



Red Hat Satellite 6.3

コンテンツ管理ガイド

Red Hat およびカスタムソースからのコンテンツの管理に関するエンドツーエンド
ガイド

Red Hat Satellite 6.3 コンテンツ管理ガイド

Red Hat およびカスタムソースからのコンテンツの管理に関するエンドツーエンドガイド

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

法律上の通知

Copyright © 2018 Red Hat, Inc.

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

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

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

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

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

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

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

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

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

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

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

All other trademarks are the property of their respective owners.

概要

本書では、Red Hat Satellite 6 でのコンテンツの管理に関するエンドツーエンドシナリオを提供します。これらのコンテンツには、RPM ファイル、ISO イメージ、Puppet モジュール、コンテナイメージなどがあります。Red Hat Satellite 6 は、アプリケーションライフサイクル全体でプロモートされた一連のコンテンツビューを使用してコンテンツを管理します。本書では、それぞれの組織に合わせたアプリケーションライフサイクルとライフサイクル環境内でホストの状態に合致するコンテンツビューの作成方法を説明します。これらのコンテンツビューは、Red Hat Satellite 6 環境でホストをプロビジョニングおよび更新する基礎となります。

目次

第1章 はじめに	5
1.1. RED HAT SATELLITE 6 コンテンツ管理の概要	5
1.2. アプリケーションライフサイクルの定義	5
1.3. コンテンツ管理タイプの定義	6
1.4. シナリオの定義	7
1.5. コンテンツ管理ストレージの定義	8
1.6. 本章のまとめ	10
第2章 組織の作成	11
2.1. 組織の作成	11
2.2. コンテキストの設定	12
2.3. 組織のデバッグ証明書の作成	12
2.4. 組織のデバッグ証明書の使用	13
2.5. 組織の削除	14
2.6. 本章のまとめ	14
第3章 サブスクリプションの管理	16
3.1. 複数の割り当てを使用した複数の組織の管理	16
3.2. カスタマーポータルでサブスクリプションの割り当ての作成	16
3.3. 割り当てへのサブスクリプションの追加	16
3.4. カスタマーポータルからのサブスクリプションマニフェストのエクスポート	17
3.5. SATELLITE SERVER へのサブスクリプションマニフェストのインポート	17
3.6. マニフェストの更新	18
3.7. コンテンツホストへのサブスクリプションのアタッチ	18
3.8. コンテンツホストのサブスクリプションの一括アップデート	19
3.9. 本章のまとめ	20
第4章 RED HAT コンテンツのインポート	21
4.1. DEFINITIVE MEDIA LIBRARY の作成	21
4.2. SATELLITE での製品およびリポジトリの使用	21
4.3. コンテンツの同期	21
4.4. ダウンロードポリシーの使用	21
4.5. 同期する RED HAT リポジトリの選択	22
4.6. RED HAT リポジトリの同期	24
4.6.1. リポジトリの復旧	25
4.6.2. 同期速度の制限	26
4.7. 同期プランの作成	27
4.8. 本章のまとめ	28
第5章 カスタムコンテンツのインポート	29
5.1. SATELLITE でのカスタム製品の使用	29
5.2. カスタム製品の作成	29
5.3. カスタム GPG キーのインポート	30
5.4. カスタム RPM リポジトリの作成	30
5.5. カスタム PUPPET リポジトリの作成	32
5.6. 個別 PUPPET モジュールの管理	33
5.7. PUPPET リポジトリの同期	34
5.8. GIT リポジトリからの PUPPET MODULES の同期	35
5.9. RED HAT SATELLITE でカスタムファイルタイプリポジトリの作成	36
5.10. RED HAT SATELLITE へのカスタムファイルタイプリポジトリへのファイルのアップロード	39
5.11. RED HAT SATELLITE のカスタムファイルタイプリポジトリからホストにファイルのダウンロード	39
5.12. ローカルディレクトリーにカスタムのファイルタイプリポジトリの作成	41

5.13. 本章のまとめ	42
第6章 アプリケーションライフサイクルの作成	43
6.1. アプリケーションライフサイクルの再検討	43
6.2. 新規アプリケーションライフサイクルの作成	43
6.3. CAPSULE SERVER へのライフサイクル環境の追加	44
6.4. アプリケーションライフサイクルでのコンテンツのプロモーション	45
6.5. RED HAT SATELLITE CAPSULE SERVER へのライフサイクル環境の追加	47
6.6. コンテンツビューのプロモート	48
6.7. SATELLITE SERVER からのライフサイクル環境の削除	48
6.8. CAPSULE SERVER からのライフサイクル環境の削除	49
6.9. 本章のまとめ	50
第7章 コンテンツビューの管理	51
7.1. コンテンツビューの理解	51
7.2. 標準コンテンツビュー	52
7.2.1. シンプルなコンテンツビューの作成	52
7.2.2. Puppet モジュールを含むコンテンツビューの作成	53
7.3. 複合コンテンツビュー	54
7.3.1. 複合コンテンツビューの作成	56
7.4. コンテンツフィルター	57
7.4.1. コンテンツフィルターの作成	59
7.5. コンテンツビューのプロモート	61
7.6. 環境とコンテンツビューへのシステム登録	62
7.6.1. サブスクリプションマネージャーへの RHEL システムの登録	62
7.6.2. サブスクリプションマネージャーへの Atomic Host の登録	63
7.7. 本章のまとめ	63
第8章 アクティベーションキーの管理	64
8.1. アクティベーションキーの作成	64
8.2. アクティベーションキーの使用	67
8.3. アクティベーションキーを使用して関連するサブスクリプションの更新	68
8.4. 自動アタッチの有効化	70
8.5. サービスレベルの設定	71
8.6. 本章のまとめ	71
第9章 エラータの管理	72
9.1. コンテンツビューによるエラータ管理	72
9.2. 利用可能なエラータの検出	73
9.3. 個別システムへのエラータの適用	75
9.4. 複数システムへのエラータの適用	76
9.5. エラータ通知のサブスクライブ	77
9.6. 本章のまとめ	77
第10章 コンテナイメージの管理	78
10.1. RED HAT CONTAINER CATALOG からのコンテナイメージのインポート	79
10.2. 他のイメージレジストリーからのコンテナイメージのインポート	80
10.3. コンテンツビューによるコンテナイメージの管理	82
10.4. DOCKER タグによるコンテナイメージの管理	83
10.5. 本章のまとめ	84
第11章 OSTREE コンテンツの管理	85
11.1. SATELLITE SERVER での OSTREE 管理の設定	85
11.2. 同期する RED HAT OSTREE コンテンツの選択	85
11.3. カスタム OSTREE コンテンツのインポート	86

11.4. コンテンツビューによる OSTREE コンテンツの管理	87
11.5. 本章のまとめ	88
第12章 ISO イメージとファイルの管理	89
12.1. RED HAT からの ISO イメージのインポート	89
12.2. 個別の ISO イメージとファイルのインポート	90
12.3. RED HAT OVAL リポジトリのインポート	91
12.4. 本章のまとめ	93
第13章 コンテンツ管理の最終処理	94
13.1. シナリオにおける目的の完了	94
13.2. システムのプロビジョニング	94
13.2.1. キックスタートリポジトリをインストールメディアとして使用	95
13.2.2. 環境への登録	95
13.2.3. 新規システムのプロビジョニング	96
付録A コンテンツストレージ向け NFS 共有の使用	97
付録B 非接続の SATELLITE SERVER へのコンテンツ ISO のインポート	99
付録C 接続済み SATELLITE SERVER へのコンテンツ ISO のインポート	101
付録D SATELLITE SERVER 間でのコンテンツ同期	104
D.1. SATELLITE SERVER、CAPSULE SERVER、および ISS	105
D.2. 前提条件	105
D.3. サポートされる同期オプション	106
D.4. チャンク ISO ファイルの使用	106
D.5. ISS の設定	106
D.5.1. エクスポート先の設定	106
D.5.2. ダウンロードポリシーの設定	108
D.6. コンテンツのエクスポート	109
D.6.1. リポジトリのエクスポート	109
D.6.2. コンテンツビューバージョンのディレクトリへのエクスポート	109
D.6.3. 増分更新	112
D.7. コンテンツのインポート	113
D.7.1. リポジトリのインポート	113
D.7.2. Red Hat リポジトリとしてのコンテンツビューのインポート	113
付録E リモートファイルタイプリポジトリの作成	115
付録F GIT を使用したテンプレートの同期	117
F.1. TEMPLATESYNC プラグインの有効化	117
F.2. TEMPLATESYNC プラグインの設定	117
F.3. テンプレートのインポートおよびエクスポート	119
F.3.1. Git リポジトリでテンプレートの同期	119
F.3.2. ローカルディレクトリ	120
F.4. 高度な GIT 設定	121
F.5. プラグインのアンインストール	121

第1章 はじめに

システム管理のコンテキストでは、コンテンツは、システム上にインストールされたソフトウェアとして定義されます。これには、ベースオペレーティングシステム、ミドルウェアサービス、エンドユーザーアプリケーションが含まれます(ただし、これらに限定されません)。Red Hat Satellite 6 は、Red Hat Enterprise Linux システム向けのさまざまな種類のコンテンツを管理するツールを提供します。このため、システム管理者は、簡単に広範なコンテンツを収集したり、コンテンツを最新の状態に保ったり、コンテンツを使用して新しいシステムをプロビジョニングして既存のシステムを更新したりできます。

本書では、コンテンツの管理方法を示すエンドツーエンドシナリオを提供します。Satellite Server を新規にインストールしたシステムの管理者を対象をしています。

1.1. RED HAT SATELLITE 6 コンテンツ管理の概要

Red Hat Satellite 6 のコンテキストでは、コンテンツ管理は複数のコンテンツタイプ向けの持続可能なリポジトリを提供するワークフローを意味します。Red Hat コンテンツについては、Red Hat Satellite 6 ではサブスクリプション情報を使用して、ユーザーに利用可能なコンテンツを認識します。つまり、Red Hat Satellite 6 は以下のものを管理するコンポーネントを使用します。

- **サブスクリプション管理**。これには Red Hat ソフトウェアサブスクリプションと関連コンテンツを安全な接続を介して管理するツールが含まれます。これにより、組織が Red Hat サブスクリプション情報を管理する手段が提供されます。
- **コンテンツ管理**。これにはコンテンツをダウンロードし、カスタムリポジトリに格納するアプリケーションが含まれます。これにより、組織は Red Hat コンテンツを格納し、さまざまな方法で整理することができるようになります。

1.2. アプリケーションライフサイクルの定義

アプリケーションライフサイクルは、Red Hat Satellite 6 のコンテンツ管理機能の中心となる概念です。アプリケーションライフサイクルは、特定の段階で特定のシステムとソフトウェアがどのように見えるかを定義します。たとえば、アプリケーションライフサイクルが単純な場合には、開発段階と実稼働段階のみになります。このような場合、アプリケーションライフサイクルは以下のようになります。

- 開発
- 実稼働

ただし、テストやベータリリースなど、より多くの段階が含まれ、アプリケーションライフサイクルが複雑になる場合があります。

- 開発
- テスト
- Beta リリース
- 実稼働

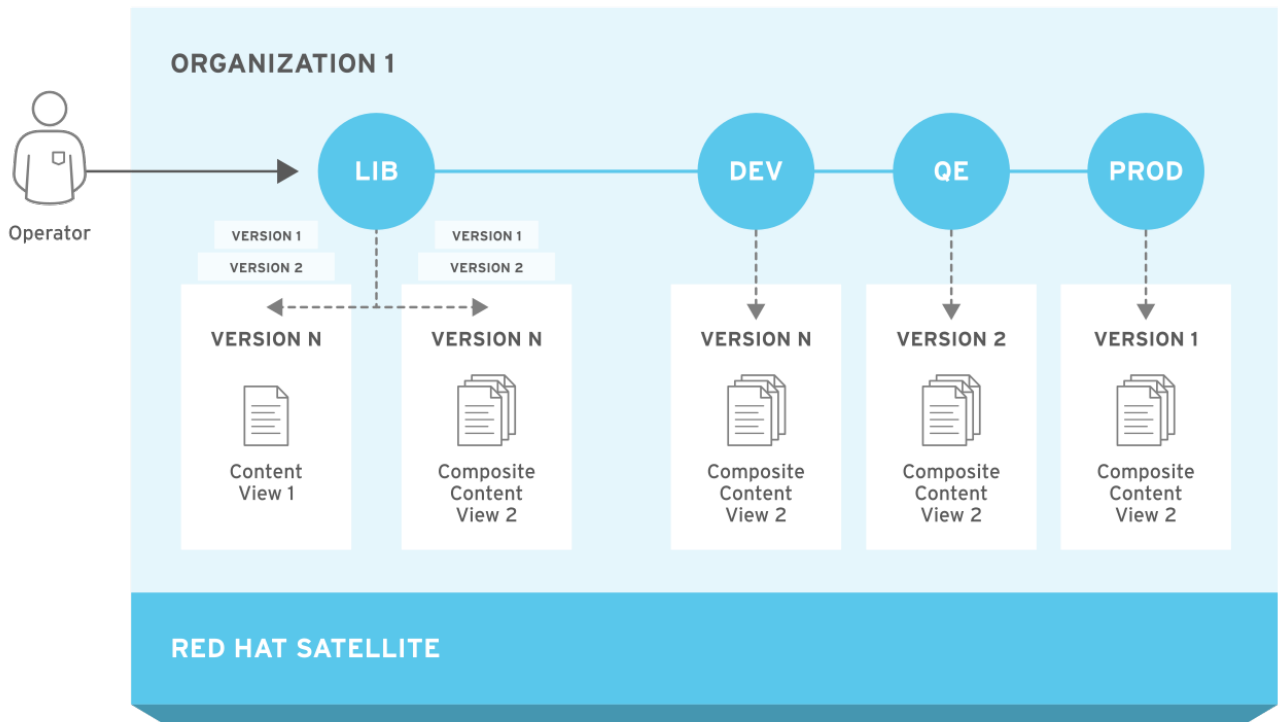
結局のところ、アプリケーションライフサイクルに含まれる段階は、それぞれの組織とソフトウェア開発方法によって異なります。Red Hat Satellite 6 はそれぞれの仕様を満たすために、アプリケーションライフサイクルの各段階をカスタマイズする方法を提供します。

Red Hat Satellite 6 では、アプリケーションライフサイクルの各段階は**環境**と呼ばれます。各環境はコンテンツの特定のコレクションを使用します。Red Hat Satellite 6 では、これらのコンテンツコレク

ションはコンテンツビューとして定義されます。各コンテンツビューは、特定の環境に含めるリポジトリ、パッケージ、および Puppet モジュールを定義できるフィルターとなります。これにより、ユーザーは各環境に指定する特定のコンテンツセットを定義できるようになります。

アプリケーションライフサイクルの概念は、進捗状況によって異なります。たとえば、アプリケーションライフサイクルの初期段階の環境では、新しい機能の開発とテストを行うために、新しいリリース前のパッケージを使用することがあります。同様に、それ以降の段階の環境では、実稼働用ソフトウェアに適した安定したパッケージのみを使用することがあります。開発中のパッケージのテストが完了したら、アプリケーションライフサイクルで、開発コンテンツビューを、実稼働環境用のコンテンツビューにプロモートすることができます。この結果、アプリケーションの開発でアプリケーションライフサイクルが進むことになります。

図1.1 Red Hat Satellite 6 アプリケーションライフサイクル



1.3. コンテンツ管理タイプの定義

Red Hat Satellite 6 では、以下のものを含むさまざまなコンテンツタイプを管理できます。

RPM パッケージ

Red Hat Satellite 6 では、Red Hat サブスクリプションに関連するリポジトリから RPM ファイルをインポートする方法が提供されます。これにより、Satellite Server は Red Hat のコンテンツ配信ネットワークから RPM ファイルをダウンロードし、ローカルに保存します。これらのリポジトリと RPM ファイルはコンテンツビューで使用できます。

キックスタートツリー

Red Hat Satellite 6 は、新しいシステムを作成するためにキックスタートツリーを取得します。新しいシステムは、ネットワークを介してこれらのキックスタートツリーにアクセスしてインストールのベースコンテンツとして使用します。また、Red Hat Satellite 6 には、事前に定義されたいくつかのキックスタートテンプレートが含まれます (独自のキックスタートテンプレートを作成することもできます)。これらのテンプレートは、新しいシステムをプロビジョニングし、インストールをカスタマイズするために使用されます。

ISO および KVM イメージ

Red Hat Satellite 6 は、インストールおよびプロビジョニング向けのメディアをダウンロードおよび管理します。たとえば、Satellite は、特定の Red Hat Enterprise Linux バージョン向けの ISO イメージおよび KVM ゲストイメージをダウンロード、保存、および管理します。

Puppet モジュール

Red Hat Satellite 6 では、RPM コンテンツとともに Puppet モジュールをアップロードできるため、プロビジョニング後にシステムの状態を設定できます。また、ユーザーはプロビジョニングプロセスの一部として Puppet クラスとパラメーターを管理することもできます。

コンテナイメージ

Red Hat Satellite 6 は、コンテナイメージのレジストリーとして機能することができます。これにより、Red Hat Enterprise Linux Atomic Host を使用してコンテナを作成する方法が提供されます。

OSTree

Red Hat Satellite 6 は OSTree ブランチをインポートし、このコンテンツを HTTP の場所に公開できます。

1.4. シナリオの定義

本書では、シナリオ例を使用して Red Hat Satellite 6 の機能を説明します。このシナリオでは、**ACME** という名前のソフトウェア開発会社が最近 Red Hat Satellite 6 をインストールし、その会社のシステム管理者が Red Hat コンテンツをインポートおよび管理したいとします。ACME の最終的な目標は以下のとおりです。

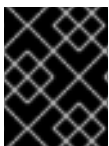
- 組織のためにコンテンツソースセットをインポートします。これには Red Hat サブスクリプションのコンテンツと独自のカスタムコンテンツが含まれます。
- ソフトウェア開発プロセスに基づいてアプリケーションライフサイクルを定義します。
- 新しい Red Hat Enterprise Linux ホストをプロビジョニングし、既存の Red Hat Enterprise Linux ホストを登録できるよう Satellite Server を準備します。

本書ではコンテンツ管理のみを取り上げます。ホストプロビジョニング、環境アーキテクチャー、Satellite Server 管理などの他の機能については、Red Hat Satellite 6 シリーズの他のガイドを参照してください。

本書では、Red Hat Satellite 6 Web UI または CLI ツール (**hammer**) のいずれかを使用する手順を説明します。いずれかの方法を選択してください。CLI を選択し、**hammer** コマンドを実行するたびに認証の詳細情報を入力しないようにするには、ローカルユーザーに対して CLI 設定ファイルを作成します。

```
# mkdir ~/.hammer
# cat > .hammer/cli_config.yml <<EOF
:foreman:
  :host: 'https://satellite.example.com/'
  :username: 'admin'
  :password: 'p@55w0rd!'
```

EOF



重要

本書での **hammer** コマンドのすべての使用箇所では、設定ファイルが使用され、認証の詳細情報を入力することはありません。

一部の CLI コマンドでは、特定のタスクを非同期的に実行する `--async` オプションを使用できます。たとえば、コンテンツを同期して監視するのではなく、コンテンツビューを非同期タスクとして公開できます。本書では、ユーザーが完了するタスクの進行状況を監視できるよう `--async` が省略されます。特定のタスクで `--async` オプションを使用する場合は、次のタスクに進む前にそのタスクを完了してください。

1.5. コンテンツ管理ストレージの定義

Red Hat Satellite 6 では、Red Hat のコンテンツ配信ネットワークと同期されるコンテンツを含む RPM および Puppet コンテンツ用リポジトリと、独自のカスタムリポジトリがホストされます。このようなリポジトリのサイズは時間の経過とともに大きくなります。したがって、それぞれの環境に合わせて適切なサイズ要件を推定し、将来の要件を適切にスケールする必要があります。

Red Hat Satellite 6 では、リポジトリコンテンツの保存と管理が `/var/lib/pulp` で行われます。このディレクトリは、スケール可能な大規模ローカルパーティションにマウントすることをお勧めします。たとえば、このパーティションを作成するには論理ボリュームマネージャー (LVM) を使用します。



重要

`/var/lib/pulp` は NFS 共有にマウントしないでください。Red Hat Satellite 6 の一部は、NFS に問題がある一時的な SQLite データベースを使用します。NFS 共有を使用する場合は、主要なソースコンテンツユニットを含む `/var/lib/pulp/content` ディレクトリのみをマウントします。Red Hat は、`/var/lib/pulp` ファイルシステムに高帯域幅で低レイテンシーのストレージを使用することをお勧めします。Red Hat Satellite には、I/O を大量に使用する多くの操作があるため、高レイテンシーで低帯域幅のストレージを使用すると、パフォーマンスが低下することがあります。

Red Hat Satellite 6 は、Red Hat のコンテンツ配信ネットワークのパッケージを同期し、保存します。これには、Red Hat Enterprise Linux などの Red Hat ソフトウェア向けのリポジトリが含まれます。Red Hat コンテンツの推奨ストレージ要件は以下のとおりです。

主要な Red Hat Enterprise Linux バージョンの実稼働フェーズ 1 の間:

- 各バイナリーパッケージリポジトリに対して最低 40GB
- 各デバッグ情報リポジトリに対して最低 80GB

主要な Red Hat Enterprise Linux バージョンの実稼働フェーズ 1 の後:

- このようなりポジトリの推定された年間増加率は、バイナリーパッケージリポジトリごとに 10GB、デバッグ情報リポジトリごとに 20GB です。

Red Hat 実稼働フェーズの詳細は「[Red Hat Enterprise Linux のライフサイクル](#)」を参照してください。



注記

すべてのリポジトリのサイズは異なります。これらの仕様は推奨にすぎず、同期を選択したリポジトリに応じて調整する必要があります。

Red Hat Satellite 6 では、ユーザーはコンテンツビューを作成できます。コンテンツビューは、特定の時点でのユーザー定義コンテンツコレクションのスナップショットとなります。これにより、既存のリポジトリからカスタマイズしたコンテンツコレクションを作成できるようになります。

コンテンツビューのコンテンツの各ユニットは、`/var/lib/pulp/content` ディレクトリーにある Definitive Media Library へのシンボリックリンクを使用します。また、コンテンツビュー内の各リポジトリーには、コンテンツビューに属するコンテンツに関するメタデータが含まれます。したがって、使用しているパッケージが最小数しかないと、コンテンツビューが使用するストレージのサイズは少なくなります。コンテンツビューを複数使用し、ビューごとに使用するパッケージが大量になると、ストレージサイズが増加します。

たとえば、Red Hat Enterprise Linux 7 RPM リポジトリーを使用しているコンテンツビューには 7000 を超えるパッケージが含まれることがあります。この場合のディスク領域は、100MB 未満のシンボリックリンクになります。ただし、以下のことを考慮してください。

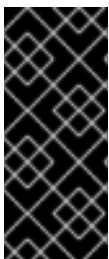
- このリポジトリーを含むコンテンツビューの数
- コンテンツビューごとのバージョンの数
- プロモートされたビューを使用しているライフサイクル環境の数
- キックスタートツリーやライブ CD コンテンツなどの追加コンテンツ

コンテンツビューが使用するストレージの量を削減するには、以下のことを行うことをお勧めします。

- コンテンツビューの未使用バージョンを削除します。ライフサイクル環境でコンテンツビューを使用せず、再使用しない場合は、削除して使用済みストレージを確保します。
- コンテンツビューでフィルターを使用します。フィルターは、コンテンツビューに表示されるコンテンツを制限します。これにより、ビューに必要なコンテンツのみを定義し、冗長なコンテンツを除外できるようになります。各コンテンツビューのサイズは大幅に削減されます。
- `/var/lib/pulp/nodes` ディレクトリーを監視します。Red Hat Satellite 6 は、このディレクトリーを使用してコンテンツビューを構築します。
- `/var/lib/pulp/published` ディレクトリーを監視します。Red Hat Satellite 6 は、このディレクトリーを使用してコンテンツビューを公開します。

また、Red Hat Satellite 6 はコンテンツ管理コンポーネント向けに以下のデータベースを使用します。

- **主要データベース:** `/var/lib/pgsql` に格納された PostgreSQL データベース。ストレージの要件は、組織、環境、登録されたシステム、コンテンツビューなどの複数の要因によって異なります。
- **コンテンツデータベース:** `/var/lib/mongodb` に格納された MongoDB データベース。ストレージの要件は、Red Hat Satellite 6 環境向けのパッケージとコンテンツビューの数によって異なります。通常は、大規模なストレージが使用されます。コンテンツデータベース向けに少なくとも 10GB を予約し、リポジトリーごとに 5GB を計画します。このディレクトリーは、スケール可能な大規模ローカルパーティションにマウントすることをお勧めします。たとえば、このパーティションを作成するには論理ボリュームマネージャー (LVM) を使用します。



重要

`/var/lib/mongodb` は NFS 共有にマウントしないでください。Red Hat は、`/var/lib/mongodb` ファイルシステムに高帯域幅で低レイテンシーのストレージを使用することをお勧めします。Red Hat Satellite には、I/O を大量に使用する多くの操作があるため、高レイテンシーで低帯域幅のストレージを使用すると、パフォーマンスが低下することがあります。

1.6. 本章のまとめ

本章では、Red Hat Satellite 6 のコンテキストのコンテンツ管理の基本的な概念を説明しました。これには、Red Hat Satellite 6 アプリケーションライフサイクルのフェーズの定義と Red Hat Satellite 6 が管理するコンテンツのさまざまなタイプが含まれます。また、本章では、シナリオ例のスコープが定義され、それぞれのコンテンツ管理のニーズを満たすストレージ要件が提供されました。

次章では、組織を作成してコンテンツを保存する方法を説明します。

第2章 組織の作成

組織は、所有者、目的、コンテンツ、セキュリティーレベルなどに基づいて Red Hat Satellite 6 リソースを論理グループに分割します。Red Hat Satellite 6 では複数の組織を作成および管理し、Red Hat サブスクリプションを分割して、各個別組織に割り当てることができます。これにより、1つの管理システムで複数の個別組織のコンテンツを管理できるようになります。以下に、組織管理の例をいくつか示します。

1つの組織

単純なシステム管理チェーンを持つ小規模な会社。この場合は、会社に対して組織を1つ作成し、コンテンツをその組織に割り当てます。

複数の組織

複数の小規模な事業単位を所有する大規模な会社 (たとえば、独立したシステム管理およびソフトウェア開発グループがある会社)。この場合は、会社と会社が所有する各事業単位に対して組織を作成します。これにより、それぞれのシステムインフラストラクチャーを分けることができます。各組織に、それぞれのニーズに基づいてコンテンツを割り当てます。

外部組織

他の組織の外部システムを管理する会社 (たとえば、クラウドコンピューティングと Web ホスティングを顧客に提供する会社)。この場合は、会社の独自のシステムインフラストラクチャーの組織に加え、外部の各会社に対して組織を作ることが考えられます。必要に応じて、各組織にコンテンツを割り当てます。

本ガイドのシナリオでは、ACME は1つの組織として機能するため、目的は ACME 向けの組織を作成し、管理することになります。Red Hat Satellite 6 のデフォルトのインストールでは、**Default_Organization** という名前のデフォルト組織が提供されます。ただし、このシナリオでは、ACME 向けのカスタム組織を作成して、設定していきます。



重要

新しいユーザーにデフォルトの組織が割り当てられていないと、そのユーザーのアクセスは制限されます。ユーザーにシステムの権限を付与するには、ユーザーをデフォルトの組織に割り当てた後にログアウトし、再度ログインします。

2.1. 組織の作成

Web UI をご利用の場合

管理 > **組織** に移動します。Satellite Server が現在管理している組織の一覧が表示されます。

新規組織 をクリックします。

以下の3つのセクションから構成される作成ウィザードが表示されます。

組織の作成

以下のような組織の基本詳細情報を提供します。

- **名前:** 組織の簡単な名前。このシナリオの場合は、**ACME** を使用します。
- **ラベル:** 組織の一意的な ID。これは、コンテンツストレージ用ディレクトリーなどの特定のASETを作成およびマップする場合に使用されます。文字、数字、アンダースコア、およびダッシュを使用し、スペースは使用しないでください。このシナリオではここでも **ACME** を使用します。
- **説明:** 組織の簡単な説明 (オプション)。このシナリオでは **Our example organization**

を使用します。

ホストの選択

すべてのホストには組織が必要です。ただし、状況によっては、ホストが孤立することがあります。たとえば、古い組織を削除すると、そのホストが孤立することがあります。このような状況では、必要に応じて、新たに作成した組織に孤立したホストを割り当てることができます。孤立したすべてのホストを割り当ての場合は **すべてを割り当て** を選択し、割り当てるホストを選択する場合は **手動割り当て** を選択します。ACME のシナリオでは孤立したホストは存在していないため、**編集に進む** をクリックして **プロパティの編集** セクションに移動します。

プロパティの編集

このセクションでは、組織に特定のインフラストラクチャーリソースを割り当てることができます。これには、ネットワークリソース、インストールメディア、キックスタートテンプレートなどが含まれます。この画面には、**管理 > 組織** に移動し、編集する組織を選択すればいつでも戻ることができます。このシナリオでは、その他の設定は必要ありません。ただし、本書では、キックスタートツリーの同期後にこのセクションに戻ります。

組織の作成後に、**送信** をクリックします。

CLI をご利用の場合

```
# hammer organization create \
--name "ACME" \
--label "ACME" \
--description "Our example organization for managing content."
```

これにより、最初の組織が作成されます。

2.2. コンテキストの設定

Red Hat Satellite 6 でコンテンツを管理する前に、コンテキストを設定する必要があります。コンテキストは、コンテンツを使用する組織を定義します。

Web UI をご利用の場合

コンテキストメニューは、画面の左上隅にあります。コンテキストを選択しないと、メニューには「すべてのコンテキスト」と示されます。このメニューにカーソルを置き、**組織** セレクターで **ACME** を選択します。これにより、コンテキストが ACME 組織に変更します。

CLI をご利用の場合

CLI を使用している場合は、**--organization "ACME"** オプションまたは **--organization-label "ACME"** オプションのいずれかを指定します。以下は例となります。

```
# hammer subscription list --organization "ACME"
```

これにより、CLI を介した各対話のコンテキストが設定されます。

2.3. 組織のデバッグ証明書の作成

組織のデバッグ証明書の新規作成:

1. **管理 > 組織** に移動します。

2. デバッグ証明書を生成する組織を選択します。
3. **生成してダウンロード** をクリックします。これにより、デバッグ証明書が生成されます。
4. 証明書ファイルを安全な場所に保存します。



注記

デバッグ証明書が組織内にダウンロードされていない場合は、プロビジョニングテンプレートのダウンロード時に自動的に生成されます。

2.4. 組織のデバッグ証明書の使用

組織のリポジトリコンテンツは、その組織のデバッグ証明書を持っている場合に、ブラウザまたは API を使用して表示できます。前のセクションで、X.509 形式の証明書の作成およびダウンロードを説明しました。ブラウザを使用する場合は、最初に X.509 証明書を、ブラウザがサポートする形式に変換し、次にその証明書をインポートする必要があります。**curl** ユーティリティでは、別のファイルへの証明書およびキーの抽出のみを行います。

Firefox での組織のデバッグ証明書の使用:

1. 「[組織のデバッグ証明書の作成](#)」に記載されている手順で、組織の証明書を作成およびダウンロードします。
2. たとえば、デフォルトの組織の X.509 証明書を開きます。

```
$ vi 'Default Organization-key-cert.pem'
```

3. このファイルの `-----BEGIN RSA PRIVATE KEY-----` から `-----END RSA PRIVATE KEY-----` までを、**key.pem** ファイルにコピーします。
4. このファイルの `-----BEGIN CERTIFICATE-----` から `-----END CERTIFICATE-----` までを、**cert.pem** ファイルにコピーします。
5. 以下のようにコマンドを入力して PKCS12 形式の証明書を作成し、パスワードまたはフレーズが要求された場合はそれを入力します。

```
$ openssl pkcs12 -keypbe PBE-SHA1-3DES -certpbe PBE-SHA1-3DES -
export -in cert.pem -inkey key.pem -out organization_label.pfx -name
organization_name
Enter Export Password:
Verifying - Enter Export Password:
```

6. 設定タブを使用して、作成した **pfx** ファイルをブラウザにインポートします。**編集 > 設定 > 詳細タブ** に移動します。**証明書** ビューの **証明書の表示** を選択して、**証明書マネージャー** を開きます。**ユーザーの証明書** タブで、**インポート** をクリックし、ロードする **pfx** ファイルを選択します。証明書の作成時に、パスワードまたはフレーズを入力するように求められます。
7. ブラウザーのアドレスバーに以下の形式の URL を入力し、リポジトリの参照を開始します。

```
http://satellite.example.com/pulp/repos/organization_label
```

Pulp は組織ラベルを使用するため、URL でも組織ラベルを使用する必要があります。

curl での組織のデバッグ証明書の使用:

1. 「[組織のデバッグ証明書の作成](#)」に記載されている手順で、組織の証明書を作成およびダウンロードします。
2. たとえば、デフォルトの組織の X.509 証明書を開きます。

```
$ vi 'Default Organization-key-cert.pem'
```

3. このファイルの -----BEGIN RSA PRIVATE KEY----- から -----END RSA PRIVATE KEY----- までを、**key.pem** ファイルにコピーします。
4. このファイルの -----BEGIN CERTIFICATE----- から -----END CERTIFICATE----- までを、**cert.pem** ファイルにコピーします。
5. リポジトリの有効な URL を確認します。前の手順で説明した参照方法を使用するか、Web UI を使用します。たとえば、Web UI を使用して **コンテンツ > 製品** に移動し、製品名を選択します。リポジトリ タブでリポジトリ名を選択し、**公開** エントリーを探します。
6. **curl** を使用してリポジトリにアクセスするには、以下のようなコマンドを入力します。

```
$ curl -k --cert cert.pem --key key.pem
http://satellite.example.com/pulp/repos/Default_Organization/Library
/content/dist/rhel/server/7/7Server/x86_64/sat-tools/6.3/os/
```

cert.pem と **key.pem** へのパスが適切な絶対パスであることを確認します。間違っているとコマンドが失敗し、メッセージも表示されません。

2.5. 組織の削除

組織の削除:

組織は、ライフサイクル環境またはホストグループに関連づけられていない場合に削除できます。削除する組織にライフサイクル環境またはホストグループが関連付けられている場合は、**組織** に移動して関連するタブをクリックします。インストール中に作成されたデフォルトの組織は、Satellite 環境で関連付けられていないホストへのプレースホルダーであるため、削除することは推奨されません。環境には常に 1 つ以上の組織が必要です。

1. **管理 > 組織** に移動します。
2. 削除する組織の名前の右側にあるリストから **削除** を選択します。
3. 警告ボックスが表示されます。

```
組織 を削除しますか?
```

4. **OK** をクリックして、組織を削除します。

2.6. 本章のまとめ

本章では、新しい組織を作成し、組織のデバッグ証明書を作成し、組織を削除し、組織をコンテンツ管理のコンテキストとして設定する方法を説明しました。

次章では、Red Hat Satellite 6 でサブスクリプションを組織にインポートする方法を説明します。サブスクリプションをインポートすると、Red Hat コンテンツを管理できるようになります。

第3章 サブスクリプションの管理

Red Hat Satellite 6 では、Red Hat のコンテンツ配信ネットワーク (CDN) からコンテンツをインポートします。Satellite Server には、お使いの Satellite の各組織に対するサブスクリプションの割り当てが含まれているマニフェストが必要です。これにより、対応するリポジトリを見つけてアクセスし、そのリポジトリからダウンロードできるようになります。すべてのサブスクリプション情報は Red Hat カスタマーポータルアカウントで入手できます。

Red Hat Satellite 6.3 には、割り当てに将来の日付が指定されたサブスクリプションを使用する機能が追加されました。これにより、既存のサブスクリプションの有効期限の前に、将来の日付が指定されたサブスクリプションがコンテンツホストに追加されたときに、リポジトリへのアクセスが妨げられることはありません。Red Hat は、自動アタッチ機能を使用せずに、現在のサブスクリプションの有効期限が切れる前に、コンテンツホストに将来の日付が指定されたサブスクリプションを手動でアタッチすることを推奨します。詳細は「[コンテンツホストへのサブスクリプションのアタッチ](#)」を参照してください。

本章では、サブスクリプションのサブセットを含むサブスクリプションの割り当てを作成する方法を説明します。

3.1. 複数の割り当てを使用した複数の組織の管理

複数の組織を管理する場合は、Satellite Server で複数の割り当てを使用できます。Satellite 6 では、各組織ごとに Satellite で設定された割り当てが 1 つ必要になります。この利点は、各組織が完全に独立したサブスクリプションを保持するため、各組織をそれぞれ独自の Red Hat Network アカウントでサポートできることです。

3.2. カスタマーポータルでサブスクリプションの割り当ての作成

サブスクリプション情報は、Red Hat カスタマーポータルでアクセスできます。そこで subscription allocation を使用して、Red Hat Satellite サーバーなどのオンプレミス管理アプリケーションで使用するサブスクリプションを割り当てることができます。

1. ブラウザーで <https://access.redhat.com/> を開き、Red Hat アカウントでログインします。
2. カスタマーポータルの左上にある **サブスクリプション** に移動します。
3. **サブスクリプション割り当て** に移動します。
4. **新規サブスクリプションの割り当てを作成** をクリックします。
5. **名前** フィールドに名前を入力します。
6. **タイプ** の一覧から、お使いの Satellite Server に一致するタイプとバージョンを選択します。
7. **作成** をクリックします。

3.3. 割り当てへのサブスクリプションの追加

以下の手順では、サブスクリプションを割り当てに追加する方法を説明します。

1. **サブスクリプション割り当て** に移動します。
2. 変更するサブスクリプションの名前を選択します。
3. **サブスクリプション** タブをクリックします。

4. **サブスクリプションの追加** をクリックします。
5. Red Hat 製品サブスクリプションの一覧が表示されます。各製品に対する**エンタイトルメントの数量**を入力します。
6. **送信** をクリックして割り当てを完了します。

割り当てにサブスクリプションを追加したら、マニフェストファイルをエクスポートします。

3.4. カスタマーポータルからのサブスクリプションマニフェストのエクスポート

1 つ以上のサブスクリプションがあるサブスクリプション割り当てを表示し、以下のいずれかからマニフェストをエクスポートできます。

- **サブスクリプション** セクションの **詳細** タブから **マニフェストのエクスポート** ボタンをクリックします。
- **サブスクリプション** タブから **マニフェストのエクスポート** ボタンをクリックします。

マニフェストをエクスポートすると、カスタマーポータルにより、選択したサブスクリプション証明書がエンコードされ、.zip アーカイブが作成されます。作成した .zip アーカイブはサブスクリプションのマニフェストで、Satellite サーバーにアップロードできます。

3.5. SATELLITE SERVER へのサブスクリプションマニフェストのインポート

マニフェストは、Red Hat Satellite 6 の Web UI と CLI の両方でインポートできます。

Web UI をご利用の場合

1. コンテキストが、使用する組織に設定されていることを確認します。
2. **コンテンツ > Red Hat サブスクリプション** に移動します。
3. **マニフェストの管理** をクリックして、組織のマニフェストページを表示します。
4. **ファイルの選択** をクリックしてサブスクリプションマニフェストを選択し、**アップロード** をクリックします。

CLI をご利用の場合

Red Hat Satellite 6 CLI を使用するには、Satellite Server にマニフェストが必要になります。ローカルクライアントシステムで、マニフェストを Satellite Server にコピーします。

```
[user@client ~]$ scp ~/manifest_file.zip root@satellite.example.com:~/.
```

以下のコマンドを実行して、マニフェストをインポートします。

```
[root@satellite ~]# hammer subscription upload \  
--file ~/manifest_file.zip \  
--organization "organization_name"
```

数分後、CLI により、マニフェストのインポートに成功したことが報告されます。

3.6. マニフェストの更新



注記

マニフェストは削除しないでください。Red Hat カスタマーポータルまたは Satellite Web UI でマニフェストを削除すると、すべてのコンテンツホストの登録が解除されます。

サブスクリプションの割り当てを変更した場合は、マニフェストを更新してその変更を更新する必要があります。たとえば、以下のいずれかを行った場合はマニフェストを更新してください。

- サブスクリプションの更新
- サブスクリプションの数量の調整
- 追加サブスクリプションの購入

Satellite のマニフェストを更新するには、以下の3つの方法から選択できます。

- Satellite Web UI の更新ボタンを使用します。Web UI で、**コンテンツ > Red Hat サブスクリプション > マニフェストの管理** に移動し、**マニフェストの更新** ボタンを選択します。
- カスタマーポータルからダウンロードし、Satellite Web UI でアップロードします。Web UI で、**コンテンツ > Red Hat サブスクリプション > マニフェストの管理** に移動し、**参照...** をクリックします。サブスクリプションマニフェストを選択し、**アップロード** をクリックします。
- カスタマーポータルからダウンロードし、CLI から Satellite Server にアップロードします。

```
[root@satellite ~]# hammer subscription upload \
--file ~/manifest_file.zip \
--organization "organization_name"
```

3.7. コンテンツホストへのサブスクリプションのタッチ

プロビジョニング時に、コンテンツホストにサブスクリプションをタッチするには、アクティベーションキーの使用が推奨されます。ただし、アクティベーションキーは既存のホストを更新することができません。新規または追加のサブスクリプション (たとえば将来の日付が指定されたサブスクリプション) を1台のホストにタッチする必要がある場合は、以下の方法で行ってください。複数のホストをアップデートする必要がある場合は、「[コンテンツホストのサブスクリプションの一括アップデート](#)」を参照してください。アクティベーションキーの詳細は「[8章 アクティベーションキーの管理](#)」を参照してください。

Web UI をご利用の場合

1. **ホスト > コンテンツホスト** に移動します。
2. 変更するサブスクリプションが割り当てられているホストの左側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** 一覧から **サブスクリプションの管理** を選択します。
4. 任意で、**検索** フィールドにキーと値を入力し、表示するサブスクリプションを絞り込みます。

5. 追加または削除するサブスクリプションの左側にあるチェックボックスを選択し、**選択した項目を追加** または **選択した項目を削除** をクリックします。
6. **完了** をクリックして変更を保存します。

CLI をご利用の場合

1. 利用可能なサブスクリプションの一覧を表示します。

```
# hammer subscription list --organization-id 1
```

2. サブスクリプションをホストにアタッチします。

```
# hammer host subscription attach --host host_name --subscription-id subscription_id
```

3.8. コンテンツホストのサブスクリプションの一括アップデート

ここで説明する方法は、インストール後に複数のコンテンツホストを同時に変更することを目的としています。Web UI およびフィルター機能を使用して変更するコンテンツホストを選択するか、Hammer コマンドラインツールの CSV ファイルエクスポート機能を使用し、CSV ファイルで設定を変更し、変更内容をアップロードします。

Web UI をご利用の場合

1. **ホスト > コンテンツホスト** に移動します。
2. 変更するサブスクリプションが割り当てられているホストの左側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** 一覧から **サブスクリプションの管理** を選択します。
4. 任意で、**検索** フィールドにキーと値を入力し、表示するサブスクリプションを絞り込みます。
5. 追加または削除するサブスクリプションの左側にあるチェックボックスを選択し、**選択した項目を追加** または **選択した項目を削除** をクリックします。
6. **完了** をクリックして変更を保存します。

CLI をご利用の場合

1. コンテンツホストの現在の状態を CSV ファイルにエクスポートします。

```
# hammer --server https://satellite.example.com csv content-hosts --export --file content_hosts.csv
```

2. CSV ファイルで必要な値を変更します。CSV ファイルの文字列を変更するには、エディター、CSV プラグイン、または **sed** を使用できます。

- a. ファイルのバックアップを作成します。

```
# cp content_hosts.csv content_hosts.csv.backup
```

- b. 以下のように、文字列を変更します。

```
# sed -i "s/1|RH1234|Red Hat Enterprise Linux Server/1|RH5678|Red Hat Enterprise Linux Server/g" content_hosts.csv
```

c. 変更箇所を確認します。以下は例となります。

```
# diff content_hosts.csv content_hosts.csv.backup
```

3. 変更したファイルを Satellite Server にアップロードします。

```
# hammer --server https://satellite.example.com csv content-hosts --file content_hosts.csv
```

3.9. 本章のまとめ

本章では、サブスクリプション割り当てとマニフェストを使用して、Red Hat カスタマーポータルからサブスクリプション情報を取得し、Satellite Server にインポートする方法を説明しました。

次章では、コンテンツ (特に Red Hat の RPM リポジトリ) をインポートする方法を説明します。

第4章 RED HAT コンテンツのインポート

この時点で Satellite Server には必要なサブスクリプション情報がインポートされています。コンテンツはシステムに追加できる状態です。本章では、Definitive Media Library (DML) の概念とコンテンツを同期して DML を作成する方法を説明します。

4.1. DEFINITIVE MEDIA LIBRARY の作成

DML は、ソフトウェアおよび設定の承認された最終的なバージョンを保存および保護するリポジトリです。つまり、DML は Satellite にインポートされたコンテンツのマスターバージョンとして機能します。これには、RPM ファイル、キックスタートツリー、ISO イメージなどの Red Hat コンテンツが含まれます。「[Red Hat Satellite 6 コンテンツ管理の概要](#)」で説明したように、Red Hat Satellite 6 では、コンテンツは DML で保存および管理されます。

4.2. SATELLITE での製品およびリポジトリの使用

Satellite では、**製品**の概念を組織単位として使用して複数のリポジトリをグループ化します。このようなりポジトリコレクションは実際の製品の概念と似ています。たとえば、Satellite で Red Hat Enterprise Linux Server を製品として見ると、その製品のリポジトリは異なるバージョン (6.0、6.1、7.0)、異なるアーキテクチャー (i386、x86_64、s390x、arm)、および異なるアドオン (オプションリポジトリ、補助リポジトリ、Virt V2V ツール) から構成されることとなります。これにより、関連するすべてのリポジトリが DML 内で統合されます。製品を使用すると、お互いに依存するリポジトリと一緒に同期されます。Red Hat リポジトリの場合、製品はリポジトリの有効後に自動的に作成されます。

本章では、Red Hat コンテンツを使用して DML を作成します。これを行うには、DML と Red Hat の製品およびリポジトリを同期します。

4.3. コンテンツの同期

DML からリポジトリを選択すると、Satellite Server により独自のリポジトリと Red Hat CDN 上のリポジトリが同期されます。これにより、Satellite Server では Red Hat のリポジトリの同一コピーが DML の一部として保持されます。Satellite Server はこのリポジトリ情報を取得し、Satellite Server のファイルシステムに保存します。最初の同期後に、DML 内のリポジトリが CDN のリポジトリと同期された状態になるような同期計画を作成できます。

最初の更新は ISO イメージを使用して実行できます。コンテンツ ISO の使用の詳細は「[付録C 接続済み Satellite Server へのコンテンツ ISO のインポート](#)」を参照してください。帯域幅制限がある場所では、以下で説明するように、**オンデマンド** または **背景** ダウンロードポリシーを使用すると、コンテンツ ISO をダウンロードおよびインポートよりも時間が短縮されることがあります。

4.4. ダウンロードポリシーの使用

Red Hat Satellite では、RPM コンテンツの同期に関する複数のダウンロードポリシーが提供されます。たとえば、コンテンツメタデータのみをダウンロードし、実際のコンテンツのダウンロードは後で行うことで時間を短縮したい場合があります。

Satellite Server では以下のポリシーが提供されます。

- **即時**: Satellite Server は、同期時にメタデータとパッケージをすべてダウンロードします。
- **オンデマンド**: Satellite Server は同期時にメタデータのみをダウンロードします。Satellite Server は、Capsule、または直接接続したクライアントが要求した場合に限り、ファイルシステムでパッケージを取得および保存します。Satellite Server が強制的にすべてのパッケージを

ダウンロードするため、Capsule に対応するリポジトリを **即時** に設定した場合は有効ではありません。

- **背景:** Satellite Server は、最初の同期後にすべてのパッケージをダウンロードする背景タスクを作成します。

最後の 2 つのポリシーは、コンテンツの同期時間を短縮するため、**レイジー同期** 機能として動作します。レイジー同期機能は **yum** リポジトリにのみ使用してください。通常のコンテンツビューと同じように、ライフサイクル環境にプロモートすることができます。

Capsule Server に、以下のポリシーを提供します。

- **即時:** Capsule Server は、同期時にメタデータとパッケージをすべてダウンロードします。Satellite Server に対応するリポジトリを **オンデマンド** に設定した場合は、Satellite Server が強制的にすべてのパッケージをダウンロードするため、この設定を使用しないでください。
- **オンデマンド:** Capsule Server は、同期時にメタデータだけをダウンロードします。Capsule は、直接接続したクライアントが要求した場合に限り、ファイルシステムでパッケージを取得して保存します。



注記

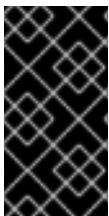
オンデマンド ダウンロードポリシーを使用すると、Capsule Server で使用できない場合に、Satellite Server からコンテンツをダウンロードします。

- **背景:** Capsule Server は、最初の同期後に、すべてのパッケージをダウンロードする背景タスクを作成します。
- **継承:** Capsule Server は、Satellite Server に対応するリポジトリから、リポジトリのダウンロードポリシーを継承します。

このポリシーは、**--enable-foreman-proxy-plugin-pulp** を **false** に設定して Capsule をインストールまたはアップデートした場合は利用できません。

4.5. 同期する RED HAT リポジトリの選択

同期するリポジトリを選択する最初の手順では、リポジトリを含む製品を特定し、リリースバージョンとベースアーキテクチャーに基づいてリポジトリを有効にします。



重要

非接続の Satellite を使用している場合は、コンテンツを同期する前に Red Hat Satellite 用のコンテンツ ISO をインポートし、Satellite Server 上で CDN URL を変更する必要があります。詳細は「[付録B 非接続の Satellite Server へのコンテンツ ISO のインポート](#)」を参照してください。

Web UI をご利用の場合

コンテンツ > **Red Hat リポジトリ** に移動します。これにより、さまざまなコンテンツタイプのタブのセットが表示されます。このページをロードするときのデフォルトのタブは **RPM** です。このタブには、RPM コンテンツを提供するサブスクリプション済みの製品一覧が含まれます。

製品と特定のリポジトリの関係は、カスケード型の階層で結ばれます。製品を選択すると、その製品のリポジトリセットの一覧が開きます。リポジトリセットを選択すると、有効にできるリポジトリの一覧が開きます。このシナリオでは、**Red Hat Enterprise Linux Server**、次に **Red Hat**

Enterprise Linux 7 Server (RPMs) を選択し、**Red Hat Enterprise Linux 7 Server RPMs x86_64 7Server** を有効にします。これにより、Red Hat Enterprise Linux 7 用の最新の RPM ファイルが有効になります。



注記

Red Hat Enterprise Linux オペレーティングシステムに 7 Server リポジトリを関連付けることと、7.X リポジトリを関連付けることの違いは、7 Server には最新アップデートがすべて含まれ、Red Hat Enterprise Linux 7.X リポジトリでは次のマイナーバージョンリリース以降のアップデートを取得しなくなることです。キックスタートリポジトリにはマイナーバージョンのみが含まれることに注意してください。

CLI をご利用の場合

製品とリポジトリの関係は同じです。製品を検索するには、以下のコマンドを使用します。

```
# hammer product list --organization "ACME"
```

製品のリポジトリのセットを一覧表示します。

```
# hammer repository-set list \
  --product "Red Hat Enterprise Linux Server" \
  --organization "ACME"
```

これにより、製品のリポジトリセット内のリポジトリが表示されます (名前と ID 番号を含む)。名前または ID 番号を使用してリポジトリを有効にします。また、リリースバージョン (**7Server**) とベースアーキテクチャー (**x86_64**) を含めます。以下は例となります。

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

このシナリオでは、Web UI または CLI を使用して ACME 向けの以下のリポジトリを有効にします。

リポジトリ	Type	説明
Red Hat Enterprise Linux 7 Server RPMs x86_64 7Server	RPM	Red Hat Enterprise Linux 7 の最新バージョン向けのリポジトリ。継続的なパッケージアップデートを受け取るために、 7.2 リポジトリの代わりに 7 Server リポジトリを使用します。

リポジトリ	Type	説明
Red Hat Satellite Tools 6.3 for RHEL 7 Server RPMs x86_64	RPM	クライアントシステム用システム管理エージェントおよびツールを含む Satellite Tools リポジトリ。新しいシステムのプロビジョニング後に、Satellite により katello-agent や Puppet などのツールがクライアントにインストールされます。継続的なパッケージアップデートを受け取るために、 7.2 リポジトリの代わりに 7 Server リポジトリを使用します。
Red Hat Enterprise Linux 7.2 Kickstart x86_64 7Server	キックスタート	Red Hat Enterprise Linux 7.2 向けキックスタートツリー。PXE を介して新しいシステムをプロビジョニングする場合にインストールメディアとして使用します。

これらのリポジトリは、このシナリオの DML 向けの初期コンテンツを提供します。それぞれのニーズに合わせて複数のリポジトリを選択できます。



注記

このシナリオでは、すべてのリポジトリで **x86_64** をベースアーキテクチャーとして使用します。

4.6. RED HAT リポジトリの同期

ここまでに、初期 DML を形成する特定のリポジトリが有効になっています。ここからは、これらのリポジトリを Red Hat CDN のリポジトリと同期します。

Web UI をご利用の場合

コンテンツ > 製品 に移動し、**Red Hat Enterprise Linux Server** を選択します。これにより、製品内で有効なリポジトリがすべて表示されます。すべてのリポジトリを選択して **同期開始** をクリックします。また、Web UI で同期の進行状況を確認することもできます。コンテンツ > 同期の状態 に移動し、製品/リポジトリツリーを展開します (または **すべて展開** をクリックします)。

CLI をご利用の場合

Red Hat Enterprise Linux Server 製品内の有効済みリポジトリを同期します。

```
# hammer product synchronize \
  --name "Red Hat Enterprise Linux Server" \
  --organization "ACME"
```

また、各リポジトリを個別に同期することもできます。製品内のすべてのリポジトリを一覧表示し、対応するリポジトリの ID 番号を使用して同期します。以下は例となります。

```
# hammer repository list \
```

```
--product "Red Hat Enterprise Linux Server" \
--organization "ACME"
# hammer repository synchronize \
--name "Red Hat Enterprise Linux 7 Server RPMs x86_64 7Server" \
--product "Red Hat Enterprise Linux Server" \
--organization "ACME"
```

同期にかかる時間は、各リポジトリのサイズとネットワーク接続の速度によって異なります。以下の表は、利用可能なインターネット帯域幅に応じてコンテンツの同期にかかる推定時間を示しています。

	単一パッケージ (10Mb)	マイナーリリース (750Mb)	メジャーリリース (6Gb)
256 Kbps	5 分 27 秒	6 時間 49 分 36 秒	2 日と 7 時間 55 分
512 Kbps	2 分 43.84 秒	3 時間 24 分 48 秒	1 日と 3 時間 57 分
T1 (1.5 Mbps)	54.33 秒	1 時間 7 分 54.78 秒	9 時間 16 分 20.57 秒
10 Mbps	8.39 秒	10 分 29.15 秒	1 時間 25 分 53.96 秒
100 Mbps	0.84 秒	1 分 2.91 秒	8 分 35.4 秒
1000 Mbps	0.08 秒	6.29 秒	51.54 秒

DML に初期コンテンツをインポートする場合は、手動による同期が必要になることがよくあります。ただし、DML が定期的に更新されるように同期計画を作成することが推奨されます。



注記

Red Hat リポジトリのダウンロードポリシーは変更できます。**Red Hat Enterprise Linux Server** 製品内のリポジトリを選択し、**ダウンロードポリシー** フィールドまでスクロールします。CLI を使用している場合は、**hammer repository update** コマンドに **--download-policy** オプションを追加して実行します。

4.6.1. リポジトリの復旧

リポジトリが破損した場合は、高度な同期を使用してそれを復旧できます。3つのオプションの中から選択できます。

- **最適同期:** リポジトリの同期時に、アップストリームの RPM との違いが検出されない RPM は回避します。
- **完全同期:** 検出した変更にかかわらずすべての RPM を同期します。特定の RPM が、アップストリームリポジトリに存在しても、ローカルリポジトリにダウンロードできない場合はこのオプションを使用します。
- **コンテンツの同期を検証:** すべての RPM を同期し、すべての RPM のチェックサムをローカルで検証します。RPM のチェックサムがアップストリームと異なっている場合は、RPM を再ダウンロードします。このオプションは **yum** リポジトリにのみ関連します。以下のいずれかのエラーが発生した場合に限りこのオプションを使用します。

- yum との同期中に、特定の RPM で **404** エラーが発生した場合。
- 特定のエラーが破損していることを示す **Package does not match intended download** エラーが発生した場合。

高度なオプションを使用して特定のリポジトリへの同期:

Web UI ユーザーの場合

1. コンテンツ > **製品** に移動します。
2. 破損したリポジトリを含む製品を選択します。
3. 同期するリポジトリの名前をクリックします。
4. **アクションの選択** メニューを展開し、**高度な同期** を選択します。
5. オプションを選択し、**同期** をクリックします。

CLI ユーザーの場合

1. リポジトリ ID の一覧を取得します。

```
# hammer repository list --organization "Default Organization"
```

2. 必要なオプションを使用して破損したリポジトリの同期:

- 最適な同期の場合:

```
# hammer repository synchronize --incremental true --id 1
```

- 完全な同期の場合:

```
# hammer repository synchronize --skip-metadata-check true --id 1
```

- コンテンツを同期する場合:

```
# hammer repository synchronize --validate-contents true --id 1
```

4.6.2. 同期速度の制限

同期の速度を制限して、利用可能な帯域幅が枯渇するのを回避したり、パフォーマンス問題を回避したりすることはできません。これは、**PULP_CONCURRENCY** パラメーターおよび **max_speed** パラメーターを設定することで行います。この設定はアップグレード時に上書きされます。設定を復元できるようにするために、アップグレード前に変更したファイルのバックアップすることが推奨されます。

1. 並行して実行する同期ジョブの数を制御するには、**/etc/default/pulp_workers** ファイルの **PULP_CONCURRENCY** パラメーターを設定します。たとえば、平行して実行するジョブの数を 1 に設定するには、**PULP_CONCURRENCY** を 1 に設定します。

```
PULP_CONCURRENCY=1
```

デフォルトでは、CPU が 8 個より少なくなるシステムでは、**PULP_CONCURRENCY** が CPU の数に設定されます。8 個以上の場合は、8 に設定されます。

2. 同期するネットワークの最大速度 (バイト毎秒) を設定するには、**max_speed** パラメーターを設定します。このパラメーターは、`/etc/pulp/server/plugins.conf.d/` ディレクトリーで各インポーターに対して個別に設定する必要があります。
 - a. たとえば、RPM コンテンツを同期する最大速度を毎秒 10 バイトに設定するには、`/etc/pulp/server/plugins.conf.d/yum_importer.json` ファイルの **"max_speed"** パラメーターを 10 に設定します。

```
# cat /etc/pulp/server/plugins.conf.d/yum_importer.json
{
  "proxy_host": null,
  "proxy_port": null,
  "proxy_username": null,
  "proxy_password": null,
  "max_speed": 10
}
```

- b. 編集後のファイル構文の検証:

```
# json_verify < /etc/pulp/server/plugins.conf.d/yum_importer.json
JSON is valid
```

3. Satellite サービスを再起動して、変更を適用します。

```
# katello-service restart
```

4.7. 同期プランの作成

同期計画では、スケジュールされた日時に DML 内のコンテンツをチェックし、更新します。Red Hat Satellite 6 では、ユーザーは同期計画を作成し、同期計画に製品を割り当てることができます。

Web UI をご利用の場合

コンテンツ > 同期プラン に移動し、**新規同期プラン** をクリックします。この UI では、同期計画に関する詳細を入力できるフィールドセットが提供されます。

- **名前:** 計画の簡単な名前。 **Example Plan** と入力します。
- **説明:** 計画の簡単な説明。 **Example Plan for ACME's repositories** と入力します。
- **間隔:** 同期をいつ実行するかを定義します。 **毎日** を選択します。
- **開始日** と **開始時刻:** 同期をいつ実行するかを定義します。今日の同期はすでに完了しているので、明日 1:00 (1AM) の同期を設定します。

保存 をクリックして計画を作成します。計画詳細ページが **詳細** と **製品** の 2 つのタブとともに表示されます。

この時点で製品を追加します。 **製品** タブをクリックし、次に **追加** をクリックします。 **Red Hat Enterprise Linux Server** 製品を選択し、 **選択を追加** をクリックします。

CLI をご利用の場合

同期プランを作成するには、以下のコマンドを実行します。

```
# hammer sync-plan create \  
--name "Red Hat Products 2" \  
--description "Example Plan for ACME's Red Hat Products" \  
--interval daily \  
--sync-date "2016-02-01 01:00:00" \  
--enabled true \  
--organization "ACME"
```

次に、その同期計画に Red Hat Enterprise Linux Server 製品を割り当てます。

```
# hammer product set-sync-plan \  
--name "Red Hat Enterprise Linux Server" \  
--sync-plan "Red Hat Products" \  
--organization "ACME"
```

この結果、Satellite Server は、毎日、Red Hat CDN に対して DML コンテンツをチェックし、Red Hat リポジトリを最新の状態にします。

4.8. 本章のまとめ

本章では、Red Hat コンテンツを ACME の Satellite Server にインポートし、同期計画を介して Red Hat コンテンツを最新の状態にする方法を説明しました。

次章では、Satellite Server の DML へカスタムコンテンツをインポートする方法を説明します。このプロセスは、カスタム製品を作成および管理する点を除いて、Red Hat コンテンツをインポートするプロセスと同じになります。

第5章 カスタムコンテンツのインポート

前章では、Definitive Media Library (DML) への Red Hat コンテンツのインポート方法を説明しました。本章では、Red Hat コンテンツと少し異なるカスタムコンテンツを説明します。事前に独自の製品を作成し、カスタマイズして、独自のリポジトリを追加します。さらに、カスタムリポジトリに Puppet モジュールを追加できます。

5.1. SATELLITE でのカスタム製品の使用

「[Satellite での製品およびリポジトリの使用](#)」では、Red Hat Satellite 6 の製品の概念と、その概念を使用してリポジトリをグループ化する方法を説明しました。Red Hat Satellite 6 では、カスタム製品を作成して複数の関連リポジトリを追加することもできます。Red Hat Satellite 6 における Red Hat コンテンツとカスタムコンテンツにはいくつかの類似点があります。

- 製品とそのリポジトリ間の関係は同じであり、リポジトリは引き続き同期する必要があります。
- カスタム製品にはクライアントがアクセスするサブスクリプション (Red Hat 製品に対するサブスクリプションに類似) が必要です。Red Hat Satellite 6 では、作成する各カスタム製品に対して新しいサブスクリプションが作成されます。

本章では、2つの関連リポジトリ (RPM コンテンツを含む RPM リポジトリと RPM コンテンツを設定するためのモジュール向け Puppet リポジトリ) を含む製品を作成します。

RPM の作成およびパッケージングの詳細は、Red Hat Enterprise Linux ドキュメンテーションの『[RPM パッケージングガイド](#)』を参照してください。

5.2. カスタム製品の作成

このシナリオでは、ACME は、PostgreSQL データベースを必要とする Web ベースアプリケーションである **Exampleware** という名前の製品を開発することを目的としています。また、ACME は、Red Hat Enterprise Linux リポジトリから PostgreSQL を使用して実稼働レベルの Exampleware を現場で使用し、新しいバージョンの PostgreSQL で Exampleware をテストすることもあります。このような場合は、新しいバージョンを同期できる PostgreSQL 向けのカスタム製品を作成します。

Web UI をご利用の場合

コンテンツ > 製品 に移動し、**新製品** をクリックして、以下の詳細情報を入力します。

- **名前:** 製品の簡単な名前。 **PostgreSQL** と入力します。
- **ラベル:** 製品の内部 ID。Red Hat Satellite 6 では、**名前** に入力した内容に基づいてこのフィールドに値が自動的に入力されます。
- **GPG キー:** 製品全体の GPG キー。特定のバージョンの PostgreSQL に対して GPG をインストールし、製品の代わりにリポジトリにアタッチするため、これは空にします。
- **同期プラン:** 製品の同期計画。これは、前の章で作成した **Example Plan** にアタッチすることができます。
- **説明:** 製品の簡単な説明。 **Content from PostgreSQL repositories** と入力します。

この情報を入力したら、**保存** をクリックします。

CLI をご利用の場合

製品を作成するには、以下のコマンドを実行します。

```
# hammer product create \
--name "PostgreSQL" \
--sync-plan "Example Plan" \
--description "Content from PostgreSQL repositories" \
--organization "ACME"
```

これにより、独自のカスタムリポジトリを作成し、そのコンテンツを同期できる新しい製品が作成されます。

5.3. カスタム GPG キーのインポート

カスタム製品を作成する前に、カスタム GPG キーを作成する必要がある場合があります。これは、PostgreSQL リポジトリとの RPM トランザクションに対してある程度のセキュリティーを提供するために使用されます。

最初に、バージョン固有のリポジトリパッケージのコピーをクライアントシステムにダウンロードします。この場合は、**pgdg-redhat95** をダウンロードします。

```
[user@client ~]$ wget http://yum.postgresql.org/9.5/redhat/rhel-7-
x86_64/pgdg-redhat95-9.5-2.noarch.rpm
```

RPM ファイルをインストールせずに抽出します。

```
[user@client ~]$ rpm2cpio pgdg-redhat95-9.5-2.noarch.rpm | cpio -idmv
```

GPG キーは、その抽出ファイルに相対的な場所である **etc/pki/rpm-gpg/RPM-GPG-KEY-PGDG-95** に存在します。

Web UI をご利用の場合

コンテンツ > **GPG キー** に移動します。**新規 Gpg キー** をクリックします。GPG キーに名前 (**PostgreSQL 9.5**) を指定し、**GPG キーのアップロード** を選択します。**参照** をクリックし、抽出した GPG キーを選択します。**保存** をクリックします。

CLI をご利用の場合

GPG キーを Satellite Server にコピーします。

```
[user@client ~]$ scp ~/etc/pki/rpm-gpg/RPM-GPG-KEY-PGDG-95
root@satellite.example.com:~/.
```

GPG キーを Satellite にアップロードします。

```
[root@satellite ~]# hammer gpg create \
--key ~/RPM-GPG-KEY-PGDG-95 \
--name "PostgreSQL 9.5" \
--organization "ACME"
```

これで、リポジトリに関連付ける GPG キーが用意できました。

5.4. カスタム RPM リポジトリの作成

通常は、実稼働レベルのサーバーでは、安定のために Red Hat Enterprise Linux に含まれる PostgreSQL のバージョンを使用します。ただし、ACME の開発者がより新しいバージョンの PostgreSQL で Exampleware をテストするとします。このとき、新しいバージョンの PostgreSQL 向けのカスタムリポジトリを作成できます。

Web UI をご利用の場合

カスタム PostgreSQL 製品を作成します。カスタム製品の作成後は、リポジトリ画面が表示されます。リポジトリの作成をクリックし、新しいリポジトリ向けのフォームを表示します。以下の詳細情報を入力します。

- **名前:** リポジトリの簡単な名前。PostgreSQL 9.5 と入力します。
- **ラベル:** リポジトリの内部 ID。Red Hat Satellite 6 では、名前に入力した内容に基づいてこのフィールドに値が自動的に入力されます。
- **タイプ:** リポジトリのタイプ。RPM ファイル (**yum**)、Puppet モジュール (**puppet**)、または Docker イメージ (**docker**) のいずれかのリポジトリを選択できます。本書のシナリオでは **yum** を選択します。新しいフィールドが表示されます。
- **URL:** ソースとして使用する外部リポジトリの URL。
。 http://yum.postgresql.org/9.5/redhat/rhel-7-x86_64/ と入力します。
- **ダウンロードポリシー:** Satellite Server が実行する同期タイプを決定します。即時を選択します。詳細は「[ダウンロードポリシーの使用](#)」を参照してください。
- **同期時のミラーリング:** アップストリームのリポジトリにないコンテンツが同期中に削除されるようにします。デフォルトでチェックが入っているので、そのままにします。
- **チェックサム:** リポジトリのチェックサム。この例ではデフォルトで SHA256 となる **Default** にします。これが Red Hat Enterprise Linux 7 で必要となるチェックサムです。Red Hat Enterprise Linux 5 以前のバージョンでは、チェックサムに SHA1 を選択します。
- **HTTP での公開:** リポジトリを HTTP で公開可能にします。このオプションは自動的に選択されます。
- **GPG キー:** このリポジトリの GPG キー。これまでに作成した PostgreSQL 9.5 GPG キーを選択します。

保存をクリックして、このリポジトリエントリを保存します。

リポジトリは同期プランを使用して定期的に更新されますが、ここで初期同期を実行します。PostgreSQL 9.5 リポジトリを選択して **同期開始** をクリックします。これでリポジトリと外部 PostgreSQL リポジトリとの同期が開始します。



注記

この同期の進捗状況では、**コンテンツ > 同期の状態** ページで確認できます。

CLI をご利用の場合

以下のコマンドを実行してリポジトリを作成します。

```
# hammer repository create \  
--name "PostgreSQL 9.5" \  
--content-type "yum" \  

```

```
--publish-via-http true \
--url http://yum.postgresql.org/9.5/redhat/rhel-7-x86_64/ \
--gpg-key "PostgreSQL 9.5" \
--product "PostgreSQL" \
--organization "ACME"
```

次にリポジトリを同期します。

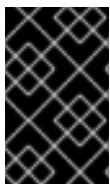
```
# hammer repository synchronize \
--name "PostgreSQL 9.5" \
--product "PostgreSQL" \
--organization "ACME"
```

これで PostgreSQL 9.5 の同期コピーが用意できました。さらに Puppet モジュールを含めて PostgreSQL サーバーを設定することができます。



注記

Web UI の製品ページには、URL ですべてのリポジトリを見つけ、自分のカスタム製品に追加するよう選択できる **リポジトリの検出** 機能があります。**リポジトリの検出** を使用して <http://yum.postgresql.org/9.5/redhat/> を検索し、Red Hat Enterprise Linux の各バージョンおよびアーキテクチャー用の PostgreSQL 9.5 リポジトリを一覧表示することができます。こうすることで、1つのソースから複数のリポジトリをインポートする時間を節約できます。



重要

Red Hat では、PostgreSQL から直接のアップストリーム RPM をサポートしていません。これらの RPM は同期プロセスのデモに使用されます。これらの RPM について問題がある場合は、PostgreSQL の開発者に連絡してください。

5.5. カスタム PUPPET リポジトリの作成

カスタム製品にも Puppet モジュールのリポジトリを含めることができます。これを含めると、ホストの状態構成を組み込む方法が提供されます。

まず、PostgreSQL 製品のリポジトリを作成します。

Web UI をご利用の場合

製品ページ (コンテンツ > 製品) を開いていることを確認します。PostgreSQL 製品をクリックするとリポジトリ一覧が表示されます。**リポジトリの作成** をクリックして、新規リポジトリのフォームを表示します。以下の詳細を入力します。

- **名前:** リポジトリの簡単な名前。 **PostgreSQL Puppet Modules** と入力します。
- **ラベル:** リポジトリの内部 ID。Red Hat Satellite 6 では、**名前** に入力した内容に基づいてこのフィールドに値が自動的に入力されます。
- **タイプ:** リポジトリのタイプ。 **puppet** を選択すると URL フィールドが表示されます。
- **URL:** ソースとして使用する外部リポジトリの URL。Puppet モジュールは手動でインポートしますが、Puppet モジュールの同期にはリポジトリソースを使用できます。

保存 をクリックして、このリポジトリエントリを保存します。

CLI をご利用の場合

以下のコマンドを実行して Puppet モジュールリポジトリを作成します。

```
# hammer repository create \  
--name "PostgreSQL Puppet Modules" \  
--content-type "puppet" \  
--product "PostgreSQL" \  
--organization "ACME"
```

これで Puppet モジュールのカスタムリポジトリができました。モジュールを追加します。

5.6. 個別 PUPPET モジュールの管理

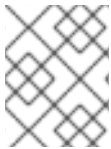
このシナリオでは、PostgreSQL 設定用の Puppet モジュールを手動でインポートします。

Puppet Forge サイト (<https://forge.puppetlabs.com/puppetlabs/postgresql>) からこのモジュールをダウンロードします。ブラウザでこのページを開き、**download latest tar.gz** をクリックしてローカルのファイルシステムに保存します。

Web UI をご利用の場合

PostgreSQL 製品のリポジトリ一覧ページを開いていることを確認します。PostgreSQL Puppet Modules リポジトリをクリックし、そのリポジトリの詳細ページを表示します。

Upload Puppet Module セクションに移動し、**参照** をクリックして、ダウンロードした PostgreSQL Puppet Module を選択してアップロードします。数秒すると、Satellite Server が **Content successfully uploaded** メッセージを表示します。



注記

Puppet モジュールを管理したり、製品から Puppet モジュールを削除するには、**Manage Puppet Modules** ページをクリックします。

CLI をご利用の場合

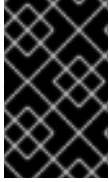
以下のコマンドで、使用中の Satellite Server のファイルシステムに Puppet モジュールをコピーします。

```
[user@client ~]$ scp ~/puppet_module.tar.gz root@satellite.example.com:~/.
```

Puppet モジュールを PostgreSQL Puppet Modules リポジトリにインポートします。

```
[root@satellite ~]# hammer repository upload-content \  
--path ~/puppet_module.tar.gz \  
--name "PostgreSQL Puppet Modules" \  
--product "PostgreSQL" \  
--organization "ACME"
```

これで RPM コンテンツを使用したサーバーをインストールして設定するのに使用する RPM コンテンツと Puppet モジュールの両方が含まれるカスタムリポジトリができました。



重要

Red Hat では、Puppet Forge からのモジュールをサポートしていません。PostgreSQL モジュールは、モジュール管理プロセスのデモのために使用されています。これらのモジュールに問題がある場合は、モジュール開発者に連絡してください。

5.7. PUPPET リポジトリの同期

Satellite Server は、アップロードした Puppet モジュールのリポジトリを作成するだけでなく、完全な Puppet モジュールリポジトリの同期ができます。この例では、Satellite Server は Puppet Forge リポジトリ全体を同期します。

Web UI をご利用の場合

コンテンツ > 製品 に移動し、**新製品** をクリックします。新しい製品のフォームが表示されます。以下の詳細情報を入力します。

- **名前:** 製品の簡単な名前。 **Puppet Forge** と入力します。
- **ラベル:** 製品の内部 ID。 Red Hat Satellite 6 では、 **名前** に入力した内容に基づいてこのフィールドに値が自動的に入力されます。
- **GPG キー:** 製品全体の GPG キー。これは空白にします。
- **同期プラン:** 製品の同期計画。これは、前の章で作成した **Example Plan** にアタッチすることができます。
- **説明:** 製品の簡単な説明。 **All modules from Puppet Forge** と入力します。

この情報を入力したら、**保存** をクリックします。

カスタム製品の作成後は、リポジトリ画面が表示されます。**リポジトリの作成** をクリックし、新しいリポジトリ用のフォームを表示します。以下の詳細情報を入力します。

- **名前:** リポジトリの簡単な名前。 **Puppet Forge Modules** と入力します。
- **ラベル:** リポジトリの内部 ID。 Red Hat Satellite 6 では、 **名前** に入力した内容に基づいてこのフィールドに値が自動的に入力されます。
- **タイプ:** リポジトリのタイプ。 **puppet** を選択すると URL フィールドが表示されます。
- **URL:** ソースとして使用する外部リポジトリの URL。 <http://forge.puppetlabs.com/> と入力します。

保存 をクリックして、このリポジトリエントリを保存します。

Puppet Forge Modules リポジトリを選択し、**同期開始** をクリックします。これで Puppet Forge から Satellite Server に全モジュールがインポートされます。

CLI をご利用の場合

製品を作成します。

```
# hammer product create \  
--name "Puppet Forge" \  
--sync-plan "Example Plan" \  

```

```
--description "All modules from Puppet Forge" \  
--organization "ACME"
```

Puppet Forge リポジトリを作成します。

```
# hammer repository create \  
--name "Puppet Forge Modules" \  
--content-type "puppet" \  
--product "Puppet Forge" \  
--organization "ACME" \  
--url http://forge.puppetlabs.com/
```

リポジトリを同期します。

```
# hammer repository synchronize \  
--name "Puppet Forge Modules" \  
--product "Puppet Forge" \  
--organization "ACME"
```



注記

Puppet Forge リポジトリには数千のモジュールが含まれてるため、同期には時間がかかる場合があります。



重要

Red Hat では、Puppet Forge からのモジュールをサポートしていません。モジュールは、同期プロセスのデモのために使用されています。これらのモジュールに問題がある場合は、モジュール開発者に連絡してください。

5.8. GIT リポジトリからの PUPPET MODULES の同期

Red Hat Satellite 6 には **pulp-puppet-module-builder** と呼ばれるユーティリティーが含まれており、これは **pulp-puppet-tools** RPM から他のシステムにインストールできます。このツールは Git リポジトリをチェックアウトし、全モジュールをビルドして、それらを Satellite 6 が同期できる構造で公開します。一般的な方法の 1 つは、Satellite Server 上でこのユーティリティーを実行し、ローカルディレクトリに公開して、そのディレクトリに対して同期するというものです。以下は例となります。

```
# mkdir /modules  
# chmod 755 /modules  
# pulp-puppet-module-builder \  
--output-dir=/modules \  
--url=git@mygitserver.com:mymodules.git \  
--branch=develop
```

これで、Git リポジトリの **develop** ブランチが **git@mygitserver.com:mymodules.git** からチェックアウトし、**/modules** に公開されます。このディレクトリを Satellite Server の新規リポジトリの URL (**file:///modules**) として追加します。以下は例となります。

Web UI をご利用の場合

カスタム製品を開き (ここでは例として **MyProduct** を使用) **リポジトリの作成** をクリックします。以下の詳細を入力します。

- **名前:** リポジトリの簡単な名前。 **Modules from Git** と入力します。
- **ラベル:** リポジトリの内部 ID。 Red Hat Satellite 6 では、 **名前** に入力した内容に基づいてこのフィールドに値が自動的に入力されます。
- **タイプ:** リポジトリのタイプ。 **puppet** を選択すると URL フィールドが表示されます。
- **URL:** ソースとして使用する外部リポジトリの URL。 **file:///modules** と入力します。

CLI をご利用の場合

Puppet Forge リポジトリを作成します。

```
# hammer repository create \
--name "Modules from Git" \
--content-type "puppet" \
--product "MyProduct" \
--organization "ACME" \
--url file:///modules
```

注記

リモートの HTTP サーバー上にモジュールを公開する場合でも同じプロセスを実行します。たとえば、Puppet モジュールを公開する標準ウェブホストとして **webserver.example.com** を使用する場合は、ホストにログインし以下のコマンドを実行します。

```
# mkdir /var/www/html/modules/
# chmod 755 /var/www/html/modules/
# pulp-puppet-module-builder \
--output-dir=/var/www/html/modules/ \
--url=git@mygitserver.com:mymodules.git \
--branch=develop
```

Satellite Server では、リポジトリの URL を <http://webserver.example.com/modules/> に設定します。

5.9. RED HAT SATELLITE でカスタムファイルタイプリポジトリの作成

Red Hat Satellite のカスタム製品には、カスタムファイルタイプのリポジトリを追加することができます。これにより、製品に任意ファイルを組み込む一般的な方法が提供されました。SSH 鍵やソースコードファイルの配布をはじめ、仮想マシンイメージや ISO ファイルなどの大容量ファイルの配布などに使用できます。

リポジトリにファイルをアップロードしたり、アップストリームの Satellite Server から同期することができます。ファイルをカスタムのファイルタイプリポジトリに置くと、特定のバージョンをコンテンツビューに追加して、様々な Capsule Server でファイルのリポジトリを利用可能にするなどの通常の Satellite 管理機能を使用できます。クライアントは、**curl -O** で、HTTP または HTTPS からファイルをダウンロードする必要があります。

Satellite Server のファイルタイプリポジトリはカスタム製品に対してのみ作成できますが、Satellite がインストールされているシステム、またはリモートの HTTP サーバーのディレクトリーに個別のファ

イルタイプリポジトリを作成して、Satellite のそのディレクトリーのコンテンツを同期できます。この方法は、Satellite リポジトリに追加するファイルが複数ある場合に便利です。

この例では、製品 (**My File Product**) を作成し、その製品にファイルタイプリポジトリ (**My Files**) を作成します。Satellite 以外のファイルタイプリポジトリの詳細は「[付録E リモートファイルタイプリポジトリの作成](#)」を参照してください。

Web UI をご利用の場合

1. カスタム製品の作成

コンテンツ > **製品** に移動し、**製品の作成** をクリックして、以下の詳細情報を入力します。

- **名前:** 製品の簡単な名前。 **My File Product** と入力します。
- **ラベル:** 製品の内部 ID。 Red Hat Satellite 6 では、 **名前** に入力した内容に基づいてこのフィールドに値が自動的に入力されます。
- **GPG キー:** 製品全体の GPG キー。これは空欄にします。
- **同期プラン:** 製品の同期計画。たとえば、前の章で作成した **Example Plan** です。
- **説明:** 製品の簡単な説明。 **My files** と入力します。

2. ファイルタイプリポジトリの作成

コンテンツ > **製品** に移動し、製品名をクリックします (この例では **My File Product**)。 **リポジトリ** タブで **新規リポジトリ** をクリックし、以下の詳細情報を入力します。

- **名前:** リポジトリの簡単な名前。 **My Files** と入力します。
- **ラベル:** リポジトリの内部 ID。 Red Hat Satellite 6 では、 **名前** に入力した内容に基づいてこのフィールドに値が自動的に入力されます。
- **タイプ:** リポジトリのコンテンツタイプ。 **file** を選択します。
- **アップストリーム URL:** ソースとして使用するアップストリームリポジトリの URL。ファイルを手動でアップロードする場合は、空白のままにできます。
- **SSL の検証:** アップストリームのリポジトリの SSL 証明書が信頼できる認証機関 (CA) によって署名されていることを確認したい場合のみ選択します。
- **アップストリームのユーザー名:** 認証に必要な場合は、アップストリームリポジトリのユーザー名を入力します。リポジトリに認証が必要ない場合はこのフィールドを空にします。
- **アップストリームのパスワード:** アップストリームリポジトリのパスワードを入力します。リポジトリに認証が必要ない場合はこのフィールドを空にします。

3. 保存 をクリックして、このリポジトリエントリーを保存します。

CLI をご利用の場合

1. カスタム製品の作成

```
# hammer product create \  
--name "My File Product" \  
--sync-plan "Example Plan" \  

```

```
--description "My files" \  
--organization "ACME"
```

オプションのパラメーター

gpg-key gpg_key_name	検索するキー名
gpg-key-id gpg_key_id	GPG キー数値 ID
sync-plan sync_plan_name	検索する同期プラン名
sync-plan-id sync_plan_id	同期プランの数値 ID

2. ファイルタイプリポジトリの作成

```
# hammer repository create \  
--name "My Files" \  
--content-type "file" \  
--product "My File Product" \  
--organization "ACME"
```

オプションのパラメーター

checksum-type sha_version	リポジトリのチェックサムです。現在、'sha1' および 'sha256' がサポートされています。
download-policy policy_name	yum リポジトリのダウンロードポリシーです ('immediate'、'on_demand'、または 'background')。
gpg-key gpg_key_name	検索するキー名
gpg-key-id gpg_key_id	GPG キー数値 ID
mirror-on-sync boolean	同期する場合に、このリポジトリをソースからミラーリングし、古い RPM を削除する必要がありますか? true または false 、 yes または no 、もしくは 1 または 0 に設定します。
publish-via-http boolean	HTTP を使用して公開する必要がありますか? true または false 、 yes または no 、 1 または 0 に設定します。
upstream-username repository_username	認証に必要な場合は、アップストリームリポジトリユーザー
upstream-password repository_password	アップストリームリポジトリユーザーのパスワード

<code>url source_repo_url</code>	ソースリポジトリの URL
<code>verify-ssl-on-sync boolean</code>	URL の SSL 証明書が信頼できる CA によって署名されているのを Katello が確認する必要がありますか? true または false 、 yes または no 、もしくは 1 または 0 に設定します。

5.10. RED HAT SATELLITE へのカスタムファイルタイプリポジトリへのファイルのアップロード

Web UI をご利用の場合

1. コンテンツ > **製品** に移動します。
2. カスタム製品の名前を選択します (例: **My File Product**)。
3. ファイルタイプリポジトリの名前を選択します (例: **My Files**)。
4. **参照** をクリックして、アップロードするファイルを選択します。
5. **アップロード** をクリックして、選択したファイルを Satellite Server にアップロードします。
6. リポジトリを公開した URL を開いて、ファイルを表示します。

CLI をご利用の場合

```
# hammer repository upload-content \
--product "My File Product" \
--name "My Files" \
--organization "ACME" \
--path example_file
```

`--path` オプションは、ファイル、ファイルディレクトリ、またはファイルの glob 表現を示します。glob は、一重引用符または二重引用符でエスケープする必要があります。

5.11. RED HAT SATELLITE のカスタムファイルタイプリポジトリからホストにファイルのダウンロード

`curl -o` を実行して、HTTPS (**HTTP での公開** リポジトリを選択している場合は HTTP でも可能) でクライアントにファイルをダウンロードします。

前提条件

- カスタムファイルタイプリポジトリがあります。詳細は「[Red Hat Satellite でカスタムファイルタイプリポジトリの作成](#)」を参照してください。
- ファイルタイプリポジトリから、クライアントにダウンロードするファイル名が必要です。
- HTTPS を使用するには、クライアントで以下の証明書が必要です。

1. **katello-server-ca.crt**。これは、安全な方法で取得する必要があります。<https://satellite.example.com/pub/> から HTTPS 経由で、ブラウザからエクスポートまたはダウンロードできます。
2. 組織のデバッグ証明書。詳細は「[組織のデバッグ証明書の作成](#)」を参照してください。

Web UI をご利用の場合

1. コンテンツ > **製品** に移動します。
2. カスタム製品の名前を選択します (例: **My File Product**)。
3. ファイルタイプリポジトリの名前を選択します (例: **My Files**)。
4. **HTTP での公開** が有効になっているかどうかを確認します。有効になっていない場合は、HTTPS を使用するための証明書が必要です。
5. リポジトリを公開した URL をコピーします。

CLI をご利用の場合

1. ファイルタイプリポジトリを一覧表示します。

```
# hammer repository list --content-type file
---|-----|-----|-----|----
ID | NAME      | PRODUCT          | CONTENT TYPE | URL
---|-----|-----|-----|----
7  | My Files  | My File Product  | file         |
---|-----|-----|-----|----
```

2. リポジトリ情報を表示します。

```
# hammer repository info --name "My Files" --product "My File
Product" --organization-id 1
```

- HTTP が有効な場合、出力は以下のようになります。

```
Publish Via HTTP:  yes
Published At:      http://satellite.example.com/pulp/isos/uuid/
```

- HTTP が無効な場合、出力は以下のようになります。

```
Publish Via HTTP:  no
Published At:      https://satellite.example.com/pulp/isos/uuid/
```

クライアントに、適切な **HTTP** または **HTTPS** の形式でコマンドの入力:

- HTTP の場合:

```
# curl -O satellite.example.com/pulp/isos/uuid/my_file
```

- HTTPS の場合:

```
# curl -O --cert ./Default\ Organization-key-cert.pem --cacert
```

```
katello-server-ca.crt satellite.example.com/pulp/isos/uuid/my_file
```

5.12. ローカルディレクトリーにカスタムのファイルタイプリポジトリーの作成

pulp-manifest を使用して、Satellite Server の外部にあるファイルのディレクトリーから、カスタムファイルタイプリポジトリーを作成します。その後、Satellite Server にファイルを同期します。ファイルタイプリポジトリーにファイルを追加すると、他のリポジトリーと同じようにファイルを操作できます。

この手順は、Satellite がインストールされているベースシステムのディレクトリーにリポジトリーを設定する方法を説明します。リモートサーバーのディレクトリーへファイルタイプのリポジトリーを作成する方法は「[付録E リモートファイルタイプリポジトリーの作成](#)」を参照してください。

ローカルディレクトリーへのファイルタイプリポジトリーの作成:

1. サーバーおよび Satellite Tools リポジトリーが有効になっていることを確認します。

```
# subscription-manager repos --enable=rhel-7-server-rpms \  
--enable=rhel-7-server-satellite-tools-6.3-rpms
```

2. Pulp マニフェストパッケージをインストールします。

```
# yum install python-pulp-manifest
```

3. HTTP サーバーのパブリックフォルダーのファイルタイプリポジトリーとして使用するディレクトリーを作成します。

```
# mkdir my_file_repo
```

4. ディレクトリーにファイルを追加して、テストファイルを作成します。

```
# touch my_file_repo/test.txt
```

5. Pulp マニフェストコマンドを入力して、マニフェストを作成します。

```
# pulp-manifest my_file_repo
```

6. マニフェストが作成されたことを確認します。

```
# ls my_file_repo  
PULP_MANIFEST test.txt
```

ローカルディレクトリーのファイルタイプリポジトリーからのファイルのインポート:

1. Satellite Server にカスタム製品が存在することを確認します。この例では **My File Product** を使用します。
2. ファイルタイプリポジトリーの作成:
 - a. Satellite Web UI で、**コンテンツ > 製品** に移動します。

- b. 製品の名前を選択します (この例では **My File Product**)。
- c. リポジトリ タブで **新規リポジトリ** を選択し、以下の詳細を入力します。
 - **名前**: リポジトリの簡単な名前。 **My Files** と入力します。
 - **ラベル**: リポジトリの内部 ID。 Red Hat Satellite 6 では、 **名前** に入力した内容に基づいてこのフィールドに値が自動的に入力されます。
 - **タイプ**: リポジトリのコンテンツタイプ。 **file** を選択します。
 - **アップストリーム URL**: ソースとして使用するリポジトリを使用したローカルディレクトリへのパス (**file:///my_file_repo** の形式)。
 - **SSL の検証**: このフィールドに何も入力されていないことを確認します。
 - **アップストリームのユーザー名**: このフィールドが何も入力されていないことを確認します。
 - **アップストリームのパスワード**: このフィールド何も入力されていないことを確認します。
3. **保存** をクリックして、このリポジトリエントリを保存します。
4. ファイルタイプリポジトリのアップデート:
 - a. **コンテンツ > 製品** に移動します。
 - b. 製品の名前を選択します (この例では **My File Product**)。
 - c. アップデートするリポジトリの名前を選択します (この例では **My Files**)。
 - d. **アクションの選択** メニューから **同期開始** を選択します。
 - e. リポジトリを公開した URL を開いて、ファイルを表示します。

5.13. 本章のまとめ

本章では、Red Hat 以外のコンテンツ向けのカスタムリポジトリを作成する方法を説明しました。これには、RPM ファイル、Puppet モジュール、カスタムファイルタイプが含まれます。

これで、ベースコンテンツがすべて Satellite Server の DML にインポートされました。これをアプリケーションのライフサイクルに使用できます。

次章では、ACME の開発および実稼働プロセスに一致するようにアプリケーションのライフサイクルを開発する方法を説明します。

第6章 アプリケーションライフサイクルの作成

「[アプリケーションライフサイクルの定義](#)」では、アプリケーションライフサイクルを定義し、それが Red Hat Satellite 6 にとって重要な理由を説明しました。本章では、アプリケーションライフサイクルのプランニングと Red Hat Satellite 6 での実装を説明します。

6.1. アプリケーションライフサイクルの再検討

ここまで見てきたように、アプリケーションライフサイクルはあるステージでシステムがどのように表示されるかを定義します。しかし、実際のライフサイクルは、組織と、組織が特定の実稼働チェーンを構成する方法によって異なります。

たとえば、メールサーバーの場合は、実際の使用における実稼働レベルのサーバーと、最新のメールサーバーパッケージをテストするテストサーバーという、単純なアプリケーションライフサイクルのみを必要とします。テストサーバーが初期段階をパスすれば、実稼働レベルのサーバーで新パッケージを使用するように設定できます。

別の例としては、ソフトウェア製品の開発ライフサイクルがあります。開発環境でソフトウェアの新しい部分を開発し、品質保証環境でこれをテストしてベータ版としてプレリリースした後、実稼働レベルのアプリケーションとしてリリースします。

各アプリケーションライフサイクルでは、**環境** と呼ばれる段階を使用します。各環境は、システムに対して特定の状態として機能します。各環境は前の環境に続くもので、アプリケーションライフサイクルを構成することになる環境チェーンを作り出します。アプリケーションライフサイクルはそれぞれ、初期の **ライブラリー** が備わった状態で開始し、これは Definitive Media Library (DML) 内の中央ソースとして機能します。ライブラリー環境には、ここまでの章で同期した全コンテンツが含まれています。ここからは、ライブラリー環境からリンクされる新たな環境を作成します。

6.2. 新規アプリケーションライフサイクルの作成

本書のシナリオでは、ACME の Exampleware を開発してリリースする単純なアプリケーションライフサイクルを作成します。初期環境にライブラリー環境を使用し、その後 **開発**、**テスト**、および **実稼働** の3つの環境をこの順に続けます。

Web UI をご利用の場合

コンテンツ > ライフサイクル環境 に移動します。お使いのアプリケーションライフサイクルにおける現在の環境が表示されます。この時点では、ライブラリー環境しか存在しません。

新規環境パス をクリックして、新規のアプリケーションライフサイクルを起動します。ライブラリーに新規環境を追加するフォームが表示されます。**名前** に **Development** と入力すると、**ラベル** が自動的に完了します。**説明** に **Environment for ACME's Development Team** と入力します。**保存** をクリックします。

環境一覧に戻ります。開発環境の新たなテーブルが表示されます。このテーブルは、新規のアプリケーションライフサイクルを示します。これで残りの環境をアプリケーションライフサイクルに追加できます。

新規テーブルの上にある **新規環境の追加** をクリックします。**名前** に **Testing**、**説明** に **Environment for ACME's Quality Engineering Team** と入力して、**保存** をクリックします。

再度 **新規環境の追加** をクリックします。**名前** に **Production**、**説明** に **Environment for ACME's Product Releases** と入力して、**保存** をクリックします。

これで **Development**、**Testing**、および **Production** という 3 つの環境が揃ったテーブルのアプリケーションライフサイクルが表示されます。

CLI をご利用の場合

各環境で **hammer lifecycle-environment create** を実行します。前の環境を指定するには、以下のように **--prior** オプションを使用します。

```
# hammer lifecycle-environment create \
--name "Development" \
--description "Environment for ACME's Development Team" \
--prior "Library" \
--organization "ACME"
# hammer lifecycle-environment create \
--name "Testing" \
--description "Environment for ACME's Quality Engineering Team" \
--prior "Development" \
--organization "ACME"
# hammer lifecycle-environment create \
--name "Production" \
--description "Environment for ACME's Product Releases" \
--prior "Testing" \
--organization "ACME"
```

hammer lifecycle-environment paths コマンドでチェーンを確認できます。

```
# hammer lifecycle-environment paths --organization "ACME"
-----
LIFECYCLE PATH
-----
Library >> Development >> Testing >> Production
-----
```

6.3. CAPSULE SERVER へのライフサイクル環境の追加

以下の手順は、Capsule Server を使用してコンテンツをプロビジョニングする環境に対するオプションです。

Capsule Server でコンテンツ機能が有効な場合は、環境を追加する必要があります。環境を追加すると、Capsule Server で Satellite Server のコンテンツを同期し、コンテンツをホストシステムに提供できます。

Capsule Server は、Satellite Server の Hammer CLI、または Web UI を使用して設定できます。

Web UI をご利用の場合

インフラストラクチャー > **Capsule** に移動し、Capsule を選択します。

編集 をクリックします。

ライフサイクル環境タブで **Env** を選択します。

カプセルのコンテンツを同期するには、概要タブの **同期** ボタンをクリックします。

以下のいずれかのオプションを選択します。

- 最適同期
- 完全同期

CLI をご利用の場合

Capsule Server の一覧を表示し、ID を書き留めます。

```
# hammer capsule list
```

その ID を使用して、Capsule Server の詳細を確認します。

```
# hammer capsule info --id capsule_id_number
```

利用可能なライフサイクル環境を確認し、環境 ID を書き留めます。

```
# hammer capsule content available-lifecycle-environments \  
--id capsule_id_number
```

利用可能なライフサイクル環境は Capsule Server に対して利用可能ですが、現在接続されていません。

ライフサイクル環境を Capsule Server に追加します。

```
# hammer capsule content add-lifecycle-environment \  
--id capsule_id_number --environment-id environment_id_number
```

Capsule Server に追加する各ライフサイクル環境に対して手順を繰り返します。

Satellite Server 環境のすべてのコンテンツを Capsule Server に同期するには、以下のコマンドを実行します。

```
# hammer capsule content synchronize --id capsule_id_number
```

Satellite Server 環境の特定のライフサイクル環境を Capsule Server と同期するには、以下のコマンドを実行します。

```
# hammer capsule content synchronize --id external_capsule_id_number \  
--environment-id environment_id_number
```

ライフサイクル環境への作業は『アーキテクチャーガイド』の「[ライフサイクル環境](#)」を参照してください。

6.4. アプリケーションライフサイクルでのコンテンツのプロモーション

アプリケーションライフサイクルのチェーンでは、コンテンツはある環境から次の環境に移動します。このプロセスは **プロモーション** と呼ばれます。プロモーションはアプリケーションライフサイクルでコンテンツを管理する基礎となるもので、これを理解することは重要です。

本ガイドのシナリオでは、ACME の Exampleware の開発が完了しており、実稼働しているとします。ACME は Exampleware コンテンツを RPM ファイルにパッケージ化し、これは別製品の一部となります。この状態では、ACME は Exampleware のバージョン 1.0 をリリースしており、このため **exampleware-1.0-0.noarch.rpm** が実稼働環境にあることとなります。

ACME では、Exampleware のパッチをリリースするために、Exampleware バージョン 1.1 の開発を開始します。この例では、アプリケーションライフサイクルの環境に以下の Exampleware のバージョンが含まれます。

開発	テスト	実稼働
exampleware-1.1-0.noarch.rpm	exampleware-1.0-0.noarch.rpm	exampleware-1.0-0.noarch.rpm

ACME では、パッチ開発の完了後、RPM をテスト環境にプロモートして、ACME の品質保証エンジニアチームがレビューできるようにします。この時点では、アプリケーションライフサイクルに以下のバージョンが含まれます。

開発	テスト	実稼働
exampleware-1.1-0.noarch.rpm	exampleware-1.1-0.noarch.rpm	exampleware-1.0-0.noarch.rpm

品質保証エンジニアチームがレビューを行う間、開発チームは Exampleware 2.0 の作業に着手します。このため、アプリケーションライフサイクルは以下のようになります。

開発	テスト	実稼働
exampleware-2.0-0.noarch.rpm	exampleware-1.1-0.noarch.rpm	exampleware-1.0-0.noarch.rpm

品質保証エンジニアチームがパッチのレビューを完了し、Exampleware 1.1 をリリースする準備が完了しました。ACME は 1.1 を実稼働環境にプロモートします。

開発	テスト	実稼働
exampleware-2.0-0.noarch.rpm	exampleware-1.1-0.noarch.rpm	exampleware-1.1-0.noarch.rpm

開発チームが Exampleware 2.0 の作業を完了し、これをテスト環境にプロモートします。

開発	テスト	実稼働
exampleware-2.0-0.noarch.rpm	exampleware-2.0-0.noarch.rpm	exampleware-1.1-0.noarch.rpm

最後に品質保証エンジニアチームがこのパッケージのレビューを行います。レビューが完了したら、パッケージを実稼働環境にプロモートします。

開発	テスト	実稼働
exampleware-2.0-0.noarch.rpm	exampleware-2.0-0.noarch.rpm	exampleware-2.0-0.noarch.rpm

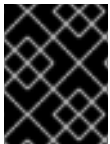
この例では、ACME の開発プロセスを Red Hat Satellite 6 アプリケーションライフサイクルにマッピングする手順を説明します。各環境には、Red Hat Satellite 6 に登録したシステムが含まれ、そのシステ

ムがアクセスできるのは、各環境に関連するリポジトリに限られます。ある環境から次の環境にパッケージをプロモートすると、プロモート先のリポジトリはパッケージの新バージョンを受け取ります。この結果、プロモート先の環境にあるシステムはパッケージを新バージョンに更新することができます。

次章では、**コンテンツビュー** という概念を説明します。これは、ライブラリーから収集されフィルタリングされたコンテンツのコレクションで、リポジトリに公開されたものです。Red Hat のリポジトリとこれまでに同期したカスタムリポジトリはソースコンテンツとして機能しますが、コンテンツビューは作成したリポジトリがどのように表示されるかをカスタマイズするものです。コンテンツビューにはパッケージが含まれ、Satellite Server はアプリケーションライフサイクルの各環境にコンテンツビューをプロモートします。ある環境でコンテンツビュー (バージョン 1.0) を使用していても、次に新バージョン (バージョン 1.1) のコンテンツビューを受け取ります。この場合、バージョン 1.0 のリポジトリは、Exampleware RPM の新バージョンが含まれているバージョン 1.1 のリポジトリに置き換えられます。

6.5. RED HAT SATELLITE CAPSULE SERVER へのライフサイクル環境の追加

新たに作成した Red Hat Satellite Capsule Server でコンテンツ機能が有効な場合は、Capsule Server に、Satellite Server からコンテンツを同期し、ホストシステムにコンテンツを同期するために有効にする環境を追加する必要があります。



重要

Satellite Capsule Server は、Satellite Server のコマンドラインインターフェース (CLI) で設定します。Satellite Server 上ですべての **hammer** コマンドを実行します。

Satellite Capsule Server への環境の追加:

1. root で Satellite Server CLI にログインします。
2. 一覧から使用する Red Hat Satellite Capsule Server を選択し、その **ID** を書き留めます。

```
# hammer capsule list
```

Satellite Capsule Server の詳細を確認するには、以下のコマンドを入力します。

```
# hammer capsule info \  
--id capsule_id_number
```

3. Red Hat Capsule Server に利用できるライフサイクル環境の一覧を確認し、**環境 ID** を書き留めます。

```
# hammer capsule content available-lifecycle-environments \  
--id capsule_id_number
```

ここで、

available-lifecycle-environments は、Satellite Capsule で利用できるライフサイクル環境ですが、現時点では Satellite Capsule にアタッチされていません。

4. Satellite Capsule Server にライフサイクル環境を追加します。

```
# hammer capsule content add-lifecycle-environment \
--id capsule_id_number \
--environment-id environment_id_number
```

ここで、

- **capsule_id_number** は、Satellite Capsule Server の ID 番号です。
- **environment_id_number** は、ライフサイクル環境の ID 番号です。

Capsule Server に追加するすべてのライフサイクル環境に対してこの手順を繰り返します。

5. Satellite Server の環境にあるコンテンツを Satellite Capsule Server に同期します。

```
# hammer capsule content synchronize \
--id capsule_id_number
```

外部 Satellite Capsule Server に複数のライフサイクル環境がある場合、同期が必要なライフサイクル環境は 1 つだけとなります。環境 ID 番号を指定して、特定の環境にターゲットを絞ることができます。

```
#hammer capsule content synchronize \
--id external_capsule_id_number \
--environment-id environment_id_number
```

6.6. コンテンツビューのプロモート

コンテンツビューと、2 つ以上のライフサイクル環境から構成される環境パスを作成したら、必要に応じてコンテンツビューを次の環境にプロモートできます。つまり、指定された環境に存在するコンテンツビューの最新バージョンが、ライフサイクル環境パスの次の環境にプロモート (つまり、コピー) されます。

コンテンツビューは、そのバージョンが存在しない環境にプロモートできます。ライフサイクル環境パスの次の環境が自動的に提示されますが、この値は上書きでき、必要に応じて別の環境にプロモートできます。

コンテンツビューのプロモート:

1. メインメニューで、**コンテンツ > コンテンツビュー** をクリックします。
2. **名前** 列で、プロモートするコンテンツビューの名前をクリックします。
3. **バージョン** タブで最新バージョンを選択し、**プロモート** をクリックします。
4. コンテンツビューをプロモートするプロモーションパスを指定し、適切なライフサイクル環境を選択し、**バージョンのプロモート** をクリックします。
5. プロモーションが完了したら、**バージョン** タブが更新され、コンテンツビューの新しいステータスが表示します。

6.7. SATELLITE SERVER からのライフサイクル環境の削除

以下の手順では、Red Hat Satellite からライフサイクル環境を削除する方法を説明します。

ライフサイクル環境の削除:

1. メインメニューで、**コンテンツ > ライフサイクル環境** をクリックします。
2. 削除するライフサイクル環境の名前をクリックし、**環境の削除** をクリックします。
3. 確認ダイアログボックスで、**削除** をクリックして環境を削除します。

**注記**

環境パスの最新の環境だけが削除できます。たとえば、3つの環境 **Library**、**Dev**、**Prod** がこの順序で存在する場合は、**Dev** を削除する前に **Prod** を削除する必要があります。**Library** 環境は削除できません。

6.8. CAPSULE SERVER からのライフサイクル環境の削除

ライフサイクル環境を Capsule Server から削除する理由は複数あります。以下のような理由があります。

- ライフサイクル環境とホストシステムとの関連性がなくなった場合
- ライフサイクル環境が Capsule Server に誤って追加された場合

Capsule Server からのライフサイクル環境の削除:

1. root ユーザーで Satellite Server CLI にログインします。
2. 一覧から使用する Capsule Server を選択し、その **ID** を書き留めます。

```
# hammer capsule list
```

Capsule Server の詳細を確認するには、以下のコマンドを実行します。

```
# hammer capsule info \  
--id capsule_id_number
```

3. Capsule Server に現在アタッチされているライフサイクル環境の一覧を確認し、**環境 ID** を書き留めます。

```
# hammer capsule content lifecycle-environments \  
--id capsule_id_number
```

4. Capsule Server からのライフサイクル環境を削除します。

```
# hammer capsule content remove-lifecycle-environment \  
--id capsule_id_number \  
--environment-id environment_id_number
```

ここで、

- **capsule_id_number** は、Capsule Server の ID 番号です。
- **environment_id_number** は、ライフサイクル環境の ID 番号です。

Capsule Server から削除するすべてのライフサイクル環境に対してこの手順を繰り返します。

5. Satellite Server の環境にあるコンテンツを Capsule Server に同期します。

```
# hammer capsule content synchronize \  
--id capsule_id_number
```

6.9. 本章のまとめ

本章では、アプリケーションライフサイクルの概念と、それが Red Hat Satellite 6 のコンテンツ管理にどのように適用されるかを説明しました。また、Red Hat Satellite 6 がアプリケーションライフサイクルの各環境にコンテンツをプロモートする方法も説明しました。

次章では、コンテンツビューと、コンテンツビューを使用して、既存の DML からリポジトリをフィルタリングする方法を説明します。

第7章 コンテンツビューの管理

本章では、様々なコンテンツビューを作成し、各種フィルターを適用する方法を説明します。

7.1. コンテンツビューの理解

Red Hat Satellite 6 では、コンテンツビューを使用して Definitive Media Library (DML) のコアリポジトリからカスタマイズリポジトリを作成します。使用するリポジトリを定義し、特定のフィルターをコンテンツに適用することで、これが作成されます。このフィルターにはパッケージフィルター、パッケージグループフィルター、およびエラータフィルターが含まれます。コンテンツビューは、特定の環境が使用するソフトウェアのバージョンを定義する方法として使用されます。前章でも説明したように、実稼働環境で古いバージョンのパッケージを含むコンテンツビューが使用され、開発環境で新しいバージョンのパッケージを含むコンテンツビューが使用されることがあります。

コンテンツビューは各環境でリポジトリセットを作成し、Satellite Server がこれを保存、管理します。アプリケーションライフサイクルの次の環境にコンテンツビューをプロモートすると、それに対応する Satellite Server のリポジトリがパッケージを更新して公開します。たとえば、Exampleware パッケージが含まれているコンテンツビューを使用するとします。

	開発	テスト	実稼働
コンテンツビューのバージョンとコンテンツ	バージョン 2 - exampleware-1.1-0.noarch.rpm	バージョン 1 - exampleware-1.0-0.noarch.rpm	バージョン 1 - exampleware-1.0-0.noarch.rpm

テストと実稼働のリポジトリには **exampleware-1.0-0.noarch.rpm** パッケージが含まれています。コンテンツビューのバージョン 2 を開発環境からテスト環境にプロモートすると、テスト環境のリポジトリが再作成され、**exampleware-1.1-0.noarch.rpm** パッケージが含まれるようになります。

	開発	テスト	実稼働
コンテンツビューのバージョンとコンテンツ	バージョン 2 - exampleware-1.1-0.noarch.rpm	バージョン 2 - exampleware-1.1-0.noarch.rpm	バージョン 1 - exampleware-1.0-0.noarch.rpm

こうすることで、システムは特定の環境専用となり、その環境が新しいコンテンツビューを使用するには更新を受け取ることができます。

フィルタリングとスナップショットを行うコンテンツビューを作成する一般的なフローは次のようになります。

1. コンテンツビューを作成します。
2. コンテンツビューに使用するリポジトリと Puppet モジュールを追加します。
3. 任意で、コンテンツビューのコンテンツを調整するフィルターを 1 つまたは複数作成します。
4. コンテンツビューを公開します。
5. コンテンツホストをコンテンツビューにアタッチします。

6. 任意で、コンテンツビューを別の環境にプロモートします。



注記

コンテンツビューでリポジトリを割り当てないと、`/etc/yum.repos.d/redhat.repo` ファイルは空になり、登録済みのシステムで更新を受け取ることができません。



注記

システムを関連付けるコンテンツビューは 1 つだけにすることができます。複数のコンテンツビューにシステムを関連付けるには、複合コンテンツビューを作成します。詳細は「[複合コンテンツビューの作成](#)」を参照してください。

7.2. 標準コンテンツビュー

標準コンテンツビューは、1 台のホストに関連付けることができる 1 つのコンテンツビューです。

7.2.1. シンプルなコンテンツビューの作成

この例では、2 つのリポジトリが含まれ、フィルターは設定されていないシンプルなコンテンツビューを作成します。Red Hat Enterprise Linux リポジトリと Satellite Tools リポジトリを使用します。このコンテンツビューには Red Hat Enterprise Linux の全 RPM が含まれるので、公開には数分かかる場合があります。

Web UI をご利用の場合

コンテンツ > コンテンツビュー に移動し、**新規ビューの作成** をクリックします。ビューの詳細 フォームが表示されるので、以下の情報を入力します。

- **名前:** ビューの簡単な名前。 **Base** と入力します。
- **ラベル:** ビューの内部 ID。Red Hat Satellite 6 では、**名前** に入力した内容に基づいてこのフィールドに値が自動的に入力されます。
- **説明:** ビューの簡単な説明。 **Base operating system** と入力します。
- **複合ビュー?:** 複合コンテンツビューを使用するかどうかを定義します。チェックを外しておきます。

保存 をクリックします。

これでコンテンツビューの新規エントリが作成され、リポジトリを追加できるようになりました。Red Hat Enterprise Linux 7 Server RPMs (Kickstart RPMs ではありません) と Red Hat Satellite Tools のリポジトリを選択し、**リポジトリの追加** をクリックします。これらのリポジトリの全パッケージがコンテンツビューに追加されます。

これでコンテンツビューを公開する準備ができました。**バージョン** に移動し、**新規バージョンの公開** をクリックします。Satellite Server によって新規バージョン (バージョン 1) の詳細が提供されます。**説明** に、このバージョンの説明を入力できるので、新規コンテンツビューの変更点を記録しておくのが便利です。**Initial content view for our operating system** と入力し、**保存** をクリックします。

Satellite Server がビューの新バージョンを作成し、ライブラリー環境に公開します。

CLI をご利用の場合

リポジトリ ID の一覧を取得します。

```
# hammer repository list --organization "ACME"
```

この例では、2つのリポジトリはそれぞれ ID に 1 と 2 を使用しています。

ID	名前
1	Red Hat Enterprise Linux 7 Server RPMs x86_64 7Server
2	Red Hat Satellite Tools 6.3 for RHEL 7 Server RPMs x86_64

コンテンツビューを作成し、リポジトリに追加します。

```
# hammer content-view create \
  --name "Base" \
  --description "Base operating system" \
  --repository-ids 1,2 \
  --organization "ACME"
```

ビューを公開します。

```
# hammer content-view publish \
  --name "Base" \
  --description "Initial content view for our operating system" \
  --organization "ACME"
```

Satellite Server がビューの新バージョンを作成し、ライブラリー環境に公開します。

7.2.2. Puppet モジュールを含むコンテンツビューの作成

この例では、1つのリポジトリが含まれ、フィルターは設定されていないコンテンツビューを作成します。PostgreSQL リポジトリを使用します。

Web UI をご利用の場合

コンテンツ > コンテンツビュー に移動し、**新規ビューの作成** をクリックします。**ビューの詳細** フォームが表示されるので、以下の情報を入力します。

- **名前:** ビューの簡単な名前。 **Database** と入力します。
- **ラベル:** ビューの内部 ID。Red Hat Satellite 6 では、**名前** に入力した内容に基づいてこのフィールドに値が自動的に入力されます。
- **説明:** ビューの簡単な説明。 **PostgreSQL Database** と入力します。
- **複合ビュー?:** 複合コンテンツビューを使用するかどうかを定義します。チェックを外しておきます。

保存 をクリックします。

これでコンテンツビューの新規エントリーが作成されます。PostgreSQL のリポジトリを選択し、**リポジトリの追加** をクリックします。PostgreSQL リポジトリの全パッケージがコンテンツビューに追加されます。

Puppet モジュール に移動し、**新規モジュールの追加** をクリックします。インポート済みの Puppet モジュールが表示されます。**postgresql** モジュールに移動し、**バージョンの選択** をクリックします。

最新を使用 のエントリーに移動し、**アクション** コラムの **バージョンの選択** をクリックします。

Puppet モジュールがコンテンツビューに追加されました。新しいバージョンを公開します。

バージョン に移動し、**新規バージョンの公開** をクリックします。説明 に **Initial RPMs and Puppet module for database** と入力し、**保存** をクリックします。

Satellite Server がビューの新バージョンを作成し、ライブラリー環境に公開します。

CLI をご利用の場合

リポジトリ ID の一覧を取得します。

```
# hammer repository list --organization "ACME"
```

必要になるのは PostgreSQL RPM リポジトリのみで、Puppet モジュールリポジトリは必要ありません。この例では、PostgreSQL RPMs の ID に 4 を使用します。

ID	名前
4	PostgreSQL 9.5

コンテンツビューを作成し、リポジトリに追加します。

```
# hammer content-view create \
  --name "Database" --description "PostgreSQL Database" \
  --repository-ids 4 \
  --organization "ACME"
```

ビューを公開します。

```
# hammer content-view publish \
  --name "Database" \
  --description "Initial RPMs and Puppet module for database" \
  --organization "ACME"
```

Satellite Server がビューの新バージョンを作成し、ライブラリー環境に公開します。

7.3. 複合コンテンツビュー

複合コンテンツビューは、複数のコンテンツビューのコンテンツを組み合わせます。たとえば、ベース OS とアプリケーションの管理に別々のコンテンツビューを使用していたとします。複合コンテンツビューを使用すると、この 2 つのコンテンツビューのコンテンツを新たなリポジトリに統合できま

す。元のコンテンツビューのリポジトリはそのまま存在しますが、組み合わせられたコンテンツには新規リポジトリが使用されます。

本ガイドのシナリオでは、企業が異なるデータベースサーバーをサポートするアプリケーションを開発しているとします。一般的なアプリケーションスタックは以下のようになります。

Exampleware スタック
アプリケーション
データベース
オペレーティングシステム

この企業が以下の 4 つのコンテンツビューを作成するとします。

- Red Hat Enterprise Linux (オペレーティングシステム)
- PostgreSQL (データベース)
- MariaDB (データベース)
- Exampleware (アプリケーション)

すると、次に 2 つの複合コンテンツビューを作成できます。1 つ目には PostgreSQL データベースを使います。

複合コンテンツビュー 1: PostgreSQL 上の Exampleware
Exampleware (アプリケーション)
PostgreSQL (データベース)
Red Hat Enterprise Linux (オペレーティングシステム)

もう 1 つには MariaDB を使います。

複合コンテンツビュー 2: MariaDB 上の Exampleware
Exampleware (アプリケーション)
MariaDB (データベース)
Red Hat Enterprise Linux (オペレーティングシステム)

これで各コンテンツビューは別々に管理、公開されます。アプリケーションスタックの新バージョンを作成すると、複合コンテンツビューの新バージョンを公開することになります。



重要

複合コンテンツビューは、各リポジトリで1つしか許可されません。たとえば、同じリポジトリを使用した2つのコンテンツビューを2つを追加しようとする、Satellite Server はエラーをレポートします。

7.3.1. 複合コンテンツビューの作成

既存のコンテンツビューを2つ組み合わせたコンテンツビューを ACME 用に作成します。

Web UI をご利用の場合

コンテンツ > コンテンツビュー に移動して **新規ビューの作成** をクリックします。以下の詳細を入力します。

- **名前: Stack**
- **説明: A stack that includes a base operating system and a database**
- **複合ビュー?:** チェックを入れます。

複合コンテンツビュー用のコンテンツビュー一覧が表示されます。Base と Database の両方のコンテンツビューを選択します。Base コンテンツビューには2つのバージョンが含まれているため、そのいずれかを選択します。ここでは、エラータフィルターが含まれているバージョン2を選択します。コンテンツビューとバージョンを選択したら、**コンテンツビューの追加** をクリックします。

新バージョンの公開 をクリックして、複合コンテンツビューを公開します。説明に **Initial version of Stack** と入力して **保存** をクリックします。

公開が完了すると、コンテンツ コラムに、追加した全コンテンツビューのパッケージ、エラータ、Puppet モジュールの数がレポートされます。

この複合コンテンツビューを **開発**、**テスト**、**実稼働** の各環境に **プロモート** します。

通常のコンテンツビューと同じように、複合コンテンツビューもアプリケーションライフサイクルの各環境で公開、プロモートされていることが確認できます。

CLI をご利用の場合

複合コンテンツビューを作成する前に、既存のコンテンツビューのバージョン ID が必要になります。

```
# hammer content-view version list \
--full-results true \
--organization "ACME"
```

本シナリオでは、Database v1.0 のバージョン ID は 5 で、Base v2.0 のバージョン ID は 6 になります。**Stack** という名前の新規の複合コンテンツビューを作成し、**--component-ids** オプションでバージョン ID を渡します。

```
# hammer content-view create \
--composite \
--name "Stack" \
--description "A stack that includes a base operating system and a
database" \
--component-ids 4,14 \
--organization "ACME"
```

複合コンテンツビューを公開します。

```
# hammer content-view publish \
--name "Stack" \
--description "Initial version of Stack" \
--organization "ACME"
```

複合コンテンツビューを全環境にプロモートします。

```
# hammer content-view version promote \
--content-view "Stack" \
--version 1 \
--to-lifecycle-environment "Development" \
--organization "ACME"
# hammer content-view version promote \
--content-view "Stack" \
--version 1 \
--to-lifecycle-environment "Testing" \
--organization "ACME"
# hammer content-view version promote \
--content-view "Stack" \
--version 1 \
--to-lifecycle-environment "Production" \
--organization "ACME"
```

7.4. コンテンツフィルター

コンテンツビューでは、フィルターを使用して特定の RPM コンテンツを含めたり制限したりします。フィルターを使用しないと、選択したリポジリーのもののがすべて含まれてしまいます。

コンテンツフィルターは以下のいずれかのタイプになります。

- **組み込み**: ビューに組み込むコンテンツを定義します。このフィルターにはコンテンツが含まれていないので、選択したリポジリーから追加するコンテンツを選択します。複数のコンテンツアイテムを組み合わせる場合には、このフィルターを使用します。
- **除外**: ビューから除外するコンテンツを定義します。このフィルターには、選択したリポジリーのコンテンツがすべて含まれているため、除外するコンテンツを選択します。リポジリーのほとんどのコンテンツを使用したいものの、ブラックリスト化されたパッケージなど、特定のパッケージは除外したい場合にこのフィルターを使用します。このフィルターでは、リポジリーにある、選択したコンテンツ以外のすべてのコンテンツを使用します。



注記

組み込みと除外のフィルターを組み合わせたコンテンツビューを公開すると、最初に組み込みフィルターが適用され、次に除外フィルターが適用されます。この場合、組み込むコンテンツを選択し、このサブセットから除外するコンテンツを選択することになります。

また、フィルターの対象となるコンテンツには以下の4つのタイプがあります。

- **パッケージ**: 名前とバージョンに基づいてパッケージにフィルターを適用します。

- **パッケージグループ:** フィルターに追加するパッケージグループを選択します。パッケージグループ一覧は、コンテンツビューに追加されたリポジトリーに基づきます。
- **エラータ - ID 別:** フィルターに追加する特定のエラータを選択します。エラーター一覧は、コンテンツビューに追加されたリポジトリーに基づきます。
- **エラータ - 日付およびタイプ別:** フィルターに追加する発行済みまたは更新済みのエラータの日付範囲およびタイプ (バグ修正、機能強化、またはセキュリティー) を選択します。



重要

フィルターは、フィルター内に記載されているパッケージの依存関係を解決するものではありません。フィルターにパッケージの依存関係を追加してください。必要な依存関係の判定には、テストが必要になる場合があります。

コンテンツフィルターの使用例を見ていきましょう。

例 1

ベースの Red Hat Enterprise Linux パッケージでリポジトリーを作成します。このフィルターでは、Red Hat Enterprise Linux リポジトリーがコンテンツビューに追加されている必要があります。

フィルター:

- **包含タイプ:** 組み込み
- **コンテンツタイプ:** パッケージグループ
- **フィルター:** **Base** パッケージグループのみを選択します。

例 2

セキュリティーアップデートを除く、特定日以降の全エラータを除外するリポジトリーを作成します。重要なセキュリティーアップデートは即座に適用すべきですが、その他のシステムアップデートを定期的に行う場合などにこれは便利です。このフィルターでは、Red Hat Enterprise Linux リポジトリーがコンテンツビューに追加されている必要があります。

フィルター:

- **包含タイプ:** 除外
- **コンテンツタイプ:** エラータ - 日付およびタイプ別
- **フィルター:** **バグ修正** と **機能強化** のエラータタイプのみを選択し、**セキュリティー** の選択は解除します。**日付タイプ** を **更新日** に設定します。エラータを制限する日付を **開始日** に設定します。**終了日** は空白にして、セキュリティー以外の新たなエラータにフィルターが適用されないようにします。

例 3

例 1 と例 2 の組み合わせで、ベース OS パッケージのみが必要ですが、最近のバグ修正と機能強化エラータを除外します。この場合、同一のコンテンツビューに 2 つのフィルターが適用されている必要があります。コンテンツビューは組み込みフィルターを最初に適用してから、除外フィルターを適用します。

フィルター 1:

- **包含タイプ:** 組み込み

- **コンテンツタイプ:** パッケージグループ
- **フィルター:** **Base** パッケージグループのみを選択します。

フィルター 2:

- **包含タイプ:** 除外
- **コンテンツタイプ:** エラータ - 日付およびタイプ別
- **フィルター:** **バグ修正** と **機能強化** のエラータタイプのみを選択し、**セキュリティ** の選択は解除します。**日付タイプ** を **更新日** に設定します。エラータを制限する日付を **開始日** に設定します。**終了日** は空白にして、**セキュリティ**以外の新たなエラータにフィルターが適用されないようにします。

コンテンツフィルターの機能例については「[How do content filters work in Satellite 6](#)」を参照してください。

7.4.1. コンテンツフィルターの作成

本ガイドのシナリオでは、ACME のベースオペレーティングシステムで特定日以降のエラータアイテムを制限するコンテンツフィルターを作成します。

Web UI をご利用の場合

コンテンツ > コンテンツビュー に移動し、ベース コンテンツビューを選択します。**Yum コンテンツ > フィルター** に移動し、**新規フィルター** をクリックします。以下の詳細を入力します。

- **名前:** **Errata Filter**
- **コンテンツタイプ:** エラータ - 日付およびタイプ別
- **包含タイプ:** 除外
- **説明:** **Exclude errata items from the last year, with the exception of security updates** (セキュリティ更新以外で、昨年からのエラータアイテムを除外)

保存 をクリックします。

エラータの**日付範囲** 画面が表示されます。ここでは、エラータの**タイプ**と**日付**の範囲が選択できます。**機能強化** と **バグ修正** のみを選択します。

- セキュリティ
- 機能強化
- バグ修正

日付タイプ では **発行日** (エラータの発行日) または **更新日** (エラータの最終更新日) を選択します。エラータが作成後に更新されていない場合は、**発行日** と **更新日** は同じになります。**発行日** を選択すると、エラータアイテムには発行日でのフィルタリングだけが行われるため、そのエラータになされた更新は除外されないことに注意してください。

開始日 で、1 年前の今日の日付を選択します。

終了日 は空白にしておきます。

保存 をクリックします。



注記

このフィルターを使用するリポジトリを指定することもできます。**影響するリポジトリ** タブを選択してリポジトリを指定します。この例では、フィルターで使用するリポジトリは1つのみです。

これでフィルターが完成しました。**新規バージョンの公開** をクリックして、完成したリポジトリを公開します。**バージョンの詳細** では **説明** に **Adding errata filter** と入力します。**保存** をクリックします。

コンテンツビューが公開されると、**コンテンツ** コラムのパッケージとエラータ (セキュリティーエラータを除く) の数が公開前のリポジトリと比べて少なくなります。これは、フィルターが前の年の、セキュリティー以外のエラータを正常に除外したことを意味します。

このコンテンツビューを **開発**、**テスト**、**実稼働** の各環境に **プロモート** します。

CLI をご利用の場合

フィルターをコンテンツビューに追加します。 `--inclusion false` オプションを使用して、フィルターを除外フィルターに設定します。

```
# hammer content-view filter create \
--name "Errata Filter" \
--type erratum --content-view "Base" \
--description "Exclude errata items from the last year, with the exception
of security updates" \
--inclusion false \
--organization "ACME"
```

フィルターにルールを追加します。

```
# hammer content-view filter rule create \
--content-view "Base" \
--content-view-filter "Errata Filter" \
--start-date "2015-01-01" \
--types enhancement,bugfix \
--date-type updated \
--organization "ACME"
```

コンテンツビューを公開します。

```
# hammer content-view publish \
--name "Base" \
--description "Adding errata filter" \
--organization "ACME"
```

ビューを各環境にプロモートします。

```
# hammer content-view version promote \
--content-view "Base" \
--version 1 \
--to-lifecycle-environment "Development" \
--organization "ACME"
# hammer content-view version promote \
--content-view "Base" \
```



```
--version 1 \
--to-lifecycle-environment "Testing" \
--organization "ACME"
# hammer content-view version promote \
--content-view "Base" \
--version 1 \
--to-lifecycle-environment "Production" \
--organization "ACME"
```

7.5. コンテンツビューのプロモート

Satellite Server はここまでに 2 つのコンテンツビューを公開し、ライブラリー環境でリポジトリが利用可能になっています。この内の 1 つのコンテンツビューをプロモートして、リポジトリが他の環境で利用可能になるようにしましょう。

注記

環境にコンテンツビューをプロモートするために、管理者以外のユーザーには以下の 2 つのパーミッションが必要になります。

1. `promote_or_remove_content_views`
2. `promote_or_remove_content_views_to_environment`.

`promote_or_remove_content_views` パーミッションは、ユーザーがプロモートできるコンテンツビューを制限します。

`promote_or_remove_content_views_to_environment` パーミッションは、コンテンツビューのプロモート先となる環境を制限します。

これらのパーミッションを使用すると、どのユーザーがどのコンテンツビューをどの環境にプロモートできるか、またはできないかということを指定できます。たとえば、テスト環境へのプロモーションはできるが、実稼働環境にはできない、という制限を設定できます。こうすることで、複数レベルの認証が提供されます。

あるユーザーがコンテンツビューをプロモートできるようになるには、この両方のパーミッションを割り当てる必要があります。

Web UI をご利用の場合

Database コンテンツビューの **バージョン** 画面を開いていることを確認します。バージョンテーブルのバージョン **1.0** で、アクションコラムにある **プロモート** をクリックします。プロモーション対象を選択することができる画面が開きます。**Development** 環境を選択し、**バージョンのプロモート** をクリックします。数分でプロモーションが完了します。

プロモート ボタンを再度クリックします。今度は **Testing** 環境を選択して **バージョンのプロモート** をクリックします。

最後に **プロモート** ボタンを再度押します。**Production** 環境を選択し、**バージョンのプロモート** をクリックします。

これでこのコンテンツビューのリポジトリが全環境に表示されます。

CLI をご利用の場合

コンテンツビューのプロモートには、毎回 `hammer content-view version promote` を使用します。

```
# hammer content-view version promote \  
--content-view "Database" \  
--version 1 \  
--to-lifecycle-environment "Development" \  
--organization "ACME"  
# hammer content-view version promote \  
--content-view "Database" \  
--version 1 \  
--to-lifecycle-environment "Testing" \  
--organization "ACME"  
# hammer content-view version promote \  
--content-view "Database" \  
--version 1 \  
--to-lifecycle-environment "Production" \  
--organization "ACME"
```

これで Database のコンテンツが全環境で利用可能になります。

7.6. 環境とコンテンツビューへのシステム登録

コンテンツビューが利用可能になったので、システムを環境とビューに登録することができます。

7.6.1. サブスクリプションマネージャーへの RHEL システムの登録

まず、テスト用の Red Hat Enterprise Linux 7 クライアントシステムに `root` ユーザーとしてログインし、Satellite Server 用のコンシューマー RPM をダウンロードします。これはホストの `pub` ディレクトリーに配置されています。たとえば、ホスト名が `satellite6.example.com` の Satellite Server の場合は、登録するクライアントで以下のコマンドを実行します。

```
[root@client ~]# rpm -Uvh http://satellite.example.com/pub/katello-ca-consumer-latest.noarch.rpm
```

以下のコマンドを実行して、Satellite Server 上の環境とコンテンツビューを一覧表示します。

```
[root@client ~]# subscription-manager environments --org "acme"
```

クライアントシステムを Satellite Server 上の環境とコンテンツビューに登録します。

```
[root@client ~]# subscription-manager register --org "acme" --environment "Development/Stack"
```



注記

クライアントシステムでは、ACME 組織に所属する Satellite Server ユーザーのユーザー名とパスワードが求められます。別の方法では、アクティベーションキーを使用してシステムを登録することもできます。これについては「[8章 アクティベーションキーの管理](#)」で説明します。

これで、クライアントシステムが、開発環境にある Stack コンテンツビューから公開されたりポジトリを使用するようになりました。

7.6.2. サブスクリプションマネージャーへの **Atomic Host** の登録

以下の手順では、サブスクリプションマネージャーで Atomic Host を登録する方法を説明します。

Satellite server から **katello-rhsm-consumer** を取得します。

```
[root@atomic_client ~]# wget http://satellite.example.com/pub/katello-rhsm-consumer
```

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

```
[root@atomic_client ~]# chmod +x katello-rhsm-consumer
```

katello-rhsm-consumer を実行します。

```
[root@atomic_client ~]# ./katello-rhsm-consumer
```

Red Hat サブスクリプションマネージャーに登録します。

```
[root@atomic_client ~]# subscription-manager register
```



注記

Atomic はアプライアンスとして機能するため、これに **katello-agent** をインストールすることは推奨されません。

7.7. 本章のまとめ

本章では、コンテンツビューを使用して DML の既存コンテンツから独自のポジトリをカスタマイズする方法を説明しました。以下の方法がありました。

- RPM がある基本的なコンテンツビューの作成
- RPM と Puppet モジュールがあるコンテンツビューの作成
- コンテンツフィルターを使用した RPM コンテンツの組み込みと除外
- 複数のコンテンツビューを統合した複合コンテンツビュー
- システムの Satellite Server への登録と、特定のコンテンツビューからのコンテンツの使用

次章では、アクティベーションキーと、それを使用してシステムを登録し、アプリケーションライフサイクル環境からコンテンツにアクセスする方法を説明します。

第8章 アクティベーションキーの管理

前章までで、コンテンツビューを公開し、その結果、Satellite Server がリポジトリを公開しました。システムは Satellite Server に登録し、それらのリポジトリからのコンテンツを使用できます。ここでシステムの登録は、Red Hat カスタマーポータルへの登録と同様の方法で行います。たとえば、Red Hat サブスクリプションマネージャー (**subscription-manager**) で `--baseurl` を使うことで、Red Hat Content Delivery Network ではなく Satellite Server を指定できます。

システム登録には 2 つの方法があります。1 つ目は Satellite Server にユーザー名とパスワードで認証する方法で、これは前章で説明しました。2 つ目はアクティベーションキーを使用する方法で、こちらを使用することが推奨されます。このキーは認証トークンとして機能します。アクティベーションキーを使用すると、システム登録とサブスクリプションのアタッチが容易にできます。ユーザーは複数のキーを作成して異なる環境やコンテンツビューに関連付けることができます。たとえば、Red Hat Enterprise Linux ワークステーション用のサブスクリプションで基本的なアクティベーションキーを作成し、これを特定の環境からのコンテンツビューに関連付けることができます。

アクティベーションキーはコンテンツホストの選択されたプロパティを定義します。コンテンツホストの登録時にアクティベーションキーを使用すると、プロセスをスピードアップし、単純化し、プロセスの一貫性を強化することができます。アクティベーションキーで以下を指定できます。

- 関連付けられるサブスクリプションおよびサブスクリプションのアタッチ動作。
- 利用可能な製品およびリポジトリ。
- ライフサイクル環境およびコンテンツビュー。
- ホストコレクションのメンバーシップ。

十分なサブスクリプションがある限り、同じアクティベーションキーを複数のコンテンツホストに適用することができます。ただし、アクティベーションキーはコンテンツホストの初期設定のみを行います。コンテンツホストを組織に登録した後は、その組織が処理する他のコンテンツをコンテンツホストに手動でアタッチすることができます。

コンテンツホストは、ホストの設定を定義するために組み合わせる複数のアクティベーションキーに関連付けることができます。設定が競合する場合は、最後に指定したアクティベーションキーが優先されます。

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、変更は、改訂されたアクティベーションキーでその後に登録される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とその使用例についての本ガイド以外の情報は、Red Hat カスタマーポータルの「[Activation Key Enhancements with Red Hat Satellite 6.1](#)」を参照してください。

8.1. アクティベーションキーの作成

アクティベーションキーを使用して、登録時にコンテンツホストをサブスクライブする方法を定義できます。アクティベーションキーで定義されるサブスクリプションの動作は以下の 2 つの要因によって変わります。

- サブスクリプションがアクティベーションキーに関連付けられているか？
- 自動アタッチオプションは有効になっているか？

上記の要因に応じて、アクティベーションキーを使用してサブスクライブするシナリオを 3 つ想定できます。

- **サブスクリプションが指定されていないアクティベーションキー:** サブスクリプションが指定されておらず、自動アタッチが有効な場合に、アクティベーションキーを使用するホストは、Satellite Server が提供する一覧から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` コマンドを実行する場合と同様です。
- **自動アタッチ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:** サブスクリプションが指定されていて自動アタッチが有効な場合に、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。
- **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:** サブスクリプションが指定されており、自動アタッチが無効な場合に、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に指定されたすべてのサブスクリプションに関連付けられます。



注記

カスタム製品 (通常は Red Hat が提供しないコンテンツを含む製品) がアクティベーションキーに割り当てられている場合、この製品は、自動アタッチの設定の有無にかかわらず、登録されたコンテンツホストに対して常に有効にされます。

Web UI をご利用の場合

1. **コンテンツ** → **アクティベーションキー** に移動し、**アクティベーションキーの作成** をクリックします。アクティベーションキーに以下の情報を提供します。
 - **名前:** アクティベーションキーの名前。この名前は、システム登録プロセス時で使用します。名前 フィールドに **development-stack** と入力します。
 - **制限:** アクティベーションキーで登録可能なシステムの最大数を定義します。**Unlimited Hosts** チェックボックスを選択します。
 - **説明:** アクティベーションキーの簡単な説明。説明 フィールドに、**Exampleware Stack in the Development Environment** と入力します。
 - **環境:** 使用する環境。環境 領域で **Development** をクリックします。
 - **コンテンツビュー:** 環境内で使用するコンテンツビュー (およびリポジトリ)。コンテンツビュー リストで **Stack** を選択します。

保存 をクリックすると、アクティベーションキーの詳細画面が表示されます。

2. 登録において、アタッチする製品と、有効にするリポジトリを定義するには、**サブスクリプション** タブに移動します。**追加** タブをクリックし、**Red Hat Enterprise Linux** サブスクリプションと **PostgreSQL** 製品の両方をクリックし、**選択項目の追加** をクリックします。



注記

自動アタッチ オプションは有効になっています。これにより、登録時にこれらの製品のアタッチが自動的に行われ、必要なリポジトリが有効になります。アクティベーションキーの **自動アタッチ** が無効になっている場合、システムはサブスクリプションやコンテンツを割り当てず、Satellite Server に登録されるのみとなります。

3. **製品コンテンツ** ページに移動します。デフォルトでは、Satellite Server が有効にしているのは以下のものだけです。

- システム要件に最も適するリポジトリ。このケースでは、Red Hat Enterprise Linux 7 Server RPMs のみです。
- カスタムコンテンツ。

シナリオでは、以下のデフォルトが設定されているはずですが。

PostgreSQL:

- PostgreSQL 9.5 - 有効にされていますか: **Yes (デフォルト)**
- PostgreSQL Puppet Modules - 有効にされていますか: **Yes (デフォルト)**

Red Hat Enterprise Linux Server:

- Red Hat Enterprise Linux 7 Server (Kickstart) - 有効にされていますか: **No (デフォルト)**
- Red Hat Satellite Tools 6.3 (for RHEL 7 Server) (RPMs) - 有効にされていますか: **No (デフォルト)**
- Red Hat Enterprise Linux 7 Server (RPMs) - 有効にされていますか: **Yes (デフォルト)**

Red Hat Satellite Tools リポジトリには設定ツール (**katello-agent** および **puppet** など) が含まれるため、これを有効にします。以下のようにします。

- Red Hat Satellite Tools 6.3 (for RHEL 7 Server) (RPMs) - 有効にされていますか: **Yes** に上書き

4. **保存** をクリックします。

CLI をご利用の場合

1. アクティベーションキーを作成します。

```
# hammer activation-key create \
--name "development-stack" \
--unlimited-content-hosts true \
--description "Exampleware Stack in the Development Environment" \
--lifecycle-environment "Development" \
--content-view "Stack" \
--organization "ACME"
```

2. サブスクリプション ID 一覧を取得します。

```
# hammer subscription list --organization "ACME"
```

3. Red Hat Enterprise Linux サブスクリプション UUID をアクティベーションキーにアタッチします。

```
# hammer activation-key add-subscription \
--name "development-stack" \
--subscription-id ff808181533518d50152354246e901aa \
--organization "ACME"
```

4. PostgreSQL 製品をアクティベーションキーにアタッチします。

```
# hammer activation-key add-subscription \
--name "development-stack" \
--subscription-id ff8081815239acdc015238fefaa10002 \
--organization "ACME"
```

5. アクティベーションキーに関連付けられている製品コンテンツを一覧表示します。

```
# hammer activation-key product-content \
--name "development-stack" \
--organization "ACME"
```

6. Red Hat Satellite Tools 6.3 リポジトリのデフォルトの自動有効化ステータスを上書きします。デフォルトでは無効になっています。有効にするには、以下のコマンドを実行します。

```
# hammer activation-key content-override \
--name "development-stack" \
--content-label rhel-7-server-satellite-tools-6.3-rpms \
--value 1 \
--organization "ACME"
```

8.2. アクティベーションキーの使用

アクティベーションキーは登録に使用されます。以下が含まれます。

- Red Hat Satellite 6 でのプロビジョニング中の新規システム登録。Red Hat Satellite 6 のキックスタートプロビジョニングテンプレートには、新規ホストの作成時に定義されるアクティベーションキーを使用してシステムを登録するコマンドが含まれています。
- 既存の Red Hat Enterprise Linux システムの登録。Red Hat サブスクリプションマネージャーが登録に Satellite Server を使用するように設定し、**subscription-manager register** コマンドの実行時にアクティベーションキーを指定します。

作成したアクティベーションキーをテストします。既存の Red Hat Enterprise Linux 7 システムを Satellite Server に登録します。

まず、Satellite Server 用のコンシューマー RPM をダウンロードします。これは、通常、ホストの web サーバーの **pub** ディレクトリに配置されています。たとえば、ホスト名が **satellite6.example.com** の Satellite Server の場合は、登録するクライアントで以下のコマンドを実行します。

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

この RPM は Satellite Server 上のリポジトリにアクセスするために必要な証明書をインストールし、Red Hat サブスクリプションマネージャーがサーバーの URL を使用するように設定します。

次に、クライアント上で Red Hat サブスクリプションマネージャーを実行します。

```
# subscription-manager register --activationkey="development-stack" \
--org="ACME"
The system has been registered with id: 744fb31c-c983-00f5-ca14-bddd0f711353
```

Satellite Server で登録を確認します。

Web UI をご利用の場合

ホスト > コンテンツホスト に移動すると、リストにシステムが表示されます。

CLI をご利用の場合

以下のコマンドを実行します。

```
# hammer content-host list --organization "ACME"
```

重要

Satellite Server にクライアントシステムを登録したら、システムに **katello-agent** パッケージをインストールして、Satellite Server にレポートできるようにします。

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

このパッケージは、Red Hat Satellite Tools 6.3 リポジトリが提供しています。

コンテンツホストの登録時に複数のアクティベーションキーを使用することができます。これにより、特定のサブスクリプションセット用にアクティベーションキーを作成してから、コンテンツホストの要件に従ってそれらを組み合わせることができます。たとえば、以下のコマンドは VDC と OpenShift の両方のサブスクリプションでコンテンツホストを ACME 組織に登録します。

```
# subscription-manager register --org "ACME" \
  --activationkey "ak-VDC,ak-OpenShift"
```

アクティベーションキーの設定で競合が生じた場合は、右端のキーが優先されます。

- 競合する設定: サービスレベル、リリースバージョン、環境、コンテンツビュー、および 製品コンテンツ。
- 競合しない設定と、ホストがその統合を取得: サブスクリプション および ホストコレクション。
- キーそのものの動作に影響を与えるが、ホストの設定には影響を与えない設定: コンテンツホストの制限 および 自動アタッチ。

8.3. アクティベーションキーを使用して関連するサブスクリプションの更新

Web UI または Hammer コマンドで、アクティベーションキーに関連付けるサブスクリプションを変更できます。Hammer CLI を使用する場合は CSV ファイルをダウンロードして、アクティベーションキー設定を変更し、変更した CSV ファイルをアップロードする必要があります。

アクティベーションキーへの変更は、変更後にプロビジョニングしたマシンにのみ適用されます。既存のコンテンツホストでサブスクリプションを更新する方法は「[コンテンツホストのサブスクリプションの一括アップデート](#)」を参照してください。

Web UI をご利用の場合

1. コンテンツ > アクティベーションキー に移動し、アクティベーションキーの名前をクリックします。

2. **サブスクリプション** タブを選択します。
3. サブスクリプションを削除するには **一覧/削除** を選択し、削除するサブスクリプションの左側にあるチェックボックスを選択し、**選択項目の削除** をクリックします。
4. サブスクリプションを追加するには、**追加** を選択し、追加するサブスクリプションの左側のチェックボックスを選択し、**選択項目の追加** をクリックします。
5. **リポジトリセット** を選択し、リポジトリのステータス設定を確認します。
6. リポジトリを有効または無効にするには、リポジトリに対してチェックボックスを選択し、**アクションの選択** リストを使用してステータスを変更します。
7. **詳細** タブを選択し、このアクティベーションキーにコンテンツビューを選択し、**保存** をクリックします。

CLI をご利用の場合

1. Satellite Server からサブスクリプションを CSV ファイルにエクスポートします (この例では **a_keys.csv**)。

```
# hammer --server https://satellite.example.com csv activation-keys \
--export --file a_keys.csv --organization "Default Organization"
```

2. 以下のようにファイルの列を表示します。

```
# column -s, -t < a_keys.csv | less -S
Name Organization Description Limit Environment Content View
Host Collections Auto-Attach Service Level Release Version
Subscriptions
```

最後の列にはサブスクリプション情報があります。これは CSV ファイルのフィールドの 1 つで、引用されるテキストにはコンマを含むことができます。以下は **サブスクリプション** フィールドの例となります。

- ""Automatic|RH1234|Red Hat Enterprise Linux Server, Standard (Physical or Virtual Nodes)|11223344|55667788""
 - '1|MCT0369|Red Hat Satellite Capsule Server|11223344|55667788'
- サブスクリプション** フィールドでは、以下の形式を使用します。
- 割り当てられるサブスクリプションの数 (**Automatic** に設定されます)。
 - サブスクリプションの ID 番号。
 - サブスクリプションの名前。
 - コントラクト番号。
 - アカウント番号。
3. CSV ファイルで必要な値を変更します。CSV プラグイン、または **sed** を使用して、**Subscriptions** フィールドの文字列を変更します。
 - a. ファイルのバックアップを作成します。

```
# cp a_keys.csv a_keys.csv.backup
```

- b. 以下のように、文字列を変更します。

```
# sed -i "s/Automatic|RH1234|Red Hat Enterprise Linux
Server/Automatic|RH4567|Red Hat Enterprise Linux Server/g"
a_keys.csv
```

- c. 変更箇所を確認します。以下は例となります。

```
# diff a_keys.csv a_keys.csv.backup
2c2
< ""Automatic|RH4567|Red Hat Enterprise Linux Server, Standard
(Physical or Virtual Nodes)|11223344|55667788""
---
> ""Automatic|RH1234|Red Hat Enterprise Linux Server, Standard
(Physical or Virtual Nodes)|11223344|55667788""
```

4. 変更したファイルを Satellite Server にアップロードします。

```
# hammer --server https://satellite.example.com csv activation-keys
\
--file a_keys.csv
```

8.4. 自動アタッチの有効化

アクティベーションキーで自動アタッチを有効にし、キーに割り当てたサブスクリプションがある場合は、サブスクリプション管理サービスが、現在インストールされている製品、アーキテクチャー、およびサービスレベルなどの設定に基づいて、最適な関連サブスクリプションを選択してアタッチします。

自動アタッチを有効にし、キーに関連するサブスクリプションがない場合、このキーは、RHEL サブスクリプションを使用し、ハイパーバイザーから RHEL Virtual Data Center (VDC) サブスクリプションを継承する仮想マシンが必要ない場合に、仮想マシンを登録するために一般的に使用されます。

自動アタッチはデフォルトで有効になっています。アクティベーションキーに関連付けられているすべてのサブスクリプションを強制的にアタッチする場合は、このオプションを無効にします。

Web UI をご利用の場合

1. コンテンツ > アクティベーションキー をクリックします。
2. 編集するアクティベーションキーの名前をクリックします。
3. サブスクリプション タブをクリックします。
4. 自動アタッチ の隣にある編集アイコンをクリックします。
5. チェックボックスにチェックを入れて自動アタッチを有効にするか、チェックを外して無効にします。
6. 保存 をクリックします。



注記

自動アタッチアクティベーションキーを使用して仮想コンテンツホストを Satellite Server に登録するには、まず `virt-who` ユーティリティーを使用して、それらのホストを Virtual Datacenter (VDC) サブスクリプションのエンタイトルメントにあるハイパーバイザーにマッピングします。この前提条件が適用されない場合、仮想ホストは一時的な仮想サブスクリプションに 24 時間登録されます。詳細は『[仮想インスタスガイド](#)』の「[仮想ゲストのサブスクリプションへの適用](#)」を参照してください。

CLI をご利用の場合

アクティベーションキー `development-stack` で自動アタッチを有効にするには、以下のコマンドを実行します。

```
# hammer activation-key update --name "development-stack" \  
--organization "ACME" --auto-attach true
```

8.5. サービスレベルの設定

アクティベーションキーを設定して、アクティベーションキーを使用して作成される新規ホストのデフォルトのサービスレベルを定義することができます。デフォルトのサービスレベルを設定すると、ホストにアタッチするのに適したサブスクリプションのみが選択されます。たとえば、アクティベーションキーのデフォルトのサービスレベルが Premium に設定されている場合、Premium サービスレベルのサブスクリプションのみが、登録時にホストに割り当てられます。

Web UI をご利用の場合

1. コンテンツ > アクティベーションキー をクリックします。
2. 編集するアクティベーションキーの名前をクリックします。
3. サービスレベル の隣にある編集アイコンをクリックします。
4. リストから必要なサービスレベルを選択します。このリストには、アクティベーションキーで利用できるサービスレベルだけが含まれます。
5. 保存 をクリックします。

CLI をご利用の場合

アクティベーションキー `development-stack` でデフォルトのサービスレベルを Premium に設定するには、以下を実行します。

```
# hammer activation-key update --name "development-stack" \  
--organization "ACME" --service-level premium
```

8.6. 本章のまとめ

本章では、アクティベーションキーの作成方法と、それを使用してシステムを Satellite Server に登録する方法を説明しました。

次章では、エラータをシステムに適用する方法などのエラータの管理を説明します。

第9章 エラータの管理

Red Hat では、品質管理およびリリースプロセスの一部として、お客様に Red Hat RPM の公式リリースのアップデートを提供しています。Red Hat では、アップデートを説明するアドバイザリーと共に、関連パッケージのグループを **エラータ** にコンパイルします。アドバイザリーには以下の 3 種類があります (重要度の高い順)。

セキュリティーアドバイザリー

パッケージで見つかったセキュリティー問題の修正を説明。セキュリティー問題の重大度のレベルは、低、中、重要、重大に分かれています。

バグ修正アドバイザリー

パッケージのバグ修正を説明。

製品の機能強化アドバイザリー

パッケージに追加された機能強化および新機能を説明。

Red Hat Satellite 6 は、リポジトリを Red Hat の Content Delivery Network (CDN) と同期する際にこれらのエラータ情報をインポートします。Red Hat Satellite 6 ではエラータを検証しフィルタリングするためのツールも提供しており、アップデートの管理が正確にできます。このようにして、関連のあるアップデートを選択し、コンテンツビューから選択したコンテンツホストに伝達することができます。



注記

エラータには、それらに含まれる最も重要なアドバイザリータイプに応じてラベルが付けられます。そのため、**製品の機能強化アドバイザリー** というラベルが付けられたエラータには機能強化の更新のみが含まれ、**バグ修正アドバイザリー** エラータにはバグ修正と機能強化の両方が含まれ、**セキュリティーアドバイザリー** にはこれら 3 つのタイプが含まれる場合があります。

Red Hat Satellite では、エラータと利用可能なコンテンツホストとの関係を表す 2 つのキーワードがあります。

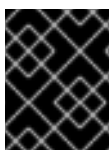
適用可能

エラータは 1 つ以上のコンテンツホストに適用されます。つまり、コンテンツホストにあるパッケージが更新されます。「適用可能」なエラータは、コンテンツホストがアクセスできる段階にはありません。

インストール可能

エラータは 1 つ以上のコンテンツホストに適用され、コンテンツホストで利用可能になっています。インストール可能なエラータはコンテンツホストのライフサイクル環境とコンテンツビューに存在しますが、インストールはされていません。このため、エラータは、コンテンツホストを管理するパーミッションがあるものの、より高いレベルでのエラータ管理の権限がないユーザーによるインストールが可能になっています。

本章では、エラータの管理方法とシステムへの適用方法を説明します。



重要

Satellite Server に登録したホストに **katello-agent** パッケージをインストールします。このパッケージは、エラータ管理に必要なサービスを提供します。

9.1. コンテンツビューによるエラータ管理

Red Hat Satellite 6 では、エラータを管理して適用するために多様な方法を提供しています。「[コンテンツフィルター](#)」で説明したように、コンテンツビューとコンテンツフィルターを使用してエラータを制限できます。フィルターには以下のものがあります。

- **ID:** リポジトリに許可するエラータを選択するフィルターを作成します。
- **日付の範囲:** 日付の範囲を定義して、その範囲内にリリースされたエラータを組み込みます。
- **タイプ:** バグ修正、機能強化、セキュリティーなどのエラータのタイプを選択して組み込みます。

ここでは例として、特定日より後のエラータを除外するコンテンツフィルターを作成します。これにより、アプリケーションライフサイクルの実稼働システムがある時点まで最新に保たれたこととなります。その後このフィルターの開始日を変更し、テスト環境に新たなエラータを導入します。こうすることで、新パッケージにアプリケーションライフサイクルとの互換性があるかどうかをテストすることができます。

コンテンツフィルターの作成方法は「[コンテンツフィルターの作成](#)」を参照してください。

コンテンツビューにエラータを追加したら、これをシステムに適用することができます。Red Hat Satellite 6 に登録した各システムにはエラータ管理画面があり、複数のエラータをシステムに適用することができます。Red Hat Satellite 6 にはエラータを検索、レビューして、複数のシステムに適用するエラータ管理機能もあります。

9.2. 利用可能なエラータの検出

以下の手順では、利用可能なエラータを表示し、フィルターする方法や、選択したアドバイザリーのメタデータを表示する方法を説明します。

1. **コンテンツ > エラータ** に移動して、利用可能なエラータの一覧を表示します。
2. ページ上部のフィルターツールを使用して、表示されるエラータの数を制限します。
 - 調べるリポジトリをリストから選択します。デフォルトでは **すべてのリポジトリ** が選択されます。
 - **適用可能** チェックボックスがデフォルトで選択され、選択されたりポジトリに適用可能なエラータだけが表示されます。**インストール可能** チェックボックスを選択すると、インストール可能なマークが付いたエラータのみが表示されます。
 - エラータの表を検索するには、以下の形式で **検索** フィールドにクエリーを入力します。

parameter operator value

検索に使用できるパラメーターの一覧は表9.1「[エラータ検索で利用できるパラメーター](#)」を参照してください。適用可能な演算子の一覧は『[Red Hat Satelliteの管理](#)』の「[詳細な検索に対してサポートされる演算子](#)」を参照してください。入力時に自動サジェスト機能が利用できません。**and** 演算子と **or** 演算子を使用してクエリーを組み合わせることもできます。たとえば、**kernel** パッケージに関するセキュリティーアドバイザリーのみを表示するには、以下を入力します。

```
type = security and package_name = kernel
```

Enter を押して検索を開始します。

3. 調べるエラータの **Errata ID** をクリックします。

- **詳細** タブには、更新されたパッケージの説明や、更新によって提供される重要な修正および機能強化が記載されています。
- **コンテンツホスト** タブでは、「[複数システムへのエラータの適用](#)」で説明したように、選択したコンテンツホストにエラータを適用できます。
- **リポジトリ** タブには、エラータが含まれているリポジトリの一覧が表示されます。リポジトリはフィルターを使用して環境やコンテンツビューで絞り込むことができ、リポジトリ名で検索できます。

表9.1 エラータ検索で利用できるパラメーター

パラメーター	説明	例
bug	Bugzilla 番号での検索。	bug = 1172165
cve	CVE 番号での検索。	cve = CVE-2015-0235
id	エラータ ID での検索。自動サジェストシステムにより、入力時に利用可能な ID の一覧が表示されます。	id = RHBA-2014:2004
issued	発行日による検索。正確な日付 (「Feb16,2015」など) を指定したり、キーワード (「Yesterday」、「1 hour ago」など) を使用したりできます。時間の範囲は、「<」演算子と「>」演算子を使用して指定できます。	issued < "Jan 12,2015"
package	完全なパッケージビルド名による検索。自動サジェストシステムにより、入力時に利用可能なパッケージの一覧が表示されます。	package = glib2-2.22.5-6.el6.i686
package_name	パッケージ名による検索。自動サジェストシステムにより、入力時に利用可能なパッケージの一覧が表示されます。	package_name = glib2
severity	セキュリティ更新によって修正される問題の重大度による検索。 Critical 、 Important 、または Moderate を指定します。	severity = Critical
title	アドバイザリーのタイトルによる検索。	title ~ openssl
type	アドバイザリーのタイプによる検索。 security 、 bugfix 、または enhancement を指定します。	type = bugfix

パラメーター	説明	例
updated	最新更新日による検索。 issued パラメーターの同じ形式を使用できます。	updated = "6 days ago"

9.3. 個別システムへのエラータの適用

この手順では、エラータをシステムに適用します。これは「[アクティベーションキーの使用](#)」からの続きで、テスト用 Red Hat Enterprise Linux 7 システムをご自分の Satellite サーバーに登録していることを前提としています。この例では、以下のエラータを適用します。

```
Errata ID:    RHSA-2016:0008
Title:       Moderate: openssl security update
Type:        security
Severity:    Moderate
Issued:      2016-01-07
Updated:     2016-01-07
Description: OpenSSL is a toolkit that implements the Secure Sockets Layer (SSL v2/v3) and Transport Layer Security (TLS v1) protocols, as well as a full-strength, general purpose cryptography library.
```

Web UI をご利用の場合

ホスト > コンテンツホスト に移動して、テストするシステムをクリックします。エラータ タブに移動します。「[コンテンツフィルターの作成](#)」で設定したフィルターにより、セキュリティーエラータの一覧が表示されます。

OpenSSL のエラータを適用していきます。検索ボックスに **title ~ openssl** と入力します。これでタイトルに **openssl** が含まれるエラータが検索されます。**RHSA-2016:0008** エラータを選択し、**選択を適用** をクリックします。確認メッセージが表示されるので、**適用** をクリックします。

Satellite Server は、選択したエラータに関連する全パッケージの更新タスクを開始します。タスクが完了すると、更新されたパッケージとその新バージョンの一覧が **詳細** セクションに表示されます。以下は例となります。

```
1:openssl-1.0.1e-51.el7_2.2.x86_64
1:openssl-libs-1.0.1e-51.el7_2.2.x86_64
```

クライアントシステムにログインし、エラータの更新を確認します。

```
[root@client ~]# yum list openssl openssl-libs
```

CLI をご利用の場合

クライアントシステムの OpenSSL エラータを一覧表示します。

```
# hammer host errata list \
--host client.example.com \
--search "title ~ openssl" \
--organization "ACME"
```

クライアントシステムに最新のエラータを適用します。Errata ID を使用して適用するエラータを特定します。

```
# hammer host errata apply \  
--host client.danssat.net \  
--errata-ids RHSA-2016:0008 \  
--organization "ACME"
```

クライアントシステムにログインし、エラータの更新を確認します。

```
[root@client ~]# yum list openssl openssl-libs
```

9.4. 複数システムへのエラータの適用

Red Hat Satellite 6 Web UI には、複数のシステムを確認してそれらにエラータを適用するエラータ管理システムがあります。この例では、「[個別システムへのエラータの適用](#)」と同じエラータ (**RHSA-2016:0008**) を使用します。

Web UI をご利用の場合

コンテンツ > エラータ に移動すると、同期リポジトリからの全エラータが表示されます。また、コンテンツホスト数 では、各エラータの適用およびインストールが可能な登録済みホストの数が表示されます。

このツールを使用して、OpenSSL エラータを 1 つシステムに適用します。検索ボックスに **title ~ openssl** と入力すると、OpenSSL に関連する全エラータが表示されます。これにはバグ修正や機能拡張が含まれますが、「[コンテンツフィルターの作成](#)」でフィルターを作成しているため、Satellite Server はこれらのエラータをテストシステムにインストールできないことに注意してください。

最新の OpenSSL エラータをクリックします。この例では **RHSA-2016:0008** になります。

このエラータの **詳細** 画面が表示され、エラータが解決する問題の説明が提示されます。

コンテンツホスト のサブタブに移動すると、このエラータが適用可能な全システム一覧が表示されます。「**エラータが現在ホストのライフサイクル環境でインストール可能なコンテンツホストのみを表示します。**」を選択し、エラータのインストール可能なシステムのみを表示します。

テストシステムを選択して、**ホストに適用** をクリックします。エラータのインストールに関する確認画面が表示されます。**確定します** をクリックします。

Satellite Server は、選択した各システムでエラータのパッケージ更新を開始します。タスクが完了したら、クライアントシステムにログインしてエラータ更新を確認します。

```
[root@client ~]# yum list openssl openssl-libs
```

CLI をご利用の場合

CLI には Web UI と同じツールがあるわけではありませんが、同様の手順を CLI コマンドで使用することができます。

全 OpenSSL エラータを一覧表示します。

```
# hammer erratum list --search "title ~ openssl" --organization "ACME"
```


インストール可能なエラータに限定するようにします。

```
# hammer erratum list \
--errata-restrict-installable true \
--search "title ~ openssl" --organization "ACME"
```

このエラータの詳細を表示します。

```
# hammer erratum info --id RHSA-2016:0008
```

このエラータを適用可能なシステムを一覧表示します。

```
# hammer host list \
--search "applicable_errata = RHSA-2016:0008" \
--organization "ACME"
```

エラータを1つのシステムに適用します。

```
# hammer host errata apply \
--host client.example.com \
--errata-ids RHSA-2016:0008
```

各クライアントシステムに以下のコマンドを実行します。実行する際は、`--host` を、各システムの名前に変更します。

```
# for HOST in `hammer \
--csv --csv-separator "|" host list \
--search "applicable_errata = RHSA-2016:0008" \
--organization "ACME" | tail -n+2 | awk \
-F "|" '{ print $2 }'`; do echo \
"== Applying to $HOST ==" ; hammer host errata apply \
--host $HOST --errata-ids RHSA-2016:0008 ; done
```

このコマンドは、RHSA-2016:0008 を適用できるホストをすべて特定し、このエラータを各ホストに適用します。

クライアントシステムにログインし、エラータの更新を確認します。

```
[root@client ~]# yum list openssl openssl-libs
```

9.5. エラータ通知のサブスクライブ

『Red Hat Satellite の管理』の「[電子メール通知の設定](#)」で説明されているように、Satellite ユーザーへのメール通知を設定できます。コンテンツビューのプロモート時や、リポジトリの同期後に、適用可能およびインストール可能なエラータの概要を受けとることができます。

9.6. 本章のまとめ

本章では、Red Hat Satellite 6 でエラータを管理してシステムに適用する方法を説明しました。

次章では、Red Hat Satellite 6 におけるコンテナ管理を説明します。

第10章 コンテナイメージの管理

コンテナ化とは、オペレーティングシステムレベルの仮想化メソッドです。標準的な仮想マシンはハイパーバイザーが各システムに割り当てるリソースを消費しますが、コンテナはホストのカーネルを直接使用する個別システムのように動作します。つまり、コンテナ内のプロセスは、ホストや他のコンテナとリソースを共有することになります。これにより、1台のホストで複数のシステムを効率的に実行できるようになります。Linux コンテナはセキュリティーを改善する一方で、アプリケーションのデプロイメントが迅速にでき、テスト、メンテナンス、トラブルシューティングが容易になります。

コンテナの詳細は、『Red Hat Enterprise Linux Atomic Host 7』の「[コンテナの使用ガイド](#)」を参照してください。

Docker は、Linux コンテナとそのアプリケーションのデプロイメントを自動化するオープンソースのプロジェクトです。これは、アプリケーションとそのランタイム依存関係をコンテナに格納してパッケージ化する機能を提供します。

Docker フォーマットのコンテナは以下で構成されています。

コンテナ

アプリケーションのサンドボックスです。各コンテナは必要な設定データが格納されているイメージをベースとしています。イメージからコンテナを起動すると、書き込み可能なレイヤーがイメージの上に追加されます。コンテナをコミットする際は毎回、変更を保存するために新規のイメージレイヤーが追加されます。

イメージ

変更されることのない、コンテナ設定の静的なスナップショットです。コンテナへの変更は新規イメージレイヤーを作成することでのみ保存されます。各イメージは1つ以上の親イメージに依存します。

プラットフォームイメージ

親を持たないイメージです。ランタイム環境、パッケージ、コンテナ化されたアプリケーションの実行に必要なユーティリティーなどを定義します。プラットフォームイメージは読み取り専用となるため、変更はすべて上に重ねられるコピーイメージに反映されます。

レジストリー

ダウンロード可能なイメージが含まれる公開または非公開のアーカイブです。レジストリーによっては、イメージをアップロードして他のユーザーに利用可能にすることもできます。Red Hat Satellite では、ローカルおよび外部レジストリーからイメージをインポートすることができます。Satellite 自体はホストのイメージレジストリーとして機能できますが、ホストは変更をレジストリーに戻すことはできません。

タグ

リポジトリにあるイメージを、通常はイメージに保存されたアプリケーションのバージョンで区別するために使用されるマークです。リポジトリは、コンテナレジストリーの同様のイメージを分類するために使用されます。イメージには固有の英数字の ID しか設定できないので、**repository:tag** などの形式で命名することで、人間が判読できる方法でイメージを識別できるようになります。

Red Hat Satellite 6 では、オンプレミスレジストリーを作成し、複数のソースからイメージをインポートし、コンテンツビューを使用してそれらをコンテナに配信できます。Satellite Server は、コンテナを実行するためのサーバーとして機能する1つ以上の Docker コンピュータリソースの作成をサポートします。これによりイメージをインポートし、このイメージをベースとしたコンテナを開始し、コンテナのアクティビティーをモニターし、その状態を新規のイメージレイヤーにコミットしてさらに伝達するようになります。

コンテンツをインポートしてコンテナイメージのライフサイクルを管理することは、RPM や Puppet モジュールといった他のコンテンツタイプの管理に似ています。製品とバンドルされたリポジトリを設定し、コンテンツビューにこれらのリポジトリを含めます。

10.1. RED HAT CONTAINER CATALOG からのコンテナイメージのインポート

Red Hat Container Catalog は、Satellite Server にインポート可能なコンテナイメージのセットを提供します。コンテンツをインポートするプロセスは、Red Hat コンテンツ用のリポジトリを有効にするものとは異なります。プロセスは以下のようになります。

1. Red Hat Container Catalog 用のカスタム製品を作成します。
2. Red Hat Container Catalog レジストリー (<http://registry.access.redhat.com/>) にリンクするリポジトリを追加します。
3. レジストリーのリポジトリと同期します。

Web UI をご利用の場合

コンテンツ > 製品 に移動し、**新製品** をクリックします。新しい製品のフォームが表示されます。以下の詳細情報を入力します。

- **名前:** 製品の簡単な名前。**Red Hat Container Catalog** と入力します。
- **ラベル:** リポジトリの内部 ID。**rhcc** と入力します。
- **GPG キー:** 製品全体の GPG キー。これは空白にします。
- **同期プラン:** 製品の同期計画。これは **Example Plan** にアタッチすることができます。
- **説明:** 製品の簡単な説明。**Red Hat Container Catalog content** と入力します。

保存 をクリックします。

カスタム製品を作成すると、製品のリポジトリ画面が表示されます。**リポジトリの作成** をクリックし、新しいリポジトリ用のフォームを表示します。以下の詳細情報を入力します。

- **名前:** リポジトリの簡単な名前。**RHEL7** と入力します。
- **ラベル:** リポジトリの内部 ID。**RHEL7** と入力します。
- **タイプ:** リポジトリのタイプ。**docker** を選択すると新たなフィールドが表示されます。
- **URL:** ソースとして使用できるレジストリーの URL。<http://registry.access.redhat.com/> と入力します。
- **アップストリームリポジトリ名:** レジストリー上のリポジトリ名。たとえば **rhel17** と入力します。



注記

<https://access.redhat.com/containers/> で他のリポジトリを検索して表示することもできます。

保存 をクリックすると、新規リポジトリが記載された製品のリポジトリ画面に戻ります。同期開始 をクリックして同期プロセスを開始します。数分すると、Satellite Server が同期を完了します。**Docker マニフェストの管理** をクリックして利用可能なマニフェストを一覧表示します。この一覧から不要なマニフェストを削除することができます。



注記

Web UI でも同期の進捗状況を確認できます。**コンテンツ > 同期の状態** に移動して、製品/リポジトリのツリーを展開します (または **すべて展開** をクリックします)。

CLI をご利用の場合

カスタムの **Red Hat Container Catalog** 製品を作成します。

```
# hammer product create \
--name "Red Hat Container Catalog" \
--sync-plan "Example Plan" \
--description "Red Hat Container Catalog content" \
--organization "ACME"
```

コンテナイメージ用のリポジトリを作成します。

```
# hammer repository create \
--name "RHEL7" \
--content-type "docker" \
--url "http://registry.access.redhat.com/" \
--docker-upstream-name "rhel7" \
--product "Red Hat Container Catalog" \
--organization "ACME"
```

次にリポジトリを同期します。

```
# hammer repository synchronize \
--name "RHEL7" \
--product "Red Hat Container Catalog" \
--organization "ACME"
```

これでコンテナイメージが同期されました。

10.2. 他のイメージレジストリーからのコンテナイメージのインポート

Red Hat Satellite 6 では、Red Hat Container Catalog からコンテナイメージをインポートするほかに、コミュニティーベースや内部のレジストリーなどの他のイメージレジストリーからコンテンツをインポートすることもできます。サードパーティーレジストリーからインポートするプロセスは、Red Hat Container Catalog からイメージをインポートするものと同様のものです。

1. カスタム製品を作成します。
2. 製品にコンテナイメージ用のリポジトリを追加し、それを外部レジストリー上にあるリポジトリにリンクします。
3. レジストリーのリポジトリと同期します。

以下の手順では、DockerHub レジストリー (<https://registry.hub.docker.com>) からの公式の Fedora イメージを使用してこの手順を実行します。

Web UI をご利用の場合

コンテンツ > 製品 に移動し、**新製品** をクリックします。新しい製品のフォームが表示されます。以下の詳細情報を入力します。

- **名前:** 製品の簡単な名前。 **Fedora** と入力します。
- **ラベル:** 製品の内部 ID。 Red Hat Satellite 6 では、 **名前** に入力した内容に基づいてこのフィールドに値が自動的に入力されます。
- **GPG キー:** 製品全体の GPG キー。これは空白にします。
- **同期プラン:** 製品の同期計画。これは **Example Plan** にアタッチすることができます。
- **説明:** 製品の簡単な説明。 **Fedora content** と入力します。

保存 をクリックします。

Fedora 向けカスタム製品を作成すると、製品のレジストリー画面が表示されます。**レジストリーの作成** をクリックし、新しいレジストリー用のフォームを表示します。以下の詳細情報を入力します。

- **名前:** レジストリーの簡単な名前。 **Fedora Docker Images** と入力します。
- **ラベル:** レジストリーの内部 ID。 Red Hat Satellite 6 では、 **名前** に入力した内容に基づいてこのフィールドに値が自動的に入力されます。
- **タイプ:** レジストリーのタイプ。 **docker** を選択すると新たなフィールドが表示されます。
- **URL:** ソースとして使用できるレジストリーの URL。 <https://registry.hub.docker.com> と入力します。
- **アップストリームレジストリー名:** レジストリー上のレジストリー名。たとえば **fedora/ssh** と入力します。



注記

<https://hub.docker.com/explore/> で他のレジストリーを検索して表示することもできます。

保存 をクリックすると、新規レジストリーが記載された製品のレジストリー画面に戻ります。**同期開始** をクリックして同期プロセスを開始します。数分すると、Satellite Server が同期を完了します。**Docker マニフェストの管理** をクリックして利用可能なマニフェストを一覧表示します。この一覧から不要なマニフェストを削除することができます。



注記

Web UI でも同期の進捗状況を確認できます。**コンテンツ > 同期の状態** に移動して、製品/レジストリーのツリーを展開します (または **すべて展開** をクリックします)。

CLI をご利用の場合

カスタムの **fedora** 製品を作成します。

■

```
# hammer product create \  
--name "Fedora" \  
--sync-plan "Example Plan" \  
--description "Fedora content" \  
--organization "ACME"
```

コンテナイメージ用のリポジトリを作成します。

```
# hammer repository create \  
--name "Fedora/SSH" \  
--content-type "docker" \  
--url "https://registry.hub.docker.com" \  
--docker-upstream-name "fedora/ssh" \  
--product "Fedora" \  
--organization "ACME"
```

次にリポジトリを同期します。

```
# hammer repository synchronize \  
--name "Fedora/SSH" \  
--product "Fedora" \  
--organization "ACME"
```

これでコンテナイメージが同期されました。

10.3. コンテンツビューによるコンテナイメージの管理

コンテンツビューを使用して、アプリケーションライフサイクルでコンテナイメージを管理します。このプロセスでは、RPM および Puppet モジュールが使用するものと同じ公開とプロモーションのメソッドを使用します。

Web UI をご利用の場合

コンテンツ > コンテンツビュー に移動し、**新規ビューの作成** をクリックします。ビューの詳細 フォームが表示されるので、以下の情報を入力します。

- **名前:** ビューの簡単な名前。 **Containers** と入力します。
- **ラベル:** ビューの内部 ID。Red Hat Satellite 6 では、**名前** に入力した内容に基づいてこのフィールドに値が自動的に入力されます。
- **説明:** ビューの簡単な説明。 **Container image for Red Hat Enterprise Linux 7** と入力します。
- **複合ビュー?:** 複合コンテンツビューを使用するかどうかを定義します。チェックを外しておきます。

保存 をクリックします。

これで新規コンテンツビューエントリが作成されます。**Docker コンテンツ** のサブタブに移動し、**追加** をクリックします。Red Hat Enterprise Linux 7 Server イメージ用のコンテナリポジトリを選択します。**リポジトリの追加** をクリックします。これで、このリポジトリからコンテナイメージがコンテンツビューに追加されます。

これでコンテンツビューを公開する準備ができました。**バージョン** に移動し、**新規バージョンの公開**

をクリックします。Satellite Server によって新規バージョン (バージョン 1) の詳細が提供されます。説明に、このバージョンに関する説明が入力できるため、新規コンテンツビューの変更点を記録しておく便利です。**Initial content view for our container image** と入力し、**保存** をクリックします。

Satellite Server がビューの新バージョンを作成し、ライブラリー環境に公開します。

プロモート をクリックして、アプリケーションライフサイクルの環境でこのコンテンツビューをプロモートすることもできます。

CLI をご利用の場合

リポジトリ ID の一覧を取得します。

```
# hammer repository list --organization "ACME"
```

この例では、コンテナ **rhel17** のリポジトリ ID が 8 となります。コンテンツビューを作成して、リポジトリを追加します。

```
# hammer content-view create \  
--name "Containers" \  
--description "Container image for Red Hat Enterprise Linux 7" \  
--repository-ids 8 \  
--organization "ACME"
```

ビューを公開します。

```
# hammer content-view publish \  
--name "Containers" \  
--description "Initial content view for our container image" \  
--organization "ACME"
```

Satellite Server がビューの新バージョンを作成し、ライブラリー環境に公開します。

10.4. DOCKER タグによるコンテナイメージの管理

コンテナイメージのなかには Docker タグを使用してバージョン番号や各バージョンについての基本的な情報を特定するものもあります。

Web UI をご利用の場合

コンテンツ > **Docker タグ** に移動して、コンテナベースの製品向けのタグ一覧を表示します。タグをクリックすると、特定イメージに関する情報が表示されます。

CLI をご利用の場合

Docker タグとその ID を一覧表示します。

```
# hammer docker tag info --id 218 --organization "ACME"
```

タグ ID を使用して Docker タグを表示します。

```
# hammer docker tag info --id 218
```

10.5. 本章のまとめ

本章では、独自のレジストリー作成やアプリケーションライフサイクルでのコンテナイメージの管理方法など、Red Hat Satellite 6 におけるコンテナイメージコンテンツの基本的な管理方法を説明しました。

次章では、OSTree コンテンツの管理方法を説明します。

第11章 OSTREE コンテンツの管理

OSTree は、起動可能で、変更しない、バージョン付きファイルシステムツリーを管理するツールです。好みのビルドシステムを使用してコンテンツをビルドサーバー上にあるこのツリー内に配置し、OSTree リポジトリを静的 HTTP にエクスポートできます。Red Hat Enterprise Linux Atomic Server は、RPM ファイルから作成された OSTree コンテンツを使用して、オペレーティングシステムを最新の状態に保ちます。

Red Hat Satellite 6 は、OSTree リポジトリと OSTree ブランチを同期し、リポジトリからブランチを管理するツールを提供します。

11.1. SATELLITE SERVER での OSTREE 管理の設定

Satellite Server 6.3 では、OSTree 管理ツールがデフォルトで有効になっています。必要に応じて、以下のコマンドを実行して、このツールが有効になっていることを確認します。

```
# satellite-installer --katello-enable-ostree=true
```

11.2. 同期する RED HAT OSTREE コンテンツの選択

Red Hat の CDN で OSTree コンテンツを選択して同期します。

Web UI をご利用の場合

コンテンツ > **Red Hat** リポジトリに移動します。これにより、さまざまなコンテンツタイプのタブの設定が表示されます。**OSTree** タブを選択し、使用する OSTree セットを見つけます。この例では、**Red Hat Enterprise Linux Atomic Host** 製品グループから **Red Hat Enterprise Linux Atomic Host Trees** をインポートします。有効なリポジトリがあることを確認してください。

コンテンツ > **製品** に移動し、**Red Hat Enterprise Linux Atomic Host** をクリックします。この時点で、この製品には **Red Hat Enterprise Linux Atomic Host Trees** ブランチセットが含まれています。このリポジトリを選択し、**同期開始** をクリックします。



注記

Web UI でも同期の進捗状況を確認できます。コンテンツ > **同期の状態** に移動して、製品/リポジトリのツリーを展開します (または **すべて展開** をクリックします)。

数分後には、Satellite Server が **rhe17** リポジトリに関連する全イメージのインポートを完了します。

CLI をご利用の場合

Red Hat Enterprise Linux Server 製品の **ostree** リポジトリを検索します。

```
# hammer repository-set list \
  --product "Red Hat Enterprise Linux Atomic Host" \
  --organization "ACME" | grep "ostree"
```

Red Hat Enterprise Linux Atomic Host 向けの **ostree** リポジトリを有効にします。この例では、このリポジトリの ID は 3822 になります。

```
# hammer repository-set enable \
```

```
--product "Red Hat Enterprise Linux Atomic Host" \  
--name "Red Hat Enterprise Linux Atomic Host (Trees)" \  
--organization "ACME"
```

自分の製品内でリポジトリを検索し、これと同期させます。この例では、リポジトリのこのバージョンの ID は 5 です。

```
# hammer repository list \  
--product "Red Hat Enterprise Linux Atomic Host" \  
--organization "ACME"  
# hammer repository synchronize \  
--name "Red Hat Enterprise Linux Atomic Host Trees" \  
--product "Red Hat Enterprise Linux Atomic Host" \  
--organization "ACME"
```

11.3. カスタム OSTREE コンテンツのインポート

Red Hat Satellite 6 では、Red Hat の CDN から OSTree コンテンツをインポートするほかに、他のソースからコンテンツをインポートすることもできます。これには、公開済みの HTTP の場所が必要になります。

Web UI をご利用の場合

コンテンツ > 製品 に移動し、**新製品** をクリックします。新しい製品のフォームが表示されます。以下の詳細情報を入力します。

- **名前:** 製品の簡単な名前。**OSTree Content** と入力します。
- **ラベル:** 製品の内部 ID。Red Hat Satellite 6 では、**名前** に入力した内容に基づいてこのフィールドに値が自動的に入力されます。
- **GPG キー:** 製品全体の GPG キー。これは空白にします。
- **同期プラン:** 製品の同期計画。これは **Example Plan** にアタッチすることができます。
- **説明:** 製品の簡単な説明。**OSTree Content** と入力します。

保存 をクリックします。

Fedora 向けカスタム製品を作成すると、製品のリポジトリ画面が表示されます。**リポジトリの作成** をクリックし、新しいリポジトリ用のフォームを表示します。以下の詳細情報を入力します。

- **名前:** リポジトリの簡単な名前。**Custom OSTree** と入力します。
- **ラベル:** リポジトリの内部 ID。Red Hat Satellite 6 では、**名前** に入力した内容に基づいてこのフィールドに値が自動的に入力されます。
- **タイプ:** リポジトリのタイプ。**ostree** を選択すると URL フィールドが表示されます。
- **URL:** ソースとして使用するレジストリーの URL。たとえば <http://www.example.com/rpm-ostree/> と入力します。

保存 をクリックすると、新規リポジトリが記載された製品のリポジトリ画面に戻ります。**同期開始** をクリックして同期プロセスを開始します。数分すると、Satellite Server が同期を完了します。



注記

Web UI でも同期の進捗状況を確認できます。**コンテンツ > 同期の状態** に移動して、製品/リポジトリのツリーを展開します (または **すべて展開** をクリックします)。

CLI をご利用の場合

カスタムの **OSTree Content** 製品を作成します。

```
# hammer product create \
--name "OSTree Content" \
--sync-plan "Example Plan" \
--description "OSTree Content" \
--organization "ACME"
```

OSTree 用のリポジトリを作成します。

```
# hammer repository create \
--name "Custom OSTree" \
--content-type "ostree" \
--url "http://www.example.com/rpm-ostree/" \
--product "OSTree Content" \
--organization "ACME"
```

次にリポジトリを同期します。

```
# hammer repository synchronize \
--name "Custom OSTree" \
--product "OSTree Content" \
--organization "ACME"
```

これで OSTree ブランチが同期されました。

11.4. コンテンツビューによる OSTREE コンテンツの管理

コンテンツビューを使用して、アプリケーションライフサイクルで OSTree ブランチを管理します。このプロセスでは、RPM および Puppet モジュールが使用するものと同じ公開とプロモーションのメソッドを使用します。

Web UI をご利用の場合

コンテンツ > コンテンツビュー に移動し、**新規ビューの作成** をクリックします。**ビューの詳細** フォームが表示されるので、以下の情報を入力します。

- **名前:** ビューの簡単な名前。 **OSTree** と入力します。
- **ラベル:** ビューの内部 ID。Red Hat Satellite 6 では、**名前** に入力した内容に基づいてこのフィールドに値が自動的に入力されます。
- **説明:** ビューの簡単な説明。 **OSTree branches for Red Hat Enterprise Atomic Host** と入力します。
- **複合ビュー?:** 複合コンテンツビューを使用するかどうかを定義します。チェックを外しておきます。

保存 をクリックします。

これで新規コンテンツビューエントリが作成されます。**OSTree コンテンツ** のサブタブに移動し、**追加** をクリックします。**Red Hat Enterprise Linux Atomic Host Trees** 用の OSTree リポジトリを選択します。**リポジトリの追加** をクリックします。これで、OSTree コンテンツが、このリポジトリからコンテンツビューに追加されます。

これでコンテンツビューを公開する準備ができました。**バージョン** に移動し、**新規バージョンの公開** をクリックします。Satellite Server によって新規バージョン (バージョン 1) の詳細が提供されます。説明に、このバージョンの説明を入力できるため、新規コンテンツビューの変更点を記録しておくに便利です。**Initial content view for our OSTree** と入力し、**保存** をクリックします。

Satellite Server がビューの新バージョンを作成し、ライブラリー環境に公開します。

プロモート をクリックして、アプリケーションライフサイクルの環境でこのコンテンツビューをプロモートすることもできます。

CLI をご利用の場合

リポジトリ ID の一覧を取得します。

```
# hammer repository list --organization "ACME"
```

この例では、OSTree のリポジトリ ID が 5 となります。コンテンツビューを作成して、リポジトリを追加します。

```
# hammer content-view create \  
--name "OSTree" \  
--description "OSTree for Red Hat Enterprise Linux Atomic Host" \  
--repository-ids 5 \  
--organization "ACME"
```

ビューを公開します。

```
# hammer content-view publish \  
--name "OSTree" \  
--description "Initial content view for our OSTree" \  
--organization "ACME"
```

Satellite Server がビューの新バージョンを作成し、ライブラリー環境に公開します。

11.5. 本章のまとめ

本章では、Red Hat およびカスタムソースから OSTree コンテンツをインポートして同期する方法や、そのコンテンツをアプリケーションライフサイクルで管理する方法など、Red Hat Satellite 6 における OSTree コンテンツの基本的な管理方法を説明しました。

次章では、ISO ファイルの管理方法を説明します。

第12章 ISO イメージとファイルの管理

Red Hat Satellite 6 には、Red Hat の CDN やその他のソースの ISO イメージを保存する機能があります。また、Satellite Server では、仮想マシンのイメージなどの他のファイルをアップロードしたり、それらをリポジトリに公開する方法が提供されます。本章では、ISO イメージとその他のファイルをインポートする基本的な手順を説明します。

12.1. RED HAT からの ISO イメージのインポート

Red Hat CDN では、特定製品の ISO イメージを提供しています。このコンテンツをインポートするプロセスは、RPM コンテンツのリポジトリを有効にするプロセスと類似のものです。

Web UI をご利用の場合

コンテンツ > **Red Hat リポジトリ** に移動します。これにより、さまざまなコンテンツタイプのタブのセットが表示されます。**ISO** タブを選択し、使用するイメージを見つけます。この例では、**Red Hat Enterprise Linux Server > Red Hat Enterprise Linux 7 Server (ISOs)** を選択し、**Red Hat Enterprise Linux 7 Server ISOs x86_64 7.2** のイメージを選択します。

コンテンツ > **製品** に移動し、**Red Hat Enterprise Linux Server** をクリックすると、この製品のリポジトリ画面が表示されます。この時点では、この製品には選択した ISO イメージのリポジトリが含まれています。このリポジトリを選択し、**同期開始** をクリックします。



注記

Web UI でも同期の進捗状況を確認できます。コンテンツ > **同期の状態** に移動して、製品/リポジトリのツリーを展開します (または **すべて展開** をクリックします)。

数分後には、Satellite Server によって、選択した全イメージのインポートが完了します。

CLI をご利用の場合

file リポジトリの Red Hat Enterprise Linux Server 製品を検索します。

```
# hammer repository-set list \
  --product "Red Hat Enterprise Linux Server" \
  --organization "ACME" | grep "file"
```

Red Hat Enterprise Linux 7.2 Server ISO の **file** リポジトリを有効にします。

```
# hammer repository-set enable \
  --product "Red Hat Enterprise Linux Server" \
  --name "Red Hat Enterprise Linux 7 Server (ISOs)" \
  --releasever 7.2 \
  --basearch x86_64 \
  --organization "ACME"
```

自分の製品内でリポジトリを検索し、これと同期させます。この例では、リポジトリのこのバージョンの ID は 40 です。

```
# hammer repository list \
  --product "Red Hat Enterprise Linux Server" \
  --organization "ACME"
# hammer repository synchronize \
```

```
--name "Red Hat Enterprise Linux 7 Server ISOs x86_64 7.2" \  
--product "Red Hat Enterprise Linux Server" \  
--organization "ACME"
```

12.2. 個別の ISO イメージとファイルのインポート

本セクションでは、Satellite Server に手動で ISO コンテンツとその他のファイルをインポートする方法を説明します。この例では、**bootdisk.iso** ファイルを Satellite Server にアップロードします。このプロセスは、カスタムの Puppet モジュールをアップロードするプロセスと同様です。

1. カスタム製品を作成します。
2. ファイルのリポジトリを製品に追加します。
3. リポジトリにアップロードするファイルを選択します。

Web UI をご利用の場合

コンテンツ > 製品 に移動し、**新製品** をクリックします。新しい製品のフォームが表示されます。以下の詳細情報を入力します。

- **名前:** 製品の簡単な名前。**Custom ISOs** と入力します。
- **ラベル:** 製品の内部 ID。Red Hat Satellite 6 では、**名前** に入力した内容に基づいてこのフィールドに値が自動的に入力されます。
- **GPG キー:** 製品全体の GPG キー。これは空白にします。
- **同期プラン:** 製品の同期計画。これは **Example Plan** にアタッチすることができます。
- **説明:** 製品の簡単な説明。**Custom ISO collection** と入力します。

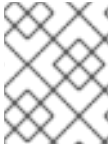
保存 をクリックします。

ISO 用のカスタム製品を作成すると、製品のリポジトリ画面が表示されます。**リポジトリの作成** をクリックし、新しいリポジトリ用のフォームを表示します。以下の詳細情報を入力します。

- **名前:** リポジトリの簡単な名前。**Bootdisk** と入力します。
- **ラベル:** リポジトリの内部 ID。Red Hat Satellite 6 では、**名前** に入力した内容に基づいてこのフィールドに値が自動的に入力されます。
- **タイプ:** リポジトリのタイプ。**file** を選択すると新たなフィールドが表示されます。
- **URL:** ソースとして使用するレジストリーの URL。これは、**PULP_MANIFEST** ファイルが保存されている Satellite が作成した他のリポジトリを同期するためのものです。この例では空白にしておきます。

保存 をクリックすると、**Bootdisk** リポジトリが記載されている製品のリポジトリ画面に戻ります。**Bootdisk** リポジトリをクリックします。

ファイルのアップロード セクションに移動し、**参照** をクリックします。ISO ファイル (この例では **bootdisk.iso**) を選択し、**アップロード** をクリックします。数秒すると、Satellite Server が **Content successfully uploaded** とレポートします。



注記

Puppet モジュールを管理したり、製品から Puppet モジュールを削除するには、**Manage Puppet Modules** ページをクリックします。

CLI をご利用の場合

カスタム製品を作成します。

```
# hammer product create \
--name "Custom ISOs" \
--sync-plan "Example Plan" \
--description "Custom ISO collection" \
--organization "ACME"
```

リポジトリを作成します。

```
# hammer repository create \
--name "Bootdisk" \
--content-type "file" \
--product "Custom ISOs" \
--organization "ACME"
```

ISO ファイルをリポジトリにアップロードします。

```
# hammer repository upload-content \
--path ~/bootdisk.iso \
--name "Bootdisk" \
--product "Custom ISOs" \
--organization "ACME"
```

これで ISO イメージを含むカスタムリポジトリが完成しました。

12.3. RED HAT OVAL リポジトリのインポート

OVAL (Open Vulnerability and Assessment Language) ファイルには、公開セキュリティーコンテンツの情報が含まれています。Red Hat Satellite 6 では、OVAL ファイルを SCAP 監査プロセスの一部として使用します。Red Hat は、複数の OVAL ファイルを格納しているリポジトリ (<https://www.redhat.com/security/data/oval/>) を提供しています。Satellite Server がこのリポジトリを同期することで、SCAP 監査向けのファイルにローカルでアクセスできるようになります。

Web UI をご利用の場合

コンテンツ > 製品 に移動し、**新製品** をクリックします。新しい製品のフォームが表示されます。以下の詳細情報を入力します。

- **名前:** 製品の簡単な名前。**OVAL Files** と入力します。
- **ラベル:** 製品の内部 ID。Red Hat Satellite 6 では、**名前** に入力した内容に基づいてこのフィールドに値が自動的に入力されます。
- **GPG キー:** 製品全体の GPG キー。これは空白にします。
- **同期プラン:** 製品の同期計画。これは **Example Plan** にアタッチすることができます。

- **説明:** 製品の簡単な説明。 **OVAL file collections** と入力します。

保存 をクリックします。

OVAL 向けのカスタム製品を作成したら、製品のリポジトリ画面が表示されます。 **リポジトリの作成** をクリックし、新しいリポジトリ用のフォームを表示します。以下の詳細情報を入力します。

- **名前:** リポジトリの簡単な名前。 **Red Hat OVAL Files** と入力します。
- **ラベル:** リポジトリの内部 ID。 Red Hat Satellite 6 では、 **名前** に入力した内容に基づいてこのフィールドに値が自動的に入力されます。
- **タイプ:** リポジトリのタイプ。 **file** を選択すると新たなフィールドが表示されます。
- **URL:** ソースとして使用するレジストリーの URL。たとえば <https://www.redhat.com/security/data/oval/> と入力します。このレポジトリには、Satellite Server が同期に使用する **PULP_MANIFEST** ファイルが含まれます。

保存 をクリックすると、新規リポジトリが記載された製品のリポジトリ画面に戻ります。このリポジトリを選択して **同期開始** をクリックします。

CLI をご利用の場合

カスタム OVAL 製品を作成します。

```
# hammer product create \
--name "OVAL Files" \
--sync-plan "Example Plan" \
--description "OVAL file collections" \
--organization "ACME"
```

OVAL リポジトリを作成します。

```
# hammer repository create \
--name "Red Hat OVAL Files" \
--content-type "file" \
--product "OVAL Files" \
--publish-via-http true \
--url https://www.redhat.com/security/data/oval/ \
--organization "ACME"
```

OVAL リポジトリを同期します。

```
# hammer repository synchronize \
--name "Red Hat OVAL Files" \
--product "OVAL Files" \
--organization "ACME"
```

ローカルの Red Hat OVAL リポジトリのテスト

OVAL コンテンツの同期が完了したら、リポジトリからの OVAL ファイルを評価するテストを実行できます。

テストの Red Hat Enterprise Linux 7 システムに **openscap-scanner** パッケージをインストールします。これには **oscap** ツールが含まれています。

■


```
# yum install openscap-scanner
```

Satellite Server から Red Hat Enterprise Linux 7 OVAL ファイルのコピーをダウンロードします。

```
# cd /tmp
# wget http://satellite.example.com/pulp/isos/ACME-OVAL_Files-
Red_Hat_OVAL_Files/Red_Hat_Enterprise_Linux_7.xml
```

oscap ツールで Red Hat Enterprise Linux 7 のテストマシンの脆弱性をスキャンします。

```
# oscap oval eval \
--results results.xml \
--report report.html ./Red_Hat_Enterprise_Linux_7.xml
```

これでテストが実行され、各定義が順守しているかどうかの情報を含む OVAL レポートが生成されます。

12.4. 本章のまとめ

本章では、Red Hat Satellite 6 での ISO およびファイルコンテンツの管理を説明しました。

次章では、コンテンツ管理プロセスを説明します。

第13章 コンテンツ管理の最終処理

コンテンツ管理は Red Hat Satellite 6 の機能の一部に過ぎません。Red Hat Satellite 6 ではシステムのプロビジョニングやシステム管理、カプセル制御、モニタリング、およびレポートなどの機能が提供されます。ただしコンテンツ管理は、Red Hat Satellite 6 エコシステムの初期段階として機能します。本章では、本ガイドでここまで説明したコンテンツ管理についておさらいし、これが他の Red Hat Satellite 6 の機能に及ぼす影響を説明します。

13.1. シナリオにおける目的の完了

本書では、ACME という架空の企業におけるエンドツーエンドのシナリオで、以下の目的を達成する方法を示してきました。

Red Hat サブスクリプションの管理

サブスクリプションマニフェストを使用して Red Hat Satellite 6 内の Red Hat コンテンツにアクセスします。サブスクリプションマニフェストを生成し、カスタマーポータルからこれをダウンロードして、Satellite Server 上の組織にインポートします。これで Satellite 環境から RPM、キックスターターコンテンツ、ISO、および Red Hat の Content Delivery Network からのコンテナイメージにアクセスできるようになります。

Definitive Media Library (DML) の作成

コンテンツ管理は、DML の作成が基本となります。DML は、コンテンツの全マスターコピー用の中央ライブラリーとして機能します。外部ソースのコンテンツを Red Hat Satellite 6 に同期して DML を形成します。この外部ソースには、Red Hat のソースやカスタムソースも含まれます。また、同期プランを使用して DML を最新に保ちます。

異なるコンテンツタイプの管理

本書では、RPM ファイルや Puppet モジュール、コンテナイメージなどの異なるタイプのコンテンツを管理する例を説明しました。

アプリケーションライフサイクルの作成

アプリケーションライフサイクルは、コンテンツ管理の核となる概念です。組織の実稼働サイクルをベースに、アプリケーションライフサイクル内に環境を作成します。次に、コンテンツをフィルタリングするコンテンツビューを定義し、作成されるリポジトリを公開して、それらをアプリケーションライフサイクル内の環境にプロモートします。アクティベーションキーを使用してシステムを特定の環境に登録します。

エラータの管理

Red Hat Satellite 6 のツールを使用してエラータを確認し、これを登録済みのシステムに適用します。各エラータには、該当システムにリモートでインストールすることが可能な関連パッケージのセットが含まれています。

コンテナイメージの管理

Satellite Server がコンテナイメージのレジストリーとして機能するようにします。Red Hat などのソースからのコンテナイメージを同期し、コンテンツビューを使用してそれらを管理し、公開します。

13.2. システムのプロビジョニング

本ガイドのシナリオにおける Satellite Server には、この時点でアプリケーションライフサイクルで管理されているコンテンツが含まれています。これで、特定の環境にシステムをプロビジョニングすることができるようになりました。ここからベアメタルシステムをプロビジョニングする方法を説明していきます。



注記

Capsule からシステムをプロビジョニングした場合は、Capsule に割り当てたコンテンツビューに、プロビジョニングした Red Hat Enterprise Linux バージョンに対するキックスタートリポジトリが含まれる必要があります。

13.2.1. キックスタートリポジトリをインストールメディアとして使用

Satellite には、ホストのインストールメディアとして使用できるキックスタートリポジトリのセットが含まれます。

キックスタートリポジトリをインストールメディアとして使用するには

1. 同期したキックスタートリポジトリをインストールメディアとして既存のコンテンツビューに追加するか、コンテンツビューを新たに作成してキックスタートリポジトリを追加します。
2. キックスタートリポジトリを追加した新しいバージョンのコンテンツビューを公開し、必要なライフサイクル環境にプロモートします。詳細は「[7章 コンテンツビューの管理](#)」を参照してください。
3. キックスタートリポジトリをインストールメディアとして使用するホストを作成します。
 - a. **ホスト > すべてのホスト** に移動し、**ホストの作成** をクリックします。
 - b. **ライフサイクル環境** 一覧から、対応するライフサイクル環境を選択します。
 - c. **コンテンツビュー** 一覧から、対応するコンテンツビューを選択します。
 - d. **コンテンツソース** 一覧から、対応するコンテンツソースを選択します。
 - e. **オペレーティングシステム** タブに移動して、対応するアーキテクチャーおよびオペレーティングシステムに移動します。
 - f. 残りの必須フィールドをすべて入力し、**送信** をクリックします。

キックスタートツリーの表示

キックスタートツリーを表示するには、以下のコマンドを入力します。

```
# hammer medium list --organization "your_organization"
```

13.2.2. 環境への登録

キックスタートプロセスの一部として、Satellite Server は、プロビジョニングテンプレートおよびスニペットのセットを使用して、キックスタートファイルを作成します。**subscription_manager_registration** スニペットは登録プロセスを制御し、主に以下の2つの機能があります。

- プロビジョニング時に選択するアクティベーションキーを使用して、プロビジョニングシステムを Satellite Server に登録します。
- Satellite Server がシステムと接続するのに使用する **katello-agent** をインストールします。キックスタートのプロビジョニングテンプレートのほとんどのデフォルトには、システムを登録するこのスニペットが含まれます。

Web UI をご利用の場合

ホスト > プロビジョニングテンプレート に移動し、**subscription_manager_registration** スニペットをクリックしてそれを表示します。

CLI をご利用の場合

以下のコマンドを実行して、**subscription_manager_registration** スニペットを表示します。

```
# hammer template dump --name subscription_manager_registration
```

キックスタートテンプレートを作成した場合は、以下の行をテンプレートに使用して、このスニペットを参照します。

```
<%= snippet "subscription_manager_registration" %>
```

13.2.3. 新規システムのプロビジョニング

新規システムの作成時には、コンテンツ管理設定から以下の点を選択します。

- アプリケーションライフサイクルの環境
- その環境におけるコンテンツビュー
- 選択した環境からの Puppet クラス
- 使用するインストールメディア

新規ホストのこれらの点は、プロビジョニング中にインストールされるパッケージや登録に使用するアクティベーションキー、更新用のリポジトリ、設定中にシステムに適用する Puppet モジュールとそのクラスに影響を与えます。また、ネットワーク情報、パーティションテーブル、カプセル、プロビジョニングテンプレートで変数として使用されるシステム固有のパラメーターなど、ホストの他の点も指定する必要があります。ホストを作成してプロビジョニングプロセスを開始する方法は『[ホストの管理](#)』の「[ホストの作成](#)」セクションを参照してください。

付録A コンテンツストレージ向け NFS 共有の使用

使用する環境ではコンテンツのストレージに十分な容量のハードディスクが必要になります。場合によっては、コンテンツのストレージに NFS 共有を使用することが便利なこともあります。本付録では、Satellite Server のコンテンツ管理コンポーネントに NFS 共有をマウントする方法を説明します。

重要

`/var/lib/pulp` は NFS 共有にマウントしないでください。Satellite Server の一部は、NFS に問題がある一時的な SQLite データベースを使用します。Red Hat では、`/var/lib/pulp` ファイルシステムに高帯域幅で低レイテンシーのストレージを使用することを推奨しています。Red Hat Satellite には、I/O を大量に使用する操作が多数あるため、高レイテンシーで低帯域幅のストレージを使用すると、パフォーマンスが低下することがあります。`/var/lib/pulp/content` ディレクトリーにのみ、NFS 共有を使用してください。

1. NFS 共有を作成します。この例では、`nfs.example.com:/satellite/content` で共有を使用します。この共有で適切なパーミッションが Satellite Server とその `apache` ユーザーに提供されるようにしてください。

2. Satellite ホスト上の Satellite サービスをシャットダウンします。

```
# katello-service stop
```

3. Satellite Server に `nfs-utils` パッケージがインストールされていることを確認します。

```
# yum install nfs-utils
```

4. `/var/lib/pulp/content` の既存のコンテンツを NFS 共有にコピーします。まず、NFS 共有を一時的な場所にマウントします。

```
# mkdir /mnt/temp  
# mount -o rw nfs.example.com:/satellite/content /mnt/temp
```

`/var/lib/pulp/content` の既存コンテンツを一時的な場所にコピーします。

```
# cp -r /var/lib/pulp/content/* /mnt/temp/.
```

5. 共有上の全ファイルで `apache` ユーザーを使用するようにパーミッションを設定します。通常、このユーザーの ID は 48 になります。

6. 一時的なストレージの場所をアンマウントします。

```
# umount /mnt/temp
```

7. `/var/lib/pulp/content` の既存コンテンツを削除します。

```
# rm -rf /var/lib/pulp/content/*
```

8. `/etc/fstab` ファイルに以下の行を追加します。

```
nfs.example.com:/satellite/content /var/lib/pulp/content nfs
rw,hard,intr,context="system_u:object_r:httpd_sys_rw_content_t:s0"
```

これでシステムの再起動後もマウントが維持されます。SELinux コンテキストを含めることを忘れないでください。

9. マウントを有効にします。

```
# mount -a
```

10. NFS 共有が **var/lib/pulp/content** にマウントしていることを確認します。

```
# df
Filesystem                1K-blocks      Used Available
Use% Mounted on
...
nfs.example.com:/satellite/content 309506048 58632800 235128224 20%
/var/lib/pulp/content
...
```

既存のコンテンツが **var/lib/pulp/content** のマウントにあることを確認します。

```
# ls /var/lib/pulp/content
```

11. Satellite ホスト上の Satellite サービスを起動します。

```
# katello-service start
```

これで Satellite Server はコンテンツの保存に NFS 共有を使用します。コンテンツの同期を実行して (「[コンテンツの同期](#)」を参照) NFS 共有が予想どおり機能することを確認してください。

付録B 非接続の SATELLITE SERVER へのコンテンツ ISO のインポート

インターネットに接続されていない閉鎖されたネットワークでホストが機能する必要がある、セキュリティレベルの高い環境でも、Satellite Server は、システムに最新のセキュリティ更新、エラータおよびパッケージを提供できます。これを実行するには、Red Hat Satellite 用のコンテンツ ISO を Red Hat カスタマーポータルからダウンロードして、Satellite Server にインポートします。



重要

お使いの Satellite Server がインターネットに接続している場合は、本セクションは必要ありません。

Red Hat カスタマーポータルから製品の ISO をダウンロードします。

1. (ウィンドウの最上部にある) ダウンロードに移動し、Red Hat Satellite を選択します。
2. コンテンツ ISO タブを開きます。サブスクリプションの全製品が記載されています。
3. 製品名、たとえば Red Hat Enterprise Linux 6 Server (x86_64) のリンクをクリックして、ISO をダウンロードします。
4. Satellite がアクセスできるディレクトリーに Satellite コンテンツ ISO をすべてコピーします。この例では `/root/isos` を使用します。
5. Satellite で、httpd で共有するローカルディレクトリーを作成します。この例では `/var/www/html/pub/sat-import/` を使用します。

```
# mkdir -p /var/www/html/pub/sat-import/
```

6. 最初の ISO のコンテンツをローカルディレクトリーにマウントし、再帰的にコピーします。

```
# mkdir /mnt/iso
# mount -o loop /root/isos/first_iso /mnt/iso
# cp -ruv /mnt/iso/* /var/www/html/pub/sat-import/
# umount /mnt/iso
# rmdir /mnt/iso
```

7. 各 ISO で上記の作業を繰り返して、コンテンツ ISO から全データを `/var/www/html/pub/sat-import/` にコピーします。
8. ディレクトリーに正しい SELinux コンテキストが設定されていることを確認します。

```
# restorecon -rv /var/www/html/pub/sat-import/
```

9. これで Satellite Server にコンテンツ ISO のコンテンツが格納されました。ただし、Satellite Server はこの場所を CDN URL として指定する必要があります。Satellite Web UI で **コンテンツ > Red Hat サブスクリプション** に移動します。
10. **マニフェストの管理** をクリックします。
11. サブスクリプションマニフェストの情報画面で **アクション** タブを選択します。

12. Red Hat プロバイダーの詳細まで移動します。**Red Hat CDN URL** の編集アイコンをクリックし、URL を、新たに作成したディレクトリーのある Satellite ホスト名に変更します。以下は例となります。
<http://server.example.com/pub/sat-import/>

13. **保存** をクリックし、「**Satellite Server へのサブスクリプションマニフェストのインポート**」に従ってマニフェストをアップロードします。

これで、Satellite は、ファイルが <http://server.example.com/pub/sat-import/> にある独自の CDN として機能するようになります。ただし、これは必須ではありません。Satellite Server が HTTP 経由でアクセス可能であれば、同一の非接続ネットワーク内の別のマシンで CDN をホストすることができます。

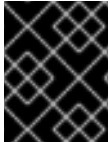
お使いの環境を非接続から接続に変更した場合は、接続されていなかった Satellite が Red Hat カスタマーポータルから直接コンテンツをプルするように設定することができます。

1. Satellite Web UI で、**コンテンツ > Red Hat サブスクリプション** に移動します。
2. **マニフェストの管理** をクリックします。
3. サブスクリプションマニフェストの情報画面で **アクション** タブを選択します。
4. Red Hat プロバイダーの詳細まで移動します。**Red Hat CDN URL** の編集アイコンをクリックし、URL を Red Hat CDN URL に変更します。
<https://cdn.redhat.com>
5. **保存** をクリックします。

これで Satellite Server は、次回の同期で、コンテンツを Red Hat カスタマーポータルから直接プルするようになります。

付録C 接続済み SATELLITE SERVER へのコンテンツ ISO のインポート

Satellite Server が Red Hat カスタマーポータルに直接接続できる場合でも、初回同期はローカルにマウントされたコンテンツ ISO から実行することができます。この同期が完了すると、ネットワーク接続でのコンテンツのダウンロードに切り替えることができます。これを実行するには、Red Hat Satellite 向けのコンテンツ ISO を Red Hat カスタマーポータルからダウンロードし、Satellite Server にこれをインポートします。帯域幅に制限がある場合は、**オンデマンド** または **背景** ダウンロードポリシーを使用する方が、上記の方法よりも効率的な場合があります。



重要

お使いの Satellite Server がインターネットに接続している場合は、本セクションは必要ありません。

この例では、Red Hat Enterprise Linux 6 リポジトリのコンテンツ ISO からの初回同期を説明します。本ガイド作成時には、DVD 21 枚分の ISO ファイルがあります。

Red Hat カスタマーポータルからのコンテンツ ISO のダウンロード

1. ブラウザーで [Red Hat カスタマーポータル](#) を開き、ログインします。
2. **ダウンロード** をクリックします。
3. **Red Hat Satellite** を選択します。
4. **コンテンツ ISO** タブを選択します。サブスクリプションの全製品が表示されます。
5. 必要なセクションを見つけます。この例では、**Red Hat Enterprise Linux 6** になります。
6. **RHEL 6 Server (x86_64) (2017-04-14T01:27:00)** などの製品名をクリックして ISO ファイルを表示します。
7. ブラウザーで、必要な ISO を、ワークステーションの **Downloads** ディレクトリーなど、ブラウザでアクセスできる場所にダウンロードします。

コンテンツ ISO のインポート

1. Satellite Server に接続している端末で、必要な全 Satellite コンテンツ ISO を一時的に保存するディレクトリーを作成します。この例では、`/tmp/isos/rhel6` を使用します。

```
# mkdir -p /tmp/isos/rhel6
```

2. ワークステーションで、ISO ファイルを Satellite Server にコピーします。

```
$ scp ~/Downloads/iso_file  
root@satellite.example.com:/tmp/isos/rhel6
```

3. Satellite Server で、ISO のマウントポイントとなるディレクトリーを作成します。

```
# mkdir /mnt/iso
```

4. 全 ISO のコンテンツを格納する作業ディレクトリーを作成します。

```
# mkdir /mnt/rhel6
```

- 最初の ISO のコンテンツを作業ディレクトリーにマウントし、再帰的にコピーします。

```
# mount -o loop /tmp/isos/iso_file /mnt/iso
# cp -ruv /mnt/iso/* /mnt/rhel6/
# umount /mnt/iso
```

- 各 ISO で上記の作業を繰り返して、コンテンツ ISO から全データを **/mnt/rhel6** にコピーします。
- 必要に応じて、マウントポイントに使用した空のディレクトリーを削除します。

```
# rmdir /mnt/iso
```

- 必要に応じて、一時的な作業ディレクトリーとそのコンテンツを削除して、スペースを取り戻します。

```
# rm -rf /tmp/isos/
```

初回同期の実行

- ディレクトリーの所有者、SELinux コンテキスト、そのコンテンツを **/var/lib/pulp** と同じものにします。

```
# chcon -R --reference /var/lib/pulp /mnt/rhel6/
# chown -R apache:apache /mnt/rhel6/
```

- /etc/pulp/content/sources/conf.d/local.conf** ファイルを作成または編集し、以下のテキストを追加します。

```
[rhel-6-server]
enabled: 1
priority: 0
expires: 3d
name: Red Hat Enterprise Linux 6 Server
type: yum
base_url:
file:///mnt/rhel6/content/dist/rhel/server/6/6Server/x86_64/os/
```

base_url のパスはコンテンツ ISO によって異なる場合があります。**base_url** で指定するディレクトリー内に **repodata** ディレクトリーが必要です。これがないと、同期は失敗します。複数のリポジトリーを同期するには、**/etc/pulp/content/sources/conf.d/local.conf** 設定ファイルで各リポジトリー向けの個別エントリーを作成します。

- Satellite Web UI で **コンテンツ > Red Hat リポジトリー** に移動し、有効にするリポジトリーを選択します。この例では、Red Hat Enterprise Linux 6 Server RPMs x86_64 6Server になります。
- コンテンツ > 同期の状態** に移動して、同期するリポジトリーを選択し、**今すぐ同期** をクリックします。

Satellite Web UI では、使用されているソースが表示されないことに留意してください。ローカルのソースに問題がある場合は、Satellite はネットワーク経由でコンテンツをプルします。プロセスを監視するには、端末に以下のコマンドを入力します (Red Hat Enterprise Linux 7 ベースシステムに限定):

```
# journalctl -f -l SYSLOG_IDENTIFIER=pulp | grep -v worker[\\-,\\. ]heartbeat
```

上記のコマンドを実行すると対話的なログが表示されます。まず Satellite Server が Red Hat カスタマーポータルに接続してリポジトリのメタデータをダウンロードして処理します。次に、ローカルリポジトリが読み込まれます。エラーが発生したら Satellite Web UI で同期をキャンセルして、設定を確認してください。

同期が成功したら、`/etc/pulp/content/sources/conf.d/local.conf` からローカルソースのエントリを削除して、このソースの接続を解除します。

付録D SATELLITE SERVER 間でのコンテンツ同期

Red Hat Satellite 6.3 では、Satellite 間の同期 (ISS) を使ってアップストリームとダウンストリームのサーバー間でコンテンツを同期します。ISS のコンテキストでは、アップストリームとはコンテンツのエクスポート元となるサーバーを指し、ダウンストリームとはインポート先となるサーバーを指します。

ISS は以下の 2 つのシナリオに対処します。

- 接続済みと非接続の Satellite Server があり、前者から後者にコンテンツを伝達する場合。
- プライマリー Satellite Server から他の Satellite Server に一部のコンテンツを伝達する場合。たとえば、IT 部門が検証したコンテンツビュー (CV) から、ダウンストリーム Satellite に **yum** コンテンツを伝達する場合。



注記

Satellite Server から Capsule Server へのコンテンツ同期では ISS は使用できないことに注意してください。Capsule Server はネイティブで同期をサポートします。詳細は『アーキテクチャーガイド』の「[Capsule Server の概要](#)」を参照してください。

Satellite 6.3 は、エクスポートをディレクトリーセット (デフォルト) または ISO ファイルとしてサポートします。エクスポートしたものは、別の Satellite Server にインポートできます。これは以前の Satellite バージョンにあった **katello-disconnected** スクリプトに代わるものです。このスクリプトはリポジトリーをディレクトリー構造にエクスポートし、これを別の Satellite Server にインポートするというものでした。Satellite 6.3 では、エクスポートとインポートの機能はすべてコマンドラインで実行されます。



注記

エクスポートされるのは RPM、キックスタート、および ISO ファイルのみです。パッケージフィルターなどのコンテンツビューの定義およびメタデータはエクスポートされません。Satellite 6.3 では、Puppet、Docker、または OSTree コンテンツのエクスポートはサポートされていません。インポートは通常のリポジトリー同期で発生し、常にライブラリー環境に届けられます。

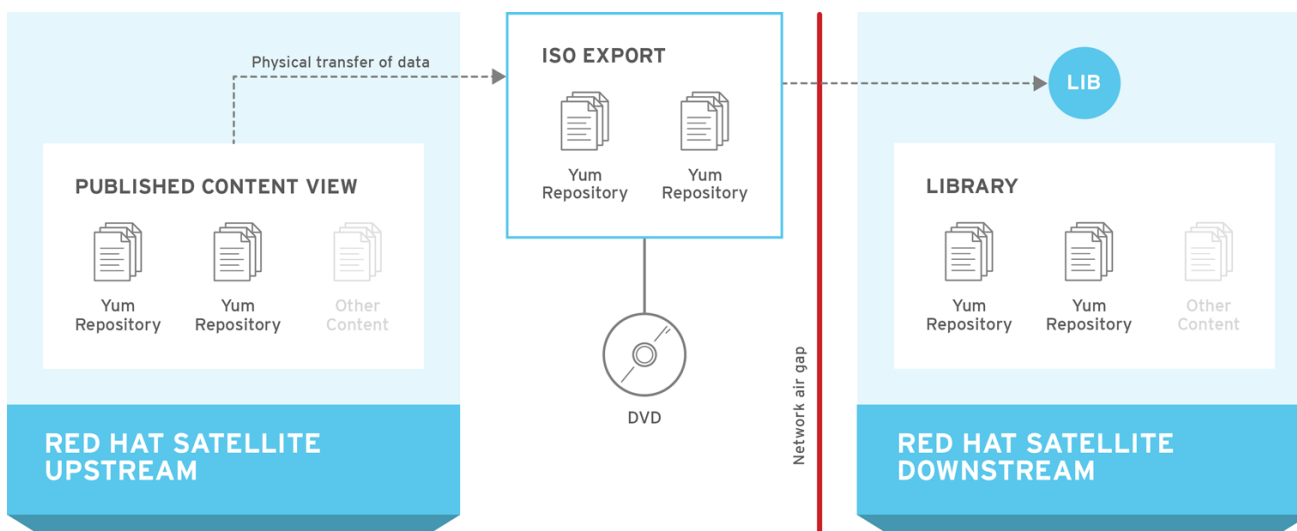
非接続のユースケースは、Satellite とそのクライアントがインターネットには接続されていないエアギャップと呼ばれるネットワークを使用するお客様に幅広く用いられています。このような非接続 Satellite にコンテンツを伝達する唯一の方法は、接続済み Satellite からのエクスポートになります。



重要

双方向の同期は非接続環境では使用されません。非接続サーバーから接続済みサーバーにコンテンツが渡されることはありません。

図D.1 エアギャップネットワークにおける ISS での情報フロー



SATELLITE6_387065_0216

D.1. SATELLITE SERVER、CAPSULE SERVER、および ISS

ISS は Red Hat Satellite のデプロイメントで独特な役割を果たします。デプロイメントに ISS を含めると、メンテナンスやバックアップを必要とする個別の Satellite Server を維持することと同等になります。ISS はクライアントにフェイルオーバーメカニズムを提供するものではなく、バックアップやリカバリーシステムとして設計されたものでもありません。これは Satellite Server 間での情報共有を図るものです。ISS の実装の主なユースケースとしては、インターネットや外部のコンテンツに接続されていない Satellite Server があり、インターネットに接続されている別の Satellite のコンテンツを同期する必要がある場合が挙げられます。セキュリティや他の理由で管理インフラストラクチャーを完全に分離する必要がある場合に、このユースケースを適用できます。

別の管理 Web UI およびプラットフォームを維持せずにローカルクライアントの管理とプロビジョニングを希望する場合は、Capsule Server のセットアップを検討してください。

D.2. 前提条件

- Red Hat Satellite 6 では Satellite 6.2 以降でのみ ISS が利用可能です。これには Red Hat Enterprise Linux 6.7、7.2、もしくはそれ以降が必要になります。
- エクスポートディレクトリーは、少なくとも 1 つの Red Hat Enterprise Linux エクスポートを受け入れられるサイズである必要があります。デフォルトではエクスポートディレクトリーは `/var/lib/pulp/katello-export/` になります。
- `/var/lib/pulp/` ディレクトリーには、エクスポートプロセス中に作成される一時ファイル用にエクスポートされるリポジトリーと同等サイズの空き容量が必要になります。これはデフォルトのエクスポートディレクトリーとは別の容量になります。
- ダウンストリームの Satellite Server には、有効にする予定のコンテンツに必須のマニフェストとエンタイトルメントが必要になります。エンタイトルメントがないダウンストリーム Satellite ではリポジトリーを有効にできません。
- リポジトリーのダウンロードポリシーは **即時** に設定する必要があります。このポリシーは Satellite が最初にメタデータと他のリポジトリー情報をダウンロードするかどうか、またリクエストがあった場合に実際のリポジトリーのみをダウンロードするかどうかを指定します。このポリシーが **即時** に設定されていないと、ISS は正常に機能しません。

必須オプションの設定方法は、「[ISS の設定](#)」で説明しています。

D.3. サポートされる同期オプション

Satellite 6.3 では以下の同期オプションがサポートされています。

- ディレクトリーまたは ISO ファイルへのリポジトリーのエクスポート。
- ディレクトリーまたは ISO ファイルへの環境または CV バージョンの全リポジトリーのエクスポート。カスタム製品はインポートのプロセス中に再作成することはできますが、Red Hat 製品はマニフェストを使用して作成する必要があるため、再作成されません。
- RPM ファイルとエラータのデータに基づく増分エクスポート。

これらの同期オプションには、コンテンツのタイプによってエクスポートとインポートの履歴詳細が含まれます。以下は例となります。

- リポジトリー同期履歴には、エクスポートの発生時刻の他にアップストリームソースの情報が含まれます。
- CV 同期履歴には、インポートの時刻、バージョン、アップストリームソースの他にエクスポート時刻とバージョンが含まれます。

D.4. チャンク ISO ファイルの使用

Satellite 6.3 はチャンク ISO ファイルへのエクスポートをサポートしています。チャンク ISO は split ISO と類似のものですが、大きな違いが 1 つあります。Satellite は ISO ファイルのサイズを追跡し、ISO に追加されているファイルの合計がそのサイズを超えると、Satellite はその ISO への書き込みを停止し、新しい ISO を作成します。この利点は ISO のファイルサイズを指定 (たとえば 4.7 GB) ことができ、かつそれより大きなリポジトリーをエクスポートできることです。これにより、DVD に書き込み可能な複数の 4.7 GB ISO ファイルが作成されます。

チャンクと分割の違いは、分割の **split** ユーティリティーは ISO のファイル形式を認識しないため、同じシリーズの次のファイル用に書き込み可能な新しい ISO ファイルを作成しないという点です。この方法では、全ファイルを 1 カ所にコピーして連結し、連結した 1 つの大きな ISO をループバックでマウントする必要があります。

`--iso-mb-size` パラメーターを使用すると、ISO エクスポートファイルのサイズを指定できます。デフォルト値は、片面の単一層 DVD のサイズである 4380 MB です。

D.5. ISS の設定

本セクションでは、ISS の設定方法を説明します。これを正しく設定しないと同期が失敗する可能性があるため、重要な手順になります。

D.5.1. エクスポート先の設定

Satellite 間の同期 (ISS) は、**pulp_export_destination** 設定にあるようにデフォルトで `/var/lib/pulp/katello-export/` ディレクトリーを使用します。このディレクトリーを変更するには、新規ディレクトリーを作成して Pulp エクスポート先を設定する必要があります。これを指定できるのは Satellite の管理者のみで、Satellite が任意のファイルシステムに書き込みないように、SELinux とその他のパーミッションも設定されています。

よく使用するディレクトリーは、エクスポート後にシンボリックリンクを作成すると便利です。エクスポートするリポジトリーと CV には、リポジトリーディレクトリー構造の前に組織名と環境名が付けられるので、パスが長すぎることになりかねません。



重要

この例で使用されているディレクトリーはデモ用です。実際に使用するエクスポートディレクトリーには必要となるエクスポート RPM と ISO ファイルに十分な容量があることを確認してください。「[コンテンツ管理ストレージの定義](#)」で、ストレージ要件を推定する方法を説明します。エクスポートプロセス時に、`/var/lib/pulp/` ディレクトリーに一時ファイルが作成されます。つまり、エクスポートプロセス時には、エクスポートされるリポジトリーの 2 倍のサイズのストレージ容量が必要になります。エクスポートが完了したら、一時ファイルは削除されます。

エクスポートディレクトリーの作成

1. エクスポートディレクトリーを作成します。

```
# mkdir /var/www/html/pub/export
```

2. **foreman** ユーザーにエクスポートディレクトリーでの書き込みおよび読み取りパーミッションを付与します。

```
# chown foreman:foreman /var/www/html/pub/export
```

3. SELinux コンテキストを設定します。

```
# semanage fcontext -a -t httpd_sys_rw_content_t \
"/var/www/html/pub/export(/.*)?"
# restorecon -RvF /var/www/html/pub/export
# ls -Zd /var/www/html/pub/export
drwxr-xr-x. foreman foreman
system_u:object_r:httpd_sys_rw_content_t:s0 /var/www/html/pub/export
```

エクスポート先の設定

CLI をご利用の場合

エクスポート先を変更するには、以下の **hammer** コマンドを入力します。

```
# hammer settings set \
--name pulp_export_destination \
--value your-export-directory
```

たとえば、エクスポート先に `/var/www/html/pub/export/` を指定するには、以下を入力します。

```
# hammer settings set \
--name pulp_export_destination \
--value /var/www/html/pub/export
```

Web UI をご利用の場合

1. Web UI で、**管理 > 設定** に移動して、**コンテンツ** タブをクリックします。

2. **名前** カラムで **pulp_export_destination** 変数を探して、**値** フィールドをクリックします。
3. **値** フィールドに `/var/www/html/pub/export` などのエクスポート先を入力して、**保存** をクリックします。

D.5.2. ダウンロードポリシーの設定

ISS では **ダウンロードポリシー** を**即時** に設定する必要があります。これをグローバルに設定すると、すべての組織で作成される新規リポジトリに適用されます。または、各リポジトリに個別に設定することもできます。デフォルト値を変更しても既存設定は変更されません。

CLI をご利用の場合

グローバルでデフォルトのダウンロードポリシーを変更するには、以下の手順を実行します。

```
# hammer settings set \  
--name default_download_policy \  
--value immediate
```

特定のリポジトリのポリシーを変更する場合は、組織のリポジトリを以下のコマンドで一覧表示できます。

```
# hammer repository list \  
--organization-label organization-label
```

既存のリポジトリのダウンロードポリシーを変更するには、以下のコマンドを使用します。

```
# hammer repository update \  
--organization-label organization-label \  
--product "Red Hat Enterprise Linux Server" \  
--name "Red Hat Enterprise Linux 7 Server Kickstart x86_64 7.2" \  
--download-policy immediate
```

Web UI をご利用の場合

Web UI を使ってグローバルでデフォルトのダウンロードポリシーを変更するには、以下の手順を実行します。

1. **管理** > **設定** に移動します。
2. **コンテンツ** タブで **default_download_policy** を探します。
3. 値フィールドの編集アイコンをクリックします。
4. 値を **即時** に設定して **保存** をクリックします。

Web UI で既存のリポジトリのダウンロードポリシーを変更するには、以下の手順を実行します。

1. **Web UI** で **コンテンツ** > **製品** に移動して、該当製品名をクリックします。
2. **リポジトリ** タブで必要なリポジトリ名をクリックし、**ダウンロードポリシー** フィールドを見つけて、**編集**アイコンをクリックします。
3. リストから **即時** を選択して、**保存** をクリックします。

D.6. コンテンツのエクスポート

本セクションでは、お使いのアップストリームサーバーから異なるタイプのコンテンツをエクスポートし、1つ以上のダウンストリームサーバーにインポートする方法を説明します。コンテンツのインポートの詳細は「[コンテンツのインポート](#)」を参照してください。

アップストリームとダウンストリームのサーバー間の同期は、現在、完全な分離を必要とするエアギャップ環境などの非接続デプロイメントでサポートされています。

D.6.1. リポジトリのエクスポート

1. **hammer repository list** コマンドを使用して、エクスポートするリポジトリを一覧表示し、エクスポートで使用する ID を特定します。
2. アップストリームサーバーからコンテンツをエクスポートするには、**hammer repository export** コマンドを使用します。このコマンドは、**pulp_export_destination** 設定で指定しているディレクトリにコンテンツをエクスポートします。ISS ではデフォルトでディレクトリにエクスポートします。**--export-to-iso 1** パラメーターを使用すると、ディレクトリではなく ISO ファイルにエクスポートすることができます。以下は例となります。

```
# hammer repository export --id 1 [--export-to-iso 1]
```



注記

--export-to-iso パラメーターを使用する場合は、1 (ISO) または 0 (ディレクトリ) のいずれかを指定する必要があります。このパラメーターにはデフォルト値がありません。

D.6.2. コンテンツビューバージョンのディレクトリへのエクスポート

コンテンツビューの特定のバージョンをディレクトリにエクスポートできます。つまり、特定の CV に、要件に合うようなラベルを付けることができます。これで、エクスポートのトラッキングと更新が容易になります。

前提条件

- CV の全リポジトリのダウンロードポリシーを **即時** に設定していることを確認します。ポリシーが **即時** に設定されていないリポジトリは、エクスポートできません。
- 製品が、必要な日に同期されることを確認します。

Web UI をご利用の場合

1. **コンテンツ > コンテンツビュー** に移動し、**新規ビューの作成** をクリックします。以下の詳細を入力して CV を作成します。
 - a. **名前**: CV の簡単な名前。 **Export_CV** と入力します。
 - b. **ラベル**: CV の内部 ID。Red Hat Satellite 6 では、**名前** に入力した内容に基づいて、このフィールドに値が自動的に入力されます。
 - c. **説明**: プレーンテキスト形式の CV の説明。
 - d. **複合ビュー?**: 複合ビューを使用するかどうかを定義します。チェックを外しておきます。
2. **送信** をクリックして変更を保存します。

3. **リポジトリの選択** 画面で、**リポジトリの選択** テーブルから新しい CV に追加するリポジトリを選択します。**リポジトリの追加** をクリックして、CV に選択したパッケージを追加します。
4. **Yum コンテンツ > フィルター** に移動し、**新規フィルター** をクリックします。以下の詳細を入力して、エラータ以外のパッケージを含むフィルターを作成します。
 - a. **名前**: フィルターの簡単な名前。**Non-errata Products** と入力します。
 - b. **コンテンツタイプ**: フィルターに追加するコンテンツタイプのリスト。**パッケージ** を選択します。
 - c. **包含タイプ**: コンテンツを CV に包含または除外するかを定義するリスト。**包含** を選択します。
 - d. **説明** - フィルターの簡単な説明 (任意)。**Include all non-errata Products** と入力します。
 - e. **保存** をクリックします。
 - f. **Include RPM** 画面で、**エラータのないすべての RPM を含めます。** のチェックを外します。
5. **Yum コンテンツ > フィルター** に移動し、**新規フィルター** にクリックします。以下の詳細を入力し、必要な日付範囲に基づいてエラータパッケージを含むフィルターを作成します。
 - a. **名前**: フィルターの簡単な名前。**Erratas untill YYYY-MM-DD** と入力します。
 - b. **コンテンツタイプ**: フィルターに追加するコンテンツタイプのリスト。**Erratum - Date and Type** を選択します。
 - c. **包含タイプ**: コンテンツを CV に包含または除外するかを定義するリスト。**包含** を選択します。
 - d. **説明**: フィルターの簡単な説明 (任意)。**Include errata products untill YYYY-MM-DD** と入力します。
 - e. **保存** をクリックします。
 - f. **On the エラータの日付範囲** 画面が表示されます。エラータイプで **セキュリティ、機能強化、バグ修正** のすべてを選択します。
 - g. **データタイプ** で **更新日** にチェックを入れます。
 - h. **開始日** および **終了日** を入力して、製品に日付のフィルターを設定します。
 - i. **保存** をクリックします。
6. **新バージョンの公開** をクリックします。**説明** フィールドにリポジトリの日付範囲を入力することが推奨されます。**Force Yum Metadata Regeneration** のチェックは外します。
7. **保存** をクリックすると、CV バージョンを公開して、エクスポートする準備ができました。

CLI をご利用の場合

1. **hammer content-view create** コマンドを使用して、新しい CV を作成します。

```
# hammer content-view create \
--name "Export_CV" \
```

```
--organization "Default Organization"
```

2. **hammer content-view add-repository** コマンドを使用して、CV にリポジトリを追加します。

```
# hammer content-view add-repository \
--name "Export_CV" \
--product "Red Hat Satellite" \
--repository "Red Hat Satellite Tools 6 for RHEL 7 Server RPMs
x86_64" \
--organization "Default Organization"
# hammer content-view add-repository \
--name "Export_CV" \
--product "Red Hat Satellite Capsule" \
--repository "Red Hat Satellite Capsule Tools 6.2 for RHEL 7 Server
RPMs x86_64" \
--organization "Default Organization"
```

3. 新しい CV にフィルターを設定するには、以下を行います。

- a. **hammer content-view filter create** コマンドを実行して、エラータ以外のパッケージを含むフィルターを作成します。

```
# hammer content-view filter create \
--content-view "Export_CV" \
--inclusion true \
--name "Non-errata_Products" \
--type rpm \
--original-packages true \
--organization "Default Organization"
```

- b. **hammer content-view filter create** コマンドを使用して、エラータパッケージを含むフィルターを作成します。

```
# hammer content-view filter create \
--content-view "Export_CV" \
--inclusion true \
--name "Erratas until YYYY-MM-DD" \
--type erratum \
--organization "Default Organization"
```

- c. **hammer content-view filter rule create** コマンドを使用して、日付範囲を定義するルールを作成します。

```
# hammer content-view filter rule create \
--content-view "Export_CV" \
--content-view-filter "Erratas until YYYY-MM-DD" \
--end-date YYYY-MM-DD \
--types security,enhancement,bugfix \
--organization "Default Organization"
```

4. **hammer content-view publish** コマンドを使用して CV バージョンを公開して、エクスポートする準備ができました。 **--description** オプションの下で、リポジトリの日付範囲を入力することが推奨されます。

```
# hammer content-view publish \
--name "Export_CV" \
--description "Repositories until YYYY-MM-DD" \
--force-yum-metadata-regeneration true \
--async \
--organization "Default Organization"
```

エクスポートするコンテンツビューバージョンの決定

1. **hammer content-view version list** コマンドを使用して、エクスポートするコンテンツビューのバージョンを指定します。以下は例となります。

```
$ hammer content-view version list \
--organization "Default Organization"
-----|-----|-----|-----
ID | NAME | VERSION | LIFECYCLE
ENVIRONMENTS
-----|-----|-----|-----
3 | Export_CV 2.0 | 2.0 | Library
2 | Export_CV 1.0 | 1.0 | Library
1 | Default Organization View 1.0 | 1.0 | Library
-----|-----|-----|-----
-----
```

コンテンツビューバージョンのエクスポート:

1. **hammer content-view version export** コマンドを使用して、コンテンツビューのバージョンをエクスポートします。

```
# hammer content-view version export --id 3
```

D.6.3. 増分更新

最新の更新を受信する際に、Satellite Server から大型リポジトリのエクスポートを回避するには、増分更新を利用する方法があります。増分更新は、特定の日時以降に、1つ以上の同期イベントで行われたローカルのリポジトリへの変更をエクスポートするものです。

増分更新のリポジトリを作成するには、**hammer repository export** コマンドで **--since** オプションを使用します。以下は例となります。

```
# hammer repository export \
--id 1 [--export-to-iso 1] \
--since ISO_Date
```

ここでの **ISO_Date** は、**2010-01-01T12:00:00Z** のような ISO 8601 形式になります。

計算に使用されるタイムスタンプは、RPM が Satellite Server で同期した時間です。たとえば、Red Hat が月曜日に RPM をリポジトリに追加して水曜日に再度追加したとすると、木曜日にはローカルリポジトリの同期はできず、火曜日の日付を使用した水曜日分の更新のみが得られます。

この機能は、変更をリポジトリに追加するほかにも、Default Organization View のコンテンツビューで便利ですが、公開済み CV ではそれほど活用できません。

増分エクスポートから同期する際には、**hammer repository synchronize** コマンドで **--incremental** オプションを追加してください。このオプションを追加せず、「同期時のミラーリング」が有効になっていると、Satellite Server はインポートを完全なインポートとして扱い、増分エクスポートにない全データを消去してしまいます。このようなシナリオから復元するには、完全なエクスポートと、そこから同期を行うため時間がかかります。

D.7. コンテンツのインポート

Red Hat Satellite 6.3 では現在、非接続環境にあるアップストリーム Satellite Server からエクスポートされたコンテンツのインポートをサポートしています。この方法は、インターネットアクセスのない非接続の Satellite Server に使用し、サーバー間で DVD などの物理的なメディアによるコンテンツの移動が必要になります。

D.7.1. リポジトリのインポート

前提条件

- アップストリームの Satellite Server リポジトリをエクスポートします。詳細は「[リポジトリのエクスポート](#)」を参照してください。

リポジトリのインポート:

1. リポジトリに対してデータを、HTTPS ではなく HTTP で利用可能にします。たとえば、エクスポートしたディレクトリーを、ダウンストリームサーバーの `/var/www/html/pub/export/` ディレクトリーにコピーします。これにより、デフォルトで HTTP で利用できるようになります。
2. Web UI で、**コンテンツ > Red Hat サブスクリプション** に移動します。
3. **マニフェストの管理** を選択します。
4. **マニフェストのインポート/削除** タブでは、**Red Hat CDN URL** アドレスフィールドを、エクスポートしたリポジトリ内で、**content** ディレクトリーおよび **listing** ファイルの場所を一致させます。
たとえば、エクスポートしたリポジトリが `/var/www/html/pub/export/Default_Organization-Red_Hat_Enterprise_Linux_7_Server_RPMs_x68_64` にある場合は、URL を http://satellite.example.com/pub/export/Default_Organization-Red_Hat_Enterprise_Linux_7_Server_RPMs_x68_64/Default_Organization/Library/ に設定します。
5. **保存** をクリックします。
6. **コンテンツ > Red Hat リポジトリ** に移動して、エクスポートしたリポジトリを選択します。

D.7.2. Red Hat リポジトリとしてのコンテンツビューのインポート

前提条件

- アップストリームの Satellite Server に Red Hat リポジトリのコンテンツビューがあること。

- アップストリームの Satellite Server からコンテンツビューをエクスポートしていること。詳細は「[コンテンツビューバージョンのディレクトリーへのエクスポート](#)」を参照してください。



注記

また、カスタムリポジトリーは、カスタム製品、たとえば切断した Satellite にインポートできます。Red Hat Content Delivery Network (CDN) の詳細は『[Architecture Guide](#)』の「[Content Delivery Network \(CDN\) Structure](#)」を参照してください。

コンテンツビューのインポート

1. データを、HTTPS ではなく、HTTP で利用可能にします。たとえば、ダウンストリームサーバーの `/var/www/html/pub/export/` ディレクトリーにコピーします。
2. Web UI で、コンテンツ > **Red Hat** サブスクリプション に移動します。
3. マニフェストの管理 を選択します。
4. マニフェストのインポート/削除 タブで、**Red Hat CDN URL** アドレスフィールドを設定し、エクスポートしたコンテンツビューの **content** ディレクトリーおよび **listing** ファイルを一致します。
たとえば、エクスポートした CV が `/var/www/html/pub/export/Default_Organization-Export_CV-v1.0` にある場合は、URL を http://satellite.example.com/pub/export/Default_Organization-Export_CV-v1.0/Default_Organization/content_views/Export_CV/1.0/ に設定します。
5. **保存** をクリックします。
6. コンテンツ > **Red Hat** リポジトリー に移動して、エクスポートしたリポジトリーをチェックします。
7. ダウンストリームサーバーで、**hammer organization update** コマンドを入力して、組織に新しいリポジトリーを追加します。以下のように、エクスポートしたコンテンツビューのバージョンに対応するディレクトリーにアドレスを設定します。

```
$ hammer organization update \  
--name "Default Organization" \  
--redhat-repository-url \  
http://satellite.example.com/pub/export/Default_Organization-Export_\  
CV-v1.0/Default_Organization/content_views/Export_CV/1.0/  
  
Organization updated
```

付録E リモートファイルタイプリポジトリーの作成

pulp-manifest を使用して、Satellite Server の外部にあるファイルのディレクトリーから、カスタムファイルタイプリポジトリーを作成します。その後、HTTP または HTTPS 経由で Satellite Server にファイルを同期します。ファイルタイプリポジトリーにファイルを追加すると、他のリポジトリーと同じようにファイルを操作できます。

この手順は、リモートサーバーのディレクトリーにリポジトリーを設定する方法を説明します。Satellite Server がインストールされているベースシステムのディレクトリーにファイルタイプリポジトリーを作成するには、「[ローカルディレクトリーにカスタムのファイルタイプリポジトリーの作成](#)」を参照してください。

前提条件

リモートファイルタイプリポジトリーを設定するには、以下の条件を満たす必要があります。

- Red Hat Enterprise Linux 7 サーバーが Satellite または Red Hat CDN に登録されている。
- Red Hat Enterprise Linux Server および Satellite Tools リポジトリーにエンタイトルメントがある。
- HTTP サーバーがインストールされている。web サーバーの設定方法は『[システム管理者ガイド](#)』における Red Hat Enterprise Linux 7 での「[Apache HTTP Server](#)」を参照してください。

リモートディレクトリーにファイルタイプリポジトリーの作成:

1. サーバーおよび Satellite Tools リポジトリーが有効になっていることを確認します。

```
# subscription-manager repos --enable=rhel-7-server-rpms \  
--enable=rhel-7-server-satellite-tools-6.3-rpms
```

2. Pulp マニフェストパッケージをインストールします。

```
# yum install python-pulp-manifest
```

3. HTTP サーバーのパブリックフォルダーのファイルタイプリポジトリーとして使用するディレクトリーを作成します。

```
# mkdir /var/www/html/pub/my_file_repo
```

4. ディレクトリーにファイルを追加して、テストファイルを作成します。

```
# touch /var/www/html/pub/my_file_repo/test.txt
```

5. Pulp マニフェストコマンドを入力して、マニフェストを作成します。

```
# pulp-manifest /var/www/html/pub/my_file_repo
```

6. マニフェストが作成されたことを確認します。

```
# ls /var/www/html/pub/my_file_repo  
PULP_MANIFEST test.txt
```

リモートのファイルタイプレポジトリーからのファイルのインポート

1. ファイルタイプリポジトリの作成

Satellite Web UI で、**コンテンツ > 製品** に移動します。製品の名前を選択します (この例では **My File Product**)。リポジトリ タブで **新規リポジトリ** を選択し、以下の詳細を入力します。

- **名前:** リポジトリの簡単な名前。 **My Files** と入力します。
- **ラベル:** リポジトリの内部 ID。Red Hat Satellite 6 では、**名前** に入力した内容に基づいてこのフィールドに値が自動的に入力されます。
- **タイプ:** リポジトリのコンテンツタイプ。 **file** を選択します。
- **アップストリーム URL:** ソースとして使用するリモートリポジトリ。
- **SSL の検証:** アップストリームのリポジトリの SSL 証明書が信頼できる認証機関 (CA) によって署名されていることを確認したい場合のみ選択します。
- **アップストリームのユーザー名:** 認証に必要な場合は、アップストリームリポジトリのユーザー名を入力します。リポジトリに認証が必要ない場合はこのフィールドを空にします。
- **アップストリームのパスワード:** アップストリームリポジトリのパスワードを入力します。リポジトリに認証が必要ない場合はこのフィールドを空にします。

2. **保存** をクリックして、このリポジトリエントリを保存します。

3. ファイルタイプリポジトリを更新するには、**コンテンツ > 製品** に移動します。製品名を選択します (この例では **My File Product**)。更新したリポジトリ名を選択します (この例では **My Files**)。

4. **アクションの選択** メニューから **同期開始** を選択します。

5. リポジトリを公開した URL を開いて、ファイルを表示します。

付録F GIT を使用したテンプレートの同期

Red Hat Satellite 6 は、Satellite Server と、Git リポジトリまたはローカルディレクトリとの間で、ジョブテンプレート、プロビジョニングテンプレート、およびパーティションテーブルテンプレートの同期を有効にします。



注記

Git リポジトリまたはローカルディレクトリと、Satellite Server との間でテンプレートを同期するの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。

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

F.1. TEMPLATESYNC プラグインの有効化

1. Satellite Server でプラグインの有効化:

```
# 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。

F.2. TEMPLATESYNC プラグインの設定

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ます。

表F.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 リポジトリで、読み取るデフォルトブランチを指定します。	Git リポジトリで、書き込むデフォルトブランチを指定します。

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
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。
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。 <ul style="list-style-type: none"> ● 更新 — 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。

パラメーター	API パラメーター名	インポートの意味	エクスポートの意味
詳細	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

F.3. テンプレートのインポートおよびエクスポート

タスクのインポートおよびエクスポートは、一連の API コールを介して利用できます。API コールは、ロールベースのアクセスコントロールシステムを使用し、これによりどのユーザーもタスクを実行できます。TemplateSync プラグインにより、Git リポジトリまたはローカルディレクトリで同期できます。

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
%>
```

F.3.1. Git リポジトリでテンプレートの同期

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

```
# 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。

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

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

- Satellite の公開キーを使用して Git サーバーを設定します。これは、`/usr/share/foreman/.ssh/id_rsa.pub` に保存されます。
- Satellite Server から、**TemplateSync** メニューに指定した Git リポジトリにテンプレートをエクスポートします。

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

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

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

```
$ curl -H "Accept:application/json,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ません。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。

F.3.2. ローカルディレクトリー

ローカルディレクトリーでリビジョン管理システムリポジトリを設定した場合は、テンプレートをローカルディレクトリーと同期すると便利です。つまり、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 に同期します。

- 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
```

- 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。
- 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ません。この動作を上書きするには、**TemplateSync** メニューの **Force import** 設定を **yes** に変更するか、**force** パラメーター **-d '{ "force": "true" }'** を import コマンドに追加します。

注記

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

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

F.4. 高度な GIT 設定

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

自己署名の Git 証明書の同意

Git サーバーで自己署名証明書の認証を使用している場合は、**git config http.sslCAPath** コマンドでその証明書を検証します。

たとえば、以下のコマンドを実行して **/cert/cert.pem** に保存されている自己署名証明書を確認します。

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

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。

F.5. プラグインのアンインストール

アンインストール後にエラーを回避するには、以下を行います。

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
```