



U Y U N I

Uyuni 2022.07

クライアント設定ガイド

2022年07月27日



目次

クライアント設定ガイドの概要	1
1. サポートされているクライアントと機能	2
1.1. サポートされているクライアントシステム	2
1.2. サポートされているツールパッケージ	2
1.3. サポートされているSUSEおよびopenSUSEクライアントの機能	3
1.4. サポートされているSUSE Linux Enterprise Server with Expanded Supportの機能	5
1.5. サポートされているSLE MicroおよびopenSUSE MicroOSクライアントの機能	7
1.6. サポートされているAlibaba Cloud Linuxの機能	9
1.7. サポートされているAlmaLinuxの機能	11
1.8. サポートされているAmazon Linuxの機能	13
1.9. サポートされているCentOSの機能	15
1.10. サポートされているDebianの機能	18
1.11. サポートされているOracleの機能	20
1.12. サポートされているRed Hat Enterprise Linuxの機能	22
1.13. サポートされているRocky Linuxの機能	25
1.14. サポートされているUbuntuの機能	27
2. 設定の基本	30
2.1. ソフトウェアチャンネル	30
2.1.1. SUSE Package Hubで提供されるパッケージ	31
2.1.2. AppStreamで提供されるパッケージ	31
2.1.3. EPELで提供されるパッケージ	31
2.1.4. Unified Installer Updates Channels on SUSE Linux Enterprise Clients	32
2.1.5. ソフトウェアリポジトリ	32
2.1.6. ソフトウェア製品	33
2.2. ブートストラップリポジトリ	33
2.2.1. ブートストラップリポジトリの作成準備	33
2.2.2. 自動モードのオプション	34
2.2.3. ブートストラップリポジトリの手動生成	35
2.2.4. ブートストラップとカスタムチャンネル	36
2.3. アクティベーションキー	36
2.3.1. 複数のアクティベーションキーの結合	38
2.3.2. 再アクティベーションキー	39
2.3.3. アクティベーションキーのベストプラクティス	40
2.4. GPGキー	41
2.4.1. クライアントでGPGキーを信頼する	41
3. クライアントの管理方法	44
3.1. Saltクライアントの接続メソッド	44
3.1.1. オンボードの詳細	44
3.1.2. Salt SSHでのプッシュ	44
3.1.3. Salt Bundle	47
3.2. 従来のクライアントの接続メソッド	49
3.2.1. SUSE Managerデーモン(rhnsd)	49
3.2.2. SSHでのプッシュ	53
4. クライアントの登録	58
4.1. クライアント登録メソッド	58
4.1.1. Web UIでクライアントを登録する	58
4.1.2. ブートストラップスクリプトを使用してクライアントを登録する	60
4.1.3. コマンドラインで登録する(Salt)	64
4.2. SUSEクライアントの登録	67
4.2.1. SUSE Linux Enterpriseクライアントの登録	67
4.2.2. SLE Microクライアントの登録	72
4.2.3. SUSE Linux Enterprise Server with Expanded Supportクライアントの登録	74

4.3. openSUSEクライアントの登録	79
4.3.1. openSUSE Leapクライアントの登録	79
4.3.2. openSUSE MicroOSクライアントの登録	83
4.4. Alibaba Cloud Linuxクライアントの登録	85
4.4.1. Alibaba Cloud Linuxクライアントの登録	86
4.5. AlmaLinuxクライアントの登録	88
4.5.1. AlmaLinuxクライアントの登録	88
4.6. Amazon Linuxクライアントの登録	92
4.6.1. Amazon Linuxクライアントの登録	92
4.7. CentOSクライアントの登録	96
4.7.1. CentOSクライアントの登録	96
4.8. Debianクライアントの登録	103
4.8.1. Debianクライアントの登録	103
4.9. Oracleクライアントの登録	108
4.9.1. Oracle Linuxクライアントの登録	108
4.10. Red Hatクライアントの登録	113
4.10.1. CDNでRed Hat Enterprise Linuxクライアントを登録する	113
4.10.2. RHUIでRed Hat Enterprise Linuxクライアントを登録する	123
4.11. Rocky Linuxクライアントの登録	133
4.11.1. Rocky Linuxクライアントの登録	133
4.12. Ubuntuクライアントの登録	138
4.12.1. Registering Ubuntu 20.04 and 22.04 Clients	138
4.12.2. Ubuntu 16.04および18.04クライアントの登録	142
4.13. クライアントをプロキシに登録する	147
4.13.1. プロキシ間でのクライアントの移動	147
4.13.2. プロキシからサーバへのクライアントの移動	148
4.13.3. Web UIを使用してクライアントをプロキシに登録する	149
4.13.4. ブートストラップスクリプトを使用して登録する(Saltと従来版)	150
4.14. パブリッククラウドでのクライアントの登録	151
4.14.1. 製品の追加とリポジトリの同期	151
4.14.2. オンデマンドイメージの準備	151
4.14.3. クライアントの登録	152
4.14.4. アクティベーションキー	153
4.14.5. Terraformによって作成されたクライアントの自動登録	153
5. クライアントのアップグレード	156
5.1. クライアント - メジャーバージョンのアップグレード	156
5.1.1. マイグレーションの準備	156
5.1.2. 自動インストールプロファイルの作成	158
5.1.3. 移行	159
5.2. コンテンツライフサイクルマネージャを使用したアップグレード	159
5.2.1. アップグレードの準備	160
5.2.2. アップグレード	161
5.3. 製品移行	162
5.3.1. 移行の実行	162
5.3.2. 製品の大量移行	163
5.4. Uyuniクライアントのアップグレード	167
5.4.1. アップグレードの準備	167
5.4.2. アップグレード	167
6. クライアントの削除	169
7. クライアントの操作	170
7.1. パッケージ管理	170
7.1.1. パッケージの検証	170
7.1.2. パッケージの比較	170
7.2. パッチ管理	171
7.2.1. パッチの作成	171

7.2.2. クライアントへのパッチの適用	173
7.3. システムのロック	174
7.3.1. 従来のクライアントのシステムのロック	174
7.3.2. Saltクライアントのシステムのロック	174
7.3.3. パッケージのロック	175
7.4. 設定管理	176
7.4.1. 設定管理用に従来のクライアントを準備する	177
7.4.2. 設定チャンネルの作成	178
7.4.3. 設定ファイル、ディレクトリ、またはシンボリックリンクの追加	178
7.4.4. クライアントを設定チャンネルにサブスクライブする	179
7.4.5. 設定ファイルの比較	179
7.4.6. 従来のクライアントにおける設定ファイルのマクロ	180
7.5. 電源管理	181
7.5.1. 電源管理とCobbler	182
7.6. 設定のスナップショット	182
7.6.1. スナップショットタグ	183
7.6.2. 大規模インストールのスナップショット	183
7.7. カスタムシステム情報	183
7.8. システムセットマネージャ	184
7.8.1. SSMでベースチャンネルを変更する	186
7.9. システムグループ	186
7.9.1. グループの作成	186
7.9.2. グループにクライアントを追加する	187
7.9.3. グループの操作	188
7.10. システムの種類	188
7.10.1. Web UIを使用して従来のクライアントをSaltに変更する	188
7.10.2. コマンドプロンプトを使用して従来のクライアントをSaltに変更する	189
8. オペレーティングシステムのインストール	190
8.1. 登録済みシステムを再インストールする	191
8.2. ネットワークを通じてインストールする(PXEブート)	192
8.2.1. DHCPサーバを準備する	193
8.2.2. プロキシを使用してTFTPツリーを同期する	194
8.3. CD-ROMまたはUSBメモリを使用してインストールする	195
8.3.1. CobblerでISOイメージを構築する	195
8.3.2. KIWIでSUSE ISOイメージを構築する	196
8.3.3. mkisofsでRedHat ISOイメージを構築する	196
8.4. 自動インストールのディストリビューション	197
8.4.1. ISOイメージに基づくディストリビューション	198
8.4.2. RPMパッケージに基づくディストリビューション	198
8.4.3. 自動インストールのディストリビューションを宣言する	199
8.5. 自動インストールプロファイル	199
8.5.1. プロファイルを宣言する	200
8.5.2. AutoYastプロファイル	202
8.5.3. キックスタートプロファイル	202
8.5.4. テンプレートの構文	203
8.6. 無人プロビジョニング	205
8.6.1. ベアメタルプロビジョニング	206
8.6.2. システムレコードを手動で作成する	206
8.7. 独自のGPGキーを使用する	207
8.7.1. PXEブート用の独自のGPGキー	208
8.7.2. CD-ROM内の独自のGPGキー	208
9. 仮想化	210
9.1. 仮想化ホストの管理	210
9.2. 仮想ゲストの作成	210
9.3. XenおよびKVMを使用した仮想化	211

9.3.1. ホストの設定	212
9.3.2. 自動インストール	213
9.3.3. VMゲストの管理	217
10. 仮想ホストマネージャ	218
10.1. VHMおよびAmazon Web Services	218
10.1.1. Amazon EC2 VHMの作成	218
10.1.2. 仮想ホストマネージャのAWS許可	219
10.2. VHMとAzure	220
10.2.1. 前提条件	220
10.2.2. Azure VHMの作成	220
10.2.3. パーミッションの割り当て	221
10.2.4. Azure UUID	222
10.3. VHMおよびGoogle Compute Engine	222
10.3.1. 前提条件	222
10.3.2. GCE VHMの作成	222
10.3.3. パーミッションの割り当て	223
10.3.4. GCE UUID	224
10.4. VHMとKubernetes	224
10.4.1. Kubernetes VHMの作成	224
10.4.2. イメージランタイムデータの取得	225
10.4.3. パーミッションと証明書	227
10.5. Nutanixによる仮想化	228
10.5.1. VHMの設定	228
10.6. VMWareによる仮想化	229
10.6.1. VHMの設定	229
10.6.2. VMWareでのSSLエラーのトラブルシューティング	230
10.7. その他のサードパーティプロバイダを使用した仮想化	231
11. クライアントのトラブルシューティング	234
11.1. 自動インストール	234
11.2. ベアメタルシステム	234
11.3. サポートが終了したCentOS 6クライアントのブートストラップ	235
11.4. サポート終了製品のブートストラップリポジトリ	236
11.5. 複製したSaltクライアント	236
11.6. FQDNS grainの無効化	237
11.7. noexecで/tmpをマウントする	237
11.8. grainを渡してイベントを開始する	237
11.9. プロキシの接続およびFQDN	238
11.10. Red Hat CDNチャンネルと複数の証明書	238
11.11. Web UIからの登録は失敗するが、エラーが表示されない	240
11.12. 古いクライアントの登録	240
11.13. ダウンとして表示されるSaltクライアントおよびDNS設定	241
12. GNU Free Documentation License	243

クライアント設定ガイドの概要

更新: 2022-07-27

クライアントの登録は、Uyuniインストール後の最初の手順であり、Uyuniで費やす時間のほとんどは、これらのクライアントを管理する時間です。

Uyuniは、広範なクライアント技術と互換性があります。広範なハードウェアオプションを使用して、従来のクライアントまたはSaltクライアントをインストールし、SUSE Linux Enterpriseまたはその他のLinuxオペレーティングシステムを実行できます。

サポートされているクライアントおよび機能の一覧については、[Client-configuration > Supported-features](#)を参照してください。

このガイドでは、異なるクライアントを登録して設定する方法に関して手動の方法と自動の方法の両方について説明します。

Chapter 1. サポートされているクライアントと機能

Uyuniは、さまざまなクライアント技術と互換性があります。 広範なハードウェアオプションを使用して、従来のクライアントまたはSaltクライアントをインストールし、SUSE Linux Enterpriseまたは他のLinuxオペレーティングシステムを実行できます。

このセクションには、サポートされているクライアントシステムのまとめが含まれています。それぞれのクライアントで使用できる機能の詳細な一覧については、次のページを参照してください。

1.1. サポートされているクライアントシステム

従来のクライアントおよびSaltクライアントでサポートされているオペレーティングシステムを次の表に示します。

この表のアイコンに意味は次のとおりです。

- ✓ このオペレーティングシステムを実行しているクライアントはSUSEでサポートされています。
- ✗ このオペレーティングシステムを実行しているクライアントはSUSEではサポートされていません。
- ? クライアントは検討中であり、後日使用できる場合と、使用できない場合があります。



クライアントオペレーティングシステムのバージョンおよびSPレベルは、Uyuniでサポートされる全般的なサポート(通常またはLTSS)の条件を基準にする必要があります。サポートされている製品バージョンの詳細については、<https://www.suse.com/lifecycle>を参照してください。



クライアントで実行されているオペレーティングシステムは、オペレーティングシステムを提供している組織によってサポートされています。

Unresolved directive in modules/client-configuration/pages/supported-features.adoc
include::../../../snippets/pages/supported-client-systems-snippet.adoc[]



DebianとUbuntuは、x86-64アーキテクチャをamd64としてリストします。

1.2. サポートされているツールパッケージ

表 1. Spacewalkのユーティリティ

ツール名	説明	サポートの有無
spacewalk-common-channels	SUSE Customer Center で提供されないチャンネルを追加します	✓
spacewalk-hostname-rename	Uyuniサーバのホスト名を変更します	✓
spacewalk-clone-by-date	特定の日までにチャンネルを複製します	✓

ツール名	説明	サポートの有無
<code>spacewalk-sync-setup</code>	ISSマスターおよびスレーブの組織マッピングを設定します	✓
<code>spacewalk-manage-channel-lifecycle</code>	チャンネルのライフサイクルを管理します	✓

1.3. サポートされているSUSEおよびopenSUSEクライアントの機能

この表には、SUSEおよびopenSUSEクライアントのさまざまな機能の使用可否がリストされています。この表では、SLES、SLED、SUSE Linux Enterprise Server for SAP、SUSE Linux Enterprise Server for HPCなど、SUSE Linux Enterpriseオペレーティングシステムのすべての亜種について記載しています。



クライアントで実行しているオペレーティングシステムは、オペレーティングシステムを提供している組織によってサポートされています。SUSE Linux EnterpriseはSUSEでサポートされています。openSUSEはSUSEコミュニティでサポートされています。

この表のアイコンに意味は次のとおりです。

- ✓: 機能はSaltクライアントと従来のクライアントの両方で使用できます
- ✗: 機能は使用できません
- ?: 機能は検討中であり、後日使用できる場合と、使用できない場合があります
- Traditional: 機能は従来のクライアントでのみサポートされています
- Salt: 機能はSaltクライアントでのみサポートされています。

表 2. SUSEおよびopenSUSEオペレーティングシステムでサポートされている機能

Feature	SUSE Linux Enterprise 12	SUSE Linux Enterprise 15	openSUSE 15
Client	✓	✓	✓
System packages	SUSE	SUSE	openSUSE Community
Registration	✓	✓	Salt
Install packages	✓	✓	Salt
Apply patches	✓	✓	Salt
Remote commands	✓	✓	Salt
System package states	Salt	Salt	Salt
System custom states	Salt	Salt	Salt
Group custom states	Salt	Salt	Salt
Organization custom states	Salt	Salt	Salt

Feature	SUSE Linux Enterprise 12	SUSE Linux Enterprise 15	openSUSE 15
System set manager (SSM)	✓	✓	Salt
Product migration	✓	✓	Salt
Basic Virtual Guest Management *	✓	✓	Salt
Advanced Virtual Guest Management *	Salt	Salt	Salt
Virtual Guest Installation (AutoYaST), as Host OS	Traditional	Traditional	✗
Virtual Guest Installation (image template), as Host OS	Salt	Salt	Salt
Virtual Guest Management	Salt	Salt	Salt
System deployment (PXE/AutoYaST)	✓	✓	✓
System redeployment (AutoYaST)	✓	✓	Salt
Contact methods	Traditional: OSAD, RHNSD, SSH-push. Salt: ZeroMQ, Salt-SSH	Traditional: OSAD, RHNSD, SSH-push. Salt: ZeroMQ, Salt-SSH	Salt: ZeroMQ, Salt-SSH
Works with Uyuni Proxy	✓	✓	Salt
Action chains	✓	✓	Salt
Staging (pre-download of packages)	✓	✓	Salt
Duplicate package reporting	✓	✓	Salt
CVE auditing	✓	✓	Salt
SCAP auditing	✓	✓	Salt
Package verification	Traditional	Traditional	✗
Package locking	Salt	Salt	Salt
System locking	Traditional	Traditional	✗
Maintenance Windows	✓	✓	✓
System snapshot	Traditional	Traditional	✗
Configuration file management	✓	✓	Salt

Feature	SUSE Linux Enterprise 12	SUSE Linux Enterprise 15	openSUSE 15
Package profiles	Traditional. Salt: Profiles supported, Sync not supported	Traditional. Salt: Profiles supported, Sync not supported	Salt: Profiles supported, Sync not supported
Power management	✓	✓	✓
Monitoring	Salt	Salt	Salt
Docker buildhost	Salt	Salt	?
Build Docker image with OS	Salt	Salt	Salt
Kiwi buildhost	Salt	?	?
Build Kiwi image with OS	Salt	?	✗
Recurring Actions	Salt	Salt	Salt
AppStreams	N/A	N/A	N/A
Yomi	✗	✓	✓

*仮想ゲスト管理:

この表では、仮想ゲスト管理は基本と高度に分割されています。

基本的な仮想ゲスト管理には、VMのリスト化、低速更新、VMのライフサイクルアクション(起動、停止、再開、一時停止)、およびVM vCPUとメモリの変更が含まれています。

高度な仮想ゲスト管理には、基本的な仮想ゲスト管理のすべての機能に加えて、高速更新、VMライフサイクルアクション(削除、リセット、電源オフ)、VMディスクの変更、ネットワークグラフィカル表示の変更、およびグラフィカル表示の設定が含まれています。

1.4. サポートされているSUSE Linux Enterprise Server with Expanded Supportの機能

この表には、SUSE Linux Enterprise Server with Expanded Supportクライアントのさまざまな機能の使用可否がリストされています。



クライアントで実行しているオペレーティングシステムは、オペレーティングシステムを提供している組織によってサポートされています。 SUSE Linux Enterprise Server with Expanded SupportはSUSEでサポートされています。

この表のアイコンに意味は次のとおりです。

- ✓: 機能はSaltクライアントと従来のクライアントの両方で使用できます
- ✗: 機能は使用できません
- ?: 機能は検討中であり、後日使用できる場合と、使用できない場合があります

- Traditional: 機能は従来のクライアントでのみサポートされています
- Salt: 機能はSaltクライアントでのみサポートされています。

表 3. SUSE Linux Enterprise Server with Expanded Supportオペレーティングシステムでサポートされている機能

機能	SLES ES 7	SLES ES 8
クライアント	✓	Salt
システムパッケージ	SUSE	SUSE
登録	✓	Salt
パッケージのインストール	✓	Salt
パッチの適用	✓	Salt
リモートコマンド	✓	Salt
システムパッケージの状態	Salt	Salt
システムカスタムの状態	Salt	Salt
グループカスタムの状態	Salt	Salt
組織カスタムの状態	Salt	Salt
システムセットマネージャ(SSM)	Salt	Salt
製品移行	なし	なし
基本的な仮想ゲスト管理*	✓	Salt
高度な仮想ゲスト管理*	Salt	Salt
仮想ゲストインストール(キックスタート)、ホストOSとして	Traditional	✗
仮想ゲストインストール(イメージテンプレート)、ホストOSとして	✓	Salt
システムの展開(PXE/キックスタート)	✓	Salt
システムの再展開(キックスタート)	✓	✗
接続メソッド	Traditional: OSAD、RHNSD、SSH-push. Salt: ZeroMQ、Salt-SSH	Salt: ZeroMQ、Salt-SSH
Uyuniプロキシでの操作	✓	Salt
アクションチェーン	✓	Salt
ステージング(パッケージの事前ダウンロード)	✓	Salt
重複パッケージの報告	✓	Salt
CVE監査	✓	Salt
SCAP監査	✓	Salt
パッケージの確認	Traditional	✗

機能	SLES ES 7	SLES ES 8
パッケージのロック	✓	?
システムのロック	Traditional	✗
保守ウィンドウ	✓	✓
システムのスナップショット	Traditional	Salt
設定ファイルの管理	✓	Salt
スナップショットとプロファイル	Traditional. Salt: プロファイルはサポートされていますが、同期はサポートされていません	Salt: プロファイルはサポートされていますが、同期はサポートされていません
電源管理	✓	Salt
監視機能	Salt	Salt
Docker buildhost	✗	✗
OSでのDockerイメージの構築	?	?
Kiwi buildhost	✗	✗
OSでのKiwiイメージの構築	✗	✗
繰り返しアクション	Salt	Salt
AppStream	なし	✓
Yomi	なし	なし

*仮想ゲスト管理:

この表では、仮想ゲスト管理は基本と高度に分割されています。

基本的な仮想ゲスト管理には、VMのリスト化、低速更新、VMのライフサイクルアクション(起動、停止、再開、一時停止)、およびVM vCPUとメモリの変更が含まれています。

高度な仮想ゲスト管理には、基本的な仮想ゲスト管理のすべての機能に加えて、高速更新、VMライフサイクルアクション(削除、リセット、電源オフ)、VMディスクの変更、ネットワークグラフィカル表示の変更、およびグラフィカル表示の設定が含まれています。

1.5. サポートされているSLE MicroおよびopenSUSE MicroOSクライアントの機能



SLE MicroおよびopenSUSE MicroOS クライアントのサポートは、テスト目的のテクノロジプレビューとして提供されるものであり、この段階ですべての機能が完全に動作するわけではありません。この機能は、Uyuniの後続バージョンですべてサポートされる予定です。運用システムでは、この機能を使用しないでください。



クライアントで実行しているオペレーティングシステムは、オペレーティングシステムを提供している組織によってサポートされています。 SLE MicroはSUSEでサポートされています。 openSUSE MicroOSはSUSEコミュニティでサポートされています。

この表のアイコンに意味は次のとおりです。

- ✓: 機能はSaltクライアントと従来のクライアントの両方で使用できます
- ✗: 機能は使用できません
- ?: 機能は検討中であり、後日使用できる場合と、使用できない場合があります
- Traditional: 機能は従来のクライアントでのみサポートされています
- Salt: 機能はSaltクライアントでのみサポートされています。

表4. SLE MicroおよびopenSUSE MicroOSオペレーティングシステムでサポートされている機能

機能	SLE MicroおよびopenSUSE MicroOS
クライアント	Salt
オペレーティングシステムパッケージ	Salt
登録	Salt
パッケージのインストール	Salt
パッチの適用(CVE IDが必要)	Salt
リモートコマンド	Salt
システムパッケージの状態	Salt
システムカスタムの状態	Salt
グループカスタムの状態	Salt
組織カスタムの状態	Salt
システムセットマネージャ(SSM)	Salt
製品の移行	?
基本的な仮想ゲスト管理*	?
高度な仮想ゲスト管理*	?
仮想ゲストインストール(キックスタート)、ホストOSとして	✗
仮想ゲストインストール(イメージテンプレート)、ホストOSとして	?
システムの配備(PXE/キックスタート)	?
システムの再配備(キックスタート)	✗
接続メソッド	Salt: ZeroMQ
Uyuniプロキシでの操作	Salt

機能	SLE MicroおよびopenSUSE MicroOS
アクションチェーン	?
ステージング(パッケージの事前ダウンロード)	?
重複パッケージの報告	Salt
CVE監査(CVE IDが必要)	Salt
SCAP監査	?
パッケージの確認	?
パッケージのロック	Salt
システムのロック	?
保守ウィンドウ	?
システムのスナップショット	×
設定ファイルの管理	Salt
スナップショットとプロファイル	Salt: プロファイルはサポートされていますが、同期 はサポートされていません
電源管理	Salt
監視機能	Salt
Docker buildhost	×
OSでのDockerイメージの構築	×
Kiwi buildhost	×
OSでのKiwiイメージの構築	Salt
繰り返しアクション	Salt
AppStreams	なし
Yomi	?

*仮想ゲスト管理:

この表では、仮想ゲスト管理は基本と高度に分割されています。

基本的な仮想ゲスト管理には、VMのリスト化、低速更新、VMのライフサイクルアクション(起動、停止、再開、一時停止)、およびVM vCPUとメモリの変更が含まれています。

高度な仮想ゲスト管理には、基本的な仮想ゲスト管理のすべての機能に加えて、高速更新、VMライフサイクルアクション(削除、リセット、電源オフ)、VMディスクの変更、ネットワークグラフィカル表示の変更、およびグラフィカル表示の設定が含まれています。

1.6. サポートされているAlibaba Cloud Linuxの機能

この表には、Alibaba Cloud Linuxクライアントのさまざまな機能の使用可否がリストされています。



クライアントで実行しているオペレーティングシステムは、オペレーティングシステムを提供している組織によってサポートされています。 Alibaba Cloud Linux はAlibaba Cloudでサポートされています。

この表のアイコンに意味は次のとおりです。

- ✓: 機能はSaltクライアントと従来のクライアントの両方で使用できます
- ✗: 機能は使用できません
- ?: 機能は検討中であり、後日使用できる場合と、使用できない場合があります
- Traditional: 機能は従来のクライアントでのみサポートされています
- Salt: 機能はSaltクライアントでのみサポートされています

表 5. Alibaba Cloud Linuxオペレーティングシステムでサポートされている機能

機能	Alibaba Cloud Linux 2
クライアント	Salt
オペレーティングシステムパッケージ	Salt
登録	Salt
パッケージのインストール	Salt
パッチの適用(CVE IDが必要)	Salt
リモートコマンド	Salt
システムパッケージの状態	Salt
システムカスタムの状態	Salt
グループカスタムの状態	Salt
組織カスタムの状態	Salt
システムセットマネージャ(SSM)	Salt
製品移行	なし
基本的な仮想ゲスト管理*	?
高度な仮想ゲスト管理*	?
仮想ゲストインストール(キックスタート)、ホストOSとして	✗
仮想ゲストインストール(イメージテンプレート)、ホストOSとして	?
システムの展開(PXE/キックスタート)	?
システムの再展開(キックスタート)	?
接続メソッド	Salt: ZeroMQ、Salt-SSH
Uyuniプロキシでの操作	Salt
アクションチェーン	Salt

機能	Alibaba Cloud Linux 2
ステージング(パッケージの事前ダウンロード)	Salt
重複パッケージの報告	Salt
CVE監査(CVE IDが必要)	Salt
SCAP監査	Salt
パッケージの確認	✗
パッケージのロック	✗
システムのロック	✗
保守ウィンドウ	✓
システムのスナップショット	✗
設定ファイルの管理	Salt
スナップショットとプロファイル	Salt: プロファイルはサポートされていますが、同期 はサポートされていません
電源管理	?
監視機能	Salt
Docker buildhost	Salt
OSでのDockerイメージの構築	Salt
Kiwi buildhost	Salt
OSでのKiwiイメージの構築	Salt
繰り返しアクション	Salt
AppStreams	なし
Yomi	なし

*仮想ゲスト管理:

この表では、仮想ゲスト管理は基本と高度に分割されています。

基本的な仮想ゲスト管理には、VMのリスト化、低速更新、VMのライフサイクルアクション(起動、停止、再開、一時停止)、およびVM vCPUとメモリの変更が含まれています。

高度な仮想ゲスト管理には、基本的な仮想ゲスト管理のすべての機能に加えて、高速更新、VMライフサイクルアクション(削除、リセット、電源オフ)、VMディスクの変更、ネットワークグラフィカル表示の変更、およびグラフィカル表示の設定が含まれています。

*従来のスタックはAlibaba Cloud Linuxで利用できますが、サポートされていません。

1.7. サポートされているAlmaLinuxの機能

この表には、AlmaLinuxクライアントのさまざまな機能の使用可否がリストされています。



クライアントで実行しているオペレーティングシステムは、オペレーティングシステムを提供している組織によってサポートされています。 AlmaLinuxはAlmaLinuxコミュニティでサポートされています。

この表のアイコンに意味は次のとおりです。

- ✓: 機能はSaltクライアントと従来のクライアントの両方で使用できます
- ✗: 機能は使用できません
- ?: 機能は検討中であり、後日使用できる場合と、使用できない場合があります
- Traditional: 機能は従来のクライアントでのみサポートされています
- Salt: 機能はSaltクライアントでのみサポートされています。

表 6. AlmaLinuxオペレーティングシステムでサポートされている機能

機能	AlmaLinux 8
クライアント	Salt (plain AlmaLinux)
システムパッケージ	AlmaLinux コミュニティ
登録	Salt
パッケージのインストール	Salt
パッチの適用	Salt
リモートコマンド	Salt
システムパッケージの状態	Salt
システムカスタムの状態	Salt
グループカスタムの状態	Salt
組織カスタムの状態	Salt
システムセットマネージャ(SSM)	Salt
製品の移行	なし
基本的な仮想ゲスト管理*	Salt
高度な仮想ゲスト管理*	Salt
仮想ゲストインストール(キックスタート)、ホストOSとして	✗
仮想ゲストインストール(イメージテンプレート)、ホストOSとして	Salt
システムの配備(PXE/キックスタート)	Salt
システムの再配備(キックスタート)	Salt
接続メソッド	Salt: ZeroMQ、Salt-SSH
Uyuniプロキシでの操作	Salt
アクションチェーン	Salt

機能	AlmaLinux 8
ステージング(パッケージの事前ダウンロード)	Salt
重複パッケージの報告	Salt
CVE監査	Salt
SCAP監査	Salt
パッケージの確認	✗
パッケージのロック	✗
システムのロック	✗
保守ウィンドウ	✓
システムのスナップショット	✗
設定ファイルの管理	Salt
スナップショットとプロファイル	Salt: プロファイルはサポートされていますが、同期 はサポートされていません
電源管理	Salt
監視機能	Salt
Docker buildhost	✗
OSでのDockerイメージの構築	✗
Kiwi buildhost	✗
OSでのKiwiイメージの構築	✗
繰り返しアクション	Salt
AppStreams	✓
Yomi	なし

*仮想ゲスト管理:

この表では、仮想ゲスト管理は基本と高度に分割されています。

基本的な仮想ゲスト管理には、VMのリスト化、低速更新、VMのライフサイクルアクション(起動、停止、再開、一時停止)、およびVM vCPUとメモリの変更が含まれています。

高度な仮想ゲスト管理には、基本的な仮想ゲスト管理のすべての機能に加えて、高速更新、VMライフサイクルアクション(削除、リセット、電源オフ)、VMディスクの変更、ネットワークグラフィカル表示の変更、およびグラフィカル表示の設定が含まれています。

1.8. サポートされているAmazon Linuxの機能

この表には、Amazon Linuxクライアントのさまざまな機能の使用可否がリストされています。



クライアントで実行しているオペレーティングシステムは、オペレーティングシステムを提供している組織によってサポートされています。 Amazon Linux はAmazonでサポートされています。

この表のアイコンに意味は次のとおりです。

- ✓: 機能はSaltクライアントと従来のクライアントの両方で使用できます
- ✗: 機能は使用できません
- ?: 機能は検討中であり、後日使用できる場合と、使用できない場合があります
- Traditional: 機能は従来のクライアントでのみサポートされています
- Salt: 機能はSaltクライアントでのみサポートされています

表 7. Amazon Linuxオペレーティングシステムでサポートされている機能

機能	Amazon Linux 2
クライアント	Salt
オペレーティングシステムパッケージ	Salt
登録	Salt
パッケージのインストール	Salt
パッチの適用(CVE IDが必要)	Salt
リモートコマンド	Salt
システムパッケージの状態	Salt
システムカスタムの状態	Salt
グループカスタムの状態	Salt
組織カスタムの状態	Salt
システムセットマネージャ(SSM)	Salt
製品移行	なし
基本的な仮想ゲスト管理*	?
高度な仮想ゲスト管理*	?
仮想ゲストインストール(キックスタート)、ホストOSとして	✗
仮想ゲストインストール(イメージテンプレート)、ホストOSとして	?
システムの展開(PXE/キックスタート)	?
システムの再展開(キックスタート)	?
接続メソッド	Salt: ZeroMQ、Salt-SSH
Uyuniプロキシでの操作	Salt
アクションチェーン	Salt

機能	Amazon Linux 2
ステージング(パッケージの事前ダウンロード)	Salt
重複パッケージの報告	Salt
CVE監査(CVE IDが必要)	Salt
SCAP監査	Salt
パッケージの確認	✗
パッケージのロック	✗
システムのロック	✗
保守ウィンドウ	✓
システムのスナップショット	✗
設定ファイルの管理	Salt
スナップショットとプロファイル	Salt: プロファイルはサポートされていますが、同期 はサポートされていません
電源管理	?
監視機能	Salt
Docker buildhost	Salt
OSでのDockerイメージの構築	Salt
Kiwi buildhost	Salt
OSでのKiwiイメージの構築	Salt
繰り返しアクション	Salt
AppStreams	なし
Yomi	なし

*仮想ゲスト管理:

この表では、仮想ゲスト管理は基本と高度に分割されています。

基本的な仮想ゲスト管理には、VMのリスト化、低速更新、VMのライフサイクルアクション(起動、停止、再開、一時停止)、およびVM vCPUとメモリの変更が含まれています。

高度な仮想ゲスト管理には、基本的な仮想ゲスト管理のすべての機能に加えて、高速更新、VMライフサイクルアクション(削除、リセット、電源オフ)、VMディスクの変更、ネットワークグラフィカル表示の変更、およびグラフィカル表示の設定が含まれています。

*従来のスタックはAmazon Linuxで利用できますが、サポートされていません。

1.9. サポートされているCentOSの機能

この表には、CentOSクライアントのさまざまな機能の使用可否がリストされています。



クライアントで実行しているオペレーティングシステムは、オペレーティングシステムを提供している組織によってサポートされています。 CentOSはCentOSコミュニティでサポートされています。

この表のアイコンに意味は次のとおりです。

- ✓: 機能はSaltクライアントと従来のクライアントの両方で使用できます
- ✗: 機能は使用できません
- ?: 機能は検討中であり、後日使用できる場合と、使用できない場合があります
- Traditional: 機能は従来のクライアントでのみサポートされています
- Salt: 機能はSaltクライアントでのみサポートされています。

表 8. CentOSオペレーティングシステムでサポートされている機能

機能	CentOS 7	CentOS 8
クライアント	✓ (プレーンCentOS)	Salt (プレーンCentOS)
システムパッケージ	CentOS コミュニティ	CentOS コミュニティ
登録	✓	Salt
パッケージのインストール	✓	Salt
パッチの適用(CVE IDが必要)	✓ (エラータで必要なサードパーティサービス)	Salt (エラータで必要なサードパーティサービス)
リモートコマンド	✓	Salt
システムパッケージの状態	Salt	Salt
システムカスタムの状態	Salt	Salt
グループカスタムの状態	Salt	Salt
組織カスタムの状態	Salt	Salt
システムセットマネージャ(SSM)	✓	Salt
製品移行	なし	なし
基本的な仮想ゲスト管理*	✓	Salt
高度な仮想ゲスト管理*	Salt	Salt
仮想ゲストインストール(キックスタート)、ホストOSとして	Traditional	✗
仮想ゲストインストール(イメージテンプレート)、ホストOSとして	✓	Salt
システムの展開(PXE/キックスタート)	✓	Salt
システムの再展開(キックスタート)	✓	Salt

機能	CentOS 7	CentOS 8
接続メソッド	Traditional: OSAD、RHNSD、SSH-push. Salt: ZeroMQ、Salt-SSH	Salt: ZeroMQ、Salt-SSH
Uyuniプロキシでの操作	✓	Salt
アクションチェーン	✓	Salt
ステージング(パッケージの事前ダウンロード)	✓	Salt
重複パッケージの報告	✓	Salt
CVE監査(CVE IDが必要)	✓	Salt
SCAP監査	✓	Salt
パッケージの確認	Traditional	✗
パッケージのロック	✓	?
システムのロック	Traditional	✗
保守ウィンドウ	✓	✓
システムのスナップショット	Traditional	✗
設定ファイルの管理	✓	Salt
スナップショットとプロファイル	Traditional. Salt: プロファイルはサポートされていますが、同期はサポートされていません	Salt: プロファイルはサポートされていますが、同期はサポートされていません
電源管理	✓	Salt
監視機能	Salt	Salt
Docker buildhost	✗	✗
OSでのDockerイメージの構築	✗	✗
Kiwi buildhost	✗	✗
OSでのKiwiイメージの構築	✗	✗
繰り返しアクション	Salt	Salt
AppStream	なし	✓
Yomi	なし	なし

*仮想ゲスト管理:

この表では、仮想ゲスト管理は基本と高度に分割されています。

基本的な仮想ゲスト管理には、VMのリスト化、低速更新、VMのライフサイクルアクション(起動、停止、再開、一時停止)、およびVM vCPUとメモリの変更が含まれています。

高度な仮想ゲスト管理には、基本的な仮想ゲスト管理のすべての機能に加えて、高速更新、VMライフサイクルアクション(削除、リセット、電源オフ)、VMディスクの変更、ネットワークグラフィカル表示の変更、お

よりグラフィカル表示の設定が含まれています。

1.10. サポートされているDebianの機能

この表には、Debianクライアントのさまざまな機能の使用可否がリストされています。



クライアントで実行しているオペレーティングシステムは、オペレーティングシステムを提供している組織によってサポートされています。 DebianはDebianコミュニティでサポートされています。

この表のアイコンに意味は次のとおりです。

- ✓: 機能はSaltクライアントと従来のクライアントの両方で使用できます
- ✗: 機能は使用できません
- ?: 機能は検討中であり、後日使用できる場合と、使用できない場合があります
- Traditional: 機能は従来のクライアントでのみサポートされています
- Salt: 機能はSaltクライアントでのみサポートされています。

表 9. Debianオペレーティングシステムでサポートされている機能

機能	Debian 9	Debian 10	Debian 11
クライアント	✓	✓	✓
システムパッケージ	Debian コミュニティ	Debian コミュニティ	Debian コミュニティ
登録	Salt	Salt	Salt
パッケージのインストール	Salt	Salt	Salt
パッチの適用	?	?	?
リモートコマンド	Salt	Salt	Salt
システムパッケージの状態	Salt	Salt	Salt
システムカスタムの状態	Salt	Salt	Salt
グループカスタムの状態	Salt	Salt	Salt
組織カスタムの状態	Salt	Salt	Salt
システムセットマネージャ(SSM)	Salt	Salt	Salt
製品移行	N/A	N/A	N/A
基本的な仮想ゲスト管理*	Salt	Salt	Salt
高度な仮想ゲスト管理*	Salt	Salt	Salt

機能	Debian 9	Debian 10	Debian 11
仮想ゲストインストール(キックスタート)、ホストOSとして	✗	✗	✗
仮想ゲストインストール(イメージテンプレート)、ホストOSとして	Salt	Salt	Salt
システムの配備(PXE/キックスタート)	✗	✗	✗
システムの再配備(キックスタート)	✗	✗	✗
接続メソッド	Salt: ZeroMQ、Salt-SSH	Salt: ZeroMQ、Salt-SSH	Salt: ZeroMQ、Salt-SSH
Uyuniプロキシでの操作	Salt	Salt	Salt
アクションチェーン	Salt	Salt	Salt
ステージング(パッケージの事前ダウンロード)	Salt	Salt	Salt
重複パッケージの報告	Salt	Salt	Salt
CVE監査	?	?	?
SCAP監査	?	?	?
パッケージの確認	✗	✗	✗
パッケージのロック	✓	✓	✓
システムのロック	✗	✗	✗
保守ウィンドウ	✓	✓	✓
システムのスナップショット	✗	✗	✗
設定ファイルの管理	Salt	Salt	Salt
パッケージプロファイル	Salt: プロファイルはサポートされていますが、同期はサポートされていません	Salt: プロファイルはサポートされていますが、同期はサポートされていません	Salt: プロファイルはサポートされていますが、同期はサポートされていません
電源管理	✓	✓	✓
監視機能	Salt	Salt	Salt
Docker buildhost	?	?	?
OSでのDockerイメージの構築	Salt	Salt	Salt
Kiwi buildhost	✗	✗	✗
OSでのKiwiイメージの構築	✗	✗	✗

機能	Debian 9	Debian 10	Debian 11
繰り返しアクション	Salt	Salt	Salt
AppStreams	なし	なし	なし
Yomi	なし	なし	なし

*仮想ゲスト管理:

この表では、仮想ゲスト管理は基本と高度に分割されています。

基本的な仮想ゲスト管理には、VMのリスト化、低速更新、VMのライフサイクルアクション(起動、停止、再開、一時停止)、およびVM vCPUとメモリの変更が含まれています。

高度な仮想ゲスト管理には、基本的な仮想ゲスト管理のすべての機能に加えて、高速更新、VMライフサイクルアクション(削除、リセット、電源オフ)、VMディスクの変更、ネットワークグラフィカル表示の変更、およびグラフィカル表示の設定が含まれています。

1.11. サポートされているOracleの機能

この表には、Oracle Linuxクライアントのさまざまな機能の使用可否がリストされています。



クライアントで実行しているオペレーティングシステムは、オペレーティングシステムを提供している組織によってサポートされています。 Oracle LinuxはOracleでサポートされています。

この表のアイコンに意味は次のとおりです。

- ✓: 機能はSaltクライアントと従来のクライアントの両方で使用できます
- ✗: 機能は使用できません
- ?: 機能は検討中であり、後日使用できる場合と、使用できない場合があります
- Traditional: 機能は従来のクライアントでのみサポートされています
- Salt: 機能はSaltクライアントでのみサポートされています

表 10. Oracle Linuxオペレーティングシステムでサポートされている機能

機能	Oracle Linux 6	Oracle Linux 7	Oracle Linux 8
クライアント	✓	✓	Salt
オペレーティングシステムパッケージ	✓	✓	Salt
登録	✓	✓	Salt
パッケージのインストール	✓	✓	Salt

機能	Oracle Linux 6	Oracle Linux 7	Oracle Linux 8
パッチの適用(CVE IDが必要)	✓	✓	Salt
リモートコマンド	✓	✓	Salt
システムパッケージの状態	Salt	Salt	Salt
システムカスタムの状態	Salt	Salt	Salt
グループカスタムの状態	Salt	Salt	Salt
組織カスタムの状態	Salt	Salt	Salt
システムセットマネージャ(SSM)	✓	✓	Salt
製品移行	なし	なし	なし
基本的な仮想ゲスト管理*	Traditional	✓	Salt
高度な仮想ゲスト管理*	✗	Salt	Salt
仮想ゲストインストール(キックスタート)、ホストOSとして	Traditional	Traditional	✗
仮想ゲストインストール(イメージテンプレート)、ホストOSとして	Traditional	✓	Salt
システムの展開(PXE/キックスタート)	✓	✓	Salt
システムの再展開(キックスタート)	Traditional	✓	Salt
接続メソッド	Traditional: OSAD、RHNSD、SSH-push. Salt: ZeroMQ、Salt-SSH	Traditional: OSAD、RHNSD、SSH-push. Salt: ZeroMQ、Salt-SSH	Salt: ZeroMQ、Salt-SSH
Uyuniプロキシでの操作	✓	✓	Salt
アクションチェーン	✓	✓	Salt
ステージング(パッケージの事前ダウンロード)	✓	✓	Salt
重複パッケージの報告	✓	✓	Salt
CVE監査(CVE IDが必要)	✓	✓	Salt
SCAP監査	✓	✓	Salt
パッケージの確認	Traditional	Traditional	✗
パッケージのロック	Traditional	✓	?
システムのロック	Traditional	Traditional	✗

機能	Oracle Linux 6	Oracle Linux 7	Oracle Linux 8
保守ウィンドウ	✓	✓	✓
システムのスナップショット	Traditional	Traditional	✗
設定ファイルの管理	✓	✓	Salt
スナップショットとプロファイル	Traditional. Salt: プロファイルはサポートされていますが、同期はサポートされていません	Traditional. Salt: プロファイルはサポートされていますが、同期はサポートされていません	Salt: プロファイルはサポートされていますが、同期はサポートされていません
電源管理	✓	✓	Salt
監視機能	Salt	Salt	Salt
Docker buildhost	✗	✗	✗
OSでのDockerイメージの構築	✗	✗	✗
Kiwi buildhost	✗	✗	✗
OSでのKiwiイメージの構築	✗	✗	✗
繰り返しアクション	Salt	Salt	Salt
AppStream	なし	なし	✓
Yomi	なし	なし	なし

*仮想ゲスト管理:

この表では、仮想ゲスト管理は基本と高度に分割されています。

基本的な仮想ゲスト管理には、VMのリスト化、低速更新、VMのライフサイクルアクション(起動、停止、再開、一時停止)、およびVM vCPUとメモリの変更が含まれています。

高度な仮想ゲスト管理には、基本的な仮想ゲスト管理のすべての機能に加えて、高速更新、VMライフサイクルアクション(削除、リセット、電源オフ)、VMディスクの変更、ネットワークグラフィカル表示の変更、およびグラフィカル表示の設定が含まれています。

1.12. サポートされているRed Hat Enterprise Linuxの機能

この表には、Red Hat Enterprise Linuxクライアントのさまざまな機能の使用可否がリストされています(拡張サポートがない場合)。



クライアントで実行しているオペレーティングシステムは、オペレーティングシステムを提供している組織によってサポートされています。 Red Hat Enterprise LinuxはRed Hatでサポートされています。

この表のアイコンに意味は次のとおりです。

- ・ ✓: 機能はSaltクライアントと従来のクライアントの両方で使用できます
- ・ ✗: 機能は使用できません
- ・ ?: 機能は検討中であり、後日使用できる場合と、使用できない場合があります
- ・ Traditional: 機能は従来のクライアントでのみサポートされています
- ・ Salt: 機能はSaltクライアントでのみサポートされています。

表 11. Red Hat Enterprise Linuxオペレーティングシステムでサポートされている機能

機能	RHEL 6	RHEL 7	RHEL 8
クライアント	✓	✓	Salt
システムパッケージ	Red Hat	Red Hat	Red Hat
登録	✓	✓	Salt
パッケージのインストール	✓	✓	Salt
パッチの適用	✓	✓	Salt
リモートコマンド	✓	✓	Salt
システムパッケージの状態	Salt	Salt	Salt
システムカスタムの状態	Salt	Salt	Salt
グループカスタムの状態	Salt	Salt	Salt
組織カスタムの状態	Salt	Salt	Salt
システムセットマネージャ(SSM)	Salt	Salt	Salt
製品移行	なし	なし	なし
基本的な仮想ゲスト管理*	Traditional	✓	Salt
高度な仮想ゲスト管理*	✗	Salt	Salt
仮想ゲストインストール(キックスタート)、ホストOSとして	Traditional	Traditional	✗
仮想ゲストインストール(イメージテンプレート)、ホストOSとして	Traditional	✓	Salt
システムの展開(PXE/キックスタート)	✓	✓	Salt
システムの再展開(キックスタート)	Traditional	✓	Salt

機能	RHEL 6	RHEL 7	RHEL 8
接続メソッド	Traditional: OSAD、RHN SD、SSH-push. Salt: ZeroMQ、Salt-SSH	Traditional: OSAD、RHN SD、SSH-push. Salt: ZeroMQ、Salt-SSH	Salt: ZeroMQ、Salt-SSH
Uyuniプロキシでの操作	✓	✓	Salt
アクションチェーン	✓	✓	Salt
ステージング(パッケージの事前ダウンロード)	✓	✓	Salt
重複パッケージの報告	✓	✓	Salt
CVE監査	✓	✓	Salt
SCAP監査	✓	✓	Salt
パッケージの確認	Traditional	Traditional	✗
パッケージのロック	Traditional	✓	?
システムのロック	Traditional	Traditional	✗
保守ウィンドウ	✓	✓	✓
システムのスナップショット	Traditional	Traditional	✗
設定ファイルの管理	✓	✓	Salt
スナップショットとプロファイル	Traditional. Salt: プロファイルはサポートされていますが、同期はサポートされていません	Traditional. Salt: プロファイルはサポートされていますが、同期はサポートされていません	Salt: プロファイルはサポートされていますが、同期はサポートされていません
電源管理	✓	✓	Salt
監視機能	Salt	Salt	Salt
Docker buildhost	✗	✗	✗
OSでのDockerイメージの構築	?	?	?
Kiwi buildhost	✗	✗	✗
OSでのKiwiイメージの構築	✗	✗	✗
繰り返しアクション	Salt	Salt	Salt
AppStream	なし	なし	✓
Yomi	なし	なし	なし

*仮想ゲスト管理:

この表では、仮想ゲスト管理は基本と高度に分割されています。

基本的な仮想ゲスト管理には、VMのリスト化、低速更新、VMのライフサイクルアクション(起動、停止、再

開、一時停止)、およびVM vCPUとメモリの変更が含まれています。

高度な仮想ゲスト管理には、基本的な仮想ゲスト管理のすべての機能に加えて、高速更新、VMライフサイクルアクション(削除、リセット、電源オフ)、VMディスクの変更、ネットワークグラフィカル表示の変更、およびグラフィカル表示の設定が含まれています。

1.13. サポートされているRocky Linuxの機能

この表には、Rocky Linuxクライアントのさまざまな機能の使用可否がリストされています。



クライアントで実行しているオペレーティングシステムは、オペレーティングシステムを提供している組織によってサポートされています。Rocky LinuxはRocky Linuxコミュニティでサポートされています。

この表のアイコンに意味は次のとおりです。

- ✓: 機能はSaltクライアントと従来のクライアントの両方で使用できます
- ✗: 機能は使用できません
- ?: 機能は検討中であり、後日使用できる場合と、使用できない場合があります
- Traditional: 機能は従来のクライアントでのみサポートされています
- Salt: 機能はSaltクライアントでのみサポートされています。

表 12. Rocky Linuxオペレーティングシステムでサポートされている機能

機能	Rocky Linux 8
クライアント	Salt (plain Rocky Linux)
システムパッケージ	Rocky Linuxコミュニティ
登録	Salt
パッケージのインストール	Salt
パッチの適用	Salt
リモートコマンド	Salt
システムパッケージの状態	Salt
システムカスタムの状態	Salt
グループカスタムの状態	Salt
組織カスタムの状態	Salt
システムセットマネージャ(SSM)	Salt
製品移行	なし
基本的な仮想ゲスト管理*	Salt
高度な仮想ゲスト管理*	Salt

機能	Rocky Linux 8
仮想ゲストインストール(キックスタート)、ホストOSとして	✗
仮想ゲストインストール(イメージテンプレート)、ホストOSとして	Salt
システムの展開(PXE/キックスタート)	Salt
システムの再展開(キックスタート)	Salt
接続メソッド	Salt: ZeroMQ、Salt-SSH
Uyuniプロキシでの操作	Salt
アクションチェーン	Salt
ステージング(パッケージの事前ダウンロード)	Salt
重複パッケージの報告	Salt
CVE監査	Salt
SCAP監査	Salt
パッケージの確認	✗
パッケージのロック	?
システムのロック	✗
保守ウィンドウ	✓
システムのスナップショット	✗
設定ファイルの管理	Salt
スナップショットとプロファイル	Salt: プロファイルはサポートされていますが、同期はサポートされていません
電源管理	Salt
監視機能	Salt
Docker buildhost	✗
OSでのDockerイメージの構築	✗
Kiwi buildhost	✗
OSでのKiwiイメージの構築	✗
繰り返しアクション	Salt
AppStream	✓
Yomi	なし

*仮想ゲスト管理:

この表では、仮想ゲスト管理は基本と高度に分割されています。

基本的な仮想ゲスト管理には、VMのリスト化、低速更新、VMのライフサイクルアクション(起動、停止、再

開、一時停止)、およびVM vCPUとメモリの変更が含まれています。

高度な仮想ゲスト管理には、基本的な仮想ゲスト管理のすべての機能に加えて、高速更新、VMライフサイクルアクション(削除、リセット、電源オフ)、VMディスクの変更、ネットワークグラフィカル表示の変更、およびグラフィカル表示の設定が含まれています。

1.14. サポートされているUbuntuの機能

この表には、Ubuntuクライアントのさまざまな機能の使用可否がリストされています。



クライアントで実行しているオペレーティングシステムは、オペレーティングシステムを提供している組織によってサポートされています。 UbuntuはCanonicalでサポートされています。

この表のアイコンに意味は次のとおりです。

- ✓: 機能はSaltクライアントと従来のクライアントの両方で使用できます
- ✗: 機能は使用できません
- ?: 機能は検討中であり、後日使用できる場合と、使用できない場合があります
- Traditional: 機能は従来のクライアントでのみサポートされています
- Salt: 機能はSaltクライアントでのみサポートされています。

表 13. Ubuntuオペレーティングシステムでサポートされている機能

Feature	Ubuntu 16.04	Ubuntu 18.04	Ubuntu 20.04
Ubuntu 22.04	Client	✓	✓
✓	✓	System packages	Canonical
Canonical	Canonical	Canonical	Registration
Salt	Salt	Salt	Salt
Install packages	Salt	Salt	Salt
Salt	Apply patches	✓	✓
✓	✓	Remote commands	Salt
Salt	Salt	Salt	System package states
Salt	Salt	Salt	Salt
System custom states	Salt	Salt	Salt
Salt	Group custom states	Salt	Salt
Salt	Salt	Organization custom states	Salt
Salt	Salt	Salt	System set manager (SSM)
Salt	Salt	Salt	Salt

Feature	Ubuntu 16.04	Ubuntu 18.04	Ubuntu 20.04
Product migration	N/A	N/A	N/A
N/A	Basic Virtual Guest Management *	Salt	Salt
Salt	Salt	Advanced Virtual Guest Management *	Salt
Salt	Salt	Salt	Virtual Guest Installation (Kickstart), as Host OS
✗	✗	✗	✗
Virtual Guest Installation (image template), as Host OS	Salt	Salt	Salt
Salt	System deployment (PXE/Kickstart)	✗	✗
✗	✗	System redeployment (Kickstart)	✗
✗	✗	✗	Contact methods
Salt: ZeroMQ, Salt-SSH	Salt: ZeroMQ, Salt-SSH	Salt: ZeroMQ, Salt-SSH	Salt: ZeroMQ, Salt-SSH
Works with Uyuni Proxy	Salt	Salt	Salt
Salt	Action chains	Salt	Salt
Salt	Salt	Staging (pre-download of packages)	Salt
Salt	Salt	Salt	Duplicate package reporting
Salt	Salt	Salt	Salt
CVE auditing	?	?	?
?	SCAP auditing	?	?
?	?	Package verification	✗
✗	✗	✗	Package locking
✓	✓	✓	✓
System locking	✗	✗	✗
✗	System snapshot	✗	✗
✗	✗	Configuration file management	Salt
Salt	Salt	Salt	Package profiles
Salt: Profiles supported, Sync not supported	Salt: Profiles supported, Sync not supported	Salt: Profiles supported, Sync not supported	Salt: Profiles supported, Sync not supported
Power management	✓	✓	✓

Feature	Ubuntu 16.04	Ubuntu 18.04	Ubuntu 20.04
✓	Monitoring	✗	Salt
Salt	Salt	Docker buildhost	?
?	?	?	Build Docker image with OS
Salt	Salt	Salt	Salt
Kiwi buildhost	✗	✗	✗
✗	Build Kiwi image with OS	✗	✗

*仮想ゲスト管理:

この表では、仮想ゲスト管理は基本と高度に分割されています。

基本的な仮想ゲスト管理には、VMのリスト化、低速更新、VMのライフサイクルアクション(起動、停止、再開、一時停止)、およびVM vCPUとメモリの変更が含まれています。

高度な仮想ゲスト管理には、基本的な仮想ゲスト管理のすべての機能に加えて、高速更新、VMライフサイクルアクション(削除、リセット、電源オフ)、VMディスクの変更、ネットワークグラフィカル表示の変更、およびグラフィカル表示の設定が含まれています。

Chapter 2. 設定の基本

広範な操作を利用できるようにするためにクライアント登録のための環境を準備するには、Uyuniで複数の手順を実行する必要があります。

このセクションには、Uyuniを正しくインストールおよびセットアップした後の 環境操作をサポートするために必要な初期設定手順のまとめが記載されています。

- Uyuniのインストールの詳細については、[Installation-and-upgrade > Install-uyuni](#)を参照してください。
- Uyuniのセットアップの詳細については、[Installation-and-upgrade > Uyuni-server-setup](#)を参照してください。

2.1. ソフトウェアチャンネル

チャンネルは、ソフトウェアパッケージをグループ化する方法です。 ソフトウェアパッケージはリポジトリによって提供され、リポジトリはチャンネルに関連付けられています。 クライアントをソフトウェアチャンネルにサブスクライブすると、クライアントは、これに関連付けられたソフトウェアをインストールし、更新できます。

Uyuniでは、チャンネルはベースチャンネルと子チャンネルに分割されます。 この方法でチャンネルを編成すると、互換性のあるパッケージのみが各システムにインストールされるようになります。 クライアントは、1つのベースチャンネルのみをサブスクライブして、登録中にクライアントのオペレーティングシステムおよびアーキテクチャに基づいて割り当てる必要があります。 ベンダによって提供される有料チャンネルでは、関連づけされたサブスクリプションを持っている必要があります。

ベースチャンネルは、特定のオペレーティングシステムの種類、バージョン、およびアーキテクチャのために構築されたパッケージから構成されています。 たとえば、SUSE Linux Enterprise Server 15 x86-64のベースチャンネルには、そのオペレーティングシステムおよびアーキテクチャと互換性のあるソフトウェアのみが含まれています。

子チャンネルはベースチャンネルに関連付けられていて、ベースチャンネルと互換性のあるパッケージのみを提供します。 システムは、ベースチャンネルの複数の子チャンネルにサブスクライブできます。 システムがベースチャンネルに割り当てられている場合、そのシステムは関連する子チャンネルをインストールできます。 たとえば、システムがSUSE Linux Enterprise Server 15 **x86_64** ベースチャンネルに割り当てられている場合、互換性のあるベースチャンネルまたは関連する子チャンネルのいずれかから利用できるパッケージのみインストールまたは更新できます。

UyuniのWeb UIで、[ソフトウェア > チャンネル一覧](#)、すべてに移動して、利用できるチャンネルをブラウズできます。 [ソフトウェア > 管理](#)、チャンネルに移動して、チャンネルを変更又は新しいチャンネルを作成できます。

カスタムチャンネルなどチャンネルを使用する方法の詳細については、[Administration > Channel-management](#)を参照してください。

2.1.1. SUSE Package Hubで提供されるパッケージ

SUSE Package HubはSUSE Linux Enterprise製品の拡張機能で、openSUSEコミュニティで提供する追加オープンソースソフトウェアを提供しています。



SUSE Package Hubのパッケージは、openSUSEコミュニティによって提供されます。パッケージはSUSEではサポートされていません。

クライアントでSUSE Linux Enterpriseオペレーティングシステムを使用している場合、SUSE Package Hub拡張機能を有効にして、これらの追加パッケージにアクセスできます。アクセスすると、クライアントのサブスクリプション先にできるSUSE Package Hubチャンネルが提供されます。

SUSE Package Hubは多数のパッケージを提供しており、大量のディスク容量を使用してパッケージの同期に長時間かかる場合があります。提供するパッケージが必要でない場合、SUSE Package Hubを有効にしないでください。

サポートされていないパッケージを誤ってインストールまたは更新しないためには、最初にすべてのSUSE Package Hubパッケージを拒否するコンテンツライフサイクル管理戦略の実装をお勧めします。その後、必要なパッケージを明示的に有効にできます。
コンテンツライフサイクル管理の詳細については、Administration > Content-lifecycleを参照してください。

2.1.2. AppStreamで提供されるパッケージ

Red Hatベースのクライアントの場合、AppStreamから追加パッケージを利用できます。ほとんどの場合、AppStreamパッケージでは、必要なソフトウェアをすべて持っていることを確認する必要があります。

UyuniのWeb UIでAppStreamパッケージを管理している場合、パッケージの更新に関して相反する推奨事項が表示される場合があります。これは、Uyuniでモジュールのメタデータを正しく解釈できないことが原因です。コンテンツライフサイクル管理(CLM)のAppStreamフィルタを使用して、AppStreamリポジトリを非モジュール型リポジトリに変換して、一部の更新操作で使用できます。CLM AppStreamフィルタの詳細については、Administration > Content-lifecycle-examplesを参照してください。

2.1.3. EPELで提供されるパッケージ

Red Hatベースのクライアントの場合、EPEL(エンタープライズ版Linux用の追加パッケージ)から追加パッケージを利用できます。EPELはオプションのパッケージリポジトリで、追加ソフトウェアが提供されます。



EPELのパッケージは、Fedoraコミュニティによって提供されます。このパッケージはSUSEではサポートされていません。

クライアントでRed Hatオペレーティングシステムを使用している場合、EPEL拡張機能を有効にして、これらの追加パッケージにアクセスできます。アクセスすると、クライアントのサブスクリプション先にできるEPELチャンネルが提供されます。

EPELは多数のパッケージを提供しており、大量のディスク容量を使用してパッケージの同期に長時間かかる場合があります。提供するパッケージが必要でない場合、EPELリポジトリを有効にしないでください。

サポートされていないパッケージを誤ってインストールまたは更新しないためには、最初にすべてのEPELパッケージを拒否するコンテンツライフサイクル管理(CLM)戦略の実装をお勧めします。その後、必要なパッケージを明示的に有効にできます。コンテンツライフサイクル管理の詳細については、Administration > Content-lifecycleを参照してください。

2.1.4. Unified Installer Updates Channels on SUSE Linux Enterprise Clients

This channel is used by the Unified Installer to ensure it is up to date before it installs the operating system. All SUSE Linux Enterprise products should have access to the installer updates channel during installation.

SUSE Linux Enterprise Serverクライアントでは、更新を含む製品を追加するときにデフォルトでインストーラ更新チャンネルが同期します。また、これらの製品チャンネルで自動インストールディストリビューションを作成するときに有効になります。

SUSE Linux Enterprise for SAPなどその他すべてのSUSE Linux Enterpriseの亜種では、インストーラ更新チャンネルを手動で追加する必要があります。そのためには、適切なSUSE Linux Enterprise Serverインストーラ更新チャンネルをこれらのSUSE Linux Enterprise亜種のベースチャンネルの下に複製します。チャンネルを複製した後にこれらのSUSE Linux Enterprise亜種の自動インストールディストリビューションを作成するとき、そのインストーラ更新チャンネルが自動的に使用されます。

2.1.5. ソフトウェアリポジトリ

リポジトリはソフトウェアパッケージを収集するために使用されます。ソフトウェアリポジトリにアクセスできる場合、リポジトリが提供するソフトウェアをインストールできます。1つ以上のリポジトリをUyuniのソフトウェアチャンネルに関連付け、クライアントをそのチャンネルに割り当て、クライアントのパッケージにインストールして更新する必要があります。

Uyuniのほとんどのデフォルトチャンネルは、正しいリポジトリに関連付けられています。カスタムチャンネルを作成している場合、アクセスできるリポジトリまたは自分で作成したリポジトリを関連付ける必要があります。

カスタムリポジトリおよびチャンネルの詳細については、Administration > Custom-channelsを参照してください。

2.1.5.1. ローカルリポジトリの場所

Saltクライアントでローカルリポジトリを設定して、Uyuniチャンネルが提供しないパッケージを提供できます。



ほとんどの場合、クライアントシステムはローカルリポジトリを必要としません。ローカルリポジトリを使用すると、クライアントで使用できるパッケージがどれかという問題が発生する可能性があります。この問題が発生すると、予期しないパッケージがインストールされる場合があります。

ローカルリポジトリは、オンボード中に無効になります。クライアントがオンボードを完了した後、ローカルリポジトリを次の場所に追加できます。

表 14. ローカルリポジトリの場所

クライアントのオペレーティングシステム	ローカルリポジトリのディレクトリ
SUSE Linux Enterprise Server	/etc/zypp/repos.d
openSUSE	/etc/zypp/repos.d
SUSE Linux Enterprise Server Expanded Support	/etc/yum.repos.d/
Red Hat Enterprise Linux	/etc/yum.repos.d/
CentOS	/etc/yum.repos.d/
Ubuntu	/etc/apt/sources.list.d/
Debian	/etc/apt/sources.list.d/

Saltクライアントでは、highstateを適用しても、ローカルリポジトリは永続性の状態が維持されます。

2.1.6. ソフトウェア製品

Uyuniでは、製品でソフトウェアを使用できます。 SUSEサブスクリプションでは、さまざまな製品にアクセスできます。 製品には、UyuniのWeb UIで管理 > セットアップウィザード > 製品に移動してブラウズし、選択できます。

製品には、任意の数のソフトウェアチャンネルが含まれてい **製品チャンネルの表示** をクリックし、製品に含まれているチャンネルを表示します。 製品を追加して正常に同期すると、製品で提供しているチャンネルにアクセスできるようになります。 Uyuniサーバとクライアントで製品のパッケージを使用できます。

プロシージャ: ソフトウェアチャンネルの追加

1. UyuniのWeb UIで、管理 > セットアップウィザード > 製品に移動します。
2. 検索バーを使用してクライアントのオペレーティングシステムおよびアーキテクチャに適切な製品を探し、適切な製品にチェックを付けます。 こうすることによって、すべての必須チャンネルに自動的にチェックが付きます。 また、**include recommended** トグルがオンになっている場合、すべての推奨チャンネルにもチェックが付きます。 矢印をクリックして関連製品の一覧を表示し、必要な追加製品にチェックが付いていることを確認します。
3. **[製品の追加]** をクリックし、製品の同期が完了するまで待機します。

詳細については、Installation-and-upgrade > Setup-wizardを参照してください。

2.2. ブートストラップリポジトリ

ブートストラップリポジトリには、ブートストラップ中にSaltまたは従来のクライアントを登録するために必要なパッケージと、クライアントにSaltをインストールするためのパッケージが含まれています。 製品を同期するとき、ブートストラップリポジトリは、自動的に作成され、Uyuniサーバに再生成されます。

2.2.1. ブートストラップリポジトリの作成準備

同期する製品を選択するとき、ブートストラップリポジトリは、必須のチャンネルすべてが完全にミラーリ

ングされるとすぐに自動的に作成されます。

プロシージャ: Web UIから同期の進捗状況を確認する

1. UyuniのWeb UIで、**ソフトウェア** > **管理** > **チャンネル**に移動し、リポジトリに関連付けられているチャンネルをクリックします。
2. [リポジトリ] タブに移動し、[同期] をクリックし、[同期状態] をクリックします。

プロシージャ: コマンドプロンプトから同期の進捗状況を確認する

1. Uyuniサーバのコマンドプロンプトで、rootとして、**tail** コマンドを使用して同期ログファイルを確認します。

```
tail -f /var/log/rhn/reposync/<channel-label>.log
```

2. それぞれの子チャンネルは、同期の進捗中にそれぞれのログを生成します。同期が完了したことを確認するには、ベースチャンネルと子チャンネルのログファイルをすべて確認する必要があります。

2.2.2. 自動モードのオプション

ブートストラップリポジトリの自動作成動作を変更できます。このセクションでは、さまざまな設定を説明します。

フラッシュモード::

フラッシュモード

デフォルトでは、既存のリポジトリは、最新パッケージでのみ更新されます。代わりに、必ず空のリポジトリで始まるように設定できます。この動作を有効にするには、**/etc/rhn/rhn.conf** で次の値を追加または編集します。

```
server.susemanager.bootstrap_repo_flush = 1
```

自動モード::

自動モード

デフォルトでは、ブートストラップリポジトリの自動再生成は有効になっています。無効にするには、**/etc/rhn/rhn.conf** で次の値を追加または編集します。

```
server.susemanager.auto_generate_bootstrap_repo = 0
```

2.2.2.1. ブートストラップデータファイルの設定

このツールは、各ディストリビューションに必要なパッケージに関する情報を含むデータファイルを使用します。データファイルは**/usr/share/susemanager/mgr_bootstrap_data.py** に保存されています。SUSEはこのファイルを定期的に更新します。このファイルを変更する場合、直接編集しないでください。代わりに、同じディレクトリにコピーを作成し、コピーを編集します。

```
cd /usr/share/susemanager/
cp mgr_bootstrap_data.py my_data.py
```

変更したら、Uyuniを設定して新しいファイルを使用します。 [/etc/rhn/rhn.conf](#) でこの値を追加または編集します。

```
server.susemanager.bootstrap_repo_datamodule = my_data
```



次の更新時、SUSEの新しいデータによって、新しいデータファイルではなく元のデータファイルが上書きされます。 SUSEによって行われた変更を使用して新しいファイルを最新に保つ必要があります。

2.2.3. ブートストラップリポジトリの手動生成

デフォルトでは、ブートストラップリポジトリは毎日再生成されます。 コマンドプロンプトからブートストラップリポジトリを手動で作成できます。

プロシージャ: SUSE Linux Enterpriseのブートストラップリポジトリの生成

1. Uyuniサーバのコマンドプロンプトで、rootとして、次のコマンド用のブートストラップリポジトリを作成するために使用できるディストリビューションをリストします。

```
mgr-create-bootstrap-repo -l
```

2. 製品ラベルとして適切なリポジトリ名を使用して、ブートストラップリポジトリを作成します。

```
mgr-create-bootstrap-repo -c SLE-version-x86_64
```

3. または、利用可能なディストリビューション一覧のディストリビューション名の横に表示されている番号を使用します。

クライアントリポジトリは [/srv/www/htdocs/pub/repositories/](#) にあります。

複数の製品(SLESとSLES for SAPなど)をミラーリング済みの場合、またはカスタムチャンネルを使用している場合、ブートストラップリポジトリを作成するときに使用する親チャンネルを指定する必要が生じる場合があります。 これは、あらゆる状況で必須ではありません。 たとえば、SLES 15の一部のバージョンには共通のコードベースがあるため、親チャンネルを指定する必要はありません。 このプロシージャは、ご使用の環境で必要な場合のみ使用します。

オプションのプロシージャ: ブートストラップリポジトリの親チャンネルの指定

1. 利用できる親チャンネルを確認します。

```
mgr-create-bootstrap-repo -c SLE-15-x86_64
Multiple options for parent channel
found. (親チャンネルの複数にオプションが表示されます。) Please use option
--with-parent-channel <label> and choose one of: (オプション --with-
-parent-channel <label>を使用し、次のいずれかを選択してください。)
- sle-product-sles15-pool-x86_64
- sle-product-sles_sap15-pool-x86_64
- sle-product-sled15-pool-x86_64
```

- 適切な親チャンネルを指定します。

```
mgr-create-bootstrap-repo -c SLE-15-x86_64 --with-parent-channel sle-
product-sled15-pool-x86_64
```

2.2.3.1. 複数アーキテクチャを含むリポジトリ

複数の異なるアーキテクチャを含むブートストラップリポジトリを作成している場合、すべてのアーキテクチャが正しく更新されることに注意を払う必要があります。たとえば、SLEのx86-64アーキテクチャおよびIBM Zアーキテクチャは、同じブートストラップリポジトリURL(</srv/www/htdocs/pub/repositories/sle/15/2/bootstrap/>)を使用します。

フ ラ ッ シ ュ オプションを有効にすると、複数のアーキテクチャのブートストラップリポジトリを生成しようとしても、生成されるアーキテクチャは1つのみです。この動作を回避するには、追加のアーキテクチャを作成するとき、コマンドプロンプトで **--no-flush** オプションを使用します。次に例を示します。

```
mgr-create-bootstrap-repo -c SLE-15-SP2-x86_64
mgr-create-bootstrap-repo --no-flush -c SLE-15-SP2-s390x
```

2.2.4. ブートストラップとカスタムチャンネル

カスタムチャンネルを使用している場合、**mgr-create-bootstrap-repo** コマンドを使用して **--with-custom-channels** オプションを使用できます。この場合、使用する親チャンネルも指定する必要があります。

カスタムチャンネルを使用すると、ブートストラップリポジトリの自動作成が失敗する場合があります。この場合、リポジトリを手動で作成する必要があります。

カスタムチャンネルの詳細については、Administration > Custom-channelsを参照してください。

2.3. アクティベーションキー

アクティベーションキーは従来のクライアントとSaltクライアントで使用し、クライアントが正しいソフトウェアのエンタイトルメントを持ち、適切なチャンネルに接続して関連グループに加入するようします。そ

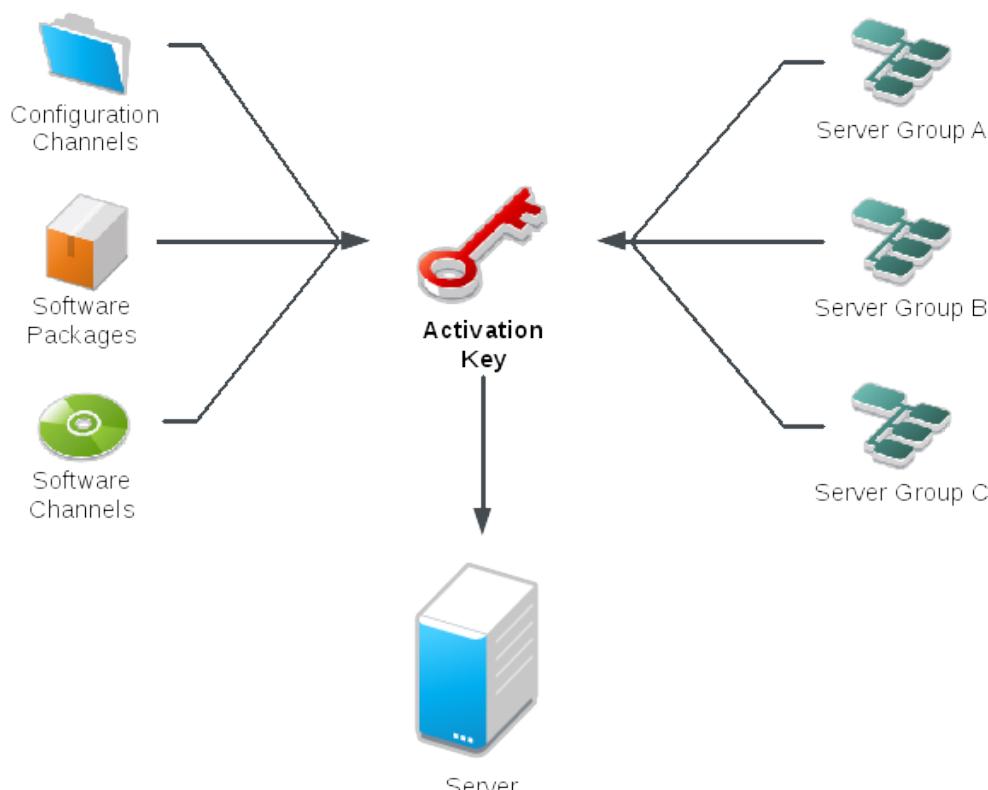
それぞれのアクティベーションキーは、キーを作成するときに設定できる組織にひもづけされます。

Uyuniでは、アクティベーションキーは、ラベルの付いた一連の設定です。 アクティベーションキーに関連付けられている設定は、すべて適用できます。そのためには、キーのラベルをパラメータとしてブートストラップスクリプトに追加します。 アクティベーションキーラベルをブートストラップスクリプトと組み合わせて使用することをお勧めします。 ブートストラップスクリプトが実行されると、そのラベルに関連付けられているすべての設定が、スクリプトを実行しているシステムに適用されます。

アクティベーションキーは以下を指定できます。

- ・ チャンネルの割り当て
- ・ システムの種類またはアドオンのエンタイトルメント
- ・ 接続メソッド
- ・ 設定ファイル
- ・ インストールするパッケージ
- ・ システムグループの割り当て

アクティベーションキーは、クライアント登録時に使用され、再使用されることはありません。 アクティベーションキーで指定する内容に関係なく、クライアントは登録後、任意の方法で変更できます。 アクティベーションキーとクライアントの関連付けは、履歴を残すためだけに記録されます。



プロセッジヤ: アクティベーションキーの作成

1. UyuniのWeb UIで、管理者としてシステム > アクティベーションキーに移動します。

2. [キ]ーの[作成]ボタンをクリックします。
3. [アクティベーションキーの詳細]ページの[説明]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。たとえば、SUSE Linux Enterprise Server 15 SP4では **SLES15-SP4** と入力します。



SUSE製品の[キー]フィールドではカンマを使用しないでください。ただし、Red Hat製品ではカンマを使用する必要があります。 詳細については、[Reference > Systems](#)を参照してください。

5. [ベースチャンネル] ドロップダウンボックスで、適切なベースソフトウェアチャンネルを選択し、関連する子チャンネルへのデータ入力を許可します。 詳細については、[reference:admin/setup-wizard.pdf](#)と[Administration > Custom-channels](#)を参照してください。
6. 必要な子チャンネルを選択します(必須のSUSE Managerツールや更新チャンネルなど)。
7. いずれかのオプションを有効にする必要がある場合は、[付属エンタイトルメント] チェックボックスにチェックを付けます。
8. [接続メソッド] は [デフォルト] のままにすることをお勧めします。
9. [汎用デフォルト] 設定には、チェックを入れないようにすることをお勧めします。
10. [アクティベーションキーの作成]をクリックしてアクティベーションキーを作成します。
11. [設定ファイルの展開] チェックボックスをチェックし、このキーの設定管理を有効にし、[アクションキーの更新]をクリックしてこの変更を保存します。



[設定ファイルの展開] チェックボックスは、アクティベーションキーを作成するまで表示されません。 設定管理を有効にする必要がある場合、前に戻つてボックスにチェックを付けます。

2.3.1. 複数のアクティベーションキーの結合

従来のクライアントでブートストラップスクリプトを実行するとき、アクティベーションキーを結合できます。 キーを結合すると、システムにインストールされる機能をより詳細に制御でき、大規模な環境や複雑な環境でのキーの重複が軽減されます。



アクティベーションキーの結合は、従来のクライアント上でのみ動作します。 Salt クライアントはアクティベーションキーの結合をサポートしていません。 Saltクライアントで結合キーを使用する場合、最初のキーのみ使用されます。

コマンドプロンプトまたはシングル自動インストールプロファイルで複数のアクティベーションキーを指定できます。

Uyuniサーバのコマンドプロンプトで、**rhnreg_ks** コマンドを使用し、カンマでキーの名前を区切れます。 Kickstartプロファイルで複数のキーを指定するには、[システム > 自動インストール](#)に移動し、使用するプロファイルを編集します。

値が競合するとクライアントの登録に失敗するため、アクティベーションキーを結合するときには注意してください。結合する前に次の値で情報の競合がないことを確認してください。

- ・ ソフトウェアパッケージ
- ・ ソフトウェアの子チャンネル
- ・ 設定チャンネル。

競合は、検出されると次のように処理されます。

- ・ ベースソフトウェアチャンネルの競合: 登録は失敗します。
- ・ システムの種類の競合: 登録は失敗します。
- ・ **enable configuration** フラグの競合: 設定管理が有効になります。
- ・ 一方のキーがシステム固有のキーである場合: 登録は失敗します。

2.3.2. 再アクティベーションキー

クライアントを再登録してすべてのUyuni設定を再取得するために、再アクティベーションキーを1回だけ使用できます。 再アクティベーションキーはクライアント固有で、システムID、履歴、グループ、およびチャンネルが含まれています。

再アクティベーションキーを作成するには、[システム]に移動し、再アクティベーションキーを作成するクライアントをクリックし、再アクティベーションタブに移動します。 [新規]をクリックして再アクティベーションキーを作成します。 後で使用できるようにキーの詳細を書き留めます。 特定のシステムIDに関連付けられていない通常のアクティベーションキーと異なり、ここで作成されるキーは、システム > アクティベーションキーページに表示されません。

Saltクライアントの場合、再アクティベーションキーを作成した後、`/etc/salt/minion.d/susemanager.conf` の **management_key** grainとして使用できます。 次に例を示します。

```
grains:
  susemanager:
    management_key: "re-1-daf44db90c0853edbb5db03f2b37986e"
```

salt-minion プロセスを再起動して再アクティベーションキーを適用します。

ブートストラップスクリプトで再アクティベーションキーを使用できます。 ブートストラップスクリプトの詳細については、Client-configuration > Registration-bootstrapを参照してください。

従来のクライアントの場合、再アクティベーションキーを作成した後、**rhnreg_ks** コマンドラインユーティリティでこのキーを使用できます。 このコマンドを実行すると、クライアントが再登録され、そのUyuni設定が復元されます。 従来のクライアントでは、再アクティベーションキーをアクティベーションキーと結合して、単一システムプロファイルで複数のキーの設定を集約できます。 次に例を示します。

```
rhnreg_ks --server=<server-url>/XMLRPC \
--activationkey=<reactivation-key>, <activationkey> \
--force
```



既存のUyuniプロファイルでクライアントを自動インストールすると、そのプロファイルは、再アクティベーションキーを使用して、システムを再登録し、その設定を復元します。プロファイルベースの自動インストール実行中は、このキーを再生成、削除、または使用しないでください。このような操作を実行すると、自動インストールは失敗します。

2.3.3. アクティベーションキーのベストプラクティス

デフォルトの親チャンネル

SUSE マネージャのデフォルトの親チャンネルを使用する場合、Uyuniは、インストールされるオペレーティングシステムに最適な親チャンネルを強制的に選択します。その場合、予期しない動作が発生する可能性があります。代わりに、それぞれのディストリビューションおよびアーキテクチャに固有のアクティベーションキーを作成することをお勧めします。

アクティベーションキーによるブートストラップ

ブートストラップスクリプトを使用している場合、各スクリプトにアクティベーションキーを作成することを検討してください。作成によって、チャンネルの割り当て、パッケージのインストール、システムグループメンバーシップ、および設定チャンネルの割り当ての整合性を取ることができます。登録後にシステムで手動操作する必要も減ります。

帯域幅の要件

アクティベーションキーを使用すると、登録時にソフトウェアが自動ダウンロードされることがあります。この動作は、帯域幅に制約がある環境では望ましくない場合があります。

次のオプションによって帯域幅使用条件が作成されます。

- ・ SUSE Product Poolチャンネルを割り当てると、対応する製品ディスクリプタパッケージが自動インストールされます。
- ・ [パッケージ] セクションのパッケージがインストールされます。
- ・ [設定] セクションのSaltの状態によっては、その内容に応じてダウンロードがトリガされる場合があります。

キーラベルの命名

読んで理解しやすい名前をアクティベーションキーに入力しないと、システムが数値の文字列を自動生成するため、キーの管理が困難になる場合があります。

キーを追跡できるようにアクティベーションキーの命名規則を検討してください。組織のインフラストラクチャに関係がある名前を付けておくと、複雑な操作の実行も簡単になります。

キーラベルを作成する場合、次のヒントを考慮してください。

- ・ OSの名前(必須): キーには、設定を指定するOS名を必ず含める必要があります。
- ・ アーキテクチャ名(推奨): 会社で稼働しているアーキテクチャ(たとえば、x86_64)が複数ある場合、アーキテクチャの種類をラベルに含めることをお勧めします。
- ・ サーバの種類の名前: このサーバの使用目的
- ・ 場所名: サーバの配置場所(部屋、ビル、部署)
- ・ 日付: 保守期間(四半期など)。
- ・ カスタム名: 組織のニーズに合う命名規則

アクティベーションキーラベルの名前の例:

```
sles15-sp4-web_server-room_129-x86_64
```

```
sles15-sp4-test_packages-blg_502-room_21-ppc64le
```



SUSE製品の [キー] フィールドではカンマを使用しないでください。ただし、Red Hat製品ではカンマを使用する必要があります。 詳細については、[Reference > Systems](#)を参照してください。

含めるチャンネル

アクティベーションキーを作成するときは、このキーに関連付けられているソフトウェアチャンネルも考慮する必要があります。キーには、特定のベースチャンネルを割り当てる必要があります。デフォルトのベースチャンネルの使用はお勧めしません。 詳細については、[Client-configuration > Registration-overview](#)でインストールしているクライアントオペレーティングシステムを参照してください。

2.4. GPGキー

クライアントではGPGキーを使用して、ソフトウェアパッケージをインストールする前にパッケージ認証の確認が行われます。信頼されているソフトウェアのみクライアントにインストールできます。

ほとんどの場合、クライアントにソフトウェアをインストールできるようにGPG設定を調整する必要はありません。

RPM packages can be signed directly, while Debian based systems sign only the metadata and use a chain of checksums to secure the packages. Most RPM based systems use signed metadata in addition to signed packages.

2.4.1. クライアントでGPGキーを信頼する

Operating systems either trust their own GPG keys directly or at least ship them installed with the minimal system. But third party packages signed by a different GPG key need manual handling. The clients can be successfully bootstrapped without the GPG key being trusted. However, you cannot install new client tool packages or update them until the keys are trusted.

Salt clients use now GPG key information entered for a software channel to manage the trusted keys. When a software channel with GPG key information is assigned to a client, the key gets trusted as soon as the channel is refreshed or the first package gets installed from this channel.

The GPG key URL which is set of a software channel must exist. In case it is a file URL, the GPG key file must be deployed on the client before the software channel is used.

The GPG keys for the Client Tools Channels of Red Hat based clients are deployed on the client into `/etc/pki/rpm-gpg/` and can be referenced with file URLs. Same is the case with the GPG keys of Expanded Support clients. Only in case a software channel is assigned to the client they will be imported and trusted by the system.



Because Debian based systems sign only metadata, there is typically no need to specify extra keys for single channels. If a user configures an own GPG key to sign the metadata as described in "Use Your Own GPG Key" in **Administration > Repo-metadata** the deployment and trust of that key is executed automatically.

2.4.1.1. User defined GPG keys

Users can define their own GPG keys to be deployed to the client.

By providing some pillar data and providing the GPG key files in the Salt filesystem, they are automatically deployed to the client.

These keys are deployed into `/etc/pki/rpm-gpg/` on RPM based operating systems and to `/usr/share/keyrings/` on Debian systems:

Define the pillar key [literal `custom_gpgkeys`] for the client you want to deploy the key to and list the names of the key file.

```
cat /etc/pillar/mypillar.sls
custom_gpgkeys:
  - my_first_gpg.key
  - my_second_gpgkey.gpg
```

Additionally in the Salt filesystem create a directory named `gpg` and store there the GPG key files with the name specified in the `custom_gpgkeys` pillar data.

```
ls -la /etc/salt/gpg/
/etc/salt/gpg/my_first_gpg.key
/etc/salt/gpg/my_second_gpgkey.gpg
```

The keys are now deployed to the client at `/etc/pki/rpm-gpg/my_first_gpg.key` and `/etc/pki/rpm-gpg/my_second_gpgkey.gpg`.

The last step is to add the URL to the GPG key URL field of the software channel. Navigate to **Software > Manage > Channels** and select the channel you want to modify. Add to **GPG key URL** the value `file:///etc/pki/rpm-gpg/my_first_gpg.key`.

2.4.1.2. GPG Keys in Bootstrap Scripts

プロシージャ: ブートストラップスクリプトを使用してクライアントでGPGキーを信頼する

1. Uyuniサーバのコマンドプロンプトで、`/srv/www/htdocs/pub/`ディレクトリの内容を確認します。このディレクトリには、使用できるすべての公開鍵が含まれています。登録クライアントに割り当てるチャンネルに適用するキーをメモします。
2. 関連するブートストラップスクリプトを開き、`ORG_GPG_KEY=`パラメータを見つけて、必要なキーを追加します。次に例を示します。

```
uyuni-gpg-pubkey-0d20833e.key
```

以前保存したキーを削除する必要はありません。



Trusting a GPG key is important for security on clients. It is the task of the admin to decide which keys are needed and can be trusted. Because a software channel cannot be used when the GPG key is not trusted, the decision of assigning a channel to a client depends on the decision of trusting the key.

Chapter 3. クライアントの管理方法

Uyuniサーバがクライアントと通信する方法は多数あります。 使用方法は、クライアントのタイプおよびネットワークアーキテクチャによって決まります。

Uyuniデーモン(**rhnscd**)は従来のクライアントシステムで動作し、Uyuniと定期的に接続し、新しい更新および通知を確認します。 Saltクライアントには適用されません。

SSHでのプッシュおよびSalt SSHでのプッシュメソッドは、クライアントがUyuniサーバに直接アクセスできない環境で使用されます。 この環境では、DMZと呼ばれるファイアウォール保護ゾーンにクライアントがあります。 内部ネットワークとの接続を開くことを認可されているシステム(Uyuniサーバなど)はDMZ内にはありません。

OSADは、Uyuniと従来のクライアントの間の代替の接続メソッドです。 OSADでは、スケジュールされているアクションを従来のクライアントがすぐに実行できます。 Saltクライアントには適用されません。

3.1. Saltクライアントの接続メソッド

ほとんどの場合、Saltクライアントは、デフォルトのブートストラップメソッドで正確に登録されます。

非接続設定でSaltクライアントを使用する必要がある場合、Salt SSHでのプッシュを設定できます。 この環境では、DMZと呼ばれるファイアウォール保護ゾーンにクライアントがあります。 Salt SSH接続メソッドの詳細については、[Client-configuration > Contact-methods-saltssh](#)を参照してください。

Saltクライアントを手動で設定してUyuniサーバに接続する必要がある場合は、正しいネットワーク詳細を使用してSaltクライアントの設定ファイルを編集します。 Salt minion設定ファイルの接続メソッドの詳細については、[Client-configuration > Registration-cli](#)を参照してください。

3.1.1. オンボードの詳細

Saltには、minionのキーを保持するための独自のデータベースがあります。これは、Uyuniデータベースと同期を保つ必要があります。 Saltでキーが受け入れられると、Uyuniでオンボードプロセスがただちに開始されます。オンボードプロセスは、**minion_id** と **machine-id** を検索して、Uyuniデータベース内で既存のシステムを探します。 何も見つからない場合は、新しいシステムが作成されます。**minion_id** または **machine-id** のエントリが見つかった場合、新しいシステムに合わせてシステムが移行されます。両方のエントリとの一致が見つかり、一致したのが同一のシステムではない場合、オンボードはエラーで中断されます。 この場合、管理者は少なくとも1つのシステムを削除して競合を解決する必要があります。

3.1.2. Salt SSHでのプッシュ

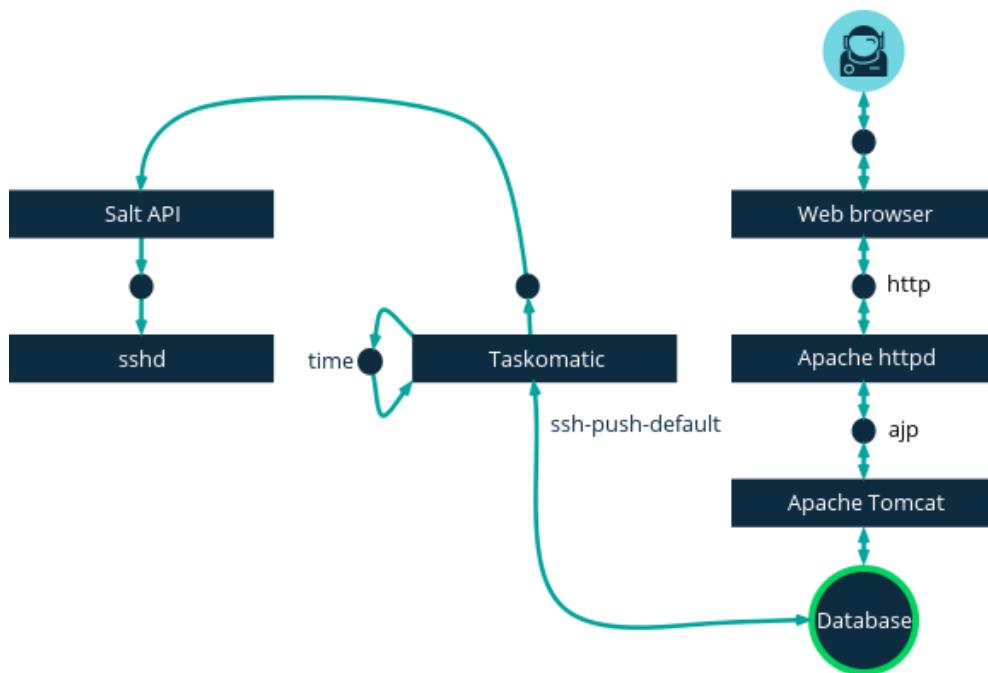
Salt SSHでのプッシュは、SaltクライアントがUyuniサーバに直接アクセスできない環境で使用されます。 この環境では、DMZと呼ばれるファイアウォール保護ゾーンにクライアントがあります。 内部ネットワークとの接続を開くことを認可されているシステム(Uyuniサーバなど)はDMZ内にはありません。

Salt SSHでのプッシュメソッドは、DMZにあるクライアントに内部ネットワークのUyuniサーバから暗号化トンネルを作成します。 すべてのアクションおよびイベントが実行された後、トンネルはクローズします。

サーバは、Salt SSHを使用して、定期的にクライアントに接続し、チェックインし、スケジュールされたアクションおよびイベントを実行します。Salt SSHの詳細については、[Salt Specialized-guides](#)を参照してください。

この接続方法は、Saltクライアントでのみ動作します。従来のクライアントでは、SSHでのプッシュを使用します。

次のイメージは、Salt SSHでのプッシュプロセスのパスを示しています。[Taskomatic](#) ブロックの左側のアイテムはすべて、Uyuniクライアントで実行されるプロセスを表します。



Salt SSHでのプッシュを使用するには、クライアントで動作しているSSHデーモンが必要で、Uyuniサーバで動作している `salt-api` デーモンによって接続できます。また、Pythonは、リモートシステムで使用できてSaltでサポートされているバージョンである必要があります。



Red Hat Enterprise Linux 5、CentOS 5、およびこれら以前のバージョンは、サポートされていないバージョンのPythonを使用しているため、サポートされません。

プロシージャ: Salt SSHでのプッシュを使用したクライアントの登録

1. UyuniのWeb UIで、[システム > ブートストラップ](#)に移動し、該当するフィールドに入力します。
2. SSHでのプッシュ接続メソッドが設定されたアクティベーションキーを選択します。アクティベーションキーの詳細については、[Client-configuration > Activation-keys](#)を参照してください。
3. [Manage System Completely via SSH] (SSHでシステムを完全に管理する) チェックボックスにチェックを付けます。
4. [ログイン] をクリックして、登録を開始します。
5. [システム > 概要](#)に移動して、システムが正しく登録されたことを確認します。

3.1.2.1. 使用可能なパラメータ

Salt SSHでのプッシュを設定している場合、ホスト、アクティベーションキー、パスワードなど、システムを登録するときに使用するパラメータを変更できます。このパスワードはブートストラップでのみ使用し、どこにも保存されません。今後のSSHセッションではすべて、キー/証明書のペアで認可されます。これらのパラメータはシステム › ブートストラップで設定されます。

sudoユーザなどシステム全体で使用される永続パラメータも設定できます。sudoユーザの設定について詳しくは、[Client-configuration > Contact-methods-pushssh](#)を参照してください。

3.1.2.2. アクションの実行

Salt SSHでのプッシュ機能は、taskomaticを使用し、`salt-ssh`を使用してスケジュール済みのアクションを実行します。taskomaticジョブは、スケジュールされたアクションを定期的に確認して実行します。従来のクライアントにおけるSSHでのプッシュと異なり、Salt SSHでのプッシュ機能は、スケジュールされたアクションに基づいて、完全な`salt-ssh`コールを実行します。

デフォルトでは、20個のSalt SSHアクションを同時に実行できます。同時実行できるアクションの個数を増やすことができます。そのためには、次の行を設定ファイルに追加し、`parallel_threads`の値を調整します。問題の発生を回避するために、同時実行アクション数を低い値に保つことをお勧めします。

```
taskomatic.com.redhat.rhn.taskomatic.task.SSHMinionActionExecutor.parallel_threads = <number>
org.quartz.threadPool.threadCount = <value of parallel_threads + 20>
```

1つのクライアントで同時実行できるアクションの個数とtaskomaticで使用されるワーカスレッドの合計数が調整されます。複数のクライアントでアクションを実行する必要がある場合、アクションは常に各クライアントで順次実行されます。

クライアントがプロキシ経由で接続されている場合、プロキシの`MaxSessions`設定を調整する必要があります。この場合、平行接続数を総クライアント数の3倍に設定します。

3.1.2.3. 今後の機能

Salt SSHでのプッシュでサポートされていない機能があります。これらの機能はSalt SSHクライアント上で動作しません。

- ・ OpenSCAPの監査
- ・ ビーコン。次の結果になります。
 - `zypper`を使用してシステムのパッケージをインストールしても、パッケージ更新が呼び出されません。
 - 仮想ホストシステムがSalt SSHベースの場合、仮想ホスト関数(たとえば、ゲストへのホスト)が動作しません。

Salt SSHの詳細については、<https://docs.saltstack.com/en/latest/topics/ssh/>を参照してください。

3.1.3. Salt Bundle

3.1.3.1. Salt Bundleの概要

Salt Bundleは、Salt Minion、Python 3、必須のPythonモジュール、およびライブラリが含まれている1つのバイナリパッケージです。

Salt BundleはPython 3に付属していて、Saltを実行するためのすべての要件です。したがって、Salt Bundleは、システムソフトウェアとしてクライアントにインストールされているPythonバージョンを使用しません。Salt Bundleは、指定のSaltバージョンの要件を満たさないクライアントにインストールできます。

Uyuni Salt Master以外のSalt Masterに接続されているSalt Minionを実行するシステムでSalt Bundleを使用することもできます。

3.1.3.2. Salt Bundleを使用してクライアントをMinionとして登録する

Salt Bundleを使用した登録方法は推奨の登録方法です。このセクションでは、現在の実装の利点と制約について説明します。Salt Bundleは、Salt、Python 3、およびSaltが依存しているPythonモジュールで構成されている **venv-salt-minion** として提供されます。Web UIを使用したブートストラップもSalt Bundleを使用しているため、Web UIを使用したブートストラップはPythonに依存しません。Salt Bundleを使用すると、クライアントがPythonインタープリターまたはモジュールを提供する必要がなくなります。

新しいクライアントをブートストラップする場合、Salt Bundleを使用した登録がデフォルトの方法です。既存のクライアントをSalt Bundleの方法に切り替えることができます。切り替える場合、**salt-minion** パッケージおよびその依存関係はインストールされたままになります。

3.1.3.2.1. Salt Minionを使用したSalt Bundleの使用

Salt Bundleは、Uyuniサーバ以外のSalt Masterによって管理されているSalt Minionと同時に使用できます。Salt Bundleが、UyuniサーバがSalt Bundleの設定ファイルを管理するクライアントにインストールされる場合、**salt-minion** の設定ファイルは管理されません。詳細については、[Salt Bundle configuration](#)を参照してください。



Uyuniサーバ以外のSalt Masterによって管理されているSalt Minionを使用してクライアントをブートストラップするには、ブートストラップスクリプトを生成するときに **mgr-bootstrap --force-bundle** を使用するか、またはブートストラップスクリプトで **FORCE_VENV_SALT_MINION** を **1** に設定することをお勧めします。Web UI **mgr_force_venv_salt_minion** を **true** に設定してブートストラップする場合、pillarをグローバルに指定できます。 詳細については、[Specialized-guides > Salt](#)を参照してください。

3.1.3.2.2. Salt MinionからSalt Bundleへの切り替え

salt-minion から **venv-salt-minion** に切り替えるためにSalt状態 **util.mgr_switch_to_venv_minion** を使用できます。移行プロセスのトラブルを回避するために、**venv-salt-minion** への切り替えは2ステップで実行することをお勧めします。

プロシージャ: `util.mgr_switch_to_venv_minion` を使用して状態を `venv-salt-minion` に切り替える

- まず、pillarを指定せずに `util.mgr_switch_to_venv_minion` を適用します。`venv-salt-minion` に切り替わり、設定ファイルなどがコピーされます。元の `salt-minion` の設定およびそのパッケージはクリーンアップされません。

```
salt <minion_id> state.apply util.mgr_switch_to_venv_minion
```

`util.mgr_switch_to_venv_minion` を適用し、`mgr_purge_non_venv_salt` を `True` に設定して `salt-minion` を削除し、`mgr_purge_non_venv_salt_files` を `True` に設定して `salt-minion` に関するすべてのファイルを削除します。この2番目の手順によって、最初の手順が処理されたことが保証され、古い設定ファイルおよび古くなった `salt-minion` パッケージが削除されます。

```
salt <minion_id> state.apply util.mgr_switch_to_venv_minion
pillar='{"mgr_purge_non_venv_salt_files": True,
"mgr_purge_non_venv_salt": True}'
```



切り替えの最初の手順をスキップして2番目の手順を実行すると、クライアント側でコマンドを実行するために使用される `salt-minion` を停止する必要があるため、状態適用プロセスは失敗する可能性があります。

他方、Salt Bundleのインストールを回避して代わりに `salt-minion` の使用を続けることも可能です。この場合、次のいずれかのオプションを指定します。

- `--no-bundle` オプションを指定して `mgr-bootstrap` を実行します。
- 生成されたブートストラップスクリプトで `AVOID_VENV_SALT_MINION` を `1` に設定します。
- ブートストラップ状態の場合、`mgr_avoid_venv_salt_minion` pillarを `True` に設定します。

3.1.3.3. Salt BundleによるSalt SSH

Salt Bundleは、クライアントに対してSalt SSHアクションを実行するときにも使用されます。

シェルスクリプトは、Saltコマンドを実行する前に `venv-salt-minion` をインストールせずにSalt Bundleをターゲットシステムに配備します。Salt BundleにはSaltのコードベース全体が含まれているため、`salt-thin` は配備されません。Salt SSH (Web UIを使用するブートストラップを含む)は、バンドル内でPython 3インターフィリターを使用します。ターゲットシステムには他のPythonインターフィリターがインストールされている必要はありません。

Bundleを使用して配備されたPython 3は、クライアントでSalt SSHセッションを処理するために使用されるため、Salt SSH (Web UIを使用したブートストラップを含む)は、システムにインストールされているPythonに依存しません。



ブートストラップリポジトリは、Web UIを使用してクライアントをブートストラップする前に作成済みである必要があります。Salt SSHは、検出したターゲットオペレーティングシステムに基づいてブートストラップリポジトリーから取得されたSalt Bundleを使用しています。 詳細については、[ブートストラップリポジトリー作成の準備](#)を参照してください



salt-thin の使用はフォールバック方法として有効にできますが、クライアントにPython 3をインストールする必要があります。 この方法は、開発目的でのみ存在しており、お勧めもサポートもしません。 `/etc/rhn/rhn.conf` 設定ファイルで `web.ssh_use_salt_thin` を `true` に設定します。

3.2. 従来のクライアントの接続メソッド

従来のクライアントは、さまざまなメソッドでUyuniサーバと通信できます。

Uyuniデーモン(`rhnsd`)は従来のクライアントシステムで動作し、Uyuniと定期的に接続し、新しい更新および通知を確認します。

SSHでのプッシュは、クライアントでUyuniサーバに直接接続できない環境で使用されます。 この環境では、DMZと呼ばれるファイアウォール保護ゾーンにクライアントはあります。 内部ネットワークとの接続を開くことを認可されているシステム(Uyuniサーバなど)はDMZ内にはありません。

OSADは、Uyuniと従来のクライアントの間の代替の接続メソッドです。 OSADでは、スケジュールされているアクションを従来のクライアントがすぐに実行できます。



With SUSE Manager 4.3 release, traditional clients have been deprecated. The release following SUSE Manager 4.3 will not support traditional clients and traditional proxies, and it is planned for the year 2023. We encourage all new deployments to use Salt clients and Salt proxies exclusively, and to migrate existing traditional clients and proxies to Salt.

+ Be aware that when migrating from traditional clients to Salt minions you do not have to delete the registered clients before. You can just register them as Salt minions and Salt will do the necessary steps with the traditional client. If you already deleted the traditional client and the registration as Salt minion is not possible anymore, see [Installation-and-upgrade > Troubleshooting](#).

3.2.1. SUSE Managerデーモン(`rhnsd`)

Uyuniデーモン(`rhnsd`)は従来のクライアントシステムで動作し、Uyuniと定期的に接続し、新しい更新および通知を確認します。 Saltクライアントには適用されません。

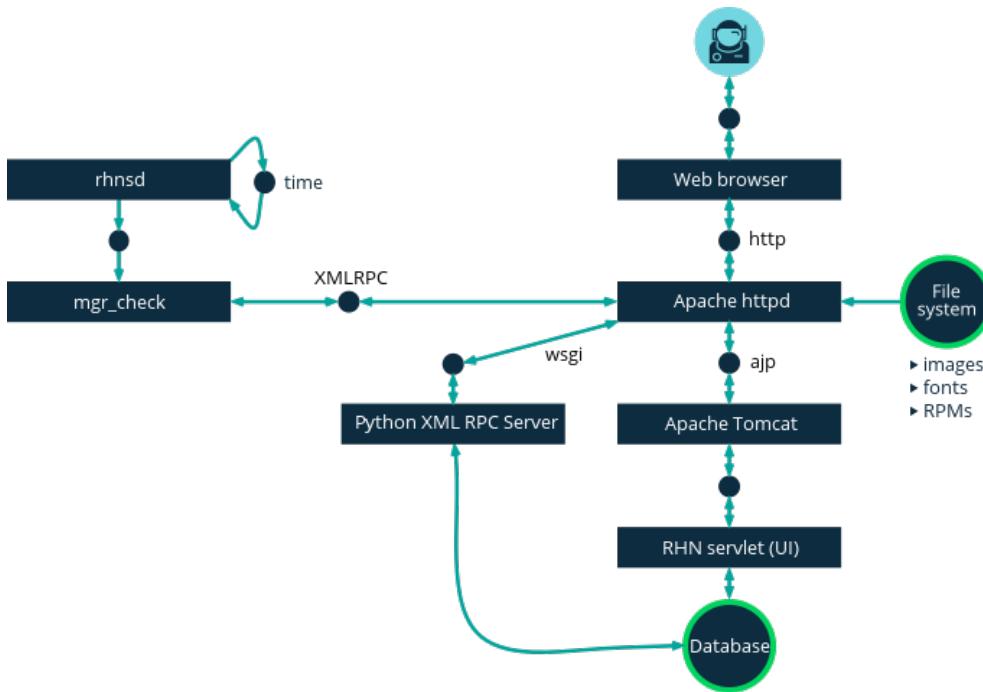
これは、SUSE Linux Enterprise 11およびRed Hat Enterprise Linux Server 6でのみ使用されます。その理由はこれらのシステムはsystemdを使用しないためです。 後者のオペレーティングシステムでは、systemdタイマー(`rhnsd.timer`)が使用され、`rhnsd.service` によって制御されます。

`/etc/init.d/rhnsd` でデーモンを起動します。

デフォルトでは4時間ごとに新しいアクションの確認が行われます。つまり、スケジュールされたアクションをクライアントで実行するには時間がかかることがあります。

更新を確認するために、**rhnsd** は、`/usr/sbin/` にある外部の **mgr_check** プログラムを実行します。このプログラムは、Uyuniへのネットワーク接続を確立する小さいアプリケーションです。Uyuniデーモンは、ネットワークポートをリッスンせず、ネットワークに直接問い合わせしません。すべてのネットワークアクティビティは、**mgr_check** ユーティリティによって実行されます。

次の図は、デフォルトの **rhnsd** プロセスのパスの概要を示しています。**Python XMLRPC server** ブロックの左側のアイテムはすべて、Uyuniクライアントで実行されるプロセスを表します。



3.2.1.1. rhnsdの設定

rhnsd 初期化スクリプトには、クライアントシステムの設定ファイルが `/etc/sysconfig/rhn/rhnsd` にあります。

デーモンの重要なパラメータはそのチェックイン頻度です。デフォルトの間隔は4時間(240分)です。設定可能な最短間隔は1時間(60分)です。間隔を1時間未満に設定すると、デフォルトの4時間(240分)に戻ります。

rhnsd 設定ファイルを変更する場合、次のコマンドをrootとしてデーモンを再起動し、変更を取得します。

```
/etc/init.d/rhnsd restart
```

rhnsd のステータスを表示するには、次のコマンドをrootとして実行します。

```
/etc/init.d/rhnsd status
```

SUSE Linux Enterprise 12以降では、デフォルトの間隔は `/etc/systemd/system/timers.target.wants/rhnsd.timer` で設定されます。このセクションでは次のようにになります。

```
[Timer]
OnCalendar=00/4:00
RandomizedDelaySec=30min
```

`systemctl` を使用して `rhnsd.timer` の上書きドロップインファイルを作成できます。

```
systemctl edit rhnsd.timer
```

たとえば、2時間の間隔を設定する場合、次のようにになります。

```
[Timer]
OnCalendar=00/2:00
```

ファイルは `/etc/systemd/system/rhnsd.timer.d/override.conf` として保存されます。

`systemd` タイマーの詳細については、`systemd.timer` および `systemctl` のマニュアルを参照してください。

3.2.1.2. OSAD

OSADは、Uyuniと従来のクライアントの間の代替の接続メソッドです。デフォルトでは、Uyuniは `rhnsd` を使用します。これは、4時間ごとにサーバに接続し、スケジュールされたアクションを実行します。OSADでは、スケジュールされているアクションを従来のクライアントがすぐに実行できます。



`rhnsd` に加えて、OSADを使用します。`rhnsd` を無効にすると、クライアントは24時間後に未確認と表示されます。

OSADには異なるコンポーネントがいくつかあります。

- ・ `osa-dispatcher` サービスは、サーバで動作し、データベースチェックを使用して、クライアントをpingする必要があるかどうか、およびアクションを実行する必要があるかどうかを判定します。
- ・ `osad` サービスは、クライアントで動作します。このサービスは、`osa-dispatcher` からpingに応答し、`mgr_check` を実行して指示された場合にはアクションを実行します。
- ・ `jabberd` サービスは、クライアントとサーバの間の通信に XMPP プロトコルを使用するデーモンです。`jabberd` サービスは認証も処理します。
- ・ `mgr_check` ツールは、クライアントで動作し、アクションを実行します。このツールは、`osa-dispatcher` サービスから通信によってトリガれます。

`osa-dispatcher` は、クエリを定期的に実行し、クライアントがネットワークアクティビティを最後に示した

タイミングを確認します。 最近アクティビティを示していないクライアントを見つけた場合、**jabberd** を使用して、Uyuniサーバで登録されているクライアントのすべてで実行している **osad** インスタンスをすべてpingします。**osad** インスタンスは、**jabberd** を使用してpingに応答します。これは、サーバ上のバックグラウンドで実行されています。**osa-dispatcher** が応答を受信すると、クライアントをオンラインとしてマークします。**osa-dispatcher** が一定の時間内に応答を受信できない場合、クライアントをオフラインとしてマークします。

OSAD対応のシステムでアクションをスケジュールすると、タスクはすぐに実行されます。**osa-dispatcher** は、実行する必要があるアクションのクライアントを定期的に確認します。 実行保留中アクションが見つかった場合、**jabberd** を使用してクライアントで **mgr_check** を実行し、次にアクションを実行します。

OSADクライアントは、サーバの完全修飾ドメイン名(FQDN)を使用して **osa-dispatcher** サービスと通信します。

osad 通信にはSSLが必要です。 SSL証明書が利用できない場合、クライアントシステムのデーモンは接続できません。 ファイアウォールルールで必要なポートの許可が設定されていることを確認します。 詳細については、[Installation-and-upgrade > Ports](#)を参照してください。

プロシージャ: OSADの有効化

1. Uyuniサーバのコマンドプロンプトで、rootとして、**osa-dispatcher** サービスを起動します。

```
systemctl start osa-dispatcher
```

2. それぞれのクライアントで、**mgr-osad** パッケージを [ツール] 子チャンネルからインストールします。
mgr-osad パッケージはクライアントにのみインストールする必要があります。**mgr-osad** パッケージをUyuniサーバにインストールする場合、**osa-dispatcher** パッケージと競合します。

3. それぞれのクライアントでrootとして **osad** サービスを起動します。

```
systemctl start osad
```

osad および **osa-dispatcher** はサービスとして実行されるため、**stop**、**restart**、**status**などの標準コマンドを使用して、管理できます。

それぞれのOSADコンポーネントは、ローカル設定ファイルを使用して設定されます。 すべてのOSADコンポーネントのデフォルトの設定パラメータを保存することをお勧めします。

Component	場所	設定ファイルへのパス
osa-dispatcher	サーバ	/etc/rhn/rhn.conf セクション: OSA の 設 定
osad	クライアント	/etc/sysconfig/rhn/osad.conf
osad ログファイル	クライアント	/var/log/osad
jabberd ログファイル	両方	/var/log/messages

OSADのトラブルシューティング

OSADクライアントをサーバに接続できない場合、または **jabberd** サービスでポート5552への応答に時間がかかる場合、開いているファイルの数が超過していることが原因である可能性があります。

それぞれのクライアントは、常に開いているサーバTCP接続が1つ必要で、1つのファイルハンドラを使用します。現在開いているファイルハンドラの数が **jabberd** での使用が許可されている最大ファイル数を超えると、**jabberd** はリクエストをキューに入れ、接続を拒否します。

この問題を解決するには、**jabberd** のファイル制限数を増やします。そのために **/etc/security/limits.conf** 設定ファイルを編集して次の行を追加します。

```
jabber soft nofile 5100
jabber hard nofile 6000
```

使用している環境で必要な制限数を計算するには、ソフト制限はクライアント数に100を加算し、ハード制限は現在のクライアント数に1000を加算します。

上記の例では、現在のクライアント数を5000と想定しているため、ソフト制限は5100、ハード制限は6000です。

/etc/jabberd/c2s.xml ファイルの **max_fds** パラメータを選択ハード制限で更新する必要があります。

```
<max_fds>6000</max_fds>
```

3.2.2. SSHでのプッシュ

SSHでのプッシュは、従来のクライアントでUyuniサーバに直接接続できない環境で使用されます。この環境では、DMZと呼ばれるファイアウォール保護ゾーンにクライアントがあります。内部ネットワークとの接続を開くことを認可されているシステム(Uyuniサーバなど)はDMZ内にはありません。

SSHでのプッシュメソッドは、DMZにあるクライアントに内部ネットワークのUyuniサーバから暗号化トンネルを作成します。すべてのアクションおよびイベントが実行された後、トンネルはクローズします。

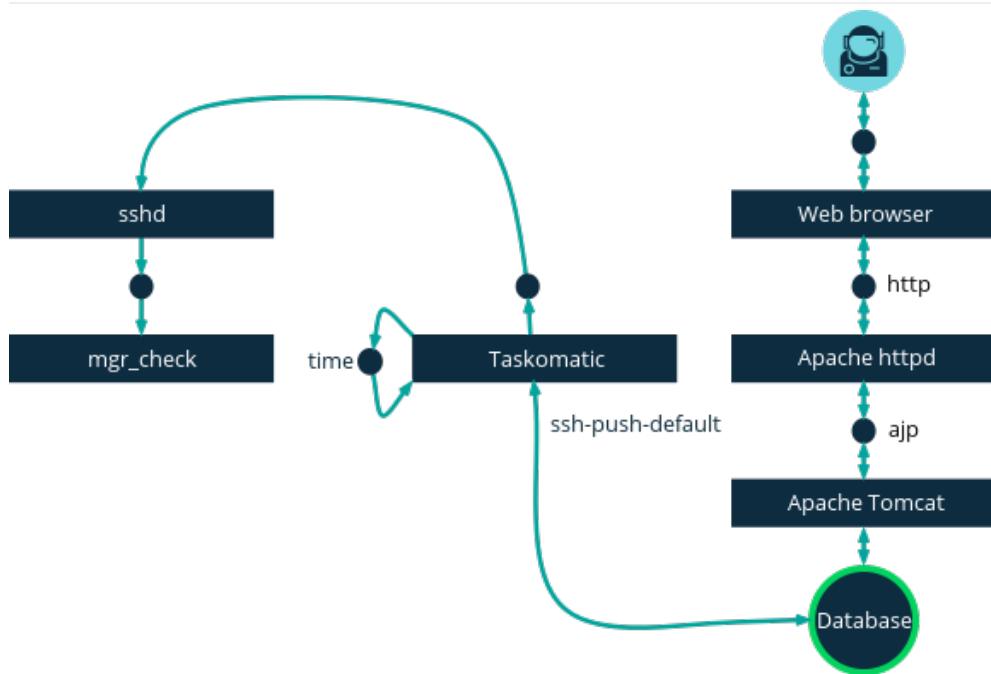
サーバは、SSHを使用して、定期的にクライアントに接続し、チェックインし、スケジュールされたアクションおよびイベントを実行します。

この接続方法は、従来のクライアントでのみ動作します。Saltクライアントでは、Salt SSHでのプッシュを使用します。



プロビジョニングモデルを使用したシステムの再インストールは、SSHでのプッシュで管理されているクライアントでは現在サポートされていません。

次のイメージは、SSHでのプッシュプロセスのパスを示しています。Taskomatic ブロックの左側のアイテムはすべて、Uyuniクライアントで実行されるプロセスを表します。



SSHでのトンネル接続では、使用できる2つのポート番号が必要です。1つ目はHTTPのトンネル用、2つ目はHTTPSでのトンネル用です(HTTPは登録プロセス中のみ必要です)。デフォルトで使用されるポート番号は **1232** と **1233** です。これらの番号を上書きするには、1024よりも大きい2つのカスタムポート番号を **/etc/rhn/rhn.conf** に追加します。

```
ssh_push_port_http = high_port_1
ssh_push_port_https = high_port_2
```

IPアドレスではなくホスト名を使用してクライアントに接続する場合、次のオプションを設定します。

```
ssh_push_use_hostname = true
```

クライアント接続を同時平行して開くために使用するスレッドの数も調整できます。デフォルトでは、2つの同時平行スレッドが使用されます。 **/etc/rhn/rhn.conf** で **taskomatic.ssh_push_workers** を設定します。

```
taskomatic.ssh_push_workers = number
```

セキュリティ上の理由から、SSHでsudoを使用して、rootとしてではなく非特権ユーザとしてシステムにアクセスする必要があります。

プロシージャ: 非特権SSHアクセスの設定

- 最新の **spacewalk-taskomatic** パッケージおよび **spacewalk-certs-tools** パッケージがUyuniサーバにインストールされていることを確認してください。
- それぞれのクライアントシステムで、適切な非特権ユーザを作成します。

3. それぞれのクライアントシステムで、`/etc/sudoers` ファイルを開き、次の行をコメントアウトします。

```
#Defaults targetpw  # ask for the password of the target user i.e.
root
#ALL    ALL=(ALL) ALL  # WARNING! Only use this together with
'Defaults  targetpw'!
```

4. それぞれのクライアントシステムの **User privilege specification** セクションで、次の行を追加します。

```
<user> ALL=(ALL) NOPASSWD:/usr/sbin/mgr_check
<user> ALL=(ALL) NOPASSWD:/home/<user>/enable.sh
<user> ALL=(ALL) NOPASSWD:/home/<user>/bootstrap.sh
```

5. それぞれのクライアントシステムの `/home/<user>/.bashrc` ファイルで、次の行を追加します。

```
PATH=$PATH:/usr/sbin
export PATH
```

6. Uyuniサーバの `/etc/rhn/rhn.conf` 設定ファイルで、次の行を追加または修正して、非特権ユーザ名を含めます。

```
ssh_push_sudo_user = <user>
```

クライアントがDMZにあり、サーバに接続できないため、`mgr-ssh-push-init` ツールを使用してクライアントをUyuniサーバに登録する必要があります。

ツールを使用するには、クライアントのホスト名またはIPアドレス、およびUyuniサーバの有効なブートストラップスクリプトへのパスが必要です。 ブートストラップの詳細については、[Client-configuration > Registration-bootstrap](#)を参照してください。

ブートストラップスクリプトは、SSHでのプッシュで設定されたスクリプトとアクティベーションキーを関連付ける必要があります。 アクティベーションキーの詳細については、[Client-configuration > Activation-keys](#)を参照してください。

始める前に、SSHトンネルに使用するポートを指定済みであることを確認する必要があります。 ポート番号を変更する前にクライアントを登録した場合、再登録する必要があります。



SSHでのプッシュによって管理されているクライアントはサーバに直接接続できません。`mgr-ssh-push-init` ツールを使用している場合、`rhnscd` デーモンは無効になっています。

プロシージャ: SSHでのプッシュを使用したクライアントの登録

1. Uyuniサーバのコマンドプロンプトで、rootとして、次のコマンドを実行します。

```
# mgr-ssh-push-init --client <client> --register \
/srv/www/htdocs/pub/bootstrap/bootstrap_script --tunnel
```

オプション: トンネルを使用しない場合、**--tunnel** オプションを削除できます。

2. オプション: **ssh_push_sudo_user** を定義済みの場合、**--notty** オプションを追加してrootのパスワードを使用できます。
3. SSH接続がアクティブであることを確認します。

```
# ssh -i /root/.ssh/id_susemanager -R <high_port>:<susemanager>:443 \
<client> zypper ref
```

例: SSHでのプッシュへのAPIアクセス

APIを使用して、使用する接続メソッドを管理できます。このPythonコードの例では、接続メソッドが**ssh-push**に設定されます。

有効な値は次のとおりです。

- **default** (pull)
- **ssh-push**
- **ssh-push-tunnel**

```
client = xmlrpclib.Server(SUMA_HOST + "/rpc/api", verbose=0)
key = client.auth.login(SUMA_LOGIN, SUMA_PASSWORD)
client.system.setDetails(key, 1000012345, {'contact_method' : 'ssh-push'})
```

クライアントを登録済みで、SSHでのプッシュを使用するようにする場合、追加手順が必要です。**mgr-ssh-push-init** ツールを使用してクライアントを設定できます。

プロシージャ: 登録済みシステムをSSHでのプッシュに移行する

1. Uyuniサーバのコマンドプロンプトで、rootとして、クライアントを設定します。

```
# mgr-ssh-push-init --client <client> \
/srv/www/htdocs/pub/bootstrap/bootstrap_script --tunnel
```

2. UyuniのWeb UIを使用して、クライアントの接続メソッドを `ssh-push` または `ssh-push-tunnel` に変更します。
3. オプション: 既存のアクティベーションキーを編集する必要がある場合、次のコマンドでできます。

```
client.activationkey.setDetails(key, '1-mykey', {'contact_method' :  
'ssh-push'})
```

プロキシを使用して接続するクライアントにもSSHでのプッシュを使用できます。 始める前にプロキシが更新されていることを確認してください。

プロシージャ: SSHでのプッシュを使用したクライアントをプロキシに登録する

1. Uyuniサーバのコマンドプロンプトで、rootとして、クライアントを設定します。

```
# mgr-ssh-push-init --client <client> \  
/srv/www/htdocs/pub/bootstrap/bootstrap_script --tunnel
```

2. Uyuniサーバのコマンドプロンプトで、SSHキーをプロキシにコピーします。

```
mgr-ssh-push-init --client <proxy>
```

Chapter 4. クライアントの登録

クライアントをUyuniサーバに登録する方法は複数あります。このセクションでは、使用できるさまざまなメソッドについて説明します。クライアントで実行するオペレーティングシステム固有の情報も含まれています。

始める前に次の項目を確認してください。

- クライアントで登録前にUyuniサーバと日時が正しく同期されている。
- アクティベーションキーを作成済みである。アクティベーションキーの作成の詳細については、[Client-configuration > Activation-keys](#)を参照してください。



Uyuniサーバをこのサーバ自体に登録しないでください。Uyuniサーバは個別に管理するか、別のUyuniサーバを使用して管理する必要があります。複数のサーバを使用する方法の詳細については、[Specialized-guides > Large-deployments](#)を参照してください。

4.1. クライアント登録メソッド

クライアントをUyuniサーバに登録する方法は複数あります。

- Saltクライアントの場合、UyuniのWeb UIを使用してクライアントを登録することをお勧めします。 詳細については、[Client-configuration > Registration-webui](#)を参照してください。
- プロセスをより詳細に制御したい場合、多数のクライアントを登録する必要がある場合、または従来のクライアントを登録している場合、ブートストラップスクリプトの作成をお勧めします。 詳細については、[Client-configuration > Registration-bootstrap](#)を参照してください。
- Saltクライアントで、さらに詳細にプロセスを制御するには、コマンド行でsingleコマンドを実行すると便利です。 詳細については、[Client-configuration > Registration-cli](#)を参照してください。

クライアントは、登録する前にUyuniサーバと日時が正しく同期されている必要があります。

まず、アクティベーションキーを作成してから、ブートストラップスクリプトまたはコマンドラインメソッドを使用する必要があります。アクティベーションキーの作成の詳細については、[Client-configuration > Activation-keys](#)を参照してください。



Uyuniサーバをこのサーバ自体に登録しないでください。Uyuniサーバは個別に管理するか、別のUyuniサーバを使用して管理する必要があります。複数のサーバを使用する方法の詳細については、[Specialized-guides > Large-deployments](#)を参照してください。

4.1.1. Web UIでクライアントを登録する

UyuniのWeb UIでクライアントを登録する方法はSaltクライアントでのみ動作します。

Web UIを使用してSaltクライアントをブートストラップしている場合、[Specialized-guides > Salt](#)を使用してクライアントでブートストラッププロセスを実行しています。Salt SSHは、Salt Bundleおよびそれに含まれているPythonインターパリターを使用します。したがって、その他のPythonインターパリターをクライアントにインストールする必要はありません。



Salt Bundleはブートストラップリポジトリに付属しているため、クライアントでブートストラッププロセスを開始する前にリポジトリを作成する必要があります。シェルスクリプトは、クライアントのオペレーティングシステムを検出し、適切なブートストラップリポジトリからSalt Bundleを配備し、ブートストラップスクリプトと同じロジックを使用します。詳細については、[ブートストラップリポジトリの作成準備](#)を参照してください。



Uyuniサーバをこのサーバ自体に登録しないでください。Uyuniサーバは個別に管理するか、別のUyuniサーバを使用して管理する必要があります。複数のサーバを使用する方法の詳細については、[Specialized-guides > Large-deployments](#)を参照してください。

プロシージャ: Web UIでクライアントを登録する

1. UyuniのWeb UIで、[システム > ブートストラップ](#)に移動します。
2. [ホスト] フィールドに、ブートストラップするクライアントの完全修飾ドメイン名(FQDN)を入力します。
3. [SSH ポート] フィールドに、クライアントを接続してブートストラップするために使用するSSHポート番号を入力します。デフォルトでは、SSHポートは **22** です。
4. [ユーザ] フィールドに、クライアントにログインするユーザ名を入力します。デフォルトでは、ユーザ名は **root** です。
5. SSHでクライアントをブートストラップするには、[認証] フィールドで、[SSH 機密鍵] にチェックを付け、クライアントへのログインに使用するSSH秘密鍵をアップロードします。SSH秘密鍵でパスフレーズが必要な場合、[SSH 機密鍵のパスフレーズ] フィールドに入力します。パスフレーズがない場合には空白のままにします。
6. パスワードでクライアントをブートストラップするには、[認証] フィールドで、[パスワード] にチェックを付け、クライアントへのログインに使用するパスワードを入力します。
7. [アクティベーションキー] フィールドで、クライアントのブートストラップに使用するソフトウェアチャンネルに関連付けられているアクティベーションキーを選択します。詳細については、[Client-configuration > Activation-keys](#)を参照してください。
8. オプション: [プロキシ] フィールドで、クライアントの登録先にするプロキシを選択します。
9. デフォルトでは、[Disable SSH Strict Key Host Checking] (SSH厳格キーhosztの確認を無効にする) チェックボックスにチェックが付いています。このチェックボックスにチェックが付いていると、ブートストラッププロセスは、手動認証なしでSSHホストキーを自動的に受け入れます。
10. オプション: [Manage System Completely via SSH] (SSHでシステムを完全に管理する) チェックボックスにチェックを付けます。このオプションにチェックを付けると、サーバへの接続にSSHを使用するようにクライアントは設定され、その他の接続方法は設定されません。
11. [ブートストラップ] をクリックして、登録を開始します。

ブートストラッププロセスが完了したら、クライアントは [システム > システム一覧] にリストされます。



SSH秘密鍵は、ブートストラッププロセス中のみ保存されます。 秘密鍵は、ブートストラップが完了するとすぐにUyuniサーバから削除されます。



Uyuniを使用してクライアントに新しいパッケージまたは更新がインストールされると、エンドユーザライセンスアグリーメント(EULA)が自動的に受け入れられます。 パッケージのEULAを確認するには、Web UIでパッケージ詳細ページを開きます。



CentOS 6、Oracle Linux 6、Red Hat Enterprise Linux 6、またはSUSE Linux Enterprise Server with Expanded Support 6の各クライアントを登録して使用するには、Uyuniサーバを設定して旧式のSSL暗号化をサポートする必要があります。 詳細については、[古いクライアントの登録をClient-configuration clients](#)で参照してください。

4.1.1.1. Handling of Locally assigned Repositories

Having repositories assigned directly to clients not served by Uyuni is not a common use case. It can cause trouble. Therfore bootstrapping via Salt disables all local repositories at the beginning of the bootstrap process.

Later, during every use of the channel state, for example when executing a Highstate or a package installation, all locally assigned repositories are disabled again.

All software packages which are used on the clients should come from channels served by Uyuni. For more information about creating a custom channel, see [Custom Channels](#) at [Administration > Custom-channels](#).

4.1.2. ブートストラップスクリプトを使用してクライアントを登録する

ブートストラップスクリプトでクライアントを登録すると、パラメータを制御できるようになり、同時に多数のクライアントを登録する必要がある場合に役立ちます。 このメソッドは、Saltクライアントと従来のクライアントの両方で動作します。

ブートストラップスクリプトを使用してクライアントを登録するには、まずブートストラップスクリプトのテンプレートを作成することをお勧めします。 このテンプレートはその後コピーして変更できます。 作成したブートストラップスクリプトは、登録時にクライアントで実行され、必要なパッケージがすべてクライアントに展開されていることを確認します。 ブートストラップスクリプトに含まれているパラメータがあります。 このパラメータによって、クライアントシステムを、アクティベーションキーやGPGキーなどそのベースチャンネルに割り当てるることができます。

リポジトリ情報を注意深く確認して、ベースチャンネルリポジトリと一致していることを確認することが重要です。 リポジトリ情報が正確に一致しないと、ブートストラップスクリプトは正しいパッケージをダウンロードできません。



すべてのクライアントにブートストラップリポジトリが必要です。製品が同期されると、SUSE Managerサーバ上で自動的に作成および再生成されます。ブートストラップリポジトリには、クライアントにSaltをインストールするためのパッケージ、Saltまたは従来のクライアントを登録するためのパッケージが用意されています。ブートストラップリポジトリの作成については、[Client-configuration > Bootstrap-repository](#)を参照してください。



GPGキーおよびUyuniクライアントツール

Uyuniクライアントツールで使用されるGPGキーは、デフォルトでは信頼されていません。ブートストラップスクリプトを作成するとき、**ORG_GPG_KEY** パラメータを使用して公開鍵の指紋を含むファイルへのパスを追加します。



openSUSE Leap 15およびSLES 15およびPython 3

デフォルトではopenSUSE Leap 15およびSLE 15はPython 3を使用します。Python 2に基づくブートストラップスクリプトは、openSUSE Leap 15システムおよびSLE 15システム用に再作成する必要があります。Python 2を使用してopenSUSE Leap 15システムまたはSLE 15システムを登録する場合、ブートストラップスクリプトは失敗します。

4.1.2.1. Web UIでのブートストラップスクリプトの作成

UyuniのWeb UIを使用して、編集できるブートストラップスクリプトを作成できます。

プロシージャ: ブートストラップスクリプトの作成

1. UyuniのWeb UIで、**管理 > マネージャ設定 > ブートストラップスクリプト**に移動します。
2. **[SUSE Manager Configuration - Bootstrap]** (SUSEマネージャ設定 - ブートストラップ) ダイアログで、従来のクライアントをインストールしている場合、**[Salt を 使用 す る ブ ー ト ス ト ラ っ プ]** チェックのチェックを外します。Saltクライアントでは、チェックを付けたままにします。
3. 必須フィールドには、前のインストール手順から取り出した値が事前に入力されています。各設定の詳細については、**Reference > Admin**を参照してください。
4. **[新]**をクリックしてスクリプトを作成します。
5. ブートストラップスクリプトが生成され、サーバの **/srv/www/htdocs/pub/bootstrap** ディレクトリに保存されます。または、HTTPS経由でブートストラップスクリプトにアクセスできます。**example.com** をUyuniサーバのホスト名に置き換えます。

```
https://<example.com>/pub/bootstrap/bootstrap.sh
```



ブートストラップスクリプトのSSLを無効にしないでください。 Web UIで [**Enable SSL**(SSLの有効化)] がチェックされていること、または設定 **USING_SSL=1** がブートストラップスクリプトに存在していることを確認してください。 SSLを無効にすると、登録プロセスでカスタムSSL証明書が必要です。 カスタム証明書の詳細については、[Administration > Ssl-certs](#)を参照してください。



CentOS 6、Oracle Linux 6、Red Hat Enterprise Linux 6、またはSUSE Linux Enterprise Server with Expanded Support 6の各クライアントを登録して使用するには、Uyuniサーバを設定して旧式のSSL暗号化をサポートする必要があります。 このエラーを解決する方法については、[古いクライアントの登録](#)を[Client-configuration > Tshoot-clients](#)で参照してください。

4.1.2.2. ブートストラップスクリプトの編集

作成したブートストラップスクリプトのテンプレートをコピーして変更し、カスタマイズできます。 Uyuniで使用するためにブートストラップスクリプトを編集するときの最小要件は、アクティベーションキーを含めることです。 ほとんどのパッケージはGPGで署名されているため、信頼できるGPGキーをシステムで用意してインストールすることも必要です。

このプロシージャでは、アクティベーションキーの正確な名前を知っている必要があります。 [ホーム > 概要](#)に移動し、「タスク」ボックスで、「[アクティベーションキーのチップ](#)」を作成します。 成したすべてのキーがこのページに一覧表示されます。 ブートストラップスクリプトで使用するキーのフルネームを、キーフィールドに表示されているように正確に入力する必要があります。 アクティベーションキーの詳細については、[Client-configuration > Activation-keys](#)を参照してください。

プロシージャ: ブートストラップスクリプトの変更

1. Uyuniサーバのコマンドラインでrootとして、ブートストラップディレクトリを次のように変更します。

```
cd /srv/www/htdocs/pub/bootstrap/
```

2. 各クライアントで使用するブートストラップスクリプトのテンプレートのコピーを2つ作成し、名前を変更します。

```
cp bootstrap.sh bootstrap-sles12.sh
cp bootstrap.sh bootstrap-sles15.sh
```

3. 変更するために **bootstrap-sles15.sh** を開きます。 次のテキストが表示されるまで下方にスクロールします。 ファイルに「**exit 1**」がある場合、その行の先頭にハッシュまたはポンド記号(#)を入力してコメントアウトします。 これによって、スクリプトがアクティブになります。 **ACTIVATION_KEYS=** フィールドにこのスクリプトのキーの名前を入力します。

```

echo "Enable this script: comment (with #'s) this block (or, at least
just"
echo "the exit below)"
echo
#exit 1

# can be edited, but probably correct (unless created during initial
install):
# NOTE: ACTIVATION_KEYS *must* be used to bootstrap a client machine.
ACTIVATION_KEYS=1-sles15
ORG_GPG_KEY=

```

4. 完了したら、ファイルを保存し、2つ目のブートストラップスクリプトでこの手順を繰り返します。



デフォルトでは、ブートストラップスクリプトは、Saltクライアントの **venv-salt-minion** がブートストラップリポジトリにある場合にはこれをインストールしようとし、ブートストラップリポジトリにSalt Bundleがない場合には **salt-minion** をインストールしようとします。何らかの理由で **salt-minion** が必要な場合、Salt Bundleをインストールせずに **salt-minion** の使用を続けることができます。詳細については、[Client-configuration > Contact-methods-saltbundle](#)を参照してください。

4.1.2.3. クライアントの接続

スクリプトの作成を完了したら、このスクリプトを使用してクライアントを登録できます。

プロシージャ: ブートストラップスクリプトの実行

1. Uyuniでrootとしてログインします。コマンドプロンプトでブートストラップディレクトリに変更します。

```
cd /srv/www/htdocs/pub/bootstrap/
```

2. 次のコマンドを実行して、クライアントでブートストラップスクリプトを実行します。EXAMPLE.COMをクライアントのホスト名に置き換えます。

```
cat bootstrap-sles15.sh | ssh root@EXAMPLE.COM /bin/bash
```

3. または、クライアントで次のコマンドを実行します。

```
curl -Sks https://server_hostname/pub/bootstrap/bootstrap-sles15.sh |
/bin/bash
```



問題を回避するには、ブートストラップスクリプトが **bash** を使用して実行されていることを確認してください。

このスクリプトは、前に作成したリポジトリディレクトリにある必要な依存関係をダウンロードします。

4. スクリプトの実行が完了すると、クライアントが正しく登録されたかどうかを確認できます。そのためには、UyuniのWeb UIを開き、**システム** > **概要**に移動して、新しいクライアントがリストされていることを確認します。
5. スクリプトを使用してSaltクライアントを登録する場合は、UyuniのWeb UIを開いて、**Salt** > **Keys**に移動し、クライアントキーを受け入れます。



Uyuniを使用してクライアントに新しいパッケージまたは更新がインストールされると、エンドユーザライセンスアグリーメント(EULA)が自動的に受け入れられます。パッケージのEULAを確認するには、Web UIでパッケージ詳細ページを開きます。

4.1.3. コマンドラインで登録する(Salt)

4.1.3.1. Saltクライアントの手動登録

ほとんどの場合、Saltクライアントは、デフォルトのブートストラップメソッドで正確に登録されます。ただし、Saltを使用してクライアントをUyuniサーバに手動で登録できます。そのためには、クライアントでSalt Minionファイルを編集し、サーバの完全修飾ドメイン名(FQDN)を指定します。このメソッドは、サーバで受信するポート4505および4506を使用します。このメソッドではUyuniサーバの設定は不要です。ただし、上記のポートを開いている必要があります。



コマンドラインで従来のクライアントを登録することができますが、その手順は長くなります。この手順についてはここでは説明しません。ブートストラップスクリプトプロシージャを使用して従来のクライアントを登録します。 詳細については、[registration-bootstrap.pdf](#)を参照してください。

このプロシージャでは、登録する前に **venv-salt-minion**(Saltbundle)または **salt-minion** パッケージをSaltクライアントにインストール済みである必要があります。両方ともさまざまな場所で設定ファイルを使用しますが、ファイル名は同じままです。systemdサービスファイル名は異なります。



この方法でブートストラップを実行できるのは、クライアントツールチャンネルまたは公式のSUSEディストリビューションの一部として **salt-minion** を使用する場合のみです。

4.1.3.2. Salt Bundleの設定

Salt Bundle (**venv-salt-minion**)

- `/etc/venv-salt-minion/`

- `/etc/venv-salt-minion/minion`
- `/etc/venv-salt-minion/minion.d/NAME.conf`
- systemdサービスファイル: `venv-salt-minion.service`

Salt bundleの詳細については、[Client-configuration > Contact-methods-saltbundle](#)を参照してください。

プロシージャ: Salt Bundle設定ファイルでクライアントを登録する

1. Saltクライアントで `minion` 設定ファイルを開きます。 設定ファイルは次の場所にあります。

```
/etc/venv-salt-minion/minion
```

または

```
/etc/venv-salt-minion/minion.d/NAME.conf
```

2. ファイルで、UyuniサーバまたはプロキシのFQDNと、アクティベーションキー(存在する場合)を追加または編集します。以下にリストされている他の設定パラメータも追加します。

```
マスタ: SERVER.EXAMPLE.COM

grains:
  susemanager:
    activation_key: "<Activation_Key_Name>"

  server_id_use_crc: adler32
  enable_legacy_startup_events: False
  enable_fqdns_grains: False
```

3. `venv-salt-minion` サービスを再起動します。

```
systemctl restart venv-salt-minion
```

4. Uyuniサーバで、新しいクライアントキーを受け入れます。 `<client>` をクライアントの名前に置き換えます。

```
salt-key -a '<client>'
```

4.1.3.3. Salt Minionの設定

Salt Minion (salt-minion)

- /etc/salt/
- /etc/salt/minion
- /etc/salt/minion.d/NAME.conf
- systemdサービスファイル: salt-minion.service

プロシージャ: Salt Minion設定ファイルでクライアントを登録する

1. Saltクライアントで **minion** 設定ファイルを開きます。 設定ファイルは次の場所にあります。

```
/etc/salt/minion
```

または

```
/etc/salt/minion.d/NAME.conf
```

2. ファイルで、UyuniサーバまたはプロキシのFQDNと、アクティベーションキー(存在する場合)を追加または編集します。以下にリストされている他の設定パラメータも追加します。

```
マスタ: SERVER.EXAMPLE.COM

grains:
  susemanager:
    activation_key: "<Activation_Key_Name>"

  server_id_use_crc: adler32
  enable_legacy_startup_events: False
  enable_fqdns_grains: False
```

3. **salt-minion** サービスを再起動します。

```
systemctl restart salt-minion
```

4. Uyuniサーバで、新しいクライアントキーを受け入れます。 **<client>** をクライアントの名前に置き換えます。

```
salt-key -a '<client>'
```



Salt minion設定ファイルの詳細については、[link:https://docs.saltstack.com/en/latest/ref/configuration/minion.html](https://docs.saltstack.com/en/latest/ref/configuration/minion.html)を参照してください。



CentOS 6、Oracle Linux 6、Red Hat Enterprise Linux 6、またはSUSE Linux Enterprise Server with Expanded Support 6の各クライアントを登録して使用するには、Uyuniサーバを設定して旧式のSSL暗号化をサポートする必要があります。このエラーを解決する方法については、**古いクライアントの登録**をClient-configuration > Tshoot-clientsで参照してください。

4.2. SUSEクライアントの登録

SUSE Linux EnterpriseおよびSUSE Linux Enterprise Server with Expanded Supportの各クライアントをUyuniサーバに登録できます。そのメソッドおよび詳細は、クライアントのオペレーティングシステムによって異なります。

始める前に、クライアントでUyuniサーバと日時が正しく同期していることを確認してください。

アクティベーションキーを作成済みである必要もあります。アクティベーションキーの作成の詳細については、Client-configuration > Activation-keysを参照してください。



Uyuniサーバをこのサーバ自身に登録しないでください。Uyuniサーバは個別に管理するか、別のUyuniサーバを使用して管理する必要があります。複数のサーバを使用する方法の詳細については、Specialized-guides > Large-deploymentsを参照してください。

4.2.1. SUSE Linux Enterpriseクライアントの登録

このセクションでは、次のSUSE Linux Enterpriseオペレーティングシステムを実行しているクライアントの登録について説明します。

- ・ SUSE Linux Enterprise Server 15 SP1
- ・ SUSE Linux Enterprise Server 15 SP2
- ・ SUSE Linux Enterprise Server 15 SP3
- ・ SUSE Linux Enterprise Server 15 SP4

以下を含むすべてのSUSE Linux Enterprise製品を準備する際には、この章の手順を使用してください。

- ・ SUSE Linux Enterprise Server for SAP
- ・ SUSE Linux Enterprise Desktop
- ・ SUSE Linux Enterprise
- ・ SUSE Linux Enterprise Real Time

これらの手順は、古いSUSE Linux Enterpriseバージョンおよびサービスパックにも使用できます。

4.2.1.1. ソフトウェアチャンネルの追加



In the following section, descriptions often default to the `x86_64` architecture. Replace it with other architectures if appropriate.

SUSE Linux EnterpriseクライアントをUyuniサーバに登録する前に、必要なソフトウェアチャンネルを追加して同期する必要があります。

このプロセッジヤで必要な製品は次のとおりです。

表 15. SLE製品 - WebUI

OSバージョン	製品名
SUSE Linux Enterprise Server 12 SP5	SUSE Linux Enterprise Server 12 SP5 x86_64
SUSE Linux Enterprise Server 15 SP1	SUSE Linux Enterprise Server 15 SP1 x86_64
SUSE Linux Enterprise Server 15 SP2	SUSE Linux Enterprise Server 15 SP2 x86_64
SUSE Linux Enterprise Server 15 SP3	SUSE Linux Enterprise Server 15 SP3 x86_64
SUSE Linux Enterprise Server 15 SP4	SUSE Linux Enterprise Server 15 SP4 x86_64

プロセッジヤ: ソフトウェアチャンネルの追加

1. UyuniのWeb UIで、**管理** > **セットアップウィザード** > **製品**に移動します。
2. 検索バーを使用してクライアントのオペレーティングシステムおよびアーキテクチャに適切な製品を探し、適切な製品にチェックを付けます。こうすることによって、すべての必須チャンネルに自動的にチェックが付きます。また、**include recommended** トグルがオンになっている場合、すべての推奨チャンネルにもチェックが付きます。矢印をクリックして関連製品の一覧を表示し、必要な追加製品にチェックが付いていることを確認します。
3. **[製品の追加]** をクリックし、製品の同期が完了するまで待機します。

または、コマンドプロンプトでチャンネルを追加できます。このプロセッジヤで必要なチャンネルは次のとおりです。

表 16. SLE製品 - CLI

OSバージョン	ベースチャンネル
SUSE Linux Enterprise Server 12 SP5	sle-product-sles12-sp5-pool-x86_64
SUSE Linux Enterprise Server 15 SP1	sle-product-sles15-sp1-pool-x86_64
SUSE Linux Enterprise Server 15 SP2	sle-product-sles15-sp2-pool-x86_64
SUSE Linux Enterprise Server 15 SP3	sle-product-sles15-sp3-pool-x86_64
SUSE Linux Enterprise Server 15 SP4	sle-product-sles15-sp4-pool-x86_64

古い製品のチャンネル名を見つけるには、Uyuniサーバのコマンドプロンプトで root になり、**mgr-sync** コマンドを使用します:

```
mgr-sync list --help
```



次に、関心のある引数を指定します。たとえば、**channels** を指定します:

```
mgr-sync list channels [-c]
```

手順: コマンドプロンプトからのソフトウェアチャンネルの追加

1. Uyuni サーバのコマンドプロンプトで root になり、**mgr-sync** コマンドを特定のチャンネルに対して実行します:

```
mgr-sync add channel <channel_label_1>
mgr-sync add channel <channel_label_2>
mgr-sync add channel <channel_label_n>
```

2. 同期は自動的に開始されます。チャンネルを手動で同期する場合、次のコマンドを使用します。

```
mgr-sync sync --with-children <channel_name>
```

3. 続行前に、同期が完了していることを確認してください。

クライアントツールを追加するには、コマンドプロンプトからこれらのチャンネルを追加します。

表 17. SUSE Linux Enterprise チャンネル - CLI

OSバージョン	クライアントチャンネル
SUSE Linux Enterprise Server 12 SP5	sles12-sp5-uyuni-client
SUSE Linux Enterprise Server 15 SP1	sles15-sp1-uyuni-client
SUSE Linux Enterprise Server 15 SP2	sles15-sp2-uyuni-client
SUSE Linux Enterprise Server 15 SP3	sles15-sp3-uyuni-client
SUSE Linux Enterprise Server 15 SP4	sles15-sp4-uyuni-client

手順: コマンドプロンプトからのソフトウェアチャンネルの追加

1. Uyuni サーバのコマンドプロンプトで root になり、**spacewalk-common-channels** コマンドを特定のチャンネルに対して実行します:

```
spacewalk-common-channels \
<ベースチャンネルのラベル> \
<子チャンネル_1> \
<子チャンネル_2> \
... <子チャンネル_n>
```

2. チャンネルの同期:

```
spacewalk-repo-sync -p <base_channel_label>
```

3. 続行前に、同期が完了していることを確認してください。

4.2.1.2. 同期ステータスの確認

プロシージャ: Web UIから同期の進捗状況を確認する

1. UyuniのWeb UIで、**ソフトウェア** > **管理** > **チャンネル**に移動し、リポジトリに関連付けられているチャネルをクリックします。
2. [リポジトリ] タブに移動し、[同期] をクリックし、[同期状態] をクリックします。

プロシージャ: コマンドプロンプトから同期の進捗状況を確認する

1. Uyuniサーバのコマンドプロンプトで、rootとして、**tail** コマンドを使用して同期ログファイルを確認します。

```
tail -f /var/log/rhn/reposync/<channel-label>.log
```

2. それぞれの子チャネルは、同期の進捗中にそれぞれのログを生成します。同期が完了したことを確認するには、ベースチャネルと子チャネルのログファイルをすべて確認する必要があります。



SUSE Linux Enterpriseチャンネルは非常に大きいことがあります。同期に数時間かかる場合があります。

4.2.1.3. クライアントでGPGキーを信頼する

4.2.1.4. クライアントでGPGキーを信頼する

Operating systems either trust their own GPG keys directly or at least ship them installed with the minimal system. But third party packages signed by a different GPG key need manual handling. The clients can be successfully bootstrapped without the GPG key being trusted. However, you cannot install new client tool packages or update them until the keys are trusted.

Salt clients use now GPG key information entered for a software channel to manage the trusted keys. When a software channel with GPG key information is assigned to a client, the key gets trusted as soon

as the channel is refreshed or the first package gets installed from this channel.

The GPG key URL which is set of a software channel must exist. In case it is a file URL, the GPG key file must be deployed on the client before the software channel is used.

The GPG keys for the Client Tools Channels of Red Hat based clients are deployed on the client into `/etc/pki/rpm-gpg/` and can be referenced with file URLs. Same is the case with the GPG keys of Expanded Support clients. Only in case a software channel is assigned to the client they will be imported and trusted by the system.



Because Debian based systems sign only metadata, there is typically no need to specify extra keys for single channels. If a user configures an own GPG key to sign the metadata as described in "Use Your Own GPG Key" in **Administration > Repo-metadata** the deployment and trust of that key is executed automatically.

4.2.1.4.1. User defined GPG keys

Users can define their own GPG keys to be deployed to the client.

By providing some pillar data and providing the GPG key files in the Salt filesystem, they are automatically deployed to the client.

These keys are deployed into `/etc/pki/rpm-gpg/` on RPM based operating systems and to `/usr/share/keyrings/` on Debian systems:

Define the pillar key [literal `custom_gpgkeys`] for the client you want to deploy the key to and list the names of the key file.

```
cat /etc/pillar/mypillar.sls
custom_gpgkeys:
  - my_first_gpg.key
  - my_second_gpgkey.gpg
```

Additionally in the Salt filesystem create a directory named `gpg` and store there the GPG key files with the name specified in the `custom_gpgkeys` pillar data.

```
ls -la /etc/salt/gpg/
/etc/salt/gpg/my_first_gpg.key
/etc/salt/gpg/my_second_gpgkey.gpg
```

The keys are now deployed to the client at `/etc/pki/rpm-gpg/my_first_gpg.key` and `/etc/pki/rpm-gpg/my_second_gpgkey.gpg`.

The last step is to add the URL to the GPG key URL field of the software channel. Navigate to **Software > Manage > Channels** and select the channel you want to modify. Add to **GPG key URL** the value `file:///etc/`

```
pki/ rpm-gpg/ my_first_gpg.key
```

4.2.1.4.2. GPG Keys in Bootstrap Scripts

プロシージャ: ブートストラップスクリプトを使用してクライアントでGPGキーを信頼する

1. Uyuniサーバのコマンドプロンプトで、`/srv/www/htdocs/pub/`ディレクトリの内容を確認します。このディレクトリには、使用できるすべての公開鍵が含まれています。登録クライアントに割り当てるチャンネルに適用するキーをメモします。
2. 関連するブートストラップスクリプトを開き、`ORG_GPG_KEY=`パラメータを見つけて、必要なキーを追加します。次に例を示します。

```
uyuni-gpg-pubkey-0d20833e.key
```

以前保存したキーを削除する必要はありません。



Trusting a GPG key is important for security on clients. It is the task of the admin to decide which keys are needed and can be trusted. Because a software channel cannot be used when the GPG key is not trusted, the decision of assigning a channel to a client depends on the decision of trusting the key.



SUSE Linux Enterprise Server 15とSUSE Linux Enterprise Server 12クライアントの両方で同じGPGキーを使用します。正しいキーは`sle12-gpg-pubkey-39db7c82.key`と呼ばれます。

4.2.1.5. クライアントの登録

SUSE Linux Enterpriseクライアントを登録するには、ブートストラップリポジトリが必要です。デフォルトでは、ブートストラップリポジトリは自動的に作成され、すべての同期製品に対して毎日再生成されます。次のコマンドを使用して、コマンドプロンプトからブートストラップリポジトリを手動で作成できます。

```
mgr-create-bootstrap-repo
```

クライアントの登録については、[Client-configuration > Registration-overview](#)を参照してください。

4.2.2. SLE Microクライアントの登録

このセクションでは、次のSLE Microオペレーティングシステムを実行しているクライアントの登録について説明します。

- SLE Micro 5.1 x86-64
- SLE Micro 5.1 ARM64



SLE Microクライアントのサポートは、テスト目的のテクノロジプレビューとして提供されるものであり、この段階ですべての機能が完全に動作するわけではありません。この機能は、Uyuniの後続バージョンですべてサポートされる予定です。運用システムでは、この機能を使用しないでください。

SLE Microは、エッジコンピューティング向けに構築された、極めて信頼性が高く軽量なオペレーティングシステムです。SUSE Linux Enterpriseのエンタープライズレベルの強化されたセキュリティおよびコンプライアンスコンポーネントを活用し、最新の不变の開発者向けOSプラットフォームと統合します。

SLE Microはトランザクション更新を使用します。トランザクション更新はアトミックであり(すべての更新はすべての更新が成功した場合にのみ適用されます)、ロールバックをサポートします。システムが再起動されるまで変更はアクティブ化されないため、実行中のシステムには影響しません。この情報は、[システム > 詳細 > 概要](#)サブタブに表示されます。

トランザクション更新の詳細については、<https://documentation.suse.com/sles/15-SP3/html/SLES-all/cha-transactional-updates.html>を参照してください。



DVDまたはISOイメージからインストールすると、`salt-transactional-update`およびSaltや`python3`などの依存関係はインストールされません。SLE MicroクライアントをUyuniに登録するにはこれらのパッケージが必要です。このクライアントを登録する前に、クライアントで`root`として次のコマンドを実行します。

```
transactional-update pkg install salt-transactional-update
```

4.2.2.1. ソフトウェアチャンネルの追加

SLE MicroクライアントをUyuniサーバに登録する前に、必要なソフトウェアチャンネルを追加して同期する必要があります。



In the following section, descriptions often default to the `x86_64` architecture. Replace it with other architectures if appropriate.

4.2.2.2. 同期ステータスの確認

プロシージャ: Web UIから同期の進捗状況を確認する

1. UyuniのWeb UIで、[ソフトウェア > 管理 > チャンネル](#)に移動し、リポジトリに関連付けられているチャンネルをクリックします。
2. [リポジトリ] タブに移動し、[同期] をクリックし、[同期状態] をクリックします。

プロシージャ: コマンドプロンプトから同期の進捗状況を確認する

1. Uyuniサーバのコマンドプロンプトで、`root`として、`tail`コマンドを使用して同期ログファイルを確認します。

```
tail -f /var/log/rhn/reposync/<channel-label>.log
```

2. それぞれの子チャンネルは、同期の進捗中にそれぞれのログを生成します。同期が完了したことを確認するには、ベースチャンネルと子チャンネルのログファイルをすべて確認する必要があります。

4.2.2.3. クライアントの登録

クライアントの登録については、[Client-configuration > Registration-overview](#)を参照してください。

4.2.3. SUSE Linux Enterprise Server with Expanded Supportクライアントの登録

このセクションでは、SUSE Linux Enterprise Server with Expanded Support (Expanded Support)オペレーティングシステムを実行している従来のクライアントおよびSaltクライアントの登録について説明します。 Expanded SupportクライアントはRed Hat Enterprise LinuxまたはCentOSに基づいています。これらは、SLESES、RESまたはRed Hat Expanded Supportと呼ばれることもあります。

SUSEによって提供されるExpanded Supportソフトウェアチャンネルは、パッケージの更新のみを提供します。パッケージそのものは提供しません。 Expanded Support クライアントを登録するには、Expanded Support製品(概要は以下を参照)を登録して必要なベースチャンネルを作成し、必要なRed HatまたはCentOSパッケージをカスタム子チャンネルとしてインポートする必要があります。 Expanded Supportソフトウェアチャンネルで提供される更新を適用する前に、初期パッケージをRed HatまたはCentOSから直接取得する必要があります。



ユーザは、Red HatまたはCentOSのベースメディアリポジトリおよびインストールメディアをアクセスできるようにする必要があります。



SUSEは、UyuniでのExpanded Supportオペレーティングシステムをサポートしていません。



従来のクライアントはExpanded Support 8では使用できません。 Expanded Support 8クライアントはSaltクライアントとしてのみサポートされます。

4.2.3.1. ソフトウェアチャンネルの追加

Expanded Supportクライアントでは、必要なパッケージの一部がRed Hat Enterprise LinuxまたはCentOSのインストールメディアに含まれています。 Expanded Supportクライアントを登録するには、その前にこれらのパッケージをインストールする必要があります。

Expanded Support製品はSUSE Customer Centerによって提供されます。 これには、クライアントツールパッケージも含まれています。

Expanded SupportクライアントをUyuniサーバに登録する前に、必要なソフトウェアチャンネルを追加して同期する必要があります。

2つの異なるチャンネルセットを選択する必要があります。一方はExpanded Support用、他方はクライアントツール用です。

正しいExpanded Supportチャンネルに関連付けられているアクティベーションキーが必要です。 アクティベーションキーの詳細については、[Client-configuration > Activation-keys](#)を参照してください。

このプロシージャで必要なチャンネルは次のとおりです。

表 18. ESチャンネル - CLI

OSバージョン	ベースチャンネル	クライアントチャンネル	ツールチャンネル
Expanded Support 6	rhel-x86_64-server-6	-	res6-suse-manager-tools-x86_64
Expanded Support 7	rhel-x86_64-server-7	-	res7-suse-manager-tools-x86_64
Expanded Support 8	rhel8-pool-x86_64	-	res8-suse-manager-tools-x86_64



Expanded Support 6 はサポート終了になっており、そのリポジトリで提供されるISOイメージは失効しています。 これらのパッケージを使用した新しいExpanded Support 6クライアントのブートストラップは失敗します。 新しいExpanded Support 6クライアントをブートストラップする必要がある場合、[Client-configuration > Tshoot-clients](#)のトラブルシューティングプロシージャに従ってください。

手順: コマンドプロンプトからのソフトウェアチャンネルの追加

1. Uyuni サーバのコマンドプロンプトで root になり、`spacewalk-common-channels` コマンドを特定のチャンネルに対して実行します:

```
spacewalk-common-channels \
<ベースチャンネルのラベル> \
<子チャンネル_1> \
<子チャンネル_2> \
... <子チャンネル_n>
```

2. チャンネルの同期:

```
spacewalk-repo-sync -p <base_channel_label>
```

3. 続行前に、同期が完了していることを確認してください。

AppStreamリポジトリにはモジュールパッケージが用意されています。UyuniのWeb UIに正しくないパッケージ情報が表示されます。Web UIまたはAPIを使用してモジュールリポジトリから直接インストールまたはアップグレードするようなパッケージ操作は実行できません。



コンテンツライフサイクル管理(CLM)でAppStreamフィルタを使用して、モジュールリポジトリを通常のリポジトリに変換できます。クライアントで `spacecmd` を使用する場合は、AppStreamフィルタを使用して `python:3.6` を必ず含めてください。

または、Salt状態を使用してSaltクライアントでモジュラー・パッケージを管理したり、クライアントで `dnf` コマンドを使用することもできます。CLMの詳細については、Administration > Content-lifecycleを参照してください。

4.2.3.1.1. ベースメディアの追加

ベースExpanded Supportチャンネルには、パッケージは含まれていません。その理由は、SUSEがRed Hat Enterprise LinuxまたはCentOSのベースメディアを提供しないためです。ベースメディアをRed HatまたはCentOSから取得する必要があります。ベースメディアは、子チャンネルとしてExpanded Support親チャンネルに追加できます。必要なパッケージのすべてがあることを確認するには、最小版やJeOSイメージではなく、フルDVDイメージを使用します。

Uyuniカスタムチャンネルを使用して、Red Hat Enterprise LinuxまたはCentOSのメディアを設定できます。ベースメディアのすべてのパッケージは、子チャンネルにミラーリングする必要があります。

チャンネルの名前は自由に選択できます。

プロシージャ: カスタムチャンネルの作成

1. UyuniサーバのWeb UIで、**ソフトウェア** > **管理** > **チャンネル**に移動します。
2. [チャンネルの作成]をクリックし、チャンネルに適切なパラメータを設定します。
3. [親チャンネル]フィールドで、適切なベースチャンネルを選択します。
4. [チャンネルの作成]をクリックします。
5. 作成する必要があるすべてのチャンネルで繰り返します。各カスタムリポジトリに1つのカスタムチャンネルが必要です。

該当するすべてのチャンネルとリポジトリを作成したことを確認できます。そのためには、**ソフトウェア** > **チャンネル一覧** > **すべて**に移動します。



Red Hat 8クライアントでは、ベースチャンネルとAppStreamチャンネルの両方を追加します。両方のチャンネルのパッケージが必要です。両方のチャンネルを追加しないと、パッケージ不足のためブートストラップリポジトリを作成できません。

モジュラー・チャンネルを使用している場合は、クライアントでPython3.6モジュールストリームを有効にする必要があります。Python 3.6を提供しない場合、`spacecmd` パッケージのインストールは失敗します。

プロシージャ: ベースメディアをカスタムチャンネルに追加する

1. Uyuniサーバのコマンドプロンプトでrootとしてベースメディアイメージを /tmp/ ディレクトリにコピーします。
2. メディアコンテンツを含むディレクトリを作成します。 <os_name> を sleses6、 sleses7、 または sleses8 のいずれかに置き換えます。

```
mkdir -p /srv/www/htdocs/pub/<os_name>
```

3. イメージをマウントします。

```
mount -o loop /tmp/<iso_filename> /srv/www/htdocs/pub/<os_name>
```

4. 前に作成した子チャンネルにパッケージをインポートします。

```
spacewalk-repo-sync -c <channel-label> -u  
file:///srv/www/htdocs/pub/<os_name>/<repopath>/
```

オプション: ベースチャンネルをコンテンツURLから追加する

または、Red Hat CDNまたはCentOSが提供するコンテンツURLにアクセスできる場合、カスタムリポジトリを作成してパッケージをミラーリングできます。

このプロシージャに必要な詳細は次のとおりです。

表 19. ESカスタムリポジトリ設定

オプション	パラメータ
リポジトリURL	Red Hat CDNまたはCentOSによって提供されるコンテンツURL
署名済みメタデータがあるかどうか	すべてのRed Hatエンタープライズリポジトリのチェックを外します
SSL CA証明書	redhat-uep (Red Hatのみ)
SSLクライアント証明書	Entitlement-Cert-date (Red Hatのみ)
SSLクライアントキー	Entitlement-Key-date (Red Hatのみ)

プロシージャ: カスタムリポジトリの作成

1. UyuniサーバのWeb UIで、[ソフトウェア](#) > [管理](#) > [リポジトリ](#)に移動します。
2.  をクリックし、リポジトリに適切なパラメータを設定します。
3.  をクリックします。

4. 作成する必要があるすべてのリポジトリで繰り返します。

すべてのチャンネルを作成済みの場合、これらのチャンネルを、作成したリポジトリと関連付けできます。

プロシージャ: チャンネルのリポジトリとの関連付け

1. UyuniサーバのWeb UIで、**ソフトウェア > 管理 > チャンネル**に移動し、関連付けるチャンネルをクリックします。
2. [リポジトリ] タブに移動し、このチャンネルと関連付けるリポジトリにチェックを付けます。
3. [リポジトリの更新] をクリックし、チャンネルとリポジトリを関連付けます。
4. 関連付ける必要があるすべてのチャンネルとすべてのリポジトリを繰り返します。
5. オプション: [同期] タブに移動し、このリポジトリの同期の繰り返しスケジュールを設定します。
6. [今すぐ同期] をクリックし、すぐに同期を開始します。

4.2.3.2. 同期ステータスの確認

プロシージャ: Web UIから同期の進捗状況を確認する

1. UyuniのWeb UIで、**ソフトウェア > 管理 > チャンネル**に移動し、リポジトリに関連付けられているチャンネルをクリックします。
2. [リポジトリ] タブに移動し、[同期] をクリックし、[同期状態] をクリックします。

プロシージャ: コマンドプロンプトから同期の進捗状況を確認する

1. Uyuniサーバのコマンドプロンプトで、rootとして、**tail** コマンドを使用して同期ログファイルを確認します。

```
tail -f /var/log/rhn/reposync/<channel-label>.log
```

2. それぞれの子チャンネルは、同期の進捗中にそれぞれのログを生成します。同期が完了したことを確認するには、ベースチャンネルと子チャンネルのログファイルをすべて確認する必要があります。



Expanded Supportチャンネルは非常に大きいことがあります。最初のチャンネル同期は数時間かかる場合があります。

最初の同期が完了したとき、このチャンネルを使用する前にチャンネルを複製することをお勧めします。この操作を実行すると、元の同期データのバックアップを作成できます。

4.2.3.3. Expanded Supportクライアントの登録

Expanded Supportクライアントは、登録できるようになりました。

クライアントの登録については、**Client-configuration > Registration-overview**を参照してください。



SUSE Linux Enterprise Server with Expanded Support 6クライアントを登録して使用するには、Uyuniサーバを設定して旧式のSSL暗号化をサポートする必要がある。このエラーを解消する方法については、[古いクライアントの登録をClient-configuration > Tshoot-clients](#)で参照してください。

4.3. openSUSEクライアントの登録

openSUSEクライアントをUyuniサーバに登録できます。そのメソッドおよび詳細は、クライアントのオペレーティングシステムによって異なります。

始める前に、クライアントでUyuniサーバと日時が正しく同期していることを確認してください。

アクティベーションキーを作成済みである必要があります。アクティベーションキーの作成の詳細については、[Client-configuration > Activation-keys](#)を参照してください。



Uyuniサーバをこのサーバ自身に登録しないでください。Uyuniサーバは個別に管理するか、別のUyuniサーバを使用して管理する必要があります。複数のサーバを使用する方法の詳細については、[Specialized-guides > Large-deployments](#)を参照してください。

4.3.1. openSUSE Leapクライアントの登録

このセクションでは、openSUSEオペレーティングシステムを実行しているSaltクライアントの登録について説明します。Uyuniは、Saltを使用するSUSE Leap 15クライアントをサポートします。従来のクライアントはサポートされていません。

ブートストラップは、リポジトリの設定やプロファイルの更新の実行など、openSUSEクライアントの起動および初期状態の実行のためにサポートされています。

4.3.1.1. ソフトウェアチャンネルの追加

openSUSEクライアントをUyuniサーバに登録する前に、必要なソフトウェアチャンネルを追加して同期する必要があります。

現在サポートされているアーキテクチャは、「`x86_64`」と「`aarch64`」です。サポートされている製品およびアーキテクチャの完全な一覧については、[Client-configuration > Supported-features](#)を参照してください。



In the following section, descriptions often default to the `x86_64` architecture. Replace it with other architectures if appropriate.

たとえば、「`x86_64`」アーキテクチャを使用する場合は、次の製品が必要です。

表 20. OpenSUSEチャンネル - CLI

OSバージョン	ベースチャンネル	クライアントチャンネル	更新チャンネル	非オープンソースチャンネル	非オープンソース更新チャンネル
openSUSE Leap 15.1	opensuse_leap_15_1	opensuse_leap_15_1-uyuni-client	opensuse_leap_15_1-updates	opensuse_leap_15_1-non-oss	opensuse_leap_15_1-non-oss-updates
openSUSE Leap 15.2	opensuse_leap_15_2	opensuse_leap_15_2-uyuni-client	opensuse_leap_15_2-updates	opensuse_leap_15_2-non-oss	opensuse_leap_15_2-non-oss-updates

表 21. OpenSUSEチャンネル - CLI

OS Version	Base Channel	Client Channel	Updates Channel	Non-OSS Channel	Non-OSS Updates Channel	Backports Updates Channel	SLE Updates Channel
openSUSE Leap 15.3	opensuse_leap15_3	opensuse_leap15_3-uyuni-client	opensuse_leap15_3-updates	opensuse_leap15_3-non-oss	opensuse_leap15_2-non-oss-updates	opensuse_leap15_3-backports-update	opensuse_leap15_3-sle-updates
openSUSE Leap 15.4	opensuse_leap15_4	opensuse_leap15_4-uyuni-client	opensuse_leap15_4-updates	opensuse_leap15_4-non-oss	opensuse_leap15_4-non-oss-updates	opensuse_leap15_4-backports-update	opensuse_leap15_4-sle-updates

手順: コマンドプロンプトからのソフトウェアチャンネルの追加

1. Uyuni サーバのコマンドプロンプトで root になり、 **spacewalk-common-channels** コマンドを特定のチャンネルに対して実行します:

```
spacewalk-common-channels \
<ベースチャンネルのラベル> \
<子チャンネル_1> \
<子チャンネル_2> \
... <子チャンネル_n>
```

2. チャンネルの同期:

```
spacewalk-repo-sync -p <base_channel_label>
```

3. 続行前に、同期が完了していることを確認してください。

4.3.1.2. 同期ステータスの確認

プロシージャ: Web UIから同期の進捗状況を確認する

1. UyuniのWeb UIで、**ソフトウェア** > **管理** > **チャンネル**に移動し、リポジトリに関連付けられているチャンネルをクリックします。
2. [リポジトリ] タブに移動し、[同期] をクリックし、[同期状態] をクリックします。

プロシージャ: コマンドプロンプトから同期の進捗状況を確認する

1. Uyuniサーバのコマンドプロンプトで、rootとして、**tail** コマンドを使用して同期ログファイルを確認します。

```
tail -f /var/log/rhn/reposync/<channel-label>.log
```

2. それぞれの子チャンネルは、同期の進捗中にそれぞれのログを生成します。同期が完了したことを確認するには、ベースチャンネルと子チャンネルのログファイルをすべて確認する必要があります。



openSUSEチャンネルは非常に大きいことがあります。同期に数時間かかる場合があります。

4.3.1.3. クライアントでGPGキーを信頼する

4.3.1.4. クライアントでGPGキーを信頼する

Operating systems either trust their own GPG keys directly or at least ship them installed with the minimal system. But third party packages signed by a different GPG key need manual handling. The clients can be successfully bootstrapped without the GPG key being trusted. However, you cannot install new client tool packages or update them until the keys are trusted.

Salt clients use now GPG key information entered for a software channel to manage the trusted keys. When a software channel with GPG key information is assigned to a client, the key gets trusted as soon as the channel is refreshed or the first package gets installed from this channel.

The GPG key URL which is set of a software channel must exist. In case it is a file URL, the GPG key file must be deployed on the client before the software channel is used.

The GPG keys for the Client Tools Channels of Red Hat based clients are deployed on the client into **/etc/pki/rpm-gpg/** and can be referenced with file URLs. Same is the case with the GPG keys of Expanded Support clients. Only in case a software channel is assigned to the client they will be imported and trusted by the system.



Because Debian based systems sign only metadata, there is typically no need to specify extra keys for single channels. If a user configures an own GPG key to sign the metadata as described in "Use Your Own GPG Key" in **Administration** > **Repo-metadata** the deployment and trust of that key is executed automatically.

4.3.1.4.1. User defined GPG keys

Users can define their own GPG keys to be deployed to the client.

By providing some pillar data and providing the GPG key files in the Salt filesystem, they are automatically deployed to the client.

These keys are deployed into `/etc/pki/rpm-gpg/` on RPM based operating systems and to `/usr/share/keyrings/` on Debian systems:

Define the pillar key [literal `custom_gpgkeys`] for the client you want to deploy the key to and list the names of the key file.

```
cat /etc/pillar/mypillar.sls
custom_gpgkeys:
  - my_first_gpg.key
  - my_second_gpgkey.gpg
```

Additionally in the Salt filesystem create a directory named `gpg` and store there the GPG key files with the name specified in the `custom_gpgkeys` pillar data.

```
ls -la /etc/salt/gpg/
/etc/salt/gpg/my_first_gpg.key
/etc/salt/gpg/my_second_gpgkey.gpg
```

The keys are now deployed to the client at `/etc/pki/rpm-gpg/my_first_gpg.key` and `/etc/pki/rpm-gpg/my_second_gpgkey.gpg`.

The last step is to add the URL to the GPG key URL field of the software channel. Navigate to **Software > Manage > Channels** and select the channel you want to modify. Add to **GPG key URL** the value `file:///etc/pki/rpm-gpg/ my_first_gpg.key`.

4.3.1.4.2. GPG Keys in Bootstrap Scripts

プロシージャ: ブートストラップスクリプトを使用してクライアントでGPGキーを信頼する

- Uyuniサーバのコマンドプロンプトで、`/srv/www/htdocs/pub/`ディレクトリの内容を確認します。このディレクトリには、使用できるすべての公開鍵が含まれています。登録クライアントに割り当てるチャンネルに適用するキーをメモします。
- 関連するブートストラップスクリプトを開き、`ORG_GPG_KEY=`パラメータを見つけて、必要なキーを追加します。次に例を示します。

```
uyuni-gpg-pubkey-0d20833e.key
```

以前保存したキーを削除する必要はありません。



Trusting a GPG key is important for security on clients. It is the task of the admin to decide which keys are needed and can be trusted. Because a software channel cannot be used when the GPG key is not trusted, the decision of assigning a channel to a client depends on the decision of trusting the key.

4.3.1.5. クライアントの登録

openSUSEクライアントを登録するには、ブートストラップリポジトリが必要です。デフォルトでは、ブートストラップリポジトリは自動的に作成され、すべての同期製品に対して毎日再生成されます。次のコマンドを使用して、コマンドプロンプトからブートストラップリポジトリを手動で作成できます。

```
mgr-create-bootstrap-repo
```

クライアントの登録については、[Client-configuration > Registration-overview](#)を参照してください。

4.3.2. openSUSE MicroOSクライアントの登録

このセクションでは、次のopenSUSE MicroOSオペレーティングシステムを実行しているクライアントの登録について説明します。

- openSUSE MicroOS



openSUSE MicroOSクライアントのサポートは、テスト目的のテクノロジプレビューとして提供されるものであり、この段階ですべての機能が完全に動作するわけではありません。この機能は、Uyuniの後続バージョンですべてサポートされる予定です。運用システムでは、この機能を使用しないでください。

4.3.2.1. ソフトウェアチャンネルの追加

openSUSE MicroOSクライアントをUyuniサーバに登録する前に、必要なソフトウェアチャンネルを追加して同期する必要があります。



In the following section, descriptions often default to the `x86_64` architecture. Replace it with other architectures if appropriate.

このプロシージャで必要なチャンネルは次のとおりです。

表 22. openSUSE MicroOSチャンネル - CLI

OSバージョン	ベースチャンネル	クライアントチャンネル	更新チャンネル
openSUSE MicroOS	<code>opensuse_tumbleweed</code>	<code>opensuse_tumbleweed-non-oss</code>	<code>opensuse_tumbleweed-update</code>

手順: コマンドプロンプトからのソフトウェアチャンネルの追加

- Uyuni サーバのコマンドプロンプトで root になり、 **spacewalk-common-channels** コマンドを特定のチャンネルに対して実行します:

```
spacewalk-common-channels \
<ベースチャンネルのラベル> \
<子チャンネル_1> \
<子チャンネル_2> \
... <子チャンネル_n>
```

- チャンネルの同期:

```
spacewalk-repo-sync -p <base_channel_label>
```

- 続行前に、同期が完了していることを確認してください。

4.3.2.2. 同期ステータスの確認

プロシージャ: Web UIから同期の進捗状況を確認する

- UyuniのWeb UIで、**ソフトウェア** > **管理** > **チャンネル**に移動し、リポジトリに関連付けられているチャンネルをクリックします。
- [リポジトリ] タブに移動し、[同期] をクリックし、[同期状態] をクリックします。

プロシージャ: コマンドプロンプトから同期の進捗状況を確認する

- Uyuniサーバのコマンドプロンプトで、rootとして、**tail** コマンドを使用して同期ログファイルを確認します。

```
tail -f /var/log/rhn/reposync/<channel-label>.log
```

- それぞれの子チャンネルは、同期の進捗中にそれぞれのログを生成します。同期が完了したことを確認するには、ベースチャンネルと子チャンネルのログファイルをすべて確認する必要があります。



openSUSE MicroOSチャンネルは非常に大きいことがあります。同期に数時間かかる場合があります。

4.3.2.3. クライアントで証明書キーを信頼する

openSUSE MicroOSはまだすべて有効になっていないため、UyuniクライアントでSSL証明書MicroOSを信頼するには手動の手順が必要です。

手順: Saltのインストールと設定

- クライアントのコマンドプロンプトで、rootとしてサーバからSSL証明書ファイルを取得します。

```
curl -k https://uyuni-server.hispa-net.com/pub/RHN-ORG-TRUSTED-SSL-CERT -o /etc/pki/trust/anchors/RHN-ORG-TRUSTED-SSL-CERT
```

2. クライアントで証明書を更新します。

```
update-ca-certificates
```

3. 必要なパッケージをインストールします。

```
transactional-update pkg install salt-minion dmidecode
```

4. クライアントを再起動します。**busybox-hostname**と競合していることを示すメッセージが表示された場合は、**[busybox-hostname] の [削除]**をクリックします。
5. このコンテンツで、**/etc/salt/minion.d/susemanager-transactional.conf**という名前の新しいファイルを作成します。

```
module_executors:
- transactional_update
- direct_call
```

クライアントを再起動するまで、Uyuniサーバでは、クライアントの正しい状態がWeb UIに表示されません。この機能は、Uyuniの後継バージョンですべてサポートされる予定です。



Saltがソフトウェアのインストールに失敗する場合は、古いバージョンのSaltを使用している可能性があります。この問題を解決するには、Saltパッケージを最新バージョンにアップグレードします。

4.3.2.4. クライアントの登録

クライアントの登録については、[Client-configuration > Registration-overview](#)を参照してください。

4.4. Alibaba Cloud Linuxクライアントの登録

Alibaba Cloud LinuxクライアントをUyuniサーバに登録できます。そのメソッドおよび詳細は、クライアントのオペレーティングシステムによって異なります。

始める前に、クライアントでUyuniサーバと日時が正しく同期していることを確認してください。



一部のAlibaba Cloud Linux 2インスタンスでは、正常に登録するために2回の試行が必要になります。

アクティベーションキーを作成済みである必要があります。 アクティベーションキーの作成の詳細については、[Client-configuration > Activation-keys](#)を参照してください。

4.4.1. Alibaba Cloud Linuxクライアントの登録

このセクションでは、Alibaba Cloud Linuxオペレーティングシステムを実行している従来のクライアントおよびSaltクライアントの登録について説明します。



従来のスタックはAlibaba Cloud Linux 2で利用できますが、サポートされていません。 Alibaba Cloud Linux 2クライアントはSaltクライアントとしてのみサポートされます。



一部のAlibaba Cloud Linux 2インスタンスでは、正常に登録するために2回の試行が必要になります。

4.4.1.1. ソフトウェアチャンネルの追加

Alibaba Cloud LinuxクライアントをUyuniサーバに登録する前に、必要なソフトウェアチャンネルを追加して同期する必要があります。



In the following section, descriptions often default to the `x86_64` architecture. Replace it with other architectures if appropriate.

このプロシージャで必要なチャンネルは次のとおりです。

表 23. Alibaba Cloud Linuxチャンネル - CLI

OSバージョン	コアチャンネル	更新チャンネル	クライアントチャンネル
Alibaba Cloud Linux 2	alibaba-2	alibaba-2-updates	alibaba-2-uyuni-client

手順: コマンドプロンプトからのソフトウェアチャンネルの追加

1. Uyuni サーバのコマンドプロンプトで root になり、`spacewalk-common-channels` コマンドを特定のチャンネルに対して実行します:

```
spacewalk-common-channels \
<ベースチャンネルのラベル> \
<子チャンネル_1> \
<子チャンネル_2> \
... <子チャンネル_n>
```

2. チャンネルの同期:

```
spacewalk-repo-sync -p <base_channel_label>
```

3. 続行前に、同期が完了していることを確認してください。



spacewalk-common-channels によって提供されるクライアントツールのチャンネルの提供元はUyuniです。SUSEではありません。

4.4.1.2. 同期ステータスの確認

プロシージャ: Web UIから同期の進捗状況を確認する

1. UyuniのWeb UIで、**ソフトウェア** > **管理** > **チャンネル**に移動し、リポジトリに関連付けられているチャンネルをクリックします。
2. [リポジトリ] タブに移動し、[同期] をクリックし、[同期状態] をクリックします。

プロシージャ: コマンドプロンプトから同期の進捗状況を確認する

1. Uyuniサーバのコマンドプロンプトで、rootとして、**tail** コマンドを使用して同期ログファイルを確認します。

```
tail -f /var/log/rhn/reposync/<channel-label>.log
```

2. それぞれの子チャンネルは、同期の進捗中にそれぞれのログを生成します。同期が完了したことを確認するには、ベースチャンネルと子チャンネルのログファイルをすべて確認する必要があります。

4.4.1.3. アクティベーションキーの作成

Alibaba Cloud Linuxチャンネルと関連付けられているアクティベーションキーを作成する必要があります。

アクティベーションキーの詳細については、**Client-configuration** > **Activation-keys**を参照してください。

4.4.1.4. クライアントの登録

Alibaba Cloud Linuxクライアントは、その他すべてのクライアントと同じ方法で登録されます。

一部のAlibaba Cloud Linux 2インスタンスは、最初の試行で登録に失敗します。

これは、Alibaba Cloud Linux 2のイメージに既知のバグがあるためです。

「python-urlgrabber3」パッケージは、Python pip/パッケージとRPM/パッケージの両方で提供されており、最初の登録試行時に競合が発生する可能性があります。

インスタンスが影響を受けるいずれかのイメージバージョンに基づいている場合、クライアントは2回目の登録試行で正しく登録されます。

クライアントの登録の詳細については、**Client-configuration** > **Registration-overview**を参照してください。

4.5. AlmaLinuxクライアントの登録

AlmaLinuxクライアントをUyuniサーバに登録できます。 そのメソッドおよび詳細は、クライアントのオペレーティングシステムによって異なります。

始める前に、クライアントでUyuniサーバと日時が正しく同期していることを確認してください。

アクティベーションキーを作成済みである必要もあります。 アクティベーションキーの作成の詳細については、[Client-configuration > Activation-keys](#)を参照してください。

4.5.1. AlmaLinuxクライアントの登録

このセクションでは、AlmaLinuxオペレーティングシステムを実行しているSaltクライアントの登録について説明します。



従来のクライアントはAlmaLinux 8では使用できません。 AlmaLinux 8クライアントはSaltクライアントとしてのみサポートされます。



AWSで作成するとき、AlmaLinuxインスタンスには、`/etc/machine-id` で常に同じ `machine-id` IDが割り当てられます。 インスタンスを作成した後に、必ず `machine-id` を再生成してください。 詳細については、[Administration > Tshoot-registerclones](#)を参照してください。

4.5.1.1. ソフトウェアチャンネルの追加



AlmaLinuxクライアントのUyuniへの登録は、ターゲットポリシーで適用されるデフォルトのSELinux設定でテストされます。 SELinuxを無効にしてAlmaLinuxクライアントをUyuniに登録する必要があります。

AlmaLinuxクライアントをUyuniサーバに登録する前に、必要なソフトウェアチャンネルを追加して同期する必要があります。

現在サポートされているアーキテクチャは、「`x86_64`」と「`aarch64`」です。 サポートされている製品およびアーキテクチャの完全な一覧については、[Client-configuration > Supported-features](#)を参照してください。



In the following section, descriptions often default to the `x86_64` architecture. Replace it with other architectures if appropriate.

このプロシージャで必要なチャンネルは次のとおりです。

表 24. AlmaLinux チャンネル - CLI

OSバージョン	ベースチャンネル	クライアントチャンネル	AppStreamチャンネル
AlmaLinux 8	almalinux8	almalinux8-uyuni-client	almalinux8-appstream

手順: コマンドプロンプトからのソフトウェアチャンネルの追加

- Uyuni サーバのコマンドプロンプトで root になり、**spacewalk-common-channels** コマンドを特定のチャンネルに対して実行します。このとき、正しいアーキテクチャを指定してください:

```
spacewalk-common-channels \
-a <アーキテクチャ> \
<ベースチャンネル名> \
<子チャンネル_1> \
<子チャンネル_2> \
... <子チャンネル_n>
```

- チャンネルの同期:

```
spacewalk-repo-sync -p <base_channel_label>
```

- 続行前に、同期が完了していることを確認してください。



spacewalk-common-channels によって提供されるクライアントツールのチャンネルの提供元はUyuniです。SUSEではありません。



AlmaLinux 8クライアントでは、ベースチャンネルとAppStreamチャンネルの両方を追加します。両方のチャンネルのパッケージが必要です。両方のチャンネルを追加しないと、パッケージ不足のためブートストラップリポジトリを作成できません。

モジュラーチャンネルを使用している場合は、クライアントでPython3.6モジュールストリームを有効にする必要があります。Python 3.6を提供しない場合、**spacecmd** パッケージのインストールは失敗します。



上流のチャンネルとUyuniチャンネルの間のAppStreamチャンネルで利用できるパッケージ数に不一致が発生する場合があります。また、同時に別の場所で追加したチャンネルを比較すると、数値が異なる場合もあります。AlmaLinuxでリポジトリを管理する方法が原因です。AlmaLinuxでは新しいバージョンがリリースされると古いバージョンのパッケージが削除されますが、Uyuniでは経過年数に関係なくすべてのバージョンが保持されます。

AppStreamリポジトリにはモジュールパッケージが用意されています。UyuniのWeb UIに正しくないパッケージ情報が表示されます。Web UIまたはAPIを使用してモジュールリポジトリから直接インストールまたはアップグレードするようなパッケージ操作は実行できません。



コンテンツライフサイクル管理(CLM)でAppStreamフィルタを使用して、モジュールリポジトリを通常のリポジトリに変換できます。クライアントで `spacecmd` を使用する場合は、AppStreamフィルタを使用して `python:3.6` を必ず含めてください。

または、Salt状態を使用してSaltクライアントでモジュラー・パッケージを管理したり、クライアントで `dnf` コマンドを使用することもできます。CLMの詳細については、[Administration > Content-lifecycle](#)を参照してください。

4.5.1.2. 同期ステータスの確認

プロシージャ: Web UIから同期の進捗状況を確認する

1. UyuniのWeb UIで、[ソフトウェア > 管理 > チャンネル](#)に移動し、リポジトリに関連付けられているチャンネルをクリックします。
2. [リポジトリ] タブに移動し、[同期] をクリックし、[同期状態] をクリックします。

プロシージャ: コマンドプロンプトから同期の進捗状況を確認する

1. Uyuniサーバのコマンドプロンプトで、rootとして、`tail` コマンドを使用して同期ログファイルを確認します。

```
tail -f /var/log/rhn/reposync/<channel-label>.log
```

2. それぞれの子チャンネルは、同期の進捗中にそれぞれのログを生成します。同期が完了したことを確認するには、ベースチャンネルと子チャンネルのログファイルをすべて確認する必要があります。

4.5.1.3. アクティベーションキーの作成

AlmaLinuxチャンネルと関連付けられているアクティベーションキーを作成する必要があります。

アクティベーションキーの詳細については、[Client-configuration > Activation-keys](#)を参照してください。

4.5.1.4. クライアントでGPGキーを信頼する

4.5.1.5. クライアントでGPGキーを信頼する

Operating systems either trust their own GPG keys directly or at least ship them installed with the minimal system. But third party packages signed by a different GPG key need manual handling. The clients can be successfully bootstrapped without the GPG key being trusted. However, you cannot install new client tool packages or update them until the keys are trusted.

Salt clients use now GPG key information entered for a software channel to manage the trusted keys. When a software channel with GPG key information is assigned to a client, the key gets trusted as soon as the channel is refreshed or the first package gets installed from this channel.

The GPG key URL which is set of a software channel must exist. In case it is a file URL, the GPG key file must be deployed on the client before the software channel is used.

The GPG keys for the Client Tools Channels of Red Hat based clients are deployed on the client into `/etc/pki/rpm-gpg/` and can be referenced with file URLs. Same is the case with the GPG keys of Expanded Support clients. Only in case a software channel is assigned to the client they will be imported and trusted by the system.



Because Debian based systems sign only metadata, there is typically no need to specify extra keys for single channels. If a user configures an own GPG key to sign the metadata as described in "Use Your Own GPG Key" in **Administration > Repo-metadata** the deployment and trust of that key is executed automatically.

4.5.1.5.1. User defined GPG keys

Users can define their own GPG keys to be deployed to the client.

By providing some pillar data and providing the GPG key files in the Salt filesystem, they are automatically deployed to the client.

These keys are deployed into `/etc/pki/rpm-gpg/` on RPM based operating systems and to `/usr/share/keyrings/` on Debian systems:

Define the pillar key [literal `custom_gpgkeys`] for the client you want to deploy the key to and list the names of the key file.

```
cat /etc/pillar/mypillar.sls
custom_gpgkeys:
  - my_first_gpg.key
  - my_second_gpgkey.gpg
```

Additionally in the Salt filesystem create a directory named `gpg` and store there the GPG key files with the name specified in the `custom_gpgkeys` pillar data.

```
ls -la /etc/salt/gpg/
/etc/salt/gpg/my_first_gpg.key
/etc/salt/gpg/my_second_gpgkey.gpg
```

The keys are now deployed to the client at `/etc/pki/rpm-gpg/my_first_gpg.key` and `/etc/pki/rpm-gpg/my_second_gpgkey.gpg`.

The last step is to add the URL to the GPG key URL field of the software channel. Navigate to **Software > Manage > Channels** and select the channel you want to modify. Add to **GPG key URL** the value `file:///etc/pki/rpm-gpg/my_first_gpg.key`.

4.5.1.5.2. GPG Keys in Bootstrap Scripts

プロシージャ: ブートストラップスクリプトを使用してクライアントでGPGキーを信頼する

1. Uyuniサーバのコマンドプロンプトで、`/srv/www/htdocs/pub`ディレクトリの内容を確認します。このディレクトリには、使用できるすべての公開鍵が含まれています。登録クライアントに割り当てるチャンネルに適用するキーをメモします。
2. 関連するブートストラップスクリプトを開き、`ORG_GPG_KEY=`パラメータを見つけて、必要なキーを追加します。次に例を示します。

```
uyuni-gpg-pubkey-0d20833e.key
```

以前保存したキーを削除する必要はありません。



Trusting a GPG key is important for security on clients. It is the task of the admin to decide which keys are needed and can be trusted. Because a software channel cannot be used when the GPG key is not trusted, the decision of assigning a channel to a client depends on the decision of trusting the key.

4.5.1.6. クライアントの登録

AlmaLinuxクライアントは、その他すべてのクライアントと同じ方法で登録されます。 詳細について
は、[Client-configuration > Registration-overview. adoc](#)を参照してください。

4.5.1.7. エラーダの管理

AlmaLinuxクライアントを更新するとき、パッケージには更新に関するメタデータが含まれています。

4.6. Amazon Linuxクライアントの登録

Amazon LinuxクライアントをUyuniサーバに登録できます。そのメソッドおよび詳細は、クライアントのオペレーティングシステムによって異なります。

始める前に、クライアントでUyuniサーバと日時が正しく同期していることを確認してください。

アクティベーションキーを作成済みである必要があります。アクティベーションキーの作成の詳細については、[Client-configuration > Activation-keys](#)を参照してください。

4.6.1. Amazon Linuxクライアントの登録

このセクションでは、Amazon Linuxオペレーティングシステムを実行している従来のクライアントおよ

びSaltクライアントの登録について説明します。



従来のクライアントはAmazon Linux 2では使用できません。Amazon Linux 2クライアントはSaltクライアントとしてのみサポートされます。



AWSで作成するとき、Amazon Linuxインスタンスには、`/etc/machine-id` で常に同じ `machine-id` IDが割り当てられます。インスタンスを作成した後に、必ず `machine-id` を再生成してください。詳細については、Administration > Tshoot-registerclonesを参照してください。

4.6.1.1. ソフトウェアチャンネルの追加

Amazon LinuxクライアントをUyuniサーバに登録する前に、必要なソフトウェアチャンネルを追加して同期する必要があります。

現在サポートされているアーキテクチャは、「`x86_64`」と「`aarch64`」です。サポートされている製品およびアーキテクチャの完全な一覧については、Client-configuration > Supported-featuresを参照してください。



In the following section, descriptions often default to the `x86_64` architecture. Replace it with other architectures if appropriate.

このプロシージャで必要なチャンネルは次のとおりです。

表 25. Amazon Linuxチャンネル - CLI

OSバージョン	コアチャンネル	クライアントチャンネル
Amazon Linux 2	<code>amazonlinux2-core</code>	<code>amazonlinux2-uyuni-client</code>



また、Amazon LinuxインスタンスでDockerを使用する場合は、必ず「`amazonlinux2-extra-docker`」チャンネルを追加して同期してください。

手順: コマンドプロンプトからのソフトウェアチャンネルの追加

1. Uyuni サーバのコマンドプロンプトで root になり、`spacewalk-common-channels` コマンドを特定のチャンネルに対して実行します:

```
spacewalk-common-channels \
<ベースチャンネルのラベル> \
<子チャンネル_1> \
<子チャンネル_2> \
... <子チャンネル_n>
```

2. チャンネルの同期:

```
spacewalk-repo-sync -p <base_channel_label>
```

3. 続行前に、同期が完了していることを確認してください。



spacewalk-common-channels によって提供されるクライアントツールのチャンネルの提供元はUyuniです。SUSEではありません。

4.6.1.2. 同期ステータスの確認

プロシージャ: Web UIから同期の進捗状況を確認する

1. UyuniのWeb UIで、**ソフトウェア** > **管理** > **チャンネル**に移動し、リポジトリに関連付けられているチャンネルをクリックします。
2. [リポジトリ] タブに移動し、[同期] をクリックし、[同期状態] をクリックします。

プロシージャ: コマンドプロンプトから同期の進捗状況を確認する

1. Uyuniサーバのコマンドプロンプトで、rootとして、**tail** コマンドを使用して同期ログファイルを確認します。

```
tail -f /var/log/rhn/reposync/<channel-label>.log
```

2. それぞれの子チャンネルは、同期の進捗中にそれぞれのログを生成します。同期が完了したことを確認するには、ベースチャンネルと子チャンネルのログファイルをすべて確認する必要があります。

4.6.1.3. アクティベーションキーの作成

Amazon Linuxチャンネルと関連付けられているアクティベーションキーを作成する必要があります。

アクティベーションキーの詳細については、**Client-configuration** > **Activation-keys**を参照してください。

4.6.1.4. クライアントでGPGキーを信頼する

4.6.1.5. クライアントでGPGキーを信頼する

Operating systems either trust their own GPG keys directly or at least ship them installed with the minimal system. But third party packages signed by a different GPG key need manual handling. The clients can be successfully bootstrapped without the GPG key being trusted. However, you cannot install new client tool packages or update them until the keys are trusted.

Salt clients use now GPG key information entered for a software channel to manage the trusted keys. When a software channel with GPG key information is assigned to a client, the key gets trusted as soon as the channel is refreshed or the first package gets installed from this channel.

The GPG key URL which is set of a software channel must exist. In case it is a file URL, the GPG key file must be deployed on the client before the software channel is used.

The GPG keys for the Client Tools Channels of Red Hat based clients are deployed on the client into `/etc/pki/rpm-gpg/` and can be referenced with file URLs. Same is the case with the GPG keys of Expanded Support clients. Only in case a software channel is assigned to the client they will be imported and trusted by the system.



Because Debian based systems sign only metadata, there is typically no need to specify extra keys for single channels. If a user configures an own GPG key to sign the metadata as described in "Use Your Own GPG Key" in **Administration > Repo-metadata** the deployment and trust of that key is executed automatically.

4.6.1.5.1. User defined GPG keys

Users can define their own GPG keys to be deployed to the client.

By providing some pillar data and providing the GPG key files in the Salt filesystem, they are automatically deployed to the client.

These keys are deployed into `/etc/pki/rpm-gpg/` on RPM based operating systems and to `/usr/share/keyrings/` on Debian systems:

Define the pillar key [literal `custom_gpgkeys`] for the client you want to deploy the key to and list the names of the key file.

```
cat /etc/pillar/mypillar.sls
custom_gpgkeys:
  - my_first_gpg.key
  - my_second_gpgkey.gpg
```

Additionally in the Salt filesystem create a directory named `gpg` and store there the GPG key files with the name specified in the `custom_gpgkeys` pillar data.

```
ls -la /etc/salt/gpg/
/etc/salt/gpg/my_first_gpg.key
/etc/salt/gpg/my_second_gpgkey.gpg
```

The keys are now deployed to the client at `/etc/pki/rpm-gpg/my_first_gpg.key` and `/etc/pki/rpm-gpg/my_second_gpgkey.gpg`.

The last step is to add the URL to the GPG key URL field of the software channel. Navigate to **Software > Manage > Channels** and select the channel you want to modify. Add to **GPG key URL** the value `file:///etc/pki/rpm-gpg/ my_first_gpg.key`.

4.6.1.5.2. GPG Keys in Bootstrap Scripts

プロシージャ: ブートストラップスクリプトを使用してクライアントでGPGキーを信頼する

1. Uyuniサーバのコマンドプロンプトで、`/srv/www/htdocs/pub/` ディレクトリの内容を確認します。このディレクトリには、使用できるすべての公開鍵が含まれています。登録クライアントに割り当てるチャンネルに適用するキーをメモします。
2. 関連するブートストラップスクリプトを開き、`ORG_GPG_KEY=` パラメータを見つけて、必要なキーを追加します。次に例を示します。

```
uyuni-gpg-pubkey-0d20833e.key
```

以前保存したキーを削除する必要はありません。



Trusting a GPG key is important for security on clients. It is the task of the admin to decide which keys are needed and can be trusted. Because a software channel cannot be used when the GPG key is not trusted, the decision of assigning a channel to a client depends on the decision of trusting the key.

4.6.1.6. クライアントの登録

Amazon Linuxクライアントは、その他すべてのクライアントと同じ方法で登録されます。 詳細については、[Client-configuration > Registration-overview](#)を参照してください。

4.7. CentOSクライアントの登録

CentOSクライアントをUyuniサーバに登録できます。 そのメソッドおよび詳細は、クライアントのオペレーティングシステムによって異なります。

始める前に、クライアントでUyuniサーバと日時が正しく同期していることを確認してください。

アクティベーションキーを作成済みである必要があります。 アクティベーションキーの作成の詳細については、[Client-configuration > Activation-keys](#)を参照してください。

4.7.1. CentOSクライアントの登録

このセクションでは、CentOSオペレーティングシステムを実行している従来のクライアントおよびSaltクライアントの登録について説明します。



CentOSクライアントは、CentOSに基づいていて、SUSE Linux Enterprise Server with Expanded Support、RES、Red Hat、またはExpanded Supportとは関係がありません。 CentOSベースメディアアリポジトリとCentOSインストールメディアへのアクセス管理、およびUyuniサーバのCentOSコンテンツデリバリネットワークへの接続は、ユーザが行います。



従来のクライアントはCentOS 8では使用できません。はSaltクライアントとしてのみサポートされます。

CentOS 8クライアント



CentOSクライアントのUyuniへの登録は、ターゲットポリシーで適用されるデフォルトのSELinux設定でテストされます。SELinuxを無効にしてCentOSクライアントをUyuniに登録する必要があります。

4.7.1.1. ソフトウェアチャンネルの追加

CentOSクライアントをUyuniサーバに登録する前に、必要なソフトウェアチャンネルを追加して同期する必要があります。

現在サポートされているアーキテクチャは、「x86_64」と「aarch64」です。サポートされている製品およびアーキテクチャの完全な一覧については、[Client-configuration > Supported-features](#)を参照してください。



In the following section, descriptions often default to the `x86_64` architecture. Replace it with other architectures if appropriate.

このプロシージャで必要なチャンネルは次のとおりです。

表 26. CentOSチャンネル - CLI

OSバージョン	ベースチャンネル	クライアントチャンネル	更新/Appstreamチャンネル
CentOS 6	centos6	centos6-uyuni-client	centos6-updates
CentOS 7	centos7	centos7-uyuni-client	centos7-updates
CentOS 8	centos8	centos8-uyuni-client	centos8-appstream



CentOS 6はサポート終了になっており、そのリポジトリで提供されるISOイメージは失効しています。これらのパッケージを使用した新しいCentOS 6クライアントのブートストラップは失敗します。新しいCentOS 6クライアントをブートストラップする必要がある場合、[Client-configuration > Tshoot-clients](#)のトラブルシューティングプロシージャに従ってください。

手順: コマンドプロンプトからのソフトウェアチャンネルの追加

1. Uyuni サーバのコマンドプロンプトで root になり、`spacewalk-common-channels` コマンドを特定のチャンネルに対して実行します。このとき、正しいアーキテクチャを指定してください:

```
spacewalk-common-channels \
-a <アーキテクチャ> \
<ベースチャンネル名> \
<子チャンネル_1> \
<子チャンネル_2> \
... <子チャンネル_n>
```

2. チャンネルの同期:

```
spacewalk-repo-sync -p <base_channel_label>
```

3. 続行前に、同期が完了していることを確認してください。



spacewalk-common-channels によって提供されるクライアントツールのチャンネルの提供元はUyuniです。SUSEではありません。



CentOS 8クライアントでは、ベースチャンネルとAppStreamチャンネルの両方を追加します。両方のチャンネルのパッケージが必要です。両方のチャンネルを追加しないと、パッケージ不足のためブートストラップリポジトリを作成できません。

モジューラーチャンネルを使用している場合は、クライアントでPython3.6モジュールストリームを有効にする必要があります。Python 3.6を提供しない場合、**spacecmd**パッケージのインストールは失敗します。



上流のチャンネルとUyuniチャンネルの間のAppStreamチャンネルで利用できるパッケージ数に不一致が発生する場合があります。また、同時に別の場所で追加したチャンネルを比較すると、数値が異なる場合もあります。CentOSでリポジトリを管理する方法が原因です。CentOSでは新しいバージョンがリリースされると古いバージョンのパッケージが削除されますが、Uyuniでは経過年数に関係なくすべてのバージョンが保持されます。

AppStreamリポジトリにはモジュールパッケージが用意されています。UyuniのWeb UIに正しくないパッケージ情報が表示されます。Web UIまたはAPIを使用してモジュールリポジトリから直接インストールまたはアップグレードするようなパッケージ操作は実行できません。



コンテンツライフサイクル管理(CLM)でAppStreamフィルタを使用して、モジュールリポジトリを通常のリポジトリに変換できます。クライアントで `spacecmd` を使用する場合は、AppStreamフィルタを使用して `python:3.6` を必ず含めてください。

または、Salt状態を使用してSaltクライアントでモジュラーパッケージを管理したり、クライアントで `dnf` コマンドを使用することもできます。CLMの詳細については、[Administration > Content-lifecycle](#)を参照してください。

4.7.1.2. 同期ステータスの確認

プロシージャ: Web UIから同期の進捗状況を確認する

1. UyuniのWeb UIで、[ソフトウェア > 管理 > チャンネル](#)に移動し、リポジトリに関連付けられているチャンネルをクリックします。
2. [リポジトリ] タブに移動し、[同期] をクリックし、[同期状態] をクリックします。

プロシージャ: コマンドプロンプトから同期の進捗状況を確認する

1. Uyuniサーバのコマンドプロンプトで、rootとして、`tail` コマンドを使用して同期ログファイルを確認します。

```
tail -f /var/log/rhn/reposync/<channel-label>.log
```

2. それぞれの子チャンネルは、同期の進捗中にそれぞれのログを生成します。同期が完了したことを確認するには、ベースチャンネルと子チャンネルのログファイルをすべて確認する必要があります。

4.7.1.3. アクティベーションキーの作成

CentOSチャンネルと関連付けられているアクティベーションキーを作成する必要があります。

アクティベーションキーの詳細については、[Client-configuration > Activation-keys](#)を参照してください。

4.7.1.4. クライアントでGPGキーを信頼する

4.7.1.5. クライアントでGPGキーを信頼する

Operating systems either trust their own GPG keys directly or at least ship them installed with the minimal system. But third party packages signed by a different GPG key need manual handling. The clients can be successfully bootstrapped without the GPG key being trusted. However, you cannot install new client tool packages or update them until the keys are trusted.

Salt clients use now GPG key information entered for a software channel to manage the trusted keys. When a software channel with GPG key information is assigned to a client, the key gets trusted as soon as the channel is refreshed or the first package gets installed from this channel.

The GPG key URL which is set of a software channel must exist. In case it is a file URL, the GPG key file must be deployed on the client before the software channel is used.

The GPG keys for the Client Tools Channels of Red Hat based clients are deployed on the client into `/etc/pki/rpm-gpg/` and can be referenced with file URLs. Same is the case with the GPG keys of Expanded Support clients. Only in case a software channel is assigned to the client they will be imported and trusted by the system.



Because Debian based systems sign only metadata, there is typically no need to specify extra keys for single channels. If a user configures an own GPG key to sign the metadata as described in "Use Your Own GPG Key" in **Administration > Repo-metadata** the deployment and trust of that key is executed automatically.

4.7.1.5.1. User defined GPG keys

Users can define their own GPG keys to be deployed to the client.

By providing some pillar data and providing the GPG key files in the Salt filesystem, they are automatically deployed to the client.

These keys are deployed into `/etc/pki/rpm-gpg/` on RPM based operating systems and to `/usr/share/keyrings/` on Debian systems:

Define the pillar key [literal `custom_gpgkeys`] for the client you want to deploy the key to and list the names of the key file.

```
cat /etc/pillar/mypillar.sls
custom_gpgkeys:
  - my_first_gpg.key
  - my_second_gpgkey.gpg
```

Additionally in the Salt filesystem create a directory named `gpg` and store there the GPG key files with the name specified in the `custom_gpgkeys` pillar data.

```
ls -la /etc/salt/gpg/
/etc/salt/gpg/my_first_gpg.key
/etc/salt/gpg/my_second_gpgkey.gpg
```

The keys are now deployed to the client at `/etc/pki/rpm-gpg/my_first_gpg.key` and `/etc/pki/rpm-gpg/my_second_gpgkey.gpg`.

The last step is to add the URL to the GPG key URL field of the software channel. Navigate to [Software > Manage > Channels](#) and select the channel you want to modify. Add to **GPG key URL** the value `file:///etc/pki/rpm-gpg/my_first_gpg.key`.

4.7.1.5.2. GPG Keys in Bootstrap Scripts

プロシージャ: ブートストラップスクリプトを使用してクライアントでGPGキーを信頼する

1. Uyuniサーバのコマンドプロンプトで、`/srv/www/htdocs/pub/`ディレクトリの内容を確認します。このディレクトリには、使用できるすべての公開鍵が含まれています。登録クライアントに割り当てるチャンネルに適用するキーをメモします。
2. 関連するブートストラップスクリプトを開き、`ORG_GPG_KEY=`パラメータを見つけて、必要なキーを追加します。次に例を示します。

```
uyuni-gpg-pubkey-0d20833e.key
```

以前保存したキーを削除する必要はありません。



Trusting a GPG key is important for security on clients. It is the task of the admin to decide which keys are needed and can be trusted. Because a software channel cannot be used when the GPG key is not trusted, the decision of assigning a channel to a client depends on the decision of trusting the key.

4.7.1.6. クライアントの登録

CentOSクライアントは、その他すべてのクライアントと同じ方法で登録されます。
詳細については、[Client-configuration > Registration-overview](#)を参照してください。



CentOS 6クライアントを登録して使用するには、Uyuniサーバを設定して旧式のSSL暗号化をサポートする必要があります。このエラーを解決する方法については、[古いクライアントの登録](#)を[Client-configuration > Overview](#)で参照してください。

4.7.1.7. エラータの管理

CentOSクライアントを更新するとき、パッケージには更新に関するメタデータは含まれていません。サードパーティのエラータサービスを使用してこの情報を取得できます。



CEFSの作成者は、パッチまたはエラータを、利便性向上を目指して努力ベースで提供していますが、これが正確であることや最新であることを保証していません。つまり、パッチ日が正しくない場合があります。また、発行されたデータが1カ月以上遅れて示されたことが少なくとも1回ありました。このような場合の情報については、<https://github.com/stevemeier/cefs/issues/28#issuecomment-656579382>および<https://github.com/stevemeier/cefs/issues/28#issuecomment-656573607>を参照してください。

パッチデータに問題または遅れがあると、信頼できないパッチ情報がUyuniサーバにインポートされる場合があります。その結果、レポート、監査、CVEの更新、またはその他のパッチ関連の情報も誤りになります。セキュリティ関連の要件や証明書の条件に応じて、パッチデータを独立して確認する方法や、異なるオペレーティングシステムを選択する方法など、このサービスを使用する方法の代替方法を検討してください。

プロシージャ: エラータサービスのインストール

1. Uyuniサーバでコマンドプロンプトからrootとして **sle-module-development-tools** モジュールを追加します。

```
SUSEConnect --product sle-module-development-tools/15.2/x86_64
```

2. エラータサービスの依存関係をインストールします。

```
zypper in perl-Text-Unidecode
```

3. **/etc/rhn/rhn.conf** で次の行を追加または編集します。

```
java.allow_adding_patches_via_api = centos7-updates-x86_64,centos7-x86_64,centos7-extras-x86_64
```

4. Tomcatを再起動します。

```
systemctl restart tomcat
```

5. エラータスクリプト用のファイルを作成します。

```
touch /usr/local/bin/cent-errata.sh
```

6. 新しいファイルを編集してこのスクリプトを含め、必要に応じてリポジトリの詳細を編集します。このスクリプトは、外部のエラータサービスからエラータの詳細をフェッチして展開し、詳細を発行します。

```

#!/bin/bash
mkdir -p /usr/local/centos
cd /usr/local/centos
rm *.xml
wget -c http://cefs.steve-meier.de/errata.latest.xml
#wget -c https://www.redhat.com/security/data/oval/com.redhat.rhsa-all.xml
wget -c https://www.redhat.com/security/data/oval/com.redhat.rhsa-RHEL7.xml.bz2
bzip2 -d com.redhat.rhsa-RHEL7.xml.bz2
wget -c http://cefs.steve-meier.de/errata-import.tar
tar xvf errata-import.tar
chmod +x /usr/local/centos/errata-import.pl
export SPACEWALK_USER='<adminname>'; export SPACEWALK_PASS='<password>'
/usr/local/centos/errata-import.pl --server '<servername>' \
--errata /usr/local/centos/errata.latest.xml \
--include-channels=centos7-updates-x86_64,centos7-x86_64,centos7-extras-x86_64 \
--publish --rhsa-oval /usr/local/centos/com.redhat.rhsa-RHEL7.xml

```

7. スクリプトを毎日実行するようcronジョブを設定します。

```
ln -s /usr/local/bin/cent-errata.sh /etc/cron.daily
```

このツールの詳細については、<https://cefs.steve-meier.de/>を参照してください。

4.8. Debianクライアントの登録

DebianクライアントをUyuniサーバに登録できます。 そのメソッドおよび詳細は、クライアントのオペレーティングシステムによって異なります。

始める前に、クライアントでUyuniサーバと日時が正しく同期していることを確認してください。

アクティベーションキーを作成済みである必要があります。 アクティベーションキーの作成の詳細については、[Client-configuration > Activation-keys](#)を参照してください。



Uyuniサーバをこのサーバ自身に登録しないでください。 Uyuniサーバは個別に管理する必要があります。

4.8.1. Debianクライアントの登録

このセクションでは、Debianオペレーティングシステムを実行しているSaltクライアントの登録について説

明します。



SUSEはDebianオペレーティングシステムをサポートしていません。 Uyuniを使用すると、Debianクライアントを管理できますが、サポートは提供されません。 Uyuniを使用してDebianクライアントを管理する方法は試験的な方法です。 この手順は、Debian 9 StretchおよびDebian 10 Busterでテストされています。稼動環境ではDebianクライアントを使用しないでください。



DebianはSaltクライアントでのみサポートされています。 従来のクライアントはサポートされていません。

ブートストラップは、初期状態の実行およびプロファイルの更新のためにDebianクライアントで使用できます。

4.8.1.1. 登録の準備

DebianクライアントをUyuniサーバに登録するには、その前に準備が必要です。

- Debian 9を使用している場合、必要なパッケージをクライアントにインストールしてから登録操作に進みます。クライアントのコマンドプロンプトでrootとして次のコマンドを実行します。

```
apt install apt-transport-https python-apt python3-apt
```

- DNSが正しく設定されていることを確認し、クライアントのエントリを提供します。または、適切なエントリを使用して、Uyuniサーバとクライアントの両方で `/etc/hosts` ファイルを設定できます。
- クライアントは、登録する前にUyuniサーバと日時が同期されている必要があります。

4.8.1.2. ソフトウェアチャンネルの追加

DebianクライアントをUyuniサーバに登録する前に、必要なソフトウェアチャンネルを追加して同期する必要があります。



In the following section, descriptions often default to the `x86_64` architecture. Replace it with other architectures if appropriate.

このプロシージャで必要なチャンネルは次のとおりです。

表 27. Debianチャンネル - CLI

OSバージョン	ベースチャンネル	クライアントチャンネル	更新チャンネル	セキュリティチャンネル
Debian 9	debian-9-pool-amd64-uyuni	debian-9-amd64-uyuni-client	debian-9-amd64-main-updates-uyuni	debian-9-amd64-main-security-uyuni

OSバージョン	ベースチャンネル	クライアントチャンネル	更新チャンネル	セキュリティチャンネル
Debian 10	debian-10-pool-amd64-uyuni	debian-10-amd64-uyuni-client	debian-10-amd64-main-updates-uyuni	debian-10-amd64-main-security-uyuni
Debian 11	debian-11-pool-amd64-uyuni	debian-11-amd64-uyuni-client	debian-11-amd64-main-updates-uyuni	debian-11-amd64-main-security-uyuni

手順: コマンドプロンプトからのソフトウェアチャンネルの追加

- Uyuni サーバのコマンドプロンプトで root になり、 `spacewalk-common-channels` コマンドを特定のチャンネルに対して実行します:

```
spacewalk-common-channels \
<ベースチャンネルのラベル> \
<子チャンネル_1> \
<子チャンネル_2> \
... <子チャンネル_n>
```

- チャンネルの同期:

```
spacewalk-repo-sync -p <base_channel_label>
```

- 続行前に、同期が完了していることを確認してください。

4.8.1.3. 同期ステータスの確認

プロシージャ: Web UIから同期の進捗状況を確認する

- UyuniのWeb UIで、**ソフトウェア** > **管理** > **チャンネル**に移動し、リポジトリに関連付けられているチャンネルをクリックします。
- [リポジトリ] タブに移動し、[同期] をクリックし、[同期状態] をクリックします。

プロシージャ: コマンドプロンプトから同期の進捗状況を確認する

- Uyuniサーバのコマンドプロンプトで、rootとして、`tail` コマンドを使用して同期ログファイルを確認します。

```
tail -f /var/log/rhn/reposync/<channel-label>.log
```

- それぞれの子チャンネルは、同期の進捗中にそれぞれのログを生成します。同期が完了したことを確認するには、ベースチャンネルと子チャンネルのログファイルをすべて確認する必要があります。



Debianチャンネルは非常に大きいことがあります。 同期に数時間かかる場合があります。

4.8.1.4. クライアントでGPGキーを信頼する

4.8.1.5. クライアントでGPGキーを信頼する

Operating systems either trust their own GPG keys directly or at least ship them installed with the minimal system. But third party packages signed by a different GPG key need manual handling. The clients can be successfully bootstrapped without the GPG key being trusted. However, you cannot install new client tool packages or update them until the keys are trusted.

Salt clients use now GPG key information entered for a software channel to manage the trusted keys. When a software channel with GPG key information is assigned to a client, the key gets trusted as soon as the channel is refreshed or the first package gets installed from this channel.

The GPG key URL which is set of a software channel must exist. In case it is a file URL, the GPG key file must be deployed on the client before the software channel is used.

The GPG keys for the Client Tools Channels of Red Hat based clients are deployed on the client into `/etc/pki/rpm-gpg/` and can be referenced with file URLs. Same is the case with the GPG keys of Expanded Support clients. Only in case a software channel is assigned to the client they will be imported and trusted by the system.



Because Debian based systems sign only metadata, there is typically no need to specify extra keys for single channels. If a user configures an own GPG key to sign the metadata as described in "Use Your Own GPG Key" in **Administration > Repo-metadata** the deployment and trust of that key is executed automatically.

4.8.1.5.1. User defined GPG keys

Users can define their own GPG keys to be deployed to the client.

By providing some pillar data and providing the GPG key files in the Salt filesystem, they are automatically deployed to the client.

These keys are deployed into `/etc/pki/rpm-gpg/` on RPM based operating systems and to `/usr/share/keyrings/` on Debian systems:

Define the pillar key [literal `custom_gpgkeys`] for the client you want to deploy the key to and list the names of the key file.

```
cat /etc/pillar/mypillar.sls
custom_gpgkeys:
  - my_first_gpg.key
  - my_second_gpgkey.gpg
```

Additionally in the Salt filesystem create a directory named `gpg` and store there the GPG key files with the name specified in the `custom_gpgkeys` pillar data.

```
ls -la /etc/salt/gpg/
/etc/salt/gpg/my_first_gpg.key
/etc/salt/gpg/my_second_gpgkey.gpg
```

The keys are now deployed to the client at `/etc/pki/rpm-gpg/my_first_gpg.key` and `/etc/pki/rpm-gpg/my_second_gpgkey.gpg`.

The last step is to add the URL to the GPG key URL field of the software channel. Navigate to **Software > Manage > Channels** and select the channel you want to modify. Add to **GPG key URL** the value `file:///etc/pki/rpm-gpg/my_first_gpg.key`.

4.8.1.5.2. GPG Keys in Bootstrap Scripts

プロシージャ: ブートストラップスクリプトを使用してクライアントでGPGキーを信頼する

1. Uyuniサーバのコマンドプロンプトで、`/srv/www/htdocs/pub/`ディレクトリの内容を確認します。このディレクトリには、使用できるすべての公開鍵が含まれています。登録クライアントに割り当てるチャンネルに適用するキーをメモします。
2. 関連するブートストラップスクリプトを開き、`ORG_GPG_KEY=`パラメータを見つけて、必要なキーを追加します。次に例を示します。

```
uyuni-gpg-pubkey-0d20833e.key
```

以前保存したキーを削除する必要はありません。



Trusting a GPG key is important for security on clients. It is the task of the admin to decide which keys are needed and can be trusted. Because a software channel cannot be used when the GPG key is not trusted, the decision of assigning a channel to a client depends on the decision of trusting the key.



Debianクライアントをインストールするには、複数のGPGキーが必要な場合があります。

4.8.1.6. クライアントの登録

Debianクライアントを登録するには、ブートストラップリポジトリが必要です。デフォルトでは、ブートストラップリポジトリは毎日再生成されます。次のコマンドを使用して、コマンドプロンプトからブートストラップリポジトリを手動で作成できます。

```
mgr-create-bootstrap-repo
```

Debian 10で、プロンプトが表示されたら **debian10-amd64-uyuni** を選択します。

クライアントの登録については、[Client-configuration > Registration-overview](#)を参照してください。

4.9. Oracleクライアントの登録

Oracle LinuxクライアントをUyuniサーバに登録できます。 そのメソッドおよび詳細は、クライアントのオペレーティングシステムによって異なります。

始める前に、クライアントでUyuniサーバと日時が正しく同期していることを確認してください。

アクティベーションキーを作成済みである必要もあります。 アクティベーションキーの作成の詳細については、[Client-configuration > Activation-keys](#)を参照してください。

4.9.1. Oracle Linuxクライアントの登録

このセクションでは、Oracle Linuxオペレーティングシステムを実行している従来のクライアントおよびSaltクライアントの登録について説明します。



従来のクライアントはOracle Linux 8では使用できません。 Oracle Linux 8クライアントはSaltクライアントとしてのみサポートされます。

4.9.1.1. ソフトウェアチャンネルの追加

Oracle LinuxクライアントをUyuniサーバに登録する前に、必要なソフトウェアチャンネルを追加して同期する必要があります。

現在サポートされているアーキテクチャは、「**x86_64**」と「**aarch64**」です。 サポートされている製品およびアーキテクチャの完全な一覧については、[Client-configuration > Supported-features](#)を参照してください。



In the following section, descriptions often default to the **x86_64** architecture. Replace it with other architectures if appropriate.

このプロシージャで必要なチャンネルは次のとおりです。

表 28. Oracleチャンネル - CLI

OSバージョン	ベースチャンネル	クライアントチャンネル	チャンネルの更新
Oracle Linux 6	oraclelinux6	oraclelinux6-uyuni-client	-
Oracle Linux 7	oraclelinux7	oraclelinux7-uyuni-client	-
Oracle Linux 8	oraclelinux8	oraclelinux8-uyuni-client	oraclelinux8-appstream



Oracle Linux 6はサポート終了になっており、そのリポジトリで提供されるISOイメージは失効しています。これらのパッケージを使用した新しいOracle Linux 6クライアントのブートストラップは失敗します。新しいOracle Linux 6クライアントをブートストラップする必要がある場合、[Client-configuration > Tshoot-clients](#) のトラブルシューティングプロシージャに従ってください。

手順: コマンドプロンプトからのソフトウェアチャンネルの追加

1. Uyuni サーバのコマンドプロンプトで root になり、**spacewalk-common-channels** コマンドを特定のチャンネルに対して実行します:

```
spacewalk-common-channels \
<ベースチャンネルのラベル> \
<子チャンネル_1> \
<子チャンネル_2> \
... <子チャンネル_n>
```

2. チャンネルの同期:

```
spacewalk-repo-sync -p <base_channel_label>
```

3. 続行前に、同期が完了していることを確認してください。



spacewalk-common-channels によって提供されるクライアントツールのチャンネルの提供元はUyuniです。SUSEではありません。



Oracle Linux 8クライアントでは、ベースチャンネルとAppStreamチャンネルの両方を追加します。両方のチャンネルのパッケージが必要です。両方のチャンネルを追加しないと、パッケージ不足のためブートストラップリポジトリを作成できません。

モジュラーチャンネルを使用している場合は、クライアントでPython3.6モジュールストリームを有効にする必要があります。Python 3.6を提供しない場合、**spacecmd** パッケージのインストールは失敗します。

AppStreamリポジトリにはモジュールパッケージが用意されています。UyuniのWeb UIに正しくないパッケージ情報が表示されます。Web UIまたはAPIを使用してモジュールリポジトリから直接インストールまたはアップグレードするようなパッケージ操作は実行できません。



コンテンツライフサイクル管理(CLM)でAppStreamフィルタを使用して、モジュールリポジトリを通常のリポジトリに変換できます。クライアントで `spacecmd` を使用する場合は、AppStreamフィルタを使用して `python:3.6` を必ず含めてください。

または、Salt状態を使用してSaltクライアントでモジュラーパッケージを管理したり、クライアントで `dnf` コマンドを使用することもできます。CLMの詳細については、[Administration > Content-lifecycle](#)を参照してください。

4.9.1.2. 同期ステータスの確認

プロシージャ: Web UIから同期の進捗状況を確認する

1. UyuniのWeb UIで、[ソフトウェア > 管理 > チャンネル](#)に移動し、リポジトリに関連付けられているチャンネルをクリックします。
2. [リポジトリ] タブに移動し、[同期] をクリックし、[同期状態] をクリックします。

プロシージャ: コマンドプロンプトから同期の進捗状況を確認する

1. Uyuniサーバのコマンドプロンプトで、rootとして、`tail` コマンドを使用して同期ログファイルを確認します。

```
tail -f /var/log/rhn/reposync/<channel-label>.log
```

2. それぞれの子チャンネルは、同期の進捗中にそれぞれのログを生成します。同期が完了したことを確認するには、ベースチャンネルと子チャンネルのログファイルをすべて確認する必要があります。

4.9.1.3. アクティベーションキーの作成

Oracle Linuxチャンネルと関連付けられているアクティベーションキーを作成する必要があります。

アクティベーションキーの詳細については、[Client-configuration > Activation-keys](#)を参照してください。

4.9.1.4. クライアントでGPGキーを信頼する

4.9.1.5. クライアントでGPGキーを信頼する

Operating systems either trust their own GPG keys directly or at least ship them installed with the minimal system. But third party packages signed by a different GPG key need manual handling. The clients can be successfully bootstrapped without the GPG key being trusted. However, you cannot install new client tool packages or update them until the keys are trusted.

Salt clients use now GPG key information entered for a software channel to manage the trusted keys. When a software channel with GPG key information is assigned to a client, the key gets trusted as soon as the channel is refreshed or the first package gets installed from this channel.

The GPG key URL which is set of a software channel must exist. In case it is a file URL, the GPG key file must be deployed on the client before the software channel is used.

The GPG keys for the Client Tools Channels of Red Hat based clients are deployed on the client into `/etc/pki/rpm-gpg/` and can be referenced with file URLs. Same is the case with the GPG keys of Expanded Support clients. Only in case a software channel is assigned to the client they will be imported and trusted by the system.



Because Debian based systems sign only metadata, there is typically no need to specify extra keys for single channels. If a user configures an own GPG key to sign the metadata as described in "Use Your Own GPG Key" in **Administration > Repo-metadata** the deployment and trust of that key is executed automatically.

4.9.1.5.1. User defined GPG keys

Users can define their own GPG keys to be deployed to the client.

By providing some pillar data and providing the GPG key files in the Salt filesystem, they are automatically deployed to the client.

These keys are deployed into `/etc/pki/rpm-gpg/` on RPM based operating systems and to `/usr/share/keyrings/` on Debian systems:

Define the pillar key [literal `custom_gpgkeys`] for the client you want to deploy the key to and list the names of the key file.

```
cat /etc/pillar/mypillar.sls
custom_gpgkeys:
  - my_first_gpg.key
  - my_second_gpgkey.gpg
```

Additionally in the Salt filesystem create a directory named `gpg` and store there the GPG key files with the name specified in the `custom_gpgkeys` pillar data.

```
ls -la /etc/salt/gpg/
/etc/salt/gpg/my_first_gpg.key
/etc/salt/gpg/my_second_gpgkey.gpg
```

The keys are now deployed to the client at `/etc/pki/rpm-gpg/my_first_gpg.key` and `/etc/pki/rpm-gpg/my_second_gpgkey.gpg`.

The last step is to add the URL to the GPG key URL field of the software channel. Navigate to **Software > Manage > Channels** and select the channel you want to modify. Add to **GPG key URL** the value `file:///etc/pki/rpm-gpg/my_first_gpg.key`.

4.9.1.5.2. GPG Keys in Bootstrap Scripts

プロシージャ: ブートストラップスクリプトを使用してクライアントでGPGキーを信頼する

1. Uyuniサーバのコマンドプロンプトで、`/srv/www/htdocs/pub`ディレクトリの内容を確認します。このディレクトリには、使用できるすべての公開鍵が含まれています。登録クライアントに割り当てるチャンネルに適用するキーをメモします。
2. 関連するブートストラップスクリプトを開き、`ORG_GPG_KEY=`パラメータを見つけて、必要なキーを追加します。次に例を示します。

```
uyuni-gpg-pubkey-0d20833e.key
```

以前保存したキーを削除する必要はありません。



Trusting a GPG key is important for security on clients. It is the task of the admin to decide which keys are needed and can be trusted. Because a software channel cannot be used when the GPG key is not trusted, the decision of assigning a channel to a client depends on the decision of trusting the key.



Oracle 8クライアントの場合、以下を使用します

```
o18-gpg-pubkey-82562EA9AD986DA3.key
```

Oracle 6または7クライアントの場合、以下を使用します

```
o167-gpg-pubkey-72F97B74EC551F0A3.key
```

4.9.1.6. クライアントの登録

Oracle Linuxクライアントは、その他すべてのクライアントと同じ方法で登録されます。 詳細については、**Client-configuration > Registration-overview**を参照してください。



Oracle Linux 6クライアントを登録して使用するには、Uyuniサーバを設定して旧式のSSL暗号化をサポートする必要があります。このエラーを解決する方法については、**古いクライアントの登録をClient-configuration > Registration-overview**参照してください。

4.10. Red Hatクライアントの登録

Red Hatコンテンツデリバリネットワーク(CDN)またはRed Hat更新インフラストラクチャ(RHUI)を使用してRed Hat Enterprise LinuxクライアントをUyuniサーバに登録できます。そのメソッドおよび詳細は、クライアントのオペレーティングシステムによって異なります。

始める前に、クライアントでUyuniサーバと日時が正しく同期していることを確認してください。

アクティベーションキーを作成済みである必要もあります。アクティベーションキーの作成の詳細については、[Client-configuration > Activation-keys](#)を参照してください。

4.10.1. CDNでRed Hat Enterprise Linuxクライアントを登録する

SUSE Linux Enterprise Server with Expanded Supportを使用するのではなく、Red Hat Enterprise Linuxクライアントを直接実行している場合、Red Hatソースを使用してパッケージを取得および更新する必要があります。このセクションでは、Red Hatコンテンツデリバリネットワーク(CDN)を使用して、Red Hat Enterprise Linuxオペレーティングシステムを実行している従来のクライアントおよびSaltクライアントを登録する方法について説明します。

代わりにRed Hat更新インフラストラクチャ(RHUI)を使用する方法については、[Client-configuration > Clients-rh-rhui](#)を参照してください。



Red Hat Enterprise Linuxクライアントは、Red Hatに基づいていて、SUSE Linux Enterprise Server with Expanded Support、RES、またはSUSE Linux Enterprise Serverとは関係ありません。Red HatベースメディアアリポジトリとRHELインストールメディアへのアクセス管理、およびUyuniサーバのRed Hatコンテンツデリバリネットワークへの接続は、ユーザが行います。使用しているすべてのRHELシステムに対してRed Hatのサポートを取得する必要があります。これを実行しないと、Red Hatの条項に違反となる場合があります。



従来のクライアントはRed Hat Enterprise Linux 6および7でのみサポートされています。Red Hat Enterprise Linux 8クライアントはSaltクライアントとしてサポートされています。

4.10.1.1. エンタイトルメントと証明書のインポート

Red Hatクライアントには、Red Hat認証局(CA)、エンタイトルメント証明書、およびエンタイトルメントキーが必要です。

エンタイトルメント証明書には、有効期限が埋め込まれていて、この期限はサポートサブスクリプションの期間と一致しています。中断を回避するには、サポートサブスクリプション期間の終わりのたびにこのプロセスを繰り返す必要があります。

Red Hatには、サブスクリプション割り当てを管理するためのサブスクリプションマネージャツールが用意されています。このツールはローカルに実行され、インストール済みの製品およびサブスクリプションを追跡します。クライアントは、サブスクリプションマネージャで登録して証明書を取得する必要があります。

Red Hatクライアントは、URLを使用してリポジトリを複製します。 URLは、Red Hatクライアントを登録した場所に応じて変わります。

Red Hatクライアントは次の3種類の方法で登録できます。

- ・ redhat.comにあるRed Hatコンテンツデリバリネットワーク(CDN)
- ・ Red Hatサテライトサーバ
- ・ クラウドのRed Hat更新インフラストラクチャ(RHUI)

このガイドでは、Red Hat CDNに登録されるクライアントについて説明します。 リポジトリコンテンツの認可済みサブスクリプションを使用して、1つ以上のシステムがCDNに登録されている必要があります。

代わりにRed Hat更新インフラストラクチャ(RHUI)を使用する方法については、[Client-configuration > Clients-rh-rhui](#)を参照してください。



クライアントシステムのサテライト証明書では、サテライトサーバおよびサブスクリプションが必要です。 サテライト証明書を使用するクライアントはUyuniサーバではサポートされていません。



エンタitleメント証明書には、有効期限が埋め込まれていて、この期限はサポートサブスクリプションの期間と一致しています。 中断を回避するには、サポートサブスクリプション期間の終わりのたびにこのプロセスを繰り返す必要があります。

Red Hatには、サブスクリプション割り当てを管理するためのサブスクリプションマネージャツールが用意されています。 このツールはクライアントシステムでローカルに実行され、インストール済みの製品およびサブスクリプションを追跡します。 サブスクリプションマネージャを使用してredhat.comを登録し、このプロシージャに従って証明書を取得します。

プロシージャ: クライアントをサブスクリプションマネージャに登録する

1. クライアントシステムのコマンドプロンプトで、サブスクリプションマネージャツールを使用して登録します。

```
subscription-manager register
```

プロンプトが表示されたら、Red Hatポータルのユーザ名とパスワードを入力します。

2. コマンドを実行します。

```
subscription-manager activate
```

3. Uyuniサーバがアクセスできる場所にエンタitleメント証明書とキーをクライアントシステムからコピーします。

```
cp /etc/pki/entitlement/ <example>/entitlement
```



エンタイトルメント証明書とキーの両方ともファイル拡張子は **.pem** です。キーにはファイル名にも **key** が含まれています。

- Red Hat CA証明書ファイルをクライアントシステムから、エンタイトルメント証明書およびキーと同じWebの場所にコピーします。

```
cp /etc/rhsm/ca/redhat-uep.pem <example>/entitlement
```

Red Hatクライアントでリポジトリを管理するには、CAおよびエンタイトルメント証明書をUyuniサーバにインポートする必要があります。この操作を実行するには、インポートプロシージャを3回実行して、3つのエントリを作成する必要があります。エンタイトルメント証明書、エンタイトルメントキーおよびRed Hat証明書にそれぞれ1つずつです。

プロシージャ: 証明書をサーバにインポートする

- UyuniサーバのWeb UIで、**システム** > **自動インストール** > **GPGキーとSSLキー**に移動します。
- [格納されているキーまたは証明書の作成]** をクリックして、エンタイトルメント証明書データを設定します。
 - [説明] フィールドに **Entitlement-Cert-date** と入力します。
 - [タイプ] フィールドで、**SSL** を選択します。
 - [アップロードするファイルの選択] フィールドで、エンタイトルメント証明書を保存した場所をブラウズし、**.pem** 証明書ファイルを選択します。
- [キーの作成]** をクリックします。
- [格納されているキーまたは証明書の作成]** をクリックして、エンタイトルメントキー用データを設定します。
 - [説明] フィールドに **Entitlement-key-date** と入力します。
 - [タイプ] フィールドで、**SSL** を選択します。
 - [アップロードするファイルの選択] フィールドで、エンタイトルメントキーを保存した場所をブラウズし、**.pem** キーファイルを選択します。
- [キーの作成]** をクリックします。
- [格納されているキーまたは証明書用]** を次のパラメータを設定します。
 - [説明] フィールドに **redhat-uep** と入力します。
 - [タイプ] フィールドで、**SSL** を選択します。
 - [アップロードするファイルの選択] フィールドで、Red Hat証明書を保存した場所をブラウズし、証明書ファイルを選択します。

7. [キー] - [の] [作成] をクリックします。

4.10.1.2. ソフトウェアチャンネルの追加

Red HatクライアントをUyuniサーバに登録する前に、必要なソフトウェアチャンネルを追加して同期する必要があります。



In the following section, descriptions often default to the `x86_64` architecture. Replace it with other architectures if appropriate.

このプロシージャで必要なチャンネルは次のとおりです。

表 29. Red Hatチャンネル - CLI

OSバージョン	ベースチャンネル	クライアントチャンネル	ツールチャンネル
Red Hat 6	<code>rhel-x86_64-server-6</code>	-	<code>res6-suse-manager-tools-x86_64</code>
Red Hat 7	<code>rhel-x86_64-server-7</code>	-	<code>res7-suse-manager-tools-x86_64</code>
Red Hat 8	<code>rhel8-pool-x86_64</code>	-	<code>res8-suse-manager-tools-x86_64</code>



Red Hat 6はサポート終了になっており、そのリポジトリで提供されるISOイメージは失効しています。これらのパッケージを使用した新しいRed Hat 6クライアントのブートストラップは失敗します。新しいRed Hat 6クライアントをブートストラップする必要がある場合、[Client-configuration > Tshoot-clients](#)のトラブルシューティングプロシージャに従ってください。

手順: コマンドプロンプトからのソフトウェアチャンネルの追加

1. Uyuni サーバのコマンドプロンプトで `root` になり、`spacewalk-common-channels` コマンドを特定のチャンネルに対して実行します:

```
spacewalk-common-channels \
<ベースチャンネルのラベル> \
<子チャンネル_1> \
<子チャンネル_2> \
... <子チャンネル_n>
```

2. チャンネルの同期:

```
spacewalk-repo-sync -p <base_channel_label>
```

3. 続行前に、同期が完了していることを確認してください。



spacewalk-common-channelsによって提供されるクライアントツールのチャンネルの提供元はUyuniです。SUSEではありません。



AppStreamリポジトリにはモジュールパッケージが用意されています。UyuniのWeb UIに正しくないパッケージ情報が表示されます。Web UIまたはAPIを使用してモジュールリポジトリから直接インストールまたはアップグレードするようなパッケージ操作は実行できません。

コンテンツライフサイクル管理(CLM)でAppStreamフィルタを使用して、モジュールリポジトリを通常のリポジトリに変換できます。クライアントで **spacecmd** を使用する場合は、AppStreamフィルタを使用して **python:3.6** を必ず含めてください。

または、Salt状態を使用してSaltクライアントでモジュラーパッケージを管理したり、クライアントで **dnf** コマンドを使用することもできます。CLMの詳細については、[Administration > Content-lifecycle](#)を参照してください。

4.10.1.3. カスタムリポジトリおよびチャンネルの準備

Red Hat CDNからソフトウェアをミラーリングするには、URLでCDNにリンクされているカスタムチャンネルおよびリポジトリをUyuniに作成する必要があります。Red Hatポータルでこれらの製品を正しく動作させるには、該当製品のエンタイトルメントが必要です。サブスクリプションマネージャツールを使用して、ミラーリングするリポジトリのURLを取得できます。

```
subscription-manager repos
```

これらのリポジトリURLを使用して、カスタムリポジトリを作成できます。クライアントを管理するために必要なコンテンツのみミラーリングできます。



Red Hatポータルに正しいエンタイトルメントがある場合、Red Hatリポジトリのカスタムバージョンのみ作成できます。

このプロセッセージャに必要な詳細は次のとおりです。

表 30. Red Hatカスタムリポジトリ設定

オプション	設定
リポジトリURL	Red Hat CDNによって提供されるコンテンツURL
署名済みメタデータがあるかどうか	すべてのRed Hatエンタイトルメントリポジトリのチェックを外します
SSL CA証明書	redhat-uep
SSLクライアント証明書	Entitlement-Cert-date
SSLクライアントキー	Entitlement-Key-date

プロセージャ: カスタムリポジトリの作成

- UyuniサーバのWeb UIで、**ソフトウェア > 管理 > リポジトリ**に移動します。
- [リポジトリの作成]**をクリックし、リポジトリに適切なパラメータを設定します。
- [リポジトリの作成]**をクリックします。
- 作成する必要があるすべてのリポジトリで繰り返します。

このプロセージャで必要なチャンネルは次のとおりです。

表 31. Red Hatカスタムチャンネル

OSバージョン	ベース製品	ベースチャンネル
Red Hat 6	RHEL6 Base x86_64	rhel6-pool-x86_64
Red Hat 7	RHEL7 Base x86_64	rhel7-pool-x86_64
Red Hat 8	RHELまたはSLES ESまたはCentOS 8 Base	rhel8-pool-x86_64



Red Hat 6はサポート終了になっており、そのリポジトリで提供されるISOイメージは失効しています。これらのパッケージを使用した新しいRed Hat 6クライアントのブートストラップは失敗します。新しいRed Hat 6クライアントをブートストラップする必要がある場合、**Client-configuration > Tshoot-clients**のトラブルシューティングプロセージャに従ってください。

プロセージャ: カスタムチャンネルの作成

- UyuniサーバのWeb UIで、**ソフトウェア > 管理 > チャンネル**に移動します。
- [チャレンネルの作成]**をクリックし、チャンネルに適切なパラメータを設定します。
- [親チャンネル]**フィールドで、適切なベースチャンネルを選択します。
- [チャレンネルの作成]**をクリックします。
- 作成する必要があるすべてのチャンネルで繰り返します。各カスタムリポジトリに1つのカスタムチャンネルが必要です。

該当するすべてのチャンネルとリポジトリを作成したことを確認できます。そのためには、**ソフトウェア > チャンネル一覧 > すべて**に移動します。



Red Hat 8クライアントでは、ベースチャンネルとAppStreamチャンネルの両方を追加します。両方のチャンネルのパッケージが必要です。両方のチャンネルを追加しないと、パッケージ不足のためブートストラップリポジトリを作成できません。

モジュラーチャンネルを使用している場合は、クライアントでPython3.6モジュールストリームを有効にする必要があります。Python 3.6を提供しない場合、**spacecmd**パッケージのインストールは失敗します。

すべてのチャンネルを作成済みの場合、これらのチャンネルを、作成したリポジトリと関連付けできます。

プロシージャ: チャンネルのリポジトリとの関連付け

1. UyuniサーバのWeb UIで、**ソフトウェア > 管理 > チャンネル**に移動し、関連付けるチャンネルをクリックします。
2. [リポジトリ] タブに移動し、このチャンネルと関連付けるリポジトリにチェックを付けます。
3. [リポジトリの更新] をクリックし、チャンネルとリポジトリを関連付けます。
4. 関連付ける必要があるすべてのチャンネルとすべてのリポジトリを繰り返します。
5. オプション: [同期] タブに移動し、このリポジトリの同期の繰り返しスケジュールを設定します。
6. [今すぐ同期] をクリックし、すぐに同期を開始します。

4.10.1.4. 同期ステータスの確認

プロシージャ: Web UIから同期の進捗状況を確認する

1. UyuniのWeb UIで、**ソフトウェア > 管理 > チャンネル**に移動し、リポジトリに関連付けられているチャンネルをクリックします。
2. [リポジトリ] タブに移動し、[同期] をクリックし、[同期状態] をクリックします。

プロシージャ: コマンドプロンプトから同期の進捗状況を確認する

1. Uyuniサーバのコマンドプロンプトで、rootとして、**tail** コマンドを使用して同期ログファイルを確認します。

```
tail -f /var/log/rhn/reposync/<channel-label>.log
```

2. それぞれの子チャンネルは、同期の進捗中にそれぞれのログを生成します。同期が完了したことを確認するには、ベースチャンネルと子チャンネルのログファイルをすべて確認する必要があります。



Red Hat Enterprise Linuxチャンネルは非常に大きいことがあります。同期に数時間かかる場合があります。

プロシージャ: オプション: Salt状態を作成して設定ファイルを展開する

1. UyuniサーバのWeb UIで、**設定 > チャンネル**に移動します。
2. [状態チャレンネルの作成] をクリックします。
 - [名前] フィールドに **subscription-manager: disable yum plugins** と入力します。
 - [ラベル] フィールドに **subscription-manager-disable-yum-plugins** と入力します。
 - [説明] フィールドに **subscription-manager: disable yum plugins** と入力します。
 - [SLS コンテンツ] フィールドは空白のままにします。
3. [設定チャレンネルの作成] をクリックします
4. [設定ファイルの作成] をクリックします

- 【ファイル名/パス】フィールドに /etc/yum/pluginconf.d/subscription-manager.conf と入力します。
- 【ファイルの内容】フィールドに次のように入力します。

```
[main]
enabled=0
```

- 【設定ファイルの作成】をクリックします
- 【Salt ファイルシステムパス】フィールドの値をメモします。
- 設定チャンネルの名前をクリックします。
- 【'init.sls' ファイルの表示/編集】をクリックします
 - 【ファイルの内容】フィールドに次のように入力します。

```
configure_subscription-manager-disable-yum-plugins:
cmd.run:
  - name: subscription-manager config
--rhsm.auto_enable_yum_plugins=0
  - watch:
    - file: /etc/yum/pluginconf.d/subscription-manager.conf
file.managed:
  - name: /etc/yum/pluginconf.d/subscription-manager.conf
  - source: salt:///etc/yum/pluginconf.d/subscription-
manager.conf
```

- 【設定ファイルの更新】をクリックします。



Salt 状態を作成して設定ファイルを開く のプロシージャはオプショナル

プロシージャ: Red Hat Enterprise Linuxクライアントのシステムグループの作成

- UyuniサーバのWeb UIで、**システム > システムグループ**に移動します。
- 【グループの作成】をクリックします。
 - 【名前】フィールドに **rhel-systems** と入力します。
 - 【説明】フィールドに **All RHEL systems** と入力します。
- 【グループの作成】をクリックします。
- 【状態】タブをクリックします。
- 【設定チャンネル】タブをクリックします。
- 検索ボックスに **subscription-manager: disable yum plugins** と入力します。

7. [検索] をクリックして状態を表示します。
8. **Assign** 列で状態のチェックボックスをクリックします。
9. [変更点の保存] をクリックします。
10. [確認] をクリックします。

RHELシステムをUyuniに追加済みの場合、これらを新しいシステムグループに割り当て、highstateを適用します。

プロシージャ: システムグループをアクティベーションキーに追加する

RHELシステムで使用したアクティベーションキーを変更して、上記で作成したシステムグループに含めます。

1. UyuniサーバのWeb UIで、**システム > アクティベーションキー**に移動します。
2. RHELシステムで使用されるそれぞれのアクティベーションキーをクリックします。
3. [グループ] タブ、[参加] サブタブに移動します。
4. [Select rhel-systems] (RHELシステムを選択) にチェックを付けます。
5. [選択されたグループに参加] をクリックします。

4.10.1.5. クライアントでGPGキーを信頼する

4.10.1.6. クライアントでGPGキーを信頼する

Operating systems either trust their own GPG keys directly or at least ship them installed with the minimal system. But third party packages signed by a different GPG key need manual handling. The clients can be successfully bootstrapped without the GPG key being trusted. However, you cannot install new client tool packages or update them until the keys are trusted.

Salt clients use now GPG key information entered for a software channel to manage the trusted keys. When a software channel with GPG key information is assigned to a client, the key gets trusted as soon as the channel is refreshed or the first package gets installed from this channel.

The GPG key URL which is set of a software channel must exist. In case it is a file URL, the GPG key file must be deployed on the client before the software channel is used.

The GPG keys for the Client Tools Channels of Red Hat based clients are deployed on the client into **/etc/pki/rpm-gpg/** and can be referenced with file URLs. Same is the case with the GPG keys of Expanded Support clients. Only in case a software channel is assigned to the client they will be imported and trusted by the system.



Because Debian based systems sign only metadata, there is typically no need to specify extra keys for single channels. If a user configures an own GPG key to sign the metadata as described in "Use Your Own GPG Key" in **Administration > Repo-metadata** the deployment and trust of that key is executed automatically.

4.10.1.6.1. User defined GPG keys

Users can define their own GPG keys to be deployed to the client.

By providing some pillar data and providing the GPG key files in the Salt filesystem, they are automatically deployed to the client.

These keys are deployed into `/etc/pki/rpm-gpg/` on RPM based operating systems and to `/usr/share/keyrings/` on Debian systems:

Define the pillar key [literal `custom_gpgkeys`] for the client you want to deploy the key to and list the names of the key file.

```
cat /etc/pillar/mypillar.sls
custom_gpgkeys:
  - my_first_gpg.key
  - my_second_gpgkey.gpg
```

Additionally in the Salt filesystem create a directory named `gpg` and store there the GPG key files with the name specified in the `custom_gpgkeys` pillar data.

```
ls -la /etc/salt/gpg/
/etc/salt/gpg/my_first_gpg.key
/etc/salt/gpg/my_second_gpgkey.gpg
```

The keys are now deployed to the client at `/etc/pki/rpm-gpg/my_first_gpg.key` and `/etc/pki/rpm-gpg/my_second_gpgkey.gpg`.

The last step is to add the URL to the GPG key URL field of the software channel. Navigate to **Software > Manage > Channels** and select the channel you want to modify. Add to **GPG key URL** the value `file:///etc/pki/rpm-gpg/ my_first_gpg.key`.

4.10.1.6.2. GPG Keys in Bootstrap Scripts

プロシージャ: ブートストラップスクリプトを使用してクライアントでGPGキーを信頼する

- Uyuniサーバのコマンドプロンプトで、`/srv/www/htdocs/pub/`ディレクトリの内容を確認します。このディレクトリには、使用できるすべての公開鍵が含まれています。登録クライアントに割り当てるチャンネルに適用するキーをメモします。
- 関連するブートストラップスクリプトを開き、`ORG_GPG_KEY=`パラメータを見つけて、必要なキーを追加します。次に例を示します。

```
uyuni-gpg-pubkey-0d20833e.key
```

以前保存したキーを削除する必要はありません。



Trusting a GPG key is important for security on clients. It is the task of the admin to decide which keys are needed and can be trusted. Because a software channel cannot be used when the GPG key is not trusted, the decision of assigning a channel to a client depends on the decision of trusting the key.

4.10.1.7. クライアントの登録

Red Hatクライアントを登録するには、ブートストラップリポジトリが必要です。デフォルトでは、ブートストラップリポジトリは自動的に作成され、すべての同期製品に対して毎日再生成されます。次のコマンドを使用して、コマンドプロンプトからブートストラップリポジトリを手動で作成できます。

```
mgr-create-bootstrap-repo
```

クライアントの登録については、[Client-configuration > Registration-overview](#)を参照してください。



Red Hat Enterprise Linux 6クライアントを登録して使用するには、Uyuniサーバを設定して旧式のSSL暗号化をサポートする必要があります。このエラーを解決する方法については、[古いクライアントの登録をClient-configuration clients](#)で参照してください。

4.10.2. RHUIでRed Hat Enterprise Linuxクライアントを登録する

SUSE Linux Enterprise Server with Expanded Supportを使用するのではなく、Red Hat Enterprise Linuxクライアントを直接実行している場合、Red Hatソースを使用してパッケージを取得および更新する必要があります。このセクションでは、Red Hat更新インフラストラクチャ(RHUI)を使用して、Red Hat Enterprise Linuxオペレーティングシステムを実行している従来のクライアントおよびSaltクライアントを登録する方法について説明します。Amazon EC2などのパブリッククラウドでクライアントを実行している場合、このメソッドを使用します。

RHUIをRed Hatコンテンツデリバリネットワーク(CDN)と組み合わせて使用して、Red Hat Enterprise Linuxサブスクリプションを管理できます。Red Hat CDNの使用については、[Client-configuration > Clients-rh-cdn](#)を参照してください。



Red Hat Enterprise Linuxクライアントは、Red Hatに基づいていて、SUSE Linux Enterprise Server with Expanded Support、RES、またはSUSE Linux Enterprise Serverとは関係がありません。UyuniサーバのRed Hat更新インフラストラクチャへの接続はユーザが行います。このRHUI証明書を使用して更新したすべてのクライアントは、正しくライセンス供与されている必要があります。クラウドプロバイダに確認し、詳細については、Red Hatのサービス条項を確認してください。



RHUIで登録されたRed Hat Enterprise Linuxクライアントの電源がオフになっている場合、Red Hatが証明書を無効と宣言する場合があります。この場合、クライアントの電源を再度オンにするか、新しいRHUI証明書を取得する必要があります。



従来のクライアントはRed Hat Enterprise Linux 6および7でのみサポートされています。 Red Hat Enterprise Linux 8クライアントはSaltクライアントとしてサポートされています。

4.10.2.1. エンタイトルメントと証明書のインポート

Red Hatクライアントには、Red Hat認証局(CA)、エンタイトルメント証明書、およびエンタイトルメントキーが必要です。

Red Hatクライアントは、URLを使用してリポジトリを複製します。 URLは、Red Hatクライアントを登録した場所に応じて変わります。

Red Hatクライアントは次の3種類の方法で登録できます。

- ・ redhat.comにあるRed Hatコンテンツデリバリネットワーク(CDN)
- ・ Red Hatサテライトサーバ
- ・ クラウドのRed Hat更新インフラストラクチャ(RHUI)

このガイドでは、Red Hat更新インフラストラクチャ(RHUI)に登録されるクライアントについて説明します。 リポジトリコンテンツの認可済みサブスクリプションを使用して、1つ以上のシステムがRHUIに登録されている必要があります。

代わりにRed Hatコンテンツデリバリネットワーク(CDN)を使用する方法については、[Client-configuration > Clients-rh-cdn](#)を参照してください。



クライアントシステムのサテライト証明書では、サテライトサーバおよびサブスクリプションが必要です。 サテライト証明書を使用するクライアントはUyuniサーバではサポートされません。

Uyuniサーバがアクセスできる場所にエンタイトルメント証明書とキーをクライアントシステムからコピーする必要があります。

キーと証明書の名前は、ここで示した名前とは多少異なる場合があります。 エンタイトルメント証明書およびRed Hat CA証明書ファイルのファイル拡張子は.crtです。 キーのファイル拡張子は.keyです。

プロシージャ: 証明書をサーバにコピーする

1. Uyuniサーバがアクセスできる場所にエンタイトルメント証明書とキーをクライアントシステムからコピーします。

Amazon EC2:

```
cp /etc/pki/rhui/product/content-<version>.crt /<example>/entitlement/
cp /etc/pki/rhui/content-<version>.key /<example>/entitlement/
```

Azure:

- 次のコマンドを使用して証明書チェーンを確認します。

```
openssl s_client -connect rhui-1.microsoft.com:443 -showcerts
```

+ サンプル出力は次のようにになります。

+

```
CONNECTED(00000003)
depth=2 C = US, O = DigiCert Inc, OU = www.digicert.com, CN = DigiCert
Global Root G2
verify return:1
depth=1 C = US, O = Microsoft Corporation, CN = Microsoft Azure TLS
Issuing CA 06
verify return:1
depth=0 C = US, ST = WA, L = Redmond, O = Microsoft Corporation, CN =
rhui-1.microsoft.com
verify return+
```

2番目の証明書(literal) **CN = Microsoft Azure**を確認します。VMで同じである場合は、証明書名をメモします。 証明書をダウンロードするには、<https://docs.microsoft.com/en-us/azure/active-directory/fundamentals/certificateAuthorities>を参照してください。 AIAリンクをクリックして、証明書をダウンロードします。証明書は、**.cer**サフィックスが付いてダウンロードされます。次のコマンドを使用して、**.crt**に変換します。

+

```
openssl x509 -inform DER -in <example.cer> -out <example.crt>
```

+ Googleクラウドプラットフォーム:

+

```
cp /etc/pki/rhui/product/content.crt /<example>/entitlement/
cp /etc/pki/rhui/key.pem /<example>/entitlement/
```

+ . Red Hat CA証明書ファイルをクライアントシステムから、エンタイトルメント証明書およびキーと同じ場所にコピーします。

+ Amazon EC2:

+

```
cp /etc/pki/rhui/cdn.redhat.com-chain.crt /<example>/entitlement
```

+ Azure:

+ 変換された証明書を/<example>/entitlement/にアップロードします

+ Googleクラウドプラットフォーム:

+

```
cp /etc/pki/rhui/ca.crt /<example>/entitlement
```

Red Hatクライアントでリポジトリを管理するには、CAおよびエンタイトルメント証明書をUyuniサーバにインポートする必要があります。この操作を実行するには、インポートプロシージャを3回実行して、3つのエントリを作成する必要があります。エンタイトルメント証明書、エンタイトルメントキーおよびRed Hat証明書にそれぞれ1つずつです。

プロシージャ: 証明書をサーバにインポートする

1. UyuniサーバのWeb UIで、**システム** > **自動インストール** > **GPGキーとSSLキー**に移動します。
2. **[格納されているキーまたは証明書の作成]**をクリックして、エンタイトルメント証明書データを設定します。
 - [説明] フィールドに**Entitlement-Cert-Date**と入力します。
 - [タイプ] フィールドで、**SSL**を選択します。
 - [アップロードするファイルの選択] フィールドで、エンタイトルメント証明書を保存した場所をブラウズし、**.crt**証明書ファイルを選択します。
3. **[キーの作成]**をクリックします。
4. **[格納されているキーまたは証明書の作成]**をクリックして、エンタイトルメントキー用データを設定します。
 - [説明] フィールドに**Entitlement-Key-Date**と入力します。
 - [タイプ] フィールドで、**SSL**を選択します。
 - [アップロードするファイルの選択] フィールドで、エンタイトルメントキーを保存した場所をブラウズし、**.key**キーファイルを選択します。
5. **[キーの作成]**をクリックします。
6. **[格納されているキーまたは証明書用]**を次のパラメータを設定します。
 - [説明] フィールドに**redhat-cert**と入力します。
 - [タイプ] フィールドで、**SSL**を選択します。
 - [アップロードするファイルの選択] フィールドで、Red Hat証明書を保存した場所をブラウズし、

証明書ファイルを選択します。

7. [証明書の作成] をクリックします。

4.10.2.2. ソフトウェアチャンネルの追加

Red HatクライアントをUyuniサーバに登録する前に、必要なソフトウェアチャンネルを追加して同期する必要があります。



In the following section, descriptions often default to the `x86_64` architecture. Replace it with other architectures if appropriate.

このプロシージャで必要なチャンネルは次のとおりです。

表 32. Red Hatチャンネル - CLI

OSバージョン	ベースチャンネル	クライアントチャンネル	ツールチャンネル
Red Hat 6	<code>rhel-x86_64-server-6</code>	-	<code>res6-suse-manager-tools-x86_64</code>
Red Hat 7	<code>rhel-x86_64-server-7</code>	-	<code>res7-suse-manager-tools-x86_64</code>
Red Hat 8	<code>rhel8-pool-x86_64</code>	-	<code>res8-suse-manager-tools-x86_64</code>



Red Hat 6はサポート終了になっており、そのリポジトリで提供されるISOイメージは失効しています。これらのパッケージを使用した新しいRed Hat 6クライアントのブートストラップは失敗します。新しいRed Hat 6クライアントをブートストラップする必要がある場合、[Client-configuration > Tshoot-clients](#)のトラブルシューティングプロシージャに従ってください。

手順: コマンドプロンプトからのソフトウェアチャンネルの追加

1. Uyuni サーバのコマンドプロンプトで root になり、`spacewalk-common-channels` コマンドを特定のチャンネルに対して実行します:

```
spacewalk-common-channels \
<ベースチャンネルのラベル> \
<子チャンネル_1> \
<子チャンネル_2> \
... <子チャンネル_n>
```

2. チャンネルの同期:

```
spacewalk-repo-sync -p <base_channel_label>
```

3. 続行前に、同期が完了していることを確認してください。



spacewalk-common-channels によって提供されるクライアントツールのチャンネルの提供元はUyuniです。SUSEではありません。



AppStreamリポジトリにはモジュールパッケージが用意されています。UyuniのWeb UIに正しくないパッケージ情報が表示されます。Web UIまたはAPIを使用してモジュールリポジトリから直接インストールまたはアップグレードするようなパッケージ操作は実行できません。

コンテンツライフサイクル管理(CLM)でAppStreamフィルタを使用して、モジュールリポジトリを通常のリポジトリに変換できます。クライアントで **spacecmd** を使用する場合は、AppStreamフィルタを使用して **python:3.6** を必ず含めてください。

または、Salt状態を使用してSaltクライアントでモジュラーパッケージを管理したり、クライアントで **dnf** コマンドを使用することもできます。CLMの詳細については、[Administration > Content-lifecycle](#)を参照してください。

RHUIを使用するには、必要なHTTPヘッダを設定ファイルに手動で追加する必要があります。このヘッダがないと、クライアントの同期を正常に実行できません。

プロシージャ: HTTPヘッダを設定ファイルに追加する

1. **X-RHUI-ID** および **X-RHUI-SIGNATURE** HTTPヘッダをRHUIインスタンスから検索します。Red Hatクライアントで次のコマンドを使用して、**169.254.169.254** でクラウドインスタンスマタデータAPIから値を取得できます。

```
echo "X-RHUI-ID=$(curl -s
http://169.254.169.254/latest/dynamic/instance-
identity/document|base64|tr -d '\n')"
echo "X-RHUI-SIGNATURE=$(curl -s
http://169.254.169.254/latest/dynamic/instance-
identity/signature|base64|tr -d '\n')"
```

2. **/etc/rhn/spacewalk-repo-sync/extr\$headers.conf** 設定ファイルを開き、次の行を追加するか、または正しい情報を使用して編集します。

```
[<channel_label_1>]
X-RHUI-ID=<value>
X-RHUI-SIGNATURE=<value>

[<channel_label_2>]
X-RHUI-ID=<value>
X-RHUI-SIGNATURE=<value>
```

上記の[literal]``<channel_label_X>``を[literal]``rhel8-baseos-repo``などのチャンネル名に置き換えます。

```
[ rhel8-baseos-repo]
X-RHUI-ID=...
X-RHUI-SIGNATURE=...
```

4.10.2.3. カスタムリポジトリおよびチャンネルの準備

RHUIからソフトウェアをミラーリングするには、URLでRHUIにリンクされているカスタムチャンネルおよびリポジトリをUyuniに作成する必要があります。Red Hatポータルでこれらの製品を正しく動作させるには、該当製品のエンタイトルメントが必要です。yumユーティリティを使用して、ミラーリングするリポジトリのURLを取得できます。

```
yum repolist -v | grep baseurl
```

これらのリポジトリURLを使用して、カスタムリポジトリを作成できます。クライアントを管理するために必要なコンテンツのみミラーリングできます。



Red Hatポータルに正しいエンタイトルメントがある場合、Red Hatリポジトリのカスタムバージョンのみ作成できます。

このプロセッジャに必要な詳細は次のとおりです。

表 33. Red Hatカスタムリポジトリ設定

オプション	設定
リポジトリURL	RHUIによって提供されるコンテンツURL
署名済みメタデータがあるかどうか	すべてのRed Hatエンタイトルメントリポジトリのチェックを外します
SSL CA証明書	redhat-cert
SSLクライアント証明書	Entitlement-Cert-Data
SSLクライアントキー	Entitlement-Key-Data

プロセッジャ: カスタムリポジトリの作成

1. UyuniサーバのWeb UIで、**ソフトウェア > 管理 > リポジトリ**に移動します。
2. **[リポジトリの作成]**をクリックし、リポジトリに適切なパラメータを設定します。
3. **[リポジトリの作成]**をクリックします。
4. 作成する必要があるすべてのリポジトリで繰り返します。

このプロセージャで必要なチャンネルは次のとおりです。

表 34. Red Hatカスタムチャンネル

OSバージョン	ベース製品	ベースチャンネル
Red Hat 6	RHEL6 Base x86_64	rhel6-pool-x86_64
Red Hat 7	RHEL7 Base x86_64	rhel7-pool-x86_64
Red Hat 8	RHELまたはSLES ESまたはCentOS 8 Base	rhel8-pool-x86_64



Red Hat 6はサポート終了になっており、そのリポジトリで提供されるISOイメージは失効しています。これらのパッケージを使用した新しいRed Hat 6クライアントのブートストラップは失敗します。新しいRed Hat 6クライアントをブートストラップする必要がある場合、[Client-configuration > Tshoot-clients](#)のトラブルシューティングプロセージャに従ってください。

プロセージャ: カスタムチャンネルの作成

1. UyuniサーバのWeb UIで、[ソフトウェア > 管理 > チャンネル](#)に移動します。
2. [\[チャンネルの作成\]](#)をクリックし、チャンネルに適切なパラメータを設定します。
3. [\[親 チャンネル\]](#) フィールドで、適切なベースチャンネルを選択します。
4. [\[チャンネルの作成\]](#)をクリックします。
5. 作成する必要があるすべてのチャンネルで繰り返します。各カスタムリポジトリに1つのカスタムチャンネルが必要です。

該当するすべてのチャンネルとリポジトリを作成したことを確認できます。そのためには、[ソフトウェア > チャンネル一覧 > すべて](#)に移動します。



Red Hat 8クライアントでは、ベースチャンネルとAppStreamチャンネルの両方を追加します。両方のチャンネルのパッケージが必要です。両方のチャンネルを追加しないと、パッケージ不足のためブートストラップリポジトリを作成できません。

すべてのチャンネルを作成済みの場合、これらのチャンネルを、作成したリポジトリと関連付けできます。

プロセージャ: チャンネルのリポジトリとの関連付け

1. UyuniサーバのWeb UIで、[ソフトウェア > 管理 > チャンネル](#)に移動し、関連付けるチャンネルをクリックします。
2. [\[リポジトリ\]](#) タブに移動し、このチャンネルと関連付けるリポジトリにチェックを付けます。
3. [\[リポジトリの更新\]](#)をクリックし、チャンネルとリポジトリを関連付けます。
4. 関連付ける必要があるすべてのチャンネルとすべてのリポジトリを繰り返します。
5. オプション: [\[同期\]](#) タブに移動し、このリポジトリの同期の繰り返しスケジュールを設定します。

6. [今すぐ同期] をクリックし、すぐに同期を開始します。

4.10.2.4. 同期ステータスの確認

プロシージャ: Web UIから同期の進捗状況を確認する

1. UyuniのWeb UIで、**ソフトウェア** > **管理** > **チャンネル**に移動し、リポジトリに関連付けられているチャンネルをクリックします。
2. [リポジトリ] タブに移動し、[同期] をクリックし、[同期状態] をクリックします。

プロシージャ: コマンドプロンプトから同期の進捗状況を確認する

1. Uyuniサーバのコマンドプロンプトで、rootとして、**tail** コマンドを使用して同期ログファイルを確認します。

```
tail -f /var/log/rhn/reposync/<channel-label>.log
```

2. それぞれの子チャンネルは、同期の進捗中にそれぞれのログを生成します。同期が完了したことを確認するには、ベースチャンネルと子チャンネルのログファイルをすべて確認する必要があります。



Red Hat Enterprise Linuxチャンネルは非常に大きいことがあります。同期に数時間かかる場合があります。

4.10.2.5. クライアントでGPGキーを信頼する

4.10.2.6. クライアントでGPGキーを信頼する

Operating systems either trust their own GPG keys directly or at least ship them installed with the minimal system. But third party packages signed by a different GPG key need manual handling. The clients can be successfully bootstrapped without the GPG key being trusted. However, you cannot install new client tool packages or update them until the keys are trusted.

Salt clients use now GPG key information entered for a software channel to manage the trusted keys. When a software channel with GPG key information is assigned to a client, the key gets trusted as soon as the channel is refreshed or the first package gets installed from this channel.

The GPG key URL which is set of a software channel must exist. In case it is a file URL, the GPG key file must be deployed on the client before the software channel is used.

The GPG keys for the Client Tools Channels of Red Hat based clients are deployed on the client into **/etc/pki/rpm-gpg/** and can be referenced with file URLs. Same is the case with the GPG keys of Expanded Support clients. Only in case a software channel is assigned to the client they will be imported and trusted by the system.



Because Debian based systems sign only metadata, there is typically no need to specify extra keys for single channels. If a user configures an own GPG key to sign the metadata as described in "Use Your Own GPG Key" in **Administration > Repo-metadata** the deployment and trust of that key is executed automatically.

4.10.2.6.1. User defined GPG keys

Users can define their own GPG keys to be deployed to the client.

By providing some pillar data and providing the GPG key files in the Salt filesystem, they are automatically deployed to the client.

These keys are deployed into `/etc/pki/rpm-gpg/` on RPM based operating systems and to `/usr/share/keyrings/` on Debian systems:

Define the pillar key [literal `custom_gpgkeys`] for the client you want to deploy the key to and list the names of the key file.

```
cat /etc/pillar/mypillar.sls
custom_gpgkeys:
  - my_first_gpg.key
  - my_second_gpgkey.gpg
```

Additionally in the Salt filesystem create a directory named `gpg` and store there the GPG key files with the name specified in the `custom_gpgkeys` pillar data.

```
ls -la /etc/salt/gpg/
/etc/salt/gpg/my_first_gpg.key
/etc/salt/gpg/my_second_gpgkey.gpg
```

The keys are now deployed to the client at `/etc/pki/rpm-gpg/my_first_gpg.key` and `/etc/pki/rpm-gpg/my_second_gpgkey.gpg`.

The last step is to add the URL to the GPG key URL field of the software channel. Navigate to **Software > Manage > Channels** and select the channel you want to modify. Add to **GPG key URL** the value `file:///etc/pki/rpm-gpg/ my_first_gpg.key`.

4.10.2.6.2. GPG Keys in Bootstrap Scripts

プロシージャ: ブートストラップスクリプトを使用してクライアントでGPGキーを信頼する

1. Uyuniサーバのコマンドプロンプトで、`/srv/www/htdocs/pub/` ディレクトリの内容を確認します。このディレクトリには、使用できるすべての公開鍵が含まれています。登録クライアントに割り当てるチャンネルに適用するキーをメモします。

- 関連するブートストラップスクリプトを開き、**ORG_GPG_KEY=**パラメータを見つけて、必要なキーを追加します。次に例を示します。

```
uyuni-gpg-pubkey-0d20833e.key
```

以前保存したキーを削除する必要はありません。



Trusting a GPG key is important for security on clients. It is the task of the admin to decide which keys are needed and can be trusted. Because a software channel cannot be used when the GPG key is not trusted, the decision of assigning a channel to a client depends on the decision of trusting the key.

4.10.2.7. クライアントの登録

Red Hatクライアントを登録するには、ブートストラップリポジトリが必要です。デフォルトでは、ブートストラップリポジトリは自動的に作成され、すべての同期製品に対して毎日再生成されます。次のコマンドを使用して、コマンドプロンプトからブートストラップリポジトリを手動で作成できます。

```
mgr-create-bootstrap-repo
```

クライアントの登録については、[Client-configuration > Registration-overview](#)を参照してください。



Red Hat Enterprise Linux 6クライアントを登録して使用するには、Uyuniサーバを設定して旧式のSSL暗号化をサポートする必要があります。詳細については、[古いクライアントの登録](#)を[Client-configuration > Activation-key](#)で参照してください。

4.11. Rocky Linuxクライアントの登録

Rocky LinuxクライアントをUyuniサーバに登録できます。そのメソッドおよび詳細は、クライアントのオペレーティングシステムによって異なります。

始める前に、クライアントでUyuniサーバと日時が正しく同期していることを確認してください。

アクティベーションキーを作成済みである必要があります。アクティベーションキーの作成の詳細については、[Client-configuration > Activation-keys](#)を参照してください。

4.11.1. Rocky Linuxクライアントの登録

このセクションでは、Rocky Linuxオペレーティングシステムを実行しているSaltクライアントの登録について説明します。



従来のクライアントはRocky Linux 8では使用できません。Rocky Linux 8クライアントはSaltクライアントとしてのみサポートされます。



Linux客户端のUyuniへの登録は、ターゲットポリシーで適用されるデフォルトのSELinux設定でテストされます。Rocky LinuxクライアントをUyuniに登録するために、SELinuxを無効にする必要はありません。

4.11.1.1. ソフトウェアチャンネルの追加

Rocky LinuxクライアントをUyuniサーバに登録する前に、必要なソフトウェアチャンネルを追加して同期する必要があります。

現在サポートされているアーキテクチャは、「x86_64」と「aarch64」です。サポートされている製品およびアーキテクチャの完全な一覧については、[Client-configuration > Supported-features](#)を参照してください。



In the following section, descriptions often default to the `x86_64` architecture. Replace it with other architectures if appropriate.

このプロシージャで必要なチャンネルは次のとおりです。

表 35. Rocky Linuxチャンネル - CLI

OSバージョン	ベースチャンネル	クライアントチャンネル	AppStreamチャンネル
Rocky Linux 8	<code>rockylinux8</code>	<code>rockylinux8-uyuni-client</code>	<code>rockylinux8-appstream</code>

手順: コマンドプロンプトからのソフトウェアチャンネルの追加

1. Uyuni サーバのコマンドプロンプトで root になり、`spacewalk-common-channels` コマンドを特定のチャンネルに対して実行します。このとき、正しいアーキテクチャを指定してください:

```
spacewalk-common-channels \
-a <アーキテクチャ> \
<ベースチャンネル名> \
<子チャンネル_1> \
<子チャンネル_2> \
... <子チャンネル_n>
```

2. チャンネルの同期:

```
spacewalk-repo-sync -p <base_channel_label>
```

3. 続行前に、同期が完了していることを確認してください。



spacewalk-common-channels によって提供されるクライアントツールのチャンネルの提供元はUyuniです。SUSEではありません。



Rocky Linux 8クライアントでは、ベースチャンネルとAppStreamチャンネルの両方を追加します。両方のチャンネルのパッケージが必要です。両方のチャンネルを追加しないと、パッケージ不足のためブートストラップリポジトリを作成できません。



上流のチャンネルとUyuniチャンネルの間のAppStreamチャンネルで利用できるパッケージ数に不一致が発生する場合があります。また、同時に別の場所で追加したチャンネルを比較すると、数値が異なる場合もあります。Rocky Linuxでリポジトリを管理する方法が原因です。Rocky Linuxでは新しいバージョンがリリースされると古いバージョンのパッケージが削除されますが、Uyuniでは経過年数に関係なくすべてのバージョンが保持されます。

モジューラーチャンネルを使用している場合は、クライアントでPython3.6モジュールストリームを有効にする必要があります。Python 3.6を提供しない場合、**spacecmd** パッケージのインストールは失敗します。

AppStreamリポジトリにはモジュールパッケージが用意されています。UyuniのWeb UIに正しくないパッケージ情報が表示されます。Web UIまたはAPIを使用してモジュールリポジトリから直接インストールまたはアップグレードするようなパッケージ操作は実行できません。



コンテンツライフサイクル管理(CLM)でAppStreamフィルタを使用して、モジュールリポジトリを通常のリポジトリに変換できます。クライアントで**spacecmd** を使用する場合は、AppStreamフィルタを使用して**python:3.6** を必ず含めてください。

または、Salt状態を使用してSaltクライアントでモジューラーパッケージを管理したり、クライアントで**dnf** コマンドを使用することもできます。CLMの詳細については、[Administration > Content-lifecycle](#)を参照してください。

4.11.1.2. 同期ステータスの確認

プロシージャ: Web UIから同期の進捗状況を確認する

1. UyuniのWeb UIで、**ソフトウェア** > **管理** > **チャンネル**に移動し、リポジトリに関連付けられているチャンネルをクリックします。
2. [リポジトリ] タブに移動し、[同期] をクリックし、[同期状態] をクリックします。

プロシージャ: コマンドプロンプトから同期の進捗状況を確認する

1. Uyuniサーバのコマンドプロンプトで、rootとして、**tail** コマンドを使用して同期ログファイルを確認します。

```
tail -f /var/log/rhn/reposync/<channel-label>.log
```

2. それぞれの子チャンネルは、同期の進捗中にそれぞれのログを生成します。同期が完了したことを確認するには、ベースチャンネルと子チャンネルのログファイルをすべて確認する必要があります。

4.11.1.3. アクティベーションキーの作成

Rocky Linuxチャンネルと関連付けられているアクティベーションキーを作成する必要があります。

アクティベーションキーの詳細については、[Client-configuration > Activation-keys](#)を参照してください。

4.11.1.4. クライアントでGPGキーを信頼する

4.11.1.5. クライアントでGPGキーを信頼する

Operating systems either trust their own GPG keys directly or at least ship them installed with the minimal system. But third party packages signed by a different GPG key need manual handling. The clients can be successfully bootstrapped without the GPG key being trusted. However, you cannot install new client tool packages or update them until the keys are trusted.

Salt clients use now GPG key information entered for a software channel to manage the trusted keys. When a software channel with GPG key information is assigned to a client, the key gets trusted as soon as the channel is refreshed or the first package gets installed from this channel.

The GPG key URL which is set of a software channel must exist. In case it is a file URL, the GPG key file must be deployed on the client before the software channel is used.

The GPG keys for the Client Tools Channels of Red Hat based clients are deployed on the client into [/etc/pki/rpm-gpg/](#) and can be referenced with file URLs. Same is the case with the GPG keys of Expanded Support clients. Only in case a software channel is assigned to the client they will be imported and trusted by the system.



Because Debian based systems sign only metadata, there is typically no need to specify extra keys for single channels. If a user configures an own GPG key to sign the metadata as described in "Use Your Own GPG Key" in [Administration > Repo-metadata](#) the deployment and trust of that key is executed automatically.

4.11.1.5.1. User defined GPG keys

Users can define their own GPG keys to be deployed to the client.

By providing some pillar data and providing the GPG key files in the Salt filesystem, they are automatically deployed to the client.

These keys are deployed into [/etc/pki/rpm-gpg/](#) on RPM based operating systems and to [/usr/share/keyrings/](#) on Debian systems:

Define the pillar key [literal `custom_gpgkeys`] for the client you want to deploy the key to and list the names of the key file.

```
cat /etc/pillar/mypillar.sls
custom_gpgkeys:
  - my_first_gpg.key
  - my_second_gpgkey.gpg
```

Additionally in the Salt filesystem create a directory named `gpg` and store there the GPG key files with the name specified in the `custom_gpgkeys` pillar data.

```
ls -la /etc/salt/gpg/
/etc/salt/gpg/my_first_gpg.key
/etc/salt/gpg/my_second_gpgkey.gpg
```

The keys are now deployed to the client at `/etc/pki/rpm-gpg/my_first_gpg.key` and `/etc/pki/rpm-gpg/my_second_gpgkey.gpg`.

The last step is to add the URL to the GPG key URL field of the software channel. Navigate to [Software > Manage > Channels](#) and select the channel you want to modify. Add to **GPG key URL** the value `file:///etc/pki/rpm-gpg/my_first_gpg.key`.

4.11.1.5.2. GPG Keys in Bootstrap Scripts

プロシージャ: ブートストラップスクリプトを使用してクライアントでGPGキーを信頼する

1. Uyuniサーバのコマンドプロンプトで、`/srv/www/htdocs/pub/`ディレクトリの内容を確認します。このディレクトリには、使用できるすべての公開鍵が含まれています。登録クライアントに割り当てるチャンネルに適用するキーをメモします。
2. 関連するブートストラップスクリプトを開き、`ORG_GPG_KEY=`パラメータを見つけて、必要なキーを追加します。次に例を示します。

```
uyuni-gpg-pubkey-0d20833e.key
```

以前保存したキーを削除する必要はありません。



Trusting a GPG key is important for security on clients. It is the task of the admin to decide which keys are needed and can be trusted. Because a software channel cannot be used when the GPG key is not trusted, the decision of assigning a channel to a client depends on the decision of trusting the key.

4.11.1.6. クライアントの登録

Rocky Linux クライアントは、その他すべてのクライアントと同じ方法で登録されます。 詳細については、[Client-configuration > Registration-overview](#)を参照してください。

4.11.1.7. エラータの管理

Rocky Linuxクライアントを更新するとき、パッケージには更新に関するメタデータが含まれています。

4.12. Ubuntuクライアントの登録

UbuntuクライアントをUyuniサーバに登録できます。 そのメソッドおよび詳細は、クライアントのオペレーティングシステムによって異なります。

始める前に、クライアントでUyuniサーバと日時が正しく同期していることを確認してください。

アクティベーションキーを作成済みである必要もあります。 アクティベーションキーの作成の詳細については、[Client-configuration > Activation-keys](#)を参照してください。

4.12.1. Registering Ubuntu 20.04 and 22.04 Clients

This section contains information about registering Salt clients running Ubuntu 20.04 LTS and 22.04 LTS operating systems.



Canonicalは、Uyuniを保証またはサポートしていません。



UbuntuはSaltクライアントでのみサポートされています。 従来のクライアントはサポートされていません。

ブートストラップは、リポジトリの設定やプロファイルの更新の実行など、Ubuntuクライアントの起動および初期状態の実行のためにサポートされています。 ただし、Ubuntuのrootユーザはデフォルトで無効になっているため、ブートストラップを使用するには、Pythonの `sudo` 特権がある既存ユーザが必要です。

4.12.1.1. ソフトウェアチャンネルの追加

UbuntuクライアントをUyuniサーバに登録する前に、必要なソフトウェアチャンネルを追加して同期する必要があります。



In the following section, descriptions often default to the `x86_64` architecture. Replace it with other architectures if appropriate.

このプロシージャで必要なチャンネルは次のとおりです。

表 36. Ubuntuチャンネル - CLI

OS Version	Base Channel	Main Channel	Updates Channel	Security Channel	Client Channel
Ubuntu 20.04	ubuntu-2004-amd64-main for amd64	ubuntu-2004-amd64-main-uyuni	ubuntu-2004-amd64-main-updates-uyuni	ubuntu-2004-amd64-main-security-uyuni	ubuntu-2004-amd64-uyuni-client
Ubuntu 22.04	ubuntu-2204-amd64-main for amd64	ubuntu-2204-amd64-main-uyuni	ubuntu-2204-amd64-main-updates-uyuni	ubuntu-2204-amd64-main-security-uyuni	ubuntu-2204-amd64-uyuni-client

Version 20.04 also requires the Universe channels:

表 37. Ubuntu 20.04 Universe Channels - CLI

Ubuntu 20.04	
Universe Channel	ubuntu-2004-amd64-universe-uyuni
Universe Updates Channel	ubuntu-2004-amd64-universe-updates-uyuni
Universe Security Updates Channel	ubuntu-2004-amd64-universe-security-uyuni
Universe Backports Channel	ubuntu-2004-amd64-universe-backports-uyuni

手順: コマンドプロンプトからのソフトウェアチャンネルの追加

1. Uyuni サーバのコマンドプロンプトで root になり、 `spacewalk-common-channels` コマンドを特定のチャンネルに対して実行します:

```
spacewalk-common-channels \
<ベースチャンネルのラベル> \
<子チャンネル_1> \
<子チャンネル_2> \
... <子チャンネル_n>
```

2. チャンネルの同期:

```
spacewalk-repo-sync -p <base_channel_label>
```

3. 続行前に、同期が完了していることを確認してください。



Ubuntuクライアントをブートストラップする前に、新しいチャンネルはすべて完全に同期されている必要があります。

4.12.1.2. 同期ステータスの確認

プロシージャ: Web UIから同期の進捗状況を確認する

1. UyuniのWeb UIで、**ソフトウェア** > **管理** > **チャンネル**に移動し、リポジトリに関連付けられているチャ

ンネルをクリックします。

2. [リポジトリ] タブに移動し、[同期] をクリックし、[同期状態] をクリックします。

プロシージャ: コマンドプロンプトから同期の進捗状況を確認する

1. Uyuniサーバのコマンドプロンプトで、rootとして、`tail` コマンドを使用して同期ログファイルを確認します。

```
tail -f /var/log/rhn/reposync/<channel-label>.log
```

2. それぞれの子チャンネルは、同期の進捗中にそれぞれのログを生成します。同期が完了したことを確認するには、ベースチャンネルと子チャンネルのログファイルをすべて確認する必要があります。



Ubuntuチャンネルは非常に大きいことがあります。同期に数時間かかる場合があります。

4.12.1.3. クライアントでGPGキーを信頼する

4.12.1.4. クライアントでGPGキーを信頼する

Operating systems either trust their own GPG keys directly or at least ship them installed with the minimal system. But third party packages signed by a different GPG key need manual handling. The clients can be successfully bootstrapped without the GPG key being trusted. However, you cannot install new client tool packages or update them until the keys are trusted.

Salt clients use now GPG key information entered for a software channel to manage the trusted keys. When a software channel with GPG key information is assigned to a client, the key gets trusted as soon as the channel is refreshed or the first package gets installed from this channel.

The GPG key URL which is set of a software channel must exist. In case it is a file URL, the GPG key file must be deployed on the client before the software channel is used.

The GPG keys for the Client Tools Channels of Red Hat based clients are deployed on the client into `/etc/pki/rpm-gpg/` and can be referenced with file URLs. Same is the case with the GPG keys of Expanded Support clients. Only in case a software channel is assigned to the client they will be imported and trusted by the system.



Because Debian based systems sign only metadata, there is typically no need to specify extra keys for single channels. If a user configures an own GPG key to sign the metadata as described in "Use Your Own GPG Key" in **Administration > Repo-metadata** the deployment and trust of that key is executed automatically.

4.12.1.4.1. User defined GPG keys

Users can define their own GPG keys to be deployed to the client.

By providing some pillar data and providing the GPG key files in the Salt filesystem, they are automatically deployed to the client.

These keys are deployed into `/etc/pki/rpm-gpg/` on RPM based operating systems and to `/usr/share/keyrings/` on Debian systems:

Define the pillar key [literal `custom_gpgkeys`] for the client you want to deploy the key to and list the names of the key file.

```
cat /etc/pillar/mypillar.sls
custom_gpgkeys:
  - my_first_gpg.key
  - my_second_gpgkey.gpg
```

Additionally in the Salt filesystem create a directory named `gpg` and store there the GPG key files with the name specified in the `custom_gpgkeys` pillar data.

```
ls -la /etc/salt/gpg/
/etc/salt/gpg/my_first_gpg.key
/etc/salt/gpg/my_second_gpgkey.gpg
```

The keys are now deployed to the client at `/etc/pki/rpm-gpg/my_first_gpg.key` and `/etc/pki/rpm-gpg/my_second_gpgkey.gpg`.

The last step is to add the URL to the GPG key URL field of the software channel. Navigate to **Software > Manage > Channels** and select the channel you want to modify. Add to **GPG key URL** the value `file:///etc/pki/rpm-gpg/ my_first_gpg.key`.

4.12.1.4.2. GPG Keys in Bootstrap Scripts

プロシージャ: ブートストラップスクリプトを使用してクライアントでGPGキーを信頼する

- Uyuniサーバのコマンドプロンプトで、`/srv/www/htdocs/pub/`ディレクトリの内容を確認します。このディレクトリには、使用できるすべての公開鍵が含まれています。登録クライアントに割り当てるチャンネルに適用するキーをメモします。
- 関連するブートストラップスクリプトを開き、`ORG_GPG_KEY=`パラメータを見つけて、必要なキーを追加します。次に例を示します。

```
uyuni-gpg-pubkey-0d20833e.key
```

以前保存したキーを削除する必要はありません。



Trusting a GPG key is important for security on clients. It is the task of the admin to decide which keys are needed and can be trusted. Because a software channel cannot be used when the GPG key is not trusted, the decision of assigning a channel to a client depends on the decision of trusting the key.

4.12.1.5. rootアクセス

Ubuntuのrootユーザはデフォルトでは無効になっています。 **sudoers** ファイルを編集して有効にできます。

プロシージャ: rootユーザーアクセスの許可

1. クライアントで、 **sudoers** ファイルを編集します。

```
sudo visudo
```

この行を **sudoers** ファイルの末尾に追加して **sudo** アクセス権をユーザに付与します。 Web UIでクライアントをブートストラップしているユーザの名前で <user> を置き換えます。

```
<user>  ALL=NOPASSWD: /usr/bin/python, /usr/bin/python2,  
/usr/bin/python3
```



このプロシージャによりrootアクセス権が付与されます。クライアントの登録に必要なパスワードは不要です。 クライアントは正常にインストールされると、root特権で実行されるため、アクセス権は不要です。 クライアントを正しくインストールした後、 **sudoers** ファイルからこの行を削除することをお勧めします。

4.12.1.6. クライアントの登録

Ubuntuクライアントを登録するには、ブートストラップリポジトリが必要です。 デフォルトでは、ブートストラップリポジトリは自動的に作成され、すべての同期製品に対して毎日再生成されます。 次のコマンドを使用して、コマンドプロンプトからブートストラップリポジトリを手動で作成できます。

```
mgr-create-bootstrap-repo
```

クライアントの登録については、 [Client-configuration > Registration-overview](#)を参照してください。

4.12.2. Ubuntu 16.04および18.04クライアントの登録

このセクションでは、Ubuntu 16.04 LTS、18.04 LTSオペレーティングシステムを実行しているSaltクライアントの登録について説明します。

Uyuniは、Saltを使用するUbuntu 16.04 LTSおよび18.04 LTSクライアントをサポートしています。



Canonicalは、Uyuniを保証またはサポートしていません。



UbuntuはSaltクライアントでのみサポートされています。従来のクライアントはサポートされていません。

ブートストラップは、リポジトリの設定やプロファイルの更新の実行など、Ubuntuクライアントの起動および初期状態の実行のためにサポートされています。ただし、Ubuntuのrootユーザはデフォルトで無効になっているため、ブートストラップを使用するには、Pythonの `sudo` 特権がある既存ユーザが必要です。

4.12.2.1. ソフトウェアチャンネルの追加

UbuntuクライアントをUyuniサーバに登録する前に、必要なソフトウェアチャンネルを追加して同期する必要があります。



In the following section, descriptions often default to the `x86_64` architecture. Replace it with other architectures if appropriate.

このプロシージャで必要なチャンネルは次のとおりです。

表 38. Ubuntuチャンネル - CLI

OSバージョン	Ubuntu 16.04	Ubuntu 18.04
ベースチャンネル	<code>ubuntu-1604-pool-amd64-uyuni</code>	<code>ubuntu-1804-pool-amd64-uyuni</code>
メインチャンネル	<code>ubuntu-1604-amd64-main-uyuni</code>	<code>ubuntu-1804-amd64-main-uyuni</code>
更新チャンネル	<code>ubuntu-1604-amd64-updates-uyuni</code>	<code>ubuntu-1804-amd64-main-updates-uyuni</code>
セキュリティチャンネル	<code>ubuntu-1604-amd64-security-uyuni</code>	<code>ubuntu-1804-amd64-main-security-uyuni</code>
ユニバースチャンネル	<code>ubuntu-1604-amd64-universe-uyuni</code>	<code>ubuntu-1804-amd64-universe-uyuni</code>
ユニバース更新チャンネル	<code>ubuntu-1604-amd64-universe-updates-uyuni</code>	<code>ubuntu-1804-amd64-universe-updates-uyuni</code>
ユニバースセキュリティ更新チャンネル	<code>ubuntu-1604-amd64-universe-security-uyuni</code>	<code>ubuntu-1804-amd64-universe-security-uyuni</code>
クライアントチャンネル	<code>ubuntu-1604-amd64-uyuni-client</code>	<code>ubuntu-1804-amd64-uyuni-client</code>



Ubuntu 16.04はサポート終了になっており、そのリポジトリで提供されるISOイメージは失効しています。これらのパッケージを使用した新しいUbuntu 16.04クライアントのブートストラップは失敗します。新しいUbuntu 16.04クライアントをブートストラップする必要がある場合、[Client-configuration > Tshoot-clients](#)のトラブルシューティングプロシージャに従ってください。

手順: コマンドプロンプトからのソフトウェアチャンネルの追加

- Uyuni サーバのコマンドプロンプトで root になり、 `spacewalk-common-channels` コマンドを特定のチャンネルに対して実行します:

```
spacewalk-common-channels \
<ベースチャンネルのラベル> \
<子チャンネル_1> \
<子チャンネル_2> \
... <子チャンネル_n>
```

- チャンネルの同期:

```
spacewalk-repo-sync -p <base_channel_label>
```

- 続行前に、同期が完了していることを確認してください。



Ubuntuクライアントをブートストラップする前に、ユニバース(ユニバースにはSaltの重要な依存関係が含まれています)を含む新しいチャンネルはすべて完全に同期されている必要があります。

4.12.2.2. 同期ステータスの確認

プロシージャ: Web UIから同期の進捗状況を確認する

- UyuniのWeb UIで、**ソフトウェア** > **管理** > **チャンネル**に移動し、リポジトリに関連付けられているチャンネルをクリックします。
- [リポジトリ] タブに移動し、[同期] をクリックし、[同期状態] をクリックします。

プロシージャ: コマンドプロンプトから同期の進捗状況を確認する

- Uyuniサーバのコマンドプロンプトで、rootとして、`tail` コマンドを使用して同期ログファイルを確認します。

```
tail -f /var/log/rhn/reposync/<channel-label>.log
```

- それぞれの子チャンネルは、同期の進捗中にそれぞれのログを生成します。同期が完了したことを確認するには、ベースチャンネルと子チャンネルのログファイルをすべて確認する必要があります。



Ubuntuチャンネルは非常に大きいことがあります。同期に数時間かかる場合があります。

4.12.2.3. クライアントでGPGキーを信頼する

4.12.2.4. クライアントでGPGキーを信頼する

Operating systems either trust their own GPG keys directly or at least ship them installed with the minimal system. But third party packages signed by a different GPG key need manual handling. The clients can be successfully bootstrapped without the GPG key being trusted. However, you cannot install new client tool packages or update them until the keys are trusted.

Salt clients use now GPG key information entered for a software channel to manage the trusted keys. When a software channel with GPG key information is assigned to a client, the key gets trusted as soon as the channel is refreshed or the first package gets installed from this channel.

The GPG key URL which is set of a software channel must exist. In case it is a file URL, the GPG key file must be deployed on the client before the software channel is used.

The GPG keys for the Client Tools Channels of Red Hat based clients are deployed on the client into `/etc/pki/rpm-gpg/` and can be referenced with file URLs. Same is the case with the GPG keys of Expanded Support clients. Only in case a software channel is assigned to the client they will be imported and trusted by the system.



Because Debian based systems sign only metadata, there is typically no need to specify extra keys for single channels. If a user configures an own GPG key to sign the metadata as described in "Use Your Own GPG Key" in **Administration > Repo-metadata** the deployment and trust of that key is executed automatically.

4.12.2.4.1. User defined GPG keys

Users can define their own GPG keys to be deployed to the client.

By providing some pillar data and providing the GPG key files in the Salt filesystem, they are automatically deployed to the client.

These keys are deployed into `/etc/pki/rpm-gpg/` on RPM based operating systems and to `/usr/share/keyrings/` on Debian systems:

Define the pillar key [literal `custom_gpgkeys`] for the client you want to deploy the key to and list the names of the key file.

```
cat /etc/pillar/mypillar.sls
custom_gpgkeys:
  - my_first_gpg.key
  - my_second_gpgkey.gpg
```

Additionally in the Salt filesystem create a directory named `gpg` and store there the GPG key files with the name specified in the `custom_gpgkeys` pillar data.

```
ls -la /etc/salt/gpg/
/etc/salt/gpg/my_first_gpg.key
/etc/salt/gpg/my_second_gpgkey.gpg
```

The keys are now deployed to the client at `/etc/pki/rpm-gpg/my_first_gpg.key` and `/etc/pki/rpm-gpg/my_second_gpgkey.gpg`.

The last step is to add the URL to the GPG key URL field of the software channel. Navigate to **Software > Manage > Channels** and select the channel you want to modify. Add to **GPG key URL** the value `file:///etc/pki/rpm-gpg/my_first_gpg.key`.

4.12.2.4.2. GPG Keys in Bootstrap Scripts

プロシージャ: ブートストラップスクリプトを使用してクライアントでGPGキーを信頼する

1. Uyuniサーバのコマンドプロンプトで、`/srv/www/htdocs/pub/`ディレクトリの内容を確認します。このディレクトリには、使用できるすべての公開鍵が含まれています。登録クライアントに割り当てるチャンネルに適用するキーをメモします。
2. 関連するブートストラップスクリプトを開き、`ORG_GPG_KEY=`パラメータを見つけて、必要なキーを追加します。次に例を示します。

```
uyuni-gpg-pubkey-0d20833e.key
```

以前保存したキーを削除する必要はありません。



Trusting a GPG key is important for security on clients. It is the task of the admin to decide which keys are needed and can be trusted. Because a software channel cannot be used when the GPG key is not trusted, the decision of assigning a channel to a client depends on the decision of trusting the key.

4.12.2.5. rootアクセス

Ubuntuのrootユーザはデフォルトでは無効になっています。 `sudoers` ファイルを編集して有効にできます。

プロシージャ: rootユーザーアクセスの許可

1. クライアントで、`sudoers` ファイルを編集します。

```
sudo visudo
```

この行を `sudoers` ファイルの末尾に追加して `sudo` アクセス権をユーザに付与します。 Web UIでクライアントをブートストラップしているユーザの名前で `<user>` を置き換えます。

```
<user> ALL=NOPASSWD: /usr/bin/python, /usr/bin/python2,
/usr/bin/python3
```



このプロセッサによりrootアクセス権が付与されます。クライアントの登録に必要なパスワードは不要です。クライアントは正常にインストールされると、root特権で実行されるため、アクセス権は不要です。クライアントを正しくインストールした後、**sudoers** ファイルからこの行を削除することをお勧めします。

4.12.2.6. クライアントの登録

Ubuntuクライアントを登録するには、ブートストラップリポジトリが必要です。デフォルトでは、ブートストラップリポジトリは自動的に作成され、すべての同期製品に対して毎日再生成されます。次のコマンドを使用して、コマンドプロンプトからブートストラップリポジトリを作成できます。

```
mgr-create-bootstrap-repo
```

クライアントの登録については、[Client-configuration > Registration-overview](#)を参照してください。

4.13. クライアントをプロキシに登録する

プロキシサーバは、Saltおよび従来のクライアントの両方のためにブローカおよびパッケージキャッシュとして動作できます。クライアントをプロキシに登録する動作は、クライアントをUyuniサーバに直接登録する動作に似ていますが、いくつかの相違点があります。

Web UI、コマンドラインにおけるコマンド、またはブートストラップスクリプトを使用してSaltクライアントをプロキシに登録するための情報が次の各セクションに記載されています。また、ブートストラップスクリプトを使用して従来のクライアントを登録するための情報も含まれています。クライアントのあるUyuniプロキシから別のプロキシまたはUyuniサーバに移動する方法もあります。

Web UI内では、Saltクライアントと従来のクライアントの両方に関する情報をプロキシページに示します。システム > システム一覧 > プロキシでプロキシの名前をクリックしてプロキシに接続するクライアントの一覧を表示し、[詳 紹] タブの [プ ロ キ シ] サブタブを選択できます。

システム > すべてでクライアントの名前をクリックしてSaltクライアントのチェーンされたプロキシの一覧を表示し、[詳 紹] タブの [接 続] サブタブを選択できます。

4.13.1. プロキシ間でのクライアントの移動

登録プロセスを繰り返すことなく、SaltおよびSaltSSHプッシュクライアントをプロキシ間で移動することができます。



従来のクライアントをプロキシ間で移動する場合、最初から登録プロセスを繰り返す必要があります。

手順: プロキシ間でSaltまたはSalt SSHプッシュクライアントを移動する

1. UyuniのWeb UIで、プロキシ間で移動するクライアントの [システムの詳細] ページに移動します。
2. [接続] タブに移動します。次に [プロキシの変更] リンクをたどって、ドロップダウンメニューを表示します。
3. [新しいプロキシ] ドロップダウンメニューから、クライアントの移動先のプロキシを選択し、[プロキシの変更] をクリックします。

手順: SSMで複数のSaltまたはSalt SSHプッシュクライアントをプロキシ間で移動する

1. UyuniのWeb UIで、システム > システム一覧に移動し、移動するそれぞれのクライアントを確認します。クライアントがシステムセットマネージャに追加されます。
2. システム > システムセットマネージャに移動し、[その他 > プロキシ] タブに移動します。
3. [新しいプロキシ] ドロップダウンメニューから、クライアントの移動先のプロキシを選択し、[プロキシの変更] をクリックします。

`system.changeProxy` API呼び出しでも同じ機能を利用できます。

4.13.1.1. 背景情報

この機能の効果は、通常のSaltクライアントとSalt SSHプッシュクライアントで異なります。

4.13.1.1.1. 通常のSaltクライアント

この機能は、Salt状態アクションをスケジュールします。これにより、`susemanager.conf` Saltクライアント設定ファイルの `master:` 設定が新しいプロキシを指すように変更されます。次に、この機能はSaltクライアントを再起動します。



`susemanager.conf` ファイルを手動で編集して `master:` を変更しても同じ効果があり、この方法もサポートされています。

minionが再起動して新しいプロキシ経由で再接続すると、サーバはデータベース内のプロキシパスを更新し、チャンネルURLを更新するための別のアクションをスケジュールします。

4.13.1.1.2. Salt SSHプッシュクライアント

この機能はデータベース内のプロキシパスをただちに更新し、チャンネルURLを更新するための新しいアクションがスケジュールされます。

4.13.2. プロキシからサーバへのクライアントの移動

Saltクライアントをプロキシからサーバに移動する場合は、プロキシリストから `なし` を選択します。

従来のクライアントをサーバに移動する場合、最初から登録プロセスを繰り返す必要があります。

4.13.3. Web UIを使用してクライアントをプロキシに登録する

Web UIを使用してSaltクライアントをUyuniプロキシに登録できます。



SLE以外のクライアント全般およびバージョン15より前のバージョンのSLEクライアントではブートストラップリポジトリが必要です。 ブートストラップリポジトリには、クライアントにSaltをインストールするためのパッケージ、Saltまたは従来のクライアントを登録するためのパッケージが用意されています。 ブートストラップリポジトリの作成については、[Client-configuration > Bootstrap-repository](#)を参照してください。

プロシージャ: Web UIを使用してクライアントをプロキシに登録する

1. UyuniのWeb UIで、**システム > ブートストラップ**に移動します。
2. [ホスト] フィールドに、ブートストラップするクライアントの完全修飾ドメイン名(FQDN)を入力します。
3. [SSHポート] フィールドに、クライアントを接続してブートストラップするために使用するSSHポート番号を入力します。 デフォルトでは、SSHポートは**22**です。
4. [ユーザ] フィールドに、クライアントにログインするユーザ名を入力します。 デフォルトでは、ユーザ名は**root**です。
5. [Authentication Method] (認証メソッド) フィールドで、クライアントのブートストラップに使用する認証メソッドを選択します。
 - パスワード認証の場合、[パスワード] フィールドに、パスワードを入力してクライアントにログインします。
 - SSH秘密鍵認証の場合、秘密鍵と関連パスフレーズを入力します。 ブートストラッププロセスの実行が完了するまでの間のみ、この鍵は保存されます。
6. [アクティベーションキー] フィールドで、クライアントのブートストラップに使用するソフトウェアチャンネルに関連付けられているアクティベーションキーを選択します。
7. [プロキシ] フィールドで、登録先にするプロキシサーバを選択します。
8. デフォルトでは、[Disable SSH Strict Key Host Checking] (SSH厳格キーホストの確認を無効にする) チェックボックスにチェックが付いています。 このチェックボックスにチェックが付いていると、ブートストラッププロセスは、手動認証なしでSSHホストキーを自動的に受け入れます。
9. オプション: [Manage System Completely via SSH] (SSHでシステムを完全に管理する) チェックボックスにチェックを付けます。 このオプションにチェックを付けると、サーバへの接続にSSHを使用するようにクライアントは設定され、その他の接続方法は設定されません。
10. [ブート] をクリックして、登録を開始します。

ブートストラッププロセスが完了したら、クライアントは [システム > システム一覧] にリストされます。

4.13.3.1. コマンドラインで登録する(Salt)

Web UIの代わりに、コマンドラインを使用して、Saltクライアントをプロキシに登録できます。 このプロシージャでは、登録する前にSaltパッケージをSaltクライアントにインストール済みである必要があります。

SLE 12ベースのクライアントでは、Advanced Systems Managementモジュールもアクティブ化しておく必要があります。



コマンドラインで従来のクライアントを登録することもできますが、その手順は長くなります。この手順についてはここでは説明しません。ブートストラップスクリプトプロセッサを使用して従来のクライアントを登録します。詳細については、[client-proxy-script.pdf](#)を参照してください。

プロセッサ: コマンドラインを使用してクライアントをプロキシに登録する

1. 次の場所にあるクライアント設定ファイルを選択します。

```
/etc/salt/minion
```

または

```
/etc/salt/minion.d/NAME.conf
```

これはminionファイルと呼ばれることもあります。

2. プロキシFQDNをマスターとしてクライアント設定ファイルに追加します。

```
master: PROXY123.EXAMPLE.COM
```

3. salt-minionサービスを再起動します。

```
systemctl restart salt-minion
```

4. サーバで、新しいクライアントキーを受け入れます。<client>をクライアントの名前に置き換えます。

```
salt-key -a '<client>'
```

4.13.4. ブートストラップスクリプトを使用して登録する(Saltと従来版)

ブートストラップスクリプトを使用してUyuniを介してSaltクライアントおよび従来のクライアントを登録できます。これは、Uyuniサーバでクライアントを登録する方法とほぼ同じです。相違点は、コマンドラインツールを使用してUyuniプロキシにブートストラップスクリプトを作成することです。その後、ブートストラップスクリプトは、必要な情報をクライアントにすべて展開します。ブートストラップスクリプトには、アクティベーションキーまたはGPGキーなどのパラメータが必要です。これらのパラメータはそれぞれの設定によって決まります。

プロシージャ: ブートストラップスクリプトを使用してクライアントをプロキシに登録する

1. Web UIを使用してUyuniサーバにクライアントアクティベーションキーを作成します。 詳細については、[Client-configuration > Activation-keys](#)を参照してください。
2. プロキシで、**mgr-bootstrap** コマンドラインツールをrootとして実行します。必要に応じて、追加のコマンドラインスイッチを使用して。ブートストラップスクリプトを調整します。 Saltクライアントではなく従来のクライアントをインストールするには、**--traditional** スイッチを使用してください。

使用できるオプションを表示するには、コマンドラインに **mgr-bootstrap --help** と入力します。

```
mgr-bootstrap --activation-keys=key-string
```

3. オプション: 結果として得られたブートストラップスクリプトを編集します。
4. クライアントで直接ブートストラップスクリプトを実行するか、または **ssh** を使用してプロキシからブートストラップスクリプトを実行します。 **<bootstrap>** をブートストラップスクリプトの名前に置き換え、**<client.example.com>** をクライアントのホスト名に置き換えます。

```
cat <bootstrap> | ssh root@<client.example.com> /bin/bash
```

4.14. パブリッククラウドでのクライアントの登録

Uyuniサーバを設定すると、クライアントの登録を開始できます。

4.14.1. 製品の追加とリポジトリの同期

クライアントに対応する製品をすでに追加し、リポジトリをUyuniに同期していることを確認してください。これは、クライアントの登録に使用されるブートストラップリポジトリを作成するために必要です。

For more information, see [installation-and-upgrade:pubcloud-setup.pdf](#).

4.14.2. オンデマンドイメージの準備

SUSEによって提供されるオンデマンドイメージから起動するインスタンスは自動的に登録され、更新されたインフラストラクチャおよびSUSE Linux Enterpriseモジュールはアクティビ化されます。 Uyuniクライアントとしてオンデマンドイメージを使用するには、使用を始める前にこの自動化を無効にする必要があります。

プロシージャ: オンデマンドイメージの準備

1. オンデマンドインスタンスにログインします。
2. コマンドプロンプトでrootとして、登録データとリポジトリを削除します。

```
registercloudguest --clean
```

3. **cloud-regionsrv-client** パッケージを削除します。

```
zypper rm cloud-regionsrv-client
```

4. 使用しているクラウドプロバイダ固有の追加パッケージを削除します。

- Amazon EC2:

```
zypper rm regionServiceClientConfigEC2 regionServiceCertsEC2
```

- Google Compute Engine:

```
zypper rm cloud-regionsrv-client-plugin-gce
regionServiceClientConfigGCE regionServiceCertsGCE
```

- Microsoft Azure:

```
zypper rm regionServiceClientConfigAzure regionServiceCertsAzure
```

UyuniをSUSE Customer Centerに登録する手順については、[Installation-and-upgrade > Server-setup](#)を参照してください。

4.14.3. クライアントの登録

UyuniのWeb ブラウザに移動し、[ホスト]、[SSH ポート]、[ユーザ]、および [パスワード] の各フィールドに [ホスト] フィールドで安定版FQDNを使用していることを確認してください。別の有効期間が短いFQDNをパブリッククラウドで使用している場合、Uyuniではホストを検索できません。



従来のクライアントをブートストラップしようとしている場合、クライアントにログインしている間にサーバのホスト名を解決できることを確認してください。サーバのFQDNをクライアントの `/etc/hosts` ローカル解決ファイルに追加する必要があります。サーバのローカルIPアドレスで `hostname -f` コマンドを使用していることを確認してください。

パブリッククラウドのイメージでは通常、ユーザ名とパスワードでSSHにログインできません。証明書でのみSSHにログインできます。Web UIからブートストラップを使用する場合、ユーザ名とSSHキーによるSSHへのログインを有効にする必要があります。この操作を実行するには、[システム > ブートストラップ](#)に移動し、認証メソッドを変更します。

クラウドプロバイダがMicrosoft Azureの場合、ユーザ名とパスワードでログインできます。この操作を実行するには、AzureUserがrootとしてパスワードなしでコマンドを実行できる必要があります。この操作を実

行するには、`/etc/sudoers.d/waagent` ファイルを開き、次の行を追加または編集します。

```
AzureUser ALL=(ALL) NOPASSWD: ALL
```



AzureUserがrootとしてパスワードなしでコマンドを実行できると、セキュリティ上のリスクが生じます。この方法の使用はテストのみにしてください。運用システムでは実行しないでください。

ブートストラッププロセスが正常に完了したら、クライアントは [システム > システム一覧] にリストされます。

- プロセスをより詳細に制御したい場合、多数のクライアントを登録する必要がある場合、または従来のクライアントを登録している場合、ブートストラップスクリプトを作成します。 詳細については、[Client-configuration > Registration-bootstrap](#)を参照してください。
- Saltクライアントで、さらに詳細にプロセスを制御するには、コマンド行でsingleコマンドを実行すると便利です。 詳細については、[Client-configuration > Registration-cli](#)を参照してください。
- パブリッククラウドイメージ(AWS AMIなど)から起動されたクライアントを登録する場合、追加の設定をして、相互に上書きしないようにする必要があります。 複製を登録する方法の詳細については、[Administration > Tshoot-registerclones](#)を参照してください。

4.14.4. アクティベーションキー

アクティベーションキーは従来のクライアントとSaltクライアントで使用し、クライアントが正しいソフトウェアのエンタイトルメントを持ち、適切なチャンネルに接続して関連グループに加入するようします。 それぞれのアクティベーションキーは、キーを作成するときに設定できる組織にひもづけされます。

アクティベーションキーの詳細については、[Client-configuration > Activation-keys](#)を参照してください。

4.14.5. Terraformによって作成されたクライアントの自動登録

Terraformによって作成された新しいクライアントはUyuniに自動的に登録できます。 次の2つの登録方法があります。

- cloud-initベースの登録
- リモート実行プロビジョナーベースの登録

4.14.5.1. cloud-initベースのクライアントの登録

新しく作成された仮想マシンを自動的に登録するには、cloud-initを活用して登録することをお勧めします。 このソリューションでは、ホストへのSSH接続を設定する必要はありません。 また、クライアントの作成に使用するツールに関係なく使用することができます。

ユーザは、マシンをUyuniに自動的に登録するために、Terraformを使用してイメージを展開するときにユーザデータのセットを渡すことができます。`user_data` はブートストラップ時にマシンを初めて起動したとき

のみ1回だけ実行されます。

cloud-initを使用してクライアントを登録する前にユーザは以下を設定する必要があります。

- ・ ブートストラップスクリプト。Client-configuration > Registration-bootstrapを参照してください
- ・ アクティベーションキー。Client-configuration > Activation-keysを参照してください

次のコマンドを実行すると、ブートストラップスクリプトがダウンロードされ、新しいマシン作成時にそのマシンが登録されます。これをcloud-initの設定に追加する必要があります。

```
curl -s http://hub-server.tf.local/pub/bootstrap/bootstrap-default.sh |  
bash -s
```



user_data が更新されてプロビジョニングが変更されると必ず、Terraformはそのマシンを破棄してから新しいIPなどを使用して再作成します。

AWSのcloud-initの詳細については、次のURLを参照してください。 . https://registry.terraform.io/providers/hashicorp/template/latest/docs/data-sources/cloudinit_config

cloud-initの例については、次のURLを参照してください。 . https://registry.terraform.io/providers/hashicorp/cloudinit/latest/docs/data-sources/cloudinit_config#example-usage

4.14.5.2. リモート実行プロビジョナーベースの登録

新しく作成した仮想マシンの自動登録の2番目のソリューションではTerraformのリモート実行プロビジョナーを使用します。

リモート実行プロビジョナーは新しく作成したマシンとやり取りします。 これは、SSH接続を開き、そのマシンでコマンドを実行できます。



リモート実行プロビジョナーを使用してクライアントを登録するとき、ユーザは、Terraformを実行しているマシンが、新しい仮想マシン作成後にそのマシンにアクセスできることを確認する必要があります。

その他の要件は、[cloud-initベースのクライアントの登録]と同じです

- ・ ブートストラップスクリプト。Client-configuration > Registration-bootstrapを参照してください
- ・ アクティベーションキー。Client-configuration > Activation-keysを参照してください

このコマンドを実行すると、ブートストラップスクリプトがダウンロードされ、新しいマシン作成時にそのマシンが登録されます。これは、実行するリモートコマンドとして定義する必要があります。

```
curl -s http://hub-server.tf.local/pub/bootstrap/bootstrap-default.sh |  
bash -s
```

リモート実行プロビジョナーの詳細については、<https://www.terraform.io/docs/provisioners/remote-exec.html>を参照してください。

Chapter 5. クライアントのアップグレード

クライアントは、基盤となるオペレーティングシステムのバージョン設定システムを使用し、定期的なアップグレードが必要です。

SUSEオペレーティングシステムを使用するSCC登録クライアントの場合、UyuniのWeb UI内でアップグレードを実行できます。サポートされているSUSE Linux Enterprise 15のアップグレードパスは、<https://documentation.suse.com/sles/15-SP3/html/SLES-all/cha-upgrade-paths.html>を参照してください。

SLE 12を実行しているクライアントをSLE 15にアップグレードするには、アップグレードは自動化されていますが、アップグレードを始める前に準備手順を実行する必要があります。 詳細については、**Client-configuration > Client-upgrades-major**を参照してください。

コンテンツライフサイクルマネージャを使用してクライアントのアップグレードを自動化することもできます。 詳細については、**Client-configuration > Client-upgrades-lifecycle**を参照してください。

サービスパックのアップグレード、openSUSE Leapのマイナーバージョンのアップグレード、openSUSE LeapからSUSE Linux Enterpriseへの移行などの製品の移行の詳細については、**Client-configuration > Client-upgrades-product-migration**を参照してください。

登録を取り消したopenSUSE Leapクライアントのアップグレードの詳細については、**Client-configuration > Client-upgrades-uyuni**を参照してください。

5.1. クライアント - メジャーバージョンのアップグレード

クライアントには、インストールされているオペレーティングシステムで利用できる最新のサービスパック(SP)があり、最新の更新がすべて適用されている必要があります。 システムが最新でありすべての更新が正しくインストールされていることを開始前に確認してください。

アップグレードは、YaSTおよびAutoYaSTによって制御されます。Zypperは使用しません。

5.1.1. マイグレーションの準備

クライアントをSLE 12からSLE 15 に移行する前に、次の作業を行う必要があります。

1. インストールメディアの準備
2. 自動インストールのディストリビューションの作成
3. アクティベーションキーの作成
4. AutoYaSTプロファイルのアップロード

プロシージャ: インストールメディア(SLE 15 SP2など)の準備

1. Uyuniサーバで、SLE 15 SP2インストールメディア用にローカルディレクトリを作成します。

```
mkdir -p /srv/images/sle15sp2
```

- インストールソースでISOイメージをダウンロードし、ISOイメージをサーバにマウントします。

```
mount -o loop DVD1.iso /mnt/
```

- マウントしたISOからローカルファイルシステムにすべてコピーします。

```
cp -r /mnt/* /srv/images/sle15sp2
```

- コピーが完了したら、ISOイメージをアンマウントします。

```
umount /mnt
```



This image is the Unified Installer and can be used for multiple autoinstallable distributions.

プロシージャ: 自動インストール可能なディストリビューションの作成

- UyuniのWeb UIで、[自動インストール可能なディストリビューション]に移動し、[自動インストール可能なディストリビューションの作成]をクリックします。
- [自動インストール可能なディストリビューションの作成]セクションで、
 - [ディストリビューションラベル]セクションに、ディストリビューションの固有の名前を入力します。半角の英字、数字、ハイフン、ピリオド、および下線のみを使用し、5文字以上にします。たとえば、`sles15sp2-x86_64`です。
 - [ツリーパス]フィールドに、インストールソースへの絶対パスを入力します。たとえば、`/srv/images/sle15sp2`です。
 - [ベースチャンネル]フィールドで、`SLE-Product-SLES15-SP2-Pool for x86_64`を選択します。
 - [インストーラ生成]フィールドで、[SUSE Linux Enterprise 15]を選択します。
 - [カーネルオプション]フィールドに、インストールでブート時にカーネルに渡すオプションを入力します。`install=`パラメータおよび`self_update=0 pt.options=self_update`パラメータはデフォルトで追加されます。
 - インストールしたシステムを初めてブートするときにカーネルに渡すオプションを [カーネルの後のオプション]セクションに入力します。
- [自動インストール可能なディストリビューションの作成]をクリックして作成します。

古いSLE 12 ベースチャンネルから新しいSLE 15チャンネルに切り替えるには、アクティベーションキーが必要です。

プロシージャ: アクティベーションキーの作成

1. UyuniサーバのWeb UIで、**システム** > **アクティベーションキー**に移動し、**[キーの作成]**をクリックします。
2. キーの説明を入力します。
3. キーを入力するか空白のままにし、自動キーを生成します。
4. オプション: 使用量を制限する場合、**[使用量]**テキストフィールドに値を入力します。
5. **SLE-Product-SLES15-SP2-Pool for x86_64**ベースチャンネルを選択します。
6. オプション: **[付属エンタイトメント]**を選択します。 詳細について
は、<https://documentation.suse.com/sles/15-SP3/html/SLES-all/article-modules.html>を参照してください。
7. **[アクティベーションキーの作成]**をクリックします。
8. **[子チャンネル]**タブをクリックし、必要なチャンネルを選択します。
9. **[キーの更新]**をクリックします。

5.1.2. 自動インストールプロファイルの作成

自動インストールプロファイルには、システムをインストールするために必要なインストールデータおよび設定データがすべて含まれています。インストール完了後に実行するスクリプトを含めることもできます。手始めに使用できるスクリプトのサンプルは、<https://github.com/SUSE/manager-build-profiles/tree/master/AutoYaST>を参照してください。

有効なAutoYaSTアップグレード設定については、

<https://doc.opensuse.org/projects/autoyast/#CreateProfile-upgrade>を参照してください。

プロシージャ: 自動インストールプロファイルの作成

1. UyuniのWeb UIで、**システム** > **自動インストール** > **プロファイル**に移動し、自動インストールプロファイルのスクリプトをアップロードします。

手始めに使用できるスクリプトのサンプルは、

<https://github.com/SUSE/manager-build-profiles/tree/master/AutoYaST>を参照してください。

2. **[カーネルオプション]**フィールドに **autoupgrade=1**と入力します。

Y2DEBUG=1オプションを含めることもできます。デバッグ設定は不要ですが、問題が発生したときの調査に役立ちます。



Azureクラウドで実行されているクライアントは、**[カーネルオプション]**に **textmode=1 console=ttyS0**を追加する必要があります。

3. 自動インストールプロファイルを貼り付けるか、またはファイルアップロードフィールドを使用します。

4. [作成] をクリックして保存します。

5. アップロードしたプロファイルで変数を設定する必要がある場合、**システム** > **自動インストール** > プロファイルに移動し、編集するプロファイルを選択し、[変数] タブに移動します。

次のフォーマットを使用して必要な変数を指定します。

```
<key>=<value>
```

5.1.3. 移行

自動インストールプロファイルで参照するチャンネルがすべて使用可能で完全に同期していることを開始前に確認してください。

`/var/log/rhn/reposync/<channel-label>.log` でミラーリングの進捗状況を監視できます。

プロシージャ: 移行

1. UyuniサーバのWeb UIで、[システム] に移動し、アップグレードするクライアントを選択します。
2. [プロビジョニング] タブに移動し、アップロードした自動インストールプロファイルを選択します。
3. [自動インストール] をスケジュールして[読み込み] に必要なタスクを追加します。
ファイルがダウンロードされ、ブートローダのエントリが変更され、再起動され、アップグレードが開始されます。

クライアントは、Uyuniサーバと次に同期するときに、再インストールジョブを受け取ります。再インストールジョブは、新しいカーネルパッケージおよびinitrdパッケージをフェッチします。また、新しいカーネルパッケージおよびinitrdパッケージへのポイントを含む新しい `/boot/grub/menu.lst` (GRUB Legacy) または `/boot/grub2/grub.cfg` (GRUB 2) を書き込みます。

クライアントが次にブートするとき、grubを使用して、新しいカーネルとそのinitrdをブートします。PXEブートはこのプロセス中に使用されません。

ジョブがフェッチされた約3分後に、クライアントは再起動するためにシャットダウンします。



Saltクライアントでは、移行が完了した後、`spacewalk/minion_script` スニペットを使用してクライアントを再登録します。

5.2. コンテンツライフサイクルマネージャを使用したアップグレード

管理するSUSE Linux Enterprise Serverクライアントが多数ある場合、コンテンツライフサイクルマネージャを使用してインプレースアップグレードを自動化できます。

5.2.1. アップグレードの準備

クライアントをアップグレードするには、その前に次の準備を行う必要があります。

- ・ コンテンツライフサイクルプロジェクトの作成
- ・ アクティベーションキーの作成
- ・ 自動インストールのディストリビューションの作成
- ・ 自動インストールプロファイルの作成

プロシージャ: コンテンツライフサイクルプロジェクトの作成

1. ディストリビューション用のコンテンツライフサイクルプロジェクトを作成します。

詳細については、[Administration > Content-lifecycle](#)を参照してください。

2. プロジェクトの名前は、短いがわかりやすい名前にします。
3. ディストリビューションに必要なソースチャンネルモジュールをすべて含めます。
4. 必要に応じてフィルタを追加し、1つ以上の環境を設定します。

プロシージャ: アクティベーションキーの作成

1. ディストリビューション用のアクティベーションキーを作成します。

詳細については、[Client-configuration > Activation-keys](#)を参照してください。

2. フィルタされたプロジェクトチャンネルのすべてがアクティベーションキーに含まれていることを確認します。

プロシージャ: 自動インストール可能なディストリビューションの作成

1. 移行するベースチャンネルごとに自動インストールのディストリビューションを作成します。

詳細については、[Client-configuration > Autoinst-distributions. adoc](#)を参照してください。

2. コンテンツライフサイクルプロジェクトの名前を表すディストリビューションラベルを付けます。
3. [インストーラ生成] フィールドで、使用しているSLESのバージョンを選択します。

プロシージャ: 自動インストールプロファイルの作成

1. アップグレードするディストリビューションおよびサービスパックごとに自動インストールプロファイルを作成します。

詳細については、[Client-configuration > Autoinst-profiles](#)を参照してください。

2. Saltクライアントと従来のクライアントには異なるプロファイルを使用する必要があります。
3. プロファイルで変数を使用して、異なるライフサイクル環境を区別できます。

自動インストールプロファイルのサンプルについては、<https://github.com/SUSE/manager-build-profiles/>

[tree/master/AutoYaST](#)を参照してください。

インプレースアップグレードを自動化するための自動インストールプロファイルでこれらの次の変数を使用します。

リスト 1. 例: 自動インストールプロファイルで使用する変数

```
registration_key=1-15sp1-demo-test
org=1
channel_prefix=15sp1-demo-test
distro_label=15sp1-demo-test
```

リスト 2. 例: 自動インストールプロファイルで使用するエントリ

```
<listentry>
  <ask_on_error config:type="boolean">true</ask_on_error>

  <media_url>https://$redhat_management_server/ks/dist/child/$channel_prefix-sle-module-web-scripting15-sp1-pool-x86_64/$distro_label</media_url>
    <name>$channel_prefix SLE-Module-Web-Scripting15-SP1 Pool for x86_64
  </name>
    <product>Web Scripting Module 15 SP1 x86_64 Pool</product>
  </listentry>
```

5.2.2. アップグレード

サーバをアップグレードする準備ができたら、クライアントをプロビジョニングできます。

プロシージャ: クライアントのプロビジョニング

1. UyuniのWeb UIで、**システム** > **システム一覧**に移動し、プロビジョニングするクライアントを選択してシステムセットマネージャに追加します。
2. **システム** > **システムセットマネージャ** > **概要**に移動し、[プロビジョニング] タブをクリックします。
3. 使用する自動インストールプロファイルを選択します。

PXEを使用できるクライアントでは、そのクライアントをプロビジョニングするとすぐに移行が自動化されます。 その他のすべてのクライアントでは、Cobblerを使用してアップグレードを実行できます。

プロシージャ: Cobblerを使用してクライアントをアップグレードする

1. コマンドプロンプトで、rootとして、利用できるCobblerプロファイルを確認します。

```
cobbler profile list
```

2. 選択したプロファイルおよびディストリビューションでISOファイルを構築します。

```
cobbler buildiso --iso=/tmp/SLE_15-sp1.iso --profiles=SLE_15
-sp1:1:Example --distro=SLE_15-sp1
```

CD-ROMを使用したクライアントのプロビジョニングの詳細については、[Client-configuration > Autoinst-cdrom](#)を参照してください。

5.3. 製品移行

製品を移行すると、SLEベースのクライアントシステムをサービスパック(SP)レベルから最新版にアップグレードできます。たとえば、SUSE Linux Enterprise 15 SP1をSUSE Linux Enterprise 15 SP2に移行できます。

openSUSE Leapを新しいマイナーバージョンまたは対応するSLE SPレベルに移行することもできます。次に例を示します。

- openSUSE Leap 15.1から15.2、または
- openSUSE Leap 15.1からSUSE Linux Enterprise 15 SP1、または
- openSUSE Leap 15.4 to SUSE Linux Enterprise 15 SP4



移行中、Uyuniは、インストール前に必要なライセンス(EULA)を自動的に受け入れます。

SUSE Linux Enterprise 12以降では、SUSE Customer Centerがサービスパックを提供している場合、SUSEはサービスパックのスキップをサポートしています。たとえば、SUSE Linux Enterprise 15からSP2にアップグレードできます。SP1はインストールされません。サポートされているSUSE Linux Enterpriseのアップグレードパスは、<https://documentation.suse.com/sles/15-SP3/html/SLES-all/cha-upgrade-paths.html#sec-upgrade-paths-supported>を参照してください。



製品の移行は、同じメジャーバージョン内でアップグレードするためのものです。SUSE Linux Enterprise 12からSUSE Linux Enterprise 15への移行には、製品の移行は使用できません。メジャーアップグレードの詳細については、[Client-configuration > Client-upgrades-major](#)を参照してください。



製品の移行にはロールバック機能はありません。移行プロシージャが始まると、ロールバックできません。万一に備えて、作業システムのバックアップを用意してください。

5.3.1. 移行の実行

製品の移行を開始する前に、保留中の更新やパッチがないことを確認してください。クライアントシステム概要詳細の「システムステータス」を確認し、提供されているすべての更新またはパッチをインストールします。クライアントシステムが最新でない場合、製品移行が失敗する可能性があります。



移行を開始する前に、ターゲット製品のすべてのチャンネルが完全に同期されていることを確認してください。Web UIで同期ステータスを確認するには、[管理 > セットアップウィザード > 製品](#)ページに移動します。

手順: 移行の実行

1. [システム > 概要](#)ページからクライアントを選択します。
2. クライアントのシステム詳細ページから、[ソフトウェア > 製品移行](#)タブに移動します。
3. ターゲットの移行パスを選択し、**[チャンネルの選択]**をクリックします。

行 - チャンネルページから、正しいベースチャンネルを選択し、**必須の子チャンネル**を含めます。

[**ベンダー変更を許可する**]にチェックを付け、ベンダを変更したパッケージをインストールできるようにします。チェックを付けると、移行の開始前に詳細を示す通知が表示されます。



openSUSE Enterpriseに移行するには、[**ベンダー変更を許可する**]オプションにチェックを付ける必要があります。

6. チャンネルを正しく設定したら**[移行のスケジュール]**をクリックします。

5.3.2. 製品の大量移行

多数のクライアントを次のSPバージョンに移行する場合、Uyuni APIコールを使用できます。



製品の大量移行操作は危険です。システムを誤ってアップグレードしないように注意してください。プロセスは徹底的にテストする必要があります。少なくとも、最初に予行演習を行ってください。

spacecmd コマンドラインツールは、**system_scheduleproductmigration** サブコマンドを提供します。このコマンドを使用して、多数のクライアントの次のマイナーバージョンへの移行をスケジュールできます。

system_scheduleproductmigration の構文の使用方法とオプションを表示するには、次のコマンドを実行します。

```
spacecmd system_scheduleproductmigration help
```

5.3.2.1. 製品の大量移行の実行

手順: 製品の大量移行の実行

1. 実行可能な移行ターゲットをリストし、移行するシステムIDをメモします。

```
spacecmd api -- system.listMigrationTargets -A 1000010001
```

2. それぞれのシステムIDに対して、`listMigrationTarget`を呼び出し、目的の製品が使用可能であることを確認します。
 - システムIDに使用可能なターゲットがある場合は、`system.scheduleProductMigration`を呼び出します。
 - 目的のターゲットを使用できない場合、そのシステムをスキップします。

次のテンプレートを環境に合わせます。

```
target = '[....]'
basechannel = 'channel-label'
system_ids = [1, 2, 3]

session = auth.login(user, pass)
for system in system_ids
  if system.listMigrationTargets(session, system).ident == target
    system.scheduleProductMigration(session, system, target, basechannel,
    [], False, <now>)
  else
    print "Cannot migrate to requested target -- skipping system"
  endif
endfor
```

5.3.2.2. 製品の大量移行の例: SLES 15 SP2からSLES 15 SP3

この例では、大量移行を容易にするためにグループが一時的に作成されます。

プロシージャ: 製品の大量移行グループの作成

1. UyuniのWeb UIで、システム > システムグループに移動し、[グループの作成]をクリックします。
2. グループに `mpm-target-sles15sp3` という名前を付けます。

同じベースチャンネルにサブスクライブしているシステムのみを、作成したグループに追加する必要があります。この例では、`SLE-Product-SLES15-SP2-Poolforx86_64` にサブスクライブされているシステムのみがグループに追加されました。



この時点でアップグレードする予定のないシステムは、グループから削除する必要があります。

プロシージャ: グループへのシステムの追加

1. グループへのクライアントの追加の詳細については、[client-configuration:system-groups.pdf](#)を参照してください。

The `spacecmd` sub-commands `system_scheduleproductmigration` and `system_listmigrationtargets` are looping over all systems that are part of the group. If there are 100 systems in the group, you will see 100 actions scheduled. It is important that all systems in the group support the same "migration target."

+ 次のコマンドを実行すると、グループ内のすべてのシステムのターゲットが取得されます。

+

```
spacecmd -- system_listmigrationtargets group:mpm-target-sles15sp3
```

+ Only select a target, which is reported for **all** systems. This command output a string of "IDs." The string is the identifier for the `MIGRATIONTARGET` of the other command.

プロシージャ: 大量移行コマンドの実行

- この例では、グループ `mpm-target-sles15sp3` 内のすべてのシステムをSLES12 SP2からSLES15 SP1にアップグレードするには、コマンドラインで次のように入力します。

```
spacecmd -- system_scheduleproductmigration group:mpm-target-sles15sp3 \
          sle-product-sles15-sp3-pool-x86_64 [190,203,195,1242] -d
```

`system_scheduleproductmigration` コマンドの構文は次のとおりです。

```
spacecmd -- system_scheduleproductmigration <SYSTEM> <BASE_CHANNEL_LABEL>
          \
          <MIGRATION_TARGET> [options]
```

詳細については、`spacecmd -- system_scheduleproductmigration help` を参照してください。

5.3.2.3. 必須の構文

<SYSTEM>

この例では、作成したグループを使用して、そのグループからすべてのシステムを選択します。

```
group:mpm-target-sles15sp3
```

<BASE_CHANNEL_LABEL>

これは、ターゲットベースチャンネルのラベルです。この場合、システムはSLES 15 SP3にアップグレードされており、ラベルは **sle-product-sles15-sp3-pool-x86_64** です。

現在ミラーリングされているすべてのベースチャンネルのリストを表示するには、次のコマンドを実行します。

```
spacecmd softwarechannel_listbasechannels.
```

現在のベースチャンネルで使用可能なターゲットでない限り、チャンネルにアップグレードできないことに注意してください。

<MIGRATION_TARGET>

グループ **group:mpm-target-sles15sp3** 内のシステムのこの値を特定するには、次のコマンドを実行します。

```
spacecmd -- system_listmigrationtargets group:mpm-target-sles15sp3
```

MIGRATION_TARGET パラメータは、次の形式で渡す必要があります。

```
[190,203,195,1242]
```

オプション

1. -s (開始時刻)
2. -d 予行演習モード (実際には処理を行わない) (実際の移行の前に予行演習を行うことを推奨)
3. -c (子チャンネル) (カンマ区切りで子チャンネルのラベルを指定する。空白は使用しないこと)

この場合、「-d」オプションが含まれています。このオプションは予行演習が成功した後に削除できます。

成功した場合は、スケジュールされたシステムごとに次のコマンド出力が表示されます。

1. システムmpm-sles152-1の製品移行のスケジューリング
2. スケジュールされたアクションID: 66

グループ内の特定のシステムのWeb UIで、アクション (この場合は予行演習) を追跡することもできます。クライアントのシステム詳細ページから、**イベント > 履歴** に移動します。予行演習中に障害が発生した場合は、システムを調査する必要があります。

すべて問題がなければ、コマンドから「-d」オプションを削除して、実際の移行を実行できます。移行が完了したら、Uyuni Web UIからシステムを再起動できます。

5.4. Uyuniクライアントのアップグレード

このセクションでは、例としてopenSUSE Leapを使用します。

5.4.1. アップグレードの準備

プロシージャ: クライアントのアップグレードの準備

- Uyuniサーバのコマンドプロンプトでrootになり、**spacewalk-common-channels**コマンドを特定のチャンネルに対して実行します。

```
spacewalk-common-channels \
opensuse_leap15_4\
opensuse_leap15_4-non-oss \
opensuse_leap15_4-non-oss-updates \
opensuse_leap15_4-updates \
opensuse_leap15_4-uyuni-client
```

- spacewalk-repo-sync**を使用して、すべてのチャンネルを完全に同期します。リポジトリのURLが定義済みの場合、[installation-and-upgrade:proxy-uyuni.pdf](#)で続行します。
- UyuniサーバのWeb UIで、**ソフトウェア > 管理 > チャンネル**に移動し、**Uyuni Client Tools for openSUSE Leap 15.4 (x86_64)**チャンネル名をクリックします。
- 右上隅の**[チャンネルの管理]**をクリックします。
- [リポジトリ]タブをクリックし、**External - Uyuni Client Tools for openSUSE Leap 15.3 (x86_64)**を選択します。
- [Update Repositories]**（リポジトリの更新）をクリックします。
- リポジトリ > 同期**サブタブに移動し、**[今すぐ同期]**をクリックします。
- openSUSE Leap 15.4 (x86_64)**および**External - openSUSE Leap 15.3 (x86_64)**で同じ操作をします。

openSUSE Leap 15.4 (x86_64)を展開し、パッケージが入力されているすべての子チャンネルを表示します。

5.4.2. アップグレード

クライアントをアップグレードするには、ソフトウェアリポジトリを置き換え、ソフトウェアを更新し、最後にクライアントを再起動します。

プロシージャ: クライアントのアップグレード

- UyuniサーバのWeb UIで、[システム]に移動し、クライアントの名前をクリックします。
[ソフトウェアチャネル]をクリックし、**カスタムチャネル**一覧にリストされているopenSUSE Leap 15.4チャンネルをベースチャンネルとして選択し、**子チャネル**ペインで、15.4子チャンネルを選択します。
[ソフトウェアチャネルの変更確認]をクリックしてから、**[確認]**をクリックします。
- ソフトウェア > パッケージ > アップグレード**をクリックし、クライアントで更新するパッケージをすべて選択します。

択してから、選択を適[パッケージのアップグレード]をクリックし、詳細を確認し、[確認]をクリックして更新を終了します。

+

+

. クライアントを再起動します。

多数のクライアントを更新する必要がある場合、Uyuniサーバでこのコマンドシーケンスのアクションチェーンを作成できます。アクションチェーンを使用して、複数のクライアントで同時に更新を実行できます。

Chapter 6. クライアントの削除

Uyuniサーバからクライアントを削除する必要がある場合、Web UIを使用して削除できます。このプロシージャは、従来のクライアントとSaltクライアントの両方で動作します。

プロシージャ: クライアントの削除

1. UyuniのWeb UIで、[システム > システム一覧](#)に移動し、削除するクライアントを選択します。
2. **[システムの削除]**をクリックします。
3. 詳細を確認し、**[プロファイルの削除]**をクリックして確認します。
4. Saltクライアントの場合、Uyuniは、追加の設定をクリーンアップしようとします。クライアントに接続できない場合、削除をキャンセルするオプションや、設定ファイルをクリーンアップせずにクライアントを削除するオプションがあります。

システムセットマネージャを使用して複数のクライアントを削除することもできます。システムセットマネージャの詳細については、[Client-configuration > System-set-manager](#)を参照してください。



従来のクライアントを削除した後にこれを自動的にクリーンアップすることはできません。手動で行う必要があります。さらに、Salt minionをクリーンアップしてもSalt自体は削除されません。



通常、従来のクライアントを削除せずにこれをSalt minionに移行します。Saltは、従来のクライアントがあることを自動検出し、必要な変更を行います。ただし、従来のクライアントを削除済みで、これをSalt minionとして再度登録する場合、[Installation-and-upgrade > Troubleshooting](#)を参照してください。

Chapter 7. クライアントの操作

クライアントの登録、アップグレード、または削除に加えて、その他の操作を実行できます。

Uyuniクライアントは、システムセットマネージャ、システムグループまたは設定管理を使用して、個別に管理またはグループに編成できます。

SUSE Manager Web UIを使用してカスタムシステム情報の取得、設定のスナップショットの管理、電源オン、電源オフ、クライアントの再起動を実行できます。

このセクションでは、これらの操作それについて詳しく説明します。

7.1. パッケージ管理

クライアントは、パッケージを使用してソフトウェアをインストール、アンインストール、およびアップグレードします。

クライアントでパッケージを管理するには、[システム]に移動し、管理するクライアントをクリックし、**システム > ソフトウェア > パッケージ**サブタブに移動します。 このセクションで使用できるオプションは、選択したクライアントのタイプ、および現在のチャンネルのサブスクリプションによって異なります。



- パッケージをインストールまたはアップグレードするとき、ライセンスまたはEULAは自動的に受け入れられます。

ほとんどのパッケージ管理アクションは、アクションチェーンに追加できます。 アクションチェーンの詳細については、**Reference > Schedule**を参照してください。

7.1.1. パッケージの検証

クライアントにインストールしたパッケージがインストール元のデータベースの現在の状態と一致していることを確認できます。 インストールしたパッケージのメタデータが、ファイルのチェックサム、ファイルサイズ、パーミッション、オーナー、グループ、タイプなど、データベースの情報と比較されます。

プロシージャ: インストールしたパッケージの検証

1. UyuniのWeb UIで、[システム]に移動し、パッケージをインストールしたクライアントをクリックし、**システム > ソフトウェア > パッケージ** > 検証サブタブに移動します。
2. 検証するパッケージを選択し、[選択したパッケージの検証]をクリックします。
3. 検証を完了したら、**システム > イベント > 履歴**に移動し、結果を表示します。

7.1.2. パッケージの比較

保存されているプロファイルでクライアントにインストールされたパッケージを比較できます。 または別のクライアントにインストールされたパッケージと比較できます。 比較を実行するとき、選択したクライアントを一致するように変更できます。

パッケージをプロファイルと比較するには、プロファイルを保存済みである必要があります。プロファイルは、現在インストールされているクライアントのパッケージから作成されます。プロファイルを作成したら使用して、同じインストール済みパッケージで別のクライアントをインストールできます。

プロシージャ: 保存されているプロファイルの作成

1. UyuniのWeb UIで、[システム]に移動し、プロファイルのベースになっているクライアントをクリックし、システム > ソフトウェア > パッケージ > プロファイルサブタブに移動します。
2. [システムプロファイルの作成]をクリックします。
3. プロファイルの名前と説明を入力し、[プロファイルの作成]をクリックします。

プロシージャ: クライアントパッケージの比較

1. UyuniのWeb UIで、[システム]に移動し、比較するクライアントをクリックし、システム > パッケージ > プロファイルサブタブに移動します。保存されているプロファイルを比較するには、プロファイルを選択し、[比較]をクリックします。
2. 別のクライアントと比較するには、クライアント名を選択し、[比較]をクリックし、2つのクライアントの差異一覧を表示します。
3. 選択したクライアントにインストールするパッケージを確認し、削除するパッケージのチェックを外し、[パッケージを同期]をクリックします。

7.2. パッチ管理

組織内でカスタムパッチを使用してクライアントを管理できます。この方法では、カスタムチャンネルのパッケージのパッチ警告の発行、パッチインストールのスケジュール、組織間のパッチの管理を実行できます。

7.2.1. パッチの作成

カスタムパッチを使用するには、パッチを作成し、これにパッケージを追加し、1つまたは複数のチャンネルに追加する必要があります。

プロシージャ: カスタムパッチの作成

1. UyuniのWeb UIで、パッチ > パッチの管理に移動し、[パッチの作成]をクリックします。
2. [パッチの作成]セクションで、次の詳細を使用します。
 - [概要] フィールドにパッチの短い説明を入力します。
 - [アドバイザリ] フィールドにパッチのラベルを入力。組織の命名規則を使用して、パッチの管理を簡単にすることをお勧めします。
 - [アドバイザリリース] フィールドにパッチのリリース番号を入力します。パッチの最初のバージョンの場合、1を使用します。
 - [アドバイザリタイプ] フィールドで、使用するパッチのタイプを選択します。修正するパッチには「バグ修正アドバイザリ」を選択します。
 - [セキュリティアドバイザリ] のアドバイザリタイプを選択した場合、使用するセキュリティレベルを「アドバイザリの重要度」フィールドで選択します。

- [製品] フィールドに、パッチが参照する製品の名前を入力します。
 - オプション: [Author] (作成者) フィールドにパッチの作成者の名前を入力します。
 - [トピック]、[説明]、[ソリューション] の各フィールドにパッチに関する追加情報を入力します。
3. オプション: [バグ] セクションで、次の詳細を使用して関連するバグの情報を指定します。
- [ID] フィールドにバグ番号を入力します。
 - [概要] フィールドにバグの短い説明を入力します。
 - [Bugzilla URL] フィールドにバグのアドレスを入力します。
 - [キーワード] フィールドにバグに関するキーワードを入力。複数のキーワードの間ではカンマを使用してください。
 - [リファレンス]、[メモ] の各フィールドにバグに関する追加情報を入力します。
 - 新しいパッチを追加するチャンネルを1つまたは複数選択します。
4. [パッチの作成] をクリックします。

既存のパッチを複製してパッチを作成することもできます。複製では、パッケージの関連づけが保持され、パッチの発行が簡素化されます。

プロシージャ: パッチの複製

1. UyuniのWeb UIで、**パッチ > パッチの複製**に移動します。
2. [潜在的に適用できる可能性のあるパッチを表示] フィールドで、複製するパッチのネルを選択します。
3. 複製する1つまたは複数のパッチを選択し、[パッチの複製] をクリックします。
4. 複製したパッチを追加するチャンネルを1つまたは複数選択します。
5. 詳細を確認し、複製を開始します。

パッチを作成したら、そのパッチにパッケージを割り当てることができます。

プロシージャ: パッチにパッケージを割り当てる

1. UyuniのWeb UIで、**パッチ > パッチの管理**に移動し、パッチのアドバイザリ名をクリックしてパッチの詳細を表示します。
2. **パッケージ > 追加タブ**に移動します。
3. [チャネル] フィールドで、パッチに割り当てるパッケージが含まれているソフトウェアチャンネルを選択し、[パッケージの表示] をクリックします。[managed packages] (管理しているすべてのパッケージ) を選択して、すべてのチャンネルで使用できるパッケージを表示できます。
4. 含めるパッケージを確認し、[パッケージの追加] をクリックします。
5. パッケージの詳細を確認し、[確認] をクリックしてパッチに適用します。
6. **パッケージ > 一覧表示/削除タブ**に移動し、パッケージが正しく割り当てられていることを確認します。

パッケージをパッチに割り当てると、パッチキャッシュが更新され、変更が反映されます。キャッシュの更新には数分かかる場合があります。

既存のパッチの詳細を変更する必要がある場合、[パッチ管理] ページから実行できます。

プロシージャ: 既存のパッチ警告の編集および削除

1. UyuniのWeb UIで、パッチ > パッチの管理に移動します。
2. パッチのアドバイザリ名をクリックし、パッチの詳細を表示します。
3. 必要に応じて変更し、[パッチの更新] をクリックします。
4. パッチを削除するには、[パッチ管理] ページでパッチを選択し、[パッチの削除] をクリックします。パッチの削除には、数分かかる場合があります。

7.2.2. クライアントへのパッチの適用

パッチの用意ができたら、クライアントに適用できます。その際、単独または他のパッチと一緒に適用できます。

パッチ内の各パッケージは、1つまたは複数のチャンネルの一部です。クライアントがチャンネルにサブスクライブされていない場合、更新はインストールされません。

対象の更新より新しいバージョンのパッケージがクライアントにすでにインストールされている場合、その更新はインストールされません。古いバージョンのパッケージがクライアントにインストールされている場合、そのパッケージはアップグレードされます。

プロシージャ: 適用可能なすべてのパッチを適用する

1. UyuniのWeb UIで、システム > 概要に移動し、更新するクライアントを選択します。
2. ソフトウェア > パッチタブに移動します。
3. [すべてを選択] をクリックして適用可能なすべてのパッチを選択します。
4. [パッチの適用] をクリックしてクライアントを更新します。

管理者特権でサインインしている場合、クライアントに対してより大規模なバッチアップグレードも実行できます。

プロシージャ: 1つのパッチを複数のクライアントに適用する

1. UyuniのWeb UIで、パッチ > パッチリストに移動します。
2. 適用するパッチを見つけ、そのパッチのシステム列で番号をクリックします。
3. パッチの適用先にするクライアントを選択し、[パッチの適用] をクリックします。
4. Confirm the list of clients to perform the update.

プロシージャ: 複数のパッチを複数のクライアントに適用する

1. UyuniのWeb UIで、システム > 概要に移動し、更新するクライアントにチェックを付け、システムセットマネージャに追加します。

2. システム > システムセットマネージャに移動し、[パッチ] タブに移動します。
3. クライアントに適用するパッチを選択し、[パッチの適用] をクリックします。
4. 更新する日時をスケジュールし、[確認] をクリックします。
5. 更新の進捗を確認するには、スケジュール > 待機中の動作に移動します。



スケジュールされたパッケージ更新は、各クライアントで設定された接続メソッドを使用してインストールされます。 詳細については、Client-configuration > Contact-methods-intro を参照してください。

7.3. システムのロック

システムのロックは、クライアントでアクションが発生しないようにするために使用されます。 たとえば、システムのロックは、クライアントの更新または再起動が行われないようにします。 これは、運用ソフトウェアを実行しているクライアントに対して、または不注意による変更が行われないようにするために役に立ちます。 アクションを実行する準備ができたときにシステムのロックを無効にできます。

システムのロックは、従来のクライアントおよびSaltクライアントでは異なる方法で実装されています。

7.3.1. 従来のクライアントのシステムのロック

従来のクライアントがロックされると、Web UIを使用してアクションをスケジュールすることができず、システム > システム一覧にあるクライアントの名前の横に南京錠のアイコンが表示されます。

プロシージャ: 従来のクライアントのシステムのロック

1. UyuniのWeb UIで、ロックするクライアントの [システムの詳細] ページに移動します。
2. [ロックの状態] で [Lock this system] (このシステムをロックする) をクリックします。このクライアントは、[Unlock this system] (このシステムのロックを解除する) をクリックするまでロックされたままになります。

ロックされた従来のクライアントでは、リモートコマンド、自動化されているパッチの更新など、一部のアクションは実行されないでいるパッチの更新を停止するには、クライアントの [システムの詳細] ページに移動し、[プロパティ] タブで [自動パッチ更新] のチェックを外します。

7.3.2. Saltクライアントのシステムのロック

Saltクライアントがロックされている場合、または停止モードになっている場合、アクションをスケジュールできず、Salt実行コマンドが無効になり、黄色のバナーが [システムの詳細] ページに表示されます。このモードでは、Web UIまたはAPIを使用してロックされているクライアントのアクションをスケジュールできますが、アクションは失敗します。



ロックメカニズムはSalt SSHクライアントでは使用できません。

プロシージャ: Saltクライアントのシステムのロック

1. UyuniのWeb UIで、ロックするクライアントの [システムの詳細] ページに移動します。
2. [式] タブに移動し、システムのロックの式にチェックを付け、[保存] をクリックします。
3. 式 > **System Lock** (システムのロック) タブに移動し、[Lock system] (システムのロック) にチェックを付け、[保存] をクリックします。このページでは、クライアントがロックされているときに特定のSaltモジュールを有効にすることもできます。
4. 変更した場合、highstateを適用する必要がある場合があります。この場合、Web UIのバナーで通知されます。システムのロック式を削除するまで、クライアントはロックされたままです。

Saltの停止モードの詳細については、<https://docs.saltstack.com/en/latest/topics/blackout/index.html> を参照してください。

7.3.3. パッケージのロック

パッケージのロックは複数のクライアントで使用できますが、さまざまな機能セットを使用できます。次のものを区別する必要があります。



1. SUSE Linux EnterpriseおよびopenSUSE (zyppベース)とRed Hat Enterprise LinuxまたはDebian クライアント、
2. 従来のクライアントとSalt クライアント。

7.3.3.1. Zyppベースのシステムでのパッケージのロック



Zypperパッケージマネージャを備えたシステムでは、従来のクライアントとSaltクライアントでパッケージのロックを使用できます。

パッケージのロックを使用して、ソフトウェアパッケージの未認可インストールや未認可アップグレードを防止します。パッケージがロックされている場合、インストールできないことを示す南京錠のアイコンが表示されます。ロックされているパッケージをインストールしようとすると、イベントログにエラーとしてレポートされます。

ロックされているパッケージは、インストール、アップグレード、または削除できません。UyuniのWeb UIを使用しても、パッケージマネージャを使用してクライアントマシンで直接操作しても同様です。ロックされているパッケージは、依存関係のあるパッケージも間接的にロックします。

プロシージャ: パッケージのロックの使用

1. 管理対象システムでソフトウェア > パッケージ > ロックタブに移動し、使用できるパッケージの一覧を表示します。
2. ロックするパッケージを選択し、[Request Lock] (ロックのリクエスト) をクリックします。ロックをアクティブ化する日時を選択します。デフォルトでは、できるだけ早くロックをアクティブ化します。ロックはすぐにはアクティブにできないことに注意してください。
3. パッケージのロックを外すには、ロック解除するパッケージを選択し、[Request Unlock] (ロック解除のリクエスト) をクリックします。ロックをアクティブ化する場合と同様に、日時を選択します。

7.3.3.2. Red Hat Enterprise LinuxやDebianのようなシステムでのパッケージのロック



一部のRed Hat Enterprise LinuxやDebianのようなシステムでは、Saltクライアントでパッケージのロックを使用できます。

Red Hat Enterprise LinuxやDebianのようなシステムでは、パッケージのロックは、ソフトウェアパッケージの未認可アップグレードや削除を防止するためにのみ使用されます。パッケージがロックされている場合、変更できないことを示す南京錠のアイコンが表示されます。ロックされているパッケージを変更しようとすると、イベントログにエラーとしてレポートされます。

ロックされているパッケージは、アップグレードまたは削除できません。UyuniのWeb UIを使用しても、パッケージマネージャを使用してクライアントマシンで直接操作しても同様です。ロックされているパッケージは、依存関係のあるパッケージも間接的にロックします。

プロシージャ: パッケージのロックの使用

1. Red Hat Enterprise Linux 7システムでは、`yum-plugin-versionlock` パッケージを `root` としてインストールします。Red Hat Enterprise Linux 8システムでは、`python3-dnf-plugin-versionlock` パッケージを `root` としてインストールします。Debianシステムでは、`apt` ツールにロック機能が含まれています。
2. 管理対象システムでソフトウェア > パッケージ > ロックタブに移動し、使用できるパッケージの一覧を表示します。
3. ロックするパッケージを選択し、**[Request Lock]** (ロックのリクエスト) をクリックします。ロックをアクティブ化する日時を選択します。デフォルトでは、できるだけ早くロックをアクティブ化します。ロックはすぐにはアクティブにできないことに注意してください。
4. パッケージのロックを外すには、ロック解除するパッケージを選択し、**[Request Unlock]** (ロック解除のリクエスト) をクリックします。ロックをアクティブ化する場合と同様に、日時を選択します。

7.4. 設定管理

クライアントそれぞれを手動で設定するのではなく、設定ファイルおよびチャンネルを使用してクライアントの設定を管理できます。

設定パラメータは、スクリプト化され、設定ファイルに保存されます。UyuniのWeb UIを使用して設定ファイルを直接書き込むことができます。または、別の場所にあるファイルをアップロードまたはリンクできます。

設定ファイルは一元管理またはローカルで管理できます。一元管理された設定ファイルは、グローバル設定チャンネルで提供され、Uyuniサーバにサブスクライブされるクライアントに適用できます。ローカル管理された設定ファイルは、一元管理された設定ファイルを上書きするために使用されます。このファイルは、設定管理特権がないUyuniユーザにとって特に便利ですが、管理しているクライアントを変更する必要があります。

設定チャンネルは、設定ファイルの編成に使用されます。クライアントを設定チャンネルにサブスクライブして、必要に応じて設定ファイルを展開できます。

設定ファイルはバージョン管理されるため、設定を追加し、クライアントで設定をテストし、必要に応じて

前のリビジョンにロールバックできます。 設定チャンネルを作成したら、さまざまな設定ファイル間の比較や同じ設定ファイルの異なるリビジョン間の比較も実行できます。

設定ファイルは一元管理またはローカルで管理できます。 一元管理された設定ファイルはグローバル設定チャンネルによって提供されます。 ローカル管理された設定ファイルはUyuniに直接作成またはアップロードされます。

使用できる設定管理機能はSaltクライアントと従来のクライアントで異なります。 次の表は、それぞれのクライアントタイプでサポートされる機能を示しています。

表 39. 設定管理でサポートされる機能

機能	Salt	従来
グローバル設定チャンネル	✓	✓
ファイルの展開	✓	✓
ファイルの比較	?	✓
ローカル管理ファイル	✗	✓
サンドボックスファイル	✗	✓
Highstateの適用	✓	✗
クライアントからのファイルのインポート	✗	✓
設定マクロ	✗	✓

7.4.1. 設定管理用に従来のクライアントを準備する

従来のクライアントでは、設定管理を使用するために追加の準備操作が必要です。 AutoYaSTまたはKickstartを使用して従来のクライアントをインストールした場合、おそらく適切なパッケージがすでに準備されています。 その他の従来のクライアントでは、そのクライアントのオペレーティングシステム用に関連するツールの子チャンネルがインストールされていることを確認します。 ソフトウェアチャンネルの詳細については、[Client-configuration > Channels](#)を参照してください。

必要なパッケージは次のとおりです。

- **mgr-cfg**: すべての **mgr-cfg-*** パッケージで必要なベースライブラリおよび関数
- **mgr-cfg-actions**: Uyuniを使用してスケジュールされた設定アクションを実行するために必要です。
- **mgr-cfg-client**: 設定管理システムのクライアントの機能へのコマンドラインインターフェースを提供します。
- **mgr-cfg-management**: Uyuni設定を管理するためのコマンドラインインターフェースを提供します。

システム > アクティベーションキーに移動し、ブートストラップ中に使用するアクティベーションキーをクリックし、[設定ファイルの配備] オプションにチェックを付けることによってブートストラッププロセス中にこれらのパッケージをインストールできます。 アクティベーションキーの詳細については、[Client-configuration > Activation-keys](#)を参照してください。

7.4.2. 設定チャンネルの作成

新しいセントラル設定チャンネルを作成するには:

プロシージャ: セントラル設定チャンネルの作成

1. UyuniのWeb UIで、**設定 > チャンネル**に移動し、**[設定チャンネルの作成]**をクリックします。
2. チャンネルの名前を入力します。
3. チャンネルのラベルを入力します。 このフィールドには、半角の英字、数字、ハイフン(-)、および下線(_)のみを含める必要があります。
4. 他のチャンネルから区別できるようにチャンネルの説明を入力します。
5. **[設定チャンネルの作成]**をクリックして新しいチャンネルを作成します。

設定チャンネルを使用して、SaltクライアントのSaltの状態を管理することもできます。

プロシージャ: Saltの状態チャンネルの作成

1. UyuniのWeb UIで、**設定 > チャンネル**に移動し、**[状態チャンネルの作成]**をクリックします。
2. チャンネルの名前を入力します。
3. チャンネルのラベルを入力します。 このフィールドには、半角の英字、数字、ハイフン(-)、および下線(_)のみを含める必要があります。
4. 他のチャンネルから区別できるようにチャンネルの説明を入力します。
5. `init.sls` ファイルの **[SLS コンテンツ]** を入力します。
6. **[設定チャンネルの作成]**をクリックして新しいチャンネルを作成します。

7.4.3. 設定ファイル、ディレクトリ、またはシンボリックリンクの追加

設定チャンネルを作成済みの場合、設定ファイル、ディレクトリ、またはシンボリックリンクを追加できます。

プロシージャ: 設定ファイル、ディレクトリ、またはシンボリックリンクの追加

1. UyuniのWeb UIで、**設定 > チャンネル**に移動し、設定ファイルに追加する設定チャンネルの名前をクリックし、**ファイルの追加 > ファイルの作成**サブタブに移動します。
2. **[ファイルタイプ]** フィールドで、テキストファイル、ディレクトリ、またはシンボリックリンクを作成するかどうかを選択します。
3. **[ファイル名/パス]** フィールドで、ファイルを展開する場所への絶対パスを入力します。
4. シンボリックリンクを作成している場合、**[シンボリックリンクの目的ファイル名/パス]** フィールドで、ターゲットのファイルおよびパスを入力します。
5. **[所有権]** フィールドおよび**[ファイルアクセス許可モード]** にファイルの**[ユーザ名/グループ]**を入力します。
6. クライアントでSELinuxが有効になっている場合、**[SELinux contexts]** (SELinuxコンテキスト) を設定して、必要なファイル属性(ユーザ、ロール、ファイルタイプなど)を有効にできます。

7. 設定ファイルにマクロが含まれている場合、マクロの先頭および末尾をマークする記号を入力します。
8. [ファイルの内容] テキストボックスで設定ファイルの内容を入力し、スクリプトドロップダウンボックスを使用して、適切なスクリプト言語を選択します。
9. [設定ファイルの作成] をクリックします。

7.4.4. クライアントを設定チャンネルにサブスクライブする

個々のクライアントを設定チャンネルにサブスクライブできます。そのためには、**システム > システム一覧** に移動し、サブスクライブするクライアントを選択し、[設定] タブに移動します複数のクライアントを設定チャンネルにサブスクライブするには、システムセットマネージャ(SSM)を使用できます。

プロシージャ: 複数のクライアントを設定チャンネルにサブスクライブする

1. UyuniのWeb UIで、**システム > システム一覧** に移動し、操作するクライアントを選択します。
2. **システム > システムセットマネージャ** に移動し、**設定 > チャンネルにサブスクライブ** サブタブに移動し、使用できる設定チャンネルの一覧を表示します。

[現在サブスクライブしているシステム] 列で番号をクリックして、設定チャンネルに現在サブスクライブされているクライアントを表示します。

4. サブスクライブ先の設定チャンネルを確認し、[続行] をクリックします。
5. 上下矢印を使用して設定チャンネルをランクします。 設定の競合が設定チャンネルで発生した場合、一覧の上の方にあるチャンネルが優先されます。
6. 選択したクライアントにチャンネルを適用する順位でサブスクライブをクリックして、現在サブスクライブしているチャンネルより低い優先度で新しいチャンネルを追加します。 最も高い順位でサブスクライブをクリックして、現在サブスクライブしているチャンネルより高い優先度で新しい既存のサブスクライブを置換をクリックして、既存のチャンネルを削除し、新しいチャンネルに置き換えます。
7. [サブスクライブの適用] をクリックします。



新しい設定チャンネルの優先度が既存のチャンネルと競合する場合、重複チャンネルが削除され、新しい優先度に応じて置き換えられます。 クライアントの設定優先度をアクションで順序変更する場合、Web UIでは続行する前に変更を確認する必要があります。

7.4.5. 設定ファイルの比較

システムセットマネージャ(SSM)を使用して、Uyuniサーバに保存されている設定ファイルを使用してクライアントに展開された設定ファイルを比較することもできます。

プロシージャ: 設定ファイルの比較

1. UyuniのWeb UIで、**システム > システム一覧** に移動して、比較する設定ファイルにサブスクライブされているクライアントを選択します。
2. **システム > システムセットマネージャ** に移動し、**設定 > ファイルの比較** サブタブに移動し、使用できる設定チャンネルの一覧を表示します。

3. オプション [システム] 列で番号をクリックして、設定ファイルに現在サブスクライブされているクライアントを表示します。
4. 比較する設定ファイルを確認し、[ファイル|アイドル|の比較|をスケジュール] をクリックします。

7.4.6. 従来のクライアントにおける設定ファイルのマクロ

1つのファイルを保存して同じ設定を共有できることは便利ですが、同じ設定ファイルの多数のバリエーションまたはシステム固有の詳細(ホスト名やMACアドレスなど)のみが異なる設定ファイルが必要になる場合があります。この場合、設定ファイル内でマクロまたは変数を使用できます。マクロまたは変数を使用すると、数百さらには数千のバリエーションを含む単一のファイルをアップロードし、配布することができます。カスタムシステム情報用の変数に加えて、次の標準マクロがサポートされています。

```
rhn.system.sid
rhn.system.profile_name
rhn.system.description
rhn.system.hostname
rhn.system.ip_address
rhn.system.custom_info(key_name)
rhn.system.net_interface.ip_address(eth_device)
rhn.system.net_interface.netmask(eth_device)
rhn.system.net_interface.broadcast(eth_device)
rhn.system.net_interface.hardware_address(eth_device)
rhn.system.net_interface.driver_module(eth_device)
```

この機能を使用するには、[設定チャネルの詳細] ページで設定ファイルをアップロードまたは作成しました。[設定チャネルの詳細] ページを開き、選択したサポート対象のマクロを含め、並べて記述するために使用するデリミタは、[マクロ開始デリミタ] フィールドと [マクロ終了デリミタ] フィールドで設定したデリミタと必ず一致させ、ファイル内の他の文字と競合しないようにしてください。デリミタは長さを2文字とし、パーセント(%)記号を含めないことをお勧めします。

たとえば、IPアドレスとホスト名のみが異なるすべてのサーバに適用される1つのファイルがあるとします。サーバごとに別の設定ファイルを管理する代わりに、`server.conf`などの単一のファイルを作成して、IPアドレスとホスト名のマクロを含めることができます。

```
hostname={| rhn.system.hostname |}
ip_address={| rhn.system.net_interface.ip_address(eth0) |}
```

Uyuni Web UIのスケジュールされたアクションまたはUyuni設定クライアント(`mgrcfg-client`)のコマンドラインのいずれかを通じて、そのファイルを個々のシステムに配布する場合、変数は、Uyuniのシステムプロファイルに記録されているシステムのホスト名とIPアドレスに置き換えられます。この例では、展開されるバージョンは次のようにになります。

```
hostname=test.example.domain.com
ip_address=177.18.54.7
```

カスタムシステム情報を取得するには、カスタム情報マクロ(`rhn.system.custom_info`)にキーラベルを挿入します。たとえば、「asset」というラベルを付けたキーを作成した場合、設定ファイルのカスタム情報マクロにそのキーラベルを追加して、そのキーラベルを含むシステムを値に代入することができます。このマクロは次のようにになります。

```
asset=@ rhn.system.custom_info(asset) @
```

そのキーの値を含むシステムにこのファイルが展開されると、マクロが変換され、次のような文字列が生成されます。

```
asset=Example#456
```

(エラーを防ぐために必要な場合などに)デフォルト値を含めるには、カスタム情報マクロに次のようにデフォルト値を付加することができます。

```
asset=@ rhn.system.custom_info(asset) = 'Asset #' @
```

値を含むシステムでは、このデフォルトはその値によって上書きされます。

システム管理を支援するために、Uyuniクライアントコンピュータ上でUyuni設定マネージャ(`mgrcfg-manager`)を利用できます。このツールは、システムに依存しないため、ファイルを変換または変更しません。`mgrcfg-manager`コマンドは、システム設定に依存しません。バイナリファイルを変更することはできません。

7.5. 電源管理

UyuniのWeb UIを使用して、電源オン、電源オフ、およびクライアントの再起動を実行できます。

この機能は、IPMIまたはRedfishプロトコルを使用し、Cobblerプロファイルを使用して管理されます。選択したクライアントには、これらのプロトコルのいずれかをサポートしている電源管理コントローラがある必要があります。

Redfishの場合、クライアントとUyuniサーバの間に有効なSSL接続を確立できることを確認してください。Redfish管理コントローラのSSLサーバ証明書を署名するために使用される認証局を信頼している必要があります。CA証明書は`.pem`フォーマットで、Uyuniサーバの`/etc/pki/trust/anchors/`に保存される必要があります。証明書を保存したら、`update-ca-certificate`を実行します。

プロシージャ: 電源管理を有効にする

1. UyuniのWeb UIで、**システム** > **システム一覧**に移動し、管理するクライアントを選択し、**プロビジョニ**

ング > 電源管理タブに移動します。

2. [タイプ] フィールドで、使用する電源管理プロトコルを選択します。
3. 電源管理サーバの詳細を入力し、適切なボタンをクリックしてアクションを実行し、[保存のみ] をクリックし、アクションを実行せずに詳細を保存します。

電源管理アクションを複数のクライアントに同時に適用できます。そのためには、クライアントをシステムセットマネージャに追加します。 システムセットマネージャの使用法の詳細については、Client-configuration > System-set-manager を参照してください。

7.5.1. 電源管理とCobbler

電源管理機能を初めて使用するとき、Cobblerシステムレコードが自動的に作成されます(まだクライアントに存在しない場合)。自動作成されたシステムレコードは、ネットワークから起動できず、ダミーのシステムイメージへの参照が含まれています。 Cobblerがプロファイルまたはイメージのないシステムレコードを現時点でサポートしていないため、この参照は必要です。

Cobbler電源管理は、フェンスエージェントツールを使用して、IPMI以外のプロトコルをサポートしています。 Uyuniでは、IPMIプロトコルとRedfishプロトコルのみがサポートされています。 クライアントを設定して、その他のプロトコルを使用できます。そのためには、`rhn.conf` 設定ファイルの `java.power_management.types` 設定パラメータにフェンスエージェント名をカンマ区切りリストとして追加します。

7.6. 設定のスナップショット

スナップショットは、設定時点でのクライアントのパッケージプロファイル、設定ファイル、およびUyuni設定を記録します。古いスナップショットにロールバックして、前の設定に戻すことができます。



スナップショットは、従来のクライアントでのみサポートされます。 Saltクライアントでは、この機能をサポートしていません。

スナップショットは、一部のアクション実行後に自動的に取得されます。 いつでもスナップショットを手動で取得することもできます。 クライアントで元に戻せないかもしれないアクションを実行する前に、その時点のスナップショットがあることを確認することをお勧めします。

スナップショットはデフォルトで有効になっています。 自動スナップショットは無効にできます。そのためには、`rhn.conf` 設定ファイルの `enable_snapshots=0` を設定します。

スナップショットを管理するには、システム > システム一覧に移動して、管理するクライアントを選択します。 選択したクライアントで現在のスナップショットをすべて一覧表示するには、プロビジョニング > スナップショットタブに移動します。 スナップショットに記録された変更の詳細を表示するには、スナップショットの名前をクリックします。 プロビジョニング > スナップショットタブでサブタブを使用して、選択したスナップショットにロールバックした変更を表示できます。

- グループメンバーシップ
- チャンネルサブスクリプション

- ・インストールされたパッケージ
- ・設定チャンネルサブスクリプション
- ・設定ファイル
- ・スナップショットタグ



スナップショットを使用して、クライアントに対するほとんどの変更をロールバックできますが、すべてではありません。たとえば、複数の更新はロールバックできません。また、製品の移行はロールバックできません。クライアントでアップグレードを実行する前に、必ずバックアップを取ってください。

7.6.1. スナップショットタグ

スナップショットタグを使用すると、わかりやすい説明をスナップショットに追加できます。タグを使用して、動作したことがわかっている最後の設定、成功したアップグレードなどスナップショットに関する追加情報を記録できます。

スナップショットタグを管理するには、**システム > システム一覧**に移動して、管理するクライアントを選択します。選択したクライアントで現在のスナップショットタグをすべて一覧表示するには、**プロビジョニングスナップショットタグ**タブに移動し、**システムタグの作成**をクリックし、説明を入力し、**[現在のスナップショットタグ]**ボタンをクリックします。

7.6.2. 大規模インストールのスナップショット

Uyuniで保持できるスナップショットの上限数はありません。つまり、クライアント、パッケージ、チャンネル、および設定変更を追加すると、スナップショットを保存するデータベースが大きくなります。

数千のクライアントを含む大規模インストールがある場合、古いスナップショットを定期的に削除するように、Uyuni APIを使用して、繰り返しスケジュールに繰り返しクリーンアップスクリプトを使用できます。または、この機能を無効にできます。そのためには、`enable_snapshots=0` (`rhn.conf` 設定ファイル内)を設定します。

7.7. カスタムシステム情報

クライアントに関してカスタマイズしたシステム情報を含めることができます。システム情報は「キー:値」ペアで定義され、クライアントに割り当てることができます。たとえば、特定のプロセッサに対して「キー:値」ペアを定義してから、そのプロセッサがインストールされているすべてのクライアントにそのキーを割り当てることができます。カスタムシステム情報は分類され、UyuniのWeb UIを使用して検索できます。

始める前に、カスタム情報を保存できるキーを作成する必要があります。

プロシージャ: カスタムシステム情報のキーの作成

1. UyuniのWeb UIで、**システム > カスタムシステム情報**に移動し、**[キーの作成]**をクリックします。
2. **[キー ラベル]** フィールドにキーの名前を追加します。スペースは使用しません。`intel-x86_64-quadcore`。

3. [説明] フィールドに必要な追加情報を入力します。

4. 必要な各キーで操作を繰り返します。

従来のクライアントの場合、この情報はUyuniデータベースに保存されます。 Saltのクライアントの場合、この情報はSalt pillarに保存されます。 次のようなコマンドを使用して、Saltクライアントからこの情報を取得できます。

```
salt $minionid pillar.get custom_info:key1
```

このコマンドは、次のような出力になります。

```
$minionid:  
val1
```

カスタムシステム情報キーを作成するとき、キーをクライアントに適用できます。

プロセッジヤ: カスタム情報キーをクライアントに適用する

1. UyuniのWeb UIで、[システム]に移動し、カスタム情報を適用するクライアントをクリックし、[詳細](#) カスタム情報タブに移動します。
2. [値の作成]をクリックします。
3. 適用する値を見つけ、キーラベルをクリックします。
4. [値] フィールドに追加情報を入力します。
5. [キーの更新]をクリックしてカスタム情報をクライアントに適用します。

設定管理の詳細については、[Client-configuration > Configuration-management](#)を参照してください。

7.8. システムセットマネージャ

システムセットマネージャ(SSM)は、同時に複数のクライアントでアクションを実行するために使用します。 SSMで一時的なクライアントセットが作成されます。これは、多数のクライアントに適用する必要がある1回限定アクションに便利です。 より永続的なセットが必要な場合、代わりにシステムグループの使用を検討してください。 システムグループの詳細については、[Client-configuration > System-groups](#)を参照してください。

SSMで使用できるアクションを次の表に示します。 この表のアイコンに意味は次のとおりです。

- ✓: このアクションはこのクライアントタイプ用にSSMで使用できます。
- ✗: このアクションはこのクライアントタイプ用にSSMで使用できません。
- ?: このアクションはこのクライアントタイプ用に検討中であり、後日サポートされる場合と、サポートされない場合があります。

表 40. 使用可能なSSMアクション

アクション	従来	Salt
システムのリスト	✓	✓
パッチのインストール	✓	✓
パッチの更新のスケジュール	✓	✓
パッケージのアップグレード	✓	✓
パッケージのインストール	✓	✓
パッケージの削除	✓	✓
パッケージの検証	✓	✗
グループを作成する	✓	✓
グループの管理	✓	✓
チャンネルのメンバーシップ	✓	✓
チャンネルサブスクリプション	✓	✗
チャンネルの展開/diff	✓	✗
クライアントの自動インストール	✓	✗
スナップショットのタグ	✓	✗
リモートコマンド	✓	✗
電源管理	✓	✗
システム設定の更新	✓	✓
ハードウェアプロファイルの更新	✓	✓
パッケージのプロファイルの更新	✓	✓
カスタム値の設定/削除	✓	✓
クライアントの再起動	✓	✓
クライアントの別の組織への移行	✓	✓
クライアントの削除	✓	✓

SSMのクライアントの選択は複数の方法で実行できます。

- システム > システム一覧に移動し、操作するクライアントにチェックを付けます。
- システム > システムグループに移動し、操作するシステムグループで [SSMで使用] をクリックします。
システムグループに移動し、操作するグループにチェックを付け、[グループでの作業] をクリックします。

操作するクライアントを選択したら、システム > システムセットマネージャに移動し、上部のメニューバーにある [システムが選択されました] アイコンをクリックします。



SSMの詳細は、UyuniのWeb UIで別の部分にある詳細と若干異なる場合があります。SSMでは、使用できるすべての更新が表示されます。そのため、最新バージョンではないかもしれないパッケージをアップグレードされる場合があります。

7.8.1. SSMでベースチャンネルを変更する

SSMを使用して、複数のクライアントのベースチャンネルを同時に変更できます。



ベースチャンネルを大幅に変更すると、影響を受けるクライアントで使用できるパッケージおよびパッチが変更されます。注意して使用してください。

プロシージャ: SSMを使用して複数のクライアントのベースチャンネルを変更する

1. UyuniのWeb UIで、**システム > システム一覧**に移動し、操作するクライアントにチェックを付け、**システム > システムセットマネージャ**に移動します。
2. [チャンネル] サブタブに移動します。
3. リストで現在のベースチャンネルを見つけ、[システム] 列に表示されている数字が正しいことを確認します。この列の数字をクリックして、変更するクライアントの詳細を表示できます。
4. [必要なベースチャンネル] フィールドで新しいベースチャンネルを選択し、[次へ] をクリックします。
5. 子チャンネルそれぞれで、[変更なし]、[サブスクライブ]、または [サブスクライブし]、[次へ] をクリックします。
6. 変更内容を確認し、いつ変更するかを選択します。
7. [確認] をクリックして、変更をスケジュールします。

7.9. システムグループ

システムグループを使用して、多数のクライアントの管理を簡単にできます。グループは、更新、設定チャネル、Saltの状態、または方式の適用など、一括アクションをクライアントで実行するために使用できます。

使用している環境で動作する方法でクライアントをグループに編成できます。たとえば、オペレーティングシステムがインストールされているクライアント、クライアントがある物理的な場所、または処理しているワークロードの種類を編成できます。クライアントは、任意の数のグループに属することができるため、さまざまな方法でグループを定義できます。

クライアントをグループに編成している場合、1つまたは複数のグループのすべてのクライアントの更新を実行できます。または、複数のグループに属しているクライアントの更新を実行できます。たとえば、すべてのSaltクライアント用に1つのグループを定義し、すべてのSLESクライアント用に別のグループを定義できます。その後、すべてのSaltクライアントの更新を実行したり、両グループに属しているクライアントを使用してすべてのSalt SLESクライアントの更新を実行できます。

7.9.1. グループの作成

グループを使用してクライアントを編成する前にグループを作成する必要があります。

プロシージャ: 新しいシステムグループの作成

1. UyuniのWeb UIで、**システム > システムグループ**に移動します。
2. **[グループの作成]**をクリックします。
3. 新しいグループの名前と説明を指定します。
4. **[グループの作成]**をクリックしてグループを保存します。
5. 必要な各グループで操作を繰り返します。

7.9.2. グループにクライアントを追加する

個々のクライアントをグループに追加したり、複数のクライアントを同時に追加できます。

プロシージャ: 1つのクライアントのグループへの追加

1. UyuniのWeb UIで、**システム > システム一覧**に移動し、追加するクライアントの名前をクリックします。
2. **グループ > 参加**タブに移動します。
3. 参加するグループを確認し、**[選択したグループに参加]**をクリックします。

プロシージャ: 複数のクライアントのグループへの追加

1. UyuniのWeb UIで、**システム > システム一覧**に移動し、クライアントを追加するグループの名前をクリックします。
2. **[ターゲットシステム]**タブに移動します。
3. 追加するクライアントを確認して、**[システムの追加]**をクリックします。

プロシージャ: SSMで複数のクライアントをグループに追加する

1. UyuniのWeb UIで、**システム > システム一覧**に移動し、追加するそれぞれのクライアントを確認します。クライアントがシステムセットマネージャに追加されます。
2. **システム > システムセットマネージャ**に移動し、**[グループ]**タブに移動します。
3. 参加するグループを見つけ、**[追加]**にチェックを付けます。
4. **[メンバーの変更]**をクリックします。
5. **[確認]**をクリックして、選択したグループにクライアントを追加します。

システムセットマネージャの詳細については、**Client-configuration > System-set-manager**を参照してください。

グループに属しているクライアントを確認できます。そのためには、**システム > システムグループ**に移動し、グループの名前をクリックし、**[システム]**タブに移動します。または、システムグループをグラフィカル表示できます。そのためには、**システム > 可視化 > システムのグループ化**に移動します。

7.9.3. グループの操作

クライアントをグループに編成すると、グループを使用して更新を管理できます。 Saltクライアントの場合、グループにあるクライアントのすべてに状態と方式を適用できます。

UyuniのWeb UIで、**システム > システムグループ**に移動します。 グループ内のいずれかのクライアントに適用できる更新がある場合、リストにアイコンが表示されます。 アイコンをクリックすると、適用できる更新に関する詳細情報が表示され、クライアントに適用されます。

同時に複数のグループを操作するご操作可能なグループを選択し、**[選択したすべてのグループで作業する]**をクリックし、すべての選択グループですべてのクライアントを選択します。

または、両方のグループに属しているクライアントを操作できます以上のグループを選択し、**[選択したすべてのクライアントのみを選択]**をクリックし、選択したすべてのグループに存在しているクライアントのみを選択します。 たとえば、すべてのSaltクライアント用に1つのグループがあり、すべてのSLESクライアント用に別のグループがあるとします。 これらのグループのインターフェクションはすべてのSalt SLESクライアントになります。

7.10. システムの種類

クライアントは、システムの種類で分類されます。 各クライアントは、両方のベースシステムの種類を備えることができ、アドオンシステムの種類が割り当てられます。

ベースシステムの種類には、従来のクライアントでは **管理**、Saltクライアントでは **Salt** が含まれます。

アドオンシステムの種類には、仮想ホストとして動作するクライアントでは **仮想ホスト**、ビルドホストとして動作するクライアントでは **コンテナビルドホスト** が含まれます。

アドオンシステムの種類は調整できます。そのためには、**システム > システム一覧 > システムの種類**に移動アドオンシステムの種類を変更するクライアントにチェックを付け、**[付属エンタイトルメント]**を選択し、**[エンタイトルメントの追加]**または**[エンタイトルメントの削除]**をクリックします。

ベースシステムの種類を **管理** から **Salt** に変更することもできます。そのためには、クライアントを再登録します。

7.10.1. Web UIを使用して従来のクライアントをSaltに変更する

従来のクライアントをSaltクライアントに変更する最もシンプルなメソッドはWeb UIでクライアントを再登録する方法です。



ベースシステムの種類を変更するには、クライアントを再登録する必要があります。 この操作を実行すると、クライアントのすべてのカスタマイズまたは設定は削除されますが、イベント履歴は保持されます。 クライアントのダウンタイムも発生します。

プロシージャ: Web UIを使用して従来のクライアントをSaltに変更する

1. UyuniのWeb UIで、**システム > システム一覧**に移動し、変更するクライアントを特定し、ホスト名をメ

モします。

2. システム>ブートストラップに移動します。
3. [ホスト] フィールドで、再登録するクライアントのホスト名を入力します。
4. 必要に応じて、他のフィールドを入力します。
5. [ブートストラップ] をクリックして、ブートストラッププロセスをスケジュールします。

クライアントの登録が完了したら、システムの種類が **Salt** である [システム一覧] にそのクライアントが表示されます。

7.10.2. コマンドプロンプトを使用して従来のクライアントをSaltに変更する

コマンドプロンプトを使用して、従来のクライアントをSaltクライアントとして再登録できます。この操作を実行するには、従来のクライアントによって使用されるパッケージを削除する必要があります。その後、Saltクライアントの好きな登録メソッドを使用してクライアントを再登録できます。



ベースシステムの種類を変更するには、クライアントを再登録する必要があります。この操作を実行すると、クライアントのすべてのカスタマイズまたは設定が削除されます。クライアントのダウンタイムも発生します。

プロシージャ: コマンドプロンプトを使用して従来のクライアントをSaltに変更する

1. 変更するクライアントのコマンドプロンプトで、パッケージマネージャを使用して、次のパッケージを削除します。

```
spacewalk-check
spacewalk-client-setup
osad
osa-common
mgr-osad
spacewalksd
mgr-daemon
rhnlib
rhnmd
```

2. 好みの登録メソッドを使用して、クライアントをSaltクライアントとして再登録します。

クライアントの登録が完了したら、システムの種類が **Salt** である [システム一覧] にそのクライアントが表示されます。

Chapter 8. オペレーティングシステムのインストール

一般に、すでに動作しているクライアントを登録します。 Uyuniに登録する直前にコンピュータに手動でインストールするか、環境にUyuniを追加する前にインストールされた既存のシステムを使用できます。

または、Uyuniを使用して、1回の手順でオペレーティングシステムをインストールしてUyuniに登録することもできます。 この方法では一部または全部が自動化されているため、インストーラの質問に答える時間を節約することができます。 これは、特にインストールと登録が必要な多くのクライアントがある場合に役立ちます。

Uyuniからオペレーティングシステムをインストールするには次のような複数の方法があります。

- ・ 登録済みのクライアントでインプレースインストールを行う
- ・ PXEブートを使用してネットワークを通じてインストールする
- ・ インストール用CD-ROMまたはUSBメモリを作成し、そのメディアでコンピュータをブートする
- ・ Uyuni for Retailソリューションの一部としてインストールする

インプレースでの再インストール方法は、以前のオペレーティングシステムがクライアントにすでにインストールされており、クライアントがUyuniにすでに登録されていることを前提としています。

インプレースインストール方法については、[Client-configuration > Autoinst-reinstall](#)を参照してください。

ネットワークブートによるインストール方法は、フォーマットされていないコンピュータで動作します。 ただし、これは次のような特定のネットワーク構成のみで実行できます。

- ・ Uyuniサーバまたはそのプロキシのいずれかが、インストール対象のコンピュータと同じローカルネットワーク上にあるか、経路にあるすべてのルータを中継できるDHCPリレーに対応している。
- ・ 新しいDHCPサーバをセットアップするか、既存のDHCPサーバを設定することができる。
- ・ インストール対象のクライアントがPXEブートに対応しており、PXEブートを実行するように設定することができる。

ネットワークブート方法については、[Client-configuration > Autoinst-pxeboot](#)を参照してください。

リムーバブルメディアを使用する方法では、このようなネットワーク上の制約を受けません。 しかし、この方法はコンピュータがCD-ROMまたはUSBメモリを読み取ることができ、各メディアからブートできることを前提としています。 また、クライアントコンピュータに対する物理的なアクセスも必要です。

リムーバブルメディアを使用する方法については、[Client-configuration > Autoinst-cdrom](#)を参照してください。

Uyuni for Retailアプローチについては、[Retail > Retail-overview](#)を参照してください。



UbuntuクライアントとDebianクライアントの自動インストールはサポートされていません。これらのオペレーティングシステムは、手動でインストールする必要があります。



The autoinstallation features of Uyuni are based on a software named Cobbler. For more information, see <https://cobbler.readthedocs.io>.

8.1. 登録済みシステムを再インストールする

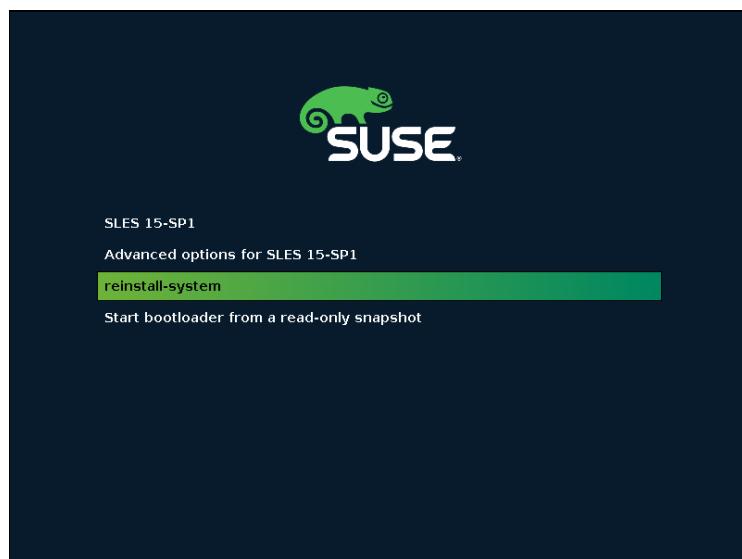
インプレースでの再インストールは、ローカルクライアントシステムから開始します。したがって、クライアントがネットワークを通じてPXEブートを実行できる必要はありません。

登録済みのクライアントをインプレースで再インストールするには、自動インストールのディストリビューションと自動インストールプロファイルを定義する必要があります。 詳細については、**Client-configuration** > **Autoinst-distributions**と**Client-configuration** > **Autoinst-profiles**を参照してください。

自動インストールプロファイルと自動インストールのディストリビューションを定義したら、再インストールを実行できます。

手順: 登録済みのクライアントを再インストールする

1. Uyuni Web UIで、**システム** > **システム一覧**に移動し、再インストールするクライアントを選択し、**プロジェクト** > **自動インストール** > **スケジュール**サブタブに移動します。
2. 作成した自動インストールプロファイルを選択し、必要に応じてプロキシを選択して、**[自動インストール]を[スケジュールしてから終了する]**をクリックします。
3. クライアントが従来のクライアントであり、osadを設定していない場合は、ジョブがフェッチされるまで待つ必要があります。
4. **プロジェクト** > **自動インストール** > **セッションの状態**に移動するか、クライアント上で直接、インストールの進行状況を監視できます。クライアントが再起動したら、[ブート]メニューで [システムの再インストール] という新しいオプションを選択します。

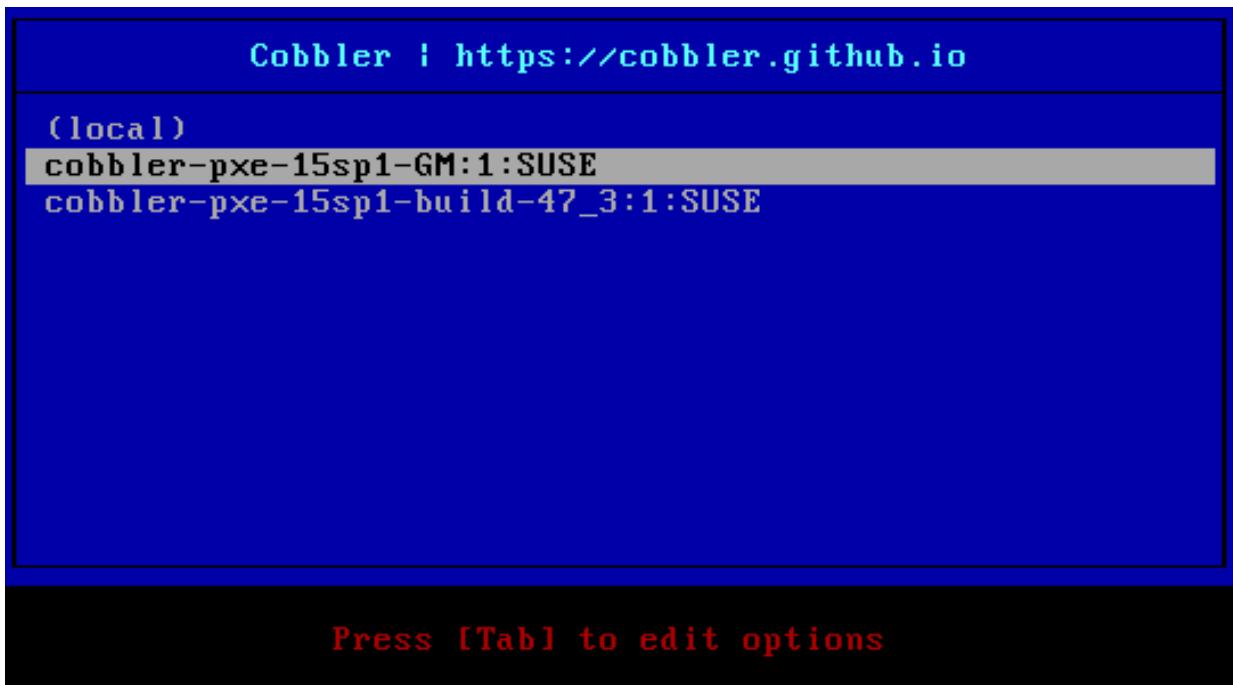


その後、インストールはHTTPプロトコルを通じて進められます。

8.2. ネットワークを通じてインストールする(PXEブート)

ネットワークブートによるインストール中に、次の処理が実行されます。

1. クライアントがPXEモードでブートされます。
2. DHCPサーバが、IPアドレスとマスク、インストールサーバのアドレス、サーバ上のブートローダファイルの名前をクライアントに提供します。
3. クライアントは、インストールサーバからTFTPプロトコルを通じてブートローダファイルをダウンロードし、実行します。
4. クライアントは、インストールに使用できるプロファイルをメニューから選択するか、いずれかのプロファイルで自動インストールを開始します。
5. クライアントは、TFTPプロトコルを通じて、そのプロファイルと一致するディストリビューション用のカーネルと初期RAMディスクをダウンロードします。
6. インストールカーネルは、インストールプログラム、KickstartまたはAutoYaSTを起動します。 それ以降、カーネルは、HTTPプロトコルを通じてサーバで提供されるリソースを使用します。
7. ディストリビューションは、KickstartまたはAutoYaSTプロファイルに従って自動的にインストールされます。
8. プロファイルは、従来のクライアントまたはSaltクライアントとしてクライアントをUyuniサーバに登録するコードスニペットを呼び出します。



インストールサーバは、Uyuniサーバまたはそのプロキシのいずれかにすることができます。 プロキシからインストールするには、事前にサーバとプロキシの間でTFTPツリーを同期させる必要があります。

DHCPサーバは、ホスト名、ルータのアドレス、ドメインネームサーバのアドレスなど、その他の構成情報

をクライアントに提供することもあります。この情報の一部は、インストールサーバをドメイン名で指定した場合など、自動インストールのために必要になることがあります。

[PXEブート]メニューで、最初の選択肢は [ローカルブート] を選択すると、ブートプロセスはローカルディスクドライブから続行されます。特定の時間内にプロファイルが選択されなかった場合は、このオプションが自動的に選択されます。これは、プロファイルを選択する人間のオペレータがいない場合に、自動インストールが開始されないようにするための安全対策です。

または、手動での操作なしで、いずれかのプロファイルからインストールを自動的に開始することもできます。これは「無人プロビジョニング」と呼ばれます。

「ベアメタル」機能は、PXEブートに基づく無人プロビジョニングの一種です。このシナリオでは、ブートローダファイルはUyuniサーバでクライアントを登録するだけで、インストールは開始しません。インプレースでの再インストールを後で実行することができます。

手順: PXEブートによるインストール

1. DHCPサーバを準備します。[DHCPサーバを準備する](#)を参照してください。
2. 自動インストールのディストリビューションを準備します。[Client-configuration > Autoinst-distributions](#)を参照してください。
3. 自動インストールプロファイルを準備します。[Client-configuration > Autoinst-profiles](#)を参照してください。
4. クライアントを再起動し、インストールするプロファイルを選択します。

他の一部の手順はオプションです。プロキシをインストールサーバとして使用するには、[プロキシを使用してTFTPツリーを同期する](#)を参照してください。無人プロビジョニングについては、[Client-configuration > Autoinst-unattended](#)を参照してください。

8.2.1. DHCPサーバを準備する

PXEブートプロセスは、DHCPを使用して、TFTPサーバを検索します。Uyuniサーバまたはそのプロキシは、TFTPサーバとして機能させることができます。

このためには、ネットワークのDHCPサーバへの管理アクセス権が必要です。TFTPブートサーバとしてインストールサーバをポイントするようにDHCP設定ファイルを編集します。

例: ISC DHCPサーバを設定する

1. DHCPサーバでrootとして `/etc/dhcpd.conf` ファイルを開きます。
2. クライアントに対する宣言を次のように変更します。

```
host myclient { (...)  
    next-server 192.168.2.1;  
    filename "pxelinux.0"; }
```

1. ファイルを保存し、`dhcpd` サービスを再起動します。

この例では、**192.168.2.1** でPXEクライアント **myclient** をインストールサーバに指定し、**pixelinux.0** ブートローダファイルの取得を指示しています。

DHCPサーバがUyuniに登録されている場合は、その代わりにDHCPd式を使用して設定することもできます。

例: DHCPd式を使用したISC DHCPサーバの設定

1. システム>システム一覧に移動し、変更するクライアントを選択し、[式] タブに移動してDHCPd式を有効にします。
2. 式の [Dhcpd] タブに移動し、[次のサーバ] フィールドに、インストールサーバのホスト名またはIPアドレスを入力します。
3. [Filename EFI] (ファイル名EFI) フィールドで、**grub/grub.efd** と入力し、EFIPXEのサポートを有効にします。
4. [ファイル名] フィールドで、**pixelinux.0** と入力し、従来のBIOSのサポートを有効にします。
5. [Save Formula] (式の保存) をクリックして設定を保存します。
6. highstateを適用します。

これですべてのホストにグローバルPXEサーバが設定されますが、ホストごとに設定することもできます。DHCPd式の詳細については、**Specialized-guides > Salt**を参照してください。

8.2.2. プロキシを使用してTFTPツリーを同期する

Uyuniサーバ上のTFTPツリーはUyuniプロキシと同期させることができます。同期するには、HTTPSポート443を開く必要があります。



プロキシを追加するたびに、ツリーの同期に時間がかかるようになります。

手順: サーバとプロキシ間のTFTPの同期

1. Uyuniサーバのコマンドプロンプトで、rootとして **susemanager-tftpsync** パッケージをインストールします。

```
zypper install susemanager-tftpsync
```

1. Uyuniプロキシのコマンドプロンプトで、rootとして **susemanager-tftpsync-recv** パッケージをインストールします。

```
zypper install susemanager-tftpsync-recv
```

1. プロキシでrootとして **configure-tftpsync.sh** スクリプトを実行します。スクリプトは、Uyuniサーバおよびプロキシのホスト名およびIPアドレス、プロキシの **tftpboot** ディレクトリの場所の詳細についてインタラクティブに問い合わせます。詳細については、**configure-tftpsync.sh --help** コマンドを使用してください。

2. サーバでrootとして `configure-tftpsync.sh` スクリプトを実行します。

```
configure-tftpsync.sh proxy1.example.com proxy2.example.com
```

3. サーバで `cobbler sync` コマンドを実行して、プロキシにファイルをプッシュします。プロキシが正しく設定されていないとこの操作は失敗します。

プロキシのリストを後で変更する場合、`configure-tftpsync.sh` スクリプトを使用して編集できます。



設定済みのプロキシを再インストールしてすべてのファイルを再プッシュする場合、`cobbler sync` を呼び出す前に `/var/lib/cobbler/pxe_cache.json` でキャッシュファイルを削除する必要があります。

8.3. CD-ROMまたはUSBメモリを使用してインストールする

Uyuniにまだ登録されていないクライアントで、PXEを通じたネットワークブートを選択できない場合は、ブート可能なCD-ROMまたはUSBメモリを使用してシステムをインストールできます。

このようなリムーバブルメディアを作成する1つの方法は、Cobblerを使用することです。 Cobblerを使用してISOイメージを作成する方法については、[CobblerでISOイメージを構築する](#)を参照してください。

もう1つの方法は、ディストリビューションに固有なメカニズムを使用することです。

- SUSEシステムでは、KIWIを使用してISOイメージを作成します。 詳細については、[KIWIでSUSE ISOイメージを構築する](#)を参照してください。
- Red Hatシステムでは、`mkisofs` を使用します。 詳細については、[mkisofsでRedHat ISOイメージを構築する](#)を参照してください。

いずれの場合も、生成されたイメージをCD-ROMまたはUSBメモリに書き込みます。

8.3.1. CobblerでISOイメージを構築する

Cobblerは、一連のディストリビューション、カーネル、およびメニューが含まれているISOブートイメージを作成できます。これはPXEインストールと同じように動作します。



CobblerによるISOの構築はIBM Zではサポートされていません。

CobblerでISOイメージを作成するには、PXEを通じてネットワークブートを使用する場合と同様に、ディストリビューションとプロファイルを作成する必要があります。 ディストリビューションの作成については、[Client-configuration > Autoinst-distributions](#)を参照してください。 プロファイルの作成については、[Client-configuration > Autoinst-profiles](#)を参照してください。

Cobblerの `buildiso` コマンドは、ブートISOの名前および出力場所を定義するパラメータを取ります。

`--distro` でディストリビューションを指定する必要があります。次に例を示します。

```
cobbler buildiso --iso=/path/to/boot.iso --distro=SLE_15-sp1
```

ブートISOには、すべてのプロファイルおよびシステムがデフォルトで含まれています。 **--profiles** オプションと **--systems** オプションで、使用するプロファイルおよびシステムを制限できます。 次に例を示します。

```
cobbler buildiso --systems="system1 system2 system3" \
--profiles="profile1 profile2 profile3 --distro=SLE_15-sp1"
```



ISOイメージをパブリック **tmp** ディレクトリに書き込むことができない場合、**/usr/lib/systemd/system/cobblerd.service** でsystemd設定を確認してください。

8.3.2. KIWIでSUSE ISOイメージを構築する

KIWIはイメージ作成システムです。 KIWIを使用して、SUSEシステムのインストール用にターゲットシステムで使用するブート可能なISOイメージを作成することができます。 システムを再起動または電源オンするとき、このイメージからブートし、AutoYaST設定をUyuniから読み込み、AutoYaSTプロファイルに応じてSUSE Linux Enterprise Serverをインストールします。

ISOイメージを使用するには、システムをブートし、プロンプトに **autoyast** と入力します(AutoYaSTブートのラベルを **autoyast** のままにしていることを想定しています)。 **Enter** キーを押してAutoYaSTのインストールを開始します。

KIWIの詳細については、<http://doc.opensuse.org/projects/kiwi/doc/>を参照してください。

8.3.3. mkisofsでRedHat ISOイメージを構築する

mkisofs を使用して、Red Hatシステムのインストール用にターゲットシステムで使用するブート可能なISOイメージを作成することができます。 システムを再起動または電源オンするとき、このイメージからブートし、Kickstart設定をUyuniから読み込み、Kickstartプロファイルに応じてRed Hat Enterprise Linuxをインストールします。

手順: mkisofsでブート可能なISOイメージを構築する

- ターゲットディストリビューションの最初のCD-ROMから **/isolinux** の内容をコピーします。
- isolinux.cfg** ファイルを編集して' **ks**' にデフォルト設定します。 '**ks**' セクションを次のように変更します。

```
label ks
kernel vmlinuz
append text ks=`url` initrd=initrd.img lang= devfs=nomount \
ramdisk_size=16438 `ksdevice`
```

IPアドレスベースのKickstart URLは次のようにになります。

```
http://`my.manager.server`/kickstart/ks/mode/ip_range
```

IPアドレスの範囲で定義されたKickstartディストリビューションは、エラーの発生を回避するために、構築元のディストリビューションと一致している必要があります。

3. オプション: **ksdevice** を使用する場合、次のようにになります。

```
ksdevice=eth0
```

ファミリ内のキックスタートプロファイルのディストリビューションを変更できます(Red Hat Enterprise Linux AS 4 を Red Hat Enterprise Linux ES 4に変更するなど)。そのためには、新しいディストリビューションラベルを指定します。複数のバージョン間(4から5)または複数の更新間(U1からU2)の移行はできません。

4. 必要に応じて、**isolinux.cfg** をさらにカスタマイズします。たとえば、複数のオプション、さまざまなブートメッセージ、または短いタイムアウト期間を追加できます。

5. 次のコマンドでISOを作成します。

```
mkisofs -o file.iso -b isolinux.bin -c boot.cat -no-emul-boot \
-boot-load-size 4 -boot-info-table -R -J -v -T isolinux/
```

isolinux/ は、ディストリビューションCDからコピーした変更済みのisolinuxファイルを含むディレクトリへの相対パスです。一方、**file.iso** は、出力ISOファイルです。これは現在のディレクトリに配置されます。

6. このISOをCD-ROMに書き込み、ディスクを挿入します。または、USBメモリに書き込んで、差し込みます。
7. システムをブートし、プロンプトに **ks** と入力します(キックスタートブートのラベルを' **ks**' のままにしている場合)。
8. **Enter** キーを押してKickstartを起動します。

8.4. 自動インストールのディストリビューション

自動インストールプロセスでは、インストールを開始するために複数のファイルが必要です。必要なファイルには、Linuxカーネル、初期RAMディスク、およびインストールモードでオペレーティングシステムをブートするために必要なその他のファイルが含まれます。

DVDイメージから必要なファイルを抽出できます。 詳細については、[ISOイメージに基づくディストリビューション](#)を参照してください。

または、**tftpboot-installation** パッケージをインストールすることもできます。 詳細については、[RPM/パッ](#)

ケージに基づくディストリビューションを参照してください。

また、これらのファイルと同じオペレーティングシステムバージョン用に、Uyuniサーバでベースチャンネルを同期させておく必要があります。

ファイルの準備が整い、ベースチャンネルを同期したら、ディストリビューションを宣言する必要があります。この操作により、インストールファイルがベースチャンネルに関連付けられます。ディストリビューションは、1つ以上のインストールプロファイルによって参照されることがあります。 詳細については、[自動インストールのディストリビューションを宣言する](#)を参照してください。

8.4.1. ISOイメージに基づくディストリビューション

この方法では、クライアントにインストールするオペレーティングシステムのインストールメディアがあることを前提としています。このメディアは通常DVD **.iso** イメージです。これには、Linuxカーネル、**initrd** ファイル、およびインストールモードでオペレーティングシステムをブートするために必要なその他のファイルが含まれています。

手順: インストールメディアからのファイルのインポート

1. インストールメディアをUyuniサーバにコピーします。 SUSEオペレーティングシステムの場合、<https://www.suse.com/download/>からインストールメディアをダウンロードできます。
2. ISOイメージをループマウントして、そのコンテンツをどこかにコピーします。

```
# mount -o loop,ro <image_name>.iso /mnt
# mkdir -p /srv/www/distributions
# cp -a /mnt /srv/www/distributions/<image_name>
# umount /mnt
```

ファイルパスをメモしておいてください。

このファイルパスは、ディストリビューションを{productname}に対して宣言するときに必要になります。

8.4.2. RPMパッケージに基づくディストリビューション

この方法は、SUSEシステムで動作します。 インストールシステム用にあらかじめパッケージされたファイルを使用するため、インストールメディアからコンテンツをインポートするよりも簡単です。

手順: インストールパッケージからファイルを抽出する

1. Uyuniサーバに、名前が **tftpboot-installation** で始まるパッケージをインストールします。 このパッケージの正確な名前は、**zypper se tftpboot-installation** コマンドで確認できます。
2. **ls -d /usr/share/tftpboot-installation/*** コマンドで、インストールファイルの場所を確認します。 ファイルパスをメモしておいてください。 このファイルパスは、ディストリビューションをUyuniに対して宣言するときに必要になります。

この手順では、Uyuniサーバに搭載されているものと同じバージョンのオペレーティングシステムをインストールする準備をします。 クライアントに異なるオペレーティングシステムやバージョンをインストールする場合は、**tftpboot-installation-*** パッケージを、これが属するディストリビューションから手動で取得する必要があります。 [パッケージ検索] 入力ボックスで、名前が**tftpboot-installation**で始まるパッケージを検索し、そのパッケージの詳細を確認します。ここには、**/var/spacewalk/**以下のローカルパスが表示されます。

8.4.3. 自動インストールのディストリビューションを宣言する

自動インストールファイルを抽出した後の次の手順は、自動インストールのディストリビューションの宣言です。

手順: 自動インストールのディストリビューションの宣言

1. UyuniのWeb UIで、**システム** > **自動インストール** > **ディストリビューション**に移動します。
2. [ディストリビューションの作成] をクリックし、次のフィールドに入力します。
 - [ディストリビューションラベル] フィールドに、自動インストール可能なディストリビューションを識別するための名前を入力します。
 - [ツリーパス] フィールドに、Uyuniサーバに保存されているインストールメディアへのパスを入力します。
 - 対応する [ベースチャネル] を選択します。これはインストールメディアと一致する必要があります。
 - [インストーラ生成] を選択します。これはインストールメディアと一致する必要があります。
 - オプション: このディストリビューションをブートするときに使用するカーネルオプションを指定します。カーネルオプションを指定する方法は複数あります。ここにはディストリビューションに当てはまるオプションのみを追加します。
3. [自動インストール可能なディストリビューションの作成] をクリックします。

準備したインストールファイルには、インストールする必要があるパッケージが含まれていない可能性があります。必要なパッケージが含まれていない場合は、[カーネルオプション] フィールドに **useonlinerepo=1** を追加します。

パッケージリポジトリには、署名されていないことがあるメタデータが含まれています。メタデータが署名されていない場合は、[カーネルオプション] フィールドに **insecure=1** を追加するか、**configuration** Autoinst-ownpgpkey の説明に従って独自のGPGキーを使用します。

これらの関連のオプションは、フルDVDの代わりに「オンラインインストーラ」ISOイメージを使用する場合や、**tftpboot-installation** パッケージを使用する場合などに必要です。

自動インストールのディストリビューションを管理するには、**システム** > **自動インストール** > **ディストリビューション**に移動します。

8.5. 自動インストールプロファイル

自動インストールプロファイルによって、オペレーティングシステムをインストールする方法が決定されま

す。たとえば、インストーラに渡す追加のカーネルパラメータを指定できます。

プロファイルの最も重要な部分は、「自動インストールファイル」です。インストールを手動で実行する場合、パーティション設定、ネットワーク情報、ユーザの詳細などの情報をインストーラに提供する必要があります。自動インストールファイルは、スクリプト形式でこの情報を提供する方法です。このタイプのファイルは、「回答ファイル」と呼ばれることもあります。

Uyuni内では、インストールするクライアントのオペレーティングシステムに応じて、2つの異なる種類のプロファイルを使用できます。

- SUSE Linux EnterpriseクライアントまたはopenSUSEクライアントの場合、AutoYaSTを使用します。
- Red Hat Enterprise Linuxクライアントの場合、Kickstartを使用します。

異なるオペレーティングシステムでクライアントをインストールする場合、AutoYaSTプロファイルとKickstartプロファイルの両方を使用できます。

- プロファイルを宣言する方法については、[プロファイルを宣言する](#)を参照してください。
- AutoYaSTプロファイルについては、[AutoYastプロファイル](#)を参照してください。
- Kickstartプロファイルについては、[Kickstartプロファイル](#)を参照してください。

プロファイルに含まれる自動インストールファイルには、変数とコードスニペットを格納できます。変数とコードスニペットについては、[テンプレートの構文](#)を参照してください。

8.5.1. プロファイルを宣言する

自動インストールファイルとディストリビューションの準備ができたら、プロファイルを作成して、Uyuniサーバで自動インストールを管理できます。プロファイルにより、選択したこのディストリビューションのインストール方法が決定されます。プロファイルを作成する1つの方法はAutoYaSTファイルまたはKickstartファイルをアップロードする方法です。または、Kickstartのみの場合、Web UIウィザードを使用できます。

手順: アップロードによる自動インストールプロファイルの作成

1. UyuniのWeb UIで、**システム > 自動インストール > プロファイル**に移動します。
2. [キックスタート/Autoyast] ファイルをアップロードをクリックします。
3. [ラベル] フィールドにプロファイルの名前を入力します。スペースは使用しません。
4. [自動インストールツリー] フィールドで、このプロファイルに使用する自動インストールのディストリビューションを選択します。
5. [仮想化タイプ] フィールドで、このプロファイルに使用する仮想化の種類を選択します。または、このプロファイルを使用して新しい仮想マシンを作成しない場合には[なし]を選択します。
6. 自動インストールファイルの内容を [ファイルの内容] フィールドにコピーするか、または [アップロードするファイル] フィールドを使用してファイルを直接アップロードします。

ここに記載する詳細については、`xref:client-configuration:autoinstall-profiles.adoc#autoyast`[AutoYastプロファイル]または`xref:client-configuration:autoinstall-profiles.adoc#kickstart` [kickstartプロファイル]を参照してください。

7. **[作成]**をクリックしてプロファイルを作成します。

プロシージャ: ウィザードでKickstartプロファイルを作成する

1. UyuniのWeb UIで、**システム** > **自動インストール** > **プロファイル**に移動します。
2. **[キックスタートプロファイルを作成]**をクリックします。
3. **[ラベル]**フィールドにプロファイルの名前を入力します。スペースは使用しません。
4. **[ベースチャンネル]**フィールドで、このプロファイルに使用するベースチャンネルを選択します。このフィールドは利用できるディストリビューションから入力されます。必要なベースチャンネルが利用できない場合、ディストリビューションを正しく作成したことを確認してください。
5. **[仮想化タイプ]**フィールドで、このプロファイルに使用する仮想化の種類を選択します。または、仮想化しない場合には**[なし]**を選択します。
6. **[次へ]**をクリックします。
7. **[配信ファイルの場所]**で、Uyuniサーバにインストールするインストールメディアへのパスを入力します。
8. **[次へ]**をクリックします。
9. クライアントのrootユーザのパスワードを入力します。
10. **[完了]**をクリックします。
11. 新しいプロファイルの詳細を確認し、必要に応じてカスタマイズします。

自動インストールプロファイルを作成している場合、**[このベースチャンネルに最新のツリーを更新]**にチェックを付けることができます。この設定では、指定ベースチャンネルに関連付けられた最新ディストリビューションをUyuniで自動選択できます。新しいディストリビューションを後で追加する場合、Uyuniは、最後に作成または変更されたディストリビューションを使用します。

仮想化の種類を変更すると、通常、プロファイルのブートローダおよびパーティションオプションを変更する必要があります。この操作を実行すると、カスタマイズが上書きされます。新しい設定または変更した設定を保存前に確認します。そのためには、**[パーティション設定]**タブに移動します。

ディストリビューションとプロファイルのカーネルオプションは統合されます。

自動インストールプロファイルの詳細および設定を変更できます。そのためには、**システム** > **自動インストール** > **プロファイル**に移動し、編集するプロファイルの名前をクリックします。または、**システム** > **システム**一覧に移動し、プロビジョニングするクライアントを選択し、**プロビジョニング** > **自動インストールサブタブ**に移動します。

8.5.2. AutoYaSTプロファイル

AutoYaSTプロファイルは、プロファイルを識別する ラベル、自動インストールのディストリビューションをポイントする 自動インストールツリー、さまざまなオプション、最も重要なAutoYaSTインストールファイルで構成されます。

AutoYaSTインストールファイルは、AutoYaSTインストーラに指示を与えるXMLファイルです。AutoYaSTでは、「制御ファイル」と呼ばれます。AutoYaSTインストールファイルの構文の詳細については、<https://doc.opensuse.org/projects/autoyast/#cha-configuration-installation-options>を参照してください。

SUSEには、独自のカスタムファイルの雛形として使用できるAutoYaSTインストールファイルのテンプレートが用意されています。このテンプレートは、<https://github.com/SUSE/manager-build-profiles> の **AutoYast** ディレクトリにあります。各プロファイルを使用するにはその前に、一部の変数を設定する必要があります。スクリプトに含まれている **README** ファイルを確認して、必要な変数を判別してください。AutoYaSTスクリプトで変数を使用する方法の詳細については、[変数](#)を参照してください。

UyuniでインストールするためのAutoYaSTインストールファイルで、最も重要なセクションを次に示します。

- **<add-on>** では、インストールに子チャンネルを追加できます。

```
https://doc.opensuse.org/projects/autoyast/#Software-Selections-additional with an ``<add-on>`` exampleを参照してください。
```

- **<general>\$SNIPPET('spacewalk/sles_no_signature_checks')</general>** は、署名のチェックを無効にします。
- **<software>** allows to specify product for the Unified Installer

```
https://doc.opensuse.org/projects/autoyast/#Software-Selections-additional with an ``<add-on>`` exampleを参照してください。
```

- **<init-scripts config:type="list">\$SNIPPET('spacewalk/minion_script ')</init-scripts>** は、クライアントをSaltクライアントとしてUyuniに登録できるようにします。

AutoYaSTの詳細については、<https://doc.opensuse.org/projects/autoyast/>を参照してください。

AutoYaSTに代わるSaltベースの最近のプロファイルには、Yomiがあります。Yomiについては、[Specialized-guides](#) › Saltを参照してください。

8.5.3. キックスタートプロファイル

Kickstartプロファイルには、多数の設定オプションがあります。プロファイルを作成するには、プロファイルをアップロードするか、専用のウィザードを使用します。

Kickstartプロファイルでは、ファイル保持一覧を使用できます。Kickstartで再インストールするクライアントにあるカスタム設定ファイルが多数ある場合、リストにしてこれらのファイルを保存し、そのリストをKickstartプロファイルに関連付けることができます。

手順: ファイル保持一覧の作成

1. UyuniのWeb UIで、**自動インストール**に移動し、**[ファイル保持一覧の作成]**をクリックします。
2. 適切なラベルを入力し、保存するすべてのファイルおよびディレクトリへの絶対パスをリストします。
3. **[一覧の作成]**をクリックします。
4. Kickstartプロファイルにファイル保持一覧を含めます。
5. **システム > 自動インストール > プロファイル**に移動して編集するプロファイルを選択し、**システムの詳細** > **ファイル保持**サブタブに移動して、含めるファイル保持一覧を選択します。



ファイル保持一覧の合計サイズは1 MBに制限されています。
 /dev/hda1
 や/dev/sda1などの特殊なデバイスは保持できません。ファイル名とディレクトリ名のみ使用できます。正規表現のワイルドカードは使用できません。

Kickstartの詳細については、Red Hatのドキュメントを参照してください。

8.5.4. テンプレートの構文

インストールファイルの一部は、インストール中に置き換えられます。変数は単一の値に置き換えられ、コードスニペットはテキストのセクション全体に置き換えられます。エスケープされた記号やセクションは置き換えられません。

CobblerはCheetahと呼ばれるテンプレートエンジンを使用して、このような置き換えを実行できます。このメカニズムにより、システムごとにプロファイルを手動で作成する必要なく、多数のシステムを再インストールできます。

自動インストールの変数やコードスニペットは、Uyuni Web UI内で作成できます。プロファイル内の**[自動インストール ファイル]**タブでは、置き換えの結果を確認できます。

- ・ 変数については、[AutoYast Profile#variables\[変数\]](#)を参照してください。
- ・ コードスニペットについては、[AutoYast Profile#code-snippets\[コードスニペット\]](#)を参照してください。
- ・ エスケープ記号またはセクション全体については、[エスケープ](#)を参照してください。

8.5.4.1. 変数

自動インストールの変数は、KickstartプロファイルおよびAutoYaSTプロファイルに値を代入するために使用できます。変数を定義するには、プロファイルから**[変数]**サブタブに移動し、テキストボックスで `name=value` ペアを作成します。

たとえば、クライアントのIPアドレスを格納する変数と、ゲートウェイのアドレスを格納する変数を作成で

きます。 次に、作成した変数は、同じプロファイルからインストールされるすべてのクライアントに対して定義できます。このためには、[変数] テキストボックスに次の行を追加します。

```
ipaddr=192.168.0.28
gateway=192.168.0.1
```

変数を使用するには、プロファイルで値の前に \$ 記号を付けて値を代入します。たとえば、Kickstartファイルの [ネットワーク] 部分は次のようにになります。

```
network --bootproto=static --device=eth0 --onboot=on --ip=$ipaddr \
--gateway=$gateway
```

\$ipaddr は **192.168.0.28** に解決され、**\$gateway** は **192.168.0.1** に解決されます。

インストールファイルでは、変数は階層的に使用します。システム変数はプロファイル変数より優先され、プロファイル変数はディストリビューション変数より優先されます。

8.5.4.2. コードスニペット

Uyuniには、多数の定義済みコードスニペットが付属しています。システム > 自動インストール > 自動インストールスニペットに移動し、既存のスニペットの一覧を表示します。

自動インストールファイルの **\$SNIPPET()** マクロに挿入してスニペットを使用します。たとえば、Kickstart では次のようにになります。

```
$SNIPPET('spacewalk/rhel_register_script')
```

または、AutoYaSTでは次のようにになります。

```
<init-scripts config:type="list">
$SNIPPET('spacewalk/sles_register_script')
</init-scripts>
```

このマクロはCobblerによって解析され、スニペットの内容に置き換えられます。独自のコードスニペットを保存して、後で自動インストールファイルで使用すること **[スニペットの作成]** をクリックして、新しいコードスニペットを作成します。

この例では、一般的なハードドライブのパーティション設定のKickstartスニペットが設定されます。

```

clearpart --all
part /boot --fstype ext3 --size=150 --asprimary
part / --fstype ext3 --size=40000 --asprimary
part swap --recommended

part pv.00 --size=1 --grow

volgroup vg00 pv.00
logvol /var --name=var vgname=vg00 --fstype ext3 --size=5000

```

たとえば、次のようにスニペットを使用します。

```
$SNIPPET('my_partition')
```

8.5.4.3. エスケープ

自動インストールファイルには、**\$(example)** のようなシェルスクリプト変数が含まれています。コンテンツはバックスラッシュ(円記号)でエスケープする必要があります **\\$(example)**。**\$** 記号をエスケープすると、テンプレートエンジンは記号を内部変数として評価しなくなります。

長いスクリプトまたは文字列は、**\#raw** ディレクティブおよび**\#end** ディレクティブで囲むことによってエスケープできます。次に例を示します。

```

#raw
#!/bin/bash
for i in {0..2}; do
    echo "$i - Hello World!"
done
#end raw

```

記号の後にスペースがある行はコメントとして扱われるため、評価されません。次に例を示します。

```

# start some section (this is a comment)
echo "Hello, world"
# end some section (this is a comment)

```

8.6. 無人プロビジョニング

「ベアメタル」機能を使用すると、汎用PXEブートイメージを使用して、ローカルネットワークに接続した直後に新しいコンピュータを登録することができます。次に、Uyuni Web UIに移動して、このコンピュータにプロファイルを割り当てます。次にクライアントをブートしたときに、そのプロファイルに従ってオペレーティングシステムがインストールされます。ベアメタルプロビジョニングについては、[ベアメタルプロビ](#)

[ジョギング](#)を参照してください。

ベアメタル機能を使用たくない場合は、Uyuniでシステムを手動で宣言することもできます。Uyuni APIを使用すると、ベアメタル機能で収集されたかのように、システムに対するシステムレコードを作成することができます。APIを使用したシステムの宣言については、[システムレコードを手動で作成する](#)を参照してください。

8.6.1. ベアメタルプロビジョニング

ベアメタルプロビジョニングオプションを有効にしている場合、Uyuniネットワークに接続されているクライアントは、電源をオンにするとすぐに組織に自動追加されます。この処理が完了すると、クライアントはシャットダウンし、[システム]一覧に表示され、インストールする準備ができます。

手順: ベアメタル機能を有効にする

1. UyuniのWeb UIで、**管理** > **マネージャ設定** > **ベアメタルシステム**に移動します。
2. [この組織に対する追加の有効化]をクリックします。

電源をオンにした新しいクライアントは、ベアメタル機能を有効にした管理者が所属している組織に追加されます。このようなクライアントは「ブートストラップ」タイプであり、通常のクライアントにするためにプロビジョニングが必要です。

新しいクライアントを追加する組織を変更するには、ベアメタル機能を無効にし、新しい組織の管理者としてログインし、機能を再有効化します。登録済みのシステムは別の組織に移行できます。そのためには[移行]タブを使用します。

この方法で登録するクライアントでは、システムセットマネージャ(SSM)を使用できます。ただし、オペレーティングシステムがまだインストールされていないため、このようなクライアントでは使用できないSSM機能があります。これは、この方法で登録したシステムを含む混合セットにも当てはまります。セットのすべてのクライアントがプロビジョニングされると、すべての機能をセットで使用できるようになります。

SSMの詳細については、[Client-configuration](#) > [System-set-manager](#)を参照してください。

手順: 「ブートストラップ」タイプのクライアントをプロビジョニングする

1. Web UIで、[システム]に移動し、プロビジョニングするクライアントを選択し、**プロビジョニング** > **自動インストールタブ**に移動します。
2. 使用するAutoYaSTプロファイルを選択し、[PXEインストール設定の作成]をクリックします。選択すると、Cobblerでシステムエントリが作成されます。
3. クライアントの電源をオンにします。

サーバは、TFTPを使用して新しいクライアントをプロビジョニングするため、プロビジョニングを正常に実行するには適切なポートおよびネットワークが正しく設定されている必要があります。

8.6.2. システムレコードを手動で作成する

APIコールを使用して、MACアドレスによって識別されるクライアントと自動インストールプロファイル間の関連付けを宣言できます。次にシステムを再起動したときに、指定したプロファイルに基づいてインストールが開始されます。

手順: 手動で宣言したプロファイルからの再インストール

- Uyuniサーバのコマンドプロンプトで、`system.createSystemRecord` APIコールを使用します。この例では、`name`をクライアントの名前に、`<profile>`をプロファイルラベルに、`<iface>`を`eth0`などのクライアント上のインターフェース名に、`<hw_addr>`を`00:25:22:71:e7:c6`などのクライアントのハードウェアアドレスに置き換えます。

```
$ spacecmd api -- --args '[ "<name>", "<profile>", "", "", \
[ {"name": "<iface>", "mac": "<hw_addr>"} ]' \
system.createSystemRecord
```

- クライアントの電源をオンにします。ネットワークからブートすると、インストール用の正しいプロファイルが選択されます。

このコマンドによって、Cobblerでシステムレコードが作成されます。カーネルオプション、クライアントのIPアドレス、クライアントのドメイン名など、追加のパラメータを指定することもできます。詳細については、`createSystemRecord` コードのAPIドキュメントを参照してください。

8.7. 独自のGPGキーを使用する

自動インストールのために使用しているリポジトリに署名されていないメタデータがある場合は、通常、自動インストールのディストリビューションのオプションとして`insecure=1`カーネルパラメータを使用し、AutoYaSTインストールファイルで`spacewalk/sles_no_signature_checks`コードスニペットを使用する必要があります。

より安全な代替方法は、独自のGPGキーを提供することです。

手順: 独自のGPGキーを追加する

- GPGキーを作成します。
- このキーを使用して、パッケージのメタデータに署名します。
- インストールメディアの初期RAMディスクにこのキーを追加します。
 - キーを作成し、そのキーを使用してメタデータに署名する方法については、[Administration > Repo-metadata](#)を参照してください。
 - ネットワークブートに使用するインストールメディアにキーを追加する方法については、[PXEブート用の独自のGPGキー](#)を参照してください。
 - CD-ROMからのブートに使用するインストールメディアにキーを追加する方法については、[CD-ROM内の独自のGPGキー](#)を参照してください。



この操作は、SUSEクライアントにのみ適用されます。



新しいGPGキーを使用してメタデータに署名した場合、オンボード済みのクライアントはこの新しいキーを認識しません。 クライアントを登録する前に、メタデータに署名することが理想的です。

リポジトリを使用するオンボード済みのクライアントの場合、修正方法は、そのクライアントでGPGキーのチェックを無効にすることです。

8.7.1. PXEブート用の独自のGPGキー

PXEブートプロセスで使用される初期RAMディスク(**initrd**)には、通常SUSEのGPGキーのみが格納されています。 パッケージをチェックするために使用できるように、このファイルに独自のキーを追加する必要があります。

手順: 初期RAMディスクにGPGキーを追加する

1. GPGキーを見つけるためにブートプロセス中に使用されるものと同じパスにディレクトリを作成します。

```
mkdir -p tftpboot/usr/lib/rpm/gnupg/keys
```

2. **.asc** サフィックスを付けてこのディレクトリにGPGキーをコピーします。

```
cp /srv/www/htdocs/pub/mgr-gpg-pub.key
tftpboot/usr/lib/rpm/gnupg/keys/mgr-gpg-pub.asc
```

3. 最上位のディレクトリ内で、コンテンツをパッケージ化し、インストールメディアファイルの一部である **initrd** に追加します。

```
cd tftpboot
find . | cpio -o -H newc | xz --check=crc32 -c >> /path/to/initrd
```

8.7.2. CD-ROM内の独自のGPGキー

mksusecd ユーティリティでインストールイメージを修正できます。 このユーティリティは、Development Toolsモジュールに含まれています。

手順: インストールISOイメージにGPGキーを追加する

1. GPGキーを見つけるためにブートプロセス中に使用されるものと同じパスにディレクトリを作成します。

```
mkdir -p initrdroot/usr/lib/rpm/gnupg/keys
```

2. **.asc** サフィックスを付けてこのディレクトリにGPGキーをコピーします。

```
cp /srv/www/htdocs/pub/mgr-gpg-pub.key  
initrdroot/usr/lib/rpm/gnupg/keys/mgr-gpg-pub.asc
```

3. **mksusecd** で既存のISOイメージを修正します。

```
mksusecd --create <new-image>.iso --initrd initrdroot/ <old-image>.iso
```

Chapter 9. 仮想化

Uyuniを使用して、通常の従来のクライアントまたはSaltクライアントに加えて、仮想化されたクライアントを管理できます。この種のインストールでは、仮想ホストは、Uyuniサーバにインストールされ、任意の数の仮想ゲストを管理します。このインストールを選択すると、複数の仮想ホストをインストールし、ゲストのグループを管理できます。

仮想化されたクライアントにある機能の範囲は、選択したサードパーティ仮想化プロバイダによって決まります。

XenおよびKVMのホストおよびゲストはUyuniで直接管理できます。そうすると、AutoYaSTまたはKickstartを使用してホストおよびゲストを自動インストールし、Web UIでゲストを管理できます。

VMWare vSphere、Nutanix AHVなどのVMWareでは、Uyuniは、仮想ホストマネージャ(VHM)を設定し、VMを制御する必要があります。そうするとホストおよびゲストを制御できますが、XenおよびKVMで可能な制御方法より限定されます。Uyuniは、VMWare vSphereまたはNutanix AHVでVMを作成または編集できません。

他のサードパーティ仮想化プロバイダはUyuniでは直接サポートされていません。ただし、プロバイダでVMのJSON設定ファイルをエクスポートできる場合、その設定ファイルをUyuniにアップロードし、VHMで管理できます。

VHMを使用して仮想化を管理する方法の詳細については、[Client-configuration > Vhm](#)を参照してください。

9.1. 仮想化ホストの管理

開始前に、仮想化ホストとして使用するクライアントで「仮想化ホスト」システムタイプが割り当てられていることを確認してください。従来のクライアントとSaltクライアントの両方を仮想化ホストとして使用できます。システム>システム一覧に移動し、仮想化ホストとして使用するクライアントの名前をクリックします。システムタイプは、「システムのプロパティ」セクションにリスト「仮想化ホスト」システムタイプがリストされていない場合、「[...に|れ|ら|の|ブ|ロ|パ|テ|イ|を|編|集|し|ま|す]」をクリックして割り当

クライアントに「仮想化ホスト」システムタイプがある場合、クライアントの「システムの詳細」ページで「仮想化」タブを使用でき「仮想化」タブでは、仮想ゲストを作成して管理し、ストレージプールおよび仮想ネットワークを管理できます。

9.2. 仮想ゲストの作成

UyuniのWeb UI内で仮想ゲストを仮想化ホストに追加できます。

プロシージャ: 仮想ゲストの作成

1. UyuniのWeb UIで、システム一覧に移動し、仮想化ホストの名前をクリックし、「仮想化」タブに移動します。
2. 「全般」セクションで、次の詳細を入力します。
 - 「ゲスト」サブタブで、[Create Guest] (ゲストの作成) をクリックします。

- [名前] フィールドにゲストの名前を入力します。
 - [ハイパーバイザ] フィールドで、使用するハイパーバイザを選択します。
 - [仮想マシンのタイプ] フィールドで、完全仮想化または部分的仮想化のいずれかを選択します。
 - [最大メモリ] フィールドに、ゲストディスクの最大サイズ制限(MB単位)を入力します。
 - [仮想CPU数] で、ゲストのvCPUの数を入力します。
 - [アーキテクチャ] フィールドで、ゲストで使用するエミュレートCPUアーキテクチャを選択します。デフォルトでは、選択したアーキテクチャは仮想ホストと一致しています。
 - [自動インストールプロファイル] フィールドで、ゲストのインストールに使用する自動インストールツールを選択します。自動インストールを使用しない場合、このフィールドを空白のままにします。
3. [ディスク] セクションで、クライアントで使用する仮想ディスクの詳細をソーステンプレートのイメージURL] フィールドで、オペレーティングシステムのイメージへのパスを入力したことを確認してください。これを実行しないと、ゲストのディスクは空になります。
 4. [ネットワーク] セクションで、クライアントで使用するネットワークインターフェースの詳細を入力します。[MACアドレス] フィールドを空白のままにして、MACアドレスを生成します。
 5. [グラフィックス] セクションで、クライアントで使用するグラフィックスドライバの詳細を入力します。
 6. ゲストを作成する時間をスケジュールし、[作成] をクリックしてゲストを作成します。
 7. 新しい仮想ゲストは、正常に作成されるとすぐに開始されます。

Uyuni Web UI内のペースメーカークラスタに仮想ゲストを追加することもできます。

プロシージャ: クラスタ管理対象仮想ゲストの作成

1. 次の追加項目を使用して、クラスタのノードの1つで仮想ゲストの作成プロシージャに従います。
 - クラスタリソースとして定義 フィールドがチェックされていることを確認します。
 - VM定義のクラスタ共有フォルダーへのパス フィールドに、ゲスト構成が保存されるタノードによって共有されるフォルダーへのパスを入力します。
 - すべてのディスクが、すべてのクラスタノードによって共有されるストレージプールに配置されていることを確認してください。

クラスタによって管理される仮想ゲストは、ライブマイグレーションできます。

9.3. XenおよびKVMを使用した仮想化

XenおよびKVMの仮想化クライアントはUyuniで直接管理できます。

まず、Uyuniサーバで仮想ホストを設定する必要があります。 今後の仮想ホストおよび仮想ゲストのAutoYaSTまたはKickstartを使用して自動インストールを設定できます。

このセクションでは、インストール後に仮想ゲストを管理する方法についても説明します。

9.3.1. ホストの設定

VMホストでXenまたはKVMを設定する方法は、関連するゲストで使用するオペレーティングシステムによって決まります。

SUSEオペレーティングシステムについては、<https://documentation.suse.com/sles/15-SP3/html/SLES-all/book-virtualization.html>でSLES仮想化に関するガイドを参照してください。

Red Hat Enterprise Linuxオペレーティングシステムについては、使用バージョンに応じてRed Hatのドキュメントを参照してください。

Uyuniは、**libvirt**を使用してゲストをインストールして管理します。ホストに**libvirdt**パッケージがインストールされている必要があります。ほとんどの場合、デフォルト設定で十分で、調整する必要はありません。ただし、ゲストのVNCコンソールに非rootユーザとしてアクセスする場合、設定変更を実行する必要があります。 設定方法の詳細については、ご使用のオペレーティングシステム用のマニュアルを参照してください。

Uyuniサーバでブートストラップスクリプトが必要です。 ブートストラップスクリプトには、ホストのアクティベーションキーを含める必要があります。 GPGキーも含めてセキュリティを強化することをお勧めします。 ブートストラップスクリプトの作成については、**Client-configuration > Registration-bootstrap**を参照してください。

ブートストラップスクリプトの準備ができたら、ホストで実行し、Uyuniサーバに登録します。 クライアントの登録の詳細については、**Client-configuration > Registration-overview**を参照してください。

Saltクライアントの場合、**仮想化ホスト**エンタitleメントを有効にする必要があります。VMの変更をすぐに表示するためには、UyuniのWeb UIでホストの [**システムの詳細**] ページで、[**プロパティ**] タブをまだは**仮想化ホスト**エンタitleメントを登録キーレベルで**エンタitleメント**セクションで、[**仮想化ホスト**] にチェックを付け、**[プロパティ]** [更新] をクリックして変更を保存します。Salt minionサービスを再起動して変更を有効にします。

```
systemctl restart salt-minion
```

従来のクライアントの場合、デフォルトでは、VMホストは**rhnsd**サービスを使用して、スケジュールされたアクションを確認します。 確認は4時間ごとに実行され、多数のクライアントが存在する環境の負荷を均等化します。そのため、アクションの実行前に、最大4時間の遅延が発生する可能性があります。VMゲストを管理している場合、この長時間の遅延は常に(特にゲストの再起動の際には)理想的ではありません。 この問題に対処するには、**rhnsd**サービスを無効にして**osad**サービスを有効にできます。**osad**サービスは、jabberプロトコルを使用してコマンドを受け取り、すぐにコマンドを実行します。

rhnsdサービスを無効にして**osad**デーモンを有効にするには、次のコマンドをrootユーザとして実行します。

```
service rhnsd stop
service rhnsd disable
```

```
service osad enable
service osad start
```

9.3.2. 自動インストール

AutoYaSTまたはKickstartを使用して、XenおよびKVMのゲストを自動的にインストールして登録できます。

ゲスト登録先のVMホストでアクティベーションキーがそれぞれのゲストで必要です。アクティベーションキーには、**プロジェクト**のエンタイトルメントと**仮想化プラットフォーム**のエンタイトルメントがあります。アクティベーションキーは、**mgr-virtualization-host**パッケージおよび**mgr-osad**パッケージにもアクセスできる必要があります。アクティベーションキーの作成の詳細については、[Client-configuration > Activation-keys](#)を参照してください。

インストール後にUyuniでゲストを自動的に登録する場合、ブートストラップスクリプトを作成する必要があります。ブートストラップスクリプトの作成については、[Client-configuration > Registration-bootstrap](#)を参照してください。



VMゲストの自動インストールは、従来のクライアントとして設定されている場合のみ機能します。Saltクライアントはテンプレートディスクイメージを使用して作成できます。AutoYaSTまたはKickstartを使用して作成することはできません。

9.3.2.1. 自動インストール可能なディストリビューションの作成

Uyuniからクライアントを自動インストールできる自動インストール可能なディストリビューションをVMに作成する必要があります。ディストリビューションは、マウントされたローカルディレクトリやリモートディレクトリから使用できたり、ループマウントされたISOイメージで使用できます。

自動インストール可能なディストリビューションの設定は、SLESまたはRed Hat Enterprise Linuxオペレーティングシステムをゲストで使用しているかどうかによって異なります。Red Hat Enterprise Linuxインストールのパッケージは、関連するベースチャンネルからフェッチされます。SUSEシステムをインストールするパッケージは、自動インストール可能なディストリビューションからフェッチされます。したがって、SLESシステムでは、自動インストール可能なディストリビューションは、完全なインストールソースである必要があります。

表 41. 自動インストール可能なディストリビューションのパス

オペレーティングシステムの種類	カーネルの場所	initrdの場所
Red Hat Enterprise Linux	<code>images/pxeboot/vmlinuz</code>	<code>images/pxeboot/initrd.img</code>
SLES	<code>boot/<arch>/loader/initrd</code>	<code>boot/<arch>/loader/linux</code>

すべてのケースで、ベースチャンネルが自動インストール可能なディストリビューションと一致していることを確認してください。

始める前に、使用しているVMホストでインストールメディアを使用できることを確認してください。これは、ネットワークリソース上、ローカルディレクトリ内、またはループマウントされたISOイメージ内にある場合があります。また、すべてのファイルおよびディレクトリが全世界で読み取ることができるかを確認

してください。

プロシージャ: 自動インストール可能なディストリビューションの作成

1. UyuniのWeb UIで、**システム** > **自動インストール** > **ディストリビューション**に移動し、**[自動インストール可能なディストリビューションの作成]**をクリックします。
2. [自動インストール可能なディストリビューションの作成] セクションで、次です。
 - [ディストリビューションラベル] セクションに、ディストリビューションの固有の名前を入力します。半角の英字、数字、ハイフン(-)、ピリオド(.)、および下線(_)のみを使用し、5文字以上にしてください。
 - [ツリーパス] フィールドに、インストールソースへの絶対パスを入力します。
 - [ベースチャンネル] フィールドで、インストールソースと一致するチャンネルを選択します。このチャンネルは、非SUSEインストール環境用のパッケージソースとして使用されます。
 - [インストーラ生成] フィールドで、インストールソースと一致するオペレーティングシステムのバージョンを選択します。
 - [カーネルオプション] フィールドに、インストールでブート時にカーネルに渡すオプションを入力します。`install=`パラメータおよび`self_update=0 pt.options=self_update`パラメータはデフォルトで追加されます。
 - インストールしたシステムを初めてブートするときにカーネルに渡すオプションを [カーネルの後のオプション] セクションに入力します。
3. **[自動インストール可能なディストリビューションの作成]**をクリックして作成します。

自動インストール可能なディストリビューションを作成するとき、これを編集できます。そのためには、**システム** > **自動インストール** > **ディストリビューション**に移動し、編集するディストリビューションを選択します。

9.3.2.2. 自動インストールプロファイルの作成およびアップロード

自動インストールプロファイルには、システムをインストールするために必要なインストールデータおよび設定データがすべて含まれています。インストール完了後に実行するスクリプトを含めることもできます。

KickstartプロファイルはUyuniのWeb UIを使用して作成できます。そのためには、**システム** > **自動インストール** > **新しくキックスタートプロファイルを作成**をクリックし、プロンプトに従って操作します。

AutoYaSTまたはKickstartの自動インストールプロファイルを手動で作成することもできます。SUSEには、独自のカスタムファイルの雛形として使用できるAutoYaSTインストールファイルのテンプレートが用意されています。これは、<https://github.com/SUSE/manager-build-profiles>にあります。

AutoYaSTを使用してSLESをインストールする場合、次のスニペットも含める必要があります。

```
<products config:type="list">
  <listentry>SLES</listentry>
</products>
```

- AutoYaSTの詳細については、[client-configuration:autoinst-profiles.pdf](#)を参照してください。
- Kickstartについては、[client-configuration:autoinst-profiles.pdf](#)を参照するか、Red Hatのインストール関連ドキュメントを参照してください。

プロシージャ: 自動インストールプロファイルのアップロード

- UyuniのWeb UIで、**システム** > **自動インストールプロファイル**に移動し、**[新規]**をクリックします。
- [自動インストールプロファイルを作成] セクションで、次のパラメータを使用します。
 - [ラベル] フィールドにプロファイルの一意の名前を入力します(半角の英字、数字、ハイフン(-)、ピリオド(.)、および下線(_)のみを使用し、7文字以上にしてください)。
 - [自動インストールツリー] フィールドで、前に作成した自動インストール可能なディストリビューションを選択します。
 - [仮想化タイプ] フィールドで、関連するゲストの種類を選択します(**KVM 仮想化ゲスト**など)。ここでは、[Xen 仮想化ホスト] を選択しないでください。
 - 自動インストールプロファイルを手動で作成する場合、[ファイルの内容] フィールドに直接入力できます。作成済みの場合、[ファイルの内容] フィールドを空白のままにします。
 - [アップロードするファイル] フィールド [Choose File] (ファイルの選択) をクリックしてシステムダイアログを使用して、アップロードするファイルを選択します。ファイルが正常にアップロードされると、ファイル名が [アップロードするファイル] フィールドに表示されます。
 - アップロードしたファイルの内容が [ファイルの内容] フィールドに表示される場合、直接編集できます。
- [作成] をクリックして変更を保存し、プロファイルを保存します。

自動インストールプロファイルを作成するとき、これを編集できます。そのためには、**システム** > **自動インストールプロファイル**に移動し、編集するプロファイルを選択します。**[作成]**をクリックして、必要な変更を行い、設定を保存します。



既存のKickstartプロファイルの [仮想化タイプ] を変更する場合、ブートローダおよびパーティションのオプションも変更する場合があり、カスタム設定を上書きすること [パーティション設定] タブを注意深く確認して、変更前にこれらの設定を確認してください。

9.3.2.3. ゲストを自動的に登録する

VMゲストを自動的にインストールするとき、Uyuniには登録されません。ゲストをインストールしてすぐに自動的に登録する場合、ブートストラップスクリプトを呼び出してゲストを登録する自動インストールプロ

ファイルにセクションを追加できます。

このセクションでは、ブートストラップスクリプトを既存のAutoYaSTプロファイルに追加する手順について説明します。

ブートストラップスクリプトの作成の詳細については、[Client-configuration > Registration-bootstrap](#)を参照してください。Kickstartでこの作業を行う方法については、Red Hatのインストール関連ドキュメントを参照してください。

プロシージャ: ブートストラップスクリプトをAutoYaSTプロファイルに追加する

1. 登録するVMゲストのアクティベーションキーがブートストラップスクリプトに含まれていることを確認してください。これはホストの `/srv/www/htdocs/pub/bootstrap_vm_guests.sh` にあります。
2. UyuniのWeb UIで、[システム > 自動インストール](#)、プロファイルに移動し、このスクリプトを関連付けるAutoYaSTプロファイルを選択します。
3. [ファイルの内容] フィールドで、次のスニペットをファイルの末尾(`</profile>`タグの直前)に追加します。スニペットのIPアドレス例を、使用中のUyuniサーバの正しいIPアドレスに置き換えてください。

```
<scripts>
  <init-scripts config:type="list">
    <script>
      <interpreter>shell </interpreter>
      <location>
        http://`192.168.1.1`/pub/bootstrap/bootstrap_vm_guests.sh
      </location>
    </script>
  </init-scripts>
</scripts>
```

4. [更新] をクリックして変更を保存します。



AutoYaSTプロファイルに `<scripts>` セクションがすでに含まれている場合、2つ目のセクションを追加しないでください。既存の `<scripts>` セクション内にブートストラップスニペットを配置します。

9.3.2.4. VMゲストの自動インストール

すべての設定が完了したら、VMゲストの自動インストールを開始できます。



各VMホストが同時にインストールできるゲストは1つだけです。複数の自動インストールをスケジュールしている場合、前のインストールが完了する前に次のインストールが始まらないようにスケジュールしてください。ゲストのインストールが別のインストールの実行中に開始すると、実行中のインストールはキャンセルされます。

1. UyuniのWeb UIで、**システム** > **概要**に移動し、ゲストをインストールするVMホストを選択します。
2. 「**仮想化**」タブ、「**プロジェクト**」サブタブに移動します。
3. 使用する自動インストールプロファイルを選択し、ゲストの一意の名前を指定します。
4. 該当する場合にはプロキシを選択し、スケジュールを入力します。
5. ゲストのハードウェアのプロファイルおよび設定オプションを変更するには、**[高度なオプション]**をクリックします。
6. **[自動インストールをスケジュールしてから終了する]**をクリックして完了します。

9.3.3. VMゲストの管理

UyuniのWeb UIを使用して、CPUやメモリの割り当て調整、シャットダウン、再起動のようなアクションなど、VMゲストを管理できます。

そのためには、XenまたはKVM VMホストをUyuniサーバに登録し、**libvirtd** サービスをホストで実行する必要があります。従来のクライアントでは、**mgr-cfg-actions** パッケージをUyuniサーバにインストールする必要があります。

UyuniのWeb UIで、**システム** > **システム一覧**に移動し、管理するゲストのVMホストをクリックします。 「**仮想化**」タブに移動し、このホストに登録されているすべてのゲストを表示し、管理機能にアクセスします。

Web UIを使用してVMゲストを管理する方法の詳細については、**Reference** > **Systems**を参照してください。

Chapter 10. 仮想ホストマネージャ

仮想ホストマネージャ(VHM)は、さまざまなクライアントの種類から情報を収集するために使用します。

VHMを使用して、プライベートクラウドまたはパブリッククラウドのインスタンスを収集し仮想化グループに編成できます。このように編成された仮想化クライアントを使用して、Taskomaticは、クライアントのデータを収集し、UyuniのWeb UIに表示します。VHMを使用すると、仮想化されたクライアントでサブスクリプションマッチングを使用することもできます。

UyuniサーバにVHMを作成して使用し、使用可能なパブリッククラウドのインスタンスを評価できます。VHMを使用して、Kubernetesで作成したクラスタを管理することもできます。

- Amazon Web ServicesでVHMを使用する方法の詳細については、[Client-configuration > Vhm-aws](#)を参照してください。
- Microsoft AzureでVHMを使用する方法の詳細については、[Client-configuration > Vhm-azure](#)を参照してください。
- Google Compute EngineでVHMを使用する方法の詳細については、[Client-configuration > Vhm-gce](#)を参照してください。
- KubernetesでVHMを使用する方法の詳細については、[Client-configuration > Vhm-kubernetes](#)を参照してください。
- NutanixでVHMを使用する方法の詳細については、[Client-configuration > Vhm-nutanix](#)を参照してください。
- VMWare vSphereでVHMを使用する方法の詳細については、[Client-configuration > Vhm-vmware](#)を参照してください。
- その他のホストでVHMを使用する方法の詳細については、[Client-configuration > Vhm-file](#)を参照してください。

10.1. VHMおよびAmazon Web Services

仮想ホストマネージャ(VHM)を使用して、Amazon Web Services (AWS)からインスタンスを収集できます。

VHMを使用すると、Uyuniは、クラスタに関する情報を取得して報告できます。VHMの詳細については、[Client-configuration > Vhm](#)を参照してください。

10.1.1. Amazon EC2 VHMの作成

仮想ホストマネージャ(VHM)はUyuniサーバ上で動作します。

virtual-host-gatherer-libcloud パッケージをUyuniサーバにインストール済みであることを確認してください。

プロシージャ: Amazon EC2 VHMの作成

1. UyuniのWeb UIで、[システム > 仮想ホストマネージャ](#)に移動します。

2. [作成]をクリックし、ドロップダウンメニューから [Amazon EC2] を選択します。
3. [Add an Amazon EC2 Virtual Host Manager] (Amazon EC2仮想ホストマネージャの追加) セクションで、次のパラメータを使用します。
 - [ラベル] フィールドにVHMのカスタム名を入力します。
 - [Access Key ID] (アクセスキーID) フィールドに、Amazonが提供するアクセスキーIDを入力します。
 - [Secret Access Key] (秘密アクセス鍵) フィールドに、Amazonインスタンスに関連付けられた秘密アクセス鍵を入力します。
 - [Region] (リージョン) フィールドに、使用するリージョンを入力します。
 - [Zone] (ゾーン) フィールドに、VMが存在するゾーンを入力します。これは、サブスクリプションマッチングを動作させるために必要です。リージョンおよびゾーンの設定の詳細については、[client-configuration:virtualization.pdf](#)を参照してください。
4. [作成]をクリックして変更を保存し、VHMを作成します。
5. [仮想ホストマネージャ] ページで、新しいVHMを選択します。
6. [プロパティ] ページで、[データの更新]をクリックし、新しいVHMを評価します。

評価されたオブジェクトおよびリソースを表示するには、システム > システム一覧 > 仮想システムに移動します。

Amazonパブリッククラウドで動作しているインスタンスは、UUIDをUyuniサーバに報告します。その際のフォーマットは、`i`に17桁の16進数をつなげたものです。

```
I1234567890abcdef0
```

10.1.2. 仮想ホストマネージャのAWS許可

セキュリティ上の理由から、タスクを実行するために可能な限り最小限の権限を常に付与してください。AWSに接続するユーザに過度な許可を持つアクセスキーを使用することはお勧めしません。

SUSE ManagerがAWSから必要な情報を収集するには、VHMにEC2インスタンスとアドレスを記述する許可が必要です。これを許可する1つの方法は、このタスクに固有の新しいIAMユーザ(IDおよびアクセス管理)を作成し、次のようにポリシーを作成して、ユーザにアタッチすることです。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeAddresses",
        "ec2:DescribeInstances"
      ],
      "Resource": "*"
    }
  ]
}
```

特定のリージョンへのアクセスを制限することで、許可をさらに制限できます。 詳細については、https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ExamplePolicies_EC2.html#iam-example-read-onlyを参照してください。

10.2. VHMとAzure

仮想ホストマネージャ(VHM)を使用して、Microsoft Azureからインスタンスを収集できます。

VHMを使用すると、Uyuniは、使用している仮想マシンに関する情報を取得して報告できます。 VHMの詳細については、**Client-configuration > Vhm**を参照してください。

10.2.1. 前提条件

作成したVHMは、Azure VMにアクセスするために、正しいパーミッションが割り当てられている必要があります。

サブスクリプション管理者としてAzureアカウントにログインし、Azureユーザアカウントとアプリケーションが正しいグループに属していることを確認してください。 アプリケーションが属しているグループによって、そのロールが決まり、パーミッションが決まります。

10.2.2. Azure VHMの作成

仮想ホストマネージャ(VHM)はUyuniサーバ上で動作します。

virtual-host-gatherer-libcloud パッケージをUyuniサーバにインストール済みであることを確認してください。

プロシージャ: Azure VHMの作成

1. UyuniのWeb UIで、**システム > 仮想ホストマネージャ**に移動します。
2. **[作成]** をクリックし、ドロップダウンメニューから **[Azure]** を選択します。

3. [Add an Azure Virtual Host Manager] (Azure仮想ホストマネージャの追加) セクションで、次のパラメータを使用します。

- [ラベル] フィールドにVHMのカスタム名を入力します。
- [Subscription ID] (サブスクリプションID) フィールドに、[Azure portal > Services > Subscriptions](#) ページにあるサブスクリプションIDを入力します。
- [Application ID] (アプリケーションID) フィールドに、このアプリケーションを登録したときに収集したアプリケーションIDを入力します。
- [Tenant ID] (テナントID) フィールドに、このアプリケーションを登録したときに収集したAzureが提供するテナントIDを入力します。
- [Secret Key] (秘密鍵) フィールドに、Azureインスタンスに関連付けられた秘密鍵を入力します。
- [Zone] (ゾーン) フィールドに、VMが存在するゾーンを入力します。たとえば、西ヨーロッパの場合、`westeurope` と入力します。これは、サブスクリプションマッチングを動作させるために必要です。

4. [作成] をクリックして変更を保存し、VHMを作成します。

5. [仮想ホストマネージャ] ページで、新しいVHMを選択します。

6. [プロパティ] ページで、[データの更新] をクリックし、新しいVHMを評価します。

評価されたオブジェクトおよびリソースを表示するには、[システム > システム一覧 > 仮想システム](#)に移動します。

10.2.3. パーミッションの割り当て

パーミッションが正しく設定されていない場合、`virtual-host-gatherer` を実行すると次のようなエラーが発生する場合があります。

```
General error: (一般エラー:) [AuthorizationFailed] The client 'client_name' with object id 'object_ID' does not have authorization to perform action 'Microsoft.Compute/virtualMachines/read' over scope '/subscriptions/not-very-secret-subscription-id' or the scope is invalid.  
([AuthorizationFailed] オブジェクトID「object_ID」のクライアント「client_name」には、アクション「Microsoft.Compute/virtualMachines/read」をスコープ「/subscriptions/not-very-secret-subscription-id」を超えて実行する権限がありません。またはスコープが無効です。) If access was recently granted, please refresh your credentials. (アクセスが最近付与された場合は、資格情報を更新してください。)
```

正しい資格情報を判断するには、Uyuniサーバのプロンプトで次のコマンドを実行します。

```
virtual-host-gatherer -i input_azure.json -o out_azure.json -vvv
```

input_azure.json ファイルには次の情報が含まれています。

```
[  
  {  
    "id": "azure_vhm",  
    "module": "Azure",  
    "subscription_id": "subscription-id",  
    "application_id": "application-id",  
    "tenant_id": "tenant-id",  
    "secret_key": "secret-key",  
    "zone": "zone"  
  }  
]
```

10.2.4. Azure UUID

Azureパブリッククラウドで実行されているインスタンスは、このUUIDをUyuniサーバに報告できます。

13f56399-bd52-4150-9748-7190aae1ff21

10.3. VHMおよびGoogle Compute Engine

仮想ホストマネージャ(VHM)を使用して、Google Compute Engine (GCE)からインスタンスを収集できます。

VHMを使用すると、Uyuniは、使用している仮想マシンに関する情報を取得して報告できます。 VHMの詳細については、[Client-configuration > Vhm](#)を参照してください。

10.3.1. 前提条件

作成したVHMは、GCE VMにアクセスするために、正しいパーミッションが割り当てられている必要があります。

Googleクラウドプラットフォームのアカウントに管理者としてログインし、クラウドのIDおよびアクセス管理(IAM)ツールを使用して、サービスアカウントに適切なロールがあることを確認してください。

10.3.2. GCE VHMの作成

仮想ホストマネージャ(VHM)はUyuniサーバ上で動作します。

VHMを実行するには、Uyuniサーバでポート443がオープンになっていて、クライアントにアクセスする必要があります。

virtual-host-gatherer-libcloud パッケージをUyuniサーバにインストール済みであることを確認してください

い。

開始する前に、GCEパネルにログインし、証明書ファイルをダウンロードします。このファイルをUyuniサーバにローカルに格納し、パスをメモします。

プロセッジヤ: GCE VHMの作成

1. UyuniのWeb UIで、**システム > 仮想ホストマネージャ**に移動します。
2. **[作成]**をクリックし、ドロップダウンメニューから **[Google Compute Engine]** を選択します。
3. **[Add a Google Compute Engine Virtual Host Manager]** (Google Compute Engineの仮想ホストマネージャの追加) セクションで、次のパラメータを使用します。
 - **[ラベル]** フィールドにVHMのカスタム名を入力します。
 - **[Service Account Email]** (サービスアカウントメール) フィールドに、サービスアカウントに関連付けられているメールアドレスを入力します。
 - **[Cert Path]** (証明書のパス) フィールドに、GCEパネルからダウンロードしたキーへのUyuniサーバのローカルパスを入力します。
 - **[プロジェクトID]** フィールドに、GCEインスタンスで使用するプロジェクトIDを入力します。
 - **[Zone]** (ゾーン) フィールドに、VMが存在するゾーンを入力します。これは、サブスクリプションマッチングを動作させるために必要です。
4. **[作成]**をクリックして変更を保存し、VHMを作成します。
5. **[仮想ホストマネージャ]**ページで、新しいVHMを選択します。
6. **[プロジェクト]**ページで、**[データタの更新]**をクリックし、新しいVHMを評価します。

評価されたオブジェクトおよびリソースを表示するには、**システム > システム一覧 > 仮想システム**に移動します。

10.3.3. パーミッションの割り当て

パーミッションが正しく設定されていない場合、**virtual-host-gatherer** を実行すると次のようなエラーが発生する場合があります。

```
ERROR: (エラー:) {'domain': 'global', 'reason': 'forbidden', 'message': "Required 'compute.zones.list' permission for 'projects/project-id'"}
ERROR: (エラー:) Could not connect to the Google Compute Engine Public Cloud using specified credentials. (指定した資格情報を使用してGoogle Compute Engineのパブリッククラウドに接続できませんでした。)
```

正しい資格情報を判断するには、Uyuniサーバのプロンプトで次のコマンドを実行します。

```
virtual-host-gatherer -i input_google.json -o out_google.json -vvv
```

input_google.json ファイルには次の情報が含まれています。

```
[
  {
    "id": "google_vhm",
    "module": "GoogleCE",
    "service_account_email": "mail@example.com",
    "cert_path": "secret-key",
    "project_id": "project-id",
    "zone": "zone"
  }
]
```

10.3.4. GCE UUID

Googleパブリッククラウドで実行されているインスタンスは、このUUIDをUyuniサーバに報告できます。

152986662232938449

10.4. VHMとKubernetes

仮想ホストマネージャ(VHM)を使用して、Kubernetesクラスタを管理できます。

VHMを使用すると、Uyuniは、クラスタに関する情報を取得して報告できます。VHMの詳細については、[Client-configuration > Vhm](#)を参照してください。

KubernetesでUyuniを使用するには、Uyuniサーバがコンテナ管理用に設定されていて、必要なすべてのチャンネルがあり、登録されているコンテナビルドホストが利用できる必要があります。

次の要件もあります。

- ・ 1つ以上のKubernetesのクラスタをネットワーク上で使用できる。
- ・ **virtual-host-gatherer-Kubernetes** パッケージがUyuniサーバにインストールされている。
- ・ Kubernetesバージョン1.5.0以上。
- ・ コンテナビルドホストにDockerバージョン1.12以上がある。

10.4.1. Kubernetes VHMの作成

Kubernetesクラスタは、UyuniにVHMとして登録されています。

Kubernetesクラスタを登録して認可する **kubeconfig** ファイルが必要です。Kubernetesのコマンドラインツールである **kubectl** を使用して **kubeconfig** ファイルを取得できます。 **kubectl config view --flatten=true** は、VHMに必要に応じて証明書ファイルが埋め込まれた設定を提供します。

プロシージャ: Kubernetes VHMの作成

1. UyuniのWeb UIで、**システム > 仮想ホストマネージャ**に移動します。
2. **[作成]**をクリックし、**[Kubernetes クラスタ]**を選択します。
3. **[Add a Kubernetes Virtual Host Manager]** (Kubernetes仮想ホストマネージャの追加) セクションで、次のパラメータを使用します。
 - **[ラベル]** フィールドにVHMのカスタム名を入力します。
 - Kubernetesクラスタに必要データが含まれている **kubeconfig** ファイルを選択します。
4. **[コンテキスト]** フィールドで、クラスタに適切なコンテキストを選択ねは **kubeconfig** ファイルで指定されています。
5. **[作成]**をクリックします。

プロシージャ: クラスタのノードを表示する

1. UyuniのWeb UIで、**システム > 仮想ホストマネージャ**に移動します。
2. Kubernetesクラスタを選択します。
3. **[Schedule refresh data]** (データの更新をスケジュールする) をクリックしてノードのデータを更新します。

ノードのデータの更新には数分かかる場合があります。 更新された情報を表示するには、ブラウザウィンドウを更新する必要があります。

接続および認証の問題は **gatherer.log** にログされます。



登録中にはノードのデータは更新されません。 データを表示するにはデータを手動で更新する必要があります。

10.4.2. イメージランタイムデータの取得

UyuniのWeb UIでKubernetesイメージに関するランタイムデータを表示できます。そのためには、**イメージ > イメージリスト**に移動します。

イメージリストの表には、3つの列があります。

- **リビジョン:**

Uyuniによってビルトされたイメージをリビルトするたびに、または外部でビルトされたイメージをインポートするたびに増加するシーケンス番号。

- **ランタイム:**

登録されたクラスタの各イメージにおける実行中インスタンスの全般的な状態。

- **インスタンス:**

Uyuniで登録されているすべてのクラスタでこのイメージを実行しているインスタンスの数。 数値の横のポップアップアイコンをクリックして数値の内訳を表示できます。

[**ランタイム**] 列には、次の状態メッセージのいずれかが表示されます。

- 全てのインスタンスがSUSE Managerと同期できています:**

実行中のすべてのインスタンスがUyuniによって追跡されているイメージの同じビルドを実行しています。

- 古いインスタンスが見つかりました:**

インスタンスの一部が古いビルドのインスタンスを実行しています。 イメージを再展開する必要があるかもしれません。

- 情報無し:**

Uyuniに含まれているイメージデータとインスタンスイメージのチェックサムが一致していません。 イメージを再展開する必要があるかもしれません。

プロシージャ: イメージのビルド

1. UyuniのWeb UIで、**イメージ > ストア**に移動します。
2. **[作成]**をクリックしてイメージストアを作成します。
3. **イメージ > プロファイル**に移動します。
4. **[作成]**をクリックしてイメージプロファイルを作成します Kubernetesへの展開に適したdockerファイルを使用する必要があります。
5. **イメージ > ビルド**に移動して、新しいプロファイルでイメージをビルドします。
6. イメージを登録済みのKubernetesクラスタのいずれかに展開します。 この操作は `kubectl` ツールを使用して実行できます。

更新データは、**イメージ > イメージリスト**にあるイメージリストに表示されます。

プロシージャ: 以前展開したイメージのインポート

1. UyuniのWeb UIで、**イメージ > イメージストア**に移動します。
2. インポートするイメージを所有しているレジストリがない場合、追加します。
3. **イメージ > イメージリスト**に移動し、**[インポート]**をクリックします。
4. 各フィールドに入力し、作成したイメージストアを選択し、**[インポート]**をクリックします。

インポートしたデータは、**イメージ > イメージリスト**にあるイメージリストに表示されます。

プロシージャ: 以前展開したイメージの再ビルド

1. UyuniのWeb UIで、**イメージ > イメージリスト**に移動し、再ビルドするイメージが含まれている行を探

し、[詳細]をクリックします。

- [ビルド状態]セクションに移動し、[再ビルト]をクリックします。再ビルトには少し時間がかかります。

再ビルトが正常に完了すると、**イメージ > イメージリスト**のイメージリストでイメージのランタイム状態が更新されます。インスタンスが前のビルトのインスタンスを実行していることをこれは示しています。



再ビルトできるのは、元々Uyuniでビルトされたイメージのみです。インポートしたイメージは再ビルトできません。

プロシージャ: 追加のランタイムデータの取得

- UyuniのWeb UIで、**イメージ > イメージリスト**に移動し、実行中のインスタンスが含まれている行を探し、[詳細]をクリックします。
- [概要情報]セクションには、[ランタイム]フィールドと[インスタンス]フィールドにデータがあります。
- [ランタイム]タブに移動します。タブには、登録されているすべてのクラスタでこのイメージを実行しているKubernetesポッドに関する情報が含まれています。このセクションの情報を次に示します。
 - ポッドの名前。
 - ポッドがあるネームスペース。
 - 指定されているポッドのコンテナのランタイム状態。

10.4.3. パーミッションと証明書



Uyuniでは `kubeconfig` ファイルにすべての証明書データが埋め込まれている場合、このファイルのみ使用できます。

UyuniからのAPIコールは次のとおりです。

- `GET /api/v1/pods`
- `GET /api/v1/nodes`

Uyuniの最小推奨パーミッションは次のとおりです。

- すべてのノードをリストするClusterRole:

```
resources: ["nodes"]
verbs: ["list"]
```

- すべてのネームスペースのポッドをリストするClusterRole(ロールのバインドはネームスペースを制限してはいけません):

```
resources: ["pods"]
verbs: ["list"]
```

/pods が403の応答を返した場合、Uyuniはクラスタ全体を無視します。

RBAC認証の操作方法の詳細については、<https://kubernetes.io/docs/admin/authorization/rbac/>を参照してください。

10.5. Nutanixによる仮想化

UyuniではNutanix AHV仮想マシンを使用できます。そのためには、仮想ホストマネージャ(VHM)を設定します。まず、UyuniサーバでVHMを設定し、使用できるVMホストを評価する必要があります。

10.5.1. VHMの設定

仮想ホストマネージャ(VHM)はUyuniサーバ上で動作します。

virtual-host-gatherer-Nutanix パッケージをUyuniサーバにインストール済みであることを確認してください。

VHMを実行するには、Uyuniサーバでポート9440がオープンになっていて、Nutanix Prism Element APIにアクセスする必要があります。

プロシージャ: Nutanix VHMの作成

1. UyuniのWeb UIで、**システム** > **仮想ホストマネージャ**に移動します。
2. **[作成]** をクリックし、**[Nutanix AHV]** を選択します。
3. **[Add a Nutanix AHV Virtual Host Manager]** (Nutanix AHV仮想ホストマネージャの追加) セクションで、次のパラメータを使用します。
 - **[ラベル]** フィールドにVHMのカスタム名を入力します。
 - **[ホスト名]** フィールドに、完全修飾ドメイン名(FQDN)またはホストIPアドレスを入力します。
 - **[ポート]** フィールドに、使用するPrism Element APIポートを入力します(**9440**など)。
 - **[ユーザ名]** フィールドに、VMホストに関連付けられているユーザ名を入力します。
 - **[パスワード]** フィールドに、VMホストユーザに関連付けられているパスワードを入力します。
4. **[作成]** をクリックして変更を保存し、VHMを作成します。
5. **[仮想ホストマネージャ]** ページで、新しいVHMを選択します。
6. **[プロパティ]** ページで、**[データの更新]** をクリックし、新しいVHMを評価します。

評価されたオブジェクトおよびリソースを表示するには、**システム** > **システム一覧** > **仮想システム**に移動します。



HTTPSを使用してブラウザからNutanix APIサーバに接続すると、**無効な証明書エラー**がログされる場合があります。このエラーが発生すると、仮想ホストマネージャからのデータの更新は失敗します。Nutanix APIサーバでは、(自己証明書ではなく)有効なSSL証明書が必要です。Nutanix SSL証明書にカスタムCA認証局を使用している場合、カスタムCA証明書をUyuniサーバの `/etc/pki/trust/anchors` にコピーします。 証明書を再度信頼します。そのためには、コマンドラインで `update-ca-certificates` コマンドを実行し、spacewalkサービスを再起動します。

VHMが作成されて設定されると、Taskomaticは、データ収集を自動的に実行します。データ収集を手動で実行する場合、**仮想ホストマネージャ**に移動し、適切なVHMを選択して **[データの更新]** をクリックします。

APIを使用してVHMに接続して仮想ホストの情報をリクエストできる **virtual-host-gatherer** というツールがUyuniに付属しています。**virtual-host-gatherer** は、オプションモジュールの概念を維持していて、各モジュールが特定のVHMを有効にします。このツールは、Taskomaticによって毎晩自動的に呼び出されます。**virtual-host-gatherer** ツールのログファイルは `/var/log/rhn/gatherer.log` にあります。

10.6. VMWareによる仮想化

Uyuniでは、ESXiやvCenterなどのVMWare vSphere仮想マシンを使用できます。そのためには、仮想ホストマネージャ(VHM)を設定します。

まず、UyuniサーバでVHMを設定し、使用できるVMホストを評価する必要があります。次に、Taskomaticは、VMのAPIを使用してデータ収集を開始できます。

10.6.1. VHMの設定

仮想ホストマネージャ(VHM)はUyuniサーバ上で動作します。

VHMを実行するには、Uyuniサーバでポート443がオープンになっていて、VMWare APIにアクセスする必要があります。

VMWareホストは、アクセスロールとパーミッションを使用して、ホストおよびゲストへのアクセスを制御します。VHMで評価するVMWareのオブジェクトまたはリソースに少なくとも **read-only** パーミッションがあることを確認してください。任意のオブジェクトまたはリソースを除外する場合、除外対象に **no-access** というマークを付けます。

新しいホストをUyuniに追加している場合、ユーザおよびオブジェクトに割り当てられているロールおよびパーミッションをUyuniで評価する必要があるかどうかを検討する必要があります。

ユーザ、ロール、およびパーミッションの詳細については、VMWare vSphereのドキュメント(<https://docs.vmware.com/en/VMware-vSphere/index.html>)を参照してください。

プロシージャ: VMWare VHMの作成

1. UyuniのWeb UIで、**システム > 仮想ホストマネージャ**に移動します。

2. [作成]をクリックし、[VMWareベース]を選択します。
3. [Add a VMWare-based Virtual Host Manager] (VMWareベースの仮想ホストマネージャの追加) セクションで、次のパラメータを使用します。
 - [ラベル] フィールドにVHMのカスタム名を入力します。
 - [ホスト名] フィールドに、完全修飾ドメイン名(FQDN)またはホストIPアドレスを入力します。
 - [ポート] フィールドに、使用するESXi APIポートを入力します(443など)。
 - [ユーザ名] フィールドに、VMホストに関連付けられているユーザ名を入力します。
 - [パスワード] フィールドに、VMホストユーザに関連付けられているパスワードを入力します。
4. [作成]をクリックして変更を保存し、VHMを作成します。
5. [仮想ホストマネージャ] ページで、新しいVHMを選択します。
6. [プロパティ] ページで、[データの更新]をクリックし、新しいVHMを評価します。

評価されたオブジェクトおよびリソースを表示するには、システム > システム一覧 > 仮想システムに移動します。



HTTPSを使用してブラウザからESXiサーバに接続すると、無効な証明書エラーがログされる場合があります。このエラーが発生すると、仮想ホストサーバからのデータの更新は失敗します。この問題を修正するには、ESXiサーバから証明書を抽出して /etc/pki/trust/anchors にコピーします。証明書を再度信頼します。そのためには、コマンドラインで `update-ca-certificates` コマンドを実行し、spacewalkサービスを再起動します。

VHMが作成されて設定されると、Taskomaticは、データ収集を自動的に実行します。データ収集を手動で実行する場合、仮想ホストマネージャに移動し、適切なVHMを選択して[データの更新]をクリックします。

APIを使用してVHMに接続して仮想ホストの情報をリクエストできる `virtual-host-gatherer` というツールがUyuniに付属しています。`virtual-host-gatherer` は、オプションモジュールの概念を維持していて、各モジュールが特定のVHMを有效地にします。このツールは、Taskomaticによって毎晩自動的に呼び出されます。`virtual-host-gatherer` ツールのログファイルは `/var/log/rhn/gatherer.log` にあります。

10.6.2. VMWareでのSSLエラーのトラブルシューティング

VMWareのインストール環境を設定中にSSLエラーが発生した場合、VMWareからCA証明書をダウンロードし、Uyuniで信頼する必要があります。

プロシージャ: VMWare CA証明書を信頼する

1. VMWareインストール環境からCA証明書をダウンロードします。そのためには、vCenterのWeb UIにログインし、[Download trusted root CA certificates] (信頼できるルートCA証明書のダウンロード) をクリックします。
2. ダウンロードしたCA証明書ファイルが.zip フォーマットの場合、アーカイブを抽出します。証明書ファイルには拡張子として番号が含まれています。たとえば、certificate.0 のようになります。

3. 証明書ファイルをUyuniサーバにコピーし、`/etc/pki/trust/anchors/` ディレクトリに保存します。
4. コピーした証明書のファイル名のサフィックスを `.crt` または `.pem` に変更します。
5. Uyuniサーバのコマンドプロンプトで、CA証明書のレコードを更新します。

```
update-ca-certificates
```

10.7. その他のサードパーティプロバイダを使用した仮想化

Xen、KVMまたはVMware以外のサードパーティ仮想化プロバイダを使用する場合、JSON設定ファイルをUyuniにインポートできます。

同様に、APIへの直接アクセスを提供しないVMWareインストール環境の場合、ファイルベースのVHMが基本的な管理機能を提供します。



このオプションは、`virtual-host-gatherer` ツールを使用して作成されたファイルをインポートするためのものです。手動で作成したファイル用には設計されていません。

プロセッジヤ: JSONファイルのエクスポートとインポート

1. VMネットワークで `virtual-host-gatherer` を実行してJSON設定ファイルをエクスポートします。
2. 生成されたファイルをUyuniサーバからアクセスできる場所に保存します。
3. UyuniのWeb UIで、**システム > 仮想ホストマネージャ**に移動します。
4. **[作成]** をクリックし、**[ファイルベース]** を選択します。
5. **[Add a file-based Virtual Host Manager]** (ファイルベースの仮想ホストマネージャの追加) セクションで、次のパラメータを使用します。
 - **[ラベル]** フィールドにVHMのカスタム名を入力します。
 - **[Url]** フィールドに、エクスポートするJSON設定ファイルへのパスを入力します。
6. **[作成]** をクリックして変更を保存し、VHMを作成します。
7. **[仮想ホストマネージャ]** ページで、新しいVHMを選択します。
8. **[プロパティ]** ページで、**[データの更新]** をクリックし、新しいVHMを評価します。

リスト 3. 例: エクスポートするJSON設定ファイル

```
{
  "examplevhhost": {
    "10.11.12.13": {
      "cpuArch": "x86_64",
      "cpuDescription": "AMD Opteron(tm) Processor 4386",
      "cpuMhz": 3092.212727,
```

```

    "cpuVendor": "amd",
    "hostIdentifier": "'vim.HostSystem:host-182'",
    "name": "11.11.12.13",
    "os": "VMware ESXi",
    "osVersion": "5.5.0",
    "ramMb": 65512,
    "totalCpuCores": 16,
    "totalCpuSockets": 2,
    "totalCpuThreads": 16,
    "type": "vmware",
    "vms": {
        "vCenter": "564d6d90-459c-2256-8f39-3cb2bd24b7b0"
    }
},
"10.11.12.14": {
    "cpuArch": "x86_64",
    "cpuDescription": "AMD Opteron(tm) Processor 4386",
    "cpuMhz": 3092.212639,
    "cpuVendor": "amd",
    "hostIdentifier": "'vim.HostSystem:host-183'",
    "name": "10.11.12.14",
    "os": "VMware ESXi",
    "osVersion": "5.5.0",
    "ramMb": 65512,
    "totalCpuCores": 16,
    "totalCpuSockets": 2,
    "totalCpuThreads": 16,
    "type": "vmware",
    "vms": {
        "49737e0a-c9e6-4ceb-aef8-6a9452f67cb5": "4230c60f-3f98-2a65-f7c3-600b26b79c22",
        "5a2e4e63-a957-426b-bfa8-4169302e4fdb": "42307b15-1618-0595-01f2-427ffcd88e",
        "NSX-gateway": "4230d43e-aafe-38ba-5a9e-3cb67c03a16a",
        "NSX-l3gateway": "4230b00f-0b21-0e9d-dfde-6c7b06909d5f",
        "NSX-service": "4230e924-b714-198b-348b-25de01482fd9"
    }
}
}
}

```

詳細については、Uyuniサーバの **virtual-host-gatherer** の関数リファレンスを参照してください。

```
man virtual-host-gatherer
```

このパッケージの **README** ファイルには、ハイパーテザの「種類」などに関する背景情報が記載されています。

```
/usr/share/doc/packages/virtual-host-gatherer/README.md
```

この関数リファレンスおよび **README** ファイルには、設定ファイルのサンプルも含まれています。

Chapter 11. クライアントのトラブルシューティング

11.1. 自動インストール

ベースチャンネルによっては、新しい自動インストールプロファイルは、必要なパッケージがないチャンネルにサブスクライブされる場合があります。

自動インストールを動作させるには、次のパッケージが必要です。

- `pyOpenSSL`
- `rhnlib`
- `libxml2-python`
- `spacewalk-koan`

この問題を解決するには、まず次の点について確認してください。

- 自動インストールプロファイルのベースチャンネルに関するツールソフトウェアチャンネルを組織およびユーザーで使用できることを確認します。
- ツールチャンネルが子チャンネルとしてUyuniで使用できることを確認します。
- 必要なパッケージおよび依存関係が関連チャンネルで使用できることを確認します。

11.2. ベアメタルシステム

ネットワークのベアメタルシステムが [システム] リストに自動的に追加されない場合、まず次の点について確認してください。

- `pxe-default-image` パッケージがインストールされている必要があります。
- ファイルのパスおよびパラメータが正しく設定されている必要があります。`rhn.conf` 設定ファイルで指定された場所に、`pxe-default-image` で提供される `vmlinuz0` ファイルと `initrd0.img` ファイルがあることを確認します。
- ベアメタルシステムをUyuniサーバに接続するネットワーク機器が正しく動作していて、そのサーバからUyuniサーバのIPアドレスにアクセスできることを確認します。
- プロビジョニングするベアメタルシステムは、ブートシーケンスでPXEブートが有効になっている必要があり、オペレーティングシステムをブートしていない必要があります。
- DHCPサーバは、ブート中にDHCPリクエストに応答する必要があります。PXEブートメッセージで次の点を確認してください。
 - DHCPサーバが目的のIPアドレスを割り当てている。
 - DHCPサーバがUyuniサーバのIPアドレスをブート用に `next-server` として割り当てている。
- Cobblerが実行中であり検出機能が有効なことを確認してください。

ブート後すぐに青いCobblerメニューが表示される場合、検出が開始されています。 正常に完了しない場合、自動シャットダウンを一時的に無効にして、問題の診断に活用してください。自動シャットダウンを無効にするには:

1. Cobblerメニューで矢印キーを使用して **pxe-default-profile** を選択し、タイマーが切れる前にTabキーを押します。
2. カーネルブートパラメータ **spacewalk-finally=running** を追加します。そのためには、統合されたエディタを使用します。その後、Enterキーを押してブートを続行します。
3. ユーザ名を **root**、パスワードを **linux** にしてシェルに入力し、デバッグを続行します。



重複プロファイル

技術的な制約により、新しいベアメタルシステムを以前に検出されたシステムと高い信頼性で区別することはできません。 したがって、ベアメタルシステムの電源を複数回オンにしないことをお勧めします。この操作を実行するとプロファイルの重複が発生します。

11.3. サポートが終了したCentOS 6クライアントのブートストラップ

CentOS 6はサポート終了になっており、このオペレーティングシステム用にクライアントリポジトリで提供されるイメージは失効しています。 これらのパッケージを使用した新しいCentOS 6クライアントのブートストラップは失敗します。 この障害は、インストールおよびブートストラップが完了しているCentOS 6クライアントでは発生しません。

新しいCentOS 6クライアントをブートストラップする必要がある場合、既存のリポジトリを編集し、正しいURLを反映します。

プロシージャ: 新しいCentOS 6クライアントのブートストラップのトラブルシューティング

1. CentOS 6クライアントのコマンドプロンプトで、**CentOS-Base.repo** ファイルを開きます。このファイルは **/etc/yum.repos.d/** ディレクトリにあります。
2. **mirrorlist** エントリを見つけます。これは **mirrorlist.centos.org** を指しています。エントリは複数存在する可能性があります。次に例を示します。

```
mirrorlist=http://mirrorlist.centos.org/?release=6&arch=$basearch&repo
=os
```

3. **mirrorlist** エントリをコメントアウトし、パッケージマネージャがURLを探さないようにします。
4. **baseurl** 行を編集し、**vault.centos.org** URLを指すようにし、CentOS 6のリポジトリを指定します。次に例を示します。

```
baseurl=https://vault.centos.org/centos/6/os/$basearch/
```

5. ファイルでリストされている各リポジトリで繰り返します。
6. クライアントをブートストラップします。 CentOSクライアントのブートストラップの詳細については、Client-configuration > Clients-centosを参照してください。

CentOS 6のサポート終了の詳細については、<http://mirror.centos.org/centos/6/readme>を参照してください。

11.4. サポート終了製品のブートストラップリポジトリ

サポートされている製品を同期するとき、ブートストラップリポジトリは、自動的に作成され、Uyuniサーバに再生成されます。 製品がサポート終了になり、サポートされなくなったが、製品の使用を継続する場合、ブートストラップリポジトリを手動で作成する必要があります。

ブートストラップリポジトリの詳細については、Client-configuration > Bootstrap-repositoryを参照してください。

プロシージャ: サポート終了製品のブートストラップリポジトリの作成

1. Uyuni サーバのコマンドプロンプトで root になり、**--force** オプションを指定してサポート対象外のブートストラップスクリプトの一覧を表示してください。たとえば下記のようになります:

```
mgr-create-bootstrap-repo --list --force
1. SLE-11-SP4-x86_64
2. SLE-12-SP2-x86_64
3. SLE-12-SP3-x86_64
```

2. 製品ラベルとして適切なリポジトリ名を使用して、ブートストラップリポジトリを作成します。

```
mgr-create-bootstrap-repo --create SLE-12-SP2-x86_64 --force
```

ブートストラップリポジトリを手動で作成しない場合、必要な製品およびブートストラップリポジトリでLTSSが使用できるかどうかを確認できます。

11.5. 複製したSaltクライアント

ハイパーバイザ複製ユーティリティを使用していて、複製したSaltクライアントを登録しようとすると、次のエラーが発生します。

残念ながら、このシステムは見つかりませんでした。

新しい複製システムのマシンIDが既存の登録済みシステムのマシンIDと同じことが原因です。 マシンIDを手動で調整してエラーに対処すると、複製したシステムを正常に登録できます。

詳細および手順については、Administration > Tshoot-registerclonesを参照してください。

11.6. FQDNS grainの無効化

FQDNS grainは、システムのすべての完全修飾DNSサービスのリストを返します。この情報の収集は、通常、高速プロセスですが、DNS設定が間違っていると、長時間かかる可能性があります。場合によっては、クライアントが無応答またはクラッシュする場合があります。

この問題を回避するには、Saltフラグを使用してFQDNS grainを無効にできます。grainを無効にした場合、ネットワークモジュールを使用して、FQDNSサービスを提供できます。この場合、クライアントが無応答になるリスクはありません。



この操作は、古いSaltクライアントにのみ適用されます。最近Saltクライアントを登録した場合、FQDNS grainはデフォルトで無効になっています。

Uyuniサーバのコマンドプロンプトで、次のコマンドを使用してFQDNS grainを無効にします。

```
salt '*' state.sls util.mgr_disable_fqdns_grain
```

このコマンドを実行すると、各クライアントが再起動され、サーバが処理する必要があるSaltイベントが生成されます。クライアント数が多い場合、バッチモードでコマンドを実行できます。

```
salt --batch-size 50 '*' state.sls util.mgr_disable_fqdns_grain
```

バッチコマンドの実行完了を待機します。 **Ctrl**+**C** でプロセスを中断しないでください。

11.7. noexecで/tmpをマウントする

Saltは、クライアントのファイルシステムの **/tmp** からリモートコマンドを実行します。したがって、**/tmp** に **noexec** オプションをマウントしないでください。

11.8. grainを渡してイベントを開始する

Saltクライアントは、起動するたびに、**machine_id** grainをUyuniに渡します。Uyuniは、このgrainを使用して、クライアントが登録されたかどうかを判定します。このプロセスでは、同期Saltコールが必要です。同期Saltコールは、その他のプロセスをブロックするため、多数のクライアントを同時に起動する場合、大幅な遅延が発生する可能性があります。

この問題を克服するために、別々のSaltコールを回避するための新しい機能がSaltに導入されました。

この機能を使用するには、この機能をサポートしているクライアントのクライアント設定に設定パラメータを追加できます。

このプロセスを簡単にするには、**mgr_start_event_grains.sls** ヘルパーSaltの状態を使用します。



この操作は、登録済みのクライアントにのみ適用されます。最近Saltクライアントを登録した場合、この設定パラメータはデフォルトで追加されています。

Uyuniサーバのコマンドプロンプトで、次のコマンドを使用して `start_event_grains` 設定ヘルパーを有効にします。

```
salt '*' state.sls util.mgr_start_event_grains
```

このコマンドを実行すると、必要な設定がクライアントの設定ファイルに追加され、クライアントを再起動したときに適用されます。クライアント数が多い場合、バッチモードでコマンドを実行できます。

```
salt --batch-size 50 '*' state.sls mgr_start_event_grains
```

11.9. プロキシの接続およびFQDN

プロキシ経由で接続されているクライアントがWeb UIに表示されることがあります、そのようなクライアントがプロキシ経由で接続されていることは示されません。完全修飾ドメイン名(FQDN)を使用して接続していない場合、Uyuniでプロキシが認識されないと、この動作が発生することがあります。

この動作を修正するには、プロキシのクライアント設定ファイルでgrainとして追加のFQDNを指定します。

```
grains:
  susemanager:
    custom_fqdns:
      - name.one
      - name.two
```

11.10. Red Hat CDNチャンネルと複数の証明書

Red Hatコンテンツデリバリネットワーク(CDN)チャンネルは複数の証明書を提供することができますが、Uyuni Web UIは単一の証明書しかインポートできません。CDNがUyuni Web UIで認識されている証明書とは異なる証明書を提示した場合、証明書が正確であっても検証が失敗し、リポジトリへのアクセスパーミッションが拒否されます。次のようなエラーメッセージが生じます。

```
[error] ([エラー])
Repository '<repo_name>' is invalid. (リポジトリ'<repo_name>'は無効です。)
<repo.pem> Valid metadata not found at specified
URL (有効なメタデータが指定されているURLで見つかりませんでした)
History: (履歴:)
- [] Error trying to read from '<repo.pem>' (
'<repo.pem>'からの読み込み時にエラーが発生しました)
- Permission to access '<repo.pem>' denied. (
'<repo.pem>'へのアクセスが拒否されました。)
Please check if the URIs defined for this repository are pointing to a
valid repository. (このリポジトリ用に定義されている
URIが有効なリポジトリを指しているかどうかを確認してください。
Skipping repository '<repo_name>' because of the above
error. (上記エラーのため、リポジトリ'<repo_name>'をスキップします。)
Could not refresh the repositories because of
errors. (エラーが発生したため、リポジトリを更新できませんでした。)
HH:MM:SS RepoMDError: Cannot access repository. Maybe repository GPG keys
are not imported (HH:MM:SS RepoMDError:
リポジトリにアクセスできません。リポジトリGPGキーがインポートされていない可能性があります)
```

この問題を解決するには、すべての有効な証明書を1つの **.pem** ファイルにマージし、Uyuniで使用する証明書を作成します。

手順: 複数のRed Hat CDN証明書の解決

1. Red Hat クライアントのコマンドプロンプトで、rootとして、**/etc/pki/entitlement/** からすべての現行の証明書を単一の **rh-cert.pem** ファイルに収集します。

```
cat 866705146090697087.pem 3539668047766796506.pem redhat-entitlement-
authority.pem > rh-cert.pem
```

2. **/etc/pki/entitlement/** からすべての現行のキーを単一の **rh-key.pem** ファイルに収集します。

```
cat 866705146090697087-key.pem 3539668047766796506-key.pem > rh-
key.pem
```

次に、**Client-configuration > Clients-rh-cdn** の手順に従って、新しい証明書を Uyuni サーバにインポートできます。

11.11. Web UIからの登録は失敗するが、エラーが表示されない

Web UIからの初期登録では、すべてのSaltクライアントがSalt SSHを使用しています。

その性質上、Salt SSHクライアントはサーバにエラーを報告しません。

ただし、Salt SSHクライアントは、エラーを検査できるログを `/var/log/salt-ssh.log` にローカルに保存します。

11.12. 古いクライアントの登録

CentOS 6、Oracle Linux 6、Red Hat Enterprise Linux 6、SUSE Linux Enterprise Server with Expanded Support 6、またはSUSE Linux Enterprise Server11の各クライアントを登録して使用するには、Uyuni サーバを設定して旧式のSSL暗号化をサポートする必要があります。

コマンドプロンプトで登録しようとすると、次のような内容を示すエラーが発生します。

```
Repository '<Repository_Name>' is invalid.
(リポジトリ'<Repository_Name>'は無効です。)
[!] Valid metadata not found at specified URL(s)
Please check if the URIs defined for this repository are pointing to a
valid repository.
(有効なメタデータが指定されているURLで見つかりませんでした。
このリポジトリ用に定義されているURIが有効なリポジトリを指しているかどうかを確認してください。
) Skipping repository '<Repository_Name>' because of the above error.
(上記のエラーのため、リポジトリ'<Repository_Name>'をスキップします。
) Download (curl) error for 'www.example.com': ('www.example.com' に
(curl)エラーをダウンロードします:
) Error code: (エラーコード:) Unrecognized error
Error message: error:1409442E:SSL routines:SSL3_READ_BYTES:tlsv1 alert
protocol version
(認識できないエラー
エラーメッセージ: エラー:1409442E:SSLルーチン:SSL3_READ_BYTES:tlsv1
警告プロトコルバージョン)
```

Web UIで登録しようとすると、次のような内容を示すエラーが発生します。

```
Rendering SLS 'base:bootstrap' failed: (SLS
'base:bootstrap'のレンダリングに失敗しました:) Jinja error: (Jinjaエラー:) >>>
No TLS 1.2 and above for RHEL6 and SLES11. (>>> RHEL6およびSLES11ではTLS
1.2以降は使用できません。) Please check your Apache config. (
Apache設定を確認してください。)
...
```

ApacheではTLS v1.2を必要とするため、この動作が発生しますが、古いオペレーティングシステムは、このバージョンのTLSプロトコルをサポートしていません。このエラーを修正するには、サーバ上のApacheが広範なプロトコルバージョンを受け入れる必要があります。Uyuniサーバでrootとして `/etc/apache2/ssl-global.conf` 設定ファイルを開き、`SSLProtocol` 行を検索し、次のように更新します。

```
SSLProtocol all -SSLv2 -SSLv3
```

この操作は、サーバ上で手動で実行する必要があります。その際、該当する場合には、プロキシでSaltの状態にします。変更後、各システムで `apache` サービスを再起動します。

11.13. ダウンとして表示されるSaltクライアントおよびDNS設定

Saltクライアントが実行されている場合でも、パッケージの更新や状態の適用などのアクションは、次のメッセージで失敗としてマークされる可能性があります。

Minionがダウンしているか、接続できませんでした。

この場合、アクションのスケジュールを変更してみてください。スケジュールの変更が成功した場合、問題の原因是DNS設定の誤りである可能性があります。

Saltクライアントが再起動されたとき、またはグレインが更新された場合、クライアントはFQDNグレインを計算し、グレインが続行されるまで応答しません。Uyuniサーバでスケジュールされたアクションが実行される場合、Uyuniサーバは、実際のアクションの前にクライアントに対して `test.ping` を実行して、クライアントが実際に実行され、アクションをトリガできることを確認します。

デフォルトでは、Uyuniサーバは `test.ping` コマンドからの応答を取得するために5秒間待機します。5秒以内に応答が受信されない場合、アクションは失敗に設定され、クライアントがダウンしているか、接続できなかったというメッセージが表示されます。

これを修正するには、クライアントのDNS解決を修正して、クライアントがFQDNの解決中に5秒間スタックしないようにします。

これができない場合は、Uyuniサーバ上の `/etc/rhn/rhn.conf` ファイルの `java.salt_presence_ping_timeout` の値を4より大きい値に増やしてみてください。

例:

```
java.salt_presence_ping_timeout = 6
```

その後、次のコマンドを使用して `spacewalk-services` を再起動します。

```
spacewalk-services restart
```



この値を大きくすると、minionに到達できないのかminionが応答しないのかをUyuniサーバが確認するのに時間がかかり、Uyuniサーバの全体的な速度が低下したり応答しなくなったりします。

Chapter 12. GNU Free Documentation License

Copyright © 2000, 2001, 2002 Free Software Foundation, Inc. 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA. Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
- C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
- D. Preserve all the copyright notices of the Document.
- E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in

the Document's license notice.

- H. Include an unaltered copy of this License.
 - I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
 - J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
 - K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
 - L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
 - M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.
 - N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.
 - O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the

Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See <http://www.gnu.org/copyleft/>.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

Copyright (c) YEAR YOUR NAME.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.

A copy of the license is included in the section entitled{ldquo}GNU Free Documentation License{rdquo}.

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with...Texts." line with this:

with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.