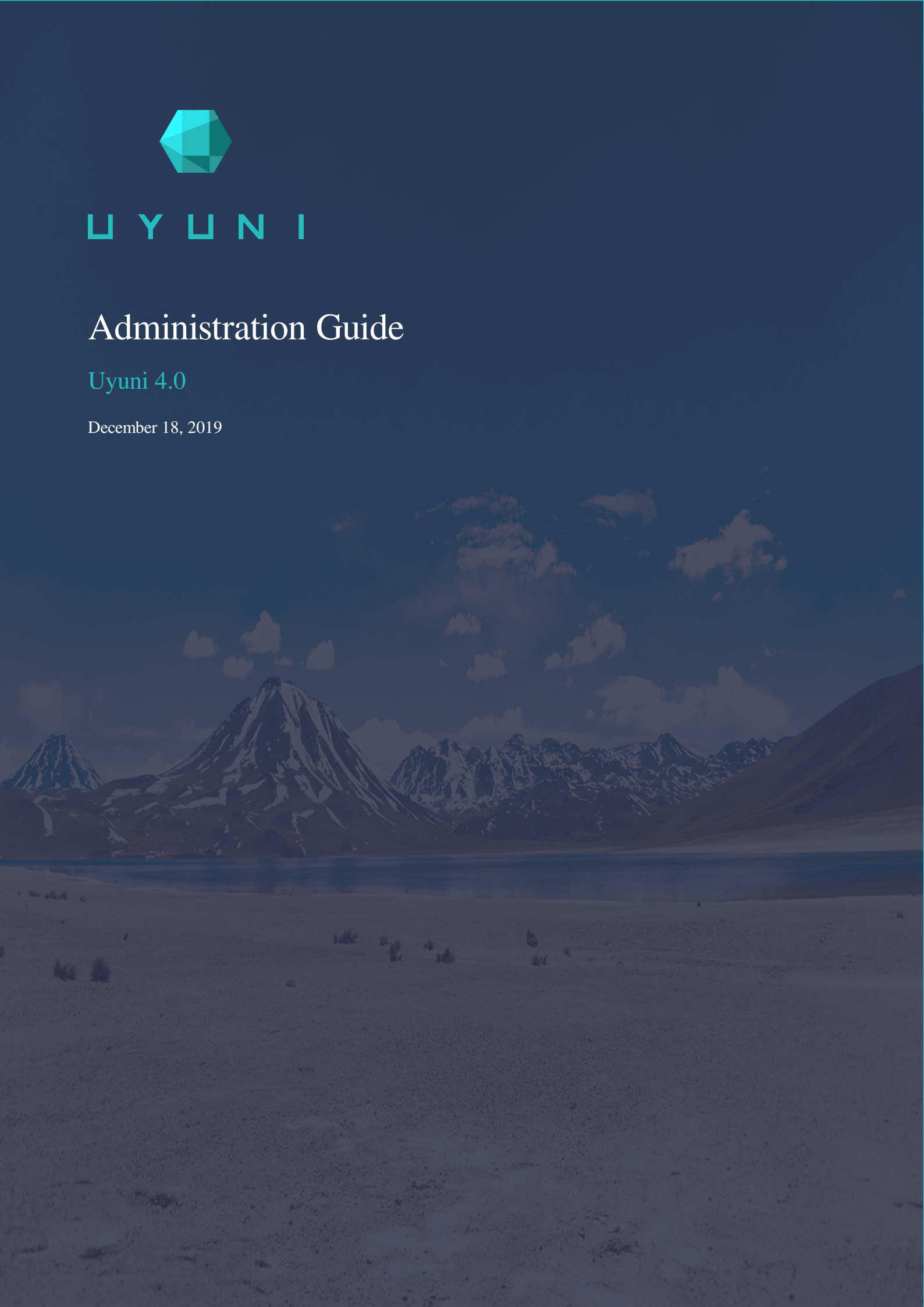


Administration Guide

Uyuni 4.0

December 18, 2019

Table of Contents

GNU Free Documentation License

Copyright © 2000, 2001, 2002 Free Software Foundation, Inc. 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA. Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document’s overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections

then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work’s title, preceding the beginning of the body of the text.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document’s license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version’s license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

```
Copyright (c) YEAR YOUR NAME.
   Permission is granted to copy, distribute and/or modify this document
   under the terms of the GNU Free Documentation License, Version 1.2
   or any later version published by the Free Software Foundation;
   with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.
   A copy of the license is included in the section entitled{ldquo}GNU
   Free Documentation License{rdquo}.
```

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the “ with…Texts.” line with this:

```
with the Invariant Sections being LIST THEIR TITLES, with the
   Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.
```

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Introduction

Publication Date: 2019-12-18

This book provides guidance on performing common administrative tasks on Uyuni.

Image Building and Management

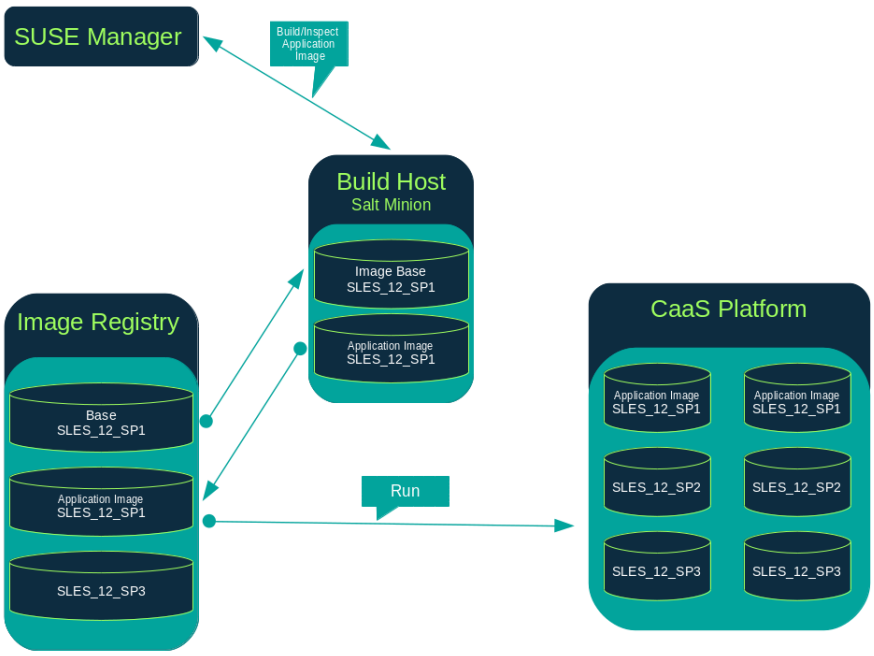
Image Building Overview

Uyuni enables system administrators to build containers and OS Images and push the result in image stores. The workflow looks like this:

1. Define an image store
2. Define an image profile and associate it with a source (either a git repository or a directory)
3. Build the image
4. Push the image to the image store

Uyuni supports two distinct build types: dockerfile, and the Kiwi image system.

Container Images



Requirements

The containers feature is available for Salt clients running SUSE Linux Enterprise Server 12 or later. Before you begin, ensure your environment meets these requirements:

- A published git repository repository containing a dockerfile and configuration scripts. The repository can be public or private, and should be hosted on GitHub, GitLab, or BitBucket.
- A properly configured image store, such as a Docker registry.

If you require a private image registry you can use an open source solution such as `Portus`. For additional information on setting up Portus as a registry provider, see the Portus Documentation.

For more information on Containers or SUSE CaaS Platform, see:

- https://documentation.suse.com/sles/15-SP1/html/SLES-all/book-sles-docker.html
- https://documentation.suse.com/suse-caasp/4/

Create a Build Host

To build images with Uyuni, you will need to create and configure a build host. Container build hosts are Salt clients running SUSE Linux Enterprise 12 or later. This section guides you through the initial configuration for a build host.

From the Uyuni Web UI, perform these steps to configure a build host:

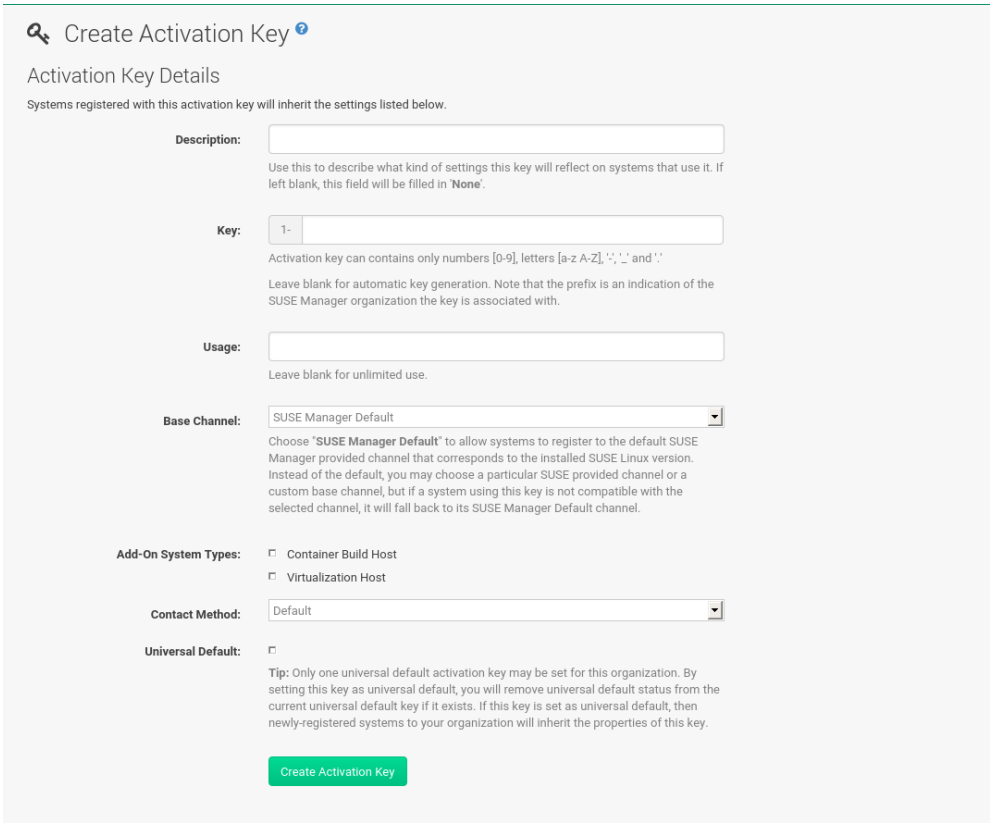
1. Select a Salt client to be designated as a build host from the **Systems › Overview** page.
2. From the `System Details` page of the selected client assign the containers modules. Go to **Software › Software Channels** and enable the containers module (for example, `SLE-Module-Containers15-Pool` and `SLE-Module-Containers15-Updates`). Confirm by clicking **`[Change Subscriptions]`**.
3. From the **System Details › Properties** page, enable `Container Build Host` from the `Add-on System Types` list and confirm by clicking **`[Update Properties]`**.
4. Install all required packages by applying `Highstate`. From the system details page select **States › Highstate** and click `Apply Highstate`. Alternatively, apply Highstate from the Uyuni Server command line:

```
salt '$your_client' state.highstate
```

Create an Activation Key for Containers

The containers built using Uyuni will use channel(s) associated to the activation key as repositories when building the image. This section will guide you into creating an ad-hoc activation key for this purpose.

To build a container, you will need an activation key that is associated with a channel other than `SUSE Manager Default`.

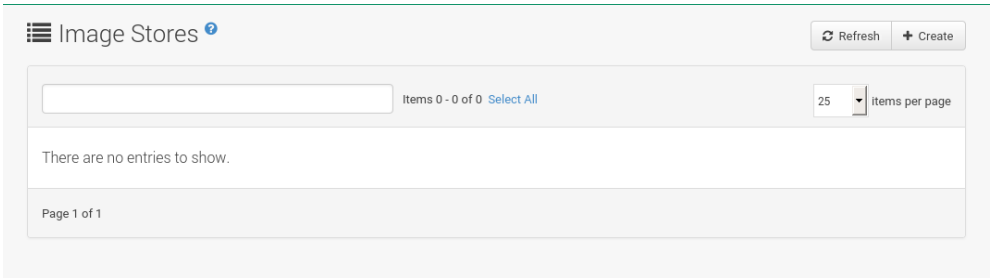


1. Select **Systems › Activation Keys**.
2. Click **`[Create Key]`**.
3. Enter a `Description` and a `Key` name. Use the drop-down menu to select the `Base Channel` to associate with this key.
4. Confirm with **`[Create Activation Key]`**.

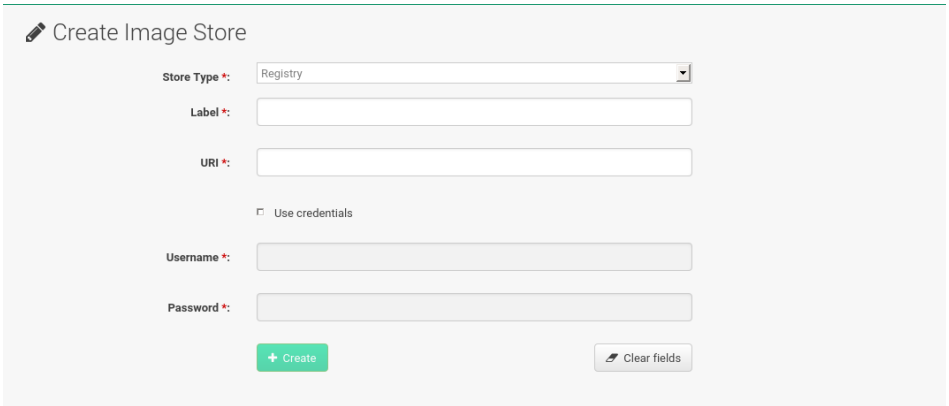
For more information, see [bp.key.managment].

Create an Image Store

All built images are pushed to an image store. This section contains information about creating an image store.



1. Select **Images › Stores**.
2. Click `Create` to create a new store.



Define a name for the image store in the `Label` field. . Provide the path to your image registry by filling in the `URI` field, as a fully qualified domain name (FQDN) for the container registry host (whether internal or external).

+

```
registry.example.com
```

+

The Registry URI can also be used to specify an image store on a registry that is already in use.

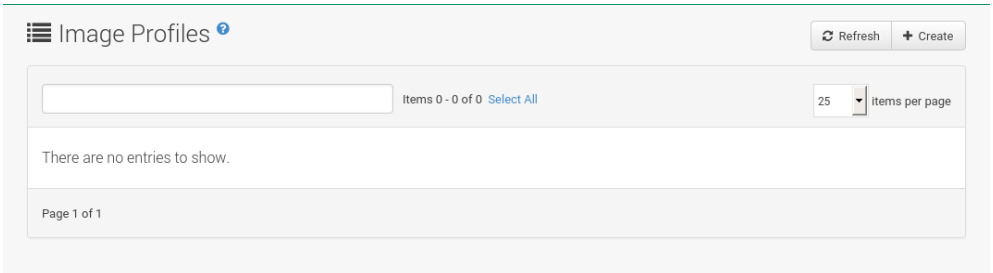
+

```
registry.example.com:5000/myregistry/myproject
```

1. Click **`[Create]`** to add the new image store.

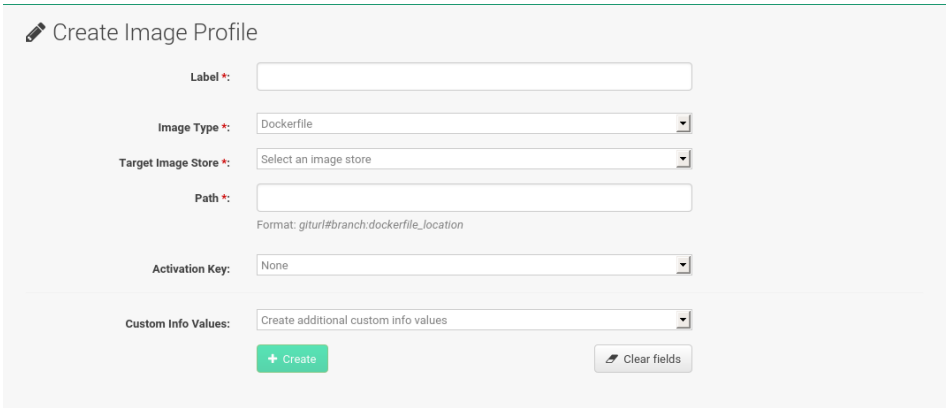
Create an Image Profile

All container images are built using an image profile, which contains the building instructions. This section contains information about creating an image profile with the Uyuni Web UI.



Procedure: Create an Image Profile

1. To create an image profile select **Images › Profiles** and click **`[Create]`**.



2. Provide a name for the image profile by filling in the `Label` field.

Only lowercase characters are permitted in container labels. If your container image tag is in a format such as `myproject/myimage`, make sure your image store registry URI contains the `/myproject` suffix.

3. Use a dockerfile as the `Image Type`.
4. Use the drop-down menu to select your registry from the `Target Image Store` field.
5. In the `Path` field, type a GitHub, GitLab or BitBucket repository URL. The URL should be be http, https, or a token authentication URL. Use one of these formats:

GitHub Path Options

- GitHub single user project repository

```
https://github.com/USER/project.git#branchname:folder
```

- GitHub organization project repository

```
https://github.com/ORG/project.git#branchname:folder
```

- GitHub token authentication

  If your git repository is private, modify the profile's URL to include authentication. Use this URL format to authenticate with a GitHub token:

```
https://USER:<AUTHENTICATION_TOKEN>@github.com/USER/project.git#master:/container/
```

GitLab Path Options

- GitLab single user project repository

```
https://gitlab.example.com/USER/project.git#master:/container/
```

- GitLab groups project repository

```
https://gitlab.example.com/GROUP/project.git#master:/container/
```

- GitLab token authentication

  If your git repository is private and not publicly accessible, you need to modify the profile's git URL to include authentication. Use this URL format to authenticate with a GitLab token:

```
https://gitlab-ci-
token:<AUTHENTICATION_TOKEN>@gitlab.example.com/USER/project.git#master:/container/
```

Specifying a git branch

If a branch is not specified, the `master` branch will be used by default. If a `folder` is not specified the image sources (dockerfile sources) are expected to be in the root directory of the GitHub or GitLab checkout.

1. Select an `Activation Key`. Activation keys ensure that images using a profile are assigned to the correct channel and packages.

Relationship Between Activation Keys and Image Profiles

When you associate an activation key with an image profile you are ensuring any image using the profile will use the correct software channel and any packages in the channel.

2. Click the **[Create]** button.

Example Dockerfile Sources

An Image Profile that can be reused is published at https://github.com/SUSE/manager-build-profiles

The `ARG` parameters ensure that the built image is associated with the desired repository version served by Uyuni. The `ARG` parameters also allow you to build image versions of SUSE Linux Enterprise Server which may differ from the version of SUSE Linux Enterprise Server used by the build host itself.

For example: The `ARG repo` parameter and the `echo` command pointing to the repository file, creates and then injects the correct path into the repository file for the desired channel version.

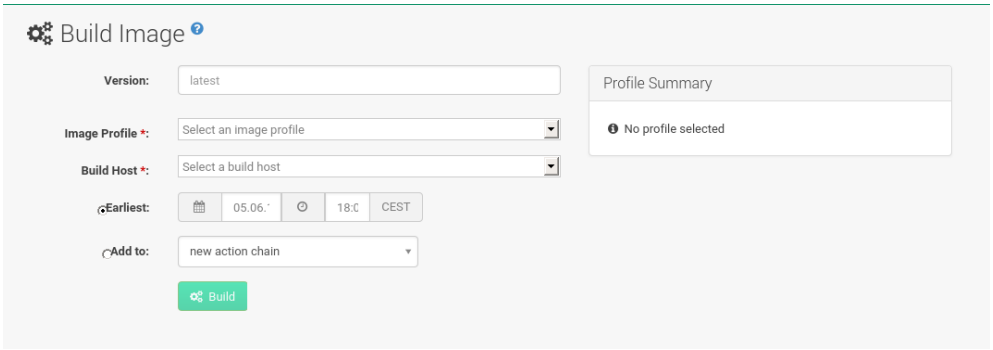
The repository version is determined by the activation key that you assigned to your image profile.

The python and python-xml packages must be installed in the container. They are required for inspecting images, and for providing the package and product list of a container to the Uyuni Web UI. If you do not install them, images will still build but the package and product list will not available in the Web UI.

```
FROM registry.example.com/sles12sp2
MAINTAINER Tux Administrator "tux@example.com"

### Begin: These lines Required for use with {productname}

ARG repo
ARG cert

# Add the correct certificate
RUN echo "$cert" > /etc/pki/trust/anchors/RHN-ORG-TRUSTED-SSL-CERT.pem

# Update certificate trust store
RUN update-ca-certificates

# Add the repository path to the image
RUN echo "$repo" > /etc/zypp/repos.d/susemanager:dockerbuild.repo

### End: These lines required for use with {productname}

# Add the package script
ADD add_packages.sh /root/add_packages.sh

# Run the package script
RUN /root/add_packages.sh

# After building remove the repository path from image
RUN rm -f /etc/zypp/repos.d/susemanager:dockerbuild.repo
```

Build an Image

There are two ways to build an image. You can select **Images › Build** from the left navigation bar, or click the build icon in the **Images › Profiles** list.



Procedure: Building an Image

1. Select **Images › Build**.
2. Add a different tag name if you want a version other than the default `latest` (only relevant to containers).

3. Select `Build Profile` and `Build Host`.

Profile Summary

Notice the `Profile Summary` to the right of the build fields. When you have selected a build profile, detailed information about the selected profile will be displayed in this area.

4. To schedule a build click the **`[Build]`** button.

Import an Image

You can import and inspect arbitrary images. Select **Images › Image List** from the left navigation bar. Complete the text boxes of the `Import` dialog. When it has processed, the imported image will be listed on the `Image List` page.

Procedure: Importing an Image

1. From **Images › Image list** click **`[Import]`** to open the `Import Image` dialog.
2. In the `Import Image` dialog complete these fields:

Image store

The registry from where the image will be pulled for inspection.

Image name

The name of the image in the registry.

Image version

The version of the image in the registry.

Build host

The build host that will pull and inspect the image.

Activation key

The activation key that provides the path to the software channel that the image will be inspected with.

3. For confirmation, click **`[Import]`**.

The entry for the image is created in the database, and an `Inspect Image` action on Uyuni is scheduled.

When it has been processed, you can find the imported image in the `Image List`. It has a different icon in the `Build` column, to indicate that the image is imported. The status icon for the imported image can also be seen on the `Overview` tab for the image.

Troubleshooting

These are some known problems when working with images:

- HTTPS certificates to access the registry or the git repositories should be deployed to the client by a custom state file.
- SSH git access using Docker is currently unsupported.
- If the python and python-xml packages are not installed in your images during the build process, reporting of installed packages or products will fail. This will result in an `unknown` update status.

OS Images

OS Images are built by the Kiwi image system. The output image is customizable and can be PXE, QCOW2, LiveCD, or other types of images.

For more information about the Kiwi build system, see the Kiwi documentation.

Requirements

The Kiwi image building feature is available for Salt clients running SUSE Linux Enterprise Server 12 and SUSE Linux Enterprise Server 11.

Kiwi image configuration files and configuration scripts must be accessible in one of these locations:

- Git repository
- HTTP hosted tarball
- Local build host directory

For an example of a complete Kiwi repository served by git, see https://github.com/SUSE/manager-build-profiles/tree/master/OSImage

Hardware Requirements for Hosts Running OS Images

Hosts running OS Images built with Kiwi need at least 1 GB of RAM. Disk space depends on the actual size of the image. For more information, see the documentation of the underlying system.

The build host must be a Salt client. Do not install the build host as a traditional client.

Create a Build Host

To build all kinds of images with Uyuni, create and configure a build host. OS Image build hosts are Salt clients running on SUSE Linux Enterprise Server 12 (SP3 or later) or SUSE Linux Enterprise Server 11 SP4.

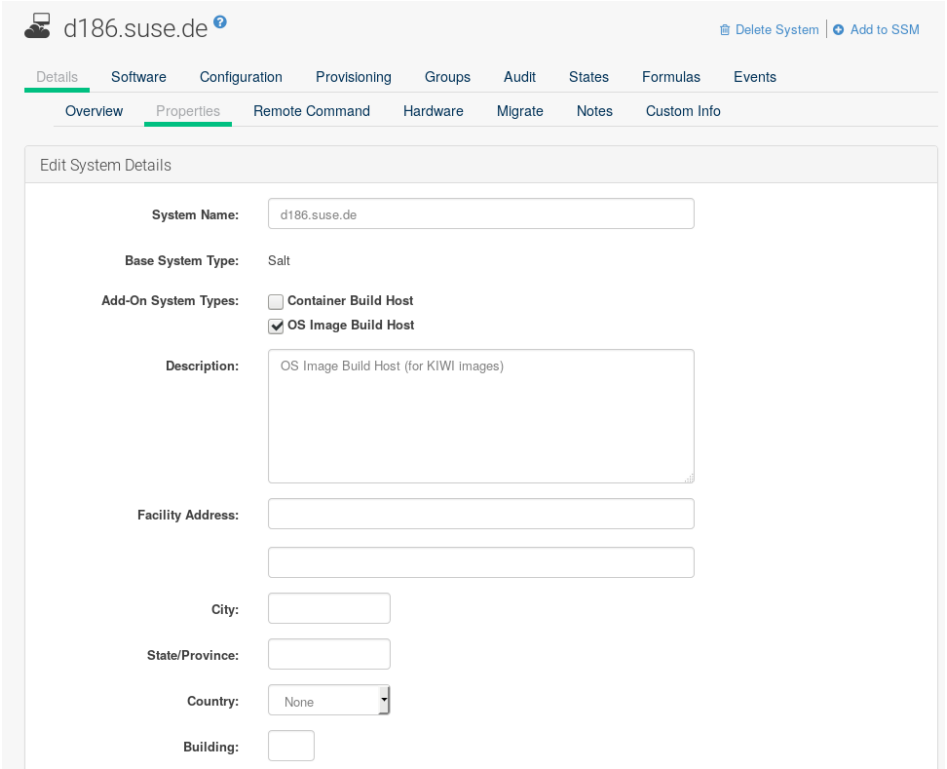
This procedure will guide you through the initial configuration for a build host.

The operating system on the build host must match the operating system on the targeted image.

For example, build SUSE Linux Enterprise Server 12 based images on a build host running SUSE Linux Enterprise Server 12 OS version. Build SUSE Linux Enterprise Server 11 based images on a build host running SUSE Linux Enterprise Server 11 OS version.

Configure the build host in the Uyuni Web UI:

1. Select a client that will be designated as a build host from the **Systems** › **Overview** page.
2. Navigate to the **System Details** › **Properties** tab, enable the `Add-on System Type OS Image Build Host`. Confirm with **`[Update Properties]`**.



3. Navigate to the **System Details** › **Software** › **Software Channels** tab, enable Uyuni Client tools (for example: `SLE-Manager-Tools12-Pool` and `SLE-Manager-Tools12-Updates`). Schedule and click **`[Confirm]`**.
4. Install Kiwi and all required packages by applying `Highstate`. From the system details page select **States** › **Highstate** and click **`[Apply Highstate]`**. Alternatively, apply Highstate from the Uyuni Server command line:

```
salt '$your_client' state.highstate
```

Uyuni Web Server Public Certificate RPM

Build host provisioning copies the Uyuni certificate RPM to the build host. This certificate is used for accessing repositories provided by Uyuni.

The certificate is packaged in RPM by the `mgr-package-rpm-certificate-osimage` package script. The package script is called automatically during a new Uyuni installation.

When you upgrade the `spacewalk-certs-tools` package, the upgrade scenario will call the package script using the default values. However if the certificate path was changed or unavailable, you will need to call the package script manually using `--ca-cert-full-path <path_to_certificate>` after the upgrade procedure has finished.

Listing 1. Package script call example

```
/usr/sbin/mgr-package-rpm-certificate-osimage --ca-cert-full-path /root/ssl-build/RHN-ORG-
TRUSTED-SSL-CERT
```

The RPM package with the certificate is stored in a salt-accessible directory such as `/usr/share/susemanager/salt/images/rhn-org-trusted-ssl-cert-osimage-1.0-1.noarch.rpm`.

The RPM package with the certificate is provided in the local build host repository `/var/lib/Kiwi/repo`.

The RPM Package with the Uyuni Certificate Must Be Specified in the Build Source

Make sure your build source Kiwi configuration contains `rhn-org-trusted-ssl-cert-osimage` as a required package in the `bootstrap` section.

Listing 2. config.xml

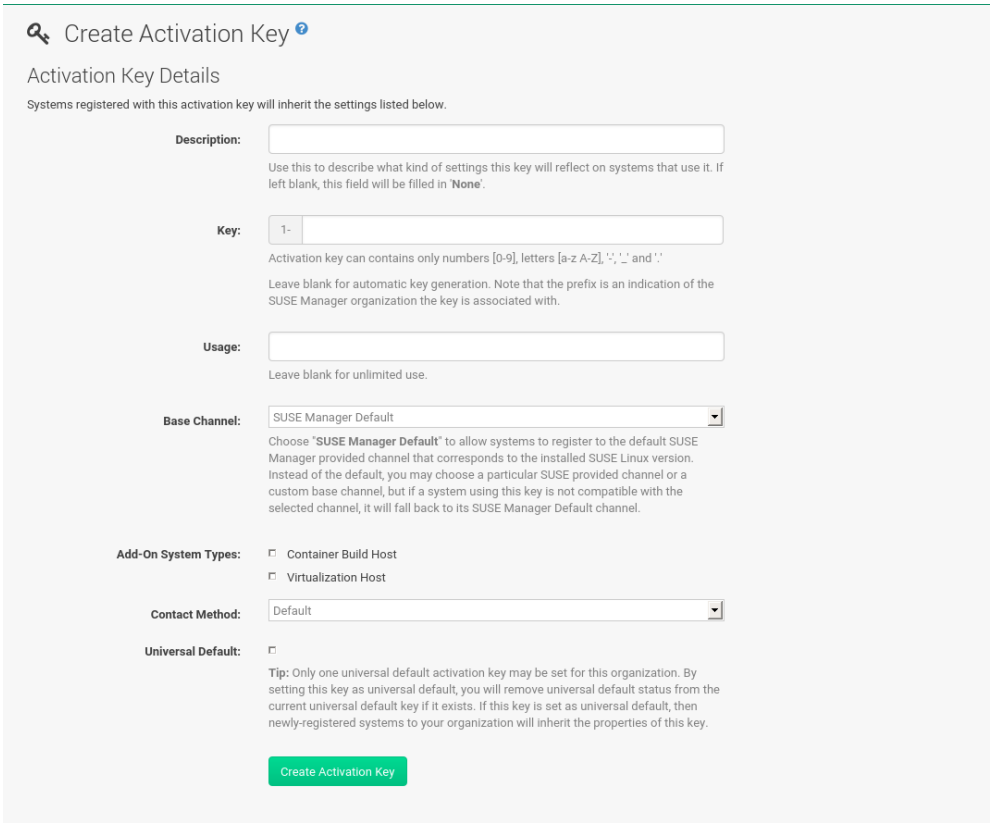
```
...
  <packages type="bootstrap">
    ...
    <package name="rhn-org-trusted-ssl-cert-osimage"
bootinclude="true"/>
  </packages>
...
```

Create an Activation Key for OS Images

Create an activation key associated with the channel that your OS Images will use as repositories when building the image.

Activation keys are mandatory for OS Image building.

To build OS Images, you will need an activation key that is associated with a channel other than `SUSE Manager Default`.



1. In the Web UI, select **Systems › Activation Keys**.
2. Click `Create Key`.
3. Enter a `Description`, a `Key` name, and use the drop-down box to select a `Base Channel` to associate with the key.
4. Confirm with **`[Create Activation Key]`**.

For more information, see [bp.key.managment].

Create an Image Store

OS Images can require a significant amount of storage space. Therefore, we recommended that the OS Image store is located on a partition of its own or on a Btrfs subvolume, separate from the root partition. By default, the image store will be located at `/srv/www/os-images`.

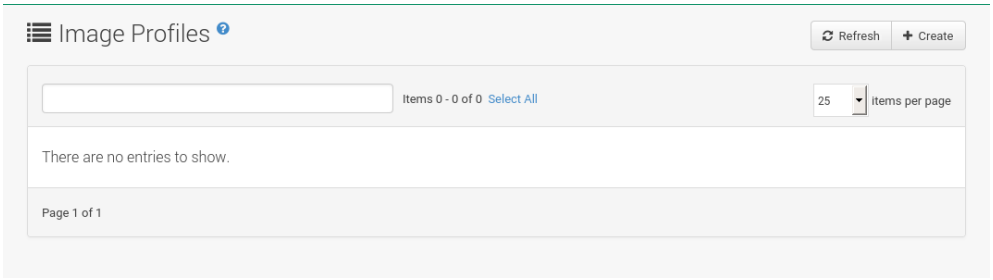
Image Stores for Kiwi Build Type

Image stores for Kiwi build type, used to build system, virtual, and other images, are not supported yet.

Images are always stored in `/srv/www/os-images/<organization id>` and are accessible via HTTP/HTTPS `https://<susemanager_host>/os-images/<organization id>`.

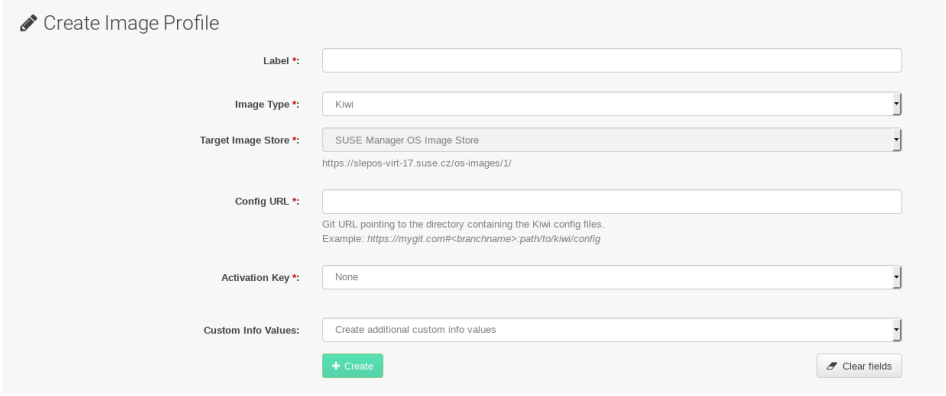
Create an Image Profile

Manage image profiles using the Web UI.



Procedure: Create an Image Profile

1. To create an image profile select from **Images › Profiles** and click **`[Create]`**.



2. In the `Label` field, provide a name for the `Image Profile`.
3. Use `Kiwi` as the `Image Type`.
4. Image store is automatically selected.
5. Enter a `Config URL` to the directory containing the Kiwi configuration files:
    a. git URI
    b. HTTPS tarball
    c. Path to build host local directory
6. Select an `Activation Key`. Activation keys ensure that images using a profile are assigned to the correct channel and packages.

Relationship Between Activation Keys and Image Profiles

When you associate activation key with an image profile you are ensuring any image using the profile will use the correct software channel and any packages in the channel.

7. Confirm with the **`[Create]`** button.

Source format options

- git/HTTP(S) URL to the repository

  URL to the git repository containing the sources of the image to be built. Depending on the layout of the repository the URL can be:

  ```
  https://github.com/SUSE/manager-build-profiles
  ```

  You can specify a branch after the `#` character in the URL. In this example, we use the `master` branch:

  ```
  https://github.com/SUSE/manager-build-profiles#master
  ```

  You can specify a directory that contains the image sources after the `:` character. In this example, we use `OSImage/POS_Image-JeOS6`:

  ```
  https://github.com/SUSE/manager-build-profiles#master:OSImage/POS_Image-JeOS6
  ```

- HTTP(S) URL to the tarball

  URL to the tar archive, compressed or uncompressed, hosted on the webserver.

  ```
  https://myimagesourceserver.example.org/MyKiwiImage.tar.gz
  ```

- Path to the directory on the build host

  Enter the path to the directory with the Kiwi build system sources. This directory must be present on the selected build host.

  ```
  /var/lib/Kiwi/MyKiwiImage
  ```

Example of Kiwi Sources

Kiwi sources consist at least of `config.xml`. Usually, `config.sh` and `images.sh` are present as well. Sources can also contain files to be installed in the final image under the `root` subdirectory.

For information about the Kiwi build system, see the Kiwi documentation.

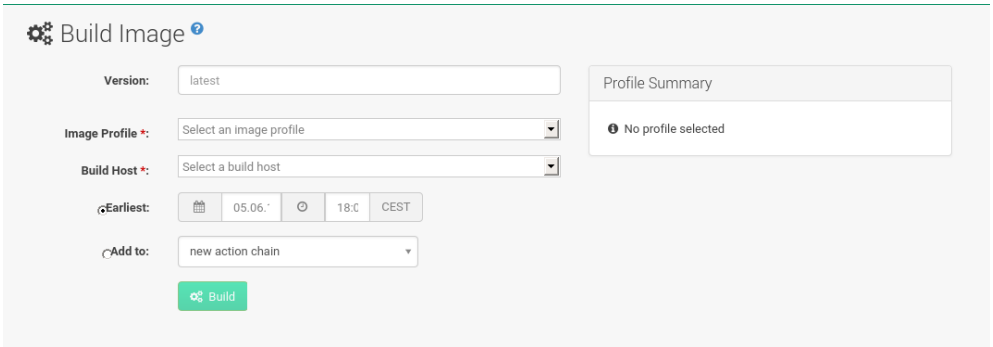
SUSE provides examples of fully functional image sources at the SUSE/manager-build-profiles public GitHub repository.

Listing 3. Example of JeOS config.xml

```
<?xml version="1.0" encoding="utf-8"?>

<image schemaversion="6.1" name="POS_Image_JeOS6">
    <description type="system">
        <author>Admin User</author>
        <contact>noemail@example.com</contact>
        <specification>SUSE Linux Enterprise 12 SP3 JeOS</specification>
    </description>
    <preferences>
        <version>6.0.0</version>
        <packagemanager>zypper</packagemanager>
        <bootsplash-theme>SLE</bootsplash-theme>
        <bootloader-theme>SLE</bootloader-theme>

        <locale>en_US</locale>
        <keytable>us.map.gz</keytable>
        <timezone>Europe/Berlin</timezone>
        <hwclock>utc</hwclock>

        <rpm-excludedocs>true</rpm-excludedocs>
        <type boot="saltboot/suse-SLES12" bootloader="grub2" checkprebuilt="true"
compressed="false" filesystem="ext3" fsmountoptions="acl" fsnocheck="true" image="pxe"
kernelcmdline="quiet"></type>
    </preferences>
    <!--    CUSTOM REPOSITORY
    <repository type="rpm-dir">
      <source path="this://repo"/>
    </repository>
    -->
    <packages type="image">
        <package name="patterns-sles-Minimal"/>
        <package name="aaa_base-extras"/> <!-- wouldn't be SUSE without that ;-) -->
        <package name="kernel-default"/>
        <package name="salt-minion"/>
        ...
    </packages>
    <packages type="bootstrap">
        ...
        <package name="sles-release"/>
        <!-- this certificate package is required to access {productname} repositories
             and is provided by {productname} automatically -->
        <package name="rhn-org-trusted-ssl-cert-osimage" bootinclude="true"/>

    </packages>
    <packages type="delete">
        <package name="mtools"/>
        <package name="initviocons"/>
        ...
    </packages>
</image>
```

Build an Image

There are two ways to build an image using the Web UI. Either select **Images › Build**, or click the build icon in the **Images › Profiles** list.



Procedure: Building an Image

1. Select **Images › Build**.
2. Add a different tag name if you want a version other than the default `latest` (applies only to containers).
3. Select the `Image Profile` and a `Build Host`.

Profile Summary

A `Profile Summary` is displayed to the right of the build fields. When you have selected a build profile, detailed information about the selected profile will show up in this area.

4. To schedule a build, click the **`[Build]`** button.

After the image is successfully built, the inspection phase begins. During the inspection phase SUSE Manager collects information about the image:

- List of packages installed in the image
- Checksum of the image
- Image type and other image details

If the built image type is `PXE`, a Salt pillar will also be generated. Image pillars are stored in the `/srv/susemanager/pillar_data/images/` directory and the Salt subsystem can access details about the generated image. Details include where the pillar is located and provided, image checksums, information needed for network boot, and more.

The generated pillar is available to all connected clients.

Troubleshooting

Building an image requires of several dependent steps. When the build fails, investigation of Salt states results can help you to identify the source of the failure. Usual checks when the build fails:

- The build host can access the build sources

- There is enough disk space for the image on both the build host and the Uyuni server
- The activation key has the correct channels associated with it
- The build sources used are valid
- The RPM package with the Uyuni public certificate is up to date and available at `/usr/share/susemanager/salt/images/rhn-org-trusted-ssl-cert-osimage-1.0-1.noarch.rpm`. For more on how to refresh a public certificate RPM, see Create a Build Host.

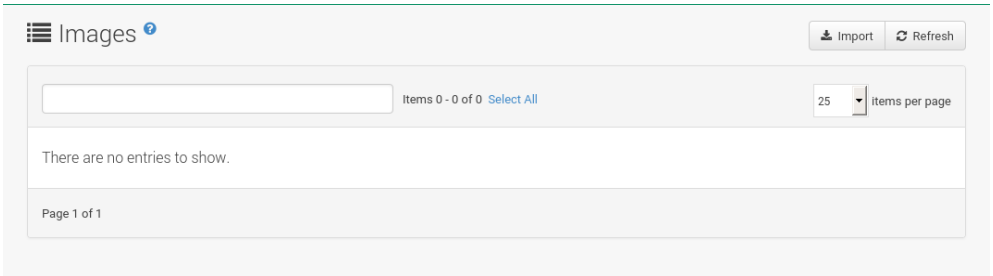
Limitations

The section contains some known issues when working with images.

- HTTPS certificates used to access the HTTP sources or git repositories should be deployed to the client by a custom state file, or configured manually.
- Importing Kiwi-based images is not supported.

List Image Profiles Available for Building

To list images available for building select **Images › Image List**. A list of all images will be displayed.



Displayed data about images includes an image `Name`, its `Version` and the build `Status`. You will also see the image update status with a listing of possible patch and package updates that are available for the image.

Clicking the **`[Details]`** button on an image will provide a detailed view. The detailed view includes an exact list of relevant patches and a list of all packages installed within the image.

The patch and the package list is only available if the inspect state after a build was successful.

Channel Management

Channels are a method of grouping software packages.

In Uyuni, channels are grouped into base and child channels, with base channels grouped by operating system type, version, and architecture, and child channels being compatible with their related base channel. When a client has been assigned to a base channel, it is only possible for that system to install the related child channels. Organizing channels in this way ensures that only compatible packages are installed on each system.

Software channels use repositories to provide packages. The channels mirror the repositories in Uyuni, and the package names and other data are stored in the Uyuni database. You can have any number of repositories associated with a channel. The software from those repositories can then be installed on clients by subscribing the client to the appropriate channel.

Clients can only be assigned to one base channel. The client can then install or update packages from the repositories associated with that base channel and any of its child channels.

Uyuni provides a number of vendor channels, which provide you everything you need to run Uyuni. Uyuni Administrators and Channel Administrators have channel management authority, which gives them the ability to create and manage their own custom channels. If you want to use your own packages in your environment, you can create custom channels. Custom channels can be used as a base channel, or you can associate them with a vendor base channel.

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

When you subscribe to a vendor channel with a traditional client, the product package will automatically be installed. On Salt clients, the packages will be added to the package state, and you will need to apply the highstate to push the changes to your systems.

Custom Channels

While Uyuni provides all required channels, you might find it useful to create custom channels specific to your environment.

Custom channels give you the ability to create your own software packages and repositories, which you can use to update your clients. They also allow you to use software provided by third party vendors in your environment.

You must have administrator privileges to be able to create and manage custom channels.

Before you create a custom channel, determine which base channel you want to associate it with, and which repositories you want to use for content.

This section gives more detail on how to create, administer, and delete custom channels.

Creating Custom Channels

If you have custom software packages that you need to install on your Uyuni systems, you can create a custom child channel to manage them. You will need to create the channel in the Uyuni Web UI and create a repository for the packages, before assigning the channel to your systems.

You can select a vendor channel as the base channel if you want to use packages provided by a vendor. Alternatively, select `none` to make your custom channel a base channel.

Custom channels will sometimes require additional security settings. Many third party vendors secure packages with GPG. If you want to use GPG-protected packages in your custom channel, you will need to trust the GPG key which has been used to sign the metadata. You can then check the `Has Signed Metadata?` check box to match the package metadata against the trusted GPG keys. For more information on importing GPG keys, see **[Reference › Systems ›]**.

Do not create child channels containing packages that are not compatible with the client system.

Procedure: Creating a Custom Channel

1. In the Uyuni Web UI, navigate to **Software › Manage › Channels**, and click **`[Create Channel]`**.
2. On the `Create Software Channel` page, give your channel a name (for example, `My Tools SLES 15 SP1 x86_64`) and a label (for example, `my-tools-sles15sp1-x86_64`). Labels must not contain space characters or capital letters.
3. In the `Parent Channel` drop down, choose the relevant base channel (for example, `SLE-Product-SLES15-SP1-Pool for x86_64`). Ensure that you choose the compatible parent channel for your packages.
4. In the `Architecture` drop down, choose the appropriate hardware architecture (for example, `x86_64`).
5. Provide any additional information in the contact details, channel access control, and GPG fields, as required for your environment.
6. Click **`[Create Channel]`**.

Procedure: Creating a Software Repository

1. In the Uyuni Web UI, navigate to **Software › Manage › Repositories**, and click **`[Create Repository]`**.
2. On the `Create Repository` page, give your repository a label (for example, `my-tools-sles15sp1-x86_64-repo`).
3. In the `Repository URL` field, provide the path to the directory with the `repodata` file (for example, `file:///opt/mytools/`). You can use any valid addressing protocol in this field.
4. Uncheck the `Has Signed Metadata?` check box.

5. OPTIONAL: Complete the SSL fields if your repository requires client certificate authentication.
6. Click **[Create Repository]**.

Procedure: Assigning the Repository to a Channel

1. Assign your new repository to your custom channel by navigating to **Software › Manage › Channels**, clicking the name of your newly created custom channel, and navigating to the Repositories tab.
2. Ensure the repository you want to assign to the channel is checked, and click **[Update Repositories]**.
3. Navigate to the Sync tab and click **[Sync Now]** to synchronize immediately. You can also set an automated synchronization schedule on this tab.

Procedure: Adding Custom Channels to an Activation Key

1. In the Uyuni Web UI, navigate to **Systems › Activation Keys**, and select the key you want to add the custom channel to.
2. On the Details tab, in the Child Channels listing, select the channel to associate. You can select multiple channels, if you need to.
3. Click **[Update Activation Key]**.

Deleting Custom Channels

You cannot delete Uyuni channels with the Web UI. Only custom channels can be deleted.

1. In the Uyuni Web UI, navigate to **Software › Manage › Channels**, and select the channel you want to delete.
2. Click **[Delete software channel]**.
3. On the Delete Channel page, check the details of the channel you are deleting, and check the Unsubscribe Systems checkbox to remove the custom channel from any systems that might still be subscribed.
4. Click **[Delete Channel]**.

When channels are deleted, the packages that are part of the deleted channel are not automatically removed. You will not be able to update packages that have had their channel deleted.

You can delete packages that are not associated with a channel in the Uyuni Web UI. Navigate to **Software › Manage › Packages**, check the packages to remove, and click **[Delete Packages]**.

Live Patching with SUSE Manager

Performing a kernel update usually requires a system reboot. Common vulnerability and exposure (CVE) patches should be applied as soon as possible, but if you cannot afford the downtime, you can use Live Patching to inject these important updates and skip the need to reboot.

The procedure for setting up Live Patching is slightly different for SLES 12 and SLES 15. Both procedures are documented in this section.

Live Patching on SLES 15

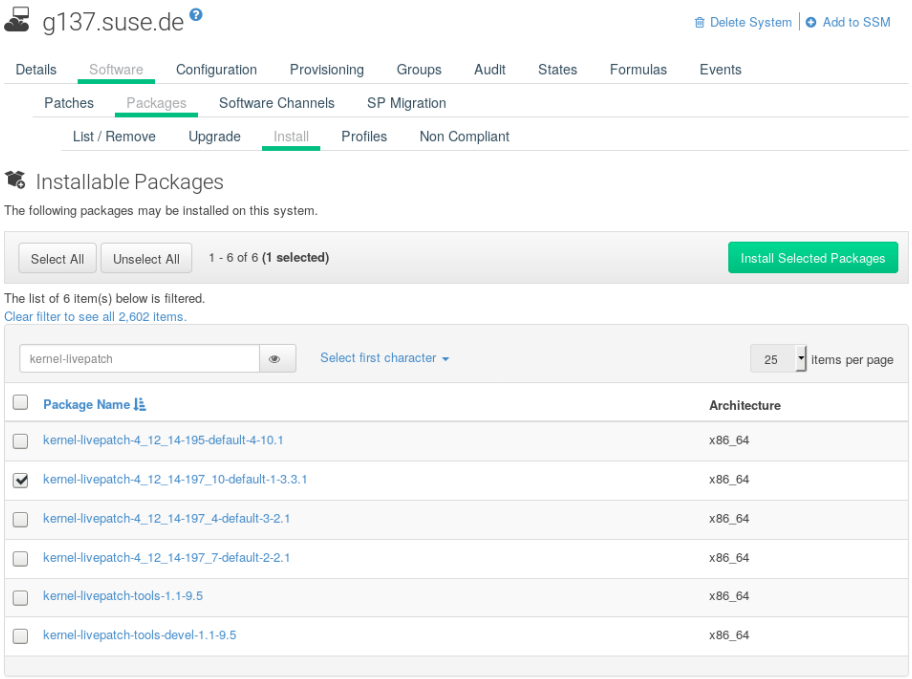
On SLES 15 systems and newer, live patching is managed by the `klp livepatch` tool.

Before you begin, ensure:

- Uyuni is fully updated
- You have one or more Salt clients running SLES 15 (SP1 or later)
- Your SLES 15 Salt clients are registered with Uyuni
- You have access to the SLES 15 channels appropriate for your architecture, including the Live Patching child channel (or channels)
- The clients are fully synchronized

Procedure: Setting up for Live Patching

1. Select the client you want to manage with Live Patching from **Systems** › **Overview**, and navigate to the **Software** › **Packages** › **Install** tab. Search for the `kernel-livepatch` package, and install it.



2. Apply the highstate to enable Live Patching, and reboot the client.
3. Repeat for each client that you want to manage with Live Patching.
4. To check that Live Patching has been enabled correctly, select the client from **Systems › System List**, and ensure that `Live Patch` appears in the `Kernel` field.

When you have the Live Patching channel installed on the client, you can clone the default vendor channel. This cloned channel will be used to manage Live Patching on your clients.

Cloned vendor channels should be prefixed by `dev` for development, `testing`, or `prod` for production. In this procedure, you will create a `dev` cloned channel, and later, you will need to promote the channel to `testing`.

Procedure: Cloning Live Patching Channels

1. At the command prompt on the client, as `root`, obtain the current package channel tree:

```
# spacewalk-manage-channel-lifecycle --list-channels
Spacewalk Username: admin
Spacewalk Password:
Channel tree:

 1. sles15-{sp-vert}-pool-x86_64
      \__ sle-live-patching15-pool-x86_64-{sp-vert}
      \__ sle-live-patching15-updates-x86_64-{sp-vert}
      \__ sle-manager-tools15-pool-x86_64-{sp-vert}
      \__ sle-manager-tools15-updates-x86_64-{sp-vert}
      \__ sles15-{sp-vert}-updates-x86_64
```

2. Use the `spacewalk-manage-channel` command with the `init` argument to automatically create a new development clone of the original vendor channel:

```
spacewalk-manage-channel-lifecycle --init -c sles15-{sp-vert}-pool-x86_64
```

3. Check that `dev-sles15-{sp-vert}-updates-x86_64` is available in your channel list.

Check the `dev` cloned channel you created, and remove any kernel updates that require a reboot.

Procedure: Removing Non-Live Kernel Patches from Cloned Channels

1. Check the current kernel version by selecting the client from **Systems › System List**, and taking note of the version displayed in the `Kernel` field.
2. In the Uyuni Web UI, select the client from **Systems › Overview**, navigate to the **Software › Manage › Channels** tab, and select `dev-sles15-sp{sp-vert}-updates-x86_64`. Navigate to the `Patches` tab, and click **`[List/Remove Patches]`**.
3. In the search bar, type `kernel` and identify the kernel version that matches the kernel currently used by your client.
4. Remove all kernel versions that are newer than the currently installed kernel.

Your channel is now set up for Live Patching, and can be promoted to `testing`. In this procedure, you will also add the Live Patching child channels to your client, ready to be applied.

Procedure: Promoting Live Patching Channels

1. At the command prompt on the client, as `root`, promote and clone the `dev-sles15-{sp-vert}-pool-x86_64` channel to a new testing channel:

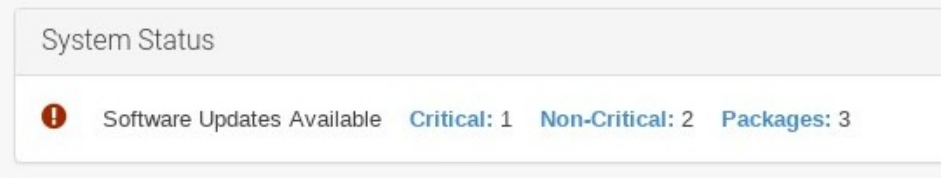
```
# spacewalk-manage-channel-lifecycle --promote -c dev-sles15-{sp-vert}-pool-x86_64
```

2. In the Uyuni Web UI, select the client from **Systems › Overview**, and navigate to the **Software › Software Channels** tab.
3. Check the new `test-sles15-sp3-pool-x86_64` custom channel to change the base channel, and check both corresponding Live Patching child channels.
4. Click **[Next]**, confirm that the details are correct, and click **[Confirm]** to save the changes.

You can now select and view available CVE patches, and apply these important kernel updates with Live Patching.

Procedure: Applying Live Patches to a Kernel

1. In the Uyuni Web UI, select the client from **Systems › Overview**. You will see a banner at the top of the screen showing the number of critical and non-critical packages available for the client:



2. Click **[Critical]** to see a list of the available critical patches.
3. Select any patch with a synopsis reading `Important: Security update for the Linux kernel`. Security bugs will also include their CVE number, where applicable.
4. OPTIONAL: If you know the CVE number of a patch you want to apply, you can search for it in **Audit › CVE Audit**, and apply the patch to any clients that require it.

Not all kernel patches are Live Patches! Non-Live kernel patches are represented by a `Reboot Required` icon located next to the `Security` shield icon. These patches will always require a reboot.

Not all security issues can be fixed by applying a live patch. Some security issues can only be fixed by applying a full kernel update and will require a reboot. The assigned CVE numbers for these issues are not included in live patches. A CVE audit will display this requirement.

Live Patching on SLES 12

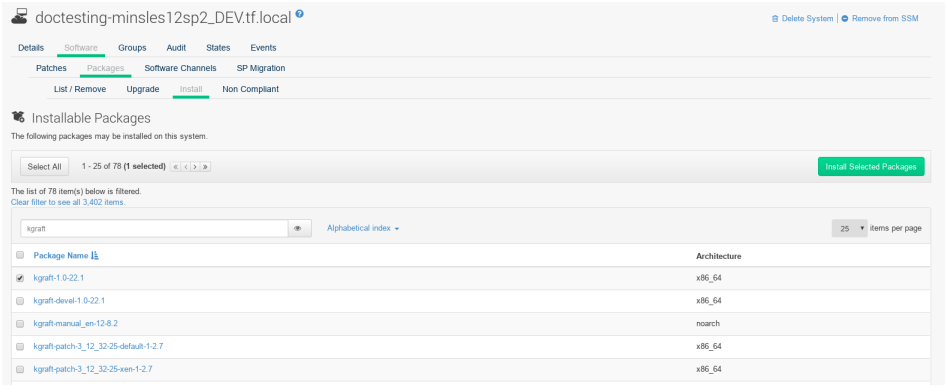
On SLES 12 systems, live patching is managed by kGraft. For in depth information covering kGraft use, see https://documentation.suse.com/sles/12-SP4/html/SLES-all/cha-kgraft.html.

Before you begin, ensure:

- Uyuni is fully updated
- You have one or more Salt clients running SLES 12 (SP1 or later)
- Your SLES 12 Salt clients are registered with Uyuni
- You have access to the SLES 12 channels appropriate for your architecture, including the Live Patching child channel (or channels)
- The clients are fully synchronized

Procedure: Setting up for Live Patching

1. Select the client you want to manage with Live Patching from **Systems** › **Overview**, and on the system details page navigate to the **Software** › **Packages** › **Install** tab. Search for the `kgraft` package, and install it.



2. Apply the highstate to enable Live Patching, and reboot the client.
3. Repeat for each client that you want to manage with Live Patching.
4. To check that Live Patching has been enabled correctly, select the client from **Systems** › **System List**, and ensure that `Live Patching` appears in the `Kernel` field.

When you have the Live Patching channel installed on the client, you can clone the default vendor channel. This cloned channel will be used to manage Live Patching on your clients.

Cloned vendor channels should be prefixed by `dev` for development, `testing`, or `prod` for production. In this procedure, you will create a `dev` cloned channel, and later, you will need to promote the channel to `testing`.

Procedure: Cloning Live Patching Channels

1. At the command prompt on the client, as `root`, obtain the current package channel tree:

```
# spacewalk-manage-channel-lifecycle --list-channels
Spacewalk Username: admin
Spacewalk Password:
Channel tree:

 1. sles12-sp4-pool-x86_64
      \__ sle-live-patching12-pool-x86_64-sp4
      \__ sle-live-patching12-updates-x86_64-sp4
      \__ sle-manager-tools12-pool-x86_64-sp4
      \__ sle-manager-tools12-updates-x86_64-sp4
      \__ sles12-sp4-updates-x86_64
```

2. Use the `spacewalk-manage-channel` command with the `init` argument to automatically create a new development clone of the original vendor channel:

```
spacewalk-manage-channel-lifecycle --init -c sles12-sp4-pool-x86_64
```

3. Check that `dev-sles12-sp4-updates-x86_64` is available in your channel list.

Check the `dev` cloned channel you created, and remove any kernel updates that require a reboot.

Procedure: Removing Non-Live Kernel Patches from Cloned Channels

1. Check the current kernel version by selecting the client from **Systems** › **System List**, and taking note of the version displayed in the `Kernel` field.
2. In the Uyuni Web UI, select the client from **Systems** › **Overview**, navigate to the **Software** › **Manage** › **Channels** tab, and select `dev-sles12-sp4-updates-x86_64`. Navigate to the `Patches` tab, and click **`[List/Remove Patches]`**.
3. In the search bar, type `kernel` and identify the kernel version that matches the kernel currently used by your client.
4. Remove all kernel versions that are newer than the currently installed kernel.

Your channel is now set up for Live Patching, and can be promoted to `testing`. In this procedure, you will also add the Live Patching child channels to your client, ready to be applied.

Procedure: Promoting Live Patching Channels

1. At the command prompt on the client, as `root`, promote and clone the `dev-sles12-sp4-pool-x86_64` channel to a new testing channel:

```
# spacewalk-manage-channel-lifecycle --promote -c dev-sles12-sp4-pool-x86_64
```

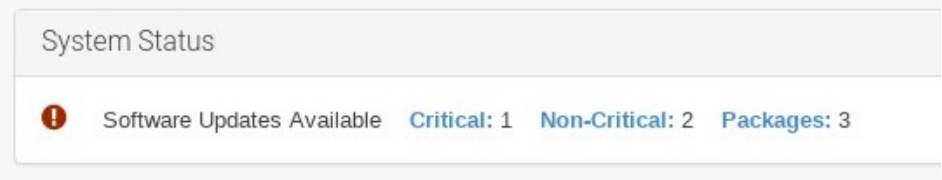
2. In the Uyuni Web UI, select the client from **Systems** › **Overview**, and navigate to the **Software** › **Software Channels** tab.
3. Check the new `test-sles12-sp4-pool-x86_64` custom channel to change the base channel, and check both corresponding Live Patching child channels.

4. Click **[Next]**, confirm that the details are correct, and click **[Confirm]** to save the changes.

You can now select and view available CVE patches, and apply these important kernel updates with Live Patching.

Procedure: Applying Live Patches to a Kernel

1. In the Uyuni Web UI, select the client from **Systems › Overview**. You will see a banner at the top of the screen showing the number of critical and non-critical packages available for the client:



2. Click **[Critical]** to see a list of the available critical patches.
3. Select any patch with a synopsis reading `Important: Security update for the Linux kernel`. Security bugs will also include their CVE number, where applicable.
4. OPTIONAL: If you know the CVE number of a patch you want to apply, you can search for it in **Audit › CVE Audit**, and apply the patch to any clients that require it.

Not all kernel patches are Live Patches! Non-Live kernel patches are represented by a `Reboot Required` icon located next to the `Security` shield icon. These patches will always require a reboot.

Not all security issues can be fixed by applying a live patch. Some security issues can only be fixed by applying a full kernel update and will require a reboot. The assigned CVE numbers for these issues are not included in live patches. A CVE audit will display this requirement.

Monitoring with Prometheus

You can monitor your Uyuni environment using Prometheus and Grafana. Uyuni Server and Proxy are able to provide self-health metrics, or install and manage a limited number of Prometheus exporters on Salt clients.

Prometheus fetches metrics using a pull mechanism, so the server must be able to establish network communications to monitored clients. Clients must have an open port and be reachable on the network.

Prometheus is a monitoring tool that is used to record real-time metrics in a time-series database. It is an open-source software project, written in Go. Metrics are collected using HTTP pulls, allowing for higher performance and scalability. For more information about Prometheus, see https://prometheus.io/docs/.

Grafana is a tool for data visualization, monitoring, and analysis. It is used to create dashboards with panels representing specific metrics over a set period of time. Grafana is commonly used together with Prometheus, but also supports other data sources such as ElasticSearch, MySQL, PostgreSQL, and Influx DB. For more information about Grafana, see https://grafana.com/docs/.

You need to install Prometheus and Grafana on a machine separate from the Uyuni Server. We recommend you use a managed Salt client as your monitoring server.

Prometheus and Grafana packages are included in the SUSE Manager Client Tools for SUSE Linux Enterprise 12 and SUSE Linux Enterprise 15.

Prometheus Metrics

Prometheus metrics are time series data, or timestamped values belonging to the same group or dimension. A metric is uniquely identified by its name and set of labels.

```
  metric name              labels                    timestamp    value
┌──────────────────────┴─────────────┐ ┌──────────────────────┴──────────────────────┐
┌──────────────┴──────────────┐ ┌────┴────┐
http_requests_total{status="200", method="GET"}  @1557331801.111  42236
```

Each application or system being monitored must expose metrics in the format above, either through code instrumentation or Prometheus exporters.

The different metric types are:

- Counter - cumulative values. For example, number of errors
- Gauge - can go up or down. For example, temperature
- Histogram - count observations in buckets
- Summary - similar to histogram, but provides totals (sum and count)

For more information about metric types, see https://prometheus.io/docs/concepts/metric_types/.

PromQL

Prometheus has its own query language called PromQL, which is a functional expression language. PromQL allows you to filter multi-dimensional time series data. It is used in all Prometheus interactions.

In PromQL, an expression can evaluate to one of three types:

- Instant vector: a set of time series containing a single sample for each time series, all sharing the same timestamp
- Range vector: a set of time series containing a range of data points over time for each time series
- Scalar: a numeric floating point value

The core part of any PromQL query is the metric name, for example, **http_requests_total**. Labels can be used as optional selectors. This example returns the total number of HTTP requests that have status **200** and method **GET**:

```
http_requests_total{status="200", method="GET"}
```

For more information about PromQL, see the official Prometheus documentation at https://prometheus.io/docs/prometheus/latest/querying/basics/.

Exporters

Exporters are libraries that help with exporting metrics from third-party systems as Prometheus metrics. Exporters are useful whenever it is not feasible to instrument a given application or system with Prometheus metrics directly. Multiple exporters can run on a monitored host to export local metrics.

The Prometheus community provides a list of official exporters, and more can be found as community contributions. For detailed information and an extensive list of exporters, see https://prometheus.io/docs/instrumenting/exporters/.

With Uyuni 4, you can set up the Server and Proxy to expose Prometheus metrics to provide insights about self-health of Uyuni. Metrics are available for these services:

- Hardware and Operating System
- Java Virtual Machines
- Apache
- Squid
- PostgreSQL
- Uyuni internals

The self-health metrics are made available by Uyuni Java application combined with Prometheus standalone exporters, running as systemd daemons.

Uyuni requires these packages to be installed on the Server and the Proxy. The packages are shipped with Uyuni Server and Proxy, but their respective systemd daemons are disabled by default.

These exporter packages are shipped with Uyuni Server:

- Node exporter: `golang-github-prometheus-node_exporter`. See https://github.com/prometheus/node_exporter.
- PostgreSQL exporter: `golang-github-wrouesnel-postgres_exporter`. See https://github.com/wrouesnel/postgres_exporter.
- JMX exporter: `prometheus-jmx_exporter`. See https://github.com/prometheus/jmx_exporter.
- Apache exporter: `golang-github-lusitaniae-apache_exporter`. See https://github.com/Lusitaniae/apache_exporter.

These exporter packages are shipped with Uyuni Proxy:

- Node exporter: `golang-github-prometheus-node_exporter`. See https://github.com/prometheus/node_exporter.
- Squid exporter: `golang-github-boynux-squid_exporter`. See https://github.com/boynux/squid-exporter.

Install and Configure Prometheus

Prometheus is installed on your monitoring server from a package, and needs configuration before you can use it to gather metrics. We recommend you use a managed Salt client as your monitoring server.

Installing Prometheus

If your monitoring server is a Uyuni client, you can install the Prometheus package using the Uyuni Web UI. Otherwise you can download and install the package on your monitoring server manually.

Procedure: Installing Prometheus

1. On the monitoring server, install the `golang-github-prometheus-prometheus` package:

```
zypper in golang-github-prometheus-prometheus
```

2. Enable the Prometheus service:

```
systemctl enable --now prometheus
```

3. Check that the Prometheus interface is loading correctly. In your browser, navigate to the URL of the

server where Prometheus is installed, and listen on port 9090 (for example, `http://example.com:9090`).

Configuring Prometheus

Prometheus requires some configuration to collect metrics and set up alarms, or to display metrics graphically in Grafana. You can configure Prometheus in the static configuration file at `/etc/prometheus/prometheus.yml`. It is important to understand how this file is structured. For example:

```
scrape_configs:
- job_name: 'suse-manager-server'
  static_configs:
    - targets:
      - 'suse-manager.local:9100'  # Node exporter
      - 'suse-manager.local:9187'  # PostgreSQL exporter
      - 'suse-manager.local:5556'  # JMX exporter (Tomcat)
      - 'suse-manager.local:5557'  # JMX exporter (Taskomatic)
      - 'suse-manager.local:9800'  # Taskomatic
    - targets:
      - 'suse-manager.local:80'    # Message queue
      labels:
        __metrics_path__: /rhn/metrics
```

For more information about configuring Prometheus, see the official Prometheus documentation at https://prometheus.io/docs/prometheus/latest/configuration/configuration/

Monitoring Salt Clients

Prometheus metrics exporters can also be used on Salt clients. The packages are available from the Uyuni client tools channels, and can be enabled and configured directly in the Uyuni Web UI. Currently, two exporters are supported:

- Node exporter: `golang-github-prometheus-node_exporter`. See https://github.com/prometheus/node_exporter.
- PostgreSQL exporter: `golang-github-wrouesnel-postgres_exporter`. See https://github.com/wrouesnel/postgres_exporter.

Installing and configuring exporters is done using a Salt formula.

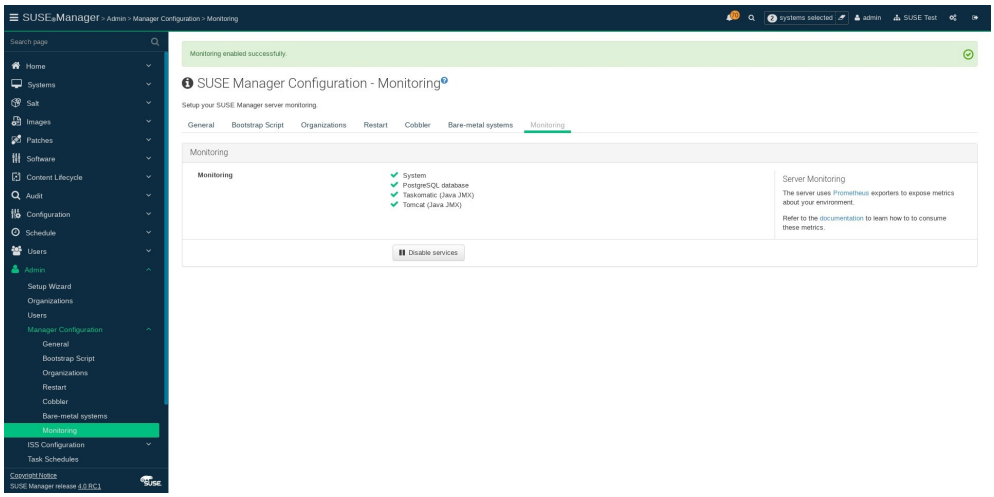
When you have the exporters installed and configured, you can begin using Prometheus to scrape metrics from monitored systems. You can do this directly through the Uyuni Web UI, or set up service discovery. Service discovery instructs Prometheus to automatically scrape metrics from systems as they are enabled.

Enable and Configure Monitoring

Procedure: Enabling Self Monitoring for Uyuni

1. In the Uyuni Web UI, navigate to **Admin › Manager Configuration › Monitoring**.

2. Click **[Enable services]**.

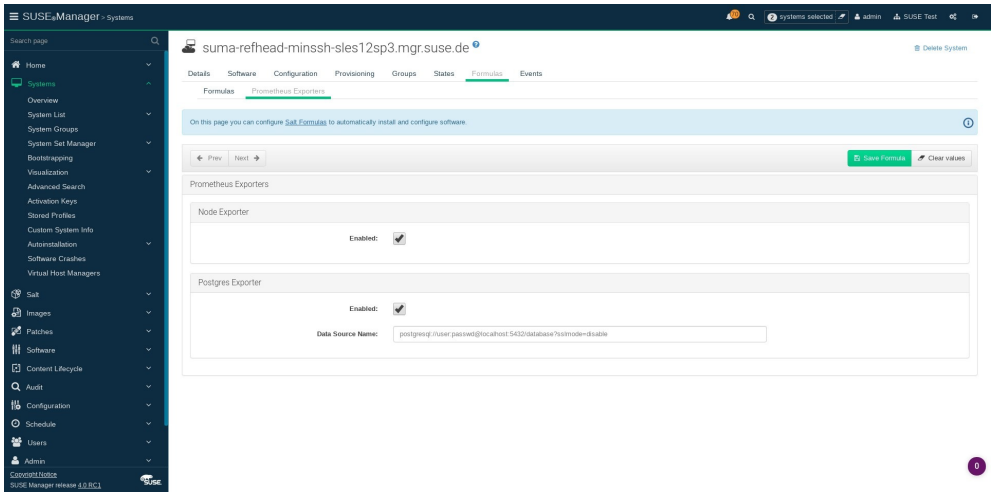


Procedure: Configuring Monitoring Formulas

1. In the SUSE Manager Web UI, open the details page of the system to be monitored, and navigate to the `Formulas` tab.
2. Check the `Monitoring` checkbox to select all monitoring formulas, and click **[Save]**.
3. Apply the highstate.

Procedure: Configuring the Exporters

1. In the SUSE Manager Web UI, open the details page of the system to be monitored, and navigate to the **Formulas › Prometheus Exporters** tab.
2. Check the `Enabled` checkbox for both the Node and the Postgres Exporter.
3. In the `Postgres Exporter` section, in the `Data Source Namer` field, enter the path to your data source (for example, `postgresql://user:passwd@localhost:5432/database?sslmode=disable`).
4. Click **[Save Formula]**.
5. Apply the highstate.



Procedure: Enable Service Discovery

This feature is a technical preview available in Uyuni 4.0.2 and later. It should not be used in production systems.

1. On the monitoring server, open the Prometheus static configuration file `/etc/prometheus/prometheus.yml`.
2. Add or update the scrape configurations section:

```
job_name: 'suma'
uyuni_sd_configs:
host: "http://your-suse-manager-server-url"
username: "apiuser"
password: "password"
```

3. Save the configuration file and restart the Prometheus service:

```
systemctl restart prometheus
```

Visualization with Grafana

The Grafana website contains dozens of dashboards uploaded by the community. For an example of the Uyuni dashboard, see https://grafana.com/dashboards/10277. For more information about dashboards, see https://grafana.com/dashboards

To use Grafana with Uyuni, you must enable metrics in the Uyuni Web UI and configure your Prometheus instance to collect those metrics.

If your monitoring server is a Uyuni client, you can install the Grafana package using the Uyuni Web UI. Otherwise you can download and install the package on your monitoring server manually.

Procedure: Setting up Grafana

1. Install the `grafana` package:

```
zypper in grafana
```

2. Enable the Grafana service:

```
systemctl enable --now grafana-server
```

3. Navigate to port 3000 in your browser.

Grafana settings are configured in `/etc/grafana/grafana.ini`.

Public Cloud

Some public cloud environments provide images for Uyuni Server and Proxy. This section discusses what you will need for running Uyuni in a public cloud, and how to set up your installation.

Public clouds provide Uyuni under a Bring Your Own Subscription (BYOS) model. This means that you must register them with the SUSE Customer Center. For more information about registering Uyuni with SUSE Customer Center, see [**Installation › General-requirements ›**].

Depending on the public cloud network you are using, you can locate the Uyuni installation images by searching for the keywords `suse`, `manager`, `proxy`, or `BYOS`.

For `SUSE Manager Server in Azure`, see [**Administration › Public-cloud-azure ›**].

Instance Requirements

Select a public cloud instance that meets the hardware requirements in [**Installation › Hardware-requirements ›**].

In addition, be aware of these considerations:

- The Uyuni setup procedure performs a forward-confirmed reverse DNS lookup. This must succeed in order for the setup procedure to complete successfully and for Uyuni to operate as expected. Therefore, it is important to perform hostname and IP configuration prior to running the Uyuni setup procedure.
- Uyuni Server and Proxy instances are expected to run in a network configuration that provides you control over DNS entries, but cannot access the wider internet. Within this network configuration DNS resolution must be provided: `hostname -f` must return the fully-qualified domain name (FQDN). DNS resolution is also important for connecting clients. DNS is dependent on the cloud framework you choose, refer to the cloud service provider documentation for detailed instructions.
- We recommend that you locate software repositories, the server database, and the proxy squid cache on an external virtual disk. This prevents data loss if the instance is unexpectedly terminated. Instructions for setting up an external virtual disk are contained in this section.

Network Setup

On a public cloud service, you must run Uyuni within a restricted network, such as VPC private subnet with an appropriate firewall setting. The instance must only be able to be accessed by machines in your specified IP ranges.

A world-accessible Uyuni instance violates the terms of the Uyuni EULA, and it will not be supported by SUSE.

To access the Uyuni Web UI, allow HTTPS when you set up your networking environment.

Set the Hostname

Uyuni requires a stable and reliable hostname. Changing the hostname at a later point can create errors.

In most public cloud environments, the method shown in this section will work correctly. However, you will have to perform the same modification for every client.

You might prefer to manage DNS resolution by creating a DNS entry in your network environment instead.

You can also manage hostname resolution by editing the `/etc/resolv.conf` file. Depending on the order of your setup, if you start the Uyuni instance prior to setting up DNS services the file may not contain the appropriate `search` directive. Check that the proper search directive exists in `/etc/resolv.conf` and add it if it is missing.

Procedure: Setting the hostname locally

1. Disable hostname setup by editing the DHCP configuration file at `/etc/sysconfig/network/dhcp`, and adding this line:

```
DHCLIENT_SET_HOSTNAME="no"
```

2. Set the hostname locally with the `hostnamectl` command. Ensure you use the system name, not the FQDN. For example, if the FQDN is `system_name.example.com`, the system name is `system_name`, and the domain name is `example.com`.

```
# hostnamectl set-hostname system_name
```

3. Create a DNS entry in your network environment for domain name resolution, or force correct resolution by editing the `/etc/hosts` file. You can find the IP address by checking your public cloud web console, or from the command line:
    - Amazon EC2 instance:

```
# ec2metadata --local-ipv4
```

    - Google Compute Engine:

```
# gcemetadata --query instance --network-interfaces --ip
```

    - Microsoft Azure:

```
# azuremetadata --internal-ip
```

In the following command, replace `<ip_address>` with IP address you retrieve from the command line above:

```
# echo "<ip_address> suma.cloud.net suma" >> /etc/hosts
```

Set up DNS Resolution

You will need to update the DNS records for the instance within the DNS service of your network environment. Refer to the cloud service provider documentation for detailed instructions:

- DNS setup on Amazon EC2
- DNS setup on Google Compute Engine
- DNS setup on Microsoft Azure

If you run a Uyuni Server instance, ensure the external storage is attached and prepared correctly, and that DNS resolution is set up as described. Start the `susemanager_setup` with YaST:

```
# /sbin/yast2 susemanager_setup
```

The Uyuni setup procedure in YaST is designed as a one pass process with no rollback or cleanup capability. Therefore, if the setup procedure is interrupted or ends with an error, it is not recommended that you repeat the setup process or attempts to manually fix the configuration. These methods are likely to result in a faulty Uyuni installation. If you experience errors during setup, start a new instance, and begin the setup procedure again on a clean system.

If you receive a message that there is not enough space available for setup, ensure that your root volume is at least 20 GB and double check that the instructions in Using Separate Storage Volume have been completed correctly.

Prior to registering instances started from on demand images remove the following packages from the instance to be registered: … cloud-regionsrv-client … **For Amazon EC2**

+ regionServiceClientConfigEC2

+ regionServiceCertsEC2 … **For Google Compute Engine**

+ cloud-regionsrv-client-plugin-gce

+ regionServiceClientConfigGCE

+ regionServiceCertsGCE … **For Microsoft Azure**

+ regionServiceClientConfigAzure

+ regionServiceCertsAzure

+ If these packages are not removed it is possible to create interference between the repositories provided by Uyuni and the repositories provided by the SUSE operated update infrastructure.

+ Additionally remove the line from the `/etc/hosts` file that contains the **susecloud.net** reference. ** If you run a Uyuni Proxy instance

+ Launch the instance, optionally with external storage configured. If you use external storage (recommended), prepare it according to Using Separate Storage Volume. It is recommended but not required to prepare the storage before configuring Uyuni proxy, as the suma-storage script will migrate any existing cached data to the external storage. After preparing the instance, register the system with the parent SUSE Manager, which could be a Uyuni Server or another Uyuni Proxy. See the **[Installation › Proxy-setup ›]** for details. When registered, run

+

```
$ /usr/sbin/configure-proxy.sh
```

+ to configure your Uyuni Proxy instance. . After the completion of the configuration step, Uyuni should be functional and running. For Uyuni Server, the setup process created an administrator user with following user name:

+ * User name: `admin`

+

Table 1. Account credentials for admin user

Amazon EC2	Google Compute Engine	Microsoft Azure
`Instance-ID`	`Instance-ID`	`Instance-Name`**-suma**

+ The current value for the `Instance-ID` or `Instance-Name` in case of the Azure Cloud, can be obtained from the public cloud Web console or from within a terminal session as follows: ** Obtain instance id from within Amazon EC2 instance

+

```
$ ec2metadata --instance-id
```

- Obtain instance id from within Google Compute Engine instance

  ```
  $ gcemetadata --query instance --id
  ```

- Obtain instance name from within Microsoft Azure instance

```
$ azuremetadata --instance-name
```

After logging in through the Uyuni Server Web UI, **change** the default password.

Uyuni Proxy does not have administration access to the Web UI. It can be managed through its parent Uyuni Server.

Using Separate Storage Volume

We recommend that the repositories and the database for Uyuni be stored on a virtual storage device. This best practice will avoid data loss in cases where the Uyuni instance may need to be terminated. These steps **must** be performed **prior** to running the YaST Uyuni setup procedure.

1. Provision a disk device in the public cloud environment, refer to the cloud service provider documentation for detailed instructions. The size of the disk is dependent on the number of distributions and channels you intend to manage with Uyuni. For sizing information refer to SUSE Manager sizing examples. A rule of thumb is 25 GB per distribution per channel.
2. Once attached the device appears as Unix device node in your instance. For the following command to work this device node name is required. In many cases the attached storage appears as **/dev/sdb**. In order to check which disk devices exists on your system, call the following command:

   ```
   $ hwinfo --disk | grep -E "Device File:"
   ```

3. With the device name at hand the process of re-linking the directories in the filesystem Uyuni uses to store data is handled by the suma-storage script. In the following example we use `/dev/sdb` as the device name.

   ```
   $ /usr/bin/suma-storage /dev/sdb
   ```

   After the call all database and repository files used by SUSE Manager Server are moved to the newly created xfs based storage. In case your instance is a Uyuni Proxy, the script will move the Squid cache, which caches the software packages, to the newly created storage. The xfs partition is mounted below the path `/manager_storage`. .

4. Create an entry in /etc/fstab (optional)

   Different cloud frameworks treat the attachment of external storage devices differently at instance boot time. Please refer to the cloud environment documentation for guidance about the fstab entry.

   If your cloud framework recommends to add an fstab entry, add the following line to the **/etc/fstab** file.

   ```
   /dev/sdb1 /manager_storage xfs defaults,nofail 1 1
   ```

Registration of Cloned Systems

Uyuni cannot distinguish between different instances that use the same system ID. If you register a second instance with the same system ID as a previous instance, Uyuni will overwrite the original system data with the new system data. This can occur when you launch multiple instances from the same image, or when an image is created from a running instance. However, it is possible to clone systems and register them successfully by deleting the cloned system's ID, and generating a new ID.

Procedure: Registering Cloned Systems

1. Clone the system using your preferred hypervisor's cloning mechanism.
2. On the cloned system, change the hostname and IP addresses, and check the `/etc/hosts` file to ensure you have the right host entries.
3. On traditional clients, stop the `rhnsd` daemon with `/etc/init.d/rhnsd stop` or, on newer systemd-based systems, with `service rhnsd stop`. Then `service osad stop`.
4. For SLES 11 or Red Hat Enterprise Linux 5 or 6 clients, run these commands:

```
# rm /var/lib/dbus/machine-id
# dbus-uuidgen --ensure
```

5. For SLES 12 or Red Hat Enterprise Linux 7 clients, run these commands:

```
# rm /etc/machine-id
# rm /var/lib/dbus/machine-id
# dbus-uuidgen --ensure
# systemd-machine-id-setup
```

6. If you are using Salt, then you will also need to run these commands:

```
# service salt-minion stop
# rm -rf /var/cache/salt
```

7. If you are using a traditional client, clean up the working files with:

```
# rm -f /etc/sysconfig/rhn/{osad-auth.conf,systemid}
```

The bootstrap should now run with a new system ID, rather than a duplicate.

If you are onboarding Salt client clones, then you will also need to check if they have the same Salt minion ID. You will need to delete the minion ID on each cloned client, using the `rm` command. Each operating system type stores this file in a slightly different location, check the table for the appropriate command.

Minion ID File Location

Each operating system stores the minion ID file in a slightly different location, check the table for the appropriate command.

Operating System	Commands

SLES 12	`rm /etc/salt/minion_id`
	
	`rm -f /etc/zypp/credentials.d/{SCCcredentials,NCCcredentials}`

SLES 11	`rm /etc/salt/minion_id`
	
	`suse_register -E`

SLES 10	`rm -rf /etc/{zmd,zypp}`
	
	`rm -rf /var/lib/zypp/` Do not delete `/var/lib/zypp/db/products/`
	
	`rm -rf /var/lib/zmd/`

Red Hat Enterprise Linux 5, 6, 7	`rm -f /etc/NCCcredentials`

When you have deleted the minion ID file, re-run the bootstrap script, and restart the client to see the cloned system in Uyuni with the new ID.

Disconnected Setup

When it is not possible to connect Uyuni to the internet, you can use it within a disconnected environment.

The repository mirroring tool (RMT) is available on SUSE Linux Enterprise 15 and later. RMT replaces the subscription management tool (SMT), which can be used on older SUSE Linux Enterprise installations.

In a disconnected Uyuni setup, RMT or SMT uses an external network to connect to SUSE Customer Center. All software channels and repositories are synchronized to a removable storage device. The storage device can then be used to update the disconnected Uyuni installation.

This setup allows your Uyuni installation to remain in an offline, disconnected environment.

Your RMT or SMT instance must be used to managed a Uyuni Server directly. It cannot be used to manage a second RMT or SMT instance, in a cascade.

For more information on RMT, see https://documentation.suse.com/sles/15-SP1/html/SLES-all/book-rmt.html.

Synchronize RMT

You can use RMT on SUSE Linux Enterprise 15 installations to manage clients running SUSE Linux Enterprise 12 or later.

We recommend you set up a dedicated RMT instance for each Uyuni installation.

Procedure: Setting up RMT

1. On the RMT instance, install the RMT package:

```
zypper in rmt-server
```

2. Configure RMT using YaST:

```
yast2 rmt
```

3. Follow the prompts to complete installation. For more information on setting up RMT, see https://documentation.suse.com/sles/15-SP1/html/SLES-all/book-rmt.html.

Procedure: Synchronizing RMT with SCC

1. On the RMT instance, list all available products and repositories for your organization:

```
rmt-cli products list --all
rmt-cli repos list --all
```

2. Synchronize all available updates for your organization:

```
rmt-cli sync
```

   You can also configure RMT to synchronize regularly using systemd.

3. Enable the products you require. For example, to enable SLES 15:

```
rmt-cli product enable sles/15/x86_64
```

4. Export the synchronized data to your removable storage. In this example, the storage medium is mounted at `/mnt/usb`:

```
rmt-cli export data /mnt/usb
```

5. Export the enabled repositories to your removable storage:

```
rmt-cli export settings /mnt/usb
```

Ensure that the external storage is mounted to a directory that is writeable by the RMT user. You can change RMT user settings in the `cli` section of `/etc/rmt.conf`.

Synchronize SMT

SMT is included with SUSE Linux Enterprise 12, and can be used to manage clients running SUSE Linux Enterprise 10 or later.

SMT requires you to create a local mirror directory on the SMT instance in order to synchronize repositories and packages.

For more details on installing and configuring SMT, see https://documentation.suse.com/sles/12-SP4/html/SLES-all/book-smt.html.

Procedure: Synchronizing SMT with SCC

1. On the SMT instance, create a database replacement file:

```
smt-sync --createdbreplacementfile /tmp/dbrepl.xml
```

2. Export the synchronized data to your removable storage. In this example, the storage medium is mounted at `/mnt/usb`:

```
smt-sync --todir /mnt/usb
smt-mirror --dbreplfile /tmp/dbrepl.xml --directory /mnt/usb \
           --fromlocalsmt -L /var/log/smt/smt-mirror-export.log
```

3. Export the enabled repositories to your removable storage:

```
rmt-cli export settings /mnt/usb
```

Ensure that the external storage is mounted to a directory that is writeable by the RMT user. You can change SMT user settings in `/etc/smt.conf`.

Synchronize a Disconnected Server

When you have removable media loaded with your SUSE Customer Center data, you can use it to synchronize your disconnected server.

Procedure: Synchronizing a Disconnected Server

1. Mount your removable media device to the Uyuni server. In this example, the mount point is `/media/disk`.
2. Open `/etc/rhn/rhn.conf` and define the mount point by adding or editing this line:

```
server.susemanager.fromdir = /media/disk
```

3. Restart the Tomcat service:

```
systemctl restart tomcat
```

4. Refresh the local data:

```
mgr-sync refresh
```

5. Perform a synchronization:

```
mgr-sync list channels
mgr-sync add channel channel-label
```

The removable disk that you use for synchronization must always be available at the same mount point. Do not trigger a synchronization, if the storage medium is not mounted. This will result in data corruption.

Inter-Server Synchronization

If you have more than one Uyuni installation, you will probably want to ensure that they stay aligned on content and permissions. Inter-Server Synchronization (ISS) allows you to connect two or more Uyuni servers and keep them up-to-date.

To set up ISS, you need to define one Uyuni server as a master, with the other as a slave. If conflicting configurations exist, the system will prioritize the master configuration.

Procedure: Setting up an ISS Master

1. In the Uyuni Web UI, navigate to **Admin › ISS Configuration › Slave Setup**, and click **[Add new master]**.
2. In the Details for new Master dialog, provide these details for the server to use as the ISS master:
    - In the Master Fully-Qualified Domain Name field, enter the FQDN of the ISS master (for example: server1.example.com).
    - In the Filename of this Master's CA Certificate field, enter the absolute path to the CA certificate on the ISS master (for example: /etc/pki/trust/anchors/RHN-ORG-TRUSTED-SSL-CERT). Click **[Add new master]** to add the ISS master.

Procedure: Setting up an ISS Slave

1. In the Uyuni Web UI, navigate to **Admin › ISS Configuration › Master Setup**, and click **[Add new slave]**.
2. In the Edit Slave Details dialog, for the server to use as the ISS slave provide these details:
    - In the Slave Fully-Qualified Domain Name field, enter the FQDN of the ISS slave (for example: server2.example.com).
    - Check the Allow Slave to Sync? checkbox to enable the slave to synchronize with the master.
    - Check the Sync All Orgs to Slave? checkbox to synchronize all organizations to this slave.
3. Click **[Create]** to add the ISS slave.
4. In the Allow Export of the Selected Organizations section, check the organizations you want to allow this slave to export to the master, and click **[Allow Orgs]**.

When you have the master and slaves set up, you can perform a synchronization from the command line on the slave, with this command:

```
mgr-inter-sync
```

Set up a Client to Master Validation Fingerprint

In highly secure network configurations you may wish to ensure your Salt clients are connecting a specific master. To set up validation from client to master enter the master's fingerprint within the `/etc/salt/minion` configuration file.

See the following procedure:

1. On the master, at the command prompt, as root, use this command to find the `master.pub` fingerprint:

   ```
   salt-key -F master
   ```

   On your client, open the `/etc/salt/minion` configuration file. Uncomment the following line and enter the master's fingerprint replacing the example fingerprint:

   ```
   master_finger: 'ba:30:65:2a:d6:9e:20:4f:d8:b2:f3:a7:d4:65:11:13'
   ```

2. Restart the salt-minion service:

   ```
   # systemctl restart salt-minion
   ```

For information on configuring security from a client, see https://docs.saltstack.com/en/latest/ref/configuration/minion.html.

Signing Repository Metadata

You will require a custom GPG key to be able to sign repository metadata.

Procedure: Generating a Custom GPG Key

1. As the root user, use the `gpg` command to generate a new key:

```
gpg --gen-key
```

2. At the prompts, select `RSA` as the key type, with a size of 2048 bits, and select an appropriate expiry date for your key. Check the details for your new key, and type `y` to confirm.
3. At the prompts, enter a name and email address to be associated with your key. You can also add a comment to help you identify the key, if desired. When you are happy with the user identity, type `O` to confirm.
4. At the prompt, enter a passphrase to protect your key.
5. The key should be automatically added to your keyring. You can check by listing the keys in your keyring:

```
gpg --list-keys
```

6. Add the password for your keyring to the `/etc/rhn/signing.conf` configuration file, by opening the file in your text editor and adding this line:

```
GPGPASS="password"
```

You can manage metadata signing on the command line using the `mgr-sign-metadata-ctl` command.

Procedure: Enabling Metadata Signing

1. You will need to know the short identifier for the key to use. You can list your available public keys in short format:

```
gpg --keyid-format short --list-keys
...
pub   rsa2048/3E7BFE0A 2019-04-02 [SC] [expires: 2021-04-01]
      A43F9EC645ED838ED3014B035CFA51BF3E7BFE0A
uid         [ultimate] SUSE Manager
sub   rsa2048/118DE7FF 2019-04-02 [E] [expires: 2021-04-01]
```

2. Enable metadata signing with the `mgr-sign-metadata-ctl` command:

```
mgr-sign-metadata-ctl enable 3E7BFE0A
OK. Found key 3E7BFE0A in keyring.
DONE. Set key 3E7BFE0A in /etc/rhn/signing.conf.
DONE. Enabled metadata signing in /etc/rhn/rhn.conf.
DONE. Exported key 4E2C3DD8 to /srv/susemanager/salt/gpg/mgr-keyring.gpg.
DONE. Exported key 4E2C3DD8 to /srv/www/htdocs/pub/mgr-gpg-pub.key.
NOTE. For the changes to become effective run:
   mgr-sign-metadata-ctl regen-metadata
```

3. You can check that your configuration is correct with this command:

```
mgr-sign-metadata-ctl check-config
```

4. Restart the services and schedule metadata regeneration to pick up the changes:

```
mgr-sign-metadata-ctl regen-metadata
```

You can also use the `mgr-sign-metadata-ctl` command to perform other tasks. Use `mgr-sign-metadata-ctl --help` to see the complete list.

Repository metadata signing is a global option. When it is enabled, it is enabled on all software channels on the server. This means that all clients connected to the server will need to trust the new GPG key to be able to install or update packages.

Procedure: Importing GPG keys on Clients

1. For RPM-based client systems, use these remote commands:

```
rpm --import http://server.example.com/pub/mgr-gpg-pub.key
```

2. For Ubuntu clients, you will need to reassign the channels, which will automatically pick up the new GPG key. You can do this through the Uyuni Web UI, or from the command line on the server with this command:

```
salt <ubuntu-client> state.apply channels
```

3. OPTIONAL: For Salt clients, you might prefer to use a state to manage your GPG keys.

Mirror Source Packages

If you build your own packages locally, or if you require the source code for your packages for legal reasons, it is possible to mirror the source packages on Uyuni Server.

Mirroring source packages can consume a significant amount of disk space.

Procedure: Mirroring Source Packages

1. Open the `/etc/rhn/rhn.conf` configuration file, and add this line:

```
server.sync_source_packages = 1
```

2. Restart the Spacewalk service to pick up the changes:

```
spacewalk-service restart
```

Currently, this feature can only be enabled globally for all repositories. It is not possible to select individual repositories for mirroring.

When this feature has been activated, the source packages will become available in the Uyuni Web UI after the next repository synchronization. They will be shown as sources for the binary package, and can be downloaded directly from the Web UI. Source packages cannot be installed on clients using the Web UI.

Authentication Methods

Uyuni supports several different authentication methods. This section discusses pluggable authentication modules (PAM) and single sign-on (SSO).

Authenticate with PAM

Uyuni supports network-based authentication systems using pluggable authentication modules (PAM). PAM is a suite of libraries that allows you to integrate Uyuni with a centralized authentication mechanism, eliminating the need to remember multiple passwords. Uyuni supports LDAP, Kerberos, and other network-based authentication systems using PAM.

Procedure: Enabling PAM

1. Create a PAM service file at `/etc/pam.d/susemanager`. A standard `/etc/pam.d/susemanager` file should look like this. It configures Uyuni to use the system wide PAM configuration:

```
#%PAM-1.0
auth     include        common-auth
account  include        common-account
password include        common-password
session  include        common-session
```

2. Enforce the use of the service file by adding this line to `/etc/rhn/rhn.conf`:

```
pam_auth_service = susemanager
```

   In this example, the PAM service file is called `susemanager`.

3. Restart the Uyuni services after a configuration change.
4. In the Uyuni Web UI, navigate to `Create User` and enable a new or existing user to authenticate with PAM.
5. Check the `Pluggable Authentication Modules (PAM)` checkbox. It is below the password and password confirmation fields.

Changing the password in the Uyuni Web UI changes only the local password on the Uyuni Server. If PAM is enabled for that user, the local password might not be used at all. In the above example, for instance, the Kerberos password will not be changed. Use the password change mechanism of your network service to change the password for these users.

To configure system-wide authentication you can use YaST. You will need to install the `yast2-ldap-client` and `yast2-kerberos-client` packages.

For more information about configuring PAM, the SUSE Linux Enterprise Server Security Guide contains a generic example that will also work for other network-based authentication methods. It also describes how to configure an Active Directory Service. For more information, see https://documentation.suse.com/sles/15-SP1/html/SLES-all/part-auth.html.

Authenticate with Single Sign-On (SSO)

This feature is provided as a technical preview. It is not supported for use in production environments.

Uyuni supports single sign-on (SSO) by implementing the Security Assertion Markup Language (SAML) 2 protocol.

Single sign-on is an authentication process that allows a user to access multiple applications with one set of credentials. SAML is an XML-based standard for exchanging authentication and authorization data. A SAML identity service provider (IdP) provides authentication and authorization services to service providers (SP), such as Uyuni. Uyuni exposes three endpoints which must be enabled for single sign-on.

SSO in Uyuni supports:

- Log in with SSO.
- Log out with service provider-initiated single logout (SLO), and Identity service provider single logout service (SLS).
- Assertion and nameId encryption.
- Assertion signatures.
- Message signatures with AuthNRequest, LogoutRequest, and LogoutResponses.
- Enable an Assertion consumer service endpoint.
- Enable a single logout service endpoint.
- Publish the SP metadata (which can be signed).

SSO in Uyuni does not support:

- Product choosing and implementation for the Identity Service Provider (IdP).
- SAML support for other products (check with the respective product documentation).

Prerequisites

Before you begin, you will need to have configured an external Identity Service Provider with these parameters. Check your IdP documentation for instructions.

You will need these endpoints:

- Assertion Consumer Service (or ACS): an endpoint to accept SAML messages to establish a session

into the Service Provider. The endpoint for ACS in Uyuni is: https://example.com/rhn/manager/sso/acs

- Single Logout Service (or SLS): an endpoint to initiate a logout request from the IdP. The endpoint for SLS in Uyuni is: https://example.com/rhn/manager/sso/sls
- Metadata: an endpoint to retrieve Uyuni metadata for SAML. The endpoint for Metadata in Uyuni is: https://example.com/rhn/manager/sso/metadata

Your IdP must have a SAML:Attribute containing the username of the IdP user domain, called `uid`. The `uid` attribute passed in the SAML:Attribute must be created in the Uyuni user base before you activate single sign-on.

After the authentication with the IdP using the user `orgadmin` is successful, you will be logged in into Uyuni as the `orgadmin` user, provided that the `orgadmin` user exists in Uyuni.

Enable SSO

Using SSO is mutually exclusive with other types of authentication: it is either enabled or disabled. SSO is disabled by default.

Procedure: Enabling SSO

1. If your users do not yet exist in Uyuni, create them first.
2. Edit `/etc/rhn/rhn.conf` and add this line at the end of the file:

```
java.sso = true
```

3. Find the parameters you want to customize in `/usr/share/rhn/config-defaults/rhn_java_sso.conf`. Insert the parameters you want to customize into `/etc/rhn/rhn.conf` and prefix them with `java.sso.`.

   For example, in `/usr/share/rhn/config-defaults/rhn_java_sso.conf` find:

```
onelogin.saml2.sp.assertion_consumer_service.url = https://YOUR-PRODUCT-HOSTNAME-OR-
IP/rhn/manager/sso/acs
```

   In order to customize it, create the corresponding option in `/etc/rhn/rhn.conf` by prefixing the option name with `java.sso.`:

```
java.sso.onelogin.saml2.sp.assertion_consumer_service.url = https://YOUR-PRODUCT-
HOSTNAME-OR-IP/rhn/manager/sso/acs
```

   To find all the occurrences you need to change, search in the file for the placeholders `YOUR-PRODUCT` and `YOUR-IDP-ENTITY`. Every parameter comes with a brief explanation of what it is

meant for.

4. Restart the spacewalk service to pick up the changes:

```
spacewalk-service restart
```

When you visit the Uyuni URL, you will be redirected to the IdP for SSO where you will be requested to authenticate. Upon successful authentication, you will be redirected to the Uyuni Web UI, logged in as the authenticated user. If you encounter problems with logging in using SSO, check the Uyuni logs for more information.

Custom SSL Certificates

You can use custom certificates with Uyuni and Uyuni Proxy.

This section covers how to use a third party SSL certificate authority with a new Uyuni installation, and replacing existing certificate with new custom certificates.

Before you begin, ensure you have:

- A certificate authority (CA) SSL public certificate
- An SSL server key
- An SSL server certificate

Your key and certificate files must be in PEM format.

The host name of the SSL keys and certificates must match the fully qualified host name of the machine you deploy them on. You can set the host names in the `X509v3 Subject Alternative Name` section of the certificate. You can also list multiple host names if your environment requires it.

If you want to use intermediate certificates, you need to merge the intermediate and root CA certificates into one file. Ensure that the intermediate certificate comes first in the combined file.

Custom Certificates for New Installations

By default, Uyuni uses a self-signed certificate. After you have completed the initial setup, you can replace the default certificate with a custom certificate.

Procedure: Installing Custom Certificates on a New Uyuni Server

1. Install the Uyuni Server according to the instructions in **[Installation › Install-intro ›]**.
2. Complete the initial setup according to **[Installation › Server-setup ›]**.
3. At the command prompt, point the SSL environment variables to the custom certificate file locations:

```
export CA_CERT=<path_to_CA_certificate_file>
export SERVER_KEY=<path_to_web_server_key>
export SERVER_CERT=<path_to_web_server_certificate>
```

4. Complete Uyuni setup:

```
yast2 susemanagersetup
```

   When you are prompted for certificate details during setup, fill in random values. The values will be overridden by the values you specified at the command prompt.

Execute the `yast2 susemanagersetup` command from the same shell you exported the environment variables from.

Custom Certificates for New Proxy Installations

By default, Uyuni Proxy uses a self-signed certificate. After you have completed the initial setup, you can replace the default certificate with a custom certificate.

Procedure: Installing Custom Certificates on a New Uyuni Proxy

1. Install the Uyuni Proxy according to the instructions in **[Installation › Install-intro ›]**.
2. Complete the initial setup according to **[Installation › Proxy-setup ›]**.
3. At the command prompt, run:

```
configure-proxy.sh
```

4. At the `Do you want to import existing certificates?` prompt, type *y*.
5. Follow the prompts to complete setup.

Re-Create Existing Server Certificates

If your existing custom certificates have expired or stopped working for any reason, you can generate a new server certificate from the existing CA.

Procedure: Re-Creating an Existing Server Certificate

1. On the Uyuni Server, at the command prompt, regenerate the server certificate:

```
rhn-ssl-tool --gen-server --dir="/root/ssl-build" --set-country="COUNTRY" \
--set-state="STATE" --set-city="CITY" --set-org="ORGANIZATION" \
--set-org-unit="ORGANIZATION UNIT" --set-email="name@example.com" \
--set-hostname="susemanager.example.top" --set-cname="example.com"
```

   Ensure that the `set-cname` parameter is the fully-qualified domain name of your Uyuni Server. You can use the the `set-cname` parameter multiple times if you require multiple aliases.

2. Install the RPM that contains the newly generated certificate. Check that you have the latest version of the RPM before running this command. The version number is incremented every time you re-create the certificates.

```
rpm -Uhv /root/ssl-build/lnx0259a/rhn-org-httpd-ssl-key-pair-lnx0259a-1.0-2.noarch.rpm
```

3. Restart services to pick up the changes:

```
spacewalk-service restart
```

Create and Replace CA and Server Certificates

If you need to create entirely new certificates for an existing installation, you need to create a combined certificate first. Clients will authenticate to the certificate with both the old and new details. Then you can go ahead and remove the old details. This maintains the chain of trust.

Be careful with this procedure! It is possible to break the trust chain between the server and clients using this procedure. If that happens, you will need an administrative user to log in to every client and deploy the CA directly.

Procedure: Creating New Certificates

1. On the Uyuni Server, at the command prompt, move the old certificate directory to a new location:

```
mv /root/ssl-build /root/old-ssl-build
```

2. Generate a new CA certificate and create an RPM:

```
rhn-ssl-tool --gen-ca --dir="/root/ssl-build" --set-country="COUNTRY" \
--set-state="STATE" --set-city="CITY" --set-org="ORGANIZATION" \
--set-org-unit="ORGANIZATION UNIT" --set-common-name="SUSE Manager CA Certificate" \
--set-email="name@example.com"
```

3. Generate a new server certificate and create an RPM:

```
rhn-ssl-tool --gen-server --dir="/root/ssl-build" --set-country="COUNTRY" \
--set-state="STATE" --set-city="CITY" --set-org="ORGANIZATION" \
--set-org-unit="ORGANIZATION UNIT" --set-email="name@example.com" \
--set-hostname="susemanager.example.top" --set-cname="example.com"
```

   Ensure that the `set-cname` parameter is the fully-qualified domain name of your Uyuni Server. You can use the the `set-cname` parameter multiple times if you require multiple aliases.

   You will need to generate a server certificate RPM for each proxy, using their host names and cnames.

When you have new certificates, you can create the combined RPMs to authenticate the clients.

Procedure: Create Combined Certificate RPMs

1. Create a new CA file that combines the old and new certificate details, and generate a new RPM:

```
mkdir /root/combined-ssl-build
cp /root/old-ssl-build/RHN-ORG-TRUSTED-SSL-CERT /root/combined-ssl-build/
cat /root/ssl-build/RHN-ORG-TRUSTED-SSL-CERT >> /root/combined-ssl-build/RHN-ORG-
TRUSTED-SSL-CERT
cp /root/old-ssl-build/*.rpm /root/combined-ssl-build/
rhn-ssl-tool --gen-ca --rpm-only --dir="/root/combined-ssl-build"
```

2. Deploy the CA certificate on the server:

```
/usr/bin/rhn-deploy-ca-cert.pl --source-dir /root/combined-ssl-build \
--target-dir /srv/www/htdocs/pub/ --trust-dir=/etc/pki/trust/anchors/
```

When you have the combined RPMs, you can deploy the combined CA certificates to your clients.

Procedure: Deploying Combined Certificates on Traditional Clients

1. On the client, create a new custom channel using these details:
   - Name: SSL-CA-Channel
   - Label: ssl-ca-channel
   - Parent Channel: <choose the parent channel of a clients>
   - Summary: SSL-CA-Channel

   For more on creating custom channels, see **[Administration › Channel-management ›]**.
2. Upload the CA certificate RPM to the channel:

```
rhnpush -c ssl-ca-channel --nosig \
--ca-chain=/srv/www/htdocs/pub/RHN-ORG-TRUSTED-SSL-CERT \
/root/combined-ssl-build/rhn-org-trusted-ssl-cert-1.0-2.noarch.rpm
```

3. Subscribe all clients to the new `SSL-CA-Channel` channel.
4. Install the CA certificate RPM on all clients by updating the channel.

Procedure: Deploying Combined Certificates on Salt Clients

1. In the Uyuni Web UI, navigate to **Systems › Overview**.
2. Check all your Salt Clients to add them to the System Set Manager (SSM).
3. Navigate to **Systems › System Set Manager › Overview**.
4. In the `States` field, click **`[Apply]`** to apply the system states.
5. In the `Highstate` page, click **`[Apply Highstate]`** to propagate the changes to the clients.

When you have every client trusting both the old and new certificates, you can go ahead and replace the server certificate on the Uyuni Server and Proxies.

Procedure: Replace Server Certificate on the Server

1. On the Uyuni Server, at the command prompt, install the RPM from the `ssl-build` directory:

```
rpm -Uhv ssl-build/susemanager/rhn-org-httpd-ssl-key-pair-susemanager-1.0-2.noarch.rpm
```

2. Restart services to pick the changes:

```
spacewalk-service restart
```

Procedure: Replace Server Certificate on the Proxy

1. On the Uyuni Proxy, at the command prompt, install the RPM from the `ssl-build` directory:

```
rpm -Uhv ssl-build/susemanager-proxy/rhn-org-httpd-ssl-key-pair-susemanager-proxy-1.0-
2.noarch.rpm
```

2. Restart services to pick up the changes:

```
rhn-proxy restart
```

3. Test that all clients still operate as expected and can use SSL to reach the Uyuni Server and any proxies.

When you have replaced the server certificates on your server and any proxies, you need to update the certificate with only the new details on all the clients. This is done by adding it to the client channels you set up previously.

Procedure: Adding the New Certificates to the Client Channel

1. Copy the combined certificate RPM into the `/root/ssl-build/` directory:

```
cp /root/combined-ssl-build/*.rpm /root/ssl-build/
```

2. Generate a new RPM with from the new certificates. Check the release number carefully to ensure you have the right certificate file:

```
rhn-ssl-tool --gen-ca --rpm-only --dir="/root/ssl-build"
```

3. Install the new local certificates on the Uyuni Server:

```
/usr/bin/rhn-deploy-ca-cert.pl --source-dir /root/ssl-build \
--target-dir /srv/www/htdocs/pub/ --trust-dir=/etc/pki/trust/anchors/
```

4. Restart services to pick up the changes:

```
spacewalk-service restart
```

5. Upload the new RPM into the channel:

```
rhnpush -c ssl-ca-channel --nosig \
--ca-chain=/srv/www/htdocs/pub/RHN-ORG-TRUSTED-SSL-CERT \
/root/ssl-build/rhn-org-trusted-ssl-cert-1.0-3.noarch.rpm
```

When you have the new certificate in the channel, you can use the Uyuni Web UI to update it on all clients and proxies, by synchronizing them with the channel. Alternatively, for Salt clients, you can use **Salt › Remote Commands**, or apply the highstate.

You will also need to update your proxies to remove the copy of the certificate and the associated RPM. Your proxies must have the same certificate content as the server. Check the `/srv/www/htdocs/pub/` directory and ensure it contains:

```
RHN-ORG-TRUSTED-SSL-CERT
rhn-org-trusted-ssl-cert-*.noarch.rpm
```

To complete the process, you need to update the database with this command:

```
/usr/bin/rhn-ssl-dbstore --ca-cert=/root/ssl-build/RHN-ORG-TRUSTED-SSL-CERT
```

If you use bootstrap, remember to also update your bootstrap scripts to reflect the new certificate information.

Backup and Restore

Back up your Uyuni installation regularly, in order to prevent data loss. Because Uyuni relies on a database as well as the installed program and configurations, it is important to back up all components of your installation. This chapter contains information on the files you need to back up, and introduces the `smdba` tool to manage database backups. It also contains information about restoring from your backups in the case of a system failure.

Backup Space Requirements

Regardless of the backup method you use, you must have available at least three times the amount of space your current installation uses. Running out of space can result in backups failing, so check this often.

Backing up Uyuni

The most comprehensive method for backing up your Uyuni installation is to back up the relevant files and directories. This can save you time in administering your backup, and can be faster to reinstall and re-synchronize in the case of failure. However, this method requires significant disk space and could take a long time to perform the backup.

If you want to only back up the required files and directories, use the following list. To make this process simpler, and more comprehensive, we recommend backing up the entire `/etc` and `/root` directories, not just the ones specified here. Some files only exist if you are actually using the related SUSE Manager feature.

- `/etc/cobbler/`
- `/etc/dhcp.conf`
- `/etc/fstab` and any ISO mountpoints you require.
- `/etc/rhn/`
- `/etc/salt`
- `/etc/sudoers`
- `/etc/sysconfig/rhn/`
- `/root/.gnupg/`
- `/root/.ssh`

  This file exists if you are using an SSH tunnel or SSH `push`. You will also need to have saved a copy of the `id-susemanager` key.

- `/root/ssl-build/`

- /srv/formula_metadata
- /srv/pillar
- /srv/salt
- /srv/susemanager
- /srv/tftpboot/
- /srv/www/cobbler
- /srv/www/htdocs/pub/
- /srv/www/os-images
- /var/cache/rhn
- /var/cache/salt
- /var/lib/cobbler/
- /var/lib/cobbler/templates/ (before version 4.0 it was /var/lib/rhn/kickstarts/)
- /var/lib/Kiwi
- /var/lib/rhn/
- /var/spacewalk/
- Plus any directories containing custom data such as scripts, Kickstart or AutoYaST profiles, and custom RPMs.

You will also need to back up your database, which you can do with the smdba tool. For more information about the smdba tool, see [**Administration** › **Backup-restore** ›].

Procedure: Restore from a Manual Backup

1. Re-install Uyuni. For more information about recovering from a backup, see [**Administration** › **Backup-restore** ›].
2. Re-synchronize your Uyuni repositories with the mgr-sync tool. For more information about the mgr-sync tool, see [syncing.suse.mgr.repositories.scc].
3. You can choose to re-register your product, or skip the registration and SSL certificate generation sections.
4. Re-install the /root/ssl-build/rhn-org-httpd-ssl-key-pair-MACHINE_NAME-VER-REL.noarch.rpm package.
5. Schedule the re-creation of search indexes next time the rhn-search service is started:

```
rhn-search cleanindex
```

This command produces only debug messages. It does not produce error messages.

6. Check whether you need to restore `/var/spacewalk/packages/`. If `/var/spacewalk/packages/` was not in your backup, you will need to restore it. If the source repository is available, you can restore `/var/spacewalk/packages/` with a complete channel synchronization:

```
mgr-sync refresh --refresh-channels
```

Check the progress by running `tail -f /var/log/rhn/reposync/`<CHANNEL_NAME>`.log` as *root*.

Administering the Database with smdba

The `smdba` tool is used for managing a local PostgreSQL database. It allows you to back up and restore your database, and manage backups. It can also be used to check the status of your database, and perform administration tasks, such as restarting.

The `smdba` tool works with local PostgreSQL databases only, it will not work with remotely accessed databases, or Oracle databases.

The `smdba` tool requires `sudo` access, in order to execute system changes. Ensure you have enabled `sudo` access for the `admin` user before you begin, by checking the `/etc/sudoers` file for this line:

```
admin   ALL=(postgres) /usr/bin/smdba
```

Check the runtime status of your database with:

```
smdba db-status
```

This command will return either `online` or `offline`, for example:

```
Checking database core...       online
```

Starting and stopping the database can be performed with:

```
smdba db-start
```

And:

```
smdba db-stop
```

Database Backup with smdba

The `smdba` tool performs a continuous archiving backup. This backup method combines a log of every change made to the database during the current session, with a series of more traditional backup files. When a crash occurs, the database state is first restored from the most recent backup file on disk, then the log of the current session is replayed exactly, to bring the database back to a current state. A continuous archiving backup with `smdba` is performed with the database running, so there is no need for downtime.

This method of backing up is stable and generally creates consistent snapshots, however it can take up a lot of storage space. Ensure you have at least three times the current database size of space available for backups. You can check your current database size by navigating to `/var/lib/pgsql/` and running `df -h`.

The `smdba` tool also manages your archives, keeping only the most recent backup, and the current archive of logs. The log files can only be a maximum file size of 16 MB, so a new log file will be created when the files reach this size. Every time you create a new backup, previous backups will be purged to release disk space. We recommend you use `cron` to schedule your `smdba` backups to ensure that your storage is managed effectively, and you always have a backup ready in case of failure.

Performing a Manual Database Backup

The `smdba` tool can be run directly from the command line. We recommend you run a manual database backup immediately after installation, or if you have made any significant changes to your configuration.

When `smdba` is run for the first time, or if you have changed the location of the backup, it will need to restart your database before performing the archive. This will result in a small amount of downtime. Regular database backups will not require any downtime.

Procedure: Performing a Manual Database Backup

1. Allocate permanent storage space for your backup. This example uses a directory located at `/var/spacewalk/`. This will become a permanent target for your backup, so ensure it will remain accessible by your server at all times.
2. In your backup location, create a directory for the backup:

   ```
   sudo -u postgres mkdir /var/spacewalk/db-backup
   ```

   Or, as root:

   ```
   install -d -o postgres -g postgres -m 700 /var/spacewalk/db-backup
   ```

3. Ensure you have the correct permissions set on the backup location:

```
chown postgres:postgres /var/spacewalk/db-backup
```

4. To create a backup for the first time, run the `smdba backup-hot` command with the `enable` option set. This will create the backup in the specified directory, and, if necessary, restart the database:

```
smdba backup-hot --enable=on --backup-dir=/var/spacewalk/db-backup
```

   This command produces debug messages and finishes sucessfully with the output:

```
INFO: Finished
```

5. Check that the backup files exist in the `/var/spacewalk/db-backup` directory, to ensure that your backup has been successful.

Scheduling Automatic Backups

You do not need to shut down your system in order to perform a database backup with `smdba`. However, because it is a large operation, database performance can slow down while the backup is running. We recommend you schedule regular database backups for a low-traffic period, to minimize disruption.

Ensure you have at least three times the current database size of space available for backups. You can check your current database size by navigating to `/var/lib/pgsql/` and running `df -h`.

Procedure: Scheduling Automatic Backups

1. Create a directory for the backup, and set the appropriate permissions (as root):

```
install -m 700 -o postgres -g postgres /var/spacewalk/db-backup
```

2. Open `/etc/cron.d/db-backup-mgr`, or create it if it does not exist, and add the following line to create the cron job:

```
0 2 * * * root /usr/bin/smdba backup-hot --enable=on --backup-dir=/var/spacewalk/db
-backup
```

3. Check the backup directory regularly to ensure the backups are working as expected.

Restoring from Backup

The `smdba` tool can be used to restore from backup in the case of failure.

Procedure: Restoring from Backup

1. Shut down the database:

```
smdba db-stop
```

2. Start the restore process and wait for it to complete:

```
smdba backup-restore start
```

3. Restart the database:

```
smdba db-start
```

4. Check if there are differences between the RPMs and the database.

```
spacewalk-data-fsck
```

Archive Log Settings

Archive logging allows the database management tool `smdba` to perform hot backups. In Uyuni with an embedded database, archive logging is enabled by default.

PostgreSQL maintains a limited number of archive logs. Using the default configuration, approximately 64 files with a size of 16 MiB are stored.

Creating a user and syncing the channels:

- SLES12-SP2-Pool-x86_64
- SLES12-SP2-Updates-x86_64
- SLE-Manager-Tools12-Pool-x86_64-SP2
- SLE-Manager-Tools12-Updates-x86_64-SP2

PostgreSQL will generate an additional roughly 1 GB of data. So it is important to think about a backup strategy and create a backups in a regular way.

Archive logs are stored at `/var/lib/pgsql/data/pg_xlog/` (postgresql).

Retrieving an Overview of Occupied Database Space

Database administrators may use the subcommand `space-overview` to get a report about occupied table spaces, for example:

```
smdba space-overview
```

outputs:

```
SUSE Manager Database Control. Version 1.5.2
Copyright (c) 2012 by SUSE Linux Products GmbH


Tablespace  | Size (Mb) | Avail (Mb) | Use %
------------+-----------+------------+------
postgres    | 7         | 49168      | 0.013
susemanager | 776       | 48399      | 1.602
```

The `smdba` command is available for PostgreSQL. For a more detailed report, use the `space-tables` subcommand. It lists the table and its size, for example:

```
smdba space-tables
```

outputs:

```
SUSE Manager Database Control. Version 1.5.2
Copyright (c) 2012 by SUSE Linux Products GmbH


Table                                | Size
-------------------------------------+-----------
public.all_primary_keys              | 0 bytes
public.all_tab_columns               | 0 bytes
public.allserverkeywordsincereboot   | 0 bytes
public.dblink_pkey_results           | 0 bytes
public.dual                          | 8192 bytes
public.evr_t                         | 0 bytes
public.log                           | 32 kB
...
```

Moving the Database

It is possible to move the database to another location. For example, move the database if the database storage space is running low. The following procedure will guide you through moving the database to a new location for use by Uyuni.

Procedure: Moving the Database

1. The default storage location for Uyuni is `/var/lib/pgsql/`. If you would like to move it, for example to `/storage/postgres/`, proceed as follows.

2. Stop the running database with (as root):

```
rcpostgresql stop
```

Shut down the running Spacewalk services with:

```
spacewalk-service stop
```

3. Copy the current working directory structure with `cp` using the `-a, --archive` option. For example:

```
cp --archive /var/lib/pgsql/ /storage/postgres/
```

This command will copy the contents of `/var/lib/pgsql/` to `/storage/postgres/pgsql/`.

The contents of the `/var/lib/pgsql` directory needs to remain the same, otherwise the Uyuni database may malfunction. You also should ensure that there is enough available disk space.

4. Mount the new database directory with:

```
mount /storage/postgres/pgsql
```

5. Make sure ownership is `postgres:postgres` and not `root:root` by changing to the new directory and running the following commands:

```
cd /storage/postgres/pgsql/
ls -l
```

Outputs:

```
total 8
drwxr-x---  4 postgres postgres   47 Jun  2 14:35 ./
```

6. Add the new database mount location to your servers fstab by editing `etc/fstab`.
7. Start the database with:

```
rcpostgresql start
```

8. Start the Spacewalk services with:

```
spacewalk-service start
```

Recovering from a Crashed Root Partition

This section provides guidance on restoring your server after its root partition has crashed. This section assumes you have setup your server similar to the procedure explained in Installation guide with separate partitions for the database and for channels mounted at `/var/lib/pgsql` and `/var/spacewalk/`.

Procedure: Recovering from a Crashed Root Partition

1. Install Uyuni. Do not mount the `/var/spacewalk` and `/var/lib/pgsql` partitions. Wait for the installation to complete before going on to the next step.
2. Shut down the services with `spacewalk-service shutdown`.
3. Shut down the database with `rcpostgresql stop`.
4. Mount `/var/spacewalk` and `/var/lib/pgsql` partitions.
5. Restore the directories listed in Backing up Uyuni.
6. Start the Spacewalk services with `spacewalk-services start`.
7. Start the database with `rcpostgresql start`.

Uyuni should now operate normally without loss of your database or synced channels.

Database Connection Information

The information for connecting to the Uyuni database is located in `/etc/rhn/rhn.conf`:

```
db_backend = postgresql
db_user = susemanager
db_password = susemanager
db_name = susemanager
db_host = localhost
db_port = 5432
db_ssl_enabled =
```

Managing Disk Space

Running out of disk space can have a severe impact on the Uyuni database and file structure which, in some cases, is not recoverable.

Uyuni monitors some directories for free disk space. You can modify which directories are monitored, and the warnings that are created. All settings are configured in the `/etc/rhn/rhn.conf` configuration file.

Monitored Directories

By default, Uyuni monitors these directories:

- `/var/lib/pgsql`
- `/var/spacewalk`
- `/var/cache`
- `/srv`

You can change which directories are monitored with the `spacecheck_dirs` parameter. You can specify multiple directories by separating them with a space.

For example:

```
spacecheck_dirs = /var/lib/pgsql /var/spacewalk /var/cache /srv
```

Thresholds

By default, Uyuni will create a warning mail when a monitored directory has less than 10% of total space available. A critical alert is created when a monitored directory falls below 5% space available.

You can change these alert thresholds with the `spacecheck_free_alert` and `spacecheck_free_critical` parameters.

For example:

```
spacecheck_free_alert = 10
spacecheck_free_critical = 5
```

Shut Down Services

By default, Uyuni will shut down the spacewalk services when the critical alert threshold is reached.

You can change this behavior with the `spacecheck_shutdown` parameter. A value of `true` will enable

the shut down feature. Any other value will disable it.

For example:

```
spacecheck_shutdown = true
```

Disable Space Checking

The space checking tool is enabled by default. You can disable it entirely with these commands:

```
systemctl stop spacewalk-diskcheck.timer
systemctl disable spacewalk-diskcheck.timer
```

Content Lifecycle Management

Content Lifecycle Management allows you to customize and test packages before updating production systems. This is especially useful if you need to apply updates during a limited maintenance window.

Content Lifecycle Management allows you to select software channels as sources, adjust them as required for your environment, and thoroughly test them before installing onto your production systems.

While you cannot directly modify vendor channels, you can clone them and then modify the clones by adding or removing packages and custom patches. You can then assign these cloned channels to test systems to ensure they work as expected. Then, when all tests pass, you can promote them to production servers.

This is achieved through a series of environments that your software channels can move through on their lifecycle. Most environment lifecycles include at least test and production environments, but you can have as many environments as you require.

This section covers the basic content lifecycle procedures, and the filters available. For more specific examples, see **[Administration › Content-lifecycle-examples ›]**.

Create a Content Lifecycle Project

1. In the Uyuni Web UI, navigate to **Content Lifecycle › Projects**, and click **[Create Project]**.
2. In the `Label` field, enter a label for your project. The `Label` field only accepts lowercase letters, numbers, periods (`.`), hyphens (`-`) and underscores (`_`).
3. In the `Name` field, enter a descriptive name for your project.
4. Click the **[Create]** button to create your project and return to the project page.
5. Click **[Attach/Detach Sources]**.
6. In the `Sources` dialog, select the source type, and select a base channel for your project. The available child channels for the selected base channel are displayed, including information on whether the channel is mandatory or recommended.
7. Check the child channels you require, and click **[Save]** to return to the project page. The software channels you selected should now be showing.
8. Click **[Attach/Detach Filters]**.
9. In the `Filters` dialog, select the filters you want to attach to the project. To create a new filter, click **[Create new Filter]**.
10. Click **[Add Environment]**.
11. In the `Environment Lifecycle` dialog, give the first environment a name and a description, and click **[Save]**. The `Name` field only accepts lowercase letters, numbers, periods (`.`), hyphens (`-`) and underscores (`_`).
12. Continue creating environments until you have all the environments for your lifecycle completed.

You can select the order of the environments in the lifecycle by selecting an environment in the `Insert before` field when you create it.

Filter Types

Uyuni allows you to create various types of filters to control the content on project build. This is the list of supported filters:

- package filtering
  - by name
  - by name, epoch, version, release and architecture
- patch filtering
  - by advisory name
  - by advisory type
  - by synopsis
  - by keyword
  - by date
  - by affected package

Package dependencies are not resolved during content filtering.

There are multiple matchers you can use with the filter. Which ones are available will depend on the filter type you choose. The full list is:

- contains
- matches (must take the form of a regular expression)
- equals
- greater
- greater or equal
- lower or equal
- lower
- later or equal

Filter `rule` Parameter

Moreover, each filter has a `rule` parameter set to either `Allow` or `Deny`. The filters are processed:

- If a package or patch satisfies a `Deny` filter, it will be excluded from the result.

- If a package or patch satisfies an `Allow` filter, it will be included in the result (even if it was excluded by a `Deny` filter).

This behavior is useful when you want to exclude large number of packages or patches using a general `Deny` filter and "cherry-pick" specific packages or patches with specific `Allow` filters.

Content filters are global in your organization and can be shared between projects.

If your project already contains built sources, when you add an environment it will automatically populate with the existing content. Content will be drawn from the previous environment of the cycle if it had one. If there is no previous environment, it will be left empty until the project sources are built again.

Build a Content Lifecycle Project

When you have created your project, defined environments, and attached sources and filters, you can build the project for the first time.

Building applies filters to the attached sources and clones them to the first environment in the project.

Procedure: Building a Content Lifecycle Project

1. In the Uyuni Web UI, navigate to **Content Lifecycle › Projects**, and select the project you want to build.
2. Review the attached sources and filters, and click **`[Build]`**.
3. You can monitor build progress in the `Environment Lifecycle` section.

After the build is finished, the environment version is increased by one and the built sources, such as software channels, can be assigned to your systems.

Promote Environments

When the project has been built, the built sources can be sequentially promoted to the environments.

Procedure: Promoting Environments

1. In the Uyuni Web UI, navigate to **Content Lifecycle › Projects**, and select the project you want to work with.
2. In the `Environment Lifecycle` section, locate the environment to promote to its successor, and click **`[Promote]`**.
3. You can monitor build progress in the `Environment Lifecycle` section.

Assign Systems to Environments

When you build and promote content lifecycle projects, it creates a tree of software channels. To add systems to the environment, assign the base and child software channels to your system using **Software › Software Channels** in the `System Details` page for the system.

Newly added cloned channels are not assigned to systems automatically. If you add or promote sources you will need to manually check and update your channel assignments.

Automatic assignment is intended to be added to Uyuni in a future version.

Generate Reports

The `spacewalk-report` command is used to produce a variety of reports. These reports can be helpful for taking inventory of your subscribed systems, users, and organizations. Using reports is often simpler than gathering information manually from the SUSE Manager Web UI, especially if you have many systems under management.

To generate reports, you must have the `spacewalk-reports` package installed.

The `spacewalk-report` command allows you to organize and display reports about content, systems, and user resources across Uyuni.

You can generate reports on:

1. System Inventory: list all the systems registered to Uyuni.
2. Patches: list all the patches relevant to the registered systems. You can sort patches by severity, as well as the systems that apply to a particular patch.
3. Users: list all registered users and any systems associated with a particular user.

To get the report in CSV format, run this command at the command prompt on the server:

```
spacewalk-report <report_name>
```

This table lists the available reports:

Table 2. `spacewalk-report` *Reports*

Report	Invoked as	Description
Actions	`actions`	All actions.
Activation Keys	`activation-keys`	All activation keys, and the entitlements, channels, configuration channels, system groups, and packages associated with them.
Activation Keys: Channels	`activation-keys-channels`	All activation keys and the entities associated with each key.
Activation Keys: Configuration	`activation-keys-config`	All activation keys and the configuration channels associated with each key.
Activation Keys: Server Groups	`activation-keys-groups`	All activation keys and the system groups associated with each key.
Activation Keys: Packages	`activation-keys-packages`	All activation keys and the packages each key can deploy.

Report	Invoked as	Description
Channel Packages	`channel-packages`	All packages in a channel.
Channel Report	`channels`	Detailed report of a given channel.
Cloned Channel Report	`cloned-channels`	Detailed report of cloned channels.
Configuration Files	`config-files`	All configuration file revisions for all organizations, including file contents and file information.
Latest Configuration Files	`config-files-latest`	The most recent configuration file revisions for all organizations, including file contents and file information.
Custom Channels	`custom-channels`	Channel metadata for all channels owned by specific organizations.
Custom Info	`custom-info`	Client custom information.
Patches in Channels	`errata-channels`	All patches in channels.
Patches Details	`errata-list`	All patches that affect registered clients.
All patches	`errata-list-all`	All patches.
Patches for Clients	`errata-systems`	Applicable patches and any registered clients that are affected.
Host Guests	`host-guests`	Host and guests mapping.
Inactive Clients	`inactive-systems`	Inactive clients.
System Inventory	`inventory`	Clients registered to the server, together with hardware and software information.
Kickstart Scripts	`kickstart-scripts`	All kickstart scripts, with details.
Kickstart Trees	`kickstartable-trees`	Kickstartable trees.
All Upgradable Versions	`packages-updates-all`	All newer package versions that can be upgraded.
Newest Upgradable Version	`packages-updates-newest`	Newest package versions that can be upgraded.
Proxy Overview	`proxies-overview`	All proxies and the clients registered to each.
Repositories	`repositories`	All repositories, with their associated SSL details, and any filters.

Report	Invoked as	Description
Result of SCAP	`scap-scan`	Result of OpenSCAP `sccdf` evaluations.
Result of SCAP	`scap-scan-results`	Result of OpenSCAP `sccdf` evaluations, in a different format.
System Data	`splice-export`	Client data needed for splice integration.
System Crash: Count	`system-crash-count`	The total number of client crashes.
System Crash: Details	`system-crash-details`	Crash details for all clients.
System Currency	`system-currency`	Number of available patches for each registered client.
System Extra Packages	`system-extra-packages`	All packages installed on all clients that are not available from channels the client is subscribed to.
System Groups	`system-groups`	System groups.
Activation Keys for System Groups	`system-groups-keys`	Activation keys for system groups.
Systems in System Groups	`system-groups-systems`	Clients in system groups.
System Groups Users	`system-groups-users`	System groups and users that have permissions on them.
History: System	`system-history`	Event history for each client.
History: Channels	`system-history-channels`	Channel event history.
History: Configuration	`system-history-configuration`	Configuration event history.
History: Entitlements	`system-history-entitlements`	System entitlement event history.
History: Errata	`system-history-errata`	Errata event history.
History: Kickstart	`system-history-kickstart`	Kickstart event history.
History: Packages	`system-history-packages`	Package event history.
History: SCAP	`system-history-scap`	OpenSCAP event history.
MD5 Certificates	`system-md5-certificates`	All registered clients using certificates with an MD5 checksum.

Report	Invoked as	Description
Installed Packages	`system-packages-installed`	Packages installed on clients.
System Profiles	`system-profiles`	All clients registered to the server, with software and system group information.
Users	`users`	All users registered to Uyuni.
MD5 Users	`users-md5`	All users for all organizations using MD5 encrypted passwords, with their details and roles.
Systems administered	`users-systems`	Clients that individual users can administer.

For more information about an individual report, run `spacewalk-report` with the option `--info` or `--list-fields-info` and the report name. The description and list of possible fields in the report will be shown.

For further information on program invocation and options, see the `spacewalk-report(8)` man page as well as the `--help`parameter of the `spacewalk-report` command.

Content Lifecycle Management Examples

This section contains some common examples of how you can use content lifecycle management. Use these examples to build your own personalized implementation.

Creating a Project for a Monthly Patch Cycle

An example project for a monthly patch cycle consists of:

- Creating a `By Date` filter
- Adding the filter to the project
- Applying the filter to a new project build
- Excluding a patch from the project
- Including a patch in the project

Creating a `By Date` filter

The `By Date` filter excludes all patches released after a specified date. This filter is useful for your content lifecycle projects that follow a monthly patch cycle.

Procedure: Creating the `By Date` Filter

1. In the Uyuni Web UI, navigate to **Content Lifecycle › Filters** and click **[Create Filter]**.
2. In the `Filter Name` field, type a name for your filter. For example, `Exclude patches by date`.
3. In the `Filter Type` field, select `Patch (Issue date)`.
4. In the `Matcher` field, `later or equal` is autoselected.
5. Select the date and time.
6. Click **[Save]**.

Adding the Filter to the Project

Procedure: Adding a Filter to a Project

1. In the Uyuni Web UI, navigate to **Content Lifecycle › Projects** and select a project from the list.
2. Click **[Attach/Detach Filters]** link to see all available filters
3. Select the new `Exclude patches by date` filter.
4. Click **[Save]**.

Applying the Filter to a new Project Build

The new filter is added to your filter list, but it still needs to be applied to the project. To apply the filter you need to build the first environment.

Procedure: Using the Filter

Click **[Build]** to build the first environment.

1. OPTIONAL: Add a message. You can use messages to help track the build history.
2. Check that the filter has worked correctly by using the new channels on a test server.
3. Click **[Promote]** to move the content to the next environment. The build will take longer if you have a large number of filters, or they are very complex.

Excluding a Patch from the Project

Tests may help you discover issues. When an issue is found, exclude the problem patch released before the `by date` filter.

Procedure: Excluding a Patch

1. In the Uyuni Web UI, navigate to **Content Lifecycle › Filters** and click **[Create Filter]**.
2. In the `Filter Name` field, enter a name for the filter. For example, `Exclude openjdk patch`.
3. In the `Filter Type` field, select `Patch (Advisory Name)`.
4. In the `Matcher` field, select `equals`.

5. In the Advisory Name field, type a name for the advisory. For example, SUSE-15-2019-1807.
6. Click **[Save]**.
7. Navigate to **Content Lifecycle › Projects** and select your project.
8. Click **[Attach/Detach Filters]** link, select Exclude openjdk patch, and click **[Save]**.

When you rebuild the project with the **[Build]** button, the new filter is used together with the by date filter we added before.

Including a Patch in the Project

In this example, you have received a security alert. An important security patch was released several days after the first of the month you are currently working on. The name of the new patch is SUSE-15-2019-2071. You need to include this new patch into your environment.

The Allow filters rule overrides the exclude function of the Deny filter rule. For more information, see [**Administration › Content-lifecycle ›**].

Procedure: Including a Patch in a Project

1. In the Uyuni Web UI, navigate to **Content Lifecycle › Filters** and click **[Create Filter]**.
2. In the Filter Name field, type a name for the filter. For example, Include kernel security fix.
3. In the Filter Type field, select Patch (Advisory Name).
4. In the Matcher field, select equals.
5. In the Advisory Name field, type SUSE-15-2019-2071, and check Allow.
6. Click **[Save]** to store the filter.
7. Navigate to **Content Lifecycle › Projects** and select your project from the list.
8. Click **[Attach/Detach Filters]**, and select Include kernel security patch.
9. Click **[Save]**.
10. Click **[Build]** to rebuild the environment.

Update an Existing Monthly Patch Cycle

When a monthly patch cycle is complete, you can update the patch cycle for the next month.

Procedure: Updating a Monthly Patch Cycle

1. In the by date field, change the date of the filter to the next month. Alternatively, create a new filter and change the assignment to the project.
2. Check if the exclude filter for SUSE-15-2019-1807 can be detached from the project. There may

be a new patch available to fix this issue.

3. Detach the `allow` filter you added previously. The patch is included by default.
4. Rebuild the project to create a new environment with patches for the next month.

Enhance a Project with Live Patching

This section covers setting up filters to create environments for live patching.

When you are preparing to use live patching, there are some important considerations

- Only ever use one kernel version on your systems. The live patching packages are installed with a specific kernel.
- Live patching updates are shipped as one patch.
- Each kernel patch that begins a new series of live patching kernels will display the `required reboot` flag. These kernel patches come with live patching tools. When you have installed them, you will need to reboot the system at least once before the next year.
- Only install live patch updates that match the installed kernel version.
- Live patches are provided as stand-alone patches. You must exclude all regular kernel patches with higher kernel version than the currently installed one.

Exclude Packages with a Higher Kernel Version

In this example you update your systems with the `SUSE-15-2019-1244` patch. This patch contains `kernel-default-4.12.14-150.17.1-x86_64`.

You need to exclude all patches which contain a higher version of `kernel-default`.

Procedure: Excluding Packages with a Higher Kernel Version

1. In the Uyuni Web UI, navigate to **Content Lifecycle › Filters**, and click **`[Create Filter]`**.
2. In the `Filter Name` field, type a name for your filter. For example, `Exclude kernel greater than 4.12.14-150.17.1`.
3. In the `Filter Type` field, select `Patch (Contains Package)`.
4. In the `Matcher` field, select `version greater than`.
5. In the `Package Name` field, type `kernel-default`.
6. Leave the the `Epoch` field empty.
7. In the `Version` field, type `4.12.14`.

8. In the `Release` field, type `150.17.1`.
9. Click **`[Save]`** to store the filter.
10. Navigate to **Content Lifecycle › Projects** and select your project.
11. Click **`[Attach/Detach Filters]`**.
12. Select `Exclude kernel greater than 4.12.14-150.17.1`, and click **`[Save]`**.

When you click **`[Build]`**, a new environment is created. The new environment contains all the kernel patches up to the version you installed.

All kernel patches with higher kernel versions are removed. Live patching kernels will stay available as long as they are not the first in a series.

Update the Project for Next Patch Month

To update the project to the next patch month you operate similar to the case before. Important is, that you do not change the "Exclude kernel greater than 4.12.14-150.17.1: …" Filter. With it you keep normal kernel-updates away, but take the latest live patches up to the selected month.

Switch to a New Kernel Version for Live Patching

Live Patching for a specific kernel version is only available for one year. After one year you must update the kernel on your systems. The following changes of the environment should be executed:

Procedure: Switch to a New Kernel Version

1. Decide which kernel version you will upgrade to. For example: `4.12.14-150.32.1`
2. Create a new kernel version Filter.
3. Detach the previous filter **Exclude kernel greater than 4.12.14-150.17.1** and attach the new filter.

Click **`[Build]`** to rebuild the environment. The new environment contains all kernel patches up to the new kernel version you selected. Systems using these channels will have the kernel update available for installation. You will need to reboot systems after they have performed the upgrade. The new kernel will remain valid for one year. All packages installed during the year will match the current live patching kernel filter.

Tuning Changelogs

Some packages have a long list of changelog entries. This data is downloaded by default, but it is not always useful information to keep. In order to limit the amount of changelog metadata which is downloaded and to save disk space, you can put a limit on how many entries to keep on disk.

This configuration option is in the `/etc/rhn/rhn.conf` configuration file. The parameter defaults to `0`, which means unlimited.

```
java.max_changelog_entries = 0
```

If you set this parameter, it will come into effect only for new packages when they are synchronized.

After changing this parameter, restart services with `spacewalk-service restart`.

You might like to delete and regenerate the cached data to remove older data.

Deleting and regenerating cached data can take a long time. Depending on the number of channels you have and the amount of data to be deleted, it can potentially take several hours. The task is run in the background by Taskomatic, so you can continue to use Uyuni while the operation completes, however you should expect some performance loss.

You can delete and request a regeneration of cached data from the command line:

```
spacewalk-sql -i
```

Then on the SQL database prompt, enter:

```
DELETE FROM rhnPackageRepodata;
INSERT INTO rhnRepoRegenQueue (id, CHANNEL_LABEL, REASON, FORCE)
(SELECT sequence_nextval('rhn_repo_regen_queue_id_seq'),
        C.label,
        'cached data regeneration',
        'Y'
        FROM rhnChannel C);
\q
```

Maintenance Window

If you work with scheduled maintenance windows, you might find it difficult to remember all the things that you need to do before, during, and after that critical downtime of the Uyuni Server. Uyuni Server related systems such as Inter-Server Synchronization Slave Servers or Uyuni Proxies are also affected and have to be considered.

SUSE recommends you always keep your Uyuni infrastructure updated. That includes servers, proxies,

and build hosts. If you do not keep the Uyuni Server updated, you might not be able to update some parts of your environment when you need to.

This section contains a checklist for your maintenance window, with links to further information on performing each of the steps.

Server

1. Apply the latest updates. See **[Upgrade › Server-update ›]**.
2. Upgrade to the latest service pack, if required. See **[Upgrade › Migrate-4x-4x ›]**.
3. Run `spacewalk-service status` and check whether all required services are up and running.

For information about database schema upgrades and PostgreSQL migrations, see **[Upgrade › Db-migration ›]**.

You can install updates using your package manager. For information on using YaST, see https://documentation.suse.com/sles/15-SP1/html/SLES-all/cha-onlineupdate-you.html. For information on using zypper, see https://documentation.suse.com/sles/15-SP1/html/SLES-all/cha-sw-cl.html#sec-zypper.

By default, several update channels are configured and enabled for the Uyuni Server. New and updated packages will become available automatically.

Client Tools

When the server is updated consider to update some tools on the clients, too. Updating `salt-minion`, `zypper`, and other related management package on clients is not a strict requirement, but it is a best practice in general. For example, a maintenance update on the server might introduce a major new Salt version. Then minions will continue to work but might experience problems later on. To avoid this always update the salt-minion package when available. SUSE makes sure that `salt-minion` can always be updated safely.

Inter-Server Synchronization Slave Server

If you are using an inter-server synchronization slave server, update it after the Uyuni Server update is complete.

For more in inter-server synchronization, see **[Administration › Iss ›]**.

Monitoring Server

If you are using a monitoring server for Prometheus, update it after the Uyuni Server update is complete.

For more information on monitoring, see **[Administration › Monitoring ›]**.

Proxy

Proxies should be updated as soon as Uyuni Server updates are complete.

In general, running a proxy connected to a server on a different version is not supported. The only exception is for the duration of updates where it is expected that the server is updated first, so the proxy could run the previous version temporarily.

Especially if you are migrating from version 3.2 to 4.0, upgrade the server first, then any proxy.

For more information, see **[Upgrade › Proxy-update ›]** and **[Upgrade › Proxy-migration ›]**.

Troubleshooting

Troubleshooting

This section contains some common problems you might encounter with Uyuni, and solutions to resolving them.

Troubleshooting Corrupt Repositories

The information in the repository metadata files can become corrupt or out of date. This can create problems with updating clients. You can fix this by removing the files and regenerating it. With an new repository data file, updates should operate as expected.

Procedure: Resolving Corrupt Repository Data

1. Remove all files from `/var/cache/rhn/repodata/<channel-label>-updates-x86_64`. If you do not know the channel label, you can find it in the Uyuni Web UI, by navigating to **Software › Channels › Channel Label**.
2. Regenerate the file from the command line:

```
spacecmd softwarechannel_regenerateyumcache <channel-label>-updates-x86_64
```

Troubleshooting Disk Space

Running out of disk space can have a severe impact on the Uyuni database and file structure which, in most cases, is not recoverable.

Uyuni monitors free space in specific directories, and has configurable alerts. For more on space management, see **[Administration › Space-management ›]**.

You can recover disk space by removing unused custom channels and redundant database entries before you run out of space entirely.

For instructions on how to delete custom channels, see **[Administration › Channel-management ›]**.

Procedure: Resolving redundant database entries

1. Use the `spacewalk-data-fsck` command to list any redundant database entries.
2. Use the `spacewalk-data-fsck --remove` command to delete them.

Troubleshooting Local Issuer Certificates

Some older bootstrap scripts create a link to the local certificate in the wrong place. This results in zypper returning an `Unrecognized error` about the local issuer certificate. You can ensure that the link to the local issuer certificate has been created correctly by checking the `/etc/ssl/certs/` directory. If you come across this problem, you should consider updating your bootstrap scripts to ensure that zypper operates as expected.

Troubleshooting Login Timeouts

By default, the Uyuni Web UI will require users to log in again after 30 minutes. Depending on your environment, you might want to adjust the login timeout value.

To adjust the value, you will need to make the change in both `rhn.conf` and `web.xml`. Ensure you set the value in seconds in `/etc/rhn/rhn.conf`, and in minutes in `web.xml`. The two values must equal the same amount of time.

For example, to change the timeout value to one hour, set the value in `rhn.conf` to 3600 seconds, and the value in `web.xml` to 60 minutes.

Procedure: Adjusting the Web UI Login Timeout Value

1. Stop services:

```
spacewalk-service stop
```

2. Open `/etc/rhn/rhn.conf` and add or edit this line to include the new timeout value in seconds:

```
web.session_database_lifetime = <Timeout_Value_in_Seconds>
```

3. Save and close the file.
4. Open `/srv/tomcat/webapps/rhn/WEB-INF/web.xml` and add or edit this line to include the new timeout value in minutes:

```
<session-timeout>Timeout_Value_in_Minutes</session-timeout>
```

5. Save and close the file.
6. Restart services:

```
spacewalk-service start
```

Troubleshooting OSAD and jabberd

Open File Count Exceeded

In some cases, the maximum number of files that jabber can open is lower than the number of connected OSAD clients.

If this occurs, OSAD clients cannot contact the SUSE Manager Server, and jabberd will take an excessive amount of time to respond on port 5222.

This fix is only required if you have more than 8192 clients connected using OSAD. In this case, we recommend you consider using Salt clients instead. For more information about tuning large scale installations, see **[Salt › Large-scale ›]**.

You can increase the number of files available to jabber by editing the jabberd local configuration file. By default, the file is located at `/etc/systemd/system/jabberd.service.d/override.conf`.

Procedure: Adjusting the Maximum File Count

1. At the command prompt, as root, open the local configuration file for editing:

```
systemctl edit jabberd
```

2. Add or edit this section:

```
[Service]
LimitNOFILE=<soft_limit>:<hard_limit>
```

   The value you choose will vary depending on your environment. For example, if you have 9500 clients, increase the soft value by 100 to 9600, and the hard value by 1000 to 10500:

```
[Unit]
LimitNOFILE=
LimitNOFILE=9600:10500
```

3. Save the file and exit the editor.

The default editor for systemctl files is vim. To save the file and exit, press *Esc* to enter `normal` mode, type *:wq* and press *Enter*.

Ensure you also update the `max_fds` parameter in `/etc/jabberd/c2s.xml`. For example: `<max_fds>10500</max_fds>`

The soft file limit is the maximum number of open files for a single process. In Uyuni the highest consuming process is `c2s`, which opens a connection per client. 100 additional files are added, here, to accommodate for any non-connection file that `c2s` requires to work correctly. The hard limit applies to all processes belonging to jabber, and also accounts for open files from the router, `c2s` and `sm` processes.

Troubleshooting Package Inconsistencies

When packages on a client are locked, Uyuni Server may not be able to correctly determine the set of applicable patches. When this occurs, package updates will be available in the Web UI, but will not appear on the client, and attempts to update the client will fail. Check package locks and exclude lists to determine if packages are locked or excluded on the client.

On the client, check package locks and exclude lists to determine if packages are locked or excluded:

- On an Expanded Support Platform, check `/etc/yum.conf` and search for `exclude=`.
- On SUSE Linux Enterprise and openSUSE, use the `zypper locks` command.

Troubleshooting Registering Cloned Clients

If you are using Uyuni to manage virtual machines, you might find it useful to create clones of your VMs. A clone is a VM that uses a primary disk that is an exact copy of an existing disk.

While cloning VMs can save you a lot of time, the duplicated identifying information on the disk can sometimes cause problems.

If you have a client that is already registered, you create a clone of that client, and then try and register the clone, you probably want Uyuni to register them as two separate clients. However, if the machine ID in both the original client and the clone is the same, Uyuni will register both clients as one system, and the existing client data will be over-written with that of the clone.

This can be resolved by changing the machine ID of the clone, so that Uyuni recognizes them as two different clients.

Each step of this procedure is performed on the cloned client. This procedure does not manipulate the original client, which will still be registered to Uyuni.

Procedure: Resolving Duplicate Machine IDs in Cloned Salt Clients

1. On the cloned machine, change the hostname and IP addresses. Make sure `/etc/hosts` contains the changes you made and the correct host entries.
2. For distributions that support systemd: If your machines have the same machine ID, delete the file on each duplicated client and re-create it:

```
# rm /etc/machine-id
# rm /var/lib/dbus/machine-id
# dbus-uuidgen --ensure
# systemd-machine-id-setup
```

3. For distributions that do not support systemd: Generate a machine ID from dbus:

```
# rm /var/lib/dbus/machine-id
# dbus-uuidgen --ensure
```

4. If your clients still have the same Salt client ID, delete the `minion_id` file on each client (FQDN will be used when it is regenerated on client restart):

```
# rm /etc/salt/minion_id
```

5. Delete accepted keys from the onboarding page and the system profile from Uyuni, and restart the client with:

```
# service salt-minion restart
```

6. Re-register the clients. Each client will now have a different `/etc/machine-id` and should be correctly displayed on the `System Overview` page.

Procedure: Resolving Duplicate Machine IDs in Cloned Traditional Clients

1. On the cloned machine, change the hostname and IP addresses. Make sure `/etc/hosts` contains the changes you made and the correct host entries.
2. Stop the `rhnsd` daemon, on Red Hat Enterprise Linux Server 6 and SUSE Linux Enterprise 11 with:

```
# /etc/init.d/rhnsd stop
```

or, on newer systemd-based systems, with:

```
# service rhnsd stop
```

3. Stop `osad` with:

```
# /etc/init.d/osad stop
```

or:

```
# service osad stop
```

or:

```
# rcosad stop
```

4. Remove the `osad` authentication configuration file and the system ID:

```
# rm -f /etc/sysconfig/rhn/{osad-auth.conf,systemid}
```

5. Delete the files containing the machine IDs:
    - SLES 12:

```
# rm /etc/machine-id
# rm /var/lib/dbus/machine-id
# dbus-uuidgen --ensure
# systemd-machine-id-setup
```

- SLES 11:

```
# suse_register -E
```

6. Remove the credential files:
   - SLES clients:

```
# rm -f /etc/zypp/credentials.d/{SCCcredentials,NCCcredentials}
```

   - Red Hat Enterprise Linux clients:

```
# rm  -f /etc/NCCcredentials
```

7. Re-run the bootstrap script. You should now see the cloned system in Uyuni without overriding the system it was cloned from.

Troubleshooting RPC Connection Timeouts

RPC connections can sometimes time out due to slow networks or a network link going down. This results in package downloads or batch jobs hanging or taking longer than expected. You can adjust the maximum time that an RPC connection can take by editing the configuration file. While this will not resolve networking problems, it will cause a process to fail rather than hang.

Procedure: Resolving RPC connection timeouts

1. On the Uyuni Server, open the `/etc/rhn/rhn.conf` file and set a maximum timeout value (in seconds):

```
server.timeout =`number`
```

2. On the Uyuni Proxy, open the `/etc/rhn/rhn.conf` file and set a maximum timeout value (in seconds):

```
proxy.timeout =`number`
```

3. On a SUSE Linux Enterprise Server client that uses zypper, open the `/etc/zypp/zypp.conf` file and set a maximum timeout value (in seconds):

```
## Valid values:  [0,3600]
## Default value: 180
download.transfer_timeout = 180
```

4. On a Red Hat Enterprise Linux client that uses yum, open the `/etc/yum.conf` file and set a maximum timeout value (in seconds):

```
timeout =`number`
```

If you limit RPC timeouts to less than `180` seconds, you risk aborting perfectly normal operations.

Troubleshooting the Saltboot Formula

Because of a problem in the computed partition size value, the saltboot formula can sometimes fail when it is created on SLE 11 SP3 clients, with an error like this:

```
          ID: disk1_partitioned
    Function: saltboot.partitioned
        Name: disk1
      Result: false
     Comment: An exception occurred in this state: Traceback (most recent call last):
  File "/usr/lib/python2.6/site-packages/salt/state.py", line 1767, in call
    **cdata['kwargs'])
  File "/usr/lib/python2.6/site-packages/salt/loader.py", line 1705, in wrapper
    return f(*args, **kwargs)
  File "/var/cache/salt/minion/extmods/states/saltboot.py", line 393, in disk_partitioned
    existing = __salt__['partition.list'](device, unit='MiB')
  File "/usr/lib/python2.6/site-packages/salt/modules/parted.py", line 177, in list_
    'Problem encountered while parsing output from parted')
CommandExecutionError: Problem encountered while parsing output from parted
```

This problem can be resolved by manually configuring the size of the partition containing the operating system. When the size is set correctly, formula creation will work as expected.

Procedure: Manually Configuring the Partition Size in the Saltboot Formula

1. In the Uyuni Web UI, navigate to **Systems** › **System Groups** and select the `Hardware Type Group` that contains the SLE 11 SP3 client that is causing the error. In the `Formulas` tab, navigate to the `Saltboot` subtab.
2. Locate the partition that contains the operating system, and in the `Partition Size` field, type the appropriate size (in MiB).
3. Click **`[Save Formula]`**, and apply the highstate to save your changes.

Troubleshooting Taskomatic

If Taskomatic crashes, repository metadata regeneration can be prevented from happening regularly.

Repository metadata regeneration is a relatively intensive process, so Taskomatic can normally take several minutes to complete it. When this occurs, package updates will be available in the Web UI, but will not appear on the client, and attempts to update the client will fail. To correct this, determine if Taskomatic is still in the process of generating repository metadata, or if a crash could have occurred. Wait for metadata regeneration to complete or restart Taskomatic after a crash in order for client updates to be carried out correctly.

Procedure: Resolving Taskomatic Problems

1. On the Uyuni Server, check the `/var/log/rhn/rhn_taskomatic_daemon.log` file to determine if any metadata regeneration processes are still running, or if a crash occurred.
2. Restart taskomatic:

```
service taskomatic restart
```

In the Taskomatic log files, you can identify the section related to metadata regeneration by looking for opening and closing lines that look like this:

```
<YYYY-DD-MM> <HH:MM:SS>,174 [Thread-584] INFO
com.redhat.rhn.taskomatic.task.repomd.RepositoryWriter - Generating new repository metadata
for channel 'cloned-2018-q1-sles12-sp3-updates-x86_64'(sha256) 550 packages, 140 errata

...

<YYYY-DD-MM> <HH:MM:SS>,704 [Thread-584] INFO
com.redhat.rhn.taskomatic.task.repomd.RepositoryWriter - Repository metadata generation for
'cloned-2018-q1-sles12-sp3-updates-x86_64' finished in 4 seconds
```