Red Hat Satellite 6.2

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

Last Updated: 2018-05-04

Red Hat Satellite 6.2 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

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、コンテナー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について説明します。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. 章の概要 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. 章の概要 12

第3章 サブスクリプションの管理 **14**
3.1. 複数のマニフェストを使用した複数の組織の管理 14
3.2. カスタマーポータルへの SATELLITE SERVER の追加 14
3.3. サブスクリプションマニフェストの作成 14
3.4. 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 15
3.5. マニフェストの更新 15
3.6. 章の概要 16

第4章 RED HAT コンテンツのインポート **17**
4.1. DEFINITIVE MEDIA LIBRARY の作成 17
4.2. SATELLITE での製品およびリポジトリーの使用 17
4.3. コンテンツの同期 17
4.4. ダウンロードポリシーの使用 17
4.5. 同期する RED HAT リポジトリーの選択 18
4.6. RED HAT リポジトリーの同期 19
4.7. 同期プランの作成 21
4.8. 章の概要 22

第5章 カスタムコンテンツのインポート **23**
5.1. SATELLITE でのカスタム製品の使用 23
5.2. カスタム製品の作成 23
5.3. カスタム GPG キーのインポート 24
5.4. カスタム RPM リポジトリーの作成 24
5.5. カスタム PUPPET リポジトリーの作成 26
5.6. 個別 PUPPET モジュールの管理 27
5.7. PUPPET リポジトリーの同期 28
5.8. GIT リポジトリーからの PUPPET MODULES の同期 29
5.9. 章の概要 30

第6章 アプリケーションライフサイクルの作成 **31**
6.1. アプリケーションライフサイクルの再検討 31
6.2. 新規アプリケーションライフサイクルの作成 31
6.3. 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 32
6.4. 章の概要 34

第7章 コンテンツビューの作成 **35**
7.1. シンプルなコンテンツビューの作成 35
7.2. PUPPET モジュールを含むコンテンツビューの作成 37
7.3. コンテンツビューのプロモート 38
7.4. コンテンツフィルターの定義 39

7.5. コンテンツフィルターの作成 41
7.6. 複合コンテンツビューの定義 43
7.7. 複合コンテンツビューの作成 44
7.8. 環境とコンテンツビューへのシステム登録 45
7.8.1. RHEL システムをサブスクリプションマネージャーに登録する 45
7.8.2. Atomic Host をサブスクリプションマネージャーに登録する 46
7.9. 章の概要 46

第8章 アクティベーションキーの管理 48
8.1. アクティベーションキーの作成 48
8.2. アクティベーションキーの使用 50
8.3. 章の概要 51

第9章 エラータの管理 52
9.1. コンテンツビューによるエラータ管理 52
9.2. 個別システムへのエラータの適用 53
9.3. 複数システムへのエラータの適用 54
9.4. 章の概要 55

第10章 コンテナーイメージの管理 56
10.1. RED HAT CONTAINER CATALOG からのコンテナーイメージのインポート 57
10.2. 他のイメージレジストリー からのコンテナーイメージのインポート 58
10.3. コンテンツビューによるコンテナーイメージの管理 60
10.4. DOCKER タグによるコンテナーイメージの管理 61
10.5. 章の概要 62

第11章 OSTREE コンテンツの管理 63
11.1. SATELLITE SERVER 上での OSTREE 管理の設定 63
11.2. 同期する RED HAT OSTREE コンテンツの選択 63
11.3. カスタム OSTREE コンテンツのインポート 64
11.4. コンテンツビューによる OSTREE コンテンツの管理 65
11.5. 章の概要 66

第12章 ISO イメージとファイルの管理 68
12.1. RED HAT からの ISO イメージのインポート 68
12.2. 個別の ISO イメージとファイルのインポート 69
12.3. RED HAT OVAL リポジトリーのインポート 70
12.4. 章の概要 72

第13章 コンテンツ管理の最終処理 73
13.1. シナリオにおける目的の完了 73
13.2. システムのプロビジョニング 73
13.3. その他のドキュメント 75

付録A コンテンツストレージ向け NFS 共有の使用 76

付録B 非接続の SATELLITE SERVER へのコンテンツ ISO のインポート 78

付録C 接続済み SATELLITE SERVER へのコンテンツ ISO のインポート 80

付録D SATELLITE SERVER 間でのコンテンツ同期 83
D.1. SATELLITE SERVER、CAPSULE SERVER、および ISS 84
D.2. 前提条件 84
D.3. サポートされる同期オプション 85
D.4. チャンク ISO ファイルの使用 85
D.5. ISS の設定 85

D.5.1. エクスポート先の設定 85
D.5.2. ダウンロードポリシーの設定 87
D.6. コンテンツのエクスポート 88
D.6.1. リポジトリーのエクスポート 88
D.6.2. コンテンツビューバージョン (CVV) のディレクトリーへのエクスポート 88
D.6.3. 増分更新 89
D.7. コンテンツのインポート 89
D.7.1. リポジトリーのインポート 90
D.7.2. Red Hat リポジトリーとしてのコンテンツビューのインポート 91

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などが含まれます (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ためにエンドツーエンドシナリオを提供します。これは、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としています。

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ーを提供するワークフローを意味します。Red Hat コンテンツの場合、Red Hat Satellite 6 はユーザーに利用可能なコンテンツを認識するためにサブスクリプション情報も使用します。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ー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。例えば、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、より複雑なアプリケーションライフサイクルには、テストやベータリリースなどのさらに多くの段階があることがあります。

- 開発

- テスト

- Beta リリース

- 実稼働

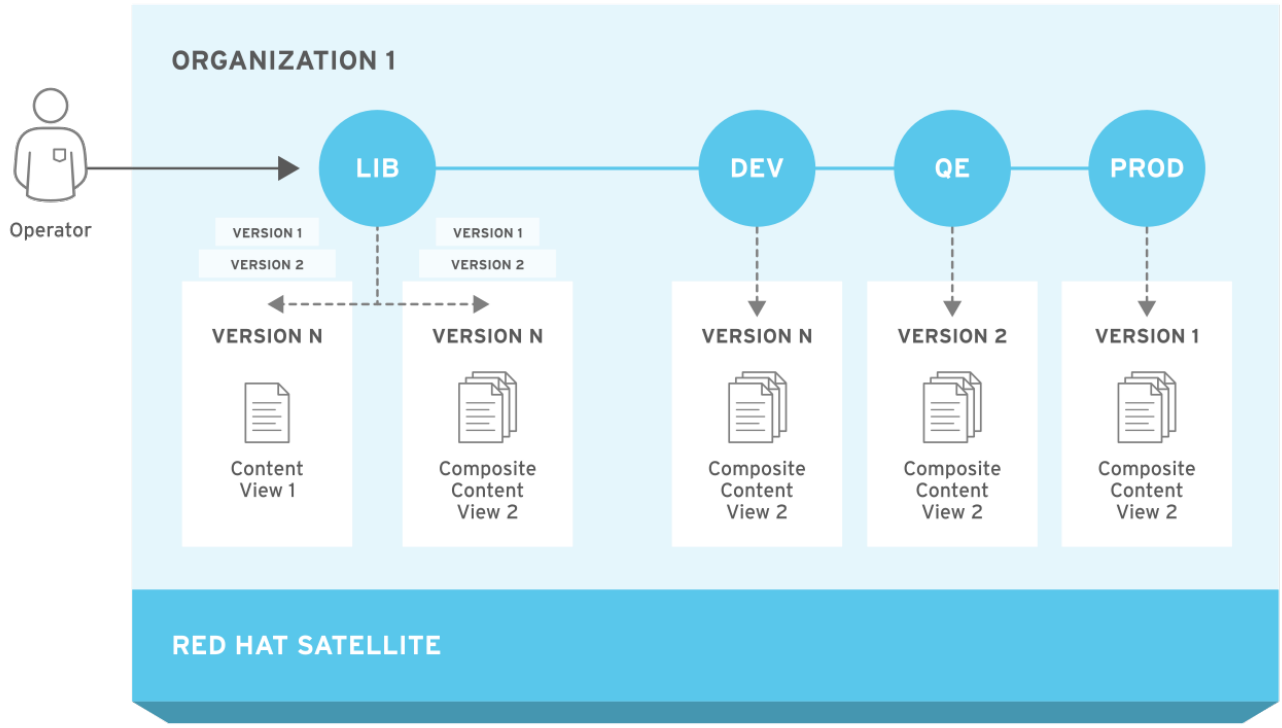
結局は、アプリケーションライフサイクルの段階は、それぞれの組織とソフトウェア開発方法によって異なります。Red Hat Satellite 6 はそれぞれの仕様を満たすために各アプリケーションライフサイクル段階をカスタマイズする方法を提供します。

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ー、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して動作します。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。例えば、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最適な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プロモートして、実稼働環境用のコンテンツビューに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行することになります。

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ーから RPM ファイルをインポートする方法が提供されます。したがって、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ーと 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。例えば、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。

コンテナーイメージ

Red Hat Satellite 6 は、コンテナー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ー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 の機能について説明します。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提供します。Red Hat Satellite 6 とのお好みの対話方法に応じていずれかを使用します。CLI を使用しており、**hammer** コマンドを実行するたびに認証詳細情報を提供したくない場合は、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、認証詳細情報は省略されます。

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ために **--async** オプションを使用できます。例えば、コンテンツを同期し、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** オプションを含める場合は、次のタスクに進む前にタスクを完了させてください。

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ーと独自のカスタムリポジトリーがホストされます。このようなリポジトリー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、リポジトリー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。例えば、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 に関する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** ディレクトリー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からのパッケージを同期し、保存します。これには、Red Hat Enterprise Linux および他の Red Hat ソフトウェア向けのリポジトリーが含まれます。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ーに対して最低 40GB
- 各デバッグ情報リポジトリーに対して最低 80GB

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

- これらのリポジトリー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ーごとに 10GB、デバッグ情報リポジトリーごとに 20GB です。

Red Hat 実稼働フェーズの詳細については、"Red Hat Enterprise Linux Life Cycle" を参照してください。

注記

すべてのリポジトリー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ー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して動作します。これにより、既存のリポジトリーからカスタマイズ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** ディレクトリーに格納された 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、最小の数のパッケージを使用しているコンテンツビューは少量のストレージを使用します。ただし、ストレージサイズは、複数のコンテンツビューを使用し、ビューごとに大量のパッケージを使用すると、増加します。

例えば、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。このディスク領域はシンボリックリンクの 100MB 未満になります。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獲得します。
- 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、コンテンツビューを構築するためにこのディレクトリーを使用します。
- **/var/lib/pulp/published** ディレクトリーを監視します。Red Hat Satellite 6 は、コンテンツビューを公開するためにこのディレクトリーを使用します。

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、大量の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最小 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。例えば、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. 章の概要

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について説明しました。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について説明します。

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、組織管理のいくつかの例を示します。

単一の組織

単一のシステム管理チェーンを持つ小規模な会社。この場合は、会社に対して単一の組織を作成し、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 (例えば、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 (例えば、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と、外部の各会社の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 は単一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、そのユーザーでログアウトし、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。これにより、Satellite Server が現在管理している組織のリストが表示されます。

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アセット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 1 つの組織が必要です。ただし、一部の状況では、ホストが孤立することがあります。例えば、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。これらの状況では、必要に応じて、新しく作成され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る場合は **すべてを割り当て** を選択し、割り当てる孤立した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ないはずなので、**編集に進む** をクリックして **プロパティーの編集** セクションに移動します。

プロパティーの編集

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ることにより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場合、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、コンテキストは ACME 組織に変更されます。

CLI をご利用の場合

CLI を使用している場合は、コマンドの最後にオプションとして **`--organization "ACME"`** または **`--organization-label "ACME"`** のいずれかを含めます。

```
# 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. 章の概要

本章では、新しい組織を作成し、コンテンツ管理のコンテキストとして設定する方法について説明しました。

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について説明しま

す。サブスクリプションがインポートされると、**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。これを行うには、Satellite Server が利用可能な製品サブスクリプション認識する必要があります。これにより、対応するリポジトリーを見つけてアクセスし、そのリポジトリー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。この情報を Satellite にインポートするために、**サブスクリプションマニフェスト**を作成します。

サブスクリプションマニフェストは、サブスクリプション情報が含まれる暗号化されたファイルのセットです。このマニフェストは Satellite Server にインポートします。Satellite Serve はこの情報を使用して CDN にアクセスし、利用可能なリポジトリーを見つけます。

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マニフェストを作成する方法について説明します。

3.1. 複数のマニフェスト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 で複数のマニフェストを使用できます。Satellite 6 では、各組織ごとにSatellite で設定された単一マニフェストが必要になります。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、複数の組織それぞれを独自の 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への SATELLITE SERVER の追加

Red Hat カスタマーポータルでは、サブスクリプション情報にアクセスできます。また、Satellite Server をサブスクリプション管理アプリケーションとして追加することもできます。これは、カスタマーポータルでサブスクリプションを Satellite に割り当てるために必要になります。

このシナリオでは、カスタマーポータルで Satellite Server をコンテンツコントリビューターとして追加します。以下の手順に従ってください。

1. ブラウザーで https://access.redhat.com/ を開き、カスタマーポータルアカウントにログインします。
2. カスタマーポータルの左上隅にある **サブスクリプション** に移動します。
3. このページにはサブスクリプション情報が表示されます。カスタマーポータルに登録されたすべてのシステムが表示された **Manage** セクションにスクロールします。このセクションにも **Subscription Management Applications** が表示されます。**Satellite** をクリックします。
4. このページには Satellite Server の一覧が表示されます。それには、以前に追加された Red Hat Satellite 5 または 6 Server が含まれます。このシナリオでは、この一覧は空であるはずです。**Register a Satellite** をクリックします。
5. カスタマーポータルでは、Satellite Server の **名前** と **バージョン** を含む Satellite に関する基本詳細情報を入力するフォームが提供されます。これらの詳細情報を入力したら、**Register** をクリックします。

この時点でカスタマーポータルには、Satellite Server のエントリーが含まれます。このページではサブスクリプションを割り当て、サブスクリプションマニフェストを作成することができます。

3.3. サブスクリプションマニフェストの作成

Satellite サーバーのカスタマーポータルページでは、サブスクリプションのグループを収集し、管理対

象システムに配布するためにサーバーに割り当てます。説明したように、Satellite Server 向けのサブスクリプションマニフェストを作成します。サブスクリプションを割り当て、ACME のマニフェストを作成するには、以下の手順を実行します。

1. Satellite のカスタマーポータルページで、**Attach a subscription** をクリックします。
2. Red Hat 製品サブスクリプションの一覧が表示されます。Satellite のサブスクリプションマニフェストに割り当てる製品を選択し、選択された各製品の **数量** も入力します。このシナリオでは Red Hat Enterprise Linux 7 のサブスクリプションを選択し、数量として 10 を入力します。**選択項目のアタッチ** をクリックして割り当てを完了します。
3. カスタマーポータルでは、選択されたサブスクリプション証明書をエンコードし、ACME サブスクリプションマニフェストである **.zip** アーカイブを作成できます。**マニフェストのダウンロード** をクリックしてマニフェストを取得します。

これで、Satellite Server にアップロードするサブスクリプションマニフェストを取得できました。

3.4. 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

コンテキストが ACME 組織に設定されていることを確認し、**コンテンツ > Red Hat サブスクリプション** に移動します。このシナリオでは、空のページが表示されます。**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。**参照...** をクリックし、サブスクリプションマニフェストを選択して、**アップロード** をクリックします。マニフェストは Satellite Server によりアップロードされ、数分後に **マニフェストの履歴** セクションに正常なインポートが報告されます。

CLI をご利用の場合

Red Hat Satellite 6 CLI を使用するには、マニフェストが Satellite Server 上にある必要があります。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。

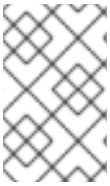
```
[user@client ~]$ scp ~/<manifest_file>.zip root@satellite.example.com:~/.
```

次に、以下のコマンドを使用してインポートします。

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

数分後に、CLI により、正常なマニフェストのインポートが報告されます。

3.5. 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が 登録解除されます。

カスタマーポータルでマニフェストに対するリポジトリーの追加と削除を行い、更新されたマニフェストを以下の 3 つのいずれかの方法で Satellite に追加します。

- 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 を使用して Satellite Server にアップロードします。

```
# hammer subscription upload --file ~/<manifest_file>.zip
```

3.6. 章の概要

本章では、サブスクリプションマニフェストを使用して Red Hat カスタマーポータルから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について説明しました。

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

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 を作成する方法について説明します。

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ーです。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ーの使用

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ーをグループ化します。このようなリポジトリーコレクションは実際の製品の概念と似ています。例えば、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ー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ー、補助リポジトリー、Virt V2V ツール) から構成されることになります。これにより、関連するすべてのリポジトリーが 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ーが一緒に同期されます。Red Hat リポジトリーの場合、製品はリポジトリー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ーとを同期します。

4.3. コンテンツの同期

DML を構成するリポジトリーを選択すると、Satellite Server により独自のリポジトリーと Red Hat CDN 上のリポジトリーが同期されます。これにより、Satellite Server では Red Hat のリポジトリーの同一コピーが DML の一部として保持されます。Satellite Server はこのリポジトリー情報を取得し、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 内のリポジトリーが CDN のリポジトリーと同期された状態であることを確認する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 の使用の詳細については、*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、以下で説明されているように **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

Satellite Server 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。例えば、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により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 は同期中にすべてのメタデータとパッケージをダウンロードします。
- **オンデマンド** - Satellite Server は同期中にメタデータのみをダウンロードします。サーバーは、クライアントが要求した場合にのみ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。

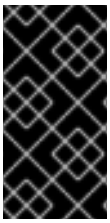
- **背景** - Satellite Server は、最初の同期後にすべてのパッケージをダウンロードするバックグラウンドタスクを作成します。

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** リポジトリーのみに使用してください。

Satellite Server に格納されたすべてのリポジトリーは、ダウンロードポリシーを使用します。ダウンロードポリシーは、それぞれのニーズに合わせて変更できます。

4.5. 同期する RED HAT リポジトリーの選択

同期するリポジトリーを選択する最初の手順では、リポジトリー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ーを有効にします。



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。詳細については、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 リポジトリー 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** である必要があります。このタブには、RPM コンテンツを提供するサブスクライブ済みのすべての製品の一覧が含まれます。

製品と特定のリポジトリーの関係は、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ーセットの一覧が開きます。リポジトリーセットを選択すると、有効にできるリポジトリーの一覧が開きます。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 リポジトリーを関連付けることと 7.X リポジトリーを関連付けることの違いは、7 Server にはすべての最新アップデートが含まれますが、Red Hat Enterprise Linux 7.X リポジトリーは次のマイナーバージョンリリース後にアップデートの取得を止めることです。キックスタートリポジトリー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

製品とリポジトリーの関係は同じです。以下のコマンドを使用して製品を検索します。

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

製品のリポジトリーセットをリストします。

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

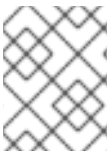
これにより、製品のリポジトリーセット内のリポジトリーが表示されます (名前と ID 番号を含む)。名前または ID 番号を使用してリポジトリーを有効にします。また、リリースバージョン (**7Server**) とベースアーキテクチャー (**x86_64**) を含めます。以下に例を示します。

```
# 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ーを有効にします。

リポジトリー	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ー。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** リポジトリーの代わりに **7 Server** リポジトリーを使用します。
Red Hat Satellite Tools 6.2 (RHEL 7 Server RPM x86_64 向け)	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 リポジトリー。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** リポジトリーの代わりに **7 Server** リポジトリーを使用します。
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ーは、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基づいて複数のリポジトリーを選択できます。



注記

このシナリオでは、すべてのリポジトリーで **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 リポジトリーの同期

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

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、製品内のすべての有効済みリポジトリーが表示されます。すべてのリポジトリーを選択し、**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/リポジトリー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ーを同期します。

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

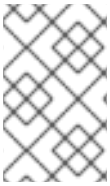
また、各リポジトリーを個別に同期することもできます。製品内のすべてのリポジトリーをリストし、対応するリポジトリーの ID 番号を使用して同期します。以下に例を示します。

```
# 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、各リポジトリー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の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

同期が完了したら、キックスタートリポジトリーがインストールメディアの Red Hat Satellite 6 のリストに表示されます。これを参照するには、**ホスト > インストールメディア** に移動します。

注記

Red Hat リポジトリー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ーを選択し、**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** コマンドを **--download-policy** オプションとともに使用します。

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および更新します。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

以下のコマンドを使用して同期計画を作成します。

```
# 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 DML コンテンツを Red Hat CDN に対してチェックし、Red Hat リポジトリーを最新の状態にします。

4.8. 章の概要

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について説明しました。

次章では、Satellite Server の DML へのカスタムコンテンツのインポートについて説明します。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は、Red Hat コンテンツのインポートと同様のものです。

第5章 カスタムコンテンツのインポート

前の章では、Definitive Media Library (DML) への Red Hat コンテンツのインポート方法について説明しました。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について説明します。事前に独自の製品を作成し、カスタマイズして、独自のリポジトリーを追加します。さらに、カスタムリポジトリーに 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ーの使用」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ーをグループ化する方法について説明しました。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ーを追加することをもできます。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ー間の関係は同じであり、リポジトリー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ー (RPM コンテンツを含む RPM リポジトリーと RPM コンテンツを設定するためのモジュール向け Puppet リポジトリー) を含む製品を作成します。

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 リポジトリーから 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よう 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。**`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ーに割り当てるため、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`** に割り当てることができます。
- **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

1 つのコマンドで製品を作成します。

```
# 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ーを作成し、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 リポジトリーとの 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ー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**) を提供し、**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、リポジトリーに関連付ける GPG キーが用意できました。

5.4. カスタム RPM リポジトリーの作成

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 の開発者はより新しいバージョンの

PostgreSQL で Exampleware をテストするとします。このような場合は、新しいバージョンの PostgreSQL 向けのカスタムリポジトリーを作成できます。

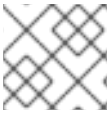
Web UI をご利用の場合

カスタム PostgreSQL 製品を作成します。カスタム製品の作成後は、リポジトリー画面が表示されます。**リポジトリーの作成** をクリックし、新しいリポジトリー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** - リポジトリーの簡単な名前。**`PostgreSQL 9.5`** と入力します。
- **ラベル** - リポジトリーの内部 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** - リポジトリーのタイプ。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ーを選択できます。本ガイドのシナリオでは、**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ーの 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ー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** - リポジトリーのチェックサム。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 およびそれ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** - リポジトリーを 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** - このリポジトリーの 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ーエントリーを保存します。

リポジトリー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 リポジトリーを選択して **同期開始** をクリックします。これでリポジトリーと外部 PostgreSQL リポジトリーとの同期が開始されます。



注記

この同期の進捗状況は、**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ーを作成します。

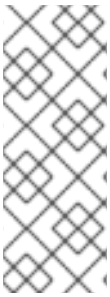
```
# hammer repository create \
--name "PostgreSQL 9.5" \
--content-type "yum" \
--publish-via-http true \
```

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

次にリポジトリーを同期します。

```
# 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 **リポジトリーの検出** 機能があり、これは URL からすべてのリポジトリーを発見し、自分のカスタム製品に追加するよう選択できるようにします。例えば、**リポジトリーの検出** を使って **`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 リポジトリーをすべて一覧表示することができます。こうすることで、単一ソースから複数のリポジトリー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 リポジトリーの作成

カスタム製品にも Puppet モジュールのリポジトリー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 製品のリポジトリーを作成します。

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てください。PostgreSQL 製品をクリックするとリポジトリー一覧が表示されます。**リポジトリーの作成** をクリックして、新規リポジトリーのフォームを表示します。以下の詳細を入力します。

- **名前** - リポジトリーの簡単な名前。**`PostgreSQL Puppet Modules`** と入力します。
- **ラベル** - リポジトリーの内部 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** - リポジトリーのタイプ。**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ーの 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ーソースを使用することができます。

保存 をクリックしてこのリポジトリーエントリーを保存します。

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ーを作成します。

```
# 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ーができました。モジュールを加えていきます。

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。ブラウザー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 製品のリポジトリー一覧ページを開いていることを確認します。PostgreSQL Puppet Modules リポジトリーをクリックし、そのリポジトリーの詳細ページを表示します。

Upload Puppet Module セクションにスクロールし、**参照** をクリックしてダウンロードした PostgreSQL Puppet Module を選択し、アップロードします。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

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

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

Puppet モジュールを PostgreSQL Puppet Modules リポジトリー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、設定するための、RPM コンテンツ と Puppet モジュールの両方を格納したカスタムリポジトリーができました。

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ついて問題がある場合は、モジュール開発者に連絡してください。

5.7. PUPPET リポジトリーの同期

アップロードした Puppet モジュールのリポジトリー作成に加え、Satellite Server は完全な Puppet モジュールリポジトリーの同期ができます。この例では、Satellite Server は Puppet Forge リポジトリー全体を同期します。

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** に割り当てることができます。
- **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、リポジトリー画面が表示されます。**リポジトリーの作成** をクリックし、新しいリポジ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** - リポジトリーの簡単な名前。**Puppet Forge Modules** と入力します。
- **ラベル** - リポジトリーの内部 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** - リポジトリーのタイプ。**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ーの 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ーエントリーを保存します。

Puppet Forge Modules リポジトリーを選択し、**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 リポジトリーを作成します。

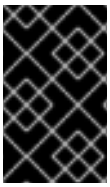
```
# hammer repository create \
--name "Puppet Forge Modules" \
--content-type "puppet" \
--product "Puppet Forge" \
--organization "ACME" \
--url http://forge.puppetlabs.com/
```

リポジトリーを同期します。

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

注記

Puppet Forge リポジトリーには数千のモジュールが格納されているので、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ついて問題がある場合は、モジュール開発者に連絡してください。

5.8. GIT リポジトリーからの 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 リポジトリー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発行します。よくある方法の 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ーに公開して、そのディレクトリーに対して同期するというものです。例を示します。

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

これで Git リポジトリーの **develop** ブランチが **git@mygitserver.com:mymodules.git** からチェックアウトされ、**/modules** に発行されます。このディレクトリーを Satellite Server の新規リポジトリーの URL (**file:///modules**) として追加します。

Web UI をご利用の場合

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

- **名前** - リポジトリーの簡単な名前。**Modules from Git** と入力します。
- **ラベル** - リポジトリーの内部 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** - リポジトリーのタイプ。**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ーの URL。**file:///modules** と入力します。

CLI をご利用の場合

Puppet Forge リポジトリーを作成します。

```
# 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発行する場合でも同じプロセスを実行します。例えば、Puppet モジュールを発行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 上では、リポジトリーの URL を **http://webserver.example.com/modules/** に設定します。

5.9. 章の概要

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ーの作成方法を説明しました。これには RPM ファイルや Puppet モジュールが含まれます。

これで、ベースコンテンツすべてが 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ニングと Red Hat Satellite 6 での実装について説明します。

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。

例えば、E メールサーバーの場合、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され、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

本ガイドのシナリオでは、ACME の Exampleware を開発、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、以下の順序で 3 つの環境を続けます:**開発**、**テスト**、および **実稼働**。

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、ライブラリー環境のみが存在します。

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`** を入力すると **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m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、**--prior** オプションを使用します。例を示します。

```
# 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. 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に渡って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になります。

ACME では Exampleware のパッチをリリースするので、Exampleware バージョン 1.1 の開発を開始します。このインスタンスでは、アプリケーションライフサイクルの環境には以下の 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

パッチの開発が完了したら、ACME は 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は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たので、Exampleware 1.1 のリリース準備が完了しました。ACME では 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。レビューが完了すると、パッケージは実稼働環境にプロモートされます。

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ステップを説明しています。これら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ーのみです。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ー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、リポジトリーに公開されたものです。Red Hat のリポジトリーとこれまでに同期したカスタムリポジトリー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ー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 のリポジトリーは、Exampleware RPM の新バージョンが含まれているバージョン 1.1 のリポジトリーに置き換えられます。

6.4. 章の概要

本章ではアプリケーションライフサイクルの概念と Red Hat Satellite 6 のコンテンツ管理について説明しました。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についても説明しました。

次章では、コンテンツビューとそれを使用したカスタマイズリポジトリーを既存の DML から作成する方法について説明します。

第7章 コンテンツビューの作成

Red Hat Satellite 6 では、コンテンツビューを使用して Definitive Media Library (DML) のコアリポジトリーからカスタマイズリポジトリーを作成します。使用するリポジトリーを定義し、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、実稼働環境では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ている間、開発環境では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にまたがるリポジトリーセットを作成し、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ある環境から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 上のリポジトリーがパッケージを更新、公開します。例えば、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ーには、**`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ーが再生成され、**`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、その環境が新バージョンのコンテンツビューを使用する際には更新を受け取ることができます。

本章では、異なるタイプのコンテンツビューの作成方法と、これに各種フィルターを適用する方法について説明します。

7.1. シンプルなコンテンツビューの作成

この例では、リポジトリー 2 つとフィルターなしというシンプルなコンテンツビューを作成します。リポジトリーは、Red Hat Enterprise Linux リポジトリーと Satellite Tools リポジトリーになります。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** のフォームが表示されます。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。

これで新規コンテンツビューのエントリーが作成され、リポジトリーを追加することができます。Red Hat Enterprise Linux 7 Server RPMs (Kickstart RPMs ではありません) と Red Hat Satellite Tools のリポジトリーを選択し、**リポジトリーの追加** をクリックします。これでコンテンツビューにこれらのリポジトリーから全パッケージが追加されます。

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 が新規バージョン (バージョン 1) についての詳細を提供し、**説明** にこのバージョンについての説明が入力できます。ここには、新規コンテンツビューの変更を記録しておくことができます。**`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

リポジトリー ID の一覧を取得します。

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

この例では、2 つのリポジトリーはそれぞれ 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 6.2 (RHEL 7 Server RPM x86_64 向け)

コンテンツビューを作成し、リポジトリーに追加します。

```
# 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. 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、リポジトリー1つとフィルターなしというコンテンツビューを作成します。リポジトリーは、PostgreSQL リポジトリーになります。

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** のフォームが表示されます。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。

これで新規コンテンツビューのエントリーが作成されます。PostgreSQL のリポジトリーを選択し、**リポジトリーの追加** をクリックします。これでコンテンツビューに PostgreSQL リポジトリーから全パッケージが追加されます。

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。これでインポート済みの Puppet モジュールが表示されます。**`postgresql`** モジュールまでスクロールし、**バージョンの選択** をクリックします。

最新を使用 のエントリーをスクロールし、**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

リポジトリー ID の一覧を取得します。

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

必要になるのは PostgreSQL RPM リポジトリーのみで、Puppet モジュールリポジトリーは必要ありません。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、リポジトリーに追加します。

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

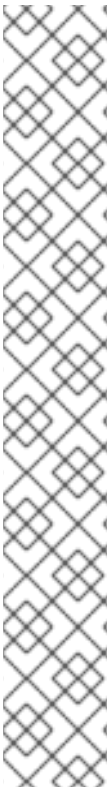
ビューを公開します。

```
# 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. 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ー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ーが他の環境で利用可能になるようにしましょう。



注記

管理者以外のユーザーは環境にコンテンツビューをプロモートするために、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。例えば、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。

プロモート ボタンを再度クリックします。**Production** 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。

これでこのコンテンツビューのリポジトリーが全環境に表示されます。

CLI をご利用の場合

`hammer content-view version promote` を毎回使用してコンテンツビューをプロモートします。

```
# 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"
```

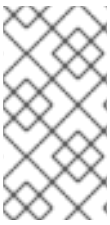
これで **Database** のコンテンツが全環境で利用可能になります。

7.4. コンテンツフィルターの定義

コンテンツビューでは、フィルターを使って特定の 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、選択したリポジトリーからのすべてのものを含めることになってしまいます。

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。このフィルターの動作は、コンテンツなしから始まり、選択したリポジトリーから追加するコンテンツを選択する、という流れになります。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。このフィルターの動作は、選択したリポジトリーからのすべてのコンテンツから始まり、除外するコンテンツを選択する、という流れになります。リポジト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、特定のパッケージを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、リポジトリーで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。

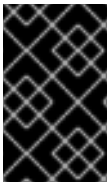


注記

組み込みと除外のフィルターの組み合わせを使用して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 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、これらを除いて定期的にシステムアップデートを実行する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** は選択解除します。**日付タイプ** を **更新日** に設定します。**開始日** をエラータを制限する日付に設定します。**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がフィルター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処理してから、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、**セキュリティー** は選択解除します。**日付タイプ** を **更新日** に設定します。**開始日** をエラータを制限する日付に設定します。**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がフィルター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、以下のアーティクルを参照してください: "How do content filters work in Satellite 6"

7.5. 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、**Base** 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`** (セキュリティー更新以外で、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。

```
[ ] セキュリティー
[X] 機能強化
[X] 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** では、エラータアイテムは発行日でフィルタリングされるだけで、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 では、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ーを指定することもできます。**影響するリポジトリー** タブを選択してリポジトリーを指定します。この例では、フィルターで使用するリポジトリーは 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ーを公開します。**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) 数が公開前のリポジトリーと比べて少なくなります。これは、フィルターが昨年から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`** オプションを使っ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"
```

7.6. 複合コンテンツビューの定義

複合コンテンツビューは、複数のコンテンツビューからコンテンツを組み合わせます。例えば、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ーに統合できます。元のコンテンツビューのリポジトリーはそのまま存在しますが、組み合わされたコンテンツには新規リポジトリーが使用されます。

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 Stack
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 つの個別コンテンツビューを開発するとします。

- 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)

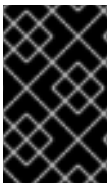
すると、次に 2 つの複合コンテンツビューを作成できます。1 つ目には PostgreSQL データベースを使います。

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個に管理、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、各リポジトリーで 1 つしか許可されません。例えば、あるリポジトリーを使用したコンテンツビュー 2 つを含めようとすると、Satellite Server はエラーをレポートします。

7.7. 複合コンテンツビューの作成

既存のコンテンツビュー 2 つを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ので、どちらかを選択します。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`** と記入して **保存** をクリックします。

公開が完了すると、**コンテンツ** コラムに含まれた全コンテンツビューのパッケージ、エラータ、Puppet モジュールカウント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"
```

7.8. 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。

7.8.1. RHEL システムをサブスクリプションマネージャーに登録する

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。例えば、ホスト名が **satellite6.example.com** の Satellite Server の場合、以下のコマンドを登録するクライアントで実行します。

```
[root@client ~]# rpm -Uvh http://satellite6.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を尋ねます。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* で説明しています。

クライアントシステムは、これで開発環境にある Stack コンテンツビューから公開されたリポジトリーを使用します。

7.8.2. Atomic Host をサブスクリプションマネージャーに登録する

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。

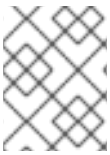
```
[root@atomic_client ~]# chmod +x katello-rhsm-consumer
```

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ので、これに **katello-agent** をインストールすることは推奨されません。

7.9. 章の概要

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リポジトリーをカスタマイズする方法について説明しました。以下の方法がありました。

- RPM のある基本的なコンテンツビューの作成
- RPM と Puppet モジュールの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った 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消費

次章では、アクティベーションキーと、それを使ったシステム登録およびアプリケーションライフサイクル環境からのコンテンツへのアクセスについて説明します。

第8章 アクティベーションキーの管理

ここまでにコンテンツビューが公開されており、その結果、Satellite Server がリポジトリーを公開しています。システムは Satellite Server に登録し、それらのリポジトリーからのコンテンツを消費できます。ここでのシステムの登録は、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。例えば、Red Hat サブスクリプションマネージャー (`subscription-manager`) で `--baseurl` を使うことにより、Red Hat Content Delivery Network ではなく Satellite Server にポイントすることができます。

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。より好ましい別の方法ではアクティベーションキーを使用します。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メントが容易にできます。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。例えば、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

Web UI をご利用の場合

コンテンツ > アクティベーションキー に移動し、**新規アクティベーションキー** をクリックします。以下の情報を入力します。

- **名前** - アクティベーションキーの名前。システムの登録プロセスでこの名前を使用します。`development-stack` と入力します。
- **コンテンツホスト制限** - このアクティベーションキーで Satellite Server が登録を許可するシステム数を定義します。**無制限** 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。`Exampleware Stack in the Development Environment` と入力します。
- **環境** - 使用する環境。`Development` を選択します。
- **コンテンツビュー** - 環境内で使用するコンテンツビュー (およびリポジトリー) 。`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。

ここで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ーを定義する必要があります。**サブスクリプション** タブに移動します。空の製品/サブスクリプション一覧が表示されます。**追加** をクリックして Red Hat Enterprise Linux サブスクリプションと PostgreSQL 製品の両方を選択し、**選択を追加** をクリックします。

注記

Auto-Attach オプションは有効になっています。これは、これらの製品を登録時に自動的にアタッチし、必要なリポジトリーを有効にします。アクティベーションキーの **Auto-Attach** が無効になっている場合は、システムはサブスクリプションやコンテンツをアタッチせずに Satellite Server に登録するだけになります。

製品コンテンツ ページに移動します。アクティベーションキーの製品に関連付けられている全リポジトリーが表示されます。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ー。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。

このシナリオでは以下のデフォルトセットがあります。

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 6.2 (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 リポジトリーには設定ツール (**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 6.2 (for RHEL 7 Server) (RPMs) - 有効にされていますか: **Yes に上書き**

保存 をクリックします。

これでシステム登録に使用するアクティベーションキーの準備ができました。

CLI をご利用の場合

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"
```

サブスクリプション ID 一覧を取得します。

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

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割り当てます。

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

```
--organization "ACME"
```

PostgreSQL 製品をアクティベーションキーに割り当てます。

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

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"
```

Red Hat Satellite Tools 6.2 リポジトリー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。以下のコマンドで有効にします。

```
# hammer activation-key content-override \
--name "stack-development" \
--content-label rhel-7-server-satellite-tools-6.1-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。

アクティベーションキーはテストすることができます。既存の Red Hat Enterprise Linux 7 システムを Satellite 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** ディレクトリーに配置されています。例えば、ホスト名が **satellite6.example.com** の Satellite Server の場合、以下のコマンドを登録するクライアントで実行します。

```
# rpm -Uvh http://satellite6.example.com/pub/katello-ca-consumer-
latest.noarch.rpm
```

この RPM は Satellite Server 上のリポジトリー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="stack-development" --
org="acme"
```

```
The system has been registered with id: 744fb31c-c983-00f5-ca14-
bddd0f711353
```

Satellite Server サーバーで登録を確認します。

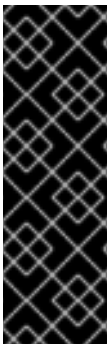
Web UI をご利用の場合

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、以下のコマンドでシステムに **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 6 Tools リポジトリーで提供しています。

8.3. 章の概要

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 に登録する方法について説明しました。

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について説明します。

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フィクスを説明。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、リポジトリーを Red Hat の Content Delivery Network (CDN) と同期する際にこれらのエラータをインポートします。Red Hat Satellite 6 ではエラータ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。

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。

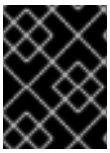
適用可能

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について説明します。



重要

Satellite に登録したホストに **katello-agent** パッケージをインストールしてください。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、エラータの管理と適用の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の定義」 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ってエラータを制限することができます。フィルターには以下のものがあります。

- **ID** - リポジトリーに許可する特定の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になります。その後に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については、「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が含まれたら、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. 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、**詳細** セクションに更新されたパッケージとその新バージョンの一覧が表示されます。例を示します。

```
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
```

9.3. 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 に移動します。これで同期リポジトリー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、各エラータを適用、インストール可能な登録済みホストの数が表示されます。

単一の OpenSSL エラータをこのツールを使ってシステムに適用します。検索ボックスに **`title ~ openssl`** と入力します。これで 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」で作成したフィルターのために、Satellite Server はこれらのエラータはテストシステムには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に制限します。

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、選択したシステム用の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"
```

エラータを単一のシステムに適用します。

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

これらのコマンドは各システムごとに実行する必要があり、**--host** はそのシステム名で置き換えます。以下のコマンドでこれを実行できます。

```
# 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 を適用可能なホストすべてを特定し、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
```

9.4. 章の概要

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について説明しました。

次章では、Red Hat Satellite 6 におけるコンテナー管理について説明します。

第10章 コンテナーイメージの管理

コンテナー化とは、システムレベルの仮想化メソッドを操作することです。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、コンテナー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、コンテナー内のプロセスは、ホストや他のコンテナーとリソースを共有することになります。これにより、単一ホストから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 コンテナー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、簡単なテスト、メンテナンス、トラブルシューティングも提供します。

コンテナーについての詳細情報は、**Red Hat Enterprise Linux Atomic Host 7** の コンテナーの使用ガイド を参照してください。

Docker は、Linux コンテナー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ー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ーは以下で構成されています。

コンテナー

アプリケーションのサンドボックスです。各コンテナー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ーを起動すると、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。コンテナー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、コンテナー設定の静的なスナップショットです。コンテナー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、コンテナー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

リポジトリーにあるイメージを、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。リポジトリーは、コンテナー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、ヒューマンリーダブルな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ーに配信することができます。**Satellite Server** は、コンテナー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** コンピュート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ーを開始し、コンテナー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できます。

コンテンツをインポートしてコンテナー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ーを設定し、コンテンツビューにこれらのリポジトリーを含めます。

10.1. RED HAT CONTAINER CATALOG からのコンテナー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ー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ー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ーを追加します。
3. レジストリーのリポジトリーと同期します。

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** - リポジトリーの内部 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`** に割り当てることができます。
- **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。

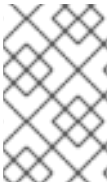
カスタム製品を作成したら、製品のリポジトリー画面が表示されます。**リポジトリーの作成** をクリックし、新しいリポジトリー用のフォームを表示します。以下の詳細情報を入力します。

- **名前** - リポジトリーの簡単な名前。**`RHEL7`** と入力します。
- **ラベル** - リポジトリーの内部 ID。**`RHEL7`** と入力します。
- **タイプ** - リポジトリーのタイプ。**`docker`** を選択すると新たなフィールドが表示されます。
- **URL** - ソースとして使用するレジストリーの 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ー名** - レジストリー上のリポジトリー名。例えば、**`rhel7`** と入力します。

注記

https://access.redhat.com/containers/ で他のリポジトリーを検索、表示することもできます。

保存 をクリックすると、新規リポジトリーが記載された製品のリポジ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。この一覧から必要に応じて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することができます。**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 **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"
```

コンテナーイメージ用のリポジトリーを作成します。

```
# 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"
```

次にリポジトリーを同期します。

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

これでコンテナー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 からのコンテナーイメージのインポート

Red Hat Container Catalog からコンテナーイメージをインポートする他に、Red Hat Satellite 6 では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. 製品にコンテナーイメージ用のリポジトリーを追加し、それを外部レジストリー上にあるリポジトリーにリンクします。
3. レジストリーのリポジトリーと同期します。

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

Web UI をご利用の場合

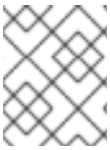
コンテンツ > 製品 に移動し、**新製品** をクリックします。新しい製品のフォームが表示されます。以下の詳細情報を入力します。

- **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** に割り当てることができます。
- **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したら、製品のリポジトリー画面が表示されます。**リポジトリーの作成** をクリックし、新しいリポジ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。

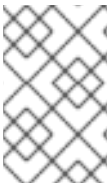
- **名前** - リポジトリーの簡単な名前。**Fedora Docker Images** と入力します。
- **ラベル** - リポジトリーの内部 ID。Red Hat Satellite 6 では、 **名前** に入力した内容に基づいてこのフィールドに値が自動的に入力されます。
- **タイプ** - リポジトリーのタイプ。**docker** を選択すると新たなフィールドが表示されます。
- **URL** - ソースとして使用するレジストリーの URL。**https://registry.hub.docker.com/** と入力します。
- **アップストリームリポジトリ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



注記

https://hub.docker.com/explore/ で他のリポジトリーを検索、表示することもできます。

保存 をクリックすると、新規リポジトリーが記載された製品のリポジ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。この一覧から必要に応じて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することができます。**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"
```

コンテナーイメージ用のリポジトリーを作成します。

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

次にリポジトリーを同期します。

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

これでコンテナー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ーイメージの管理

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にわたってコンテナー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** のフォームが表示されます。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ーが作成されます。**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ーリポジトリーを選択します。**リポジトリーの追加** をクリックします。これでこのリポジトリーからコンテナー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 が新規バージョン (バージョン 1) についての詳細を提供し、**説明** にこのバージョンについての説明が入力できます。ここには、新規コンテンツビューの変更を記録しておくことができます。**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に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

リポジトリー ID の一覧を取得します。

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

この例では、コンテナー **rhel7** のリポジトリーは ID に **8** を使用します。コンテンツビューを作成して、リポジトリーを追加します。

```
# 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ーイメージの管理

コンテナー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、コンテナー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ついての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. 章の概要

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にわたるコンテナーイメージの管理方法など、**Red Hat Satellite 6** におけるコンテナーイメージコンテンツの基本的な管理方法について説明しました。

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、起動可能で変更されない、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 リポジトリーを静的 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 コンテンツを使って、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 リポジトリーと OSTree ブランチを同期し、リポジトリーからブランチを管理するツールを提供します。

11.1. SATELLITE SERVER 上での OSTREE 管理の設定

デフォルトでは Satellite Server には OSTree 管理ツールは含まれていません。このため、**satellite-installer** を実行して必要なパッケージをインストールし、ツールを設定する必要があります。

Satellite Server に **root** ユーザーとしてログインし、以下のコマンドを実行します。

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

インストーラーが必要なパッケージ (**ostree**、**pulp-ostree-plugins**、および **tfm-rubygem-katello_ostree**) をダウンロードし、Satellite サーバーが OSTree 管理ツールを使用するように設定されます。

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 リポジトリー 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。**OSTree** タブを選択し、使用する OSTree セットまでスクロールします。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。有効なリポジトリー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** のブランチセットが含まれています。このリポジトリーを選択し、**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することができます。**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)。

数分後には、Satellite Server が **rhel7** リポジトリー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** リポジトリーを検索します。

```
# 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** リポジトリーを有効にします。この例では、このリポジトリーの 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ーを検索し、これと同期させます。この例では、リポジトリー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 の CDN から OSTree コンテンツをインポートするほかに、Red Hat Satellite 6 は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** に割り当てることができます。
- **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したら、製品のリポジトリー画面が表示されます。**リポジトリーの作成** をクリックし、新しいリポジ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** - リポジトリーの簡単な名前。**Custom OSTree** と入力します。
- **ラベル** - リポジトリーの内部 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** - リポジトリーのタイプ。**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。**http://www.example.com/rpm-ostree/** と入力します。

保存 をクリックすると、新規リポジトリーが記載された製品のリポジ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。

注記

Web UI でも同期の進捗状況を確認することができます。**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 **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 用のリポジトリーを作成します。

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

次にリポジトリーを同期します。

```
# 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にわたって 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** のフォームが表示されます。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ーが作成されます。**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 リポジトリーを選択します。**リポジトリーの追加**をクリックします。これでこのリポジトリーから OSTree コンテンツ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 が新規バージョン (バージョン 1) についての詳細を提供し、**説明** にこのバージョンについての説明が入力できます。ここには、新規コンテンツビューの変更を記録しておくことができます。**`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に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

リポジトリー ID の一覧を取得します。

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

この例では、OSTree のリポジトリーは ID に 5 を使用します。コンテンツビューを作成して、リポジトリーを追加します。

```
# 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. 章の概要

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にわたって管理する方法など、Red Hat Satellite 6 における OSTreeコンテンツの基本的な管理方法について説明しました。

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やその他のソースからの 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ー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。

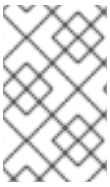
12.1. RED HAT からの ISO イメージのインポート

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ー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 リポジトリー 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、使用するイメージまでスクロールします。この例では、**Red Hat Enterprise Linux Server > Red Hat Enterprise Linux 7 Server (ISOs)** から **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ー画面が表示されます。この時点でこの製品には選択した ISO イメージのリポジトリーが含まれています。このリポジトリーを選択し、**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することができます。**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)。

数分後には、Satellite Server が選択した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

file リポジトリーの 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** リポジトリーを有効にします。

```
# 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ーを検索し、これと同期させます。この例では、リポジトリー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について説明します。この例では、**bootdisk.iso** という名前のファイルを 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のものです。

1. カスタム製品を作成します。
2. ファイルのリポジトリーを製品に追加します。
3. リポジトリー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** に割り当てることができます。
- **説明** - 製品の簡単な説明。**Custom ISO collection** と入力します。

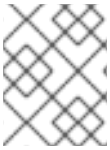
保存 をクリックします。

ISO 用のカスタム製品を作成したら、製品のリポジトリー画面が表示されます。**リポジトリーの作成** をクリックし、新しいリポジ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** - リポジトリーの簡単な名前。**Bootdisk** と入力します。
- **ラベル** - リポジトリーの内部 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** - リポジトリーのタイプ。**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** ファイルを格納している Satellite が作成した他のリポジトリー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** リポジトリーが記載された製品のリポジトリー画面に戻ります。**Bootdisk** リポジトリーをクリックします。

ファイルのアップロード セクションにスクロールし、**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"
```

リポジトリーを作成します。

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

ISO ファイルをリポジトリー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ーが完成しました。

12.3. RED HAT OVAL リポジトリーのインポート

Open Vulnerability and Assessment Language (OVAL) ファイルには、公開セキュリティーコンテンツについて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ー (https://www.redhat.com/security/data/oval/) を提供しています。Satellite Server がこのリポジトリーを同期することで、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`** に割り当てることができます。

- **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、製品のリポジトリー画面が表示されます。**リポジトリーの作成** をクリックし、新しいリポジ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** - リポジトリーの簡単な名前。**Red Hat OVAL Files** と入力します。
- **ラベル** - リポジトリーの内部 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** - リポジトリーのタイプ。**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。**https://www.redhat.com/security/data/oval/** と入力します。このリポジトリーには、Satellite Server が同期に使用する **PULP_MANIFEST** ファイルが格納されています。

保存 をクリックすると、新規リポジトリーが記載された製品のリポジトリー画面に戻ります。このリポジトリーを選択して **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 リポジトリーを作成します。

```
# 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 リポジトリーを同期します。

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

ローカルの Red Hat OVAL リポジトリーのテスト

OVAL コンテンツの同期が完了したら、リポジトリーからの OVAL ファイルでテストを実行することができます。

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、各定義が順守されているかどうかについての情報を含むOVAL レポートが生成されます。

12.4. 章の概要

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について説明しました。

次章では、コンテンツ管理プロセスについて説明します。

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、およびレポーティング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r Red Hat Satellite 6 の機能に及ぼす影響について説明します。

13.1. シナリオにおける目的の完了

本ガイドでは、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って 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 上の組織にこれをインポートします。これで Satellite 環境から RPM、キックスタートコンテンツ、ISO、および Red Hat の Content Delivery Network からのコンテナー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 の作成が基礎となります。DML は、コンテンツのすべての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。外部ソースからのコンテンツを 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、同期プランを使って DML を最新に保ちます。

異なるコンテンツタイプの管理

本ガイドでは、RPM ファイルや Puppet モジュール、コンテナー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てきました。

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ーを公開して、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っ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 のツールを使っ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。

コンテナーイメージの管理

Satellite Server がコンテナー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 および他のソースからのコンテナーイメージを同期し、コンテンツビューを使ってそれらを管理、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にわたって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可能です。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。

インストールメディアの使用

Red Hat コンテンツのインポート時に最初に注意する点は、Red Hat Enterprise Linux 7 を格納して

いるキックスタートツリーを同期したということです。Red Hat Satellite 6 は自動でこのキックスタートツリーを組織向けのインストールメディアとして追加します。Web UI を使用している場合は **ホスト > インストールメディア** に移動し、CLI を使用している場合は以下のコマンドを実行すると、このキックスタートツリーが確認できます。

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

Satellite Server には、プロビジョニングプロセス中に選択するインストールメディアを格納しているキックスタートのテンプレートセットも含まれています。

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とスニペットを使ったキックスタートファイルを作成します。特に 1 つのスニペット (**subscription_manager_registration**) が登録プロセスを制御します。Web UI では **ホスト > プロビジョニングテンプレート** に移動して **subscription_manager_registration** スニペットをクリックするとこれを確認できます。CLI では、以下のコマンドで同様の結果が得られます。

```
# hammer template dump --name subscription_manager_registration
```

スニペットは以下のようになります。

```
<% if @host.params['kt_activation_keys'] %>
# add subscription manager
yum -t -y -e 0 install subscription-manager
rpm -ivh <%= subscription_manager_configuration_url(@host) %>

echo "Registering the System"
subscription-manager register --org="<%= @host.rhsm_organization_label
%>" --name="<%= @host.name %>" --activationkey="<%=
@host.params['kt_activation_keys'] %>"

<% if @host.operatingsystem.name == "RedHat" %>
  # add the rhel rpms to install katello agent
  subscription-manager repos --enable=rhel-*-satellite-tools-*-rpms
<% end %>

echo "Installing Katello Agent"
yum -t -y -e 0 install katello-agent
chkconfig goferd on
<% end %>
```

このスニペットには 2 つの機能があります。1 つ目は、プロビジョニング中に選択したアクティベーションキーを使って、プロビジョニングされたシステムを Satellite Server に登録することです。2 つ目は、**katello-agent** をインストールすることです。Satellite Server はこれを使ってシステムと通信します。デフォルトのキックスタートプロビジョニングテンプレートのほとんどには、システム登録のためにこのスニペットが含まれています。独自のキックスタートテンプレートを作成する場合は、テンプレートに以下の行を使ってこのスニペットを参照するようにしてください。

```
<%= snippet "subscription_manager_registration" %>
```

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
- 希望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ー、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、ネットワーク情報やパーティションテーブル、カプセル、および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。

  Web UI では、**ホスト > 新規ホスト** に移動して新規システムを作成します。順番に各タブ (**ホスト**、**Puppet クラス**、**ネットワーク**、**オペレーティングシステム**、**パラメーター**、**追加情報**) でシステム向けの必須情報を入力します。これが完了して**送信** をクリックすると、プロビジョニングプロセスが開始されます。

  CLI では **`hammer host create`** で新規ホストのプロビジョニングを行います。

13.3. その他のドキュメント

以下のガイドでは、Red Hat Satellite 6 の関連情報と次のステップの例が提供されます。

Red Hat Satellite 6 ホスト設定ガイド

インフラストラクチャー管理やプロビジョニングなど、Red Hat Satellite 6 の主要機能についてのガイド
https://access.redhat.com/documentation/ja-jp/red_hat_satellite/6.2/html/host_configuration_guide/

Red Hat Satellite 6 Provisioning Guide

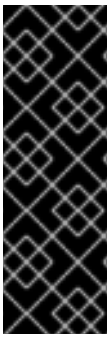
Red Hat Satellite Server から物理ホストと仮想ホストのプロビジョニングを実行するためのガイド
https://access.redhat.com/documentation/en/red-hat-satellite/6.2/paged/provisioning-guide/

Red Hat Satellite 6 Puppet Guide

独自の Puppet モジュールを構築し、これを Red Hat Satellite 6 にインポートするためのガイド
https://access.redhat.com/documentation/en/red-hat-satellite/6.2/paged/puppet-guide/

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について説明します。



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 に関する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。**/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という、セキュリティー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ー更新、エラータおよびパッケージを提供することができます。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、例えば Red Hat Enterprise Linux 6 Server (x86_64)(2015-03-12) 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 すべてをコピーします。この例では **/root/isos** を使います。
5. Satellite 上で httpd を使って共有され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/isoe
# 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 から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 としてポイント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スクロールします。**Red Hat CDN URL** の編集アイコンをクリックし、URL を 新規に作成されたディレクトリーのある Satellite ホスト名に変更します。例を示します。
    `http://server.example.com/pub/sat-import/`

13. **保存** クリックしてから、「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。

お使いの環境が非接続から接続に変更された場合は、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。

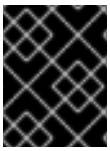
4. Red Hat プロバイダーの詳細までスクロールします。**Red Hat CDN URL** の編集アイコンをクリックし、URL を Red Hat CDN URL に変更します。
    `https://cdn.redhat.com`

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直接 Red Hat カスタマーポータルから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可能です。この同期が完了すると、ネットワーク接続から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 **バックグラウンド** 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 リポジトリーのコンテンツ ISO カラの初回同期について説明します。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、ブラウザー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 Satellite コンテンツ ISO すべてを一時的に保存するための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
```

5. 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
```

6. 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。

7. 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
```

8. 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

1. ディレクトリーの所有者と 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/
```

2. **/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は **repodata** ディレクトリーを格納している必要があります。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。

3. 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。

4. **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
```

状態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ー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ーが読み込まれます。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ーを削除して、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 6.2 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。

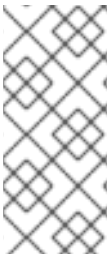
ISS は以下の 2 つのシナリオに対処します。

- 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。例えば、IT 部門が検証したコンテンツビュー (CV) から **yum** コンテンツをダウンストリーム Satellite に伝達する場合。

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 Red Hat アーキテクチャーガイド を参照してください。

Satellite 6.2 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。エクスポートされたものは、別の Satellite Server にインポート可能になります。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 6.2 では、エクスポートとインポート機能はすべてコマンドラインで実行されます。



注記

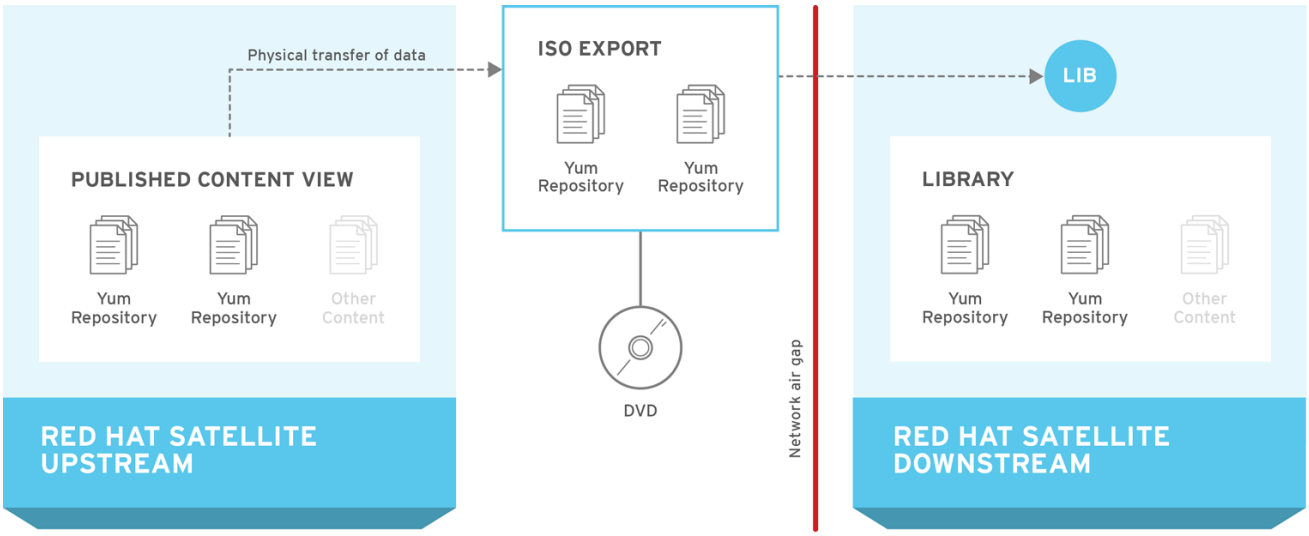
エクスポートされるのは RPM、キックスタート、および ISO ファイルのみです。パッケージフィルターなどのコンテンツビューのメタデータはエクスポートされません。Satellite 6.2 では、Puppet、Docker、または OSTree コンテンツのエクスポートはサポートされていません。インポートは通常のリポジトリー同期で発生し、常にライブラリー環境に届けられます。

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。セキュリティー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 6.2 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って作成する必要があるので、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。例えば以下のものです。

- 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 6.2 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、1 つ大きな違いがあります。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することができ (例えば 4.7 GB)、かつそれよりお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、それらを連結し、その単一の大きな 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 パラメーターを使うと、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 の設定方法について説明します。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ので、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** 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しないように SELinux とその他のパーミッションも設定されています。

よく使用され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。エクスポートされるリポジトリーと 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」ではストレージ要件を推定する方法についての情報があります。エクスポートプロセス中には、**/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中には、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

コマンドラインでエクスポート先を変更するには、以下のフォーマットで **hammer** コマンドを使用します。

```
# hammer settings set \
--name pulp_export_destination \
--value your-export-directory
```

例えば、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. **管理 > 設定** に移動して、**Katello** タブをクリックします。

2. **名前** のコラムで **pulp_export_destination** 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ーに適用されます。または、各リポジトリー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す。

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、以下のコマンドを使用します。

```
# hammer settings set \
--name default_download_policy \
--value immediate
```

特定のリポジトリーのポリシーを変更する場合は、組織のリポジトリーを以下のコマンドで一覧表示できます。

```
# hammer repository list \
--organization-label <organization-label>
```

既存のリポジトリー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. **Katello** 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ー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. **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. **リポジトリー** タブで必要なリポジトリーをクリックし、**ダウンロードポリシー**フィールドの編集アイコンをクリックします。

3. ドロップダウンリストから **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について説明します。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、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. リポジトリーのエクスポート

アップストリームサーバーからのコンテンツのエクスポートには **hammer repository export** コマンドを使用します。このコマンドは、**pulp_export_destination** で指定されているディレクトリー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ー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、ディレクトリーではなく ISO ファイルにエクスポートすることができます。例を示します。

```
# 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 (ディレクトリー) 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はありません。

D.6.2. コンテンツビューバージョン (CVV) のディレクトリーへのエクスポート

リポジトリーはすべて、コンテンツビューバージョンでディレクトリーにエクスポートすることができます。つまり、特定の CV に要件に合うようなラベル付けをすることで (例えば "Repos to Export")、整理が容易になります。これはすべてが単一のディレクトリー内にあるためです。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。

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** コマンドを使ってエクスポートする CVV を判断します。例を示します。

```
$ hammer content-view version list \
--organization "Organization_Name"
---|----------------------|---------|----------------------
ID | NAME                 | VERSION | LIFECYCLE ENVIRONMENTS
---|----------------------|---------|----------------------
14 | cdn.megacorp.com 1.0 | 1.0     | Library, Production
---|----------------------|---------|----------------------
```

CVV をエクスポートするには以下を実行します。

1. **hammer content-view version export** コマンドで CV をエクスポートします。

```
# hammer content-view version export --id 14
```

エクスポートディレクトリーにリンクを追加すると、サブスクリプション URL での使用が容易になります。例を示します。

```
$ tree megacorp-cdn_megacorp_com-v1.0
megacorp-cdn_megacorp_com-v1.0
|-- megacorp
    |-- content_views
    |   |-- cdn_megacorp_com
    |   |   |-- 1.0
```

バージョンディレクトリーへのシンボリックリンクを作成し、そのリンクを使用します。

```
$ ln -s megacorp-cdn_megacorp_com-v1.0/megacorp/ \
content_views/cdn_megacorp_com/1.0 cdn-latest
```

サブスクリプション CDN URL がエクスポート元をポイントするように設定するには、シンボリックリンクを使用します。

```
$ hammer organization update \
--name "Mega Subsidiary" \
--redhat-repository-url \
http://megacorp.com/pub/cdn-latest

Organization updated
```

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ー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。これは、特定の日時以降に 1 つ以上の同期イベントごとになされたローカルのリポジトリー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ーを作成するには、**hammer repository export** コマンドで **--since** オプションを使用します。例を示します。

```
# 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 Satellite Server で同期された時間です。例えば、Red Hat が月曜日に RPM をリポジトリー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ーの同期はできず、火曜日の日付を使って水曜日分の更新のみが得られます。

変更をリポジトリーに追加するほかに、この機能は 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、**--incremental** オプションを **hammer repository synchronize** コマンドで使用するようにしてください。このオプションを使用せず、リポジトリーの "Mirror on Sync" 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リカバリーするには、完全なエクスポートとそこからの同期という時間のかかるものになります。

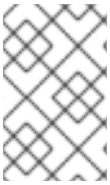
D.7. コンテンツのインポート

Red Hat Satellite 6.2 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. リポジトリーのインポート

この方法は、ダウンストリームサーバーで **hammer repository sync** コマンドを使用するもので、カスタムまたは増分コンテンツをインポートする場合に必要になります。



注記

ダウンストリームサーバーには、実際には "import" というコマンドはありません。プロセスは通常のリポジトリー同期で、アップストリームサーバーからエクスポートされたコンテンツをソースとして使用します。

ダウンストリームサーバーにコンテンツをインポートする前に、何がエクスポートされたかを確認します。アップストリームサーバーで **hammer repository-set list** コマンドを実行し、エクスポートされたものを確認します。例を示します。

```
$ hammer repository-set list \
--organization "Mega Subsidiary" \
--product "Red Hat Enterprise Linux Server" \
| grep kickstart

1952 | kickstart | Red Hat Enterprise Linux 6 Server (Kickstart)
1951 | kickstart | Red Hat Enterprise Linux 5 Server (Kickstart)
2455 | kickstart | Red Hat Enterprise Linux 7 Server (Kickstart)
```

リポジトリーの有効化

1. ダウンストリームで **hammer repository-set enable** コマンドを使ってエクスポートファイルからのリポジトリーを有効にします。例を示します。

```
$ hammer repository-set enable \
--organization "Mega Subsidiary" \
--product "Red Hat Enterprise Linux Server" \
--basearch x86_64 --releasever 7.2 --id 2455
```

リポジトリーの同期

1. 該当組織のリポジトリーを一覧表示して、ID を確認します。

```
$ hammer repository list --organization "Mega Subsidiary"
---|-------------------------------------------------------|
ID | NAME                                                  |
44 | Red Hat Enterprise Linux 7 Server Kickstart x86_64 7.2 |
```

2. **hammer repository synchronize** コマンドを使って必要なリポジトリーを同期します。例を示します。

```
$ hammer repository synchronize --id 44
```

増分エクスポートから同期する際には、**--incremental** オプションを使用してください。

D.7.2. Red Hat リポジトリー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 リポジトリー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。
- エクスポートされたコンテンツビューをある場所またはメディアにコピーして、ダウンストリームシステムから利用可能になっていること。

コンテンツビューのインポート

1. エクスポートされたコンテンツビューのある場所もしくはメディアをダウンストリームの Satellite Server にマウントします。
2. データを HTTPS ではなく HTTP で利用可能にします。例えば **`/var/www/html/pub/export/`** にコピーします。
3. 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。
4. **マニフェストの管理** をクリックします。
5. **アクション** タブでアドレスフィールドを選択するか、**Red Hat CDN URL** の右側にある編集アイコンをクリックします。
6. アドレスをエクスポートされたコンテンツビュー内のディレクトリーの場所に設定します。例えば、エクスポートが **`/export/Default_Organization-Test_Product-One/Default_Organization/Library/`** を含んでいる場合、URL を **`http://localhost/pub/export/Default_Organization-Test_Product-One/Default_Organization/Library`** と設定します。