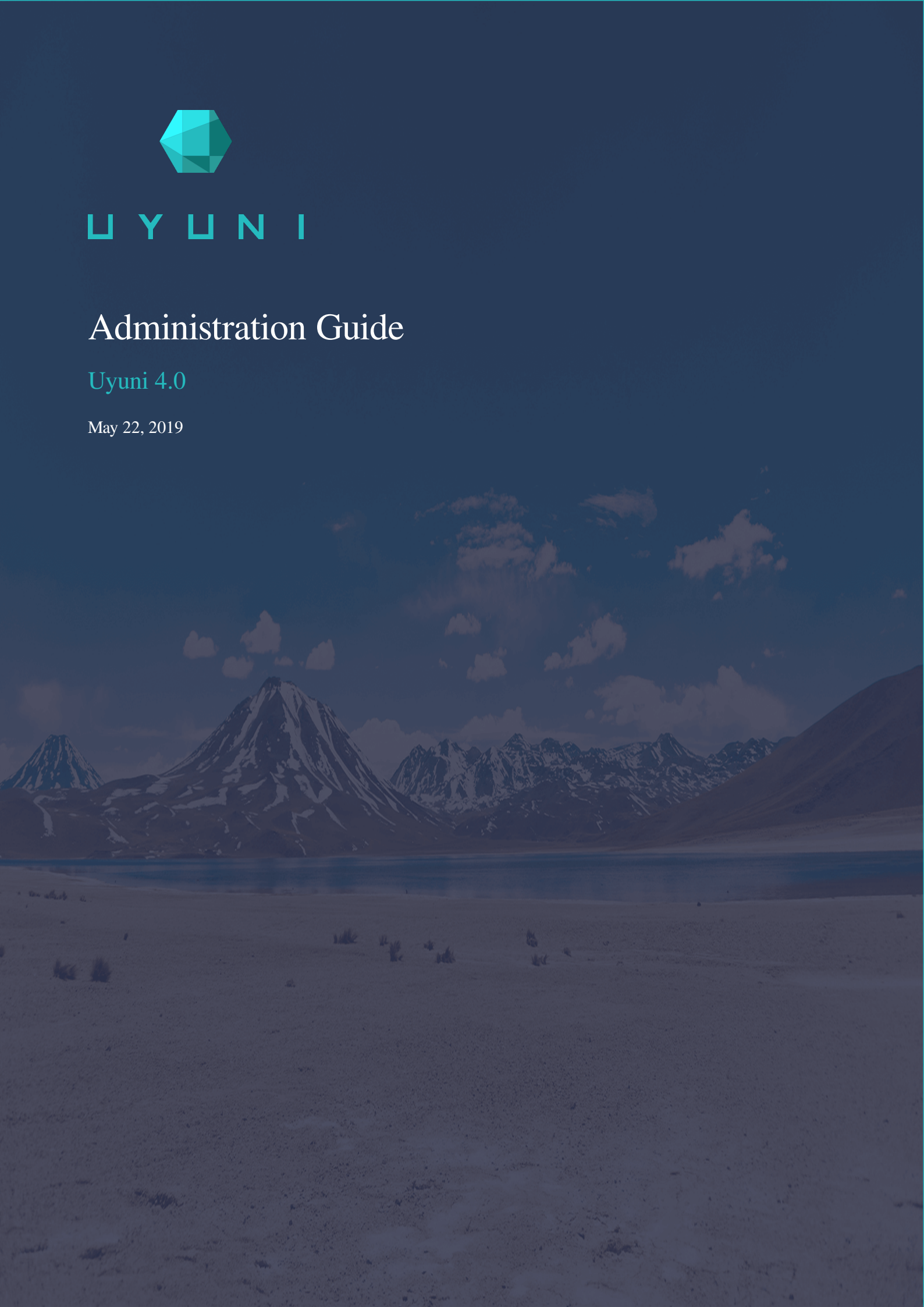


Administration Guide

Uyuni 4.0

May 22, 2019

Table of Contents

GNU Free Documentation License

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

TODO: Introduction to the Admin Guide

Image Building and Management

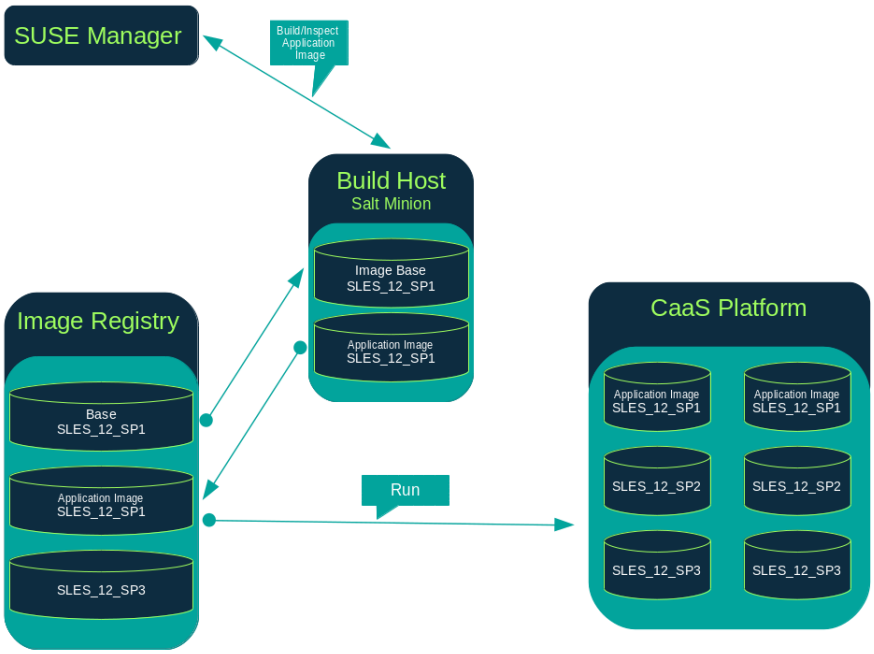
Image Building Overview

Uyuni enables system administrators to build containers, systems, and virtual images. Uyuni helps with creating Image Stores and managing Image Profiles.

Uyuni supports two distinct build types:

- Dockerfile-for more information, see Container Images
- Kiwi image system-for more information, see OS Images

Container Images



Requirements

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

- An existing external GitHub or internal GitLab repository containing a Dockerfile and configuration scripts (example scripts are provided in this chapter).
- A properly configured image registry.

Registry Provider Solutions

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 CaaS Platform, see:

- SUSE Linux Enterprise Server 12 SP3 Docker Guide
- SUSE CaaS Platform 2 Documentation

Creating a Build Host

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

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

1. Select a minion to be designated as a build host from the **Systems › Overview** page.
2. From the `System Details` page for the selected minion assign the containers modules by going to **Software › Software Channels** and enabling `SLE-Module-Containers12-Pool` and `SLE-Module-Containers12-Updates`. Confirm by clicking **[Change Subscriptions]**.
3. From the **System Details › Properties** page, enable `Add-on System Type` and `Container Build Host` 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:

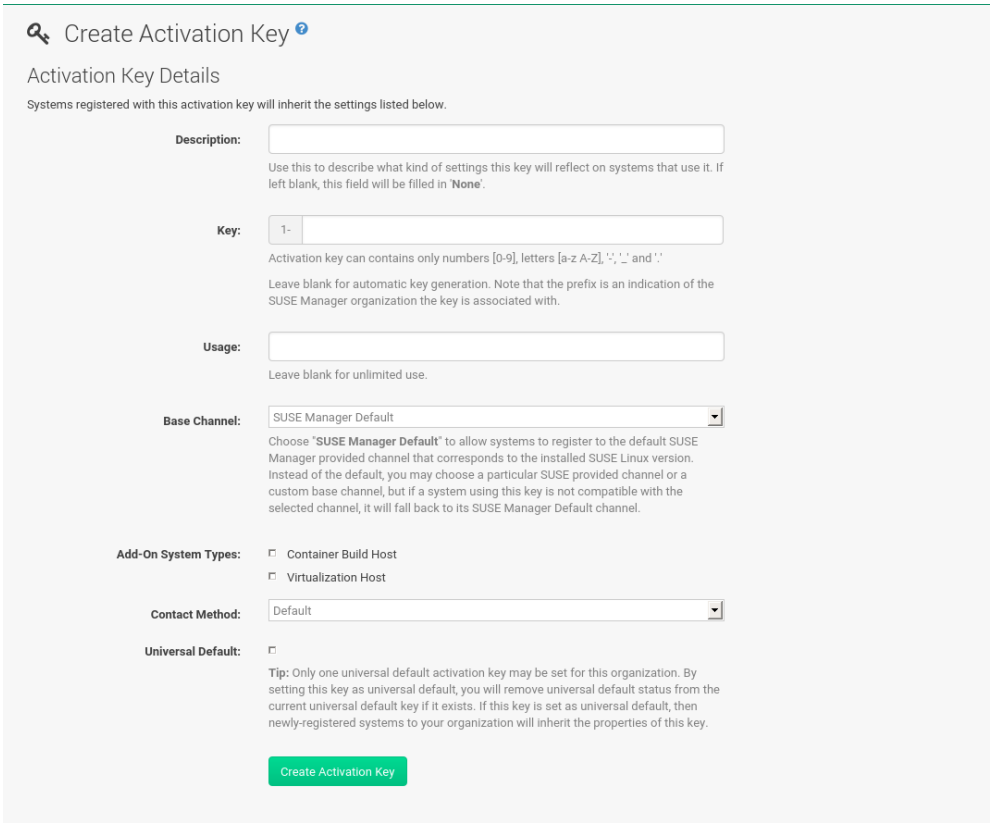
```
salt '$your_minion' state.highstate
```

Define Container Build Channels with an Activation Key

Create an activation key associated with the channel that your images will use.

Relationship Between Activation Keys and Image Profiles

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

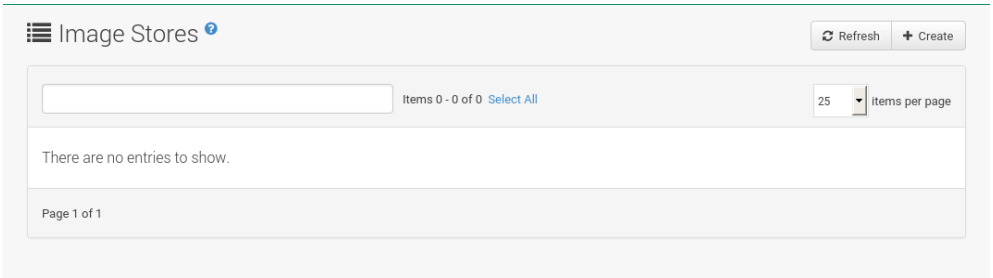


1. Select **Main Menu › Systems › Activation Keys**.
2. Click **[Create Key]**.
3. Enter a `Description` and a `Key` name. Use the drop-down menu to select the `Base Channel` to associate with this key.
4. Confirm with **[Create Activation Key]**.

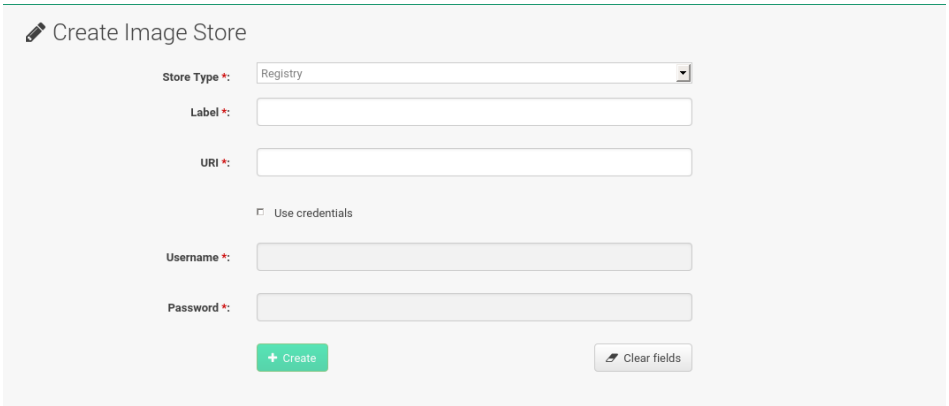
For more information, see [bp.key.managment].

Creating an Image Store

Define a location to store all of your images by creating an Image Store.



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



3. Uyuni currently provides support only for the `Registry` store type. Define a name for the image store in the `Label` field.
4. 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
```

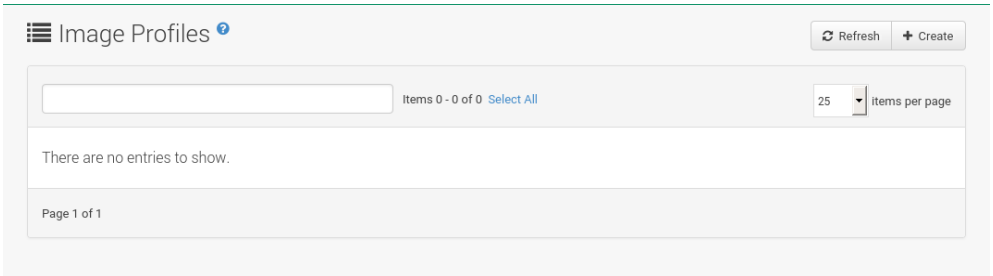
The Registry URI can also be used to specify an image store on a used registry.

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

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

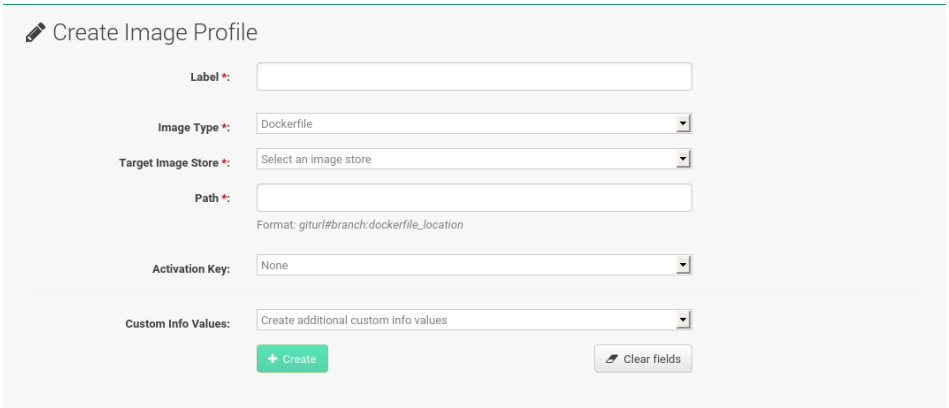
Creating an Image Profile

Manage Image Profiles from the `Image Profile` page.



Procedure: Create an Image Profile

1. To create an image profile select **Image › 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. Enter a Github or Gitlab repository URL (http, https, or token authentication) in the `Path` field using one of the following 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 and not publicly accessible, you need to modify the profile's git 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 Github or Gitlab 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 and add_packages Script

This section contains an example Dockerfile. You specify a Dockerfile that will be used during image building when creating an image profile. A Dockerfile and any associated scripts should be stored within an internal or external Github or Gitlab repository:

Required Dockerfile Lines

The Dockerfile provides access to a specific repository version served by Uyuni. This example Dockerfile is used by Uyuni to trigger a build job on a build host minion. The `ARG` parameters ensure that the image that is built 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.

```
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
```

This is an example `add_packages.sh` script for use with your Dockerfile:

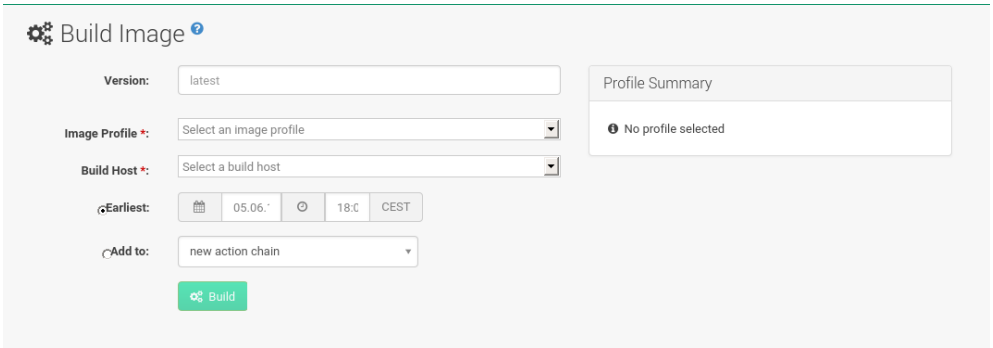
```
#!/bin/bash
set -e

zypper --non-interactive --gpg-auto-import-keys ref

zypper --non-interactive in python python-xml aaa_base aaa_base-extras net-tools timezone vim
less sudo tar
```

Packages Required for Inspecting Your Images

To inspect images and provide the package and product list of a container to the Uyuni Web UI you will need to install python and python-xml within the container. If these packages remain uninstalled, your images will still build, but the package and product list will be unavailable from the Web UI.

Building 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: Build an Image

1. For this example 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.

Importing an Image

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

Procedure: Import an Image

1. From **Images › Images** 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.

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

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

Once it has been processed, you can find the imported image in the `Images` list. It has a different icon in the `Build` column, to indicate that the image is imported (see screenshot). The status icon for the imported image can also be seen on the `Overview` tab for the image.

Troubleshooting

These are some known problems that you might encounter when working with images:

- HTTPS certificates to access the registry or the git repositories should be deployed to the minion by a custom state file.
- SSH git access using Docker is currently unsupported. You may test it, but SUSE will not provide support.
- If the python and python-xml packages are not installed in your images during the build process, Salt cannot run within the container and 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. They can be of various types: PXE, QCOW2, LiveCD images, and others.

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

Requirements

The Kiwi image building feature is available for Salt minions running SUSE Linux Enterprise Server 12. It is currently not supported to build SUSE Linux Enterprise 15 images.

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

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

Example scripts are provided in the following sections.

Hardware Requirements for Hosts Running OS Images

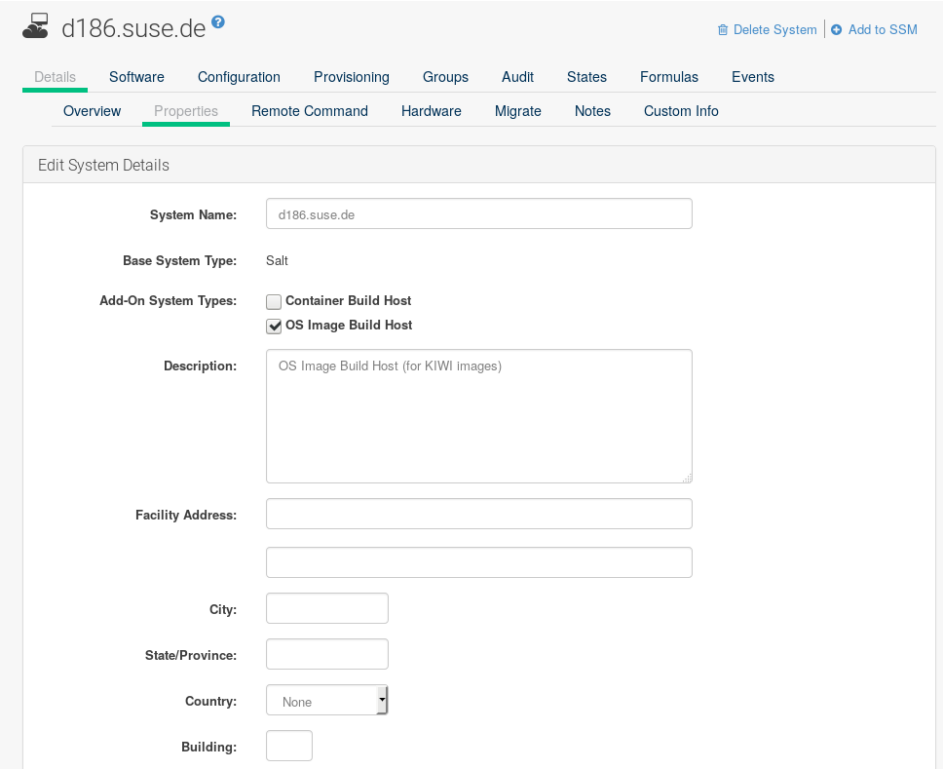
Hosts running OS images built with Kiwi need at least 1 GB of RAM. Disk space depends on the actual size of the image. For more information, see the documentation of the underlying system.

Creating a Build Host

To build all kinds of images with Uyuni, create and configure a build host. OS image build hosts are Salt minions running SUSE Linux Enterprise Server 12 (SP3 or later). This procedure will guide you though the initial configuration for a build host.

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

1. Select a minion that will be designated as a build host from the **Main Menu** › **Systems** › **Overview** page.
2. From the **System Details** › **Properties** page, enable the `Add-on System Type: OS Image Build Host` and confirm with **[Update Properties]**.



3. From the **System Details › Software › Software Channels** page, enable `SLE-Manager-Tools12-Pool` and `SLE-Manager-Tools12-Updates` (or a later version). 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_minion' 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

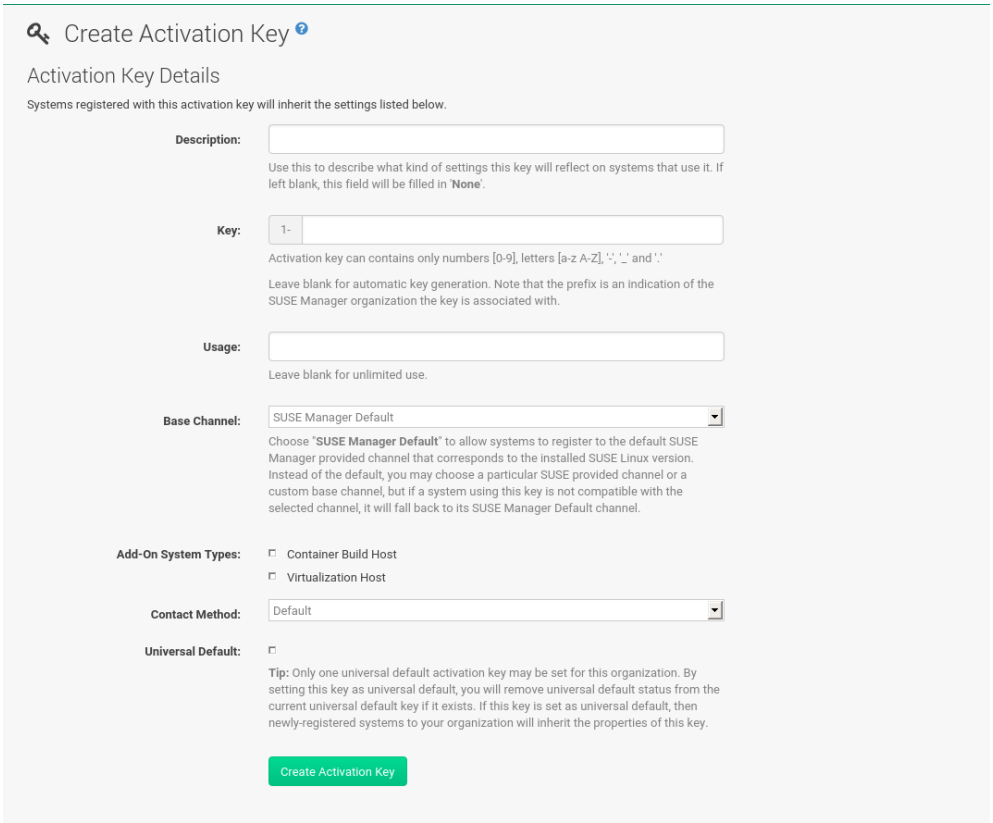
```
...
  <packages type="bootstrap">
    ...
    <package name="rhn-org-trusted-ssl-cert-osimage"
bootinclude="true"/>
  </packages>
...
```

Define Kiwi Build Channels with an Activation Key

Create an activation key associated with the channel that your images will use. Activation keys are mandatory for OS Image building.

Relationship Between Activation Keys and Image Profiles

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 **Main Menu › 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].

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`.

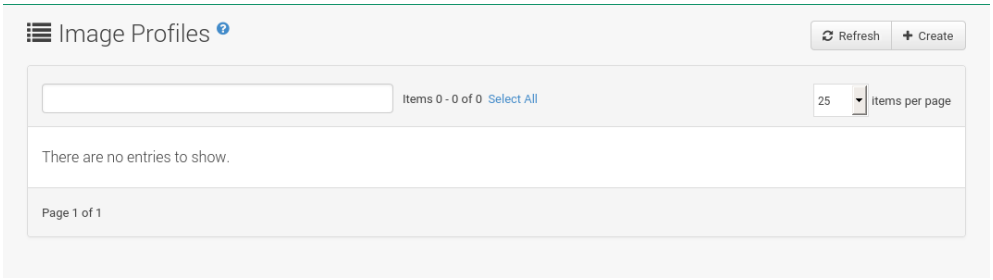
Image stores for Kiwi build type

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>`

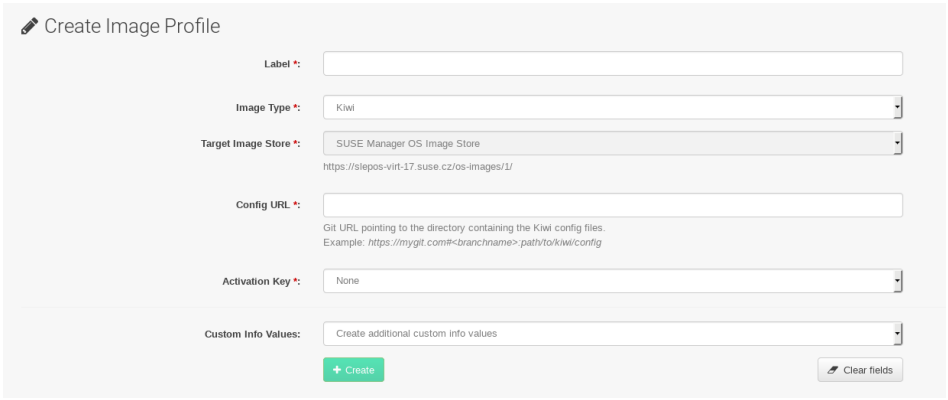
Creating an Image Profile

Manage Image Profiles using the Web UI.



Procedure: Create an Image Profile

1. To create an image profile select from **Main Menu** › **Images** › **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 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.

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.

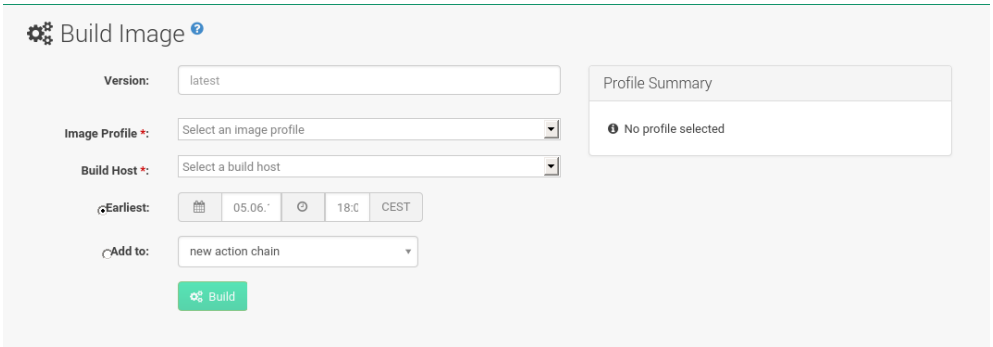
SUSE provides examples of fully functional image sources at the SUSE/manager-build-profiles public GitHub repository.

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>
```

Building an Image

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



Procedure: Build an Image

1. Select **Main Menu › 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.

Image Inspection and Salt Integration

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 minions.

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 Creating a Build Host.

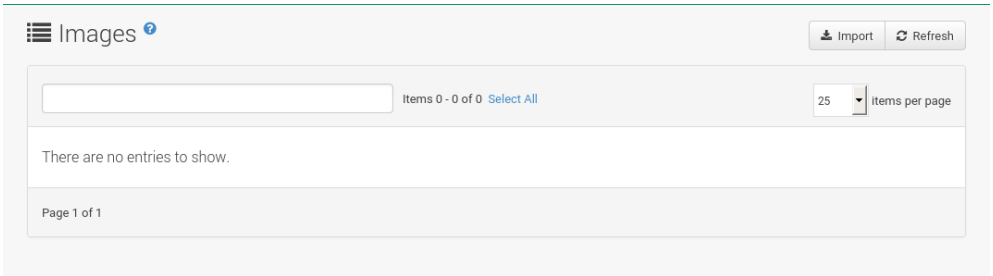
Limitations

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 minion by a custom state file, or configured manually.
- Importing Kiwi-based images is not supported.

Listing Image Profiles Available for Building

To list images available for building select **Main Menu** › **Images** › **Images**. 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 including 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.

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** › **Systems 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. sles15-sp{sp-ver}-pool-x86_64
      \__ sle-live-patching15-pool-x86_64-sp{sp-ver}
      \__ sle-live-patching15-updates-x86_64-sp{sp-ver}
      \__ sle-manager-tools15-pool-x86_64-sp{sp-ver}
      \__ sle-manager-tools15-updates-x86_64-sp{sp-ver}
      \__ sles15-sp{sp-ver}-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{sp-ver}-pool-x86_64
```

3. Check that `dev-sles15-spSP1-updates-x86_64` is available in your channel list.

Now you can 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 › Systems 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-spSP1-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-spSP1-pool-x86_64` channel to a new testing channel:

```
# spacewalk-manage-channel-lifecycle -promote -c dev-sles15-sp{sp-ver}-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: image::live_patching_criticalupdates.png[scaledwidth=80%]
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

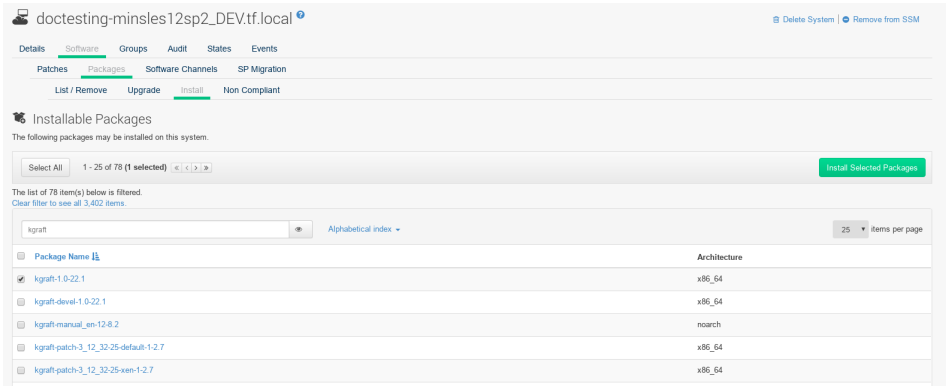
On SLES 12 systems, live patching is managed by kGraft. For in depth information covering kGraft use, see https://www.suse.com/documentation/sles-12/singlehtml/book_sle_admin/book_sle_admin.html#cha.kgraft.

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 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 › Systems 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-sp{sp-ver}-pool-x86_64
      \__ sle-live-patching12-pool-x86_64-sp{sp-ver}
      \__ sle-live-patching12-updates-x86_64-sp{sp-ver}
      \__ sle-manager-tools12-pool-x86_64-sp{sp-ver}
      \__ sle-manager-tools12-updates-x86_64-sp{sp-ver}
      \__ sles12-sp{sp-ver}-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-sp{sp-ver}-pool-x86_64
```

3. Check that `dev-sles12-spSP1-updates-x86_64` is available in your channel list.

Now you can 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 › Systems 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-spSP1-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-spSP1-pool-x86_64` channel to a new testing channel:

```
# spacewalk-manage-channel-lifecycle -promote -c dev-sles12-sp{sp-ver}-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-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: image::live_patching_criticalupdates.png[scaledwidth=80%]
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 Icinga

Introduction

This chapter provides guidance on the setup of an Icinga server using SLES 15 SP1. For more information, see the Official Icinga documentation: http://docs.icinga.org/latest/en/.

Installation and Basic Configuration

Icinga packages are found in the `SLE-Manager-Tools15-Updates x86_64`.

Icinga Installation Location

Do not install Icinga on the Uyuni server. Install Icinga on a stand-alone SUSE Linux Enterprise client.

Procedure: Installation and Basic Configuration

1. Register the new client with Uyuni and subscribe it to the Uyuni client and update channels. SLES 15 and later include these channels by default.
2. Install the required Icinga packages on the new client:

```
zypper in icinga icinga-idoutils-pgsql postgresql postgresql94-server \
monitoring-plugins-all apache2
```

3. Edit the `/etc/icinga/objects/contacts.cfg` file and add the email address which you will use for reciving alerts.

```
define contact {
  contact_name      icingaadmin          ; Short name of user
  use               generic-contact      ; Inherit default values
  alias             Icinga Admin         ; Full name of user
  email             icinga@localhost     ; <<*** CHANGE THIS TO YOUR EMAIL ADDRESS ***
}
```

4. Enable postgres on boot and start the database:

```
systemctl enable postgresql.service
systemctl start postgresql.service
```

5. Create the database and user for Icinga:

```
>psql
 postgres=# ALTER USER postgres WITH PASSWORD '<newpassword>';
 postgres=# CREATE USER icinga;
 postgres=# ALTER USER icinga WITH PASSWORD 'icinga';
 postgres=# CREATE DATABASE icinga;
 postgres=# GRANT ALL ON DATABASE icinga TO icinga;
 postgres=# \q
 exit
```

6. Adjust client authentication rights located in `/var/lib/pgsql/data/pg_hba.conf` to match the following:

```
# TYPE  DATABASE        USER            ADDRESS                 METHOD
local   icinga         icinga                                  trust
local   all            postgres                                ident

# "local" is for Unix domain socket connections only
local   all             all                                     trust
# IPv4 local connections:
host    all             all             127.0.0.1/32            trust
# IPv6 local connections:
host    all             all             ::1/128                 trust
# Allow replication connections from localhost, by a user with the
# replication privilege.
#local   replication     postgres                                peer
#host    replication     postgres        127.0.0.1/32            ident
#host    replication     postgres        ::1/128                 ident
```

Placement of Authentication Settings

Ensure the local entries for icinga authentication settings are placed above all other local entries or you will get an error when configuring the database schema. The entries in `pg_hba.conf` are read from top to bottom.

7. Reload the Postgres service:

```
systemctl reload postgresql.service
```

8. Configure the database schema by running the following command in `/usr/share/doc/packages/icinga-idoutils-pgsql/pgsql/`:

```
psql -U icinga -d icinga < pgsql.sql
```

9. Edit the following lines in `/etc/icinga/ido2db.cfg` to switch from the default setting of mysql to postgres:

```
vi /etc/icinga/ido2db.cfg

 db_servertype=pgsql
 db_port=5432
```

Open Firewall Port

Allow port 5432 through your firewall or you will not be able to access the WebGUI.

10. Create an icinga admin account for logging into the web interface:

```
htpasswd -c /etc/icinga/htpasswd.users icingaadmin
```

11. Enable and start all required services:

```
systemctl enable icinga.service
systemctl start icinga.service
systemctl enable ido2db.service
systemctl start ido2db.service
systemctl enable apache2.service
systemctl start apache2.service
```

12. Login to the WebGUI at: http://localhost/icinga.

This concludes setup and initial configuration of Icinga.

Icinga and NRPE Quickstart

The following sections provides an overview on monitoring your Uyuni server using Icinga. You will add Uyuni as a host to Icinga and use a Nagios script/plugin to monitor running services via NRPE (Nagios Remote Plugin Executor). This section does not attempt to cover all monitoring solutions Icinga has to offer but should help you get started.

Procedure: Adding Uyunito Icinga for Monitoring

1. On your Uyuni server install the required packages:

```
zypper install nagios-nrpe susemanager-nagios-plugin insserv nrpe monitoring-plugins-
nrpe
```

2. Modify the NRPE configuration file located at:

```
/etc/nrpe.cfg
```

Edit or add the following lines:

```
server_port=5666
nrpe_user=nagios
nrpe_group=nagios
allowed_hosts=Icinga.example.com
dont_blame_nrpe=1
command[check_systemd.sh]=/usr/lib/nagios/plugins/check_systemd.sh $ARG1$
```

Variable definitions:

server_port

The variable `server_port` defines the port nrpe will listen on. The default port is 5666. This port must be opened in your firewall.

nrpe_user

The variables `nrpe_user` and `nrpe_group` control the user and group IDs that nrpe will run under. Uyuni probes need access to the database, therefore nrpe requires access to database credentials stored in `/etc/rhn/rhn.conf`. There are multiple ways to achieve this. You may add the user `nagios` to the group `www` (this is already done for other IDs such as tomcat); alternatively you can simply have nrpe run with the effective group ID `www` in `/etc/rhn/rhn.conf`.

allowed_hosts

The variable `allowed_hosts` defines which hosts nrpe will accept connections from. Enter the FQDN or IP address of your Icinga server here.

dont_blame_nrpe

The use of variable `dont_blame_nrpe` is unavoidable in this example. `nrpe` commands by default will not allow arguments being passed due to security reasons. However, in this example you should pass the name of the host you want information on to nrpe as an argument. This action is only possible when setting the variable to 1.

command[check_systemd.sh]

You need to define the command(s) that nrpe can run on Uyuni. To add a new nrpe command specify a command call by adding `command` followed by square brackets containing the actual nagios/icinga plugin name. Next define the location of the script to be called on your Uyuni server. Finally the variable `$ARG1$` will be replaced by the actual host the Icinga server would like information about. In the example above, the command is named `check_systemd.sh`. You can specify any name you like but keep in mind the command name is the actual script stored in `/usr/lib/nagios/plugins/` on your Uyuni server. This name must also match your probe definition on the Icinga server. *This will be described in greater detail later in the chapter. The check_systemd.sh script/plugin will also be provided in a later section.*

3. One your configuration is complete load the new nrpe configuration as root with:

```
systemctl start nrpe
```

This concludes setup of nrpe.

Add a Host to Icinga

To add a new host to Icinga create a host.cfg file for each host in `/etc/icinga/conf.d/`. For example `susemanager.cfg`:

```
define host {
  host_name           susemanager
  alias               SUSE Manager
  address             192.168.1.1
  check_period        24x7
  check_interval      1
  retry_interval      1
  max_check_attempts  10
  check_command       check-host-alive
}
```

Place the host IP address you want to add to Icinga on the `Address` line.

After adding a new host restart Icinga as root to load the new configuation:

```
systemctl restart icinga
```

Adding Services to Icinga

To add services for monitoring on a specific host define them by adding a service definition to your host.cfg file located in `/etc/icinga/conf.d`. For example you can monitor if a systems SSH service is running with the following service definition.

```
define service {
  host_name           susemanager
  use                 generic-service
  service_description SSH
  check_command       check_ssh
  check_interval      60
}
```

After adding any new services restart Icinga as root to load the new configuration:

```
systemctl restart icinga
```

Creating Icinga Hostgroups

You can create hostgroups to simplify and visualize hosts logically. Create a `hostgroups.cfg` file located in `/etc/icinga/conf.d/` and add the following lines:

```
define hostgroup {
  hostgroup_name  ssh_group
  alias           ssh group
  members         susemanager,mars,jupiter,pluto,examplehost4
}
```

The `members` variable should contain the `host_name` from within each host.cfg file you created to represent your hosts. Every time you add an additional host by creating a host.cfg ensure you add the host_name to the members list of included hosts if you want it to be included within a logical hostgroup.

After adding several hosts to a hostgroup restart Icinga as root to load the new configuration:

```
systemctl restart icinga
```

Creating Icinga Servicegroups

You can create logical groupings of services as well. For example if you would like to create a group of essential Uyuni services which are running define them within a `servicegroups.cfg` file placed in `/etc/icinga/conf.d/`:

```
#Servicegroup 1
define servicegroup {
  servicegroup_name     SUSE Manager Essential Services
  alias                 Essential Services
}

#Servicegroup 2
define servicegroup {
  servicegroup_name     Client Patch Status
  alias                 SUSE Manager 3 Client Patch Status
}
```

Within each host's `host.cfg` file add a service to a servicegroup with the following variable:

```
define service {
  use                 generic-service
  service_description SSH
  check_command       check_ssh
  check_interval      60
  servicegroups       SUSE Manager Essential Services
}
```

All services that include the `servicegroups` variable and the name of the servicegroup will be added to the specified servicegroup. After adding services to a servicegroup restart Icinga as root to load the new configuation:

```
systemctl restart icinga
```

Monitoring Systemd Services

The following section provides information on monitoring uptime of critical Uyuni services.

Procedure: Monitoring Running Systemd Services

1. As root create a new plugin file called `check_systemd.sh` in `/usr/lib/nagios/plugins/` on your Uyuni server:

```
vi /usr/lib/nagios/plugins/ check_systemd.sh
```

2. For this example you will use an opensource community script to monitor Systemd services. You may also wish to write your own.

```
#!/bin/bash
# Copyright (C) 2016 Mohamed El Morabity <melmorabity@fedoraproject.com>
#
# This module is free software: you can redistribute it and/or modify it under
# the terms of the GNU General Public License as published by the Free Software
# Foundation, either version 3 of the License, or (at your option) any later
# version.
#
# This software is distributed in the hope that it will be useful, but WITHOUT
# ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS
# FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License along with
# this program. If not, see <http://www.gnu.org/licenses/>.

PLUGINDIR=$(dirname $0)
. $PLUGINDIR/utils.sh


if [ $# -ne 1 ]; then
    echo "Usage: ${0##*/} <service name>" >&2
    exit $STATE_UNKNOWN
fi

service=$1

status=$(systemctl is-enabled $service 2>/dev/null)
r=$?
if [ -z "$status" ]; then
    echo "ERROR: service $service doesn't exist"
    exit $STATE_CRITICAL
fi

if [ $r -ne 0 ]; then
    echo "ERROR: service $service is $status"
    exit $STATE_CRITICAL
fi

systemctl --quiet is-active $service
if [ $? -ne 0 ]; then
    echo "ERROR: service $service is not running"
    exit $STATE_CRITICAL
fi

echo "OK: service $service is running"
exit $STATE_OK
```

A current version of this script can be found at: https://github.com/melmorabity/nagios-plugin-systemd-service/blob/master/check_systemd_service.sh

Non-supported 3rd Party Plugin

The script used in this example is an external script and is not supported by SUSE.

Always check to ensure scripts are not modified or contain malicous code before using them on production machines.

3. Make the script executable:

```
chmod 755 check_systemd.sh
```

4. On your SUSE manager server add the following line to the `nrpe.cfg` located at `/etc/nrpe.cfg` :

```
# SUSE Manager Service Checks
command[check_systemd.sh]=/usr/lib/nagios/plugins/check_systemd.sh $ARG1$
```

This will allow the Icinga server to call the plugin via nrpe on Uyuni.

5. Provide proper permissions by adding the script to the sudoers file:

```
visudo
```

```
nagios  ALL=(ALL)       NOPASSWD:/usr/lib/nagios/plugins/check_systemd.sh
Defaults:nagios !requiretty
```

You can also add permissions to the entire plugin directory instead of allowing permissions for individual scripts:

```
nagios  ALL=(ALL)       NOPASSWD:/usr/lib/nagios/plugins/
```

6. On your Icinga server define the following command within `/etc/icinga/objects/commands.cfg` :

```
define command {
        command_name    check-systemd-service
        command_line    /usr/lib/nagios/plugins/check_nrpe -H $HOSTADDRESS$ -c
check_systemd.sh -a $ARG1$
}
```

7. Now you will add the following critical services to be montitored to your Uyuni host file:
    - auditlog-keeper.service
    - jabberd.service
    - spacewalk-wait-for-jabberd.service
    - tomcat.service
    - spacewalk-wait-for-tomcat.service
    - salt-master.service
    - salt-api.service
    - spacewalk-wait-for-salt.service
    - apache2.service
    - osa-dispatcher.service
    - rhn-search.service
    - cobblerd.service
    - taskomatic.service
    - spacewalk-wait-for-taskomatic.service

    On your Icinga server add the following service blocks to your Uyuni host file `susemanager.cfg` file located in `/etc/icinga/conf.d/`. (This configuration file was created in the previous section *Adding a Host to Icinga*.)

```
# Monitor Audit Log Keeper
define service {
       use                     generic-service
       host_name               susemanager
       check_interval          1
       active_checks_enabled   1
       service_description     Audit Log Keeper Service
       servicegroups           SUSE Manager Essential Services
       check_command           check-systemd-service!auditlog-keeper.service

}

# Monitor Jabberd
define service {
       use                     generic-service
       host_name               susemanager
       check_interval          1
       active_checks_enabled   1
       service_description     Jabberd Service
       servicegroups           SUSE Manager Essential Services
       check_command           check-systemd-service!jabberd.service

}

# Monitor Spacewalk Wait for Jabberd
define service{
       use                     generic-service
       host_name               susemanager
       check_interval          1
```

```
        active_checks_enabled  1
        service_description    Spacewalk Wait For Jabberd Service
        servicegroups          SUSE Manager Essential Services
        check_command          check-systemd-service!spacewalk-wait-for-
jabberd.service
}

# Monitor Tomcat
define service{
        use                    generic-service
        host_name              susemanager
        check_interval         1
        active_checks_enabled  1
        service_description    Tomcat Service
        servicegroups          SUSE Manager Essential Services
        check_command          check-systemd-service!tomcat.service
}

# Monitor Spacewalk Wait for Tomcat
define service{
        use                    generic-service
        host_name              susemanager
        check_interval         1
        active_checks_enabled  1
        service_description    Spacewalk Wait For Tomcat Service
        servicegroups          SUSE Manager Essential Services
        check_command          check-systemd-service!spacewalk-wait-for-
tomcat.service
}

# Monitor Salt Master
define service{
        use                    generic-service
        host_name              susemanager
        check_interval         1
        active_checks_enabled  1
        service_description    Salt Master Service
        servicegroups          SUSE Manager Essential Services
        check_command          check-systemd-service!salt-master.service
}

# Monitor Salt API
define service{
        use                    generic-service
        host_name              susemanager
        check_interval         1
        active_checks_enabled  1
        service_description    Salt API Service
        servicegroups          SUSE Manager Essential Services
        check_command          check-systemd-service!salt-api.service
}

# Monitor Spacewalk Wait for Salt
define service{
        use                    generic-service
        host_name              susemanager
        check_interval         1
        active_checks_enabled  1
        service_description    Spacewalk Wait For Salt Service
        servicegroups          SUSE Manager Essential Services
        check_command          check-systemd-service!spacewalk-wait-for-salt.service
}

# Monitor apache2
define service{
        use                    generic-service
        host_name              susemanager
        check_interval         1
```

```
        active_checks_enabled  1
        service_description    Apache2 Service
        servicegroups          SUSE Manager Essential Services
        check_command          check-systemd-service!apache2.service
}

# Monitor osa dispatcher
define service{
        use                    generic-service
        host_name              susemanager
        check_interval         1
        active_checks_enabled  1
        service_description    Osa Dispatcher Service
        servicegroups          SUSE Manager Essential Services
        check_command          check-systemd-service!osa-dispatcher.service
}

# Monitor rhn search
define service{
        use                    generic-service
        host_name              susemanager
        check_interval         1
        active_checks_enabled  1
        service_description    RHN Search Service
        servicegroups          SUSE Manager Essential Services
        check_command          check-systemd-service!rhn-search.service
}

# Monitor Cobblerd
define service{
        use                    generic-service
        host_name              susemanager
        check_interval         1
        active_checks_enabled  1
        service_description    Cobblerd Service
        servicegroups          SUSE Manager Essential Services
        check_command          check-systemd-service!cobblerd.service
}

# Monitor taskomatic
define service{
        use                    generic-service
        host_name              susemanager
        check_interval         1
        active_checks_enabled  1
        service_description    Taskomatic Service
        servicegroups          SUSE Manager Essential Services
        check_command          check-systemd-service!taskomatic.service
}

# Monitor wait for taskomatic
define service{
        use                    generic-service
        host_name              susemanager
        check_interval         1
        active_checks_enabled  1
        service_description    Spacewalk Wait For Taskomatic Service
        servicegroups          SUSE Manager Essential Services
        check_command          check-systemd-service!spacewalk-wait-for-
taskomatic.service
}
```

Each of these service blocks will be passed as the check-systemd-service!$ARG1$ variable to SUSE manager server via nrpe. You probably noticed the servicegroups parameter was also included. This adds each service to a servicegroup and has been defined in a

`servicesgroups.cfg` file located in `/etc/icinga/conf.d/`:

```
define servicegroup {
       servicegroup_name     SUSE Manager Essential Services
       alias                 Essential Services
}
```

8. Restart Icinga:

```
systemctl restart icinga
```

Using the check_suma_patches Plugin

You can use the `check_suma_patches` plugin to check if any machines connected to Uyuni as clients require a patch or an update. The following procedure will guide you through the setup of the check_suma_patches plugin.

Procedure: Setup check_suma_patches

1. On your Uyuni server open `/etc/nrpe.cfg` and add the following lines:

```
# SUSE Manager check_patches
command[check_suma_patches]=sudo /usr/lib/nagios/plugins/check_suma_patches $ARG1$
```

2. On your Icinga server open `/etc/icinga/objects/commands.cfg` and define the following command:

```
define command{
        command_name    check_suma
        command_line    /usr/lib/nagios/plugins/check_nrpe -H 192.168.1.1 -c $ARG1$ -a
$HOSTNAME$
}
```

3. On your Icinga server open any of your Uyuni client host configration files located at `/etc/icinga/conf.d/clients.cfg` and add the following service definition:

```
define service {
        use                             generic-service
        host_name                       client-hostname
        service_description             Available Patches for client-host_name
        servicegroups                   Client Patch Status
        check_command                   check_suma!check_suma_patches
}
```

4. In the above service definition notice that this host is included in the servicegroup labeled *Client Patch Status*. Add the following servicegroup definition to `/etc/icinga/conf.d/servicegroups.cfg` to create a servicegroup:

```
define servicegroup {
       servicegroup_name     Client Patch Status
       alias                 SUSE Manager 3 Client Patch Status
}
```

5.
   - `OK:System is up to date`
   - `Warning: At least one patch or package update is available`
   - `Critical:At least one security/critical update is available`
   - `Unspecified:The host cannot be found in the SUSE Manager database or the host name is not unique`

This concludes setup of the `check_suma_patches` plugin.

Using the check_suma_lastevent Plugin

You can use the `check_suma_lastevent` plugin to display the last action executed on any host.

The following procedure will guide you through the setup of the check_suma_patches plugin.

Procedure: Setup check_suma_lastevent

1. On your Uyuni server open `/etc/nrpe.cfg` and add the following lines:

```
# Check SUSE Manager Hosts last events
command[check_events]=sudo /usr/lib/nagios/plugins/check_suma_lastevent $ARG1$
```

2. On the Icinga server open `/etc/icinga/objects/commands.cfg` and add the following lines:

```
define command {
        command_name    check_events
        command_line    /usr/lib/nagios/plugins/check_nrpe -H manager.suse.de -c $ARG1$
-a $HOSTNAME$
}
```

3. On your Icinga server add the following line to a host.cfg service definition:

```
define service{
        use                             generic-service
        host_name                       hostname
        service_description             Last Events
        check_command                   check_events!check_suma_lastevent
}
```

4. Status will be reported as follows:
   - `OK:Last action completed successfully`

- `Warning: Action is currently in progress`
- `Critical:Last action failed`
- `Unspecified:The host cannot be found in the Uyuni database or the host name is not unique`

This concludes setup of the `check_suma_lastevent` plugin.

Additional Resources

For more information, see Icinga's official documentation located at http://docs.icinga.org/latest/en.

For some excellent time saving configuration tips and tricks not covered in this guide, see the following section located within the official documentation: http://docs.icinga.org/latest/en/objecttricks.html

Kubernetes

Prerequisites

The prerequisites listed below should be met before proceeding.

- At least one *Kubernetes* or _SUSE CaaS Platform _ cluster available on your network
- Uyuni configured for container management

Required channels are present, a registered build host available etc.

- virtual-host-gatherer-Kubernetes package installed on your Uyuni server

Requirements

- Kubernetes version 1.5.0 or higher. Alternatively use SUSE CaaS Platform *(SUSE CaaS Platform includes Kubernetes 1.5.0 by default)*
- Docker version 1.12 or higher on the container build host

To enable all Kubernetes related features within the Web UI, the virtual-host-gatherer-Kubernetes package must be installed.

Register Kubernetes as a Virtual Host Manager

Kubernetes clusters are registered with SUSE Manager as `virtual host managers`. Registration and authorization begins with importing a `kubeconfig` file using Kubernetes official command line tool `kubectl`.

Procedure: Registering a Kubernetes Cluster with Uyuni

1. Select **Systems › Virtual Host Managers** from the navigation menu.
2. Expand the `Create` dropdown in the upper right corner of the page and select **Kubernetes Cluster** .
3. Input a label for the new Virtual Host Manager.
4. Select the `kubeconfig` file which contains the required data for the Kubernetes cluster.
5. Select the correct *context* for the cluster, as specified in the kubeconfig file.
6. Click `Create`.

View the List of Nodes in a Cluster

1. Select **Systems › Virtual Host Managers** from the navigation menu.
2. Select the desired Kubernetes cluster to view it.

3. Node data is not refreshed during registration. To refresh node data, click on `Schedule refresh data`.
4. Refresh the browser. If the node data is not available wait a few moments and try again.

Obtain Runtime Data about Images

See the following steps to find runtime data for images.

1. Select **Images › Images** from the navigation menu.
2. In the image list table, take notice of the new runtime columns. These are labeled: `Revision`, `Runtime` and `Instances`. Initially these columns will not provide useful data.
    - `Revision`: An artificial sequence number which increments on every rebuild for manager-built images, or on every reimport for externally built images.
    - `Runtime`: Overall status of the running instances of the image throughout the registered clusters. The status can be one of the following:
        - All instances are consistent with SUSE Manager: All the running instances are running the same build of the image as tracked by SUSE Manager.
        - Outdated instances found: Some of the instances are running an older build of the image. A redeploy of the image into the pod may be required.
        - No information: The checksum of the instance image does not match the image data contained in SUSE Manager. A redeploy of the image into the pod may be required.
    - `Instances`: Number of instances running this image across all the clusters registered in SUSE Manager. A breakdown of numbers can be seen by clicking on the pop-up icon next to the number.

Build an image for deployment in Kubernetes

The following steps will help you build an image for deployment in Kubernetes.

1. Under **Images › Stores**, create an image store.
2. Under **Images › Profiles**, create an image profile (with a Dockerfile which is suitable to deploy to Kubernetes).
3. Under **Images › Build**, build an image with the new profile and wait for the build to finish.
4. Deploy the image into one of the registered Kubernetes clusters (via `kubectl`).
5. Notice the updated data in `Runtime` and `Instances` columns in the respective image row.

Import a Previously Deployed Image in Kubernetes

The following steps will guide you through importing a previously deployed image in Kubernetes.

1. Select an image that has already been deployed to any of your registered Kubernetes clusters.
2. Add the registry owning the image to SUSE Manager as an image store.
3. Select **Images › Images** , click `Import` from the top-right corner, fill in the form fields and click `Import`.
4. Notice the updated data in `Runtime` and `Instances` columns in the respective image row.

Obtain Additional Runtime Data

The following steps will help you find additional runtime data.

1. Select to **Images › Images** , click the `Details` button on the right end of a row which has running instances.
2. Under the `Overview` tab, notice the data in `Runtime` and `Instances` fields under `Image Info` section.
3. Select the `Runtime` tab.
4. Here is a breakdown of the Kubernetes pods running this image in all the registered clusters including the following data:
    - Pod name
    - Namespace which the pod resides in
    - The runtime status of the container in the specific pod. Status icons are explained in the preceeding example.

Rebuild a Previously Deployed Image in Kubernetes

The following steps will guide you through rebuilding an image which has been deployed to a Kubernetes cluster.

1. Go to **Images › Images** , click the Details button on the right end of a row which has running instances. The image must be manager-built.
2. Click the `Rebuild` button located under the `Build Status` section and wait for the build to finish.
3. Notice the change in the `Runtime` icon and title, reflecting the fact that now the instances are running a previous build of the image.

Role Based Access Control Permissions and Certificate Data

Currently, only kubeconfig files containing all embedded certificate data may be used with SUSE Manager

The API calls from Uyuni are:

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

According to this list, the minimum recommended permissions for Uyuni should be as follows:

- A ClusterRole to list all the nodes:

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

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

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

Due to a a 403 response from /pods, the entire cluster will be ignored by SUSE Manager.

For more information on working with RBAC Authorization see: https://kubernetes.io/docs/admin/authorization/rbac/

Virtual Hosts

Inventorying vCenter/vSphere ESXi Hosts with Uyuni

Foreign virtual hosts (such as vCenter and vSphere ESXi) can be inventoried using the `Virtual Host Manager`. From the vSphere Client you can define roles and permissions for vCenter and vSphere ESXi users allowing vSphere objects and resources to be imported and inventoried by Uyuni. Objects and resources are then displayed as foreign hosts on the Uyuni **Systems › Virtual Systems** page.

The following sections will guide you through:

- Requirements
- Overview of permissions and roles
- Adding vCenter and vSphere ESXi hosts to Uyuni

Requirements

This table displays the default API communication port and required access rights for inventorying objects and resources:

Ports / Permissions	Description
443	Default port that Uyuni uses to access the ESXi API for obtaining infrastructure data
read-only	All vCenter/ESXi objects and resources that should be inventoried by the Virtual Host Manager should be at least assigned the *read-only* role. Mark objects and resources with *no-access* to exclude them from the inventory.

Permissions and Roles Overview

This section will guide you through assigning user permissions and roles in vCenter/ESXi.

A user is someone who has been authorized to access an ESXi host. The Virtual Host Manager (located on the SUSE Manager server) will inventory ESXi data defined by assigned roles and permissions on a user account.

For example: The user *John* has been assigned the *read-only* access role to all servers and datacenters in his company with one exception. John's account has been assigned the *no-access* role on the company's *Financial Database server*. You decide to use John's user account and add the ESXi host to SUSE Manager. During the inventory the *Financial Database server* will be excluded.

Keep user access roles in mind when planning to add ESXi hosts to SUSE manager. Note that SUSE Manager will not inventory any objects or resources assigned with the *no-access* role on any user account.

User Roles/Permissions

When planning to add new ESXi hosts to SUSE Manager, consider if the roles and permissions assigned users require need to be inventoried by SUSE Manager.

Adding New Users and Assigning Roles

See the official vSphere documentation on adding new users and assigning roles.

- Authentication and User Management

Inventorying vCenter/vSphere ESXi Hosts

This procedure guides you through inventorying a VSphere ESXi host with Uyuni.

1. From the Uyuni Web UI select **Main Menu › Systems › Virtual Host Managers** from the left navigation bar.
2. From the upper right corner of the *Virtual Host Managers* page select **[Create]** VMWare-based.
3. From the *Add a VMware-based Virtual Host Manager* page complete these fields with your ESXi host data:

Label

Custom name for your Virtual Host Manager

Hostname

Fully-qualified domain name (FQDN) or host IP address

Port

Default ESXi API port

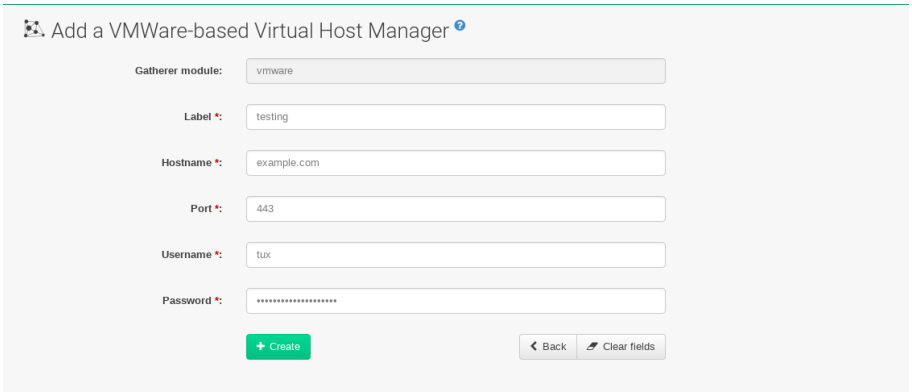
Username

Assign a username

Remember that only objects and resources which match a user's defined role will be inventoried. Set the user's role on objects and resources you want inventoried to *read-only*.

Password

ESXi users password



4. Click the **[Create]** button.
5. From the **Systems › Virtual Host Managers** page select the new Virtual Host manager.
6. From the **Virtual Host Managers › Properties** page click the **[Refresh]** button.

If you do not refresh the data from a new Virtual Host Manager, host data will not be inventoried and therefore will not be displayed under **Systems › Virtual Systems**.

7. View inventoried ESXi host objects and resources by selecting **Systems › Virtual Systems** .

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: `http://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-org-ssl`).
3. 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, provide these details for the Server to use as the ISS slave:
    - In the `Slave Fully-Qualified Domain Name` field, enter the FQDN of the ISS slave (for example: `http://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 master, with this command:

```
mgr-inter-sync
```

Setup a Minion to Master Validation Fingerprint

In highly secure network configurations you may wish to ensure your minions are connecting a specific master. To setup validation from minion to master enter the masters fingerprint within the `/etc/salt/minion` configuration file. See the following procedure:

1. On the master enter the following command as root and note the fingerprint:

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

   On your minion, open the minion configuration file located in `/etc/salt/minion`. Uncomment the following line and enter the masters 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 more information on configuring security from a minion see: https://docs.saltstack.com/en/latest/ref/configuration/minion.html

Signing Repository Metadata

TODO

Explain why repository metadata should/would be signed.

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/keyname.key
rpm --import http://server.example.com/pub/company.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-minion> state.apply channels
```

3. OPTIONAL: For salt minions, 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.

Note that this 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
```

1. 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. 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

Authentication Via PAM

As security measures become increasingly complex, SUSE Manager supports network-based authentication systems via Pluggable Authentication Modules (PAM). PAM is a suite of libraries that allows to integrate SUSE Manager with a centralized authentication mechanism, thus eliminating the need to remember multiple passwords. SUSE Manager supports LDAP, Kerberos, and other network-based authentication systems via PAM. To enable SUSE Manager to use PAM in your organization's authentication infrastructure, follow the steps below.

1. Set up a PAM service file (default location: `/etc/pam.d/susemanager`) then enforce its use by adding the following line to `/etc/rhn/rhn.conf` :

```
pam_auth_service = susemanager
```

This assumes the PAM service file is named susemanager.

2. To enable a new or existing user to authenticate with PAM, proceed to the **Create User** page and select the checkbox labeled Pluggable Authentication Modules (PAM) positioned below the password and password confirmation fields.
3. To authenticate a SLES system against Kerberos add the following lines to `/etc/pam.d/susemanager` :

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

To register a Red Hat Enterprise Linux System against Kerberos add the following lines to `/etc/pam.d/susemanager`

```
#%PAM-1.0
 auth        required      pam_env.so
 auth        sufficient    pam_krb5.so no_user_check
 auth        required      pam_deny.so
 account     required      pam_krb5.so no_user_check
```

+

1. YaST can now be used to configure PAM, when packages such as yast2-ldap-client and yast2-kerberos-client are installed; for detailed information on configuring PAM, see the SUSE Linux Enterprise Server Security Guide https://www.suse.com/documentation/sles-12/book_security/data/

part_auth.html. This example is not limited to Kerberos; it is generic and uses the current server configuration. Note that only network based authentication services are supported.

Changing Passwords

Changing the password on the SUSE Manager Web interface changes only the local password on the SUSE Manager server. But this password may not be used at all if PAM is enabled for that user. In the above example, for instance, the Kerberos password will not be changed.

Authentication Via eDirectory and PAM

1. First check to ensure eDirectory authentication is working with your current OS for example:

```
#getent passwd
```

2. If users are returned from eDirectory then create the following file:

```
# cat /etc/pam.d/susemanager
```

3. And add the following content:

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

4. Finally add the following lines to the Uyuni conf file:

```
# grep -i pam /etc/rhn/rhn.conf
 pam_auth_service = susemanager
```

5. You may now create users with the same id that appears on eDirectory and mark the Use PAM check-box from the SUSE Manager WebUI.

Example Quest VAS Active Directory Authentication Template

If you are using Quest VAS for active directory authentication, you can use the following `/etc/pam.d/susemanager` file.

```
#%PAM-1.0
auth       required       pam_env.so
auth       sufficient     pam_vas3.so no_user_check
auth       requisite      pam_vas3.so echo_return
auth       required       pam_deny.so
account    required       pam_vas3.so no_user_check
```

Authentication Via 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, 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 (please check with the respective product documentation).

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.

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. Copy parameters you want to customize from `/usr/share/rhn/config-defaults/rhn_java_sso.conf` into `/etc/rhn/rhn.conf` and proceed by inserting the parameters you want to customize by prefixing them with `java.sso.`.

Example:

1. in `/usr/share/rhn/config-defaults/rhn_java_sso.conf`:

```
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. . Restart 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.

Using a Custom SSL Certificate

The following section will guide you through using a custom certificate with Uyuni 4.0 and SUSE Manager Proxy 4.0.

Prerequisites

The following list provides requirements for using a custom certificate.

- A Certificate Authority (CA) SSL public certificate file
- A Web server SSL private key file
- A Web server SSL public certificate file
- Key and Certificate files must be in PEM format

Hostname and SSL Keys

The hostname of the web server's SSL keys and relevant certificate files must match the hostname of the machine which they will be deployed on.

Intermediate Certificates

In case you want to use CAs with intermediate certificates, merge the intermediate and root CA certificates into one file. It is important that the intermediate certificate comes first within the combined file.

Setup

After completing YaST firstboot procedures, export your current environment variables and point them to the correct SSL files to be imported. Running these commands will make the default certificate obsolete after executing the `yast2 susemanagersetup` command. For more information on YaST firstboot, see https://www.suse.com/documentation/suse-manager-3/singlehtml/suse_manager21/book_susemanager_install/book_susemanager_install.html#sec.manager.inst.setup.

1. Export the environment variables and point to the SSL files to be imported:

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

2. Execute Uyuni setup with

```
yast2 susemanagersetup
```

   Proceed with the default setup. Upon reaching the Certificate Setup window during YaST installation, fill in random values, as these will be overridden with the values specified in

[bp.cert.custom.setup.proc.export].

Shell Requirements

Make sure that you execute `yast2 susemanagersetup` from within the same shell the environment variables were exported from.

Using a Custom Certificate with SUSE Manager Proxy

After completing the installation with yast found in [advanced.topics.proxy.quickstart] continue with a modified [at.manager.proxy.run.confproxy] procedure:

1. Execute `configure-proxy.sh`.
2. When prompted with:

   ```
   Do you want to import existing certificates?
   ```

   Answer with `y` .
3. Continue by following the script prompts.

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 profiles, AutoYaST, and custom RPMs.

You will also need to back up your database, which you can do by using the `smdba` tool, which is explained in Administering the Database with smdba.

Procedure: Restore from a Manual Backup

1. Re-install Uyuni. For more information, see Recovering from a Crashed Root Partition.
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. If you did not have /var/spacewalk/packages/ in your backup, but the source repository still exists, you can restore it by performing a complete channel synchronization with:

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

You can 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 the smdba db-status command. This command will return either online or offline:

```
smdba db-status
Checking database core...       online
```

Starting and stopping the database can be performed with smdba db-start and smdba db-stop.

```
smdba db-start
Starting core...       done
```

```
smdba db-stop
Stopping the SUSE Manager database...
Stopping core:         done
```

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 run 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:

```
# 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

In SUSE Manager with an embedded database, archive logging is enabled by default. This feature allows the database management tool `smdba` to perform hot backups.

With archive log enabled, even more data is stored on the hard disk:

- 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
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
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 if your database storage space is running low. The following procedure will guide you through moving the database to a new location for use by SUSE Manager.

Procedure: Moving the Database

1. The default storage location for SUSE Manager 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:

   ```
   # 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 SUSE Manager database may malfunction. You also should ensure 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
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
```

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 Getting Started 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. Start by installing SLES12 SP2 and the SUSE Manager Extension. Do not mount the `/var/spacewalk` and `/var/lib/pgsql` partitions.
2. Once installation of SUSE Manager has completed shutdown services with `spacewalk-service shutdown` and the database with `rcpostgresql stop`.

3. Mount your `/var/spacewalk` and `/var/lib/pgsql` partitions and restore the directories listed in section one.
4. Start SUSE Manager services and the database with `spacewalk-services start` and `rcpostgresql start`
5. SUSE Manager should now operate normally without loss of your database or synced channels.

Database Connection Information

The information for connecting to the SUSE Manager 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 =
```

Tuning Apache and Tomcat

Altering Apache and Tomcat Parameters

Apache and Tomcat Parameters should only be modified with support or consulting as these parameters can have severe and catastrophic performance impacts on your server when improperly adjusted. SUSE will not be able to provide support for catastrophic failure when these advanced parameters are modified without consultation. Tuning values for Apache httpd and Tomcat requires that you align these parameters with your server hardware. Furthermore testing of these altered values should be performed within a test environment.

Apache's httpd MaxClients Parameter

The `MaxClients` setting determines the number of Apache httpd processes, and thus limits the number of client connections that can be made at the same time (SUSE Manager uses the pre-fork MultiProcessing Modules). The default value for `MaxClients` in SUSE Manager is 150. If you need to set the `MaxClients` value greater than 150, Apache httpd's ServerLimit setting and Tomcat's `maxThreads` must also be increased accordingly (see below).

The Apache httpd `MaxClients` parameter must always be less or equal than Tomcat's `maxThreads` parameter!

If the `MaxClients` value is reached while the software is running, new client connections will be queued and forced to wait, this may result in timeouts. You can check the Apache httpd's `error.log` for details:

```
[error] Server reached MaxClients setting, consider increasing the MaxClients setting
```

The default `MaxClients` parameter can be overridden on SUSE Manager by editing the `server-tuning.conf` file located at `/etc/apache2/`. For example `server-tuning.conf` file:

```
# prefork MPM
   <IfModule prefork.c>
           # number of server processes to start
           # http://httpd.apache.org/docs/2.2/mod/mpm_common.html#startservers
           StartServers         5
           # minimum number of server processes which are kept spare
           # http://httpd.apache.org/docs/2.2/mod/prefork.html#minspareservers
           MinSpareServers      5
           # maximum number of server processes which are kept spare
           # http://httpd.apache.org/docs/2.2/mod/prefork.html#maxspareservers
           MaxSpareServers     10
           # highest possible MaxClients setting for the lifetime of the Apache process.
           # http://httpd.apache.org/docs/2.2/mod/mpm_common.html#serverlimit
           ServerLimit        150
           # maximum number of server processes allowed to start
           # http://httpd.apache.org/docs/2.2/mod/mpm_common.html#maxclients
           MaxClients         150
           # maximum number of requests a server process serves
           # http://httpd.apache.org/docs/2.2/mod/mpm_common.html#maxrequestsperchild
           MaxRequestsPerChild  10000
   </IfModule>
```

Whenever the Apache httpd `MaxClients` parameter is changed, the `ServerLimit` must also be updated to the same value, or the change will have no effect.

Tomcat's maxThreads Parameter

Tomcat's `maxThreads` represents the maximum number of request processing threads that it will create. This value determines the maximum number of simultaneous requests that it is able to handle. All HTTP requests to the SUSE Manager server (from clients, browsers, XMLRPC API scripts, etc.) are handled by Apache httpd, and some of them are routed to Tomcat for further processing. It is thus important that Tomcat is able to serve the same amount of simultaneous requests that Apache httpd is able to serve in the worst case. The default value for SUSE Manager is 200 and should always be equal or greater than Apache httpd's `MaxClients`. The `maxThreads` value is located within the `server.xml` file located at `/etc/tomcat/`.

Example relevant lines in `server.xml`:

```
<Connector port="8009" protocol="AJP/1.3" redirectPort="8443" URIEncoding="UTF-8"
address="127.0.0.1" maxThreads="200" connectionTimeout="20000"/>
<Connector port="8009" protocol="AJP/1.3" redirectPort="8443" URIEncoding="UTF-8"
address="::1" maxThreads="200" connectionTimeout="20000"/>
```

Tuning Notes

When configuring Apache httpd's `MaxClients` and Tomcat's `maxThreads` parameters you should also take into consideration that each HTTP connection will need one or more database connections. If the RDBMS is not able to serve an adequate amount of connections, issues will arise. See the following equation for a rough calculation of the needed amount of database connections:

```
((3 * java_max) +  apache_max + 60)
```

Where:

- 3 is the number of Java processes the server runs with pooled connections (Tomcat, Taskomatic and Search)
- java_max is the maximum number of connections per Java pool (20 by default, changeable in `/etc/rhn/rhn.conf` via the hibernate.c3p0.max_size parameter)
- apache_max is Apache httpd's `MaxClients`
- 60 is the maximum expected number of extra connections for local processes and other uses

Tuning Large Deployments

In the following sections find considerations about a big scale deployment. In this context, a big scale comprises 1000 minions or more.

General Recommendations

SUSE recommends the following in a big scale Uyuni deployment:

- Uyuni servers should have at least 8 recent x86 cores, 32 GiB of RAM, and, most important, fast I/O devices such as at least an SSD (2 SSDs in RAID-0 are strongly recommended).
- Proxies with many minions (hundreds) should have at least 2 recent x86 cores and 16 GiB of RAM.
- Use one SUSE Manager Proxy per 500-1000 clients. Keep into account that download time depends on network capacity. Here is a rough example calculation with physical link speed of 1 GB/s:

  ```
  400 Megabytes  *        3000       /      119 Megabyte/s        / 60
  = 169 Minutes
  ```

  This is:

  ```
  Size of updates * Number of minions / Theoretical download speed / 60
  ```

- Depending on hardware you can accept hundreds of minion keys.
- Plan time for onboarding minions- at least one hour per 1000 minions.
- It is not recommended onboarding more than approx. 1000 minions directly to the Uyuni server- proxies should be used instead. This is because every minion can use up to 3 TCP connections simultaneously, and too many TCP connections can cause performance issues.
- If the following error appears in output of `dmesg`, you probably have an excessive number of minions attached to a single Uyuni server or proxy for the ARP cache to contain all of their addresses:

  ```
  kernel: neighbour table overflow
  ```

  In that case, increase the ARP cache values via `sysctl`, for example, by adding the following lines to `/etc/sysctl.conf`:

  ```
  net.ipv4.neigh.default.gc_thresh1 = 4096
  net.ipv4.neigh.default.gc_thresh2 = 8192
  net.ipv4.neigh.default.gc_thresh3 = 16384
  net.ipv4.neigh.default.gc_interval = 60
  net.ipv4.neigh.default.gc_stale_time = 120
  ```

Start Small and Scale Up

Always start small and scale up gradually. Keep the server monitored in order to identify possible issues early.

Tuning Proposals

SUSE proposes the following tuning settings in a big scale Uyuni deployment:

- Increase the maximum Tomcat heap memory to face a potentially long queue of Salt return results. Set 8 GiB instead of the current default 1 GiB: parameter `Xmx1G` in `/etc/sysconfig/tomcat` (affects onboarding and Action execution).
- Increase the number of Taskomatic workers, allowing to parallelize work on a high number of separate jobs. Set parameter `org.quartz.threadPool.threadCount = 100` in `/etc/rhn/rhn.conf` (affects onboarding and staging).
- Allow Taskomatic to check for runnable jobs more frequently to reduce latency. Set parameter `org.quartz.scheduler.idleWaitTime = 1000` in `/etc/rhn/rhn.conf` (affects onboarding, staging and Action execution).
- Increase Tomcat's Salt return result workers to allow parallelizing work on a high number of Salt return results. Set parameter `java.message_queue_thread_pool_size = 100` in `/etc/rhn/rhn.conf` (affects patching).
- Increase the number of PostgreSQL connections available to Java applications (Tomcat, Taskomatic) according to the previous parameters, otherwise extra workers will starve waiting for a connection. Set parameter `hibernate.c3p0.max_size = 150` in `/etc/rhn/rhn.conf` (affects all minion operations). Make sure enough PostgreSQL connections are configured before changing this parameter - refer to `smdba system-check autotuning --help` to get automatic tuning of the PostgreSQL configuration file while changing the number of available connections. Additional manual tuning is usually not necessary but might be required depending on scale and exact use cases.
- Increase the number of Taskomatic's `minion-action-executor` worker threads allowing to parallelize the scheduling of Actions to minions. Set parameter `taskomatic.com.redhat.rhn.taskomatic.task.MinionActionExecutor.parallel_threads = 8` in `/etc/rhn/rhn.conf` (affects all minion operations, especially staging).
- Increase Salt's presence ping timeouts if responses might come back later than the defaults. Set parameters `java.salt_presence_ping_timeout = 20` and `java.salt_presence_ping_gather_job_timeout = 20` in `/etc/rhn/rhn.conf` (affects all minion operations).
- Increase the number of Salt master workers so that more requests can run in parallel (otherwise Tomcat and Taskomatic workers will starve waiting for the Salt API, and Salt will not be able to serve files timely). Set parameter `worker_threads: 100` in `/etc/salt/master.d/susemanager.conf` (affects onboarding and patching).
    - Increase this parameter further if file management states fail with the error "Unable to manage file: Message timed out"

    - Note that Salt master workers can consume significant amounts of RAM (typically about 70 MB per worker). It is recommended to keep usage monitored when increasing this value and to do so in relatively small increments (eg. 20) until failures are no longer produced.

- Disable daily comparison of configuration files. Click on **Admin** › **Task Schedules**, then on the **[compare-configs-default]** link, then on the **[Disable Schedule]** button and finally on **[Delete Schedule]**.
- Increase the maximum heap memory for the search daemon to be able to index many minions. Set 4 GiB instead of the current default 512 MB: add `rhn-search.java.maxmemory=4096` in `/etc/rhn/rhn.conf` (affects background indexing only).

Note that increasing the number of Postgres connections will require more RAM, make sure the Uyuni server is monitored and swap is never used.

Also note the above settings should be regarded as guidelines-they have been tested to be safe but care should be exercised when changing them, and consulting support is highly recommended.

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 and, once all tests pass, apply 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 feature is not yet complete! The documentation for this feature is being offered as a preview of changes to come.

Managing Content Lifecycle Projects

Procedure: Creating a Content Lifecycle Project

1. In the Uyuni Web UI, navigate to **Content Lifecycle Management** › **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 will be 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 new 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.

Procedure: Using a Content Lifecycle Project

1. TBD …

Troubleshooting

Troubleshooting

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

Before you begin troubleshooting, you might want to produce some reports from your system to help you understand what is going on.

Producing Reports

The `spacewalk-report` command is used to produce a variety of reports for system administrators. These reports can be helpful for taking inventory of your entitlements, 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.

`spacewalk-reports` *Package*

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

`spacewalk-report` allows administrators to organize and display reports about content, systems, and user resources across SUSE Manager. Using `spacewalk-report`, you can receive reports on:

1. System Inventory: lists all of the systems registered to SUSE Manager.
2. Entitlements: lists all organizations on SUSE Manager, sorted by system or channel entitlements.
3. Patches: lists all the patches relevant to the registered systems and sorts patches by severity, as well as the systems that apply to a particular patch.
4. Users: lists all the users registered to SUSE Manager and any systems associated with a particular user.

`spacewalk-report` allows administrators to organize and display reports about content, systems, and user resources across SUSE Manager. To get the report in CSV format, run the following at the command line of your SUSE Manager server.

```
spacewalk-report report_name
```

The following reports are available:

Table 1. `spacewalk-report` *Reports*

Report	Invoked as	Description
Channel Packages	`channel-packages`	List of packages in a channel.
Channel Report	`channels`	Detailed report of a given channel.

Report	Invoked as	Description
Cloned Channel Report	`cloned-channels`	Detailed report of cloned channels.
Custom Info	`custom-info`	System custom information.

Report	Invoked as	Description
Entitlements	`entitlements`	Lists all organizations on SUSE Manager with their system or channel entitlements.
Patches in Channels	`errata-channels`	Lists of patches in channels.

Report	Invoked as	Description
Patches Details	`errata-list`	Lists all patches that affect systems registered to SUSE Manager.
All patches	`errata-list-all`	Complete list of all patches.

Report	Invoked as	Description
Patches for Systems	`errata-systems`	Lists applicable patches and any registered systems that are affected.
Host Guests	`host-guests`	List of host-guests mapping.

Report	Invoked as	Description
Inactive Systems	`inactive-systems`	List of inactive systems.
System Inventory	`inventory`	List of systems registered to the server, together with hardware and software information.

Report	Invoked as	Description
Kickstart Trees	`kickstartable-trees`	List of kickstartable trees.
All Upgradable Versions	`packages-updates-all`	List of all newer package versions that can be upgraded.

Report	Invoked as	Description
Newest Upgradable Version	`packages-updates-newest`	List of only newest package versions that can be upgraded.
Result of SCAP	`scap-scan`	Result of OpenSCAP sccdf eval.

Report	Invoked as	Description
Result of SCAP	`scap-scan-results`	Result of OpenSCAP sccdf eval, in a different format.
System Data	`splice-export`	System data needed for splice integration.

Report	Invoked as	Description
System Groups	`system-groups`	List of system groups.
Activation Keys for System Groups	`system-groups-keys`	List of activation keys for system groups.

Report	Invoked as	Description
Systems in System Groups	`system-groups-systems`	List of systems in system groups.
System Groups Users	`system-groups-users`	Report of system groups users.

Report	Invoked as	Description
Installed Packages	`system-packages-installed`	List of packages installed on systems.
Users in the System	`users`	Lists all users registered to SUSE Manager.

Report	Invoked as	Description
Systems administered	`users-systems`	List of systems 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 invocations and options, see the `spacewalk-report(8)` man page as well as the `--help`parameter of the `spacewalk-report`.

Troubleshooting Corrupt Repositories

The information in the repository data file can become corrupt or out of date. This can create problems with updating the server. 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/sles15-sp1-updates-x86_64`
2. Regenerate the file from the command line:

```
spacecmd softwarechannel_regenerateyumcache sles{sles-version}-{sp-version-l}-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. 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 [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 OSAD and jabberd

Cause: Consequence: Fix: Result:

Open File Count Exceeded

`SYMPTOMS`: OSAD clients cannot contact the SUSE Manager Server, and jabberd requires long periods of time to respond on port 5222.

`CAUSE`: The number of maximum files that a jabber user can open is lower than the number of connected clients. Each client requires one permanently open TCP connection and each connection requires one file handler. The result is jabberd begins to queue and refuse connections.

`CURE`: Edit the `/etc/security/limits.conf` to something similar to the following: `jabbersoftnofile<#clients + 100> jabberhardnofile<#clients + 1000>`

This will vary according to your setup. For example in the case of 5000 clients: `jabbersoftnofile5100 jabberhardnofile6000`

Ensure you update the `/etc/jabberd/c2s.xml` max_fds parameter as well. For example: `<max_fds>6000</max_fds>`

`EXPLANATION`: The soft file limit is the limit of the maximum number of open files for a single process. In SUSE Manager 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 the jabber user, and accounts for open files from the router, s2s and sm processes additionally.

jabberd Database Corruption

SYMPTOMS: After a disk is full error or a disk crash event, the jabberd database may have become corrupted. jabberd may then fail to start during spacewalk-service start:

```
Starting spacewalk services...
   Initializing jabberd processes...
       Starting router                                                          done
       Starting sm startproc:  exit status of parent of /usr/bin/sm: 2
failed
   Terminating jabberd processes...
```

/var/log/messages shows more details:

```
jabberd/sm[31445]: starting up
jabberd/sm[31445]: process id is 31445, written to /var/lib/jabberd/pid/sm.pid
jabberd/sm[31445]: loading 'db' storage module
jabberd/sm[31445]: db: corruption detected! close all jabberd processes and run db_recover
jabberd/router[31437]: shutting down
```

CURE: Remove the jabberd database and restart. Jabberd will automatically re-create the database:

```
spacewalk-service stop
 rm -Rf /var/lib/jabberd/db/*
 spacewalk-service start
```

An alternative approach would be to test another database, but SUSE Manager does not deliver drivers for this:

```
rcosa-dispatcher stop
 rcjabberd stop
 cd /var/lib/jabberd/db
 rm *
 cp /usr/share/doc/packages/jabberd/db-setup.sqlite .
 sqlite3 sqlite.db < db-setup.sqlite
 chown jabber:jabber *
 rcjabberd start
 rcosa-dispatcher start
```

Capturing XMPP Network Data for Debugging Purposes

If you are experiencing bugs regarding OSAD, it can be useful to dump network messages in order to help with debugging. The following procedures provide information on capturing data from both the client and server side.

Procedure: Server Side Capture

1. Install the tcpdump package on the SUSE Manager Server as root: `zypper in tcpdump`
2. Stop the OSA dispatcher and Jabber processes with `rcosa-dispatcher stop` and `rcjabberd stop.`
3. Start data capture on port 5222: `tcpdump -s 0 port 5222 -w server_dump.pcap`
4. Start the OSA dispatcher and Jabber processes: `rcosa-dispatcher start` and `rcjabberd start`.
5. Open a second terminal and execute the following commands: `rcosa-dispatcher start` and `rcjabberd start`.
6. Operate the SUSE Manager server and clients so the bug you formerly experienced is reproduced.
7. Once you have finished your capture re-open terminal 1 and stop the capture of data with: `CTRL+c`

Procedure: Client Side Capture

1. Install the tcpdump package on your client as root: `zypper in tcpdump`
2. Stop the OSA process: `rcosad stop`.
3. Begin data capture on port 5222: `tcpdump -s 0 port 5222 -w client_client_dump.pcap`
4. Open a second terminal and start the OSA process: `rcosad start`
5. Operate the SUSE Manager server and clients so the bug you formerly experienced is reproduced.
6. Once you have finished your capture re-open terminal 1 and stop the capture of data with: `CTRL+c`

Engineering Notes: Analyzing Captured Data

This section provides information on analyzing the previously captured data from client and server.

1. Obtain the certificate file from your SUSE Manager server: /etc/pki/spacewalk/jabberd/server.pem
2. Edit the certificate file removing all lines before `----BEGIN RSA PRIVATE KEY-----`, save it as key.pem
3. Install Wireshark as root with: `zypper in wireshark`
4. Open the captured file in wireshark.
5. From **Eidt ›]menu:Preferences[** select SSL from the left pane.
6. Select RSA keys list: **Edit ›]menu:New[**
    - IP Address any
    - Port: 5222
    - Protocol: xmpp
    - Key File: open the key.pem file previously edited.

- Password: leave blank

For more information see also:

- https://wiki.wireshark.org/SSL
- https://bugs.wireshark.org/bugzilla/show_bug.cgi?id=3444

Troubleshooting Package Inconsistencies

Packages can sometimes be locked or taskomatic can experience problems, which creates problems with metadata regeneration. 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 any processes are running, or if a crash could have occured. Check package locks and exclude lists to determine if packages are locked or excluded on the client. When you have located the problematic process, the metadata can be regenerated and synchronization occurs as expected.

Procedure: Resolving Package Inconsisties

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

```
/etc/init.d/taskomatic restart
```

3. Check package locks and exclude lists to determine if packages are locked or excluded on the client:
   - On Expanded Support Platform, check `/etc/yum.conf` and search for `exclude=`.
   - On SLES, use the `zypper locks` command.

Troubleshooting Registering Cloned Minions

Sometimes a cloned client (either traditional or Salt) will use the same machine ID as the system they are a clone of. This results in Uyuni only recognizing one system, rather than two different systems. This can be resolved by changing the machine ID of the cloned system, so that Uyuni recognizes them as two different clients.

Each step in this section is performed on the cloned system. This procedure does not manipulate the original system, which will still be registered to Uyuni. The cloned virtual machine should have a different UUID from the original (the UUID is generated by your hypervisor) or SUSE Manager will overwrite the original system data with the new one.

Procedure: Resolving Duplicate Machine IDs in Cloned Salt Clients

1. For SLES 12: If your machines have the same machine IDs then delete the file on each minion and recreate it:

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

ones

2. For SLES 11: As there is no systemd machine ID, generate one from dbus:

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

3. If your machines still have the same minion ID then delete the minion_id file on each minion (FQDN will be used when it is regenerated on minion restart):

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

4. Delete accepted keys from the Onboarding page and the system profile from Uyuni, and restart the minion with:

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

5. Re-register the clients. Each minion will now have a different `/etc/machine-id` and should now 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, and make sure `/etc/hosts` contains the changes you made and the correct host entries.
2. Stop rhnsd daemon with:

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

or:

```
# rcrhnsd 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
```

   - SLES 10:

```
# rm -rf /etc/{zmd,zypp}
# rm -rf /var/lib/zypp/!(db)
# rm -rf /var/lib/zmd/
```

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 SUSE Manager without overwriting 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) for this parameter:

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

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

```
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) for this parameter:

```
## 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) for this parameter:

```
timeout =`number`
```

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