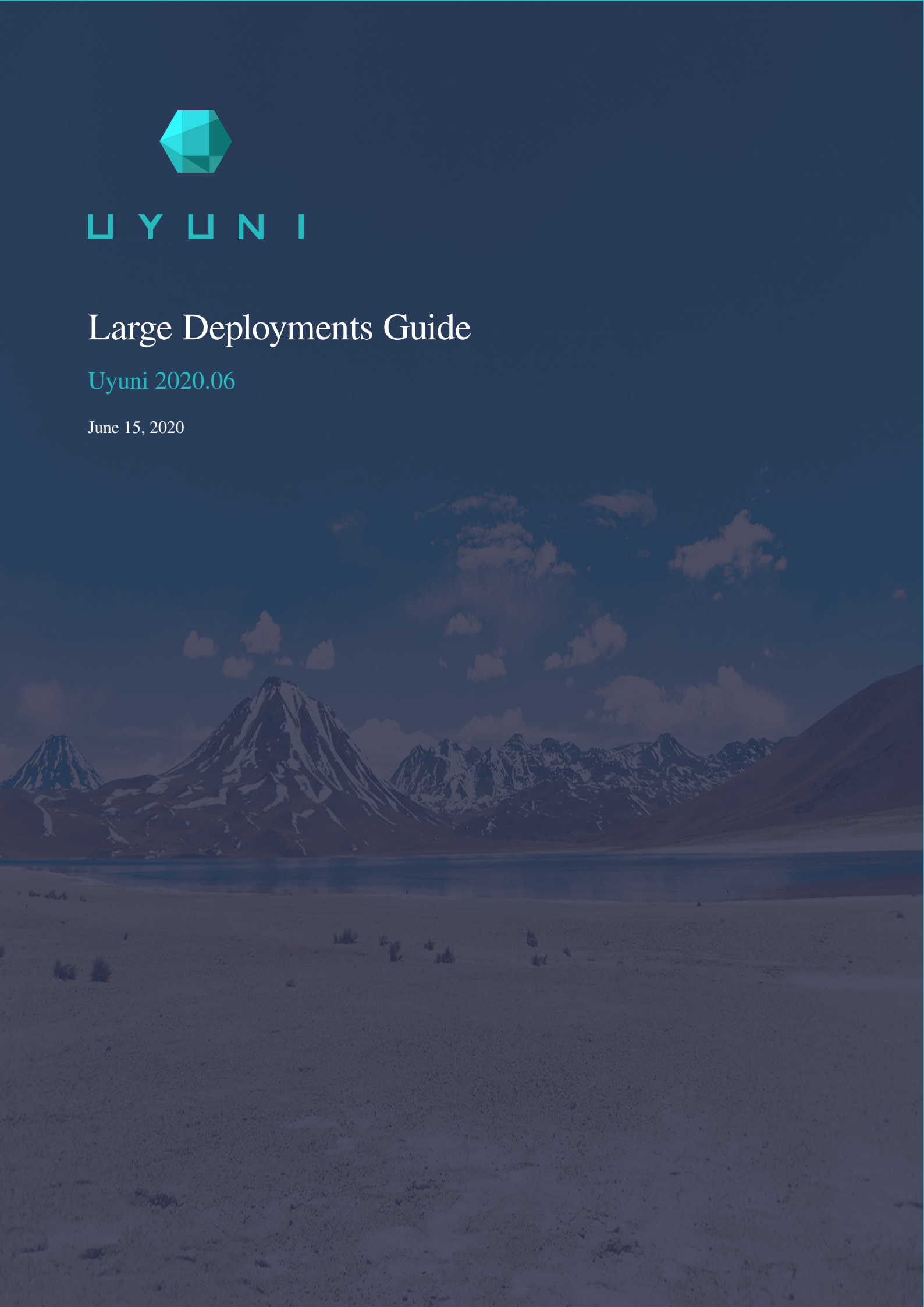


Large Deployments Guide

Uyuni 2020.06

June 15, 2020

Table of Contents

GNU Free Documentation License 1
Introduction 8
Hardware Requirements 8
Using a Single Server to Manage Large Scale Deployments 10
Operation Recommendations 10
Salt Client Onboarding Rate 10
Salt Clients and the RNG 10
Clients Running with Unaccepted Salt Keys 10
Disabling the Salt Mine 11
Disable Unnecessary Taskomatic jobs 11
Swap and Monitoring 12
AES Key Rotation 12
Using Multiple Servers to Manage Large Scale Deployments 13
Hub Requirements 13
Hub Installation 13
Using the Hub API 14
Hub XMLRPC API Namespaces 14
Hub XMLRPC API Authentication Modes 15
Authentication Examples 15
Managing Large Scale Deployments in a Retail Environment 19
Tuning Large Scale Deployments 20
The Tuning Process 20
Environmental Variables 22
Parameters 22
`MaxClients` 22
`ServerLimit` 23
`maxThreads` 23
Tomcat's `-Xmx` 24
`java.message_queue_thread_pool_size` 24
`java.salt_batch_size` 25
`java.salt_presence_ping_timeout` 25
`java.salt_presence_ping_gather_job_timeout` 26
`java.taskomatic_channel_repodata_workers` 27
`taskomatic.java.maxmemory` 27
`org.quartz.threadPool.threadCount` 28
`org.quartz.scheduler.idleWaitTime` 28
`MinionActionExecutor.parallel_threads` 29
`hibernate.c3p0.max_size` 29
`rhn-search.java.maxmemory` 29
`shared_buffers` 30
`max_connections` 30
`work_mem` 31
`effective_cache_size` 31
`thread_pool` 32
`worker_threads` 32

swappiness 33
Memory Usage 34

GNU Free Documentation License

0. PREAMBLE

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

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

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

1. APPLICABILITY AND DEFINITIONS

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

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

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

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

then there are none.

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

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

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

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

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

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

2. VERBATIM COPYING

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

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

3. COPYING IN QUANTITY

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

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

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

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

4. MODIFICATIONS

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

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

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

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

D. Preserve all the copyright notices of the Document.

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

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

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

H. Include an unaltered copy of this License.

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

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

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

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

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

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

O. Preserve any Warranty Disclaimers.

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

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

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

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

5. COMBINING DOCUMENTS

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

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

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

6. COLLECTIONS OF DOCUMENTS

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

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

7. AGGREGATION WITH INDEPENDENT WORKS

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

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

8. TRANSLATION

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

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

9. TERMINATION

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

10. FUTURE REVISIONS OF THIS LICENSE

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

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

ADDENDUM: How to use this License for your documents

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

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

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

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

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

Introduction

Publication Date: 2020-06-15

Uyuni is designed by default to work on small and medium scale installations. For installations with more than 1000 clients per Uyuni Server, adequate hardware sizing and parameter tuning must be performed.

There is no hard maximum number of supported systems. Many factors can affect how many clients can reliably be used in a particular installation. Factors can include which features are used, and how the hardware and systems are configured.

Large installations require standard Salt clients. These instructions cannot be used in environments using traditional clients or Salt SSH minions.

There are two main ways to manage large scale deployments. You can manage them with a single Uyuni Server, or you can use multiple servers in a hub. Both methods are described in this book.

Additionally, if you are operating within a Retail environment, you can use SUSE Manager for Retail to manage large deployments of point-of-service terminals. There is an introduction to SUSE Manager for Retail in this book.

Tuning and monitoring large scale deployments can differ from smaller installations. This book contains guidance for both tuning and monitoring within larger installations.

Hardware Requirements

Not all problems can be solved with better hardware, but choosing the right hardware is an absolute necessity for large scale deployments.

The minimum requirements for the Uyuni Server are:

- Eight or more recent x86_64 CPU cores.
- 32 GiB RAM. For installations with thousands of clients, use 64 GB or more.
- Fast I/O storage devices, such as locally-attached SSDs. For PostgreSQL data directories, we recommend locally-attached RAID-0 SSDs.

If the Uyuni Server is virtualized, enable the `elevator=noop` kernel command line option, for the best input/output performance. You can check the current status with `cat /sys/block/<DEVICE>/queue/scheduler`. This command will display a list of available schedulers with the currently active one in brackets. To change the scheduler before a reboot, use `echo noop > /sys/block/<DEVICE>/queue/scheduler`.

The minimum requirements for the Uyuni Proxy are:

- One Uyuni Proxy per 500-1000 clients, depending on available network bandwidth.

- Two or more recent x86_64 CPU cores.
- 16 GB RAM, and sufficient storage for caching.

Clients should never be directly attached to the Uyuni Server in production systems.

In large scale installations, the Uyuni Proxy is used primarily as a local cache for content between the server and clients. Using proxies in this way can substantially reduce download time for clients, and decrease Server egress bandwidth use.

The number of clients per proxy will affect the download time. Always take network structure and available bandwidth into account.

We recommend you estimate the download time of typical usage to determine how many clients to connnect to each proxy. To do this, you will need to estimate the number of package upgrades required in every patch cycle. You can use this formula to calculate the download time:

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

For example, the total time needed to transfer 400 MB of upgrades through a physical link speed of 1 GB/s to 3000 clients:

```
400 MB  * 3000 / 119 MB/s / 60 = 169 min
```

Using a Single Server to Manage Large Scale Deployments

This section discusses how to set up a single Uyuni Server to manage a large number of clients. It contains some recommendations for hardware and networking, and an overview of the tuning parameters that you need to consider in a large scale deployment.

Operation Recommendations

This section contains a range of recommendations for large scale deployments.

Always start small and scale up gradually. Monitor the server as you scale to identify problems early.

Salt Client Onboarding Rate

The rate at which Uyuni can onboard clients is limited and depends on hardware resources. Onboarding clients at a faster rate than Uyuni is configured for will build up a backlog of unprocessed keys. This slows down the process and can potentially exhaust resources. We recommend that you limit the acceptance key rate programmatically. A safe starting point would be to onboard a client every 15 seconds. You can do that with this command:

```
for k in $(salt-key -l un|grep -v Unaccepted); do salt-key -y -a $k; sleep 15; done
```

Salt Clients and the RNG

All communication to and from Salt clients is encrypted. During client onboarding, Salt uses asymmetric cryptography, which requires available entropy from the Random Number Generator (RNG) facility in the kernel. If sufficient entropy is not available from the RNG, it will significantly slow down communications. This is especially true in virtualized environments. Ensure enough entropy is present, or change the virtualization host options.

You can check the amount of available entropy with the `cat /proc/sys/kernel/random/entropy_avail`. It should never be below 100-200.

Clients Running with Unaccepted Salt Keys

Idle clients which have not been onboarded, that is clients running with unaccepted Salt keys, consume more resources than idle clients that have been onboarded. Generally, this consumes about an extra 2.5 Kb/s of inbound network bandwidth per client. For example, 1000 idle clients will consume about 2.5 Mb/s extra. This consumption will reduce almost to zero when onboarding has been completed for all clients. Limit the number of non-onboarded clients for optimal performance.

Disabling the Salt Mine

In older versions, Uyuni used a tool called Salt mine to check client availability. The Salt mine would cause clients to contact the server every hour, which created significant load. With the introduction of a more efficient mechanism in Uyuni 3.2, the Salt mine is no longer required. Instead, the Uyuni Server uses Taskomatic to ping only the clients that appear to have been offline for twelve hours or more, with all clients being contacted at least once in every twenty four hour period by default. You can adjust this by changing the `web.system_checkin_threshold` parameter in `rhn.conf`. The value is expressed in days, and the default value is `1`.

Newly registered Salt clients will have the Salt mine disabled by default. If the Salt mine is running on your system, you can reduce load by disabling it. This is especially effective if you have a large number of clients.

Disable the Salt mine by running this command on the server:

```
salt '*' state.sls util.mgr_mine_config_clean_up
```

This will restart the clients and generate some Salt events to be processed by the server. If you have a large number of clients, handling these events could create excessive load. To avoid this, you can execute the command in batch mode with this command:

```
salt --batch-size 50 '*' state.sls util.mgr_mine_config_clean_up
```

You will need to wait for this command to finish executing. Do not end the process with `Ctrl`+`C`.

Disable Unnecessary Taskomatic jobs

To minimize wasted resources, you can disable non-essential or unused Taskomatic jobs.

You can see the list of Taskomatic jobs in the Uyuni Web UI, at **Admin › Task Schedules**.

To disable a job, click the name of the job you want to disable, select `Disable Schedule`, and click **[Update Schedule]**.

To delete a job, click the name of the job you want to delete, and click **[Delete Schedule]**.

We recommend disabling these jobs:

- Daily comparison of configuration files: `compare-configs-default`
- Hourly synchronization of Cobbler files: `cobbler-sync-default`
- Daily gatherer and subscription matcher: `gatherer-matcher-default`

Do not attempt to disable any other jobs, as it could prevent Uyuni from functioning correctly.

Swap and Monitoring

It is especially important in large scale deployments that you keep your Uyuni Server constantly monitored and backed up.

Swap space use can have significant impacts on performance. If significant non-transient swap usage is detected, you can increase the available hardware RAM.

You can also consider tuning the Server to consume less memory. For more information on tuning, see **[Salt › Large-scale-tuning ›]**.

AES Key Rotation

Communications from the Salt Master to clients is encrypted with a single AES key. The key is rotated when:

- The `salt-master` process is restarted, or
- Any minion key is deleted (for example, when a client is deleted from Uyuni)

After the AES key has been rotated, all clients must re-authenticate to the master. By default, this happens next time a client receives a message. If you have a large number of clients (several thousands), this can cause a high CPU load on the Uyuni Server. If the CPU load is excessive, we recommend that you delete keys in batches, and in off-peak hours if possible, to avoid overloading the server.

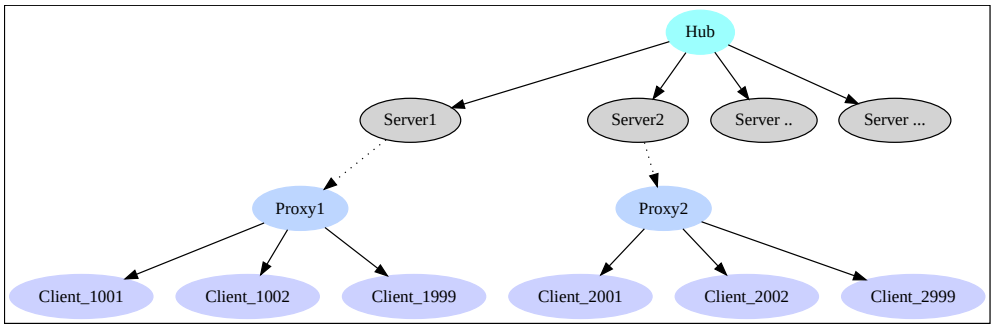
For more information, see:

- https://docs.saltstack.com/en/latest/topics/tutorials/intro_scale.html#too-many-minions-re-authing
- https://docs.saltstack.com/en/getstarted/system/communication.html

Using Multiple Servers to Manage Large Scale Deployments

If you need to manage a large number of clients, in most cases you can do so with a single Uyuni Server, tuned appropriately. However, if you need to manage tens of thousands of clients, you might find it easier to use multiple Uyuni Servers, in a hub, to manage them.

Uyuni Hub helps you manage very large deployments. The typical Hub topology looks like this:



Hub Requirements

To set up a Hub installation, you will require:

- One central Uyuni Server, which acts as the Hub Server.
- Two or more additional Uyuni Servers, registered to the Hub as Salt clients. This document refers to these as peripheral servers. Register peripheral servers to the Hub Server directly, do not use a proxy.
- Any number of clients registered to the peripheral servers.
- Ensure the Hub Server and all peripheral servers are running Uyuni 4.1 or higher.

The Hub Server must not have clients registered to it. Clients should only be registered to the peripheral servers.

You will need credentials to access the XMLRPC APIs on each server, including the Hub Server.

Hub Installation

Before you begin, you need to install the XMLRPC API package, and configure the Hub Server to use the API.

Procedure: Installing and Configuring the XMLRPC API

1. On the Hub Server, or on a host that has access to all peripheral servers' XMLRPC APIs, install the `hub-xmlrpc-api` package. The package is available in the Uyuni 2020.06 repositories.
2. Verify if the configuration file available at `/etc/hub/hub-xmlrpc-api-config.json` is pointing to the correct Hub instance and modify it as required.

3. OPTIONAL: In the configuration file, check the settings for the timeout parameters `connect_timeout` and `read_write_timeout`. The values for these parameters are defined in seconds.
4. Restart services to pick up the configuration changes.

Using the Hub API

The Hub XMLRPC API is a service that you can run on any machine using the systemd unit file `hub-xmlrpc-api.service`. You can launch the API from the command prompt:

```
systemctl start hub-xmlrpc-api
```

When it is running, connect to the service using port 8888 using any XMLRPC-compliant client libraries.

For examples, see **[Large-deployments › Hub-auth ›]**.

Logs are saved in `/var/logs/hub/hub-xmlrpc-api.log`. Logs are rotated weekly, or when the log file size reaches the specified limit. By default, the log file size limit is 10 MB.

Hub XMLRPC API Namespaces

Hub XMLRPC API operates in a similar way to the Uyuni API. For Uyuni API documentation, see https://documentation.suse.com/suma.

The Hub XMLRPC API exposes the same methods that are available from the server's XMLRPC APIs, with a few differences in parameter and return types. Additionally, the Hub XMLRPC API supports some Hub-specific end points which are not available in the Uyuni API.

Hub supports three different namespaces:

- The `hub` namespace is used to target the Hub XMLRPC API Server. It supports all the methods that are available on a Uyuni Server XMLRPC API as well as Hub-specific XMLRPC endpoints. Hub-specific methods are primarily related to authentication.
- The `unicast` namespace is used to target a single peripheral server registered in the hub. It redirects any call transparently to one specific server and returns any value as if the server's XMLRPC API endpoint was used directly.
- The `multicast` namespace is used to target multiple peripheral servers registered in the hub. It redirects any call transparently to all the specified servers and returns the results in the form of a `map`.

Some important considerations for hub namespaces:

- Individual server IDs can be obtained using `client.hub.listServerIds(hubKey)`.

- The `unicast` namespace assumes all methods receive `hubKey` and serverID as their first two parameters, then any other parameter as specified by the regular Server API.

```
client.unicast.[namespace].[methodname](hubKey, serverId, param1, param2)
```

- You can obtain `hubKey` using different authentication methods. For more information, see **[Large-deployments › Hub-auth ›]**.
- The `multicast` namespace assumes all methods receive `hubKey`, a list of ServerIDs, then lists of per-server parameters as specified by the regular server API. The return value is a map, with `Successful` and `Failed` entries.

```
client.multicast.[namespace].[methodname](hubKey, [serverId1, serverId2], [param1_s1,
param1_s2], [param2_s1, param2_s2])
```

Hub XMLRPC API Authentication Modes

Hub supports three different authentication modes:

- Manual mode (default): API credentials must be explicitly provided for each server.
- Relay mode: the credentials used to authenticate with the Hub are also used to authenticate to each server. You must provide a list of servers to connect to.
- Auto-connect mode: credentials are reused for each server, and any peripheral server you have access to is automatically connected.

Authentication Examples

This section provides examples of each authentication method.

Example: Manual Authentication

In manual mode, credentials have to be explicitly provided for each peripheral server before you can connect to it.

A typical workflow for manual authentication is:

1. Credentials for the Hub are passed to the `login` method, and a session key for the Hub is returned (`hubKey`).
2. Using the session key from the previous step, Uyuni Server IDs are obtained for all the peripheral servers attached to the Hub via the `hub.listServerIds` method
3. Credentials for each peripheral server are provided to the `attachToServers` method. This performs authentication against each server's XMLRPC API endpoint.
4. A `multicast` call is performed on a set of servers. This is defined by `serverIds`, which contains

the IDs of the servers to target. In the background, `system.list_system` is called on each server's XMLRPC API

5. Hub aggregates the results and returns the response in the form of a `map`. The map has two entries:
    - `Successful`: list of responses for those clients where the call succeeded.
    - `Failed`: list of responses for those clients where the call failed.

If you want to call a method on just one Uyuni Server, then Hub API also provides a `unicast` namespace. In this case, the response will be a single value and not a map, in the same way as if you called that Uyuni server's API directly.

Listing 1. Example Python Script for Manual Authentication:

```
#!/usr/bin/python
import xmlrpclib

HUB_URL = "<HUB_URL>"
HUB_LOGIN = "<LOGIN>"
HUB_PASSWORD = "<PASSWORD>"

client = xmlrpclib.Server(HUB_URL, verbose=0)

hubKey = client.hub.login(HUB_LOGIN, HUB_PASSWORD)

# Get the server IDs
serverIds = client.hub.listServerIds(hubKey)

# Authenticate each server with its own credentials
usernames = [HUB_LOGIN for s in serverIds]
passwords = [HUB_PASSWORD for s in serverIds]
client.hub.attachToServers(hubKey, serverIds, usernames, passwords)

# Perform the operation
systemsPerServer = client.multicast.system.list_systems(hubKey, serverIds)
successfulResponses = systemsPerServer["Successful"]["Responses"]
failedResponses = systemsPerServer["Failed"]["Responses"]

for system in successfulResponses:
  print (system)

#logout
client.auth.logout(hubKey)
```

Example: Relay Authentication

In relay authentication mode, the credentials used to sign in to the Hub API are also used to sign in into the APIs of the peripheral servers the user wants to work with. In this authentication mode, it is assumed that the same credentials are valid for every server, and that they correspond to a user with appropriate permissions.

After signing in, you must call the `attachToServers` method. This method defines the servers to target in all subsequent calls.

A typical workflow for relay authentication is:

1. Credentials for the Hub are passed to the `loginWithAuthRelayMode` method, and a session key for the Hub is returned (`hubKey`).
2. Using the session key from the previous step, Uyuni Server IDs are obtained for all the peripheral servers attached to the Hub via the `hub.listServerIds` method
3. A call to `attachToServers` is made, and the same credentials used to sign in to the Hub are passed to each server. This performs authentication against each server's XMLRPC API endpoint.
4. A `multicast` call is performed on a set of servers. This is defined by `serverIds`, which contains the IDs of the servers to target. In the background, `system.list_system` is called on each server's XMLRPC API.
5. Hub aggregates the results and returns the response in the form of a `map`. The map has two entries:
    - `Successful`: list of responses for those clients where the call succeeded.
    - `Failed`: list of responses for those clients where the call failed.

Listing 2. Example Python Script for Relay Authentication:

```
#!/usr/bin/python
import xmlrpclib


HUB_URL = "http://localhost:8888/hub/rpc/api"
HUB_LOGIN = "admin"
HUB_PASSWORD = "admin"

client = xmlrpclib.Server(<HUB_URL>, verbose=0)

hubKey = client.hub.loginWithAuthRelayMode(<HUB_LOGIN>, <HUB_PASSWORD>)

#Get the server IDs
serverIds = client.hub.listServerIds(hubKey)

#authenticate those servers(same credentials will be used as of hub to authenticate)
client.hub.attachToServers(hubKey, serverIds)

# perform the needed operation
systemsPerServer = client.multicast.system.list_systems(hubKey, serverIds)
successfulResponses = systemsPerServer["Successful"]["Responses"]
failedResponses = systemsPerServer["Failed"]["Responses"]

for system in successfulResponses:
  print (system)

#logout
client.auth.logout(hubKey)
```

Example: Auto-Connect Authentication

Auto-connect mode is similar to relay mode, it uses the Hub credentials to sign in in to all peripheral servers. However, there is no need to use the `attachToServers` method, as auto-connect mode connects to all available peripheral servers. This occurs at the same time as you sign in to the Hub.

A typical workflow for auto-connect authentication is:

1. Credentials for the Hub are passed to the `loginWithAutoconnectMode` method, and a session key for the Hub is returned (`hubKey`).
2. A `multicast` call is performed on a set of servers. This is defined by `serverIds`, which contains the IDs of the servers to target. In the background, `system.list_system` is called on each server's XMLRPC API.
3. Hub aggregates the results and returns the response in the form of a `map`. The map has two entries:
    - `Successful`: list of responses for those clients where the call succeeded.
    - `Failed`: list of responses for those clients where the call failed.

Listing 3. Example Python Script for Auto-Connect Authentication:

```
#!/usr/bin/python
import xmlrpclib


HUB_URL = "http://localhost:8888/hub/rpc/api"
HUB_LOGIN = "admin"
HUB_PASSWORD = "admin"

client = xmlrpclib.Server(<HUB_URL>, verbose=0)

hubKey = client.hub.loginWithAutoconnectMode(<HUB_LOGIN>, <HUB_PASSWORD>)

#Get the server IDs
serverIds = client.hub.listServerIds(hubKey)

# perform the needed operation
systemsPerServer = client.multicast.system.list_systems(hubKey, serverIds)
successfulResponses = systemsPerServer["Successful"]["Responses"]
failedResponses = systemsPerServer["Failed"]["Responses"]

for system in successfulResponses:
  print (system)

#logout
client.auth.logout(hubKey)
```

Managing Large Scale Deployments in a Retail Environment

SUSE Manager for Retail 2020.06 is an open source infrastructure management solution, optimized and tailored specifically for the retail industry. It uses the same technology as SUSE Manager, but is customized to address the needs of retail organizations.

SUSE Manager for Retail is designed for use in retail situations where customers can use point-of-service terminals to purchase or exchange goods, take part in promotions, or collect loyalty points. In addition to retail installations, it can also be used for novel purposes, such as maintaining student computers in an educational environment, or self-service kiosks in banks or hospitals.

SUSE Manager for Retail is intended for use in installations that include servers, workstations, point-of-service terminals, and other devices. It allows administrators to install, configure, and update the software on their servers, and manage the deployment and provisioning of point-of-service machines.

Point-of-Service (PoS) terminals can come in many different formats, such as point-of-sale terminals, kiosks, digital scales, self-service systems, and reverse-vending systems. Every terminal, however, is provided by a vendor, who set basic information about the device in the firmware. SUSE Manager for Retail accesses this vendor information to determine how best to work with the terminal in use.

In most cases, different terminals will require a different operating system (OS) image to ensure they work correctly. For example, an information kiosk has a high-resolution touchscreen, where a cashier terminal might only have a very basic display. While both of these terminals require similar processing and network functionality, they will require different OS images. The OS images ensure that the different display mechanisms work correctly.

For more information about setting up and using SUSE Manager for Retail, see **[Retail › Retail-introduction ›]**.

Tuning Large Scale Deployments

Uyuni is designed by default to work on small and medium scale installations. For installations with more than 1000 clients per Uyuni Server, adequate hardware sizing and parameter tuning must be performed.

The instructions in this section can have severe and catastrophic performance impacts when improperly used. In some cases, they can cause Uyuni to completely cease functioning. Always test changes before implementing them in a production environment. During implementation, take care when changing parameters. Monitor performance before and after each change, and revert any steps that do not produce the expected result.

We strongly recommend that you contact SUSE Consulting for assistance with tuning.

SUSE will not provide support for catastrophic failure when these advanced parameters are modified without consultation.

Tuning is not required on installations of fewer than 1000 clients. Do not perform these instructions on small or medium scale installations.

The Tuning Process

Any Uyuni installation is subject to a number of design and infrastructure constraints that, for the purposes of tuning, we call environmental variables. Environmental variables can include the total number of clients, the number of different operating systems under management, and the number of software channels.

Environmental variables influence, either directly or indirectly, the value of most configuration parameters. During the tuning process, the configuration parameters are manipulated to improve system performance.

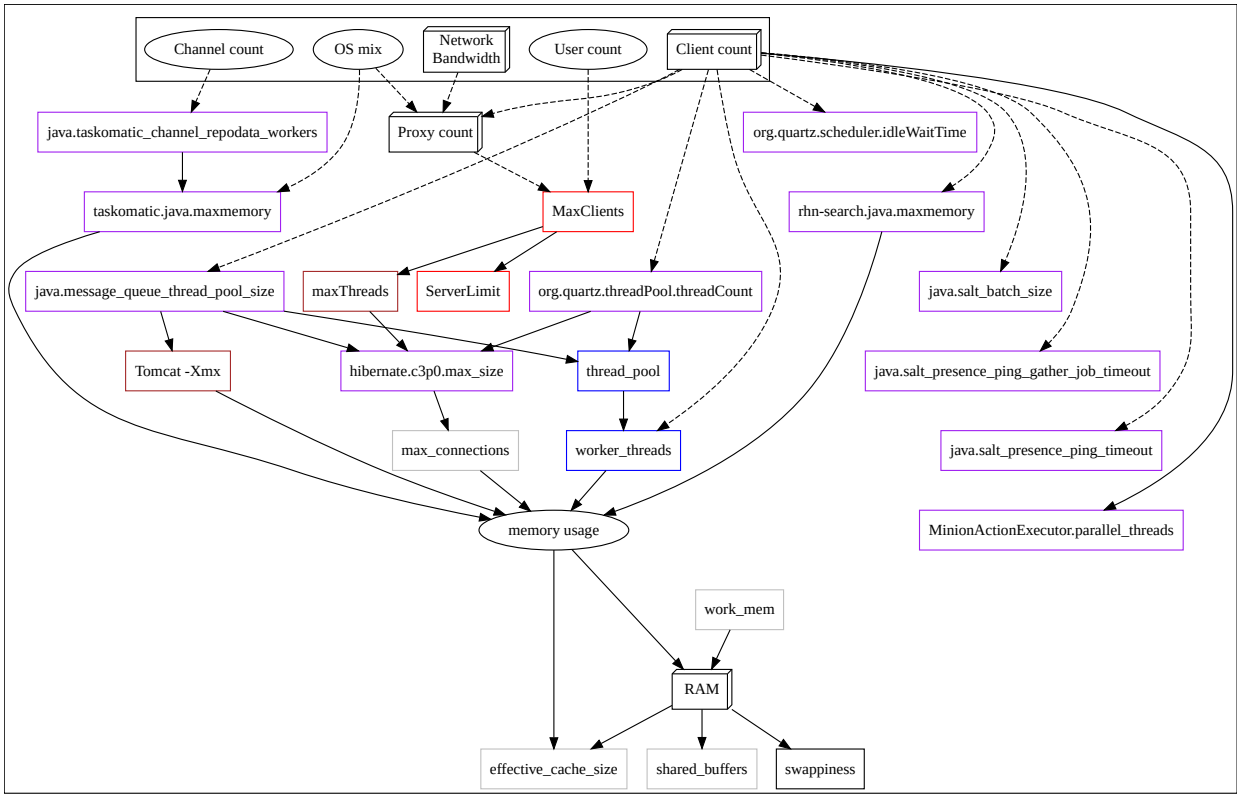
Before you begin tuning, you will need to estimate the best setting for each environment variable, and adjust the configuration parameters to suit.

To help you with the estimation process, we have provided you with a dependency graph. Locate the environmental variables on the dependency graph to determine how they will influence other variables and parameters.

Environmental variables are represented by graph nodes in a rectangle at the top of the dependency graph. Each node is connected to the relevant parameters that might need tuning. Consult the relevant sections in this document for more information about recommended values.

Tuning one parameter might require tuning other parameters, or changing hardware, or the infrastructure. When you change a parameter, follow the arrows from that node on the graph to determine what other parameters might need adjustment. Continue through each parameter until you have visited all nodes on

the graph.



Key to the Dependency Graph

- 3D boxes are hardware design variables or constraints
- Oval-shaped boxes are software or system design variables or constraints
- Rectangle-shaped boxes are configurable parameters, color-coded by configuration file:
  - Red: Apache `httpd` configuration files
  - Blue: Salt configuration files
  - Brown: Tomcat configuration files
  - Grey: PostgreSQL configuration files
  - Purple: `/etc/rhn/rhn.conf`
- Dashed connecting lines indicate a variable or constraint that might require a change to another parameter
- Solid connecting lines indicate that changing a configuration parameter requires checking another one to prevent issues

After the initial tuning has been completed, you will need to consider tuning again in these cases:

- If your tuning inputs change significantly
- If special conditions arise that require a certain parameter to be changed. For example, if specific warnings appear in a log file.

- If performance is not satisfactory

To re-tune your installation, you will need to use the dependency graph again. Start from the node where significant change has happened.

Environmental Variables

This section contains information about environmental variables (inputs to the tuning process).

Network Bandwidth

A measure of the typically available egress bandwith from the Uyuni Server host to the clients or Uyuni Proxy hosts. This should take into account network hardware and topology as well as possible capacity limits on switches, routers, and other network equipment between the server and clients.

Channel count

The number of expected channels to manage. Includes any vendor-provided, third-party, and cloned or staged channels.

Client count

The total number of actual or expected clients. It is important to tune any parameters in advance of a client count increase, whenever possible.

OS mix

The number of distinct operating system versions that managed clients have installed. This is ordered by family (SUSE Linux Enterprise, openSUSE, Red Hat Enterprise Linux, or Ubuntu based). Storage and computing requirements are different in each case.

User count

The expected maximum amount of concurrent users interacting with the Web UI plus the number of programs simultaneously using the XMLRPC API. Includes `spacecmd`, `spacewalk-clone-by-date`, and similar.

Parameters

This section contains information about the available parameters.

`MaxClients`

Description	The maximum number of HTTP requests served simultaneously by Apache httpd. Proxies, Web UI, and XMLRPC API clients each consume one. Requests exceeding the parameter will be queued and might result in timeouts.

Tune when	User count and proxy count increase significantly and this line appears in `/var/log/apache2/error_log`: `[…] [mpm_prefork:error] [pid …] AH00161: server reached MaxRequestWorkers setting, consider raising the MaxRequestWorkers setting`.
Value default	150
Value recommendation	150-500
Location	`/etc/apache2/server-tuning.conf`, in the `prefork.c` section
Example	`MaxClients = 200`
After changing	Immediately change `ServerLimit` and check `maxThreads` for possible adjustment.
Notes	This parameter was renamed to `MaxRequestWorkers`, both names are valid.
More information	https://httpd.apache.org/docs/2.4/en/mod/mpm_common.html#maxrequestworkers

`ServerLimit`

Description	The number of Apache httpd processes serving HTTP requests simultaneously. The number must equal `MaxClients`.
Tune when	`MaxClients` changes
Value default	150
Value recommendation	The same value as `MaxClients`
Location	`/etc/apache2/server-tuning.conf`, in the `prefork.c` section
Example	`ServerLimit = 200`
More information	https://httpd.apache.org/docs/2.4/en/mod/mpm_common.html#serverlimit

`maxThreads`

Description	The number of Tomcat threads dedicated to serving HTTP requests

Tune when	`MaxClients` changes. `maxThreads` must always be equal or greater than `MaxClients`
Value default	150
Value recommendation	The same value as `MaxClients`
Location	`/etc/tomcat/server.xml`
Example	`<Connector port="8009" protocol="AJP/1.3" redirectPort="8443" URIEncoding="UTF-8" address="127.0.0.1" maxThreads="200" connectionTimeout="20000"/>`
More information	https://tomcat.apache.org/tomcat-9.0-doc/config/http.html

Tomcat's `-Xmx`

Description	The maximum amount of memory Tomcat can use
Tune when	`java.message_queue_thread_pool_size` is increased or `OutOfMemoryException` errors appear in `/var/log/rhn/rhn_web_ui.log`
Value default	1 GiB
Value recommendation	4-8 GiB
Location	`/etc/sysconfig/tomcat`
Example	`JAVA_OPTS="··· -Xmx8G ···"`
After changing	Check memory usage
More information	https://docs.oracle.com/javase/8/docs/technotes/tools/windows/java.html

`java.message_queue_thread_pool_size`

Description	The maximum number of threads in Tomcat dedicated to asynchronous operations, including handling of incoming Salt events
Tune when	Client count increases significantly
Value default	5
Value recommendation	50 - 150
Location	`/etc/rhn/rhn.conf`

Example	`java.message_queue_thread_pool_size = 50`
After changing	Check `hibernate.c3p0.max_size`, as each thread consumes a PostgreSQL connection, starvation might happen if the allocated connection pool is insufficient. Check `thread_pool`, as each thread might perform Salt API calls, starvation might happen if the allocated Salt thread pool is insufficient. Check Tomcat's `-Xmx`, as each thread consumes memory, `OutOfMemoryException` might be raised if insufficient.
More information	`man rhn.conf`

`java.salt_batch_size`

Description	The maximum number of minions concurrently executing a scheduled action.
Tune when	Client count reaches several thousands and actions are not executed quickly enough.
Value default	200
Value recommendation	200-500
Location	`/etc/rhn/rhn.conf`
Example	`java.salt_batch_size = 300`
After changing	Check memory usage. Monitor memory usage closely before and after the change.
More information	**[Salt › Salt-rate-limiting › Salt Rate Limiting]**

`java.salt_presence_ping_timeout`

Description	Before any action is executed on a client, a presence ping is executed to make sure the client is reachable. This parameter sets the amount of time before a second command (`find_job`) is sent to the client to verify its presence. Having many clients typically means some will respond faster than others, so this timeout could be raised to accommodate for the slower ones.

Tune when	Client count increases significantly, or some clients are responding correctly but too slowly, and Uyuni excludes them from calls. This line appears in `/var/log/rhn/rhn_web_ui.log`: `"Got no result for <COMMAND> on minion <MINION_ID> (minion did not respond in time)"`
Value default	4 seconds
Value recommendation	4-400 seconds
Location	`/etc/rhn/rhn.conf`
Example	`java.salt_presence_ping_timeout = 40`
More information	**[Salt › Salt-timeouts › Salt Timeouts]**

`java.salt_presence_ping_gather_job_timeout`

Description	Before any action is executed on a client, a presence ping is executed to make sure the client is reachable. After `java.salt_presence_ping_timeout` seconds have elapsed without a response, a second command (`find_job`) is sent to the client for a final check. This parameter sets the number of seconds after the second command after which the client is definitely considered offline. Having many clients typically means some will respond faster than others, so this timeout could be raised to accommodate for the slower ones.
Tune when	Client count increases significantly, or some clients are responding correctly but too slowly, and Uyuni excludes them from calls. This line appears in `/var/log/rhn/rhn_web_ui.log`: `"Got no result for <COMMAND> on minion <MINION_ID> (minion did not respond in time)"`
Value default	1 second
Value recommendation	1-100 seconds
Location	`/etc/rhn/rhn.conf`
Example	`java.salt_presence_ping_gather_job_timeout = 10`
More information	**[Salt › Salt-timeouts › Salt Timeouts]**

java.taskomatic_channel_repodata_workers

Description	Whenever content is changed in a software channel, its metadata needs to be recomputed before clients can use it. Channel-altering operations include the addition of a patch, the removal of a package or a repository synchronization run. This parameter specifies the maximum number of Taskomatic threads that Uyuni will use to recompute the channel metadata. Channel metadata computation is both CPU-bound and memory-heavy, so raising this parameter and operating on many channels simultaneously could cause Taskomatic to consume significant resources, but channels will be available to clients sooner.
Tune when	Channel count increases significantly (more than 50), or more concurrent operations on channels are expected.
Value default	2
Value recommendation	2-10
Location	`/etc/rhn/rhn.conf`
Example	`java.taskomatic_channel_repodata_workers = 4`
After changing	Check `taskomatic.java.maxmemory` for adjustment, as every new thread will consume memory
More information	`man rhn.conf`

taskomatic.java.maxmemory

Description	The maximum amount of memory Taskomatic can use. Generation of metadata, especially for some OSs, can be memory-intensive, so this parameter might need raising depending on the managed OS mix.
Tune when	`java.taskomatic_channel_repodata_workers` increases, OSs are added to Uyuni (particularly Red Hat Enterprise Linux or Ubuntu), or `OutOfMemoryException` errors appear in `/var/log/rhn/rhn_taskomatic_daemon.log`.
Value default	4096 MiB

Value recommendation	4096-16384 MiB
Location	`/etc/rhn/rhn.conf`
Example	`taskomatic.java.maxmemory = 8192`
After changing	Check memory usage.
More information	`man rhn.conf`

`org.quartz.threadPool.threadCount`

Description	The number of Taskomatic worker threads. Increasing this value allows Taskomatic to serve more clients in parallel.
Tune when	Client count increases significantly
Value default	20
Value recommendation	20-200
Location	`/etc/rhn/rhn.conf`
Example	`org.quartz.threadPool.threadCount = 100`
After changing	Check `hibernate.c3p0.max_size` and `thread_pool` for adjustment
More information	http://www.quartz-scheduler.org/documentation/2.4.0-SNAPSHOT/configuration.html

`org.quartz.scheduler.idleWaitTime`

Description	Cycle time for Taskomatic. Decreasing this value lowers the latency of Taskomatic.
Tune when	Client count is in the thousands.
Value default	5000 ms
Value recommendation	1000-5000 ms
Location	`/etc/rhn/rhn.conf`
Example	`org.quartz.scheduler.idleWaitTime = 1000`
More information	http://www.quartz-scheduler.org/documentation/2.4.0-SNAPSHOT/configuration.html

MinionActionExecutor.parallel_threads

Description	Number of Taskomatic threads dedicated to sending commands to Salt clients as a result of actions being executed.
Tune when	Client count is in the thousands.
Value default	1
Value recommendation	1-10
Location	`/etc/rhn/rhn.conf`
Example	`taskomatic.com.redhat.rhn.taskomatic.task.MinionActionExecutor.parallel_threads = 10`

hibernate.c3p0.max_size

Description	Maximum number of PostgreSQL connections simultaneously available to both Tomcat and Taskomatic. If any of those components requires more concurrent connections, their requests will be queued.
Tune when	`java.message_queue_thread_pool_size` or `maxThreads` increase significantly, or when `org.quartz.threadPool.threadCount` has changed significantly. Each thread consumes one connection in Taskomatic and Tomcat, having more threads than connections might result in starving.
Value default	20
Value recommendation	100 to 200, higher than the maximum of `java.message_queue_thread_pool_size` + `maxThreads` and `org.quartz.threadPool.threadCount`
Location	`/etc/rhn/rhn.conf`
Example	`hibernate.c3p0.max_size = 100`
After changing	Check `max_connections` for adjustment.
More information	https://www.mchange.com/projects/c3p0/#maxPoolSize

rhn-search.java.maxmemory

Description	The maximum amount of memory that the `rhn-search` service can use.
Tune when	Client count increases significantly, and `OutOfMemoryException` errors appear in `journalctl -u rhn-search`.
Value default	512 MiB
Value recommendation	512-4096 MiB
Location	`/etc/rhn/rhn.conf`
Example	`rhn-search.java.maxmemory = 4096`
After changing	Check memory usage.

`shared_buffers`

Description	The amount of memory reserved for PostgreSQL shared buffers, which contain caches of database tables and index data.
Tune when	RAM changes
Value default	25% of total RAM
Value recommendation	25-40% of total RAM
Location	`/var/lib/pgsql/data/postgresql.conf`
Example	`shared_buffers = 8192MB`
After changing	Check memory usage.
More information	https://www.postgresql.org/docs/10/runtime-config-resource.html#GUC-SHARED-BUFFERS

`max_connections`

Description	Maximum number of PostgreSQL connections available to applications. More connections allow for more concurrent threads/workers in various components (in particular Tomcat and Taskomatic), which generally improves performance. However, each connection consumes resources, in particular `work_mem` megabytes per sort operation per connection.
Tune when	`hibernate.c3p0.max_size` changes significantly, as that parameter determines the maximum number of connections available to Tomcat and Taskomatic

Value default	400
Value recommendation	2 * hibernate.c3p0.max_size + 50, if less than 1000
Location	`/var/lib/pgsql/data/postgresql.conf`
Example	`max_connections = 250`
After changing	Check memory usage. Monitor memory usage closely before and after the change.
More information	https://www.postgresql.org/docs/10/runtime-config-connection.html#GUC-MAX-CONNECTIONS

`work_mem`

Description	The amount of memory allocated by PostgreSQL every time a connection needs to do a sort or hash operation. Every connection (as specified by `max_connections`) might make use of an amount of memory equal to a multiple of `work_mem`.
Tune when	Individual query operations are too slow, and value is below 5 MB
Value recommendation	2-20 MB
Location	`/var/lib/pgsql/data/postgresql.conf`
Example	`work_mem = 10MB`
After changing	check if the Uyuni Server might need additional RAM.
More information	https://www.postgresql.org/docs/10/runtime-config-resource.html#GUC-WORK-MEM

`effective_cache_size`

Description	Estimation of the total memory available to PostgreSQL for caching. It is the explicitly reserved memory (`shared_buffers`) plus any memory used by the kernel as cache/buffer.
Tune when	Hardware RAM or memory usage increase significantly

Value recommendation	Start with 75% of total RAM. For finer settings, use `shared_buffers` + free memory + buffer/cache memory. Free and buffer/cache can be determined via the `free -m` command (`free` and `buff/cache` in the output respectively)
Location	`/var/lib/pgsql/data/postgresql.conf`
Example	`effective_cache_size = 24GB`
After changing	Check memory usage
Notes	This is an estimation for the query planner, not an allocation.
More information	https://www.postgresql.org/docs/10/runtime-config-query.html#GUC-EFFECTIVE-CACHE-SIZE

`thread_pool`

Description	The number of worker threads serving Salt API HTTP requests. A higher number can improve parallelism of Uyuni Server-initiated Salt operations, but will consume more memory.
Tune when	`java.message_queue_thread_pool_size` or `org.quartz.threadPool.threadCount` are changed. Starvation can occur when there are more Tomcat or Taskomatic threads making simultaneous Salt API calls than there are Salt API worker threads.
Value default	100
Value recommendation	100-500, but should be higher than the sum of `java.message_queue_thread_pool_size` and `org.quartz.threadPool.threadCount`
Location	`/etc/salt/master.d/susemanager.conf`, in the `rest_cherrypy` section.
Example	`thread_pool: 100`
After changing	Check `worker_threads` for adjustment.
More information	https://docs.saltstack.com/en/latest/ref/netapi/all/salt.netapi.rest_cherrypy.html#performance-tuning

`worker_threads`

Description	The number of `salt-master` worker threads that process commands and replies from minions and the Salt API. Increasing this value, assuming sufficient resources are available, allows Salt to process more data in parallel from minions without timing out, but will consume significantly more RAM (typically about 70 MiB per thread).
Tune when	Client count increases significantly, `thread_pool` increases significantly, or `SaltReqTimeoutError` or `Message timed out` errors appear in `/var/log/salt/master`.
Value default	8
Value recommendation	8-200
Location	`/etc/salt/master.d/tuning.conf`
Example	`worker_threads: 50`
After changing	Check memory usage. Monitor memory usage closely before and after the change.
More information	https://docs.saltstack.com/en/latest/ref/configuration/master.html#worker-threads

`swappiness`

Description	How aggressively the kernel moves unused data from memory to the swap partition. Setting a lower parameter typically reduces swap usage and results in better performance, especially when RAM memory is abundant.
Tune when	RAM increases, or swap is used when RAM memory is sufficient.
Value default	60
Value recommendation	1-60. For 128 GB of RAM, 10 is expected to give good results.
Location	`/etc/sysctl.conf`
Example	`vm.swappiness = 20`
More information	https://documentation.suse.com/sles/15-SP1/html/SLES-all/cha-tuning-memory.html#cha-tuning-memory-vm

Memory Usage

Adjusting some of the parameters listed in this section can result in a higher amount of RAM being used by various components. It is important that the amount of hardware RAM is adequate after any significant change.

To determine how RAM is being used, you will need to check each process that consumes it.

Operating system

Stop all Uyuni services and inspect the output of `free -h`.

Java-based components

This includes Taskomatic, Tomcat, and `rhn-search`. These services support a configurable memory cap.

The Uyuni Server

Depends on many factors and can only be estimated. Measure PostgreSQL reserved memory by checking `shared_buffers`, permanently. You can also multiply `work_mem` and `max_connections`, and multiply by three for a worst case estimate of per-query RAM. You will also need to check the operating system buffers and caches, which are used by PostgreSQL to host copies of database data. These often automatically occupy any available RAM.

It is important that the Uyuni Server has sufficient RAM to accommodate all of these processes, especially OS buffers and caches, to have reasonable PostgreSQL performance. We recommend you keep several gigabytes available at all times, and add more as the database size on disk increases.

Whenever the expected amount of memory available for OS buffers and caches changes, update the `effective_cache_size` parameter to have PostgreSQL use it correctly. You can calculate the total available by finding the total RAM available, less the expected memory usage.

To get a live breakdown of the memory used by services on the Uyuni Server, use this command:

```
pidstat -p ALL -r --human 1 60 | tee pidstat-memory.log
```

This command will save a copy of displayed data in the `pidstat-memory.log` file for later analysis.