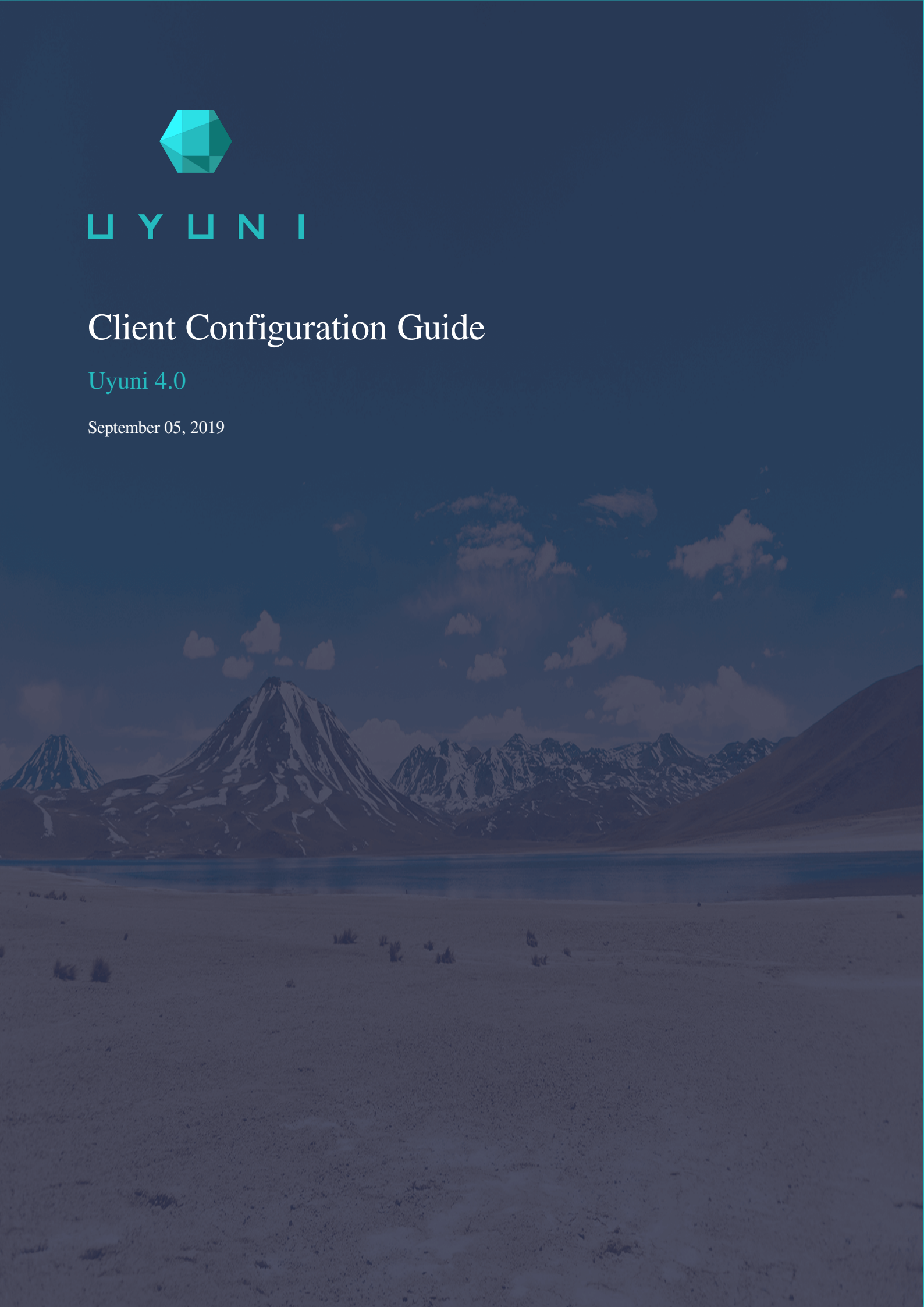


# Client Configuration Guide

## Uyuni 4.0

September 05, 2019

# Table of Contents

# Introduction

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

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

For a complete list of supported clients, see **[ Installation › Client-requirements › ]**.

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

## Compare Traditional and Salt Clients

This table lists the availability of various features for traditional and Salt clients. The icons in this table indicate:

- ✔ the feature is available
- ✖ the feature is not available.
- ? the feature is under consideration, and may or may not be made available at a later date

*Table 1. Features of Traditional and Salt Clients*

| Feature | Notes | Traditional | Salt |
|---|---|---|---|
| Registration | | ✔ | ✔ |
| Install Packages | | ✔ | ✔ |
| Apply Patches | | ✔ | ✔ |
| Remote Commands | | ✔ | ✔ |
| System Package States | | ✖ | ✔ |
| System Custom States | | ✖ | ✔ |
| Group Custom States | | ✖ | ✔ |
| Organization Custom States | | ✖ | ✔ |
| System Set Manager | | ✔ | ✔ |
| Service Pack Migration | | ✔ | ✔ |
| Virtual Host Management | | ✔ | ✔ |
| Virtual Guest Installation | | ✔ | ? |
| System Redeployment | With Auto-installation | ✔ | ? |

| Feature | Notes | Traditional | Salt |
|---|---|---|---|
| Contact Methods | Between server and client | osad, rhnsd, ssh push | zeromq (Salt default), salt-ssh |
| SUSE Manager Proxy | | ✔ | ✔ |
| Action Chains | | ✔ | ✔ |
| Software Crash Reporting | | ✔ | ✖ |
| Duplicate Package Reporting | | ✔ | ✔ |
| SCAP Auditing | | ✔ | ✔ |
| Support for Multiple Organizations | | ✔ | ✔ |
| Package Verification | | ✔ | ? |
| System Locking | | ✔ | ? |
| Configuration File Management | | ✔ | ✔ |
| Snapshots and Profiles | | ✔ | ? (Profiles are supported, sync is not) |
| Power Management | | ✔ | ✔ |

# Client Registration Overview

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

For Salt clients, the simplest method is to register your clients using the Uyuni Web UI.

If you need to have more control over the process, want to register many clients, or are registering traditional clients, we recommend creating a bootstrap script.

Both methods are described in this manual.

## Registering Clients with the Web UI

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

*Procedure: Registering clients in the Web UI*

1. In the Uyuni Web UI, navigate to **Systems › Bootstrapping**.
2. In the `Host` field, type the fully qualified domain name (FQDN) of the client to be bootstrapped.
3. In the `SSH Port` field, type the SSH port number to use to connect and bootstrap the client. By default, the SSH port is `22`.
4. In the `User` field, type the username to log in to the client. By default, the username is `root`.
5. In the `Password` field, type password to log in to the client.
6. In the `Activation Key` field, select the activation key that is associated with the software channel you want to use to bootstrap the client.
7. 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.
8. OPTIONAL: Check the `Manage System Completely via SSH` checkbox. If you check this option, the client will be configured to use SSH for its connection to the Server, and no other connection method will be configured.
9. Click **`[Bootstrap]`** to begin registration. When the bootstrap process has completed, your client will be listed at **Systems › System List**.

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

## Registering Clients with a Bootstrap Script

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

clients.

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

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

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

*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.

*SLES 15 and Python 3*

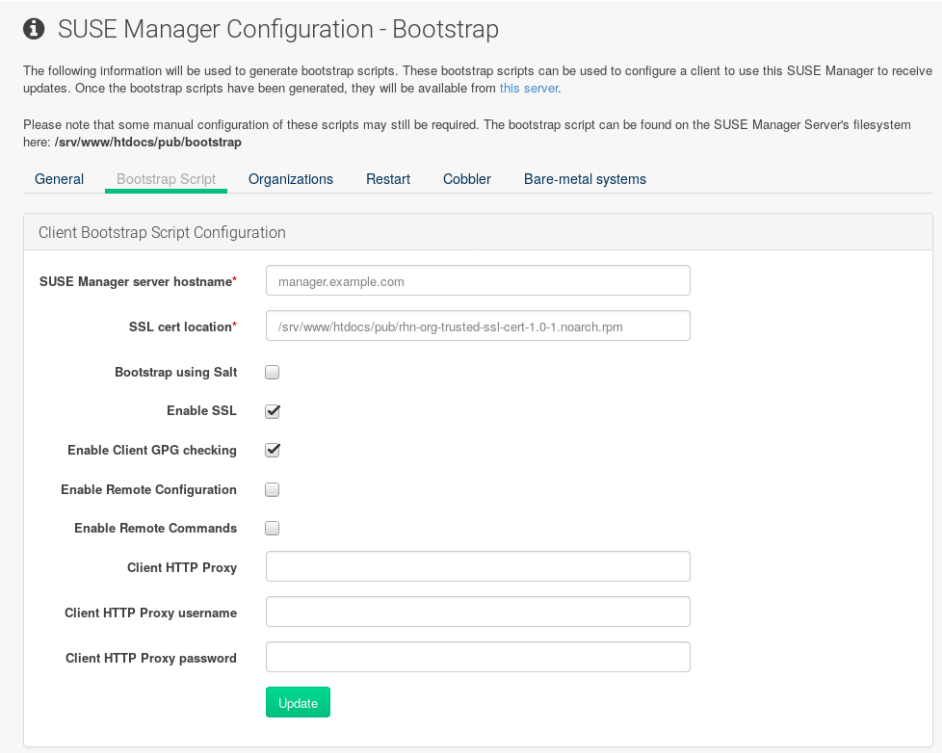
SLE 15 uses Python 3 by default. Bootstrap scripts based on Python 2 must be re-created for SLE 15 systems. Attempting to register SLE 15 systems using Python 2 bootstrap scripts will fail.

## Create a Bootstrap Script

This procedure describes how to generate a bootstrap script.

*Procedure: Creating a Bootstrap Script*

1. In the Uyuni Web UI, navigate to **Admin › Manager Configuration › Bootstrap Script**.
2. In the `SUSE Manager Configuration - Bootstrap` dialog, uncheck the `Bootstrap using Salt` checkbox if you are installing a traditional client. For Salt clients, leave it checked. Use default settings and click the **`[Update]`** button.




*Using SSL*

Unchecking **Enable SSL** in the Web UI or setting `USING_SSL=0` in the bootstrap script is not recommended. If you disable SSL nevertheless you will need to manage custom CA certificates to be able to run the registration process successfully.

3. A template bootstrap script is generated and stored on the server's file system in the `/srv/www/htdocs/pub/bootstrap` directory.

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

The bootstrap script is also available at `https://example.com/pub/bootstrap/bootstrap.sh`.

## 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 will also need to have trusted GPG keys on your system to install them.

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

**Overview** and click on `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.

*Procedure: Modifying a Bootstrap Script*

1. Login as root from the command line on your Uyuni server.
2. Navigate to the bootstrap directory with:

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

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

```
cp bootstrap.sh bootstrap-sles11.sh
cp bootstrap.sh bootstrap-sles12.sh
```

4. Open `sles12.sh` for modification. Scroll down and modify both lines marked in green. You must comment out `exit 1` with a hash mark (`#`) to activate the script and then enter the name of the key for this script in the `ACTIVATION_KEYS=` field as follows:

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

# can be edited, but probably correct (unless created during initial install):
# NOTE: ACTIVATION_KEYS *must* be used to bootstrap a client machine.
ACTIVATION_KEYS=1-sles12
ORG_GPG_KEY=
```

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

## Connect Clients

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

*Procedure: Running the Bootstrap Script*

1. On the Uyuni Server, log in as root at the command prompt, and navigate to this directory:

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

2. Run this command to execute the bootstrap script on the client:

```
cat MODIFIED-SCRIPT.SH | ssh root@example.com /bin/bash
```

The script will execute and proceed to download the required dependencies located in the

repositories directory you created earlier.

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

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

## Package Locks

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

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

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

*Procedure: Using Package Locks*

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

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

2. Navigate to the **Software › Packages › Lock** tab on the managed system to see a list of all available packages.
3. Select the packages to lock, and click **`[Request Lock]`**. You can also choose to enter a date and time for the lock to activate. Leave the date and time blank if you want the lock to activate as soon as possible. Note that the lock might not activate immediately.
4. To remove a package lock, select the packages to unlock and click **`[Request Unlock]`**. Leave the date and time blank if you want the lock to deactivate as soon as possible. Note that the lock might not deactivate immediately.

# Activation Keys

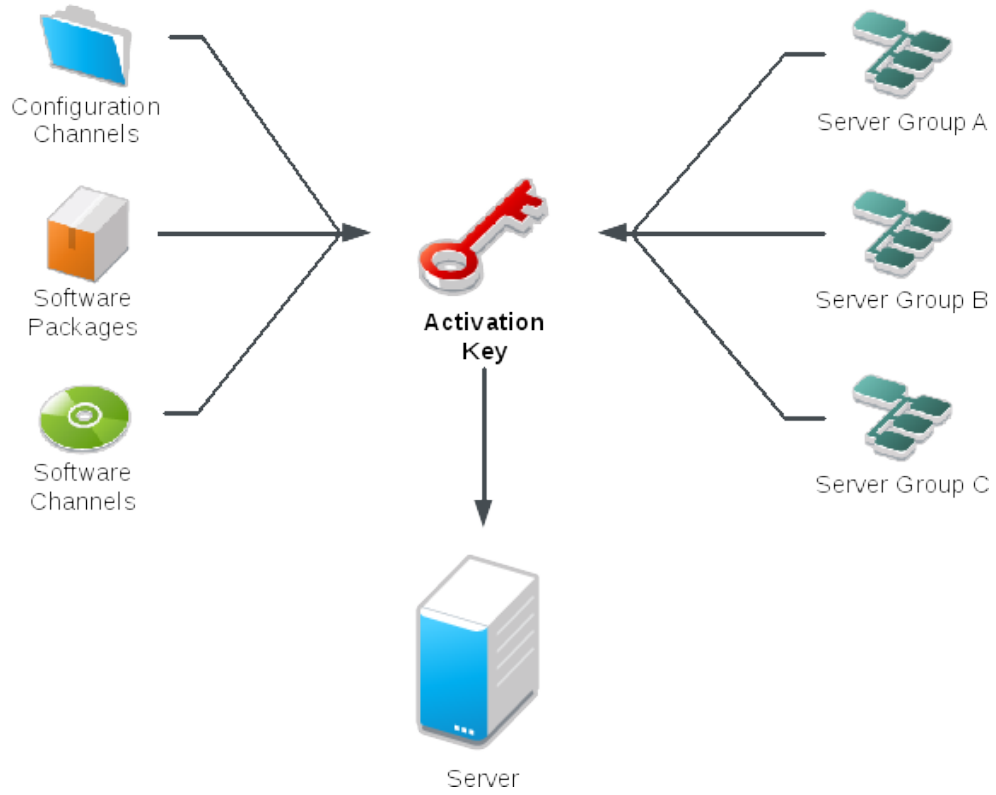
Activation keys are used with traditional and Salt clients to ensure that your clients have the correct software entitlements, are connecting to the appropriate channels, and are subscribed to the relevant groups. Each activation key is bound to an organization, which you can set when you create the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

In Uyuni, an activation key is a group of configuration settings with a label. You can apply all

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

An activation key can specify:

- Channel Assignment
- System Types (Traditionally called Add-on Entitlements)
- Contact Method
- Configuration Files
- Packages to be Installed
- System Group Assignment




*Procedure: Creating an Activation Key*

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

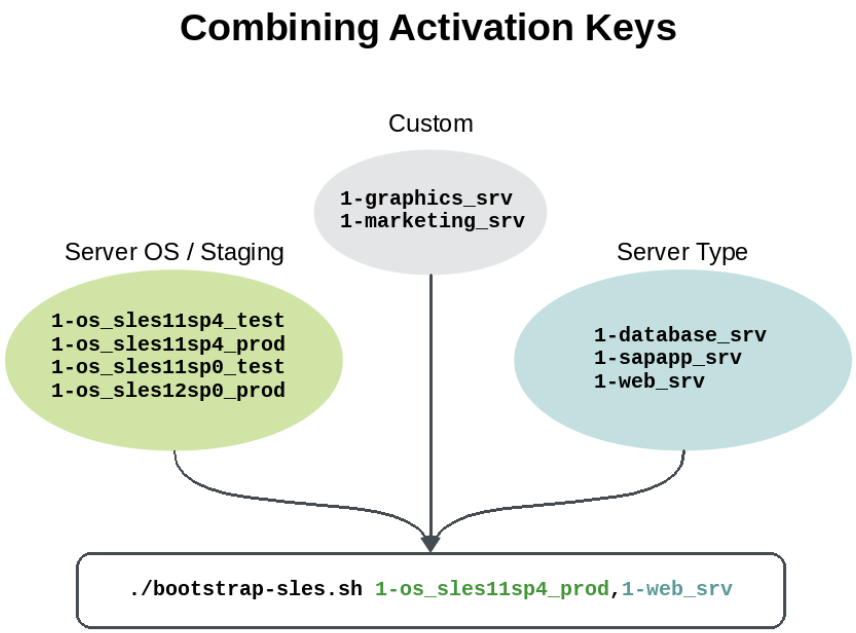
Do not use commas in the Key field for any SUSE products. However, you **must** use commas for Red Hat Products. For more information, see [ **Reference › System-details ›** ].

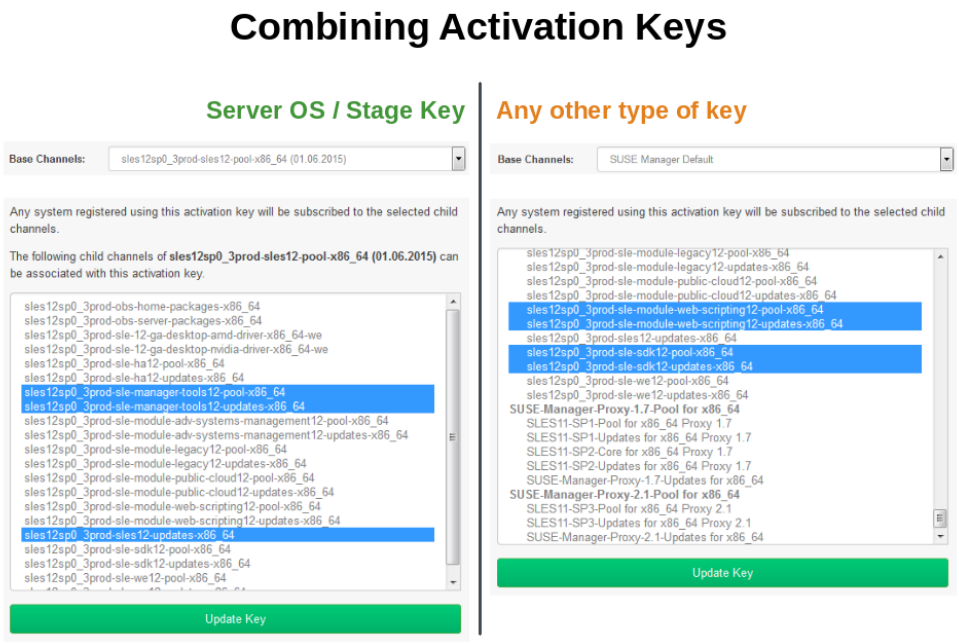
5. In the Base Channels drop-down box, select the appropriate base software channel, and allow the relevant child channels to populate.
6. Select the child channels you need (for example, the mandatory SUSE Manager tools and updates channels).
7. We recommend you leave the Contact Method set to Default.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

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

## Combining Activation Keys

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






## Activation Key Best Practices

### *Default Parent Channel*

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

### *Bootstrapping with Activation Keys*

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

### *Bandwidth Requirements*

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

These options create bandwidth usage:

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

### *Key Label Naming*

If you do not enter a human-readable name for your activation keys, the system will automatically

generate a number string, which can make it difficult to manage your keys.

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

When creating key labels, consider these tips:

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

Example activation key label names:

```
sles12-sp2-web_server-room_129-x86_64
```

```
sles12-sp2-test_packages-blg_502-room_21-ppc64le
```

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

*Included Channels*

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

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

- Channels to be included:
  - suse-manager-tools
- Typical packages to be included:
  - mgr-osad (pushing tasks)
    - Installs `python-jabberpy` and `pyxml` as dependencies

    - `mgr-cfg-actions` (Remote Command, Configuration Management)
        - Installs `mgr-cfg` and `mgr-cfg-client` as dependencies

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

Typical packages to be included:

- osad (pushing tasks): Installs `python-jabberpy` and `pyxml` as dependencies
- `rhncfg-actions` (Remote Command, Configuration Managment): Installs `rhncfg` and `rhncfg-client` as dependencies

# Registering Clients on a Proxy

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

This section contains information on registering Salt clients on a proxy using the Web UI, or with a bootstrap script.

Within the Web UI, proxy pages will show information about both Salt and traditional clients.

You can see a list of clients that are connected to a proxy by clicking on the name of the proxy in **Main Navigation › Systems › Systems › Proxy**, selecting the `Details` tab, and then selecting the `Proxy` tab.

A list of chained proxies for a Salt client can be seen by clicking on the name of the client in **Main Navigation › Systems › All**, selecting the `Details` tab, and then selecting the `Connection` tab.

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

## Registering with the Web UI on a Proxy

Registering Salt clients to a Uyuni Proxy using the Web UI is similar to registering clients directly with the Uyuni Server.

*Procedure: Registering clients to a proxy in the Web UI*

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

Instead of the Web UI, you can use the command line to register a Salt client through a proxy. This procedure requires that you have installed the Salt package on the Salt client before registration, and have activated the `Advanced` systems module.

*Procedure: Registering clients to a proxy using the command line*

1. Add the proxy FQDN as the master in the clients configuration file located at:

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

   or:

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

2. Add the FQDN to the minion file:

   ```
   master: proxy123.example.com
   ```

   Save and restart the salt-minion service:

   ```
   systemctl restart salt-minion
   ```

3. On the Server, accept the new client key with:

   ```
   salt-key -a 'client'
   ```

   The client connects to the proxy exclusively for Salt operations and normal HTTP package downloads.

## Registering with Bootstrap on a Proxy

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

*Procedure: Registering clients to a proxy with a bootstrap script*

1. Create a client activation key on the Uyuni server using the Web UI. See **[ Client-configuration › Clients-and-activation-keys › ]**.
2. On the proxy, execute the `mgr-bootstrap` command line tool as root. If needed, use the additional command line switches to tune your bootstrap script. To install a traditional client instead of a Salt client, ensure you use the `--traditional` switch.

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

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

3. Optional: edit the resulting bootstrap script.
4. Execute the bootstrap script on the clients.

# Introduction

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

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

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

The Cobbler installation server allows you to automate bare-metal installations. It uses DHCP to access a PXE boot server, and can be used in virtualized environments.

It is also possible to perform a disconnected setup, in an environment where you cannot access the internet.

## AutoYaSt

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

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

AutoYaST can be used to schedule a registered system to be installed with a new operating system and package profile, or you can use it to install a new system that was not previously registered, or does not yet have an operating system installed.

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

### Before you Begin

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

- A DHCP server is not required for AutoYaST, but it can make things easier. If you are using static IP addresses, you should select static IP while developing your AutoYaST profile.
- Host the AutoYaST distribution trees via HTTP, provided by Uyuni.
- If you are performing a bare metal AutoYaST installation, use these settings:
    - Configure DHCP to assign the required networking parameters and the bootloader program location.
    - In the bootloader configuration file, specify the kernel and appropriate kernel options to be used.

## Build a Bootable ISO

You will need to create a bootable ISO image to be used by the target system for installation. When the system is rebooted or switched on, it boots from the image, loads the 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.

This is managed by the KIWI image system. For more information about KIWI, see http://doc.opensuse.org/projects/kiwi/doc/.

## Integrate with PXE

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

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

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

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

For more information about PXE boot, see https://www.suse.com/documentation/sles-15/singlehtml/book_sle_deployment/book_sle_deployment.html#cha.deployment.prep_pxe.

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

# Kickstart

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

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

Kickstart can be used to schedule a registered system to be installed with a new operating system and package profile, or you can use it to install a new system that was not previously registered, or does not yet have an operating system installed.

For more information about Kickstart, see the Red Hat documentation.

## Before you Begin

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

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

## Build a Bootable ISO

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

*Building a Bootable ISO*

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

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

   IP address-based Kickstart URLs will look like this:

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

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

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

   ```
   ksdevice=eth0
   ```

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

4. Customize `isolinux.cfg` further as required. For example, you can add multiple options, different boot messages, or shorter timeout periods.
5. Create the ISO with this command:

```
mkisofs -o file.iso -b isolinux.bin -c boot.cat -no-emul-boot \
  -boot-load-size 4 -boot-info-table -R -J -v -T isolinux/
```

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

6. Burn the ISO to CD-ROM and insert the disk.
7. Boot the system and type `ks` at the prompt (if you left the label for the Kickstart boot as 'ks').
8. Press `Enter` to start Kickstart.

### Integrating with PXE

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

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

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

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

For more information about PXE boot, see https://www.suse.com/documentation/sles-15/singlehtml/book_sle_deployment/book_sle_deployment.html#cha.deployment.prep_pxe.

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

## Cobbler

Cobbler is an installation server that allows you to perform unattended system installations. It can be used on server, client, or guest systems, and in virtual environments.

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

- Installation environment analysis using the `cobbler check` command
- Multi-site installation server configuration using the `cobbler replicate` command
- Virtual machine guest installation automation with the `koan` client-side tool

- Building installation ISOs with PXE-like menus using the `cobbler buildiso` command (for Uyuni systems with x86_64 architecture)

For more detailed Cobbler documentation, see http://cobbler.github.io/manuals/.

SUSE only supports Cobbler functions that are available in the Uyuni Web UI, or through the Uyuni API. All features documented here are supported.

Cobbler is not currently supported within SUSE Manager for Retail environments. If you intend to use your installation with SUSE Manager for Retail formulas, do not configure Cobbler.

## Cobbler Requirements

To use Cobbler for system installation with PXE, you will require a TFTP server. Uyuni installs a TFTP server by default. To PXE boot systems, you will require a DHCP server, or have access to a network DHCP server. Edit the `/etc/dhcp.conf` configuration file to change `next-server` to the hostname or IP address of your Cobbler server.

Cobbler requires an open HTTP port to synchronize data between the Server and the Proxy. By default, Cobbler uses port 80, but you can configure it to use port 443 instead if that suits your environment.

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

## Configure Cobbler

Cobbler configuration is primarily managed using the `/etc/cobbler/settings` file. Cobbler will run with the default settings unchanged. All configurable settings are explained in detail in the `/etc/cobbler/settings` file, including information on each setting, and recommendations.

Cobbler uses DHCP to automate bare metal installations from a PXE boot server. You must have administrative access to the network's DHCP server, or be able to configure DHCP directly on the the Cobbler server.

If you have an existing DHCP server, you will need to edit the DHCP configuration file so that it points to the Cobbler server and PXE boot image.

*Procedure: Configuring DHCP for Cobbler*

1. On the DHCP server, as root, open the `/etc/dhcpd.conf` file.
2. Append a new class with options for performing PXE boot installation. For example:

```
allow booting;
allow bootp;
class "PXE"
{match if substring(option vendor-class-identifier, 0, 9) = "PXEClient";
next-server 192.168.2.1;
filename "pxelinux.0";}
```

This example:

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

3. Save the file.

*Procedure: Configuring PXE Boot in KVM*

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

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

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

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

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

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

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

Alternatively, you can use the `net-edit` subcommand, which will also perform some error checking.

*Listing 1. Example: Minimal Network XML Description for KVM*

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

## TFTP

Uyuni uses the `atftpd` daemon, but it can also use TFTP. The `atftpd` daemon is the recommended method for PXE services, and is installed by default. The default configuration works in most cases. However, if you need to change the configuration, do so using YaST Services Manager.

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

You can also synchronize Cobbler-generated TFTP contents to a Uyuni Proxy.

*Procedure: Installing TFTP*

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

```
zypper install susemanager-tftpsync
```

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

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

*Procedure: Configuring TFTP on a Proxy*

1. On the Uyuni Proxy, as root, run the `configure-tftpsync.sh` script.
2. The script will interactively ask you for details on the host names and IP addresses of the Uyuni Server and Proxy, as well for the location of the `tftpboot` directory on the Proxy.

For more information, use the `configure-tftpsync.sh --help` command.

*Procedure: Configuring TFTP on a Server*

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

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

2. Run the `cobbler sync` command to push the files to the proxy. This will fail if you have not configured the proxies correctly.
3. If you want to change the list of proxies later on, you can use the `configure-tftpsync.sh` script to edit them.

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

## Synchronize and Start the Cobbler Service

The Uyuni Server must be able to access the Uyuni Proxy systems directly. You cannot push using a proxy.

Before you start the Cobbler service, check that all the prerequisites are configured according to your requirements. You can do this by running the `cobbler check` command.

When your configuration is correct, start the Uyuni service:

```
/usr/sbin/spacewalk-service start
```

Do not start or stop the `cobblerd` service independent of the Uyuni service. Doing so can cause errors. Always use `/usr/sbin/spacewalk-service` to start or stop Uyuni.

## Kickstart Templates

Kickstart files are used to automate Red Hat Enterprise Linux client installations. Kickstart templates are used to describe how to create Kickstart files. To help with creating Kickstart templates, you can create Kickstart variables within the Uyuni Web UI. This allows you to create and manage large numbers of profiles and systems, without having to manually create Kickstart files for each.

Kickstart templates are shared by various profiles and systems that can each have their own variables and values. These variables modify the templates, and a template engine parses the template and variables into a usable Kickstart file.

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

For more information on creating Kickstart profile variables, see **[ Reference › Systems › ]**.

## Kickstart Template Syntax

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

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

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

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

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

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

Kickstart templates use syntax rules that rely on punctuation symbols. To avoid clashes, they need to be properly treated.

If the template contains shell script variables like `$(example)`, the content needs to be escaped with a backslash: `\$(example)`. If the variable is defined in the template, the templating engine will evaluate it correctly. If there is no such variable, the content will be left unchanged. Escaping the *$* symbol will prevent the templating engine from evaluating the symbol as an internal variable.

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

```
#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)
```

For more information about Kickstart templates and Cobbler, see https://fedorahosted.org/cobbler/wiki/KickstartTemplating

### Kickstart Snippets

Kickstart snippets are sections of Kickstart code that can be called by a `$SNIPPET()` function. The snippet is parsed by Cobbler and substituted with the contents of the snippet.

This example sets up a snippet for a common hard drive partition configuration:

```
clearpart --all
part /boot --fstype ext3 --size=150 --asprimary
part / --fstype ext3 --size=40000 --asprimary
part swap --recommended

part pv.00 --size=1 --grow

volgroup vg00 pv.00
logvol /var --name=var vgname=vg00 --fstype ext3 --size=5000
```

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

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

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

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

## Build ISOs with Cobbler

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

Building ISOs with Cobbler is not supported on IBM Z.

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

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

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

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

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

## Bare Metal Provisioning

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

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

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

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

### Enable Bare Metal Systems Management

Bare metal systems management can be enabled or disabled in the Uyuni Web UI by navigating to **Admin › SUSE Manager Configuration › Bare-metal systems**.

New systems are added to the organization of the administrator who enabled the bare metal systems management feature. To change the organization, log in as an Administrator of the required organization, and re-enable the feature.

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

### Provision Bare Metal Systems

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

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

## Disconnected Setup

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

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

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

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

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

For more information on RMT, see https://www.suse.com/documentation/sles-15/book_rmt/data/book_rmt.html.

### Synchronize RMT

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

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

*Procedure: Setting up RMT*

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

```
zypper in rmt-server
```

2. Configure RMT using YaST:

```
yast2 rmt
```

3. Follow the prompts to complete installation. For more information on setting up RMT, see

https://www.suse.com/documentation/sles-15/book_rmt/data/book_rmt.html.

*Procedure: Synchronizing RMT with SCC*

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

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

2. Synchronize all available updates for your organization:

```
rmt-cli sync
```

   You can also configure RMT to synchronize regularly using systemd.

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

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

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

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

5. Export the enabled repositories to your removable storage:

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

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

## Synchronize SMT

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

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

For more details on installing and configuring SMT, see https://www.suse.com/documentation/sles-12/book_smt/data/book_smt.html.

*Procedure: Synchronizing SMT with SCC*

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

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

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

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

3. Export the enabled repositories to your removable storage:

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

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

## Synchronize a Disconnected Server

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

*Procedure: Synchronizing a Disconnected Server*

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

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

3. Restart the Tomcat service:

```
systemctl restart tomcat
```

4. Refresh the local data:

```
mgr-sync refresh
```

5. Perform a synchronization:

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

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

# Other Clients

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

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

## Registering Red Hat Enterprise Linux Clients

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

### Set up a Red Hat Enterprise Linux Client

Ensure that your client meets these requirements before you start:

- 8 GB RAM or more
- Two or more physical or virtual CPU cores
- Access to Red Hat Enterprise Linux installation and subscription media
- An LVM or an NFS mount is recommended

You will need to ensure you provision enough disk space. The `/var/spacewalk` directory contains all mirrored RPMs, and can take a significant amount of disk space. For example, the Red Hat Enterprise Linux 6 x86_64 channels require over 90 GB.

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

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

You are responsible for arranging access to Red Hat base media repositories and RHEL installation media. You must obtain support from either Red Hat or SUSE for all your RHEL systems. If you do not do this, you might be violating your terms with Red Hat.

### Red Hat Enterprise Linux Channel Management

The base Red Hat Enterprise Linux software channel does not contain any packages. This is because SUSE does not provide Red Hat Enterprise Linux base media. You will need to obtain base media from Red Hat, which you can then add as a child channel to the Red Hat Enterprise Linux parent channel.

The Red Hat Enterprise Linux and tools channels are provided by SUSE Customer Center. You can synchronize your client with the `mgr-sync` command to get them.

Because the Red Hat Enterprise Linux channels are particularly large, it can take up to 24 hours for an initial channel synchronization to complete. When you have completed the initial synchronization, we recommended you clone the channel before working with it. This provides you with a backup of the original synchronization data.

The following procedure guides you through setup of the Red Hat Enterprise Linux media as a Uyuni channel. All packages on the media will be mirrored into a child channel located under the distribution name and architecture.

*Procedure: Setting up a Red Hat Enterprise Linux Channel*

1. In the Uyuni Web UI, navigate to **Channels › Manage Software Channels**, and click **[Create Channel]**.
2. Fill in the channel details, and add the channel as a child to the corresponding Red Hat Enterprise Linux distribution channel for your architecture. The base parent channel will not contain any packages.
3. Modify the associated activation key to include your new child channel.
4. At the command prompt, as root, copy your installation disk image to the `/tmp` directory.
5. Create a directory to contain the media content:

```
mkdir -p /srv/www/htdocs/pub/rhel
```

6. Mount the ISO:

```
mount -o loop /tmp/name_of_iso /srv/www/htdocs/pub/rhel
```

7. Synchronize the packages with `spacewalk-repo-sync`:

   For Red Hat Enterprise Linux 7:

```
spacewalk-repo-sync -c channel_name -u https://127.0.0.1/pub/rhel/
Repo URL: https://127.0.0.1/pub/rhel/
Packages in repo:              [...]
Packages already synced:       [...]
Packages to sync:              [...]
[...]
```

   For Red Hat Enterprise Linux 6:

```
spacewalk-repo-sync -c channel_name -u https://127.0.0.1/pub/rhel/Server/
Repo URL: https://127.0.0.1/pub/rhel/Server/
Packages in repo:              [...]
Packages already synced:       [...]
Packages to sync:              [...]
[...]
```

Sometimes, the `spacewalk-repo-sync` will stop running during a synchronization, which will give this error:

```
[Errno 256] No more mirrors to try.
```

If this occurs, you can run `spacewalk-repo-sync` in debugging mode to determine the error.

Start debugging mode:

```
export URLGRABBER_DEBUG=DEBUG
```

Check the output:

```
/usr/bin/spacewalk-repo-sync --channel _<channel-label>_ --type yum
```

Disable debug mode:

```
unset URLGRABBER_DEBUG``
```

## Register Red Hat Enterprise Linux Clients

Before you register Red Hat Enterprise Linux clients to your Uyuni Server, check that you have the corresponding Red Hat Enterprise Linux product enabled, and the required channels are fully synchronized.

You will also need an activation key associated with the Red Hat Enterprise Linux channel. For more information on activation keys, see **[ Client-configuration › Clients-and-activation-keys › ]**.

Missing packages will cause your registration to fail. For Red Hat Enterprise Linux clients, packages are contained on the Red Hat Enterprise Linux installation media. Ensure you have loop-mounted the installation media, and added it as a child channel to the base Red Hat Enterprise Linux channel.

**Red Hat Enterprise Linux 7**

- Product: Red Hat Enterprise Linux 7
- Mandatory channels: `rhel-x86_64-server-7` , `res7-suse-manager-tools-x86_64` , `res7-x86_64`

**Red Hat Enterprise Linux 6**

- Product: Red Hat Enterprise Linux 6
- Mandatory channels: `rhel-x86_64-server-6` , `res6-suse-manager-tools-x86_64` , `res6-x86_64`

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

- In the UyuniWeb UI, navigate to **Admin › Setup Wizard** and select the **SUSE Products** tab. This dialog displays a completion bar for each product when they are being synchronized.
- You can also check the synchronization log file at the command prompt. Use the `cat` or `tail -f` command to view the `/var/log/rhn/reposync/channel-label.log` file. If you use this method, remember that base channels can contain multiple child channels. Each of the child channels will generate its own log during the synchronization progress. You will need to check all the base and child channel log files to be sure that the synchronization is complete.

When you are ready to register your Red Hat Enterprise Linux client, follow the instructions in **[ Client-configuration › Registration-overview › ]**.

## Registering {centos} Clients

This section contains information about registering traditional and Salt clients running {centos} operating systems.

### Set up a {centos} Client

The `spacewalk-utils` package contains a number of upstream command line tools required for client administration. You will also require the `spacewalk-common-channels` tool. Keep in mind SUSE only provides support for `spacewalk-clone-by-date` and `spacewalk-manage-channel-lifecycle` tools.

The `/etc/rhn/spacewalk-common-channels.ini` configuration file must contain the channel references you want to add. If a channel is not listed, check the latest version for updates: https://github.com/spacewalkproject/spacewalk/tree/master/utils

You will also need an activation key associated with the {centos} channel. For more information on activation keys, see **[ Client-configuration › Clients-and-activation-keys › ]**.

*Procedure: Adding Channels and Repositories*

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

```
zypper in spacewalk-utils
```

2. Add the {centos} base, updates, and client channels using the `spacewalk-common-channels` script:

```
spacewalk-common-channels -u admin -p`secret`-a x86_64 'centos7'
spacewalk-common-channels -u admin -p`secret`-a x86_64 'centos7-updates'
spacewalk-common-channels -u admin -p`secret`-a x86_64 'centos7-uyuni-client-x86_64'
```

*Procedure: Synchronizing {centos} Clients*

1. In the Uyuni Web UI, navigate to **Main Menu › Software › Manage**, and select the base channel you want to synchronize.
2. In the `Repositories` tab, navigate to the `Sync` subtab, and click **`[Sync Now]`**. You can also create a regular synchronization schedule on this page.

When you have prepared you Uyuni Server, you can install your {centos} client using your preferred installation media and method.

*Procedure: Setting up a {centos} Client*

1. At the command prompt, copy all relevant GPG keys to `/srv/www/htdocs/pub`. If you intend to use a bootstrap script to register your client, you can add the GPG keys to your bootstrap script.
2. Check that the client machine can resolve itself and your Uyuni Server using DNS.
3. Check that there is an entry in `/etc/hosts` for the real IP address of the client.
4. Create an activation key called `centos7` on the Uyuni Server that points to the correct parent and child channels, including the {centos} base repository, updates, and client channels.

When you are ready to register your {centos} client, follow the instructions in **[ Client-configuration › Registration-overview › ]**.

## Registering Ubuntu Clients

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

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

Supported features:

- Bootstrapping
- Synchronizing `.deb` channels
- Assigning `.deb` channels to clients
- GPG signing `.deb` repositories
- Information displayed in System details pages
- Package install, update, and remove
- Package install using `Package States`
- Configuration and state channels

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 in order to use bootstrapping, you will require an existing user with `sudo` privileges for Python.

Some actions are not yet supported:

- Patch and errata support
- Bare metal installations, PXE booting, and virtual host provisioning
- Live patching
- CVE Audit
- If you use are using a repository from storage media (`server.susemanager.fromdir = ...` option in rhn.conf), Ubuntu Client Tools will not work.

Canonical does not endorse or support SUSE Manager.

## Prepare to Register Ubuntu Clients

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

*Procedure: Adding the Ubuntu Channels*

1. Install the `spacewalk-utils` package:

```
sudo zypper in spacewalk-utils
```

2. At the command prompt on the Uyuni Server, as root, add the Ubuntu channels:

```
sudo spacewalk-common-channels ubuntu-1804-pool-amd64-uyuni ubuntu-1804-amd64-main-uyuni
\
ubuntu-1804-amd64-main-update-uyuni ubuntu-1804-amd64-main-security-uyuni \
ubuntu-1804-amd64-universe-uyuni ubuntu-1804-amd64-uyuni-client
```

3. Synchronize the new custom channels. You can check the progress of your synchronization from the command line with this command:

```
tail -f /var/log/rhn/reposync.log /var/log/rhn/reposync/*
```

4. To use bootstrap with Ubuntu, you will need to create a bootstrap repository. You can do this from the command line with `mgr-create-bootstrap-repo`:

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

For more information on creating custom repositories, see **[ Administration › Channel-management › ]**.