# Table of Contents

GNU Free Documentation License .......................... 1
Introduction .................................................. 8

## Supported Clients and Features

- Supported Client Systems ........................................ 9
- Supported SUSE and openSUSE Client Features .................. 10
- Supported SUSE Linux Enterprise Server with Expanded Support Features ........................................ 13
- Supported Red Hat Enterprise Linux Features .................. 15
- Supported CentOS Features ..................................... 18
- Supported Ubuntu Features ..................................... 20

## SLE Client Registration Overview

- Register Clients with the Web UI ................................ 23
- SLES 11 SSL errors .............................................. 24
- Register Clients with a Bootstrap Script ....................... 24
  - Create a Bootstrap Script ................................... 25
  - Editing a Bootstrap Script ................................ 26
  - Connect Clients ............................................. 26
  - Package Locks ............................................... 27
- Register on the Command Line (Salt) ............................ 28
- SLES 11 SSL errors .............................................. 29
- Activation Keys ................................................ 29
  - Combining Activation Keys ................................... 31
  - Activation Key Best Practices ................................ 32

## SLE Client Registration on a Proxy

- Register in the Web UI (Salt Only) ......................... 35
- Register on the Command Line (Salt) ......................... 36
- Registering with a Bootstrap Script (Salt and Traditional) ................................................................ 37

## Automate Client Installation

- Preparation .................................................... 38
  - Upload a Profile ............................................. 39
  - Variables and snippets ..................................... 40
- Autoinstallation with AutoYaST ................................ 41
  - Bare Metal Provisioning ..................................... 41
  - API Provisioning ............................................. 42
  - Advanced PXE Installation Configuration .................. 42
- Kickstart ....................................................... 42
  - Before you Begin ............................................ 43
  - Build a Bootable ISO ........................................ 43
  - Integrating with PXE .......................................... 44

## Cobbler

- Cobbler Requirements .......................................... 45
- Configure Cobbler ............................................... 45
- TFTP ............................................................. 47
- Synchronize and Start the Cobbler Service .................... 48
- Autoinstallation Templates ..................................... 49
- Build ISOs with Cobbler ....................................... 51
- Bare Metal Provisioning ....................................... 51
<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>VHM and Kubernetes</td>
<td>90</td>
</tr>
<tr>
<td>Create a Kubernetes VHM</td>
<td>90</td>
</tr>
<tr>
<td>Retrieve Image Runtime Data</td>
<td>91</td>
</tr>
<tr>
<td>Permissions and Certificates</td>
<td>93</td>
</tr>
<tr>
<td>VHM and SUSE CaaS Platform</td>
<td>94</td>
</tr>
<tr>
<td>Onboarding CaaS nodes</td>
<td>94</td>
</tr>
<tr>
<td>Software Channels</td>
<td>96</td>
</tr>
<tr>
<td>Custom Channels</td>
<td>96</td>
</tr>
<tr>
<td>Bootstrap Repository</td>
<td>97</td>
</tr>
<tr>
<td>Prepare to Create a Bootstrap Repository</td>
<td>97</td>
</tr>
<tr>
<td>Options for Automatic Mode</td>
<td>97</td>
</tr>
<tr>
<td>Flush Mode</td>
<td>97</td>
</tr>
<tr>
<td>Automatic Mode</td>
<td>97</td>
</tr>
<tr>
<td>Configure Bootstrap Data File</td>
<td>98</td>
</tr>
<tr>
<td>Manually Generate a Bootstrap Repository</td>
<td>98</td>
</tr>
<tr>
<td>Contact Methods</td>
<td>100</td>
</tr>
<tr>
<td>SUSE Manager Daemon (rhnsd)</td>
<td>100</td>
</tr>
<tr>
<td>Configure rhnsd</td>
<td>101</td>
</tr>
<tr>
<td>Push via SSH</td>
<td>102</td>
</tr>
<tr>
<td>Push via Salt SSH</td>
<td>106</td>
</tr>
<tr>
<td>OSAD</td>
<td>107</td>
</tr>
<tr>
<td>Using the System Set Manager</td>
<td>110</td>
</tr>
<tr>
<td>Setting up System Set Manager</td>
<td>110</td>
</tr>
<tr>
<td>Using System Set Manager</td>
<td>110</td>
</tr>
<tr>
<td>Troubleshooting Clients</td>
<td>111</td>
</tr>
<tr>
<td>Bare Metal Systems</td>
<td>111</td>
</tr>
<tr>
<td>Cloned Salt Clients</td>
<td>111</td>
</tr>
<tr>
<td>Disabling the FQDNS grain</td>
<td>112</td>
</tr>
<tr>
<td>Mounting /tmp with noexec</td>
<td>112</td>
</tr>
<tr>
<td>Passing Grains to a Start Event</td>
<td>112</td>
</tr>
<tr>
<td>Proxy Connections and FQDN</td>
<td>113</td>
</tr>
<tr>
<td>AutoYast Example File</td>
<td>114</td>
</tr>
<tr>
<td>Minimalist AutoYaST Profile for Automated Installations and Useful Enhancements</td>
<td>114</td>
</tr>
</tbody>
</table>
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 worldwide, 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:
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.
Registering clients is the first step after installing Uyuni, and most of the time you spend with Uyuni will be spent on maintaining those clients.

Uyuni is compatible with a range of client technologies: you can install traditional or Salt clients, running SUSE Linux Enterprise or another Linux operating system, with a range of hardware options.

For a complete list of supported clients and features, see [Client-configuration › Supported-features › ].

This guide discusses how to register and configure different clients, both manually and automatically.
Supported Clients and Features

Uyuni is compatible with a range of client technologies. You can install traditional or Salt clients, running SUSE Linux Enterprise or another Linux operating system, with a range of hardware options.

This section contains summary of supported client systems. For a detailed list of features available on each client, see the following pages.

Supported Client Systems

Supported operating systems for traditional and Salt clients are listed in this table.

The icons in this table indicate:

- ✓ clients running this operating system are supported by SUSE
- ✗ clients running this operating system are not supported by SUSE
- ? clients are under consideration, and may or may not be supported at a later date.

Client operating system versions and SP levels must be under general support (normal or LTSS) to be supported with Uyuni. For details on supported product versions, see https://www.suse.com/lifecycle.

Table 1. Supported Client Systems

<table>
<thead>
<tr>
<th>Operating System</th>
<th>Architecture</th>
<th>Traditional Clients</th>
<th>Salt Clients</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUSE Linux Enterprise 15</td>
<td>x86_64, POWER, IBM Z, ARM</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>SUSE Linux Enterprise 12</td>
<td>x86_64, POWER, IBM Z, ARM</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>SUSE Linux Enterprise 11</td>
<td>x86, x86_64, Itanium, IBM POWER, IBM Z</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>SUSE Linux Enterprise Server for SAP</td>
<td>x86_64, POWER</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>openSUSE Leap 15.1</td>
<td>x86_64</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>SUSE Linux Enterprise Server-ES 8</td>
<td>x86_64</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>SUSE Linux Enterprise Server-ES 7</td>
<td>x86_64</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>SUSE Linux Enterprise Server-ES 6</td>
<td>x86, x86_64</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
Supported SUSE and openSUSE Client Features

This table lists the availability of various features on SUSE and openSUSE clients. This table covers all variants of the SUSE Linux Enterprise operating system, including SLES, SLED, SUSE Linux Enterprise Server for SAP, and SUSE Linux Enterprise Server for HPC.

The icons in this table indicate:

- ✔️ the feature is available on both traditional and Salt clients
- ✗ the feature is not available
- 🎨 the feature is under consideration, and may or may not be made available at a later date
- Traditional the feature is supported only on traditional clients
- Salt the feature is supported only on Salt clients.

<table>
<thead>
<tr>
<th>Feature</th>
<th>SUSE Linux Enterprise 11</th>
<th>SUSE Linux Enterprise 12</th>
<th>SUSE Linux Enterprise 15</th>
<th>openSUSE 15.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Operating system packages</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Registration</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>Salt</td>
</tr>
<tr>
<td>Install packages</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>Salt</td>
</tr>
<tr>
<td>Feature</td>
<td>SUSE Linux Enterprise 11</td>
<td>SUSE Linux Enterprise 12</td>
<td>SUSE Linux Enterprise 15</td>
<td>openSUSE 15.1</td>
</tr>
<tr>
<td>----------------------------------------</td>
<td>--------------------------</td>
<td>--------------------------</td>
<td>--------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>Apply patches</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>Salt</td>
</tr>
<tr>
<td>Remote commands</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>Salt</td>
</tr>
<tr>
<td>System package states</td>
<td>Salt</td>
<td>Salt</td>
<td>Salt</td>
<td>Salt</td>
</tr>
<tr>
<td>System custom states</td>
<td>Salt</td>
<td>Salt</td>
<td>Salt</td>
<td>Salt</td>
</tr>
<tr>
<td>Group custom states</td>
<td>Salt</td>
<td>Salt</td>
<td>Salt</td>
<td>Salt</td>
</tr>
<tr>
<td>Organization custom states</td>
<td>Salt</td>
<td>Salt</td>
<td>Salt</td>
<td>Salt</td>
</tr>
<tr>
<td>System set manager (SSM)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>Salt</td>
</tr>
<tr>
<td>Service pack migration</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>Salt</td>
</tr>
<tr>
<td>Basic Virtual Guest Management 🌱</td>
<td>Traditional</td>
<td>✓</td>
<td>✓</td>
<td>‘Salt</td>
</tr>
<tr>
<td>Advanced Virtual Guest Management 🌱</td>
<td>✗</td>
<td>Salt</td>
<td>Salt</td>
<td>Salt</td>
</tr>
<tr>
<td>Virtual Guest Installation (AutoYaST), as Host OS</td>
<td>Traditional</td>
<td>Traditional</td>
<td>Traditional</td>
<td>✗</td>
</tr>
<tr>
<td>Virtual Guest Installation (image template), as Host OS</td>
<td>✗</td>
<td>Salt</td>
<td>Salt</td>
<td>Salt</td>
</tr>
<tr>
<td>Virtual Guest Management</td>
<td>✗</td>
<td>Salt</td>
<td>Salt</td>
<td>Salt</td>
</tr>
<tr>
<td>System deployment (PXE/AutoYaST)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>System redeployment (AutoYaST)</td>
<td>Traditional</td>
<td>✓</td>
<td>✓</td>
<td>Salt</td>
</tr>
<tr>
<td>Feature</td>
<td>SUSE Linux Enterprise 11</td>
<td>SUSE Linux Enterprise 12</td>
<td>SUSE Linux Enterprise 15</td>
<td>openSUSE 15.1</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>--------------------------</td>
<td>--------------------------</td>
<td>--------------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>Contact methods</td>
<td><strong>Traditional:</strong> OSAD, RHNSD, SSH-push.</td>
<td><strong>Traditional:</strong> OSAD, RHNSD, SSH-push.</td>
<td><strong>Traditional:</strong> OSAD, RHNSD, SSH-push.</td>
<td><strong>Salt:</strong> ZeroMQ, Salt-SSH</td>
</tr>
<tr>
<td></td>
<td><strong>Salt:</strong> ZeroMQ, Salt-SSH</td>
<td><strong>Salt:</strong> ZeroMQ, Salt-SSH</td>
<td><strong>Salt:</strong> ZeroMQ, Salt-SSH</td>
<td></td>
</tr>
<tr>
<td>Works with Uyuni Proxy</td>
<td>✅</td>
<td>✅</td>
<td>✅</td>
<td><strong>Salt</strong></td>
</tr>
<tr>
<td>Action chains</td>
<td>✅</td>
<td>✅</td>
<td>✅</td>
<td><strong>Salt</strong></td>
</tr>
<tr>
<td>Software crash reporting</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Staging (pre-download of packages)</td>
<td>✅</td>
<td>✅</td>
<td>✅</td>
<td><strong>Salt</strong></td>
</tr>
<tr>
<td>Duplicate package reporting</td>
<td>✅</td>
<td>✅</td>
<td>✅</td>
<td><strong>Salt</strong></td>
</tr>
<tr>
<td>CVE auditing</td>
<td>✅</td>
<td>✅</td>
<td>✅</td>
<td><strong>Salt</strong></td>
</tr>
<tr>
<td>SCAP auditing</td>
<td>✅</td>
<td>✅</td>
<td>✅</td>
<td><strong>Salt</strong></td>
</tr>
<tr>
<td>Package verification</td>
<td><strong>Traditional</strong></td>
<td><strong>Traditional</strong></td>
<td><strong>Traditional</strong></td>
<td>✗</td>
</tr>
<tr>
<td>Package locking</td>
<td><strong>Traditional</strong></td>
<td><strong>Traditional</strong></td>
<td><strong>Traditional</strong></td>
<td>✗</td>
</tr>
<tr>
<td>System locking</td>
<td><strong>Traditional</strong></td>
<td><strong>Traditional</strong></td>
<td><strong>Traditional</strong></td>
<td>✗</td>
</tr>
<tr>
<td>System snapshot</td>
<td><strong>Traditional</strong></td>
<td><strong>Traditional</strong></td>
<td><strong>Traditional</strong></td>
<td>✗</td>
</tr>
<tr>
<td>Configuration file management</td>
<td>✅</td>
<td>✅</td>
<td>✅</td>
<td><strong>Salt</strong></td>
</tr>
<tr>
<td>Package profiles</td>
<td><strong>Traditional.</strong> Salt: Profiles supported, Sync not supported</td>
<td><strong>Traditional.</strong> Salt: Profiles supported, Sync not supported</td>
<td><strong>Traditional.</strong> Salt: Profiles supported, Sync not supported</td>
<td><strong>Salt:</strong> Profiles supported, Sync not supported</td>
</tr>
<tr>
<td>Power management</td>
<td>✅</td>
<td>✅</td>
<td>✅</td>
<td>✅</td>
</tr>
<tr>
<td>Monitoring</td>
<td>✕</td>
<td><strong>Salt</strong></td>
<td><strong>Salt</strong></td>
<td><strong>Salt</strong></td>
</tr>
<tr>
<td>Docker buildhost</td>
<td>✕</td>
<td><strong>Salt</strong></td>
<td><strong>Salt</strong></td>
<td>✕</td>
</tr>
</tbody>
</table>
Virtual Guest Management:

In this table, virtual guest management is split into basic and advanced.

Basic virtual guest management includes listing VMs, slow refresh, VM lifecycle actions (start, stop, resume, pause), and modifying VM vCPU and Memory.

Advanced virtual guest management includes fast refresh, VM lifecycle actions (delete, reset, power off), modifying VM disk, network, graphical display, and graphical display configuration.

Supported SUSE Linux Enterprise Server with Expanded Support Features

This table lists the availability of various features on SUSE Linux Enterprise Server with Expanded Support clients.

The icons in this table indicate:

- ✔️ the feature is available on both traditional and Salt clients
- ✗ the feature is not available
- ❓ the feature is under consideration, and may or may not be made available at a later date
- Traditional the feature is supported only on traditional clients
- Salt the feature is supported only on Salt clients.

<table>
<thead>
<tr>
<th>Feature</th>
<th>SUSE Linux Enterprise 11</th>
<th>SUSE Linux Enterprise 12</th>
<th>SUSE Linux Enterprise 15</th>
<th>openSUSE 15.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Build Docker image with OS</td>
<td>✗</td>
<td>Salt</td>
<td>Salt</td>
<td>Salt</td>
</tr>
<tr>
<td>Kiwi buildhost</td>
<td>✗</td>
<td>Salt</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>Build Kiwi image with OS</td>
<td>✗</td>
<td>Salt</td>
<td>?</td>
<td>✗</td>
</tr>
</tbody>
</table>

Table 3. Supported Features on SUSE Linux Enterprise Server with Expanded Support Operating Systems

<table>
<thead>
<tr>
<th>Feature</th>
<th>SLES ES 6</th>
<th>SLES ES 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Operating system packages</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Registration</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Install packages</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Feature</td>
<td>SLES ES 6</td>
<td>SLES ES 7</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>-----------</td>
<td>-----------</td>
</tr>
<tr>
<td>Apply patches</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Remote commands</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>System package states</td>
<td>Salt</td>
<td>Salt</td>
</tr>
<tr>
<td>System custom states</td>
<td>Salt</td>
<td>Salt</td>
</tr>
<tr>
<td>Group custom states</td>
<td>Salt</td>
<td>Salt</td>
</tr>
<tr>
<td>Organization custom states</td>
<td>Salt</td>
<td>Salt</td>
</tr>
<tr>
<td>System set manager (SSM)</td>
<td>Salt</td>
<td>Salt</td>
</tr>
<tr>
<td>Service pack migration</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Basic Virtual Guest Management *</td>
<td>Traditional</td>
<td>✓</td>
</tr>
<tr>
<td>Advanced Virtual Guest Management ✗</td>
<td>✗</td>
<td>Salt</td>
</tr>
<tr>
<td>Virtual Guest Installation (Kickstart), as Host OS</td>
<td>Traditional</td>
<td>Traditional</td>
</tr>
<tr>
<td>Virtual Guest Installation (image template), as Host OS</td>
<td>Traditional</td>
<td>✓</td>
</tr>
<tr>
<td>System deployment (PXE/Kickstart)</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>System redeployment (Kickstart)</td>
<td>Traditional</td>
<td>✓</td>
</tr>
<tr>
<td>Works with Uyuni Proxy</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Action chains</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Software crash reporting</td>
<td>✗</td>
<td>Traditional</td>
</tr>
<tr>
<td>Staging (pre-download of packages)</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Duplicate package reporting</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>CVE auditing</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>SCAP auditing</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Feature</td>
<td>SLES ES 6</td>
<td>SLES ES 7</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>-----------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>Package verification</td>
<td>Traditional</td>
<td>Traditional</td>
</tr>
<tr>
<td>Package locking</td>
<td>Traditional</td>
<td>Traditional</td>
</tr>
<tr>
<td>System locking</td>
<td>Traditional</td>
<td>Traditional</td>
</tr>
<tr>
<td>System snapshot</td>
<td>Traditional</td>
<td>Traditional</td>
</tr>
<tr>
<td>Configuration file management</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Snapshots and profiles</td>
<td>Traditional.</td>
<td>Traditional.</td>
</tr>
<tr>
<td>Salt: Profiles supported, Sync</td>
<td>Salt: Profiles supported, Sync not supported</td>
<td>not supported</td>
</tr>
<tr>
<td>Power management</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Monitoring</td>
<td>Salt</td>
<td>Salt</td>
</tr>
<tr>
<td>Docker buildhost</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Build Docker image with OS</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>Kiwi buildhost</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Build Kiwi image with OS</td>
<td>✗</td>
<td>✗</td>
</tr>
</tbody>
</table>

**Virtual Guest Management:**

In this table, virtual guest management is split into basic and advanced.

Basic virtual guest management includes listing VMs, slow refresh, VM lifecycle actions (start, stop, resume, pause), and modifying VM vCPU and Memory.

Advanced virtual guest management includes fast refresh, VM lifecycle actions (delete, reset, power off), modifying VM disk, network, graphical display, and graphical display configuration.

**Supported Red Hat Enterprise Linux Features**

This table lists the availability of various features on native Red Hat Enterprise Linux clients (without Expanded Support).

The icons in this table indicate:

- ✔ the feature is available on both traditional and Salt clients
- ✗ the feature is not available
- ✎ the feature is under consideration, and may or may not be made available at a later date
- **Traditional** the feature is supported only on traditional clients
- **Salt** the feature is supported only on Salt clients.

### Table 4. Supported Features on Red Hat Enterprise Linux Operating Systems

<table>
<thead>
<tr>
<th>Feature</th>
<th>RHEL 6</th>
<th>RHEL 7</th>
<th>RHEL 8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client</td>
<td>✔️</td>
<td>✔️</td>
<td>Salt</td>
</tr>
<tr>
<td>Operating system packages</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Registration</td>
<td>✔️</td>
<td>✔️</td>
<td>Salt</td>
</tr>
<tr>
<td>Install packages</td>
<td>✔️</td>
<td>✔️</td>
<td>Salt</td>
</tr>
<tr>
<td>Apply patches</td>
<td>✔️</td>
<td>✔️</td>
<td>Salt</td>
</tr>
<tr>
<td>Remote commands</td>
<td>✔️</td>
<td>✔️</td>
<td>Salt</td>
</tr>
<tr>
<td>System package states</td>
<td>Salt</td>
<td>Salt</td>
<td>Salt</td>
</tr>
<tr>
<td>System custom states</td>
<td>Salt</td>
<td>Salt</td>
<td>Salt</td>
</tr>
<tr>
<td>Group custom states</td>
<td>Salt</td>
<td>Salt</td>
<td>Salt</td>
</tr>
<tr>
<td>Organization custom states</td>
<td>Salt</td>
<td>Salt</td>
<td>Salt</td>
</tr>
<tr>
<td>System set manager (SSM)</td>
<td>Salt</td>
<td>Salt</td>
<td>Salt</td>
</tr>
<tr>
<td>Service pack migration</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Basic Virtual Guest Management 🃁</td>
<td>Traditional</td>
<td>✔️</td>
<td>Salt</td>
</tr>
<tr>
<td>Advanced Virtual Guest Management 🃁</td>
<td>✗</td>
<td>Salt</td>
<td>Salt</td>
</tr>
<tr>
<td>Virtual Guest Installation (Kickstart), as Host OS</td>
<td>Traditional</td>
<td>Traditional</td>
<td>✗</td>
</tr>
<tr>
<td>Virtual Guest Installation (image template), as Host OS</td>
<td>Traditional</td>
<td>✔️</td>
<td>Salt</td>
</tr>
<tr>
<td>System deployment (PXE/Kickstart)</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>System redeployment (Kickstart)</td>
<td>Traditional</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Feature</td>
<td>RHEL 6</td>
<td>RHEL 7</td>
<td>RHEL 8</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>---------------------------------------------</td>
<td>---------------------------------------------</td>
<td>---------------------------------------------</td>
</tr>
<tr>
<td>Contact methods</td>
<td><strong>Traditional</strong>: OSAD, RHNSD, SSH-push.</td>
<td><strong>Traditional</strong>: OSAD, RHNSD, SSH-push.</td>
<td><strong>Salt</strong>: ZeroMQ, Salt-SSH</td>
</tr>
<tr>
<td></td>
<td><strong>Salt</strong>: ZeroMQ, Salt-SSH</td>
<td><strong>Salt</strong>: ZeroMQ, Salt-SSH</td>
<td></td>
</tr>
<tr>
<td>Works with Uyuni Proxy</td>
<td>![✔️]</td>
<td>![✔️]</td>
<td><strong>Salt</strong></td>
</tr>
<tr>
<td>Action chains</td>
<td>![✔️]</td>
<td>![✔️]</td>
<td><strong>Salt</strong></td>
</tr>
<tr>
<td>Software crash reporting</td>
<td>![✗]</td>
<td>![Traditional]</td>
<td>![✗]</td>
</tr>
<tr>
<td>Staging (pre-download of packages)</td>
<td>![✔️]</td>
<td>![✔️]</td>
<td><strong>Salt</strong></td>
</tr>
<tr>
<td>Duplicate package reporting</td>
<td>![✔️]</td>
<td>![✔️]</td>
<td><strong>Salt</strong></td>
</tr>
<tr>
<td>CVE auditing</td>
<td>![✔️]</td>
<td>![✔️]</td>
<td><strong>Salt</strong></td>
</tr>
<tr>
<td>SCAP auditing</td>
<td>![✔️]</td>
<td>![✔️]</td>
<td><strong>Salt</strong></td>
</tr>
<tr>
<td>Package verification</td>
<td><strong>Traditional</strong></td>
<td><strong>Traditional</strong></td>
<td>![✗]</td>
</tr>
<tr>
<td>Package locking</td>
<td><strong>Traditional</strong></td>
<td><strong>Traditional</strong></td>
<td>![✗]</td>
</tr>
<tr>
<td>System locking</td>
<td><strong>Traditional</strong></td>
<td><strong>Traditional</strong></td>
<td>![✗]</td>
</tr>
<tr>
<td>System snapshot</td>
<td><strong>Traditional</strong></td>
<td><strong>Traditional</strong></td>
<td>![✗]</td>
</tr>
<tr>
<td>Configuration file management</td>
<td>![✔️]</td>
<td>![✔️]</td>
<td><strong>Salt</strong></td>
</tr>
<tr>
<td>Snapshots and profiles</td>
<td><strong>Traditional, Salt</strong>: Profiles supported, Sync not supported</td>
<td><strong>Traditional, Salt</strong>: Profiles supported, Sync not supported</td>
<td><strong>Salt</strong>: Profiles supported, Sync not supported</td>
</tr>
<tr>
<td>Power management</td>
<td>![✔️]</td>
<td>![✔️]</td>
<td><strong>Salt</strong></td>
</tr>
<tr>
<td>Monitoring</td>
<td><strong>Salt</strong></td>
<td><strong>Salt</strong></td>
<td><strong>Salt</strong></td>
</tr>
<tr>
<td>Docker buildhost</td>
<td>![✗]</td>
<td>![✗]</td>
<td>![✗]</td>
</tr>
<tr>
<td>Build Docker image with OS</td>
<td>![?]</td>
<td>![?]</td>
<td>![?]</td>
</tr>
<tr>
<td>Kiwi buildhost</td>
<td>![✗]</td>
<td>![✗]</td>
<td>![✗]</td>
</tr>
<tr>
<td>Build Kiwi image with OS</td>
<td>![✗]</td>
<td>![✗]</td>
<td>![✗]</td>
</tr>
</tbody>
</table>
Virtual Guest Management:

In this table, virtual guest management is split into basic and advanced.

Basic virtual guest management includes listing VMs, slow refresh, VM lifecycle actions (start, stop, resume, pause), and modifying VM vCPU and Memory.

Advanced virtual guest management includes fast refresh, VM lifecycle actions (delete, reset, power off), modifying VM disk, network, graphical display, and graphical display configuration.

Supported CentOS Features

This table lists the availability of various features on CentOS clients.

CentOS is not currently an officially supported client operating system. It may or may not be supported in a future version of Uyuni. However, CentOS with an Expanded Support subscription is supported.

The icons in this table indicate:

- ✔ the feature is available on both traditional and Salt clients
- ✗ the feature is not available
- ? the feature is under consideration, and may or may not be made available at a later date
- Traditional the feature is supported only on traditional clients
- Salt the feature is supported only on Salt clients.

Table 5. Supported Features on CentOS Operating Systems

<table>
<thead>
<tr>
<th>Feature</th>
<th>CentOS 6</th>
<th>CentOS 7</th>
<th>CentOS 8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client</td>
<td>![CentOS](plain CentOS) / ✔️ (with Expanded Support)</td>
<td>![CentOS](plain CentOS) / ✔️ (with Expanded Support)</td>
<td>![CentOS](plain CentOS) / Salt (with Expanded Support)</td>
</tr>
<tr>
<td>Operating system packages</td>
<td>![CentOS](plain CentOS) / ✔️ (with Expanded Support)</td>
<td>![CentOS](plain CentOS) / ✔️ (with Expanded Support)</td>
<td>![CentOS](plain CentOS) / Salt (with Expanded Support)</td>
</tr>
<tr>
<td>Registration</td>
<td>✔️</td>
<td>✔️</td>
<td>Salt</td>
</tr>
<tr>
<td>Install packages</td>
<td>✔️</td>
<td>✔️</td>
<td>Salt</td>
</tr>
<tr>
<td>Apply patches (requires CVE ID)</td>
<td>✔️ (third-party service required for errata)</td>
<td>✔️ (third-party service required for errata)</td>
<td>Salt (third-party service required for errata)</td>
</tr>
<tr>
<td>Remote commands</td>
<td>✔️</td>
<td>✔️</td>
<td>Salt</td>
</tr>
<tr>
<td>System package states</td>
<td>Salt</td>
<td>Salt</td>
<td>Salt</td>
</tr>
<tr>
<td>Feature</td>
<td>CentOS 6</td>
<td>CentOS 7</td>
<td>CentOS 8</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>----------</td>
<td>----------</td>
<td>----------</td>
</tr>
<tr>
<td>System custom states</td>
<td>Salt</td>
<td>Salt</td>
<td>Salt</td>
</tr>
<tr>
<td>Group custom states</td>
<td>Salt</td>
<td>Salt</td>
<td>Salt</td>
</tr>
<tr>
<td>Organization custom states</td>
<td>Salt</td>
<td>Salt</td>
<td>Salt</td>
</tr>
<tr>
<td>System set manager (SSM)</td>
<td>✔️</td>
<td>✔️</td>
<td>Salt</td>
</tr>
<tr>
<td>Service pack migration</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Basic Virtual Guest Management ✺</td>
<td>Traditional</td>
<td>✔️</td>
<td>Salt</td>
</tr>
<tr>
<td>Advanced Virtual Guest Management ✺</td>
<td>✗</td>
<td>Salt</td>
<td>Salt</td>
</tr>
<tr>
<td>Virtual Guest Installation (Kickstart), as Host OS</td>
<td>Traditional</td>
<td>Traditional</td>
<td>✗</td>
</tr>
<tr>
<td>Virtual Guest Installation (image template), as Host OS</td>
<td>Traditional</td>
<td>✔️</td>
<td>Salt</td>
</tr>
<tr>
<td>System deployment (PXE/Kickstart)</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>System redeployment (Kickstart)</td>
<td>Traditional</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Works with Uyuni Proxy</td>
<td>✔️</td>
<td>✔️</td>
<td>Salt</td>
</tr>
<tr>
<td>Action chains</td>
<td>✔️</td>
<td>✔️</td>
<td>Salt</td>
</tr>
<tr>
<td>Software crash reporting</td>
<td>✗</td>
<td>Traditional</td>
<td>✗</td>
</tr>
<tr>
<td>Staging (pre-download of packages)</td>
<td>✔️</td>
<td>✔️</td>
<td>Salt</td>
</tr>
<tr>
<td>Duplicate package reporting</td>
<td>✔️</td>
<td>✔️</td>
<td>Salt</td>
</tr>
<tr>
<td>CVE auditing (requires CVE ID)</td>
<td>✔️</td>
<td>✔️</td>
<td>Salt</td>
</tr>
<tr>
<td>Feature</td>
<td>CentOS 6</td>
<td>CentOS 7</td>
<td>CentOS 8</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>----------</td>
<td>----------</td>
<td>----------</td>
</tr>
<tr>
<td>SCAP auditing</td>
<td>✔️</td>
<td>✔️</td>
<td>Salt</td>
</tr>
<tr>
<td>Package verification</td>
<td>Traditional</td>
<td>Traditional</td>
<td>✗</td>
</tr>
<tr>
<td>Package locking</td>
<td>Traditional</td>
<td>Traditional</td>
<td>✗</td>
</tr>
<tr>
<td>System locking</td>
<td>Traditional</td>
<td>Traditional</td>
<td>✗</td>
</tr>
<tr>
<td>System snapshot</td>
<td>Traditional</td>
<td>Traditional</td>
<td>✗</td>
</tr>
<tr>
<td>Configuration file management</td>
<td>✔️</td>
<td>✔️</td>
<td>Salt</td>
</tr>
<tr>
<td>Snapshots and profiles</td>
<td>Traditional. Salt: Profiles supported, Sync not supported</td>
<td>Traditional. Salt: Profiles supported, Sync not supported</td>
<td>Salt: Profiles supported, Sync not supported</td>
</tr>
<tr>
<td>Power management</td>
<td>✔️</td>
<td>✔️</td>
<td>Salt</td>
</tr>
<tr>
<td>Monitoring</td>
<td>Salt</td>
<td>Salt</td>
<td>Salt</td>
</tr>
<tr>
<td>Docker buildhost</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Build Docker image with OS</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Kiwi buildhost</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Build Kiwi image with OS</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
</tbody>
</table>

_virtual Guest Management:

In this table, virtual guest management is split into basic and advanced.

Basic virtual guest management includes listing VMs, slow refresh, VM lifecycle actions (start, stop, resume, pause), and modifying VM vCPU and Memory.

Advanced virtual guest management includes fast refresh, VM lifecycle actions (delete, reset, power off), modifying VM disk, network, graphical display, and graphical display configuration.

**Supported Ubuntu Features**

This table lists the availability of various features on Ubuntu clients.

The icons in this table indicate:

- ✔️ the feature is available on both traditional and Salt clients
- ✗ the feature is not available
- ❔ the feature is under consideration, and may or may not be made available at a later date
- **Traditional** the feature is supported only on traditional clients
- **Salt** the feature is supported only on Salt clients.

### Table 6. Supported Features on Ubuntu Operating Systems

<table>
<thead>
<tr>
<th>Feature</th>
<th>Ubuntu 16.04</th>
<th>Ubuntu 18.04</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Operating system packages</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Registration</td>
<td>Salt</td>
<td>Salt</td>
</tr>
<tr>
<td>Install packages</td>
<td>Salt</td>
<td>Salt</td>
</tr>
<tr>
<td>Apply patches</td>
<td>❔</td>
<td>❔</td>
</tr>
<tr>
<td>Remote commands</td>
<td>Salt</td>
<td>Salt</td>
</tr>
<tr>
<td>System package states</td>
<td>Salt</td>
<td>Salt</td>
</tr>
<tr>
<td>System custom states</td>
<td>Salt</td>
<td>Salt</td>
</tr>
<tr>
<td>Group custom states</td>
<td>Salt</td>
<td>Salt</td>
</tr>
<tr>
<td>Organization custom states</td>
<td>Salt</td>
<td>Salt</td>
</tr>
<tr>
<td>System set manager (SSM)</td>
<td>Salt</td>
<td>Salt</td>
</tr>
<tr>
<td>Service pack migration</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Basic Virtual Guest Management ✻</td>
<td>Salt</td>
<td>Salt</td>
</tr>
<tr>
<td>Advanced Virtual Guest Management ✻</td>
<td>Salt</td>
<td>Salt</td>
</tr>
<tr>
<td>Virtual Guest Installation (Kickstart), as Host OS</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Virtual Guest Installation (image template), as Host OS</td>
<td>Salt</td>
<td>Salt</td>
</tr>
<tr>
<td>System deployment (PXE/Kickstart)</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>System redeployment (Kickstart)</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Contact methods</td>
<td>Salt: ZeroMQ, Salt-SSH</td>
<td>Salt: ZeroMQ, Salt-SSH</td>
</tr>
<tr>
<td>Feature</td>
<td>Ubuntu 16.04</td>
<td>Ubuntu 18.04</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>---------------------------</td>
<td>---------------------------</td>
</tr>
<tr>
<td>Works with Uyuni Proxy</td>
<td>Salt</td>
<td>Salt</td>
</tr>
<tr>
<td>Action chains</td>
<td>Salt</td>
<td>Salt</td>
</tr>
<tr>
<td>Software crash reporting</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Staging (pre-download of packages)</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>Duplicate package reporting</td>
<td>Salt</td>
<td>Salt</td>
</tr>
<tr>
<td>CVE auditing</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>SCAP auditing</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>Package verification</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Package locking</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>System locking</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>System snapshot</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Configuration file management</td>
<td>Salt</td>
<td>Salt</td>
</tr>
<tr>
<td>Package profiles</td>
<td>Salt: Profiles supported, Sync not supported</td>
<td>Salt: Profiles supported, Sync not supported</td>
</tr>
<tr>
<td>Power management</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Monitoring</td>
<td>x</td>
<td>Salt</td>
</tr>
<tr>
<td>Docker buildhost</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>Build Docker image with OS</td>
<td>Salt</td>
<td>Salt</td>
</tr>
<tr>
<td>Kiwi buildhost</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Build Kiwi image with OS</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>

* Virtual Guest Management:

In this table, virtual guest management is split into basic and advanced.

Basic virtual guest management includes listing VMs, slow refresh, VM lifecycle actions (start, stop, resume, pause), and modifying VM vCPU and Memory.

Advanced virtual guest management includes fast refresh, VM lifecycle actions (delete, reset, power off), modifying VM disk, network, graphical display, and graphical display configuration.
SLE Client Registration Overview

There are several ways to register clients to your Uyuni Server. The preferred methods are described in the following sections:

- For Salt clients, we recommend that you register clients using the Uyuni Web UI. For more information, see [Client-configuration › Registration-webui ›].
- If you want more control over the process, have to register many clients, or are registering traditional clients, we recommend that you create a bootstrap script. For more information, see [Client-configuration › Registration-bootstrap ›].
- For Salt clients and even more control over the process, executing single commands on the command line can be useful. For more information, see [Client-configuration › Registration-cli ›].

The client must have the date and time synchronized correctly with the Uyuni server before registration.

You must create an activation key first, to use bootstrap script or command line method. For more information about creating activation keys, see [Client-configuration › Clients-and-activation-keys ›].

Register Clients with the Web UI

Registering clients with the Uyuni Web UI works for Salt clients only.

**Procedure: Registering Clients with the Web UI**

1. In the Uyuni Web UI, navigate to Systems › Bootstrapping.
2. In the Host field, type the fully-qualified domain name (FQDN) of the client to be bootstrapped.
3. In the SSH Port field, type the SSH port number to use to connect and bootstrap the client. By default, the SSH port is 22.
4. In the User field, type the username to log in to the client. By default, the username is root.
5. In the Password field, type password to log in to the client.
6. In the Activation Key field, select the activation key that is associated with the software channel you want to use to bootstrap the client. For more information, see [Client-configuration › Clients-and-activation-keys ›].
7. OPTIONAL: In the Proxy field, select the proxy to register the client to.
8. By default, the Disable SSH Strict Key Host Checking checkbox is selected. This allows the bootstrap process to automatically accept SSH host keys without requiring you to manually authenticate.
9. OPTIONAL: Check the Manage System Completely via SSH checkbox. If you check this option, the client will be configured to use SSH for its connection to the server, and no other connection method will be configured.
10. Click [Bootstrap] to begin registration.
When the bootstrap process has completed, your client will be listed at **Systems › System List**.

When new packages or updates are installed on the client using Uyuni, any end user license agreements (EULAs) are automatically accepted. To review a package EULA, open the package details page in the Web UI.

### SLES 11 SSL errors

SLES 11 clients can sometimes have SSL errors which make some operations unusable, including package management and bootstrapping. In this case, you will see an error like this:

```plaintext
Repository 'SLES11-SP4-SUSE-Manager-Tools x86_64' is invalid.
[!] Valid metadata not found at specified URL(s)
Please check if the URIs defined for this repository are pointing to a valid repository.
Skipping repository 'SLES11-SP4-SUSE-Manager-Tools x86_64' because of the above error.
Download (curl) error for 'www.example.com':
  Error code: Unrecognized error
  Error message: error:1409442E:SSL routines:SSL3_READ_BYTES:tlsv1 alert protocol version
```

This occurs because Apache requires TLS v1.2, but older versions of SLES do not support this version of the TLS protocol. To fix this error, you need to force Apache to accept a greater range of protocol versions. Open the `/etc/apache2/ssl-global.conf` configuration file, locate the `SSLProtocol` line, and update it to read:

```
SSLProtocol all -SSLv2 -SSLv3
```

This will need to be done manually on the server, and with a Salt state on the Proxy. Restart the **apache** service on each system after making the changes.

### Register Clients with a Bootstrap Script

Registering clients with a bootstrap script gives you control over parameters, and can help if you have to register a large number of clients at once. This method works for both Salt and traditional clients.

To register clients using a bootstrap script, we recommend you create a template bootstrap script to begin, which can then be copied and modified. The bootstrap script you create is executed on the client when it is registered, and ensures all the necessary packages are deployed to the client. There are some parameters contained in the bootstrap script, which ensure the client system can be assigned to its base channel, including activation keys and GPG keys.

It is important that you check the repository information carefully, to ensure it matches the base channel repository. If the repository information does not match exactly, the bootstrap script will not be able to download the correct packages.
A bootstrap repository is needed for non-SLE clients in general and for SLE clients before version 15. A bootstrap offers packages for installing Salt on clients and for registering Salt or traditional clients. For information about creating a bootstrap repository, see [Client-configuration › Bootstrap-repository › ].

If you are bootstrapping Salt clients using the Web UI, you will need to ensure that the client system has Python installed before you begin. For Salt clients running SUSE Linux Enterprise Server 12 or older, you will also require the python-xml package.

openSUSE Leap 15 and SLES 15 and Python 3

openSUSE Leap 15 and SLE 15 use Python 3 by default. Bootstrap scripts based on Python 2 must be re-created for openSUSE Leap 15 and SLE 15 systems. Attempting to register openSUSE Leap 15 or SLE 15 systems using Python 2 bootstrap scripts will fail.

Create a Bootstrap Script

You can use the Uyuni Web UI to create an editable bootstrap script.

Procedure: Creating a Bootstrap Script

1. In the Uyuni Web UI, navigate to Admin › Manager Configuration › Bootstrap Script.

2. In the SUSE Manager Configuration - Bootstrap dialog, uncheck the Bootstrap using Salt checkbox if you are installing a traditional client. For Salt clients, leave it checked.

3. The required fields are pre-populated with values derived from previous installation steps. For details on each setting, see [Reference › Admin › ].

4. Click [Update] create the script.

5. The bootstrap script is generated and stored on the server in the /srv/www/htdocs/pub/bootstrap directory. Alternatively, you can access the bootstrap script over HTTPS. Replace example.com with the host name of your Uyuni Server:

   https://example.com/pub/bootstrap/bootstrap.sh

   Do not disable SSL in your bootstrap script. Ensure that Enable SSL is checked in the Web UI, or that the setting USING_SSL=1 exists in the bootstrap script. If you disable SSL, the registration process requires custom SSL certificates. For more about custom certificates, see [Administration › Ssl-certs › ].
Editing a Bootstrap Script

You can copy and modify the template bootstrap script you created to customize it. A minimal requirement when modifying a bootstrap script for use with Uyuni is the inclusion of an activation key. Most packages are signed with GPG, so you will also need to have trusted GPG keys on your system to install them.

In this procedure, you will need to know the exact name of your activation keys. Navigate to Home › Overview and, in the Tasks box, click Manage Activation Keys. All keys created for channels are listed on this page. You must enter the full name of the key you wish to use in the bootstrap script exactly as presented in the key field. For more information about activation keys, see [Client-configuration › Clients-and-activation-keys › ].

Procedure: Modifying a Bootstrap Script

1. On your Uyuni server, as root at the command line change to the bootstrap directory with:

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

2. Create and rename two copies of the template bootstrap script for use with each of your clients.

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

3. Open `bootstrap-sles12.sh` for modification. Scroll down until you can see the text shown below. If `exit 1` exists in the file, comment it out by typing a hash or pound sign (`#`) at the beginning of the line. This activates the script. Enter the name of the key for this script in the `ACTIVATION_KEYS=` field:

   ```bash
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-sles12
ORG_GPG_KEY=
   ```

4. When you have finished, save the file, and repeat this procedure for the second bootstrap script.

Connect Clients

When you have finished creating your script, you can use it to register clients.

Procedure: Running the Bootstrap Script

1. On the Uyuni Server, log in as root. At the command prompt, and change to the bootstrap directory:
2. Run this command to execute the bootstrap script on the client; replace **EXAMPLE.COM** with the host name of your client:

```
cat bootstrap-sles12.sh | ssh root@EXAMPLE.COM /bin/bash
```

The script will execute and proceed to download the required dependencies located in the repositories directory you created earlier.

3. When the script has finished running, you can check that your client is registered correctly by opening the Uyuni Web UI and navigating to **Systems › Overview** to ensure the new client is listed.

When new packages or updates are installed on the client using Uyuni, any end user license agreements (EULAs) are automatically accepted. To review a package EULA, open the package detail page in the Web UI.

### Package Locks

Package locks can only be used on traditional clients that use the Zypper package manager. The feature is not currently supported on Red Hat Enterprise Linux or Salt clients.

Package locks are used to prevent unauthorized installation or upgrades to software packages on traditional clients. When a package has been locked, it will show a padlock icon, indicating that it can not be installed. Any attempt to install a locked package will be reported as an error in the event log.

Locked packages can not be installed, upgraded, or removed, either through the Uyuni Web UI, or directly on the client machine using a package manager. Locked packages will also indirectly lock any dependent packages.

#### Procedure: Using Package Locks

1. On the client machine, install the `zypp-plugin-spacewalk` package as **root**:

```
zypper in zypp-plugin-spacewalk
```

2. Navigate to the **Software › Packages › Lock** tab on the managed system to see a list of all available packages.

3. Select the packages to lock, and click **[Request Lock]**. You can also choose to enter a date and time for the lock to activate. Leave the date and time blank if you want the lock to activate as soon as possible. Note that the lock might not activate immediately.

4. To remove a package lock, select the packages to unlock and click **[Request Unlock]**. Leave
the date and time blank if you want the lock to deactivate as soon as possible. Note that the lock might not deactivate immediately.

Register on the Command Line (Salt)

Instead of the Web UI, you can use the command line to register a Salt client. This procedure requires that you have installed the Salt package on the Salt client before registration. For SLE 12 based clients, you also must have activated the Advanced Systems Management module.

Registring on the command line is also possible with traditional clients, but it requires more steps. It is not covered here. Use the bootstrap script procedure to register traditional clients. For more information, see registration-bootstrap.pdf.

Procedure: Registering Clients Using the Command Line

1. Choose a client configuration file located at:

   /etc/salt/minion

   or:

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

   This is sometimes also called a minion file.

2. Add the Uyuni Server or Proxy FQDN as the master, and the activation key, to the client configuration file:

   master: SERVER.EXAMPLE.COM
   server_id_use_src: adler32
   enable_legacy_startup_events: False
   enable_fqdns_grains: False
   grains:
     susemanager:
       activation-key: "<Activation_Key_Name>"

3. Restart the salt-minion service:

   systemctl restart salt-minion

4. On the Uyuni Server, accept the new client key; replace <client> with the name of your client:

   salt-key -a '<client>'
SLES 11 SSL errors

SLES 11 clients can sometimes have SSL errors which make some operations unusable, including package management and bootstrapping. In this case, you will see an error like this:

```
Repository 'SLES11-SP4-SUSE-Manager-Tools x86_64' is invalid.
[] Valid metadata not found at specified URL(s)
Please check if the URIs defined for this repository are pointing to a valid repository.
Skipping repository 'SLES11-SP4-SUSE-Manager-Tools x86_64' because of the above error.
Download (curl) error for 'www.example.com':
Error code: Unrecognized error
Error message: error:1409442E:SSL routines:SSL3_READ_BYTES:tlsv1 alert protocol version
```

This occurs because Apache requires TLS v1.2, but older versions of SLES do not support this version of the TLS protocol. To fix this error, you need to force Apache to accept a greater range of protocol versions. Open the `/etc/apache2/ssl-global.conf` configuration file, locate the `SSLProtocol` line, and update it to read:

```
SSLProtocol all -SSLv2 -SSLv3
```

This will need to be done manually on the server, and with a Salt state on the Proxy. Restart the `apache` service on each system after making the changes.

Activation Keys

Activation keys are used with traditional and Salt clients to ensure that your clients have the correct software entitlements, are connecting to the appropriate channels, and are subscribed to the relevant groups. Each activation key is bound to an organization, which you can set when you create the key.

In Uyuni, an activation key is a group of configuration settings with a label. You can apply all configuration settings associated with an activation key by adding its label as a parameter to a bootstrap script. We recommend you use an activation key label in combination with a bootstrap script. When the bootstrap script is executed all configuration settings associated with the label are applied to the system the script is run on.

An activation key can specify:

- Channel Assignment
- System Types (Traditionally called Add-on Entitlements)
- Contact Method
- Configuration Files
- Packages to be Installed
- System Group Assignment
Procedure: Creating an Activation Key

1. In the Uyuni Web UI, as an administrator, navigate to Systems › Activation Keys.
2. Click the [Create Key] button.
3. On the Activation Key Details page, in the Description field, enter a name for the activation key.
4. In the Key field, enter the distribution and service pack associated with the key. For example, SLES12-SP4 for SUSE Linux Enterprise Server 12 SP4.

   ! Do not use commas in the Key field for any SUSE products. However, you must use commas for Red Hat Products. For more information, see [Reference › Systems › ].

5. In the Base Channels drop-down box, select the appropriate base software channel, and allow the relevant child channels to populate. For more information, see reference:admin/setup-wizard.pdf and [Administration › Custom-channels › ].
6. Select the child channels you need (for example, the mandatory SUSE Manager tools and updates channels).
7. We recommend you leave the Contact Method set to Default.
8. We recommend you leave the Universal Default setting unchecked.
9. Click [Create Activation Key] to create the activation key.
10. Check the **Configuration File Deployment** check box to enable configuration management for this key, and click [**Update Activation Key**] to save this change.

The **Configuration File Deployment** check box does not appear until after you have created the activation key. Ensure you go back and check the box if you need to enable configuration management.

**Combining Activation Keys**

You can combine activation keys when executing the bootstrap script on your clients. Combining keys allows for more control on what is installed on your systems and reduces duplication of keys for large or complex environments.

**Combining Activation Keys**
Activation Key Best Practices

Default Parent Channel

Avoid using the SUSE Manager Default parent channel. This setting forces Uyuni to choose a parent channel that best corresponds to the installed operating system, which can sometimes lead to unexpected behavior. Instead, we recommend you create activation keys specific to each distribution and architecture.

Bootstrapping with Activation Keys

If you are using bootstrap scripts, consider creating an activation key for each script. This will help you align channel assignments, package installation, system group memberships, and configuration channel assignments. You will also need less manual interaction with your system after registration.

Bandwidth Requirements

Using activation keys might result in automatic downloading of software at registration time, which might not be desirable in environments where bandwidth is constrained.

These options create bandwidth usage:

- Assigning a SUSE Product Pool channel will result in the automatic installation of the corresponding product descriptor package.
- Any package in the Packages section will be installed.
- Any Salt state from the Configuration section might trigger downloads depending on its contents.

Key Label Naming

If you do not enter a human-readable name for your activation keys, the system will automatically...
generate a number string, which can make it difficult to manage your keys.

Consider a naming scheme for your activation keys to help you keep track of them. Creating names which are associated with your organization’s infrastructure will make it easier for you when performing more complex operations.

When creating key labels, consider these tips:

- **OS naming (mandatory):** Keys should always refer to the OS they provide settings for
- **Architecture naming (recommended):** Unless your company is running on one architecture only, for example x86_64, then providing labels with an architecture type is a good idea.
- **Server type naming:** What is, or what will this server be used for?
- **Location naming:** Where is the server located? Room, building, or department?
- **Date naming:** Maintenance windows, quarter, etc.
- **Custom naming:** What naming scheme suits your organizations needs?

Example activation key label names:

```
sles12-sp2-web_server-room_129-x86_64
sles12-sp2-test_packages-blg_502-room_21-ppc64le
```

Do not use commas in the **Key** field for any SUSE products. However, you **must** use commas for Red Hat Products. For more information, see [Reference › Systems ›](#).

### Included Channels

When creating activation keys you also need to keep in mind which software channels will be associated with it.

Keys should have a specific base channel assigned to them, for example: **SLES12-SP2-Pool-x86_64**. If this is not the case, Uyuni cannot use specific stages. Using the default base channel is not recommended and may cause problems.

- **Channels to be included:**
  - suse-manager-tools
- **Typical packages to be included:**
  - mgr-osad (pushing tasks)
    - Installs **python-jabberpy** and **pyxml** as dependencies
- **mgr-cfg-actions** (Remote Command, Configuration Management)
  - Installs *mgr-cfg* and *mgr-cfg-client* as dependencies

The *suse-manager-tools* channel is mandatory.

Typical packages to be included:

- osad (pushing tasks): Installs *python-jabberpy* and *pyxml* as dependencies
- **rhncfg-actions** (Remote Command, Configuration Management): Installs *rhncfg* and *rhncfg-client* as dependencies
SLE Client Registration on a Proxy

Proxy servers can act as a broker and package cache for both Salt and traditional clients. Registering clients on a Uyuni Proxy is similar to registering them directly on Uyuni, with a few key differences.

These sections contain information on registering Salt clients on a proxy using the Web UI, commands on the command line, or a bootstrap script. There is also information on registering traditional clients using a bootstrap script.

Within the Web UI, proxy pages will show information about both Salt and traditional clients. You can see a list of clients that are connected to a proxy by clicking the name of the proxy in Systems › System List › Proxy, then select the Proxy subtab of the Details tab.

A list of chained proxies for a Salt client can be seen by clicking the name of the client in Systems › All, then select the Connection subtab of the Details tab.

If you decide to move any of your clients between proxies or the server you will need to repeat the registration process from the beginning.

Register in the Web UI (Salt Only)

Using the Web UI is similar to registering clients directly with the Uyuni Server.

A bootstrap repository is needed for non-SLE clients in general and for SLE clients before version 15. A bootstrap offers packages for installing Salt on clients and for registering Salt or traditional clients. For information about creating a bootstrap repository, see [Client-configuration › Bootstrap-repository › ].

Procedure: Registering Clients to a Proxy in the Web UI

1. In the Uyuni Web UI, navigate to Systems › Bootstrapping.
2. In the Host field, type the fully-qualified domain name (FQDN) of the client to be bootstrapped.
3. In the SSH Port field, type the SSH port number that will be used to connect and bootstrap the client. By default, the SSH port is 22.
4. In the User field, type the username to log in to the client. By default, the username is root.
5. In the Password field, type password to log in to the client.
6. In the Activation Key field, select the activation key that is associated with the software channel you want to use to bootstrap the client.
7. In the Proxy field, select the proxy server you want to register to.
8. By default, the Disable SSH Strict Key Host Checking checkbox is selected. This allows the bootstrap process to automatically accept SSH host keys without requiring you to manually authenticate.
9. **OPTIONAL:** Check the *Manage System Completely via SSH* checkbox. If you check this option, the client will be configured to use SSH for its connection to the server, and no other connection method will be configured.

10. Click **[Bootstrap]** to begin registration.

When the bootstrap process has completed, your client will be listed at **Systems › System List**.

**Register on the Command Line (Salt)**

Instead of the Web UI, you can use the command line to register a Salt client to a proxy. This procedure requires that you have installed the Salt package on the Salt client before registration. For SLE 12 based clients, you also must have activated the *Advanced Systems Management* module.

Registering traditional clients on the command line is also possible, but it requires more steps. It is not covered here. Use the bootstrap script procedure to register traditional clients. For more information, see *script-client-proxy.pdf*.

*Procedure: Registering Clients to a Proxy Using the Command Line*

1. Choose a client configuration file located at:

   `/etc/salt/minion`

   or:

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

   This is sometimes also called a minion file.

2. Add the proxy FQDN as the **master** to the client configuration file:

   `master: PROXY123.EXAMPLE.COM`

3. Restart the **salt-minion** service:

   `systemctl restart salt-minion`

4. On the server, accept the new client key; replace `<client>` with the name of your client:

   `salt-key -a '<client>'`
Registering with a Bootstrap Script (Salt and Traditional)

Registering clients (either traditional or Salt) via SUSE Manager Proxy with a bootstrap script is done almost the same way as registering clients directly with the Uyuni Server. The difference is that you create the bootstrap script on the SUSE Manager Proxy with a command line tool. The bootstrap script then deploys all necessary information to the clients. The bootstrap script requires some parameters (such as activation keys or GPG keys) that depend on your specific setup.

Procedure: Registering Clients to a Proxy with a Bootstrap Script

1. Create a client activation key on the Uyuni server using the Web UI. For more information, see [Client-configuration › Clients-and-activation-keys ›](#).

2. On the proxy, execute the `mgr-bootstrap` command line tool as root. If needed, use the additional command line switches to tune your bootstrap script. To install a traditional client instead of a Salt client, ensure you use the `--traditional` switch.

   To view available options type `mgr-bootstrap --help` from the command line:

   ```
   # mgr-bootstrap --activation-keys=key-string
   ```

3. Optional: edit the resulting bootstrap script.

4. Execute the bootstrap script on the clients.
Automate Client Installation

AutoYaST and Kickstart configuration files allow you to automate client system installations. This is useful if you need to install a large number of clients.

For SUSE Linux Enterprise clients, use AutoYaST. When you have created an AutoYaST file, you can upload and manage it using the Uyuni Web UI.

For Red Hat Enterprise Linux clients, use Kickstart. Kickstart files are created, modified, and managed within the Uyuni Web UI.

We recommend that you use PXE boot for installing clients. PXE booting requires a DHCP server that points to your Uyuni Server. The Uyuni Server then acts as a TFTP server.

The TFTP environment is generated with Cobbler. Cobbler can also generate a bootable ISO image. The ISO image can be used to install machines when PXE boot is not an option; for more information, see [Client-configuration › Cobbler › ].

Preparation

A configured distribution and an autoinstallation profile is required.

Procedure: Preparing a Distribution

1. Provide the files required to start an installation. Unpack an installation medium such as a DVD image on your Server. It contains the Linux kernel, an initrd, and other files required to boot the OS in installation mode.

2. In the Uyuni Web UI, navigate to System › Autoinstallation › Distributions.

3. In the Autoinstallable Distributions dialog, click Create Distribution:
   - In the Distribution Label field, enter a name to identify your autoinstallable distribution.
   - In the Tree Path field, enter the path to an installation tree located on your Uyuni Server.
   - Select the matching Base Channel mirrored on the Uyuni Server. This base channel must represent the distribution you want to install. It can be the Vendor, Custom, or Cloned Channels.
   - The Installer Generation should also match.
   - Optionally, you can specify kernel options which should be added when booting this distribution. Note that there are multiple places where you can provide kernel options. Only add options here that are generic for the distribution.

4. Click [Create Autoinstallable Distribution].

For more information, see [Reference › Systems › ].

Procedure: Preparing a Profile
1. In the Uyuni Web UI, navigate to System › Autoinstallation › Profiles.

2. In the Autoinstallation Profiles dialog, add the profile for your autoinstallation. It can be an AutoYaST or Kickstart profile.

3. There are two ways to create profiles:
   - Create a Kickstart profile using a wizard
   - Upload an externally created profile (Kickstart or AutoYaST)

For more information about the Kickstart wizard, see [Reference › Systems ›].

Upload a Profile

Profiles require a label, and an Autoinstallation Tree (distribution).

Upload the Kickstart or AutoYaST profile. You can write your own kickstart or AutoYaST profile directly in the Web UI, or create the profile and upload it from your local filesystem.

AutoYaST is able to dump a profile from an existing installation. This can be used as template, but will need to be edited to make it usable by Uyuni.

Procedure: Making an AutoYaST Template Usable by Uyuni

1. Change the add-on section and add Uyuni URLs. The URL must use this format:

   http://$redhat_management_server/ks/dist/child/<channel-label>/<distribution-label>

2. Replace <channel-label> and <distribution-label> with the correct labels. You can also use a variable for distribution-label. Ensure that the distribution label corresponds to the autoinstallable distribution you selected. You can only configure child channels in this file. The channels must be children of the base channel you selected in the distribution you use for this profile. You do not need to specify a base channel. The base channel is defined in the distribution.

3. Register the system after it is installed. For this step we provide script snippets that can be used.

   For Salt managed clients, use the spacewalk/minion_script snippet:

   ```
   <scripts>
   <init-scripts config:type="list">
   $SNIPPET('spacewalk/minion_script')
   </init-scripts>
   </scripts>
   ```

   For traditional clients, use the spacewalk/sles_register_script snippet:
For registering Salt clients, you must accept the Salt key on the Uyuni Server before you attempt autoinstallation.

If you change the distribution label, it will not automatically change the `install=` kernel option. You will need to manually change the kernel options to match the distribution label.

For more information about autoinstallation profiles, see [Reference › Systems ›].

Variables and snippets

Profiles are not finalized until they are requested by a client. This allows you to use variables in profiles. You can define profile variables in the Web UI by navigating to Profiles › Variables.

Some common variables are:

**redhat_management_server**
The server Red Hat clients register to (automatically set).

**org**
The organization ID where this profile is created (automatically set).

**registration_key**
The key used in the registration snippets. By specifying this variable, you can set the activation key to be used to register the system.

**dont_register**
If specified, the registration will be skipped.

**allow_config_actions**
If set to 1, it will allow traditional configuration management (traditional only).

**allow_remote_commands**
If set to 1, it will allow traditional remote command execution (traditional only).

**dont_disable_automatic_onlineupdate**
If set, the automatic online update will stay enabled (SUSE OSes only).
Autoinstallation with AutoYaST

When you have prepared an autoinstallation profile, you can autoinstall your clients.

To start autoinstallation the client must already be known to Uyuni. You can use bare metal provisioning to bring clients into Uyuni. Alternatively, you can use the Uyuni API.

Bare Metal Provisioning

Bare metal provisioning is supported on clients with AMD or Intel x86_64 processors, and at least 1 GB of RAM.

Uyuni Server uses Cobbler over TFTP to connect to bare metal clients for provisioning. Check that you have a DHCP server and that you have set the next-server configuration parameter to match the Uyuni server IP address or hostname.

When you have the bare metal provisioning option enabled, any bare metal client connected to the Uyuni network is automatically added to the organization as soon as it is powered on. The provisioning process can take a few minutes. When it is complete, the client will be shut down, and it appears in the Systems list, ready to be installed.

Procedure: Provisioning Bare Metal Clients

1. In the Uyuni Web UI, navigate to Admin › Manager Configuration › Bare-metal systems.
2. Click [Enable adding to this organization].
3. Navigate to Systems, locate your bare metal clients in the list, and click the client you want to provision.
4. Select the Provisioning › Autoinstallation tab.
5. Select the AutoYaST profile to use, and start the autoinstallation.

You cannot schedule autoinstallation for bare metal clients. Bare metal clients will be automatically installed when they are correctly configured and powered on.

New bare metal clients are added to the organization that belongs to the administrator who enabled the bare metal feature. To change the organization clients are added to, disable the bare metal feature, log in as the administrator of the new organization, and then re-enable the feature.

You can use the System Set Manager (SSM) with bare metal clients. However, not all SSM features are
available for bare metal clients, because they do not yet have an operating system installed. This also applies to mixed sets that include bare metal systems. All features will become available to the set when all the clients in the set have been provisioned. For more information on SSM, see [Client-configuration › Using-ssm ›].

API Provisioning

You can use API calls at the command prompt to bring clients into Uyuni for autoinstallation.

Procedure: Provisioning Using the API

1. At the command prompt, use the `system.createSystemRecord` or `system.createSystemProfile` API calls. In this example, replace `<hw_addr>` with a hardware address such as `00:25:22:71:e7:c6` and `<name>` with the name of your client:

    ```
    spacecmd api -- --args '
    "systemname", "hwAddress": "<hw_addr>", "hostname": "<name>"
    
    system.createSystemProfile
    ```

2. In the Uyuni Web UI, navigate to Systems, locate your new clients in the list, and click the client you want to provision.

3. Select the Provisioning › Autoinstallation tab.

4. Select the AutoYaST profile to use, and start the autoinstallation. Alternatively, you can schedule the autoinstallation for a later time.

Advanced PXE Installation Configuration

If the client needs to be installed for the first time, you can use the Create PXE installation configuration option. This option creates a PXE boot configuration. When you power on the client, it boots from the network and the correct profile is selected for installation.

If the client is already managed, click [Schedule Autoinstallation and Finish] to start the installation.

For more information about AutoYaST, see https://doc.opensuse.org/projects/autoyast/.

Kickstart

When you install a Red Hat Enterprise Linux client, there are a number of questions you need to answer. To automate installation, you can create a Kickstart file with all the answers to those questions, so that no user intervention is required.

Kickstart files can be kept on a server and read by individual clients during installation. The same Kickstart file is used to install multiple clients.

Kickstart can be used to schedule a registered system to be installed with a new operating system and package profile, or you can use it to install a new system that was not previously registered, or does not
yet have an operating system installed.
For more information about Kickstart, see the Red Hat documentation.

Before you Begin

Some preparation is required for your infrastructure to handle Kickstart installations. Before you create a Kickstart profile, consider:

- A DHCP server is not required for kickstarting, but it can make things easier. If you are using static IP addresses, select static IP while developing your Kickstart profile.
- An FTP server can be used instead of hosting the Kickstart distribution tree using HTTP.
- If you are performing a bare metal Kickstart installation, use these settings:
  - Configure DHCP to assign the required networking parameters and the bootloader program location.
  - In the bootloader configuration file, specify the kernel and appropriate kernel options to be used.

Build a Bootable ISO

You will need to create a bootable ISO image to be used by the target system for installation. When the system is rebooted or switched on, it boots from the image, loads the Kickstart configuration from your Uyuni, and installs Red Hat Enterprise Linux according to the Kickstart profile.

Building a Bootable ISO

1. Copy the contents of `/isolinux` from the first CD-ROM of the target distribution.
2. Edit the `isolinux.cfg` file to default to ‘ks’. Change the ‘ks’ section to read:

```plaintext
label ks
kernel vmlinuz
append text ks='url\'initrd=initrd.img lang= devfs=nomount \ramdisk_size=16438\ksdevice'
```

IP address-based Kickstart URLs will look like this:

```plaintext
http://`my.manager.server`/kickstart/ks/mode/ip_range
```

The Kickstart distribution defined via the IP range should match the distribution from which you are building, or errors will occur.

3. OPTIONAL: If you want to use the `ksdevice`, it looks like:

```plaintext
ksdevice=eth0
```
It is possible to change the distribution for a Kickstart profile within a family, such as Red Hat Enterprise Linux AS 4 to Red Hat Enterprise Linux ES 4, by specifying the new distribution label. Note that you cannot move between versions (4 to 5) or between updates (U1 to U2).

4. Customize `isolinux.cfg` further as required. For example, you can add multiple options, different boot messages, or shorter timeout periods.

5. Create the ISO with this command:

   ```
   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/
   ```

   Note that `isolinux/` is the relative path to the directory containing the modified isolinux files copied from the distribution CD, while `file.iso` is the output ISO file, which is placed into the current directory.

6. Burn the ISO to CD-ROM and insert the disk.

7. Boot the system and type `ks` at the prompt (if you left the label for the Kickstart boot as 'ks').

8. Press `Enter` to start Kickstart.

**Integrating with PXE**

Instead of using a bootable ISO image, you can use a PXE image instead. This is less error-prone, allows Kickstart installation from bare metal, and integrates with existing PXE/DHCP environments.

To use this method, make sure your systems have network interface cards (NICs) that support PXE. You will need to install and configure a PXE server, ensure DHCP is running, and place the installation repository on an HTTP server that is reachable by the Uyuni Server.

Upload the Kickstart profile to the Uyuni Server using the Uyuni Web UI.

When the AutoYaST profile has been created, use the URL from the Autoinstallation Overview page as the image location.


For more information about autoinstallation profiles, see [Reference › Systems › ].

**Cobbler**

Cobbler is an installation server that allows you to perform unattended system installations. Cobbler is installed on the Uyuni Server.

SUSE only supports Cobbler functions that are available in the Uyuni Web UI, or through the Uyuni API. Only supported features are documented here.
If you intend to use your installation with SUSE Manager for Retail formulas, do not follow this guide to configure Cobbler on the branch server. In SUSE Manager for Retail installations, the TFTPD formula manages these settings. For more information about the TFTPD formula, see [Salt › Formula-tftpd ›].

This section explains the Cobbler features most commonly used with Uyuni:

- The `cobbler sync` command is triggered from Uyuni Server and generate the TFTP boot environment
- Installation environment analysis using the `cobbler check` command
- Virtual machine guest installation automation with the `koan` client-side tool
- Building installation ISOs with PXE-like menus using the `cobbler buildiso` command (for Uyuni systems with x86_64 architecture)

For more information about Cobbler, see https://cobbler.readthedocs.io.

**Cobbler Requirements**

To use Cobbler for system installation with PXE, you will require a TFTP server. Uyuni installs a TFTP server by default. To PXE boot systems, you will require a DHCP server, or have access to a network DHCP server.

Cobbler uses host names as a unique key for each system. If you are using the `pxe-default-image` to onboard bare metal systems, make sure every system has a unique host name. Non-unique host names will cause all systems with the same host name to have the configuration files overwritten when a provisioning profile is assigned.

**Configure Cobbler**

Cobbler configuration is primarily managed using the `/etc/cobbler/settings` file. Cobbler will run with the default settings unchanged. All configurable settings are explained in detail in the `/etc/cobbler/settings` file.

The PXE boot process uses DHCP to find the TFTP boot server. The Uyuni Server can act as such a TFTP boot server and Cobbler can generate the content for it. You must have administrative access to the network’s DHCP server. Edit the DHCP configuration file so that it points to the Uyuni Server as the TFTP boot server:

*Procedure: Example for Configuring the ISC DHCP Server*

1. On the DHCP server, as root, open the `/etc/dhcpd.conf` file.
2. Append a new class with options for performing PXE boot installation. For example:
allow booting;
allow bootp;
class "PXE"
  {match if substring(option vendor-class-identifier, 0, 9) = "PXEClient";
   next-server 192.168.2.1;
   filename "pxelinux.0";}

This example:

- Enables the bootp protocol for network booting.
- Creates a class called PXE.
- Identifies systems as PXEClient if they are configured with PXE as the first boot priority.
- Directs PXE Clients to the Cobbler server at 192.168.2.1.
- Retrieves the pxelinux.0 bootloader file.

3. Save the file.

Procedure: Configuring PXE Boot in KVM

While it is possible to use KVM with PXE booting, it can be unreliable. We do not recommend you use this on production systems.

1. Use the virsh command to produce a dump of the current network XML description:

```bash
virsh net-dumpxml --inactive network > network.xml
```

2. Open the XML dump file at network.xml and add a bootp parameter within the <dhcp> element:

```xml
<bootp file='!/pxelinux.0' server='192.168.100.153'/>
```

3. Use the virsh command to install the updated description:

```bash
virsh net-define network.xml
```

Alternatively, you can use the net-edit subcommand, which will also perform some error checking.
Listing 1. Example: Minimal Network XML Description for KVM

```xml
<network>
  <name>default</name>
  <uuid>1da84185-31b5-4c8b-9ee2-a7f5ba39a7ee</uuid>
  <forward mode='nat'>
    <nat>
      <port start='1024' end='65535'/>
    </nat>
  </forward>
  <bridge name='virbr0' stp='on' delay='0'/>
  <mac address='52:54:00:29:59:18'/>
  <domain name='default'/>
  <ip address='192.168.100.1' netmask='255.255.255.0'>
    <dhcp>
      <range start='192.168.100.128' end='192.168.100.254'/>
      <bootp file='/pxelinux.0' server='192.168.100.153'/>
    </dhcp>
  </ip>
</network>
```

TFTP

Uyuni uses the **tftp** daemon. The **tftp** daemon is the recommended method for PXE services, and is installed by default. The default configuration works in most cases. However, if you need to change the configuration, use the YaST Services Manager.

The TFTP service must be running so it can serve the **pxelinux.0** boot image. Start YaST and use **System › Services Manager** to configure the **tftp** daemon.

You can also synchronize Cobbler-generated TFTP contents to a Uyuni Proxy. For synchronization, HTTPS port 443 must be open.

**Procedure: Installing TFTP**

1. On the Uyuni Server, as root, install the **susemanager-tftpsync** package:

   ```
   zypper install susemanager-tftpsync
   ```

2. On the Uyuni Proxy, as root user, install the **susemanager-tftpsync-recv** package:

   ```
   zypper install susemanager-tftpsync-recv
   ```

**Procedure: Configuring TFTP on a Proxy**

1. On the Uyuni Proxy, as root, run the **configure-tftpsync.sh** script.

2. The script will interactively ask you for details on the host names and IP addresses of the Uyuni Server and Proxy, as well for the location of the **tftpboot** directory on the Proxy.

For more information, use the **configure-tftpsync.sh --help** command.
Procedure: Configuring TFTP on a Server

1. On the Uyuni Server, as root, run the `configure-tftpsync.sh` script.

```
configure-tftpsync.sh proxy1.example.com proxy2.example.com
```

2. Run the `cobbler sync` command to push the files to the proxy. This will fail if you have not configured the proxies correctly.

3. If you want to change the list of proxies later on, you can use the `configure-tftpsync.sh` script to edit them.

   If you reinstall an already configured proxy and want to push all the files again, you must remove the cache file at `/var/lib/cobbler/pxe_cache.json` before you call `cobbler sync`.

Background Information about the Synchronization Process

A `cobbler sync` is a rebuild of every file Cobbler touched. On Uyuni, `cobbler sync` does the following actions:

1. Run pre-sync triggers. This can be any number of shell scripts.
2. Delete all files and directories that are not allowed in `/srv/www/cobbler/`.
3. Create all needed directories.
4. Delete all elements inside the directories.
5. Create the TFTP directory.
6. Write the DHCP files if management is enabled (unsupported). For more information, see Configure Cobbler.
7. Do the same with DNS (unsupported).
8. Clean up the cache.
9. Run `rsync` if rsync management is enabled.
10. Run post-sync triggers. This can be any number of shell scripts (unsupported).

Uyuni also adds or removes, or edits systems that are in Cobbler. Those actions trigger a so-called lite sync process. This sync only touches files and directories that are related to the change which triggered it.

Synchronize and Start the Cobbler Service

When tftpsync is configured, the Uyuni Server must be able to access the Uyuni Proxy systems directly.
Do not start or stop the **cobblerd** service independent of the Uyuni service. Doing so can cause errors. Always use `/usr/sbin/spacewalk-service` to start or stop Uyuni.

Check that all the prerequisites the Cobbler service requires, are configured according to your requirements. You can do this by running the **cobbler check** command.

When you had to change the configuration, restart the Uyuni service:

```
/usr/sbin/spacewalk-service restart
```

**Autoinstallation Templates**

AutoYaST or Kickstart profiles are used to automate SUSE Linux Enterprise or Red Hat Enterprise Linux client installations. Templates are used to describe how to create autoinstallation profiles. You can create autoinstallation variables within the Uyuni Web UI. This allows you to create and manage large numbers of profiles and systems, without having to manually create profiles for each.

Cobbler uses a template engine called Cheetah that provides support for templates, variables, and snippets.

For more information on creating profiles, see [Reference › Systems ›](

**Kickstart Template Syntax**

Kickstart templates can have static values for certain common items such as PXE image file names, subnet addresses, and common paths such as `/etc/sysconfig/network-scripts/`. However, templates differ from standard Kickstart files in their use of variables.

For example, a standard Kickstart file might have a networking section like this:

```
network --device=eth0 --bootproto=static --ip=192.168.100.24 \
   --netmask=255.255.255.0 --gateway=192.168.100.1 --nameserver=192.168.100.2
```

In a Kickstart template file, the networking section would look like this instead:

```
network --device=$net_dev --bootproto=static --ip=$ip_addr \
   --netmask=255.255.255.0 --gateway=$my_gateway --nameserver=$my_nameserver
```

These variables are substituted with the values set in your Kickstart profile variables or in your system detail variables. If the same variable is defined in both the profile and the system detail, then the system detail variable takes precedence.

Kickstart templates use syntax rules that rely on punctuation symbols. To avoid clashes, they need to be properly treated.
If the template contains shell script variables like `$(example)`, the content needs to be escaped with a backslash: `\$(example)`. If the variable is defined in the template, the templating engine will evaluate it correctly. If there is no such variable, the content will be left unchanged. Escaping the `$` symbol will prevent the templating engine from evaluating the symbol as an internal variable.

Long scripts or strings can be escaped by wrapping them with the `#raw` and `#end raw` directives. For example:

```bash
#raw
#!/bin/bash
for i in {0..2}; do
  echo "$i - Hello World!"
done
#end raw
```

Any line with a `#` symbol followed by a whitespace is treated as a comment and is therefore not evaluated. For example:

```bash
#start some section (this is a comment)
echo "Hello, world"
#end some section (this is a comment)
```

Kickstart Snippets

Kickstart snippets are sections of Kickstart code that can be called by a `$SNIPPET()` function. The snippet is parsed by Cobbler and substituted with the contents of the snippet. This example sets up a snippet for a common hard drive partition configuration:

```bash
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
```

Save this snippet of the configuration to a file in `/var/lib/cobbler/snippets/`, where Cobbler can access it.

Use the snippet by calling the `$SNIPPET()` function in your Kickstart templates. For example:

```bash
$SNIPPET('my_partition')
```

Cobbler will parse the function with the snippet of code contained in the `my_partition` file.
**Build ISOs with Cobbler**

Cobbler can create ISO boot images that contain a set of distributions, kernels, and a menu, that work similar to a PXE installation.

Building ISOs with Cobbler is not supported on IBM Z.

The Cobbler `buildiso` command takes parameters to define the name and output location of the boot ISO. For example:

```
cobbler buildiso --iso=/path/to/boot.iso
```

The boot ISO includes all profiles and systems by default. You can limit which profiles and systems are used, with the `--profiles` and `--systems` options. For example:

```
cobbler buildiso --systems="system1 system2 system3" \
   --profiles="profile1 profile2 profile3"
```

If you cannot write an ISO image to a public `tmp` directory, check your systemd settings in `/usr/lib/systemd/system/cobblerd.service`.

**Bare Metal Provisioning**

Systems that have not yet been provisioned are called bare metal systems. You can provision bare metal systems using Cobbler. Once a bare metal system has been provisioned in this way, it will appear in the `Systems` list, where you can perform regular provisioning with autoinstallation, for a completely unattended installation.

To successfully provision a bare metal system, you will require a fully patched Uyuni server.

The system to be provisioned must have x86_64 architecture, with at least 2 GB RAM, and be capable of PXE booting.

The server uses TFTP to provision the bare metal client, so the appropriate port and networks must be configured correctly in order for provisioning to be successful. In particular, ensure that you have a DHCP server, and have set the `next-server` parameter to the Uyuni server IP address or hostname.

**Enable Bare Metal Systems Management**

Bare metal systems management can be enabled or disabled in the Uyuni Web UI by navigating to Admin > SUSE Manager Configuration > Bare-metal systems.
New systems are added to the organization of the administrator who enabled the bare metal systems management feature. To change the organization, log in as an Administrator of the required organization, and re-enable the feature.

When the feature has been enabled, any bare metal system connected to the server network will be automatically added to the organization when it is powered on. The process can take a few minutes, and the system will automatically shut down when it is complete. The system will now be visible in the Systems › System list. Click on the name of the system to see basic information. For more details, go to the Properties, Notes, and Hardware tabs. You can migrate bare metal systems to other organizations if required, using the Migrate tab.

Provision Bare Metal Systems

Provisioning bare metal systems is similar to provisioning other systems, and can be done using the Provisioning tab. However, you will not be able to schedule provisioning, it will happen automatically as soon as the system is configured and powered on.

System Set Manager can be used with bare metal systems. However, not all SSM features are available, because bare metal systems do not have an operating system installed. This also applies to mixed sets that contain bare metal systems. All features will be re-enabled if the bare metal systems are removed from the set.
Other Clients

It is possible to register clients using operating systems from Red Hat, CentOS, or Ubuntu.

This section contains information specific to clients running operating systems other than those provided by SUSE.

Registering SUSE Linux Enterprise Server with Expanded Support Clients

This section contains information about registering traditional and Salt clients running SUSE Linux Enterprise Server with Expanded Support (Expanded Support) operating systems.

Expanded Support clients are based on Red Hat Enterprise Linux or CentOS.

They are sometimes also called SLESES, RES or Red Hat Expanded Support.

You are responsible for arranging access to Red Hat or CentOS base media repositories and installation media.

You must obtain support from SUSE for all your Expanded Support systems.

Traditional clients are not available on Expanded Support 8. Expanded Support 8 clients are only supported as Salt clients.

Server Requirements

Before you begin, check that your Uyuni Server meets the requirements at [Installation › Hardware-requirements ›].

Taskomatic uses one CPU core, and requires at least 3072 MB of RAM. To ensure that taskomatic has access to enough memory, open the /etc/rhn/rhn.conf configuration file, and add this line:

```
taskomatic.java.maxmemory=3072
```

Restart Taskomatic:

```
systemctl restart taskomatic
```

Add Client Tools

For Expanded Support clients, some required packages are contained on the Red Hat Enterprise Linux or CentOS installation media. You must have these packages installed before you can register a Expanded
Support client.

The Expanded Support product is provided by SUSE Customer Center. This also includes the client tools package.

Before you register Expanded Support clients to your Uyuni Server, check that you have the corresponding Expanded Support product enabled, and the required channels are fully synchronized.

You need to select two different sets of channels, one for Expanded Support and the other for the Client Tools.

You will need an activation key associated with the correct Expanded Support channels. For more information about activation keys, see [Client-configuration › Clients-and-activation-keys ›].

For Expanded Support 8 clients, add both the BaseOS and Appstream channels. You will require packages from both channels. If you do not add both channels, you will not be able to create the bootstrap repository, due to missing packages.

Procedure: Adding Client Tools Channels

1. On the Uyuni Server, add the appropriate Expanded Support channels:
   - For Expanded Support 6:
     From the Web UI, add RHEL Expanded Support 6 x86_64.
     From the command prompt, add rhel-x86_64-server-6 and res6-suse-manager-tools-x86_64.
   - For Expanded Support 7:
     From the Web UI, add RHEL Expanded Support 7 x86_64 and SUSE Linux Enterprise Client Tools RES7 x86_64.
     From the command prompt, add rhel-x86_64-server-7 and res7-suse-manager-tools-x86_64.
   - For Expanded Support 8:
     From the Web UI, add RHEL8 Base x86_64 and SUSE Manager Tools for RHEL and ES 8 x86_64.
     From the command prompt, add rhel-x86_64-server-8 and res8-suse-manager-tools-x86_64.

2. Synchronize the Uyuni Server with the SUSE Customer Center. You can do this using the Web UI, or by running `mgr-sync` at the command prompt.
3. Add the new channel to your activation key.

There are two ways to check if a channel has finished synchronizing:

- In the Uyuni Web UI, navigate to Admin › Setup Wizard and select the SUSE Products tab. This dialog displays a completion bar for each product when they are being synchronized.

- Check the synchronization log file at the command prompt with `tail -f /var/log/rhn/reposync/channel-label.log` file.

Each child channel will generate its own log during the synchronization progress. You will need to check all the base and child channel log files to be sure that the synchronization is complete.

The Expanded Support channels can be very large. The initial channel synchronization can sometimes take up to 24 hours.

When the initial synchronization is complete, we recommended you clone the channel before you work with it. This gives you a backup of the original synchronization data.

Add Base Media

The base Expanded Support channel does not contain any packages, because SUSE does not provide Red Hat Enterprise Linux or CentOS base media. You will need to obtain base media from Red Hat or CentOS, which you can add as a child channel to the Expanded Support parent channel. To ensure you have all the packages you need, use a full DVD image, not a minimal or JeOS image.

You can use Uyuni custom channels to set up the Red Hat Enterprise Linux or CentOS media. All packages on the base media are mirrored into a child channel.

**Procedure: Creating a Red Hat Enterprise Linux or CentOS Custom Channel**

- SLES ES 6 and 7, require only one child channel for the os repository
- SLES ES 8 requires two child channels for both the BaseOS and AppStream repositories If you do not add both channels, you will not be able to create the bootstrap repository, due to missing packages.

1. In the Uyuni Web UI, navigate to Software › Manage › Channels.
2. Click [Create Channel] and set these parameters:
   - In the Channel Name field, type a name for your channel, specifying the OS name and architecture.
   - In the Channel Label field, type a label for your channel, specifying the OS name and architecture.
In the **Parent Channel** field, select the corresponding Red Hat Enterprise Linux or CentOS distribution channel for your architecture. The parent channel will not contain any packages.

In the **Architecture** field, select the appropriate architecture.

In the **Repository Checksum Type** field, select **sha1**.

In the **Channel Summary** field, type a summary for your channel, specifying the OS name and architecture.

In the **Organization Sharing** field, select **Public**.

3. Click [Create Channel].

4. Add the new channel to your activation key.

**Procedure: Adding Base Media to Custom Channels**

1. On the Uyuni Server, at the command prompt, as root, copy the base media image to the `/tmp/` directory.

2. Create a directory to contain the media content. Replace `<os_name>` with either `sleses6`, `sleses7` or `sleses8`:

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

3. Mount the image:

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

When the image is mounted, you can synchronize the base media. To manually synchronize the channels, navigate to **Software › Manage › Channels**. Click each channel in the list, and navigate to menu: [Repositories > Sync]. Click [Sync Now] to begin synchronization immediately. You can also create a synchronization schedule from this screen.

**Monitor Synchronization Progress**

You can check if a channel has finished synchronizing:

- In the Uyuni Web UI, navigate to **Admin › Setup Wizard** and select the **SUSE Products** tab. This dialog displays a completion bar for each product when they are being synchronized.

- Check the synchronization log file at the command prompt with `tail -f /var/log/rhn/reposync/channel-label.log`. Each child channel will generate its own log during the synchronization progress.
You will need to check all the base and child channel log files to be sure that the synchronization is complete.

Register Expanded Support Clients

You Expanded Support clients are now ready to be registered.

For more information on registering your clients, see [Client-configuration › Registration-overview ›].

Registering Red Hat Enterprise Linux Clients

This section contains information about registering traditional and Salt clients running Red Hat Enterprise Linux operating systems.

Red Hat Enterprise Linux clients are based on Red Hat and are unrelated to SUSE Linux Enterprise Server with Expanded Support, RES, Red Hat, or SUSE Linux Enterprise Server. You are responsible for arranging access to Red Hat base media repositories and RHEL installation media, as well as connecting Uyuni Server to the Red Hat content delivery network. You must obtain support from Red Hat for all your RHEL systems. If you do not do this, you might be violating your terms with Red Hat.

Traditional clients are not available on Red Hat Enterprise Linux 8. Red Hat Enterprise Linux 8 clients are only supported as Salt clients.

Server Requirements

Before you begin, check that your Uyuni Server meets the requirements at [Installation › Hardware-requirements ›].

Taskomatic uses one CPU core, and requires at least 3072 MB of RAM. To ensure that taskomatic has access to enough memory, open the /etc/rhn/rhn.conf configuration file, and add this line:

```
taskomatic.java.maxmemory=3072
```

Restart Taskomatic:

```
systemctl restart taskomatic
```

Import Entitlements and CA Certificate

Red Hat clients require a Red Hat certificate authority (CA) and entitlement certificate, and an entitlement key.
Entitlement certificates are embedded with expiration dates, which match the length of the support subscription. To avoid disruption, you will need to repeat this process at the end of every support subscription period.

Red Hat supply a subscription manager tool to manage subscription assignments. It runs locally to track installed products and subscriptions. Clients must be registered with the subscription manager to obtain certificates.

Red Hat clients use a URL to replicate repositories. The URL will change depending on where the Red Hat client is registered.

Red Hat clients can be registered in three different ways:

- Red Hat content delivery network (CDN) at redhat.com
- Red Hat Satellite Server
- Red Hat update infrastructure (RHUI) in the cloud

This guide covers clients registered to Red Hat CDN. You must have at least one system registered to the CDN, with an authorized subscription for repository content.

Entitlement certificates for RHUI (cloud-based systems) only allow you to download content, not repository data. Satellite certificates for client systems require a Satellite server and subscription. Clients using Satellite certificates are not supported with Uyuni Server.

Entitlement certificates are embedded with expiration dates, which match the length of the support subscription. To avoid disruption, you will need to repeat this process at the end of every support subscription period.

Red Hat supplies the subscription-manager tool to manage subscription assignments. It runs locally on the client system to track installed products and subscriptions. Register to redhat.com with subscription-manager, then follow this procedure to obtain certificates.

Procedure: Registering Clients to Subscription Manager

1. On the client system, at the command prompt, register with the subscription manager tool:

   ```bash
   subscription-manager register
   ```

   Enter your Red Hat Portal username and password when prompted.

2. Copy your entitlement certificate and key from the client system, to a location that the Uyuni Server can access:

   ```bash
   cp /etc/pki/entitlement/ /<example>/entitlement/
   ```
Your entitlement certificate and key will both have a file extension of **.pem**. The key will also have key in the filename.

3. Copy the Red Hat CA Certificate file from the client system, to the same web location as the entitlement certificate and key:

   ```bash
   cp /etc/rhsm/ca/redhat-uep.pem /example/entitlement
   ```

To manage repositories on your Red Hat client, you need to import the CA and entitlement certificates to the Uyuni Server. This requires three entries: one each for the entitlement certificate, the entitlement key, and the Red Hat certificate.

**Procedure: Importing Certificates to the Server**

1. On the Uyuni Server Web UI, navigate to Systems › Autoinstallation › GPG and SSL Keys.

2. Click [Create Stored Key/Cert] and set these parameters for the entitlement certificate:
   - In the **Description** field, type **Entitlement-Cert-date**.
   - In the **Type** field, select **SSL**.
   - In the **Select file to upload** field, browse to the location where you saved the entitlement certificate, and select the **.pem** certificate file.

3. Click [Create Key].

4. Click [Create Stored Key/Cert] and set these parameters for the entitlement key:
   - In the **Description** field, type **Entitlement-key-date**.
   - In the **Type** field, select **SSL**.
   - In the **Select file to upload** field, browse to the location where you saved the entitlement key, and select the **.pem** key file.

5. Click [Create Key].

6. Click [Create Stored Key/Cert] and set these parameters for the Red Hat certificate:
   - In the **Description** field, type **redhat-uep**.
   - In the **Type** field, select **SSL**.
   - In the **Select file to upload** field, browse to the location where you saved the Red Hat certificate, and select the certificate file.

7. Click [Create Key].

**Repository Management**

You can use the subscription manager tool to get the URLs of the repositories you want to mirror:
You can use these repository URLs to create custom repositories. This allows you to mirror only the content you need to manage your clients.

For Red Hat 8 clients, add both the BaseOS and Appstream channels. You will require packages from both channels. If you do not add both channels, you will not be able to create the bootstrap repository, due to missing packages.

You can only create custom versions of Red Hat repositories if you have the correct entitlements in your Red Hat Portal.

**Procedure: Creating Custom Repositories**

1. On the Uyuni Server Web UI, navigate to **Software › Manage › Repositories**.

2. Click **[Create Repository]** and set these parameters for the entitlement certificate:
   - In the **Repository Label** field, type `rhel-7-server-rpms`.
   - In the **Repository URL** field, type the URL of the repository to mirror. For example, `https://cdn.redhat.com/content/dist/rhel/server/7/7Server/x86_64/os/`.
   - In the **Has Signed Metadata?** field, uncheck all Red Hat Enterprise Repositories.
   - In the **SSL CA Certificate** field, select `redhat-uep`.
   - In the **SSL Client Certificate** field, select `Entitlement-Cert-date`.
   - In the **SSL Client Key** field, select `Entitlement-Key-date`.
   - Leave all other fields as the default values.

3. Click **[Create Repository]**.

4. Repeat for every repository you want to define.

When you have created the custom repositories, you can create corresponding custom channels.

**Procedure: Creating Custom Channels**

1. On the Uyuni Server Web UI, navigate to **Software › Manage › Channels**.

2. Click **[Create Channel]** and set these parameters for the entitlement certificate. Ensure you use the correct RHEL version:
   - In the **Channel Name** field, type `RHEL 7 x86_64`.
   - In the **Channel Label** field, type `rhel7-x86_64-server`.
   - In the **Parent Channel** field, select `None`.
   - In the **Architecture** field, select `x86_64`. 
In the **Repository Checksum Type** field, select **sha1**.

In the **Channel Summary** field, type **RHEL 7 x86_64**.

In the **Organization Sharing** field, select **Public**.

3. Click **[Create Channel]**.

4. Navigate to the **Repositories** tab, check the appropriate repository, and click **[Update repositories]**.

5. OPTIONAL: Navigate to the **Sync** tab to set a recurring schedule for synchronization of this repository.

6. Click **[Sync Now]** to begin synchronization immediately.

---

Red Hat Enterprise Linux channels can be very large. Synchronization can sometimes take several hours.

When you have created the custom channels and synchronized them with the repositories, you can create child channels.

**Procedure: Creating Child Channels**

1. On the Uyuni Server Web UI, navigate to **Software › Manage › Channels**.

2. Click **[Create Channel]** and set these parameters for the entitlement certificate. Ensure you use the correct RHEL version:

   - In the **Channel Name** field, type **RHEL 7 x86_64**.
   - In the **Channel Label** field, type **rhel7-x86_64-extras**.
   - In the **Parent Channel** field, select **rhel7-x86_64-server**.
   - In the **Architecture** field, select **x86_64**.
   - In the **Repository Checksum Type** field, select **sha1**.
   - In the **Channel Summary** field, type **RHEL 7 x86_64 Extras**.
   - In the **Organization Sharing** field, select **Public**.

3. Click **[Create Channel]**.

4. Navigate to the **Repositories** tab, check the appropriate repository, and click **[Update repositories]**.

5. OPTIONAL: Navigate to the **Sync** tab to set a recurring schedule for synchronization of this repository.

6. Click **[Sync Now]** to begin synchronization immediately.

---

Red Hat Enterprise Linux channels can be very large. Synchronization can sometimes take several hours.
Add Client Tools

When you have set up all the custom channels, you can add the client tools.

For this section, you will require an activation key. For more information about activation keys, see [Client-configuration › Clients-and-activation-keys › ].

Your SUSE Manager subscription entitles you to the tools channels for SUSE Linux Enterprise Server with Expanded Support (also known as Red Hat Expanded Support or RES). You must use the client tools channel to create the bootstrap repository. This procedure applies to both traditional and Salt minions.

Procedure: Adding Client Tools Channels

1. On the Uyuni Server, add the appropriate Expanded Support channels:
   
   ◦ For Expanded Support 6:
   
   From the Web UI, add **RHEL6 Base x86_64** and **SUSE Linux Enterprise Client Tools RES6 x86_64**.
   
   From the command prompt, add **rhel-x86_64-server-6** and **res6-suse-manager-tools-x86_64**.
   
   ◦ For Expanded Support 7:
   
   From the Web UI, add **RHEL7 Base x86_64** and **SUSE Linux Enterprise Client Tools RES7 x86_64**.
   
   From the command prompt, add **rhel-x86_64-server-7** and **res7-suse-manager-tools-x86_64**.
   
   ◦ For Expanded Support 8:
   
   From the Web UI, add **RHEL8 Base x86_64** and **SUSE Manager Tools for RHEL and ES 8 x86_64**. You will also need to add the Appstream channel.
   
   From the command prompt, add **rhel-x86_64-server-8** and **res8-suse-manager-tools-x86_64**.

2. Synchronize the Uyuni Server with the SUSE Customer Center. You can do this using the Web UI, or by running **mgr-sync** at the command prompt.

3. Add the new channel to your activation key.

You can choose to disable the Red Hat Enterprise Linux subscription-manager yum plugins.

The yum plugins are disabled with a configuration Salt state.
This procedure is optional.

**Procedure: Creating a Salt State to Deploy Configuration Files**

1. On the Uyuni Server Web UI, navigate to **Configuration › Channels**.

2. Click **[Create State Channel]**
   - In the **Name** field, type `subscription-manager: disable yum plugins`.
   - In the **Label** field, type `subscription-manager-disable-yum-plugins`.
   - In the **Description** field, type `subscription-manager: disable yum plugins`.
   - In the **SLS Contents** field, leave it empty.

3. Click **[Create Config Channel]**

4. Click **[Create Configuration File]**
   - In the **Filename/Path** field, type `/etc/yum/pluginconf.d/subscription-manager.conf`.
   - In the **File Contents** field, type:

     ```
     [main]
     enabled=0
     ```

1. Click **[Create Configuration File]**

2. Take note of the value of the field **Salt Filesystem Path**.

3. Click on the name of the Configuration Channel.

4. Click on **View/Edit 'init.sls' File**
   - In the **File Contents** field, type:

     ```
     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
     ```

1. Click **[Update Configuration File]**

**Procedure: Creating a System Group for Red Hat Enterprise Linux Clients**

1. On the Uyuni Server Web UI, navigate to **Systems › System Groups**.

2. Click **[Create Group]**.
   - In the **Name** field, type `rhel-systems`. 
In the Description field, type All RHEL systems.

3. Click [Create Group].

4. Click States tab.

5. Click Configuration Channels tab.

6. Type subscription-manager: disable yum plugins at the search box.

7. Click [Search] and the state will appear.

8. Click the checkbox for the state at the Assign column.

9. Click [Save changes].

10. Click [Confirm].

If you already have RHEL systems added to Uyuni, assign them to the new system group, and then apply the highstate.

Procedure: Adding the System Group to Activation Keys

You need to modify the activation keys you used for RHEL systems to include the system group created above.

1. On the Uyuni Server Web UI, navigate to Systems › Activation Keys.

2. For each the Activation Keys you used for RHEL systems, click on it and:

3. Navigate to the Groups tab, and the Join subtab.

4. Check Select rhel-systems.

5. Click [Join Selected Groups].

Trust GPG Keys on Clients

By default, Red Hat Enterprise Linux does not trust the GPG key for Uyuni Expanded Support client tools.

The clients can be successfully bootstrapped without the GPG key being trusted. However, they will not be able to install new client tool packages or update them. If this occurs, add GPG key to the ORG_GPG_KEY= parameter in all Red Hat Enterprise Linux bootstrap scripts.

For example, for SLES ES 6 (RES6-SUSE-Manager-Tools) use:

```
sle11-gpg-pubkey-307e3d54.key
```

For example, for SLES ES 7 (RES7-SUSE-Manager-Tools) and SLES ES 8 (RES8-SUSE-Manager-Tools), and Ubuntu 16.04 (Ubuntu-16.04-SUSE-Manager-Tools) and Ubuntu 18.04 (Ubuntu-18.04-SUSE-Manager-Tools) use:
You will find all keys available on the server in /srv/www/htdocs/pub/:

- sle12-gpg-pubkey-39db7c82.key
- ptf-gpg-pubkey-b37b98a9.key
- res-gpg-pubkey-0182b964.key
- sle10-gpg-pubkey-9c800aca.key
- sle11-gpg-pubkey-307e3d54.key
- sle12-gpg-pubkey-39db7c82.key
- sle12-reserve-gpg-pubkey-50a3dd1c.key

You do not need to delete any previously stored keys.

If you are bootstrapping clients from the Uyuni Web UI, you will need to use a Salt state to trust the key. Create the Salt state and assign it to the organization. You can then use an activation key and configuration channels to deploy the key to the clients.

Register Clients

To register your Red Hat clients, you need a bootstrap repository. By default, bootstrap repositories are automatically created, and regenerated daily for all synchronized products. You can manually create the bootstrap repository from the command prompt, using this command:

```
mgr-create-bootstrap-repo --with-custom-channels
```

For more information on registering your clients, see [Client-configuration › Registration-overview › ].

Package Management and Red Hat Enterprise Linux 8 Clients

If you are using Red Hat Enterprise Linux 8 clients, you cannot perform package operations such as installing or upgrading directly from modular repositories like the Red Hat Enterprise Linux Appstream repository. You can use the Appstream filter with content lifecycle management to transform modular repositories into regular repositories.

For more information about content lifecycle management, see [Administration › Content-lifecycle › ].

Registering CentOS Clients

This section contains information about registering traditional and Salt clients running CentOS operating systems.
CentOS clients are based on CentOS and are unrelated to SUSE Linux Enterprise Server with Expanded Support, RES, Red Hat, or Expanded Support. You are responsible for arranging access to CentOS base media repositories and CentOS installation media, as well as connecting Uyuni Server to the CentOS content delivery network.

Traditional clients are not available on CentOS 8. CentOS 8 clients are only supported as Salt clients.

Server Requirements

Before you begin, check that your Uyuni Server meets the requirements at [Installation › Hardware-requirements › ].

Taskomatic uses one CPU core, and requires at least 3072 MB of RAM. To ensure that taskomatic has access to enough memory, open the /etc/rhn/rhn.conf configuration file, and add this line:

```
taskomatic.java.maxmemory=3072
```

Restart Taskomatic:

```
systemctl restart taskomatic
```

Channel and Repository Management

For CentOS 8 clients, add both the BaseOS and Appstream channels. You will require packages from both channels. If you do not add both channels, you will not be able to create the bootstrap repository, due to missing packages.

Procedure: Adding Channels and Repositories

1. At the command prompt on the Uyuni Server, as root, install the spacewalk-utils package:

```
zypper in spacewalk-utils
```

2. Add the CentOS base, updates, and client channels, specifying the CentOS version and architecture:

   - For CentOS 6:

```
spacewalk-common-channels -a x86_64 centos6 \
centos6-uyuni-client centos6-updates
```

   - For CentOS 7:
For CentOS 7:

```bash
spacewalk-common-channels -a x86_64 centos7 \ centos7-uyuni-client centos7-updates
```

For CentOS 8:

```bash
spacewalk-common-channels -a x86_64 centos8 \ centos8-uyuni-client centos8-appstream
```

The client tools channel provided by `spacewalk-common-channels` is sourced from Uyuni and not from SUSE.

**Procedure: Synchronizing CentOS repositories**

1. In the Uyuni Web UI, navigate to **Software › Manage**, and check every CentOS channel.
2. In the **Repositories** tab, navigate to the **Sync** subtab, and click **[Sync Now]**. You can also create a regular synchronization schedule on this page.

**Create an Activation Key**

You will need to create an activation key that is associated with your CentOS channels.

For more information on activation keys, see [Client-configuration › Clients-and-activation-keys ›](#).

**Register Clients**

CentOS clients are registered in the same way as all other clients. For more information, see [Client-configuration › Registration-overview ›](#).

**Manage Errata**

When you update CentOS clients, the packages do not include metadata about the updates. You can use a third-party errata service to obtain this information.

The third-party errata service described here is provided and maintained by the community. It is not supported by SUSE.

**Procedure: Installing an Errata Service**

1. From the command prompt, as root, add the `sle-module-development-tools` module:

   ```bash
   SUSEConnect --product sle-module-development-tools/15.1/x86_64
   ```

2. Install the errata service:
zypper in perl-Text-Unidecode

3. Create a file for your errata script:

```
touch /usr/local/bin/cent-errata.sh
```

4. Edit the new file to include this script, editing the repository details as required. This script fetches the errata details from an external errata service, unpacks it, and publishes the details:

```
#!/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 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-x86_64-updates,centos7-x86_64,centos7-x86_64-extras
--publish --rhsa-oval /usr/local/centos/com.redhat.rhsa-RHEL7.xml
```

5. Set up a cron job to run the script daily:

```
ln -s /usr/local/bin/cent-errata.sh /etc/cron.daily
```

For more information on this tool, see https://cefs.steve-meier.de/.

**Registering Ubuntu Clients**

This section contains information about registering Salt clients running Ubuntu operating systems.

Uyuni supports Ubuntu 16.04 LTS and 18.04 LTS Clients using Salt. Traditional clients are not supported.

⚠️ Canonical does not endorse or support Uyuni.

Bootstrapping is supported for starting Ubuntu clients and performing initial state runs such as setting repositories and performing profile updates. However, the root user on Ubuntu is disabled by default, so to use bootstrapping, you will require an existing user with `sudo` privileges for Python.
Prepare to Register

Some preparation is required before you can register Ubuntu clients to the Uyuni Server.

Procedure: Adding client tools channels

Before you begin, ensure you have the Ubuntu product enabled, and have synchronized the Ubuntu channels for SUSE Customer Center:

You can do this from Web UI or using command prompt, at your choice.

For Ubuntu 16.04:

• From the Web UI, add Ubuntu 16.04 and SUSE Linux Enterprise Client Tools Ubuntu 1604 amd64.

• From the command prompt, add ubuntu-16.04-pool-amd64 and ubuntu-16.04-suse-manager-tools-amd64.

For Ubuntu 18.04:

• From the Web UI, add Ubuntu 18.04 and SUSE Linux Enterprise Client Tools Ubuntu 1804 amd64.

• From the command prompt, add ubuntu-18.04-pool-amd64 and ubuntu-18.04-suse-manager-tools-amd64.

The mandatory channels do not contain Ubuntu upstream packages. The repositories and channels for synchronizing upstream content must be configured manually.

In the Uyuni Web UI, navigate to Software › Channel List › All. Verify that you have a base channel and a child channel for your architecture.

For example, for Ubuntu 18.04:

• Base channel: ubuntu-18.04-pool for amd64

• Child channel: Ubuntu-18.04-SUSE-Manager-Tools for amd64

Procedure: Creating Custom Channels and Repositories

You need to manually create three repositories:

• For main

• For main updates

• For main security

These examples are for Ubuntu 18.04 (bionic). Make sure you change the values to match the Ubuntu
In these procedures, we use http://archive.ubuntu.com/ for Ubuntu repositories. You can replace this URL with a more appropriate mirror.

1. On the Uyuni Server Web UI, navigate to Software › Manage › Repositories.

2. Click [Create Repository] and set these parameters for the main repository:
   - In the Repository Label field, type ubuntu-bionic-main.
   - In the Repository URL field, type http://archive.ubuntu.com/ubuntu/dists/bionic/main/binary-amd64/.
   - In the Repository Type field, select deb.
   - Leave all other fields as the default values.

3. Click [Create Repository].

4. Navigate to Software › Manage › Repositories.

5. Click [Create Repository] and set these parameters for the main-updates repository:
   - In the Repository Label field, type ubuntu-bionic-main-updates.
   - In the Repository URL field, type http://archive.ubuntu.com/ubuntu/dists/bionic-updates/main/binary-amd64/.
   - In the Repository Type field, select deb.
   - Leave all other fields as the default values.

6. Click [Create Repository].

7. Navigate to Software › Manage › Repositories.

8. Click [Create Repository] and set these parameters for the main-security repository:
   - In the Repository Label field, type ubuntu-bionic-main-security.
   - In the Repository Type field, select deb.
   - Leave all other fields as the default values.

9. Click [Create Repository].

When you have created the repositories, you can create the custom channels, one for each repository:

1. On the Uyuni Server Web UI, navigate to Software › Manage › Channels.

2. Click [Create Channel] and set these parameters for the entitlement certificate:
   - In the Channel Name field, type Ubuntu 18.04 LTS AMD64 Main.
- In the **Channel Label** field, type `ubuntu-1804-amd64-main`.
- In the **Parent Channel** field, select `ubuntu-18.04-pool for amd64`.
- In the **Architecture** field, select **AMD64 Debian**.
- In the **Repository Checksum Type** field, select `sha1`.
- In the **Channel Summary** field, type `Ubuntu 18.04 LTS AMD64 Main`.
- In the **Organization Sharing** field, select **Public**.

3. Click [Create Channel].

4. Navigate to **Software › Manage › Channels**.

5. Click [Create Channel] and set these parameters for the entitlement certificate.

   - In the **Channel Name** field, type `Ubuntu 18.04 LTS AMD64 Main Updates`.
   - In the **Channel Label** field, type `ubuntu-1804-amd64-main-updates`.
   - In the **Parent Channel** field, select `ubuntu-18.04-pool for amd64`.
   - In the **Architecture** field, select **AMD64 Debian**.
   - In the **Repository Checksum Type** field, select `sha1`.
   - In the **Channel Summary** field, type `Ubuntu 18.04 LTS AMD64 Main Updates`.
   - In the **Organization Sharing** field, select **Public**.

6. Click [Create Channel].

7. Navigate to **Software › Manage › Channels**.

8. Click [Create Channel] and set these parameters for the entitlement certificate.

   - In the **Channel Name** field, type `Ubuntu 18.04 LTS AMD64 Main Security`.
   - In the **Channel Label** field, type `ubuntu-1804-amd64-main-security`.
   - In the **Parent Channel** field, select `ubuntu-18.04-pool for amd64`.
   - In the **Architecture** field, select **AMD64 Debian**.
   - In the **Repository Checksum Type** field, select `sha1`.
   - In the **Channel Summary** field, type `Ubuntu 18.04 LTS AMD64 Main Security`.
   - In the **Organization Sharing** field, select **Public**.

9. Click [Create Channel].

Your custom channels should use this structure, at **Software › Channel List › All** (example for Ubuntu 18.04):

- Base channel (vendor): `ubuntu-18.04-pool for amd64`
• Child custom channel: Ubuntu 18.04 LTS AMD64 Main
• Child custom channel: Ubuntu 18.04 LTS AMD64 Main Updates
• Child custom channel: Ubuntu 18.04 LTS AMD64 Main Security
• Child vendor channel: Ubuntu-18.04-SUSE-Manager-Tools for amd64

When you have the channels set up, associate each channel with the appropriate repository, and synchronize them (optionally configure scheduled synchronization).

To do this, proceed to Software › Manage › Channels, and for each channel you created, click on it and:

1. Navigate to the Repositories
2. Navigate to the Sync and click [Sync Now] to begin synchronization immediately.
3. You can also setup a scheduled synchronization from this screen.

You need all the new channels fully synchronized before bootstrapping any Ubuntu client.

Procedure: Alternative Method for Adding Ubuntu Channels and Repositories

1. At the command prompt on the Uyuni Server, as root, install the spacewalk-utils package:

   ```
   zypper in spacewalk-utils
   ```

2. Add the Ubuntu channels.

   For Ubuntu 16.04:

   ```
   spacewalk-common-channels \
   ubuntu-1604-amd64-main \
   ubuntu-1604-amd64-main-updates \
   ubuntu-1604-amd64-main-security
   ```

   For Ubuntu 18.04:

   ```
   spacewalk-common-channels \
   ubuntu-1804-amd64-main \
   ubuntu-1804-amd64-main-updates \
   ubuntu-1804-amd64-main-security
   ```

When you have the channels set up, associate each channel with the appropriate repository, and synchronize them. You can also configure a synchronization schedule.

To manually synchronize the channels, navigate to Software › Manage › Channels. Click each channel in the list, and:
1. Navigate to the **Repositories** tab.

2. Navigate to the **Sync** subtab.

3. Click **[Sync Now]** to begin synchronization immediately.

4. You can also create a synchronization schedule from this screen.

Ubuntu channels can be very large. Synchronization can sometimes take several hours.

When you have the channels set up, associate each channel with the appropriate repository, and synchronize the channels (optionally configure scheduled synchronization).

To manually synchronize the channels, navigate to **Software › Manage › Channels**. Click each channel in the list, and navigate to menu: [Repositories › Sync]. Click **[Sync Now]** to begin synchronization immediately. You can also create a synchronization schedule from this screen.

### Monitor Synchronization Progress

You can check if a channel has finished synchronizing:

- In the Uyuni Web UI, navigate to **Admin › Setup Wizard** and select the **SUSE Products** tab.

  This dialog displays a completion bar for each product when they are being synchronized.

- Check the synchronization log file at the command prompt with `tail -f /var/log/rhn/reposync/channel-label.log`.

  Each child channel will generate its own log during the synchronization progress.

You will need to check all the base and child channel log files to be sure that the synchronization is complete.

### Root Access

The root user on Ubuntu is disabled by default. You can enable it by editing the `sudoers` file.

**Procedure: Granting Root User Access**

1. On the client, edit the `sudoers` file:

   ```
   sudo visudo
   ```

   Grant `sudo` access to the user by adding this line to the `sudoers` file. Replace `<user>` with the name of the user that will be used to bootstrap the client in the Web UI:
This procedure grants root access without requiring a password, which is required for registering the client. When the client is successfully installed it will run with root privileges, so the access is no longer required. We recommend that you remove the line from the `sudoers` file after the client has been successfully installed.

Register Clients

To register your Ubuntu clients, you need a bootstrap repository. By default, bootstrap repositories are automatically created, and regenerated daily for all synchronized products. You can manually create the bootstrap repository from the command prompt, using this command:

```
mgr-create-bootstrap-repo --with-custom-channels
```

For more information on registering your clients, see [Client-configuration › Registration-overview ›].

Registering openSUSE Clients

This section contains information about registering Salt clients running openSUSE operating systems. Uyuni supports openSUSE Leap 15.1 Clients using Salt. Traditional clients are not supported.

Bootstrapping is supported for starting openSUSE clients and performing initial state runs such as setting repositories and performing profile updates.

Prepare to Register

Some preparation is required before you can register openSUSE clients to the Uyuni Server.

Monitor Synchronization Progress

There are two ways to check if a channel has finished synchronizing:

- In the Uyuni Web UI, navigate to Admin › Setup Wizard and select the SUSE Products tab. This dialog displays a completion bar for each product when they are being synchronized.

- Check the synchronization log file at the command prompt:

  ```
  [command]"tail -f /var/log/rhn/reposync/<channel-label>.log".
  ```

  Each child channel will generate its own log during the synchronization progress. You will need to
check all the base and child channel log files to be sure that the synchronization is complete.

Register Clients

To register your openSUSE clients, you need a bootstrap repository. By default, bootstrap repositories are automatically created, and regenerated daily for all synchronized products. You can manually create the bootstrap repository from the command prompt, using this command:

```
mgr-create-bootstrap-repo --with-custom-channels
```

For more information on registering your clients, see [Client-configuration › Registration-overview ›].
Virtualization

You can use Uyuni to manage virtualized clients in addition to regular traditional or Salt clients. In this type of installation, a virtual host is installed on the Uyuni Server to manage any number of virtual guests. If you choose to, you can install several virtual hosts to manage groups of guests.

The range of capabilities that virtualized clients have depends on the third-party virtualization provider you choose.

Xen and KVM hosts and guests can be managed directly in Uyuni. This enables you to autoinstall hosts and guests using AutoYaST or Kickstart, and manage guests in the Web UI.

For VMWare, including VMWare vSphere, Uyuni requires you to set up a virtual host manager (VHM) to control the VMs. This gives you control over the hosts and guests, but in a more limited way than available with Xen and KVM.

Other third-party virtualization providers are not directly supported by Uyuni. However, if your provider allows you to export a JSON configuration file for the VM, you can upload that configuration file to Uyuni and manage it with a VHM.

Virtualization with Xen and KVM

Xen and KVM virtualized clients can be managed directly in Uyuni.

To begin, you will need to set up a virtual host on your Uyuni Server. You can then set up autoinstallation using AutoYaST or Kickstart for future virtual hosts, and for virtual guests.

This section also includes information about administering your virtual guests after they have been installed.

Host Setup

The way that you set up Xen or KVM on a VM host depends on what operating system you want to use on its associated guests.


For Red Hat Enterprise Linux operating systems, refer to the Red Hat documentation for your version.

Uyuni uses `libvirt` to install and manage guests. You must have the `libvirtd` package installed on your host. In most cases, the default settings are usually sufficient, and you should not need to adjust them. However, if you want to access the VNC console on your guests as a non-root user, you will need to perform some configuration changes. For more information about how to set this up, consult the relevant documentation for your operating system.

You will require a bootstrap script on the Uyuni Server. Your bootstrap script must include the activation
key for your host. We also recommend that you include your GPG key for additional security. For more on creating a bootstrap script, see [Client-configuration › Registration-bootstrap ›].

When your bootstrap script is ready, execute it on the host to register it with the Uyuni Server. For more on client registration, see [Client-configuration › Registration-overview ›].

For Salt clients, you will need to enable the Virtualization Host entitlement. This allows you to see VM changes instantly. To do this, in the Uyuni Web UI, navigate to the System Details page for the host, and click on the Properties tab. Alternatively, the Virtualization Host entitlement can be added at the registration key level. In the Add-On System Types section, check Virtualization Host, and click [Update Properties] to save the changes. Restart the Salt minion service to activate the change:

```
systemctl restart salt-minion
```

For traditional clients, by default, VM hosts use the rhnsd service to check for scheduled actions. The check occurs every four hours, in order to balance load in environments where there are a lot of clients. This can create delays of up to four hours before an action is carried out. When you are managing VM guests, this long delay is not always ideal, especially for actions like rebooting a guest. To address this, you can disable the rhnsd service, and enable the osad service. The osad service receives commands using a jabber protocol, and will execute commands instantly.

To disable the rhnsd service, and enable the osad daemon, run these commands as the root user:

```
service rhnsd stop
service rhnsd disable
service osad enable
service osad start
```

Autoinstallation

You can use AutoYaST or Kickstart to automatically install and register Xen and KVM guests.

You will require an activation key for the VM host you want to register the guests to, and for each guest. Your activation key must have the provisioning and Virtualization Platform entitlements. Your activation key must also have access to the mgr-virtualization-host and mgr-osad packages. For more on creating activation keys, see [Client-configuration › Clients-and-activation-keys ›].

If you want to automatically register the guests with Uyuni after installation, you will need to create a bootstrap script. For more on creating a bootstrap script, see [Client-configuration › Registration-bootstrap ›].
Autoinstallation of VM guests works only if they are configured as Traditional clients. Salt clients can be created using a template disk image, but not by using AutoYaST or Kickstart.

Create an Autoinstallable Distribution

You will need to create an autoinstallable distribution on the VM host to be able to autoinstall clients from Uyuni. The distribution can be made available from a mounted local or remote directory, or on a loop-mounted ISO image.

The configuration of the autoinstallable distribution will differ depending on whether you are using a SLES or Red Hat Enterprise Linux operating system on your guests. The packages for a Red Hat Enterprise Linux installation are fetched from the associated base channel. Packages for installing SUSE systems are fetched from the autoinstallable distribution. Therefore, for SLES systems, the autoinstallable distribution must be a complete installation source.

Table 7. Paths for autoinstallable distributions

<table>
<thead>
<tr>
<th>Operating System Type</th>
<th>Kernel Location</th>
<th>initrd Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Red Hat Enterprise Linux</td>
<td>images/pxeboot/vmlinuz</td>
<td>images/pxeboot/initrd.img</td>
</tr>
<tr>
<td>SLES</td>
<td>boot/&lt;arch&gt;/loader/initrd</td>
<td>boot/&lt;arch&gt;/loader/linux</td>
</tr>
</tbody>
</table>

In all cases, ensure that the base channel matches the autoinstallable distribution.

Before you begin, ensure you have a installation media available to your VM Host. It can be on a network resource, a local directory, or an loop-mounted ISO image. Additionally, ensure that all files and directories are world-readable.

Procedure: Creating an Autoinstallable Distribution

1. In the Uyuni Web UI, navigate to Systems › Autoinstallation › Distributions and click Create Distribution.
2. In the Create Autoinstallable Distribution section, use these parameters:
   
   - In the Distribution Label section, type a unique name for the distribution. Use only letters, numbers, hyphens (-), periods (.), and underscores (_), and ensure the name is longer than four characters.
   
   - In the Tree Path field, type an absolute path to the installation source.
   
   - In the Base Channel field, select the channel that matches the installation source. This channel is used as the package source for non-SUSE installations.
   
   - In the Installer Generation field, select the operating system version that matches the installation source.
In the **Kernel Options** field, type any options to be passed to the kernel when booting for the installation. The `install=` parameter and the `self_update=0` parameter are added by default.

In the **Post Kernel Options** section, type any options to be passed to the kernel when booting the installed system for the first time.

3. Click [Create Autoinstallable Distribution] to save.

When you have created an autoinstallable distribution, you can edit it by navigating to Systems › Autoinstallation › Distributions and selecting the distribution you want to edit.

---

### Create and Upload an Autoinstallation Profile

Autoinstallation profiles contain all the installation and configuration data needed to install a system. They can also contain scripts to be executed after the installation is complete.

Kickstart profiles can be created using the Uyuni Web UI, by navigating to Systems › Autoinstallation › Profiles, clicking [Create New Kickstart File], and following the prompts. You can also create AutoYaST or Kickstart autoinstallation profiles by hand.

An example AutoYaST profile that includes a script for registering the client with Uyuni is available in [Client-configuration › Autoyast-example › ]. If you are using AutoYaST to install SLES, you will also need to include this snippet:

```xml
<products config:type="list">
  <listentry>SLES</listentry>
</products>
```

- For more on AutoYaST, see [Client-configuration › Client-automating-installation › ].
- For more on Kickstart, see [Client-configuration › Kickstart › ], or refer to the Red Hat documentation for your installation.

---

### Procedure: Uploading an Autoinstallation Profile

1. In the Uyuni Web UI, navigate to Systems › Autoinstallation › Profiles and click [Upload Kickstart/AutoYaST File].

2. In the **Create Autoinstallation Profile** section, use these parameters:
   - In the **Label** field, type a unique name for the profile. Use only letters, numbers, hyphens (-), periods (.), and underscores (_), and ensure the name is longer than six characters.
   - In the **Autoinstall Tree** field, select the autoinstallable distribution you created earlier.
   - In the **Virtualization Type** field, select the relevant Guest type (for example, **KVM Virtualized Guest**). Do not choose **Xen Virtualized Host** here.
   - **OPTIONAL:** If you want to manually create your autoinstallation profile, you can type it directly into the **File Contents** field. If you have a file already created, leave the **File**
In the **File to Upload** field, click [Choose File], and use the system dialog to select the file to upload. If the file is successfully uploaded, the filename will be shown in the **File to Upload** field.

- The contents of the uploaded file will be shown in the **File Contents** field. If you need to make edits, you can do so directly.

3. Click [Create] to save your changes and store the profile.

When you have created an autoinstallation profile, you can edit it by navigating to **Systems › Autoinstallation › Profiles** and selecting the profile you want to edit. Make the desired changes and save your settings by clicking [Create].

If you change the **Virtualization Type** of an existing Kickstart profile, it might also modify the bootloader and partition options, potentially overwriting any custom settings. Carefully review the **Partitioning** tab to verify these settings before making changes.

### Automatically Register Guests

When you install VM guests automatically, they are not registered to Uyuni. If you want your guests to be automatically registered as soon as they are installed, you can add a section to the autoinstallation profile that invokes a bootstrap script, and registers the guests.

This section gives instructions for adding a bootstrap script to an existing AutoYaST profile.

For more on creating a bootstrap script, see [Client-configuration › Registration-bootstrap › ] for instructions on how to do this for `{kickstart}`, refer to the Red Hat documentation for your installation.

**Procedure: Adding a Bootstrap Script to an AutoYaST Profile**

1. Ensure your bootstrap script contains the activation key for the VM guests you want to register with it, and that is located on the host at `/srv/www/htdocs/pub/bootstrap_vm_guests.sh`.

2. In the Uyuni Web UI, navigate to **Systems › Autoinstallation › Profiles**, and select the AutoYaST profile to associate this script with.

3. In the **File Contents** field, add this snippet at the end of the file, immediately before the closing `</profile>` tag. Ensure you replace the example IP address in the snippet with the correct IP address for your Uyuni Server:
4. Click **Update** to save your changes.

If your AutoYaST profile already contains a `<scripts>` section, do not add a second one. Place the bootstrap snippet inside the existing `<scripts>` section.

### Autoinstall VM Guests

Once you have everything set up, you can start to autoinstall your VM guests.

Each VM host can only install one guest at a time. If you are scheduling more than one autoinstallation, make sure you time them so that the next installation does not begin before the previous one has completed. If a guest installation starts while another one is still running, the running installation will be canceled.

1. In the Uyuni Web UI, navigate to **Systems > Overview**, and select the VM host you want to install guests on.
2. Navigate to the **Virtualization** tab, and the **Provisioning** subtab.
3. Select the autoinstallation profile you want to use, and specify a unique name for the guest.
4. Choose a proxy if applicable and enter a schedule.
5. To change the guest’s hardware profile and configuration options, click **[Advanced Options]**.
6. Click **[Schedule Autoinstallation and Finish]** to complete.

### Manage VM Guests

You can use the Uyuni Web UI to manage your VM Guests, including actions like shutting down, restarting, and adjusting CPU and memory allocations.

To do this, you will need your Xen or KVM VM host registered to the Uyuni Server, and have the **libvirtd** service running on the host. For traditional clients, you will also need the **mgr-cfg-actions** package installed on your Uyuni Server.

In the Uyuni Web UI, navigate to **Systems > System List**, and click on the VM host for the guests you want to manage. Navigate to the **Virtualization** tab to see all guests registered to this host, and access the management functions.
Virtualization with VMWare

You can use VMWare vSphere virtual machines, including ESXi and vCenter, with Uyuni by setting up a virtual host manager (VHM).

To begin, you will need to set up a VHM on your Uyuni Server, and inventory the available VM hosts. Taskomatic can then begin data collection using the VMs API.

VHM Setup

The Virtual Host Manager (VHM) runs on the Uyuni Server.

To run a VHM, your Uyuni Server will need to have port 443 open, in order to access the VMWare API.

VMWare hosts use access roles and permissions to control access to hosts and guests. Ensure that any VMWare objects or resources that you want to be inventoried by the VHM have at least read-only permissions. If you want to exclude any objects or resources, mark them with no-access.

When you are adding new hosts to Uyuni, you will need to consider if the roles and permissions that have been assigned to users and objects need to be inventoried by Uyuni.

For more on users, roles, and permissions, see the VMWare vSphere documentation: https://docs.vmware.com/en/VMware-vSphere/index.html

Procedure: Creating a VMWare VHM

1. In the Uyuni Web UI, navigate to Systems › Virtual Host Managers.
2. Click [Create] and select VMWare-based.
3. In the Add a VMWare-based Virtual Host Manager section, use these parameters:
   - In the Label field, type a custom name for your VHM.
   - In the Hostname field, type the fully-qualified domain name (FQDN) or host IP address.
   - In the Port field, type the ESXi API port to use (for example, 443).
   - In the Username field, type the username associated with the VM host.
   - In the Password field, type the password associated with the VM host user.
4. Click [Create] to save your changes and create the VHM.
5. On the Virtual Host Managers page select the new VHM.
6. On the Properties page, click [Refresh Data] to inventory the new VHM.

To see which objects and resources have been inventoried, navigate to Systems › System List › Virtual Systems.
Connecting to the ESXi server from a browser using HTTPS can sometimes log an invalid certificate error. If this occurs, refreshing the data from the virtual hosts server will fail. To correct the problem, extract the certificate from the ESXi server, and copy it to /etc/pki/trust/anchors. Re-trust the certificate by running the update-ca-certificates command on the command line, and restart the spacewalk services.

After your VHM has been created and configured, Taskomatic will run data collection automatically. If you want to manually perform data collection, navigate to Systems › Virtual Host Managers, select the appropriate VHM, and click [Refresh Data].

Uyuni ships with a tool called virtual-host-gatherer that can connect to VHMs using their API, and request information about virtual hosts. virtual-host-gatherer maintains the concept of optional modules, where each module enables a specific VHM. This tool is automatically invoked nightly by Taskomatic. Log files for the virtual-host-gatherer tool are located at /var/log/rhn/gather.log.

Virtualization with Other Third Party Providers

If you want to use a third-party virtualization provider other than Xen, KVM, or VMware, you can import a JSON configuration file to Uyuni.

Similarly, if you have a VMWare installation that does not provide direct access to the API, a file-based VHM will provide you with some basic management features.

This option is for importing files that have been created with the virtual-host-gatherer tool. It is not designed for manually created files.

Procedure: Exporting and Importing a JSON File

1. Export the JSON configuration file by running virtual-host-gatherer on the VM network.
2. Save the produced file to a location accessible by your Uyuni Server.
3. In the Uyuni Web UI, navigate to Systems › Virtual Host Managers.
4. Click [Create] and select File-based.
5. In the Add a file-based Virtual Host Manager section, use these parameters:
   - In the Label field, type a custom name for your VHM.
   - In the Url field, type the path to your exported JSON configuration file.
6. Click [Create] to save your changes and create the VHM.
7. On the Virtual Host Managers page, select the new VHM.
8. On the Properties page, click [Refresh Data] to inventory the new VHM.
Listing 2. Example: Exported JSON configuration file:

```json
{
    "examplevhost": {
        "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": {
                "4230c6bf-3f98-2a65-f7c3-600b26b79c22": "4230d43e-aafe-38ba-5a9e-3cb67c03a16a",
                "5a2e4e63-a957-426b-bfa8-4169302e4fdd": "42307b15-1618-0595-01f2-427f4cddd88e",
                "NSX-gateway": "4230d43e-aafe-38ba-5a9e-3cb67c03a16a",
                "NSX-3gateway": "4230b00f-0b21-0e9d-dfde-6c7b06909d5f",
                "NSX-service": "4230e924-b714-198b-348b-25de01482fd9"
            }
        }
    }
}
```

For more information, see the man page on your Uyuni server for `virtual-host-gatherer`:

```bash
man virtual-host-gatherer
```

The `README` file of that package provides background information about the `type` of a hypervisor, etc.:

```bash
/usr/share/doc/packages/virtual-host-gatherer/README.md
```

The man page and the `README` file also contain example configuration files.
Virtual Host Managers

Virtual Host Managers (VHMs) are used to gather information from a range of client types.

VHMs can be used to collect private or public cloud instances and organize them into virtualization groups. With your virtualized clients organized this way, Taskomatic collects data on the clients for display in the Uyuni Web UI. VHMs also allow you to use subscription matching on your virtualized clients.

You can create a VHM on your Uyuni Server, and use it to inventory available public cloud instances. You can also use a VHM to manage clusters created with Kubernetes and SUSE CaaS Platform.

For more information on using a VHM with Microsoft Azure, see [Client-configuration › Vhm-azure ›]. For more information on using a VHM with Amazon Web Services, see [Client-configuration › Vhm-aws ›]. For more information on using a VHM with Google Compute Engine, see [Client-configuration › Vhm-gce ›]. For more information on using a VHM with Kubernetes, see [Client-configuration › Vhm-kubernetes ›].

SUSE Support and VM Zones

Public cloud providers use regions to define the physical geographic location of the datacenter providing virtual machines. For example, US-East, or Asia.

Regions are then further divided into zones. For example, the US-East region might contain zones called us-east-2a and us-east-2b, among others.

SUSE uses the zone of a virtual machine to determine the appropriate subscription to provide. If all of your VMs are provided by the same zone, you are within the terms and conditions of the 1-2 Virtual Machines subscription.

If your VMs are provided by different zones, even if they are within the same region, you might not meet the conditions of the 1-2 Virtual Machines subscription. In this case, check your subscription carefully.

For BYOS instances (bring your own subscription), all installed products are passed to the subscription matcher. If your public cloud instances are PAYG (pay as you go), their base products will be excluded from the subscription matcher counting.

The calculation about whether an instance is PAYG or BYOS is done at the time of registration or when a hardware refresh action is executed.

For more information, see https://www.suse.com/products/terms_and_conditions.pdf or contact SUSE.

VHM and Amazon Web Services

You can use a Uyuni VHM to gather instances from Amazon Web Services (AWS).
The VHM allows Uyuni to obtain and report information about your clusters. For more information on VHMs, see [Client-configuration › Vhm › ].

Create an Amazon EC2 VHM

The Virtual Host Manager (VHM) runs on the Uyuni Server.

Ensure you have installed the virtual-host-gatherer-libcloud package on the Uyuni Server.

Procedure: Creating an Amazon EC2 VHM

1. In the Uyuni Web UI, navigate to Systems › Virtual Host Managers.
2. Click [Create] and select Amazon EC2 from the drop-down menu.
3. In the Add an Amazon EC2 Virtual Host Manager section, use these parameters:
   - In the Label field, type a custom name for your VHM.
   - In the Access Key ID field, type the access key ID provided by Amazon.
   - In the Secret Access Key field, type the secret access key associated with the Amazon instance.
   - In the Region field, type the region to use.
   - In the Zone field, type the zone your VM is located in. This is required for subscription matching to work.
4. Click [Create] to save your changes and create the VHM.
5. On the Virtual Host Managers page, select the new VHM.
6. On the Properties page, click [Refresh Data] to inventory the new VHM.

To see which objects and resources have been inventoried, navigate to Systems › System List › Virtual Systems.

Instances running on the Amazon public cloud will report this UUID to Uyuni Server:

```
i-1234567890abcdef0
```

VHM and Azure

You can use a Uyuni VHM to gather instances from Microsoft Azure.

The VHM allows Uyuni to obtain and report information about your virtual machines. For more information on VHMs, see [Client-configuration › Vhm › ].
Create an Azure VHM

The Virtual Host Manager (VHM) runs on the Uyuni Server.

Ensure you have installed the virtual-host-gatherer-libcloud package on the Uyuni Server.

Procedure: Creating an Azure VHM

1. In the Uyuni Web UI, navigate to Systems › Virtual Host Managers.

2. Click [Create] and select Azure from the drop-down menu.

3. In the Add an Azure Virtual Host Manager section, use these parameters:
   - In the Label field, type a custom name for your VHM.
   - In the Subscription ID field, type the subscription ID provided by Azure.
   - In the Application ID field, type the application ID provided by Azure.
   - In the Tenant ID field, type the tenant ID provided by Azure.
   - In the Secret Key field, type the secret key associated with the Azure instance.
   - In the Zone field, type the zone your VM is located in. This is required for subscription matching to work.

4. Click [Create] to save your changes and create the VHM.

5. On the Virtual Host Managers page, select the new VHM.

6. On the Properties page, click [Refresh Data] to inventory the new VHM.

To see which objects and resources have been inventoried, navigate to Systems › System List › Virtual Systems.

Assigning Permissions

The VHM you create needs to have the correct permissions assigned, in order for it to access the Azure VM.

Log in to your Azure account as the subscription administrator, and ensure that the Azure user account and application are in the correct groups. The group that the application is in determines the role it has, and therefore the permissions.

If the permissions are not set correctly, you might receive an error like this when you run virtual-host-gatherer:

General error: [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. If access was recently granted, please refresh your credentials.
To determine the correct credentials, run this command at the prompt on the Uyuni Server:

```
virtual-host-gatherer -i input_azure.json -o out_azure.json -vvv
```

The `input_azure.json` file should contain this information:

```
[
  {
    "id": "azure_vhm",
    "module": "Azure",
    "subscription_id": "subscription-id",
    "application_id": "application-id",
    "tenant_id": "tenant-id",
    "secret_key": "secret-key",
    "zone": "zone"
  }
]
```

**Azure UUID**

Instances running on the Azure public cloud will report this UUID to the Uyuni Server:

```
13f56399-bd52-4150-9748-7190aae1ff21
```

**VHM and Google Compute Engine**

You can use a Uyuni VHM to gather instances from Google Compute Engine (GCE).

The VHM allows Uyuni to obtain and report information about your virtual machines. For more information on VHMs, see [Client-configuration › Vhm ›](#).

**Create a GCE VHM**

The Virtual Host Manager (VHM) runs on the Uyuni Server.

To run a VHM, your Uyuni Server will need to have port 443 open, in order to access the clients.

Ensure you have installed the `virtual-host-gatherer-libcloud` package on the Uyuni Server.

Before you begin, log in to the GCE panel, and download a certificate file. Store this file locally on your Uyuni Server, and take note of the path.

**Procedure: Creating a GCE VHM**

1. In the Uyuni Web UI, navigate to Systems › Virtual Host Managers.

2. Click [Create] and select Google Compute Engine from the drop-down menu.

3. In the Add a Google Compute Engine Virtual Host Manager section, use these
parameters:

- In the **Label** field, type a custom name for your VHM.
- In the **Service Account Email** field, type the email address associated with your Google account.
- In the **Cert Path** field, type the path to the certificate downloaded from the GCE panel.
- In the **Project ID** field, type the project ID used by the GCE instance.
- In the **Zone** field, type the zone your VM is located in. This is required for subscription matching to work.

4. Click **[Create]** to save your changes and create the VHM.

5. On the **Virtual Host Managers** page, select the new VHM.

6. On the **Properties** page, click **[Refresh Data]** to inventory the new VHM.

To see which objects and resources have been inventoried, navigate to **Systems › System List › Virtual Systems**.

**Assigning Permissions**

The VHM you create needs to have the correct permissions assigned, in order for it to access the GCE VM.

Log in to your Google Cloud Platform account as an administrator, and use the Cloud Identity and Access Management (IAM) tool to ensure that the service account has the appropriate roles. You will also need to ensure that the VM has been assigned the `VM` role.

If the permissions are not set correctly, you might receive an error like this when you run `virtual-host-gatherer`:

```
ERROR: {'domain': 'global', 'reason': 'forbidden', 'message': "Required 'compute.zones.list' permission for 'projects/project-id'"
ERROR: Could not connect to the Google Compute Engine Public Cloud using specified credentials.
```

To determine the correct credentials, run this command at the prompt on the Uyuni Server:

```
virtual-host-gatherer -i input_google.json -o out_google.json -vvv
```

The **input_google.json** file should contain this information:
GCE UUID

Instances running on the Google public cloud will report this UUID to Uyuni Server:

152986662232938449

VHM and Kubernetes

You can use a Uyuni VHM to manage Kubernetes clusters.

The VHM allows Uyuni to obtain and report information about your clusters. For more information on VHMs, see [Client-configuration › Vhm › ].

To use Uyuni with Kubernetes, you will need to have your Uyuni Server configured for container management, with all required channels present, and a registered container build host available.

You will also require:

- At least one Kubernetes or SUSE CaaS Platform cluster available on your network.
- The virtual-host-gatherer-Kubernetes package installed on the Uyuni Server.
- Kubernetes version 1.5.0 or higher, or SUSE CaaS Platform.
- Docker version 1.12 or higher on the container build host.

Create a Kubernetes VHM

Kubernetes clusters are registered with Uyuni as a VHM.

You will need a kubeconfig file to register and authorize your Kubernetes cluster. You can get a kubeconfig file using the Kubernetes command line tool kubectl. If you are using SUSE CaaS Platform, you can download the file from the Velum interface.

Procedure: Creating a Kubernetes VHM

1. In the Uyuni Web UI, navigate to Systems › Virtual Host Managers.
2. Click [Create] and select Kubernetes Cluster.
3. In the **Add a Kubernetes Virtual Host Manager** section, use these parameters:
   - In the **Label** field, type a custom name for your VHM.
   - Select the **kubeconfig** file that contains the required data for the Kubernetes cluster.
4. In the **context** field, select the appropriate context for the cluster. This is specified in the **kubeconfig** file.
5. Click **[Create]**.

**Procedure: Viewing the Nodes in a Cluster**

1. In the Uyuni Web UI, navigate to **Systems › Virtual Host Managers**.
2. Select the Kubernetes cluster.
3. Refresh the node data by clicking **[Schedule refresh data]**.

The node data can take a few moments to update. You might need to refresh your browser window to see the updated information.

Any connection or authentication problems are logged to **gatherer.log**.

Node data is not refreshed during registration. You will need to manually refresh the data to see it.

**Retrieve Image Runtime Data**

You can view runtime data about Kubernetes images in the Uyuni Web UI, by navigating to **Images › Image List**.

The image list table contains three columns:

- **Revision**: A sequence number that increments on every rebuild for images built by Uyuni, or on every import for externally built images.

- **Runtime**: Overall status of the running instances for each image in registered clusters.

- **Instances**: Number of instances running this image across all the clusters registered in Uyuni. You can see a breakdown of numbers by clicking the pop-up icon next to the number.

The **Runtime** column displays one of these status messages:
• **All instances are consistent with SUSE Manager:**

  All the running instances are running the same build of the image as tracked by Uyuni.

• **Outdated instances found:**

  Some of the instances are running an older build of the image. You might need to redeploy the image.

• **No information:**

  The checksum of the instance image does not match the image data contained in Uyuni. You might need to redeploy the image.

---

**Procedure: Building an Image**

1. In the Uyuni Web UI, navigate to **Images › Stores**.
2. Click **[Create]** to create an image store.
3. Navigate to **Images › Profiles**.
4. Click **[Create]** to create an image profile. You will need to use a dockerfile that is suitable to deploy to Kubernetes.
5. Navigate to **Images › Build** to build an image with the new profile.
6. Deploy the image into one of the registered Kubernetes clusters. You can do this with the **kubectl** tool.

The updated data should now be available in the image list at **Images › Image List**.

---

**Procedure: Importing a Previously Deployed Image**

1. In the Uyuni Web UI, navigate to **Images › Image Stores**.
2. Add the registry that owns the image you want to import, if it is not already there.
3. Navigate to **Images › Image List** and click **[Import]**.
4. Complete the fields, select the image store you created, and click **[Import]**.

The imported image should now be available in the image list at **Images › Image List**.

---

**Procedure: Rebuilding a Previously Deployed Image**

1. In the Uyuni Web UI, navigate to **Images › Image List**, locate the row that contains the image you want to rebuild, and click **[Details]**.
2. Navigate to the **Build Status** section, and click **[Rebuild]**. The rebuild can take some time to complete.

When the rebuild has successfully completed, the runtime status of the image is updated in the image list at **Images › Image List**. This shows that the instances are running a previous build of the image.
You can only rebuild images if they were originally built with Uyuni. You cannot rebuild imported images.

Procedure: Retrieving Additional Runtime Data

1. In the Uyuni Web UI, navigate to Images › Image List, locate the row that contains the running instance, and click [Details].
2. Navigate to the Overview tab. In the Image Info section, there is data in the Runtime and Instances fields.
3. Navigate to the Runtime tab. This section contains information about the Kubernetes pods running this image in all the registered clusters. The information in this section includes:
   - Pod name.
   - Namespace which the pod resides in.
   - The runtime status of the container in the specific pod.

Permissions and Certificates

You can only use kubeconfig files with Uyuni if they contain all embedded certificate data.

The API calls from Uyuni are:

- GET /api/v1/pods
- GET /api/v1/nodes

The minimum recommended permissions for Uyuni are:

- A ClusterRole to list all the nodes:

  ```
  resources: ["nodes"]
  verbs: ["list"]
  ```

- A ClusterRole to list pods in all namespaces (role binding must not restrict the namespace):

  ```
  resources: ["pods"]
  verbs: ["list"]
  ```

If /pods returns a 403 response, the entire cluster will be ignored by Uyuni.

For more information on working with RBAC Authorization, see https://kubernetes.io/docs/admin/authorization/rbac/.
VHM and SUSE CaaS Platform

You can use a Uyuni VHM to gather instances from SUSE CaaS Platform.

The VHM allows Uyuni to obtain and report information about your virtual machines. For more information on VHMs, see [Client-configuration › Vhm › ].

Onboarding CaaSP nodes

You can register each SUSE CaaS Platform node to Uyuni using the same method as you would a Salt client. For more information, see [Client-configuration › Registration-overview › ].

We recommend that you create an activation key to associate SUSE CaaS Platform channels, and to onboard the associated nodes. For more on activation keys, see [Client-configuration › Clients-and-activation-keys › ].

If you are using cloud-init, we recommended that you use a bootstrap script in the cloud-init configuration. For more on bootstrapping, see [Client-configuration › Registration-bootstrap › ].

When you have added the SUSE CaaS Platform nodes to Uyuni, the registered system will be locked automatically. When a system is locked, the Web UI shows a warning and you can schedule actions using the Web UI or the API, but the action will fail.

The locking mechanism works only with plain Salt minions (locking is not supported with salt-ssh minions).

You can enable or disable the system lock using the System Lock formula. When the system lock is disabled, all operations are permitted.

The System Lock formula is enabled automatically if SUSE CaaS Platform is detected on the node.

Updates related to Kubernetes are managed using the skuba-update tool. For more information, see https://documentation.suse.com/suse-caasp/4/html/caasp-admin/_cluster_updates.html.
When using Salt or Uyuni (either via UI or API) on any SUSE CaaS Platform nodes:

- Do not apply a patch (if the patch is marked as interactive)
- Do not mark a system to automatically install patches
- Do not perform an SP migration
- Do not reboot a node
- Do not issue any power management action via Cobbler
- Do not install a package if it breaks or conflicts the `patterns-caasp-Node-x.y`
- Do not remove a package if it breaks or conflicts or is one of the packages related with the `patterns-caasp-Node-x.y`
- Do not upgrade a package if it breaks or conflicts or is one of the packages related with the `patterns-caasp-Node-x.y`

Issuing those operations could render your SUSE CaaS Platform cluster unusable. Uyuni will not stop you from issuing these operations if the system is not locked.
Software Channels

Channels are a method of grouping software packages. In Uyuni, channels are divided into base channels and child channels. Organizing channels in this way ensures that only compatible packages are installed on each system.

A base channel consists of packages built for a specific operating system type, version, and architecture. For example, all of the packages in SUSE Linux Enterprise Server 12 for the x86_64 architecture make up a base channel. The list of packages in SUSE Linux Enterprise Server 12 for the s390x architecture make up a different base channel. A system must be subscribed to only one base channel, which is assigned automatically during registration based on the SUSE Linux Enterprise release and system architecture. For paid channels provided by a vendor, you must have an associated subscription.

A child channel is associated with a specific base channel and provides only packages that are compatible with that base channel. A system can be subscribed to multiple child channels of its base channel. When a system has been assigned to a base channel, it is only possible for that system to install the related child channels. For example, if a system has been assigned to the SUSE Linux Enterprise Server 12 x86_64 base channel, they will only be able to install or update packages compatible with SUSE Linux Enterprise Server 12 x86_64.

In the Uyuni Web UI you can browse your available channels by navigating to Software › Channel List › All. You can modify or create new channels by navigating to Software › Manage › Channels.

Custom Channels

If you require packages that are not provided by the standard Uyuni base channels, you can create custom channels. Uyuni Administrators and Channel Administrators have channel management authority, which gives them the ability to create and manage their own custom channels.

For more on creating custom channels, see [ Administration › Custom-channels › ].
Bootstrap Repository

A bootstrap repository contains packages for installing Salt on clients, as well as the required packages for registering Salt or traditional clients during bootstrapping. Bootstrap repositories are automatically created and regenerated on the Uyuni Server for every synchronized product.

Prepare to Create a Bootstrap Repository

When you select a product for synchronization, the bootstrap repository is automatically created as soon as all mandatory channels are fully mirrored.

There are two ways to check if a channel has finished synchronizing:

- In the Uyuni Web UI, navigate to Admin › Setup Wizard and select the SUSE Products tab. This dialog displays a completion bar for each product when they are being synchronized. Some products also require extensions to be synchronized.

- You can also check the synchronization log file at the command prompt. Use the `cat` or `tail -f` command to view the `/var/log/rhn/reposync/channel-label.log` file. If you use this method, remember that base channels can contain multiple child channels. Each of the child channels will generate its own log during the synchronization progress. You will need to check all the base and child channel log files to be sure that the synchronization is complete.

Options for Automatic Mode

You can change how the automated bootstrap repository creation works. This section details the various settings.

Flush Mode

By default, every regeneration starts with an empty repository and copies only the latest packages into it. To disable this behavior, add or edit this value in `/etc/rhn/rhn.conf`:

```
server.susemanager.bootstrap_repo_flush = 0
```

Automatic Mode

By default, automated regeneration of the bootstrap repositories is enabled. To disable it, add or edit this value in `/etc/rhn/rhn.conf`:

```
server.susemanager.auto_generate_bootstrap_repo = 0
```
Configure Bootstrap Data File

The tool uses a data file with information about which packages are required for each distribution. The data file is stored at `/usr/share/susemanager/mgr_bootstrap_data.py`. SUSE updates this file regularly. If you want to make changes to this file, do not edit it directly. Instead, create a copy in the same directory and edit your copy:

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

When you have made your changes, configure Uyuni to use the new file. Add or edit this value in `/etc/rhn/rhn.conf`:

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

On the next update, the new data from SUSE will overwrite the original data file, not the new one. You will need to keep the new file up to date with changes provided by SUSE.

Manually Generate a Bootstrap Repository

By default, bootstrap repositories are regenerated daily. You can manually create the bootstrap repository from the command prompt:

**Procedure: Generating the Bootstrap Repository for SUSE Linux Enterprise**

1. At the command prompt on the Uyuni Server, as root, list the available bootstrap repositories:

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

2. Create the bootstrap repository, using the appropriate repository name as the product label:

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

The client repository is located in `/srv/www/htdocs/pub/repositories/`.

**Procedure: Specify a Bootstrap Repository**

If you have mirrored more than one SUSE Linux Enterprise 15 Product (for example, SLES and SLES for SAP), you can specify the one you are actually interested in.

1. Check what bootstrap repositories you have available:
2. Specify the appropriate repository:

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

There are a number of ways that the Uyuni Server can communicate with clients. Which one you use depends on your network architecture.

The Uyuni daemon (rhnsd) runs on traditional client systems and periodically connects with Uyuni to check for new updates and notifications. It does not apply to Salt clients.

Push via SSH and Push via Salt SSH are used in environments where clients cannot reach the Uyuni Server directly. In this environment, clients are located in a firewall-protected zone called a DMZ. No system within the DMZ is authorized to open a connection to the internal network, including the Uyuni Server.

OSAD is an alternative contact method between Uyuni and its clients. OSAD allows registered client systems to execute scheduled actions immediately.

SUSE Manager Daemon (rhnsd)

The Uyuni daemon (rhnsd) runs on traditional client systems and periodically connects with Uyuni to check for new updates and notifications. It does not apply to Salt clients.

It is only used on SUSE Linux Enterprise 11 and Red Hat Enterprise Linux Server 6, as these systems do not use systemd. On later operating systems, a systemd timer (rhnsd.timer) is used and controlled by rhnsd.service.

Start the daemon with /etc/init.d/rhnsd.

By default, it will check every four hours for new actions. This means it can take some time for clients to execute scheduled actions.

To check for updates, rhnsd runs the external mgr_check program located in /usr/sbin/. This is a small application that establishes the network connection to Uyuni. The SUSE Manager daemon does not listen on any network ports or talk to the network directly. All network activity is performed by the mgr_check utility.

This figure provides an overview of the default rhnsd process path. All items left of the Python XMLRPC server block represent processes running on a Uyuni client.
Configure rhnsd

The `rhnsd` initialization script has a configuration file on the client system at `/etc/sysconfig/rhn/rhnsd`.

An important parameter for the daemon is its check-in frequency. The default interval time is four hours (240 minutes). The minimum allowed time interval is one hour (60 minutes). If you set the interval below one hour, it will change back to the default of 4 hours (240 minutes).

If you modify the `rhnsd` configuration file, execute this command as root to restart the daemon and pick up your changes:

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

To see the status of `rhnsd`, use this command as root:

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

On SUSE Linux Enterprise 12 and later, the default time interval is set in `/etc/systemd/system/timers.target.wants/rhnsd.timer`, in this section:

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

You can create an overriding drop-in file for `rhnsd.timer` using `systemctl`:
systemctl edit rhnsd.timer

For example, if you want to configure a two-hour time interval:

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

The file will be saved as `/etc/systemd/system/rhnsd.timer.d/override.conf`.

For more information about system timers, see the `systemd.timer` and `systemctl` manpages.

**Push via SSH**

Push via SSH is used in environments where traditional clients cannot reach the Uyuni Server directly. In this environment, clients are located in a firewall-protected zone called a DMZ. No system within the DMZ is authorized to open a connection to the internal network, including the Uyuni Server.

The Push via SSH method creates an encrypted tunnel from the Uyuni Server on the internal network to the clients located on the DMZ. After all actions and events are executed, the tunnel is closed.

The server uses SSH to contact the clients at regular intervals, checking in and performing scheduled actions and events.

This contact method works for traditional clients only. For Salt clients, use Push via Salt SSH.

Re-installing systems using the provisioning model is not currently supported on clients managed with push via SSH.

This image demonstrates the push via SSH process path. All items left of the Taskomatic block represent processes running on a Uyuni client.
For tunneling connections via SSH, two available port numbers are required, one for tunneling HTTP and the second for tunneling via HTTPS (HTTP is only necessary during the registration process). The port numbers used by default are 1232 and 1233. To overwrite these, you can add two custom port numbers greater than 1024 to /etc/rhn/rhn.conf:

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

If you would like your clients to be contacted using their hostnames instead of an IP address, set this option:

```
ssh_push_use_hostname = true
```

It is also possible to adjust the number of threads to use for opening client connections in parallel. By default, two parallel threads are used. Set `taskomatic.ssh_push_workers` in /etc/rhn/rhn.conf:

```
taskomatic.ssh_push_workers = number
```

For security reasons, you might want to use sudo with SSH, to access the system as an unprivileged user instead of as root.

*Procedure: Configuring Unprivileged SSH Access*

1. Ensure you have the latest `spacewalk-taskomatic` and `spacewalk-certs-tools` packages installed on the Uyuni Server.
2. On each client system, create an appropriate unprivileged user.
3. On each client system, open the `/etc/sudoers` file and comment out these lines:

```
#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. On each client system, in the **User privilege specification** section, add these lines:

```
<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. On each client system, in the `/home/user/.bashrc` file, add these lines:

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

6. On the Uyuni Server, in the `/etc/rhn/rhn.conf` configuration file, add or amend this line to include the unprivileged username:

```
ssh_push_sudo_user = <user>
```

Because clients are in the DMZ and cannot reach the server, you need to use the `mgr-ssh-push-init` tool to register them with the Uyuni Server.

To use the tool, you will need the client hostname or IP address, and the path to a valid bootstrap script on the Uyuni Server. For more information about bootstrapping, see [Client-configuration › Registration-bootstrap ›](#).

The bootstrap script will need to have an activation key associated with it that is configured for Push via SSH. For more information on activation keys, see [Client-configuration › Clients-and-activation-keys ›](#).

Before you begin, you need to ensure that you have specified which ports to use for SSH tunneling. If you have registered clients before changing the port numbers, they will need to be registered again.

![Info]

Clients that are managed with Push via SSH cannot reach the server directly. When you use the `mgr-ssh-push-init` tool, the `rhnsd` daemon is disabled.

**Procedure: Registering Clients with Push via SSH**

1. At the command prompt on the Uyuni Server, as root, execute this command:

```
# mgr-ssh-push-init --client <client> --register \
/srv/www/htdocs/pub/bootstrap/bootstrap_script --tunnel
```
OPTIONAL: You can remove the `--tunnel` option, if you do not want to use tunneling.

2. Verify that the SSH connection is active:

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

**Example: API Access to Push via SSH**

You can use the API to manage which contact method to use. This example Python code sets the contact method to `ssh-push`.

Valid values are:

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

```python
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'})
```

If you have a client that has already been registered, and you want to migrate it to use Push via SSH, some extra steps are required. You can use the `mgr-ssh-push-init` tool to set up your client.

**Procedure: Migrating Registered Systems to Push via SSH**

1. At the command prompt on the Uyuni Server, as root, set up the client:

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

2. Using the Uyuni Web UI, change the client’s contact method to `ssh-push` or `ssh-push-tunnel`.

3. OPTIONAL: If you need to edit an existing activation key, you can do so with this command:

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

You can also use Push via SSH for clients that connect using a Uyuni Proxy. Ensure your proxy is updated before you begin.

**Procedure: Registering Clients with Push via SSH to a Proxy**

1. At the command prompt on the Uyuni Proxy, as root, set up the client:
Push via Salt SSH

Push via Salt SSH is used in environments where Salt clients cannot reach the Uyuni Server directly. In this environment, clients are located in a firewall-protected zone called a DMZ. No system within the DMZ is authorized to open a connection to the internal network, including the Uyuni Server.

The Push via Salt SSH method creates an encrypted tunnel from the Uyuni Server on the internal network to the clients located on the DMZ. After all actions and events are executed, the tunnel is closed.

The server uses the `salt-ssh` tool to contact the clients at regular intervals, checking in and performing scheduled actions and events. For more information about Salt SSH, see [Salt › Salt-ssh ›](#).

This contact method works for Salt clients only. For traditional clients, use Push via SSH.

This image demonstrates the Push via Salt SSH process path. All items left of the Taskomatic block represent processes running on a Uyuni client.

To use Push via Salt SSH, you must have the SSH daemon running on the client, and reachable by the `salt-api` daemon running on the Uyuni Server. Additionally, Python must be available on the remote system, and be a version supported by Salt.
Red Hat Enterprise Linux 5, CentOS 5, and earlier are not supported, as they use unsupported versions of Python.

**Procedure: Registering Clients with Push via Salt SSH**

1. In the Uyuni Web UI, navigate to **Systems › Bootstrapping** and complete the appropriate fields.

2. Select an activation key with the Push via SSH contact method configured. For more information about activation keys, see [Client-configuration › Clients-and-activation-keys › ].

3. Check the Manage system completely via SSH checkbox.

4. Click [Bootstrap] to begin registration.

5. Confirm that the system has been registered correctly by navigating to **Systems › Overview**.

When you are configuring Push via Salt SSH, you can modify parameters that are used when a system is registered, including the host, activation key, and password. The password is used only for bootstrapping, it is not saved anywhere. All future SSH sessions are authorized via a key/certificate pair. These parameters are configured in **Systems › Bootstrapping**.

You can also configure persistent parameters that are used system-wide, including the sudo user. For more information on configuring the sudo user, see the Push via SSH section in this chapter.

The Push via Salt SSH feature uses taskomatic to execute scheduled actions using `salt-ssh`. The taskomatic job periodically checks for scheduled actions and executes them. Unlike Push via SSH on traditional clients, the Push via Salt SSH feature executes a complete `salt-ssh` call based on the scheduled action.

There are some features that are not yet supported on Push via Salt SSH. These features will not work on Salt SSH clients:

- OpenSCAP auditing
- Beacons, resulting in:
  - Installing a package on a system using `zypper` will not invoke the package refresh.
  - Virtual Host functions (for example, a host to guests) will not work if the virtual host system is Salt SSH-based.

For more information about Salt SSH, see [https://docs.saltstack.com/en/latest/topics/ssh/](https://docs.saltstack.com/en/latest/topics/ssh/).

**OSAD**

OSAD is an alternative contact method between Uyuni and its clients. By default, Uyuni uses `rhnsd`, which contacts the server every four hours to execute scheduled actions. OSAD allows registered client systems to execute scheduled actions immediately.
OSAD has several distinct components:

- The **osa-dispatcher** service runs on the server, and uses database checks to determine if clients need to be pinged, or if actions need to be executed.
- The **osad** service runs on the client. It responds to pings from **osa-dispatcher** and runs **mgr_check** to execute actions when directed to do so.
- The **jabberd** service is a daemon that uses the **XMPP** protocol for communication between the client and the server. The **jabberd** service also handles authentication.
- The **mgr_check** tool runs on the client to execute actions. It is triggered by communication from the **osa-dispatcher** service.

The **osa-dispatcher** periodically runs a query to check when clients last showed network activity. If it finds a client that has not shown activity recently, it will use **jabberd** to ping all **osad** instances running on all clients registered with your Uyuni server. The **osad** instances respond to the ping using **jabberd**, which is running in the background on the server. When the **osa-dispatcher** receives the response, it marks the client as online. If the **osa-dispatcher** fails to receive a response within a certain period of time, it marks the client as offline.

When you schedule actions on an OSAD-enabled system, the task will be carried out immediately. The **osa-dispatcher** periodically checks clients for actions that need to be executed. If an outstanding action is found, it uses **jabberd** to execute **mgr_check** on the client, which will then execute the action.

OSAD clients use the fully qualified domain name (FQDN) of the server to communicate with the **osa-dispatcher** service.

SSL is required for **osad** communication. If SSL certificates are not available, the daemon on your client systems will fail to connect. Make sure your firewall rules are set to allow the required ports. For more information, see [tab.install.ports.server].

### Procedure: Enabling OSAD

1. At the command prompt on the Uyuni Server, as root, start the **osa-dispatcher** service:

   ```bash
   systemctl start osa-dispatcher
   ```

2. On each client, install the **mgr-osad** package from the **Tools** child channel. The **mgr-osad** package should be installed on clients only. If you install the **mgr-osad** package on your Uyuni Server, it will conflict with the **osa-dispatcher** package.

3. On each client, as root, start the **osad** service:
Because `osad` and `osa-dispatcher` are run as services, you can use standard commands to manage them, including `stop`, `restart`, and `status`.

Each OSAD component is configured using local configuration files. We recommend you keep the default configuration parameters for all OSAD components.

<table>
<thead>
<tr>
<th>Component</th>
<th>Location</th>
<th>Path to Configuration File</th>
</tr>
</thead>
<tbody>
<tr>
<td>osa-dispatcher</td>
<td>Server</td>
<td><code>/etc/rhn/rhn.conf</code> Section: OSA configuration</td>
</tr>
<tr>
<td>osad</td>
<td>Client</td>
<td><code>/etc/sysconfig/rhn/osad.conf</code></td>
</tr>
<tr>
<td>osad log file</td>
<td>Client</td>
<td><code>/var/log/osad</code></td>
</tr>
<tr>
<td>jabberd log file</td>
<td>Both</td>
<td><code>/var/log/messages</code></td>
</tr>
</tbody>
</table>

**Troubleshooting OSAD**

If your OSAD clients cannot connect to the server, or if the `jabberd` service takes a lot of time responding to port 5552, it could be because you have exceeded the open file count.

Every client needs one always-open TCP connection to the server, which consumes a single file handler. If the number of file handlers currently open exceeds the maximum number of files that `jabberd` is allowed to use, `jabberd` will queue the requests, and refuse connections.

To resolve this issue, you can increase the file limits for `jabberd` by editing the `/etc/security/limits.conf` configuration file and adding these lines:

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

Calculate the limits required for your environment by adding 100 to the number of clients for the soft limit, and 1000 to the current number of clients for the hard limit.

In the example above, we have assumed 500 current clients, so the soft limit is 5100, and the hard limit is 6000.

You will also need to update the `max_fds` parameter in the `/etc/jabberd/c2s.xml` file with your chosen hard limit:

```
<max_fds>6000</max_fds>
```
Using the System Set Manager

System Set Manager (SSM) is used to administrate groups of systems, rather than performing actions on one system at a time. It works for both Salt and traditional clients.

For a complete list of the tasks that you can perform with the SSM, see [Reference › Systems › ].

Setting up System Set Manager

You need to select which systems or system group you want to work with before you can use SSM to perform operations.

You can access SSM in three different ways:

- Navigate to Systems › System List, select systems you want to work with, and navigate to Systems › System Set Manager.
- Navigate to Systems › System Groups, and click [Use in SSM] for the system group you want to work with.
- Navigate to Systems › System Groups, select the group you want to work with, and click [Work with Group].

Using System Set Manager

The details you see in SSM might differ slightly from the details available in other parts of the Uyuni Web UI. If you are looking at the details of a single system in the Web UI, then you will only be able to see the latest available versions of package updates. When you look at package updates in SSM, all available versions will be shown. This is intended to make it easier for system administrators to manage package versions, and choose to upgrade to packages that might not be the latest version.
Troubleshooting Clients

Bare Metal Systems

If a bare metal system on the network is not automatically added to the Systems list, check these things first:

- You must have the pxe-default-image package installed.
- File paths and parameters must be configured correctly. Check that the vmlinuz0 and initrd0.img files, which are provided by pxe-default-image, are in the locations specified in the rhn.conf configuration file.
- Ensure the networking equipment connecting the bare metal system to the Uyuni server is working correctly, and that you can reach the Uyuni server IP address from the server.
- The bare metal system to be provisioned must have PXE booting enabled in the boot sequence, and must not be attempting to boot an operating system.
- The DHCP server must be responding to DHCP requests during boot. Check the PXE boot messages to ensure that:
  - the DHCP server is assigning the expected IP address
  - the DHCP server is assigning the the Uyuni server IP address as next-server for booting.
- Ensure Cobbler is running, and that the Discovery feature is enabled.

If you see a blue Cobbler menu shortly after booting, discovery has started. If it does not complete successfully, temporarily disable automatic shutdown in order to help diagnose the problem. To disable automatic shutdown:

1. Select pxe-default-profile in the Cobbler menu with the arrow keys, and press the Tab key before the timer expires.
2. Add the kernel boot parameter spacewalk-finally=running using the integrated editor, and press Enter to continue booting.
3. Enter a shell with the username root and password linux to continue debugging.

Duplicate profiles

Due to a technical limitation, it is not possible to reliably distinguish a new bare metal system from a system that has previously been discovered. Therefore, we recommended that you do not power on bare metal systems multiple times, as this will result in duplicate profiles.

Cloned Salt Clients

If you have used your hypervisor clone utility, and attempted to register the cloned Salt client, you might get this error:
We're sorry, but the system could not be found.

This is caused by the new, cloned, system having the same machine ID as an existing, registered, system. You can adjust this manually to correct the error and register the cloned system successfully.

For more information and instructions, see [Administration › Tshoot-registerclones › ].

Disabling the FQDNS grain

The FQDNS grain returns the list of all the fully-qualified DNS services in the system. Collecting this information is usually a fast process, but if the DNS settings have been misconfigured, it could take a much longer time. In some cases, the client could become unresponsive, or crash.

To prevent this problem, you can disable the FQDNS grain with a Salt flag. If you disable the grain, you can use a network module to provide FQDNS services, without the risk of the client becoming unresponsive.

This only applies to older Salt clients. If you registered your Salt client recently, the FQDNS grain is disabled by default.

On the Uyuni Server, at the command prompt, use this command to disable the FQDNS grain:

```
salt '*' state.sls util.mgr_disable_fqdns_grain
```

This command restarts each client and generate Salt events that the server needs to process. If you have a large number of clients, you can execute the command in batch mode instead:

```
salt --batch-size 50 '*' state.sls util.mgr_disable_fqdns_grain
```

Wait for the batch command to finish executing. Do not interrupt the process with Ctrl+C.

Mounting /tmp with noexec

Salt runs remote commands from /tmp on the client’s filesystem. Therefore you must not mount /tmp with the noexec option.

Passing Grains to a Start Event

Every time a Salt client starts, it passes the machine_id grain to Uyuni. Uyuni uses this grain to determine if the client is registered. This process requires a synchronous Salt call. Synchronous Salt calls block other processes, so if you have a lot of clients start at the same time, the process could create significant delays.
To overcome this problem, a new feature has been introduced in Salt to avoid making a separate synchronous Salt call.

To use this feature, you can add a configuration parameter to the client configuration, on clients that support it.

To make this process easier, you can use the `mgr_start_event_grains.sls` helper Salt state.

This only applies to already registered clients. If you registered your Salt client recently, this config parameter is added by default.

On the Uyuni Server, at the command prompt, use this command to enable the `start_event_grains` configuration helper:

```
salt '*' state.sls util.mgr_start_event_grains
```

This command adds the required configuration into the client’s configuration file, and applies it when the client is restarted. If you have a large number of clients, you can execute the command in batch mode instead:

```
salt --batch-size 50 '*' state.sls mgr_start_event_grains
```

**Proxy Connections and FQDN**

Sometimes clients connected through a Uyuni Proxy appear in the Web UI, but do not show that they are connected through a proxy. This can occur if you are not using the fully-qualified domain name (FQDN) to connect, and the proxy is not known to Uyuni.

To correct this behavior, specify additional FQDNs as grains in the client configuration file on the proxy:

```
grains:
  susemanager:
    custom_fqdns:
    - name.one
    - name.two
```
AutoYaST Example File

Minimalist AutoYaST Profile for Automated Installations and Useful Enhancements

The AutoYaST profile in this section installs a SUSE Linux Enterprise Server system with all default installation options including a default network configuration using DHCP. After the installation is finished, a bootstrap script located on the Uyuni server is executed in order to register the freshly installed system with Uyuni. You need to adjust the IP address of the Uyuni server, the name of the bootstrap script, and the root password according to your environment:

```xml
<user>
  ...
  <username>root</username>
  <user_password>'linux'</user_password>
  </user>

<location>http://'192.168.1.1'/pub/bootstrap/'my_bootstrap.sh'</location>
```

The complete AutoYaST file:
Use this enhancement fragment to add child channels:
Replace \texttt{channel-label} and \texttt{distribution-label} with the correct labels (such as \texttt{sles12-sp4-updates-x86\_64} and \texttt{sles12-sp4-x86\_64}). Ensure that the distribution label corresponds to the Autoinstallable Distribution. Set the variables (such as \texttt{\$c\_server}) according to your environment. For more information about variables, see [Reference › Systems ›].

Here is a literal example for \texttt{sles12-sp4-x86\_64}:

```xml
<add-on>
  <add_on_products config:type="list">
    <listentry>
      <!-- SLES12 Updates -->
      <media_url>http://192.168.150.10/ks/dist/child/dev-sles12-sp4-updates-x86_64/dev-sles12sp4</media_url>
      <product>SLES 12 Updates</product>
      <product_dir/></product_dir>
      <name>SLES12 Updates</name>
    </listentry>
    <listentry>
      <!-- SLES12 SUSE Manager Tools Pool -->
      <product>SLES 12 Pool SUSE Manager Tools</product>
      <product_dir/></product_dir>
      <name>SLES12 Pool SUSE Manager Tools</name>
    </listentry>
    <listentry>
      <!-- SLES12 SUSE Manager Tools Updates -->
      <product>SLES 12 Updates SUSE Manager Tools</product>
      <product_dir/></product_dir>
      <name>SLES12 Updates SUSE Manager Tools</name>
    </listentry>
  </add_on_products>
</add-on>
```
Add the Updates Channel

It is required that you add the updates tools channel to the `<add-on>` AutoYaST snippet section. This ensures your systems are provided with an up-to-date version of the `libzypp` package. If you do not include the updates tools channel, you will encounter 400 errors. In this example, the `(DISTRIBUTION_NAME)` is replaced with the name of the autoinstallation distribution from Systems › Autoinstallation › Distributions.

```
<listentry>
  <ask_on_error config:type="boolean">true</ask_on_error>
  <media_url>http://$redhat_management_server/ks/dist/child/sles12-sp2-updates-x86_64/(DISTRIBUTION_NAME)</media_url>
  <name>sles12 sp2 updates</name>
  <product>SLES12</product>
  <product_dir></product_dir>
</listentry>
```