Red Hat Satellite 6.15

コンテンツの管理

Red Hat およびカスタムソースからコンテンツをインポートし、ライフサイクル環境全体でアプリケーションのライフサイクルを管理し、コンテンツビューを使用してコンテンツをフィルタリングし、Satellite Server 間でコンテンツを同期します。

Last Updated: 2024-07-26

Red Hat Satellite 6.15 コンテンツの管理

Red Hat およびカスタムソースからコンテンツをインポートし、ライフサイクル環境全体でアプリケーションのライフサイクルを管理し、コンテンツビューを使用してコンテンツをフィルタリングし、Satellite Server 間でコンテンツを同期します。

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

法律上の通知

概要

本書を使用して、Satellite 6 のコンテンツを理解し、管理します。コンテンツの例には、RPM ファイルおよび ISO イメージが含まれます。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このコンテンツを管理します。本書では、それぞれの組織に合わせたアプリケーションライフサイクルとライフサイクル環境内でホストの状態に合致するコンテンツビューの作成方法を説明します。このようなコンテンツビューは、最終的に Red Hat Satellite 6 環境でホストのプロビジョニングおよび更新の基盤となります。

目次

RED HAT ドキュメントへのフィードバック (英語のみ)

Red Hat ドキュメントに関するご意見やご感想をお寄せください。また、改善点があればお知らせください。

フィードバックを提供するには、Red Hat Jira の **Create Issue** フォームを使用します。Jira の問題は Red Hat Satellite Jira プロジェクトに作成され、その進捗状況を追跡できます。

前提条件

- Red Hat アカウント が登録されている。

手順

1. **Create Issue** にアクセスします。Jira でログインエラーが表示された場合は、フォームにリダイレクトされた後、ログインして続行します。
2. **Summary** フィールドと **Description** フィールドに入力します。**Description** フィールドに、ドキュメントの URL、章またはセクション番号、および問題の詳しい説明を入力します。フォーム内の他のフィールドは変更しないでください。
3. **Create** をクリックします。

第1章 コンテンツ管理の概要

Satellite のコンテキストでは、**コンテンツは**、システムにインストールされているソフトウェアとして定義されます。これには、ベースオペレーティングシステム、ミドルウェアサービス、エンドユーザーアプリケーションが含まれますが、これらに限定されません。Red Hat Satellite では、ソフトウェアライフサイクルの各段階で、Red Hat Enterprise Linux システムのさまざまな種類のコンテンツを管理できます。

Red Hat Satellite は、以下の内容を管理します。

サブスクリプションの管理

これにより、組織で Red Hat サブスクリプション情報を管理できます。

コンテンツ管理

これにより、組織は Red Hat コンテンツを保存し、さまざまな方法で整理する手段を提供します。

1.1. RED HAT SATELLITE のコンテンツタイプ

Red Hat Satellite を使用すると、さまざまなコンテンツタイプをインポートして管理できます。

たとえば、Satellite は次のコンテンツタイプをサポートしています。

RPM パッケージ

RPM パッケージを、Red Hat サブスクリプションに関連するリポジトリーからインポートします。Satellite Server は、Red Hat コンテンツ配信ネットワークから RPM パッケージをダウンロードし、ローカルに保存します。これらのリポジトリーとその RPM パッケージを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

キックスタートツリーをインポートしてホストをプロビジョニングします。新しいシステムは、ネットワーク経由でこれらのキックスタートツリーにアクセスし、インストールのベースコンテンツとして使用します。Red Hat Satellite には、事前定義されたキックスタートテンプレー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こともできます。

ISO および KVM イメージ

インストールおよびプロビジョニングのメディアをダウンロードして管理します。たとえば、Satellite は、特定の Red Hat Enterprise Linux および Red Hat 以外のオペレーティングシステムの ISO イメージおよびゲストイメージをダウンロード、保存、および管理します。

カスタムのファイルタイプ

SSL 証明書、ISO イメージ、OVAL ファイルなど、必要なあらゆる種類のファイルのカスタムコンテンツを管理します。

第2章 RED HAT サブスクリプションの管理

Red Hat Satellite は、Red Hat 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できます。Satellite では、対応するリポジトリーからコンテンツを検索し、アクセスし、ダウンロードするために Red Hat サブスクリプションマニフェストが必要です。Red Hat サブスクリプションマニフェストには、Satellite Server の各組織へのサブスクリプション割り当てを含める必要があります。サブスクリプション情報はすべて、Red Hat カスタマーポータルのアカウントで確認できます。

この章のタスクを完了するには、カスタマーポータルで Red Hat サブスクリプションマニフェストを作成する必要があります。

エンタイトルメントベースのサブスクリプションモデルは非推奨であり、今後のリリースで削除されることに注意してください。Red Hat では、代わりに Simple Content Access のアクセスベースのサブスクリプションモデルを使用することを推奨します。

カスタマーポータルで Red Hat サブスクリプションマニフェストを作成、管理、およびエクスポートするには、**Subscription Central** の オンライン接続されている Satellite Server に向けたマニフェストの作成および管理 を参照してください。

この章を参照して、Satellite Web UI 内で Red Hat サブスクリプションマニフェストをインポートし、管理します。

サブスクリプションの割り当てと組織

複数のサブスクリプション割り当てがある場合は、複数の組織を管理できます。Satellite では、Satellite Server で設定した組織ごとに 1 つの割り当てが必要です。この利点は、各組織が別々のサブスクリプションを維持し、それぞれ独自の Red Hat アカウントで複数の組織をサポートできることです。

未来の日付のサブスクリプション

サブスクリプション割り当てでは、未来の日付のサブスクリプションを使用できます。既存のサブスクリプションの有効期限前に、未来の日付が指定されたサブスクリプションをコンテンツホストに追加する場合は、リポジトリーへのアクセスが中断されず、そのまま利用できます。

現在のサブスクリプションの有効期限が切れる前に、未来の日付のサブスクリプションをコンテンツホストに手動でアタッチします。自動アタッチ方法には依存せず、別の目的で設計されており、機能しない可能性があります。詳細は、「コンテンツホストへの Red Hat サブスクリプションのアタッチ」 を参照してください。

将来の日付のサブスクリプションがアクティブになったら、マニフェストを更新してリポジトリーのコンテンツを同期する必要があります。詳細は、「Red Hat サブスクリプションマニフェストの更新およびリフレッシュ」 を参照してください。

関連情報

- **オフラインネットワーク環境での Satellite Server のインストール** の カスタム CDN からコンテンツを使用するための Satellite Server の設定

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

以下の手順を使用して、Red Hat サブスクリプションマニフェストを Satellite Server にインポートします。

注記

Simple Content Access (SCA) は、マニフェストではなく組織で設定されます。マニフェストをインポートしても、組織の Simple Content Access のステータスは変更されません。

前提条件

- Red Hat サブスクリプションマニフェストがある。
  - Satellite がオンライン接続されている場合は、Red Hat Hybrid Cloud Console を使用してマニフェストを作成し、エクスポートします。詳細は、**Subscription Central** の オンライン接続されている Satellite Server に向けたマニフェストの作成および管理 を参照してください。
  - Satellite がオンライン接続されていない場合は、Red Hat カスタマーポータルを使用してマニフェストを作成し、エクスポートします。詳細は、**Subscription Central** の 接続されていない Satellite Server のマニフェストの使用 を参照してください。

手順

1. Satellite Web UI で、コンテキストが、使用する組織に設定されていることを確認します。
2. Satellite Web UI で、**Content** > **Subscriptions** に移動し、**Manage Manifest** をクリックします。
3. **Manage Manifest** ウィンドウで、**Choose File** をクリックします。
4. Red Hat サブスクリプションマニフェストファイルが保存されている場所に移動し、**Open** をクリックします。

CLI 手順

1. Red Hat サブスクリプションマニフェストファイルをローカルマシンから Satellite Server にコピーします。

```
$ scp ~/manifest_file.zip root@satellite.example.com:~/.
```

2. Satellite Server に **root** ユーザーとしてログインし、Red Hat サブスクリプションマニフェストファイルをインポートします。

```
# hammer subscription upload \
--file ~/manifest_file.zip \
--organization "My_Organization"
```

リポジトリーを有効にし、Red Hat コンテンツをインポートすることができるようになりました。詳細は、**コンテンツの管理** の コンテンツのインポート を参照してください。

2.2. RED HAT サブスクリプションの特定

Red Hat サブスクリプションマニフェストを Satellite Server にインポートすると、マニフェストからのサブスクリプションがサブスクリプションウィンドウにリスト表示されます。サブスクリプションが大量にある場合には、結果をフィルタリングして、特定のサブスクリプションを見つけることができます。

前提条件

- Red Hat サブスクリプションマニフェストファイルを Satellite Server にインポートしている。詳細は、「Red Hat サブスクリプションマニフェストの Satellite Server へのインポート」 を参照してください。

手順

1. Satellite Web UI で、コンテキストが、使用する組織に設定されていることを確認します。
2. Satellite Web UI で、**Content** > **Subscriptions** に移動します。
3. サブスクリプションウィンドウで **検索** フィールドをクリックし、検索条件のリストを表示して検索クエリーをビルドします。
4. 検索条件を選択して、他のオプションを表示します。
5. 検索クエリーをビルドしたら、検索アイコンをクリックします。

たとえば、**検索** フィールドにカーソルを置き、**期限切れ**を選択して、スペースキーを押すと、別のリストが表示され、>、<、または = を配置するオプションが表示されます。> を選択してスペースキーを押すと、自動オプションの別のリストが表示されます。独自の条件を入力することも可能です。

2.3. サブスクリプション割り当てへの RED HAT サブスクリプションの追加

以下の手順を使用して、Red Hat Satellite Web UI でサブスクリプション割り当てにサブスクリプションを追加します。

前提条件

- Red Hat サブスクリプションマニフェストファイルを Satellite Server にインポートしている。詳細は、「Red Hat サブスクリプションマニフェストの Satellite Server へのインポート」 を参照してください。

手順

1. Satellite Web UI で、コンテキストが、使用する組織に設定されていることを確認します。
2. Satellite Web UI で、**Content** > **Subscriptions** に移動します。
3. サブスクリプションウィンドウで、**サブスクリプションの追加** をクリックします。
4. 追加するサブスクリプションの各行で、**Quantity to Allocate (割り当て可能)** 列に数量を入力します。
5. **Submit** をクリックします。

2.4. サブスクリプション割り当てからの RED HAT サブスクリプションの削除

以下の手順を使用して、Red Hat Satellite Web UI でサブスクリプション割り当てからサブスクリプションを削除します。

注記

マニフェストを削除することはできません。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、コンテンツホストのエンタイトルメントがすべて解除されます。

前提条件

- Red Hat サブスクリプションマニフェストファイルを Satellite Server にインポートしている。詳細は、「Red Hat サブスクリプションマニフェストの Satellite Server へのインポート」 を参照してください。

手順

1. Satellite Web UI で、コンテキストが、使用する組織に設定されていることを確認します。
2. Satellite Web UI で、**Content** > **Subscriptions** に移動します。
3. 削除するサブスクリプションの各行で、該当のチェックボックスを選択します。
4. **Delete** をクリックしてから、削除を確定します。

2.5. RED HAT サブスクリプションマニフェストの更新およびリフレッシュ

サブスクリプションの割り当てを変更する場合には、マニフェストをリフレッシュして変更を反映する必要があります。たとえば、以下のいずれかを行った場合はマニフェストをリフレッシュする必要があります。

- サブスクリプションの更新
- サブスクリプションの量の調整
- 追加のサブスクリプションの購入

Satellite Web UI でマニフェストを直接更新できます。または、変更を含む更新済みのマニフェストをインポートできます。

手順

1. Satellite Web UI で、コンテキストが、使用する組織に設定されていることを確認します。
2. Satellite Web UI で、**Content** > **Subscriptions** に移動します。
3. サブスクリプションウィンドウで、**マニフェストの管理** をクリックします。
4. マニフェストの管理ウィンドウで、**リフレッシュ** をクリックします。

2.6. コンテンツホストへの RED HAT サブスクリプションのアタッチ

プロビジョニングプロセス中は、コンテンツホストにサブスクリプションをアタッチする主要な方法として、アクティベーションキーを使用します。ただし、アクティベーションキーは既存のホストを更新できません。新規または追加のサブスクリプション (たとえば未来の日付が指定されたサブスクリプション) を 1 台のホストにアタッチする必要がある場合は、以下の手順を使用します。

重要

この手順は、Satellite で Simple Content Access (SCA) が無効になっている場合にのみ有効です。SCA を有効にすると、ホストにサブスクリプションを接続する必要がなくなります。新しく作成された組織では、SCA がデフォルトで有効になっていることに注意してください。SCA の詳細は、Simple Content Access を参照してください。

複数ホストの更新に関する詳細情報は、「複数のホストでの Red Hat サブスクリプションの更新」 を参照してください。

アクティベーションキーに関する詳細情報は、*9章 アクティベーションキーの管理* を参照してください。

Satellite サブスクリプション

Satellite では、管理する Red Hat Enterprise Linux ホストごとに、以前は Red Hat Enterprise Linux Smart Management と呼ばれていた Red Hat Enterprise Linux Satellite サブスクリプションが必要です。

ただし、Satellite サブスクリプションを各コンテンツホストにアタッチする必要はありません。Satellite サブスクリプションは製品証明書に関連付けられていないため、Satellite のコンテンツホストに自動的にアタッチできません。Satellite サブスクリプションをコンテンツホストに追加しても、コンテンツやリポジトリーへのアクセスは提供されません。必要に応じて、独自の記録または追跡の目的で、Satellite サブスクリプションをマニフェストに追加できます。

前提条件

- Red Hat サブスクリプションマニフェストファイルを Satellite Server にインポートしている。詳細は、「Red Hat サブスクリプションマニフェストの Satellite Server へのインポート」 を参照してください。

手順

1. Satellite Web UI で、コンテキストが、使用する組織に設定されていることを確認します。
2. Satellite Web UI で、**Hosts** > **Content Hosts** に移動します。
3. 変更するサブスクリプションのコンテンツホストの行ごとに、該当するチェックボックスを選択します。
4. **Select Action** リストから **Manage Subscriptions** を選択します。
5. 必要に応じて、**検索** フィールドにキーと値を入力し、表示するサブスクリプションを絞り込みます。
6. 追加または削除したいサブスクリプションの左側にあるチェックボックスを選択し、必要に応じて **Add Selected** または **Remove Selected** をクリックします。
7. **完了** をクリックして変更を保存します。

CLI 手順

1. Satellite Server で、利用可能なすべてのサブスクリプションをリスト表示します。

```
# hammer subscription list \
--organization-id My_Organization_ID
```

2. ホストにサブスクリプションをアタッチします。

```
# hammer host subscription attach \
--host My_Host_Name \
--subscription-id My_Subscription_ID
```

2.7. 複数のホストでの RED HAT サブスクリプションの更新

インストール後の変更を複数のコンテンツホストに同時に行うには、この手順を使用します。

重要

この手順は、Satellite で Simple Content Access (SCA) が無効になっている場合にのみ有効です。SCA を有効にすると、ホストにサブスクリプションを接続する必要がなくなります。新しく作成された組織では、SCA がデフォルトで有効になっていることに注意してください。SCA の詳細は、Simple Content Access を参照してください。

手順

1. Satellite Web UI で、コンテキストが、使用する組織に設定されていることを確認します。
2. Satellite Web UI で、**Hosts** > **Content Hosts** に移動します。
3. 変更するサブスクリプションのコンテンツホストの行ごとに、該当するチェックボックスを選択します。
4. **Select Action** リストから **Manage Subscriptions** を選択します。
5. 必要に応じて、**検索** フィールドにキーと値を入力し、表示するサブスクリプションを絞り込みます。
6. 追加または削除するサブスクリプションの左側にあるチェックボックスを選択し、必要に応じて **Add Selected** または **Remove Selected** をクリックします。
7. **完了** をクリックして変更を保存します。

2.8. コンテンツ配信ネットワークの構造

Red Hat コンテンツ配信ネットワーク (CDN) (**cdn.redhat.com**) は、地理的に分散された一連の静的 Web サーバーで、システムが使用することを目的としたコンテンツおよびエラータが含まれます。このコンテンツには、Subscription Manager を使用して登録されたシステムまたは Satellite Web UI を介して直接アクセスできます。この CDN のアクセス可能なサブネットは、Red Hat サブスクリプション管理 または Satellite Server を使用して、システムに割り当てられているサブスクリプションから設定します。

Red Hat コンテンツ配信ネットワークは、有効なユーザーのみがアクセスできるように、X.509 証明書認証によって保護されます。

CDN のディレクトリー構造

```
$ tree -d -L 11
└── content 1
    ├── beta 2
    │   └── rhel 3
```

```
│       └── server 4
│           └── 7 5
│               └── x86_64 6
│                   └── sat-tools 7
└── dist
    └── rhel
        └── server
            └── 7
                ├── 7.2
                │   └── x86_64
                │       └── kickstart
                └── 7Server
                    └── x86_64
                        └── os
```

1. **content** ディレクトリー。
2. コンテンツのライフサイクルに関するディレクトリー。**beta** (ベータコードの場合)、**dist** (実稼働環境の場合)、**eus** (延長更新サポートの場合) などが一般的なディレクトリーとなります。
3. 製品名に関するディレクトリー。通常は、Red Hat Enterprise Linux を表す **rhel**。
4. 製品の種類に関するディレクトリー。Red Hat Enterprise Linux の場合は **server**、**workstation**、**computenode** などになります。
5. リリースバージョンに関するディレクトリー (例: **7**、**7.2**、**7Server**)。
6. ベースアーキテクチャーに関するディレクトリー (例: **i386** または **x86_64**)。
7. リポジトリー名に関するディレクトリー (例: **sat-tools**、**kickstart**、**rhscl**)。一部のコンポーネントには、変更可能な追加サブディレクトリーが含まれます。

このディレクトリー構造は、Red Hat Subscription Manifest でも使用されます。

第3章 代替コンテンツソースの管理

代替コンテンツソースは、同期時にコンテンツをダウンロードする代替パスを定義します。コンテンツ自体は、代わりのコンテンツソースからダウンロードされますが、メタデータは設定に応じて Satellite Server またはアップストリーム URL からダウンロードされます。コンテンツがローカルファイルシステムまたは近くのネットワークにある場合は、代替コンテンツソースを使用して同期を高速化できます。Satellite Server および Capsule の代替コンテンツソースを設定できます。作成後または変更後は、代替コンテンツソースを更新する必要があります。毎週の cron ジョブにより、すべての代替コンテンツソースが更新されます。また、Satellite Web UI または Hammer CLI を使用して、代替コンテンツソースを手動で更新することもできます。Satellite Server に関連付けられた代替コンテンツソース、または複数の組織に接続された Capsule Server は、すべての組織に影響します。

代替コンテンツソースには 3 つのタイプがあります。

Custom

カスタムの代替コンテンツソースは、ネットワークまたはファイルシステムにある任意のアップストリームリポジトリーからコンテンツをダウンロードします。

Simplified

簡易代替コンテンツソースは、選択した製品の Satellite Server からアップストリームリポジトリー情報をコピーします。簡易代替コンテンツソースは、Capsule からアップストリームリポジトリーへの接続が Satellite Server からの接続よりも高速な場合に、最適です。簡易代替コンテンツソースを作成するときに Red Hat 製品を選択すると、コンテンツが Red Hat CDN から Capsule にダウンロードされます。

RHUI

RHUI の代替コンテンツソースは、Red Hat Update Infrastructure サーバーからコンテンツをダウンロードします。Satellite Web UI には、ネットワークパスの検索や認証情報のインポートに役立つ例が用意されています。RHUI 代替コンテンツソースは RHUI バージョン 4 以降であり、デフォルトのインストール設定を使用する必要があります。たとえば、AWS RHUI は、独自の認証要件が求められるインストールシナリオを使用するため、サポートされていません。

代替コンテンツソースのパーミッション要件

管理者以外のユーザーが代替コンテンツソースを管理するには、以下の権限が必要です。

1. **view_smart_proxies**
2. **view_content_credentials**
3. **view_organizations**
4. **view_products**

上記の権限に加えて、ユーザーが実行できるアクションに応じて、代替コンテンツソースに固有の権限を割り当てます。

1. **view_alternate_content_sources**
2. **create_alternate_content_sources**
3. **edit_alternate_content_sources**
4. **destroy_alternate_content_sources**

3.1. カスタム代替コンテンツソースの設定

前提条件

- リポジトリーで SSL 認証が必要な場合は、SSL 証明書とキーを Satellite にインポートしておく。詳細は、**コンテンツの管理** の カスタム SSL 証明書のインポート を参照してください。
- 代替コンテンツソースパスは、指定したサブパスが追加されたベース URL で構成されることに注意してください。たとえば、ベース URL が **https://server.example.com** で、サブパスが **rhel7/** および **rhel8/** の場合、**https://server.example.com/rhel7/** および **https://server.example.com/rhel8/** の両方が検索されます。

手順

1. Satellite Web UI で、**Content** > **Alternate Content Sources** に移動します。
2. **Add Source** をクリックし、**Source type** を **Custom** に設定します。
3. **Content type** を選択します。
4. **Name** フィールドに、代替コンテンツソースの名前を入力します。
5. オプション: **Description** フィールドに、ACS の説明を入力します。
6. 代替コンテンツソースを同期する Capsules を選択します。
7. 代替コンテンツソースのベース URL を入力します。
8. サブパスのコンマ区切りリストを入力します。
9. 必要に応じて、**Manual Authentication** または **Content Authentication** の認証情報を提供します。
10. SSL 検証が必要な場合は、**Verify SSL** を有効にして、SSL CA 証明書を選択します。
11. 詳細を確認し、**Add** をクリックします。
12. **Content** > **Alternate Content Sources** に移動し、新しく作成した代替コンテンツソースの横にある縦の省略記号をクリックして、**Refresh** を選択します。

CLI 手順

1. Satellite Server で以下のコマンドを入力します。

```
# hammer alternate-content-source create \
--alternate-content-source-type custom \
--base-url "https://local-repo.example.com:port" \
--name "My_ACS_Name" \
--smart-proxy-ids My_Capsule_ID
```

2. 新しく作成した代替コンテンツソースが一覧表示されるかどうかを確認します。

```
# hammer alternate-content-source list
```

3. 代替コンテンツソースを更新します。

```
# hammer alternate-content-source refresh --id My_Alternate_Content_Source_ID
```

4. 代替コンテンツソースを同期する Capsule を追加します。

```
# hammer alternate-content-source update \
--id My_Alternate_Content_Source_ID \
--smart-proxy-ids My_Capsule_ID
```

5. 代替コンテンツソースを更新します。

```
# hammer alternate-content-source refresh --id My_Alternate_Content_Source_ID
```

3.2. 簡易代替コンテンツソースの設定

手順

1. Satellite Web UI で、**Content** > **Alternate Content Sources** に移動します。
2. **Add Source** をクリックし、**Source type** を**Simplified** に設定します。
3. **Content type** を選択します。
4. **Name** フィールドに、代替コンテンツソースの名前を入力します。
5. オプション: **Description** フィールドに、ACS の説明を入力します。
6. 代替コンテンツソースを同期する Capsules を選択します。
7. オプション: ACS で Capsule Server の HTTP プロキシーを使用する場合は、**Use HTTP proxies** を選択します。
8. 代替コンテンツソースを使用する製品を選択します。
9. 詳細を確認し、**Add** をクリックします。
10. **Content** > **Alternate Content Sources** に移動し、新しく作成した代替コンテンツソースの横にある垂直省略記号をクリックして、**Refresh** を選択します。

CLI 手順

1. Simplified ACS を作成します。

```
# hammer alternate-content-source create \
--alternate-content-source-type simplified \
--name My_ACS_Name \
--product-ids My_Product_ID \
--smart-proxy-ids My_Capsule_ID
```

2. 新しく作成された ACS がリストにあるかどうかを確認します。

```
# hammer alternate-content-source list
```

3. ACS を更新します。

```
# hammer alternate-content-source refresh --id My_ACS_ID
```

3.2.1. Red Hat CDN からの Capsule の直接同期

簡易代替コンテンツソースを使用して、Red Hat CDN からコンテンツを直接同期するように Capsule を設定できます。

手順

1. Satellite Web UI で、**Content** > **Alternate Content Sources** に移動します。
2. **Add Source** をクリックし、**Source type** を**Simplified** に設定します。
3. **Content type** を **Yum** に設定します。
4. **Name** フィールドに、代替コンテンツソースの名前を入力します。
5. オプション: **Description** フィールドに、代替コンテンツソースの説明を入力します。
6. Red Hat CDN から直接同期する Capsule を選択します。
7. オプション: ACS で Capsule Server の HTTP プロキシーを使用する場合は、**Use HTTP proxies** を選択します。
8. Red Hat CDN から Capsule に同期する必要がある Red Hat 製品を選択します。
9. 詳細を確認し、**Add** をクリックします。
10. **Content** > **Alternate Content Sources** に移動し、新しく作成した代替コンテンツソースの横にある垂直省略記号をクリックして、**Refresh** を選択します。

Capsule は、Satellite ではなく Red Hat CDN からコンテンツをダウンロードするようになります。

3.3. RHUI 代替コンテンツソースの設定

前提条件

- **Red Hat Update Infrastructure の設定と管理**の Red Hat Update Infrastructure Management Tool を使用したクライアント資格証明書の作成 の説明に従って、RHUA ノード上の必要なリポジトリーのクライアント資格証明書を生成します。
- クライアントエンタイトルメント証明書を Satellite にインポートします。詳細は、**コンテンツの管理** の カスタム SSL 証明書のインポート を参照してください。
- 必要なリポジトリーのサブパスのリストを取得します。RHUA サーバーで次のコマンドを実行します。

```
# rhui-manager repo info --repo_id My_Repo_ID
```

- 代替コンテンツソースパスは、指定したサブパスが追加されたベース URL で構成されることに注意してください。たとえば、ベース URL が **https://server.example.com** で、サブパスが **rhel7/** および **rhel8/** の場合、**https://server.example.com/rhel7/** および **https://server.example.com/rhel8/** の両方が検索されます。

手順

1. Satellite Web UI で、**Content** > **Alternate Content Sources** に移動します。

2. **Add Source** をクリックし、**Source type** を **RHUI** に設定します。
3. Satellite Web UI で提供されるコマンドを使用して、RHUI 証明書を生成します。目的のリポジトリーのリポジトリーラベルを指定していることを確認してください。
4. **Name** フィールドに、代替コンテンツソースの名前を入力します。
5. オプション: **Description** フィールドに、ACS の説明を入力します。
6. 代替コンテンツソースを同期する Capsules を選択します。
7. オプション: ACS で Capsule の HTTP プロキシーを使用する場合は、**Use HTTP proxies** を選択します。
8. Red Hat Update Infrastructure CDS ノードのベース URL を入力します。
9. サブパスのコンマ区切りリストを入力します。
10. 必要に応じて、**Content Credentials** を入力します。
11. SSL 検証が必要な場合は、**Verify SSL** を有効にして、SSL CA 証明書を選択します。
12. 詳細を確認し、**Add** をクリックします。
13. **Content** > **Alternate Content Sources** に移動し、新しく作成した代替コンテンツソースの横にある垂直省略記号をクリックして、**Refresh** を選択します。

CLI 手順

1. Satellite Server で以下のコマンドを入力します。

```
# hammer alternate-content-source create \
--alternate-content-source-type rhui \
--base-url "https://rhui-cds-node/pulp/content" \
--name "My_ACS_Name" \
--smart-proxy-ids My_Capsule_ID \
--ssl-client-cert-id My_SSL_Client_Certificate_ID \
--ssl-client-key-id My_SSL_Client_Key_ID \
--subpaths path/to/repo/1/,path/to/repo/2/ \
--verify-ssl 1
```

2. 新しく作成した代替コンテンツソースが一覧表示されるかどうかを確認します。

```
# hammer alternate-content-source list
```

3. 代替コンテンツソースを更新します。

```
# hammer alternate-content-source refresh --id My_Alternate_Content_Source_ID
```

4. 代替コンテンツソースを同期する Capsule を追加します。

```
# hammer alternate-content-source update \
--id My_Alternate_Content_Source_ID \
--smart-proxy-ids My_Capsule_ID
```

5. 代替コンテンツソースを更新します。

```
# hammer alternate-content-source refresh --id My_Alternate_Content_Source_ID
```

第4章 コンテンツのインポート

本章では、さまざまな種類のカスタムコンテンツを Satellite にインポートする方法を概説します。たとえば、特定のタイプのカスタムコンテンツの情報については、以下の章を使用できますが、基本的な手順は同じです。

- *12章ISO イメージの管理*
- *14章カスタムファイルタイプコンテンツの管理*

4.1. SATELLITE の製品とリポジトリー

Satellite における Red Hat コンテンツとカスタムコンテンツの両方に類似点があります。

- 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、Red Hat 製品に対するサブスクリプションと同様に、ホストがアクセスするためのサブスクリプションが必要です。Satellite は、作成するカスタム製品ごとにサブスクリプションを作成します。

Red Hat コンテンツはすでに製品に分類されています。たとえば、Red Hat Enterprise Linux Server は Satellite の **製品** です。その製品のリポジトリーは、さまざまなバージョン、アーキテクチャー、アドオンで構成されます。Red Hat リポジトリーの場合、製品はリポジトリーを有効にした後に自動的に作成されます。詳細は、「Red Hat リポジトリーの有効化」 を参照してください。

他のコンテンツは、必要に応じてカスタム製品に分類できます。たとえば、EPEL (Extra Packages for Enterprise Linux) 製品を作成し、それに "EPEL 7 x86_64" リポジトリーを追加することができます。

RPM の作成およびパッケージングの詳細は、**Red Hat Enterprise Linux RPM パッケージガイド**を参照してください。

4.2. 製品とリポジトリーのベストプラクティス

- 製品およびコンテンツビューごとに 1 つのコンテンツタイプを使用します (例: yum コンテンツのみ)。
- ファイルリポジトリーを HTTP 経由で利用できるようにします。**Protected** を true に設定すると、コンテンツをダウンロードするのに、グローバルデバッグ証明書の使用が必要になります。
- Hammer スクリプトまたは Ansible Playbook を使用して、複数の製品とリポジトリーの作成を自動化します。
- Red Hat コンテンツについては、Red Hat マニフェストを Satellite にインポートします。詳細は、*2章Red Hat サブスクリプションの管理* を参照してください。
- **Upstream URL** を使用してリポジトリーにコンテンツをアップロードしないでください。代わりに、**Upstream URL** を設定せずに、コンテンツの同期先およびアップロード先となるリポジトリーを作成します。
  すでに別のリポジトリーと同期しているリポジトリーにコンテンツをアップロードすると、ミラーリングポリシーとコンテンツタイプによってはコンテンツが上書きされる可能性があります。

4.3. カスタム SSL 証明書のインポート

外部ソースからカスタムコンテンツを同期する前に、SSL 証明書をカスタム製品にインポートする必要がある場合があります。これには、同期するアップストリームリポジトリーのクライアント証明書および鍵または CA 証明書が含まれる可能性があります。

パッケージのダウンロードに SSL 証明書とキーが必要な場合は、Satellite に追加できます。

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

手順

1. Satellite Web UI で **Content** > **Content Credentials** に移動します。Content Credentials ウィンドウで、**Create Content Credential** をクリックします。
2. **Name** フィールドに、SSL 証明書の名前を入力します。
3. **Type** のリストから、**SSL Certificate** を選択します。
4. **Content Credentials Content** フィールドに SSL 証明書を貼り付けるか、**Browse** をクリックして SSL 証明書をアップロードします。
5. **Save** をクリックします。

CLI 手順

1. SSL 証明書を Satellite Server にコピーします。

```
$ scp My_SSL_Certificate root@satellite.example.com:~/.
```

または、SSL 証明書をオンラインソースから Satellite Server にダウンロードします。

```
$ wget -P ~ http://upstream-satellite.example.com/pub/katello-server-ca.crt
```

2. SSL 証明書を Satellite にアップロードします。

```
# hammer content-credential create \
--content-type cert \
--name "My_SSL_Certificate" \
--organization "My_Organization" \
--path ~/My_SSL_Certificate
```

4.4. カスタム製品の作成

カスタム製品を作成し、リポジトリーをカスタム製品に追加します。Satellite Web UI の代わりに CLI を使用する場合は、CLI 手順 を参照してください。

手順

1. Satellite Web UI で **Content** > **Products** に移動して、**Create Product** をクリックします。
2. **Name** フィールドに、製品の名前を入力します。Satellite では、**Name** に入力した内容に基づいて **Label** フィールドに自動的に入力されます。
3. オプション: **GPG Key** の一覧から、製品の GPG キーを選択します。

4. オプション: **SSL CA Cert** の一覧から、製品の SSL CA 証明書を選択します。
5. オプション: **SSL Client Cert** の一覧から、製品の SSL クライアント証明書を選択します。
6. オプション: **SSL Client Key** の一覧から、製品の SSL クライアントキーを選択します。
7. オプション: **Sync Plan** の一覧から、既存の同期プランを選択するか、**Create Sync Plan** をクリックし、製品要件の同期プランを作成します。
8. **Description** フィールドには、製品の説明を入力します。
9. **Save** をクリックします。

CLI 手順

製品を作成するには、以下のコマンドを実行します。

```
# hammer product create \
--name "My_Product" \
--sync-plan "Example Plan" \
--description "Content from My Repositories" \
--organization "My_Organization"
```

4.5. カスタム RPM リポジトリーの追加

以下の手順を使用して、Satellite でカスタム RPM リポジトリーを追加します。Satellite Web UI の代わりに CLI を使用する場合は、CLI 手順 を参照してください。

Satellite Web UI の製品ウィンドウには、**Repo Discovery** 機能があり、URL からすべてのリポジトリーを見つけ、どのリポジトリーを自分のカスタム製品に追加するかを選択できます。たとえば、**Repo Discovery** を使用して **https://download.postgresql.org/pub/repos/yum/16/redhat/** を検索し、さまざまな Red Hat Enterprise Linux バージョンおよびアーキテクチャーのすべてのリポジトリーをリスト表示できます。これにより、1 つの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。

カスタム RPM のサポート

Red Hat は、サードパーティーのサイトから直接アップストリームの 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 に関する問題については、サードパーティーの開発者に連絡してください。

手順

1. Satellite Web UI で、**Content** > **Products** に移動し、使用する製品を選択してから **New Repository** をクリックします。
2. **Name** フィールドに、リポジトリーの名前を入力します。Satellite では、 **Name** に入力した内容に基づいて **Label** フィールドに自動的に入力されます。
3. オプション: **Description** フィールドに、新規リポジトリーの説明を入力します。
4. **タイプ** のリストから、リポジトリーのタイプとして **yum** を選択します。
5. オプション: **Restrict to Architecture** のリストから、アーキテクチャーを選択します。アーキテクチャーに関係なく、すべてのホストでリポジトリーを利用できるようにする場合は、**No restriction** を選択してください。

6. オプション: **Restrict to OS Version**の一覧から、OS バージョンを選択します。OS のバージョンに関係なく、すべてのホストでリポジトリーを利用できるようにする場合は、**No restriction** を選択してください。

7. オプション: **Upstream URL** フィールドに、ソースとして使用する外部リポジトリーの URL を入力します。Satellite は、**http://**、**https://**、および **file://** の 3 つのプロトコルをサポートしています。**file://** リポジトリーを使用している場合は、**/var/lib/pulp/sync_imports/** ディレクトリーの下に配置する必要があります。
アップストリーム URL を入力しない場合は、パッケージを手動でアップロードできます。

8. オプション: **Ignore SRPMs** のチェックボックスをオンにして、ソース RPM パッケージを Satellite との同期から除外します。

9. オプション: **Treeinfo file should have INI format** というエラーが表示された場合は、**Ignore treeinfo** チェックボックスをオンにします。**treeinfo** ファイルをスキップすると、キックスタートに関連するすべてのファイルがリポジトリーから削除されます。

10. アップストリームのリポジトリーの SSL 証明書が信頼できる CA によって署名されていることを確認する場合は、**Verify SSL** のチェックボックスを選択します。

11. オプション: 認証に必要な場合は、**Upstream Username** フィールドに、アップストリームリポジトリーのユーザー名を入力します。リポジトリーに認証が必要ない場合は、このフィールドをクリアします。

12. オプション: **Upstream Password** フィールドに、アップストリームリポジトリーに対応するパスワードを入力します。リポジトリーに認証が必要ない場合は、このフィールドをクリアします。

13. オプション: **Upstream Authentication Token** フィールドに、認証用のアップストリームリポジトリーユーザーのトークンを指定します。リポジトリーに認証が必要ない場合は、このフィールドを空欄のままにします。

14. **Download Policy** リストから、Satellite Server が実行する同期の種類を選択します。詳細は、「ダウンロードポリシーの概要」 を参照してください。

15. **ミラーリングポリシー** リストから、Satellite Server が実行するコンテンツの同期のタイプを選択します。詳細は、「ミラーリングポリシーの概要」 を参照してください。

16. オプション: **Retain package versions**フィールドに、パッケージごとに保持するバージョンの数を入力します。

17. **HTTP Proxy Policy** フィールドで、HTTP プロキシーを選択します。

18. **Checksum**の一覧から、リポジトリーのチェックサムタイプを選択します。

19. オプション: **Unprotected** のチェックボックスをオフにして、このリポジトリーにアクセスするためにサブスクリプションエンタイトルメント証明書を要求することができます。デフォルトでは、リポジトリーは HTTP 経由で公開されます。

20. オプション: **GPG Key** の一覧から、製品の GPG キーを選択します。

21. オプション: **SSL CA Cert** フィールドで、リポジトリーの SSL CA 証明書を選択します。

22. オプション: **SSL Client cert** フィールドで、リポジトリーの SSL Client Certificate を選択します。

23. オプション: **SSL Client Key** フィールドで、リポジトリーの SSL Client Key を選択します。

24. **Save** をクリックして、リポジトリーを作成します。

CLI 手順

1. 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 \
--arch "My_Architecture" \
--content-type "yum" \
--gpg-key-id My_GPG_Key_ID \
--name "My_Repository" \
--organization "My_Organization" \
--os-version "My_OS_Version" \
--product "My_Product" \
--publish-via-http true \
--url My_Upstream_URL
```

続行して、リポジトリーを同期します。

4.6. RED HAT リポジトリーの有効化

外部ネットワークアクセスで HTTP プロキシーの使用が必要な場合は、サーバー用にデフォルトの HTTP プロキシーを設定します。詳細は、デフォルトの HTTP プロキシーの Satellite への追加 を参照してください。

同期するリポジトリーを選択するには、まずリポジトリーが含まれる製品を特定し、適切なリリースバージョンおよび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する必要があります。

Red Hat Enterprise Linux 8 ホストの場合:

Red Hat Enterprise Linux 8 ホストをプロビジョニングするには、**Red Hat Enterprise Linux 8 for x86_64 - AppStream (RPMs)** リポジトリーおよび **Red Hat Enterprise Linux 8 for x86_64 - BaseOS (RPMs)** リポジトリーが必要です。

Red Hat Enterprise Linux 7 ホストの場合:

Red Hat Enterprise Linux 7 ホストをプロビジョニングするには、**Red Hat Enterprise Linux 7 Server (RPMs)** リポジトリーが必要です。

Red Hat Enterprise Linux オペレーティングシステムのバージョンと **7Server** リポジトリーまたは、**7.X** リポジトリーのいずれかに関連付ける場合の相違点は、**7Server** リポジトリーには最新の更新がすべて含まれますが、**Red Hat Enterprise Linux 7.X** リポジトリーでは次のマイナーリリースが出たら更新の受信を停止する点です。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。

手順

1. Satellite Web UI で、**Content** > **Red Hat Repositories** に移動します。
2. リポジトリーを検索するには、リポジトリー名を入力するか、**Recommended Repositories** ボタンをオンの位置に切り替えて、必要なリポジトリーの一覧を表示します。
3. 利用可能なリポジトリーペインで、リポジトリーをクリックしてリポジトリーセットをデプロイメントします。

4. 必要な基本アーキテクチャーおよびリリースバージョンの横にある Enable アイコンをクリックします。

CLI 手順

1. 製品を検索するには、以下のコマンドを実行します。

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

2. 製品に設定されたリポジトリーを一覧表示します。

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

3. 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。**7Server** などのリリースバージョンと、**x86_64** などのベースアーキテクチャーを含めます。

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

4.7. リポジトリーの同期

コンテンツを Satellite にダウンロードするには、リポジトリーを同期する必要があります。この手順は、リポジトリーの初期同期や、必要に応じてリポジトリーを手動で同期するために使用できます。

組織のすべてのリポジトリーを同期することもできます。詳細は、「組織の全リポジトリーの同期」 を参照してください。

同期プランを作成し、定期的に更新できるようにします。詳細は、「同期プランの作成」 を参照してください。

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 7 時間 55 分
512 Kbps	2 分 43.84 秒	3 時間 24 分 48 秒	1 日 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

	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)
1000 Mbps	0.08 秒	6.29 秒	51.54 秒

手順

1. Satellite Web UI で、**Content** > **Products** に移動し、同期が必要なリポジトリーを含む製品を選択します。
2. 同期するリポジトリーを選択し、**Sync Now** をクリックします。
3. オプション: Satellite Web UI で同期の進捗状況を表示するには、**Content** > **Sync Status** に移動して、対応する製品またはリポジトリーツリーを展開します。

CLI 手順

- 製品全体を同期します。

```
# hammer product synchronize \
--name "My_Product" \
--organization "My_Organization"
```

- 個々のリポジトリーを同期します。

```
# hammer repository synchronize \
--name "My_Repository" \
--organization "My_Organization" \
--product "My Product"
```

4.8. 組織の全リポジトリーの同期

以下の手順を使用して、組織内の全リポジトリーを同期します。

手順

1. SSH を使用して Satellite Server にログインします。
2. 次の Bash スクリプトを実行します。

```
ORG="My_Organization"

for i in $(hammer --no-headers --csv repository list --organization $ORG --fields Id)
do
  hammer repository synchronize --id ${i} --organization $ORG --async
done
```

4.9. ダウンロードポリシーの概要

Red Hat Satellite には、RPM コンテンツ同期用のダウンロードポリシーが複数あります。たとえば、コンテンツのメタデータだけをダウンロードして、実際のコンテンツのダウンロードを後で延期したい場合があります。

Satellite Server には以下のポリシーがあります。

即時

Satellite Server は、同期時にメタデータとパッケージをすべてダウンロードします。

オンデマンド

Satellite Server は同期時にメタデータのみをダウンロードします。Satellite Server は、Capsule または直接接続されたクライアントの要求があると、ファイルシステムへのパッケージの取得と保存のみを行います。この設定は、Satellite Server が強制的にすべてのパッケージをダウンロードするため、Capsule の対応するリポジトリーを **Immediate** に設定しても効果はありません。

オンデマンド ポリシーは、コンテンツの同期時間を短縮するので、**遅延同期** 機能として動作します。遅延同期機能は Yum リポジトリーにのみ使用してください。通常どおりに、コンテンツビューにパッケージを追加して、ライフサイクル環境にプロモートできます。

Capsule Server には、以下のポリシーがあります。

即時

Capsule Server は、同期時にメタデータとパッケージをすべてダウンロードします。Satellite Server で対応するリポジトリーを **オンデマンド** に設定した場合は、Satellite Server が強制的にすべてのパッケージをダウンロードするので、この設定を使用しないでください。

オンデマンド

Capsule Server は、同期時にメタデータのみをダウンロードします。Capsule Server は、直接接続されたクライアントが要求したときのみ、ファイルシステム上でパッケージを取得して保存します。**オンデマンド** ダウンロードポリシーを使用すると、Capsule Server でコンテンツを入手できない場合には、コンテンツが Satellite Server からダウンロードされます。

継承

Capsule Server は、Satellite Server の対応するリポジトリーから、リポジトリーのダウンロードポリシーを継承します。

ストリーミングダウンロードポリシー

Capsule のストリーミングダウンロードポリシーにより、Capsule はコンテンツのキャッシュを回避できます。Capsule からコンテンツが要求されると、Capsule はプロキシーとして機能し、Satellite から直接コンテンツを要求します。

4.10. デフォルトのダウンロードポリシーの変更

Satellite が全組織で作成したリポジトリーに適用するデフォルトのダウンロードポリシーを設定できます。

Red Hat のリポジトリーか、Red Hat 以外のカスタムリポジトリーかによって、Satellite は別の設定を使用します。デフォルト値を変更しても、既存の設定は変更されません。

手順

1. Satellite Web UI で、**Administer** > **Settings** に移動します。
2. **Content** タブをクリックします。
3. 要件に応じて、デフォルトのダウンロードポリシーを変更します。

- Red Hat リポジトリーのデフォルトのダウンロードポリシーを変更するには、**Default Red Hat Repository download policy** 設定の値を変更します。
- デフォルトのカスタムリポジトリーダウンロードポリシーを変更するには、**Default Custom Repository download policy** 設定の値を変更します。

CLI 手順

- デフォルトの Red Hat リポジトリーダウンロードポリシーを **immediate** または **on_demand** のいずれかに変更するには、以下のコマンドを入力します。

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

- Red Hat 以外のカスタムリポジトリーのデフォルトダウンロードポリシーを **immediate** または **on_demand** のいずれかに変更するには、以下のコマンドを入力します。

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

4.11. リポジトリーのダウンロードポリシーの変更

リポジトリーのダウンロードポリシーを設定できます。

手順

1. Satellite Web UI で、**Content** > **Products** に移動します。
2. 必要な製品名を選択します。
3. **Repositories** タブで、必要なリポジトリー名をクリックし、**Download Policy** フィールドを見つけ、編集アイコンをクリックします。
4. リストから、必要なダウンロードポリシーを選択し、**Save** をクリックします。

CLI 手順

1. 組織のリポジトリーをリスト表示します。

```
# hammer repository list \
--organization-label My_Organization_Label
```

2. リポジトリーのダウンロードポリシーを **immediate** または **on_demand** に変更します。

```
# hammer repository update \
--download-policy immediate \
--name "My_Repository" \
--organization-label My_Organization_Label \
--product "My_Product"
```

4.12. ミラーリングポリシーの概要

ミラーリングにより、ローカルリポジトリーがアップストリームリポジトリーと正確に同期します。前回の同期以降、次の同期でコンテンツがアップストリームリポジトリーから削除された場合、そのコンテンツはローカルリポジトリーからも削除されます。

ミラーリングポリシーを使用して、リポジトリーを同期するときにリポジトリーデータおよびコンテンツのミラーリングをより細かく制御できます。たとえば、リポジトリーのリポジトリーをミラーリングできない場合は、このリポジトリーのコンテンツのみをミラーリングするようにミラーリングポリシーを設定できます。

Satellite Server には、次のミラーリングポリシーがあります。

加法

コンテンツもリポジトリーデータもミラーリングされません。したがって、最後の同期以降に追加された新しいコンテンツのみがローカルリポジトリーに追加され、何も削除されません。

コンテンツのみ

コンテンツのみをミラーリングし、レポデータはミラーリングしません。一部のリポジトリーはメタデータミラーリングをサポートしていません。そのような場合、ミラーリングポリシーをコンテンツのみに設定して、コンテンツのみをミラーリングできます。

完全なミラーリング

コンテンツおよびレポデータをミラーリングします。これが最速の方法です。このミラーリングポリシーは、Yum コンテンツでのみ使用できます。

警告

完全なミラーリングのミラーリングポリシーが適用されたリポジトリーのメタデータを再公開しないでください。完全なミラーリングのミラーリングポリシーが適用されたリポジトリーを含むコンテンツビューについても同様です。

4.13. リポジトリーのミラーリングポリシーの変更

リポジトリーのミラーリングポリシーを設定できます。

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

手順

1. Satellite Web UI で、**Content** > **Products** に移動します。
2. 製品名を選択します。
3. **Repositories** タブで、リポジトリー名をクリックし、**Mirroring Policy** フィールドを見つけて、編集アイコンをクリックします。
4. リストからミラーリングポリシーを選択し、**Save** をクリックします。

CLI 手順

1. 組織のリポジトリーをリスト表示します。

```
# hammer repository list \
--organization-label My_Organization_Label
```

2. リポジトリーのミラーリングポリシーを **additive**、**mirror_complete**、または **mirror_content_only** に変更します。

```
# hammer repository update \
--id 1 \
--mirroring-policy mirror_complete
```

4.14. カスタム RPM リポジトリーへのコンテンツのアップロード

個別の RPM およびソース RPM をカスタム RPM リポジトリーにアップロードできます。RPM は、Satellite Web UI または Hammer CLI を使用してアップロードできます。ソース RPM をアップロードするには、Hammer CLI を使用する必要があります。

手順

1. Satellite Web UI で、**Content** > **Products** に移動します。
2. カスタム製品の名前をクリックします。
3. **リポジトリー** タブで、カスタム RPM リポジトリーの名前をクリックします。
4. **Upload Package** で **Browse...** をクリックし、アップロードする RPM を選択します。
5. **Upload** をクリックします。

このリポジトリー内のすべての RPM を表示するには、**Content Counts** 配下の **Packages** の横にある数字をクリックします。

CLI 手順

- 以下のコマンドを入力して、RPM をアップロードします。

```
# hammer repository upload-content \
--id My_Repository_ID \
--path /path/to/example-package.rpm
```

- 以下のコマンドを入力してソース RPM をアップロードします。

```
# hammer repository upload-content \
--content-type srpm \
--id My_Repository_ID \
--path /path/to/example-package.src.rpm
```

アップロードが完了すると、**hammer srpm list** コマンドおよび **hammer srpm info --id srpm_ID** コマンドを使用してソース RPM に関する情報を表示できます。

4.15. CAPSULE でのコンテンツ数の更新

Capsule で同期済みのコンテンツが有効になっている場合、Capsule に関連付けられた環境で使用可能なコンテンツの数を更新できます。このとき、Capsule で使用可能な環境内のコンテンツビューが表示

されます。その後、コンテンツビューを展開して、そのコンテンツビューバージョンに関連付けられているリポジトリーを表示できます。

手順

1. Satellite Web UI で、**Infrastructure** > **Capsules** に移動し、同期されたコンテンツを表示する Capsule を選択します。
2. **Overview** タブを選択します。
3. **Content Sync** の下で **Synchronize** ボタンを切り替えて、**Optimized Sync** または **Complete Sync** を実行し、コンテンツ数を更新する Capsule を同期します。
4. **コンテンツ** タブを選択します。
5. **>** をクリックして **Environment** を選択し、それらの Capsule で利用可能なコンテンツビューを表示します。
6. **>** をクリックしてコンテンツビューを展開し、コンテンツビューで使用可能なリポジトリーと環境の特定のバージョンを表示します。
7. yum リポジトリーに固有の **Packages** の下にあるコンテンツ数を表示します。
8. **Additional content** で、エラータ、パッケージグループ、ファイル、コンテナータグ、コンテナーマニフェスト、および Ansible コレクションの数を表示します。
9. 環境の右側の列にある縦の省略記号をクリックし、**Refresh counts** をクリックして、**Packages** の下の Capsule で同期されたコンテンツ数を更新します。

4.16. カスタムポートでコンテンツの同期を許可する SELINUX の設定

SELinux は、特定のポートでのみコンテンツの同期のために Satellite へのアクセスを許可します。デフォルトでは、次のポートで実行されている Web サーバーへの接続が許可されています: 80、81、443、488、8008、8009、8443、および 9000。

手順

1. Satellite で、コンテンツの同期用に SELinux が許可するポートを確認するには、以下のコマンドを実行します。

```
# semanage port -l | grep ^http_port_t
http_port_t     tcp      80, 81, 443, 488, 8008, 8009, 8443, 9000
```

2. コンテンツ同期用のポート (例: 10011) を許可するように SELinux を設定するには、以下のようにコマンドを入力します。

```
# semanage port -a -t http_port_t -p tcp 10011
```

4.17. 破損したリポジトリーの回復

リポジトリーの破損の場合は、高度な同期を使用して復元できます。これには 3 つのオプションがあります。

最適化された同期

アップストリームのパッケージとの違いが検出されないパッケージをバイパスして、リポジトリーを同期します。

完全な同期

検出された変更に関係なく、すべてのパッケージを同期します。特定のパッケージがアップストリームリポジトリーに存在していても、ローカルリポジトリーにダウンロードできなかった場合は、このオプションを使用します。

コンテンツチェックサムの確認

すべてのパッケージを同期してから、すべてのパッケージのチェックサムをローカルで検証します。RPM のチェックサムがアップストリームと異なる場合は、RPM をもう一度ダウンロードします。このオプションは、Yum コンテンツにのみ関連します。以下のいずれかのエラーがある場合は、このオプションを使用します。

- 特定のパッケージでは、**yum** との同期中に **404** エラーが発生します。
- **Package does not match intended download** エラー。特定のパッケージが破損していることを意味します。

手順

1. Satellite Web UI で、**Content** > **Products** に移動します。
2. 破損したリポジトリーを含む製品を選択します。
3. 同期するリポジトリーの名前を選択します。
4. 最適化された同期または完全な同期を実行するには、**Select Action** メニューから **Advanced Sync** を選択します。
5. 必要なオプションを選択して、**Sync** をクリックします。
6. オプション: チェックサムを確認するには、**Select Action** メニューから **Verify Content Checksum** をクリックします。

CLI 手順

1. リポジトリー ID のリストを取得します。

```
# hammer repository list \
--organization "My_Organization"
```

2. 必要なオプションを使用して破損したリポジトリーを同期します。
   - 最適な同期の場合:

```
# hammer repository synchronize \
--id My_ID
```

   - 完全同期の場合:

```
# hammer repository synchronize \
--id My_ID \
--skip-metadata-check true
```

- 検証コンテンツの同期について以下を実行します。

```
# hammer repository synchronize \
--id My_ID \
--validate-contents true
```

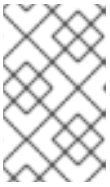
4.18. リポジトリーメタデータの再公開

リポジトリーのコンテンツに基づいて配布されるべきコンテンツがリポジトリーディストリビューションに含まれていない場合、リポジトリーメタデータを再公開できます。

この手順は注意して使用してください。Red Hat は、リポジトリーを完全に同期するか、新しいコンテンツビューバージョンを公開して、破損したメタデータを修復することを推奨します。

手順

1. Satellite Web UI で、**Content** > **Products** に移動します。
2. メタデータを再公開するリポジトリーを含む製品を選択します。
3. **Repositories** タブで、リポジトリーを選択します。
4. リポジトリーのメタデータを再公開するには、**Select Action** メニューから **Republish Repository Metadata** をクリックします。



注記

メタデータはリポジトリーのアップストリームソースからそのままコピーされるため、このアクションは **完全なミラーリング** ポリシーを使用するリポジトリーでは使用できません。

4.19. コンテンツビューメタデータの再公開

コンテンツビューのメタデータを再公開するには、この手順を使用します。

手順

1. Satellite Web UI で、**Content** > **Lifecycle** > **Content Views** に移動します。
2. コンテンツビューを選択します。
3. **Versions** タブで、コンテンツビューバージョンを選択します。
4. 当該コンテンツビューバージョンのメタデータを再公開するには、縦の省略記号アイコンから **Republish repository metadata** をクリックします。

リポジトリーメタデータを再公開すると、**Complete Mirroring** ポリシーに準拠していないコンテンツビューバージョン内のすべてのリポジトリーのメタデータが再生成されます。

4.20. HTTP プロキシーの追加

この手順を使用して、HTTP プロキシーを Satellite に追加します。次に、製品、リポジトリー、およびサポートされるコンピュートリソースに使用する HTTP プロキシーを指定できます。

前提条件

HTTP プロキシーは、次のホストへのアクセスを許可する必要があります。

ホスト名	ポート	プロトコル
subscription.rhsm.redhat.com	443	HTTPS
cdn.redhat.com	443	HTTPS
*.akamaiedge.net	443	HTTPS
cert.console.redhat.com (Red Hat Insights を使用している場合)	443	HTTPS
api.access.redhat.com (Red Hat Insights を使用している場合)	443	HTTPS
cert-api.access.redhat.com (Red Hat Insights を使用している場合)	443	HTTPS

Satellite Server がプロキシーを使用して subscription.rhsm.redhat.com および cdn.redhat.com と通信する場合には、プロキシーはこれらの通信に対して SSL インスペクションを実行できません。

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

手順

1. Satellite Web UI で、**Infrastructure** > **HTTP Proxies** に移動して、**New HTTP Proxy** を選択します。
2. **Name** フィールドで、HTTP プロキシーの名前を入力します。
3. **URL** フィールドには、ポート番号を含めて HTTP プロキシーの URL を入力します。
4. HTTP プロキシーの認証が必要な場合は、**Username** と **Password** を入力します。
5. オプション: **Test URL** フィールドに HTTP プロキシー URL を入力してから **Test Connection** をクリックし、Satellite から HTTP プロキシーに接続できることを確認します。
6. **Locations** タブで、ロケーションを追加します。
7. **Organization** タブをクリックして、組織を追加します。
8. **Submit** をクリックします。

CLI 手順

- Satellite Server で、以下のコマンドを入力して HTTP プロキシーを追加します。

```
# hammer http-proxy create \
--name proxy-name \
--url proxy-URL:port-number
```

-

HTTP プロキシーの認証が必要な場合は、**--username name** および **--password password** オプションを追加します。

詳細は、ナレッジベースの記事 How to access Red Hat Subscription Manager (RHSM) through a firewall or proxy を参照してください。

4.21. 製品の HTTP プロキシーポリシーの変更

ネットワークトラフィックを細かく制御するために、製品ごとに HTTP プロキシーポリシーを設定できます。製品の HTTP プロキシーポリシーは、個別のリポジトリーに異なるポリシーを設定しない限り、製品内のすべてのリポジトリーに適用されます。

個々のリポジトリーに HTTP プロキシーポリシーを設定するには、「リポジトリーの HTTP プロキシーポリシーの変更」 を参照してください。

手順

1. Satellite Web UI で、**Content** > **Products** に移動し、変更が必要な各製品の横にあるチェックボックスを選択します。
2. **Select Action** リストから **Manage HTTP Proxy** を選択します。
3. 一覧から **HTTP Proxy Policy** を選択します。
   - **Global Default**: グローバルデフォルトプロキシー設定を使用します。
   - **No HTTP Proxy**: グローバルデフォルトプロキシーが設定されている場合でも、HTTP プロキシーを使用しないでください。
   - **Use specific HTTP Proxy**: 一覧から **HTTP Proxy** を選択します。HTTP プロキシーを Satellite に追加してから、リストからプロキシーを選択する必要があります。詳細は、「HTTP プロキシーの追加」 を参照してください。
4. **Update** をクリックします。

4.22. リポジトリーの HTTP プロキシーポリシーの変更

ネットワークトラフィックを細かく制御するために、リポジトリーごとに HTTP プロキシーポリシーを設定できます。Satellite Web UI の代わりに CLI を使用する場合は、CLI 手順 を参照してください。

製品内のすべてのリポジトリーに同じ HTTP プロキシーポリシーを設定するには、「製品の HTTP プロキシーポリシーの変更」 を参照してください。

手順

1. Satellite Web UI で、**Content** > **Products** に移動し、リポジトリーが含まれる製品名をクリックします。
2. **Repositories** タブで、リポジトリー名をクリックします。
3. **HTTP Proxy** フィールドを見つけ、編集アイコンをクリックします。
4. 一覧から **HTTP Proxy Policy** を選択します。
   - **Global Default**: グローバルデフォルトプロキシー設定を使用します。

- **No HTTP Proxy**: グローバルデフォルトプロキシーが設定されている場合でも、HTTP プロキシーを使用しないでください。
- **Use specific HTTP Proxy**: 一覧から **HTTP Proxy** を選択します。HTTP プロキシーを Satellite に追加してから、リストからプロキシーを選択する必要があります。詳細は、「HTTP プロキシーの追加」 を参照してください。

5. **Save** をクリックします。

CLI 手順

- Satellite Server で、使用する HTTP プロキシーポリシーを指定して、以下のコマンドを入力します。

```
# hammer repository update \
--http-proxy-policy HTTP_Proxy_Policy \
--id Repository_ID
```

--http-proxy-policy に以下のいずれかのオプションを指定します。

- **none**: グローバルデフォルトプロキシーが設定されている場合でも、HTTP プロキシーを使用しないでください。
- **global_default_http_proxy**: グローバルデフォルトプロキシー設定を使用します。
- **use_selected_http_proxy**: **--http-proxy My_HTTP_Proxy_Name** または **--http-proxy-id My_HTTP_Proxy_ID** を使用して HTTP プロキシーを指定します。新しい HTTP プロキシーを Satellite に追加するには、「HTTP プロキシーの追加」 を参照してください。

4.23. 同期プランの作成

同期プランでは、スケジュールされた日時にコンテンツをチェックし、更新します。Satellite では、同期プランを作成し、製品をプランに割り当てることができます。

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

手順

1. Satellite Web UI で **Content** > **Sync Plans** に移動して、**New Sync Plan** をクリックします。
2. **Name** フィールドに、プランの名前を入力します。
3. オプション: **Description** フィールドには、プランの説明を入力します。
4. **Interval** リストから、プランを実行する間隔を選択します。
5. **Start Date** と **Start Time** リストから、同期プランの実行を開始するタイミングを選択します。
6. **Save** をクリックします。

CLI 手順

1. 同期プランを作成するには、以下のコマンドを入力します。

```
# hammer sync-plan create \
```

```
--description "My_Description" \
--enabled true \
--interval daily \
--name "My_Products" \
--organization "My_Organization" \
--sync-date "2023-01-01 01:00:00"
```

2. 組織で利用可能な同期プランを表示し、同期プランが作成されたことを確認します。

```
# hammer sync-plan list --organization "My_Organization"
```

4.24. 同期プランの製品への割り当て

同期プランでは、スケジュールされた日時にコンテンツをチェックし、更新します。Satellite では、同期プランを製品に割り当てて、コンテンツを定期的に更新できます。

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

手順

1. Satellite Web UI で、**Content > Products** に移動します。
2. 製品を選択します。
3. **詳細** タブで、ドロップダウンメニューから **同期プラン** を選択します。

CLI 手順

1. 同期プランを製品に割り当てます。

```
# hammer product set-sync-plan \
--name "My_Product_Name" \
--organization "My_Organization" \
--sync-plan "My_Sync_Plan_Name"
```

4.25. 複数の製品への同期プランの割り当て

以下の手順を使用して、最低でも 1 度同期され、1 つ以上のリポジトリーが含まれる組織の製品に同期プランを割り当てます。

手順

1. 次の Bash スクリプトを実行します。

```
ORG="My_Organization"
SYNC_PLAN="daily_sync_at_3_a.m"

hammer sync-plan create --name $SYNC_PLAN --interval daily --sync-date "2023-04-5
03:00:00" --enabled true --organization $ORG
for i in $(hammer --no-headers --csv --csv-separator="|" product list --organization $ORG --
per-page 999 | grep -vi not_synced | awk -F'|' '$5 != "0" { print $1}')
```

```
do
  hammer product set-sync-plan --sync-plan $SYNC_PLAN --organization $ORG --id $i
done
```

2. スクリプトの実行後、同期プランに割り当てられた製品を表示します。

```
# hammer product list --organization $ORG --sync-plan $SYNC_PLAN
```

4.26. 同期プランのベストプラクティス

- 製品に同期プランを追加し、コンテンツを定期的に同期して、同期中の Satellite の負荷を低く抑えます。コンテンツを同期する頻度は、少なくするよりも多くします。たとえば、コンテンツを月に 1 回ではなく毎日同期する同期プランを設定します。
- Hammer スクリプトまたは Ansible Playbook を使用して、同期プランの作成と更新を自動化します。
- **Custom Cron** ツールを使用して複数の同期プランを作成し、同期タスクを数時間にわたって分散してタスクの負荷を軽減します。

表4.1 cron 式の例

cron 式	説明
0 22 * * 1-5	月曜日から金曜日まで毎日 22:00 に実行
30 3 * * 6,0	毎週土曜日と日曜日の 3:30 に実行
30 2 8-14 * *	毎月 8 日から 14 日まで毎日 2:30 に実行

4.27. 同期の同時実行の制限

デフォルトでは、各リポジトリー同期ジョブは一度に最大 10 個のファイルを取得できます。これは、リポジトリーごとに調整できます。

制限を増やすとパフォーマンスが向上しますが、アップストリームサーバーのオーバーロードが発生したり、リクエストを拒否し始める可能性があります。アップストリームサーバーがリクエストを拒否したためにリポジトリーの同期に失敗する場合は、制限を下げてみてください。

CLI 手順

```
# hammer repository update \
--download-concurrency 5 \
--id Repository_ID \
--organization "My_Organization"
```

4.28. カスタム GPG キーのインポート

クライアントが署名されたカスタムコンテンツを使用している場合は、クライアントが適切な GPG キーでパッケージのインストールを検証するよう設定されていることを確認してください。これにより、承認されたソースからのパッケージのみをインストールできるようになります。

Red Hat コンテンツは適切な GPG キーでに設定されているため、Red Hat リポジトリーの GPG キー管理はサポートされていません。

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

前提条件

Satellite で使用する RPM コンテンツへの署名に使用する GPG キーのコピーがあることを確認する。ほとんどの RPM ディストリビューションプロバイダーは、GPG キーを Web サイトで提供しています。これは、RPM から手動で抽出することもできます。

1. バージョン固有のリポジトリーパッケージのコピーをローカルマシンにダウンロードします。

```
$ wget http://www.example.com/9.5/example-9.5-2.noarch.rpm
```

2. インストールせずに RPM ファイルをデプロイメントします。

```
$ rpm2cpio example-9.5-2.noarch.rpm | cpio -idmv
```

GPG キーは、**etc/pki/rpm-gpg/RPM-GPG-KEY-EXAMPLE-95** での抽出に相対的に配置されています。

手順

1. Satellite Web UI で **Content** > **Content Credentials** に移動して、ウィンドウの右上の **Create Content Credential** をクリックします。
2. リポジトリーの名前を入力し、**タイプ** のリストから **GPG キー** を選択します。
3. GPG キーを **コンテンツ認証情報の内容** フィールドに貼り付けるか、**参照** をクリックして、インポートする GPG キーファイルを選択します。
   カスタムリポジトリーに複数の GPG キーで署名されたコンテンツが含まれる場合は、コンテンツの **認証情報の内容** フィールドに、各キーの間に新しい行が含まれる GPG キーをすべて入力する必要があります。以下に例を示します。

```
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQINBFy/HE4BEADttv2TCPzVrre+aJ9f5QsR6oWZMm7N5Lwxjm5x5zA9BLiPPGFN
4aTUR/g+K1S0aqCU+ZS3Rnxb+6fnBxD+COH9kMqXHi3M5UNzbp5WhCdUpISXjjpU
XIFFWBPuBfyr/FKRknFH15P+9kLZLxCpVZZLsweLWCuw+JKCMmnA
=F6VG
-----END PGP PUBLIC KEY BLOCK-----

-----BEGIN PGP PUBLIC KEY BLOCK-----

mQINBFw467UBEACmREzDeK/kuScCmfJfHJa0Wgh/2fbJLLt3KSvsgDhORIptf+PP
OTFDlKuLkJx99ZYG5xMnBG47C7ByoMec1j94YeXczuBbynOyyPlvduma/zf8oB9e
Wl5GnzcLGAnUSRamfqGUWcyMMinHHIKIc1X1P4I=
=WPpI
-----END PGP PUBLIC KEY BLOCK-----
```

4. **Save** をクリックします。

CLI 手順

1. GPG キーを Satellite Server にコピーします。

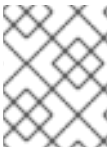
```
$ scp ~/etc/pki/rpm-gpg/RPM-GPG-KEY-EXAMPLE-95 root@satellite.example.com:~/.
```

2. GPG キーを Satellite にアップロードします。

```
# hammer content-credentials create \
--content-type gpg_key \
--name "My_GPG_Key" \
--organization "My_Organization" \
--path ~/RPM-GPG-KEY-EXAMPLE-95
```

4.29. カスタムリポジトリーを特定のオペレーティングシステムまたは SATELLITE のアーキテクチャーに制限する

特定のオペレーティングシステムのバージョンまたはアーキテクチャーを持つホストでのみカスタムリポジトリーを使用できるように Satellite を設定できます。たとえば、カスタムリポジトリーを Red Hat Enterprise Linux 9 ホストのみに制限できます。



注記

カスタム製品のアーキテクチャーとオペレーティングシステムのバージョンのみを制限します。Satellite は、これらの制限を Red Hat リポジトリーに自動的に適用します。

手順

1. Satellite Web UI で、**Content** > **Products** に移動します。
2. 制限するリポジトリーセットを含む製品をクリックします。
3. **Repositories** タブで、制限するリポジトリーをクリックします。
4. **Publishing Settings** セクションで、次のオプションを設定します。
   - オペレーティングシステムのバージョンを制限するには、**Restrict to OS version** を設定します。
   - アーキテクチャーを制限するには、**Restrict to architecture** を設定します。

第5章 ホストによるコンテンツへのアクセスの制限

Satellite は、ホストによるコンテンツへのアクセスを制限するための方法を複数提供します。Satellite が管理するコンテンツの特定のサブセットへのアクセスをホストに許可するには、次の戦略を使用できます。Red Hat は、以下に記載されている順序で戦略を実行するよう検討することを推奨します。

コンテンツビューとライフサイクル環境

コンテンツビューとライフサイクル環境を使用し、必要に応じてコンテンツビューフィルターを組み込みます。
コンテンツビューの詳細は、*7章 コンテンツビューの管理* を参照してください。

ライフサイクル環境の詳細は、*6章 アプリケーションのライフサイクルの管理* を参照してください。

コンテンツのオーバーライド

デフォルトでは、Satellite によってホストされるコンテンツは有効または無効にできます。カスタム製品では、リポジトリーはデフォルトで常に無効になりますが、Red Hat 製品は、特定のリポジトリーに応じてデフォルトで有効または無効にすることができます。リポジトリーを有効にすると、ホストはリポジトリーパッケージまたはその他のコンテンツにアクセスできるようになり、利用可能なコンテンツをダウンロードしてインストールできるようになります。
リポジトリーが無効になっている場合、ホストはリポジトリーのコンテンツにアクセスできません。コンテンツのオーバーライドは、任意のリポジトリーのデフォルトの有効化値である **Enabled** または **Disabled** をオーバーライドする方法を提供します。ホストまたはアクティベーションキーにコンテンツオーバーライドを追加できます。

ホストにコンテンツオーバーライドを追加する方法の詳細は、**ホストの管理** の ホストのリポジトリーの有効化と無効化 を参照してください。

アクティベーションキーにコンテンツオーバーライドを追加する方法の詳細は、「アクティベーションキーでのリポジトリーの有効化と無効化」 を参照してください。

複合コンテンツビュー

複合コンテンツビューを使用すると、複数のコンテンツビューを組み合わせ、ホストが複数のコンテンツビューのコンテンツにアクセスできるようになります。複合コンテンツビューの詳細は、「複合コンテンツビューの作成」 を参照してください。

アーキテクチャーと OS バージョンの制限

カスタム製品では、製品を使用できる **yum** リポジトリーのアーキテクチャーと OS バージョンに制限を設定できます。たとえば、カスタムリポジトリーを **Red Hat Enterprise Linux 8** に制限した場合、そのリポジトリーは Red Hat Enterprise Linux 8 を実行しているホストでのみ使用できます。
アーキテクチャーと OS バージョンの制限は、すべての戦略の中で最も優先されます。これらの制限を、コンテンツオーバーライド、コンテンツビューの変更、またはライフサイクル環境の変更によってオーバーライドまたは無効化することはできません。このため、アーキテクチャーまたは OS バージョンの制限を使用する前に、前述の他の戦略を検討することを推奨します。Red Hat リポジトリーは、アーキテクチャーと OS バージョンの制限を自動的に設定します。

リリースバージョン

Red Hat Enterprise Linux ドットリリースリポジトリーなどの特定の Red Hat リポジトリーには、リポジトリーメタデータに **リリースバージョン** が含まれます。リリースバージョンは、ホストの **システム目的** プロパティーで指定されたリリースバージョンと比較されます。この比較に基づいて、コンテンツへのアクセスが制限される場合があります。システム目的属性の設定に関する詳細は、**ホストの管理** の Red Hat Satellite でのホストの作成 を参照してください。

すべての戦略を組み込む

特定のパッケージまたはリポジトリーは、次のすべてに該当する場合にのみホストで使用できます。

- リポジトリーが、ホストのコンテンツビューおよびライフサイクル環境に含まれている。
- ホストのコンテンツビューにリポジトリーが追加された後、ホストのコンテンツビューが公開されている。
- リポジトリーが、コンテンツビューフィルターによって除外されていない。
- リポジトリーがデフォルトで有効になっているか、コンテンツオーバーライドを使用して **Enabled** にオーバーライドされている。
- リポジトリーにアーキテクチャーまたは OS バージョンの制限がないか、アーキテクチャーまたは OS バージョンの制限がホストに一致している。
- 特定の Red Hat リポジトリーでは、リリースバージョンが設定されていないか、リリースバージョンがホストのリリースバージョンと一致している。

アクティベーションキーの管理

アクティベーションキーを使用すると、これらの戦略の一部でワークフローを簡略化できます。アクティベーションキーを使用して、次のアクションを実行できます。

- ホストをコンテンツビューとライフサイクル環境に割り当てる。
- コンテンツオーバーライドをホストに追加する。
- リリースバージョンを含む、ホスト上のシステム目的属性を設定する。

アクティベーションキーは、登録中のホストにのみ影響します。ホストがすでに登録されている場合、上記の属性はホストごとに個別に変更することも、コンテンツホストの一括アクションを通じて変更することもできます。詳細は、**コンテンツの管理** の アクティベーションキーの管理 を参照してください。

第6章 アプリケーションのライフサイクルの管理

本章では、Satellite におけるアプリケーションライフサイクルと、Satellite および Capsule のアプリケーションライフサイクルの作成および削除方法を説明します。

6.1. アプリケーションのライフサイクルの概要

アプリケーションライフサイクル は、Satellite 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そのソフトウェアがどのように見えるかを定義します。たとえば、アプリケーションライフサイクルは単純である可能性があり、開発段階と実稼働段階のみが存在する可能性があります。この場合、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、ベータリリースやテストフェーズなど、追加の段階が含まれる場合があります。これにより、アプリケーションライフサイクルに別のステージが追加されます。

- 開発
- テスト
- ベータリリース
- 実稼働

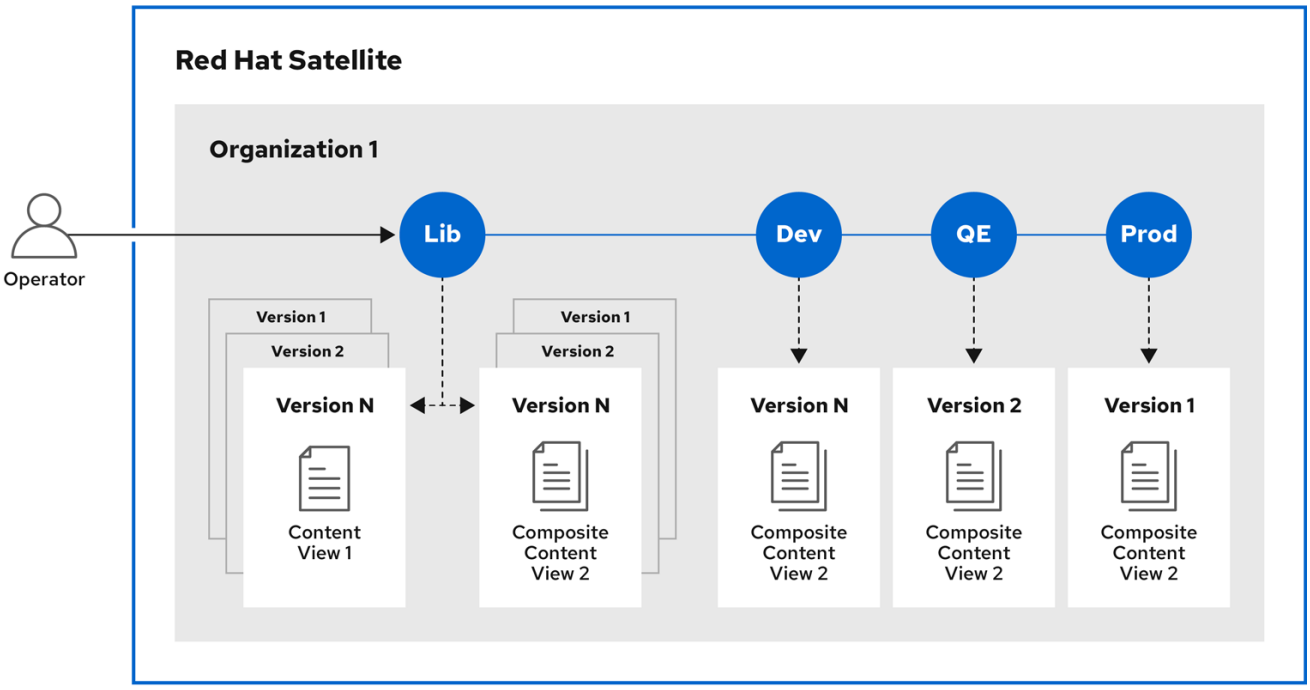
Satellite は、仕様に合わせて各アプリケーションライフサイクルの段階をカスタマイズする方法を提供します。

アプリケーションライフサイクルの各ステージは、Satellite では**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。Satellite は、これらのコンテンツコレクションをコンテンツビューとして定義します。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およびパッケージを定義できるフィルターとして機能します。これにより、各環境に指定する特定のコンテンツセットを定義する方法が提供されます。

たとえば、メールサーバーには、実際に使用する実稼働レベルのサーバーがあり、最新のメールサーバーパッケージを試すテストサーバーがある単純なアプリケーションライフサイクルのみが必要となります。テストサーバーが初期フェーズをパスしたら、実稼働レベルのサーバーが新しいパッケージを使用するように設定できます。

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

図6.1 Satellite アプリケーションのライフサイクル



6.2. アプリケーションライフサイクルのコンテンツプロモーション

アプリケーションライフサイクルチェーンでは、ある環境から次の環境へコンテンツを移動すると、これは **プロモーション** と呼ばれます。

例: Satellite ライフサイクル環境全体でのコンテンツプロモーション

各環境には、Red Hat Satellite に登録されているシステムセットが含まれます。これらのシステムは、環境に関連するリポジトリーにのみアクセスできます。別の環境にパッケージをプロモートすると、ターゲット環境のリポジトリーで新しいパッケージバージョンを受け取ります。これにより、ターゲット環境の各システムは、新しいパッケージバージョンに更新できます。

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

パッチ開発が完了したら、パッケージをテスト環境にプロモートし、品質保証エンジニアチームがパッチをレビューできるようにします。アプリケーションライフサイクルには、各環境に以下のパッケージバージョンが含まれます。

開発	テスト	実稼働
example_software-1.1-0.noarch.rpm	**example_software-1.1-0.noarch.rpm**	**example_software**-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行う間、開発チームは **example_software** 2.0 で作業を開始します。これにより、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
example_software-2.0-0.noarch.rpm	**example_software**-1.1-0.noarch.rpm	**example_software**-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ます。これで **example_software** 1.1 をリリースする準備が整いました。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
example_software-2.0-0.noarch.rpm	**example_software**-1.1-0.noarch.rpm	**example_software-1.1-0.noarch.rpm**

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

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

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

開発	テスト	実稼働
example_software-2.0-0.noarch.rpm	**example_software**-2.0-0.noarch.rpm	**example_software-2.0-0.noarch.rpm**

詳細は、「コンテンツビューのプロモート」 を参照してください。

6.3. ライフサイクル環境のベストプラクティス

- 複数のライフサイクル環境パスを使用して、連続的なコンテンツ消費ステージを複数実装します。各ステージに、**実稼働** ライフサイクル環境などの特定のコンテンツセットを含めます。
- Hammer スクリプトまたは Ansible Playbook を使用して、ライフサイクル環境の作成を自動化します。
- デフォルトのユースケース: 各ライフサイクル環境パスの固定ステージ (例: **開発**、**テスト**、**実稼働**)。
    - コンテンツビューをライフサイクル環境にプロモートします (たとえば、**テスト** から **実稼働** へ)。このコンテンツビューまたは複合コンテンツビューを使用するすべてのコンテンツホストが、**実稼働** ライフサイクル環境からパッケージをインストールできます。これらのパッケージは自動的にインストールまたは更新されないことに注意してください。

    - コンテンツホストのパッチ適用中にエラーが発生した場合は、ホストを以前のバージョンのコンテンツビューに割り当てます。これはパッケージが利用可能性にのみ影響し、インストール済みパッケージがダウングレードされることはありません。

- 別のユースケース: ライフサイクル環境のステージを固定コンテンツ用 (四半期ごとの更新など) に使用し、エラータからの増分更新を含む新しいマイナーバージョンのみを公開します。

    - コンテンツホストにパッチを適用するときに、Satellite Web UI、Satellite API、Hammer CLI、またはアクティベーションキーを使用して、ライフサイクル環境を **2023-Q4** から **2024-Q1** に変更します。

    - メリット: ライフサイクル環境を確認することで、ホストが受信するソフトウェアパッケージを直接確認できます。

    - デメリット: **開発**、**テスト**、**実稼働** などのステージが明確に定義されていないと、コンテンツのプロモートが動的ではなくなります。

- 複数のライフサイクル環境パスを使用して、たとえば Web サーバーホストとデータベースホストを分離するために、さまざまな環境に対して複数のステージを定義します。

- Capsule Server はライフサイクル環境を使用してコンテンツを同期します。コンテンツを複数のライフサイクル環境パスに分けると、コンテンツがより効率的に同期されます。特定の Capsule Server で、単一のライフサイクル環境パス内の 1 つのオペレーティングシステムのコンテンツだけを提供すると、必要なコンテンツのみが同期されます。

6.4. ライフサイクル環境パスの作成

ソフトウェアを開発およびリリースするためのアプリケーションライフサイクルを作成するには、**ライブラリー** 環境を初期環境として使用して、環境パスを作成します。次に、オプションで環境パスに環境を追加します。

手順

1. Satellite Web UI で、**Content** > **Lifecycle** > **Lifecycle Environments** に移動します。
2. **新規環境パス** をクリックして、新しいアプリケーションライフサイクルを開始します。
3. **名前** フィールドに、環境の名前を入力します。
4. **説明** フィールドには、お使いの環境の説明を入力します。
5. **Save** をクリックします。
6. オプション: 環境パスに環境を追加するには、**新しい環境の追加** をクリックし、**名前** と **説明** フィールドを入力します。続いて、**以前の環境** リストから以前の環境を選択します。

CLI 手順

1. 環境パスを作成するには、**hammer lifecycle-environment create** コマンドを入力し、**--prior** オプションでライブラリー環境を指定します。

```
# hammer lifecycle-environment create \
--name "Environment Path Name" \
--description "Environment Path Description" \
--prior "Library" \
--organization "My_Organization"
```

2. オプション: 環境パスに環境を追加するには、**hammer lifecycle-environment create** コマンドを入力し、**--prior** オプションを使用して親環境を指定します。

```
# hammer lifecycle-environment create \
--name "Environment Name" \
--description "Environment Description" \
--prior "Prior Environment Name" \
--organization "My_Organization"
```

3. ライフサイクル環境チェーンを表示するには、以下のコマンドを入力します。

```
# hammer lifecycle-environment paths --organization "My_Organization"
```

6.5. CAPSULE SERVER へのライフサイクル環境の追加

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

ライブラリー ライフサイクル環境は、CDN がリポジトリーを更新するたびに自動的に Capsule が同期をトリガーするため、Capsule Server に割り当てないでください。自動で同期される場合、Capsule 上の複数のシステムリソースや Satellite と Capsule 間のネットワーク帯域幅、および Capsule 上の利用可能なディスク領域が消費される可能性があります。

Satellite Server の Hammer CLI または Satellite Web UI を使用できます。

手順

1. Satellite Web UI で、**Infrastructure** > **Capsule** に移動し、ライフサイクルを追加する Capsule を選択します。
2. **Edit** をクリックしてから、**Lifecycle Environments** タブをクリックします。
3. 左側のメニューから、Capsule に追加するライフサイクル環境を選択し、**Submit** をクリックします。
4. Capsule のコンテンツを同期するには、**Overview** タブをクリックして **Synchronize** をクリックします。
5. **Optimized Sync** または **Complete Sync** を選択します。
同期の各タイプの定義については、リポジトリーの復旧 を参照してください。

CLI 手順

1. Satellite Server で、Capsule Server の全リストを表示するには、以下のコマンドを入力します。

```
# hammer capsule list
```

ライフサイクルの追加先となる Capsule の Capsule ID をメモします。

2. ID を使用して、Capsule の詳細を確認します。

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

3. Capsule Server で利用可能なライフサイクル環境を表示するには、以下のコマンドを入力して、ID と組織名を書き留めます。

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

4. ライフサイクル環境を Capsule Server に追加します。

```
# hammer capsule content add-lifecycle-environment \
--id My_capsule_ID \
--lifecycle-environment-id My_Lifecycle_Environment_ID
--organization "My_Organization"
```

Capsule Server に追加するライフサイクル環境ごとに繰り返します。

5. Satellite から Capsule にコンテンツを同期します。

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

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

- Satellite Server から Capsule Server に特定のライフサイクル環境を同期するには、以下のコマンドを入力します。

```
# hammer capsule content synchronize \
--id My_capsule_ID
--lifecycle-environment-id My_Lifecycle_Environment_ID
```

- メタデータをチェックせずに、Satellite Server から Capsule Server にすべてのコンテンツを同期するには、以下を実行します。

```
# hammer capsule content synchronize \
--id My_capsule_ID \
--skip-metadata-check true
```

これは、Satellite Web UI で **Complete Sync** を選択することと同じです。

6.6. SATELLITE SERVER からのライフサイクル環境の削除

以下の手順を使用して、ライフサイクル環境を削除します。

手順

1. Satellite Web UI で、**Content** > **Lifecycle Environments** に移動します。
2. 削除するライフサイクル環境の名前をクリックし、**環境の削除** をクリックします。
3. **削除** をクリックして環境を削除します。

CLI 手順

1. 組織のライフサイクル環境をリスト表示し、削除するライフサイクル環境の名前を書き留めます。

```
# hammer lifecycle-environment list \
--organization "My_Organization"
```

2. **hammer lifecycle-environment delete** コマンドを使用して環境を削除します。

```
# hammer lifecycle-environment delete \
--name "My_Environment" \
--organization "My_Organization"
```

6.7. CAPSULE SERVER からのライフサイクル環境の削除

ライフサイクル環境がホストシステムまたは環境に関連しなくなった場合に、Capsule Server からライフサイクル環境を削除できます。

Capsule からライフサイクル環境を削除するには、Satellite Web UI と Hammer CLI の両方を使用できます。

手順

1. Satellite Web UI で、**Infrastructure** > **Capsules** に移動し、ライフサイクルを削除する Capsule を選択します。
2. **Edit** をクリックしてから、**Lifecycle Environments** タブをクリックします。
3. 右のメニューから、Capsule から削除するライフサイクル環境を選択し、**Submit** をクリックします。
4. Capsule のコンテンツを同期するには、**Overview** タブをクリックしてから **Synchronize** をクリックします。
5. **Optimized Sync** または **Complete Sync** を選択します。

CLI 手順

1. リストから Capsule Server を選択し、その **ID** を書き留めます。

```
# hammer capsule list
```

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

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

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

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

4. ライフサイクル環境を Capsule Server から削除します。

```
# hammer capsule content remove-lifecycle-environment \
--id My_Capsule_ID \
--lifecycle-environment-id My_Lifecycle_Environment_ID
```

Capsule Server から削除するライフサイクル環境ごとに、この手順を繰り返します。

5. Satellite Server の環境から Capsule Server にコンテンツを同期します。

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

第7章 コンテンツビューの管理

Red Hat Satellite はコンテンツビューを使用して、意図的にキュレートしたコンテンツのサブセットにホストがアクセスできるようにします。これを実行するには、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します。

スナップショットのフィルタリングおよび作成を行うためのコンテンツビューを作成するための一般的なワークフローは以下のとおりです。

1. コンテンツビューを作成します。
2. コンテンツビューに必要なリポジトリーを 1 つ以上追加します。
3. オプション: 1 つ以上のフィルターを作成して、コンテンツビューのコンテンツを絞り込みます。詳細は、「コンテンツフィルターの例」 を参照してください。
4. オプション: コンテンツビューのパッケージの依存関係を解決します。詳細は、「パッケージの依存関係の解決」 を参照してください。
5. コンテンツビューを公開します。
6. オプション: コンテンツビューを別の環境にプロモートします。詳細は、「コンテンツビューのプロモート」 を参照してください。
7. コンテンツホストをコンテンツビューにアタッチします。

リポジトリーがコンテンツビューに関連付けられていない場合、**/etc/yum.repos.d/redhat.repo** ファイルは空のままとなり、登録済みのシステムは更新を受け取ることができません。

ホストは 1 つのコンテンツビューにのみ関連付けることができます。複数のコンテンツビューにホストを関連付けるには、複合コンテンツビューを作成します。詳細は、「複合コンテンツビューの作成」 を参照してください。

7.1. RED HAT SATELLITE のコンテンツビュー

コンテンツビューは、ホストがアクセスできるコンテンツの意図的にキュレーションされたサブセットです。コンテンツビューを作成することで、特定の環境または Capsule Server で使用されるソフトウェアバージョンを定義できます。

各コンテンツビューは、各環境にわたってリポジトリーのセットを作成します。これらのリポジトリーは、Satellite Server によって保存および管理されます。たとえば、次の方法でコンテンツビューを作成できます。

- 実稼働環境用の古いパッケージバージョンを含むコンテンツビューと、**開発** 環境用の新しいパッケージバージョンを含む別のコンテンツビュー。
- オペレーティングシステムに必要なパッケージリポジトリーを含むコンテンツビューと、アプリケーションに必要なパッケージリポジトリーを含む別のコンテンツビュー。
- コンテンツビューを管理するためのモジュール方式の複合コンテンツビュー。たとえば、オペレーティングシステムを管理するためのコンテンツには 1 つのコンテンツビューを使用し、アプリケーションを管理するためのコンテンツには別のコンテンツビューを使用できます。両方のコンテンツビューを組み合わせた複合コンテンツビューを作成することにより、各コンテンツビューのリポジトリーをマージする新しいリポジトリーが作成されます。ただし、コンテンツビューのリポジトリーは引き続き存在するため、個別に管理し続けることもできます。

デフォルトの組織ビュー

デフォルトの組織ビュー は、Satellite に同期されているすべてのコンテンツを対象とした、アプリケーション制御のコンテンツビューです。コンテンツビューとライフサイクル環境を設定しなくても、Satellite の **ライブラリー** 環境にホストを登録して、**デフォルトの組織ビュー** を使用できます。

環境間でのコンテンツビューのプロモート

アプリケーションライフサイクルでコンテンツビューをある環境から次の環境にプロモートすると、Satellite はリポジトリーを更新し、パッケージを公開します。

例7.1 パッケージを 開発 から テスト へプロモートする

テスト および **実稼働** 用のリポジトリーには、**my-software-1.0-0.noarch.rpm** パッケージが含まれています。

	開発	テスト	実稼働
コンテンツビューのバージョン	バージョン 2	バージョン 1	バージョン 1
コンテンツビューの内容	**my-software**-1.1-0.noarch.rpm	**my-software**-1.0-0.noarch.rpm	**my-software**-1.0-0.noarch.rpm

コンテンツビューのバージョン 2 を **開発** から **テスト** にプロモートすると、**テスト** のリポジトリーが更新され、**my-software-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	バージョン 2	**バージョン 2**	バージョン 1
コンテンツビューの内容	**my-software**-1.1-0.noarch.rpm	**my-software-1.1-0.noarch.rpm**	**my-software**-1.0-0.noarch.rpm

これにより、ホストは特定の環境に指定されますが、その環境でコンテンツビューの新しいバージョンが使用されると、更新を受け取るようになります。

7.2. コンテンツビューのベストプラクティス

- Red Hat Enterprise Linux や追加ソフトウェア (**Apache-2.4** や **PostgreSQL-16.2**) などのコンテンツをコンテンツビューでバンドルすると、メンテナンスが容易になります。コンテンツビューが小さすぎる場合は、多くのメンテナンスが必要になります。
- 毎日更新されるコンテンツが必要な場合は、**Default Organization View** コンテンツビューを使用します。これは、すべてのリポジトリーからの最新の同期コンテンツを含んでおり、Library ライフサイクル環境で使用できます。
- 複合コンテンツビューは、より高い柔軟性が要求される状況でのみ使用します。たとえば、あるコンテンツビューを毎週更新し、別のコンテンツビューを毎月更新する場合などです。

- 複合コンテンツビューを使用する場合は、コンテンツビューを公開してから、複合コンテンツビューを公開します。複合コンテンツビューにバンドルするコンテンツビューの数が増えるほど、コンテンツの変更や更新に必要な労力も増えます。
- コンテンツビューが複合コンテンツビューにのみバンドルされている場合、そのコンテンツビューのライフサイクル環境を設定する必要はありません。
- Hammer スクリプトまたは Ansible Playbook を使用して、複合コンテンツビューとライフサイクル環境の作成と公開を自動化します。cron ジョブ、systemd タイマー、または反復ロジックを使用して可視性を確保します。
- 公開した各コンテンツビューまたは複合コンテンツビューのバージョンの説明に変更と日付を追加します。新しいライフサイクル環境へのコンテンツの移動など、最近のアクティビティーは、コンテンツ自体の最近の変更に関係なく、Satellite Web UI に日付別に表示されます。
- 新しいコンテンツビューまたは複合コンテンツビューを公開すると、新しいメジャーバージョンが作成されます。エラータの増分更新により、マイナーバージョンの番号が増加します。このカウンターを変更またはリセットすることはできないことに注意してください。

7.3. コンテンツホストのパッチ適用に関するベストプラクティス

- ホストを Satellite に登録するには、**subscription-manager** パッケージ、**katello-host-tools** パッケージ、およびそれらの依存関係が含まれる Satellite Client 6 が必要です。詳細は、**ホストの管理** の ホストの登録 を参照してください。
- Satellite Web UI を使用して、ホストからパッケージをインストール、アップグレード、削除します。SSH と Ansible を使用して、ジョブテンプレートでコンテンツホストを更新できます。
- Satellite Web UI を使用してコンテンツホストにエラータを適用します。デフォルトのパッケージマネージャーを使用してホスト上のパッケージにパッチを適用する場合、Satellite はパッケージとリポジトリーのリストを受信し、適用可能なエラータと利用可能な更新を再計算します。
- ジョブテンプレートを変更または置き換えて、カスタムのステップを追加します。これにより、ホスト上でコマンドを実行したり、スクリプトを実行したりできるようになります。
- ホストで一括アクションを実行する場合、特にパッケージをアップグレードする場合は、オペレーティングシステムのメジャーバージョンごとにアクションをまとめます。
- 一括アクションを実行する際には、**via remote execution – customize first** を選択し、ホストにパッチを適用する時間を定義します。
- Satellite 上のリポジトリーおよび割り当てられたコンテンツビューに含まれていないパッケージにエラータを適用することはできません。
- **rpm** または **dpkg** を使用してインストールされたパッケージへの変更は、**apt**、**yum**、または **zypper** の次回の実行時に Satellite に送信されます。

7.4. コンテンツビューの作成

以下の手順を使用してシンプルなコンテンツビューを作成します。Satellite Web UI の代わりに CLI を使用する場合は、CLI 手順 を参照してください。

前提条件

コンテンツビューごとに、コンテンツビューでパッケージの依存関係を解決するかどうかを指定できますが、デフォルトの Satellite 設定を変更して、すべてのコンテンツビューでパッケージの解決を有効化または無効化することを推奨します。詳細は、「パッケージの依存関係の解決」 を参照してください。

手順

1. Satellite Web UI で、**Content** > **Lifecycle** > **Content Views** に移動します。
2. **Create content view**をクリックします。
3. **名前** フィールドに、ビューの名前を入力します。Satellite では、入力した名前から **ラベル** フィールドに自動的に入力されます。
4. **説明** フィールドに、ビューの説明を入力します。
5. **Type** フィールドで、**Content view** または **Composite content view**ビューを選択します。
6. オプション: このコンテンツビューを公開するたびに依存関係を自動的に解決する場合は、**Solve dependencies** のチェックボックスを選択します。依存関係の解決により公開時間が遅くなり、使用するコンテンツビューフィルターが無視される可能性があります。また、エラータの依存関係を解決する際に、エラーが発生する可能性があります。
7. **Create content view**をクリックします。

コンテンツビューの手順

1. **Create content view**をクリックして、コンテンツビューを作成します。
2. **Repositories** タブで、**Type** リストからコンテンツビューに追加するリポジトリーを選択し、追加する利用可能なリポジトリーの横にあるチェックボックスを選択して、**Add repositories** をクリックします。
3. **Publish new version** をクリックし、**Description** フィールドに、変更をログに記録するバージョンに関する情報を入力します。
4. オプション: プロモーションパスを有効にするには、**Promote** をクリックします。**利用可能なプロモーションパスからライフサイクル環境を選択して、新しいバージョンをプロモートすることができます。**
5. **Next** をクリックします。
6. **Review** ページで、公開しようとしている環境を確認できます。
7. **Finish** をクリックします。

コンテンツビューは、**Content Views** ページで表示できます。コンテンツビューに関する詳細情報を表示するには、コンテンツビュー名をクリックします。コンテンツビューにホストを登録するには、**ホストの管理** の ホストの登録 を参照してください。

CLI 手順

1. リポジトリー ID のリストを取得します。

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

2. 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 \
--description "My_Content_View" \
--name "My_Content_View" \
--organization "My_Organization" \
--repository-ids 1,2
```

--repository-ids オプションを使用すると、**hammer repository list** コマンドの出力で ID を確認できます。

3. ビューを公開します。

```
# hammer content-view publish \
--description "My_Content_View" \
--name "My_Content_View" \
--organization "My_Organization"
```

4. オプション: 既存のコンテンツビューにリポジトリーを追加するには、以下のコマンドを入力します。

```
# hammer content-view add-repository \
--name "My_Content_View" \
--organization "My_Organization" \
--repository-id repository_ID
```

Satellite Server は、新しいバージョンのビューを作成し、ライブラリー環境に公開します。

7.5. モジュールストリームの表示

Satellite では、コンテンツビューにリポジトリーのモジュールストリームを表示できます。

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

手順

1. Satellite Web UI で、公開されたバージョンの **Content View** > **Module Streams** に移動して、コンテンツタイプで利用可能なモジュールストリームを表示します。
2. **Search** フィールドを使用して、特定のモジュールを検索します。
3. モジュールに関する情報を表示するには、モジュールとそれに対応するタブをクリックして、**Details**、**Repositories**、**Profiles**、および **Artifacts** を含めます。

CLI 手順

1. 組織の一覧を表示します。

```
# hammer organization list
```

2. 組織のすべてのモジュールストリームを表示します。

```
# hammer module-stream list \
--organization-id My_Organization_ID
```

7.6. コンテンツビューのプロモート

以下の手順を使用して、異なるライフサイクル環境全体でコンテンツビューをプロモートします。Satellite Web UI の代わりに CLI を使用する場合は、CLI 手順 を参照してください。

コンテンツビュープロモーションのパーミッション要件

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 パーミッションで、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。

prove_or_remove_content_views_to_environment パーミッションは、ユーザーがコンテンツビューをプロモートできる環境を制限します。

このパーミッションを使用すると、特定の環境に、特定のコンテンツビューをプロモートするが、他の環境にはプロモートできないように、ユーザーパーミッションを割り当てることができます。たとえば、テスト環境へのプロモートを許可し、実稼働環境にはできないようにユーザーを制限できます。

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

手順

1. Satellite Web UI で、**Content** > **Lifecycle** > **Content Views** に移動します。
2. プロモートするコンテンツビューを選択します。
3. プロモートするバージョンを選択し、縦の省略記号アイコンをクリックして **プロモート** をクリックします。
4. コンテンツビューをプロモートする環境を選択し、**Promote** 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 手順

- 各ライフサイクル環境に対して Hammer を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 "My_Organization"
# hammer content-view version promote \
--content-view "Database" \
--version 1 \
--to-lifecycle-environment "Testing" \
--organization "My_Organization"
# hammer content-view version promote \
--content-view "Database" \
```

```
--version 1 \
--to-lifecycle-environment "Production" \
--organization "My_Organization"
```

これで、データベースのコンテンツがすべての環境で利用可能になります。

- あるいは、次の Bash スクリプトを使用して、組織内のすべてのライフサイクル環境にわたってコンテンツビューをプロモートすることもできます。

```
ORG="My_Organization"
CVV_ID=My_Content_View_Version_ID

for i in $(hammer --no-headers --csv lifecycle-environment list --organization $ORG | awk -F, {'print $1'} | sort -n)
do
  hammer content-view version promote --organization $ORG --to-lifecycle-environment-id $i --id $CVV_ID
done
```

検証

- コンテンツビューのバージョンに関する情報を表示して、必要なライフサイクル環境にプロモートされていることを確認します。

```
# hammer content-view version info --id My_Content_View_Version_ID
```

次のステップ

- コンテンツビューにホストを登録するには、**ホストの管理** の ホストの登録 を参照してください。

7.7. 複合コンテンツビューの概要

複合コンテンツビューは、複数のコンテンツビューのコンテンツを組み合わせます。たとえば、オペレーティングシステムとアプリケーションを管理するコンテンツビューが別々の場合があります。複合コンテンツビューを使用して、両方のコンテンツビューのコンテンツを新規リポジトリーにマージできます。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、組み合わせたコンテンツには新規リポジトリーも存在します。

さまざまな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する場合には、example_application は以下のように表示されます。

example_software
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

4 つの別々のコンテンツビューの例:

- 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)
- **example_software** (アプリケーション)

以前のコンテンツビューから、2 つの複合コンテンツビューを作成できます。

PostgreSQL データベースの複合コンテンツビューの例:

複合コンテンツビュー 1 – PostgreSQL の example_software
example_software (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)

MariaDB の複合コンテンツビューの例:

複合コンテンツビュー 2 – MariaDB の example_software
example_software (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)

その後、各コンテンツビューは個別に管理および公開されます。アプリケーションのバージョンを作成すると、複合コンテンツビューの新規バージョンを公開します。複合コンテンツビューの作成時に **Auto Publish** オプションを選択することもできます。その場合、複合コンテンツビューに含まれるコンテンツビューが再公開されると、複合コンテンツビューは自動的に再公開されます。

リポジトリーの制限事項

Docker リポジトリーを複合コンテンツビューに複数回含めることはできません。たとえば、同じ Docker リポジトリーを使用した 2 つのコンテンツビューを複合コンテンツビューに追加しようとすると、Satellite Server はエラーを報告します。

7.8. 複合コンテンツビューの作成

以下の手順を使用して複合コンテンツビューを作成します。Satellite Web UI の代わりに CLI を使用する場合は、CLI 手順 を参照してください。

手順

1. Satellite Web UI で、**Content** > **Lifecycle** > **Content Views** に移動します。
2. **Create content view** をクリックします。

3. **Create content view**ウィンドウで、**Name** フィールドにビューの名前を入力します。Red Hat Satellite では、入力した名前から **ラベル** フィールドに自動的に入力されます。
4. オプション: **Description** フィールドに、ビューの説明を入力します。
5. **Type** タブで、**Composite content view**を選択します。
6. オプション: コンテンツビューが再公開されたときに複合コンテンツビューの新しいバージョンを自動的に公開する場合は、**Auto publish** チェックボックスをオンにします。
7. **Create content view**をクリックします。
8. **Content views** タブで、複合コンテンツビューに追加するコンテンツビューを選択し、**Add content views** をクリックします。
9. **Add content views**ウィンドウで、各コンテンツビューのバージョンを選択します。
10. オプション: コンテンツビューを最新バージョンに自動的に更新する場合は、**Always update to latest version** チェックボックスをオンにします。
11. **Add** をクリックしてから、**Publish new version** をクリックします。
12. オプション: **Description** フィールドに、コンテンツビューの説明を入力します。
13. **Publish** ウィンドウで **Promote** スイッチを設定し、ライフサイクル環境を選択します。
14. **Next** をクリックし、**Finish** をクリックします。

CLI 手順

1. 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 をリスト表示します。

```
# hammer content-view version list \
--organization "My_Organization"
```

2. 新しい複合コンテンツビューを作成します。**--auto-publish** オプションを **yes** に設定すると、そのコンテンツビューを含むコンテンツビューが再公開されると、複合コンテンツビューは自動的に再公開されます。

```
# hammer content-view create \
--composite \
--auto-publish yes \
--name "Example_Composite_Content_View" \
--description "Example composite content view" \
--organization "My_Organization"
```

3. 複合コンテンツビューにコンテンツビューを追加します。コマンド内のコンテンツビュー、コンテンツビューのバージョン、および組織は、ID または名前で識別できます。複数のコンテンツビューを複合コンテンツビューに追加するには、含む必要のあるコンテンツビューごとにこの手順を繰り返します。
   - コンテンツビューで **Always update to latest version**オプションが有効になっている場合:

```
# hammer content-view component add \
--component-content-view-id Content_View_ID \
```

```
--composite-content-view "Example_Composite_Content_View" \
--latest \
--organization "My_Organization"
```

- コンテンツビューの **Always update to latest version** オプションが無効になっている場合:

```
# hammer content-view component add \
--component-content-view-id Content_View_ID \
--composite-content-view "Example_Composite_Content_View" \
--component-content-view-version-id Content_View_Version_ID \
--organization "My_Organization"
```

4. 複合コンテンツビューを公開します。

```
# hammer content-view publish \
--name "Example_Composite_Content_View" \
--description "Initial version of composite content view" \
--organization "My_Organization"
```

5. 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 "Example_Composite_Content_View" \
--version 1 \
--to-lifecycle-environment "Development" \
--organization "My_Organization"
# hammer content-view version promote \
--content-view "Example_Composite_Content_View" \
--version 1 \
--to-lifecycle-environment "Testing" \
--organization "My_Organization"
# hammer content-view version promote \
--content-view "Example_Composite_Content_View" \
--version 1 \
--to-lifecycle-environment "Production" \
--organization "My_Organization"
```

7.9. コンテンツフィルターの概要

コンテンツビューでは、フィルターを使用して特定の Yum 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コンテンツが含まれます。

コンテンツフィルターには、以下の 2 つのタイプがあります。

表7.1 フィルタータイプ

フィルタータイプ	説明
包含	コンテンツなしの状態から開始し、選択したリポジトリーから追加するコンテンツを選択します。このフィルターを使用して、複数のコンテンツアイテムを組み合わせます。

フィルタータイプ	説明
Exclude	選択したリポジトリーからすべてのコンテンツを使用し、削除するコンテンツを選択します。特定のパッケージを除外しながら、特定のコンテンツリポジトリーの大部分を使用する場合は、このフィルターを使用します。このフィルターは、選択したコンテンツ以外のリポジトリー内のコンテンツをすべて使用します。

Include および Exclude フィルターの組み合わせ

Include と Exclude 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 Include フィルターがトリガーされ、次に Exclude フィルターがトリガーされます。この場合には、追加するコンテンツを選択してから、包含のサブセットから除外するコンテンツを選択します。

コンテンツタイプ

以下のコンテンツタイプに基づいて、コンテンツをフィルターできます。

表7.2 コンテンツタイプ

コンテンツタイプ	説明
RPM	名前とバージョン番号に基づいてパッケージをフィルタリングします。**RPM** オプションは、モジュール以外の RPM パッケージとエラータをフィルタリングします。ソース RPM はこのフィルターの影響を受けず、引き続きコンテンツビューで使用できます。
パッケージグループ	パッケージグループに基づいてパッケージをフィルタリングします。パッケージグループのリストは、コンテンツビューに追加されたリポジトリー別になっています。
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。エラータリストは、コンテンツビューに追加されたリポジトリー別になっています。
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された日付範囲およびエラータタイプ (バグ修正、機能拡張、またはセキュリティー) を選択します。
モジュールストリーム	特定のモジュールストリームを含めるか除外するかを選択します。**モジュールストリーム** オプションは、モジュール RPM とエラータをフィルタリングしますが、選択したモジュールストリームに関連付けられるモジュール以外のコンテンツをフィルターしません。
コンテナーイメージタグ	特定のコンテナーイメージタグを含めるか除外するかを選択します。

7.10. パッケージの依存関係の解決

Satellite は、コンテンツビューを公開するときに、コンテンツビュー内のパッケージの依存関係を依存リポジトリーに追加できます。これを設定するには、**依存関係の解決** を有効にします。

依存関係の解決は、たとえば、1 つのパッケージをコンテンツビューのバージョンに増分的に追加する場合に役立ちます。そのようなパッケージをインストールするには、依存関係の解決を有効にする必要がある場合があります。

ただし、依存関係の解決はほとんどの状況では不要です。以下に例を示します。

- コンテンツビューにセキュリティーエラータを増分的に追加する場合、依存関係の解決によってコンテンツビューの公開に大幅な遅延が発生する可能性があり、大きなメリットがありません。
- 新しいエラータのパッケージに、古いコンテンツビューバージョンのパッケージと互換性のない依存関係が含まれている可能性があります。依存関係の解決を使用してエラータを増分的に追加すると、不要なパッケージが含まれる可能性があります。代わりに、コンテンツビューを更新することを検討してください。

注記

依存関係の解決では、コンテンツビューのリポジトリー内のパッケージのみが考慮されます。クライアントにインストールされているパッケージは考慮されません。たとえば、コンテンツビューに AppStream のみが含まれている場合、依存する BaseOS コンテンツは、公開時に依存関係の解決の対象にはなりません。

詳細は、**コンテンツの管理** の リポジトリー依存関係の解決の制限 を参照してください。

依存関係を解決すると、次の問題が発生する可能性があります。

コンテンツビューの公開が大幅に遅延する

Satellite は、コンテンツビュー内のすべてのリポジトリーの依存関係を調べます。したがって、リポジトリーが増えると公開時間も長くなります。
この問題を軽減するには、リポジトリーを減らして複数のコンテンツビューを使用し、それらを複合コンテンツビューに結合します。

依存パッケージのコンテンツビューフィルターが無視される

Satellite は、フィルター内のルールよりもパッケージの依存関係の解決を優先します。
たとえば、セキュリティー目的でフィルターを作成しても、依存関係の解決を有効にすると、Satellite は安全でないと思われるパッケージを追加する可能性があります。

この問題を軽減するには、フィルタリングルールを慎重にテストして、必要な依存関係を確認します。依存関係の解決に不要なパッケージが含まれている場合は、追加パッケージやエラータが必要とするコアの基本依存関係を手動で特定します。

> **例7.2 除外フィルターと依存関係の解決を組み合わせる**
>
> コンテンツビューフィルターを使用して Red Hat Enterprise Linux 8.3 を再作成し、それ以降の Red Hat Enterprise Linux 8 マイナーリリースから選択したエラータを含めるとします。これを実現するには、Red Hat Enterprise Linux 8.3 リリース日以降の、必要な一部を除くほとんどのエラータを除外するフィルターを作成します。次に、依存関係の解決を有効にします。
>
> このような状況では、予想よりも多くのパッケージが依存関係の解決の対象となる可能性があります。その結果、ホストが Red Hat Enterprise Linux 8.3 マシンから逸脱したものになります。

追加のエラータとパッケージが必要ない場合は、コンテンツビューのフィルタリングを設定しないでください。代わりに、Satellite Web UI の **Content** > **Red Hat Repositories** ページで、Red Hat Enterprise Linux 8.3 リポジトリーを有効にして使用してください。

例7.3 パッケージを除外すると DNF の依存関係の解決が不可能になる場合がある

いくつかの除外パッケージを含む Red Hat Enterprise Linux 8.3 リポジトリーを作成すると、**dnf update** が失敗することがあります。

問題を解決するには、依存関係の解決を有効にしないでください。代わりに、**dnf** からのエラーを調査し、欠落している依存関係の除外を停止するようにフィルターを調整します。

そうしないと、依存関係の解決によってリポジトリーが Red Hat Enterprise Linux 8.3 から逸脱したものに可能性があります。

7.11. コンテンツビューの依存関係の解決を有効にする

コンテンツビューの依存関係の解決を有効にするには、以下の手順を使用します。

前提条件

- 依存関係の解決は、限られた状況でのみ役立ちます。有効にする前に、必ず 「パッケージの依存関係の解決」 を読んで理解してください。

手順

1. Satellite Web UI で、**Content** > **Lifecycle** > **Content Views** に移動します。
2. コンテンツビューのリストから、必要なコンテンツビューを選択します。
3. **Details** タブで、**Solve dependencies** を切り替えます。

7.12. コンテンツフィルターの例

以下の手順に従って、カスタムコンテンツフィルターを構築します。

注記

フィルターを使用すると、コンテンツビューの公開に要する時間が大幅に長くなる可能性があります。たとえば、フィルターを使用しなければコンテンツビューの公開タスクが数分で完了する場合、エラータの Exclude または Include フィルターを追加した後は 30 分かかる可能性があります。

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 リポジトリーをコンテンツビューに追加する必要があります。

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

セキュリティー更新を除く、特定の日付以降のすべてのエラータを除外するリポジトリーを作成します。これは、重要なセキュリティー更新 (すぐに適用する必要がある) を除き、システムの更新を定期的に実行する場合に便利です。このフィルターでは、Red Hat Enterprise Linux リポジトリーをコンテンツビューに追加する必要があります。

フィルター:

- **包含タイプ:** Exclude
- **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**機能拡張**のエラータタイプのみを選択し、**セキュリティー**の選択は解除します。**日付タイプ**を**更新日**に設定します。エラータを制限する日付を**開始日**に設定します。**終了日** は空白にして、セキュリティー以外の新たなエラータがフィルターされるようにします。

例 3

例 1 と例 2 の組み合わせにより、オペレーティングシステムパッケージのみが必要で、最新のバグ修正および機能拡張のエラータを除外します。これには、同じコンテンツビューにアタッチされた 2 つのフィルターが必要です。コンテンツビューは、最初に Include フィルターを処理してから、Exclude フィルターを処理します。

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:** Exclude
- **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**機能拡張**のエラータタイプのみを選択し、**セキュリティー**の選択は解除します。**日付タイプ**を**更新日**に設定します。エラータを制限する日付を**開始日**に設定します。**終了日** は空白にして、セキュリティー以外の新たなエラータがフィルターされるようにします。

例 4

コンテンツビューで、特定のモジュールストリームをフィルタリングします。

フィルター 1:

- **包含タイプ:** 組み込み
- **コンテンツタイプ:** モジュールストリーム

- **フィルター:** コンテンツビューに必要な特定のモジュールストリーム (例: **ant**) のみを選択し、**Add Module Stream** をクリックします。

フィルター 2:

- **包含タイプ:** Exclude
- **コンテンツタイプ:** パッケージ
- **フィルター**: コンテンツビューから除外するモジュール以外のパッケージをフィルタリングするルールを追加します。パッケージをフィルタリングしない場合、コンテンツビューフィルターには、モジュールストリーム **ant** に関連付けられるすべてのモジュール以外のパッケージが含まれます。すべての * パッケージを除外するルールを追加するか、除外するパッケージ名を指定します。

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

7.13. YUM コンテンツのコンテンツフィルターの作成

Yum コンテンツを含むコンテンツビューをフィルタリングして、特定のパッケージ、パッケージグループ、エラータ、またはモジュールストリームを含めたり除外したりできます。フィルターは、**名前**、**バージョン**、および **アーキテクチャー** の組み合わせに基づいています。

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

フィルターをビルドする方法の例については、「コンテンツフィルターの例」 を参照してください。

手順

1. Satellite Web UI で、**Content** > **Lifecycle** > **Content Views** に移動します。
2. コンテンツビューを選択します。
3. **Filters** タブで、**Create filter** をクリックします。
4. 名前を入力します。
5. **Content type** リストから、コンテンツタイプを選択します。
6. **Inclusion Type** リストから、**Include filter** または **Exclude filter** のいずれかを選択します。
7. オプション: **説明** フィールドに、フィルターの説明を入力します。
8. **Create filter** をクリックして、コンテンツフィルターを作成します。
9. **コンテンツタイプ** に入力した内容に応じて、必要なフィルターを作成するルールを追加します。
10. フィルターを **リポジトリーのサブセットに適用する** か、**すべてのリポジトリーに適用する** 場合は選択します。
11. **新規バージョンの公開** をクリックして、フィルタリングされたリポジトリーを公開します。
12. オプション: **Description** フィールドには、変更の説明を入力します。
13. **Create filter** をクリックして、新しいバージョンのコンテンツビューを公開します。

このコンテンツビューを全環境にプロモートできます。

CLI 手順

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

```
# hammer content-view filter create \
--name "Errata Filter" \
--type erratum --content-view "Example_Content_View" \
--description "My latest filter" \
--inclusion false \
--organization "My_Organization"
```

2. フィルターにルールを追加します。

```
# hammer content-view filter rule create \
--content-view "Example_Content_View" \
--content-view-filter "Errata Filter" \
--start-date "YYYY-MM-DD" \
--types enhancement,bugfix \
--date-type updated \
--organization "My_Organization"
```

3. コンテンツビューを公開します。

```
# hammer content-view publish \
--name "Example_Content_View" \
--description "Adding errata filter" \
--organization "My_Organization"
```

4. ビューを各環境にプロモートします。

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

7.14. 複数のコンテンツビューバージョンの削除

複数のコンテンツビューバージョンを同時に削除できます。

手順

1. Satellite Web UI で、**Content** > **Lifecycle** > **Content Views** に移動します。
2. バージョンを削除するコンテンツビューを選択します。
3. **バージョン** タブで、削除するバージョンのチェックボックスをオンにします。
4. コンテンツビューのリストの上部にある縦の省略記号アイコンをクリックします。
5. **削除** をクリックして、削除のウィザードが開き、影響を受ける環境が表示されます。
6. 影響を受ける環境がない場合は、詳細を確認して **削除** をクリックします。
7. 影響を受ける環境がある場合は、削除する前にホストまたはアクティベーションキーを再割り当てします。
8. アクションの詳細を確認します。
9. **Delete** をクリックします。

7.15. 検索フィルターのクリア

Search テキストボックスでキーワードを使用して特定のコンテンツタイプを検索しても結果が返されない場合は、**Clear search** をクリックしてすべての検索クエリーをクリアし、**Search** テキストボックスをリセットします。

フィルターを使用して **Type** テキストボックスで特定のリポジトリーを検索し、検索で結果が返されない場合は、**Clear filters** をクリックして有効なフィルターをすべて消去し、**Type** テキストボックスをリセットします。

7.16. コンテンツビューの空の状態の標準化

コンテンツビューにフィルターがリストされていない場合は、**Create filter** をクリックします。モーダルが開き、フィルター作成の次の手順が表示されます。新しいフィルターを追加して新しいコンテンツタイプを作成するには、次の手順に従います。

7.17. コンテンツビューバージョンの比較

次の手順を使用して、Satellite のコンテンツビューバージョンの機能を比較します。

手順

1. Satellite Web UI で、**Content** > **Lifecycle** > **Content Views** に移動します。
2. バージョンを比較するコンテンツビューを選択します。
3. **Version** タブで、比較する 2 つのバージョンの横にあるチェックボックスを選択します。
4. **Compare** をクリックします。
   Compare 画面では、バージョンのドロップダウンメニューに事前に選択したバージョンが表示され、いずれかのバージョンで見つかったすべてのコンテンツタイプのタブが表示されます。結果をフィルタリングして、同じコンテンツタイプ、異なるコンテンツタイプ、またはすべてのコンテンツタイプのみを表示できます。ドロップダウンメニューからコンテンツビューバージョンを選択して、異なるコンテンツビューバージョンを比較できます。

7.18. アーカイブされたコンテンツビューバージョンの分散

Distribute archived content view versions設定を使用すると、プロモートされていないコンテンツビューバージョンのリポジトリーを他のリポジトリーとともに Satellite コンテンツ Web アプリケーションでホストできるようになります。これは、デバッグ中にコンテンツビューのバージョンにどのようなコンテンツが存在するかを確認するのに役立ちます。

手順

1. Satellite Web UI で、**Administer** > **Settings** に移動します。
2. **Content** タブをクリックします。
3. **Distribute archived content view versions**パラメーターを **Yes** に設定します。
4. **Submit** をクリックします。
   これにより、ライフサイクル環境のないコンテンツビューバージョンのリポジトリーを **satellite.example.com/pulp/content/My_Organization/content_views/My_Content_View/My_Content_View_Version/** で配布できるようになります。

注記

設定が有効になると、プロモートされていない以前のコンテンツビューバージョンは分散されません。新しいコンテンツビューバージョンのみが分散されます。

第8章 SATELLITE SERVER 間でのコンテンツの同期

複数の Satellite Server を使用した Satellite セットアップでは、Inter-Satellite Synchronization (ISS) を使用して、1 つのアップストリームサーバーから 1 つ以上のダウンストリームサーバーにコンテンツを同期できます。

インフラストラクチャーのデプロイ方法に応じて、Satellite には 2 つの可能な ISS 設定があります。シナリオに応じて、ISS 用の Satellite を設定します。詳細は、**切断されたネットワーク環境での Satellite Server のインストール の Inter-Satellite Synchronization のシナリオ を** 参照してください。

Pulp エクスポートパスを変更するには、**Red Hat ナレッジベース** の Hammer content export fails with "Path '/the/path' is not an allowed export path" を参照してください。

8.1. エクスポートとインポートを使用したコンテンツの同期

エクスポートとインポートのワークフローを使用してコンテンツを同期するには、複数の方法があります。

- アップストリームの Satellite Server をコンテンツストアとして使用します。つまり、コンテンツビューのバージョンではなく、ライブラリー全体を同期します。このアプローチは、最も単純なエクスポート/インポートワークフローを提供します。このような場合、コンテンツビューのバージョンをダウンストリームに管理できます。詳細は、「アップストリームの Satellite Server のコンテンツストアとしての使用」 を参照してください。
- アップストリームの Satellite Server を使用して、コンテンツビューのバージョンを同期します。このアプローチにより、Satellite Server 間で同期されるコンテンツをより詳細に制御できます。詳細は、「アップストリーム Satellite サーバーを使用してコンテンツビューのバージョンを同期する」 を参照してください。
- 単一のリポジトリーを同期します。これは、コンテンツビューの同期アプローチを使用しているが、追加のリポジトリーを既存のコンテンツビューに追加せずに同期したい場合に便利です。詳細は、「単一リポジトリーの同期」 を参照してください。

注記

エクスポートとインポートを使用してコンテンツを同期するには、ダウンストリームとアップストリームの両方の Satellite Server で同じメジャー、マイナー、パッチバージョンの Satellite が必要です。

アップストリームとダウンストリームの Satellite バージョンを一致させることができない場合は、以下を使用できます。

- 同期可能なエクスポートとインポート。
- インターネットに接続されたアップストリーム Satellite とアップストリーム Satellite に接続されたダウンストリーム Satellite による Satellite 間同期 (ISS)。

8.1.1. アップストリームの Satellite Server のコンテンツストアとしての使用

このシナリオでは、コンテンツを管理するのではなく、アップストリームの Satellite Server を更新用のコンテンツストアとして使用します。ダウンストリームの Satellite Server を使用して、分離されたネットワークの背後にあるすべてのインフラストラクチャーのコンテンツを管理します。アップストリームの Satellite Server からライブラリーのコンテンツをエクスポートし、ダウンストリームの Satellite Server にインポートします。

アップストリームの Satellite Server の場合

1. 次のいずれかの方法で、リポジトリーが **Immediate** ダウンロードポリシーを使用していることを確認します。

   a. **On Demand** を使用する既存のリポジトリーの場合は、リポジトリーの詳細ページでダウンロードポリシーを **Immediate** に変更します。

   b. 新しいリポジトリーの場合、Red Hat リポジトリーを有効にする前に、**Default Red Hat Repository download policy** 設定が **Immediate** に設定されていること、およびカスタムリポジトリーの**Default download policy** が **Immediate** に設定されていることを確認します。

   詳細は、「ダウンロードポリシーの概要」 を参照してください。

2. 同期するコンテンツを有効にします。詳細は、「Red Hat リポジトリーの有効化」 を参照してください。
   カスタムコンテンツを同期する場合は、まず カスタム製品を作成 し、製品リポジトリーを同期します。

3. 有効なコンテンツを同期します。

   a. 最初のエクスポートでは、**complete** ライブラリーのエクスポートを実行して、同期したすべてのコンテンツがエクスポートされるようにします。これにより、1 つ以上のダウンストリーム Satellite Server に後でインポートできるコンテンツアーカイブが生成されます。完全なライブラリーのエクスポートの実行に関する詳細は、「ライブラリー環境のエクスポート」 を参照してください。

   b. アップストリームの Satellite Server の今後のすべての更新を増分的にエクスポートします。これにより、最近の一連の更新のみを含む、より無駄のないコンテンツアーカイブが生成されます。たとえば、新規リポジトリーを有効にして同期すると、次のエクスポートされたコンテンツアーカイブには、新たに有効なリポジトリーからのみコンテンツが含まれます。増分ライブラリーのエクスポートの実行に関する詳細は、「ライブラリー環境の増分的なエクスポート」 を参照してください。

ダウンストリームの Satellite Server の場合

1. アップストリームの Satellite Server からエクスポートされたコンテンツをハードディスクに移動します。

2. **/var/lib/pulp/imports** 配下のディレクトリー内に配置します。

3. 「ライブラリー環境へのインポート」 に説明されている手順に従って、コンテンツを組織にインポートします。
   その後必要に応じて、コンテンツビューまたはライフサイクル環境を使用してコンテンツを管理できます。

8.1.2. アップストリーム Satellite サーバーを使用してコンテンツビューのバージョンを同期する

このシナリオでは、アップストリームの Satellite Server をコンテンツストアとして使用するだけでなく、分離されたネットワークの背後にあるすべてのインフラストラクチャーのコンテンツを同期するためにも使用します。CDN からの更新をコンテンツビューとライフサイクル環境にキュレートします。指定されたライフサイクル環境にコンテンツをプロモートしたら、アップストリームの Satellite Server からコンテンツをエクスポートし、ダウンストリームの Satellite Server にインポートできます。

アップストリームの Satellite Server の場合

1. 次のいずれかの方法で、リポジトリーが **Immediate** ダウンロードポリシーを使用していることを確認します。

   a. **On Demand** を使用する既存のリポジトリーの場合は、リポジトリーの詳細ページでダウンロードポリシーを **Immediate** に変更します。

   b. 新しいリポジトリーの場合、Red Hat リポジトリーを有効にする前に、**Default Red Hat Repository download policy** 設定が **Immediate** に設定されていること、およびカスタムリポジトリーの**Default download policy** が **Immediate** に設定されていることを確認します。

   詳細は、「ダウンロードポリシーの概要」 を参照してください。

2. 同期するコンテンツを有効にします。詳細は、「Red Hat リポジトリーの有効化」 を参照してください。
   カスタムコンテンツを同期する場合は、まず カスタム製品を作成 し、製品リポジトリーを同期します。

3. 有効なコンテンツを同期します。

   a. 最初のエクスポートでは、エクスポートするコンテンツビューのバージョンで **complete** バージョンエクスポートを実行します。詳細は、「コンテンツビューバージョンのエクスポート」 を参照してください。これにより、1 つ以上のダウンストリーム Satellite Server にインポートできるコンテンツアーカイブが生成されます。

   b. 接続された Satellite Server のすべての今後の更新を増分的にエクスポートします。これにより、最新の更新セットからのみ変更が含まれるよりスリムなコンテンツアーカイブが生成されます。たとえば、コンテンツビューに新しいリポジトリーがある場合、このエクスポートしたコンテンツアーカイブには最新の変更のみが含まれます。詳細は、「コンテンツビューバージョンの増分エクスポート」 を参照してください。

   c. 新しいコンテンツがある場合は、増分をエクスポートする前に、このコンテンツを含むコンテンツビューを再公開します。詳細は、*7章 コンテンツビューの管理* を参照してください。これにより、エクスポートする適切なコンテンツを持つコンテンツビューのバージョンが新たに作成されます。

ダウンストリームの Satellite Server の場合

1. アップストリームの Satellite Server からエクスポートされたコンテンツをハードディスクに移動します。

2. **/var/lib/pulp/imports** 配下のディレクトリー内に配置します。

3. コンテンツを希望の組織にインポートします。詳細は、「コンテンツビューバージョンのインポート」 を参照してください。これにより、エクスポートしたコンテンツアーカイブからコンテンツビューバージョンが作成され、続いてコンテンツが適切にインポートされます。

8.1.3. 単一リポジトリーの同期

このシナリオでは、単一のリポジトリーをエクスポートおよびインポートします。

アップストリームの Satellite Server の場合

1. 次のいずれかの方法で、リポジトリーが**Immediate** ダウンロードポリシーを使用していることを確認します。

   a. **On Demand** を使用する既存のリポジトリーの場合は、リポジトリーの詳細ページでダウンロードポリシーを **Immediate** に変更します。

   b. 新しいリポジトリーの場合、Red Hat リポジトリーを有効にする前に、**Default Red Hat Repository download policy** 設定が **Immediate** に設定されていること、およびカスタムリポジトリーの**Default download policy** が **Immediate** に設定されていることを確認します。

   詳細は、「ダウンロードポリシーの概要」 を参照してください。

2. 同期するコンテンツを有効にします。詳細は、「Red Hat リポジトリーの有効化」 を参照してください。
   カスタムコンテンツを同期する場合は、まず カスタム製品を作成 し、製品リポジトリーを同期します。

3. 有効なコンテンツを同期します。

   a. 最初のエクスポートで、**complete** リポジトリーエクスポートを実行して、同期されたすべてのコンテンツがエクスポートされるようにします。これにより、1 つ以上のダウンストリーム Satellite Server に後でインポートできるコンテンツアーカイブが生成されます。完全なリポジトリーエクスポートの実行の詳細については、「リポジトリーのエクスポート」 を参照してください。

   b. アップストリームの Satellite Server の今後のすべての更新を増分的にエクスポートします。これにより、最近の一連の更新のみを含む、より無駄のないコンテンツアーカイブが生成されます。増分ライブラリーのエクスポートの実行に関する詳細は、「リポジトリーの増分エクスポート」 を参照してください。

ダウンストリームの Satellite Server の場合

1. アップストリームの Satellite Server からエクスポートされたコンテンツをハードディスクに移動します。

2. **/var/lib/pulp/imports** 配下のディレクトリー内に配置します。

3. コンテンツを組織にインポートします。「リポジトリーのインポート」を参照してください。その後必要に応じて、コンテンツビューまたはライフサイクル環境を使用してコンテンツを管理できます。

8.2. カスタムリポジトリーの同期

Inter-Satellite Synchronization Network Sync を使用する場合は、Red Hat リポジトリーは自動的に設定されますが、カスタムリポジトリーは設定されません。この手順を使用して、Inter-Satellite Synchronization (ISS) ネットワーク同期を介して、オンライン接続されている Satellite Server 上のカスタムリポジトリーからオフラインの Satellite Server にコンテンツを同期します。

オフラインの Satellite Server の手順を完了する前に、オンライン接続された Satellite Server の手順に従ってください。

オンライン接続された Satellite Server

1. Satellite Web UI で、**Content** > **Products** に移動します。

2. カスタム製品をクリックします。
3. カスタムリポジトリーをクリックします。
4. **Published At:** URL をコピーします。
5. ネットワーク接続されていない Satellite Server で手順を続行します。

ネットワーク接続されていない Satellite Server

1. ネットワーク接続されている Satellite Server から **katello-server-ca.crt** ファイルをダウンロードします。

```
# curl http://satellite.example.com/pub/katello-server-ca.crt
```

2. **katello-server-ca.crt** の内容を使用して SSL コンテンツ認証情報を作成します。SSL コンテンツ認証情報の作成の詳細は、「カスタム SSL 証明書のインポート」 を参照してください。
3. Satellite Web UI で、**Content** > **Products** に移動します。
4. 以下を使用してカスタム製品を作成します。
   - **Upstream URL**: 先ほどコピーしたリンクを貼り付けます。
   - **SSL CA Cert**: ネットワーク接続されている Satellite Server から転送された SSL 証明書を選択します。

   カスタム製品の作成の詳細は、「カスタム製品の作成」 を参照してください。

これらの手順を完了すると、切断された Satellite Server 上にカスタムリポジトリーが適切に設定されます。

8.3. ライブラリー環境のエクスポート

組織のライブラリー環境にあるすべての Yum リポジトリーのコンテンツを Satellite Server からアーカイブファイルにエクスポートし、このアーカイブファイルを使用して、別の Satellite Server または別の Satellite Server の組織に同じリポジトリーを作成することができます。エクスポートしたアーカイブファイルには、以下のデータが含まれます。

- コンテンツビューバージョンのメタデータが含まれる JSON ファイル
- 組織のライブラリー環境からのすべてのリポジトリーを含むアーカイブファイル

Satellite Server は、ライブラリー環境に含まれる RPM、キックスタートファイル、および Docker コンテンツのみをエクスポートします。

前提条件

- エクスポートディレクトリーに、エクスポートに対応できる空き容量があることを確認する。
- **/var/lib/pulp/exports** ディレクトリーに、エクスポートプロセス中に作成された一時ファイルにエクスポートされるリポジトリーのサイズと同じ空き容量があることを確認する。
- エクスポートするライブラリーライフサイクル環境内の全リポジトリーでダウンロードポリシーを **即時** に設定していることを確認する。詳細は、「ダウンロードポリシーの概要」 を参照してください。

- エクスポートする製品が、必要な日付に同期されることを確認する。

手順

1. エクスポートする組織名または ID を使用します。

```
# hammer content-export complete library --organization="My_Organization"
```

2. エクスポートしたコンテンツビューバージョンが含まれるアーカイブが、エクスポートディレクトリーにあることを確認します。

```
# ls -lh /var/lib/pulp/exports/My_Organization/Export-Library/1.0/2021-03-02T03-35-24-00-00
total 68M
-rw-r--r--. 1 pulp pulp 68M Mar  2 03:35 export-1e25417c-6d09-49d4-b9a5-23df4db3d52a-20210302_0335.tar.gz
-rw-r--r--. 1 pulp pulp 333 Mar  2 03:35 export-1e25417c-6d09-49d4-b9a5-23df4db3d52a-20210302_0335-toc.json
-rw-r--r--. 1 pulp pulp 443 Mar  2 03:35 metadata.json
```

インポートするには、**tar.gz**、**toc.json**、および **metadata.json** の 3 つのファイルすべてが必要です。

3. 組織にコンテンツビューの **Export-Library** が新たに作成されました。このコンテンツビューには、この組織に属するすべてのリポジトリーが含まれます。このコンテンツビューの新しいバージョンが自動的に公開され、エクスポートされます。

チャンクを使用したエクスポート

多くの場合、エクスポートされたアーカイブのコンテンツのサイズは数ギガバイトになる可能性があります。小さいサイズまたはチャンクに分割する必要がある場合、分割するには、export コマンドで **--chunk-size-gb** フラグを直接使用できます。以下の例では、**--chunk-size-gb=2** を指定して、**2 GB** のチャンクでアーカイブを分割する方法を確認できます。

```
# hammer content-export complete library \
--chunk-size-gb=2 \
--organization="My_Organization"

Generated /var/lib/pulp/exports/My_Organization/Export-Library/2.0/2021-03-02T04-01-25-00-00/metadata.json

# ls -lh /var/lib/pulp/exports/My_Organization/Export-Library/2.0/2021-03-02T04-01-25-00-00/
```

8.4. ライブラリー環境の同期可能な形式でのエクスポート

組織のライブラリー環境にあるすべての yum リポジトリー、キックスタートリポジトリー、およびファイルリポジトリーのコンテンツを、カスタム CDN を作成し、HTTP/HTTPS を介してカスタム CDN からコンテンツを同期するために使用できる同期可能な形式にエクスポートできます。

その後、生成されたコンテンツをローカル Web サーバー上で提供し、インポート元の Satellite Server または別の Satellite Server 組織で同期できます。

コンテンツのインポートにより、生成されたコンテンツを使用して、別の Satellite Server または別の Satellite Server 組織に同じリポジトリーを作成できます。エクスポートされたアーカイブをインポートすると、インポートする Satellite Server で通常のコンテンツビューが作成または更新されます。詳細

は、「コンテンツビューバージョンのインポート」 を参照してください。

以下のコンテンツを、Satellite Server から同期可能な形式でエクスポートできます。

- Yum リポジトリー
- Kickstart リポジトリー
- ファイルリポジトリー

Ansible、Deb、Docker コンテンツをエクスポートすることはできません。

エクスポートには、インポートする Satellite Server での同期に使用できる Yum 形式のリポジトリーのパッケージ、**リスト** ファイル、およびメタデータを含むディレクトリーが含まれます。

前提条件

- エクスポートするライブラリーライフサイクル環境内の全リポジトリーでダウンロードポリシーを **即時** に設定していることを確認する。詳細は、「ダウンロードポリシーの概要」 を参照してください。
- エクスポートする製品が、必要な日付に同期されることを確認します。
- コンテンツをエクスポートするユーザーに **Content Exporter** ロールがあることを確認します。

手順

1. エクスポートする組織名または ID を使用します。

```
# hammer content-export complete library \
--organization="My_Organization" \
--format=syncable
```

2. オプション: エクスポートされたコンテンツがエクスポートディレクトリーにあることを確認します。

```
# du -sh /var/lib/pulp/exports/My_Organization/Export-My_Repository/1.0/2021-03-02T03-35-24-00-00
```

8.5. 同期可能なエクスポートのインポート

手順

- 組織名または ID を使用して、同期可能なエクスポートをインポートします。

```
# hammer content-import library
--organization="My_Organization"
--path="My_Path_To_Syncable_Export"
```

注記

同期可能なエクスポートは、**/etc/pulp/settings.py** で指定されている **ALLOWED_IMPORT_PATHS** の 1 つに配置する必要があります。デフォルトでは、これには **/var/lib/pulp/imports** が含まれます。

8.6. ライブラリー環境の増分的なエクスポート

ライブラリーコンテンツのエクスポートは、システムリソースにとって非常にコストのかかる操作となる可能性があります。複数の Red Hat Enterprise Linux ツリーがある組織は、Satellite Server で数ギガバイトの領域を占める可能性があります。

このような場合は、前回のエクスポート以降に変更されたコンテンツのみを含む増分エクスポートを作成できます。増分エクスポートでは、通常、完全エクスポートよりもアーカイブファイルが小さくなります。

次の例は、組織のライブラリー内にある全リポジトリーの増分エクスポートを示しています。

手順

1. 増分エクスポートを作成します。

```
# hammer content-export incremental library \
--organization="My_Organization"
```

同期可能なエクスポートを作成する場合は、**--format=syncable** を追加します。デフォルトでは、Satellite はインポート可能なエクスポートを作成します。

次のステップ

- オプション: エクスポートされたデータを表示します。

```
# find /var/lib/pulp/exports/My_Organization/Export-Library/
```

8.7. コンテンツビューバージョンのエクスポート

コンテンツビューのバージョンを Satellite Server からアーカイブファイルにエクスポートし、このアーカイブファイルを使用して、別の Satellite Server か、別の Satellite Server の組織に同じコンテンツビューバージョンを作成できます。Satellite は、通常のコンテンツビューとして複合コンテンツビューをエクスポートします。複合性は保持されません。エクスポートされたアーカイブをインポートすると、ダウンストリームの Satellite Server で通常のコンテンツビューが作成または更新されます。エクスポートしたアーカイブファイルには、以下のデータが含まれます。

- コンテンツビューバージョンのメタデータが含まれる JSON ファイル
- コンテンツビューバージョンに含まれるすべてのリポジトリーを含むアーカイブファイル

エクスポートできるのは、コンテンツビューのバージョンに追加された Yum リポジトリー、キックスタートファイル、および Docker コンテンツのみです。Satellite では、以下の内容はエクスポートされません。

- コンテンツビューの定義およびメタデータ (パッケージフィルターなど)

前提条件

コンテンツビューをエクスポートするには、エクスポートする Satellite Server が、以下の条件を満たしていることを確認します。

- エクスポートディレクトリーに、エクスポートに対応できる空き容量があることを確認する。
- **/var/lib/pulp/exports** ディレクトリーに、エクスポートプロセス中に作成された一時ファイルにエクスポートされるリポジトリーのサイズと同じ空き容量があることを確認する。
- エクスポートするコンテンツビュー内のすべてのリポジトリーで、ダウンロードポリシーを **即時** に設定していることを確認する。詳細は、「ダウンロードポリシーの概要」 を参照してください。
- エクスポートする製品が、必要な日付に同期されることを確認する。
- コンテンツをエクスポートするユーザーに **Content Exporter** ロールがあることを確認します。

コンテンツビューバージョンをエクスポートする方法

1. エクスポート可能なコンテンツビューのバージョンをリスト表示します。

```
# hammer content-view version list \
--content-view="My_Content_View" \
--organization="My_Organization"

---|----------|---------|-------------|----------------------
ID | NAME     | VERSION | DESCRIPTION | LIFECYCLE ENVIRONMENTS
---|----------|---------|-------------|----------------------
5  | view 3.0 | 3.0     |             | Library
4  | view 2.0 | 2.0     |             |
3  | view 1.0 | 1.0     |             |
---|----------|---------|-------------|----------------------
```

コンテンツビューバージョンをエクスポートします。

1. 必要なバージョンのバージョン番号を取得します。以下の例は、エクスポート用のバージョン **1.0** を対象としています。

```
# hammer content-export complete version \
--content-view="Content_View_Name" \
--version=1.0 \
--organization="My_Organization"
```

2. エクスポートしたコンテンツビューバージョンが含まれるアーカイブが、エクスポートディレクトリーにあることを確認します。

```
# ls -lh /var/lib/pulp/exports/My_Organization/Content_View_Name/1.0/2021-02-25T18-59-26-00-00/
```

コンテンツを正常にインポートするには、**tar.gz** アーカイブファイル、**toc.json** および **metadata.json** などの 3 つのファイルがすべて必要です。

チャンクを使用したエクスポート

多くの場合、エクスポートされたアーカイブのコンテンツのサイズは数ギガバイトになる場合がありま

す。小さいサイズまたはチャンクに分割することを推奨します。分割するには、**--chunk-size-gb** オプションを指定して **hammer content-export** コマンドを使用します。以下の例では、**--chunk-size-gb=2** を使用してアーカイブを **2 GB** のチャンクに分割します。

```
# hammer content-export complete version \
--chunk-size-gb=2 \
--content-view="Content_View_Name" \
--organization="My_Organization" \
--version=1.0
# ls -lh /var/lib/pulp/exports/My_Organization/view/1.0/2021-02-25T21-15-22-00-00/
```

8.8. コンテンツビューバージョンを同期可能な形式でエクスポートする

コンテンツビューのバージョンを、カスタム CDN の作成に使用できる同期可能な形式にエクスポートできます。コンテンツビューをエクスポートした後、次のいずれかを実行できます。

- HTTP/HTTPS 経由でカスタム CDN からコンテンツを同期します。
- **Hammer content-import** を使用してコンテンツをインポートします。これには、エクスポートサーバーとインポートサーバーの両方で Satellite 6.15 を実行する必要があることに注意してください。

その後、インポートする Satellite Server または別の Satellite Server 組織のローカル Web サーバーを使用して、生成されたコンテンツを提供できます。

Syncable Format エクスポートを直接インポートすることはできません。代わりに、インポートする Satellite Server で以下を行う必要があります。

- 生成されたコンテンツを、インポートする Satellite Server がアクセスできる HTTP/HTTPS Web サーバーにコピーします。
- CDN 設定を **Custom CDN** に更新します。
- Web サーバーを指すように CDN URL を設定します。
- オプション: Web サーバーで必要な場合は、SSL/TLS CA 認証情報を設定します。
- リポジトリーを有効にします。
- リポジトリーを同期します。

以下のコンテンツを、Satellite Server から同期可能な形式でエクスポートできます。

- Yum リポジトリー
- Kickstart リポジトリー
- ファイルリポジトリー

Ansible、Deb、Docker コンテンツをエクスポートすることはできません。

エクスポートには、インポートする Satellite Server での同期に使用できる Yum 形式のリポジトリーのパッケージ、**リスト** ファイル、およびメタデータを含むディレクトリーが含まれます。

前提条件

- エクスポートするコンテンツビュー内のすべてのリポジトリーのダウンロードポリシーが、**即時** に設定されていることを確認します。詳細は、「ダウンロードポリシーの概要」 を参照してください。
- エクスポートする製品が、必要な日付に同期されることを確認します。
- コンテンツをエクスポートするユーザーに **Content Exporter** ロールがあることを確認します。

コンテンツビューバージョンをエクスポートする方法

- エクスポート可能なコンテンツビューのバージョンをリスト表示します。

```
# hammer content-view version list \
--content-view="My_Content_View" \
--organization="My_Organization"
```

手順

1. 必要なバージョンのバージョン番号を取得します。以下の例は、エクスポート用のバージョン **1.0** を対象としています。

```
# hammer content-export complete version \
--content-view="Content_View_Name" \
--version=1.0 \
--organization="My_Organization" \
--format=syncable
```

2. オプション: エクスポートされたコンテンツがエクスポートディレクトリーにあることを確認します。

```
# ls -lh /var/lib/pulp/exports/My_Organization/My_Content_View_Name/1.0/2021-02-25T18-59-26-00-00/
```

8.9. コンテンツビューバージョンの増分エクスポート

コンテンツビューの完全なバージョンをエクスポートすると、システムリソースの点で非常にコストがかかる操作になる可能性があります。複数の Red Hat Enterprise Linux ツリーを持つコンテンツビューバージョンは、Satellite Server で数ギガバイトの領域を占める可能性があります。

このような場合は、前回のエクスポート以降に変更されたコンテンツのみを含む増分エクスポートを作成できます。増分エクスポートでは、通常、完全エクスポートよりもアーカイブファイルが小さくなります。

手順

1. 増分エクスポートを作成します。

```
# hammer content-export incremental version \
--content-view="My_Content_View" \
--organization="My_Organization" \
--version="My_Content_View_Version"
```

同期可能なエクスポートを作成する場合は、**--format=syncable** を追加します。デフォルトでは、Satellite はインポート可能なエクスポートを作成します。

次のステップ

- オプション: エクスポートしたコンテンツビューを表示します。

```
# find
/var/lib/pulp/exports/My_Organization/My_Exported_Content_View/My_Content_View_Version/
```

- エクスポートしたコンテンツビューバージョンを Satellite Server にインポートできます。詳細は、「コンテンツビューバージョンのインポート」 を参照してください。

8.10. リポジトリーのエクスポート

組織のライブラリー環境にあるリポジトリーのコンテンツを Satellite Server からエクスポートできます。このアーカイブファイルを使用して、別の Satellite Server または別の Satellite Server 組織に同じリポジトリーを作成できます。

次のコンテンツを Satellite Server からエクスポートできます。

- Ansible リポジトリー
- Kickstart リポジトリー
- Yum リポジトリー
- ファイルリポジトリー
- Docker コンテンツ

エクスポートには次のデータが含まれます。

- リポジトリーメタデータを含む 2 つの JSON ファイル。
- 組織のライブラリー環境からのリポジトリーのコンテンツを含む 1 つ以上のアーカイブファイル。

インポートするには、すべてのファイル **tar.gz**、**toc.json**、**metadata.json** が必要です。

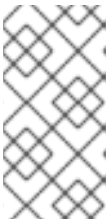
前提条件

- エクスポートディレクトリーに、エクスポートに対応できる十分な空き容量があることを確認します。
- **/var/lib/pulp/exports** ディレクトリーに、エクスポートするすべてのリポジトリーのサイズに相当する十分な空きストレージスペースがあることを確認してください。
- エクスポートするライブラリーライフサイクル環境内のリポジトリーでダウンロードポリシーを **即時** に設定していることを確認する。詳細は、「ダウンロードポリシーの概要」 を参照してください。
- エクスポートする製品が、必要な日付に同期されることを確認する。

手順

1. 製品名およびリポジトリー名を使用してリポジトリーをエクスポートします。

```
# hammer content-export complete repository \
--name="My_Repository" \
--organization="My_Organization" \ --product="_My_Product"
```



注記

エクスポートされたアーカイブのサイズは、リポジトリー内のパッケージの数とサイズによって異なります。エクスポートされたアーカイブをチャンクに分割する場合は、**--chunk-size-gb** 引数を使用してリポジトリーをエクスポートし、**--chunk-size-gb=2** のようにサイズをギガバイト単位の整数値で制限します。

2. オプション: エクスポートされたアーカイブがエクスポートディレクトリーにあることを確認します。

```
# ls -lh /var/lib/pulp/exports/My_Organization/Export-My_Repository/1.0/2022-09-02T03-35-24-00-00/
```

8.11. 同期可能な形式でのリポジトリーのエクスポート

組織のライブラリー環境にあるリポジトリーのコンテンツを、カスタム CDN を作成し、HTTP/HTTPS を介してカスタム CDN からコンテンツを同期するために使用できる同期可能な形式にエクスポートできます。

その後、インポートする Satellite Server または別の Satellite Server 組織のローカル Web サーバーを使用して、生成されたコンテンツを提供できます。

Syncable Format エクスポートを直接インポートすることはできません。代わりに、インポートする Satellite Server で以下を行う必要があります。

- 生成されたコンテンツを、インポートする Satellite Server がアクセスできる HTTP/HTTPS Web サーバーにコピーします。
- CDN 設定を **Custom CDN** に更新します。
- Web サーバーを指すように CDN URL を設定します。
- オプション: Web サーバーで必要な場合は、SSL/TLS CA 認証情報を設定します。
- リポジトリーを有効にします。
- リポジトリーを同期します。

以下のコンテンツを、Satellite Server から同期可能な形式でエクスポートできます。

- Yum リポジトリー
- Kickstart リポジトリー
- ファイルリポジトリー

Ansible、Deb、Docker コンテンツをエクスポートすることはできません。

エクスポートには、インポートする Satellite Server での同期に使用できる Yum 形式のリポジトリーのパッケージ、**リスト** ファイル、およびメタデータを含むディレクトリーが含まれます。

前提条件

- エクスポートするライブラリーライフサイクル環境内のリポジトリーでダウンロードポリシーを **即時** に設定していることを確認する。詳細は、「ダウンロードポリシーの概要」 を参照してください。

手順

1. リポジトリー名または ID を使用してリポジトリーをエクスポートします。

```
# hammer content-export complete repository \
--organization="My_Organization" \
--product="My_Product" \
--name="My_Repository" \
--format=syncable
```

2. オプション: エクスポートされたコンテンツがエクスポートディレクトリーにあることを確認します。

```
# du -sh /var/lib/pulp/exports/My_Organization/Export-My_Repository/1.0/2021-03-02T03-35-24-00-00
```

8.12. リポジトリーの増分エクスポート

リポジトリーのエクスポートは、システムリソースの観点から非常にコストのかかる操作になる可能性があります。一般的な Red Hat Enterprise Linux ツリーは、Satellite Server 上で数ギガバイトのスペースを占める場合があります。

このような場合は、**インクリメンタルエクスポート** を使用して、前回のエクスポート以降に変更されたコンテンツのみをエクスポートできます。増分エクスポートでは、通常、完全エクスポートよりもアーカイブファイルが小さくなります。

以下の例は、ライブラリーライフサイクル環境でのリポジトリーの増分エクスポートを示しています。

手順

1. 増分エクスポートを作成します。

```
# hammer content-export incremental repository \
--name="My_Repository" \
--organization="My_Organization" \
--product="My_Product"
```

2. オプション: エクスポートされたデータを表示します。

```
# ls -lh /var/lib/pulp/exports/My_Organization/Export-My_Repository/3.0/2021-03-02T03-35-24-00-00/
total 172K
-rw-r--r--. 1 pulp pulp  20M Mar  2 04:22 export-436882d8-de5a-48e9-a30a-17169318f908-20210302_0422.tar.gz
```

```
-rw-r--r--. 1 pulp pulp  333 Mar  2 04:22 export-436882d8-de5a-48e9-a30a-17169318f908-20210302_0422-toc.json
-rw-r--r--. 1 root root  492 Mar  2 04:22 metadata.json
```

8.13. 同期可能な形式でのリポジトリーの増分エクスポート

リポジトリーのエクスポートは、システムリソースの観点から非常にコストのかかる操作になる可能性があります。一般的な Red Hat Enterprise Linux ツリーは、Satellite Server 上で数ギガバイトのスペースを占める場合があります。

このような場合は、**インクリメンタルエクスポート** を使用して、前回のエクスポート以降に変更されたコンテンツのみをエクスポートできます。通常、増分エクスポートでは、完全エクスポートよりもアーカイブファイルが小さくなります。

以下の手順は、ライブラリーライフサイクル環境でのリポジトリーの増分エクスポートを示しています。

手順

1. 増分エクスポートを作成します。

```
# hammer content-export incremental repository \
--format=syncable \
--name="My_Repository" \
--organization="My_Organization" \
--product="My_Product"
```

2. オプション: エクスポートされたデータを表示します。

```
# find /var/lib/pulp/exports/Default_Organization/My_Product/2.0/2023-03-09T10-55-48-05-00/ -name "*.rpm"
```

8.14. エクスポートの追跡

Satellite はすべてのエクスポートの記録を保持します。アップストリームの Satellite Server でコンテンツをエクスポートするたびに、エクスポートは記録され、将来のクエリーのために維持されます。レコードを使用して、エクスポートを整理および管理できます。これは、特に増分的にエクスポートする場合に役立ちます。

アップストリームの Satellite Server からいくつかのダウンストリームの Satellite Server 用にコンテンツをエクスポートする場合、特定のサーバー用にエクスポートされたコンテンツを追跡することもできます。これにより、どのコンテンツがどこにエクスポートされたかを追跡できます。

エクスポート中に **--destination-server** 引数を使用して、ターゲットサーバーを指定します。このオプションは、すべての **content-export** 操作で利用できます。

ライブラリーのエクスポート先の追跡

- ライブラリーをエクスポートするときの宛先サーバーを指定します。

```
# hammer content-export complete library \
--destination-server=My_Downstream_Server_1 \
--organization="My_Organization" \
```

```
--version=1.0
```

コンテンツビューのエクスポート先の追跡

- コンテンツビューバージョンをエクスポートするときの宛先サーバーを指定します。

```
# hammer content-export complete version \
--content-view="Content_View_Name" \
--destination-server=My_Downstream_Server_1 \
--organization="My_Organization" \
--version=1.0
```

エクスポートレコードのクエリー

- 次のコマンドを使用して、コンテンツエクスポートをリスト表示します。

```
# hammer content-export list --organization="My_Organization"
```

8.15. ライブラリー環境へのインポート

エクスポートされたライブラリーのコンテンツを、別の Satellite Server 上の組織のライブラリーのライフサイクル環境にインポートできます。ライブラリー環境からコンテンツをエクスポートする方法の詳細については、「ライブラリー環境のエクスポート」 を参照してください。

前提条件

- エクスポートファイルは、**/var/lib/pulp/imports** 配下のディレクトリーにある必要があります。
- エクスポートされたコンテンツに Red Hat リポジトリーがある場合、インポートする組織のマニフェストには、エクスポート内に含まれる製品のサブスクリプションが含まれている。
- コンテンツをインポートするユーザーに、**Content Importer** ロールを割り当てる。

手順

1. エクスポートしたファイルを、インポート先の Satellite Server の **/var/lib/pulp/imports** のサブディレクトリーにコピーします。
2. インポートディレクトリーとその内容の所有権を **pulp:pulp** に設定します。

```
# chown -R pulp:pulp /var/lib/pulp/imports/2021-03-02T03-35-24-00-00
```

3. 所有権が正しく設定されていることを確認します。

```
# ls -lh /var/lib/pulp/imports/2021-03-02T03-35-24-00-00
total 68M
-rw-r--r--. 1 pulp pulp 68M Mar  2 04:29 export-1e25417c-6d09-49d4-b9a5-23df4db3d52a-
20210302_0335.tar.gz
-rw-r--r--. 1 pulp pulp 333 Mar  2 04:29 export-1e25417c-6d09-49d4-b9a5-23df4db3d52a-
20210302_0335-toc.json
-rw-r--r--. 1 pulp pulp 443 Mar  2 04:29 metadata.json
```

1. インポート先の組織を特定します。
2. ライブラリーコンテンツを Satellite Server にインポートするには、次のコマンドを入力します。

```
# hammer content-import library \
--organization="My_Organization" \
--path=/var/lib/pulp/imports/2021-03-02T03-35-24-00-00
```

フルパス **/var/lib/pulp/imports/My_Exported_Library_Dir** を入力する必要があることに注意してください。相対パスは機能しません。

3. ライブラリーコンテンツのインポートを確認するには、製品およびリポジトリーのコンテンツを確認します。ターゲット組織に **Import-Library** というコンテンツビューが新たに作成されます。このコンテンツビューは、ライブラリーコンテンツのインポートを容易にするために使用されます。
デフォルトでは、このコンテンツビューは Satellite Web UI に表示されません。**Import-Library** は、ホストに直接割り当てることを意図したものではありません。代わりに、通常どおり、ホストを **Default Organization View** または別のコンテンツビューに割り当てます。

8.16. WEB サーバーからライブラリー環境へのインポート

エクスポートされたライブラリーコンテンツを Web サーバーから別の Satellite Server にある組織のライブラリーライフサイクル環境に直接インポートできます。ライブラリー環境からコンテンツをエクスポートする方法の詳細については、「ライブラリー環境のエクスポート」 を参照してください。

前提条件

- エクスポートされたファイルが同期可能な形式である。
- エクスポートされたファイルに HTTP/HTTPS 経由でアクセスできる。
- エクスポートされたコンテンツに Red Hat リポジトリーがある場合、インポートする組織のマニフェストには、エクスポート内に含まれる製品のサブスクリプションが含まれている。
- コンテンツビューバージョンをインポートするユーザーに **Content Importer** ロールが割り当てられている。

手順

1. インポート先の組織を特定します。
2. ライブラリーコンテンツを Satellite Server にインポートするには、次のコマンドを入力します。

```
# hammer content-import library \
--organization="My_Organization" \
--path=http://server.example.com/pub/exports/2021-02-25T21-15-22-00-00/
```

ターゲット組織に **Import-Library** というコンテンツビューが新たに作成されます。このコンテンツビューは、ライブラリーコンテンツのインポートを容易にするために使用されます。

デフォルトでは、このコンテンツビューは Satellite Web UI に表示されません。**Import-Library** は、ホストに直接割り当てることを意図したものではありません。代わりに、ホストを **Default Organization View** または別のコンテンツビューに割り当てます。

8.17. コンテンツビューバージョンのインポート

エクスポートされたコンテンツビューバージョンをインポートして、別の Satellite Server 上の組織に同じコンテンツを持つバージョンを作成できます。コンテンツビューバージョンのエクスポートの詳細は、「コンテンツビューバージョンのエクスポート」 を参照してください。

コンテンツビューバージョンをインポートすると、メジャーバージョン番号、マイナーバージョン番号が同じで、同じパッケージとエラータを含む同じリポジトリーが含まれます。カスタムリポジトリー、製品、およびコンテンツビューは、インポートする組織に存在しない場合は自動的に作成されます。

前提条件

- エクスポートファイルは、**/var/lib/pulp/imports** 配下のディレクトリーにある必要があります。
- エクスポートされたコンテンツに Red Hat リポジトリーがある場合、インポートする組織のマニフェストには、エクスポート内に含まれる製品のサブスクリプションが含まれている。
- コンテンツビューバージョンをインポートするユーザーに **Content Importer** ロールが割り当てられている。

手順

1. エクスポートしたファイルを、インポート先の Satellite Server の **/var/lib/pulp/imports** のサブディレクトリーにコピーします。
2. インポートディレクトリーとその内容の所有権を **pulp:pulp** に設定します。

```
# chown -R pulp:pulp /var/lib/pulp/imports/2021-02-25T21-15-22-00-00/
```

3. 所有権が正しく設定されていることを確認します。

```
# ls -lh /var/lib/pulp/imports/2021-02-25T21-15-22-00-00/
```

4. コンテンツビューバージョンを Satellite Server にインポートするには、次のコマンドを入力します。

```
# hammer content-import version \
--organization=My_Organization \
--path=/var/lib/pulp/imports/2021-02-25T21-15-22-00-00/
```

フルパス **/var/lib/pulp/imports/My_Exported_Version_Dir** を入力する必要があることに注意してください。相対パスは機能しません。

5. コンテンツビューバージョンが正常にインポートされたことを確認するには、組織のコンテンツビューバージョンをリスト表示します。

```
# hammer content-view version list \
--organization-id=My_Organization_ID
```

8.18. WEB サーバーからのコンテンツビューバージョンのインポート

エクスポートしたコンテンツビューバージョンを Web サーバーから直接インポートして、別の Satellite Server 上の組織内に同じコンテンツを含むバージョンを作成できます。コンテンツビューバー

ジョンのエクスポートの詳細は、「コンテンツビューバージョンのエクスポート」 を参照してください。

コンテンツビューバージョンをインポートすると、メジャーバージョン番号、マイナーバージョン番号が同じで、同じパッケージとエラータを含む同じリポジトリーが含まれます。カスタムリポジトリー、製品、およびコンテンツビューは、インポートする組織に存在しない場合は自動的に作成されます。

前提条件

- エクスポートされたファイルが同期可能な形式である。
- エクスポートされたファイルに HTTP/HTTPS 経由でアクセスできる。
- エクスポートされたコンテンツに Red Hat リポジトリーがある場合、インポートする組織のマニフェストには、エクスポート内に含まれる製品のサブスクリプションが含まれている。
- コンテンツビューバージョンをインポートするユーザーに **Content Importer** ロールが割り当てられている。

手順

- コンテンツビューバージョンを Satellite Server にインポートします。

```
# hammer content-import version \
--organization=My_Organization \
--path=http://server.example.com/pub/exports/2021-02-25T21-15-22-00-00/
```

8.19. リポジトリーのインポート

エクスポートされたリポジトリーを別の Satellite Server の組織にインポートできます。リポジトリーのコンテンツのエクスポートの詳細については、「リポジトリーのエクスポート」 を参照してください。

前提条件

- エクスポートファイルが **/var/lib/pullp/imports** の下のディレクトリーにある。
- エクスポートに Red Hat リポジトリーが含まれている場合、インポート組織のマニフェストには、エクスポートに含まれる製品のサブスクリプションが含まれている。
- コンテンツをインポートするユーザーに、**Content Importer** ロールを割り当てる。

手順

1. エクスポートしたファイルを、インポート先の Satellite Server の **/var/lib/pulp/imports** のサブディレクトリーにコピーします。
2. インポートディレクトリーとその内容の所有権を **pulp:pulp** に設定します。

```
# chown -R pulp:pulp /var/lib/pulp/imports/2021-03-02T03-35-24-00-00
```

3. 所有権が正しく設定されていることを確認します。

```
# ls -lh /var/lib/pulp/imports/2021-03-02T03-35-24-00-00
```

```
total 68M
-rw-r--r--. 1 pulp pulp 68M Mar  2 04:29 export-1e25417c-6d09-49d4-b9a5-23df4db3d52a-
20210302_0335.tar.gz
-rw-r--r--. 1 pulp pulp 333 Mar  2 04:29 export-1e25417c-6d09-49d4-b9a5-23df4db3d52a-
20210302_0335-toc.json
-rw-r--r--. 1 pulp pulp 443 Mar  2 04:29 metadata.json
```

1. インポート先の組織を特定します。
2. リポジトリーのコンテンツを Satellite Server にインポートするには、次のコマンドを入力します。

```
# hammer content-import repository \
--organization="My_Organization" \
--path=/var/lib/pulp/imports/2021-03-02T03-35-24-00-00
```

フルパス **/var/lib/pulp/imports/My_Exported_Repo_Dir** を入力する必要があることに注意してください。相対パスは機能しません。

3. リポジトリーをインポートしたことを確認するには、製品とリポジトリーの内容を確認してください。

8.20. WEB サーバーからのリポジトリーのインポート

エクスポートされたリポジトリーを Web サーバーから別の Satellite Server 上の組織に直接インポートできます。リポジトリーのコンテンツのエクスポートの詳細は、「リポジトリーのエクスポート」 を参照してください。

前提条件

- エクスポートされたファイルが同期可能な形式である。
- エクスポートされたファイルに HTTP/HTTPS 経由でアクセスできる。
- エクスポートに Red Hat リポジトリーが含まれている場合、インポート組織のマニフェストには、エクスポートに含まれる製品のサブスクリプションが含まれている。
- コンテンツビューバージョンをインポートするユーザーに **Content Importer** ロールが割り当てられている。

手順

1. インポート先の組織を選択します。
2. リポジトリーを Satellite Server にインポートするには、次のコマンドを入力します。

```
# hammer content-import repository \
--organization="My_Organization" \
--path=http://server.example.com/pub/exports/2021-02-25T21-15-22-00-00/
```

8.21. HAMMER CLI チートシートを使用したコンテンツのエクスポートとインポート

表8.1 Export

目的	コマンド
組織のライブラリーの完全なエクスポート	**hammer content-export complete library --organization="My_Organization"**
組織のライブラリーの増分エクスポート (すでに何かをエクスポート済みであることが前提)。	**hammer content-export incremental library --organization="My_Organization"**
コンテンツビューバージョンの完全なエクスポート	**hammer content-export complete version --content-view="My_Content_View" --version=1.0 --organization="My_Organization"**
開発環境にプロモートされたコンテンツビューバージョンをエクスポートします。	**hammer content-export complete version --content-view="My_Content_View" --organization="My_Organization" --lifecycle-environment="Dev"**
小規模なチャンクでのコンテンツビューのエクスポート (2 GB スラブ)。	**hammer content-export complete version --content-view="My_Content_View" --version=1.0 --organization="My_Organization" --chunk-size-gb=2**
コンテンツビューバージョンの増分エクスポート (すでに何かをエクスポート済みであることが前提)。	**hammer content-export incremental version --content-view="My_Content_View" --version=2.0 --organization="My_Organization"**
リポジトリーの完全なエクスポート	**hammer content-export complete repository --product="My_Product" --name="My_Repository" --organization="My_Organization"**
リポジトリーの増分エクスポート (すでに何かをエクスポート済みであることが前提)	**hammer content-export incremental repository --product="My_Product" --name="My_Repository" --organization="My_Organization"**
エクスポートのリスト表示	**hammer content-export list --content-view="My_Content_View" --organization="My_Organization"**

表8.2 インポート

目的	コマンド
組織のライブラリーへのインポート	**hammer content-import library --organization="My_Organization" --path="/var/lib/pulp/imports/My_Exported_Library_Dir"**
コンテンツビューバージョンにインポートします。	**hammer content-import version --organization="My_Organization" --path="/var/lib/pulp/imports/My_Exported_Version_Dir"**

目的	コマンド
リポジトリーのインポート	**hammer content-import repository --organization="My_Organization" --path="/var/lib/pulp/imports/My_Exported_Repo_Dir"**

第9章 アクティベーションキーの管理

アクティベーションキーは、システム登録とサブスクリプションのアタッチを自動化する方法を提供します。複数のキーを作成して、異なる環境とコンテンツビューに関連付けることができます。たとえば、Red Hat Enterprise Linux ワークステーション用のサブスクリプションで基本のアクティベーションキーを作成し、これを特定の環境のコンテンツビューに関連付けることができます。

重要

Satellite で Simple Content Access (SCA) が有効になっている場合は、アクティベーションキーにサブスクリプションをアタッチすることはできません。SCA を有効にすると、ホストにサブスクリプションを接続する必要がなくなります。新しく作成された組織では、SCA がデフォルトで有効になっていることに注意してください。SCA の詳細は、Simple Content Access を参照してください。

コンテンツホストの登録時にアクティベーションキーを使用して、プロセスにかかる時間を短縮するだけでなく、プロセスの簡潔性や一貫性を向上します。アクティベーションキーは、ホストが登録されている場合にのみ使用できます。アクティベーションキーに変更が加えられた場合には、それ以降、改訂されたアクティベーションキーで登録されるホストにだけ変更が適用されます。これらの変更は、既存のホストには加えられません。

アクティベーションキーを使用して、コンテンツホストの次のプロパティーを定義できます。

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

ホストの作成と登録の間のコンテンツビューの競合

ホストをプロビジョニングすると、Satellite は、ホストグループまたはホスト設定で設定したコンテンツビューからのプロビジョニングテンプレートとその他のコンテンツを使用します。ホストの登録時に、アクティベーションキーからのコンテンツビューが、ホストグループまたはホスト設定からの元のコンテンツビューを上書きします。次に、Satellite は、ホストの再ビルドなど、今後のすべてのタスクにアクティベーションキーからのコンテンツビューを使用します。

ホストの再ビルド時に、使用するコンテンツビューを、ホストグループやホスト設定ではなく、アクティベーションキーで設定するようにします。

複数のコンテンツホストでの同じアクティベーションキーの使用

サブスクリプションが十分にある場合には、同じアクティベーションキーを複数のコンテンツホストに適用できます。ただし、アクティベーションキーはコンテンツホストの初期設定のみを行います。コンテンツホストを組織に登録した後は、組織のコンテンツをコンテンツホストに手動でアタッチできます。

コンテンツホストでの複数のアクティベーションキーの使用

コンテンツホストは、複数のアクティベーションキーを関連付けることで、組み合わせてホストを設定できます。設定の競合が発生した場合には、最後に指定したアクティベーションキーが優先されます。以下のようにホストグループのパラメーターを設定して優先順位を指定できます。

■

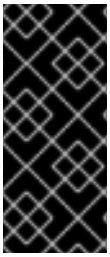
```
$ hammer hostgroup set-parameter \
--hostgroup "My_Host_Group" \
--name "My_Activation_Key" \
--value "name_of_first_key", "name_of_second_key", ...
```

9.1. アクティベーションキーのベストプラクティス

- ユースケースごとにアクティベーションキーを作成します。これにより、ホスト上のコンテンツの管理が構造化、モジュール化、簡素化されます。
- コンテンツとライフサイクル環境がわかる命名規則をアクティベーションキーに使用します (例: **red-hat-enterprise-linux-webserver**)。
- Hammer スクリプトまたは Ansible Playbook を使用してアクティベーションキーの管理を自動化します。

9.2. アクティベーションキーの作成

アクティベーションキーを使用して、登録時にホストにアタッチするサブスクリプションの特定のセットを定義できます。アクティベーションキーに追加するサブスクリプションは、関連するコンテンツビュー内で利用可能である必要があります。



重要

Satellite で Simple Content Access (SCA) が有効になっている場合は、アクティベーションキーにサブスクリプションをアタッチすることはできません。SCA を有効にすると、ホストにサブスクリプションを接続する必要がなくなります。新しく作成された組織では、SCA がデフォルトで有効になっていることに注意してください。SCA の詳細は、Simple Content Access を参照してください。

サブスクリプションマネージャーは、以下の要因に応じたさまざまな方法で、サブスクリプションをアタッチします。

- サブスクリプションがアクティベーションキーに関連付けられているか？
- 自動アタッチオプションは有効になっているか？
- Red Hat Enterprise Linux 8 ホストの場合: アクティベーションキーにシステムの目的が設定されていますか？

Satellite は、ホストにインストールされている製品に対してのみサブスクリプションを自動的にアタッチします。拡張更新サポート (EUS) など、デフォルトで Red Hat Enterprise Linux にインストールされていない製品が記載されていないサブスクリプションについては、必要なサブスクリプションを指定したアクティベーションを使用し、自動アタッチを無効にします。

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

1. サブスクリプションを自動的にアタッチするアクティベーションキー。
   サブスクリプションの指定なしで、自動アタッチが有効な場合に、アクティベーションキーを使用するホストは、アクティベーションキーと関連するコンテンツビューが提供する最適なサブスクリプションを検索します。これは、**subscription-manager --auto-attach** コマンドを実

行する場合と類似しています。Red Hat Enterprise Linux 8 ホストの場合、アクティベーションキーを設定し、登録時にシステムの目的をホストに設定して、サブスクリプションの自動アタッチメントを強化できます。

2. 自動アタッチ用にカスタムのサブスクリプションを指定するアクティベーションキー。
サブスクリプションが指定されていて、自動アタッチが有効な場合に、アクティベーションキーを使用するホストは、アクティベーションキーで指定されたリストから最適なサブスクリプションを選択します。アクティベーションキーにシステムの目的を設定しても、このシナリオには影響しません。

3. サブスクリプションセットが指定されたアクティベーションキー。
サブスクリプションが指定されていて、自動アタッチが無効な場合、アクティベーションキーを使用するホストは、アクティベーションキーに指定されたすべてのサブスクリプションに関連付けられます。アクティベーションキーにシステムの目的を設定しても、このシナリオには影響しません。

カスタム製品

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

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

手順

1. Satellite Web UI で **Content** > **Lifecycle** > **Activation Keys** に移動して、**Create Activation Key** をクリックします。

2. **Name** フィールドに、アクティベーションキーの名前を入力します。

3. 制限を設定する場合は、**Unlimited hosts** のチェックボックスをオフにし、**Limit** フィールドに、アクティベーションキーで登録できるシステムの最大数を入力します。アクティベーションキーを使用して登録するホストに制限を設けない場合は、**Unlimited Hosts** チェックボックスが選択されていることを確認します。

4. オプション: **Description** フィールドに、アクティベーションキーの説明を入力します。

5. **Environment** リストから、使用する環境を選択します。

6. **Content View** リストから、使用するコンテンツビューを選択します。

7. Simple Content Access (SCA) が有効な場合:

    a. **リポジトリーセット** タブで、指定したリポジトリーのみが有効になっていることを確認します。

8. SCA が有効になっていない場合は、以下のようになります。

    a. **サブスクリプション** タブをクリックしてから、**追加** サブメニューをクリックします。

    b. 事前に作成したサブスクリプションのチェックボックスをクリックします。

    c. **Add Selected** をクリックします。

9. **Save** をクリックします。

10. オプション: Red Hat Enterprise Linux 8 ホストの場合、**システム目的** セクションで、システム目的でアクティベーションキーを設定し、登録時にホストに設定して、サブスクリプションの自動アタッチメントを強化できます。

CLI 手順

1. アクティベーションキーを作成します。

```
# hammer activation-key create \
--name "My_Activation_Key" \
--unlimited-hosts \
--description "Example Stack in the Development Environment" \
--lifecycle-environment "Development" \
--content-view "Stack" \
--organization "My_Organization"
```

2. オプション: Red Hat Enterprise Linux 8 ホストの場合、以下のコマンドを入力して、システムの目的でアクティベーションキーを設定し、登録時にホストに設定してサブスクリプションの自動アタッチメントを強化します。

```
# hammer activation-key update \
--organization "My_Organization" \
--name "My_Activation_Key" \
--service-level "Standard" \
--purpose-usage "Development/Test" \
--purpose-role "Red Hat Enterprise Linux Server" \
--purpose-addons "addons"
```

3. サブスクリプション ID リストを取得します。

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

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

```
# hammer activation-key add-subscription \
--name "My_Activation_Key" \
--subscription-id My_Subscription_ID \
--organization "My_Organization"
```

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

   a. Simple Content Access (SCA) が有効な場合:

```
# hammer activation-key product-content \
--content-access-mode-all true \
--name "My_Activation_Key" \
--organization "My_Organization"
```

   b. SCA が有効になっていない場合は、以下のようになります。

```
# hammer activation-key product-content \
--name "My_Activation_Key" \
--organization "My_Organization"
```

6. Satellite Client 6 リポジトリーのデフォルトの自動有効化ステータスをオーバーライドします。デフォルトのステータスは無効に設定されています。有効にするには、以下のコマンドを実行します。

```
# hammer activation-key content-override \
--name "My_Activation_Key" \
--content-label rhel-7-server-satellite-client-6-rpms \
--value 1 \
--organization "My_Organization"
```

9.3. アクティベーションキーに関連付けられたサブスクリプションの更新

重要

この手順は、Satellite で Simple Content Access (SCA) が無効になっている場合にのみ有効です。SCA を有効にすると、ホストにサブスクリプションを接続する必要がなくなります。新しく作成された組織では、SCA がデフォルトで有効になっていることに注意してください。SCA の詳細は、Simple Content Access を参照してください。

以下の手順を使用して、アクティベーションキーに関連付けられたサブスクリプションを変更します。Satellite Web UI の代わりに CLI を使用する場合は、CLI 手順 を参照してください。

アクティベーションキーへの変更は、変更後にプロビジョニングしたマシンにのみ適用されます。既存のコンテンツホストでサブスクリプションを更新する方法は 「複数のホストでの Red Hat サブスクリプションの更新」 を参照してください。

手順

1. Satellite Web UI で、**Content** > **Lifecycle** > **Activation Keys** に移動し、アクティベーションキーの名前をクリックします。
2. **Subscriptions** タブをクリックします。
3. サブスクリプションを削除するには **List/Remove** を選択してから、削除するサブスクリプションの左側にあるチェックボックスを選択し、**Remove Selected** をクリックします。
4. サブスクリプションを追加するには、**Add** を選択してから、追加するサブスクリプションの左側のチェックボックスを選択し、**Add Selected** をクリックします。
5. **リポジトリーセット** タブをクリックし、リポジトリーのステータス設定を確認します。
6. リポジトリーを有効または無効にするには、リポジトリーのチェックボックスを選択してから、**Select Action** リストを使用してステータスを変更します。
7. **Details** タブをクリックし、このアクティベーションキーのコンテンツビューを選択し、**Save** をクリックします。

CLI 手順

1. 現在アクティベーションキーが含まれているサブスクリプションをリスト表示します。

```
# hammer activation-key subscriptions \
--name My_Activation_Key \
--organization "My_Organization"
```

2. アクティベーションキーから必要なサブスクリプションを削除します。

```
# hammer activation-key remove-subscription \
--name "My_Activation_Key" \
--subscription-id ff808181533518d50152354246e901aa \
--organization "My_Organization"
```

--subscription-id オプションで、UUID またはサブスクリプションの ID のいずれかを使用できます。

3. 新しいサブスクリプションをアクティベーションキーにアタッチします。

```
# hammer activation-key add-subscription \
--name "My_Activation_Key" \
--subscription-id ff808181533518d50152354246e901aa \
--organization "My_Organization"
```

--subscription-id オプションで、UUID またはサブスクリプションの ID のいずれかを使用できます。

4. 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リスト表示します。

```
# hammer activation-key product-content \
--name "My_Activation_Key" \
--organization "My_Organization"
```

5. 必要な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。

```
# hammer activation-key content-override \
--name "My_Activation_Key" \
--content-label content_label \
--value 1 \
--organization "My_Organization"
```

有効化する場合は、**--value** オプションに **1** を、無効化する場合は **0** を入力します。

9.4. アクティベーションキーを使用したホストの登録

アクティベーションキーを使用して、以下のタスクを完了できます。

- Red Hat Satellite を使用したプロビジョニング中に新規ホストを登録する。Red Hat Satellite のキックスタートプロビジョニングテンプレートには、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ホストを登録するコマンドが含まれています。
- 既存の Red Hat Enterprise Linux ホストを登録します。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。

Satellite Web UI、Hammer CLI、または Satellite API のホスト登録機能を使用して、Satellite にホストを登録できます。詳細は、**ホストの管理** の ホストの登録 を参照してください。

手順

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

2. **Activation Keys** リストから、ホストに割り当てるアクティベーションキーを選択します。
3. **Generate** をクリックして登録コマンドを作成します。
4. **ファイル** アイコンをクリックして、コマンドをクリップボードにコピーします。
5. SSH を使用してホストに接続し、登録コマンドを実行します。
6. **/etc/yum.repos.d/redhat.repo** ファイルをチェックして、適切なリポジトリーが有効であることを確認します。

CLI 手順

1. Hammer CLI を使用してホスト登録コマンドを生成します。

```
# hammer host-registration generate-command \
--activation-keys "My_Activation_Key"
```

ホストが Satellite Server の SSL 証明書を信頼しない場合は、登録コマンドに **--insecure** フラグを追加して SSL 検証を無効にすることができます。

```
# hammer host-registration generate-command \
--activation-keys "My_Activation_Key" \
--insecure true
```

2. SSH を使用してホストに接続し、登録コマンドを実行します。
3. **/etc/yum.repos.d/redhat.repo** ファイルをチェックして、適切なリポジトリーが有効であることを確認します。

API の手順

1. Satellite API を使用してホスト登録コマンドを生成します。

```
# curl -X POST https://satellite.example.com/api/registration_commands \
--user "My_User_Name" \
-H 'Content-Type: application/json' \
-d '{ "registration_command": { "activation_keys": ["My_Activation_Key_1,
My_Activation_Key_2"] }}'
```

ホストが Satellite Server の SSL 証明書を信頼しない場合は、登録コマンドに **--insecure** フラグを追加して SSL 検証を無効にすることができます。

```
# curl -X POST https://satellite.example.com/api/registration_commands \
--user "My_User_Name" \
-H 'Content-Type: application/json' \
-d '{ "registration_command": { "activation_keys": ["My_Activation_Key_1,
My_Activation_Key_2"], "insecure": true }}'
```

アクティベーションキーを使用すると、その環境を簡単に指定できます。詳細は、**コンテンツの管理** の アクティベーションキーの管理 を参照してください。

コマンドライン引数としてパスワードを入力するには、**username:password** 構文を使用します。これにより、パスワードがシェル履歴に保存される可能性があることに注意してください。または、パスワードの代わりに一時的なパーソナルアクセストークンを使用することもで

きます。Satellite Web UI でトークンを生成するには、**My Account** > **Personal Access Tokens** に移動します。

2. SSH を使用してホストに接続し、登録コマンドを実行します。
3. **/etc/yum.repos.d/redhat.repo** ファイルをチェックして、適切なリポジトリーが有効であることを確認します。

複数のアクティベーションキー

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

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

競合の設定

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

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

9.5. 自動アタッチの有効化

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

重要

この手順は、Satellite で Simple Content Access (SCA) が無効になっている場合にのみ有効です。SCA を有効にすると、ホストにサブスクリプションを接続する必要がなくなります。新しく作成された組織では、SCA がデフォルトで有効になっていることに注意してください。SCA の詳細は、Simple Content Access を参照してください。

自動アタッチを有効にして、キーに関連付けられたサブスクリプションを持たないことができます。このタイプのキーは、仮想マシンが物理サブスクリプションを消費するのではなく、ハイパーバイザーからホストベースのサブスクリプションを継承する場合に、仮想マシンを登録するために一般的に使用されます。詳細は、**仮想マシンサブスクリプションの設定** を参照してください。

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

手順

1. Satellite Web UI で、**Content** > **Lifecycle** > **Activation Keys** に移動します。
2. 編集するアクティベーションキーの名前をクリックします。
3. **Subscriptions** タブをクリックします。
4. **Auto-Attach** の隣にある編集アイコンをクリックします。
5. チェックボックスにチェックを入れて自動アタッチを有効にするか、チェックを外して無効にします。
6. **Save** をクリックします。

CLI 手順

- アクティベーションキーで自動アタッチを有効にするには、以下のコマンドを入力します。

```
# hammer activation-key update --name "My_Activation_Key" \
--organization "My_Organization" --auto-attach true
```

9.6. サービスレベルの設定

アクティベーションキーで作成した新規ホストのデフォルトのサービスレベルを定義するように、アクティベーションキーを設定できます。デフォルトのサービスレベルを設定すると、ホストにアタッチするのに適したサブスクリプションのみが選択されます。たとえば、アクティベーションキーのデフォルトのサービスレベルが Premium に設定されている場合には、Premium サービスレベルのサブスクリプションのみが、登録時にホストにアタッチされます。

手順

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

CLI 手順

- アクティベーションキーでサービスレベルを Premium に設定します。

```
# hammer activation-key update \
--name "My_Activation_Key" \
--organization "My_Organization" \
--service-level premium
```

9.7. アクティベーションキーでのリポジトリーの有効化と無効化

Simple Content Access (SCA) ユーザーは、Satellite Web UI を使用して、アクティベーションキーでリポジトリーを有効または無効にできます。

手順

1. Satellite Web UI で、**Content** > **Lifecycle** > **Activation Keys** に移動します。
2. アクティベーションキーを選択します。
3. **Repository Sets** タブを選択します。
4. 必要に応じて、ドロップダウンから **Repository type** 列を **Custom** または **Red Hat** にフィルタリングできます。
5. 目的のリポジトリーを選択するか、**Select All** チェックボックスをクリックしてすべてのリポジトリーを選択します。
6. **Select Action** リストから、**Override to Enabled**、**Override to Disabled**、または **Reset to Default** を選択します。

第10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 の公式リリースの更新を提供しています。Red Hat では、更新を説明するアドバイザリーと共に、関連パッケージのグループを **エラータ** にコンパイルします。アドバイザリーには以下の 3 種類があります (重要度の高い順)。

セキュリティーアドバイザリー

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

バグ修正アドバイザリー

パッケージのバグ修正を説明。

製品の機能拡張アドバイザリー

パッケージに追加された機能拡張および新機能を説明。

Red Hat Satellite 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情報をインポートします。Red Hat Satellite ではエラータを検証してフィルタリングするためのツールも提供しており、更新の管理が正確にできます。このようにして、関連のある更新を選択し、コンテンツビューから選択したコンテンツホストに伝播できます。

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

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

適用可能

1 つ以上のコンテンツホストに適用されるエラータ。つまり、コンテンツホストに存在するパッケージを更新します。これらのエラータはコンテンツホストに適用されますが、状態が **インストール可能** に変わるまでは、エラータのインストール準備はできていません。インストール可能なエラータは自動的に適用されます。

インストール可能

1 つ以上のコンテンツホストに適用され、コンテンツホストにインストールできるエラータ。インストール可能なエラータは、ライフサイクル環境および関連するコンテンツビューからコンテンツホストで利用できますが、まだインストールされていません。

本章では、エラータの管理方法と 1 つのホストまたは複数のホストへの適用方法を説明します。

10.1. エラータのベストプラクティス

- エラータを使用すると、影響を受けない他のパッケージを無駄に更新することなく、凍結中のコンテンツセットにセキュリティー問題に対するパッチを追加できます。
- Hammer スクリプトまたは Ansible Playbook を使用して、エラータ管理を自動化します。
- コンテンツホストページでエラータを表示し、現在のコンテンツビューとライフサイクル環境のエラータを、最新の同期済みパッケージを含む Library ライフサイクル環境と比較します。適用できるエラータは、ホストのライフサイクルのコンテンツビューバージョンに含まれるエラータだけです。適用可能なエラータを推奨事項として表示し、増分コンテンツビューを作成してホストにエラータを提供できます。詳細は、「増分コンテンツビューへのエラータの追加」 を参照してください。

10.2. 利用可能なエラータの検出

以下の手順では、利用可能なエラータを表示し、フィルタリングする方法や、選択したアドバイザリーのメタデータを表示する方法を説明します。Satellite Web UI の代わりに CLI を使用する場合は、CLI 手順 を参照してください。

手順

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

     ```
     parameter operator value
     ```

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

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

     Enter を押して検索を開始します。
3. 調べるエラータの **Errata ID** をクリックします。
   - **説明** タブには、更新されたパッケージの説明や、更新によって提供される重要な修正および機能拡張が記載されています。
   - **コンテンツホスト** タブでは、「複数ホストへのエラータの適用」で説明したように、選択したコンテンツホストにエラータを適用できます。
   - **リポジトリー** タブには、エラータが含まれているリポジトリーのリストが表示されます。リポジトリーはフィルターを使用して環境やコンテンツビューで絞り込むことができ、リポジトリー名で検索できます。

新しいホストページを使用して、使用可能なエラータを表示し、インストールするエラータを選択することもできます。

1. Satellite Web UI で、**Hosts > All Hosts** に移動し、必要なホストを選択します。
2. ホストに関連付けられたエラータがある場合、新しいホストページのインストール可能なエラータカードに、セキュリティーアドバイザリー、バグ修正、および拡張機能の内訳を示すインタラクティブな円グラフが表示されます。

3. 新しいホストページで、**Content** タブを選択します。
4. コンテンツページで、 **Errata** タブを選択します。
5. このページには、選択したホストのインストール可能なエラータが表示されます。
6. インストールするエラータのチェックボックスをクリックします。
7. リモート実行を使用するには **Apply via Remote Execution** を選択し、リモート実行をカスタマイズする場合は **Apply via customized remote execution** を選択します。
8. **Submit** をクリックします。

CLI 手順

- 全組織で利用可能なエラータを表示するには、以下のコマンドを実行します。

```
# hammer erratum list
```

- 特定のエラータの詳細を表示するには、以下のコマンドを実行します。

```
# hammer erratum info --id erratum_ID
```

- **--search** オプションを指定してクエリーを入力し、エラータを検索します。たとえば、選択した製品に適用可能なエラータで、指定したバグが含まれるものを順番に表示し、セキュリティーエラータが一番上に表示されるようにするには、以下のコマンドを入力します。

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

10.3. エラータ検索で利用できるパラメーター

パラメーター	説明	例
bug	Bugzilla 番号での検索。	**bug = 1172165**
cve	CVE 番号での検索。	**cve = CVE-2015-0235**
id	エラータ ID での検索。自動サジェストシステムにより、入力時に利用可能な ID のリストが表示されます。	**id = RHBA-2014:2004**
issued	発行日による検索。正確な日付 (Feb16,2015 など) を指定したり、キーワード (Yesterday、1 hour ago など) を使用したりできます。時間の範囲は、<演算子と>演算子を使用して指定できます。	**issued < "Jan 12,2015"**

パラメーター	説明	例
package	完全なパッケージビルド名による検索。自動サジェストシステムにより、入力時に利用可能なパッケージのリストが表示されます。	**package = glib2-2.22.5-6.el6.i686**
package_name	パッケージ名による検索。自動サジェストシステムにより、入力時に利用可能なパッケージのリストが表示されます。	**package_name = glib2**
severity	セキュリティー更新によって修正される問題の重大度による検索。**Critical**、**Important**、または **Moderate** を指定します。	**severity = Critical**
title	アドバイザリーのタイトルによる検索。	**title ~ openssl**
type	アドバイザリーのタイプによる検索。**security**、**bugfix**、または **enhancement** を指定します。	**type = bugfix**
updated	最新更新日による検索。**issued** パラメーターの同じ形式を使用できます。	**updated = "6 days ago"**

10.4. インストール可能なエラータの適用

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

手順

1. Satellite Web UI で、**Hosts > All Hosts**に移動し、必要なホストを選択します。
2. ホストに関連付けられているエラータがある場合は、新しいホストページの Installable Errata カードに表示されます。
3. **Content** タブでは、**Errata** には選択したホストのインストール可能なエラータが表示されます。
4. インストールするエラータのチェックボックスをクリックします。
5. リモート実行を使用するには、ホストに追加するエラータの横にある縦 3 つのドットのアイコンを使用して、**Apply via Remote Execution** を選択します。リモート実行をカスタマイズする場合は、**Apply via customized remote execution** を選択します。
6. **Submit** をクリックします。

10.5. エラータ通知のサブスクライブ

Satellite ユーザー向けに電子メール通知を設定することができます。ユーザーには、適用かつインストール可能なエラータのサマリーや、コンテンツビューのプロモーションに関する通知、またはリポジトリーの同期完了通知が送信されます。詳細は、**Red Hat Satellite の管理** の メール通知の設定 を参照してください。

10.6. リポジトリー依存関係の解決の制限

Satellite では、コンテンツビューに増分更新を使用すると、リポジトリーの依存関係の問題の一部が解決されます。ただし、リポジトリーレベルでの依存関係の解決は引き続き問題となります。

新しい依存関係でリポジトリーの更新が利用できるようになると、Satellite は、既存のリポジトリーパッケージで利用可能な古いバージョンがある場合でも、依存関係を解決するために、パッケージの最新バージョンを取得します。これにより、パッケージのインストール時に依存関係の解決が必要な問題がさらに発生することがあります。

シナリオ例

クライアント上のリポジトリーには、依存関係 **example_repository-libs-1.0** のパッケージ **example_repository-1.0** があります。リポジトリーには、別のパッケージ **example_tools-1.0** もあります。

セキュリティーエラータは、パッケージ **example_tools-1.1** で利用できるようになります。**example_tools-1.1** パッケージは、依存関係として **example_repository-libs-1.1** パッケージが必要です。

コンテンツビューを増分更新すると、**example_tools-1.1**、**example_tools-1.0**、**example_repository-libs-1.1** がリポジトリーに含まれます。リポジトリーには、**example_repository-1.0** と **example_repository-libs-1.0** のパッケージもあります。コンテンツビューの増分更新で、パッケージ **example_repository-1.1** が追加されなかったことに注意してください。これらすべてのパッケージは **dnf** を使用してインストールできるため、潜在的な問題は検出されません。ただし、クライアントが **example_tools-1.1** パッケージをインストールすると、**example_repository-libs-1.0** と **example_repository-libs-1.1** の両方をインストールできないため、依存関係の解決が必要な問題が発生します。

現在、この問題の回避策はありません。パッケージの基本セットからエラータが適用されるまでの期間が長く、マイナー Y リリースが増えるほど、依存関係の解決の問題が発生する可能性が高くなります。

10.7. エラータ用のコンテンツビューフィルターの作成

コンテンツフィルターを使用して、エラータを制限できます。以下のようなフィルターを使用します。

- **ID**:-: 結果として表示されるリポジトリーに含めることができるように、特定のエラータを選択します。
- **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追加します。
- **タイプ**:-: バグ修正、機能拡張、セキュリティーなどのエラータのタイプを選択して追加します。

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できます。

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

前提条件

- 必要なエラータを含むリポジトリーを指定してコンテンツビューを作成している。詳細は、「コンテンツビューの作成」 を参照してください。

手順

1. Satellite Web UI で、**Content** > **Lifecycle** > **Content Views** に移動します。
2. エラータを適用するために使用するコンテンツビューを選択します。
3. **Yum コンテンツ** > **フィルター**を選択し、**新規フィルター**をクリックします。
4. **名前** フィールドで、**Errata Filter** を入力します。
5. **コンテンツタイプ** リストから **エラータ - 日付およびタイプ** を選択します。
6. **Inclusion Type** リストから **Exclude** を選択してください。
7. **Description** フィールドに **Exclude errata items from YYYY-MM-DD** を入力します。
8. **Save** をクリックします。
9. **Errata Type**の場合、除外するエラータタイプのチェックボックスを選択します。たとえば、特定の日付以降の機能拡張やバグ修正エラータを除外し、セキュリティーエラータすべてを含めるには、**Enhancement** および **Bugfix** のチェックボックスを選択し、**Security** のチェックボックスの選択を解除します。
10. **Date Type** の場合、2 つのチェックボックスからいずれかを選択します。
    - エラータの発行日については **発行日** を選択します。
    - エラータの最終更新日については **更新日** を選択します。
11. **Start Date** を選択して、すべてのエラータを除外するか、選択した日付以降のエラータを除外します。
12. **End Date** フィールドは空白にしておきます。
13. **Save** をクリックします。
14. **新規バージョンの公開** をクリックして、表示されているリポジトリーを公開します。
15. **説明** フィールドに **Adding errata filter** と入力します。
16. **Save** をクリックします。
    コンテンツビューの公開が完了すると、**コンテンツ** 列に、初期リポジトリーからのパッケージとエラータの数が減ったことが報告される点に留意してください。これは、前年のセキュリティー以外のエラータがフィルターにより正常に除外されたためです。
17. **バージョン** タブをクリックします。
18. 公開バージョンの右側にある **プロモート** をクリックします。
19. コンテンツビューバージョンをプロモートする環境を選択します。

20. **説明** フィールドに、プロモートの説明を入力します。
21. **Promote Version** をクリックして、必要とされる環境全体に、このコンテンツビューバージョンをプロモートします。

CLI 手順

1. エラータのフィルターを作成します。

```
# hammer content-view filter create \
--content-view "My_Content_View" \
--description "Exclude errata items from the YYYY-MM-DD" \
--name "My_Filter_Name" \
--organization "My_Organization" \
--type "erratum"
```

2. フィルタールールを作成して、指定の **開始日** 以降のエラータすべてを除外します。

```
# hammer content-view filter rule create \
--content-view "My_Content_View" \
--content-view-filter="My_Content_View_Filter" \
--organization "My_Organization" \
--start-date "YYYY-MM-DD" \
--types=security,enhancement,bugfix
```

3. コンテンツビューを公開します。

```
# hammer content-view publish \
--name "My_Content_View" \
--organization "My_Organization"
```

4. コンテンツビューをライフサイクル環境にプロモートし、そこに含まれるエラータをそのライフサイクル環境で利用できるようにします。

```
# hammer content-view version promote \
--content-view "My_Content_View" \
--organization "My_Organization" \
--to-lifecycle-environment "My_Lifecycle_Environment"
```

10.8. 増分コンテンツビューへのエラータの追加

エラータが利用できるがインストールできない場合には、増分コンテンツビューバージョンを作成して、エラータをコンテンツホストに追加できます。たとえば、コンテンツビューがバージョン 1.0 の場合は、コンテンツビューバージョン 1.1 になり、公開時に、コンテンツビューバージョン 2.0 になります。

重要

コンテンツビューのバージョンが古い場合、機能拡張エラータを増分的に追加する際に、非互換性が発生する可能性があります。これは、機能拡張は通常、リポジトリー内の最新のソフトウェア向けに設計されているためです。

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

手順

1. Satellite Web UI で、**Content** > **Content Types** > **Errata** に移動します。
2. **エラータ** のリストから、適用するエラータの名前をクリックします。
3. エラータを適用するコンテンツホストを選択し、**ホストに適用** をクリックします。これにより、コンテンツビューの増分更新が作成されます。
4. エラータをコンテンツホストに適用する場合は、**Apply Errata to Content Hosts immediately after publishing** のチェックボックスを選択します。
5. **確認** をクリックして、エラータを適用します。

CLI 手順

1. エラータと対応する ID をリスト表示します。

```
# hammer erratum list
```

2. 異なる content-view バージョンと対応する ID をリスト表示します。

```
# hammer content-view version list
```

3. content-view バージョンに単一のエラータを適用します。コンマ区切りのリストとして、さらに ID を追加できます。

```
# hammer content-view version incremental-update \
--content-view-version-id 319 --errata-ids 34068b
```

10.9. エラータのホストへの適用

以下の手順を使用して、エラータをレビューし、ホストに適用します。

前提条件

- Red Hat から利用可能な最新のエラータと、Red Hat Satellite リポジトリーを同期している。詳細は、「リポジトリーの同期」 を参照してください。
- Satellite Server の環境およびコンテンツビューにホストを登録している。詳細は、**ホストの管理** の ホストの登録 を参照してください。
- リモート実行用にホストを設定します。リモート実行ジョブの詳細は、**ホストの管理** の リモートジョブの設定およびセットアップ を参照してください。

ホストにエラータを適用する手順は、そのオペレーティングシステムによって異なります。

10.9.1. Red Hat Enterprise Linux 9 を実行しているホストへのエラータの適用

以下の手順を使用して、Red Hat Enterprise Linux 9 を実行しているホストに対するエラータを確認して適用します。

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

手順

1. Satellite Web UI で、**Hosts** > **Content Hosts** に移動し、エラータを適用するホストを選択します。
2. **エラータ** タブに移動してエラータのリストを表示します。
3. 適用するエラータを選択し、**Apply Selected (選択した項目を適用)** をクリックします。確認画面で、**適用** をクリックします。
4. 選択したエラータに関連付けられた全パッケージを更新するタスクが完了したら、**詳細** タブをクリックして更新済みのパッケージを表示します。

CLI 手順

1. ホストのすべてのエラータをリスト表示します。

```
# hammer host errata list \
--host client.example.com
```

2. エラータが含まれるモジュールのストリームを検索します。

```
# hammer erratum info --id ERRATUM_ID
```

3. ホストで、モジュールストリームを更新します。

```
# dnf update Module_Stream_Name
```

10.9.2. Red Hat Enterprise Linux 8 を実行しているホストへのエラータの適用

以下の手順を使用して、Red Hat Enterprise Linux 8 を実行しているホストに対するエラータを確認して適用します。

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

手順

1. Satellite Web UI で、**Hosts** > **Content Hosts** に移動し、エラータを適用するホストを選択します。
2. **エラータ** タブに移動してエラータのリストを表示します。
3. 適用するエラータを選択し、**Apply Selected (選択した項目を適用)** をクリックします。確認画面で、**適用** をクリックします。
4. 選択したエラータに関連付けられた全パッケージを更新するタスクが完了したら、**詳細** タブをクリックして更新済みのパッケージを表示します。

CLI 手順

1. ホストのすべてのエラータをリスト表示します。

```
# hammer host errata list \
--host client.example.com
```

2. エラータが含まれるモジュールのストリームを検索します。

```
# hammer erratum info --id ERRATUM_ID
```

3. ホストで、モジュールストリームを更新します。

```
# dnf update Module_Stream_Name
```

10.9.3. Red Hat Enterprise Linux 7 を実行しているホストへのエラータの適用

以下の手順を使用して、Red Hat Enterprise Linux 7 を実行しているホストに対するエラータを確認して適用します。

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

手順

1. Satellite Web UI で、**Hosts** > **Content Hosts** に移動し、エラータを適用するホストを選択します。
2. **エラータ** タブに移動してエラータのリストを表示します。
3. 適用するエラータを選択し、**Apply Selected (選択した項目を適用)** をクリックします。確認画面で、**適用** をクリックします。
4. 選択したエラータに関連付けられた全パッケージを更新するタスクが完了したら、**詳細** タブをクリックして更新済みのパッケージを表示します。

CLI 手順

1. ホストのすべてのエラータをリスト表示します。

```
# hammer host errata list \
--host client.example.com
```

2. ホストに最新のエラータを適用します。エラータ ID を使用して適用するエラータを特定します。
Remote Execution の使用

```
# hammer job-invocation create \
--feature katello_errata_install \
--inputs errata=ERRATUM_ID1,ERRATUM_ID2 \
--search-query "name = client.example.com"
```

10.10. 複数ホストへのエラータの適用

以下の手順を使用して、エラータをレビューし、複数の RHEL ホストに適用します。

前提条件

- Red Hat から利用可能な最新のエラータと、Red Hat Satellite リポジトリーを同期している。詳細は、「リポジトリーの同期」 を参照してください。

- Satellite Server の環境およびコンテンツビューにホストを登録している。詳細は、**ホストの管理** の ホストの登録 を参照してください。
- リモート実行用にホストを設定します。リモート実行ジョブの詳細は、**ホストの管理** の リモートジョブの設定およびセットアップ を参照してください。

手順

1. Satellite Web UI で、**Content** > **Content Types** > **Errata** に移動します。
2. 適用するエラータ名をクリックします。
3. **コンテンツホスト** タブをクリックします。
4. エラータの適用先のホストを選択し、**ホストへの適用** をクリックします。
5. **Confirm** をクリックします。

CLI 手順

1. インストール可能な全エラータを表示します。

```
# hammer erratum list \
--errata-restrict-installable true \
--organization "Default Organization"
```

2. エラータのいずれかを複数のホストに適用します。
Remote Execution の使用

```
# hammer job-invocation create \
--feature katello_errata_install \
--inputs errata=ERRATUM_ID \
--search-query "applicable_errata = ERRATUM_ID"
```

次の Bash スクリプトを使用して、このエラータが利用可能な各ホストにエラータを適用します。

```
for HOST in hammer --csv --csv-separator "|" host list --search "applicable_errata = ERRATUM_ID" --organization "Default Organization" | tail -n+2 | awk -F "|" '{ print $2 }' ; do
  echo "== Applying to $HOST ==" ; hammer host errata apply --host $HOST --errata-ids ERRATUM_ID1,ERRATUM_ID2 ;
done
```

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

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

```
# hammer task list
```

4. 選択したタスクの状態を表示します。

```
# hammer task progress --id task_ID
```

10.11. ホストコレクションへのエラータの適用

Remote Execution の使用

```
# hammer job-invocation create \
--feature katello_errata_install \
--inputs errata=ERRATUM_ID1,ERRATUM_ID2,... \
--search-query "host_collection = HOST_COLLECTION_NAME"
```

第11章 コンテナーイメージの管理

Satellite では、さまざまなソースからコンテナーイメージをインポートして、コンテンツビューを使用して外部コンテナーに分散できます。

Red Hat Enterprise Linux Atomic Host 7 のコンテナーの詳細は、**Red Hat Enterprise Linux Atomic Host 7** の コンテナーの使用 を参照してください。

Red Hat Enterprise Linux 8 のコンテナーの詳細は、**Red Hat Enterprise Linux 8**の コンテナーの構築、実行、および管理 を参照してください。

Red Hat Enterprise Linux 9 のコンテナーの詳細は、**Red Hat Enterprise Linux 9**の コンテナーの構築、実行、および管理 を参照してください。

11.1. コンテナーイメージのインポート

Red Hat レジストリーまたは他のイメージレジストリーからコンテナーイメージリポジトリーをインポートできます。

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

リポジトリー検出を使用した手順

1. Satellite Web UI で **Content** > **Products** に移動して、**Repo Discovery** をクリックします。
2. **Repository Type** リストから **Container Images** を選択します。
3. **Registry to Discover** フィールドには、イメージのインポート元となるレジストリーの URL を入力します。
4. **Registry Username** フィールドには、コンテナーのイメージレジストリーのユーザー名に対応する名前を入力します。
5. **Registry Password** フィールドには、入力したユーザー名に対応するパスワードを入力します。
6. **Registry Search Parameter** フィールドには、検索の絞り込みに使用する検索条件を入力して、**Discover** をクリックします。
7. オプション: **Discovered Repository** リストをさらに絞り込むには、**Filter** フィールドに、使用する追加の検索条件を入力します。
8. **Discovered Repository** リストからインポートするリポジトリーを選択して、**Create Selected** をクリックします。
9. オプション: このコンテナーリポジトリーのダウンロードポリシーを **on demand** に変更するには、「リポジトリーのダウンロードポリシーの変更」 を参照してください。
10. オプション: **Product** リストから製品を作成するには、**New Product** を選択します。
11. **Name** フィールドに製品名を入力します。
12. オプション: **Repository Name** と **Repository Label** のコラムで、リポジトリー名とラベルを編集できます。
13. **Run Repository Creation** をクリックします。

14. リポジトリーの作成が完了したら、各新規リポジトリーをクリックして詳細情報を確認できます。

15. オプション: リポジトリーにインポートするコンテンツをフィルタリングするには、リポジトリーをクリックして、**Limit Sync Tags** に移動します。これをクリックし、Satellite への同期コンテンツを制限するタグを編集または追加します。

16. Satellite Web UI で、**Content** > **Products** に移動し、製品の名前を選択します。

17. 新規リポジトリーを選択し、**Sync Now** をクリックして同期プロセスを開始します。

リポジトリーの手動作成を伴う手順

1. Satellite Web UI で、**Content** > **Products** に移動します。必要な製品の名前をクリックします。

2. **New repository** をクリックします。

3. **Type** リストから **docker** を選択します。リポジトリーの詳細を入力し、**Save** をクリックします。

4. 新しいリポジトリーを選択し、**Sync Now** をクリックします。

次のステップ

- 同期の進捗状況を表示するには、**Content** > **Sync Status** に移動して、リポジトリーツリーをデプロイメントします。
- 同期が完了したら、**Container Image Manifests** をクリックして利用可能なマニフェストをリスト表示します。また、必要のなくなったマニフェストは、このリストから削除できます。

CLI 手順

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

```
# hammer product create \
--description "My_Description" \
--name "Red Hat Container Catalog" \
--organization "My_Organization" \
--sync-plan "My_Sync_Plan"
```

2. 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 \
--content-type "docker" \
--docker-upstream-name "rhel7" \
--name "RHEL7" \
--organization "My_Organization" \
--product "Red Hat Container Catalog" \
--url "http://registry.access.redhat.com/"
```

3. リポジトリーを同期します。

```
# hammer repository synchronize \
--name "RHEL7" \
```

```
--organization "My_Organization" \
--product "Red Hat Container Catalog"
```

関連情報

- 手動での製品とリポジトリーの作成方法の詳細は、*4章 コンテンツのインポート* を参照してください。

11.2. コンテナー名のパターンの管理

Satellite を使用してコンテナーの作成や管理を行う場合には、コンテナーはコンテンツビューのバージョン間や異なるステージの Satellite ライフサイクル環境間を移動するので、コンテナー名はステージごとに変更されます。たとえば、アップストリームのリポジトリーから、**ssh** 名を使用してコンテナーイメージを同期する場合には、そのイメージを Satellite 製品と組織に追加してコンテンツビューの一部として公開する時に、コンテナーイメージは **my_organization_production-custom_spin-my_product-custom_ssh** という名前になる可能性があります。これが原因で、コンテナーイメージをプルする時に問題が発生する可能性があります。理由は、コンテナーレジストリーに、コンテナー名のインスタンスが 1 つしか含まれていない可能性があるためです。Satellite の命名規則の問題を回避するには、デフォルト名を上書きするようにレジストリー名のパターンを設定して、コンテナー名が後で使用するときに明確になるようにします。

制限事項

レジストリー名のパターンを使用してコンテナーの命名規則を管理する場合には、レジストリーの命名パターンが理由でグローバルー意名を生成する必要があるので、命名時に競合の問題が発生する可能性があります。以下に例を示します。

- **repository.docker_upstream_name** のレジストリー名パターンを設定した場合に、**Production** ライフサイクルと同じリポジトリー名のコンテナーコンテンツが含まれるコンテンツビューを公開またはプロモートできません。
- **lifecycle_environment.name** のレジストリー名パターンを設定した場合には、同じ名前を指定して 2 つ目のコンテナーリポジトリーを作成することができません。

コンテナーに対してレジストリーの命名パターンを定義する場合には、注意を払って進めて行く必要があります。

手順

レジストリー名パターンで、コンテナーの命名を管理するには、以下の手順を実行します。

1. Satellite Web UI で、**Content** > **Lifecycle** > **Lifecycle Environments** に移動します。
2. ライフサイクル環境を作成するか、既存のライフサイクル環境を選択して編集します。
3. **コンテナーイメージのレジストリー** エリアで、**レジストリー名のパターン** の右側にある編集アイコンをクリックします。
4. 変数リストと例を使用して、必要とされるレジストリー名のパターンを判断します。
5. **レジストリー名のパターン** フィールドに、使用するレジストリー名のパターンを入力します。たとえば、**repository.docker_upstream_name** を使用するには、以下を入力します。

```
<%= repository.docker_upstream_name %>
```

6. **Save** をクリックします。

11.3. コンテナーレジストリーの認証管理

Satellite からコンテナーイメージにアクセスするための認証設定を管理できます。デフォルトでは、Satellite のコンテナーイメージにアクセスするには認証が必要です。

ライフサイクル環境の Satellite イメージレジストリーに含まれるコンテナーイメージにアクセスするのにユーザーの認証をするかどうかを指定できます。たとえば、認証要件なしに **Production** ライフサイクルからコンテナーイメージにアクセスできるようにしたり、認証済みのユーザーだけに **Development** および **QA** 環境へのアクセスを制限したりすることもできます。

手順

1. Satellite Web UI で、**Content** > **Lifecycle** > **Lifecycle Environments** に移動します。
2. 認証を管理するライフサイクル環境を選択します。
3. このライフサイクル環境に認証なしでアクセスできるようにするには、**Unauthenticated Pull** のチェックボックスを選択します。認証なしのアクセスを制限するには、**Unauthenticated Pull** のチェックボックスのチェックを外します。
4. **Save** をクリックします。

11.4. 認証局を信頼するように PODMAN と DOCKER を設定する

Podman は、**/etc/containers/certs.d/** および **/etc/docker/certs.d/** の 2 つのパスを使用して CA ファイルを見つけます。

ルート CA ファイルをこれらの場所のいずれかにコピーし、サーバーのホスト名によって判別されるパスを使用して、**ca.crt** ファイルに名前を付けます。

次の例では、ユースケースに応じて、**hostname.example.com** を **satellite.example.com** または **capsule.example.com** に置き換えます。

- 最初に、以下を使用して関連する場所を作成する必要がある場合があります。

```
# mkdir -p /etc/containers/certs.d/hostname.example.com
```

  または

```
# mkdir -p /etc/docker/certs.d/hostname.example.com
```

- podman を使用する場合

```
# cp rootCA.pem /etc/containers/certs.d/hostname.example.com/ca.crt
```

- または、Docker を使用している場合は、ルート CA ファイルを同等の Docker ディレクトリーにコピーします。

```
# cp rootCA.pem /etc/docker/certs.d/hostname.example.com/ca.crt
```

レジストリーにログインする際に、**--tls-verify=false** オプションを使用する必要がなくなります。

```
$ podman login hostname.example.com
```

```
Username: admin
Password:
Login Succeeded!
```

11.5. コンテナーレジストリーの使用

Podman および Docker を使用して、コンテナーレジストリーからコンテンツを取得できます。

Capsule のコンテナーレジストリー

コンテンツを含む Capsule では、Container Gateway Capsule プラグインはコンテナーレジストリーとして機能します。Katello から認証情報をキャッシュし、受信した要求を Pulp にプロキシーします。Container Gateway は、コンテンツを含む Capsules でデフォルトで利用できます。

手順

コンテナーレジストリーにログインします。

```
# podman login satellite.example.com
```

コンテナーイメージをリスト表示します。

```
# podman search satellite.example.com/
```

コンテナーイメージをプルします。

```
# podman pull satellite.example.com/my-image:<optional_tag>
```

第12章 ISO イメージの管理

Satellite を使用して、Red Hat のコンテンツ配信ネットワーク (CDN) または他のソースからの ISO イメージを保存できます。仮想マシンイメージなどの他のファイルをアップロードしたり、リポジトリーに公開したりすることも可能です。

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手順は、RPM コンテンツのリポジトリーを有効にする手順と似ています。

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

手順

1. Satellite Web UI で、**Content** > **Red Hat Repositories** に移動します。
2. **検索** フィールドで、**Red Hat Enterprise Linux 7 Server (ISOs)** などのイメージ名を入力します。
3. 利用可能なリポジトリーウィンドウで、**Red Hat Enterprise Linux 7 Server (ISOs)**をデプロイメントします。
4. **x86_64 7.2** エントリーでは、**有効化** アイコンをクリックして、対象のイメージのリポジトリーを有効にします。
5. Satellite Web UI で、**Content** > **Products** に移動して、**Red Hat Enterprise Linux Server** をクリックします。
6. Red Hat Enterprise Linux Server ウィンドウの **Repositories** タブをクリックして、**Red Hat Enterprise Linux 7 Server ISOs x86_64 7.2** をクリックします。
7. Red Hat Enterprise Linux 7 Server ISOs x86_64 7.2 ウィンドウの右上で、**アクションの選択** をクリックして、**同期開始** を選択します。

同期のステータスの表示方法

- Satellite Web UI で、**Content** > **Sync Status** に移動し、**Red Hat Enterprise Linux Server**を展開します。

CLI 手順

1. **file** リポジトリーの Red Hat Enterprise Linux Server 製品を特定します。

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

2. 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 "My_Organization"
```

3. 製品のリポジトリーを見つけます。

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

4. 製品のリポジトリーを同期します。

```
# hammer repository synchronize \
--name "Red Hat Enterprise Linux 7 Server ISOs x86_64 7.2" \
--product "Red Hat Enterprise Linux Server" \
--organization "My_Organization"
```

12.2. 個別の ISO イメージとファイルのインポート

以下の手順を使用して、ISO コンテンツとその他のファイルを Satellite Server に手動でインポートします。ファイルをインポートするには、Satellite Web UI または Hammer CLI を使用して、以下の手順を実行してください。ただし、アップロードするファイルのサイズが 15 MB よりも大きい場合は、Hammer CLI を使用してリポジトリーにアップロードする必要があります。

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

手順

1. Satellite Web UI で **Content** > **Products** に移動して、製品ウィンドウで **Create Product** をクリックします。
2. **名前** フィールドで製品を識別するための名前を入力します。この名前が **ラベル** フィールドに投入されます。
3. オプション: **GPG Key** フィールドには、製品の GPG キーを入力します。
4. オプション: **同期プラン** リストから製品の同期プランを選択します。
5. オプション: **Description** フィールドには、製品の説明を入力します。
6. **Save** をクリックします。
7. 製品ウィンドウで、新製品をクリックし、**リポジトリーの作成** をクリックします。
8. **Name** フィールドに、リポジトリーの名前を入力します。これにより、自動的に **ラベル** フィールドにデータが投入されます。
9. **タイプ** のリストから **ファイル** を選択します。
10. **Upstream URL** フィールドに、ソースとして使用するレジストリーの URL を入力します。**アップストリームのユーザー名** と **アップストリームのパスワード** フィールドには対応するユーザー名とパスワードを追加します。
11. **Save** をクリックします。

12. 新しいリポジトリーを選択します。
13. **ファイルのアップロード** に移動し、**参照** をクリックします。
14. **.iso** ファイルを選択して **アップロード** をクリックします。

CLI 手順

1. カスタム製品を作成します。

```
# hammer product create \
--name "My_ISOs" \
--sync-plan "Example Plan" \
--description "My_Product" \
--organization "My_Organization"
```

2. リポジトリーを作成します。

```
# hammer repository create \
--name "My_ISOs" \
--content-type "file" \
--product "My_Product" \
--organization "My_Organization"
```

3. 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 \
--id repo_ID \
--organization "My_Organization"
```

第13章 ANSIBLE コンテンツの管理

Ansible コレクションを複数のソースから Satellite Server にインポートできます。

Satellite での Ansible 統合の詳細は、**Ansible 統合を使用した設定の管理** を参照してください。

13.1. ANSIBLE コレクションの同期

Satellite では、Private Automation Hub、**console.redhat.com**、およびその他の Satellite インスタンスからの Ansible コレクションを同期できます。Ansible コレクションは、同期後、**コンテンツ** の下の Satellite Web UI メニューの新しいリポジトリータイプとしてSatellite に表示されます。

手順

1. Satellite Web UI で、**Content** > **Products** に移動します。
2. 必要な製品名を選択します。
3. **製品** ウィンドウで、リポジトリーを作成する製品の名前を選択します。
4. **リポジトリー** タブをクリックして、**新規リポジトリー** をクリックします。
5. **Name** フィールドに、リポジトリーの名前を入力します。
   ラベル フィールドには、名前に基づいて自動的に入力されます。
6. **タイプ** リストから、**ansible コレクション** を選択します。
7. **アップストリーム URL** フィールドに、アップストリームコレクションリポジトリーの URL を入力します。
   URL には、任意の Ansible Galaxy エンドポイントを指定できます。例:
   https://console.redhat.com/api/automation-hub/
8. オプション: **Requirements.yml** フィールドで、エンドポイントから同期するコレクションのリストとそのバージョンを指定できます。
   コレクションのリストを指定しない場合は、エンドポイントのすべてが同期されます。

   ```
   ---
   collections:
   - name: my_namespace.my_collection
     version: 1.2.3
   ```

   詳細は、**Galaxy ユーザーガイド** の Installing roles and collections from the same requirements.yml file を参照してください。
9. 認証します。
   a. **Private Automation Hub** から Satellite を同期するには、 **Auth Token** フィールドにトークンを入力します。
      詳細は、**Connect to Hub** ガイドの Connect Private Automation Hub を参照してください。
   b. **console.redhat.com** から Satellite を同期するには、 **Auth Token** フィールドにトークンを入力し、**Auth URL** フィールドに SSO URL を入力します。
      詳細は、**Automation Hub の概要** を参照してください。

c. Satellite から Satellite を同期するには、両方の認証フィールドを空白のままにします。

10. **Save** をクリックします。
11. Ansible Collections リポジトリーに移動します。
12. **アクションの選択** メニューから **同期開始** を選択します。

第14章 カスタムファイルタイプコンテンツの管理

Satellite で、SSH キーおよびソースコードファイル、仮想マシンイメージや ISO ファイルなどの大容量ファイルの管理と配布が必要になる場合があります。これには、Red Hat Satellite のカスタム製品にカスタムファイルタイプのリポジトリーを追加します。こうすることで、製品に任意のファイルを組み込む一般的な方法が提供されます。

リポジトリーにファイルをアップロードし、アップストリームの Satellite Server からファイルを同期できます。ファイルをカスタムのファイルタイプリポジトリーに追加すると、特定のバージョンをコンテンツビューに追加して、さまざまな Capsule Server でファイルのリポジトリーを利用可能にするなど、通常の Satellite 管理機能を使用できます。**curl -O** を使用して、HTTP または HTTPS 経由でクライアントにファイルをダウンロードする必要があります。

Satellite Server のファイルタイプリポジトリーはカスタム製品に対してのみ作成できますが、リポジトリーソースは柔軟に作成できます。Satellite Server またはリモート HTTP サーバーのディレクトリーに独立したリポジトリーソースを作成し、そのディレクトリーの内容を Satellite に同期できます。この方法は、Satellite リポジトリーに追加するファイルが複数ある場合に便利です。

14.1. カスタムファイルタイプリポジトリーのローカルソースの作成

Satellite が **Pulp Manifest** コマンドを使用してインストールされているベースシステムで、ファイルのディレクトリーから、カスタムファイルタイプリポジトリーを作成できます。その後、ファイルをリポジトリーに同期し、カスタムファイルタイプのコンテンツを他のコンテンツタイプと同様に管理できます。

以下の手順を使用して、Satellite がインストールされているベースシステムのディレクトリーにリポジトリーを設定します。リモートサーバーのディレクトリーにファイルタイプリポジトリーを作成するには、「カスタムファイルタイプリポジトリーのリモートソースの作成」 を参照してください。

手順

1. Utils リポジトリーが有効になっていることを確認します。

```
# subscription-manager repos \
--enable=rhel-8-for-x86_64-appstream-rpms \
--enable=rhel-8-for-x86_64-baseos-rpms \
--enable=satellite-utils-6.15-for-rhel-8-x86_64-rpms
```

2. satellite-utils モジュールを有効にします。

```
# dnf module enable satellite-utils
```

3. Pulp マニフェストパッケージをインストールします。

```
# satellite-maintain packages install python3.11-pulp_manifest
```

このコマンドは、Satellite サービスを停止し、**satellite-installer** を再実行することに注意してください。または、サービスの停止によるダウンタイムを防ぐために、次を使用できます。

```
# satellite-maintain packages unlock
# satellite-maintain packages install python39-pulp_manifest
# satellite-maintain packages lock
```

4. 次のように、ファイルタイプリポジトリーとして使用するディレクトリーを作成します。

```
# mkdir -p /var/lib/pulp/local_repos/my_file_repo
```

5. 親フォルダーを許可されたインポートパスに追加します。

```
# satellite-installer --foreman-proxy-content-pulpcore-additional-import-paths
/var/lib/pulp/local_repos
```

6. ディレクトリーにファイルを追加して、テストファイルを作成します。

```
# touch /var/lib/pulp/local_repos/my_file_repo/test.txt
```

7. Pulp Manifest コマンドを入力して、マニフェストを作成します。

```
# pulp-manifest /var/lib/pulp/local_repos/my_file_repo
```

8. マニフェストが作成されたことを確認します。

```
# ls /var/lib/pulp/local_repos/my_file_repo
PULP_MANIFEST test.txt
```

これで、ローカルソースをカスタムファイルタイプリポジトリーとしてインポートできるようになりました。**file:**// URL スキームとディレクトリーの名前を使用して、**Upstream URL** (**file:///var/lib/pulp/local_repos/my_file_repo** など) を指定します。詳細は、「カスタムファイルタイプリポジトリーの作成」 を参照してください。

ディレクトリーの内容を更新する場合は、Pulp Manifest を再実行し、Satellite でリポジトリーを同期します。詳細は、「リポジトリーの同期」 を参照してください。

14.2. カスタムファイルタイプリポジトリーのリモートソースの作成

Pulp Manifest を使用して、Satellite Server の外部にあるファイルのディレクトリーから、カスタムファイルタイプリポジトリーソースを作成します。その後、HTTP または HTTPS を介してファイルを Satellite Server のリポジトリーに同期し、カスタムファイルタイプのコンテンツを他のコンテンツタイプと同様に管理できます。

以下の手順を使用して、リモートサーバーのディレクトリーにリポジトリーを設定します。Satellite Server がインストールされているベースシステムのディレクトリーにファイルタイプリポジトリーを作成するには、「カスタムファイルタイプリポジトリーのローカルソースの作成」 を参照してください。

前提条件

- Satellite または Red Hat CDN に登録された Red Hat Enterprise Linux 8 を実行するサーバーがあります。
- サーバーには、Red Hat Enterprise Linux Server および Red Hat Satellite Utils リポジトリーへのエンタイトルメントがあります。
- HTTP サーバーがインストールされている。Web サーバーの設定の詳細は、Red Hat Enterprise Linux 8 の **さまざまな種類のサーバーのデプロイメント** の Apache HTTP Web サーバーの設定 を参照してください。

手順

1. サーバー上で、必要なリポジトリーを有効にします。

```
# subscription-manager repos \
--enable=rhel-8-for-x86_64-appstream-rpms \
--enable=rhel-8-for-x86_64-baseos-rpms \
--enable=satellite-utils-6.15-for-rhel-8-x86_64-rpms
```

2. satellite-utils モジュールを有効にします。

```
# dnf module enable satellite-utils
```

3. Pulp マニフェストパッケージをインストールします。

```
# dnf install python3.11-pulp_manifest
```

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

```
# mkdir /var/www/html/pub/my_file_repo
```

5. ディレクトリーにファイルを追加して、テストファイルを作成します。

```
# touch /var/www/html/pub/my_file_repo/test.txt
```

6. Pulp Manifest コマンドを入力して、マニフェストを作成します。

```
# pulp-manifest /var/www/html/pub/my_file_repo
```

7. マニフェストが作成されたことを確認します。

```
# ls /var/www/html/pub/my_file_repo
PULP_MANIFEST test.txt
```

これで、リモートソースをカスタムファイルタイプリポジトリーとしてインポートできるようになりました。ディレクトリーへのパスを使用して、**Upstream URL** (**http://satellite.example.com/my_file_repo** など) を指定します。詳細は、「カスタムファイルタイプリポジトリーの作成」 を参照してください。

ディレクトリーの内容を更新する場合は、Pulp Manifest を再実行し、Satellite でリポジトリーを同期します。詳細は、「リポジトリーの同期」 を参照してください。

14.3. カスタムファイルタイプリポジトリーの作成

カスタムファイルタイプリポジトリーを作成する手順は、リポジトリー作成時に **ファイル** タイプを選択する以外は、カスタムコンテンツの作成手順と同じです。製品を作成してから、カスタムリポジトリーを追加する必要があります。

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

手順

1. Satellite Web UI で、**Content > Products** に移動します。
2. リポジトリーを作成する製品を選択します。
3. **Repositories** タブで、**New Repository** をクリックします。
4. **Name** フィールドに、リポジトリーの名前を入力します。Satellite では、名前を基に **ラベル** フィールドに値が自動的に入力されます。
5. オプション: **Description** フィールドに、新規リポジトリーの説明を入力します。
6. **Type** のリストから、リポジトリーのタイプとして **file** を選択します。
7. オプション: **Upstream URL** フィールドに、ソースとして使用するアップストリームリポジトリーの URL を入力します。アップストリーム URL を入力しない場合は、パッケージを手動でアップロードできます。詳細は、「カスタムファイルタイプリポジトリーへのファイルのアップロード」 を参照してください。
8. リポジトリーの SSL 証明書が信頼できる CA によって署名されていることを確認するには、**Verify SSL** を選択します。
9. オプション: 認証に必要な場合は、**Upstream Username** フィールドに、アップストリームリポジトリーのユーザー名を入力します。リポジトリーに認証が必要ない場合は、このフィールドをクリアします。
10. オプション: **Upstream Password** フィールドに、アップストリームリポジトリーに対応するパスワードを入力します。リポジトリーに認証が必要ない場合は、このフィールドをクリアします。
11. オプション: **Upstream Authentication Token** フィールドに、認証用のアップストリームリポジトリーユーザーのトークンを指定します。リポジトリーに認証が必要ない場合は、このフィールドを空欄のままにします。
12. **ミラーリングポリシー** リストから、Satellite Server が実行するコンテンツの同期のタイプを選択します。詳細は、「ミラーリングポリシーの概要」 を参照してください。
13. **HTTP Proxy Policy** フィールドで、HTTP プロキシーを選択します。デフォルトでは、**Global Default** HTTP プロキシーを使用します。
14. オプション: **Unprotected** のチェックボックスをオフにして、このリポジトリーにアクセスするためにサブスクリプションエンタイトルメント証明書を要求することができます。デフォルトでは、リポジトリーは HTTP 経由で公開されます。
15. オプション: **SSL CA Cert** フィールドで、リポジトリーの SSL CA 証明書を選択します。
16. オプション: **SSL Client Cert** フィールドで、リポジトリーの SSL Client Certificate を選択します。
17. オプション: **SSL Client Key** フィールドで、リポジトリーの SSL Client Key を選択します。
18. **Save** をクリックして、リポジトリーを作成します。

CLI 手順

1. カスタム製品を作成します。

```
# hammer product create \
```

```
--description "My_Files" \
--name "My_File_Product" \
--organization "My_Organization" \
--sync-plan "My_Sync_Plan"
```

表14.1 hammer product create コマンドのオプションパラメーター

オプション	説明
--gpg-key-id gpg_key_id	GPG キー数値 ID
--sync-plan-id sync_plan_id	同期プランの数値 ID
--sync-plan sync_plan_name	検索する同期プラン名

2. **file** タイプリポジトリーを作成します。

```
# hammer repository create \
--content-type "file" \
--name "My_Files" \
--organization "My_Organization" \
--product "My_File_Product"
```

表14.2 hammer repository create コマンドのオプションパラメーター

オプション	説明
--checksum-type sha_version	リポジトリーのチェックサム (**sha1** または **sha256**)
--download-policy policy_name	リポジトリーのダウンロードポリシー (**immediate** または **on_demand**)
--gpg-key-id gpg_key_id	GPG キー数値 ID
--gpg-key gpg_key_name	検索するキー名
--mirror-on-sync boolean	同期する場合に、このリポジトリーをソースからミラーリングし、古いパッケージを削除する必要がありますか?**true** または **false**、**yes** または **no**、もしくは **1** または **0** に設定します。
--publish-via-http boolean	HTTP を使用して公開する必要もありますか ?**true** または **false**、**yes** または **no**、もしくは **1** または **0** に設定します。
--upstream-password repository_password	アップストリームリポジトリーユーザーのパスワード

オプション	説明
--upstream-username repository_username	認証に必要な場合は、アップストリームリポジトリーユーザー
--url My_Repository_URL	リモートリポジトリーの URL
--verify-ssl-on-sync boolean	リモートリポジトリーのアップストリーム SSL 証明書が信頼できる CA によって署名されていることを確認する**true** または **false**、**yes** または **no**、もしくは **1** または **0** に設定します。

14.4. カスタムファイルタイプリポジトリーへのファイルのアップロード

ファイルをカスタムファイルタイプリポジトリーにアップロードするには、以下の手順を行います。

手順

1. Satellite Web UI で、**Content** > **Products** に移動します。
2. カスタム製品の名前を選択します。
3. ファイルタイプリポジトリーの名前を選択します。
4. **参照** をクリックして、アップロードするファイルを選択します。
5. **アップロード** をクリックして、選択したファイルを Satellite Server にアップロードします。
6. リポジトリーを公開した URL を開いて、ファイルを表示します。

CLI 手順

```
# hammer repository upload-content \
--id repo_ID \
--organization "My_Organization" \
--path example_file
```

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

14.5. カスタムファイルタイプリポジトリーからホストにファイルをダウンロードする

curl -O を使用して HTTPS 経由でクライアントにファイルをダウンロードできます。また、リポジトリーの **Unprotected** オプションが選択されている場合は、オプションで HTTP 経由でもダウンロードできます。

前提条件

- カスタムファイルタイプリポジトリーがある。詳細は、「カスタムファイルタイプリポジトリーの作成」 を参照してください。

- ファイルタイプリポジトリーから、クライアントにダウンロードするファイル名を把握しておく。
- HTTPS を使用するには、クライアントで以下の証明書を用意しておく。
  - **katello-server-ca.crt**。詳細は、**Red Hat Satellite の管理** の Katello ルート CA 証明書のインポート を参照してください。
  - 組織のデバッグ証明書。詳細は、**Red Hat Satellite の管理** の 組織のデバッグ証明書の作成 を参照してください。

手順

1. Satellite Web UI で、**Content** > **Products** に移動します。
2. カスタム製品の名前を選択します。
3. ファイルタイプリポジトリーの名前を選択します。
4. **Unprotected** チェックボックスを選択して、HTTP 経由で公開されているリポジトリーにアクセスしてください。
5. **Published At** URL をコピーします。
6. クライアントで、Satellite Server からファイルをダウンロードします。
   - HTTPS の場合:

```
# curl \
--cacert ./_katello-server-ca.crt \
--cert ./_My_Organization_key-cert.pem \
--remote-name \
https://satellite.example.com/pulp/content/My_Organization_Label/Library/custom/My_Product_Label/My_Repository_Label/My_File
```

   - HTTP の場合:

```
# curl \
--remote-name \
http://satellite.example.com/pulp/content/My_Organization_Label/Library/custom/My_Product_Label/My_Repository_Label/My_File
```

CLI 手順

1. ファイルタイプリポジトリーをリスト表示します。

```
# hammer repository list --content-type file
---|------------|------------------|--------------|----
ID | NAME       | PRODUCT          | CONTENT TYPE | URL
---|------------|------------------|--------------|----
7  | My_Files   | My_File_Product  | file         |
---|------------|------------------|--------------|----
```

2. リポジトリー情報を表示します。

```
# hammer repository info \
--name "My_Files" \
--organization-id My_Organization_ID \
--product "My_File_Product"
```

Unprotected が有効になっている場合、出力は次のようになります。

```
Publish Via HTTP: yes
Published At:
https://satellite.example.com/pulp/content/My_Organization_Label/Library/custom/My_File_Product_Label/My_Files_Label/
```

Unprotected が有効になっていない場合、出力は次のようになります。

```
Publish Via HTTP: no
Published At:
https://satellite.example.com/pulp/content/My_Organization_Label/Library/custom/My_File_Product_Label/My_Files_Label/
```

3. クライアントで、Satellite Server からファイルをダウンロードします。

- HTTPS の場合:

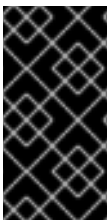
```
# curl \
--cacert ./_katello-server-ca.crt \
--cert ./_My_Organization_key-cert.pem \
--remote-name \
https://satellite.example.com/pulp/content/My_Organization_Label/Library/custom/My_Product_Label/My_Repository_Label/My_File
```

- HTTP の場合:

```
# curl \
--remote-name \
http://satellite.example.com/pulp/content/My_Organization_Label/Library/custom/My_Product_Label/My_Repository_Label/My_File
```

付録A コンテンツストレージ向け NFS 共有の使用

使用する環境ではコンテンツのストレージに十分な容量のハードディスクが必要になります。場合によっては、コンテンツのストレージに NFS 共有を使用することが便利なこともあります。本付録では、Satellite Server のコンテンツ管理コンポーネントに NFS 共有をマウントする方法を説明します。



重要

/var/lib/pulp のファイルシステムには、帯域幅が高く、レイテンシーの低いストレージを使用してください。Red Hat Satellite には I/O を大量に使用する操作が多数あるので、レイテンシーが高く、帯域幅の低いストレージを使用すると、パフォーマンスが低下する問題が発生する可能性があります。

手順

1. NFS 共有を作成します。この例では、**nfs.example.com:/Satellite/pulp** 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 Server で Satellite サービスを停止します。

```
# satellite-maintain 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。

```
# satellite-maintain packages install nfs-utils
```

4. **/var/lib/pulp** の既存のコンテンツを NFS 共有にコピーする必要があります。まず、NFS 共有を一時的なロケーションにマウントします。

```
# mkdir /mnt/temp
# mount -o rw nfs.example.com:/Satellite/pulp /mnt/temp
```

/var/lib/pulp の既存コンテンツを一時的なロケーションにコピーします。

```
# cp -r /var/lib/pulp/* /mnt/temp/.
```

5. 共有上の全ファイルで **pulp** 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。
6. 一時的なストレージのロケーションをアンマウントします。

```
# umount /mnt/temp
```

7. **/var/lib/pulp** の既存コンテンツを削除します。

```
# rm -rf /var/lib/pulp/*
```

8. **/etc/fstab** ファイルに以下の行を追加します。

```
nfs.example.com:/Satellite/pulp    /var/lib/pulp   nfs
rw,hard,intr,context="system_u:object_r:pulpcore_var_lib_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** 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/pulp 309506048 58632800 235128224  20% /var/lib/pulp
...
```

既存のコンテンツが **var/lib/pulp** のマウントにあることを確認します。

```
# ls /var/lib/pulp
```

11. Satellite Server で Satellite サービスを開始します。

```
# satellite-maintain 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 NFS 共有が予想どおり機能することを確認してください。詳細は、「リポジトリーの同期」 を参照してください。

付録B キックスタートリポジトリーのインポート

キックスタートリポジトリーは、コンテンツ ISO イメージでは提供されません。オフラインの Satellite でキックスタートリポジトリーを使用するには、使用する Red Hat Enterprise Linux のバージョンのバイナリー DVD ISO ファイルをダウンロードし、キックスタートファイルを Satellite にコピーする必要があります。

B.1. RED HAT ENTERPRISE LINUX 9 のキックスタートリポジトリーのインポート

以下の手順を使用して、Red Hat Enterprise Linux 9 のキックスタートリポジトリーをインポートします。

手順

1. Red Hat カスタマーポータル access.redhat.com/downloads に移動し、ログインします。
2. **Red Hat Enterprise Linux**をクリックします。
3. 一覧から製品バリアントと製品バージョンを選択します。(例: 製品バリアント **Red Hat Enterprise Linux for x86_64** および製品バージョン **9.0**)
4. **Red Hat Enterprise Linux 9.0 Binary DVD**など、完全インストールイメージを見つけ、**Download Now** をクリックします。最小限の ISO を使用してホストをプロビジョニングすることはできないことに注意してください。
5. ダウンロードが完了したら、ISO イメージを Satellite Server にコピーします。
6. Satellite Server で、マウントポイントを作成し、そのロケーションに ISO イメージを一時的にマウントします。

```
# mkdir /mnt/iso
# mount -o loop rhel-binary-dvd.iso /mnt/iso
```

rhel-binary-dvd.iso は、ISO イメージの名前に置き換えます。

7. Red Hat Enterprise Linux 9 AppStream および BaseOS Kickstart リポジトリーのディレクトリーを作成します。

```
# mkdir --parents /var/www/html/pub/satellite-import/content/dist/rhel9/9.0/x86_64/appstream/kickstart
# mkdir --parents /var/www/html/pub/satellite-import/content/dist/rhel9/9.0/x86_64/baseos/kickstart
```

8. ISO イメージから **kickstart** ファイルをコピーします。

```
# cp -a /mnt/iso/AppStream/* /var/www/html/pub/satellite-import/content/dist/rhel9/9.0/x86_64/appstream/kickstart
# cp -a /mnt/iso/BaseOS/* /mnt/iso/images/ /var/www/html/pub/satellite-import/content/dist/rhel9/9.0/x86_64/baseos/kickstart
```

BaseOS の場合は、**/mnt/iso/images/** ディレクトリーのコンテンツもコピーする必要があることに注意してください。

9. 次のエントリーをリストファイルに追加します。
/var/www/html/pub/satellite-import/content/dist/rhel9/9.0/x86_64/appstream/listing ファイルで改行して **kickstart** を追加します。

   /var/www/html/pub/satellite-import/content/dist/rhel9/9.0/x86_64/baseos/listing ファイルで改行して **kickstart** を追加します。

   /var/www/html/pub/satellite-import/content/dist/rhel8/listing ファイルで改行してバージョン番号を追加します。たとえば、Red Hat Enterprise Linux 9.0 バイナリー ISO の場合は **9.0** を追加します。

10. ISO イメージから **.treeinfo** ファイルをコピーします。

```
# cp /mnt/iso/.treeinfo /var/www/html/pub/satellite-
import/content/dist/rhel9/9.0/x86_64/appstream/kickstart/treeinfo
# cp /mnt/iso/.treeinfo /var/www/html/pub/satellite-
import/content/dist/rhel9/9.0/x86_64/baseos/kickstart/treeinfo
```

11. **/var/www/html/pub/satellite-import/content/dist/rhel9/9.0/x86_64/baseos/kickstart/treeinfo** ファイルを開いて編集します。

12. **[general]** セクションで、以下の変更を加えます。

    - **packagedir = AppStream/Packages** を **packagedir = Packages** に変更します。
    - **repository = AppStream** を **repository = .** に変更します。
    - **variant = AppStream** を **variant = BaseOS** に変更します。
    - **variants = AppStream,BaseOS** を **variants = BaseOS** に変更します。

13. **[tree]** セクションで、**variants = AppStream,BaseOS** を **variants = BaseOS** に変更します。

14. **[variant-BaseOS]** セクションで、以下の変更を加えます。

    - **packages = BaseOS/Packages** を **packages = Packages** に変更します。
    - **repository = BaseOS** を **repository = .** に変更します。

15. **[media]** および **[variant-AppStream]** のセクションを削除します。

16. ファイルを保存してから閉じます。

17. **/var/www/html/pub/satellite-import/content/dist/rhel9/9.0/x86_64/baseos/kickstart/treeinfo** ファイルが以下の形式であることを確認します。

```
[checksums]
images/efiboot.img =
sha256:c01c18acc6778d6e66c8d0872bac59bfd7219ccf3cfa70a5c605c0fb37f33a83
images/install.img =
sha256:ddd08e5a5d92edee150f91ff4f12f39253eae72ff496465cf1b2766fe4a4df49
images/pxeboot/initrd.img =
sha256:a09a8ec89d485d71ed1bdad83584d6d816e67448221172d9aad97886cd70adca
images/pxeboot/vmlinuz =
sha256:6e523d7c3266e26c695923ab12b2873b16b0c61fb2e48ade608ad8998821584b

[general]
```

```
; WARNING.0 = This section provides compatibility with pre-productmd treeinfos.
; WARNING.1 = Read productmd documentation for details about new format.
arch = x86_64
family = Red Hat Enterprise Linux
name = Red Hat Enterprise Linux 9.0.0
packagedir = Packages
platforms = x86_64,xen
repository = .
timestamp = 1571146127
variant = BaseOS
variants = BaseOS
version = 9.0.0

[header]
type = productmd.treeinfo
version = 1.2

[images-x86_64]
efiboot.img = images/efiboot.img
initrd = images/pxeboot/initrd.img
kernel = images/pxeboot/vmlinuz

[images-xen]
initrd = images/pxeboot/initrd.img
kernel = images/pxeboot/vmlinuz

[release]
name = Red Hat Enterprise Linux
short = RHEL
version = 9.0.0

[stage2]
mainimage = images/install.img

[tree]
arch = x86_64
build_timestamp = 1571146127
platforms = x86_64,xen
variants = BaseOS

[variant-BaseOS]
id = BaseOS
name = BaseOS
packages = Packages
repository = .
type = variant
uid = BaseOS
```

18. **/var/www/html/pub/satellite-import/content/dist/rhel9/9.0/x86_64/appstream/kickstart/treeinfo** ファイルを開いて編集します。

19. **[general]** セクションで、以下の変更を加えます。

    - **packagedir = AppStream/Packages** を **packagedir = Packages** に変更します。

- **repository = AppStream** を **repository = .** に変更します。
- **variants = AppStream,BaseOS** を **variants = AppStream** に変更します。

20. **[tree]** セクションで、**variants = AppStream,BaseOS** を **variants = AppStream** に変更します。
21. **[variant-AppStream]** セクションで、以下の変更を加えます。
    - **packages = AppStream/Packages** を **packages = Packages** に変更します。
    - **repository = AppStream** を **repository = .** に変更します。
22. ファイルから次のセクションを削除します: **[checksums]**、**[images-x86_64]**、**[images-xen]**、**[media]**、**[stage2]**、**[variant-BaseOS]**。
23. ファイルを保存してから閉じます。
24. **/var/www/html/pub/satellite-import/content/dist/rhel9/9.0/x86_64/appstream/kickstart/treeinfo** ファイルが以下の形式であることを確認します。

```
[general]
; WARNING.0 = This section provides compatibility with pre-productmd treeinfos.
; WARNING.1 = Read productmd documentation for details about new format.
arch = x86_64
family = Red Hat Enterprise Linux
name = Red Hat Enterprise Linux 9.0.0
packagedir = Packages
platforms = x86_64,xen
repository = .
timestamp = 1571146127
variant = AppStream
variants = AppStream
version = 9.0.0

[header]
type = productmd.treeinfo
version = 1.2

[release]
name = Red Hat Enterprise Linux
short = RHEL
version = 9.0.0

[tree]
arch = x86_64
build_timestamp = 1571146127
platforms = x86_64,xen
variants = AppStream

[variant-AppStream]
id = AppStream
name = AppStream
packages = Packages
```

```
repository = .
type = variant
uid = AppStream
```

25. マウントされたバイナリー DVD ISO イメージを使用する予定がない場合は、ディレクトリーをアンマウントして削除します。

```
# umount /mnt/iso
# rmdir /mnt/iso
```

26. Satellite Web UI で、キックスタートリポジトリーを有効にします。

B.2. RED HAT ENTERPRISE LINUX 8 のキックスタートリポジトリーのインポート

以下の手順を使用して、Red Hat Enterprise Linux 8 のキックスタートリポジトリーをインポートします。

手順

1. Red Hat カスタマーポータル access.redhat.com/downloads に移動し、ログインします。
2. **Red Hat Enterprise Linux**をクリックします。
3. 一覧から製品バリアントと製品バージョンを選択します。(例: 製品バリアント **Red Hat Enterprise Linux for x86_64** および製品バージョン **8.1**)
4. **Red Hat Enterprise Linux 8.1 Binary DVD**など、完全インストールイメージを見つけ、**Download Now** をクリックします。
5. ダウンロードが完了したら、ISO イメージを Satellite Server にコピーします。
6. Satellite Server で、マウントポイントを作成し、そのロケーションに ISO イメージを一時的にマウントします。

```
# mkdir /mnt/iso
# mount -o loop rhel-binary-dvd.iso /mnt/iso
```

rhel-binary-dvd.iso は、ISO イメージの名前に置き換えます。

7. Red Hat Enterprise Linux 8 AppStream および BaseOS Kickstart リポジトリーのディレクトリーを作成します。

```
# mkdir --parents /var/www/html/pub/satellite-
import/content/dist/rhel8/8.1/x86_64/appstream/kickstart
# mkdir --parents /var/www/html/pub/satellite-
import/content/dist/rhel8/8.1/x86_64/baseos/kickstart
```

8. ISO イメージから **kickstart** ファイルをコピーします。

```
# cp -a /mnt/iso/AppStream/* /var/www/html/pub/satellite-
import/content/dist/rhel8/8.1/x86_64/appstream/kickstart
# cp -a /mnt/iso/BaseOS/* /mnt/iso/images/ /var/www/html/pub/satellite-
import/content/dist/rhel8/8.1/x86_64/baseos/kickstart
```

BaseOS の場合は、**/mnt/iso/images/** ディレクトリーのコンテンツもコピーする必要があることに注意してください。

9. 次のエントリーをリストファイルに追加します。
/var/www/html/pub/satellite-import/content/dist/rhel8/8.1/x86_64/appstream/listing ファイルで改行して **kickstart** を追加します。

   /var/www/html/pub/satellite-import/content/dist/rhel8/8.1/x86_64/baseos/listing ファイルで改行して **kickstart** を追加します。

   /var/www/html/pub/satellite-import/content/dist/rhel8/listing ファイルで改行してバージョン番号を追加します。たとえば、Red Hat Enterprise Linux 8.1 バイナリー ISO の場合は **8.1** を追加します。

10. ISO イメージから **.treeinfo** ファイルをコピーします。

```
# cp /mnt/iso/.treeinfo /var/www/html/pub/satellite-
import/content/dist/rhel8/8.1/x86_64/appstream/kickstart/treeinfo
# cp /mnt/iso/.treeinfo /var/www/html/pub/satellite-
import/content/dist/rhel8/8.1/x86_64/baseos/kickstart/treeinfo
```

11. **/var/www/html/pub/satellite-import/content/dist/rhel8/8.1/x86_64/baseos/kickstart/treeinfo** ファイルを開いて編集します。

12. **[general]** セクションで、以下の変更を加えます。

    - **packagedir = AppStream/Packages** を **packagedir = Packages** に変更します。
    - **repository = AppStream** を **repository = .** に変更します。
    - **variant = AppStream** を **variant = BaseOS** に変更します。
    - **variants = AppStream,BaseOS** を **variants = BaseOS** に変更します。

13. **[tree]** セクションで、**variants = AppStream,BaseOS** を **variants = BaseOS** に変更します。

14. **[variant-BaseOS]** セクションで、以下の変更を加えます。

    - **packages = BaseOS/Packages** を **packages = Packages** に変更します。
    - **repository = BaseOS** を **repository = .** に変更します。

15. **[media]** および **[variant-AppStream]** のセクションを削除します。

16. ファイルを保存してから閉じます。

17. **/var/www/html/pub/satellite-import/content/dist/rhel8/8.1/x86_64/baseos/kickstart/treeinfo** ファイルが以下の形式であることを確認します。

```
[checksums]
images/efiboot.img =
sha256:c01c18acc6778d6e66c8d0872bac59bfd7219ccf3cfa70a5c605c0fb37f33a83
images/install.img =
sha256:ddd08e5a5d92edee150f91ff4f12f39253eae72ff496465cf1b2766fe4a4df49
images/pxeboot/initrd.img =
sha256:a09a8ec89d485d71ed1bdad83584d6d816e67448221172d9aad97886cd70adca
images/pxeboot/vmlinuz =
```

```
sha256:6e523d7c3266e26c695923ab12b2873b16b0c61fb2e48ade608ad8998821584b

[general]
; WARNING.0 = This section provides compatibility with pre-productmd treeinfos.
; WARNING.1 = Read productmd documentation for details about new format.
arch = x86_64
family = Red Hat Enterprise Linux
name = Red Hat Enterprise Linux 8.1.0
packagedir = Packages
platforms = x86_64,xen
repository = .
timestamp = 1571146127
variant = BaseOS
variants = BaseOS
version = 8.1.0

[header]
type = productmd.treeinfo
version = 1.2

[images-x86_64]
efiboot.img = images/efiboot.img
initrd = images/pxeboot/initrd.img
kernel = images/pxeboot/vmlinuz

[images-xen]
initrd = images/pxeboot/initrd.img
kernel = images/pxeboot/vmlinuz

[release]
name = Red Hat Enterprise Linux
short = RHEL
version = 8.1.0

[stage2]
mainimage = images/install.img

[tree]
arch = x86_64
build_timestamp = 1571146127
platforms = x86_64,xen
variants = BaseOS

[variant-BaseOS]
id = BaseOS
name = BaseOS
packages = Packages
repository = .
type = variant
uid = BaseOS
```

18. **/var/www/html/pub/satellite-import/content/dist/rhel8/8.1/x86_64/appstream/kickstart/treeinfo** ファイルを開いて編集します。

19. **[general]** セクションで、以下の変更を加えます。

- **packagedir = AppStream/Packages** を **packagedir = Packages** に変更します。
- **repository = AppStream** を **repository = .** に変更します。
- **variants = AppStream,BaseOS** を **variants = AppStream** に変更します。

20. **[tree]** セクションで、**variants = AppStream,BaseOS** を **variants = AppStream** に変更します。
21. **[variant-AppStream]** セクションで、以下の変更を加えます。
    - **packages = AppStream/Packages** を **packages = Packages** に変更します。
    - **repository = AppStream** を **repository = .** に変更します。
22. ファイルから次のセクションを削除します: **[checksums]**、**[images-x86_64]**、**[images-xen]**、**[media]**、**[stage2]**、**[variant-BaseOS]**。
23. ファイルを保存してから閉じます。
24. **/var/www/html/pub/satellite-import/content/dist/rhel8/8.1/x86_64/appstream/kickstart/treeinfo** ファイルが以下の形式であることを確認します。

```
[general]
; WARNING.0 = This section provides compatibility with pre-productmd treeinfos.
; WARNING.1 = Read productmd documentation for details about new format.
arch = x86_64
family = Red Hat Enterprise Linux
name = Red Hat Enterprise Linux 8.1.0
packagedir = Packages
platforms = x86_64,xen
repository = .
timestamp = 1571146127
variant = AppStream
variants = AppStream
version = 8.1.0

[header]
type = productmd.treeinfo
version = 1.2

[release]
name = Red Hat Enterprise Linux
short = RHEL
version = 8.1.0

[tree]
arch = x86_64
build_timestamp = 1571146127
platforms = x86_64,xen
variants = AppStream

[variant-AppStream]
id = AppStream
name = AppStream
packages = Packages
```

```
repository = .
type = variant
uid = AppStream
```

25. マウントされたバイナリー DVD ISO イメージを使用する予定がない場合は、ディレクトリーをアンマウントして削除します。

```
# umount /mnt/iso
# rmdir /mnt/iso
```

26. Satellite Web UI で、キックスタートリポジトリーを有効にします。

B.3. RED HAT ENTERPRISE LINUX 7 のキックスタートリポジトリーのインポート

以下の手順を使用して、Red Hat Enterprise Linux 7 のキックスタートリポジトリーをインポートします。

手順

1. Red Hat カスタマーポータル access.redhat.com/downloads に移動し、ログインします。
2. **Red Hat Enterprise Linux**をクリックします。
3. **Product Variant** リストの上にある **Switch to version 7 and below**をクリックします。
4. 一覧から製品バリアントと製品バージョンを選択します。(例: 製品バリアント **Red Hat Enterprise Linux for x86_64** および製品バージョン **7.9**)
5. **Red Hat Enterprise Linux 7.9 Binary DVD**など、完全インストールイメージを見つけ、**Download Now** をクリックします。
6. ダウンロードが完了したら、ISO イメージを Satellite Server にコピーします。
7. Satellite Server で、マウントポイントを作成し、そのロケーションに ISO イメージを一時的にマウントします。

```
# mkdir /mnt/iso
# mount -o loop rhel-binary-dvd.iso /mnt/iso
```

rhel-binary-dvd.iso は、ISO イメージの名前に置き換えます。

8. kickstart ディレクトリーを作成します。

```
# mkdir --parents /var/www/html/pub/satellite-
import/content/dist/rhel/server/7/7.9/x86_64/kickstart/
```

9. ISO イメージから **kickstart** ファイルをコピーします。

```
# cp -a /mnt/iso/* /var/www/html/pub/satellite-
import/content/dist/rhel/server/7/7.9/x86_64/kickstart/
```

10. 次のエントリーをリストファイルに追加します。

/var/www/html/pub/satellite-import/content/dist/rhel/server/7/listing ファイルで改行してバージョン番号を追加します。たとえば、Red Hat Enterprise Linux 7.9 バイナリー ISO の場合は **7.9** を追加します。

/var/www/html/pub/satellite-import/content/dist/rhel/server/7/7.9/listing ファイルで改行してアーキテクチャーを追加します。(例: **x86_64**)

/var/www/html/pub/satellite-import/content/dist/rhel/server/7/7.9/x86_64/listing ファイルで改行して **kickstart** を追加します。

11. ISO イメージから **.treeinfo** ファイルをコピーします。

```
# cp /mnt/iso/.treeinfo /var/www/html/pub/satellite-
import/content/dist/rhel/server/7/7.9/x86_64/kickstart/treeinfo
```

12. マウントされたバイナリー DVD ISO イメージを使用する予定がない場合は、ディレクトリーをアンマウントして削除します。

```
# umount /mnt/iso
# rmdir /mnt/iso
```

13. Satellite Web UI で、キックスタートリポジトリーを有効にします。