



U Y U N I

Uyuni 2024.01

Client Configuration Guide

January 30 2024



# Table of Contents

Client Configuration Guide Overview	1
1. Supported Clients and Features	2
1.1. Supported Client Systems . . . . .	2
1.2. Supported Tools Packages . . . . .	3
1.3. Supported SUSE and openSUSE Client Features . . . . .	3
1.4. Supported SLE Micro Client Features . . . . .	6
1.5. openSUSE Leap Micro Client Features . . . . .	9
1.6. Supported Alibaba Cloud Linux Features . . . . .	11
1.7. Supported AlmaLinux Features . . . . .	13
1.8. Supported Amazon Linux Features . . . . .	15
1.9. Supported CentOS Features . . . . .	18
1.10. Supported Debian Features . . . . .	20
1.11. Supported openEuler Features . . . . .	22
1.12. Supported Oracle Features . . . . .	25
1.13. Supported Raspberry Pi OS Features . . . . .	27
1.14. Supported Red Hat Enterprise Linux Features . . . . .	30
1.15. Supported Rocky Linux Features . . . . .	32
1.16. Supported Ubuntu Features . . . . .	34
2. Configuration Basics	38
2.1. Software Channels . . . . .	38
2.1.1. Packages Provided by SUSE Package Hub . . . . .	38
2.1.2. Packages Provided by AppStream . . . . .	39
2.1.3. Packages Provided by EPEL . . . . .	39
2.1.4. Unified Installer Updates Channels on SUSE Linux Enterprise Clients . . . . .	39
2.1.5. Software Repositories . . . . .	40
2.1.6. Software Products . . . . .	41
2.2. Bootstrap Repository . . . . .	41
2.2.1. Prepare to Create a Bootstrap Repository . . . . .	42
2.2.2. Options for Automatic Mode . . . . .	42
2.2.3. Manually Generate a Bootstrap Repository . . . . .	43
2.2.4. Bootstrap and Custom Channels . . . . .	44
2.3. Activation Keys . . . . .	44
2.3.1. Reactivation Keys . . . . .	46
2.3.2. Activation Key Best Practices . . . . .	47
2.4. GPG Keys . . . . .	48
2.4.1. Trust GPG Keys on Clients . . . . .	48
3. Client Contact Methods	51
3.1. Default contact method for Salt clients . . . . .	51
3.1.1. Onboarding Details . . . . .	52
3.2. SSH Push contact method . . . . .	52
3.2.1. Available parameters . . . . .	54
3.2.2. Action Execution . . . . .	55
3.2.3. Future features . . . . .	55
3.3. SSH Push (with tunnel) contact method . . . . .	55
3.4. Salt Bundle . . . . .	58
3.4.1. What is Salt Bundle? . . . . .	58

3.4.2. Client Registration with Salt Bundle as a Minion . . . . .	58
3.4.3. SSH Push with the Salt Bundle . . . . .	59
3.4.4. Extend Salt Bundle with Python packages using pip . . . . .	60
<b>4. Client Registration . . . . .</b>	<b>61</b>
<b>4.1. Client Registration Methods . . . . .</b>	<b>61</b>
4.1.1. Register Clients with the Web UI . . . . .	61
4.1.2. Register Clients with a Bootstrap Script . . . . .	63
4.1.3. Register on the Command Line . . . . .	66
<b>4.2. SUSE Client Registration . . . . .</b>	<b>68</b>
4.2.1. Registering SUSE Linux Enterprise Clients . . . . .	69
4.2.2. Registering SLE Micro Clients . . . . .	72
<b>4.3. openSUSE Client Registration . . . . .</b>	<b>77</b>
4.3.1. Registering openSUSE Leap Clients . . . . .	77
4.3.2. Registering openSUSE Leap Micro Clients . . . . .	79
<b>4.4. Alibaba Cloud Linux Client Registration . . . . .</b>	<b>81</b>
4.4.1. Registering Alibaba Cloud Linux Clients . . . . .	81
<b>4.5. AlmaLinux Client Registration . . . . .</b>	<b>83</b>
4.5.1. Registering AlmaLinux Clients . . . . .	83
<b>4.6. Amazon Linux Client Registration . . . . .</b>	<b>86</b>
4.6.1. Registering Amazon Linux Clients . . . . .	86
<b>4.7. CentOS Client Registration . . . . .</b>	<b>88</b>
4.7.1. Registering CentOS Clients . . . . .	89
<b>4.8. Debian Client Registration . . . . .</b>	<b>93</b>
4.8.1. Registering Debian Clients . . . . .	93
<b>4.9. OpenEuler Client Registration . . . . .</b>	<b>96</b>
4.9.1. Registering openEuler Clients . . . . .	96
<b>4.10. Oracle Client Registration . . . . .</b>	<b>99</b>
4.10.1. Registering Oracle Linux Clients . . . . .	100
<b>4.11. Raspberry Pi OS Client Registration . . . . .</b>	<b>102</b>
4.11.1. Registering Raspberry Pi OS Clients . . . . .	102
<b>4.12. Red Hat Client Registration . . . . .</b>	<b>106</b>
4.12.1. Registering Red Hat Enterprise Linux Clients with CDN . . . . .	107
4.12.2. Registering Red Hat Enterprise Linux clients with RHUI . . . . .	115
<b>4.13. Rocky Linux Client Registration . . . . .</b>	<b>120</b>
4.13.1. Registering Rocky Linux Clients . . . . .	120
<b>4.14. Ubuntu Client Registration . . . . .</b>	<b>123</b>
4.14.1. Registering Ubuntu 20.04 and 22.04 Clients . . . . .	123
<b>4.15. Register Clients to a Proxy . . . . .</b>	<b>126</b>
4.15.1. Move Clients between Proxies . . . . .	126
4.15.2. Move Clients from Proxies to the Server . . . . .	127
4.15.3. Register Clients to a Proxy with the Web UI . . . . .	127
4.15.4. Registering with a Bootstrap Script (Salt and Traditional) . . . . .	129
<b>4.16. Registering clients on a public cloud . . . . .</b>	<b>130</b>
4.16.1. Add Products and Synchronize Repositories . . . . .	130
4.16.2. Prepare on-demand images . . . . .	130
4.16.3. Register clients . . . . .	131
4.16.4. Activation keys . . . . .	132
4.16.5. Automatic registration of clients created by Terraform . . . . .	132
<b>5. Client Upgrades . . . . .</b>	<b>134</b>

5.1. Client - Major version upgrade . . . . .	134
5.1.1. Prepare to migrate . . . . .	134
5.1.2. Create an autoinstallation profile . . . . .	136
5.1.3. Migration . . . . .	137
5.2. Upgrade Using the Content Lifecycle Manager . . . . .	137
5.2.1. Prepare to Upgrade . . . . .	137
5.2.2. Upgrade . . . . .	139
5.3. Product Migration . . . . .	139
5.3.1. Single System Migration . . . . .	140
5.3.2. Product Mass Migration . . . . .	140
5.4. Upgrade Uyuni Clients . . . . .	143
5.4.1. Prepare to Upgrade . . . . .	143
5.4.2. Upgrade . . . . .	144
6. Client Deletion . . . . .	146
6.1. Delete a Client with the Web UI . . . . .	146
6.2. Delete a Salt Client on the Command Line (with API Call) . . . . .	146
6.3. Delete a Client from the Command Line . . . . .	146
6.3.1. Salt Client . . . . .	146
7. Client Operations . . . . .	148
7.1. Package Management . . . . .	148
7.1.1. Verify Packages . . . . .	148
7.1.2. Compare Packages . . . . .	148
7.2. Patch Management . . . . .	149
7.2.1. Create Patches . . . . .	149
7.2.2. Apply Patches to Clients . . . . .	151
7.3. System Locking . . . . .	152
7.3.1. System Locks on Salt Clients . . . . .	152
7.3.2. Package Locks . . . . .	152
7.4. Configuration Management . . . . .	153
7.4.1. Create Configuration Channels . . . . .	154
7.4.2. Add Configuration Files, Directories, or Symbolic Links . . . . .	155
7.4.3. Subscribe Clients to Configuration Channels . . . . .	156
7.4.4. Compare Configuration Files . . . . .	156
7.4.5. Jinja templating on Salt clients . . . . .	157
7.5. Power Management . . . . .	157
7.5.1. Power Management and Cobbler . . . . .	158
7.6. Custom System Information . . . . .	158
7.7. System Set Manager . . . . .	159
7.7.1. Change Base Channels in SSM . . . . .	160
7.8. System Groups . . . . .	161
7.8.1. Create Groups . . . . .	161
7.8.2. Add Clients to Groups . . . . .	161
7.8.3. Work with Groups . . . . .	162
7.9. System Types . . . . .	163
8. Operating System Installation . . . . .	164
8.1. Reinstall Registered Systems . . . . .	165
8.2. Install via the Network (PXE Boot) . . . . .	165
8.2.1. Prepare the DHCP Server . . . . .	167
8.2.2. Synchronize the TFTP Tree with Proxies . . . . .	168

8.2.3. GRUB EFI binary names for different architectures . . . . .	169
8.3. Install via a CD-ROM or a USB Key . . . . .	169
8.3.1. Build an ISO Image With Cobbler . . . . .	169
8.3.2. Build a SUSE ISO Image With KIWI . . . . .	170
8.3.3. Build a Red Hat ISO Image With Cobbler . . . . .	170
8.4. Autoinstallable Distributions . . . . .	170
8.4.1. Distribution Based on an ISO Image . . . . .	171
8.4.2. Distribution Based on a RPM Package . . . . .	171
8.4.3. Declare an Autoinstallable Distribution . . . . .	172
8.5. Autoinstallation Profiles . . . . .	172
8.5.1. Declare the Profile . . . . .	173
8.5.2. AutoYaST Profiles . . . . .	174
8.5.3. Kickstart profiles . . . . .	175
8.5.4. Templates syntax . . . . .	176
8.6. Unattended Provisioning . . . . .	178
8.6.1. Bare Metal Provisioning . . . . .	178
8.6.2. Create a System Record Manually . . . . .	179
8.7. Use Your Own GPG Key . . . . .	179
8.7.1. Own GPG Key for PXE Boot . . . . .	180
8.7.2. Own GPG key in a CD-ROM . . . . .	180
9. Virtualization . . . . .	182
9.1. Manage Virtualized Hosts . . . . .	182
9.2. Create Virtual Guests . . . . .	182
9.3. Virtualization with Xen and KVM . . . . .	183
9.3.1. Host setup . . . . .	184
9.3.2. Autoinstall VM Guests . . . . .	185
9.3.3. Manage VM Guests . . . . .	188
10. Virtual Host Managers . . . . .	190
10.1. VHM and Amazon Web Services . . . . .	190
10.1.1. Create an Amazon EC2 VHM . . . . .	190
10.1.2. AWS Permissions for Virtual Host Manager . . . . .	191
10.2. VHM and Azure . . . . .	192
10.2.1. Prerequisites . . . . .	192
10.2.2. Create an Azure VHM . . . . .	192
10.2.3. Assigning permissions . . . . .	193
10.2.4. Azure UUID . . . . .	193
10.3. VHM and Google Compute Engine . . . . .	193
10.3.1. Prerequisites . . . . .	194
10.3.2. Create a GCE VHM . . . . .	194
10.3.3. Assigning Permissions . . . . .	194
10.3.4. GCE UUID . . . . .	195
10.4. VHM and Kubernetes . . . . .	195
10.4.1. Create a Kubernetes VHM . . . . .	196
10.4.2. Retrieve Image Runtime Data . . . . .	196
10.4.3. Permissions and Certificates . . . . .	198
10.5. Virtualization with Nutanix . . . . .	199
10.5.1. VHM Setup . . . . .	199
10.6. Virtualization with VMWare . . . . .	200
10.6.1. VHM Setup . . . . .	200

10.6.2. Troubleshooting SSL Errors on VMWare.....	201
10.7. Virtualization with Other Third Party Providers.....	202
11. GNU Free Documentation License	204

# Client Configuration Guide Overview

**Updated:** 2024-01-30

Registering clients is the first step after installing Uyuni, and most of the time you spend with Uyuni is spent on maintaining those clients.

Uyuni is compatible with SUSE Linux Enterprise or another Linux operating system and with a range of hardware options.

For a complete list of supported clients and features, see [Client-configuration > Supported-features](#).

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

# Chapter 1. Supported Clients and Features

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

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

## 1.1. Supported Client Systems

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

The icons in this table indicate:

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



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



The operating system running on a client is supported by the organization that supplies the operating system.

*Table 1. Supported Client Systems*

Operating System	Architecture	Salt Clients
SUSE Linux Enterprise 15, 12	x86-64, ppc64le, IBM Z, aarch64	✓
SUSE Linux Enterprise Server for SAP 15, 12	x86-64, ppc64le	✓
SLE Micro	x86-64, aarch64, s390x	✓
openSUSE Leap 15	x86-64, aarch64	✓
openSUSE Leap Micro	x86-64, aarch64	✓
Alibaba Cloud Linux 2	x86-64, aarch64	✓
AlmaLinux 9, 8	x86-64, aarch64	✓
Amazon Linux 2	x86-64, aarch64	✓
Amazon Linux 2023	x86-64, aarch64	✓
CentOS 7	x86-64, ppc64le, aarch64	✓

Operating System	Architecture	Salt Clients
Debian 12, 11, 10	x86-64	✓
openEuler 22.03	x86-64, aarch64	✓
Oracle Linux 9, 8, 7	x86-64, aarch64	✓
Raspberry Pi OS 12	arm64, armhf	✓
Red Hat Enterprise Linux 9, 8, 7	x86-64	✓
Rocky Linux 9, 8	x86-64, aarch64	✓
Ubuntu 22.04, 20.04	amd64	✓



Debian and Ubuntu list the x86-64 architecture as amd64.

When the distribution reaches end-of-life, it enters grace period of 3 months when the support is considered deprecated. After that period, the product is considered unsupported. Any support may only be available on the best-effort basis.

For more information about end-of-life dates, see <https://endoflife.software/operating-systems>.

## 1.2. Supported Tools Packages

The `spacewalk-utils` and `spacewalk-utils-extras` packages can provide additional services and features.

*Table 2. Spacewalk Utilities*

Tool Name	Description	Supported?
<code>spacewalk-common-channels</code>	Add channels not provided by SUSE Customer Center	✓
<code>spacewalk-hostname-rename</code>	Change the hostname of the Uyuni Server	✓
<code>spacewalk-clone-by-date</code>	Clone channels by a specific date	✓
<code>spacewalk-sync-setup</code>	Set up ISS master and slave organization mappings	✓
<code>spacewalk-manage-channel-lifecycle</code>	Manage channel lifecycles	✓

## 1.3. Supported SUSE and openSUSE Client Features

This table lists the availability of various features on SUSE and openSUSE clients. This table covers all

variants of the SUSE Linux Enterprise operating system, including SLES, SLED, SUSE Linux Enterprise Server for SAP, and SUSE Linux Enterprise Server for HPC.



- The operating system you run on a client is supported by the organization that supplies the operating system. SUSE Linux Enterprise is supported by SUSE. openSUSE is supported by the SUSE community.

The icons in this table indicate:

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

*Table 3. Supported Features on SUSE and openSUSE Operating Systems*

Feature	SUSE Linux Enterprise 12	SUSE Linux Enterprise 15	openSUSE 15
Client	✓	✓	✓
System packages	SUSE	SUSE	openSUSE Community
Registration	✓	✓	Salt
Install packages	✓	✓	Salt
Apply patches	✓	✓	Salt
Remote commands	✓	✓	Salt
System package states	Salt	Salt	Salt
System custom states	Salt	Salt	Salt
Group custom states	Salt	Salt	Salt
Organization custom states	Salt	Salt	Salt
System set manager (SSM)	✓	✓	Salt
Product migration	✓	✓	Salt
Basic Virtual Guest Management *	✓	✓	Salt

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

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

#### \* Virtual Guest Management:

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

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

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

## 1.4. Supported SLE Micro Client Features



- The operating system you run on a client is supported by the organization that supplies the operating system. SLE Micro is supported by SUSE.

The icons in this table indicate:

- ✓ the feature is available on both Salt and traditional clients
- ✗ the feature is not available

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

**Table 4. Supported Features on SLE Micro Operating Systems**

Feature	SLE Micro
Client	Salt
Operating system packages	Salt
Registration	Salt
Install packages	Salt
Apply patches (requires CVE ID)	Salt
Remote commands	Salt
System package states	Salt
System custom states	Salt
Group custom states	Salt
Organization custom states	Salt
System set manager (SSM)	Salt
Product migration	Salt
Basic Virtual Guest Management *	Salt
Advanced Virtual Guest Management *	Salt
Virtual Guest Installation (Kickstart), as Host OS	Salt
Virtual Guest Installation (image template), as Host OS	Salt
System deployment (PXE/Kickstart)	Salt
System redeployment (Kickstart)	Salt
Contact methods	Salt: ZeroMQ
Works with Uyuni Proxy	Salt
Action chains	Salt
Staging (pre-download of packages)	?
Duplicate package reporting	Salt

Feature	SLE Micro
CVE auditing (requires CVE ID)	Salt
SCAP auditing	?
Package verification	?
Package locking	Salt
System locking	?
Maintenance Windows	?
System snapshot	✗
Configuration file management	Salt
Snapshots and profiles	Salt: Profiles supported, Sync not supported
Power management	Salt
Monitoring server	✗
Monitored clients <b>**</b>	Salt
Docker buildhost	✗
Build Docker image with OS	✗
Kiwi buildhost	✗
Build Kiwi image with OS	Salt
Recurring Actions	Salt
AppStreams	N/A
Yomi	?

**\*** Virtual Guest Management:

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

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

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

**\*\*** On SLE Micro, only the Node exporter and the Blackbox exporter are available.

## 1.5. openSUSE Leap Micro Client Features



openSUSE Leap Micro is supported by the SUSE community.

The icons in this table indicate:

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

*Table 5. Supported Features on openSUSE Leap Micro Operating Systems*

Feature	openSUSE Leap Micro
Client	Salt
Operating system packages	Salt
Registration	Salt
Install packages	Salt
Apply patches (requires CVE ID)	Salt
Remote commands	Salt
System package states	Salt
System custom states	Salt
Group custom states	Salt
Organization custom states	Salt
System set manager (SSM)	Salt
Product migration	Salt
Basic Virtual Guest Management *	Salt
Advanced Virtual Guest Management *	Salt
Virtual Guest Installation (Kickstart), as Host OS	Salt
Virtual Guest Installation (image template), as Host OS	Salt
System deployment (PXE/Kickstart)	Salt
System redeployment (Kickstart)	Salt

Feature	openSUSE Leap Micro
Contact methods	Salt: ZeroMQ
Works with Uyuni Proxy	Salt
Action chains	Salt
Staging (pre-download of packages)	?
Duplicate package reporting	Salt
CVE auditing (requires CVE ID)	Salt
SCAP auditing	?
Package verification	?
Package locking	Salt
System locking	?
Maintenance Windows	?
System snapshot	✗
Configuration file management	Salt
Snapshots and profiles	Salt: Profiles supported, Sync not supported
Power management	Salt
Monitoring server	✗
Monitored clients	Salt
Docker buildhost	✗
Build Docker image with OS	✗
Kiwi buildhost	✗
Build Kiwi image with OS	Salt
Recurring Actions	Salt
AppStreams	N/A
Yomi	?

### \* Virtual Guest Management:

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

Basic virtual guest management includes listing VMs, slow refresh, VM lifecycle actions (start, stop,

resume, pause), and modifying VM vCPU and Memory.

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

## 1.6. Supported Alibaba Cloud Linux Features

This table lists the availability of various features on Alibaba Cloud Linux clients.



The operating system you run on a client is supported by the organization that supplies the operating system. Alibaba Cloud Linux is supported by Alibaba Cloud.

The icons in this table indicate:

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

*Table 6. Supported Features on Alibaba Cloud Linux Operating Systems*

Feature	Alibaba Cloud Linux 2
Client	Salt
Operating system packages	Salt
Registration	Salt
Install packages	Salt
Apply patches (requires CVE ID)	Salt
Remote commands	Salt
System package states	Salt
System custom states	Salt
Group custom states	Salt
Organization custom states	Salt
System set manager (SSM)	Salt
Product migration	N/A

Feature	Alibaba Cloud Linux 2
Basic Virtual Guest Management *	?
Advanced Virtual Guest Management *	?
Virtual Guest Installation (Kickstart), as Host OS	✗
Virtual Guest Installation (image template), as Host OS	?
System deployment (PXE/Kickstart)	?
System redeployment (Kickstart)	?
Contact methods	Salt: ZeroMQ, Salt-SSH
Works with Uyuni Proxy	Salt
Action chains	Salt
Staging (pre-download of packages)	Salt
Duplicate package reporting	Salt
CVE auditing (requires CVE ID)	Salt
SCAP auditing	Salt
Package verification	✗
Package locking	✗
System locking	✗
Maintenance Windows	✓
System snapshot	✗
Configuration file management	Salt
Snapshots and profiles	Salt: Profiles supported, Sync not supported
Power management	?
Monitoring server	✗
Monitored clients	Salt
Docker buildhost	Salt
Build Docker image with OS	Salt
Kiwi buildhost	Salt

Feature	Alibaba Cloud Linux 2
Build Kiwi image with OS	Salt
Recurring Actions	Salt
AppStreams	N/A
Yomi	N/A

#### \* Virtual Guest Management:

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

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

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

\* The traditional stack is available on Alibaba Cloud Linux but it is unsupported.

## 1.7. Supported AlmaLinux Features

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



- The operating system you run on a client is supported by the organization that supplies the operating system. AlmaLinux is supported by the AlmaLinux community.

The icons in this table indicate:

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

*Table 7. Supported Features on AlmaLinux Operating Systems*

Feature	AlmaLinux 9	AlmaLinux 8
Client	Salt (plain AlmaLinux)	Salt (plain AlmaLinux)
System packages	AlmaLinux Community	AlmaLinux Community

Feature	AlmaLinux 9	AlmaLinux 8
Registration	Salt	Salt
Install packages	Salt	Salt
Apply patches	Salt	Salt
Remote commands	Salt	Salt
System package states	Salt	Salt
System custom states	Salt	Salt
Group custom states	Salt	Salt
Organization custom states	Salt	Salt
System set manager (SSM)	Salt	Salt
Product migration	N/A	N/A
Basic Virtual Guest Management *	Salt	Salt
Advanced Virtual Guest Management *	Salt	Salt
Virtual Guest Installation (Kickstart), as Host OS	✗	✗
Virtual Guest Installation (image template), as Host OS	Salt	Salt
System deployment (PXE/Kickstart)	Salt	Salt
System redeployment (Kickstart)	Salt	Salt
Contact methods	Salt: ZeroMQ, Salt-SSH	Salt: ZeroMQ, Salt-SSH
Works with Uyuni Proxy	Salt	Salt
Action chains	Salt	Salt
Staging (pre-download of packages)	Salt	Salt
Duplicate package reporting	Salt	Salt
CVE auditing	Salt	Salt
SCAP auditing	Salt	Salt
Package verification	✗	✗

Feature	AlmaLinux 9	AlmaLinux 8
Package locking	✗	✗
System locking	✗	✗
Maintenance Windows	✓	✓
System snapshot	✗	✗
Configuration file management	Salt	Salt
Snapshots and profiles	Salt: Profiles supported, Sync not supported	Salt: Profiles supported, Sync not supported
Power management	Salt	Salt
Monitoring server	✗	✗
Monitored clients	Salt	Salt
Docker buildhost	✗	✗
Build Docker image with OS	✗	✗
Kiwi buildhost	✗	✗
Build Kiwi image with OS	✗	✗
Recurring Actions	Salt	Salt
AppStreams	✓	✓
Yomi	N/A	N/A

### \* Virtual Guest Management:

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

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

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

## 1.8. Supported Amazon Linux Features

This table lists the availability of various features on Amazon Linux clients.



- The operating system you run on a client is supported by the organization that supplies the operating system. Amazon Linux is supported by Amazon.

The icons in this table indicate:

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

*Table 8. Supported Features on Amazon Linux Operating Systems*

Feature	Amazon Linux 2	Amazon Linux 2023
Client	Salt	Salt
Operating system packages	Salt	Salt
Registration	Salt	Salt
Install packages	Salt	Salt
Apply patches (requires CVE ID)	Salt	Salt
Remote commands	Salt	Salt
System package states	Salt	Salt
System custom states	Salt	Salt
Group custom states	Salt	Salt
Organization custom states	Salt	Salt
System set manager (SSM)	Salt	Salt
Product migration	N/A	N/A
Basic Virtual Guest Management *	?	?
Advanced Virtual Guest Management *	?	?
Virtual Guest Installation (Kickstart), as Host OS	✗	✗
Virtual Guest Installation (image template), as Host OS	?	?
System deployment (PXE/Kickstart)	?	?
System redeployment (Kickstart)	?	?

Feature	Amazon Linux 2	Amazon Linux 2023
Contact methods	Salt: ZeroMQ, Salt-SSH	Salt: ZeroMQ, Salt-SSH
Works with Uyuni Proxy	Salt	Salt
Action chains	Salt	Salt
Staging (pre-download of packages)	Salt	Salt
Duplicate package reporting	Salt	Salt
CVE auditing (requires CVE ID)	Salt	Salt
SCAP auditing	Salt	Salt
Package verification	✗	✗
Package locking	✗	✗
System locking	✗	✗
Maintenance Windows	✓	✓
System snapshot	✗	✗
Configuration file management	Salt	Salt
Snapshots and profiles	Salt: Profiles supported, Sync not supported	Salt: Profiles supported, Sync not supported
Power management	?	?
Monitoring server	✗	✗
Monitored clients	Salt	Salt
Docker buildhost	Salt	Salt
Build Docker image with OS	Salt	Salt
Kiwi buildhost	Salt	Salt
Build Kiwi image with OS	Salt	Salt
Recurring Actions	Salt	Salt
AppStreams	N/A	N/A
Yomi	N/A	N/A

### \* Virtual Guest Management:

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

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

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

\* The traditional stack is available on Amazon Linux but it is unsupported.

## 1.9. Supported CentOS Features

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



- The operating system you run on a client is supported by the organization that supplies the operating system. CentOS is supported by the CentOS community.

The icons in this table indicate:

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

*Table 9. Supported Features on CentOS Operating Systems*

Feature	CentOS 7
Client	✓ (plain CentOS)
System packages	CentOS Community
Registration	✓
Install packages	✓
Apply patches (requires CVE ID)	✓ (third-party service required for errata)
Remote commands	✓
System package states	Salt
System custom states	Salt
Group custom states	Salt
Organization custom states	Salt
System set manager (SSM)	✓
Product migration	N/A

Feature	CentOS 7
Basic Virtual Guest Management *	✓
Advanced Virtual Guest Management *	Salt
Virtual Guest Installation (Kickstart), as Host OS	✗
Virtual Guest Installation (image template), as Host OS	✓
System deployment (PXE/Kickstart)	✓
System redeployment (Kickstart)	✓
Contact methods	Salt: ZeroMQ, Salt-SSH
Works with Uyuni Proxy	✓
Action chains	✓
Staging (pre-download of packages)	✓
Duplicate package reporting	✓
CVE auditing (requires CVE ID)	✓
SCAP auditing	✓
Package verification	✗
Package locking	✓
System locking	✗
Maintenance Windows	✓
System snapshot	✗
Configuration file management	✓
Snapshots and profiles	Salt: Profiles supported, Sync not supported
Power management	✓
Monitoring server	✗
Monitored clients	Salt
Docker buildhost	✗
Build Docker image with OS	✗
Kiwi buildhost	✗

Feature	CentOS 7
Build Kiwi image with OS	✗
Recurring Actions	Salt
AppStreams	N/A
Yomi	N/A

### \* Virtual Guest Management:

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

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

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

## 1.10. Supported Debian Features

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



- The operating system you run on a client is supported by the organization that supplies the operating system. Debian is supported by the Debian community.

The icons in this table indicate:

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

*Table 10. Supported Features on Debian Operating Systems*

Feature	Debian 10	Debian 11	Debian 12
Client	✓	✓	✓
System packages	Debian Community	Debian Community	Debian Community
Registration	Salt	Salt	Salt
Install packages	Salt	Salt	Salt

Feature	Debian 10	Debian 11	Debian 12
Apply patches	?	?	?
Remote commands	Salt	Salt	Salt
System package states	Salt	Salt	Salt
System custom states	Salt	Salt	Salt
Group custom states	Salt	Salt	Salt
Organization custom states	Salt	Salt	Salt
System set manager (SSM)	Salt	Salt	Salt
Product migration	N/A	N/A	N/A
Basic Virtual Guest Management *	Salt	Salt	Salt
Advanced Virtual Guest Management *	Salt	Salt	Salt
Virtual Guest Installation (Kickstart), as Host OS	✗	✗	✗
Virtual Guest Installation (image template), as Host OS	Salt	Salt	Salt
System deployment (PXE/Kickstart)	✗	✗	✗
System redeployment (Kickstart)	✗	✗	✗
Contact methods	Salt: ZeroMQ, Salt-SSH	Salt: ZeroMQ, Salt-SSH	Salt: ZeroMQ, Salt-SSH
Works with Uyuni Proxy	Salt	Salt	Salt
Action chains	Salt	Salt	Salt
Staging (pre-download of packages)	Salt	Salt	Salt
Duplicate package reporting	Salt	Salt	Salt
CVE auditing	?	?	?
SCAP auditing	?	?	?

Feature	Debian 10	Debian 11	Debian 12
Package verification	✗	✗	✗
Package locking	✓	✓	✓
System locking	✗	✗	✗
Maintenance Windows	✓	✓	✓
System snapshot	✗	✗	✗
Configuration file management	Salt	Salt	Salt
Package profiles	Salt: Profiles supported, Sync not supported	Salt: Profiles supported, Sync not supported	Salt: Profiles supported, Sync not supported
Power management	✓	✓	✓
Monitoring server	✗	✗	✗
Monitoring clients	Salt	Salt	Salt
Docker buildhost	?	?	?
Build Docker image with OS	Salt	Salt	Salt
Kiwi buildhost	✗	✗	✗
Build Kiwi image with OS	✗	✗	✗
Recurring Actions	Salt	Salt	Salt
AppStreams	N/A	N/A	N/A
Yomi	N/A	N/A	N/A

### \* Virtual Guest Management:

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

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

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

## 1.11. Supported openEuler Features

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



The operating system you run on a client is supported by the organization that supplies the operating system. openEuler is supported by the openEuler community.

The icons in this table indicate:

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

*Table 11. Supported Features on openEuler Operating Systems*

Feature	openEuler 22.03
Client	Salt (plain openEuler)
System packages	openEuler Community
Registration	Salt
Install packages	Salt
Apply patches	Salt
Remote commands	Salt
System package states	Salt
System custom states	Salt
Group custom states	Salt
Organization custom states	Salt
System set manager (SSM)	Salt
Product migration	N/A
Basic Virtual Guest Management *	Salt
Advanced Virtual Guest Management *	Salt
Virtual Guest Installation (Kickstart), as Host OS	✗
Virtual Guest Installation (image template), as Host OS	Salt

Feature	openEuler 22.03
System deployment (PXE/Kickstart)	Salt
System redeployment (Kickstart)	Salt
Contact methods	Salt: ZeroMQ, Salt-SSH
Works with Uyuni Proxy	Salt
Action chains	Salt
Staging (pre-download of packages)	Salt
Duplicate package reporting	Salt
CVE auditing	Salt
SCAP auditing	Salt
Package verification	✗
Package locking	✗
System locking	✗
Maintenance Windows	✓
System snapshot	✗
Configuration file management	Salt
Snapshots and profiles	Salt: Profiles supported, Sync not supported
Power management	Salt
Monitoring	Salt
Docker buildhost	✗
Build Docker image with OS	✗
Kiwi buildhost	✗
Build Kiwi image with OS	✗
Recurring Actions	Salt
Yomi	N/A

### \* Virtual Guest Management:

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

Basic virtual guest management includes listing VMs, slow refresh, VM lifecycle actions (start, stop,

resume, pause), and modifying VM vCPU and Memory.

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

## 1.12. Supported Oracle Features

This table lists the availability of various features on Oracle Linux clients.



- ! The operating system you run on a client is supported by the organization that supplies the operating system. Oracle Linux is supported by Oracle.

The icons in this table indicate:

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

*Table 12. Supported Features on Oracle Linux Operating Systems*

Feature	Oracle Linux 7	Oracle Linux 8	Oracle Linux 9
Client	✓	Salt	Salt
Operating system packages	✓	Salt	Salt
Registration	✓	Salt	Salt
Install packages	✓	Salt	Salt
Apply patches (requires CVE ID)	✓	Salt	Salt
Remote commands	✓	Salt	Salt
System package states	Salt	Salt	Salt
System custom states	Salt	Salt	Salt
Group custom states	Salt	Salt	Salt
Organization custom states	Salt	Salt	Salt

Feature	Oracle Linux 7	Oracle Linux 8	Oracle Linux 9
System set manager (SSM)	✓	Salt	Salt
Product migration	N/A	N/A	N/A
Basic Virtual Guest Management *	✓	Salt	Salt
Advanced Virtual Guest Management *	Salt	Salt	Salt
Virtual Guest Installation (Kickstart), as Host OS	Traditional	✗	✗
Virtual Guest Installation (image template), as Host OS	✓	Salt	Salt
System deployment (PXE/Kickstart)	✓	Salt	Salt
System redeployment (Kickstart)	✓	Salt	Salt
Contact methods	Traditional: OSAD, RHNSD, SSH-push. Salt: ZeroMQ, Salt-SSH	Salt: ZeroMQ, Salt-SSH	Salt: ZeroMQ, Salt-SSH
Works with Uyuni Proxy	✓	Salt	Salt
Action chains	✓	Salt	Salt
Staging (pre-download of packages)	✓	Salt	Salt
Duplicate package reporting	✓	Salt	Salt
CVE auditing (requires CVE ID)	✓	Salt	Salt
SCAP auditing	✓	Salt	Salt
Package verification	Traditional	✗	✗
Package locking	✓	?	?
System locking	Traditional	✗	✗
Maintenance Windows	✓	✓	✓

Feature	Oracle Linux 7	Oracle Linux 8	Oracle Linux 9
System snapshot	Traditional	✗	✗
Configuration file management	✓	Salt	Salt
Snapshots and profiles	Traditional. Salt: Profiles supported, Sync not supported	Salt: Profiles supported, Sync not supported	Salt: Profiles supported, Sync not supported
Power management	✓	Salt	Salt
Monitoring server	✗	✗	✗
Monitored clients	Salt	Salt	Salt
Docker buildhost	✗	✗	✗
Build Docker image with OS	✗	✗	✗
Kiwi buildhost	✗	✗	✗
Build Kiwi image with OS	✗	✗	✗
Recurring Actions	Salt	Salt	Salt
AppStreams	N/A	✓	✓
Yomi	N/A	N/A	N/A

#### \* Virtual Guest Management:

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

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

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

## 1.13. Supported Raspberry Pi OS Features

This table lists the availability of various features on Raspberry Pi OS clients.



- The operating system you run on a client is supported by the organization that supplies the operating system. Raspberry Pi OS is supported by the Raspberry Pi OS community.

The icons in this table indicate:

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

*Table 13. Supported Features on Raspberry Pi OS Operating Systems*

Feature	Raspberry Pi OS 12
Client	✓
System packages	Raspberry Pi OS Community
Registration	Salt
Install packages	Salt
Apply patches	?
Remote commands	Salt
System package states	Salt
System custom states	Salt
Group custom states	Salt
Organization custom states	Salt
System set manager (SSM)	Salt
Product migration	N/A
Basic Virtual Guest Management *	Salt
Advanced Virtual Guest Management *	Salt
Virtual Guest Installation (Kickstart), as Host OS	✗
Virtual Guest Installation (image template), as Host OS	Salt
System deployment (PXE/Kickstart)	✗
System redeployment (Kickstart)	✗
Contact methods	Salt: ZeroMQ, Salt-SSH
Works with Uyuni Proxy	Salt

Feature	Raspberry Pi OS 12
Action chains	Salt
Staging (pre-download of packages)	Salt
Duplicate package reporting	Salt
CVE auditing	?
SCAP auditing	?
Package verification	✗
Package locking	✓
System locking	✗
Maintenance Windows	✓
System snapshot	✗
Configuration file management	Salt
Package profiles	Salt: Profiles supported, Sync not supported
Power management	✓
Monitoring server	✗
Monitoring clients	Salt
Docker buildhost	?
Build Docker image with OS	Salt
Kiwi buildhost	✗
Build Kiwi image with OS	✗
Recurring Actions	Salt
AppStreams	N/A
Yomi	N/A

### \* Virtual Guest Management:

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

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

Advanced virtual guest management includes all features of Basic virtual guest management plus fast

refresh, VM lifecycle actions (delete, reset, power off), modifying VM disk, network, graphical display, and graphical display configuration.

## 1.14. Supported Red Hat Enterprise Linux Features

This table lists the availability of various features on native Red Hat Enterprise Linux clients.



- The operating system you run on a client is supported by the organization that supplies the operating system. Red Hat Enterprise Linux is supported by Red Hat.

The icons in this table indicate:

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

*Table 14. Supported Features on Red Hat Enterprise Linux Operating Systems*

Feature	RHEL 7	RHEL 8	RHEL 9
Client	✓	Salt	Salt
System packages	Red Hat	Red Hat	Red Hat
Registration	✓	Salt	Salt
Install packages	✓	Salt	Salt
Apply patches	✓	Salt	Salt
Remote commands	✓	Salt	Salt
System package states	Salt	Salt	Salt
System custom states	Salt	Salt	Salt
Group custom states	Salt	Salt	Salt
Organization custom states	Salt	Salt	Salt
System set manager (SSM)	Salt	Salt	Salt
Product migration	N/A	N/A	N/A
Basic Virtual Guest Management *	✓	Salt	Salt

Feature	RHEL 7	RHEL 8	RHEL 9
Advanced Virtual Guest Management *	Salt	Salt	Salt
Virtual Guest Installation (Kickstart), as Host OS	✗	✗	✗
Virtual Guest Installation (image template), as Host OS	✓	Salt	Salt
System deployment (PXE/Kickstart)	✓	Salt	Salt
System redeployment (Kickstart)	✓	Salt	Salt
Contact methods	Salt: ZeroMQ, Salt-SSH	Salt: ZeroMQ, Salt-SSH	Salt: ZeroMQ, Salt-SSH
Works with Uyuni Proxy	✓	Salt	Salt
Action chains	✓	Salt	Salt
Staging (pre-download of packages)	✓	Salt	Salt
Duplicate package reporting	✓	Salt	Salt
CVE auditing	✓	Salt	Salt
SCAP auditing	✓	Salt	Salt
Package verification	✗	✗	✗
Package locking	✓	?	?
System locking	✗	✗	✗
Maintenance Windows	✓	✓	✓
System snapshot	✗	✗	✗
Configuration file management	✓	Salt	Salt
Snapshots and profiles	Salt: Profiles supported, Sync not supported	Salt: Profiles supported, Sync not supported	Salt: Profiles supported, Sync not supported
Power management	✓	Salt	Salt
Monitoring server	✗	✗	✗

Feature	RHEL 7	RHEL 8	RHEL 9
Monitored clients	Salt	Salt	Salt
Docker buildhost	✗	✗	✗
Build Docker image with OS	?	?	?
Kiwi buildhost	✗	✗	✗
Build Kiwi image with OS	✗	✗	✗
Recurring Actions	Salt	Salt	Salt
AppStreams	N/A	✓	✓
Yomi	N/A	N/A	N/A

### \* Virtual Guest Management:

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

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

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

## 1.15. Supported Rocky Linux Features

This table lists the availability of various features on Rocky Linux clients.



- The operating system you run on a client is supported by the organization that supplies the operating system. Rocky Linux is supported by the Rocky Linux community.

The icons in this table indicate:

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

*Table 15. Supported Features on Rocky Linux Operating Systems*

Feature	Rocky Linux 8	Rocky Linux 9
Client	Salt (plain Rocky Linux)	Salt (plain Rocky Linux)
System packages	Rocky Linux Community	Rocky Linux Community
Registration	Salt	Salt
Install packages	Salt	Salt
Apply patches	Salt	Salt
Remote commands	Salt	Salt
System package states	Salt	Salt
System custom states	Salt	Salt
Group custom states	Salt	Salt
Organization custom states	Salt	Salt
System set manager (SSM)	Salt	Salt
Product migration	N/A	N/A
Basic Virtual Guest Management *	Salt	Salt
Advanced Virtual Guest Management *	Salt	Salt
Virtual Guest Installation (Kickstart), as Host OS	✗	✗
Virtual Guest Installation (image template), as Host OS	Salt	Salt
System deployment (PXE/Kickstart)	Salt	Salt
System redeployment (Kickstart)	Salt	Salt
Contact methods	Salt: ZeroMQ, Salt-SSH	Salt: ZeroMQ, Salt-SSH
Works with Uyuni Proxy	Salt	Salt
Action chains	Salt	Salt
Staging (pre-download of packages)	Salt	Salt
Duplicate package reporting	Salt	Salt
CVE auditing	Salt	Salt

Feature	Rocky Linux 8	Rocky Linux 9
SCAP auditing	Salt	Salt
Package verification	✗	✗
Package locking	?	?
System locking	✗	✗
Maintenance Windows	✓	✓
System snapshot	✗	✗
Configuration file management	Salt	Salt
Snapshots and profiles	Salt: Profiles supported, Sync not supported	Salt: Profiles supported, Sync not supported
Power management	Salt	Salt
Monitoring server	✗	✗
Monitored clients	Salt	Salt
Docker buildhost	✗	✗
Build Docker image with OS	✗	✗
Kiwi buildhost	✗	✗
Build Kiwi image with OS	✗	✗
Recurring Actions	Salt	Salt
AppStreams	✓	✓
Yomi	N/A	N/A

#### \* Virtual Guest Management:

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

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

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

## 1.16. Supported Ubuntu Features

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



- The operating system you run on a client is supported by the organization that supplies the operating system. Ubuntu is supported by Canonical.

The icons in this table indicate:

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

*Table 16. Supported Features on Ubuntu Operating Systems*

Feature	Ubuntu 20.04	Ubuntu 22.04
Client	✓	✓
System packages	Canonical	Canonical
Registration	Salt	Salt
Install packages	Salt	Salt
Apply patches	✓	✓
Remote commands	Salt	Salt
System package states	Salt	Salt
System custom states	Salt	Salt
Group custom states	Salt	Salt
Organization custom states	Salt	Salt
System set manager (SSM)	Salt	Salt
Product migration	N/A	N/A
Basic Virtual Guest Management *	Salt	Salt
Advanced Virtual Guest Management *	Salt	Salt
Virtual Guest Installation (Kickstart), as Host OS	✗	✗
Virtual Guest Installation (image template), as Host OS	Salt	Salt

Feature	Ubuntu 20.04	Ubuntu 22.04
System deployment (PXE/Kickstart)	✗	✗
System redeployment (Kickstart)	✗	✗
Contact methods	Salt: ZeroMQ, Salt-SSH	Salt: ZeroMQ, Salt-SSH
Works with Uyuni Proxy	Salt	Salt
Action chains	Salt	Salt
Staging (pre-download of packages)	Salt	Salt
Duplicate package reporting	Salt	Salt
CVE auditing	?	?
SCAP auditing	?	?
Package verification	✗	✗
Package locking	✓	✓
System locking	✗	✗
System snapshot	✗	✗
Configuration file management	Salt	Salt
Package profiles	Salt: Profiles supported, Sync not supported	Salt: Profiles supported, Sync not supported
Power management	✓	✓
Monitoring	Salt	Salt
Docker buildhost	?	?
Build Docker image with OS	Salt	Salt
Kiwi buildhost	✗	✗
Build Kiwi image with OS	✗	✗
Recurring Actions	Salt	Salt
AppStreams	N/A	N/A
Yomi	N/A	N/A

\* Virtual Guest Management:

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

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

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

# Chapter 2. Configuration Basics

Uyuni requires a number of steps to prepare the environment for clients registration before a wide range of its operations can be utilized.

This section contains summary of the initial configuration steps that are necessary to support environment operations following successful Uyuni installation and setting up.

- For more information about installing Uyuni, see [Installation-and-upgrade > Install-uyuni](#).
- For more information about setting up Uyuni, see [Installation-and-upgrade > Uyuni-server-setup](#).

## 2.1. Software Channels

Channels are a method of grouping software packages. Software packages are provided by repositories, and repositories are associated with channels. Subscribing a client to a software channel allows the client to install and update any of the software associated with it.

In Uyuni, channels are divided into base channels and child channels. Organizing channels in this way ensures that only compatible packages are installed on each system. A client must be subscribed to only one base channel, assigned during registration based on the client operating system and architecture. For paid channels provided by a vendor, you must have an associated subscription.

A base channel consists of packages built for a specific operating system type, version, and architecture. For example, the SUSE Linux Enterprise Server 15 x86-64 base channel contains only software compatible with that operating system and architecture.

A child channel is associated with a base channel and provides only packages that are compatible with the base channel. A system can be subscribed to multiple child channels of its base channel. When a system has been assigned to a base channel, it is only possible for that system to install the related child channels. For example, if a system has been assigned to the SUSE Linux Enterprise Server 15 **x86\_64** base channel, they can only install or update packages made available through the compatible base channel, or any of its associated child channels.

In the Uyuni Web UI you can browse your available channels by navigating to [Software > Channel List > All](#). You can modify or create new channels by navigating to [Software > Manage > Channels](#).

For more on using channels, including custom channels, see [Administration > Channel-management](#).

### 2.1.1. Packages Provided by SUSE Package Hub

SUSE Package Hub is an extension to SUSE Linux Enterprise products that provides additional open source software provided by the openSUSE community.



- The packages in SUSE Package Hub are provided by the openSUSE community.
- They are not supported by SUSE.

If you are using SUSE Linux Enterprise operating systems on your clients, you can enable the SUSE Package Hub extension to access these additional packages. This provides the SUSE Package Hub channels, which you can subscribe your clients to.

SUSE Package Hub provides a large number of packages, which can take a long time to synchronize and consume a large amount of disk space. Do not enable SUSE Package Hub unless you require the packages it provides.

To avoid unintentionally installing or updating unsupported packages, we recommend that you implement a content lifecycle management strategy that initially denies all SUSE Package Hub packages. You can then explicitly enable the specific packages you require. For more information about content lifecycle management, see [Administration > Content-lifecycle](#).

### 2.1.2. Packages Provided by AppStream

For Red Hat based clients, additional packages are available through AppStream. In most cases, the AppStream packages are required to ensure that you have all the software you need.

When you are managing AppStream packages in the Uyuni Web UI, you might notice that you see contradicting suggestions for package updates. This is due to the Uyuni not being able to interpret the modular metadata correctly. You can use the content lifecycle management (CLM) AppStream filter to transform AppStream repositories into non-modular repositories for use with some upgrade operations. For more information about the CLM AppStream filters, see [Administration > Content-lifecycle-examples](#).

### 2.1.3. Packages Provided by EPEL

For Red Hat based clients, additional packages are available through EPEL (extra packages for enterprise Linux). EPEL is an optional package repository that provides additional software.



- The packages in EPEL are provided by the Fedora community. They are not supported by SUSE.

If you are using Red Hat operating systems on your clients, you can enable the EPEL extension to access these additional packages. This provides the EPEL channels, which you can subscribe your clients to.

EPEL provides a large number of packages, which can take a long time to synchronize and consume a large amount of disk space. Do not enable the EPEL repositories unless you require the packages it provides.

To avoid unintentionally installing or updating unsupported packages, we recommended that you implement a content lifecycle management (CLM) strategy that initially denies all EPEL packages. You can then explicitly enable the specific packages you require. For more information about content lifecycle management, see [Administration > Content-lifecycle](#).

### 2.1.4. Unified Installer Updates Channels on SUSE Linux Enterprise Clients

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

For SUSE Linux Enterprise Server clients the installer updates channel is synchronized by default when you add a product that contains them, and are enabled when you create an autoinstallable distribution with these product channels.

For all other SUSE Linux Enterprise variants, including SUSE Linux Enterprise for SAP, you must add the installer updates channel manually. To do this, clone the appropriate SUSE Linux Enterprise Server installer updates channel below the base channel of these SUSE Linux Enterprise variants. When creating an autoinstallable distribution for these SUSE Linux Enterprise variants after the channel was cloned, it is used automatically.

## 2.1.5. Software Repositories

Repositories are used to collect software packages. When you have access to a software repository, you can install any of the software that the repository provides. You must have at least one repository associated with your software channels in Uyuni to assign clients to the channel and install and update packages on the client.

Most default channels in Uyuni are already associated with the correct repositories. If you are creating custom channels, you need to associate a repository that you have access to, or that you have created yourself.

For more information about custom repositories and channels, see [Administration > Custom-channels](#).

### 2.1.5.1. Local Repository Locations

You can configure local repositories on clients, to provide packages that are not supplied by Uyuni channels.



In most cases, client systems do not require local repositories. Local repositories can lead to problems knowing which packages are available on the client. This can lead to installing unexpected packages.

Local repositories are disabled during onboarding.

Local repositories will be disabled each time a channel state is executed. For example, when applying the highstate or performing a package action.

If local repositories should stay enabled after onboarding, the following pillar must be set for the affected client:

Edit the `/srv/pillar/top.sls` file:

```
base:
  'minionid':
```

```
- localrepos
```

Edit the `/srv/pillar/localrepos.sls` file:

```
mgr_disable_local_repos: False
```

After a client has completed onboarding, you can add local repositories to these locations:

*Table 17. Local Repository Locations*

Client Operating System	Local Repository Directory
SUSE Linux Enterprise Server	<code>/etc/zypp/repos.d</code>
openSUSE	<code>/etc/zypp/repos.d</code>
Red Hat Enterprise Linux and similar derivatives	<code>/etc/yum.repos.d/</code>
Ubuntu	<code>/etc/apt/sources.list.d/</code>
Debian	<code>/etc/apt/sources.list.d/</code>

## 2.1.6. Software Products

In Uyuni, software is made available in products. Your SUSE subscription allows you to access a range of different products, which you can browse and select in the Uyuni Web UI by navigating to **Admin > Setup Wizard > Products**.

Products contain any number of software channels. Click the **Show product's channels** icon to see the channels included in the product. When you have added a product and synchronized successfully, you have access to the channels provided by the product, and can use the packages in the product on your Uyuni Server and clients.

*Procedure: Adding Software Channels*

1. In the Uyuni Web UI, navigate to **Admin > Setup Wizard > Products**.
2. Locate the appropriate products for your client operating system and architecture using the search bar, and check the appropriate product. This will automatically check all mandatory channels. Also all recommended channels are checked as long as the **include recommended** toggle is turned on. Click the arrow to see the complete list of related products, and ensure that any extra products you require are checked.
3. Click **Add Products** and wait until the products have finished synchronizing.

For more information, see **Installation-and-upgrade > Setup-wizard**.

## 2.2. Bootstrap Repository

A bootstrap repository contains required packages for registering clients during bootstrapping. When products are synchronized, bootstrap repositories are automatically created and regenerated on the Uyuni Server.

### 2.2.1. Prepare to Create a Bootstrap Repository

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

*Procedure: Checking Synchronization Progress from the Web UI*

1. In the Uyuni Web UI, navigate to **Software > Manage > Channels**, then click the channel associated to the repository.
2. Navigate to the **Repositories** tab, then click **Sync** and check **Sync Status**.

*Procedure: Checking Synchronization Progress from the Command Prompt*

1. At the command prompt on the Uyuni Server, as root, use the **tail** command to check the synchronization log file:

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

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

### 2.2.2. Options for Automatic Mode

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

*Flush Mode::*

#### Flush Mode

By default, existing repositories are updated only with the latest packages. You can configure it to always start with an empty repository instead. To enable this behavior, add or edit this value in **/etc/rhn/rhn.conf**:

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

*Automatic Mode::*

#### Automatic Mode

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

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

### 2.2.2.1. Configure Bootstrap Data File

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

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

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

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



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

### 2.2.3. Manually Generate a Bootstrap Repository

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

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

- At the command prompt on the Uyuni Server, as root, list the available distributions to create bootstrap repositories for:

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

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

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

- Alternatively, use the number shown next to the distribution name in the list of available distributions.

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

If you have mirrored more than one product (for example, SLES and SLES for SAP), or if you use custom channels, you might need to specify the parent channel to use when creating the bootstrap repository. This is not required in every situation. For example, some SLES 15 versions have common code bases, so there is no need to specify a parent channel. Use this procedure only if your environment requires it.

*OPTIONAL Procedure: Specifying a Parent Channel for a Bootstrap Repository*

1. Check which parent channels you have available:

```
mgr-create-bootstrap-repo -c SLE-15-x86_64
Multiple options for parent channel found. Please use option
--with-parent-channel <label> and choose one of:
- sle-product-sles15-pool-x86_64
- sle-product-sles_sap15-pool-x86_64
- sle-product-sled15-pool-x86_64
```

2. Specify the appropriate parent channel:

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

#### 2.2.3.1. Repositories with Multiple Architectures

If you are creating bootstrap repositories that include multiple different architectures, you need to be careful that all architectures are updated correctly. For example, the x86-64 and IBM Z architectures for SLE use the same bootstrap repository URL at </srv/www/htdocs/pub/repositories/sle/15/2/bootstrap/>.

When the **flush** option is enabled, and you attempt to generate the bootstrap repository for multiple architectures, only one architecture is generated. To avoid this, use the **--no-flush** option at the command prompt when creating additional architectures. For example:

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

#### 2.2.4. Bootstrap and Custom Channels

If you are using custom channels, you can use the **--with-custom-channels** option with the **mgr-create-bootstrap-repo** command. In this case, you also need to specify the parent channel to use.

Automatic creation of a bootstrap repository might fail if you are using custom channels. In this case, you need to create the repository manually.

For more information about custom channels, see **Administration > Custom-channels**.

## 2.3. Activation Keys

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

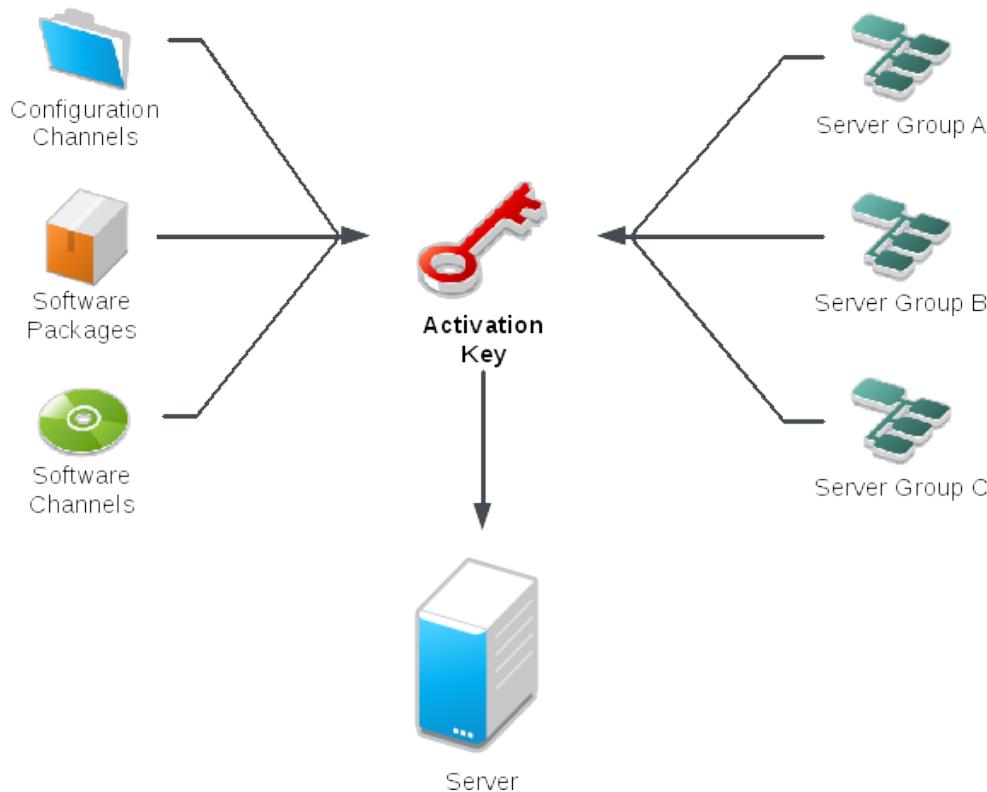
In Uyuni, an activation key is a group of configuration settings with a label. You can apply all configuration settings associated with an activation key by adding its label as a parameter to a bootstrap

script. We recommend you use an activation key label in combination with a bootstrap script. When the bootstrap script is executed all configuration settings associated with the label are applied to the system the script is run on.

An activation key can specify:

- Channel assignment
- System types or add-on entitlements
- Contact method
- Configuration files
- Packages to be installed
- System group assignment

Activation keys are used at the time a client is registered, and not used again. After the client has been registered, the client can be changed in any way, regardless of what the activation key specifies. The association between the activation key and the client is recorded only for historical purposes.



#### *Procedure: Creating an Activation Key*

1. In the Uyuni Web UI, as an administrator, navigate to **Systems > Activation Keys**.
2. Click the **Create Key** button.
3. On the **Activation Key Details** page, in the **Description** field, enter a description of the activation key.

4. In the **Key** field, enter a name for the activation key. For example, **SLES15-SP5** for SUSE Linux Enterprise Server 15 SP5.



- Do not use commas or double quotes in the **Key** field for any SUSE products. However, you **must** use commas for Red Hat Products.
- All other characters are allowed, but <> (){} (this includes the space) are removed automatically.
- If the field is left empty, a random string is generated.

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



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

### 2.3.1. Reactivation Keys

Reactivation keys can be used once only to re-register a client and regain all Uyuni settings. Reactivation keys are client-specific, and include the system ID, history, groups, and channels.

To create a reactivation key, navigate to **Systems**, click the client to create a reactivation key for, and navigate to the **Details > Reactivation** tab. Click **Generate New Key** to create the reactivation key. Record the details of the key for later use. Unlike typical activation keys, which are not associated with a specific system ID, keys created here do not show up on the **Systems > Activation Keys** page.

After you have created a reactivation key, you can use it as the **management\_key** grain in **/etc/salt/minion.d/susemanager.conf**. For example:

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

Restart the `venv-salt-minion` or `salt-minion` process to apply the reactivation key.

You can use a reactivation key with a bootstrap script. For more information about bootstrap scripts, see [Client-configuration > Registration-bootstrap](#).



- If you autoinstall a client with its existing Uyuni profile, the profile uses the reactivation key to re-register the system and restore its settings. Do not regenerate, delete, or use this key while a profile-based autoinstallation is in progress. Doing so causes the autoinstallation to fail.

## 2.3.2. Activation Key Best Practices

### *Default Parent Channel*

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

### *Bootstrapping with Activation Keys*

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

### *Bootstrapping LTSS clients*

If you are bootstrapping clients with LTSS subscription, include the LTSS channels during activation key creation.

### *Bandwidth Requirements*

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

These options create bandwidth usage:

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

### *Key Label Naming*

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

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

When creating key labels, consider these tips:

- OS naming (mandatory): Keys should always refer to the OS they provide settings for
- Architecture naming (recommended): Unless your company is running on one architecture only, for example x86\_64, then providing labels with an architecture type is a good idea.
- Server type naming: What is this server being used for?
- Location naming: Where is the server located? Room, building, or department?
- Date naming: Maintenance windows, quarter, etc.
- Custom naming: What naming scheme suits your organizations needs?

Example activation key label names:

`sles15-sp4-web_server-room_129-x86_64`

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

#### *Included Channels*

When creating activation keys you also need to keep in mind which software channels are associated with it. Keys should have a specific base channel assigned to them. Using the default base channel is not recommended. For more information, see the client operating system you are installing at **Client-configuration > Registration-overview**.

## 2.4. GPG Keys

Clients use GPG keys to check the authenticity of software packages before they are installed. Only trusted software can be installed on clients.

In most cases, you do not need to adjust the GPG settings to be able to install software on your clients.

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

### 2.4.1. Trust GPG Keys on Clients

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

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

The GPG key URL parameter in the software channel page can contain multiple key URLs separated by "whitespace". In case it is a file URL, the GPG key file must be deployed on the client before the software channel is used.

The GPG keys for the Client Tools Channels of Red Hat based clients are deployed on the client into `/etc/pki/rpm-gpg/` and can be referenced with file URLs.

Only in case a software channel is assigned to the client they will be imported and trusted by the system.



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

#### 2.4.1.1. User defined GPG keys

Users can define custom GPG keys to be deployed to a client.

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

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

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

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

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

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

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

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

### 2.4.1.2. GPG Keys in Bootstrap Scripts

*Procedure: Trusting GPG Keys on Clients Using a Bootstrap Script*

1. On the Uyuni Server, at the command prompt, check the contents of the `/srv/www/htdocs/pub/` directory. This directory contains all available public keys. Take a note of the key that applies to the channel assigned to the client you are registering.
2. Open the relevant bootstrap script, locate the `ORG_GPG_KEY=` parameter and add the required key. For example:

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

You do not need to delete any previously stored keys.



- Trusting a GPG key is important for security on clients. It is the task of the administrator to decide which keys are needed and can be trusted. A software channel cannot be assigned to a client when the GPG key is not trusted.

## Chapter 3. Client Contact Methods

There are three contact methods how the Uyuni Server can communicate with clients using the Salt protocol.



The so-called traditional contact protocol is no longer supported with Uyuni 5.0 and later. Before upgrading from Uyuni 4 to 5, traditional clients and thus traditional proxies have to be migrated to Salt or replaced with Salt proxies.

Which Salt contact method you use depends on the type of client, and your network architecture:

### Default (salt)

is the default choice and recommended unless there are specific needs. For more information, see [Client-configuration > Contact-methods-salt](#).

### SSH Push

is useful only if network restrictions make it impossible for clients to establish contact to the server. It is supported with the Salt Bundle only. This contact method has serious limitations. For more information, see [Client-configuration > Contact-methods-saltssh](#).

### SSH Push (with tunnel)

is the same as SSH Push but with using a secured communication tunnel. For more information, see [Client-configuration > Contact-methods-saltssh-tunnel](#).

Implementation of Salt communication software:

### Salt Bundle

is a single binary package containing Salt Minion, Python 3, required Python modules, and libraries. Thus the Salt contact method is independent from software installed on the client. Using the Salt Bundle is the default. It is the only supported implementation for the SSH Push or SSH Push (with tunnel) contact method. For more information, see [Client-configuration > Contact-methods-saltbundle](#).

### Salt Minion

is Salt software installed on the client system.

## 3.1. Default contact method for Salt clients

The default contact method using the Salt protocol is recommended unless there are specific needs. For more information about Salt in general, see [Specialized-guides > Salt](#).

Software updates are generally pushed from the server to the client. Connections are initiated from the client. This means you must open ports on the server, not on clients. The Salt clients are also known as Salt minions. Uyuni Server installs a daemon on every client.

If you need to use Salt clients in a disconnected setup you can configure SSH Push as a contact method.

With this contact method, clients can be located in a firewall-protected zone called a DMZ. For more information about SSH Push, see [Client-configuration > Contact-methods-saltssh](#).

### 3.1.1. Onboarding Details

Salt has its own database to keep the keys for the minions. This needs to be kept in sync with the Uyuni database. As soon as the key is accepted in Salt, the onboarding process in Uyuni starts. The onboarding process will look for existing systems in the Uyuni database by searching for the `minion_id` and the `machine-id`. If nothing is found, the new system gets created. In case an entry with the `minion_id` or the `machine-id` is found, that system will be migrated to match the new system. In case there is a match for both entries, and they are not the same system, the onboarding will be aborted with an error. In this case the administrator needs to resolve the conflict by removing at least one of the systems.

## 3.2. SSH Push contact method

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

SSH Push is also to use if installing a daemon agent on clients is not possible.



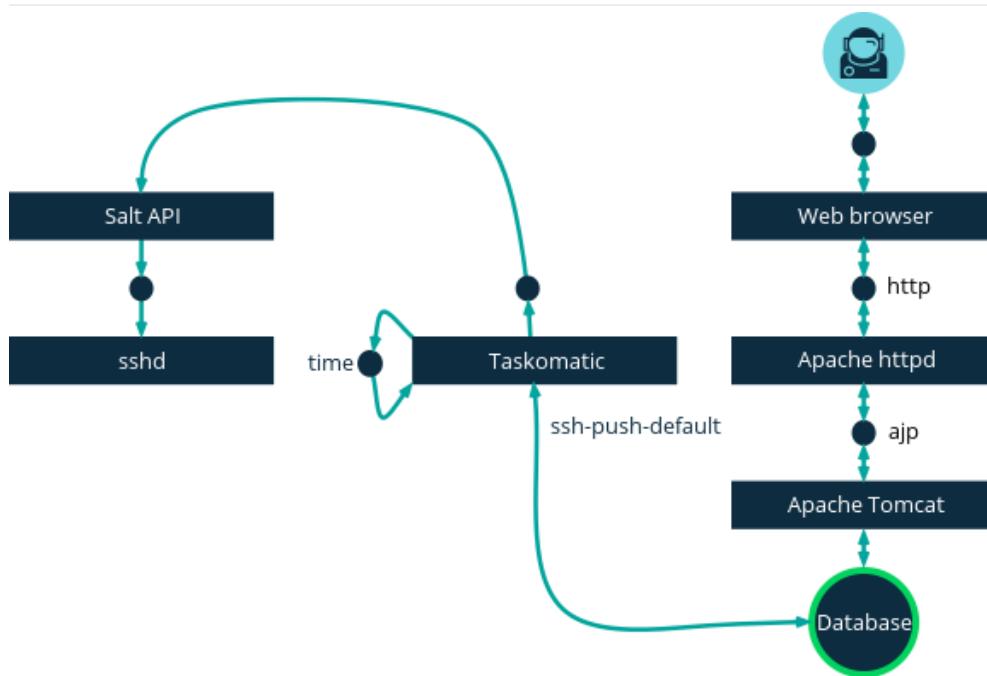
The SSH Push method has serious limitations. It does not scale well, and consumes more Server resources and network bandwidth than the plain Salt method (`default`). The Push SSH method is not at all supported with large setups (1000 clients and more).

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



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

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



To use SSH Push, you must have the SSH daemon running on the client, and reachable by the [salt-api](#) daemon running on the Uyuni Server. Additionally, the required Python version will be deployed with the Salt Bundle on the client system.

Before beginning with the following registering procedure, define an activation key with the SSH Push contact method configured. This method expects that there is a direct connection with HTTPS to the Server.

You need to use the Web UI or API to register these clients with the Uyuni Server. See the following procedure or example.

#### *Procedure: Registering clients with SSH Push*

1. In the Uyuni Web UI, navigate to **Systems > Bootstrapping** and complete the appropriate fields.
2. Select an activation key with the SSH Push contact method configured. For more information about activation keys, see **Client-configuration > Activation-keys**.
3. Check the **Manage system completely via SSH** checkbox.
4. Click **Bootstrap** to begin registration.
5. Confirm that the system has been registered correctly by navigating to **Systems > Overview**.

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

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

Valid values are:

- **default** (pull)

- **ssh-push**
- **ssh-push-tunnel**

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

### 3.2.1. Available parameters

When you are configuring SSH Push, you can modify parameters that are used when a system is registered, including the host, activation key, and password. The password is used only for bootstrapping, it is not saved anywhere. All future SSH sessions are authorized via a key/certificate pair. These parameters are configured in **Systems > Bootstrapping**.

You can also configure persistent parameters that are used system-wide, including the sudo user to access the system as an unprivileged user instead of as root.

*Procedure: Configuring unprivileged SSH access*

1. Ensure you have the latest **spacewalk-taskomatic** and **spacewalk-certs-tools** packages installed on the Uyuni Server.
2. On each client system, create an appropriate unprivileged user.
3. On each client system, edit the **sudoers** file:

```
sudo visudo
```

4. Grant **sudo** access to the user by adding this line at the end of the **sudoers** file. Replace **<user>** with the name of the user that is bootstrapping the client in the Web UI:

```
<user> ALL=NOPASSWD: /usr/bin/python3, /var/tmp/venv-salt-minion/bin/python
```



This procedure grants root access without requiring a password, which is required for registering the client. When the client is successfully installed it runs with root privileges, so the access is no longer required. We recommend that you remove the line from the **sudoers** file after the client has been successfully installed.

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

```
ssh_push_sudo_user = <user>
```

After changing this configuration parameter you must restart services such as **salt-secrets-**

`config.service`, `tomcat.service`, and `taskomatic.service`. To cover all needed services, it is the best to restart `spacewalk-service` as root:

```
spacewalk-service restart
```

### 3.2.2. Action Execution

The SSH Push feature uses taskomatic to execute scheduled actions using `salt-ssh`. The taskomatic job periodically checks for scheduled actions and executes them. The SSH Push feature executes a complete `salt-ssh` call based on the scheduled action.

By default, twenty Salt SSH actions can be executed at a time. You can increase the number of actions that can be executed in parallel, by adding these lines to your configuration file, and adjusting the value of `parallel_threads` upwards. We recommend you keep the number of parallel actions low, to avoid problems:

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

This adjusts the number of actions that can run in parallel on any one client and the total number of worker threads used by taskomatic. If actions needs to be run on multiple clients, actions are always executed sequentially on each client.

If the clients are connected through a proxy, you need to adjust the `MaxSessions` settings on the proxy. In this case, set the number of parallel connections to be three times the total number of clients.

### 3.2.3. Future features

There are some features that are not yet supported on SSH Push. These features do not work on Salt SSH clients:

- OpenSCAP auditing
- Beacons, resulting in:
  - Installing a package on a system using `zypper` does not invoke the package refresh.
  - Virtual Host functions (for example, a host to guests) does not work if the virtual host system is Salt SSH-based.

For more information about Salt SSH, see [Specialized-guides > Salt](#) and <https://docs.saltstack.com/en/latest/topics/ssh/>.

## 3.3. SSH Push (with tunnel) contact method

SSH Push (with tunnel) (`ssh-push-tunnel`) is used in environments where clients cannot reach the

Uyuni Server directly. In this environment, clients are located in a firewall-protected zone called a DMZ. No system within the DMZ is authorized to open a connection to the internal network, including the Uyuni Server.

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

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



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



The tunnel is used to provide the access to the server through the encrypted tunnel. The repositories assigned to the SSH Push client (with tunnel) are provided through this tunnel only, thus it is not possible to use package manager tools directly from the client system because the repositories are available only while the tunnel is up. In other words, access is only possible if the session is initiated by the server. All package managing operations on the client can be performed from the server side only.

For tunneling connections via SSH, a port number is required for tunneling via HTTPS. The port number used by default is **1233**. To overwrite it, you can add a custom port numbers greater than 1024 to **/etc/rhn/rhn.conf**:

```
ssh_push_port_https = high_port
```

After changing this configuration parameter you must restart services such as **salt-secrets-config.service**, **tomcat.service**, and **taskomatic.service**. To cover all needed services, it is the best to restart **spacewalk-service** as root:

```
spacewalk-service restart
```

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

*Procedure: Configuring Unprivileged SSH Access*

1. On each client system, create an appropriate unprivileged user.
2. On each client system, edit the **sudoers** file:

```
sudo visudo
```

3. Grant **sudo** access to the user by adding this line at the end of the **sudoers** file. Replace **<user>** with the name of the user that is bootstrapping the client in the Web UI:

```
<user> ALL=NOPASSWD: /usr/bin/python3, /var/tmp/venv-salt-minion/bin/python
```



This procedure grants root access without requiring a password, which is required for registering the client. When the client is successfully installed it runs with root privileges, so the access is no longer required. We recommend that you remove the line from the **sudoers** file after the client has been successfully installed.

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

```
ssh_push_sudo_user = <user>
```

After changing this configuration parameter you must restart services such as **salt-secrets-config.service**, **tomcat.service**, and **taskomatic.service**. To cover all needed services, it is the best to restart **spacewalk-service** as root:

```
spacewalk-service restart
```

You need to use the Web UI or API to register these clients with the Uyuni Server.

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

- For more information about bootstrapping, see **Client-configuration > Registration-bootstrap**.
- For more information about bootstrapping, see [client-configuration:activation-keys.pdf](#).

*Example: API access to SSH Push (with tunnel)*

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

Valid values are:

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

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

## 3.4. Salt Bundle

### 3.4.1. What is Salt Bundle?

The Salt Bundle is a single binary package containing Salt Minion, Python 3, required Python modules, and libraries.

The Salt Bundle is shipped with Python 3 and all the requirements for Salt to run. Thus Salt Bundle does not use the Python version install on the client as system software. The Salt Bundle can be installed on clients that do not meet the requirements for a given Salt version.

It is also possible to use the Salt Bundle on systems that run a Salt Minion connected to a Salt Master other than the Uyuni Salt Master.

### 3.4.2. Client Registration with Salt Bundle as a Minion

The registration method with the Salt Bundle is the recommended registration method. This section explains the advantages and limitations of the current implementation. The Salt Bundle is provided as the **venv-salt-minion** that consists of Salt, Python 3, and the Python modules Salt depends on. Bootstrapping with Web UI is using Salt Bundle as well, so bootstrapping with Web UI is not Python dependant. Using the Salt Bundle, it is no longer needed that the client provides any Python interpreter or modules.

If you bootstrap new clients, registration with the Salt Bundle is the default method. You can switch existing clients to the Salt Bundle method. If you switch, the **salt-minion** package and its dependencies will stay installed.

#### 3.4.2.1. Using the Salt Bundle with the Salt Minion

The Salt Bundle can be used with the Salt Minion managed by the Salt Master other than Uyuni Server at the same time. If the Salt Bundle is installed on a client Uyuni Server will manage the configuration files of the Salt Bundle, the configuration files of **salt-minion** will not be managed in this case. For more information, see [Salt Bundle configuration](#).



- To bootstrap a client with the Salt Minion managed by the Salt Master other than Uyuni Server it is recommended to use **mgr-bootstrap --force -bundle** when generating the bootstrap script, or to set **FORCE\_VENV\_SALT\_MINION** to 1 in the bootstrap script.
- For bootstrapping with Web UI **mgr\_force\_venv\_salt\_minion** set to **true** pillar can be specified globally. For more information, see [Specialized-guides > Salt](#).

#### 3.4.2.2. Switching from Salt Minion to Salt Bundle

The Salt state **util.mgr\_switch\_to\_venv\_minion** is available to switch from **salt-minion** to **venv-salt-minion**. It is recommended to switch to **venv-salt-minion** in two steps to avoid

trouble with shifting processes:

*Procedure: Switching with `util.mgr_switch_to_venv_minion` state to `venv-salt-minion`*

1. Apply `util.mgr_switch_to_venv_minion` with no pillar specified first. This will result in the switch to `venv-salt-minion` with copying configuration files etc. It will not clean up the original `salt-minion` configurations and its packages.

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

2. Apply `util.mgr_switch_to_venv_minion` with `mgr_purge_non_venv_salt` set to `True` to remove `salt-minion` and with `mgr_purge_non_venv_salt_files` set to `True` to remove all the files related to `salt-minion`. This second step ensures the first step was processed, and then removes the old configuration files and the now obsolete `salt-minion` package.

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



In case of running the second step of switching with skipping the first step, state apply process could fail as it requires stopping the `salt-minion` which is used to execute the command on the client side.

On the other hand, it is also possible to avoid installing the Salt Bundle and keep using `salt-minion` instead. In this case, specify one of these options:

- Execute `mgr-bootstrap` with `--no-bundle` option.
- Set `AVOID_VENV_SALT_MINION` to `1` in the generated bootstrap script.
- For bootstrap state set the `mgr_avoid_venv_salt_minion` pillar to `True`.

### 3.4.3. SSH Push with the Salt Bundle

The Salt Bundle is also used when performing SSH Push actions to clients.

A shell script deploys the Salt Bundle onto the target system without installing `venv-salt-minion` before any Salt command is executed. Because the Salt Bundle contains the whole Salt code base, no `salt-thin` is deployed. SSH Push (including bootstrapping using the Web UI) uses the Python 3 interpreter within the bundle. The target system does not need to have any other Python interpreter installed.

The Python 3 deployed with the Bundle is used to handle SSH Push session on the client, so SSH Push (including bootstrapping with Web UI) is not dependant on Python installed on the system.

Using `salt-thin` can be enabled as a fallback method, but it requires Python 3 to be installed on the client. This method is not recommended nor supported and exists for development purposes only. Set

`web.ssh_use_salt_thin` to `true` in the `/etc/rhn/rhn.conf` configuration file.



- The bootstrap repository must be created before bootstrapping the client with Web UI. SSH Push is using the Salt Bundle taken from the bootstrap repository based on the detected target operating system. For more information, see [client-configuration:bootstrap-repository.pdf](#).
- SSH Push is using `/var/tmp` to deploy Salt Bundle to and execute Salt commands on the client with the bundled Python. Therefore you must not mount `/var/tmp` with the `noexec` option. It is not possible to bootstrap the clients, which have `/var/tmp` mounted with `noexec` option, with the Web UI because the bootstrap process is using SSH Push to reach a client.

#### 3.4.4. Extend Salt Bundle with Python packages using pip

The Salt Bundle includes `pip` to make it possible to extend the functionality of the bundled Salt Minion with extra Python packages.

By default, `salt <minion_id> pip.install <package-name>` installs the Python package specified by `<package_name>` into `/var/lib/venv-salt-minion/local`.



If needed, the path `/var/lib/venv-salt-minion/local` can be overridden by setting the `VENV_PIP_TARGET` environment variable for the `venv-salt-minion.service`. It is recommended to use a systemd drop-in configuration file for the service. It could be done with the configuration file `/etc/systemd/system/venv-salt-minion.service.d/10-pip-destination.conf`:

```
[Service]
Environment=VENV_PIP_TARGET=/new/path/local/venv-salt-minion/pip
```



The Python packages installed through `pip` are not changing on updating the Salt Bundle. To ensure that such packages are available and functional after an update, it is recommended to install them with a Salt state that is applied after Salt Bundle updates.

# Chapter 4. Client Registration

There are several ways to register clients to your Uyuni Server. This section covers the various available methods. It also contains information specific to the operating system you intend to run on the client.

Before you begin, check that:

- The client has the date and time synchronized correctly with the Uyuni Server before registration.
- You have created an activation key. For more information about creating activation keys, see [Client-configuration > Activation-keys](#).



Do not register the Uyuni Server to itself. The Uyuni Server must be managed individually or by using another separate Uyuni Server. For more information about using multiple servers, see [Specialized-guides > Large-deployments](#).

## 4.1. Client Registration Methods

There are several ways to register clients to your Uyuni Server.

- For only a few clients, we recommend that you register clients using the Uyuni Web UI. For more information, see [Client-configuration > Registration-webui](#).
- If you want more control over the process, or have to register many clients, we recommend that you create a bootstrap script. For more information, see [Client-configuration > Registration-bootstrap](#).
- For even more control over the process, executing a single command on the command line can be useful. For more information, see [Client-configuration > Registration-cli](#).

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

You must create an activation key first, to use bootstrap script or command line method. For more information about creating activation keys, see [Client-configuration > Activation-keys](#).



Do not register the Uyuni Server to itself. The Uyuni Server must be managed individually or by using another separate Uyuni Server. For more information about using multiple servers, see [Specialized-guides > Large-deployments](#).

### 4.1.1. Register Clients with the Web UI

Bootstrapping clients with the Web UI is using [Specialized-guides > Salt](#) to execute the bootstrap process on the client. Salt SSH uses the Salt Bundle and its included Python interpreter. Therefore, no other Python interpreter needs to be installed on the client.



As Salt Bundle is shipped with the bootstrap repository, the repository must be created before starting the bootstrap process for the client. A shell script detects the operating system on the client and deploys the Salt Bundle from the appropriate bootstrap repository, using the same logic as the bootstrap script. For

more information, see [Prepare to Create a Bootstrap Repository](#).



Do not register the Uyuni Server to itself. The Uyuni Server must be managed individually or by using another separate Uyuni Server. For more information about using multiple servers, see [Specialized-guides > Large-deployments](#).

*Procedure: Registering Clients with the Web UI*

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

When the bootstrap process has completed, your client is listed at **Systems > System List**.



SSH private keys are stored only for the duration of the bootstrapping process. They are deleted from the Uyuni Server as soon as bootstrapping is complete.



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

#### 4.1.1.1. Handling of Locally assigned Repositories

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

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

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

### 4.1.2. Register Clients with a Bootstrap Script

Registering clients with a bootstrap script gives you control over parameters, and can help if you have to register a large number of clients at once.

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

It is important that you check the repository information carefully, to ensure it matches the base channel repository. If the repository information does not match exactly, the bootstrap script cannot download the correct packages.



All clients need a bootstrap repository. It is automatically created and regenerated on the Uyuni Server when products are synchronized. A bootstrap repository includes packages for installing Salt on clients, and for registering clients.

For more information about creating a bootstrap repository, see [Client-configuration > Bootstrap-repository](#).

#### *GPG Keys and Uyuni Client Tools*



The GPG key used by Uyuni Client Tools is not trusted by default. When you create your bootstrap script, add a path to the file containing the public key fingerprint with the `ORG_GPG_KEY` parameter.



openSUSE Leap 15 and SLE 15 use Python 3 by default. Bootstrap scripts based on Python 2 must be re-created for openSUSE Leap 15 and SLE 15 systems. If you register openSUSE Leap 15 or SLE 15 systems using Python 2, the bootstrap script fails.

#### 4.1.2.1. Create a Bootstrap Script with `mgr-bootstrap`

The `mgr-bootstrap` command generates custom bootstrap scripts. A bootstrap script is used by Uyuni

client systems for simplifying their initial registration and configuration.

The arguments **--activation-keys** and **--script**, are the only mandatory arguments. On the Uyuni Server, as root at the command line execute it with the mandatory arguments. Replace **<ACTIVATION\_KEY>** and **<EDITED\_NAME>** with your values:

```
mgr-bootstrap --activation-keys=<ACTIVATION_KEY> --script=bootstrap-<EDITED_NAME>.sh
```

The **mgr-bootstrap** command offers several other options, including the ability to set a specific hostname, set specific GPG keys, and set the registration method (salt-minion or salt-bundle).

For more information, see the **mgr-bootstrap** man page, or run **mgr-bootstrap --help**.

#### 4.1.2.2. Create a Bootstrap Script from Web UI

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

*Procedure: Creating a Bootstrap Script*

1. In the Uyuni Web UI, navigate to **Admin > Manager Configuration > Bootstrap Script**.
2. The required fields are pre-populated with values derived from previous installation steps. For details on each setting, see **Reference > Admin**.
3. Click **Update** to create the script.
4. The bootstrap script is generated and stored on the server in the **/srv/www/htdocs/pub/bootstrap** directory. Alternatively, you can access the bootstrap script over HTTPS. Replace **<example.com>** with the host name of your Uyuni Server:

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



- Do not disable SSL in your bootstrap script. Ensure that **Enable SSL** is checked in the Web UI, or that the setting **USING\_SSL=1** exists in the bootstrap script. If you disable SSL, the registration process requires custom SSL certificates.
- For more information about custom certificates, see **Administration > Ssl-certs**.

#### 4.1.2.3. Edit a Bootstrap Script

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

In this procedure, you need to know the exact name of your activation keys. Navigate to **Home >**

**Overview** and, in the **Tasks** box, click **Manage Activation Keys**. All keys created for channels are listed on this page. You must enter the full name of the key you wish to use in the bootstrap script exactly as presented in the key field. For more information about activation keys, see **Client-configuration > Activation-keys**.

*Procedure: Modifying a Bootstrap Script*

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

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

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

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

3. Open **bootstrap-sles15.sh** for modification. Scroll down until you can see the text shown below. If **exit 1** exists in the file, comment it out by typing a hash or pound sign (#) at the beginning of the line. This activates the script. Enter the name of the key for this script in the **ACTIVATION\_KEYS=** field:

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

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

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



By default, bootstrap script will try to install **venv-salt-minion** if it is available in the bootstrap repository and **salt-minion** if there is no Salt bundle in the bootstrap repository. It is possible to avoid installing Salt bundle and keep using **salt-minion** if you need it for some reason.

For more information, see **Client-configuration > Contact-methods-saltbundle**.

#### 4.1.2.4. Connect Clients

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

*Procedure: Running the Bootstrap Script*

1. On the Uyuni Server, log in as root. At the command prompt, and change to the bootstrap directory:

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

- Run this command to execute the bootstrap script on the client; replace **EXAMPLE.COM** with the host name of your client:

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

- Alternatively, on the client, run this command:

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



- To avoid problems, make sure the bootstrap script is executed using **bash**.

This script downloads the required dependencies located in the repositories directory you created earlier.

- When the script has finished running, you can check that your client is registered correctly by opening the Uyuni Web UI and navigating to **Systems > Overview** to ensure the new client is listed.
- If you used the script to register the Salt client, open the Uyuni Web UI and navigate to **Salt > Keys** to accept the client key.



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

### 4.1.3. Register on the Command Line

#### 4.1.3.1. Manual client registration

In most cases, Salt clients are registered accurately with the default bootstrap methods. However, you can use Salt to register the client to the Uyuni Server manually by editing the Salt minion file on the client, and providing the fully qualified domain name (FQDN) of the server. This method uses ports 4505 and 4506 inbound to the server. This method requires no configuration on the Uyuni Server aside from ensuring that these ports are open.

This procedure requires that you have installed the **venv-salt-minion** (Salt bundle) or the **salt-minion** package on the Salt client before registration. Both use configuration files in different locations and filenames remain the same. The systemd service filename is different.



- Bootstrapping this way will only work if you use the **salt-minion** being part of the client tools channels or of an official SUSE distributions.

#### 4.1.3.2. Salt Bundle configuration

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

- `/etc/venv-salt-minion/`
- `/etc/venv-salt-minion/minion`
- `/etc/venv-salt-minion/minion.d/NAME.conf`
- systemd service file: `venv-salt-minion.service`

For more information about the Salt bundle, see [Client-configuration > Contact-methods-saltbundle](#).

*Procedure: Registering Clients with Salt Bundle Configuration File*

1. On the Salt client, open the **minion** configuration file. The configuration file is either located at:

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

or:

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

2. In the file add or edit the FQDN of the Uyuni Server or Proxy, and the activation key if any. Also add the other configuration parameters listed below.

```
master: SERVER.EXAMPLE.COM
grains:
  susemanager:
    activation_key: "<Activation_Key_Name>"
server_id_use_crc: adler32
enable_legacy_startup_events: False
enable_fqdns_grains: False
```

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

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

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

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

#### 4.1.3.3. Client configuration

### Client (**salt-minion**)

- `/etc/salt/`

- `/etc/salt/minion`
- `/etc/salt/minion.d/NAME.conf`
- systemd service file: `salt-minion.service`

*Procedure: Registering Clients with Salt Minion Configuration File*

1. On the Salt client, open the `minion` configuration file. The configuration file is either located at:

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

or:

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

2. In the file add or edit the FQDN of the Uyuni Server or Proxy, and the activation key if any. Also add the other configuration parameters listed below.

```
master: SERVER.EXAMPLE.COM
grains:
  susemanager:
    activation_key: "<Activation_Key_Name>"
server_id_use_crc: adler32
enable_legacy_startup_events: False
enable_fqdns_grains: False
```

3. Restart the `salt-minion` service:

```
systemctl restart salt-minion
```

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

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

For more information about the Salt minion configuration file, see <https://docs.saltstack.com/en/latest/ref/configuration/minion.html>.

## 4.2. SUSE Client Registration

You can register SUSE Linux Enterprise clients to your Uyuni Server.

The method and details varies depending on the operating system of the client.

Before you start, ensure that the client has the date and time synchronized correctly with the Uyuni

Server.

You must also have created an activation key. For more information about creating activation keys, see [Client-configuration > Activation-keys](#).



- Do not register a Uyuni Server to itself. The Uyuni Server must be managed individually or by using another separate Uyuni Server. For more information about using multiple servers, see [Specialized-guides > Large-deployments](#).

## 4.2.1. Registering SUSE Linux Enterprise Clients

This section contains information about registering clients running SUSE Linux Enterprise operating systems.

Use the instructions in this chapter for preparing all SUSE Linux Enterprise products, including:

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

You can also use these instructions for older SUSE Linux Enterprise versions and service packs.

### 4.2.1.1. Add Software Channels



- In the following section, descriptions often default to the [x86\\_64](#) architecture. Replace it with other architectures if appropriate.

Before you register SUSE Linux Enterprise clients to your Uyuni Server, you need to add the required software channels, and synchronize them.

The products you need for this procedure are:

*Table 18. SLE Products - WebUI*

OS Version	Product Name
SUSE Linux Enterprise Server 15 SP5	SUSE Linux Enterprise Server 15 SP5 x86_64
SUSE Linux Enterprise Server 15 SP4	SUSE Linux Enterprise Server 15 SP4 x86_64
SUSE Linux Enterprise Server 15 SP3	SUSE Linux Enterprise Server 15 SP3 x86_64
SUSE Linux Enterprise Server 15 SP2	SUSE Linux Enterprise Server 15 SP2 x86_64
SUSE Linux Enterprise Server 15 SP1	SUSE Linux Enterprise Server 15 SP1 x86_64
SUSE Linux Enterprise Server 12 SP5	SUSE Linux Enterprise Server 12 SP5 x86_64

**Procedure: Adding Software Channels**

1. In the Uyuni Web UI, navigate to **Admin > Setup Wizard > Products**.
2. Locate the appropriate products for your client operating system and architecture using the search bar, and check the appropriate product. This will automatically check all mandatory channels. Also all recommended channels are checked as long as the **include recommended** toggle is turned on. Click the arrow to see the complete list of related products, and ensure that any extra products you require are checked.
3. Click **Add Products** and wait until the products have finished synchronizing.

Alternatively, you can add channels at the command prompt. The channels you need for this procedure are:

*Table 19. SLE Products - CLI*

OS Version	Base Channel
SUSE Linux Enterprise Server 15 SP5	sle-product-sles15-sp5-pool-x86_64
SUSE Linux Enterprise Server 15 SP4	sle-product-sles15-sp4-pool-x86_64
SUSE Linux Enterprise Server 15 SP3	sle-product-sles15-sp3-pool-x86_64
SUSE Linux Enterprise Server 15 SP2	sle-product-sles15-sp2-pool-x86_64
SUSE Linux Enterprise Server 15 SP1	sle-product-sles15-sp1-pool-x86_64
SUSE Linux Enterprise Server 12 SP5	sle-product-sles15-sp5-pool-x86_64

To find channel names of older products, at the command prompt on the Uyuni Server, as root, use the **mgr-sync** command:

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

Then specify the argument you are interested in. For example, **channels**:

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

**Procedure: Adding Software Channels at the Command Prompt**

1. At the command prompt on the Uyuni Server, as root, use the **mgr-sync** command to add the appropriate channels:

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

2. Synchronization starts automatically. If you want to synchronize the channels manually, use:

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

3. Ensure the synchronization is complete before continuing.

To add the client tools, add these channels from the command prompt:

*Table 20. SUSE Linux Enterprise Channels - CLI*

OS Version	Client Channel
SUSE Linux Enterprise Server 15 SP5	sles15-sp5-uyuni-client
SUSE Linux Enterprise Server 15 SP4	sles15-sp4-uyuni-client
SUSE Linux Enterprise Server 15 SP3	sles15-sp3-uyuni-client
SUSE Linux Enterprise Server 15 SP2	sles15-sp2-uyuni-client
SUSE Linux Enterprise Server 15 SP1	sles15-sp1-uyuni-client
SUSE Linux Enterprise Server 12 SP5	sles12-sp5-uyuni-client

*Procedure: Adding Software Channels at the Command Prompt*

1. At the command prompt on the Uyuni Server, as root, use the **spacewalk-common-channels** command to add the appropriate channels:

```
spacewalk-common-channels \
<base_channel_label> \
<child_channel_label_1> \
<child_channel_label_2> \
... <child_channel_label_n>
```

2. If **automatic synchronization** is turned off, synchronize the channels:

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

3. Ensure the synchronization is complete before continuing.

#### 4.2.1.2. Check Synchronization Status

*Procedure: Checking Synchronization Progress from the Web UI*

1. In the Uyuni Web UI, navigate to **Software > Manage > Channels**, then click the channel associated to the repository.
2. Navigate to the **Repositories** tab, then click **Sync** and check **Sync Status**.

*Procedure: Checking Synchronization Progress from the Command Prompt*

1. At the command prompt on the Uyuni Server, as root, use the **tail** command to check the

synchronization log file:

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

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



- SUSE Linux Enterprise channels can be very large. Synchronization can sometimes take several hours.

#### 4.2.1.3. Manage GPG Keys

Clients use GPG keys to check the authenticity of software packages before they are installed. Only trusted software can be installed on clients.



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

For more information about GPG keys, see [Client-configuration > Gpg-keys](#).



- Use the same GPG key for both SUSE Linux Enterprise Server 15 and SUSE Linux Enterprise Server 12 clients. The correct key is called **sle12-gpg-pubkey-39db7c82.key**.

#### 4.2.1.4. Register Clients

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

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

For more information on registering your clients, see [Client-configuration > Registration-overview](#).

#### 4.2.2. Registering SLE Micro Clients

This section contains information about registering clients running SLE Micro operating systems 5.1, 5.2, 5.3, 5.4, and 5.5 on x86-64, arm64, and IBM Z (s390x) architectures.

The SLE Micro is an ultra-reliable, lightweight operating system purpose built for edge computing. It leverages the enterprise hardened security and compliance components of SUSE Linux Enterprise and merges them with a modern, immutable, developer-friendly OS platform.

The SLE Micro uses transactional updates. Transactional updates are atomic (all updates are applied only

if all updates succeed) and support rollbacks. They do not affect a running system as no changes are activated until after the system is rebooted. This information is displayed in the **Systems > Details > Overview** subtab.

For more information on transactional updates and rebooting, see <https://documentation.suse.com/sles/html/SLES-all/cha-transactional-updates.html>.

#### 4.2.2.1. Add Software Channels

Before you register SLE Micro clients to your Uyuni Server, you need to add the required software channels, and synchronize them.



In the following section, descriptions often default to the **x86\_64** architecture. Replace it with other architectures if appropriate.

The products you need for this procedure are:

*Table 21. SLE Micro Products - WebUI*

OS Version	Product Name
SLE Micro 5.5 x86-64	SUSE Linux Enterprise Micro 5.5 x86_64
SLE Micro 5.5 arm64	SUSE Linux Enterprise Micro 5.5 aarch64
SLE Micro 5.5 s390x	SUSE Linux Enterprise Micro 5.5 s390x
SLE Micro 5.4 x86-64	SUSE Linux Enterprise Micro 5.4 x86_64
SLE Micro 5.4 arm64	SUSE Linux Enterprise Micro 5.4 aarch64
SLE Micro 5.4 s390x	SUSE Linux Enterprise Micro 5.4 s390x
SLE Micro 5.3 x86-64	SUSE Linux Enterprise Micro 5.3 x86_64
SLE Micro 5.3 arm64	SUSE Linux Enterprise Micro 5.3 aarch64
SLE Micro 5.3 s390x	SUSE Linux Enterprise Micro 5.3 s390x
SLE Micro 5.2 x86-64	SUSE Linux Enterprise Micro 5.2 x86_64
SLE Micro 5.2 arm64	SUSE Linux Enterprise Micro 5.2 aarch64
SLE Micro 5.2 s390x	SUSE Linux Enterprise Micro 5.2 s390x
SLE Micro 5.1 x86-64	SUSE Linux Enterprise Micro 5.1 x86_64
SLE Micro 5.1 arm64	SUSE Linux Enterprise Micro 5.1 aarch64
SLE Micro 5.1 s390x	SUSE Linux Enterprise Micro 5.1 s390x

*Procedure: Adding Software Channels*

1. In the Uyuni Web UI, navigate to **Admin > Setup Wizard > Products**.
2. Locate the appropriate products for your client operating system and architecture using the search bar, and check the appropriate product. This will automatically check all mandatory channels. Also all recommended channels are checked as long as the **include recommended** toggle is turned on. Click the arrow to see the complete list of related products, and ensure that any extra products you require are checked.
3. Click **Add Products** and wait until the products have finished synchronizing.

Alternatively, you can add channels at the command prompt. The channels you need for this procedure are:

*Table 22. SLE Micro Products - CLI*

OS Version	Base Channel	Updates Channel
SLE Micro 5.5 x86-64	sle-micro-5.5-pool-x86_64	sle-micro-5.5-updates-x86_64
SLE Micro 5.5 arm64	sle-micro-5.5-pool-arm64	sle-micro-5.5-updates-arm64
SLE Micro 5.5 IBM Z (s390x)	sle-micro-5.5-pool-s390x	sle-micro-5.5-updates-s390x
SLE Micro 5.4 x86-64	sle-micro-5.4-pool-x86_64	sle-micro-5.4-updates-x86_64
SLE Micro 5.4 arm64	sle-micro-5.4-pool-arm64	sle-micro-5.4-updates-arm64
SLE Micro 5.4 IBM Z (s390x)	sle-micro-5.4-pool-s390x	sle-micro-5.4-updates-s390x
SLE Micro 5.3 x86-64	sle-micro-5.3-pool-x86_64	sle-micro-5.3-updates-x86_64
SLE Micro 5.3 arm64	sle-micro-5.3-pool-arm64	sle-micro-5.3-updates-arm64
SLE Micro 5.3 IBM Z (s390x)	sle-micro-5.3-pool-s390x	sle-micro-5.3-updates-s390x
SLE Micro 5.2 x86-64	suse-microos-5.2-pool-x86_64	suse-microos-5.2-updates-x86_64
SLE Micro 5.2 arm64	suse-microos-5.2-pool-aarch64	suse-microos-5.2-updates-aarch64
SLE Micro 5.2 IBM Z (s390x)	suse-microos-5.2-pool-s390x	suse-microos-5.2-updates-s390x
SLE Micro 5.1 x86-64	suse-microos-5.1-pool-x86_64	suse-microos-5.1-updates-x86_64
SLE Micro 5.1 arm64	suse-microos-5.1-pool-aarch64	suse-microos-5.1-updates-aarch64
SLE Micro 5.1 IBM Z (s390x)	suse-microos-5.1-pool-s390x	suse-microos-5.1-updates-s390x

*Procedure: Adding Software Channels at the Command Prompt*

1. At the command prompt on the Uyuni Server, as root, use the **mgr-sync** command to add the appropriate channels:

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

2. Synchronization starts automatically. If you want to synchronize the channels manually, use:

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

3. Ensure the synchronization is complete before continuing.

To add the client tools, add these channels from the command prompt:

*Table 23. SLE Micro Channels - CLI*

OS Version	Client Channel
SLE Micro 5.4	sle-micro-5.4-uyuni-client
SLE Micro 5.3	suse-micro-5.3-uyuni-client
SLE Micro 5.2	suse-microos-5.2-uyuni-client
SLE Micro 5.1	sle-microos-5.1-uyuni-client

*Procedure: Adding Software Channels at the Command Prompt*

1. At the command prompt on the Uyuni Server, as root, use the **spacewalk-common-channels** command to add the appropriate channels:

```
spacewalk-common-channels \
<base_channel_label> \
<child_channel_label_1> \
<child_channel_label_2> \
... <child_channel_label_n>
```

2. If **automatic synchronization** is turned off, synchronize the channels:

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

3. Ensure the synchronization is complete before continuing.

#### 4.2.2.2. Check Synchronization Status

*Procedure: Checking Synchronization Progress from the Web UI*

1. In the Uyuni Web UI, navigate to **Software > Manage > Channels**, then click the channel associated to the repository.
2. Navigate to the **Repositories** tab, then click **Sync** and check **Sync Status**.

**Procedure: Checking Synchronization Progress from the Command Prompt**

- At the command prompt on the Uyuni Server, as root, use the **tail** command to check the synchronization log file:

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

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

**4.2.2.3. Register Clients**

SLE Micro clients require a reboot after registering. Although a reboot is automatically scheduled after registration is completed, it is respecting the default reboot manager maintenance window. This window may be several hours after the client is registered. It is advisable to manually reboot the client after the registration script finishes, to speed up the registration and to see the system appear in the system list.

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

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

For more information on registering your clients, see **Client-configuration > Registration-overview**.

When using a bootstrap script with SLE Micro systems, ensure that the certificate section of the script has this content:

```
ORG_CA_CERT=RHN-ORG-TRUSTED-SSL-CERT
ORG_CA_CERT_IS_RPM_YN=0
```

Either edit the bootstrap script directly and add the settings, or create the bootstrap script with these parameters:

```
mgr-bootstrap --script=bootstrap-sle-micro.sh \
--ssl-cert=/srv/www/htdocs/pub/RHN-ORG-TRUSTED-SSL-CERT
```

**4.2.2.4. Reboot SLE Micro**

SLE Micro is a transactional system. Transactional updates in general support several reboot methods. It is recommended to use **systemd** for rebooting in systems managed by Uyuni. Using other methods can lead to undesired behavior.

When bootstrapping a transactional system on Uyuni, `systemd` will be configured as the reboot method (`REBOOT_METHOD`), if the system is in its default configuration. Such a configuration allows Uyuni to control the reboot action, and rebooting can be performed immediately or scheduled with Uyuni as wanted.

#### 4.2.2.4.1. Background Information

By default, the reboot method during client installation is set to `auto`. With the `auto` boot method, `rebootmgrd` will be used to reboot the system according to the configured policies if the service is running. Policies can be to reboot instantly or during a maintenance window. For more information, see the `rebootmgrd(8)` man page. Otherwise if `rebootmgrd` is not running, Uyuni will call `systemctl reboot`.



Using any method different from `systemd` may cause undesired behavior.

## 4.3. openSUSE Client Registration

You can register openSUSE and openSUSE Leap Micro clients to your Uyuni Server. The method and details varies depending on the operating system of the client.

Before you start, ensure that the client has the date and time synchronized correctly with the Uyuni Server.

You must also have created an activation key. For more information about creating activation keys, see [Client-configuration > Activation-keys](#).



Do not register a Uyuni Server to itself. The Uyuni Server must be managed individually or by using another separate Uyuni Server. For more information about using multiple servers, see [Specialized-guides > Large-deployments](#).

### 4.3.1. Registering openSUSE Leap Clients

This section contains information about registering clients running openSUSE operating systems. Uyuni supports openSUSE Leap 15 clients using Salt.

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

#### 4.3.1.1. Add Software Channels

Before you register openSUSE clients to your Uyuni Server, you need to add the required software channels, and synchronize them.

The architectures currently supported are: `x86_64` and `aarch64`. For full list of supported products and architectures, see [Client-configuration > Supported-features](#).



- In the following section, descriptions often default to the **x86\_64** architecture.
- Replace it with other architectures if appropriate.

For example, when working with **x86\_64** architecture, you need this product:

*Table 24. openSUSE Channels - CLI*

OS Version	openSUSE Leap 15.5	openSUSE Leap 15.4
Base Channel	opensuse_leap15_5	opensuse_leap15_4
Client Channel	opensuse_leap15_5-uyuni-client	opensuse_leap15_4-uyuni-client
Updates Channel	opensuse_leap15_5-updates	opensuse_leap15_4-updates
Non-OSS Channel	opensuse_leap15_5-non-oss	opensuse_leap15_4-non-oss
Non-OSS Updates Channel	opensuse_leap15_5-non-oss-updates	opensuse_leap15_4-non-oss-updates
Backports Updates Channel	opensuse_leap15_5-backports-updates	opensuse_leap15_4-backports-updates
SLE Updates Channel	opensuse_leap15_5-sle-updates	opensuse_leap15_4-sle-updates

*Procedure: Adding Software Channels at the Command Prompt*

- At the command prompt on the Uyuni Server, as root, use the **spacewalk-common-channels** command to add the appropriate channels:

```
spacewalk-common-channels \
<base_channel_label> \
<child_channel_label_1> \
<child_channel_label_2> \
... <child_channel_label_n>
```

- If **automatic synchronization** is turned off, synchronize the channels:

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

- Ensure the synchronization is complete before continuing.

#### 4.3.1.2. Check Synchronization Status

*Procedure: Checking Synchronization Progress from the Web UI*

- In the Uyuni Web UI, navigate to **Software > Manage > Channels**, then click the channel associated to the repository.
- Navigate to the **Repositories** tab, then click **Sync** and check **Sync Status**.

*Procedure: Checking Synchronization Progress from the Command Prompt*

- At the command prompt on the Uyuni Server, as root, use the **tail** command to check the synchronization log file:

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

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



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

#### 4.3.1.3. Manage GPG Keys

Clients use GPG keys to check the authenticity of software packages before they are installed. Only trusted software can be installed on clients.



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

For more information about GPG keys, see [Client-configuration > Gpg-keys](#).

#### 4.3.1.4. Register Clients

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

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

For more information on registering your clients, see [Client-configuration > Registration-overview](#).

### 4.3.2. Registering openSUSE Leap Micro Clients

This section contains information about registering clients running openSUSE Leap Micro operating systems on x86-64 and aarch64 architectures.

The openSUSE Leap Micro is an ultra-reliable, lightweight operating system purpose built for edge computing. It leverages the enterprise hardened security and compliance components of SUSE Linux Enterprise and merges them with a modern, immutable, developer-friendly OS platform.

The openSUSE Leap Micro uses transactional updates. Transactional updates are atomic (all updates are applied only if all updates succeed) and support rollbacks. They do not affect the running system because no changes are activated until the system is rebooted. This information is displayed in the [Systems >](#)

## Details > Overview subtab.

For more information on transactional updates and rebooting, see <https://documentation.suse.com/sles/html/SLES-all/cha-transactional-updates.html>.

*Table 25. openSUSE Channels - CLI*

OS Version	openSUSE Leap Micro 5.5	openSUSE Leap Micro 5.4	openSUSE Leap Micro 5.3
Base Channel	opensuse_micro5_5	opensuse_micro5_4	opensuse_micro5_3
Client Channel	opensuse_micro5_5-uyuni-client	opensuse_micro5_4-uyuni-client	opensuse_micro5_3-uyuni-client
SLE Updates Channel	opensuse_micro5_5-sle-updates	opensuse_micro5_4-sle-updates	opensuse_micro5_3-sle-updates

*Procedure: Adding Software Channels at the Command Prompt*

- At the command prompt on the Uyuni Server, as root, use the `spacewalk-common-channels` command to add the appropriate channels:

```
spacewalk-common-channels \
<base_channel_label> \
<child_channel_label_1> \
<child_channel_label_2> \
... <child_channel_label_n>
```

- If `automatic synchronization` is turned off, synchronize the channels:

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

- Ensure the synchronization is complete before continuing.

### 4.3.2.1. Check Synchronization Status

*Procedure: Checking Synchronization Progress from the Web UI*

- In the Uyuni Web UI, navigate to **Software > Manage > Channels**, then click the channel associated to the repository.
- Navigate to the **Repositories** tab, then click **Sync** and check **Sync Status**.

*Procedure: Checking Synchronization Progress from the Command Prompt*

- At the command prompt on the Uyuni Server, as root, use the `tail` command to check the synchronization log file:

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

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



- openSUSE Leap Micro channels can be very large. Synchronization can sometimes take several hours.

#### 4.3.2.2. Register Clients



- openSUSE Leap Micro clients require reboot after registering. Reboot is automatically scheduled after registration is completed, but it is respecting the default reboot manager maintenance window. This window may be several hours after the client is registered. To speed up openSUSE Leap Micro registration, manually reboot the client after the registration script finishes.

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

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

For more information on registering your clients, see [Client-configuration > Registration-overview](#).

## 4.4. Alibaba Cloud Linux Client Registration

You can register Alibaba Cloud Linux clients to your Uyuni Server. The method and details vary depending on the operating system of the client.

Before you start, ensure that the client has the date and time synchronized correctly with the Uyuni Server.

You must also have created an activation key. For more information about creating activation keys, see [Client-configuration > Activation-keys](#).

### 4.4.1. Registering Alibaba Cloud Linux Clients

This section contains information about registering clients running Alibaba Cloud Linux operating systems.



- Some Alibaba Cloud Linux 2 instances will need two tries to register successfully.

#### 4.4.1.1. Add Software Channels

Before you register Alibaba Cloud Linux clients to your Uyuni Server, you need to add the required software channels, and synchronize them.



- In the following section, descriptions often default to the **x86\_64** architecture.
- Replace it with other architectures if appropriate.

The channels you need for this procedure are:

*Table 26. Alibaba Cloud Linux Channels - CLI*

OS Version	Core Channel	Updates Channel	Client Channel
Alibaba Cloud Linux 2	alibaba-2	alibaba-2-updates	alibaba-2-uyuni-client

*Procedure: Adding Software Channels at the Command Prompt*

1. At the command prompt on the Uyuni Server, as root, use the **spacewalk-common-channels** command to add the appropriate channels:

```
spacewalk-common-channels \
<base_channel_label> \
<child_channel_label_1> \
<child_channel_label_2> \
... <child_channel_label_n>
```

2. If **automatic synchronization** is turned off, synchronize the channels:

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

3. Ensure the synchronization is complete before continuing.



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

#### 4.4.1.2. Check Synchronization Status

*Procedure: Checking Synchronization Progress from the Web UI*

1. In the Uyuni Web UI, navigate to **Software > Manage > Channels**, then click the channel associated to the repository.
2. Navigate to the **Repositories** tab, then click **Sync** and check **Sync Status**.

*Procedure: Checking Synchronization Progress from the Command Prompt*

1. At the command prompt on the Uyuni Server, as root, use the **tail** command to check the synchronization log file:

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

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

#### 4.4.1.3. Create an Activation Key

You need to create an activation key that is associated with your Alibaba Cloud Linux channels.

For more information on activation keys, see [Client-configuration > Activation-keys](#).

#### 4.4.1.4. Register Clients

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

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

For more information on registering your clients, see [Client-configuration > Registration-overview](#).

Some Alibaba Cloud Linux 2 instances will fail to register on the first try. This is due to a known bug in the Alibaba Cloud Linux 2 image.

The `python-urlgrabber3` package is provided both as a Python pip package and an RPM package, which can cause a conflict on the first attempt to register.

If your instance is based on one of the affected image versions, the client should register correctly on the second registration attempt.

## 4.5. AlmaLinux Client Registration

You can register AlmaLinux clients to your Uyuni Server. The method and details vary depending on the operating system of the client.

Before you start, ensure that the client has the date and time synchronized correctly with the Uyuni Server.

You must also have created an activation key. For more information about creating activation keys, see [Client-configuration > Activation-keys](#).

### 4.5.1. Registering AlmaLinux Clients

This section contains information about registering clients running AlmaLinux operating systems.



When created at AWS, AlmaLinux instances always have the same `machine-id` at `/etc/machine-id`. Make sure you regenerate the `machine-id` after the instance is created. For more information, see [Administration > Troubleshooting](#).

#### 4.5.1.1. Add Software Channels



Registering AlmaLinux clients to Uyuni is tested with the default SELinux configuration of **enforcing** with a **targeted** policy. You do not need to disable SELinux to register AlmaLinux clients to Uyuni.

Before you can register AlmaLinux clients to your Uyuni Server, you need to add the required software channels, and synchronize them.

The architectures currently supported are: **x86\_64** and **aarch64**, on version 9 additionally ppc64le and s390x. For full list of supported products and architectures, see **Client-configuration > Supported-features**.



In the following section, descriptions often default to the **x86\_64** architecture. Replace it with other architectures if appropriate.

The channels you need for this procedure are:

*Table 27. AlmaLinux Channels - CLI*

OS Version	Base Channel	Client Channel	AppStream Channel
AlmaLinux 9	almalinux9	almalinux9-uyuni-client	almalinux9-appstream
AlmaLinux 8	almalinux8	almalinux8-uyuni-client	almalinux8-appstream

*Procedure: Adding Software Channels at the Command Prompt*

- At the command prompt on the Uyuni Server, as root, use the **spacewalk-common-channels** command to add the appropriate channels. Ensure you specify the correct architecture:

```
spacewalk-common-channels \
-a <architecture> \
<base_channel_name> \
<child_channel_name_1> \
<child_channel_name_2> \
... <child_channel_name_n>
```

- If **automatic synchronization** is turned off, synchronize the channels:

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

- Ensure the synchronization is complete before continuing.



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



For AlmaLinux 9 and AlmaLinux 8 clients, add both the Base and AppStream channels. You require packages from both channels. If you do not add both channels, you cannot create the bootstrap repository, due to missing packages.

If you are using modular channels, you must enable the Python 3.6 module stream on the AlmaLinux 8 client. If you do not provide Python 3.6, the installation of the **spacecmd** package will fail.



You might notice some disparity in the number of packages available in the AppStream channel between upstream and the Uyuni channel. You might also see different numbers if you compare the same channel added at a different point in time. This is due to the way that AlmaLinux manages their repositories. AlmaLinux removes older version of packages when a new version is released, while Uyuni keeps all of them, regardless of age.



The AppStream repository provides modular packages. This results in the Uyuni Web UI showing incorrect package information. You cannot perform package operations such as installing or upgrading directly from modular repositories using the Web UI or API.

Alternatively, you can use Salt states to manage modular packages, or use the **dnf** command on the client. For more information about CLM, see [Administration > Content-lifecycle](#).

#### 4.5.1.2. Check Synchronization Status

*Procedure: Checking Synchronization Progress from the Web UI*

1. In the Uyuni Web UI, navigate to **Software > Manage > Channels**, then click the channel associated to the repository.
2. Navigate to the **Repositories** tab, then click **Sync** and check **Sync Status**.

*Procedure: Checking Synchronization Progress from the Command Prompt*

1. At the command prompt on the Uyuni Server, as root, use the **tail** command to check the synchronization log file:

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

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

#### 4.5.1.3. Create an Activation Key

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

For more information on activation keys, see [Client-configuration > Activation-keys](#).

#### 4.5.1.4. Manage GPG Keys

Clients use GPG keys to check the authenticity of software packages before they are installed. Only trusted software can be installed on clients.



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

For more information about GPG keys, see [Client-configuration > Gpg-keys](#).

#### 4.5.1.5. Register Clients

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

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

For more information on registering your clients, see [Client-configuration > Registration-overview](#).

#### 4.5.1.6. Manage Errata

When you update AlmaLinux clients, the packages include metadata about the updates.

## 4.6. Amazon Linux Client Registration

You can register Amazon Linux clients to your Uyuni Server. The method and details vary depending on the operating system of the client.

Before you start, ensure that the client has the date and time synchronized correctly with the Uyuni Server.

You must also have created an activation key. For more information about creating activation keys, see [Client-configuration > Activation-keys](#).

### 4.6.1. Registering Amazon Linux Clients

This section contains information about registering clients running Amazon Linux operating systems.



When created at AWS, Amazon Linux 2 instances always have the same **machine-id** id at `/etc/machine-id`. If you are creating Amazon Linux 2 instances, make sure you regenerate the **machine-id** after the instances are created. For more information, see [Administration > Troubleshooting](#).

Amazon Linux 2023 is not affected by this.

#### 4.6.1.1. Add Software Channels

Before you register Amazon Linux clients to your Uyuni Server, you need to add the required software

channels, and synchronize them.

The architectures currently supported are: **x86\_64** and **aarch64**. For full list of supported products and architectures, see [Client-configuration > Supported-features](#).



- In the following section, descriptions often default to the **x86\_64** architecture. Replace it with other architectures if appropriate.

The channels you need for this procedure are:

*Table 28. Amazon Linux Channels - CLI*

OS Version	Base Channel	Client Channel
Amazon Linux 2023	amazonlinux2023	amazonlinux2023-uyuni-client
Amazon Linux 2	amazonlinux2-core	amazonlinux2-uyuni-client



- For Amazon Linux 2, make sure you also add and synchronize **amazonlinux2-extra-docker** channel if you plan to use Docker at your Amazon Linux instances.



- For Amazon Linux 2023, make sure you also add and synchronize **amazonlinux2023-kernel-livepatch** channel if you plan to use Kernel Live patches at your Amazon Linux instances.

#### *Procedure: Adding Software Channels at the Command Prompt*

1. At the command prompt on the Uyuni Server, as root, use the **spacewalk-common-channels** command to add the appropriate channels:

```
spacewalk-common-channels \
<base_channel_label> \
<child_channel_label_1> \
<child_channel_label_2> \
... <child_channel_label_n>
```

2. If **automatic synchronization** is turned off, synchronize the channels:

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

3. Ensure the synchronization is complete before continuing.



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

#### *4.6.1.2. Check Synchronization Status*

##### *Procedure: Checking Synchronization Progress from the Web UI*

1. In the Uyuni Web UI, navigate to **Software > Manage > Channels**, then click the channel associated to the repository.
2. Navigate to the **Repositories** tab, then click **Sync** and check **Sync Status**.

*Procedure: Checking Synchronization Progress from the Command Prompt*

1. At the command prompt on the Uyuni Server, as root, use the **tail** command to check the synchronization log file:

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

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

#### 4.6.1.3. Create an Activation Key

You need to create an activation key that is associated with your Amazon Linux channels.

For more information on activation keys, see **Client-configuration > Activation-keys**.

#### 4.6.1.4. Manage GPG Keys

Clients use GPG keys to check the authenticity of software packages before they are installed. Only trusted software can be installed on clients.



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

For more information about GPG keys, see **Client-configuration > Gpg-keys**.

#### 4.6.1.5. Register Clients

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

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

For more information on registering your clients, see **Client-configuration > Registration-overview**.

## 4.7. CentOS Client Registration

You can register CentOS clients to your Uyuni Server. The method and details varies depending on the operating system of the client.

Before you start, ensure that the client has the date and time synchronized correctly with the Uyuni Server.

You must also have created an activation key. For more information about creating activation keys, see [Client-configuration > Activation-keys](#).

## 4.7.1. Registering CentOS Clients

This section contains information about registering clients running CentOS operating systems.



You are responsible for arranging access to CentOS base media repositories and CentOS installation media, as well as connecting Uyuni Server to the CentOS content delivery network.



Registering CentOS clients to Uyuni is tested with the default SELinux configuration of **enforcing** with a **targeted** policy. You do not need to disable SELinux to register CentOS clients to Uyuni.

### 4.7.1.1. Add Software Channels

Before you can register CentOS clients to your Uyuni Server, you need to add the required software channels, and synchronize them.

The architectures currently supported are: **x86\_64** and **aarch64**. For full list of supported products and architectures, see [Client-configuration > Supported-features](#).



In the following section, descriptions often default to the **x86\_64** architecture. Replace it with other architectures if appropriate.

The channels you need for this procedure are:

*Table 29. CentOS Channels - CLI*

OS Version	Base Channel	Client Channel	Updates/Appstream Channel
CentOS 7	centos7	centos7-uyuni-client	centos7-updates

#### *Procedure: Adding Software Channels at the Command Prompt*

- At the command prompt on the Uyuni Server, as root, use the **spacewalk-common-channels** command to add the appropriate channels. Ensure you specify the correct architecture:

```
spacewalk-common-channels \
-a <architecture> \
<base_channel_name> \
<child_channel_name_1> \
<child_channel_name_2> \
... <child_channel_name_n>
```

2. If [automatic synchronization](#) is turned off, synchronize the channels:

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

3. Ensure the synchronization is complete before continuing.



The client tools channel provided by [spacewalk-common-channels](#) is sourced from Uyuni and not from SUSE.

If you are using modular channels, you must enable the Python 3.6 module stream on the client. If you do not provide Python 3.6, the installation of the [spacecmd](#) package will fail.



You might notice some disparity in the number of packages available in the AppStream channel between upstream and the Uyuni channel. You might also see different numbers if you compare the same channel added at a different point in time. This is due to the way that CentOS manages their repositories. CentOS removes older version of packages when a new version is released, while Uyuni keeps all of them, regardless of age.



The AppStream repository provides modular packages. This results in the Uyuni Web UI showing incorrect package information. You cannot perform package operations such as installing or upgrading directly from modular repositories using the Web UI or API.

Alternatively, you can use Salt states to manage modular packages, or use the [dnf](#) command on the client. For more information about CLM, see [Administration > Content-lifecycle](#).

#### 4.7.1.2. Check Synchronization Status

##### *Procedure: Checking Synchronization Progress from the Web UI*

1. In the Uyuni Web UI, navigate to **Software > Manage > Channels**, then click the channel associated to the repository.
2. Navigate to the **Repositories** tab, then click **Sync** and check **Sync Status**.

##### *Procedure: Checking Synchronization Progress from the Command Prompt*

1. At the command prompt on the Uyuni Server, as root, use the [tail](#) command to check the synchronization log file:

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

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

#### 4.7.1.3. Create an Activation Key

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

For more information on activation keys, see **Client-configuration > Activation-keys**.

#### 4.7.1.4. Manage GPG Keys

Clients use GPG keys to check the authenticity of software packages before they are installed. Only trusted software can be installed on clients.



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

For more information about GPG keys, see **Client-configuration > Gpg-keys**.

#### 4.7.1.5. Register Clients

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

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

For more information on registering your clients, see **Client-configuration > Registration-overview**.

#### 4.7.1.6. Manage Errata

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



The authors of CEFS provide patches or errata on a best-effort basis, in the hope they are useful but with no guarantees of correctness or currency. This could mean that the patch dates could be incorrect, and in at least one case, the published data was shown to be more than a month old. For more information on these cases, see <https://github.com/stevemeier/cefs/issues/28#issuecomment-656579382> and <https://github.com/stevemeier/cefs/issues/28#issuecomment-656573607>.

Any problems or delays with the patch data might result in unreliable patch information being imported to your Uyuni Server. This would cause reports, audits, CVE updates, or other patch-related information to also be incorrect. Please consider alternatives to using this service, such as independently verifying patch data, or choosing a different operating system, depending on your security-related requirements and certifications criteria.

**Procedure: Installing an Errata Service**

1. On the Uyuni Server, from the command prompt, as root, add the **sle-module-development-tools** module:

```
SUSEConnect --product sle-module-development-tools/15.2/x86_64
```

2. Install errata service dependencies:

```
zypper in perl-Text-Unidecode
```

3. Add or edit this line in **/etc/rhn/rhn.conf**:

```
java.allow_adding_patches_via_api = centos7-x86_64-updates,centos7-x86_64,centos7-x86_64-extras
```

4. Restart Tomcat:

```
systemctl restart tomcat
```

5. Create a file for your errata script:

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

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

```
#!/bin/bash
mkdir -p /usr/local/centos
cd /usr/local/centos
rm *.xml
wget -c http://cefs.steve-meier.de/errata.latest.xml
wget -c https://www.redhat.com/security/data/oval/v2/RHEL7/rhel-7.oval.xml.bz2
bzip2 -d rhel-7.oval.xml.bz2
wget -c http://cefs.steve-meier.de/errata-import.tar
tar xvf errata-import.tar
chmod +x /usr/local/centos/errata-import.pl
export SPACEWALK_USER='<adminname>';export SPACEWALK_PASS='<password>'
/usr/local/centos/errata-import.pl --server '<servername>' \
--errata /usr/local/centos/errata.latest.xml \
--include-channels=centos7-x86_64-updates,centos7-x86_64,centos7-x86_64-extras \
--publish --rhsa-oval /usr/local/centos/rhel-7.oval.xml
```

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

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

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

## 4.8. Debian Client Registration

You can register Debian clients to your Uyuni Server. The method and details varies depending on the operating system of the client.

Before you start, ensure that the client has the date and time synchronized correctly with the Uyuni Server.

You must also have created an activation key. For more information about creating activation keys, see **Client-configuration > Activation-keys**.

### 4.8.1. Registering Debian Clients

This section contains information about registering clients running Debian operating systems.

Bootstrapping can be used with Debian clients for performing initial state runs, and for profile updates.

#### 4.8.1.1. Prepare to Register

Some preparation is required before you can register Debian clients to the Uyuni Server:

- Ensure DNS is correctly configured and provides an entry for the client. Alternatively, you can configure the **/etc/hosts** files on both the Uyuni Server and the client with the appropriate entries.
- The client must have the date and time synchronized with the Uyuni Server before registration.

#### 4.8.1.2. Add Software Channels

Before you can register Debian clients to your Uyuni Server, you need to add the required software channels, and synchronize them.



- In the following section, descriptions often default to the **x86\_64** architecture.  
Replace it with other architectures if appropriate.

The channels you need for this procedure are:

*Table 30. Debian Channels - CLI*

OS Version	Base Channel	Client Channel	Updates Channel	Security Channel
Debian 12	debian-12-pool-amd64-uyuni	debian-12-amd64-uyuni-client	debian-12-amd64-main-updates-uyuni	debian-12-amd64-main-security-uyuni
Debian 11	debian-11-pool-amd64-uyuni	debian-11-amd64-uyuni-client	debian-11-amd64-main-updates-uyuni	debian-11-amd64-main-security-uyuni

OS Version	Base Channel	Client Channel	Updates Channel	Security Channel
Debian 10	debian-10-pool-amd64-uyuni	debian-10-amd64-uyuni-client	debian-10-amd64-main-updates-uyuni	debian-10-amd64-main-security-uyuni

*Procedure: Adding Software Channels at the Command Prompt*

- At the command prompt on the Uyuni Server, as root, use the **spacewalk-common-channels** command to add the appropriate channels:

```
spacewalk-common-channels \
<base_channel_label> \
<child_channel_label_1> \
<child_channel_label_2> \
... <child_channel_label_n>
```

- If **automatic synchronization** is turned off, synchronize the channels:

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

- Ensure the synchronization is complete before continuing.

**4.8.1.3. Check Synchronization Status***Procedure: Checking Synchronization Progress from the Web UI*

- In the Uyuni Web UI, navigate to **Software > Manage > Channels**, then click the channel associated to the repository.
- Navigate to the **Repositories** tab, then click **Sync** and check **Sync Status**.

*Procedure: Checking Synchronization Progress from the Command Prompt*

- At the command prompt on the Uyuni Server, as root, use the **tail** command to check the synchronization log file:

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

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



- Debian channels can be very large. Synchronization can sometimes take several hours.

**4.8.1.4. Manage GPG Keys**

Clients use GPG keys to check the authenticity of software packages before they are installed. Only trusted software can be installed on clients.



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

For more information about GPG keys, see [Client-configuration > Gpg-keys](#).



Debian clients can require multiple GPG keys to be installed.

When synchronizing third-party Debian repositories, you will need to import the appropriate GPG key on the server. If the GPG key is missing, synchronization will fail.

For Debian repositories, only the metadata is signed. Therefore importing a GPG key for the software channel is not needed. Packages will not be re-signed by Uyuni.

To see which GPG keys are already imported to Uyuni Server, run the following command:

```
sudo gpg --homedir /var/lib/spacewalk/gpgdir --list-keys
```

To import a new GPG key, use the **--import** parameter:

```
sudo gpg --homedir /var/lib/spacewalk/gpgdir --import <filename>.gpg
```

#### 4.8.1.5. Root Access

The root user on Debian is disabled by default for SSH access.

To be able to onboard using a regular user, you need to edit the **sudoers** file.

##### *Procedure: Granting Root User Access*

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

```
sudo visudo
```

Grant **sudo** access to the user by adding this line at the end of the **sudoers** file. Replace **<user>** with the name of the user that is bootstrapping the client in the Web UI:

```
<user> ALL=NOPASSWD: /usr/bin/python, /usr/bin/python2, /usr/bin/python3,  
/var/tmp/venv-salt-minion/bin/python
```



This procedure grants root access without requiring a password, which is required for registering the client. When the client is successfully installed it runs with root privileges, so the access is no longer required. We recommend that you remove the line from the **sudoers** file after the client has been successfully

- installed.

#### 4.8.1.6. Register Clients

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

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

For more information on registering your clients, see [Client-configuration > Registration-overview](#).

## 4.9. OpenEuler Client Registration

You can register openEuler clients to your Uyuni Server. The method and details varies depending on the operating system of the client.

Before you start, ensure that the client has the date and time synchronized correctly with the Uyuni Server.

You must also have created an activation key. For more information about creating activation keys, see [Client-configuration > Activation-keys](#).

### 4.9.1. Registering openEuler Clients

This section contains information about registering clients running openEuler operating systems.



- When created at AWS, openEuler instances always have the same **machine-id** id at **/etc/machine-id**. Make sure you regenerate the **machine-id** after the instance is created. For more information, see [Administration > Troubleshooting](#).

#### 4.9.1.1. Add Software Channels

Before you register openEuler clients to your Uyuni Server, you need to add the required software channels, and synchronize them.

The architectures currently supported are: **x86\_64** and **aarch64**. For full list of supported products and architectures, see [Client-configuration > Supported-features](#).



- In the following section, descriptions often default to the **x86\_64** architecture. Replace it with other architectures if appropriate.

The channels you need for this procedure are:

*Table 31. openEuler Channels - CLI*

OS Version	Core Channel	Client Channel	Update Channel	EPOL Channel	Everything Channel
openEuler 22.03	openeuler2203	openeuler2203-uyuni-client	openeuler2203-update	openeuler2203-epol	openeuler2203-everything

*Procedure: Adding Software Channels at the Command Prompt*

- At the command prompt on the Uyuni Server, as root, use the **spacewalk-common-channels** command to add the appropriate channels:

```
spacewalk-common-channels \
<base_channel_label> \
<child_channel_label_1> \
<child_channel_label_2> \
... <child_channel_label_n>
```

- If **automatic synchronization** is turned off, synchronize the channels:

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

- Ensure the synchronization is complete before continuing.



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

**4.9.1.2. Check Synchronization Status***Procedure: Checking Synchronization Progress from the Web UI*

- In the Uyuni Web UI, navigate to **Software > Manage > Channels**, then click the channel associated to the repository.
- Navigate to the **Repositories** tab, then click **Sync** and check **Sync Status**.

*Procedure: Checking Synchronization Progress from the Command Prompt*

- At the command prompt on the Uyuni Server, as root, use the **tail** command to check the synchronization log file:

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

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

**4.9.1.3. Create an Activation Key**

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

For more information on activation keys, see [Client-configuration > Activation-keys](#).

#### 4.9.1.4. Trust GPG Keys on Clients

#### 4.9.1.5. Trust GPG Keys on Clients

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

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

The GPG key URL parameter in the software channel page can contain multiple key URLs separated by "whitespace". In case it is a file URL, the GPG key file must be deployed on the client before the software channel is used.

The GPG keys for the Client Tools Channels of Red Hat based clients are deployed on the client into `/etc/pki/rpm-gpg/` and can be referenced with file URLs.

Only in case a software channel is assigned to the client they will be imported and trusted by the system.



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

##### 4.9.1.5.1. User defined GPG keys

Users can define custom GPG keys to be deployed to a client.

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

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

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

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

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

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

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

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

#### 4.9.1.5.2. GPG Keys in Bootstrap Scripts

*Procedure: Trusting GPG Keys on Clients Using a Bootstrap Script*

1. On the Uyuni Server, at the command prompt, check the contents of the `/srv/www/htdocs/pub/` directory. This directory contains all available public keys. Take a note of the key that applies to the channel assigned to the client you are registering.
2. Open the relevant bootstrap script, locate the `ORG_GPG_KEY=` parameter and add the required key. For example:

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

You do not need to delete any previously stored keys.



Trusting a GPG key is important for security on clients. It is the task of the administrator to decide which keys are needed and can be trusted. A software channel cannot be assigned to a client when the GPG key is not trusted.

#### 4.9.1.6. Register Clients

openEuler clients are registered in the same way as all other clients. For more information, see **Client-configuration > Registration-overview**.

## 4.10. Oracle Client Registration

You can register Oracle Linux clients to your Uyuni Server. The method and details varies depending on the operating system of the client.

Before you start, ensure that the client has the date and time synchronized correctly with the Uyuni Server.

You must also have created an activation key. For more information about creating activation keys, see

## Client-configuration > Activation-keys.

### 4.10.1. Registering Oracle Linux Clients

This section contains information about registering clients running Oracle Linux operating systems.



Direct synchronizing Unbreakable Linux Network (ULN) repositories with Uyuni are not currently supported. An Oracle Local Distribution for ULN must be used. For more information about setting up a local ULN mirror, see the Oracle documentation provided at <https://docs.oracle.com/en/operating-systems/oracle-linux/software-management/sfw-mgmt-UseSoftwareDistributionMirrors.html#local-uln-mirror>.

#### 4.10.1.1. Add Software Channels

Before you register Oracle Linux clients to your Uyuni Server, you need to add the required software channels, and synchronize them.

The architectures currently supported are: **x86\_64** and **aarch64**. For full list of supported products and architectures, see **Client-configuration > Supported-features**.



In the following section, descriptions often default to the **x86\_64** architecture. Replace it with other architectures if appropriate.

The channels you need for this procedure are:

*Table 32. Oracle Channels - CLI*

OS Version	Base Channel	Client Channel	Updates Channel
Oracle Linux 9	oraclelinux9	oraclelinux9-uyuni-client	oraclelinux9-appstream
Oracle Linux 8	oraclelinux8	oraclelinux8-uyuni-client	oraclelinux8-appstream
Oracle Linux 7	oraclelinux7	oraclelinux7-uyuni-client	-

#### *Procedure: Adding Software Channels at the Command Prompt*

- At the command prompt on the Uyuni Server, as root, use the **spacewalk-common-channels** command to add the appropriate channels:

```
spacewalk-common-channels \
<base_channel_label> \
<child_channel_label_1> \
<child_channel_label_2> \
... <child_channel_label_n>
```

- If **automatic synchronization** is turned off, synchronize the channels:

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

3. Ensure the synchronization is complete before continuing.



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



For Oracle Linux 9 and Oracle Linux 8 clients, add both the Base and AppStream channels. You require packages from both channels. If you do not add both channels, you cannot create the bootstrap repository, due to missing packages.

If you are using modular channels, you must enable the Python 3.6 module stream on the client. If you do not provide Python 3.6, the installation of the **spacecmd** package will fail.



The AppStream repository provides modular packages. This results in the Uyuni Web UI showing incorrect package information. You cannot perform package operations such as installing or upgrading directly from modular repositories using the Web UI or API.

Alternatively, you can use Salt states to manage modular packages, or use the **dnf** command on the client. For more information about CLM, see **Administration > Content-lifecycle**.

#### 4.10.1.2. Check Synchronization Status

*Procedure: Checking Synchronization Progress from the Web UI*

1. In the Uyuni Web UI, navigate to **Software > Manage > Channels**, then click the channel associated to the repository.
2. Navigate to the **Repositories** tab, then click **Sync** and check **Sync Status**.

*Procedure: Checking Synchronization Progress from the Command Prompt*

1. At the command prompt on the Uyuni Server, as root, use the **tail** command to check the synchronization log file:

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

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

#### 4.10.1.3. Create an Activation Key

You need to create an activation key that is associated with your Oracle Linux channels.

For more information on activation keys, see **Client-configuration > Activation-keys**.

#### 4.10.1.4. Manage GPG Keys

Clients use GPG keys to check the authenticity of software packages before they are installed. Only trusted software can be installed on clients.



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

For more information about GPG keys, see [Client-configuration > Gpg-keys](#).



For Oracle Linux 9 and Oracle Linux 8 clients use

```
ol8-gpg-pubkey-82562EA9AD986DA3.key
```

For Oracle Linux 7 clients use

```
ol67-gpg-pubkey-72F97B74EC551F0A3.key
```

#### 4.10.1.5. Register Clients

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

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

For more information on registering your clients, see [Client-configuration > Registration-overview](#).

### 4.11. Raspberry Pi OS Client Registration

You can register Raspberry Pi OS clients to your Uyuni Server. The method and details depend on the operating system of the client.

Before you start, ensure that the client has the date and time synchronized correctly with the Uyuni Server.

You must also have created an activation key. For more information about creating activation keys, see [Client-configuration > Activation-keys](#).

#### 4.11.1. Registering Raspberry Pi OS Clients

This section contains information about registering clients running Raspberry Pi OS operating systems.

Bootstrapping can be used with Raspberry Pi OS clients for performing initial state runs, and for profile updates.

#### 4.11.1.1. Prepare to Register

Some preparation is required before you can register Raspberry Pi OS clients to the Uyuni Server:

- Ensure DNS is correctly configured and provides an entry for the client. Alternatively, you can configure the `/etc/hosts` files on both the Uyuni Server and the client with the appropriate entries.
- The client must have the date and time synchronized with the Uyuni Server before registration.

#### 4.11.1.2. Add Software Channels

Before you register Raspberry Pi OS clients to your Uyuni Server, you need to add the required software channels, and synchronize them.

The architectures currently supported are: arm64 and armhf. For full list of supported products and architectures, see [Client-configuration > Supported-features](#).

The channels you need for this procedure are:

*Table 33. Raspberry Pi OS Channels - CLI*

Channel description	arm64	armhf
Base Channel	raspberrypios-12-pool-arm64-uyuni	raspberrypios-12-pool-armhf-uyuni
Client Channel	raspberrypios-12-arm64-uyuni-client	raspberrypios-12-armhf-uyuni-client
Updates Channel	raspberrypios-12-arm64-main-updates-uyuni	-
Contributions Channel	raspberrypios-12-arm64-contrib-uyuni	raspberrypios-12-armhf-contrib-uyuni
Non Free Channel	raspberrypios-12-arm64-non-free-uyuni	raspberrypios-12-armhf-non-free-uyuni
Non Free Firmware Channel	raspberrypios-12-arm64-non-free-firmware-uyuni	-
Raspberry Channel	raspberrypios-12-arm64-raspberry-uyuni	raspberrypios-12-armhf-raspberry-uyuni
Contribution Updates	raspberrypios-12-arm64-contrib-updates-uyuni	-

Channel description	arm64	armhf
Non Free Updates	raspberrypios-12-arm64-non-free-updates-uyuni	-
Non Free Firmware Updates	raspberrypios-12-arm64-non-free-firmware-updates-uyuni	-
Security Main Channel	raspberrypios-12-arm64-main-security-uyuni	-
Security Contribution Channel	raspberrypios-12-arm64-contrib-security-uyuni	-
Security Non Free Channel	raspberrypios-12-arm64-non-free-security-uyuni	-
Security Non Free Firmware Channel	raspberrypios-12-arm64-non-free-firmware-security-uyuni	-
RPI Channel	-	raspberrypios-12-armhf-rpi-uyuni

*Procedure: Adding Software Channels at the Command Prompt*

- At the command prompt on the Uyuni Server, as root, use the **spacewalk-common-channels** command to add the appropriate channels:

```
spacewalk-common-channels \
<base_channel_label> \
<child_channel_label_1> \
<child_channel_label_2> \
... <child_channel_label_n>
```

- If **automatic synchronization** is turned off, synchronize the channels:

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

- Ensure the synchronization is complete before continuing.



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



- You need all the new channels fully synchronized before bootstrapping any Raspberry Pi OS client.

#### 4.11.1.3. Check Synchronization Status

*Procedure: Checking Synchronization Progress from the Web UI*

- In the Uyuni Web UI, navigate to **Software > Manage > Channels**, then click the channel associated

to the repository.

2. Navigate to the **Repositories** tab, then click **Sync** and check **Sync Status**.

*Procedure: Checking Synchronization Progress from the Command Prompt*

1. At the command prompt on the Uyuni Server, as root, use the **tail** command to check the synchronization log file:

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

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



- Raspberry Pi OS channels can be very large. Synchronization can sometimes take several hours.

#### 4.11.1.4. Create an Activation Key

You need to create an activation key that is associated with your Raspberry Pi OS channels.

For more information on activation keys, see **Client-configuration > Activation-keys**.

#### 4.11.1.5. Manage GPG Keys

Clients use GPG keys to check the authenticity of software packages before they are installed. Only trusted software can be installed on clients.



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

For more information about GPG keys, see **Client-configuration > Gpg-keys**.



- Raspberry Pi OS clients can require multiple GPG keys to be installed.

When synchronizing third-party Raspberry Pi OS repositories, you will need to import the appropriate GPG key on the server. If the GPG key is missing, synchronization will fail.

For Raspberry Pi OS repositories, only the metadata is signed. Therefore importing a GPG key for the software channel is not needed. Packages will not be re-signed by Uyuni.

To see which GPG keys are already imported to Uyuni Server, run the following command:

```
sudo gpg --homedir /var/lib/spacewalk/gpgdir --list-keys
```

To import a new GPG key, use the `--import` parameter:

```
sudo gpg --homedir /var/lib/spacewalk/gpgdir --import <filename>.gpg
```

#### 4.11.1.6. Root Access

The root user on Raspberry Pi OS is disabled by default for SSH access.

To be able to onboard using a regular user, you need to edit the `sudoers` file.

*Procedure: Granting Root User Access*

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

```
sudo visudo
```

Grant `sudo` access to the user by adding this line at the end of the `sudoers` file. Replace `<user>` with the name of the user that is bootstrapping the client in the Web UI:

```
<user> ALL=NOPASSWD: /usr/bin/python, /usr/bin/python2, /usr/bin/python3,  
/var/tmp/venv-salt-minion/bin/python
```



This procedure grants root access without requiring a password, which is required for registering the client. When the client is successfully installed it runs with root privileges, so the access is no longer required. We recommend that you remove the line from the `sudoers` file after the client has been successfully installed.

#### 4.11.1.7. Register Clients

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

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

For more information on registering your clients, see [Client-configuration > Registration-overview](#).

## 4.12. Red Hat Client Registration

You can register Red Hat Enterprise Linux clients to your Uyuni Server using either the Red Hat content delivery network (CDN), or Red Hat update infrastructure (RHUI). The method and details varies depending on the operating system of the client.

Before you start, ensure that the client has the date and time synchronized correctly with the Uyuni Server.

You must also have created an activation key. For more information about creating activation keys, see [Client-configuration > Activation-keys](#).

### 4.12.1. Registering Red Hat Enterprise Linux Clients with CDN

This section contains information about using the Red Hat content delivery network (CDN) to register clients running Red Hat Enterprise Linux operating systems.

For information about using Red Hat update infrastructure (RHUI) instead, see [Client-configuration > Clients-rh-rhui](#).



You are responsible for arranging access to Red Hat base media repositories and RHEL installation media, as well as connecting Uyuni Server to the Red Hat content delivery network. You must obtain support from Red Hat for all your RHEL systems. If you do not do this, you might be violating your terms with Red Hat.

#### 4.12.1.1. Import Entitlements and Certificates

Red Hat clients require a Red Hat certificate authority (CA) and entitlement certificate, and an entitlement key.

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

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

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

Red Hat clients can be registered in three different ways:

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

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

For information about using Red Hat update infrastructure (RHUI) instead, see [Client-configuration > Clients-rh-rhui](#).



- Satellite certificates for client systems require a Satellite server and subscription.
- Clients using Satellite certificates are not supported with Uyuni Server.



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

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

*Procedure: Registering Clients to Subscription Manager*

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

```
subscription-manager register
```

Enter your Red Hat Portal username and password when prompted.

2. Run command:

```
subscription-manager activate
```

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

```
cp /etc/pki/entitlement/ <example>/entitlement/
```



- Your entitlement certificate and key both have a file extension of **.pem**. The key also has **key** in the filename.

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

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

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

*Procedure: Importing Certificates to the Server*

1. On the Uyuni Server Web UI, navigate to **Systems > Autoinstallation > GPG and SSL Keys**.
2. Click **Create Stored Key/Cert** and set these parameters for the entitlement certificate:
  - In the **Description** field, type **Entitlement-Cert-date**.

- In the **Type** field, select **SSL**.
  - In the **Select file to upload** field, browse to the location where you saved the entitlement certificate, and select the **.pem** certificate file.
3. Click **Create Key**.
4. Click **Create Stored Key/Cert** and set these parameters for the entitlement key:
- In the **Description** field, type **Entitlement-key-date**.
  - In the **Type** field, select **SSL**.
  - In the **Select file to upload** field, browse to the location where you saved the entitlement key, and select the **.pem** key file.
5. Click **Create Key**.
6. Click **Create Stored Key/Cert** and set these parameters for the Red Hat certificate:
- In the **Description** field, type **redhat-uep**.
  - In the **Type** field, select **SSL**.
  - In the **Select file to upload** field, browse to the location where you saved the Red Hat certificate, and select the certificate file.
7. Click **Create Key**.

#### 4.12.1.2. Add Software Channels

Before you register Red Hat clients to your Uyuni Server, you need to add the required software channels, and synchronize them.



In the following section, descriptions often default to the **x86\_64** architecture. Replace it with other architectures if appropriate.

The channels you need for this procedure are:

*Table 34. Red Hat Channels - CLI*

OS Version	Base Channel	Client Channel	Tools Channel
Red Hat 9	rhel9-pool-uyuni	-	rhel9-uyuni-client
Red Hat 8	rhel8-pool-uyuni	-	rhel8-uyuni-client
Red Hat 7	rhel7-pool-uyuni	-	rhel7-uyuni-client

#### *Procedure: Adding Software Channels at the Command Prompt*

1. At the command prompt on the Uyuni Server, as root, use the **spacewalk-common-channels** command to add the appropriate channels. Ensure you specify the correct architecture:

```
spacewalk-common-channels \
```

```
-a <architecture> \
<base_channel_name> \
<child_channel_name_1> \
<child_channel_name_2> \
... <child_channel_name_n>
```

2. If [automatic synchronization](#) is turned off, synchronize the channels:

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

3. Ensure the synchronization is complete before continuing.



The client tools channel provided by [spacewalk-common-channels](#) is sourced from Uyuni and not from SUSE.



The AppStream repository provides modular packages. This results in the Uyuni Web UI showing incorrect package information. You cannot perform package operations such as installing or upgrading directly from modular repositories using the Web UI or API.

Alternatively, you can use Salt states to manage modular packages, or use the `dnf` command on the client. For more information about CLM, see [Administration > Content-lifecycle](#).

#### 4.12.1.3. Prepare Custom Repositories and Channels

To mirror the software from the Red Hat CDN, you need to create custom channels and repositories in Uyuni that are linked to the CDN by a URL. You must have entitlements to these products in your Red Hat Portal for this to work correctly. You can use the subscription manager tool to get the URLs of the repositories you want to mirror:

```
subscription-manager repos
```

You can use these repository URLs to create custom repositories. This allows you to mirror only the content you need to manage your clients.



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

The details you need for this procedure are:

*Table 35. Red Hat Custom Repository Settings*

Option	Setting
Repository URL	The content URL provided by Red Hat CDN
Has Signed Metadata?	Uncheck all Red Hat Enterprise repositories

Option	Setting
SSL CA Certificate	redhat-uep
SSL Client Certificate	Entitlement-Cert-date
SSL Client Key	Entitlement-Key-date

*Procedure: Creating Custom Repositories*

1. On the Uyuni Server Web UI, navigate to **Software > Manage > Repositories**.
2. Click **Create Repository** and set the appropriate parameters for the repository.
3. Click **Create Repository**.
4. Repeat for all repositories you need to create.

The channels you need for this procedure are:

*Table 36. Red Hat Custom Channels*

OS Version	Base Channel
Red Hat 9	rhel9-pool-uyuni
Red Hat 8	rhel8-pool-uyuni
Red Hat 7	rhel7-pool-uyuni

*Procedure: Creating Custom Channels*

1. On the Uyuni Server Web UI, navigate to **Software > Manage > Channels**.
2. Click **Create Channel** and set the appropriate parameters for the channels.
3. In the **Parent Channel** field, select the appropriate base channel.
4. Click **Create Channel**.
5. Repeat for all channels you need to create. There should be one custom channel for each custom repository.

You can check that you have created all the appropriate channels and repositories, by navigating to **Software > Channel List > All**.



- For Red Hat 8 clients, add both the Base and AppStream channels. You require packages from both channels. If you do not add both channels, you cannot create the bootstrap repository, due to missing packages.

If you are using modular channels, you must enable the Python 3.6 module stream on the client. If you do not provide Python 3.6, the installation of the **spacecmd** package will fail.

When you have created all the channels, you can associate them with the repositories you created:

**Procedure: Associating Channels with Repositories**

1. On the Uyuni Server Web UI, navigate to **Software > Manage > Channels**, and click the channel to associate.
2. Navigate to the **Repositories** tab, and check the repository to associate with this channel.
3. Click **Update Repositories** to associate the channel and the repository.
4. Repeat for all channels and repositories you need to associate.
5. OPTIONAL: Navigate to the **Sync** tab to set a recurring schedule for synchronization of this repository.
6. Click **Sync Now** to begin synchronization immediately.

**4.12.1.4. Check Synchronization Status****Procedure: Checking Synchronization Progress from the Web UI**

1. In the Uyuni Web UI, navigate to **Software > Manage > Channels**, then click the channel associated to the repository.
2. Navigate to the **Repositories** tab, then click **Sync** and check **Sync Status**.

**Procedure: Checking Synchronization Progress from the Command Prompt**

1. At the command prompt on the Uyuni Server, as root, use the **tail** command to check the synchronization log file:

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

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



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

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

1. On the Uyuni Server Web UI, navigate to **Configuration > Channels**.
2. Click **Create State Channel**.
  - In the **Name** field, type **subscription-manager: disable yum plugins**.
  - In the **Label** field, type **subscription-manager-disable-yum-plugins**.
  - In the **Description** field, type **subscription-manager: disable yum plugins**.
  - In the **SLS Contents** field, leave it empty.
3. Click **Create Config Channel**
4. Click **Create Configuration File**

- In the **Filename/Path** field type `/etc/yum/pluginconf.d/subscription-manager.conf`.
- In the **File Contents** field type:

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

5. Click **Create Configuration File**
6. Take note of the value of the field **Salt Filesystem Path**.
7. Click on the name of the Configuration Channel.
8. Click on **View/Edit 'init.sls' File**

- In the **File Contents** field, type:

```
configure_subscription-manager-disable-yum-plugins:
cmd.run:
  - name: subscription-manager config --rhsm.auto_enable_yum_plugins=0
  - watch:
    - file: /etc/yum/pluginconf.d/subscription-manager.conf
file.managed:
  - name: /etc/yum/pluginconf.d/subscription-manager.conf
  - source: salt://etc/yum/pluginconf.d/subscription-manager.conf
```

9. Click **Update Configuration File**.



The **Creating a Salt State to Deploy Configuration Files** procedure is optional.

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

1. On the Uyuni Server Web UI, navigate to **Systems > System Groups**.
2. Click **Create Group**.
  - In the **Name** field, type `rhel-systems`.
  - In the **Description** field, type `All RHEL systems`.
3. Click **Create Group**.
4. Click **States** tab.
5. Click **Configuration Channels** tab.
6. Type `subscription-manager: disable yum plugins` at the search box.
7. Click **Search** to see the state.
8. Click the checkbox for the state at the **Assign** column.
9. Click **Save changes**.

## 10. Click **Confirm**.

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

### *Procedure: Adding the System Group to Activation Keys*

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

1. On the Uyuni Server Web UI, navigate to **Systems > Activation Keys**.
2. For each the Activation Keys you used for RHEL systems, click on it and:
3. Navigate to the **Groups** tab, and the **Join** subtab.
4. Check **Select rhel-systems**.
5. Click **Join Selected Groups**.

### 4.12.1.5. Manage GPG Keys

Clients use GPG keys to check the authenticity of software packages before they are installed. Only trusted software can be installed on clients.



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

For more information about GPG keys, see **Client-configuration > Gpg-keys**.



For the Red Hat custom channels, if you want to check the **Enable GPG Check** field, you need to populate the **GPG Key URL** field. You can use the file URL of the GPG key stored in the directory **/etc/pki/rpm-gpg** of the Red Hat minion.

Example: `file:///etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release`

For the complete list of the Red Hat product signing keys, see <https://access.redhat.com/security/team/key>.

### 4.12.1.6. Register Clients

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

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

For more information on registering your clients, see [Client-configuration > Registration-overview](#).

## 4.12.2. Registering Red Hat Enterprise Linux clients with RHUI

i

This section contains information about using Red Hat update infrastructure (RHUI) to register clients running Red Hat Enterprise Linux operating systems.

If you are running clients in a public cloud, such as Amazon EC2, use this method.

It is possible to use RHUI in conjunction with the Red Hat content delivery network (CDN) to manage your Red Hat Enterprise Linux subscriptions. For information about using Red Hat CDN, see [Client-configuration > Clients-rh-cdn](#).



You are responsible for connecting Uyuni Server to the Red Hat update infrastructure. All clients that get updates using this RHUI certificate need to be correctly licensed, please check with your cloud provider and the Red Hat terms of service for more information.



When Red Hat Enterprise Linux clients registered with RHUI are switched off, Red Hat might declare the certificate invalid. In this case, you need to turn the client on again, or get a new RHUI certificate.

### 4.12.2.1. Import entitlements and certificates

In the past it was required to import the certificates and entitlement data manual into Uyuni Server. This task has been automated by using the same mechanism as for SUSE PAYG instances. For more information, see [Installation-and-upgrade > Connect-payg](#).

This guide covers clients registered to Red Hat update infrastructure (RHUI). You must have at least one system registered to RHUI, with an authorized subscription for repository content.

For information about using Red Hat content delivery network (CDN) instead, see [Client-configuration > Clients-rh-cdn](#).



Satellite certificates for client systems require a Satellite server and subscription. Clients using Satellite certificates are not supported with Uyuni Server.



The PAYG connection regular checks with the client to get the latest authentication data. It is important that the client stays running and is regularly updated. If this does not happen, repository synchronization will fail with authentication errors at some point in time.



Update any Red Hat 7 instance before connecting it.



A Red Hat 9 instance needs to be configured with the crypto policy **LEGACY** to

- be able to connect it. Execute `sudo update-crypto-policies --set LEGACY` to configure it accordingly.

#### 4.12.2.2. Connecting to Red Hat update infrastructure

*Procedure: Connecting new Red Hat instance*

1. In the Uyuni Web UI, navigate to **Admin > Setup Wizard > PAYG**, and click **Add PAYG**.
2. Start with the page section **PAYG connection Description**.
3. In the **Description** field, add the description.
4. Move to the page section **Instance SSH connection data**.
5. In the **Host** field, enter the instance DNS or IP address to connect from Uyuni.
6. In the **SSH Port** field, enter the port number or use default value 22.
7. In the **User** field, enter the username as specified in the cloud.
8. In the **Password** field, enter the password.
9. In the **SSH Private Key** field, enter the instance key.
10. In the **SSH Private Key Passphrase** field, enter the key passphrase.



- Authentication keys must always be in PEM format.

If you are not connecting directly to the instance, but via SSH bastion, proceed with [Procedure: Adding SSH bastion connection data](#).

Otherwise, continue with [Procedure: Finishing Red Hat connecting](#).

*Procedure: Adding SSH bastion connection data*

1. Navigate to the page section **Bastion SSH connection data**.
2. In the **Host** field, enter the bastion hostname.
3. In the **SSH Port** field, enter the bastion port number.
4. In the **User** field, enter the bastion username.
5. In the **Password** field, enter the bastion password.
6. In the **SSH Private Key** field, enter the bastion key.
7. In the **SSH Private Key Passphrase** field, enter the bastion key passphrase.

Complete the setup with [Procedure: Finishing Red Hat connecting](#).

*Procedure: Finishing Red Hat connecting*

1. To complete adding new Red Hat connection data, click **Create**.
2. Return to PAYG connection data **Details** page. The updated connection status is displayed on the

top section named **Information**.

3. Connection status is shown in **Admin > Setup Wizard > Pay-as-you-go** screen, too.
4. If the authentication data for the instance is correct, the column **Status** shows **Credentials successfully updated**.



If invalid data is entered at any point, the newly created instance is shown in **Admin > Setup Wizard > PAYG**, with column **Status** displaying an error message.

As soon as the authentication data is available on the server, repositories were added for all available repositories on the connected instance. The repositories can be seen in **Software > Manage > Repositories**



A Red Hat connection will create custom repositories which are owned by organization 1 by default. If a different organization should own the autogenerated repositories, configure `java.rhui_default_org_id` in `/etc/rhn/rhn.conf`.

This only defines and updates the repositories. If you want to use a repository for a managed client, you need to specify a Software Channel and connect the repositories to it.

#### 4.12.2.3. Add software channels

Before you register Red Hat clients to your Uyuni Server, you need to add the required software channels, and synchronize them.



In the following section, descriptions often default to the **x86\_64** architecture. Replace it with other architectures if appropriate.

The channels you need for this procedure are:

*Table 37. Red Hat Channels - CLI*

OS Version	Base Channel	Client Channel	Tools Channel
Red Hat 7	rhel7-pool-uyuni	-	rhel7-uyuni-client
Red Hat 8	rhel8-pool-uyuni	-	rhel8-uyuni-client
Red Hat 9	rhel9-pool-uyuni	-	rhel9-uyuni-client

##### *Procedure: Adding Software Channels at the Command Prompt*

1. At the command prompt on the Uyuni Server, as root, use the **spacewalk-common-channels** command to add the appropriate channels. Ensure you specify the correct architecture:

```
spacewalk-common-channels \
-a <architecture> \
```

```
<base_channel_name> \
<child_channel_name_1> \
<child_channel_name_2> \
... <child_channel_name_n>
```

- If [automatic synchronization](#) is turned off, synchronize the channels:

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

- Ensure the synchronization is complete before continuing.



The client tools channel provided by [spacewalk-common-channels](#) is sourced from Uyuni and not from SUSE.



The AppStream repository provides modular packages. This results in the Uyuni Web UI showing incorrect package information. You cannot perform package operations such as installing or upgrading directly from modular repositories using the Web UI or API.

Alternatively, you can use Salt states to manage modular packages, or use the [dnf](#) command on the client. For more information about CLM, see [Administration > Content-lifecycle](#).

#### 4.12.2.4. Prepare custom channels

To mirror the software from RHUI, you need to create custom channels in Uyuni that are linked to autogenerated repositories.

The channels you need for this procedure are:

*Table 38. Red Hat Custom Channels*

OS Version	Base Channel
Red Hat 7	rhel7-pool-uyuni
Red Hat 8	rhel8-pool-uyuni
Red Hat 9	rhel9-pool-uyuni

*Procedure: Creating Custom Channels*

- On the Uyuni Server Web UI, navigate to **Software > Manage > Channels**.
- Click **Create Channel** and set the appropriate parameters for the channels.
- In the **Parent Channel** field, select the appropriate base channel.
- Click **Create Channel**.
- Repeat for all channels you need to create. There should be one custom channel for each custom

repository.

You can check that you have created all the appropriate channels and repositories, by navigating to **Software > Channel List > All**.



- For Red Hat 8 clients, add both the Base and AppStream channels. You require packages from both channels. If you do not add both channels, you cannot create the bootstrap repository, due to missing packages.

When you have created all the channels, you can associate them with the repositories you created:

*Procedure: Associating Channels with Repositories*

1. On the Uyuni Server Web UI, navigate to **Software > Manage > Channels**, and click the channel to associate.
2. Navigate to the **Repositories** tab, and check the repository to associate with this channel.
3. Click **Update Repositories** to associate the channel and the repository.
4. Repeat for all channels and repositories you need to associate.
5. OPTIONAL: Navigate to the **Sync** tab to set a recurring schedule for synchronization of this repository.
6. Click **Sync Now** to begin synchronization immediately.

#### 4.12.2.5. Check synchronization status

*Procedure: Checking Synchronization Progress from the Web UI*

1. In the Uyuni Web UI, navigate to **Software > Manage > Channels**, then click the channel associated to the repository.
2. Navigate to the **Repositories** tab, then click **Sync** and check **Sync Status**.

*Procedure: Checking Synchronization Progress from the Command Prompt*

1. At the command prompt on the Uyuni Server, as root, use the **tail** command to check the synchronization log file:

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

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



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

#### 4.12.2.6. Manage GPG keys

Clients use GPG keys to check the authenticity of software packages before they are installed. Only

trusted software can be installed on clients.



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

For more information about GPG keys, see [Client-configuration > Gpg-keys](#).

#### 4.12.2.7. Register clients

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

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

For more information on registering your clients, see [Client-configuration > Registration-overview](#).

## 4.13. Rocky Linux Client Registration

You can register Rocky Linux clients to your Uyuni Server. The method and details vary depending on the operating system of the client.

Before you start, ensure that the client has the date and time synchronized correctly with the Uyuni Server.

You must also have created an activation key. For more information about creating activation keys, see [Client-configuration > Activation-keys](#).

### 4.13.1. Registering Rocky Linux Clients

This section contains information about registering clients running Rocky Linux operating systems.



Registering Rocky Linux clients to Uyuni is tested with the default SELinux configuration of **enforcing** with a **targeted** policy. You do not need to disable SELinux to register Rocky Linux clients to Uyuni.

#### 4.13.1.1. Add Software Channels

Before you can register Rocky Linux clients to your Uyuni Server, you need to add the required software channels, and synchronize them.

The architectures currently supported are: **x86\_64** and **aarch64**, on version 9 additionally ppc64le and s390x. For full list of supported products and architectures, see [Client-configuration > Supported-features](#).



In the following section, descriptions often default to the [x86\\_64](#) architecture. Replace it with other architectures if appropriate.

The channels you need for this procedure are:

*Table 39. Rocky Linux Channels - CLI*

OS Version	Base Channel	Client Channel	AppStream Channel
Rocky Linux 9	rockylinux9	rockylinux9-uyuni-client	rockylinux9-appstream
Rocky Linux 8	rockylinux8	rockylinux8-uyuni-client	rockylinux8-appstream

*Procedure: Adding Software Channels at the Command Prompt*

- At the command prompt on the Uyuni Server, as root, use the [spacewalk-common-channels](#) command to add the appropriate channels. Ensure you specify the correct architecture:

```
spacewalk-common-channels \
-a <architecture> \
<base_channel_name> \
<child_channel_name_1> \
<child_channel_name_2> \
... <child_channel_name_n>
```

- If [automatic synchronization](#) is turned off, synchronize the channels:

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

- Ensure the synchronization is complete before continuing.



The client tools channel provided by [spacewalk-common-channels](#) is sourced from Uyuni and not from SUSE.



For Rocky Linux 8 and Rocky Linux 9 clients, add both the Base and AppStream channels. You require packages from both channels. If you do not add both channels, you cannot create the bootstrap repository, due to missing packages.



You might notice some disparity in the number of packages available in the AppStream channel between upstream and the Uyuni channel. You might also see different numbers if you compare the same channel added at a different point in time. This is due to the way that Rocky Linux manages their repositories. Rocky Linux removes older version of packages when a new version is released, while Uyuni keeps all of them, regardless of age.

If you are using modular channels with Rocky Linux 8, you must enable the Python 3.6 module stream on the client. If you do not provide Python 3.6, the installation of the [spacecmd](#) package will fail.



The AppStream repository provides modular packages. This results in the Uyuni Web UI showing incorrect package information. You cannot perform package operations such as installing or upgrading directly from modular repositories using the Web UI or API.

Alternatively, you can use Salt states to manage modular packages, or use the `dnf` command on the client. For more information about CLM, see [Administration > Content-lifecycle](#).

#### 4.13.1.2. Check Synchronization Status

*Procedure: Checking Synchronization Progress from the Web UI*

1. In the Uyuni Web UI, navigate to **Software > Manage > Channels**, then click the channel associated to the repository.
2. Navigate to the **Repositories** tab, then click **Sync** and check **Sync Status**.

*Procedure: Checking Synchronization Progress from the Command Prompt*

1. At the command prompt on the Uyuni Server, as root, use the `tail` command to check the synchronization log file:

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

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

#### 4.13.1.3. Create an Activation Key

You need to create an activation key that is associated with your Rocky Linux channels.

For more information on activation keys, see [Client-configuration > Activation-keys](#).

#### 4.13.1.4. Manage GPG Keys

Clients use GPG keys to check the authenticity of software packages before they are installed. Only trusted software can be installed on clients.



Trusting a GPG key is important for security on clients. It is the task of the administrator to decide which keys are needed and can be trusted. A software channel cannot be assigned to a client when the GPG key is not trusted.

For more information about GPG keys, see [Client-configuration > Gpg-keys](#).

#### 4.13.1.5. Register Clients

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

bootstrap repository from the command prompt, using this command:

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

For more information on registering your clients, see [Client-configuration > Registration-overview](#).

#### 4.13.1.6. Manage Errata

When you update Rocky Linux clients, the packages include metadata about the updates.

## 4.14. Ubuntu Client Registration

You can register Ubuntu clients to your Uyuni Server. The method and details varies depending on the operating system of the client.

Before you start, ensure that the client has the date and time synchronized correctly with the Uyuni Server.

You must also have created an activation key. For more information about creating activation keys, see [Client-configuration > Activation-keys](#).

### 4.14.1. Registering Ubuntu 20.04 and 22.04 Clients

This section contains information about registering clients running Ubuntu 20.04 LTS and 22.04 LTS operating systems.

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



Canonical does not endorse or support Uyuni.

#### 4.14.1.1. Add Software Channels

Before you register Ubuntu clients to your Uyuni Server, you need to add the required software channels, and synchronize them.



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

The channels you need for this procedure are:

*Table 40. Ubuntu Channels - CLI*

OS Version	Base Channel	Main Channel	Updates Channel	Security Channel	Client Channel
Ubuntu 22.04	ubuntu-2204-pool-amd64-uyuni	ubuntu-2204-amd64-main-uyuni	ubuntu-2204-amd64-main-updates-uyuni	ubuntu-2204-amd64-main-security-uyuni	ubuntu-2204-amd64-uyuni-client
Ubuntu 20.04	ubuntu-2004-pool-amd64-uyuni	ubuntu-2004-amd64-main-uyuni	ubuntu-2004-amd64-main-updates-uyuni	ubuntu-2004-amd64-main-security-uyuni	ubuntu-2004-amd64-uyuni-client

Version 20.04 also requires the Universe channels:

*Table 41. Ubuntu 20.04 Universe Channels - CLI*

Ubuntu 20.04	
Universe Channel	ubuntu-2004-amd64-universe-uyuni
Universe Updates Channel	ubuntu-2004-amd64-universe-updates-uyuni
Universe Security Updates Channel	ubuntu-2004-amd64-universe-security-uyuni
Universe Backports Channel	ubuntu-2004-amd64-universe-backports-uyuni

*Procedure: Adding Software Channels at the Command Prompt*

- At the command prompt on the Uyuni Server, as root, use the **spacewalk-common-channels** command to add the appropriate channels:

```
spacewalk-common-channels \
<base_channel_label> \
<child_channel_label_1> \
<child_channel_label_2> \
... <child_channel_label_n>
```

- If **automatic synchronization** is turned off, synchronize the channels:

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

- Ensure the synchronization is complete before continuing.



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

#### 4.14.1.2. Check Synchronization Status

*Procedure: Checking Synchronization Progress from the Web UI*

- In the Uyuni Web UI, navigate to **Software > Manage > Channels**, then click the channel associated

to the repository.

2. Navigate to the **Repositories** tab, then click **Sync** and check **Sync Status**.

*Procedure: Checking Synchronization Progress from the Command Prompt*

1. At the command prompt on the Uyuni Server, as root, use the **tail** command to check the synchronization log file:

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

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



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

#### 4.14.1.3. Manage GPG Keys

Clients use GPG keys to check the authenticity of software packages before they are installed. Only trusted software can be installed on clients.



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

For more information about GPG keys, see **Client-configuration > Gpg-keys**.

#### 4.14.1.4. Root Access

The root user on Ubuntu is disabled by default for SSH access.

To be able to onboard using a regular user, you need to edit the **sudoers** file.



- This issue happens with self-installed versions of Ubuntu. If the default user has been granted administrative privileges during installation time, a password is required to perform privilege escalation using **sudo**. With cloud instances this does not happen because **cloud-init** automatically creates a file under **/etc/sudoers.d** and grants privilege escalation through **sudo** without the need for a password.

*Procedure: Granting Root User Access*

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

```
sudo visudo
```

Grant `sudo` access to the user by adding this line at the end of the `sudoers` file. Replace `<user>` with the name of the user that is bootstrapping the client in the Web UI:

```
<user> ALL=NOPASSWD: /usr/bin/python, /usr/bin/python2, /usr/bin/python3,  
/var/tmp/venv-salt-minion/bin/python
```



This procedure grants root access without requiring a password, which is required for registering the client. When the client is successfully installed it runs with root privileges, so the access is no longer required. We recommend that you remove the line from the `sudoers` file after the client has been successfully installed.

#### 4.14.1.5. Register Clients

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

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

For more information on registering your clients, see [Client-configuration > Registration-overview](#).

## 4.15. Register Clients to a Proxy

Proxy servers can act as a broker and package cache for clients. Registering clients to a proxy is similar to registering them directly to the Uyuni Server, with a few key differences.

These sections contain information on registering clients to a proxy using the Web UI, commands on the command line, or a bootstrap script. There are also procedures how you can move clients from one Uyuni Proxy to another one or to the Uyuni Server.

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

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

### 4.15.1. Move Clients between Proxies

You can move Salt and Salt SSH push clients between proxies without the need to repeat the registration process.



You cannot move chained proxies. Instead of moving a chained proxy, create a new proxy, move the clients, and delete the old proxy.

**Procedure: Moving Salt or Salt SSH Push Client between Proxies**

1. In the Uyuni Web UI, navigate to the **System Details** page for the client you want to move between proxies.
2. Navigate to the **Connection** tab. Then follow the **Change proxy** link to see the drop-down menu.
3. From the **New Proxy** drop-down menu select the proxy you want the client to move to, and click **Change Proxy**.

**Procedure: Moving Multiple Salt or Salt SSH Push Clients between Proxies with SSM**

1. In the Uyuni Web UI, navigate to **Systems > System List** and check each client to move, this adds the clients to the system set manager.
2. Navigate to **Systems > System Set Manager**, and go to the **Misc > Proxy``** tab.
3. From the **New Proxy** drop-down menu select the proxy you want the clients to move to, and click **Change Proxy**.

The same functionality is also available with the `system.changeProxy` API call.

**4.15.1.1. Background Information**

The effect of this function differs between normal Salt clients and Salt SSH push clients.

**4.15.1.1.1. Normal Salt Clients**

The function schedules a Salt state action, which modifies `master:` setting in the `susemanager.conf` Salt client configuration file to point to the new proxy. Then the function restarts the Salt client.



Changing `master:` by manually editing the `susemanager.conf` file has the same effect and is supported, too.

When the client restarts and reconnects via the new proxy, the server updates the proxy path in the database and schedules another action for refreshing the channel URLs.

**4.15.1.1.2. Salt SSH Push Clients**

The function updates the proxy path immediately in the database and new action for refreshing the channel URLs is scheduled.

**4.15.2. Move Clients from Proxies to the Server**

If you want to move a Salt client from a proxy to the server, select **None** from proxy list.

**4.15.3. Register Clients to a Proxy with the Web UI**

You can register Salt clients to the Uyuni Proxy using the Web UI.

A bootstrap repository is needed for non-SLE clients in general and for SLE clients before version 15. A

bootstrap repository offers packages for installing Salt on clients and for registering Salt or traditional clients.

For information about creating a bootstrap repository, see [Client-configuration > Bootstrap-repository](#).

*Procedure: Registering Clients to a Proxy with the Web UI*

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

When the bootstrap process has completed, your client is listed at **Systems > System List**.

#### 4.15.3.1. Register on the Command Line

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

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

1. Choose a client configuration file located at:

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

or:

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

This is sometimes also called a minion file.

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

```
master: PROXY123.EXAMPLE.COM
```

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

```
systemctl restart salt-minion
```

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

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

#### 4.15.4. Registering with a Bootstrap Script (Salt and Traditional)

You can register clients through the Uyuni Proxy with a bootstrap script. This is done almost the same way as registering clients directly with the Uyuni Server. The difference is that you create the bootstrap script on the Uyuni Proxy with a command line tool. The bootstrap script then deploys all necessary information to the clients. The bootstrap script requires some parameters such as activation keys or GPG keys. These parameters depend on your specific setup.

*Procedure: Registering Clients to a Proxy with a Bootstrap Script*

1. Create a client activation key on the Uyuni server using the Web UI. For more information, see [Client-configuration > Activation-keys](#).
2. On the proxy, execute the **mgr-bootstrap** command line tool as root. If needed, use the additional command line switches to tune your bootstrap script.

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

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

3. OPTIONAL: Edit the resulting bootstrap script.
4. Execute the bootstrap script directly on the clients or from the proxy with **ssh**. Replace **<bootstrap>** with the name of the bootstrap script and **<client.example.com>** with the host name of your client:

```
cat <bootstrap> | ssh root@<client.example.com> /bin/bash
```

## 4.16. Registering clients on a public cloud

When you have your Uyuni Server set up, you are ready to start registering clients.

### 4.16.1. Add Products and Synchronize Repositories

Ensure you have already added the corresponding products for your clients and synced the repositories to Uyuni. This is required to create the bootstrap repositories used for registering clients.

For more information, see [installation-and-upgrade:pubcloud-setup.pdf](#).

### 4.16.2. Prepare on-demand images

An instance started from an on-demand image provided by SUSE is automatically registered, and the update infrastructure and SUSE Linux Enterprise modules are activated. To use your on-demand image as a Uyuni client, you need to disable this automation before you begin.

*Procedure: Preparing on-demand images*

1. Log in to the on-demand instance.
2. At the command prompt, as root, remove the registration data and repositories:

```
registercloudguest --clean
```

3. Remove the **cloud-regionsrv-client** package:

```
zypper rm cloud-regionsrv-client
```

4. Remove additional packages specific to your cloud provider:

- Amazon EC2:

```
zypper rm regionServiceClientConfigEC2 regionServiceCertsEC2
```

- Google Compute Engine:

```
zypper rm cloud-regionsrv-client-plugin-gce regionServiceClientConfigGCE  
regionServiceCertsGCE
```

- Microsoft Azure:

```
zypper rm regionServiceClientConfigAzure regionServiceCertsAzure
```

For instructions on registering Uyuni to SUSE Customer Center, see [Installation-and-upgrade > Server-setup](#).

### 4.16.3. Register clients

In the Uyuni Web UI, navigate to **Systems > Bootstrapping**, then fill in the **Host**, **SSH Port**, **User**, and **Password** fields. Make sure you use stable FQDNs for the **Host** field, or Uyuni cannot find your host when your Public Cloud gives you a different short-lived FQDNS.



If you are attempting to bootstrap traditional clients, check that you can resolve the host name of the server while you are logged in to the client. You might need to add the FQDN of the server to `/etc/hosts` local resolution file on the client. Check using the `hostname -f` command with the local IP address of the server.

Public cloud images usually do not allow SSH login with username and password, but only SSH with a certificate. If you want to use bootstrap from the Web UI, you need to enable SSH login with username and SSH key. You can do this by navigating to **Systems > Bootstrapping** and changing the authentication method.

If your cloud provider is Microsoft Azure, you can log in with username and password. To do this, you need to allow the AzureUser to run commands as root without a password. To do this, open the `/etc/sudoers.d/waagent` file, and add or edit this line:

```
AzureUser ALL=(ALL) NOPASSWD: ALL
```



Allowing the AzureUser to run commands as root without a password carries a security risk. Use this method for testing only. Do not do this for production systems.

When the bootstrap process has completed successfully, your client is listed at **Systems > System List**.

- If you want more control over the process, have to register many clients, or are registering traditional clients, create a bootstrap script. For more information, see [Client-configuration > Registration-bootstrap](#).
- For Salt clients and even more control over the process, executing single commands on the command line can be useful. For more information, see [Client-configuration > Registration-cli](#).
- When registering clients launched from a public cloud image (for example, AWS AMI), you need to do some additional configuration to prevent them from over-writing each other. For more information about registering clones, see [Administration > Troubleshooting](#).

## 4.16.4. Activation keys

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

For more on activation keys, see [Client-configuration > Activation-keys](#).

## 4.16.5. Automatic registration of clients created by Terraform

New clients created by Terraform can be automatically registered to Uyuni. Registration can be achieved in two ways:

- **cloud-init** based registration
- remote execution provisioner based registration

### 4.16.5.1. **cloud-init** based client registration

Registering by leveraging **cloud-init** is the preferred way of automatic registering of the newly created virtual machines. This solution avoids configuring an SSH connection to the host. It can also be used regardless of the tool used for client creation.

User can pass the set of user data when deploying the image with Terraform, to automatically register the machine to Uyuni. **user\_data** is run only once at bootstrap, and only the first time the machine is started.

Before using **cloud-init** to register clients, the user must configure:

- Bootstrap script. For more information, see [Client-configuration > Registration-bootstrap](#).
- Activation keys. For more information, see [Client-configuration > Activation-keys](#).

The following command will download the bootstrap script and register the new machine when it is created. It should be added to the **cloud-init** configuration:

```
curl -s http://hub-server.tf.local/pub/bootstrap/bootstrap-default.sh | bash -s
```



Anytime **user\_data** is updated to change the provisioning, Terraform will destroy and then recreate the machines with a new IP, etc.

For more information about **cloud-init** on AWS, see [https://registry.terraform.io/providers/hashicorp/template/latest/docs/data-sources/cloudinit\\_config](https://registry.terraform.io/providers/hashicorp/template/latest/docs/data-sources/cloudinit_config)#example-usage.

For a **cloud-init** example, see [https://registry.terraform.io/providers/hashicorp/cloudinit/latest/docs/data-sources/cloudinit\\_config#example-usage](https://registry.terraform.io/providers/hashicorp/cloudinit/latest/docs/data-sources/cloudinit_config#example-usage).

### 4.16.5.2. `remote-exec` provisioner based registration

The second solution for automatic registering of the newly created virtual machines uses Terraform's `remote-exec` provisioner.

`remote-exec` provisioner interacts with the newly created machines. It opens an SSH connection and can run commands on that machine.



When using `remote-exec` provisioner to register clients, the user must ensure that the machine running Terraform will have access to the new virtual machine after its creation.

The remaining requirements are the same as when using `cloud-init` based client registration:

- Bootstrap script. For more information, see [Client-configuration > Registration-bootstrap](#).
- Activation keys. For more information, see [Client-configuration > Activation-keys](#).

The following command will download the bootstrap script and register the new machine when it is created. It should be defined as the remote command to run:

```
curl -s http://hub-server.tf.local/pub/bootstrap/bootstrap-default.sh | bash -s
```

For more information about `remote-exec` provisioner, see <https://www.terraform.io/docs/provisioners/remote-exec.html>.

# Chapter 5. Client Upgrades

Clients use the versioning system of their underlying operating system, and require regular upgrades.

For SCC registered clients using SUSE operating systems, you can perform upgrades within the Uyuni Web UI. For supported SUSE Linux Enterprise 15 upgrade paths, see <https://documentation.suse.com/sles/15-SP4/html/SLES-all/cha-upgrade-paths.html>

To upgrade clients running SLE 12 to SLE 15, the upgrade is automated, but you need to do some preparation steps before you begin. For more information, see **Client-configuration > Client-upgrades-major**.

You can also automate client upgrades using the content lifecycle manager. For more information, see **Client-configuration > Client-upgrades-lifecycle**.

For more information about product migration such as service pack upgrades, openSUSE Leap minor version upgrades, or openSUSE Leap to SUSE Linux Enterprise migrations, see **Client-configuration > Client-upgrades-product-migration**.

For more information about upgrading unregistered openSUSE Leap clients, see **Client-configuration > Client-upgrades-uyuni**.

## 5.1. Client - Major version upgrade

Your clients must have the latest available service pack (SP) for the installed operating system, with all the latest updates applied. Before you start, ensure that the system is up to date and all updates have been installed successfully.

The upgrade is controlled by YaST and AutoYaST, it does not use Zypper.

### 5.1.1. Prepare to migrate

Before you can migrate your client from SLE 12 to SLE 15 , you need to:

#### *Procedure*

1. Prepare installation media
2. Create an auto-installable distribution
3. Create an activation key
4. Upload an AutoYaST profile

#### *Procedure: Preparing installation media (for example, SLE 15 SP2)*

1. On the Uyuni Server, create a local directory for the SLE 15 SP2 installation media:

```
mkdir -p /srv/images/sle15sp2
```

2. Download an ISO image with the installation sources, and mount the ISO image on your server:

```
mount -o loop DVD1.iso /mnt/
```

3. Copy everything from the mounted ISO to your local file system:

```
cp -r /mnt/* /srv/images/sle15sp2
```

4. When the copy is complete, unmount the ISO image:

```
umount /mnt
```



This image is the Unified Installer and can be used for multiple autoinstallable distributions.

*Procedure: Creating an autoinstallable distribution*

1. In the Uyuni Web UI, navigate to **Systems > Autoinstallation > Distributions** and click **Create Distribution**.
2. In the **Create Autoinstallable Distribution** section, use these parameters:
  - In the **Distribution Label** section, type a unique name for the distribution. Use only letters, numbers, hyphens, periods, and underscores, and ensure the name is longer than four characters. For example, **sles15sp2-x86\_64**.
  - In the **Tree Path** field, type an absolute path to the installation source. For example, **/srv/images/sle15sp2**.
  - In the **Base Channel** field, select **SLE-Product-SLES15-SP2-Pool for x86\_64**.
  - In the **Installer Generation** field, select **SUSE Linux Enterprise 15**.
  - In the **Kernel Options** field, type any options to be passed to the kernel when booting for the installation. The **install=** parameter and the **self\_update=0 pt.options=self\_update** parameter are added by default.
  - In the **Post Kernel Options** section, type any options to be passed to the kernel when booting the installed system for the first time.
3. Click **Create Autoinstallable Distribution** to save.

To switch from the old SLE 12 base channel to the new SLE 15 channel, you need an activation key.

*Procedure: Creating an activation key*

1. In the Uyuni Server Web UI, navigate to **Systems > Activation Keys** and click **Create Key**.
2. Enter a description for your key.

3. Enter a key or leave it blank to generate an automatic key.
4. OPTIONAL: If you want to limit the usage, enter your value in the **Usage** text field.
5. Select the **SLE-Product-SLES15-SP2-Pool for x86\_64** base channel.
6. OPTIONAL: Select any **Add-On System Types**. For more information, see <https://documentation.suse.com/sles/15-SP4/html/SLES-all/article-modules.html>.
7. Click **Create Activation Key**.
8. Click the **Child Channels** tab and select the required channels.
9. Click **Update Key**.

### 5.1.2. Create an autoinstallation profile

Autoinstallation profiles contain all the installation and configuration data needed to install a system. They can also contain scripts to be executed after the installation is complete. For example scripts that you can use as a starting point, see <https://github.com/SUSE/manager-build-profiles/tree/master/AutoYaST>.

For valid AutoYaST upgrade settings, see <https://doc.opensuse.org/projects/autoyast/#CreateProfile-upgrade>.

*Procedure: Creating an autoinstallation profile*

1. In the Uyuni Web UI, navigate to **Systems > Autoinstallation > Profiles** and upload your autoinstallation profile script.

For example scripts that you can use as a starting point, see

<https://github.com/SUSE/manager-build-profiles/tree/master/AutoYaST>.

2. In the **Kernel Options** field, type **autoupgrade=1**.

Optionally, you can also include the **Y2DEBUG=1** option. The debug setting is not required but can help with investigating any future problems you might encounter.



Clients running in Azure cloud must add **textmode=1 console=ttyS0** to **Kernel Options**.

3. Paste the autoinstallation profile or use the file upload field.
4. Click **Create** to save.
5. When the uploaded profile requires variables to be set, navigate to **Systems > Autoinstallation > Profiles**, select the profile to edit, and navigate to the **Variables** tab.

Specify the required variables, using this format:

```
<key>=<value>
```

### 5.1.3. Migration

Before you begin, check that all the channels referenced in the autoinstallation profile are available and fully synchronized.

You can monitor the mirroring progress in `/var/log/rhn/reposync/<channel-label>.log`.

*Procedure: Migrating*

1. In the Uyuni Server Web UI, navigate to **Systems** and select the client to be upgraded.
2. Navigate to the **Provisioning** tab, and select the autoinstallation profile you uploaded.
3. Click **Schedule Autoinstallation and Finish**. The system downloads the required files, change the bootloader entries, reboot, and start the upgrade.

Next time the client synchronizes with the Uyuni Server, it receives a re-installation job. The re-installation job fetches the new kernel and initrd packages. It also writes a new `/boot/grub/menu.lst` (GRUB Legacy) or `/boot/grub2/grub.cfg` (GRUB 2), containing pointers to the new kernel and initrd packages.

When the client next boots, it uses grub to boot the new kernel with its initrd. PXE booting is not used during this process.

Approximately three minutes after the job was fetched, the client goes down for reboot.



For clients, use the `spacewalk/minion_script` snippet to register the client again after migration has completed.

## 5.2. Upgrade Using the Content Lifecycle Manager

When you have many SUSE Linux Enterprise Server clients to manage, you can automate in-place upgrades using the content lifecycle manager.

### 5.2.1. Prepare to Upgrade

Before you can upgrade your clients, you need to make these preparations:

- Create a content lifecycle project
- Create an activation key
- Create an autoinstallable distribution
- Create an autoinstallation profile

*Procedure: Creating a Content Lifecycle Project*

1. Create a content lifecycle project for your distribution.

For more information, see **Administration > Content-lifecycle**.

2. Ensure you choose a short but descriptive name for your project.
3. Include all source channel modules that you require for your distribution.
4. Add filters as required, and set up at least one environment.

*Procedure: Creating an Activation Key*

1. Create an activation key for your distribution.

For more information, see **Client-configuration > Activation-keys**.

2. Ensure your activation key includes all filtered project channels.

*Procedure: Creating an Autoinstallable Distribution*

1. Create an autoinstallable distribution for every base channel you want to migrate.

For more information, see **Client-configuration > Autoinst-distributions**.

2. Give your distribution a label that references the name of the content lifecycle project.
3. In the **Installer Generation** field, select the SLES version you are using.

*Procedure: Creating an Autoinstallation Profile*

1. Create an autoinstallation profile for every target distribution and service pack you want to upgrade to.

For more information, see **Client-configuration > Autoinst-profiles**.

2. You can use variables in the profile to distinguish between the different lifecycle environments.

For example autoinstallation profiles, see <https://github.com/SUSE/manager-build-profiles/tree/master/AutoYaST>.

Use these variables in your autoinstallation profiles for automating in-place upgrades:

*Listing 1. Example: Variables for Use in Autoinstallation Profiles*

```
registration_key=1-15sp1-demo-test
org=1
channel_prefix=15sp1-demo-test
distro_label=15sp1-demo-test
```

*Listing 2. Example: Entry for Use in Autoinstallation Profiles*

```
<listentry>
<ask_on_error config:type="boolean">true</ask_on_error>
<media_url>https://$redhat_management_server/ks/dist/child/$channel_prefix-sle-module-
web-scripting15-sp1-pool-x86_64/$distro_label</media_url>
<name>$channel_prefix SLE-Module-Web-Scripting15-SP1 Pool for x86_64 </name>
<product>Web Scripting Module 15 SP1 x86_64 Pool</product>
</listentry>
```

## 5.2.2. Upgrade

When you have prepared the server for the upgrade, you can provision the clients.

*Procedure: Provisioning the Clients*

1. In the Uyuni Web UI, navigate to **Systems > System List**, and select the clients you want to provision to add them to the system set manager.
2. Navigate to **Systems > System Set Manager > Overview** and click the **Provisioning** tab.
3. Select the autoinstallation profile to use.

For clients that are able to use PXE, the migration is automated as soon as you have provisioned them. For all other clients, you can use Cobbler to perform the upgrade.

*Procedure: Using Cobbler to Upgrade Clients*

1. At the command prompt, as root, check the available Cobbler profiles:

```
cobbler profile list
```

2. Build the ISO file with your chosen profile and distribution:

```
cobbler buildiso --iso=/tmp/SLE_15-sp1.iso --profiles=SLE_15-sp1:1:Example  
--distro=SLE_15-sp1
```

For more information about using CD-ROMs to provision clients, see **Client-configuration > Autoinst-cdrom**.

## 5.3. Product Migration

Product migration allows you to upgrade SLE-based client systems from an Service Pack (SP) level to a later one. For example, you can migrate SUSE Linux Enterprise Server 15 SP1 to SUSE Linux Enterprise Server 15 SP2.

Product migration is for upgrading within the same major version. You cannot use product migration to migrate from SUSE Linux Enterprise Server 12 to SUSE Linux Enterprise Server 15. For more information about major upgrades, see **Client-configuration > Client-upgrades-major**.

You can also migrate openSUSE Leap to a later minor version or to the corresponding SUSE Linux Enterprise Server SP level, for example:

- openSUSE Leap 15.1 to 15.2
- openSUSE Leap 15.1 to SUSE Linux Enterprise Server 15 SP1
- openSUSE Leap 15.5 to SUSE Linux Enterprise Server 15 SP5

In SUSE Linux Enterprise Server 12 and later, SUSE supports service pack skipping if SUSE Customer

Center provides it. For example, you can upgrade from SUSE Linux Enterprise Server 15 to SP2, without installing SP1.

For supported SUSE Linux Enterprise Server upgrade paths, see <https://documentation.suse.com/sles/15-SP4/html/SLES-all/cha-upgrade-paths.html#sec-upgrade-paths-supported>.



- During migration, Uyuni automatically accepts any required licenses (EULAs) before installation.

### 5.3.1. Single System Migration

Before starting the product migration:

- Ensure there are no pending updates or patches. Check the **System Status** on the client system's **Details > Overview** page, and install all offered updates or patches. If your client system is not up to date, product migration may fail.
- Make sure all the channels of the target product are fully synchronized. To check the synchronization status in the Web UI, navigate to the **Admin > Setup Wizard > Products** page.
- Ensure you have a working system backup available, in case of an emergency. Product migration does not have a rollback feature. When the migration procedure has begun, rolling back is not possible.

*Procedure: Performing a Single System Migration*

1. From the **Systems > Overview** page, select a client.
2. From the system details page of the client, navigate to the **Software > Product Migration** tab.
3. Select the target migration path and click **Select Channels**.
4. From the **Product Migration - Channels** page select the correct base channel, including **Mandatory Child Channels** and any additional **Optional Child Channels**.
5. OPTIONAL: Check **Allow Vendor Change** to allow packages that have changed vendors to be installed. If this occurs, a notification is shown with details before the migration is started.



- To migrate openSUSE Leap to SUSE Linux Enterprise Server, you must check the **Allow Vendor Change** option.

6. Click **Schedule Migration** when your channels have been configured properly.

### 5.3.2. Product Mass Migration

If you want to migrate a large number of clients to the next SP version, you can use Uyuni API calls.

The **spacecmd** commandline tool provides a **system\_scheduleproductmigration** sub command, which can be used to schedule a migration for a large number of clients to the next minor version.

#### 5.3.2.1. Perform a Product Mass Migration



The product mass migration operation is dangerous and the process should be tested thoroughly. At least, do a dry-run first.

Be careful not to upgrade systems unintentionally.

#### *Procedure: Performing a Product Mass Migration*

1. List available migration targets, and take note of the system IDs you want to migrate:

```
spacecmd api -- system.listMigrationTargets -A 1000010001
```

2. For each system ID, call **listMigrationTarget** and check that the desired target product is available.
  - If the system ID has an available target, call **system.scheduleProductMigration**.
  - If the desired target is not available, skip the system.
3. Adapt this template for your environment:

```
target = '....'
basechannel = 'channel-label'
system_ids = [1, 2, 3]

session = auth.login(user, pass)
for system in system_ids
    if system.listMigrationTargets(session, system).ident == target
        system.scheduleProductMigration(session, system, target, basechannel, [], False,
<now>)
    else
        print "Cannot migrate to requested target -- skipping system"
    endif
endfor
```

#### 5.3.2.2. Example: SLES 15 SP2 to SLES 15 SP3

For this example, a group will be created temporarily to facilitate the mass migration.

#### *Procedure: Creating a Mass Product Migration Group*

1. In the Uyuni Web UI, navigate to **Systems > System Groups**, and click **Create Group**.
2. Name the group **mppm-target-sles15sp3**.
  - Only systems subscribed to the same base channel should be added to the created group. In the example, only systems subscribed to **SLE-Product-SLES15-SP2-Pool for x86\_64** should be added to the group.

For more information about adding clients to groups, see [client-configuration:system-groups.pdf](#).

#### *Procedure: Adding Systems to the Group*

1. Get the targets for all the systems in the group by running:

```
spacecmd -- system_listmigrationtargets group:mpm-target-sles15sp3
```

2. The command output a string of "IDs."
  - Only select a target, which is reported for **all** systems.
  - The string is the identifier for the **MIGRATIONTARGET** of the other command.



- The **spacecmd** sub-commands **system\_scheduleproductmigration** and **system\_listmigrationtargets** are looping over all systems that are part of the group.
- If there are 100 systems in the group, you will see 100 actions scheduled.
- All systems in the group must support the same migration target.

#### *Procedure: Running the Mass Migration Command*

1. The syntax for the **system\_scheduleproductmigration** command is as follows:

```
spacecmd -- system_scheduleproductmigration <SYSTEM> <BASE_CHANNEL_LABEL> \
<MIGRATION_TARGET> [options]
```

2. For this example to upgrade all systems in the group **mpm-target-sles15sp3** from SLES 12 SP2 to SLES 15 SP, enter on the command line:

```
spacecmd -- system_scheduleproductmigration group:mpm-target-sles15sp3 \
sle-product-sles15-sp3-pool-x86_64 "[190,203,195,1242]" -d
```

#### **5.3.2.2.1. Mandatory Syntax Explained**

To see syntax usage and options for **system\_scheduleproductmigration**, run:

```
spacecmd system_scheduleproductmigration help
```

#### **<SYSTEM>**

For this example we will use the group we created to select all of the systems from that group:

```
group:mpm-target-sles15sp3
```

#### **<BASE\_CHANNEL\_LABEL>**

This is the label for the target base channel. In this case, the system is being upgraded to SLES 15 SP3, and the label is **sle-product-sles15-sp3-pool-x86\_64**.

To see a list of all base channels currently mirrored, run:

```
spacecmd softwarechannel_listbasechannels.
```

Keep in mind you cannot upgrade to a channel unless it is an available target for your current base channel.

## <MIGRATION\_TARGET>

To identify this value for systems in the group **group:mpm-target-sles15sp3**, run:

```
spacecmd -- system_listmigrationtargets group:mpm-target-sles15sp3
```

The **MIGRATION\_TARGET** parameter must be passed in the following format; note necessary shell quotation to prevent sideeffects with brackets:

```
"[190,203,195,1242]"
```

## Options

- **-S** START\_TIME
- **-d** pass this flag, if you want to do a dry run (it is recommended to run a dry run before the actual migration)
- **-c** CHILD\_CHANNELS (comma-separated child channels labels with no spaces)

In this case we included the **-d** option, which can be removed after a successful dry run.

If successful, the command output for each scheduled system will look like this:

```
Scheduling Product migration for system mpm-sles152-1
Scheduled action ID: 66
```

You can also track the action, in this case the dry run, in the Web UI for a given system in the group. From the system details page of the client, navigate to **Events > History**. If there are any failures during the dry run, the system should be investigated.

If all is well, the **-d** option can be removed from the command to run the real migration. After the migration is complete, you can reboot the system from the Uyuni Web UI.

## 5.4. Upgrade Uyuni Clients

In this section, we use openSUSE Leap as an example.

### 5.4.1. Prepare to Upgrade

*Procedure: Preparing the Client Upgrade*

- At the command prompt on the Uyuni Server, as root, use the **spacewalk-common-channels** command to add the appropriate channels.

```
spacewalk-common-channels \
opensuse_leap15_4 \
opensuse_leap15_4-non-oss \
opensuse_leap15_4-non-oss-updates \
opensuse_leap15_4-updates \
opensuse_leap15_4-uyuni-client
```

- Fully synchronize all channels with **spacewalk-repo-sync**. In case of already defined repository URLs, continue with [installation-and-upgrade:proxy-uyuni.pdf](#).
- In the Uyuni Server Web UI, navigate to **Software > Manage > Channels** and click the **Uyuni Client Tools for openSUSE Leap 15.4 (x86\_64)** channel name.
- In the upper right corner, click **Manage Channel**.
- Click the **Repositories** tab, and select **External - Uyuni Client Tools for openSUSE Leap 15.3 (x86\_64)**.
- Click **Update Repositories**.
- Navigate to **Repositories > Sync** subtab, and click **Sync Now**.
- Do the same with **openSUSE Leap 15.4 (x86\_64)** and **External - openSUSE Leap 15.3 (x86\_64)**.

Unfold **openSUSE Leap 15.4 (x86\_64)** to see all child channels populated with packages.

### 5.4.2. Upgrade

To upgrade a client you replace the software repositories and update the software, and finally reboot the client.

*Procedure: Upgrading the Client*

- In the Uyuni Server Web UI, navigate to **Systems** and click the name of the client.
- Click **Software > Software Channels**, and as the base channel select the openSUSE Leap 15.5 channel that is listed in the **Customs Channels** list.
- In the **Child Channels** pane, select the 15.5 child channels.
- Click **Next**, and **Confirm Software Channel Change** with **Confirm**.
- Click **Software > Packages > Upgrade**, and select all the packages to be updated on the client, and then apply the selection. Click **Upgrade Packages**, check the details, and click **Confirm** to complete the update.
- Reboot the client.

If you need to update many clients, you can create an action chain of this command sequence on the Uyuni Server. You can use the action chain to perform updates on multiple clients at the same time.

# Chapter 6. Client Deletion

If you need to remove a client from your Uyuni Server, you can use the Web UI to delete it. You can also remove a client from the command line.

## 6.1. Delete a Client with the Web UI

*Procedure: Deleting a Client*

1. In the Uyuni Web UI, navigate to **Systems > System List** and select the client to delete.
2. Click **Delete System**.
3. Check the details and click **Delete Profile** to confirm.
4. For Salt clients, Uyuni attempts to clean up additional configuration. If the client cannot be contacted, you are given the option to cancel the deletion, or delete the client without cleaning up the configuration files.

You can also delete multiple clients using the system set manager. For more information about the system set manager, see **Client-configuration > System-set-manager**.



- Cleaning up a Salt client only disables Salt and stops the service if possible. It does not uninstall the package.

## 6.2. Delete a Salt Client on the Command Line (with API Call)

*Procedure: Deleting a Client from the Server*

1. Delete the client with the FQDN (Fully Qualified Domain Name):

```
spacecmd system_delete FQDN
```

**spacecmd system\_delete** also deletes the Salt key.

**system\_delete** offers the following options:

```
usage: system_delete [options] <SYSTEMS>
options:
  -c TYPE - Possible values:
    * 'FAIL_ON_CLEANUP_ERR' - fail in case of cleanup error,
    * 'NO_CLEANUP' - do not cleanup, just delete,
    * 'FORCE_DELETE' - try cleanup first but delete server anyway
      in case of error
```

## 6.3. Delete a Client from the Command Line

### 6.3.1. Salt Client

This process is only for Uyuni clients, do not run it on the Uyuni Server itself.



- You must not execute the following procedure on clients running Red Hat Enterprise Linux, Debian, or clones. Instead of **zypper** use equivalent packager commands such as **yum**, **dnf**, or **apt**.

*Procedure: Deleting SLES 12 and 15 Salt Clients*

1. Stop the salt-minion service:

```
systemctl stop salt-minion
```

2. Remove the repositories and configuration files:

```
rm /etc/zypp/repos.d/susemanager\*:channels.repo  
rm -r /etc/sysconfig/rhn/  
rm -r /etc/salt/
```

3. Remove Client Packages:

```
zypper rm salt salt-minion python*-salt sle-manager-tools-release
```

*Procedure: Salt Bundle Client - Manual Registration Cleanup*

1. To unregister, run:

```
systemctl stop venv-salt-minion  
zypper rm -y venv-salt-minion  
rm /etc/zypp/repos.d/susemanager\*:channels.repo /etc/venv-salt-minion/*  
rm -r /etc/venv-salt-minion/*
```

For information about the Salt bundle, see **Client-configuration > Contact-methods-saltbundle**.

This process is only for Uyuni clients, do not run it on the Uyuni Server itself.

# Chapter 7. Client Operations

In addition to registering, upgrading, installing software, or deleting clients other operations can be performed. Uyuni clients can be managed individually or as a set of clients. A set of clients can grouped with System Set Manager for a one time action or permanently with System Groups. For advanced client configuratin options general Configuration Management is available. You can obtain Custom System Information, power on or off, and reboot clients using the Uyuni Web UI.

The following sections contain detailed descriptions for each of these operations.

## 7.1. Package Management

Clients use packages to install, uninstall, and upgrade software.

To manage packages on a client, navigate to **Systems**, click the client to manage, and navigate to the **Systems > Software > Packages** subtab. The options available in this section vary depending on the type of client you have selected, and its current channel subscriptions.



- When packages are installed or upgraded, licenses or EULAs are automatically accepted.

Most package management actions can be added to action chains. For more about action chains, see **Reference > Schedule**.

### 7.1.1. Verify Packages

You can check that packages you have installed on a client match the current state of the database they were installed from. The metadata of the installed package is compared to information in the database, including the file checksum, file size, permissions, owner, group, and type.

#### *Procedure: Verifying Installed Packages*

1. In the Uyuni Web UI, navigate to **Systems**, click the client the package is installed on, and navigate to the **Systems > Software > Packages > Verify** subtab.
2. Select the packages you want to verify and click **Verify Selected Packages**.
3. When the verification is complete, navigate to **Systems > Events > History** to see the results.

### 7.1.2. Compare Packages

You can compare the packages installed on a client with a stored profile, or with packages installed on another client. When the comparison is made, you can choose to modify the selected client to match.

To compare packages against a profile, you need to have stored a profile. Profiles are created from the packages on a currently installed client. When the profile has been created, you can use it to install more clients with the same packages installed.

*Procedure: Creating a Stored Profile*

1. In the Uyuni Web UI, navigate to **Systems**, click the client to base your profile off, and navigate to the **Systems > Software > Packages > Profiles** subtab.
2. Click **Create System Profile**.
3. Type a name and description for your profile and click **Create Profile**.

*Procedure: Comparing Client Packages*

1. In the Uyuni Web UI, navigate to **Systems**, click the client to compare, and navigate to the **Systems > Software > Packages > Profiles** subtab. To compare with a stored profile, select the profile and click **Compare**.
2. To compare with another client, select the client name and click **Compare** to see a list of differences between the two clients.
3. Check packages you want to install on the selected client, uncheck packages you want to remove, and click **Sync Packages to**.

## 7.2. Patch Management

You can use custom patches within your organization to manage clients. This allows you to issue patch alerts for packages in custom channels, schedule patch installation, and manage patches across organizations.

### 7.2.1. Create Patches

To use a custom patch, you need to create the patch, add packages to it and add it to one or more channels.

*Procedure: Creating a Custom Patch*

1. In the Uyuni Web UI, navigate to **Patches > Manage Patches**, click **Create Patch**.
2. In the **Create Patch** section, use these details:
  - In the **Synopsis** field, type a short description of the patch.
  - In the **Advisory** field, type a label for the patch. We recommend you devise a naming convention for your organization to make patch management easier.
  - In the **Advisory Release** field, enter a release number for your patch. For example, if this is the first version of this patch, use **1**.
  - In the **Advisory Type** field, select the type of patch to use. For example, **Bug Fix Advisory** for a patch that fixes errors.
  - If you selected an advisory type of **Security Advisory**, in the **Advisory Severity** field, select the severity level to use.
  - In the **Product** field, type the name of the product this patch refers to.

- OPTIONAL: In the **Author** field, type the name of the author of the patch.
  - Complete the **Topic**, **Description**, and **Solution** fields with further information about the patch.
3. OPTIONAL: In the **Bugs** section, specify the information of any related bugs, using these details:
- In the **ID** field, enter the bug number.
  - In the **Summary** field, type a short description of the bug.
  - In the **Bugzilla URL** field, type the address of the bug.
  - In the **Keywords** field, type any keywords related to the bug. Use a comma between each keyword.
  - Complete the **References** and **Notes** fields with further information about the bug.
  - Select one or more channels to add the new patch to.
4. Click **Create Patch**.

You can also create patches by cloning an existing one. Cloning preserves package associations and simplifies issuing patches.

*Procedure: Cloning Patches*

1. In the Uyuni Web UI, navigate to **Patches > Clone Patches**.
2. In the **View patches potentially applicable to:** field, select the software channel for the patch you want to clone.
3. Select the patch or patches you want to clone, and click **Clone Patches**.
4. Select one or more channels to add the cloned patch to.
5. Confirm the details to begin the clone.

When you have created a patch, you can assign packages to it.

*Procedure: Assigning Packages to a Patch*

1. In the Uyuni Web UI, navigate to **Patches > Manage Patches**, and click the the advisory name of the patch to see the patch details.
2. Navigate to the **Packages > Add** tab.
3. In the **Channel** field, select the software channel that contains the packages you want to assign to the patch, and click **View Packages**. You can select **All managed packages** to see the available packages in all channels.
4. Check the packages you want to include, and click **Add Packages**.
5. Confirm the details of the packages, and click **Confirm** to assign them to the patch.
6. Navigate to the **Packages > List/Remove** tab to check that the packages have been assigned correctly.

When packages are assigned to a patch, the patch cache is updated to reflect the changes. The cache update might take a couple of minutes.

If you need to change the details of an existing patch, you can do so from the [Patches Management](#) page.

*Procedure: Editing and Deleting Existing Patch Alerts*

1. In the Uyuni Web UI, navigate to **Patches > Manage Patches**.
2. Click the advisory name of the patch to see the patch details.
3. Make the changes as required, and click **Update Patch**.
4. To delete a patch, select the patch in the [Patches Management](#) page, and click **Delete Patches**. Deleting patches might take a few minutes.

## 7.2.2. Apply Patches to Clients

When a patch is ready, you can apply it to clients either singly, or with other patches.

Each package within a patch is part of one or more channels. If the client is not subscribed to the channel, the update is not installed.

If the client has a more recent version of a package already installed, the update is not installed. If the client has an older version of the package installed, the package is upgraded.

*Procedure: Applying All Applicable Patches*

1. In the Uyuni Web UI, navigate to **Systems > Overview** and select the client you want to update.
2. Navigate to the **Software > Patches** tab.
3. Click **Select All** to select all applicable patches.
4. Click **Apply Patches** to update the client.

If you are signed in with Administrator privileges, you can also perform larger batch upgrades for clients.

*Procedure: Applying a Single Patch to Multiple Clients*

1. In the Uyuni Web UI, navigate to **Patches > Patch List**.
2. Locate the patch you want to apply, and click the number under the **Systems** column for that patch.
3. Select the clients you want to apply the patch to, and click **Apply Patches**.
4. Confirm the list of clients to perform the update.

*Procedure: Applying Multiple Patches to Multiple Clients*

1. In the Uyuni Web UI, navigate to **Systems > Overview** and check the clients you want to update to add them to the system set manager.
2. Navigate to **Systems > System Set Manager** and naviagte to the **Patches** tab.

3. Select the patches you want to apply to the clients and click **Apply Patches**.
4. Schedule a date and time for the update to occur, and click **Confirm**.
5. To check the progress of the update, navigate to **Schedule > Pending Actions**.



Scheduled package updates are installed using the contact method configured for each client. For more information, see **Client-configuration > Contact-methods-intro**.

## 7.3. System Locking

System locks are used to prevent actions from occurring on a client. For example, a system lock prevents a client from being updated or restarted. This is useful for clients running production software, or to prevent accidental changes. You can disable the system lock when you are ready to perform actions.

### 7.3.1. System Locks on Salt Clients

When a Salt client is locked, or put into blackout mode, no actions can be scheduled, Salt execution commands are disabled, and a yellow banner is displayed on the **System Details** page. In this mode, actions can be scheduled for the locked client using the Web UI or the API, but the actions fail.



The locking mechanism is not available for Salt SSH clients.

*Procedure: System Locking a Salt Client*

1. In the Uyuni Web UI, navigate to the **System Details** page for the client you want to lock.
2. Navigate to the **Formulas** tab, check the system lock formula, and click **Save**.
3. Navigate to the **Formulas > System Lock** tab, check **Lock system**, and click **Save**. On this page, you can also enable specific Salt modules while the client is locked.
4. When you have made your changes, you might need to apply the highstate. In this case, a banner in the Web UI notifies you. The client remains locked until you remove the system lock formula.

For more information about blackout mode in Salt, see <https://docs.saltstack.com/en/latest/topics/blackout/index.html>.

### 7.3.2. Package Locks

Package locking can be used on several clients, but different feature sets are available. You must differentiate between SUSE Linux Enterprise and openSUSE (zypp-based) versus Red Hat Enterprise Linux or Debian clients.

#### 7.3.2.1. Package Locks on Zypp-based Systems

Package locks are used to prevent unauthorized installation or upgrades to software packages. When a package has been locked, it shows a padlock icon, indicating that it cannot be installed. Any attempt to install a locked package is reported as an error in the event log.

Locked packages cannot be installed, upgraded, or removed, neither through the Uyuni Web UI, nor directly on the client machine using a package manager. Locked packages also indirectly lock any dependent packages.

*Procedure: Using Package Locks*

1. Navigate to the **Software > Packages > Lock** tab on the managed system to see a list of all available packages.
2. Select the packages to lock, and click **Request Lock**. Pick date and time for the lock to activate. By default, the lock is activated as soon as possible. Note that the lock might not activate immediately.
3. To remove a package lock, select the packages to unlock and click **Request Unlock**. Pick date and time as with activating the lock.

#### 7.3.2.2. Package Locks on Red Hat Enterprise Linux- and Debian-like Systems



- Some Red Hat Enterprise Linux- and Debian-like systems have package locking available on Salt clients.

On Red Hat Enterprise Linux- and Debian-like systems, package locks are only used to prevent unauthorized upgrades or removals to software packages. When a package has been locked, it shows a padlock icon, indicating that it cannot be changed. Any attempt to change a locked package is reported as an error in the event log.

Locked packages cannot be upgraded or removed, neither through the Uyuni Web UI, nor directly on the client machine using a package manager. Locked packages also indirectly lock any dependent packages.

*Procedure: Using Package Locks*

1. On the Red Hat Enterprise Linux 7 systems, install the `yum-plugin-versionlock` package as `root`. On the Red Hat Enterprise Linux 8 systems, install the `python3-dnf-plugin-versionlock` package as `root`. On Debian systems, the `apt` tool has the locking feature included.
2. Navigate to the **Software > Packages > Lock** tab on the managed system to see a list of all available packages.
3. Select the packages to lock, and click **Request Lock**. Pick date and time for the lock to activate. By default, the lock is activated as soon as possible. Note that the lock might not activate immediately.
4. To remove a package lock, select the packages to unlock and click **Request Unlock**. Pick date and time as with activating the lock.

## 7.4. Configuration Management

You can use configuration files and channels to manage configuration for your clients, rather than configuring each client manually.

Configuration parameters are scripted and stored in configuration files. You can write configuration files directly using the Uyuni Web UI, or you can upload or link to files that exist in other locations.

Configuration files can be centrally managed. Centrally managed configuration files are provided by global configuration channels and can be applied to any client subscribed to the Uyuni Server.

Configuration channels are used to organize configuration files. You can subscribe clients to configuration channels, and deploy configuration files as required.

Configuration files are version-controlled, so you can add configuration settings, test them on your clients, and roll back to a previous revision as required. When you have created your configuration channels, you can also perform comparisons between various configuration files, and between revisions of the same configuration file.

Centrally managed configuration files are provided by global configuration channels.

The available configuration management features are different for Salt and traditional clients. This table shows the supported features on different client types.

The icons in this table indicate:

- ✓ features are supported by SUSE
- ✗ features are not supported by SUSE
- ? features are under consideration, and may or may not be supported at a later date.

*Table 42. Configuration Management Supported Features*

Feature	Salt	Traditional
Global Configuration Channels	✓	✓
Deploying Files	✓	✓
Comparing Files	?	✓
Locally Managed Files	✓ (with Salt features)	✓
Sandbox Files	✗	✓
Applying the Highstate	✓	✗
File Import from a Client	✗	✓
Jinja Templating	✓	✗
Configuration Macros	✓ with Salt features (grains, pillar data, etc.)	✓

## 7.4.1. Create Configuration Channels

### 7.4.1.1. Central Configuration Channel

To create a new central configuration channel:

**Procedure: Creating Central Configuration Channel**

1. In the Uyuni Web UI, navigate to **Configuration > Channels**, and click **Create Config Channel**.
2. Type a name for the channel.
3. Type a label for the channel. This field must contain only letters, numbers, hyphens (-) and underscores (\_).
4. Type a description for the channel that allows you to distinguish it from other channels.
5. Click **Create Config Channel** to create the new channel.

**7.4.1.2. Salt State Channel**

You can also use a configuration channel to manage Salt states on Salt clients:

**Procedure: Creating a Salt State Channel**

1. In the Uyuni Web UI, navigate to **Configuration > Channels**, and click **Create State Channel**.
2. Type a name for the channel.
3. Type a label for the channel. This field must contain only letters, numbers, hyphens (-) and underscores (\_).
4. Type a description for the channel that allows you to distinguish it from other channels.
5. Type the **SLS Contents** for the **init.sls** file.
6. Click **Create Config Channel** to create the new channel.

**7.4.2. Add Configuration Files, Directories, or Symbolic Links**

When you have created a configuration channel you can add a configuration file, directory, or symbolic link:

**Procedure: Adding a Configuration File, Directory, or Symbolic Link**

1. In the Uyuni Web UI, navigate to **Configuration > Channels**, and click the name of the configuration channel that you want to add a configuration file to, and navigate to the **Add Files > Create File** subtab.
2. In the **File Type** field, choose whether you want to create a text file, directory, or symbolic link.
3. In the **Filename/Path** field, type the absolute path to the location where the file should be deployed.
4. If you are creating a symbolic link, type the target file and path in the **Symbolic Link Target Filename/Path** field.
5. Type the **User name** and **Group name** for the file in the **Ownership** field, and the **File Permissions Mode**.
6. If the client has SELinux enabled, you can configure **SELinux contexts** to enable the required file attributes (for example: user, role, and file type).

7. If the configuration file includes a macro, enter the symbol that marks the beginning and end of the macro.
8. Enter the configuration file contents in the **File Contents** text box, using the script drop-down box to choose the appropriate scripting language.
9. Click **Create Configuration File**.

### 7.4.3. Subscribe Clients to Configuration Channels

You can subscribe individual clients to configuration channels by navigating to **Systems > System List**, selecting the client you want to subscribe, and navigating to the **Configuration** tab. To subscribe multiple clients to a configuration channel, you can use the system set manager (SSM).

*Procedure: Subscribing Multiple Clients to Configuration Channels*

1. In the Uyuni Web UI, navigate to **Systems > Systems List** and select the clients you want to work with.
2. Navigate to **Systems > System Set Manager**, and go to the **Configuration > Subscribe to Channels** subtab to see the list of available configuration channels.
3. OPTIONAL: Click the number in the **Systems currently subscribed** column to see which clients are currently subscribed to the configuration channel.
4. Check the configuration channels you want to subscribe to, and click **Continue**.
5. Rank the configuration channels using the up and down arrows. Where settings conflicts occur between configuration channels, channels closer to the top of the list take precedence.
6. Determine how the channels are applied to the selected clients. Click **Subscribe With Lowest Priority** to add the new channels at a lower priority than currently subscribed channels. Click **Subscribe with Highest Priority** to add the new channels at a higher priority than currently subscribed channels. Click **Replace Existing Subscriptions** to remove existing channels and replace them with the new channels.
7. Click **Apply Subscriptions**.



If new configuration channel priorities conflict with existing channels, the duplicate channel is removed and replaced according to the new priority. If the client's configuration priorities are going to be reordered by an action, the Web UI requires you to confirm the change before proceeding.

### 7.4.4. Compare Configuration Files

You can also use the system set manager (SSM) to compare the configuration file deployed on clients with the configuration file stored on the Uyuni Server.

*Procedure: Comparing Configuration Files*

1. In the Uyuni Web UI, navigate to **Systems > Systems List** and select the clients subscribed to the configuration files you want to compare.

2. Navigate to **Systems > System Set Manager**, and go to the **Configuration > Compare Files** subtab to the list of available configuration files.
3. OPTIONAL: Click the number in the **Systems** column to see which clients are currently subscribed to the configuration file.
4. Check the configuration files to compare, and click **Schedule File Comparison**.

#### 7.4.5. Jinja templating on Salt clients

Jinja templating is possible on Salt clients. Jinja provides variables from pillars or grains. They can be used in configuration files or Salt states.

For more information, see <https://docs.saltproject.io/salt/user-guide/en/latest/topics/jinja.html> with this example:

```
{% if grains.os_family == 'RedHat' %}
  {% set dns_cfg = '/etc/named.conf' %}
{% elif grains.os_family == 'Debian' %}
  {% set dns_cfg = '/etc/bind/named.conf' %}
{% else %}
  {% set dns_cfg = '/etc/named.conf' %}
{% endif %}
dns_conf:
  file.managed:
    - name: {{ dns_cfg }}
    - source: salt://dns/files/named.conf
```

## 7.5. Power Management

You can power on, power off, and reboot clients using the Uyuni Web UI.

This feature uses either the IPMI or Redfish protocol and is managed using a Cobbler profile. The selected client must have a power management controller supporting one of these protocols.

For Redfish, ensure you can establish a valid SSL connection between the client and the Uyuni Server. You must have trusted the certificate authority used to sign the SSL Server Certificate of the Redfish management controller. The CA certificate must be in **.pem** format, and stored on the Uyuni Server at **/etc/pki/trust/anchors/**. When you have saved the certificate, run **update-ca-certificate**.

#### *Procedure: Enabling Power Management*

1. In the Uyuni Web UI, navigate to **Systems > Systems List**, select the client you want to manage, and navigate to the **Provisioning > Power Management** tab.
2. In the **Type** field, select the power management protocol to use.
3. Complete the details for the power management server, and click the appropriate button for the action to take, or click **Save only** to save the details without taking any action.

You can apply power management actions to multiple clients at the same time by adding them to the system set manager. For more information about using the system set manager, see **Client-configuration**

› **System-set-manager.**

### 7.5.1. Power Management and Cobbler

The first time you use a power management feature, a Cobbler system record is automatically created, if one does not yet exist for the client. These automatically created system records are not bootable from the network, and include a reference to a dummy system image. This is needed because Cobbler does not currently support system records without profiles or images.

Cobbler power management uses fence-agent tools to support protocols other than IPMI. Only IPMI and Redfish protocols are supported by Uyuni. You can configure your client to use other protocols by adding the fence-agent names as a comma-separated list to the `java.power_management.types` configuration parameter in the `rhn.conf` configuration files.

## 7.6. Custom System Information

You can include customized system information about your clients. System information is defined as key:value pairs, which can be assigned to clients. For example, you can define a key:value pair for a particular processor, then assign that key to all clients that have that processor installed. Custom system information is categorized, and can be searched using the Uyuni Web UI.

Before you begin, you need to create a key that allows you to store custom information.

*Procedure: Creating a Custom System Information Key*

1. In the Uyuni Web UI, navigate to **Systems > Custom System Info**, and click **Create Key**.
2. In the **Key Label** field, add a name for your key. Do not use spaces. For example, `intel-x86_64-quadcore`.
3. In the **Description** field, provide any additional information required.
4. Repeat for each key you require.

The information is available via Salt pillar. You can retrieve this information with a command such as:

```
salt $minionid pillar.get custom_info:key1
```

This command will result in an output such as:

```
$minionid:
  val1
```

When you have created some custom system information keys, you can apply them to clients.

*Procedure: Applying Custom Information Keys to Clients*

1. In the Uyuni Web UI, navigate to **Systems**, click the client to apply custom information to, and

navigate to the **Details > Custom Info** tab.

2. Click **Create Value**.
3. Locate the value you want to apply, and click the key label.
4. In the **Value** field, provide any additional information.
5. Click **Update Key** to apply the custom information to the client.

For more information about configuration management, see **Client-configuration > Configuration-management**.

## 7.7. System Set Manager

The system set manager (SSM) is used to perform actions on more than one client at a time. SSM creates ephemeral sets of clients, making it useful for one-off actions that you need to apply to a number of clients. If you want more permanent sets, consider using system groups instead. For more information about system groups, see **Client-configuration > System-groups**.

The actions available for use in SSM are listed in this table. The icons in this table indicate:

- ✓ this action is available in SSM for this client type
- ✗ this action is not available in SSM for this client type
- ? this action is under consideration for this client type, and may or may not be supported at a later date.

*Table 43. Available SSM Actions*

Action	Traditional	Salt
List systems	✓	✓
Install patches	✓	✓
Schedule patch updates	✓	✓
Upgrade packages	✓	✓
Install packages	✓	✓
Remove packages	✓	✓
Verify packages	✓	✗
Create groups	✓	✓
Manage groups	✓	✓
Channel memberships	✓	✓
Channel subscriptions	✓	✗

Action	Traditional	Salt
Deploy/diff channels	✓	✗
Autoinstall clients	✓	✗
Tag for snapshot	✓	✗
Remote commands	✓	✗
Power management	✓	✗
Update system preferences	✓	✓
Update hardware profiles	✓	✓
Update package profiles	✓	✓
Set/remove custom values	✓	✓
Reboot clients	✓	✓
Migrate clients to another organization	✓	✓
Delete clients	✓	✓

You can select clients for the SSM in several ways:

- Navigate to **Systems > System List** and check the clients you want to work with.
- Navigate to **Systems > System Groups**, and click **Use in SSM** for the system group you want to work with.
- Navigate to **Systems > System Groups**, check the group you want to work with, and click **Work with Group**.

When you have selected the clients you want to work with, navigate to **Systems > System Set Manager**, or click the **systems selected** icon in the top menu bar.



- The details in SSM might differ slightly from the details in other parts of the Uyuni Web UI. In SSM, all available updates are shown. This allows you to upgrade to packages that might not be the latest version.

### 7.7.1. Change Base Channels in SSM

You can use SSM to change the base channel of more than one client at the same time.



- Changing the base channel significantly changes the packages and patches available to the affected clients. Use with caution.

*Procedure: Using SSM to Change Base Channels for Multiple Clients*

1. In the Uyuni Web UI, navigate to **Systems > System List**, check the clients you want to work with, and navigate to **Systems > System Set Manager**.
2. Navigate to the **Channels** subtab.
3. Locate the current base channel in the list, and verify that the number shown in the **Systems** column is correct. You can click the number in this column to see more details of the clients you are changing.
4. Select the new base channel in the **Desired base Channel** field, and click **Next**.
5. For each child channel, select **No change**, **Subscribe**, or **Unsubscribe**, and click **Next**.
6. Check the changes you are making, and choose a time for the action to occur.
7. Click **Confirm** to schedule the changes.

## 7.8. System Groups

You can use system groups to make it easier to manage a large number of clients. Groups can be used to perform bulk actions on clients such as applying updates, configuration channels, salt states, or formulas.

You can organize clients into groups in any way that works for your environment. For example, you could organize clients on which operating system is installed, which physical location they are in, or the type of workload they are handling. Clients can be in any number of groups, so you can define your groups in different ways.

When you have clients organized into groups, you can perform updates on all clients in one or more groups, or on intersections between groups. For example, you can define one group for all clients with web server software, and another group for all SLES clients. You can then perform updates on all clients with web server software, or use the intersection between the groups, and update all SLES clients with web server software.

### 7.8.1. Create Groups

You need to create some groups before you can use them to organize your clients.

*Procedure: Creating a New System Group*

1. In the Uyuni Web UI, navigate to **Systems > System Groups**.
2. Click **Create Group**.
3. Give your new group a name and a description.
4. Click **Create Group** to save your group.
5. Repeat for each group you require.

### 7.8.2. Add Clients to Groups

You can add individual clients to your groups, or add multiple clients at the same time.

**Procedure: Adding A Single Client to a Group**

1. In the Uyuni Web UI, navigate to **Systems > System List** and click the name of the client to add.
2. Navigate to the **Groups > Join** tab.
3. Check the group to join and click **Join Selected Groups**.

**Procedure: Adding Multiple Clients to a Group**

1. In the Uyuni Web UI, navigate to **Systems > System Groups** and click the name of the group to add clients to.
2. Navigate to the **Target systems** tab.
3. Check the clients to add and click **Add Systems**.

**Procedure: Adding Multiple Clients to a Group with SSM**

1. In the Uyuni Web UI, navigate to **Systems > System List** and check each client to add, this adds the clients to the system set manager.
2. Navigate to **Systems > System Set Manager**, and go to the **Groups** tab.
3. Locate the group to join and check **Add**.
4. Click **Alter Membership**.
5. Click **Confirm** to join the clients to the selected group.

For more information about the system set manager, see **Client-configuration > System-set-manager**.

You can see which clients are in a group by navigating to **Systems > System Groups**, clicking the name of the group, and navigating to the **Systems** tab. Alternatively, you can see a graphical representation of your system groups by navigating to **Systems > Visualization > Systems Grouping**.

### 7.8.3. Work with Groups

When you have your clients arranged into groups, you can use your groups to manage updates.

In the Uyuni Web UI, navigate to **Systems > System Groups**. The list shows an icon if there are updates available for any of the clients in the group. The list shows a question mark icon if checking the update status is disabled for any of the clients in the group. Click the icon to see more information about the updates available and to apply them to the clients.

You can also work with more than one group at a time. Select the groups you want to work with, and click **Work with union** to select every client in every selected group.

Alternatively, you can work on intersections of groups. Select two or more groups, and click **Work with intersection** to select only those clients that exist in all the selected groups. For example, you might have one group for all clients with web server software, and another group for all SLES clients. The intersection of these groups would be all SLES clients with web server software.

## 7.9. System Types

Clients are categorized by system type. Every client can have both a base system type, and an add-on system type assigned.

Base system type is **Salt** for all clients.

Add-on system types include **Virtualization Host**, for clients that operate as virtual hosts, and **Container Build Host** for clients that operate as a build host.

You can adjust the add-on system type by navigating to **Systems > System List > System Types**. Check the clients you want to change the add-on system type for, select the **Add-On System Type**, and click either **Add System Type** or **Remove System Type**.

# Chapter 8. Operating System Installation

Generally, you register clients that are already running. You might have installed these machines manually just before registering them to Uyuni, or they might be pre-existing systems that were installed before you added Uyuni to your environment.

Alternatively, you can use Uyuni to help you install an operating system and register it to Uyuni in one go. This method is partially or totally automated, so you can save time answering installer questions, and is especially useful if you have many clients you want to install and register.

There are several ways to install an operating system from Uyuni:

- in-place, on clients that are already registered;
- over the network, using PXE boot;
- preparing an installation CD-ROM or an USB key, and then going to the machine to boot on that medium;
- as part of the Uyuni for Retail solution.

The in-place reinstallation method assumes that a previous operating system has already been installed on the client, and that the client has already been registered to Uyuni.

For information about the in-place installation method, see [Client-configuration > Autoinst-reinstall](#).

The network boot installation method works on unformatted machines. However, it can only be performed in certain network configurations:

- the Uyuni Server, or one of its proxies, are on the same local network as the machine you want to install, or you have a DHCP relay that allows you to cross all routers in between;
- you are able to set up a new DHCP server or to configure an existing one;
- the client to install is able to boot with PXE, and you can configure it to do so.

For information about the network boot method, see [Client-configuration > Autoinst-pxeboot](#).

The removable medium method allows you to bypass these network constraints. However, it assumes the machine is able to read CD-ROMs or USB keys, and boot from them. It also requires physical access to the client machine.

For information about the removable media method, see [Client-configuration > Autoinst-cdrom](#).

For information about the Uyuni for Retail approach, see [Retail > Retail-overview](#).



- Autoinstallation of Ubuntu and Debian clients is not supported. These operating systems must be installed manually.

The autoinstallation features of Uyuni are based on a software named Cobbler. For more information

about Cobbler, see <https://cobbler.readthedocs.io>.



- SUSE only supports Cobbler functions that are available in the Uyuni Web UI, or through the Uyuni API. The sole command-line command supported by Cobbler is **buildiso**. Only supported features are documented here.

## 8.1. Reinstall Registered Systems

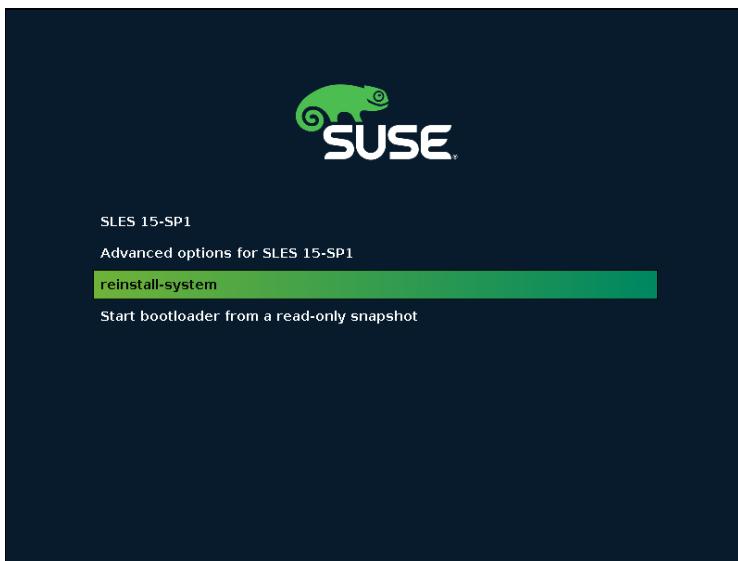
The in-place reinstallation starts from the local client system. There is therefore no need for the client to be able to boot over network with PXE.

To reinstall a registered client in-place, you must define an autoinstallable distribution and an autoinstallation profile. For information, see **Client-configuration > Autoinst-distributions** and **Client-configuration > Autoinst-profiles**.

When you have defined the autoinstallation profile and distribution, you can perform the reinstallation.

### *Procedure: Reinstall an Already Registered Client*

1. In the Uyuni Web UI, navigate to **Systems > Systems List**, select the client to reinstall, and go to the **Provisioning > Autoinstallation > Schedule** subtab.
2. Select the autoinstallation profile that you prepared, select a proxy if needed, and click **Schedule** **Autoinstallation and Finish**.
3. You can monitor progress of the installation by navigating to **Provisioning > Autoinstallation > Session Status**, or on the client directly. The client reboots, and in the boot menu selects a new choice called **reinstall-system**.

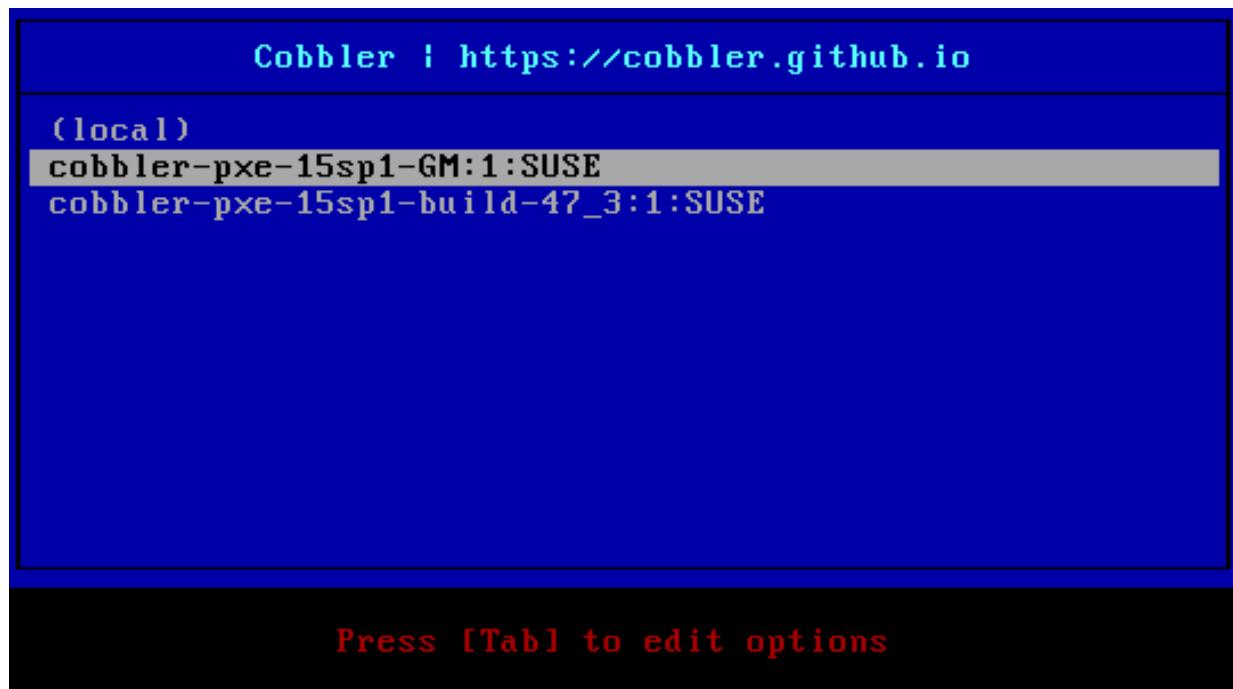


The installation then proceeds over HTTP protocol.

## 8.2. Install via the Network (PXE Boot)

During a network boot installation:

1. The client boots in PXE mode.
2. A DHCP server provides the client with an IP address and mask, the address of an installation server, and the name of a bootloader file on that server.
3. The client downloads the bootloader file via the TFTP protocol from the installation server and executes it.
4. The client is presented with a choice of possible profiles to install from a menu, or starts automatically installing one of the profiles.
5. The client downloads via the TFTP protocol a kernel and an initial RAM disk for the distribution matching that profile.
6. The installation kernel starts an installation program, either Kickstart or AutoYaST. From now on it uses resources provided on the server via the HTTP protocol.
7. The distribution is installed automatically according to the Kickstart or AutoYaST profile.
8. The profile calls a code snippet that registers the client to the Uyuni Server.



The installation server can either be the Uyuni Server or one of its proxies. To install from a proxy, you must synchronize the TFTP tree between the server and the proxy before you begin.

The DHCP server may also provide other configuration information to the client, like a host name, the address of a router and the address of a domain name server. Some of this information might be needed for the autoinstallation, for example if you specify the installation server by its domain name.

In the PXE boot menu, the first choice is **Local boot**. If you select this, the boot process continues from the local disk drives. This option is selected automatically if no profile was selected after a certain time. This is a security measure to prevent from starting an autoinstallation if no human operator is there

to select one of the profiles.

Alternatively, the installation can start automatically from one of the profiles without manual intervention. This is called "unattended provisioning."

The "bare metal" functionality is a kind of unattended provisioning based on PXE boot. In this scenario, the bootloader file only registers the client at the Uyuni Server and does not start the installation. You can then trigger an in-place reinstallation later.

#### *Procedure: Installing with PXE Boot*

1. Prepare the DHCP server, see [Prepare the DHCP Server](#).
2. Prepare an autoinstallable distribution, see [Client-configuration > Autoinst-distributions](#).
3. Prepare an autoinstallation profile, see [Client-configuration > Autoinst-profiles](#).
4. Reboot the client, and chose a profile to install.

Some other steps are optional. To use a proxy as the installation server, see [Synchronize the TFTP Tree with Proxies](#). For unattended provisioning, see [Client-configuration > Autoinst-unattended](#).

### 8.2.1. Prepare the DHCP Server

The PXE boot process uses DHCP to find the TFTP server. The Uyuni Server or its proxies can act as such a TFTP server.

You must have administrative access to the network's DHCP server. Edit the DHCP configuration file so that it points to the installation server as the TFTP boot server.

#### *Example: Configuring the ISC DHCP Server*

1. On the DHCP server, as root, open the `/etc/dhcpd.conf` file.
2. Modify the declaration for your client:

```
host myclient { (...)  
    next-server 192.168.2.1;  
    filename "pxelinux.0"; }
```

1. Save the file and restart the `dhcpd` service.

This example directs the PXE client `myclient` to the installation server at `192.168.2.1` and instructs it to retrieve the `pxelinux.0` bootloader file.

Alternatively, if your DHCP server is registered at Uyuni, you can configure it using the DHCPd formula instead:

#### *Example: Configuring the ISC DHCP Server Using the DHCPd Formula*

1. Navigate to **Systems > System List**, select the client you want to change, and go to the **Formulas**

tab to enable the DHCPd formula.

2. Go to the formula's **Dhcpd** tab, and in the **Next Server** field, enter the hostname or IP address of the installation server.
3. In the **Filename EFI** field, type **grub/shim.EFI** to enable EFI PXE support.
4. In the **Filename** field, type **pxelinux.0** to enable legacy BIOS support.
5. Click **Save Formula** to save your configuration.
6. Apply the highstate.



If you do not use Secure Boot, type **grub/grubx86.efi** instead of **grub/shim.EFI** in the **Filename EFI** field.



For different architectures, see table [GRUB EFI binary names for different architectures](#).



Using Cobbler managed DHCP is unsupported with Uyuni.

This sets up a global PXE server for all the hosts, you can also have per-host settings. For more information about the DHCPd formula, see [Specialized-guides > Salt](#).

### 8.2.2. Synchronize the TFTP Tree with Proxies

You can synchronize the TFTP tree on the Uyuni Server with the Uyuni Proxy. For synchronization, HTTPS port 443 must be open.



Every added proxy slows down the tree synchronization.

*Procedure: Synchronizing TFTP between Server and Proxy*

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

```
zypper install susemanager-tftpsync
```

1. On the Uyuni Proxy, at the command prompt, as root, install the **susemanager-tftpsync-recv** package:

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

1. On the proxy, as root, run the **configure-tftpsync.sh** script. The script interactively asks you for details on the host names and IP addresses of the Uyuni Server and Proxy, as well for the location of the **tftpboot** directory on the Proxy. For more information, use the **configure-tftpsync.sh --help** command.
2. On the server, as root, run the **configure-tftpsync.sh** script.

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

- Run the `cobbler sync` command on the server to push the files to the proxy. This fails if you have not configured the proxies correctly.

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



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

### 8.2.3. GRUB EFI binary names for different architectures

*Table 44. GRUB EFI binary names for different architectures*

Architecture	GRUB EFI binary name
aarch64	grubaa64.efi
x86-64	grubx86.efi
ppc64le	grub.ppc64le

## 8.3. Install via a CD-ROM or a USB Key

For clients that are not yet registered to Uyuni, and if network boot over PXE is not an option, a bootable CD-ROM or USB key can be used to install the system.

One option to prepare such a removable medium is to use Cobbler. For information about using Cobbler to prepare an ISO image, see [Build an ISO Image With Cobbler](#).

For SUSE systems, it is often recommended to prepare an ISO image using KIWI. For more information, see [Build a SUSE ISO Image With KIWI](#).

In all cases, you use the resulting image to burn a CD-ROM or prepare a USB key.

### 8.3.1. Build an ISO Image With Cobbler

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



- Building ISOs with Cobbler is not supported on IBM Z.

In order to prepare an ISO image with Cobbler, you need to prepare a distribution and a profile, similar to using network boot over PXE. For information about creating a distribution, see [Client-configuration > Autoinst-distributions](#). For information about creating a profile, see [Client-configuration > Autoinst-](#)

profiles.

The Cobbler **buildiso** command takes parameters to define the name and output location of the boot ISO. Specifying the distribution with **--distro** is mandatory when running **buildiso** command.

```
cobbler buildiso --iso=/path/to/boot.iso --distro=<your-distro-label>
```



- [: You must use distro and profile labels as listed by Cobbler, and not simply as shown in the UI.

To list the names of distributions and profiles stored by Cobbler, run the commands:

```
# cobbler distro list
# cobbler profile list
```

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

```
cobbler buildiso --systems="system1 system2 system3" \
--profiles="<your-profile1-label> <your-profile2-label> <your-profile3-label>" \
--distro=<your-distro-label>
```



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

### 8.3.2. Build a SUSE ISO Image With KIWI

KIWI is an image creation system. You can use KIWI to create a bootable ISO image to be used by the target system for installation of a SUSE system. When the system is rebooted or switched on, it boots from the image, loads the AutoYaST configuration from your Uyuni, and installs SUSE Linux Enterprise Server according to the AutoYaST profile.

To use the ISO image, boot the system and type **autoyast** at the prompt (assuming you left the label for the AutoYaST boot as **autoyast**). Press **Enter** to begin the AutoYaST installation.

For more information about KIWI, see <http://doc.opensuse.org/projects/kiwi/doc/>.

### 8.3.3. Build a Red Hat ISO Image With Cobbler

For more information, see [client-configuration:autoinst-cdrom.pdf](#).

## 8.4. Autoinstallable Distributions

The autoinstallation process relies on several files to initiate the installation. These files include the Linux kernel, an initial RAM disk, and other files required to boot the operating system in installation

mode.

You can extract the needed files from a DVD image. For information, see [Distribution Based on an ISO Image](#).

Alternatively, you can install the **tftpboot-installation** package. For information, see [Distribution Based on a RPM Package](#).

You must also have a base channel synchronized on your Uyuni Server for the same operating system version as those files.

When you have the files ready, and the base channel synchronized, you need to declare the distribution. This operation associates the installation files to the base channel. The distribution can be referred to by one or more installation profiles. For information, see [Declare an Autoinstallable Distribution](#).

### 8.4.1. Distribution Based on an ISO Image

This method assumes you have installation media for the operating system you want to install on the clients. This is usually a DVD **.iso** image that contains the Linux kernel, an **initrd** file, and other files required to boot the operating system in installation mode.

*Procedure: Importing Files from Installation Media*

1. Copy the installation media to your Uyuni Server. For SUSE operating systems, you can download installation media from <https://www.suse.com/download/>.
2. Loop-mount the ISO image, and copy its contents somewhere:

```
# mount -o loop,ro <image_name>.iso /mnt
# mkdir -p /srv/www/distributions
# cp -a /mnt /srv/www/distributions/<image_name>
# umount /mnt
```

3. Take a note of the file path. You will need it when you declare the distribution to Uyuni.

### 8.4.2. Distribution Based on a RPM Package

This method works on SUSE systems. It is simpler than importing contents from an installation media, because it uses prepackaged files for your installation system.

*Procedure: Extracting Files from an Installation Package*

1. On the Uyuni Server, install the package whose name starts with **tftpboot-installation**. You can determine its exact name with the command **zypper se tftpboot-installation**
2. Determine where the installation files went with the command **ls -d /usr/share/tftpboot-installation/\***. Take a note of the file path. You will need it when you declare the distribution to Uyuni.

This procedure prepares the installation of the same operating system version as the one that powers your

Uyuni Server. If you want to install a different operating system or version on the client, you need to manually get the package **tftpboot-installation-\*** from the distribution it belongs to. In the **Package Search** input box Uyuni, search the packages whose names start with **tftpboot-installation**, then look at the package's details. They show the local path below **/var/spacewalk/**.

### 8.4.3. Declare an Autoinstallable Distribution

The next step after extracting the autoinstallation files is to declare an autoinstallable distribution.

*Procedure: Declaring an Autoinstallable Distribution*

1. In the Uyuni Web UI, navigate to **Systems > Autoinstallation > Distributions**.
2. Click **Create Distribution**, and complete these fields:
  - In the **Distribution Label** field, enter a name to identify your autoinstallable distribution.
  - In the **Tree Path** field, enter the path to the installation media saved on your Uyuni Server.
  - Select the matching **Base Channel**. This must match the installation media.
  - Select the **Installer Generation**. This must match the installation media.
  - OPTIONAL: Specify kernel options to use when booting this distribution. There are multiple ways to provide kernel options. Only add options here that are generic for the distribution.
3. Click **Create Autoinstallable Distribution**.

The installation files that you prepared might not contain the packages you need to install. If they are not included, add **useonlinerepo=1** to the **Kernel Options** field.

The package repositories contain metadata that can be unsigned. If the metadata is unsigned, add **insecure=1** to the **Kernel Options** field, or use your own GPG key as explained in **Client-configuration > Autoinst-owngpgkey**.

These kernel options are needed for example when you use the "online installer" ISO images instead of the full DVD, or when you use the **tpboot-installation** package.

Navigate to **Systems > Autoinstallation > Distributions** to manage your autoinstallable distributions.

## 8.5. Autoinstallation Profiles

An autoinstallation profile determines how the operating system will be installed. For example, you can specify additional kernel parameters to be passed to the installer.

The most important part of the profile is the "autoinstallation file." When you perform an installation manually, you must provide information to the installer, such as partitioning and networking information and user details. The autoinstallation file is a method of providing this information in a scripted form. This type of file is sometimes also referred to as an "answers file."

Within Uyuni, you can use two different types of profiles, depending on the operating system of the clients you want to install:

- For SUSE Linux Enterprise or openSUSE clients, use AutoYaST.
- For Red Hat Enterprise Linux clients, use Kickstart.

You can use both AutoYaST and Kickstart profiles if you want to install clients with different operating systems.

- For information about how to declare profiles, see [Declare the Profile](#)
- For information about AutoYaST profiles, see [AutoYaST Profiles](#).
- For information about Kickstart profiles, see [Kickstart Profiles](#).

The autoinstallation file contained in the profile can include variables and code snippets. For information about variables and code snippets, see [Templates Syntax](#).

### 8.5.1. Declare the Profile

When you have prepared an autoinstallation file and distribution, you can create profiles to manage autoinstallation on your Uyuni Server. The profile will determine how to install this distribution you selected. One way to create a profile is to upload an AutoYaST or Kickstart file. Alternatively, for Kickstart only, you can use the Web UI wizard.

*Procedure: Creating an Autoinstallation Profile by Upload*

1. In the Uyuni Web UI, navigate to **Systems > Autoinstallation > Profiles**.
2. Click **Upload Kickstart/AutoYaST File**.
3. In the **Label** field, type a name for the profile. Do not use spaces.
4. In the **Autoinstall Tree** field, select the autoinstallable distribution to use for this profile.
5. In the **Virtualization Type** field, select the type of virtualization to use for this profile, or select **None** if you do not want to use this profile to create a new virtual machine.
6. Copy the contents of your autoinstallation file into the **File Contents** field, or upload the file directly using the **File to Upload** field.

For more information about the details to include here, see [AutoYaST Profiles](#) or [Kickstart Profiles](#).

7. Click **Create** to create the profile.

*Procedure: Creating a Kickstart Profile by Wizard*

1. In the Uyuni Web UI, navigate to **Systems > Autoinstallation > Profiles**.
2. Click **Create Kickstart Profile**.
3. In the **Label** field, type a name for the profile. Do not use spaces.

4. In the **Base Channel** field, select the base channel to use for this profile. This field is populated from the distributions available. If the base channel you need is not available, check that you have created the distribution correctly.
5. In the **Virtualization Type** field, select the type of virtualization to use for this profile, or select **None** for no virtualization.
6. Click **Next**.
7. In the **Distribution File Location** type the path to the installation media installed on the Uyuni Server.
8. Click **Next**.
9. Provide a password for the root user on the client.
10. Click **Finish**.
11. Review the details of your new profile, and customize as required.

When you are creating your autoinstallation profile, you can check **Always use the newest Tree for this base channel**. This setting allows Uyuni to automatically pick the latest distribution that is associated with the specified base channel. If you add new distributions later, Uyuni uses the most recently created or modified.

Changing the **Virtualization Type** usually requires changes to the profile bootloader and partition options. This can overwrite your customization. Verify new or changed settings before saving them, by navigating to the **Partitioning** tab.

The kernel options from the distribution and the profile are combined.

You can change the details and settings of your autoinstallation profiles by navigating to **Systems > Autoinstallation > Profiles** and clicking the name of the profile you want to edit. Alternatively, navigate to **Systems > System List**, select the client you want to provision, and navigate to the **Provisioning > Autoinstallation** subtab.

## 8.5.2. AutoYaST Profiles

An AutoYaST profile consists of a **Label** that identifies the profile, an **Autoinstall Tree** that points to an autoinstallable distribution, various options, and, most importantly, an AutoYaST installation file.

The AutoYaST installation file is an XML file that give directions to the AutoYaST installer. AutoYaST calls it a "control file." For the full syntax of AutoYaST installation files, see <https://doc.opensuse.org/projects/autoyast/#cha-configuration-installation-options>.

SUSE provides templates of AutoYaST installation files that you can use as a starting point for your own custom files. You will find the templates at <https://github.com/SUSE/manager-build-profiles> in the **AutoYast** directory. Each of these profiles requires you to set some variables before you use it. Check the **README** file included with the script to determine which variables you need. For more information about using variables in AutoYaST scripts, see **Variables**.

These are the most important sections in the AutoYaST installation file for installing with Uyuni:

- <**add-on**> allows to add child channels to the installation. For an example, see <https://doc.opensuse.org/projects/autoyast/#Software-Selections-additional>.
  - `<general>\$SNIPPET('spacewalk/sles\_no\_signature\_checks')</general>` disables signature checks
  - <**software**> allows to specify product for the Unified Installer
    - See <https://doc.opensuse.org/projects/autoyast/#CreateProfile-Software> with a "<software>" example
  - <**init-scripts**  
**config:type="list">\$SNIPPET('spacewalk/minion\_script')</init-scripts>**
- allows the client to register to Uyuni as a Salt client.

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

A more recent, Salt-based alternative to AutoYaST, is Yomi. For information about Yomi, see **Specialized-guides > Salt**.

### 8.5.3. Kickstart profiles

Kickstart profiles offer a large number of configuration options. To create these profiles, you can upload them, or use a dedicated wizard.

Kickstart profiles allow you to use file preservation lists. If you have many custom configuration files located on a client you want to reinstall with Kickstart, you can save them as a list, and associate that list with the Kickstart profile.

*Procedure: Creating a file preservation list*

1. In the Uyuni Web UI, navigate to **Systems > Autoinstallation > File Preservation** and click **Create File Preservation List**.
2. Enter a suitable label, and list absolute paths to all files and directories you want to save.
3. Click **Create List**.
4. Include the file preservation list in your Kickstart profile.
5. Navigate to **Systems > Autoinstallation > Profiles** and select the profile you want to edit, go to the **System Details > File Preservation** subtab, and select the file preservation list to include.



File preservation lists are limited to a total size of 1 MB. Special devices like `/dev/hda1` and `/dev/sda1` cannot be preserved. Only use file and directory names, you cannot use regular expression wildcards.

For more information about Kickstart, see the Red Hat documentation.

## 8.5.4. Templates syntax

Parts of your installation file are replaced during the installation. Variables are replaced with single values, and code snippets are replaced with whole sections of text. Escaped symbols or sections are not replaced.

A template engine called Cheetah allows Cobbler to do these replacements. This mechanism allows you to reinstall large numbers of systems, without having to manually create profiles for each of them.

You can create autoinstallation variables and code snippets within the Uyuni Web UI. Within a profile, the **Autoinstallation File** tab allows you to see the result of the substitutions.

- For information about variables, see [Variables](#).
- For information about code snippets, see [Code Snippets](#).
- For information about escaping symbols or text blocks, see [Escaping](#).

### 8.5.4.1. Variables

Autoinstallation variables can be used to substitute values into Kickstart and AutoYaST profiles. To define a variable, from the profile, navigate to the **Variables** subtab, and create a **name=value** pair in the text box.

For example, you could create a variable that holds the IP address of the client, and another that holds the address of its gateway. Those variables can then be defined for all the clients installed from the same profile. To do that, add these lines to the **Variables** text box:

```
ipaddr=192.168.0.28
gateway=192.168.0.1
```

To use the variable, prepend a **\$** sign in the profile to substitute the value. For example, the **network** part of a Kickstart file may look like the following:

```
network --bootproto=static --device=eth0 --onboot=on --ip=$ipaddr \
--gateway=$gateway
```

The **\$ipaddr** is resolved to **192.168.0.28**, and the **\$gateway** to **192.168.0.1**.

In installation files, variables use a hierarchy. System variables take precedence over profile variables, which in turn take precedence over distribution variables.

### 8.5.4.2. Code snippets

Uyuni comes with a large number of predefined code snippets. Navigate to **Systems > Autoinstallation > Autoinstallation Snippets** to see the list of existing snippets.

Use a snippet by inserting the **\$SNIPPET()** macro in your autoinstallation file. For example, in Kickstart:

```
$SNIPPET('spacewalk/rhel_register_script')
```

Or, in AutoYaST:

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

The macro is parsed by Cobbler and substituted with the contents of the snippet. You can also store your own code snippets to use in autoinstallation files later on. Click [Create Snippet](#) to create a new code snippet.

This example sets up a Kickstart snippet for a common hard drive partition configuration:

```
clearpart --all
part /boot --fstype ext3 --size=150 --asprimary
part / --fstype ext3 --size=40000 --asprimary
part swap --recommended

part pv.00 --size=1 --grow

volgroup vg00 pv.00
logvol /var --name=var vgname=vg00 --fstype ext3 --size=5000
```

Use the snippet with, for example:

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

#### 8.5.4.3. Escaping

If the autoinstallation file contains shell script variables like **\$(example)**, the content needs to be escaped with a backslash: **\\$(example)**. Escaping the **\$** symbol prevents the templating engine from evaluating the symbol as an internal variable.

Text blocks such as code fragments or scripts can be escaped by wrapping them with the **\#raw** and **\#end raw** directives. For example:

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

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

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

## 8.6. Unattended Provisioning

The "bare metal" functionality allows you to register any new machine as soon as it connects to the local network, using a generic PXE boot image. You then go to Uyuni Web UI and assign a profile to this machine. The next time the client boots, the operating system is installed according to that profile. For information about the bare metal provisioning, see [Bare Metal Provisioning](#).

If you do not want to use the bare metal functionality, you can still declare manually the system at Uyuni. The Uyuni API allows you to create system records for systems as if they were collected by the bare metal functionality. For information about declaring systems with the API, see [Create a System Record Manually](#).

### 8.6.1. Bare Metal Provisioning

When you have the bare metal provisioning option enabled, any client connected to the Uyuni network is automatically added to the organization as soon as it is powered on. When this operation is complete, the client is shut down, and it appears in the **Systems** list, ready to be installed.

*Procedure: Enable the Bare Metal Functionality*

1. In the Uyuni Web UI, navigate to **Admin > Manager Configuration > Bare-metal systems**.
2. Click **Enable adding to this organization**.

New clients that are powered on are added to the organization that belongs to the administrator who enabled the bare metal feature. They are of "bootstrap" type, and still need to be provisioned to become regular clients.

To change the organization new clients are added to, disable the bare metal feature, log in as the administrator of the new organization, and then re-enable the feature. You can migrate already registered systems to other organizations, using the **Migrate** tab.

You can use the system set manager (SSM) with clients that are registered that way. However, not all SSM features are available for those clients, because they do not yet have an operating system installed. This also applies to mixed sets that include systems registered that way. All features become available to the set when all the clients in the set have been provisioned. For more information on SSM, see [Client-configuration > System-set-manager](#).

*Procedure: Provision "Bootstrap" Type clients*

1. In the Uyuni Web UI, navigate to **Systems**, select the client to provision, and go to the **Provisioning > Autoinstallation** tab.
2. Select the AutoYaST profile to use, and click **Create PXE installation configuration**. This option creates a system entry at Cobbler.

---

### 3. Power on the client.

The server uses TFTP to provision the new client, so the appropriate port and networks must be configured correctly for provisioning to be successful.

#### 8.6.2. Create a System Record Manually

You can use an API call to declare an association between a client identified by its MAC address and an autoinstallation profile. Next time the system reboots, it starts the installation based on the specified profile.

*Procedure: Reinstallation From a Manually Declared Profile*

1. On the Uyuni Server, at the command prompt, use the `system.createSystemRecord` API call. In this example, replace `name` with the name of your client, `<profile>` with the profile label, `<iface>` with the name of the interface on the client such as `eth0`, and `<hw_addr>` with its hardware address such as `00:25:22:71:e7:c6`:

```
$ spacecmd api --args '[{"name": "<name>", "profile": "<profile>", "", "", "", \\
[ {"name": "<iface>", "mac": "<hw_addr>"} ]]' \\
system.createSystemRecord
```

2. Power on the client. It boots from the network, and the correct profile is selected for installation.

This command creates a system record at Cobbler. You may also specify additional parameters, like kernel options, the IP address of the client, and its domain name. For more information, see the API documentation for `createSystemRecord` call.

## 8.7. Use Your Own GPG Key

If the repositories you are using for autoinstallation have unsigned metadata, you usually have to use the `insecure=1` kernel parameter as an option of the autoinstallable distribution, and use a `spacewalk/sles_no_signature_checks` code snippet in the AutoYaST installation file.

A safer alternative is to provide your own GPG key.



This technique applies to SUSE clients only.

*Procedure: Include your own GPG key*

1. Create a GPG key.
2. Use it to sign the package's metadata.
3. Add it to the initial RAM disk of your installation media.
  - For information on how to create the key and sign the metadata with it, see **Administration > Repo-metadata**.
  - For information on how to add the key to installation media used for network boot, see **Own**

## GPG Key for PXE Boot.

- For information on how to add the key to installation media used for a boot from CD-ROM, see [Own GPG Key in a CD-ROM](#).



When you signed the metadata with your new GPG key, any already onboarded client will not know about the new key. Ideally, you should sign the metadata before you register any client.

For already onboarded clients that use those repositories, the workaround is to disable GPG key checking on them.

### 8.7.1. Own GPG Key for PXE Boot

The initial RAM disk (**initrd**) used by PXE boot process normally contains SUSE's GPG key only. You must add our own key to this file, so it can be used to check the packages.

*Procedure: Add a GPG key to the initial RAM disk*

- Create a directory with a path identical to the one that is used during the boot process to find the GPG key:

```
mkdir -p tftpboot/usr/lib/rpm/gnupg/keys
```

- Copy your GPG key into this directory with the suffix **.asc**:

```
cp /srv/www/htdocs/pub/mgr-gpg-pub.key tftpboot/usr/lib/rpm/gnupg/keys/mgr-gpg-pub.asc
```

- Inside the top level directory, package the content and append it to the **initrd** that is part of your installation media files:

```
cd tftpboot
find . | cpio -o -H newc | xz --check=crc32 -c >> /path/to/initrd
```

### 8.7.2. Own GPG key in a CD-ROM

You can modify an installation image with the **mksusecd** utility. This utility is contained in the Development Tools module.

*Procedure: Add a GPG key to an Installation ISO image*

- Create a directory with a path identical to the one that is used during the boot process to find the GPG key:

```
mkdir -p initrdroot/usr/lib/rpm/gnupg/keys
```

- 
2. Copy your GPG key into this directory with the suffix **.asc**:

```
cp /srv/www/htdocs/pub/mgr-gpg-pub.key initrdroot/usr/lib/rpm/gnupg/keys/mgr-gpg-pub.asc
```

3. Amend the existing ISO image with **mksusecd**:

```
mksusecd --create <new-image>.iso --initrd initrdroot/ <old-image>.iso
```

# Chapter 9. Virtualization

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

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

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

For VMWare, including VMWare vSphere, and Nutanix AHV, Uyuni requires you to set up a virtual host manager (VHM) to control the VMs. This gives you control over the hosts and guests, but in a more limited way than available with Xen and KVM; Uyuni cannot create or edit VMs on VMWare vSphere or Nutanix AHV.

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

For more information about using VHMs to manage virtualization, see [Client-configuration > Vhm](#).

## 9.1. Manage Virtualized Hosts

Before you begin, ensure that the client you want to use as a virtualization host has the **Virtualization Host** system type assigned to it. Navigate to **Systems > Systems List** and click the name of the client to use as a virtualization host. If the **Virtualization Host** system type is not listed, initialize the **Virtualization Host** formula. For more information, see [client-configuration:virt-xenkvm.pdf](#).

When a client has the **Virtualization Host** system type, the **Virtualization** tab is available in the System Details page for the client. The **Virtualization** tab allows you to create and manage virtual guests, and manage storage pools and virtual networks.

## 9.2. Create Virtual Guests

You can add virtual guests to your virtualization hosts within the Uyuni Web UI.

### *Procedure: Creating a Virtual Guest*

1. In the Uyuni Web UI, navigate to **Systems > Systems List**, click the name of the virtualization host, and navigate to the **Virtualization** tab.
2. In the **General** section, complete these details:
  - In the **Guests** subtab, click **Create Guest**.

- In the **Name** field, type the name of the guest.
  - In the **Hypervisor** field, select the hypervisor to use.
  - In the **Virtual Machine Type** field, select either fully virtualized or para-virtualized.
  - In the **Maximum Memory** field, type the upper size limit for the guest disk, in MiB.
  - In the **Virtual CPU count**, type the number of vCPUs for the guest.
  - In the **Architecture** field, select the emulated CPU architecture to use on the guest. By default, the architecture selected matches the virtual host.
  - In the **Auto-installation Profile** field, select the auto-installation tool to use to install the guest. Leave this field blank if you do not want to use auto-installation.
3. In the **Disks** section, complete the details of the virtual disk to use with the client. In the **Source template image URL** field, ensure you type the path to an operating system image. If you do not do this, your guest is created with an empty disk.
  4. In the **Networks** section, complete the details of the virtual network interface to use with the client. Leave the **MAC address** field blank to generate a MAC address.
  5. In the **Graphics** section, complete the details of the graphics driver to use with the client.
  6. Schedule a time for the guest to be created, and click **Create** to create the guest.
  7. The new virtual guest starts as soon as it has successfully been created.

You can add virtual guests on a pacemaker cluster within the Uyuni Web UI, too.

*Procedure: Creating a cluster managed Virtual Guest*

1. Follow the **Creating a Virtual Guest** procedure on one of the nodes of the cluster with the following additions:
  - Ensure the **Define as a cluster resource** field is checked.
  - In the **Path to the cluster shared folder for VM definitions** field, type the path to a folder shared by all cluster nodes where the guest configuration will be stored.
  - Make sure every disk is located on a storage pool shared by all the cluster nodes.

Virtual Guests managed by a cluster can be live migrated.

## 9.3. Virtualization with Xen and KVM

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

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

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

### 9.3.1. Host setup

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

- For SUSE operating systems, see the SLES Virtualization Guide available from <https://documentation.suse.com/sles/15-SP4/html/SLES-all/book-virtualization.html>.
- For Red Hat Enterprise Linux operating systems, refer to the Red Hat documentation for your version.

The **Virtualization Host** formula will help you with initializing the host. For more information, see [client-configuration:virt-xenkvm.pdf](#).

#### 9.3.1.1. Background information

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

You need a bootstrap script on the Uyuni Server. The bootstrap script must include the activation key for your host. We also recommend that you include your GPG key for additional security. For more on creating a bootstrap script, see [Client-configuration > Registration-bootstrap](#).

When the bootstrap script is ready, use it to register the host with the Uyuni Server. For more information about client registration, see [Client-configuration > Registration-overview](#).

#### 9.3.1.2. Initialize **Virtualization Host**

With the **Virtualization Host** formula, initialize the host:

##### *Procedure: Initializing **Virtualization Host***

1. In the Uyuni Web UI, navigate to the **System Details** page for the host, and click the **Formulas** tab.
2. Select the **Virtualization Host** formula, and click **Save**.
3. Click the **Virtualization Host** subtab.
4. Check the settings, and click **Save Formula**.
5. Apply the Highstate for the changes to take effect.
6. Restart the **salt-minion** service to activate the new configuration:

```
systemctl restart salt-minion
```

## 9.3.2. Autoinstall VM Guests

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

You need an activation key for the VM host you want to register the guests to, and for each guest. The activation key must have the **provisioning** and **Virtualization Platform** entitlements. For more information about creating activation keys, see **Client-configuration > Activation-keys**.

If you want to automatically register the guests with Uyuni after installation, you need to create a bootstrap script. For more information about creating a bootstrap script, see **Client-configuration > Registration-bootstrap**.

### 9.3.2.1. Create an Autoinstallable Distribution

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

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

*Table 45. Paths for autoinstallable distributions*

Operating System Type	Kernel Location	initrd Location
Red Hat Enterprise Linux	<code>images/pxeboot/vmlinuz</code>	<code>images/pxeboot/initrd.img</code>
SUSE	<code>boot/&lt;arch&gt;/loader/initrd</code>	<code>boot/&lt;arch&gt;/loader/linux</code>

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

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

#### *Procedure: Creating an Autoinstallable Distribution*

1. In the Uyuni Web UI, navigate to **Systems > Autoinstallation > Distributions** and click **Create Distribution**.
2. In the **Create Autoinstallable Distribution** section, use these parameters:
  - In the **Distribution Label** section, type a unique name for the distribution. Use only letters, numbers, hyphens (-), periods (.), and underscores (\_), and ensure the name is longer than four characters.

- In the **Tree Path** field, type an absolute path to the installation source.
- In the **Base Channel** field, select the channel that matches the installation source. This channel is used as the package source for non-SUSE installations.
- In the **Installer Generation** field, select the operating system version that matches the installation source.
- In the **Kernel Options** field, type any options to be passed to the kernel when booting for the installation. The `install=` parameter and the `self_update=0` `pt.options=self_update` parameter are added by default.
- In the **Post Kernel Options** section, type any options to be passed to the kernel when booting the installed system for the first time.

3. Click **Create Autoinstallable Distribution** to save.

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

### 9.3.2.2. Create and Upload an Autoinstallation Profile

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

Kickstart profiles can be created using the Uyuni Web UI, by navigating to **Systems > Autoinstallation > Profiles**, clicking **Create New Kickstart File**, and following the prompts.

You can also create AutoYaST or Kickstart autoinstallation profiles by hand. SUSE provides templates of AutoYaST installation files that you can use as a starting point for your own custom files. You will find them at <https://github.com/SUSE/manager-build-profiles>.

If you are using AutoYaST to install SLES, you also need to include this snippet:

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

- For more on AutoYaST, see [client-configuration:autoinst-profiles.pdf](#).
- For more on Kickstart, see [client-configuration:autoinst-profiles.pdf](#), or refer to the Red Hat documentation for your installation.

#### *Procedure: Uploading an Autoinstallation Profile*

1. In the Uyuni Web UI, navigate to **Systems > Autoinstallation > Profiles** and click **Upload Kickstart/AutoYaST File**.
2. In the **Create Autoinstallation Profile** section, use these parameters:
  - In the **Label** field, type a unique name for the profile. Use only letters, numbers, hyphens (-),

periods (.), and underscores (\_), and ensure the name is longer than six characters.

- In the **Autoinstall Tree** field, select the autoinstallable distribution you created earlier.
- In the **Virtualization Type** field, select the relevant Guest type (for example, **KVM Virtualized Guest**). Do not choose **Xen Virtualized Host** here.
- OPTIONAL: If you want to manually create your autoinstallation profile, you can type it directly into the **File Contents** field. If you have a file already created, leave the **File Contents** field blank.
- In the **File to Upload** field, click **Choose File**, and use the system dialog to select the file to upload. If the file is successfully uploaded, the filename is shown in the **File to Upload** field.
- The contents of the uploaded file is shown in the **File Contents** field. If you need to make edits, you can do so directly.

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

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



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

### 9.3.2.3. Automatically Register Guests

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

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

For more information about creating a bootstrap script, see **Client-configuration > Registration-bootstrap**. For instructions on how to do this for Kickstart, refer to the Red Hat documentation for your installation.

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

1. Ensure your bootstrap script contains the activation key for the VM guests you want to register with it, and that is located on the host at `/srv/www/htdocs/pub/bootstrap_vm_guests.sh`.
2. In the Uyuni Web UI, navigate to **Systems > Autoinstallation > Profiles**, and select the AutoYaST profile to associate this script with.
3. In the **File Contents** field, add this snippet at the end of the file, immediately before the closing `</profile>` tag. Ensure you replace the example IP address **192.168.1.1** in the snippet with the

correct IP address for your Uyuni Server:

```
<scripts>
  <init-scripts config:type="list">
    <script>
      <interpreter>shell</interpreter>
      <location>
        http://192.168.1.1/pub/bootstrap/bootstrap_vm_guests.sh
      </location>
    </script>
  </init-scripts>
</scripts>
```

4. Click **Update** to save your changes.



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

#### 9.3.2.4. Autoinstall VM Guests

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



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

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

#### 9.3.3. Manage VM Guests

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

To do this, you need your Xen or KVM VM host registered to the Uyuni Server, and have the **libvirt** service running on the host.

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

For more information about managing VM guests using the Web UI, see [Reference > Systems](#).

# Chapter 10. Virtual Host Managers

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

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

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

- For more information on using a VHM with Amazon Web Services, see [Client-configuration > Vhm-aws](#).
- For more information on using a VHM with Microsoft Azure, see [Client-configuration > Vhm-azure](#).
- For more information on using a VHM with Google Compute Engine, see [Client-configuration > Vhm-gce](#).
- For more information on using a VHM with Kubernetes, see [Client-configuration > Vhm-kubernetes](#).
- For more information on using a VHM with Nutanix, see [Client-configuration > Vhm-nutanix](#).
- For more information on using a VHM with VMWare vSphere, see [Client-configuration > Vhm-vmware](#).
- For more information on using a VHM with other hosts, see [Client-configuration > Vhm-file](#).

## 10.1. VHM and Amazon Web Services

You can use a virtual host manager (VHM) to gather instances from Amazon Web Services (AWS).

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

### 10.1.1. Create an Amazon EC2 VHM

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

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

*Procedure: Creating an Amazon EC2 VHM*

1. In the Uyuni Web UI, navigate to **Systems > Virtual Host Managers**.
2. Click **Create** and select **Amazon EC2** from the drop-down menu.
3. In the **Add an Amazon EC2 Virtual Host Manager** section, use these parameters:

- In the **Label** field, type a custom name for your VHM.
  - In the **Access Key ID** field, type the access key ID provided by Amazon.
  - In the **Secret Access Key** field, type the secret access key associated with the Amazon instance.
  - In the **Region** field, type the region to use.
  - In the **Zone** field, type the zone your VM is located in. This is required for subscription matching to work. For more information about setting regions and zones, see [client-configuration:virtualization.pdf](#).
4. Click **Create** to save your changes and create the VHM.
  5. On the **Virtual Host Managers** page, select the new VHM.
  6. On the **Properties** page, click **Refresh Data** to inventory the new VHM.

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

Instances running on the Amazon public cloud report a UUID to the Uyuni Server in the format of an **i** followed by seventeen hexadecimal digits:

```
I1234567890abcdef0
```

### 10.1.2. AWS Permissions for Virtual Host Manager

For security reasons, always grant the least privilege possible for a task to be performed. Using an Access Key with excessive permissions for users connecting to AWS is not advised.

For SUSE Manager to gather the information required from AWS, the VHM needs permission to describe EC2 instances and addresses. One method to grant this is to create a new IAM user (Identity and Access Management) specific to this task, create a policy as follows and attach to the user:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeAddresses",
        "ec2:DescribeInstances"
      ],
      "Resource": "*"
    }
  ]
}
```

You can limit permissions more by restricting access to specific regions. For more information, see [https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ExamplePolicies\\_EC2.html#iam-example-read](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ExamplePolicies_EC2.html#iam-example-read)

---

only.

## 10.2. VHM and Azure

You can use a virtual host manager (VHM) to gather instances from Microsoft Azure.

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

### 10.2.1. Prerequisites

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

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

### 10.2.2. Create an Azure VHM

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

Ensure you have installed the [virtual-host-gatherer-libcloud](#) package on the Uyuni Server.

*Procedure: Creating an Azure VHM*

1. In the Uyuni Web UI, navigate to **Systems > Virtual Host Managers**.
2. Click **Create** and select **Azure** from the drop-down menu.
3. In the **Add an Azure Virtual Host Manager** section, use these parameters:
  - In the **Label** field, type a custom name for your VHM.
  - In the **Subscription ID** field, type the subscription ID found in [Azure portal > Services > Subscriptions](#) page.
  - In the **Application ID** field, type the application ID that you collected when you registered the application.
  - In the **Tenant ID** field, type the tenant ID provided by Azure that you collected when you registered the application.
  - In the **Secret Key** field, type the secret key associated with the Azure instance.
  - In the **Zone** field, type the zone your VM is located in. For example, for West Europe, enter **westeurope**. This is required for subscription matching to work.
4. Click **Create** to save your changes and create the VHM.
5. On the **Virtual Host Managers** page, select the new VHM.

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

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

### 10.2.3. Assigning permissions

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

```
General error: [AuthorizationFailed] The client 'client_name' with object id 'object_ID' does
not have authorization to perform action 'Microsoft.Compute/virtualMachines/read' over scope
'/subscriptions/not-very-secret-subscription-id' or the scope is invalid. If access was
recently granted, please refresh your credentials.
```

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

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

The **input\_azure.json** file should contain this information:

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

### 10.2.4. Azure UUID

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

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

## 10.3. VHM and Google Compute Engine

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

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

### 10.3.1. Prerequisites

The VHM you create needs to have the correct permissions assigned to access the GCE VM.

Log in to your Google Cloud Platform account as an administrator, and use the Cloud Identity and Access Management (IAM) tool to ensure that the service account has the appropriate roles.

### 10.3.2. Create a GCE VHM

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

To run a VHM, your Uyuni Server needs to have port 443 open, to access the clients.

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

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

*Procedure: Creating a GCE VHM*

1. In the Uyuni Web UI, navigate to **Systems > Virtual Host Managers**.
2. Click **Create** and select **Google Compute Engine** from the drop-down menu.
3. In the **Add a Google Compute Engine Virtual Host Manager** section, use these parameters:
  - In the **Label** field, type a custom name for your VHM.
  - In the **Service Account Email** field, type the email address associated with your service account.
  - In the **Cert Path** field, type the local path on the Uyuni Server to the key that you downloaded from the GCE panel.
  - In the **Project ID** field, type the project ID used by the GCE instance.
  - In the **Zone** field, type the zone your VM is located in. This is required for subscription matching to work.
4. Click **Create** to save your changes and create the VHM.
5. On the **Virtual Host Managers** page, select the new VHM.
6. On the **Properties** page, click **Refresh Data** to inventory the new VHM.

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

### 10.3.3. Assigning Permissions

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

## host-gatherer:

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

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

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

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

```
[  
  {  
    "id": "google_vhm",  
    "module": "GoogleCE",  
    "service_account_email": "mail@example.com",  
    "cert_path": "secret-key",  
    "project_id": "project-id",  
    "zone": "zone"  
  }  
]
```

### 10.3.4. GCE UUID

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

```
152986662232938449
```

## 10.4. VHM and Kubernetes

You can use a virtual host manager (VHM) to manage Kubernetes clusters.

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

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

You also require:

- At least one Kubernetes cluster available on your network.
- The `virtual-host-gatherer-Kubernetes` package installed on the Uyuni Server.
- Kubernetes version 1.5.0 or higher.

- Docker version 1.12 or higher on the container build host.

### 10.4.1. Create a Kubernetes VHM

Kubernetes clusters are registered with Uyuni as a VHM.

You will need a **kubeconfig** file to register and authorize your Kubernetes cluster. You can get a **kubeconfig** file using the Kubernetes command line tool **kubectl config view --flatten=true** provides the configuration with the certificate files embedded as needed for the VHM.

*Procedure: Creating a Kubernetes VHM*

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

*Procedure: Viewing the Nodes in a Cluster*

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

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

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



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

### 10.4.2. Retrieve Image Runtime Data

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

The image list table contains three columns:

- **Revision:**

---

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

- **Runtime:**

Overall status of the running instances for each image in registered clusters.

- **Instances:**

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

The **Runtime** column displays one of these status messages:

- **All instances are consistent with SUSE Manager:**

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

- **Outdated instances found:**

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

- **No information:**

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

*Procedure: Building an Image*

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

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

*Procedure: Importing a Previously Deployed Image*

1. In the Uyuni Web UI, navigate to **Images > Image Stores**.
2. Add the registry that owns the image you want to import, if it is not already there.

3. Navigate to **Images > Image List** and click **Import**.
4. Complete the fields, select the image store you created, and click **Import**.

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

*Procedure: Rebuilding a Previously Deployed Image*

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

When the rebuild has successfully completed, the runtime status of the image is updated in the image list at **Images > Image List**. This shows that the instances are running a previous build of the image.



- You can only rebuild images if they were originally built with Uyuni. You cannot rebuild imported images.

*Procedure: Retrieving Additional Runtime Data*

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

### 10.4.3. Permissions and Certificates



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

The API calls from Uyuni are:

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

The minimum recommended permissions for Uyuni are:

- A ClusterRole to list all the nodes:

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

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

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

If `/pods` returns a 403 response, the entire cluster is ignored by Uyuni.

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

## 10.5. Virtualization with Nutanix

You can use Nutanix AHV virtual machines with Uyuni by setting up a virtual host manager (VHM). To begin, you need to set up a VHM on your Uyuni Server, and inventory the available VM hosts.

### 10.5.1. VHM Setup

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

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

To run a VHM, your Uyuni Server must have port 9440 open to access the Nutanix Prism Element API.

*Procedure: Creating a Nutanix VHM*

1. In the Uyuni Web UI, navigate to **Systems > Virtual Host Managers**.
2. Click **Create** and select **Nutanix AHV**.
3. In the **Add a Nutanix AHV Virtual Host Manager** section, use these parameters:
  - In the **Label** field, type a custom name for your VHM.
  - In the **Hostname** field, type the fully qualified domain name (FQDN) or host IP address.
  - In the **Port** field, type the Prism Element API port to use (for example, **9440**).
  - In the **Username** field, type the username associated with the VM host.
  - In the **Password** field, type the password associated with the VM host user.
4. Click **Create** to save your changes and create the VHM.
5. On the **Virtual Host Managers** page select the new VHM.
6. On the **Properties** page, click **Refresh Data** to inventory the new VHM.

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

## Systems.



Connecting to the Nutanix Prism API server from a browser using HTTPS can sometimes log an **invalid certificate** error. If this occurs, refreshing the data from the virtual host manager fails. A valid SSL certificate (not self-signed) is required on your Nutanix API server. If you are using a custom CA authority for your Nutanix SSL certificate, copy the custom CA certificate to **/etc/pki/trust/anchors** on the Uyuni Server.

Re-trust the certificate by running the **update-ca-certificates** command on the command line, and restart the spacewalk services.

After your VHM has been created and configured, Taskomatic runs data collection automatically. If you want to manually perform data collection, navigate to **Systems > Virtual Host Managers**, select the appropriate VHM, and click **Refresh Data**.

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

## 10.6. Virtualization with VMWare

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

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

### 10.6.1. VHM Setup

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

To run a VHM, your Uyuni Server needs to have port 443 open, to access the VMWare API.

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

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

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

*Procedure: Creating a VMWare VHM*

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

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



Connecting to the ESXi server from a browser using HTTPS can sometimes log an **invalid certificate** error. If this occurs, refreshing the data from the virtual hosts server fails. To correct the problem, extract the certificate from the ESXi server, and copy it to **/etc/pki/trust/anchors**. Re-trust the certificate by running the **update-ca-certificates** command on the command line, and restart the spacewalk services.

After your VHM has been created and configured, Taskomatic runs data collection automatically. If you want to manually perform data collection, navigate to **Systems > Virtual Host Managers**, select the appropriate VHM, and click **Refresh Data**.

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

## 10.6.2. Troubleshooting SSL Errors on VMWare

If you see SSL errors while configuring your VMWare installation, you need to download the CA certificate file from VMWare, and trust it on Uyuni.

### *Procedure: Trusting VMWare CA Certificates*

1. Download the CA Certificate from your VMWare installation. You can do this by logging in to your vCenter Web UI, and clicking **Download trusted root CA certificates**.

2. If the downloaded CA certificates file is in **.zip** format, extract the archive. The certificate files have a number as an extension. For example, **certificate.0**.
3. Copy the certificate files to your Uyuni Server, and save them to the **/etc/pki/trust/anchors/** directory.
4. Change the filename suffix on the copied certificate to either **.crt** or **.pem**.
5. On the Uyuni Server, at the command prompt, update the CA certificate record:

```
update-ca-certificates
```

## 10.7. Virtualization with Other Third Party Providers

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

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



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

### *Procedure: Exporting and Importing a JSON File*

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

### *Listing 3. Example: Exported JSON configuration file:*

```
{
  "examplevh": {
    "10.11.12.13": {
      "cpuArch": "x86_64",
      "cpuDescription": "AMD Opteron(tm) Processor 4386",
      "cpuMhz": 3092.212727,
      "cpuVendor": "amd",
      "hostIdentifier": "'vim.HostSystem:host-182'",
```

```
        "name": "11.11.12.13",
        "os": "VMware ESXi",
        "osVersion": "5.5.0",
        "ramMb": 65512,
        "totalCpuCores": 16,
        "totalCpuSockets": 2,
        "totalCpuThreads": 16,
        "type": "vmware",
        "vms": {
            "vCenter": "564d6d90-459c-2256-8f39-3cb2bd24b7b0"
        }
    },
    "10.11.12.14": {
        "cpuArch": "x86_64",
        "cpuDescription": "AMD Opteron(tm) Processor 4386",
        "cpuMhz": 3092.212639,
        "cpuVendor": "amd",
        "hostIdentifier": "'vim.HostSystem:host-183'",
        "name": "10.11.12.14",
        "os": "VMware ESXi",
        "osVersion": "5.5.0",
        "ramMb": 65512,
        "totalCpuCores": 16,
        "totalCpuSockets": 2,
        "totalCpuThreads": 16,
        "type": "vmware",
        "vms": {
            "49737e0a-c9e6-4ceb-aef8-6a9452f67cb5": "4230c60f-3f98-2a65-f7c3-600b26b79c22",
            "5a2e4e63-a957-426b-bfa8-4169302e4fdb": "42307b15-1618-0595-01f2-427ffccdd88e",
            "NSX-gateway": "4230d43e-aafe-38ba-5a9e-3cb67c03a16a",
            "NSX-l3gateway": "4230b00f-0b21-0e9d-dfde-6c7b06909d5f",
            "NSX-service": "4230e924-b714-198b-348b-25de01482fd9"
        }
    }
}
```

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

## **man virtual-host-gatherer**

The [README](#) file of that package provides background information about the [type](#) of a hypervisor, etc.:

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

The man page and the **README** file also contain example configuration files.

# Chapter 11. GNU Free Documentation License

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

## 0. PREAMBLE

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

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

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

## 1. APPLICABILITY AND DEFINITIONS

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

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

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

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

---

then there are none.

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

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

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

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

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

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

## 2. VERBATIM COPYING

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

---

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

### 3. COPYING IN QUANTITY

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

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

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

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

### 4. MODIFICATIONS

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

- A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
- C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

- 
- D. Preserve all the copyright notices of the Document.
  - E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
  - F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
  - G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
  - H. Include an unaltered copy of this License.
  - I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
  - J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
  - K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
  - L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
  - M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.
  - N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.
  - O. Preserve any Warranty Disclaimers.

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

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

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

---

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

## 5. COMBINING DOCUMENTS

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

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

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

## 6. COLLECTIONS OF DOCUMENTS

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

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

## 7. AGGREGATION WITH INDEPENDENT WORKS

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

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

## 8. TRANSLATION

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

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

## 9. TERMINATION

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

## 10. FUTURE REVISIONS OF THIS LICENSE

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

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

## ADDENDUM: How to use this License for your documents

Copyright (c) YEAR YOUR NAME.

Permission is granted to copy, distribute and/or modify this document  
under the terms of the GNU Free Documentation License, Version 1.2  
or any later version published by the Free Software Foundation;  
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.  
A copy of the license is included in the section entitled "GNU  
Free Documentation License".