Red Hat Update Infrastructure 3.0

System Administrator's Guide

Installation, Configuration, and Administration of Red Hat Update Infrastructure

Last Updated: 2018-10-18

Red Hat Update Infrastructure 3.0 System Administrator's Guide

Installation, Configuration, and Administration of Red Hat Update Infrastructure

Legal Notice

Abstract

The Red Hat Update Infrastructure 3.0 System Administrator's Guide provides requirements and instructions to help cloud providers implement and configure Red Hat Update Infrastructure. It also provides step-by-step instructions for performing administrative tasks, such as adding or removing content delivery servers, load balancers, and custom repositories.

Table of Contents

CHAPTER 1. ABOUT RED HAT UPDATE INFRASTRUCTURE 6
1.1. NEW FEATURES IN THIS RED HAT UPDATE INFRASTRUCTURE RELEASE 7
1.2. INSTALLATION OPTIONS 7
1.2.1. Option 1: Full Installation 8
1.2.2. Option 2: Installation with an Existing Storage Solution 8
1.2.3. Option 3: Installation with an Existing Load Balancing Solution 8
1.2.4. Option 4: Installation with Existing Storage and Load Balancing Solutions 9
1.3. RED HAT UPDATE INFRASTRUCTURE COMPONENTS 10
1.3.1. Red Hat Update Appliance 10
1.3.2. Content Delivery Server 11
1.3.3. HAProxy 12
1.3.4. Repositories, Containers, and Content 13
1.4. CONTENT PROVIDER TYPES 13
1.5. UTILITY AND COMMAND-LINE INTERFACE COMMANDS 13
1.6. COMPONENT COMMUNICATIONS 14

CHAPTER 2. INFORMATION REQUIRED FOR INSTALLATION 15

CHAPTER 3. PREREQUISITES FOR INSTALLING RED HAT UPDATE INFRASTRUCTURE 17

CHAPTER 4. REGISTER RED HAT UPDATE INFRASTRUCTURE AND ATTACH SUBSCRIPTIONS 20
4.1. INSTALL RED HAT ENTERPRISE LINUX 20
4.2. REGISTER RED HAT UPDATE INFRASTRUCTURE 20
4.3. ATTACH A SUBSCRIPTION TO THE RED HAT UPDATE APPLIANCE 21
4.4. ATTACH A SUBSCRIPTION TO THE CDS NODES 22
4.5. ATTACH A SUBSCRIPTION TO THE HAPROXY NODES 22
4.6. ENABLE THE REQUIRED REPOSITORIES 23

CHAPTER 5. SHARED STORAGE 25
5.1. GLUSTER STORAGE 25
5.1.1. Create Shared Storage 25
5.1.2. Extend the Storage Volume 27
5.2. CREATE NFS STORAGE 28

CHAPTER 6. INSTALL RED HAT UPDATE INFRASTRUCTURE 30
6.1. GENERATE AN RSA KEY PAIR 30
6.1.1. RSA Key Pair for Version 2 of the SSH Protocol 30
6.1.2. ECDSA Key Pair for Version 2 of the SSH Protocol 31
6.2. APPLY UPDATES 32
6.3. MOUNT RED HAT UPDATE INFRASTRUCTURE ISO 32
6.4. RUN THE SETUP_PACKAGE_REPOS SCRIPT 32
6.5. INSTALL THE RHUI-INSTALLER SCRIPT 33
6.6. RUN THE RHUI-INSTALLER 33
6.7. CHANGE THE INITIAL PASSWORD 34
6.8. REGISTER A RED HAT SUBSCRIPTION IN RHUI 36

CHAPTER 7. ADD A CONTENT DELIVERY SERVER 38

CHAPTER 8. ADD AN HAPROXY LOAD BALANCER 40

CHAPTER 9. CREATE AND SYNCHRONIZE A RED HAT REPOSITORY 42
9.1. CREATE A REPOSITORY 42
9.2. SYNCHRONIZE A REPOSITORY 43
9.3. CHECK REPOSITORY SYNCHRONIZATION 45

CHAPTER 10. CLIENT ENTITLEMENT CERTIFICATE AND CLIENT CONFIGURATION RPM 49
10.1. CREATE AN ENTITLEMENT CERTIFICATE 49
10.2. CREATE A CLIENT CONFIGURATION RPM 50
10.3. INSTALL A CLIENT RPM 51
10.4. WORKING WITH THE EUS CHANNEL 52

CHAPTER 11. CREATE CLIENT PROFILES FOR THE RED HAT UPDATE INFRASTRUCTURE SERVERS 54
11.1. GENERATE GPG KEYS 54
11.2. SET UP CUSTOM REPOSITORIES 56
11.3. CREATE AN ENTITLEMENT CERTIFICATE 58
11.4. CREATE A CLIENT CONFIGURATION RPM 58
11.5. INSTALL THE CLIENT CONFIGURATION RPM ON A CLIENT NODE 58

CHAPTER 12. CREATE CLIENT IMAGES AND TEMPLATES 60
12.1. IMAGE REQUIREMENTS 60
12.2. RED HAT UPDATE INFRASTRUCTURE INTEGRATION 60
12.3. TEMPLATE PREPARATION 61

CHAPTER 13. CERTIFIED CLOUD AND SERVICE PROVIDER CERTIFICATION WORKFLOW 63

CHAPTER 14. MANAGE CONTENT 64
14.1. AVAILABLE CHANNELS 64
14.2. MANAGE THE LINUX SOFTWARE REPOSITORIES 64
14.2.1. List the Available Repositories 64
14.2.2. Display the Repository Information 65
14.2.3. Add a Red Hat Repository 65
14.2.4. Delete a Red Hat Repository 65
14.2.5. List the RPM Packages in a Repository 66
14.2.6. Create a Custom Repository 66
14.2.7. Upload Packages to a Custom Repository 68
14.2.8. Delete Packages from a Custom Repository 68
14.3. ORPHANED CONTENT UNITS 70
14.4. MANAGE THE CONTENT DELIVERY SERVER NODES 71
14.5. WORKING WITH CONTAINERS 72
14.6. MANAGE THE CONTENT DELIVERY SERVER DOCKER CONTENT 73
14.6.1. Docker Content in Red Hat Update Infrastructure 73
14.6.2. Add a Container to Red Hat Update Infrastructure 73
14.6.3. Synchronize the docker Repository 75
14.6.4. Generate the docker Client Configuration 76
14.6.5. Install a RPM on the Client 76
14.6.6. Test the docker pull Command on the Client 77
14.7. ATOMIC HOST AND OSTREE CONTENT 78
14.7.1. Add an Atomic Host Repository 79
14.7.2. Synchronize the OSTree Repository 79
14.7.3. Generate a Client Configuration Package on the RHUA 80
14.7.4. Configure Atomic Host 81
14.7.5. Test the ostree pull Command with Atomic Host 81

CHAPTER 15. MANAGE CERTIFICATES AND KEYS 83
15.1. RED HAT UPDATE APPLIANCE CERTIFICATES 83
15.2. CONTENT DELIVERY SERVER CERTIFICATES 83
15.3. CLIENT CERTIFICATES 83
15.4. DISPLAY AND MANAGE CERTIFICATES 84
15.4.1. List the Entitled Products for a Certificate 84

15.4.2. List Custom Repository Entitlements 85
15.4.3. Upload a Content Certificate 85

CHAPTER 16. RED HAT UPDATE INFRASTRUCTURE 3.0 STATUS CODES, LOG FILES, AND CONFIGURATION FILES 87

CHAPTER 17. UPGRADE RED HAT UPDATE INFRASTRUCTURE 89

CHAPTER 18. BACK UP AND RESTORE RED HAT UPDATE INFRASTRUCTURE 91
18.1. BACK UP THE RED HAT UPDATE APPLIANCE 91
18.2. RESTORE THE RED HAT UPDATE APPLIANCE 92
18.3. BACK UP A CONTENT DELIVERY SERVER 93
18.4. RESTORE A CONTENT DELIVERY SERVER 94
18.5. BACK UP AN HAPROXY SERVER 95
18.6. RESTORE AN HAPROXY SERVER 95

CHAPTER 19. MIGRATE TO A NEW LOAD BALANCER, OR CHANGE THE NAME OF AN EXISTING LOAD BALANCER 96

APPENDIX A. RED HAT UPDATE INFRASTRUCTURE MANAGEMENT TOOL MENUS AND COMMANDS 97

APPENDIX B. RED HAT UPDATE INFRASTRUCTURE COMMAND-LINE INTERFACE 100
B.1. CERT 100
B.2. PACKAGES 101
B.3. REPO 101
B.4. STATUS 102
B.5. CLIENT 102
B.6. SUBSCRIPTIONS 103

APPENDIX C. RESOLVE COMMON PROBLEMS IN RED HAT UPDATE INFRASTRUCTURE 104

APPENDIX D. API REFERENCE IN RED HAT UPDATE INFRASTRUCTURE 3.0 108
D.1. REPOSITORY APIS 108
D.1.1. Creation, Deletion, and Configuration 108
D.1.1.1. Create a Repository 108
D.1.1.2. Update a Repository 109
D.1.1.3. Associate an Importer to a Repository 111
D.1.1.4. Associate a Distributor with a Repository 112
D.1.1.5. Update an Importer Associated with a Repository 114
D.1.1.6. Disassociate an Importer from a Repository 115
D.1.1.7. Update a Distributor Associated with a Repository 115
D.1.1.8. Disassociate a Distributor from a Repository 116
D.1.1.9. Delete a Repository 117
D.1.2. Retrieval 117
D.1.2.1. Retrieve a Single Repository 117
D.1.2.2. Retrieve All Repositories 119
D.1.2.3. Advanced Search for Repositories 120
D.1.2.4. Retrieve Importers Associated with a Repository 123
D.1.2.5. Retrieve an Importer Associated with a Repository 124
D.1.2.6. Retrieve Distributors Associated with a Repository 124
D.1.2.7. Retrieve a Distributor Associated with a Repository 125
D.1.2.8. Advanced Search for Distributors 126
D.1.3. Synchronization 127
D.1.3.1. Sync a Repository 127
D.1.3.2. Download a Repository 127
D.1.3.3. Scheduling a Sync 128

D.1.3.4. Updating a Scheduled Sync 129
D.1.3.5. Deleting a Scheduled Sync 130
D.1.3.6. Listing All Scheduled Syncs 130
D.1.3.7. Listing a Single Scheduled Sync 131
D.1.3.8. Retrieving Sync History 132
D.1.4. Publication 133
D.1.4.1. Publish a Repository 133
D.1.4.2. Scheduling a Publish 133
D.1.4.3. Updating a Scheduled Publish 135
D.1.4.4. Deleting a Scheduled Publish 135
D.1.4.5. Listing All Scheduled Publishes 136
D.1.4.6. Listing a Single Scheduled Publish 137
D.1.4.7. Retrieving Publish History 137
D.1.5. Content Retrieval 138
D.1.5.1. Advanced Unit Search 138
D.2. TASK MANAGEMENT 139
D.2.1. Task Report 140
D.2.2. Polling Task Progress 141
D.2.3. Cancelling a Task 141
D.2.4. Listing Tasks 142
D.2.5. Deleting Completed Tasks 142
D.2.6. Searching for Tasks 142
D.3. TASK GROUP MANAGEMENT 143
D.3.1. Cancelling Tasks in a Task Group 143
D.3.2. Task Group Summary 144
D.3.3. Polling Task Group Progress 144

APPENDIX E. GNU GENERAL PUBLIC LICENSES 146
E.1. GNU GENERAL PUBLIC LICENSE VERSION 2.0 146
E.2. GNU GENERAL PUBLIC LICENSE VERSION 3.0 150

APPENDIX F. RED HAT UPDATE INFRASTRUCTURE SYSTEM ADMINISTRATOR'S GUIDE DOCUMENT REVISION HISTORY 161

CHAPTER 1. ABOUT RED HAT UPDATE INFRASTRUCTURE

Red Hat Update Infrastructure (RHUI) is a highly scalable, highly redundant framework that enables you to manage repositories and content. It also enables cloud providers to deliver content and updates to Red Hat Enterprise Linux (RHEL) instances. Based on the upstream Pulp project, RHUI allows cloud providers to locally mirror Red Hat-hosted repository content, create custom repositories with their own content, and make those repositories available to a large group of end users through a load-balanced content delivery system.

The Red Hat Update Infrastructure 3.0 System Administrator's Guide will help system administrators prepare their infrastructure for participation in the Red Hat Certified Cloud and Service Provider program.

This guide documents the steps necessary to install and configure the Red Hat Update Appliance (RHUA), content delivery servers (CDSs), repositories, shared storage, and load balancing. Experienced RHEL system administrators are the target audience. System administrators with limited Red Hat Enterprise Linux skills should consider engaging Red Hat Consulting to provide a Red Hat Certified Cloud Provider Architecture Service.

The Red Hat Update Infrastructure 3.0 System Administrator's Guide also provides guidance to system administrators when configuring, managing, and updating RHUI. This guide discusses:

- the various RHUI components
- content provider types
- the command-line interface (CLI) used to manage the components
- utility commands
- certificate management
- content management.

See Appendix A, *Red Hat Update Infrastructure Management Tool Menus and Commands* for a list of all of the available menus and commands in the Red Hat Update Infrastructure Management Tool.

See Appendix B, *Red Hat Update Infrastructure Command-Line Interface* for a list of the functions that can also be run from a standard shell prompt.

See Appendix C, *Resolve Common Problems in Red Hat Update Infrastructure* for some of the common known issues and possible solutions.

See Appendix D, *API Reference in Red Hat Update Infrastructure 3.0* for a list of the Pulp APIs used in Red Hat Update Infrastructure.

See the following resources for more information on the various Red Hat Update Infrastructure components.

- Red Hat Enterprise Linux
- Red Hat Gluster Storage
- HAProxy
- High Availability Add-On
- Pulp

- Docker
- Puppet

1.1. NEW FEATURES IN THIS RED HAT UPDATE INFRASTRUCTURE RELEASE

Red Hat Update Infrastructure 3.0 offers:

- easy installation using Puppet.
- code rebased to Pulp 2.8 to be consistent with the code base in Red Hat Satellite 6.
- faster access to content due to reworked architecture for automated installations.
- default use of Red Hat Gluster Storage as shared storage to speed up content availability at the CDS and eliminate the need for synchronization.
- high-availability deployment to reduce the error of one CDS not being synchronized with another CDS.
- a load balancer/HAProxy node that is client-facing. (This functionality was integrated previously into the CDS logic.)
- certificates managed by the **`rhui-installer`** and **`rhui-manager`** commands.
- updates to *yum.repos.d/**, certificates, and keys to use a new unified URL.
- removal of client-side load balancing functionality from rhui-lb.py.
- support for Docker and OSTree (atomic) content.

1.2. INSTALLATION OPTIONS

The following table presents the various Red Hat Update Infrastructure components.

Table 1.1. Red Hat Update Infrastructure Components and Functions

Component	Acronym	Function	Alternative
Red Hat Update Appliance	RHUA	Downloads new packages from the Red Hat content delivery network and copies new packages to each CDS node.	None
Content Delivery Server	CDS	Provides the Yum repositories that clients connect to for the updated packages	None
HAProxy	None	Provideds load balancing across CDS nodes	Existing storage solution

Component	Acronym	Function	Alternative
Gluster Storage	None	Provides shared storage	Existing storage solution

At a high level, this guide describes how to perform the following installation tasks.

Table 1.2. Red Hat Update Infrastructure Installation Tasks

Installation Task	Performed on
Install Red Hat Enterprise Linux 6 or 7	RHUA, CDS, and HAProxy
Subscribe the system	RHUA, CDS, and HAProxy
Attach a Red Hat Update Infrastructure subscription	RHUA, CDS, and HAProxy
Apply updates	RHUA, CDS and HAProxy
Mount the Red Hat Update Infrastructure ISO (optional)	RHUA, CDS, and HAProxy
Run the **setup_package_repos** script (optional if using the RHUI 3 ISO)	RHUA, CDS, and HAProxy
Install rhui-installer	RHUA
Run rhui-installer	RHUA

There are several scenarios for setting up your cloud environment. This guide documents Option 1: Full Installation and includes notes and remarks about how to alter the installation for other scenarios.

1.2.1. Option 1: Full Installation

- A RHUA
- Three or more CDS nodes with Gluster Storage
- Two or more HAProxy servers

1.2.2. Option 2: Installation with an Existing Storage Solution

- A RHUA
- Two or more CDS nodes with an existing storage solution
- Two or more HAProxy servers

1.2.3. Option 3: Installation with an Existing Load Balancing Solution

- A RHUA

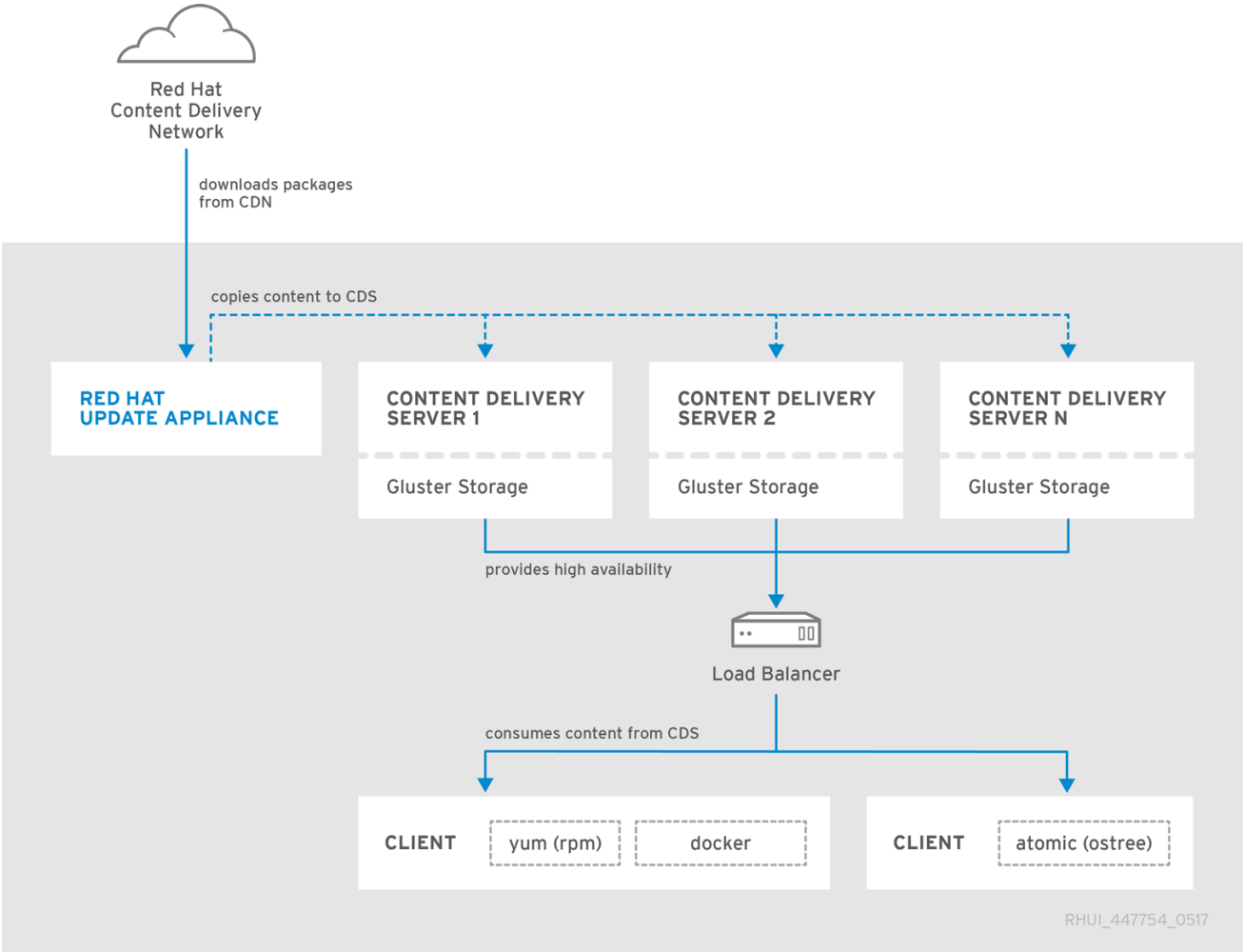
- Three or more CDS nodes with Gluster Storage
- Existing load balancer

1.2.4. Option 4: Installation with Existing Storage and Load Balancing Solutions

- A RHUA
- Two or more CDS nodes with an existing storage solution
- Existing load balancer

The following figure depicts a high-level view of how the various RHUI components interact with each other.

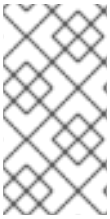
Figure 1.1. Red Hat Update Infrastructure Overview



NOTE

You can use the ISO to install Red Hat Update Infrastructure 3.0, or you can connect to the Red Hat Customer Portal to download Red Hat Update Infrastructure 3.0.

Install the RHUA and CDS nodes on separate _X86 servers (bare metal or virtual machines). You cannot install Red Hat Update Infrastructure without the RHUI ISO and an appropriate content certificate, provided by Red Hat. Ensure all _X86 servers (bare metal or virtual machines) and networks that connect to RHUI can access the ISO.



NOTE

- Despite its name, the RHUA is not actually shipped as an appliance; it is an RPM installed on an instance in the cloud.
- See the Release Notes before setting up Red Hat Update Infrastructure 3.0.

1.3. RED HAT UPDATE INFRASTRUCTURE COMPONENTS

The following subsections describe each RHUI component.

1.3.1. Red Hat Update Appliance

There is one RHUA per RHUI installation, though in many cloud environments there will be one RHUI installation per region or datacenter. For example, Amazon's EC2 cloud comprises several regions. In every region, there is a separate RHUI set up with its own RHUA node.

The RHUA:

- downloads new packages from the Red Hat content delivery network (CDN). The RHUA is the only RHUI component that connects to Red Hat, and you can configure the RHUA's synchronization schedule.
- copies new packages to each CDS node.
- verifies the RHUI installation's health and writes the results to a file located on the RHUA. Monitoring solutions use this file to determine the RHUI installation's health.
- provides a human-readable view of the RHUI installation's health through a CLI tool.

The RHUI uses two main configuration files: */etc/rhui/rhui-tools.conf* and */etc/rhui-installer/answers.yaml*.

The */etc/rhui/rhui-tools.conf* configuration file contains general options used by the RHUA, such as the default file locations for certificates, and default configuration parameters for the Red Hat CDN synchronization. This file normally does not require editing.

The Red Hat Update Infrastructure Management Tool generates the */etc/rhui-installer/answers.yaml* configuration file based on user-inputted values. It contains all the information that drives the running of a RHUA in a particular region. An example configuration includes the destination on the RHUA to download packages and a list of CDS nodes (host names) in the RHUI installation.

The RHUA employs several services to synchronize, organize, and distribute content for easy delivery:

- **Pulp**: The service that oversees management of the supporting services, providing a user interface for users to interact with.
- **MongoDB**: A NoSQL database used to keep track of currently synchronized repositories, packages, and other crucial metadata. MongoDB stores values in BSON (Binary JSON) objects.
- **Qpid**: An Apache-based messaging broker system that allows the RHUA to interact securely with the CDSs to inform them of desired actions against the RHUA (synchronize, remove, adjust repositories, and so on). This allows for full control over the RHUI appliance from just the RHUA system.

The following considerations might apply.

- MongoDB's files appear to take up a large amount of room on the file system and are sometimes larger than the database content itself. This is normal behavior based on Mongo's allocation method. For more information, see RHUIs mongodb files are larger than the actual database contents.
- If MongoDB fails to start, clearing database locks and performing a repair is often effective as outlined in Red Hat Update Infrastructure fails to start due to a MongoDB startup error.

1.3.2. Content Delivery Server

The CDS nodes provide the repositories that clients connect to for the updated content. There can be as few as one CDS. Because RHUI provides a load balancer with failover capabilities, we recommended that you use multiple CDS nodes.

The CDSs host content to end-user RHEL systems. While there is no required number of systems, the CDS works in a round-robin style load-balanced fashion (A, B, C, A, B, C) to deliver content to end-user systems. The CDS uses HTTP to host content to end-user systems via httpd-based yum repositories.

During configuration, you specify the CDS's directory where packages are synchronized. Similar to the RHUA, the only requirement is that you mount the directory on the CDS. It is up to the cloud provider to determine the best course of action when allocating the necessary devices. The Red Hat Update Infrastructure Management Tool configuration RPM takes care of linking the package directory with the Apache configuration to serve it.

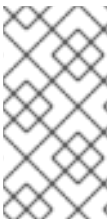
If NFS is used, **`rhui-installer`** can configure an NFS share on the RHUA to store the content as well as a directory on the CDSs to mount the NFS share. The following **`rhui-manager`** options control these settings:

- **`--remote-fs-mountpoint`** is the file system location where the remote file system share should be mounted (default: */var/lib/rhui/remote_share*)
- **`--remote-fs-server`** is the remote mount point for a shared file system to use, for example, **`nfs.example.com:/path/to/share`** (default: **`nfs.example.com:/export`**)

If these default values are used, the */export* directory on the RHUA and the */var/lib/rhui/remote_share* directory on each CDS are identical. For example, the *published* subdirectory has the following structure if Yum, Docker, and OSTree repositories are already synchronized.

```
[root@rhua ~]# ls /export/published/
docker  ostree  yum

[root@cds01 ~]# ls /var/lib/rhui/remote_share/published/
docker  ostree  yum
```

The expected usage is that each CDS will keep its own copy of the packages. It is possible the cloud provider will use some form of shared storage (such as Gluster Storage) that the RHUA writes packages to and each CDS reads from.



NOTE

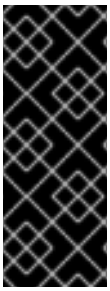
The storage solution must provide an NFS endpoint for mounting on the RHUA and CDSs. If local storage is implemented, shared storage is needed for the cluster to work. If you want to provide local storage to the RHUA, configure the RHUA to function as the NFS server with a *rhua.example.com:/path/to/nfs/share* endpoint configured.

The only nonstandard logic that takes place on each CDS is the entitlement certificate checking. This checking ensures that the client making requests on the Yum or OSTree repositories is authorized by the cloud provider to access those repositories. The check ensures:

- that the entitlement certificate was signed by the cloud provider's Certificate Authority (CA) Certificate. The CA Certificate is installed on the CDS as part of its configuration to facilitate this verification.
- the requested URI matches an entitlement found in the client's entitlement certificate.

If the CA verification fails, the client will see an SSL error. See the CDS's Apache logs under */var/log/httpd/* for more information.

```
[root@cds01 ~]# ls -1 /var/log/httpd/
access_log
cds.example.com_access_ssl.log
cds.example.com_error_ssl.log
crane_access_ssl.log
crane_error_ssl.log
default_error.log
error_log
```



IMPORTANT

- The Apache configuration is handled through the */etc/httpd/conf.d/cds_ssl.conf* file during the CDS installation.
- If multiple clients experience problems updating against a repository, this may indicate a problem with the RHUI itself. See Yum generates 'Errno 14 HTTP Error 401: Authorization Required' while accessing RHUI CDS for more details.

1.3.3. HAProxy

If more than one CDS is used, a load-balancing solution must be in place to spread client HTTPS requests across all servers. RHUI ships with HAProxy, but it is up to you to choose what load-balancing solution (for example, the one from the cloud provider) to use during the installation. If HAProxy is used, you must also decide how many nodes to bring in. See HAProxy Configuration for more information.

Clients are not configured to go directly to a CDS; their repository files are configured to point to HAProxy, the RHUI load balancer. HAProxy is a TCP/HTTP reverse proxy particularly suited for high-availability environments. HAProxy:

- routes HTTP requests depending on statically assigned cookies.
- spreads the load among several servers while assuring server persistence through the use of HTTP cookies.
- switches to backup servers in the event a main server fails.
- accepts connections to special ports dedicated to service monitoring.
- stops accepting connections without breaking existing ones.
- adds, modifies, and deletes HTTP headers in both directions.
- blocks requests matching particular patterns.

- persists client connections to the correct application server depending on application cookies.
- reports detailed status as HTML pages to authenticated users from a URI intercepted from the application.

With RHEL 7, the load balancer technology is included in the base operating system. With RHEL 6, the load balancer technology is in the Red Hat Enterprise Linux Load Balancer repository. The load balancer must be installed on a separate node.

NOTE

If you are using an existing load balancer, ensure ports 5000 and 443 are configured in the load balancer for the cds-lb-hostname forwarded to the pool and that all CDSs in the cluster are in the load balancer's pool. You do not need to follow the steps in Chapter 8, *Add an HAProxy Load Balancer*.

Keep in mind that when clients fail to connect successfully, it is important to review the httpd logs on the CDS under */var/log/httpd/* to ensure that any requests reached the CDS. If not, issues such as DNS or general network connectivity may be at fault.

1.3.4. Repositories, Containers, and Content

A repository is a storage location for software packages (RPMs). RHEL uses **`yum`** commands to search a repository, download, install, and configure the RPMs. The RPMs contain all the dependencies needed to run an application. RPMs also download updates for software in your repositories.

RHEL 7 implements Linux containers using core technologies such as control groups (cgroups) for resource management, namespaces for process isolation, and SELinux for security, enabling secure multiple tenancy and reducing the potential for security exploits. Linux containers enable rapid application deployment, simpler testing, maintenance, and troubleshooting while improving security. Using RHEL 7 with Docker allows you to increase staff efficiency, deploy third-party applications faster, enable a more agile development environment, and manage resources more tightly.

There are two general scenarios for using Linux containers in RHEL 7. You can work with host containers as a tool for application sandboxing, or you can use the extended features of image-based containers. When you launch a container from an image, a writable layer is added on top of this image. Every time you commit a container (using the **`docker commit`** command), a new image layer is added to store your changes.

Content, as it relates to RHUI, is the software (such as RPMs) that you download from the Red Hat CDN for use on the RHUA and the CDS nodes. The RPMs provide the files necessary to run specific applications and tools. Clients are granted access by a set of SSL content certificates and keys provided by an rpm package, which also provides a set of generated yum repository files.

See What Channels Can Be Delivered at Red Hat's Certified Certified Cloud & Service Provider (CCSP) Partners? for more information.

1.4. CONTENT PROVIDER TYPES

There are three types of cloud computing environments: public cloud, private cloud, and hybrid cloud. This guide focuses on public and private clouds. We assume the guide's audience understands the implications of using public, private, and hybrid clouds.

1.5. UTILITY AND COMMAND-LINE INTERFACE COMMANDS

See Appendix A, *Red Hat Update Infrastructure Management Tool Menus and Commands* for a list of the functions you can perform using Red Hat Update Infrastructure Management Tool.

See Appendix B, *Red Hat Update Infrastructure Command-Line Interface* for a list of the functions that can also be run from a standard shell prompt.

1.6. COMPONENT COMMUNICATIONS

All RHUI components use the HTTPS communication protocol over port 443.

Table 1.3. Red Hat Update Infrastructure Communication Protocols

Source	Destination	Protocol	Purpose
Red Hat Update Appliance	Red Hat Content Delivery Network	HTTPS	Downloads packages from Red Hat
Load Balancer	Content Delivery Server	HTTPS	Forwards the client's yum, docker or ostree request
Client	Load Balancer	HTTPS	Used by yum, docker, or ostree on the client to download packages from a content delivery server

Report a bug

CHAPTER 2. INFORMATION REQUIRED FOR INSTALLATION

Use the following checklist to help you gather all the required information before you install the Red Hat Update Appliance (RHUA) and the content delivery server (CDS) nodes.

Table 2.1. Red Hat Update Infrastructure Requirements Checklist

Required Information	Information Usage	Resources and Note
Red Hat Credentials		Red Hat Customer Portal
	Network and firewall requirements for the RHUA and each CDS	It is possible for a CDS to have a client-facing host name that differs from the host name used for intra-Red Hat Update Infrastructure communication. If you are using client-facing host names, note each CDS's client-facing FQDN and the corresponding IP address.
	Proxy for access to the Red Hat content delivery network	Proxy settings for Red Hat Update Infrastructure set automatically during the installation. They are set on the CDS nodes in the */etc/yum.conf* files, where you configure the repositories.
Content Repository Size	Storage space for the RPM packages required by Red Hat Update Infrastructure*	See Section 1.4.3 Storage in the Red Hat Satellite 6.1 Installation Guide for specific storage requirements.
Client Profiles	A client profile determines Red Hat Update Infrastructure content that is available to the client and the CDS from which the client downloads that content.	

*All repositories are placed in the */var/lib/pulp* directory. Create this directory only if you need to create a new mount point for it; otherwise, the system creates it automatically during the installation process. All repositories are synchronized with CDS nodes under */var/lib/pulp-cds*.

NOTE

Consider using a separate storage volume for the installation if you expect to store a large amount of data.

NOTE

Each RHUI server (RHUA node or CDS node) requires a separate file system of the required size. It is important to use technologies such as LVM, SAN, or NAS storage that allow the content repository to grow if needed. The current *rhel-7-server-rpms* repository is 12 GB, and the current *rhel-6-server-rpms* repository is 29 GB. These repositories will grow as the product does. These repositories are examples of the large sizes needed to hold all of the packages in any given environment.

Report a bug

CHAPTER 3. PREREQUISITES FOR INSTALLING RED HAT UPDATE INFRASTRUCTURE

The cloud provider provides the following technical prerequisites:

1. completion of the initial stages of the Red Hat Certified Cloud & Service Provider (CCSP) certification, including review of the client's :
    a. virtualization, image creation, and instance provisioning technologies, tools, and processes.
    b. proposed process for measuring and reporting consumption of Red Hat software.
    c. proposed process for notifying customers of errata updates to Red Hat software.
    d. proposed process for making images that include Red Hat software available to customers, including image life-cycle management and retiring outdated images.
    See Product Documentation for Red Hat Certified Cloud and Service Provider Certification Browse Knowledgebase for more information.
2. Self-signed certificates are typically used for Red Hat Update Infrastructure (RHUI) deployment. If SSL certificates signed by a third-party certificate authority will be used, they have been obtained by the client and reviewed by Red Hat.

NOTE

The Red Hat Consultant can assist with the development of self-signed certificates, and their use will not affect the user experience of the client's customers.

3. The client will provide systems, virtual machines, or tenant instances for installation of all Red Hat Update Appliances (RHUAs), external load balancers, and content delivery servers (CDSs), configured as described below.
4. Make sure access to RHEL 7 and the RHUI bits (by ISO or subscription) are available.
5. A minimal RHUI installation includes three required servers: one RHUA and two CDSs (physical or virtual) configured as follows:
    a. Red Hat Enterprise Linux (RHEL) 6.7 or greater with **Minimal** installation recommended
    b. SELinux on
    c. Two CPUs, AMD64 processor architecture
    d. 4 GB memory minimum
    e. 10 GB disk for operating system
    f. 50 GB disk per major RHEL release
    g. Each CDS node with a 500 GB local block device dedicated to the GlusterFS brick (if Gluster Storage is used)
6. Certification generation using openssl requires one server, new or existing, configured as follows:
    a. RHEL 6.7 or greater with **Minimal** installation recommended

a. RHEL 6.7 or greater with **Minimal** installation recommended

b. SELinux enabled

c. Two CPUs, AMD64 processor architecture

d. 2 GB memory

e. 6 GB disk for operating system

7. Image certification is performed on RHEL guest templates as provided, typically one RHEL 6 guest and one RHEL 7 guest.

   a. Minimum 10 GB disk for operating system

   b. iptables on

   c. SELinux enabled

   d. If password authentication is on, must use strongest possible hash

   e. Default logging on

8. The client's network must be properly configured for the RHUI.

   a. IP addresses must be allocated for all RHUAs, CDSs, and external load balancers (if any).

   b. DNS records (forward and reverse) have been created for all IP addresses. Example: rhua.company.com, cds1.company.com, cds2.company.com, and certs.company.com

NOTE

If the server has multiple network interface cards (NICs), the fully qualified domain name of the RHUA and the CDSs must be resolved to the IP of the NIC that is used to communicate between the RHUA and the CDSs.

RHUI uses DNS to reach the CDN. In most cases, your instance should be preconfigured to talk to the proper DNS servers hosted as part of the cloud's infrastructure. If you run your own DNS servers or update your client DNS configuration, there is a chance you will see errors similar to **yum Could not contact any CDS load balancers**. In these cases, check that your DNS server is forwarding to the cloud's DNS servers for the request or that your DNS client is configured to fall back to the cloud's DNS server for name resolution.

Using more than one HAProxy node requires a round-robin DNS entry for the host name used as the value of the *--cds-lb-hostname* parameter when rhui-installer is run (*cds.example.com* in this guide) that resolves to the IP addresses of all HAProxy nodes. This Knowledgebase solution presents one way to configure a round-robin DNS. In the context of RHUI, these will be the IP addresses of the HAProxy nodes, and they are to be mapped to the host name specified as *--cds-lb-hostname* while calling rhui-installer.

See HAProxy Configuration for more information.

Red Hat Enterprise Linux 7 uses firewalld for port manipulation, whereas Red Hat Enterprise Linux 6 uses iptables.

9. All required network ports are open.

Table 3.1. Required Network Port Settings

Connection	Port	Usage
RHUA to cdn.redhat.com	443/TCP	Content Delivery
RHUA to CDSs	22/TCP	Initial SSH configuration
RHUA to HAProxy servers	22/TCP	Initial SSH configuration
CDS to RHUA	8140/TCP	Puppet
HAProxy to RHUA	8140/TCP	Puppet
Clients to CDS or HAProxy	443/TCP	
Clients to CDS or HAProxy	5000/TCP	Docker
HAProxy to CDS	443/TCP	Load balancing
HAProxy to CDS	5000/TCP	Docker load balancing
GlusterFS ports	24007/TCP, 49152-4/TCP	Storage
NFS ports	2049/TCP	File system

10. Network proxy settings between RHUA and the Red Hat CDN are configured appropriately.
11. Network proxy settings between the CDSs and the clients via *yum.conf* are configured appropriately.
12. A round-robin DNS entry if more than one HAProxy node is used

Report a bug

CHAPTER 4. REGISTER RED HAT UPDATE INFRASTRUCTURE AND ATTACH SUBSCRIPTIONS

4.1. INSTALL RED HAT ENTERPRISE LINUX

1. Install Red Hat Enterprise Linux on the Red Hat Update Appliance (RHUA), each content delivery server (CDS), and on the HAProxy load balancer if you are using it. See the Red Hat Enterprise Linux 7 Installation Guide for installation details.

2. Attach an appropriate subscription to each node. See Chapter 6 of the Red Hat Enterprise Linux 7 System Administrator's Guide for subscription details.

4.2. REGISTER RED HAT UPDATE INFRASTRUCTURE

1. Register the system that you are going to use as the RHUA instance.

```
[root@rhua ~]# subscription-manager register --type=rhui --username
<admin-example> --password <secret>
Registering to: subscription.rhsm.redhat.com:443/subscription
The system has been registered with ID: <a12b34c5-6d78-9ef1-2345-
ghi678jk91l2m>
```

NOTE

If you are using an existing system as the RHUA and the RHUA has an attached subscription, you will see **`This system is already registered. Use --force to override`** when you try to register it using **`# subscription-manager register --type=rhui`**. You can override the subscription by adding **`--force`** to the command line argument. Another option is to unregister the system (**`# subscription-manager unregister`**) and register it again (**`# subscription-manager register --type=rhui`**).

2. Register each CDS node that will be used unless existing, external, already registered systems are used.

```
[root@cds1 ~]# subscription-manager register --type=rhui --username
<admin-example> --password <secret>
Registering to: subscription.rhsm.redhat.com:443/subscription
The system has been registered with ID: <a12b34c5-6d78-9ef1-2345-
ghi678jk91l2m>
```

3. Register each HAProxy node that will be used unless existing, external, already registered systems are used.

```
[root@haproxy1 ~]# subscription-manager register --type=rhui --
username <admin-example> --password <secret>
Registering to: subscription.rhsm.redhat.com:443/subscription
The system has been registered with ID: <a12b34c5-6d78-9ef1-2345-
ghi678jk91l2m>
```

The new system will be available on the Customer Portal, and the new RHUA instance will not have any subscriptions applied to it.

4.3. ATTACH A SUBSCRIPTION TO THE RED HAT UPDATE APPLIANCE

1. Check for available subscriptions to add to the RHUA.

```
[root@rhua ~]# subscription-manager list --available
+-------------------------------------------+
    Available Subscriptions
+-------------------------------------------+
Subscription Name:   Red Hat Enterprise Linux Atomic Host for
Certified Cloud
                     and Service Providers (via Red Hat Update
Infrastructure)
Provides:            Red Hat Enterprise Linux Atomic Host Beta from
RHUI
                     Red Hat Enterprise Linux Atomic Host from RHUI
SKU:                 RH00731
Contract:            11312089
Pool ID:             8a85f9815a6c4c9d015a6c6acb373ed9
Provides Management: No
Available:           19
Suggested:           1
Service Level:       Premium
Service Type:        L1-L3
Subscription Type:   Standard
Ends:                02/22/2018
System Type:         Physical

Subscription Name:   Red Hat Update Infrastructure and RHEL Add-Ons
for
                     Providers
Provides:            dotNET on RHEL (for RHEL Server) from RHUI
                     Red Hat Enterprise Linux Server from RHUI
                     Red Hat Software Collections (for RHEL Server)
from RHUI
                     Red Hat Enterprise Linux for SAP from RHUI
                     Red Hat Enterprise Linux Resilient Storage
(for RHEL
                     Server) from RHUI
                     Red Hat Enterprise Linux Scalable File System
(for RHEL
                     Server) from RHUI
                     Red Hat Enterprise Linux Server - Extended
Update Support
                     from RHUI
                     dotNET on RHEL Beta (for RHEL Server) from
RHUI
                     Red Hat Enterprise Linux for SAP Hana from
RHUI
                     RHEL Software Test Suite (for RHEL Server)
from RHUI
                     Red Hat Enterprise Linux High Availability
(for RHEL
                     Server) from RHUI
                     Red Hat Update Infrastructure
                     Red Hat Enterprise Linux Load Balancer (for
```

```
RHEL Server)
                    from RHUI
SKU:                RC1116415
Contract:           11314314
Pool ID:            8a85f9815a71f0bd015a72445adf0223
Provides Management: No
Available:          20
Suggested:          1
Service Level:      Premium
Service Type:       L1-L3
Subscription Type:  Standard
Ends:               02/23/2018
System Type:        Physical
```

2. Use the Pool ID for your subscription to attach the subscription. Because there are two SKUs and two subscription names, you need to run **`subscription-manager attach --pool=<Pool ID>`** for each <Pool ID>.

```
# subscription-manager attach --pool=<Pool ID>
Successfully attached a subscription for: <Subscription_Name>

# subscription-manager attach --pool=<Pool ID>
Successfully attached a subscription for: <Subscription_Name>
```

4.4. ATTACH A SUBSCRIPTION TO THE CDS NODES

1. Check for available subscriptions that you can add to the CDS nodes.

```
[root@<cds1> ~]# subscription-manager list --available
+-------------------------------------------+
    Available Subscriptions
+-------------------------------------------+
Subscription Name:   <Your_Subscription_Name>
```

2. Use the Pool ID of **Red Hat Update Infrastructure and RHEL Add-Ons for Providers** subscription. This subscription provides access to Red Hat Enterprise Linux and Gluster Storage.

```
# subscription-manager attach --pool=<Pool_ID>
Successfully attached a subscription for: Red Hat Update
Infrastructure and RHEL Add-Ons for Providers
```

4.5. ATTACH A SUBSCRIPTION TO THE HAPROXY NODES

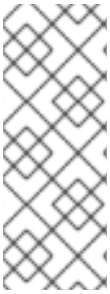
1. Check for available subscriptions that you can add to the HAProxy nodes.

```
[root@<haproxy1> ~]# subscription-manager list --available
+-------------------------------------------+
    Available Subscriptions
+-------------------------------------------+
Subscription Name:   <Your_Subscription_Name>
```

2. Use the Pool ID of **Red Hat Update Infrastructure and RHEL Add-Ons for Providers** subscription. This subscription provides access to Red Hat Enterprise Linux and HAProxy.

```
# subscription-manager attach --pool=<Pool_ID>
Successfully attached a subscription for: Red Hat Update
Infrastructure and RHEL Add-Ons for Providers
```

4.6. ENABLE THE REQUIRED REPOSITORIES



NOTE

The *rhel-7-server-rhui-rpms* repository is the base Red Hat Enterprise Linux repository and should provide the necessary packages. The *rhel-7-server-rhui-rpms* is the same as the *rhel-7-server-rpms* repository and is used automatically when you register the system with **`--type=rhui`**.

The same concept holds true for the relevant Red Hat Enterprise Linux 6 repositories.

The RHUA and CDS nodes require Red Hat Enterprise Linux installations with the base packages and with all repositories disabled except for *rhel-7-server-rpms* (or *rhel-6-server-rpms* for Red Hat Enterprise Linux 6). This requirement also means any third-party configurations or software that is not directly necessary for the direct operation of the server cannot be installed. This restriction includes hardening or other non-Red Hat security software.

1. List the enabled repositories to verify that your system is correctly subscribed.

```
# yum repolist enabled
Loaded plugins: search-disabled-repos
rhel-7-server-rhui-rpms

repo id                                              repo name
status
!local-rhui3                                         local-rhui3
101
!rhui-REGION-client-config-server-7/x86_64           Red Hat Update
Infrastructure 2.0 Client Configuration Server 7       6
!rhui-REGION-rhel-server-releases/7Server/x86_64  Red Hat Enterprise
Linux Server 7 (RPMs)                          13,578
!rhui-REGION-rhel-server-rh-common/7Server/x86_64 Red Hat Enterprise
Linux Server 7 RH Common (RPMs)                  209
repolist: 13,894
```

2. Disable all repositories for the RHUA and enable the relevant repository. Use the first command for RHEL 7 and the second command for RHEL 6.

```
[root@<rhua> ~]# subscription-manager repos --disable=*;
subscription-manager repos --enable=rhel-7-server-rhui-rpms

[root@<rhua> ~]# subscription-manager repos --disable=*;
subscription-manager
repos --enable=rhel-6-server-rhui-rpms
```

3. Enable the RHUI 3 repository. Use the first command for RHEL 7 and the second command for RHEL 6.

```
[root@<rhua> ~]# subscription-manager repos --enable=rhel-7-server-
rhui-3-rpms

[root@<rhua> ~]# subscription-manager repos --enable=rhel-6-server-
rhui-3-rpms
```

4. Disable all repositories for the CDS nodes and enable the relevant repository. Use the first command for RHEL 7 and the second command for RHEL 6.

```
[root@<cds$> ~]# subscription-manager repos --disable=*;
subscription-manager repos --enable=rhel-7-server-rhui-rpms --
enable=rh-gluster-3-for-rhel-7-server-rhui-rpms

[root@<cds$> ~]# subscription-manager repos --disable=*;
subscription-manager repos --enable=rhel-6-server-rhui-rpms --
enable=rhs-3-for-rhel-6-server-rhui-rpms
```

5. Enable the RHUI 3 repository. Use the first command for RHEL 7 and the second command for RHEL 6.

```
[root@<cds$> ~]# subscription-manager repos --enable=rhel-7-server-
rhui-3-rpms

[root@<cds$> ~]# subscription-manager repos --enable=rhel-6-server-
rhui-3-rpms
```

6. Disable all repositories for the HAProxy nodes and enable the relevant repository. Use the first command for RHEL 7 and the second command for RHEL 6.

```
[root@<haproxy$> ~]# subscription-manager repos --disable=*;
subscription-manager repos --enable=rhel-7-server-rhui-rpms

[root@<haproxy$> ~]# subscription-manager repos --disable=*;
subscription-manager repos --enable=rhel-6-server-rhui-rpms --
enable=rhel-lb-for-rhel-6-server-rhui-rpms
```

7. Enable the RHUI 3 repository. Use the first command for RHEL 7 and the second command for RHEL 6.

```
[root@<haproxy$> ~]# subscription-manager repos --enable=rhel-7-
server-rhui-3-rpms

[root@<haproxy$> ~]# subscription-manager repos --enable=rhel-6-
server-rhui-3-rpms
```

Report a bug

CHAPTER 5. SHARED STORAGE

The Red Hat Update Appliance (RHUA) and content delivery servers (CDSs) need a shared storage volume that is accessible by both. Red Hat Gluster Storage is provided with Red Hat Enterprise Linux (RHEL) 6 and 7, but you can use any Network File System (NFS) solution.

5.1. GLUSTER STORAGE

5.1.1. Create Shared Storage

NOTE

glusterfs-server is available only with the appropriate subscription.

See the Red Hat Gluster Storage documentation for installation and administration details. In particular, see Section 11.15 of the Red Hat Gluster Storage 3.4 Administration Guide for split-brain management.

WARNING

As of Red Hat Gluster Storage 3.4, two-way replication without arbiter bricks is considered deprecated. Existing volumes that use two-way replication without arbiter bricks remain supported for this release. New volumes with this configuration are not supported. Red Hat no longer recommends the use of two-way replication without arbiter bricks and plans to remove support entirely in future versions of Red Hat Gluster Storage. This change affects both replicated and distributed-replicated volumes that do not use arbiter bricks.

Two-way replication without arbiter bricks is being deprecated because it does not provide adequate protection from split-brain conditions. Even in distributed-replicated configurations, two-way replication cannot ensure that the correct copy of a conflicting file is selected without the use of a tie-breaking node.

Red Hat strongly recommends using three-node Gluster Storage volumes.

Information about three-way replication is available in Section 5.6.2, Creating Three-way Replicated Volumes and Section 5.7.2, Creating Three-way Distributed Replicated Volumes of the Red Hat Gluster Storage 3.4 Administration Guide.

One concern regarding shared storage is the inability to expand block storage size if the disk usage approaches 100% because of raw disk. Gluster Storage usually works on a physical server, and its bricks are internal storage. With a physical server, the disks of bricks cannot be extended if they are assigned to an entire physical internal disk. According to general storage practice, the brick should be placed on the Logical Volume Manager (LVM.)

The following steps describe how to create a shared volume on LVM using Gluster Storage of three nodes and install required packages. Refer to the product documentation if you are using a different storage solution.

1. Run the following steps on all CDS nodes. The example shows **cds1**.

a. For Red Hat Enterprise Linux 7, run the following command.

```
[root@cds1 ~]# yum install glusterfs-server glusterfs-cli rh-
rhua-selinux-policy
```

b. For Red Hat Enterprise Linux 6, run the following command.

```
[root@cds1 ~]# yum install xfsprogs glusterfs-server glusterfs-
cli rh-rhua-selinux-policy
```

2. Initialize the physical volume on the new disk.

```
# pvcreate /dev/vdb
```

3. Create a Volume Group on /dev/vdb.

```
# vgcreate vg_gluster /dev/vdb
```

4. Create a logical volume on LVM.

```
# lvcreate -n lv_brick1 -l 100%FREE vg_gluster
```

5. Format the device.

```
mkfs.xfs -f -i size=512 /dev/mapper/vg_gluster-lv_brick1
```

6. Create a mount directory, mount the disk, enable glusterd, and start glusterd.

```
# mkdir -p /export/xvdb; mount /dev/mapper/vg_gluster-lv_brick1
/export/xvdb; mkdir -p /export/xvdb/brick; systemctl enable
glusterd.service; systemctl start glusterd.service
```

7. Add the following entry in **/etc/fstab** on each CDS node.

```
/dev/mapper/vg_gluster-lv_brick1 /export/xvdb xfs defaults 0 0
```

8. Run the following steps on only one CDS node, for example, **cds1**.

```
[root@cds1 ~]# gluster peer probe cds2.example.com
peer probe: success.
[root@cds1 ~]# gluster peer probe cds3.example.com
peer probe: success.
```

NOTE

Make sure DNS resolution is working. A bad name resolution error is shown below.

```
[root@cds1 ~]# gluster peer probe <cds[23].example.com hostnames>
 peer probe: failed: Probe returned with Transport endpoint is not
connected
```

IMPORTANT

The Gluster peer probe might also fail with **`peer probe: failed: Probe returned with Transport endpoint is not connected`** when there is a communication or port issue. A workaround to this failure is to disable the firewalld service. If you prefer not to disable the firewall, you can allow the correct ports as described in Section 3.1, Verifying Port Access of the Red Hat Gluster Storage Administration Guide 3.4.

9. Before proceeding, verify that the peer connections were successful. You should see a similar output.

```
[root@cds1 ~]# gluster peer status
Number of Peers: 2
Hostname: cds2.v3.example.com
Uuid: 6cb9fdf9-1486-4db5-a438-24c64f47e63e
State: Peer in Cluster (Connected)
Hostname: cds3.v3.example.com
Uuid: 5e0eea6c-933d-48ff-8c2f-0228effa6b82
State: Peer in Cluster (Connected)
```

```
[root@cds1 ~]# gluster volume create rhui_content_0 replica 3 \
cds1.example.com:/export/xvdb/brick
cds2.example.com:/export/xvdb/brick \
cds3.example.com:/export/xvdb/brick
volume create: rhui_content_0: success: please start the volume to
access data
```

```
[root@cds1 ~]# gluster volume start rhui_content_0
volume start: rhui_content_0: success
```

5.1.2. Extend the Storage Volume

You can extend a disk's volume if it is approaching its capacity by adding a new disk of the same size to each CDS node and running the following commands on each CDS node. The name of the device file representing the disk depends on the technology you use, but if the first disk was **`/dev/vdb`**, the second can be **`/dev/vdc`**. Replace the device file in the following procedure with the actual device file name.

1. Initialize the physical volume on the new disk.

```
# pvcreate /dev/vdc
```

2. Extend the logical volume group.

```
# vgextend vg_gluster /dev/vdc
```

3. Extend the logical volume itself by the amount of free disk space on the new physical volume.

```
# lvextend vg_gluster/lv_brick1 /dev/vdc
```

4. Expand the file system.

```
# xfs_growfs /dev/mapper/vg_gluster-lv_brick1
```

5. Run **df** on the RHUA node to confirm that the mounted Gluster Storage volume has the expected new size.

5.2. CREATE NFS STORAGE

You can set up an NFS server for the content managed by RHUI on the RHUA node or on a dedicated machine. The following procedure describes how to set up storage using NFS.

IMPORTANT

Using a dedicated machine allows CDS nodes, and mainly your RHUI clients, to continue to work if something happens to the RHUA node. Red Hat recommends that you set up an NFS server on a dedicated machine.

1. Install the **nfs-utils** package on the node hosting the NFS server, on the RHUA node (if it differs), and also on all your CDS nodes.

```
# yum install nfs-utils
```

2. Edit the **/etc/exports** file on the NFS server. Choose a suitable directory to hold the RHUI content and allow the RHUA node and all your CDS nodes to access it. For example, to use the **/export** directory and make it available to all systems in the example.com domain, put the following line to **/etc/exports**.

```
/export *.example.com(rw,no_root_squash)
```

3. Create the directory for the RHUI content as defined in **/export**.

```
# mkdir /export
```

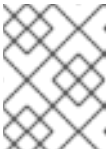
4. Start and enable the NFS service.

   a. On RHEL 7, run the following command.

```
# systemctl start nfs
# systemctl start rpcbind
# systemctl enable nfs-server
# systemctl enable rpcbind
```

   b. On RHEL 6, run the following command.

```
# service nfs start
# service rpcbind start
# chkconfig nfs on
# chkconfig rpcbind on
```



NOTE

If you are using an existing NFS server and the NFS service is already running, use **`restart`** instead of **`start`**.

5. Test your setup. On a CDS node, run the following commands, which assume that the NFS server has been set up on a machine named **`filer.example.com`**.

```
# mkdir /mnt/nfstest
# mount filer.example.com:/export /mnt/nfstest
# touch /mnt/nfstest/test
```

   You should not get any error messages.

6. To clean up after this test, remove the **`test`** file, unmount the remote share, and remove the **`test`** directory.

```
# rm /mnt/nfstest/test
# umount /mnt/nfstest
# rmdir /mnt/nfstest
```

   Your NFS server is now set up. See Section 8.7. NFS Server Configuration for more information on NFS server configuration for RHEL 7. See Chapter 9. Network File System (NFS) for more information on NFS server configuration for RHEL 6.

Report a bug

CHAPTER 6. INSTALL RED HAT UPDATE INFRASTRUCTURE

The following sections describe how to install Red Hat Update Infrastructure.

6.1. GENERATE AN RSA KEY PAIR

IMPORTANT

It is necessary to generate the RSA key pair on the Red Hat Update Appliance (RHUA) node and copy the public key to content delivery server (CDS) and HAProxy nodes so **`rhui-manager`** can set up the CDS and HAProxy nodes.

6.1.1. RSA Key Pair for Version 2 of the SSH Protocol

Follow these steps to generate an RSA key pair for version 2 of the SSH protocol.

1. Generate an RSA key pair.

```
[USER@rhua ~]$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/USER/.ssh/id_rsa):
```

2. Press **`Enter`** to confirm the default location, **`~/.ssh/id_rsa`**, for the newly created key.

3. Leave the passphrase field blank. The CDS installation and registration will fail if a passphrase is provided while generating the key pair.

```
Your identification has been saved in /home/USER/.ssh/id_rsa.
Your public key has been saved in /home/USER/.ssh/id_rsa.pub.
The key fingerprint is:
e7:97:c7:e2:0e:f9:0e:fc:c4:d7:cb:e5:31:11:92:14
USER@rhua.example.com
The key's randomart image is:
+--[ RSA 2048]----+
|             E.  |
|            . .  |
|             o . |
|              . .|
|        S .    . |
|         + o o ..|
|          * * +oo|
|           O +..=|
|           o*  o.|
+-----------------+
```

4. By default, the permissions of the **`~/.ssh/`** directory are set to *rwx------* or *700* expressed in octal notation. This is to ensure that only <$USER> can view the contents. If required, this can be confirmed with the following command.

```
[USER@rhua ~]$ ls -ld ~/.ssh
drwx------. 2 USER USER 54 Nov 25 16:56 /home/USER/.ssh/
```

5. Copy the public key to the HAProxy and CDS nodes.

```
[USER@rhua ~]$ ssh-copy-id user@<haproxy1>
[USER@rhua ~]$ ssh-copy-id user@<cds1>
[USER@rhua ~]$ ssh-copy-id user@<cds2>
```

This command copies the most recently modified **~/.ssh/id*.pub** public key if it is not yet installed. Alternatively, specify the public key's file name.

```
[USER@rhua ~]$ ssh-copy-id -i ~/.ssh/id_rsa.pub user@hostname
```

This command copies the content of **~/.ssh/id_rsa.pub** into the **~/.ssh/authorized_keys** file on the machine to which you want to connect. If the file already exists, the keys are appended to its end.

6.1.2. ECDSA Key Pair for Version 2 of the SSH Protocol

Follow these steps to generate an ECDSA key pair for version 2 of the SSH protocol.

1. Generate an ECDSA key pair.

```
[USER@rhua ~]$ ssh-keygen -t ecdsa
Generating public/private ecdsa key pair.
Enter file in which to save the key (/home/USER/.ssh/id_ecdsa):
```

2. Press **Enter** to confirm the default location, **~/.ssh/id_ecdsa**, for the newly created key.

3. Leave the passphrase field blank. The CDS installation and registration will fail if a passphrase is provided while generating the key pair.

```
[USER@rhua ~]$
Your identification has been saved in /home/USER/.ssh/id_ecdsa.
Your public key has been saved in /home/USER/.ssh/id_ecdsa.pub.
The key fingerprint is:
fd:1d:ca:10:52:96:21:43:7e:bd:4c:fc:5b:35:6b:63
USER@rhua.example.com
The key's randomart image is:
+--[ECDSA  256]---+
|       .+ +o     |
|       . =.o     |
|        o o +  ..|
|         + + o  +|
|        S o o oE.|
|           + oo+.|
|            + o  |
|                 |
|                 |
+-----------------+
```

4. By default, the permissions of the **~/.ssh/** directory are set to **rwx------** or **700** expressed in octal notation. This is to ensure that only <$USER> can view the contents. If required, this can be confirmed.

```
[USER@rhua ~]$ ls -ld ~/.ssh
drwx------. 2 USER USER 54 Nov 25 16:56 /home/USER/.ssh/
```

5. Copy the public key to the HAProxy and CDS nodes.

```
[USER@rhua ~]$ ssh-copy-id user@<haproxy1>
[USER@rhua ~]$ ssh-copy-id user@<cds1>
[USER@rhua ~]$ ssh-copy-id user@<cds2>
```

This command copies the most recently modified **~/.ssh/id*.pub** public key if it is not yet installed. Alternatively, specify the public key's file name.

```
[USER@rhua ~]$ ssh-copy-id -i ~/.ssh/id_ecdsa.pub USER@hostname
```

This command copies the content of **~/.ssh/id_ecdsa.pub** into the **~/.ssh/authorized_keys** on the machine to which you want to connect. If the file already exists, the keys are appended to its end.

6.2. APPLY UPDATES

1. Before installing Red Hat Update Appliance (RHUA) packages, apply any available operating system updates to all nodes (RHUA, content delivery server [CDS], and HAProxy) and reboot.

2. Verify that all configuration changes have persisted.

WARNING

Make sure the host name of the RHUA is set correctly. If the host name is unset and its value is reported as **localhost.localdomain** or **localhost**, you will not be able to proceed.

6.3. MOUNT RED HAT UPDATE INFRASTRUCTURE ISO

NOTE

This section is optional and can be skipped if you are using a subscription to install Red Hat Update Infrastructure 3.0.

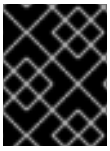
To install the RHUI packages on the RHUA node, CDS nodes, and HAProxy nodes, mount the ISO to a suitable directory (or burn the ISO to a CD, insert the CD, and mount the ISO), and enter the mount point.

6.4. RUN THE SETUP_PACKAGE_REPOS SCRIPT

NOTE

This section is optional and can be skipped if you are using a subscription to install Red Hat Update Infrastructure 3.0.

The **setup_package_repos** script is provided in the root directory of the RHUI ISO image.



IMPORTANT

The **setup_package_repos** script creates a Yum configuration entry that requires the RHUI packages to be signed with the Red Hat Release key.

Perform the following step on the RHUA node, CDS nodes, and HAProxy nodes to install the RHUI packages.

1. Execute the script from the mount point for the applicable system component.

```
[root@rhua ~]# ./setup_package_repos
[root@<cds1> ~]# ./setup_package_repos
[root@<haproxy1> ~]# ./setup_package_repos
```

6.5. INSTALL THE RHUI-INSTALLER SCRIPT

1. Install the **rhui-installer** script.

```
[root@rhua ~]# yum install -y rhui-installer
```

```
This script will install the RHUI packages on the current machine.
- Ensuring we are in an expected directory.
- Copying installation files.
- Creating a Repository File
- Importing the gpg key.
- Installation repository will remain configured for future package
installs.
- Installation media can now be safely unmounted.

Installation packages are now available on this system. If you are
installing a RHUA, please run yum install -y rhui-installer; rhui-
installer.
If you are installing a CDS, please log into the RHUA and run rhui-
manager to begin the installation. Do not run rhui-installer to
install a CDS.
```

6.6. RUN THE RHUI-INSTALLER

The rhui-installer sets the initial Red Hat Update Infrastructure login password and displays it in its output. It is also written in the */etc/rhui-installer/answers.yaml* file. You can override the initial password with the **`--rhui-manager-password`** option. If you want to change the initial password later, you can only change it via the rhui-manager tool. Run the **`rhui-installer --help`** command to see the full list of rhui-installer options.

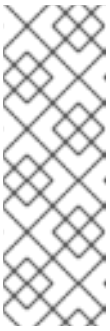
1. Run **`rhui-installer`** on the RHUA.

```
[root@rhua ~]# rhui-installer --remote-fs-type=glusterfs --remote-
fs-server=cds1.example.com:rhui_content_0 --cds-lb-
hostname=cds.example.com

Installing          Done
[100%]
```

```
[.....................................................................
...........]

Success!
The initial credentials are admin / <system-generated password>

Re-running the installer will not update your password.

The full log is at /var/log/kafo/configuration.log
```

Following are explanations of the command arguments.

- **`--remote-fs-type=glusterfs`** means the remote file system type is GlusterFS.
- **`--remote-fs-server=cds1.example.com`** means the name of the remote file system server is cds1.example.com
- **`rhui_content_0`** means the name of the GlusterFS volume on cds1.example.com
- **`--cds-lb-hostname=cds.example.com`** means the name of the load balancer on cds1.example.com is cds.example.com.



NOTE

During installation, the cds-lb-hostname option is not included and prepopulates from the answers file provided with the rhui-installer RPM. The host name is preset in the answers file to *cds.example.com*, and certificates are created for the RHUI environment with this cds-lb-hostname included. See the procedure in Chapter 19, *Migrate to a New Load Balancer, or Change the Name of an Existing Load Balancer* for details on changing the name of a load balancer.

If using NFS, the rhui-installer command line is different. Instead of

```
--remote-fs-type=glusterfs --remote-fs-
server=cds1.example.com:rhui_content_0
```

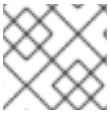
specify the NFS server name and the exported directory, joined by the colon sign, as the parameter of the **`--remote-fs-server`** option. For example:

```
--remote-fs-server=filer.example.com:/export
```

2. Verify that the remote share is mounted.

```
[root@rhua ~]# mount | grep rhui

cds1.example.com:rhui_content_0 on /var/lib/rhui/remote_share type
fuse.glusterfs
(rw,relatime,user_id=0,group_id=0,default_permissions,allow_other,ma
x_read=131072)
```

6.7. CHANGE THE INITIAL PASSWORD



NOTE

Rerunning the rhui-installer will not update the rhui-manager login password.

1. Navigate to the Red Hat Update Infrastructure Management Tool home screen.

```
[root@rhua ~]# rhui-manager

Previous authentication credentials could not be found. Logging into
the RHUI.

If this is the first time using the RHUI, it is recommended to
change the user's password in the User Management section of RHUI
Tools.
```

2. Enter the RHUI Username (admin) and RHUI Password (provided by the rhui-installer output). The initial password is also stored in **`/etc/rhui-installer/answers.yaml`**.

3. After successfully logging in for the first time, you should change the password. Press **u** on the Red Hat Update Infrastructure Management Tool home screen to select **manage RHUI users**.

```
              -= Red Hat Update Infrastructure Management Tool =-

-= Home =-

   r   manage repositories
   c   manage content delivery servers (CDS)
   l   manage HAProxy load-balancer instances
   s   synchronization status and scheduling
   e   create entitlement certificates and client configuration RPMs
   n   manage Red Hat entitlement certificates
   sm  manage Red Hat subscriptions
   u   manage RHUI users

                                                        Connected:
rhua.example.com
```

4. Press **p** to select **p change a user's password (followed by logout)**.

```
--------------------------------------------------------------------
--------------
= Red Hat Update Infrastructure Management Tool =

= User Manager =

  p   change a user's password (followed by logout)

                                                  Connected:
rhua.example.com

--------------------------------------------------------------------
--------------
rhui (users) => p
```

```
Warning: After password change you will be logged out.
Use ctrl-c to cancel password change.
Username: admin
```

5. Enter the new password and press **Enter**. Re-enter the new password and press **Enter**.

```
New Password:
Re-enter Password:

Password successfully updated.
------------------------------------------------------------------
--------------
```

6.8. REGISTER A RED HAT SUBSCRIPTION IN RHUI

1. Navigate to the **Red Hat Update Infrastructure Management Tool** home screen.

```
[root@rhua ~]# rhui-manager
```

2. Press **sm** to select **manage Red Hat subscriptions**.

```
             -= Red Hat Update Infrastructure Management Tool =-

-= Home =-

   r   manage repositories
   c   manage content delivery servers (CDS)
   l   manage HAProxy load-balancer instances
   s   synchronization status and scheduling
   e   create entitlement certificates and client configuration RPMs
   n   manage Red Hat entitlement certificates
   sm  manage Red Hat subscriptions
   u   manage RHUI users

                                                      Connected:
rhua.example.com
```

3. Type **r** to select **register a Red Hat subscription in RHUI**.

```
             -= Red Hat Update Infrastructure Management Tool =-


-= Subscriptions Manager =-

   l   list registered Red Hat subscriptions
   a   list available Red Hat subscriptions
   r   register a Red Hat subscription in RHUI
   d   unregister a Red Hat subscription in RHUI

                                                      Connected:
rhua.example.com
```

4. Register the subscriptions by entering the number beside each subscription that you want to include, or enter **a** to select all of them.

5. Press **c** when your are finished selecting the subscriptions. The Red Hat Update Infrastructure Management Tool displays the subscriptions to be registered and prompts for confirmation.

6. Press **y** to proceed. A screen message indicates each successful registration.

7. Check that the correct subscriptions have been registered by pressing **l** to access the **list registered Red Hat subscriptions** screen.

Report a bug

CHAPTER 7. ADD A CONTENT DELIVERY SERVER

The Red Hat Update Infrastructure Management Tool provides several options for configuring a content delivery server (CDS) within the Red Hat Update Infrastructure (RHUI). Adding a CDS differs in Red Hat Update Infrastructure 3.0 from Red Hat Update Infrastructure 2.1.3. Rather than adding a preconfigured CDS as done in Red Hat Update Infrastructure 2.1.3, the **Add** step in the Red Hat Update Appliance (RHUA) also installs CDS services.

1. Make sure sshd is running on the CDS node and that ports 443 and 5000 are open.

2. Navigate to the Red Hat Update Infrastructure Management Tool home screen.

```
[root@rhua ~]# rhui-manager
```

3. Press **c** to select **manage content delivery servers (CDS)**.

```
             -= Red Hat Update Infrastructure Management Tool =-

-= Home =-

   r   manage repositories
   c   manage content delivery servers (CDS)
   l   manage HAProxy load-balancer instances
   s   synchronization status and scheduling
   e   create entitlement certificates and client configuration RPMs
   n   manage Red Hat entitlement certificates
   sm  manage Red Hat subscriptions
   u   manage RHUI users

                                                        Connected:
rhua.example.com
```

4. Type **a** to select **register (add) a new Content Delivery Server instance**.

```
--------------------------------------------------------------------
--------------
= Red Hat Update Infrastructure Management Tool =

= Content Delivery Server (CDS) Management =

l   list all known CDS instances managed by the RHUI
a   register (add) a new CDS instance
r   reinstall and reapply configuration to an existing CDS instance
d   unregister (delete) a CDS instance from the RHUI

                                                  Connected:
rhua.example.com
--------------------------------------------------------------------
--------------
rhui (cds) => a
```

5. Enter the host name of the CDS to add.

```
Hostname of the CDS instance to register:
cds1.example.com
```

6. Enter the user name that will have SSH access to the CDS and have sudo privileges.

```
Username with SSH access to <cds1.example.com> and sudo privileges:
root
```

7. Enter the absolute path to the SSH private key for logging in to the CDS and press **Enter**.

```
Absolute path to an SSH private key to log into <cds1.example.com>
as root:
/root/.ssh/id_rsa
.....................................................................
.....
The following CDS has been successfully added:

Hostname:             <cds1.example.com>
SSH Username:      root
SSH Private Key:  /root/.ssh/id_rsa

The CDS will now be configured:
.....
The CDS was successfully configured.
```

8. If adding the content delivery server fails, check that the CDS daemon pulp-cds is running and that the firewall rules permit access between the RHUA and the CDS.

9. Run the **mount** command to see if Gluster Storage is mounted as read-write.

```
[root@rhua ~]# mount | grep cds1.example.com

cds1.example.com:rhui_content_0 on /var/lib/rhui/remote_share type
fuse.glusterfs
(rw,relatime,user_id=0,group_id=0,default_permissions,allow_other,ma
x_read=131072)
```

10. After successful configuration, repeat these steps for any remaining CDSs. You can also add a CDS using the command-line interface.

```
[root@rhua ~]# rhui cds add cds1.example.com root /root/.ssh/id_rsa
-u
```

Report a bug

CHAPTER 8. ADD AN HAPROXY LOAD BALANCER

If you are using multiple HAProxy nodes and have not set up a round-robin DNS entry for the host name that you used as the value of the *--cds-lb-hostname* parameter when you ran rhui-installer (as instructed in Section 6.6, "Run the rhui-installer") that resolves to the IP addresses of all HAProxy nodes, see Chapter 3, *Prerequisites for Installing Red Hat Update Infrastructure* for more details.

1. Make sure sshd is running on the HAProxy node and that ports 443 and 5000 are open.

2. Navigate to the Red Hat Update Infrastructure Management Tool home screen.

```
[root@rhua ~]# rhui-manager
```

3. In the **Red Hat Update Infrastructure Management Tool** home screen, press **l** to select **manage HAProxy load-balancer instances**.

```
              -= Red Hat Update Infrastructure Management Tool =-

-= Home =-

   r   manage repositories
   c   manage content delivery servers (CDS)
   l   manage HAProxy load-balancer instances
   s   synchronization status and scheduling
   e   create entitlement certificates and client configuration RPMs
   n   manage Red Hat entitlement certificates
   sm  manage Red Hat subscriptions
   u   manage RHUI users

                                                        Connected:
rhua.example.com
```

4. Press **a** to select **register (add) a new HAProxy Load-balancer instance**.

```
--------------------------------------------------------------------
--------------
= Red Hat Update Infrastructure Management Tool =

= Load-balancer (HAProxy) Management =

l   list all known HAProxy Load-balancer instances managed by the
RHUI
a   register (add) a new HAProxy Load-balancer instance
r   reinstall and reapply configuration to an existing HAProxy Load-
balancer instance
d   unregister (delete) a HAProxy Load-balancer instance from the
RHUI

                                                  Connected:
rhua.example.com
--------------------------------------------------------------------
--------------

rhui (loadbalancers) => a
```

5. Enter the host name of the HAProxy to add.

```
Hostname of the HAProxy Load-balancer instance to register:
haproxy1.example.com
```

6. Enter the user name that will have SSH access to the HAProxy load balancer and have sudo privileges.

```
Username with SSH access to haproxy1.example.com and sudo
privileges:
root
```

7. Enter the absolute part to the SSH private key for logging in to the HAProxy load balancer and press **Enter**.

```
Absolute path to an SSH private key to log into haproxy1.example.com
as root:
/root/.ssh/id_rsa
.....................................................................
.....

The following HAProxy Load-balancer has been successfully added:

Hostname:            <haproxy1.example.com>
SSH Username:   root

SSH Private Key:  /root/.ssh/id_rsa

The HAProxy Load-balancer will now be configured:
……..

The HAProxy Load-balancer was successfully configured.
```

8. After successful configuration, repeat these steps for any remaining HAProxy load balancers. You can also add an HAProxy load balancer through the command-line interface.

```
# rhui haproxy add <haproxy1.example.com> root /root/.ssh/id_rsa -u
```

Report a bug

CHAPTER 9. CREATE AND SYNCHRONIZE A RED HAT REPOSITORY

9.1. CREATE A REPOSITORY

1. On the Red Hat Update Appliance (RHUA), log in to the Red Hat Update Infrastructure Management Tool.

```
[root@rhua ~]# rhui-manager
```

2. In the Red Hat Update Infrastructure Management Tool home screen, press **r** to select **manage repositories**.

```
             -= Red Hat Update Infrastructure Management Tool =-

-= Home =-

   r   manage repositories
   c   manage content delivery servers (CDS)
   l   manage HAProxy load-balancer instances
   s   synchronization status and scheduling
   e   create entitlement certificates and client configuration RPMs
   n   manage Red Hat entitlement certificates
   sm  manage Red Hat subscriptions
   u   manage RHUI users

                                                        Connected:
rhua.example.com
```

3. Press **a** to select **add a new Red Hat content repository**.

```
------------------------------------------------------------------
------------------
         -= Red Hat Update Infrastructure Management Tool =-

= Repository Management =-

l   list repositories currently managed by the RHUI
i   display detailed information on a repository
a   add a new Red Hat content repository
c   create a new custom repository (RPM content only)
d   delete a repository from the RHUI
u   upload content to a custom repository (RPM content only)
p   list packages in a repository (RPM content only)

                                                  Connected:
rhua.example.com
------------------------------------------------------------------
--------------

rhui (repo) => a
```

4. Wait for the Red Hat Update Infrastructure Management Tool to determine the entitled repositories. This might take several minutes.

```
                                                    Connected:
rhua.example.com

--------------------------------------------------------------------------------

rhui (repo) => a

Loading latest entitled products from Red Hat...

Determining undeployed products...
```

5. The Red Hat Update Infrastructure Management Tool prompts for a selection method.

```
... product list calculated

Import Repositories:

1  - All in Certificate
2  - By Product
3  - By Repository

Enter value (1-3) or 'b' to abort:
```

6. Press **2** to select the **By Product** method.

7. Add Red Hat repositories to the RHUA by entering the number beside each repository that you want to include. The only repositories that will display are Red Hat repositories that are included in your entitlement certificate but have not yet been added.

8. Press **c** when your are finished selecting the repositories. The Red Hat Update Infrastructure Management Tool displays the repositories to be deployed and prompts for confirmation.

9. Press **y** to proceed. A screen message indicates each successful deployment.

10. Check that the correct repositories have been installed by pressing **1** to access the **list repositories currently managed by the RHUI** screen.

9.2. SYNCHRONIZE A REPOSITORY

The initial synchronization of Red Hat content can take a while, typically 10 to 20 minutes. Begin synchronizing as soon as possible.

1. Navigate to the Red Hat Update Infrastructure Management Tool home screen.

```
[root@rhua ~]# rhui-manager
```

2. Press **s** to select **synchronization status and scheduling**.

```
              -= Red Hat Update Infrastructure Management Tool =-
```

```
-= Home =-

   r   manage repositories
   c   manage content delivery servers (CDS)
   l   manage HAProxy load-balancer instances
   s   synchronization status and scheduling
   e   create entitlement certificates and client configuration RPMs
   n   manage Red Hat entitlement certificates
   sm  manage Red Hat subscriptions
   u   manage RHUI users

                                                        Connected:
rhua.example.com
```

3. Press **sr** to select **sync an individual repository immediately**.

```
------------------------------------------------------------------
--------------
         -= Red Hat Update Infrastructure Management Tool =-

-= Synchronization Status =-

dr  display repo sync summary
vr  view the details of the last repository sync
sr  sync an individual repository immediately

                                                  Connected:
rhua.example.com

------------------------------------------------------------------
--------------
```

4. Select the repository and press **c** to confirm.

5. Press **y** to proceed.

6. Enter **dr** to select **display repo sync summary**.

```
------------------------------------------------------------------
--------------

rhui (sync) => dr
------------------------------------------------------------------
--------------
         -= Red Hat Update Infrastructure Management Tool =-

-= Repository Synchronization Status =-

Last Refreshed: 13:59:27

(updated every 5 seconds, ctrl+c to exit)

Next Sync                Last Sync                Last Result

------------------------------------------------------------------
```

```
--------------

Red Hat Enterprise Linux 7 Server - Extras from RHUI (RPMs) (x86_64)

02-29-2016 19:54          02-29-2016 13:59          Success

                                                Connected:
rhua.example.com
------------------------------------------------------------------
--------------
```

9.3. CHECK REPOSITORY SYNCHRONIZATION

1. Run the following command to check whether a repository is being synchronized.

```
[root@rhua ~]# rhui-manager status
```

2. If you see a running job, as shown below, **do not reboot**. Wait until the result is something other than **Running**.

```
[root@rhua ~]# rhui-manager status
Red Hat Enterprise Linux High Availability (for RHEL 6 Server)
(RPMs) from RHUI (6Server-x86_64)  Success

Red Hat Enterprise Linux High Availability (for RHEL 7 Server)
(RPMs) from RHUI (7Server-x86_64)  Running
```

3. If you must reboot, or if the system was rebooted for a reason beyond your control, check the output of **`rhui-manager status`** when the system is up. You can also check the Pulp task list with the following command.

```
[root@rhua ~]# pulp-admin -u admin -p admin tasks list
```

4. Use the actual password of the admin user after the **`-p`** switch. If you see **No tasks found**, then the synchronization process was safely interrupted. You can now synchronize the repository manually in rhui-manager, or you can wait for the next available time slot, during which the repository will synchronize automatically.

5. Run **`rhui-manager status`** or the aforementioned pulp-admin command at any time to determine the status of the repository.
   Sometimes an attempt to synchronize a repository whose synchronization was interrupted can fail. The state is **Running**, and there is also a running Pulp task, but nothing is being transferred. If the status has been **Running** for an excessive amount of time, check whether there is an issue by examining the **`repository synchronization status`**.

6. Find out the name of the repository in the Pulp task list. For example,

```
[root@rhua ~]# pulp-admin -u admin -p admin tasks list
+------------------------------------------------------------------
-------+
                                  Tasks
+------------------------------------------------------------------
-------+
```

```
Operations:  sync
Resources:   rhel-x86_64-6-rhui-2-rpms-6Server-x86_64 (repository)
State:       Running
Start Time:  2017-05-24T08:16:48Z
Finish Time: Incomplete
Task Id:     6e44a32d-2e11-41f2-bbe6-996531c5cda0

Operations:  publish
Resources:   rhel-x86_64-6-rhui-2-rpms-6Server-x86_64 (repository)
State:       Waiting
Start Time:  Unstarted
Finish Time: Incomplete
Task Id:     13f5d531-2f0d-4a73-9d27-b4fd126b6f13
```

7. Check the synchronization status. If you see the following output, there is a problem.

```
[root@rhua ~]# pulp-admin -u admin -p admin rpm repo sync status --
repo-id rhel-x86_64-6-rhui-2-rpms-6Server-x86_64
+----------------------------------------------------------------
-------+
      Repository Status [rhel-x86_64-6-rhui-2-rpms-6Server-x86_64]
+----------------------------------------------------------------
-------+

This command may be exited via ctrl+c without affecting the request.


Downloading metadata...
[-]
... completed

Downloading repository content...
[-]
[==================================================] 100%
RPMs:       0/0 items
Delta RPMs: 0/0 items

... completed

Downloading distribution files...
[==================================================] 100%
Distributions: 0/0 items
... completed

Importing errata...
[-]
... completed

Importing package groups/categories...
[-]
... completed

Cleaning duplicate packages...
[-]
... completed
```

8. You have to cancel the open Pulp tasks and try synchronizing the affected repository again to resolve the problem and ensure that the repository is available on the RHUA. Use pulp-admin again as follows to view the task IDs.

```
[root@rhua ~]# pulp-admin -u admin -p admin tasks list
+----------------------------------------------------------------
-------+
                                  Tasks
+----------------------------------------------------------------
-------+

Operations:  sync
Resources:   rhel-x86_64-6-rhui-2-rpms-6Server-x86_64 (repository)
State:       Running
Start Time:  2017-05-24T08:16:48Z
Finish Time: Incomplete
Task Id:     6e44a32d-2e11-41f2-bbe6-996531c5cda0

Operations:  publish
Resources:   rhel-x86_64-6-rhui-2-rpms-6Server-x86_64 (repository)
State:       Waiting
Start Time:  Unstarted
Finish Time: Incomplete
Task Id:     13f5d531-2f0d-4a73-9d27-b4fd126b6f13
```

9. Cancel the tasks.

```
[root@rhua ~]# pulp-admin -u admin -p admin tasks cancel --task-id
6e44a32d-2e11-41f2-bbe6-996531c5cda0
Task cancel is successfully initiated.

[root@rhua ~]# pulp-admin -u admin -p admin tasks cancel --task-id
13f5d531-2f0d-4a73-9d27-b4fd126b6f13
Task cancel is successfully initiated.
```

10. Run **`rhui-manager status`** and verify that the status is **Canceled**.

11. Try synchronizing the repository again in rhui-manager. Use the same pulp-admin commands as earlier to monitor the progress. If you run into the same problem again, the repository is likely in an irreparable state and will have to be removed from RHUA and added again.

12. View the Pulp task list to obtain the task IDs.

13. Cancel the running and waiting tasks.

14. Use **`rhui-manager`** to remove the repository.

15. In edge cases, the repository cannot be removed because rhui-manager reports: "Task deletion is still occurring, any actions taken while this is not complete can result in errors." If that happens, restart the Pulp services.

    a. For RHEL 6:

```
# service pulp_workers restart; service pulp_resource_manager
restart; service pulp_celerybeat restart
```

b. For RHEL 7:

```
# systemctl restart pulp_workers; systemctl restart
pulp_resource_manager; systemctl restart pulp_celerybeat
```

16. Replace **`restart`** with **`status`** in the above commands to verify each service has restarted.
17. Add the repository again in rhui-manager.
18. Synchronize it immediately, or wait for the next automated synchronization.

Report a bug

CHAPTER 10. CLIENT ENTITLEMENT CERTIFICATE AND CLIENT CONFIGURATION RPM

10.1. CREATE AN ENTITLEMENT CERTIFICATE

1. On the RHUA, log in to the Red Hat Update Infrastructure Management Tool.

```
[root@rhua ~]# rhui-manager
```

2. In the Red Hat Update Infrastructure Management Tool home screen, press **e** to select **create entitlement certificates and client configuration RPMs**.

```
              -= Red Hat Update Infrastructure Management Tool =-

-= Home =-

   r   manage repositories
   c   manage content delivery servers (CDS)
   l   manage HAProxy load-balancer instances
   s   synchronization status and scheduling
   e   create entitlement certificates and client configuration RPMs
   n   manage Red Hat entitlement certificates
   sm  manage Red Hat subscriptions
   u   manage RHUI users

                                                        Connected:
rhua.example.com
```

3. Press **e** to select **generate an entitlement certificate**.

```
--------------------------------------------------------------------
--------------
         -= Red Hat Update Infrastructure Management Tool =-

-= Client Entitlement Management =-

e   generate an entitlement certificate
c   create a client configuration RPM from an entitlement
certificate

                                                  Connected:
rhua.example.com
--------------------------------------------------------------------
--------------

rhui (client) => e
```

4. Select which repositories to include in the entitlement certificate by typing the number of the repository at the prompt. Typing the number of a repository places a checkmark next to the name of that repository. Continue until all repositories you want to add have been checked.

IMPORTANT

Include only repositories for a single Red Hat Enterprise Linux version in a single entitlement. Adding repositories for multiple Red Hat Enterprise Linux versions will lead to an unusable yum configuration file.

5. Press **c** at the prompt to confirm.

6. Enter a name for the certificate. This name helps identify the certificate within the Red Hat Update Infrastructure Management Tool and to generate the name of the certificate and key files.

```
Name of the certificate. This will be used as the name of the
certificate file
(name.crt) and its associated private key (name.key). Choose
something that will
help identify the products contained with it:
```

7. Enter a path to save the certificate to. Leave the field blank to save it to the current working directory.

8. Enter the number of days the certificate should be valid for. Leave the field blank for 365 days. The details of the repositories to be included in the certificate display.

9. Press **y** at the prompt to confirm the information and create the entitlement certificate.

```
Repositories to be included in the entitlement certificate:

Red Hat Repositories

 Red Hat Enterprise Linux for SAP (RHEL 6 Server) (RPMs) from RHUI

Proceed? (y/n) y
.................................................+++
Entitlement certificate created at
/root/clientcert/rhuiclientexample.crt

----------------------------------------------------------------
--------------
```

10.2. CREATE A CLIENT CONFIGURATION RPM

1. From the Red Hat Update Infrastructure Management Tool home screen, press **e** to select **create entitlement certificates and client configuration RPMs**.

```
              -= Red Hat Update Infrastructure Management Tool =-

-= Home =-

   r   manage repositories
   c   manage content delivery servers (CDS)
   l   manage HAProxy load-balancer instances
   s   synchronization status and scheduling
   e   create entitlement certificates and client configuration RPMs
   n   manage Red Hat entitlement certificates
```

```
   sm  manage Red Hat subscriptions
   u   manage RHUI users

                                                        Connected:
rhua.example.com
```

2. Press **c** to select **create a client configuration RPM from an entitlement certificate**.

3. Enter the full path of a local directory to save the configuration files to.

```
Full path to local directory in which the client configuration files
generated by this tool
should be stored (if this directory does not exist, it will be
created):

/tmp
```

4. Enter the RPM's name.

```
clientrpmtest
```

5. Enter the version of the configuration RPM. The default version is 2.0.

6. Enter the full path to the entitlement certificate authorizing the client to access specific channels.

```
Full path to the entitlement certificate authorizing the client to
access
specific channels:
/root/clientcert/rhuiclientexample.crt
```

7. Enter the full path to the private key for the entitlement certificate.

```
Full path to the private key for the above entitlement certificate:
/root/clientcert/rhuiclientexample.key
```

8. Enter the port to serve Docker content on. Port 5000 is the default.

```
Port to serve Docker content on (default 5000):
```

9. Select any unprotected custom repositories to be included in the client configuration.

```
- 1 : unprotected_repo1
```

10. Press **c** to confirm selections or **?** for more commands.

```
Successfully created client configuration RPM.
Location: /tmp/clientrpmtest-2.0/build/RPMS/noarch/clientrpmtest-
2.0-1.noarch.rpm
```

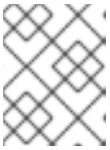
10.3. INSTALL A CLIENT RPM

1. Copy the RPM to the client machine and install it using yum.

```
[client1 ~]# yum localinstall <rpm>
```

2. Verify the RPM has been installed.

```
[client1 ~]# yum list <rpm>
```



NOTE

The rhn-plugin and subscription-manager plugin will be disabled in yum after the client RPM has been installed.

3. View yum repositories to ensure the repository was added and packages are available for installing.

```
[client1 ~]# yum repolist
Loaded plugins: search-disabled-repos, security
rhui-rhel-sap-for-rhel-6-server-rhui-rpms                | 2.0 kB
00:00
rhui-rhel-sap-for-rhel-6-server-rhui-rpms/primary        |  26 kB
00:00
rhui-rhel-sap-for-rhel-6-server-rhui-rpms
77/77
repo id                                    repo name
status
rhui-rhel-sap-for-rhel-6-server-rhui-rpms Red Hat Enterprise Linux
for SA 77
repolist: 77
```

4. Install a package from that repository.

```
[client1 ~]# yum install compat-locales-sap
```

10.4. WORKING WITH THE EUS CHANNEL

By default, clients are set to take content from the main release channel, for example, 7Server or 6Server. If you have synchronized the EUS channel, you need to set the concrete version of the release. After installing the client RPM, the client machine has the **rhui-set-release** tool available.

1. To set the yum **releasever** variable to **version**, which creates the **/etc/yum/vars/releasever** file, which in turn makes yum use EUS repositories with this particular version, run the following command.

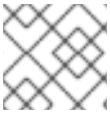
```
rhui-set-release --set <version>
```

2. To unset the **releasever** variable (to remove the file), run the following command.

```
rhui-set-release --unset
```

3. To print the currently set version, run the following command.

```
rhui-set-release
```



NOTE

If there is no output, it means no particular version is set.

Report a bug

CHAPTER 11. CREATE CLIENT PROFILES FOR THE RED HAT UPDATE INFRASTRUCTURE SERVERS

11.1. GENERATE GPG KEYS

1. Create a GPG key that you can use to sign custom packages (including client configuration RPMs) for the Red Hat Update Infrastructure (RHUI) client profile.

   - A 4,096-bit RSA key is used because this profile will be used for RHUI servers that run on Red Hat Enterprise Linux (RHEL) 6 or RHEL 7. Gathering sufficient random data to generate a 4,096-bit key may take a significant amount of time, particularly if the Red Hat Update Appliance (RHUA) is a virtual machine. The disk activity created by a repository or content delivery server (CDS) synchronization may speed up the process.

   - The name of the client profile RPM (in this case, **rhui-client-rhui**), which will be created in a later step, is used as the comment portion of the user ID. It is recommended that a different signing key be used for each client profile; the client profile name is used to distinguish the user IDs of the different keys.

```
# gpg --gen-key

gpg (GnuPG) 2.0.14; Copyright (C) 2009 Free Software Foundation,
Inc.

This is free software; you are free to change and redistribute
it. There is NO WARRANTY, to the extent permitted by law.

Please select what kind of key you want:

1. RSA and RSA (default)
2. DSA and Elgamal
3. DSA (sign only)
4. RSA (sign only)

Your selection? 4

RSA keys may be between 1024 and 4096 bits long.

What keysize do you want? (2048) 4096
Requested keysize is 4096 bits

Please specify how long the key should be valid.

0 = key does not expire
 = key expires in n days
w = key expires in n weeks
m = key expires in n months
y = key expires in n years

Key is valid for? (0) 0
Key does not expire at all.

Is this correct? y

GnuPG needs to construct a user ID to identify your key.
```

```
Real name: $YOURNAME

Email address: $USER@$HOST.com

Comment: rhui-client-rhui

You selected this user ID:

“$USERID (rhui-client-rhui) <$USER@$HOST.com>"

Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit? o

You need a Passphrase to protect your secret key.
```

2. Enter a high-quality password and record it in a secure location.

```
gpg: key EDD092F4 marked as ultimately trusted

public and secret key created and signed.

gpg: checking the trustdb

gpg: 3 marginal(s) needed, 1 complete(s) needed, PGP trust model

gpg: depth: 0   valid:   1   signed:   0   trust:  0-, 0q, 0n, 0m,
0f, 1u

pub    4096R/EDD092F4  2015-11-25

Key fingerprint = 1139 932A 26E2 981A 1341 D636 0DDB B5F6 EDD0 925F4

uid Red Hat $USERID (rhui-client-rhui) <user@redhat.com>

Note that this key cannot be used for encryption. You may want to use
the command “--edit-key” to generate a subkey for this purpose.
```

3. Create a second key. This time choose option 3, DSA (sign only), as the key type and enter 1024 bits as the key size. These options create a key that can be used to sign RPMs for both RHEL 6 and RHEL 7. Use **`rhui-client-all`** as the comment portion of the user ID.

4. Export the two keys by running the following commands.

```
# mkdir /root/rpm-gpg
# gpg --export --armor rhui-client-rhui >> /root/rpm-gpg/rhui-
client-rhui
# gpg --export --armor rhui-client-all >> /root/rpm-gpg/rhui-client-
all
```

GPG defaults to substring matching when searching for keys. It is only necessary to specify the unique portion of the user ID (the client profile RPM name in this case). The traditional RPM-GPG-KEY- prefix will be added to the GPG key file names when the Red Hat Update Infrastructure Management Tool creates client configuration packages.

11.2. SET UP CUSTOM REPOSITORIES

Create custom repositories that can be used to distribute updated client configuration packages or other non-Red Hat software to the RHUI servers. A protected repository for 64-bit RHUI servers (for example, *client-rhui-x86_64*) will be the preferred vehicle for distributing new non-Red Hat packages (such as an updated client configuration package) to the RHUI servers.

Like Red Hat content repositories, all of which are protected, protected custom repositories that differ only in processor architecture (i386 versus AMD64) are consolidated into a single entitlement within an entitlement certificate, using the **`$basearch`** yum variable.

In the event of certificate problems, an unprotected repository for RHUI servers can be used as a fallback method for distributing updated RPMs to the RHUI servers.

1. Navigate to the Red Hat Update Infrastructure Management Tool home screen.

```
[root@rhua ~]# rhui-manager
```

2. From the **Repository Management** screen, press **c** to select **create a new custom repository (RPM content only)**.

```
--------------------------------------------------------------------------------
             -= Red Hat Update Infrastructure Management Tool =-

-= Repository Management =-

   l   list repositories currently managed by the RHUI
   i   display detailed information on a repository
   a   add a new Red Hat content repository
   ad  add a new Red Hat docker container
   c   create a new custom repository (RPM content only)
   d   delete a repository from the RHUI
   u   upload content to a custom repository (RPM content only)
   p   list packages in a repository (RPM content only)

                                                    Connected:
rhua.example.com
--------------------------------------------------------------------------------
```

3. Enter a unique ID for the repository. Only alphanumeric characters, _ (underscore), and - (hyphen) are permitted. You cannot use spaces in the unique ID. For example, *repo1*, *repo_1*, and *repo-1* are all valid entries.

```
Unique ID for the custom repository (alphanumerics, _, and - only):
```

4. Enter a display name for the repository. This name is used to identify the repository within the Red Hat Update Infrastructure Management Tool.

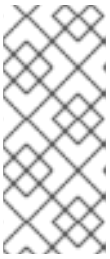
5. Specify the path that will host the repository. The path must be unique across all repositories hosted by RHUI. For example, if you specify the path at this step as *some/unique/name*, then the repository will be located at *//server/pulp/repos/some/unique/name*.

6. Select **sha256** as the checksum type to be used for the repository metadata.

NOTE

Use **sha256** when you create a custom repository for RHEL 6 or RHEL 7. Use **sha1** if you create repositories for RHEL 5 client.

7. Choose whether to protect the new repository. If you answer no to this question, any client can access the repository. If you answer yes, only clients with an appropriate entitlement certificate can access the repository.



NOTE

As the name implies, the content in an unprotected repository is available to any system that requests it, without any need for a client entitlement certificate. Be careful when using an unprotected repository to distribute any content, particularly content such as updated client configuration RPMs, which will then provide access to protected repositories.

Use of unprotected repositories is a “break glass in case of emergency” course of action.

If you choose to protect the new repository, the Red Hat Update Infrastructure Management Tool will ask for the entitlement path. It will also suggest the entitlement path based on the repository’s relative path.

Client entitlement certificates contain the download URLs that they are allowed to access. The RHUI analyzes the contents of the certificate to determine if the repository requested matches any of the permitted URLs, which determines whether to allow the client to authenticate. For example, if an entitlement certificate grants access to */some/unique/name* and the request is made to a repository located at *//server/pulp/repos/some/unique/name/os/repodata*, the RHUI will approve the request and grant the authentication because the path begins with one of the entitled download URLs. The URL only needs to begin with the correct information; it does not need to match exactly.

NOTE

If the */some/unique/name* repository that was created in pulp-admin was not added in a custom group, */some/unique/name* is not displayed in the Red Hat Update Infrastructure Management Tool. If you try to create a repository with the same ID */some/unique/name* in the Red Hat Update Infrastructure Management Tool and are not aware that */some/unique/name* repository was created in pulp-admin, you will see a message saying "A repository with ID */some/unique/name* already exists".

Entitlements can also contain variables, as long as yum knows the value for the variable. The two most common variables to use are **`$basearch`** and **`$releasever`**, which are populated with details of the client making the request. For example, if an entitlement certificate grants access to */unique-name/$basearch/bar* and the request is made to a repository located at *//server/pulp/repos/unique-name/x86_64/bar*, the RHUI will approve the request and grant the authentication because the path matches when the variable is populated.

The Red Hat Update Infrastructure Management Tool suggests a path to use based on the variables you used when you gave it a path for the repository. Leave the field blank to accept the suggested path.

The Red Hat Update Infrastructure Management Tool will ask if you want GNU Privacy Guard

(GPG) signature turned on for content in that repository. If you press **y**, you will be asked if the content will be signed by Red Hat. Answering yes will include Red Hat's GPG key in the repository configuration. You are then asked if the content will be signed by a custom GPG key. Answering yes will prompt for a path to a public GPG key to include in the repository configuration. You can continue entering multiple paths to public GPG keys.

```
Should the repository require clients to perform a GPG check and
verify packages are signed by a GPG key? (y/n) y

Will the repository be used to host any Red Hat GPG signed
content? (y/n) y

Will the repository be used to host any custom GPG signed content?
(y/n) y

Enter the absolute path to the public key of the GPG key pair:
/root/rpm-gpg/rhui-client-rhui.gpg

Would you like to enter another public key? (y/n) y

Enter the absolute path to the public key of the GPG key pair:
/root/rpm-gpg/rhui-client-all.gpg

Would you like to enter another public key? (y/n) n
```

8. The details of the new repository displays. Press **y** at the prompt to confirm the information and create the repository.

11.3. CREATE AN ENTITLEMENT CERTIFICATE

See Section 10.1, "Create an Entitlement Certificate" for details.

11.4. CREATE A CLIENT CONFIGURATION RPM

See Section 10.2, "Create a Client Configuration RPM" for details.

11.5. INSTALL THE CLIENT CONFIGURATION RPM ON A CLIENT NODE

1. Install the client configuration RPM on each client node that requires updating.

```
# yum install /path/to/client_custom.rpm
```

2. The client configuration RPM will configure a yum repository called *rhui-$ORIGINALNAME*. Use **`yum update`** to update each node.

```
# yum update
```

NOTE

Running **`yum update`** pulls updates from all enabled yum repositories. To pull updates from the *rhui-rhui-3 yum* repository only, use the following command.

```
# yum --disablerepo=* --enablerepo=rhui-rhui-3 update
```

Report a bug

CHAPTER 12. CREATE CLIENT IMAGES AND TEMPLATES

The exact nature of the Red Hat Enterprise Linux (RHEL) images to be created depends on the technology stack in your environment. In all cases, the goal is to create an artifact (image, template, and so on) that will meet certain criteria when instantiated.

12.1. IMAGE REQUIREMENTS

The following requirements apply to certified cloud images, most of which are default behaviors or configurations.

- Red Hat packages may not be altered, rebuilt, or replaced.
- SELinux should be enabled and in enforcing mode.
- If used, iptables should be blocking access to all ports other than SSH (and any other ports required for proper operation of the cloud infrastructure).
- Local passwords should use a hashing algorithm at least as strong as the default for that RHEL version (SHA-512 for Red Hat Enterprise Linux 7).
- Disk size should be at least 6 GB.
- File system type should be ext4 (Red Hat Enterprise Linux 6) or xfs (Red Hat Enterprise Linux 7).
- sshd should be enabled for remote access.
- Syslog configuration should be unchanged from the operating system default.

See the Cloud Image Certification Policy Guide for more details.

12.2. RED HAT UPDATE INFRASTRUCTURE INTEGRATION

1. Integrate the image with the Red Hat Update Infrastructure (RHUI) by transferring the RHUI entitlement RPM and GPG key to the target RHEL client system.
2. Install the appropriate client configuration RPM.

```
# yum install <rhui-client-rhel7>
```

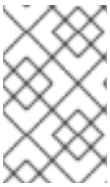
3. Import the Red Hat release GPG key (*/etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release*) into the entitlement RPM, along with any custom repository keys.
4. Import the entitlement RPM GPG key.

```
#rpm --import <rhui-client-rhui>
```

5. Updates will come from RHUI instead of the Red Hat Subscription Manager (rhsm); turn off rhsm by editing *./rhsm.conf* to reflect **enabled=0**.
6. Optionally (but strongly recommended), run the yum update command to apply all available updates.

12.3. TEMPLATE PREPARATION

The image must be sanitized to make it suitable for use as a template. This script can be used to sanitize a virtual machine image in preparation for use as a template. It is compatible with Red Hat Enterprise Linux 6 and Red Hat Enterprise Linux 7 images.



NOTE

The script might require modification in some environments. Review this script carefully before use and make sure that the changes it makes to the image are compatible with your environment.

```
#!/bin/bash

# RHEL 7
if ! [[ `runlevel | cut -d " " -f 2` =~ ^[1S]$ ]]; then
echo "Please *boot* to runlevel 1"
exit 3
fi

# Kill udev
killall -9 udevd

# Clean out /root
rm -rf /root/*
rm -f /root/.bash_history
rm -rf /root/.ssh

# SSH host keys
rm -f /etc/ssh/ssh_host_*
# Remove all files in /var that are not owned by an RPM

for FILE in `find /var -type f`; do
rpm -qf --quiet "$FILE" || rm -f "$FILE"

done

# Remove empty directories in /var that are not owned by an RPM

until [ "$REMOVED_DIR" = false ]; do
     REMOVED_DIR=false
     for DIR in `find /var -type d -empty`; do
    if ! rpm -qf --quiet "$DIR"; then
         REMOVED_DIR=true
         rmdir "$DIR"
    fi
 done

done

# Truncate any remaining files in /var/log
for FILE in `find /var/log -type f`; do
   echo -n > "$FILE"

done
```

```
# Make sure the RPM GPG key has been imported
rpm --import /etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release 2> /dev/null

# Remove MAC addresses from /etc/sysconfig/network-scripts/ifcfg-*
for FILE in /etc/sysconfig/network-scripts/ifcfg-*; do

   sed -i /^HWADDR/d "$FILE"

done

# Remove auto-generated udev rules for CD-ROM and network devices
rm -f /etc/udev/rules.d/70-persistent-{cd,net}.rules

# Clean out /tmp
find /tmp -mindepth 1 -delete
```

1. Copy the script to */mktemplate.sh* and reboot the system to **runlevel 1**.

NOTE

Do not change to runlevel 1 instead of rebooting (with init 1, for example). Changing to runlevel 1 leaves certain daemons running that are not running when the system is booted to single-user mode (notably rsyslog).

2. When the system has rebooted into single-user mode, execute the following commands.

```
# unset HISTFILE
# chmod 0755  /mktemplate.sh
# /mktemplate.sh
# rm -f /mktemplate.sh
# poweroff
```

Report a bug

CHAPTER 13. CERTIFIED CLOUD AND SERVICE PROVIDER CERTIFICATION WORKFLOW

The Certified Cloud Provider Agreement requires that Red Hat certifies the images (templates) from which tenant instances are created to ensure a fully supported configuration for end customers. There are two methods for certifying the images for Red Hat Enterprise Linux and Red Hat Enterprise Linux 7. The preferred method is to use the Certified Cloud & Service Provider (CCSP) image certification workflow.

See the Red Hat Certified Cloud and Service Provider Certification Workflow Guide for more information.

After certifications have been reviewed by Red Hat, a pass/fail will be assigned and certification will be posted to on the public Red Hat certification website at Certified Service and Product Catalogs.

Report a bug

CHAPTER 14. MANAGE CONTENT

Red Hat Update Infrastructure (RHUI) can be configured to create and use a repository that will update the RHUI installation. The Red Hat Update Infrastructure Management Tool can create the repository and generate an entitlement certificate and client configuration RPM. The RPM is installed on the Red Hat Update Appliance (RHUA) and each content delivery server (CDS) node. Future updates can be downloaded and installed using **`yum update`**.

14.1. AVAILABLE CHANNELS

Red Hat's Certified Cloud & Service Provider (CCSP) Partners control what channels and packages are delivered through their service. See this Knowledgebase article for the most current information regarding what channels are available. The repositories are available for use with:

- Red Hat Enterprise Linux 7
- Red Hat Enterprise Linux 7 for SAP Applications
- Red Hat Enterprise Linux 7 for SAP HANA
- Red Hat Enterprise Linux 6
- Red Hat Enterprise Linux 6 for SAP Applications
- Red Hat Enterprise Linux 6 for SAP HANA
- Red Hat Enterprise Linux 5
- Red Hat Enterprise Linux 5 Extended Life Cycle Support

Contact your CCSP if a required channel is missing. You can learn more about what is available by browsing the Certification Catalog.

14.2. MANAGE THE LINUX SOFTWARE REPOSITORIES

A repository is a server node that contains downloadable software for a Linux distribution. You use yum to search for, install, and control RPMs from the repository to your RHUA and CDS nodes.

14.2.1. List the Available Repositories

1. Navigate to the Red Hat Update Infrastructure Management Tool.

```
[root@rhua ~]# rhui-manager
```

2. In the Red Hat Update Infrastructure Management Tool home screen, press **`r`** to select **manage repositories**.

3. Press **`l`** to select **list repositories currently managed by the RHUI**.

```
                                                    Connected:
rhua.example.com
```

```
------------------------------------------------------------------
--------------
rhui (repo) => l
```

14.2.2. Display the Repository Information

You can use the **Repository Management** screen to display information about a particular repository.

1. From the **Repository Management** screen, press **i**. The output contains all repositories that are managed by Red Hat Update Infrastructure.
2. Select which repository to view by typing the repository's number at the prompt. Typing the number of a repository places a checkmark next to the name of that repository. You can also choose the range of repositories, for instance, by entering **1** - **5**.
3. Continue until all repositories you want to view are checked.
4. Press **c** at the prompt to confirm.

```
Name:              RHEL RHUI Server 7 Containers (7Server-x86_64)
Type:              Red Hat
Relative Path:
content/dist/rhel/rhui/server/7/7Server/x86_64/containers/
GPG Check:         Yes
Custom GPG Keys:   (None)
Red Hat GPG Key:   Yes
Package Count:     0
Last Sync:          Never
Next Sync:         11-30-2015 19:38
```

14.2.3. Add a Red Hat Repository

Load the specific repositories for entitled products before you add a new Red Hat repository.

See Section 9.1, "Create a Repository" for details.

14.2.4. Delete a Red Hat Repository

When the Red Hat Update Infrastructure Management Tool deletes a Red Hat repository, it deletes the repository from the RHUA and all applicable CDS nodes.

NOTE

The repository content remains on the disk and takes up disk space. This content is known as an orphan content unit, or an orphan for short. See Section 14.3, "Orphaned Content Units" for more details.

1. From the **Repository Management** screen, press **d** at the prompt to delete a Red Hat repository. A list of all repositories currently being managed by RHUI displays.
2. Select which repositories to delete by typing the number of the repository at the prompt. Typing the number of a repository places a checkmark next to the name of that repository. You can also choose the range of repositories, for instance, by entering **1** - **5**.

3. Continue until all repositories you want to delete are checked.

4. Press **c** at the prompt to confirm.

NOTE

After you delee the repositories, the client configuration RPMs that refer to the deleted repositories will not be available to be used by yum.

14.2.5. List the RPM Packages in a Repository

When listing repositories within the Red Hat Update Infrastructure Management Tool, only repositories that contain fewer than 100 packages display their contents. Results with more than 100 packages only display a package count.

1. To see a complete list, regardless of how many packages are contained within a repository, press **r** at the **Home** screen to access the **Repository Management** screen.

2. Press **p** to select **list packages in a repository (RPM content only)**.

3. Select the number of the repository you want to view. The Red Hat Update Infrastructure Management Tool asks if you want to filter the results. Leave the line blank to see the results without a filter.

4. Alternatively, type the first few letters of the RPM name you are looking for to filter the results.

14.2.6. Create a Custom Repository

Use a protected repository or 64-bit RHUI servers when you are distributing new non-Red Hat packages to RHUI servers. For example, use *client-rhui-x86_64* if you are distributing an updated client configuration package.

Like Red Hat content repositories, all of which are protected, protected custom repositories that differ only in processor architecture (i386 versus AMD64) are consolidated into a single entitlement within an entitlement certificate, using the **$basearch** yum variable.

If certificate validation prevents access, you can use an unprotected server repository to distribute RPMs to the RHUI servers.

1. From the **Repository Management** screen, press **c** to access the **create a new custom repository (RPM content only)** screen.

2. Enter a unique ID for the repository. Only alphanumeric characters, _ (underscore), and - (hyphen) are permitted. You cannot use spaces in the unique ID. For example, *repo1*, *repo_1*, and *repo-1* are all valid entries.

3. Enter a display name for the repository. This name is used to identify the repository within the Red Hat Update Infrastructure Management Tool.

4. Specify the path that will host the repository. The path must be unique across all repositories hosted by Red Hat Update Infrastructure. For example, if you specify the path at this step as *some/unique/name*, then the repository will be located at *//<server>/pulp/repos/some/unique/name*.

5. Select **sha256** as the checksum type to be used for the repository metadata.

6. Choose whether to protect the new repository. If you answer no to this question, any client can access the repository. If you answer yes, only clients with an appropriate entitlement certificate can access the repository.

NOTE

As the name implies, the content in an unprotected repository is available to any system that requests it, without any need for a client entitlement certificate. Be careful when using an unprotected repository to distribute any content, particularly content such as updated client configuration RPMs, which will then provide access to protected repositories.

Use of unprotected repositories is a “break glass in case of emergency” course of action.

7. If you choose to protect the new repository, the Red Hat Update Infrastructure Management Tool will ask for the entitlement path. It will also suggest the entitlement path based on the repository’s relative path.
Client entitlement certificates contain the download URLs that they are allowed to access. The RHUI analyzes the contents of the certificate to determine if the repository requested matches any of the permitted URLs, which determines whether to allow the client to authenticate. For example, if an entitlement certificate grants access to */some/unique/name* and the request is made to a repository located at *//server/pulp/repos/some/unique/name/os/repodata*, RHUI will approve the request and grant the authentication because the path begins with one of the entitled download URLs. The URL only needs to begin with the correct information; it does not need to match exactly.

Entitlements can also contain variables, as long as yum knows the value for the variable. The two most common variables to use are **`$basearch`** and **`$releasever`**, which are populated with details of the client making the request. For example, if an entitlement certificate grants access to */unique-name/$basearch/bar* and the request is made to a repository located at *//server/pulp/repos/unique-name/x86_64/bar*, RHUI will approve the request and grant the authentication because the path matches when the variable is populated.

The Red Hat Update Infrastructure Management Tool suggests a path to use based on the variables you used when you gave it a path for the repository. Leave the field blank to accept the suggested path.

The Red Hat Update Infrastructure Management Tool will ask if you want GNU Privacy Guard (GPG) signature turned on for content in that repository. If you press **`y`**, you will be asked if the content will be signed by Red Hat. Answering yes will include Red Hat’s GPG key in the repository configuration. You are then asked if the content will be signed by a custom GPG key. Answering yes will prompt for a path to a public GPG key to include in the repository configuration. You can continue entering multiple paths to public GPG keys.

```
Should the repository require clients to perform a GPG check and
verify packages are signed by a GPG key? (y/n) y

Will the repository be used to host any Red Hat GPG signed content?
(y/n) y

Will the repository be used to host any custom GPG signed content?
(y/n) y

Enter the absolute path to the public key of the GPG key pair:
```

```
/tmp/rhuitest1.gpg

Would you like to enter another public key? (y/n) y

Enter the absolute path to the public key of the GPG key pair:
/root/rpm-gpg/rhui-client-rhui.gpg

Would you like to enter another public key? (y/n) n
```

8. The details of the new repository display. Press **y** at the prompt to confirm the information and create the repository.

14.2.7. Upload Packages to a Custom Repository

You can upload multiple packages at a time and to upload to more than one repository at a time. Packages are uploaded to the RHUA immediately but are not available on the CDS node until the next time the CDS node synchronizes.

1. From the **Repository Management** screen, press **u** at the prompt to upload new packages to a particular repository. A list of all available custom repositories displays.

NOTE

You cannot upload packages to Red Hat repositories.

2. Select which custom repository to add the packages to by typing the number of the repository at the prompt. Typing the number of a repository places a checkmark next to the name of that repository. Continue until all repositories you want to add to have been checked.

3. Press **c** at the prompt to confirm.

4. Specify the location of the RPMs to upload. This can be a single .rpm file, or it can be a directory containing several .rpm files. If you specify a directory, all .rpm files in that directory are uploaded. The details of the new packages to upload display.

5. Press **y** at the prompt to confirm the information and upload the packages.

```
The following RPMs will be uploaded:

origin-1.0-1.noarch.rpm
parent-1.0-1.noarch.rpm
patb-0.1-2.x86_64.rpm
rh-amazon-rhui-client-rhs30-2.2.124-1.el7.noarch.rpm

Proceed? (y/n) y
```

14.2.8. Delete Packages from a Custom Repository

1. From the **Repository Management** screen, press **p** to list packages in a repository (RPM content only). Enter the number of the custom repository that you want to delete packages from and press **Enter**.

```
rhui (repo) => p
```

```
Choose a repository:
  1  - HP Packages for Testing
Enter value (1-1) or 'b' to abort: 1

Enter the first few characters (case insensitive) of an RPM to
filter the results
(blank line for no filter):


Only filtered results that contain less than 100 packages will have
their
contents displayed. Results with more than 100 packages will display
a package count only.

Packages:
  hprest-1.5-79.x86_64.rpm
  hpsum-7.6.0-86.rhel7.x86_64.rpm
  ilorest-2.2.2-6.x86_64.rpm         <========== Goal, delete this
package
  sum-8.2.0-53.rhel7.x86_64.rpm
```

2. Use the **pulp-admin** command to list the repository information, including the repo_id.

```
# pulp-admin --username admin --password "redhat" repo list --snip--
Id:                              custom_repo1
Display Name:           HP Packages for Testing
Description:               HP Packages for Testing
Content Unit Counts:
  Rpm: 4
```

3. List the package information.

```
# pulp-admin --username admin --password "redhat" rpm repo content
rpm --repo-id "custom_repo1" --str-eq="filename=ilorest-2.2.2-
6.x86_64.rpm"

Arch:         x86_64
Buildhost:    bls11u3x64001.sde.rdlabs.hpecorp.net
Checksum:
570b98fff1943819e554ff5d643f674a1aa00fc1b362900badfdc4bd0943ce06
Checksumtype: sha256
Description:  Command line interface for managing HPE ProLiant
Servers  Authors:
              --------     Hewlett Packard Enterprise
Epoch:        0
Filename:     ilorest-2.2.2-6.x86_64.rpm
License:      Copyright 2016 Hewlett Packard Enterprise Development
LP
Name:         ilorest
Provides:     config(ilorest) = 2.2.2-6-0, ilorest = 2.2.2-6-0,
ilorest(x86-64)
              = 2.2.2-6-0, ilorest_chif.so()(64bit)
Release:      6
Requires:     /bin/sh, /bin/sh, libc.so.6()(64bit),
              libc.so.6(GLIBC_2.2.5)(64bit), libc.so.6(GLIBC_2.3)
```

```
(64bit),
              libdl.so.2()(64bit), libdl.so.2(GLIBC_2.2.5)(64bit),
              libz.so.1()(64bit), rtld(GNU_HASH)
Vendor:       Hewlett Packard Enterprise Company
Version:      2.2.2
```

4. Delete the package from the custom repository.

```
# pulp-admin --username admin --password "redhat" rpm repo remove
rpm --repo-id "custom_repo1" --str-eq="filename=ilorest-2.2.2-
6.x86_64.rpm"
This command may be exited via ctrl+c without affecting the request.

[\]
Running...

Units Removed:
  ilorest-2.2.2-6-x86_64
```

5. Update the metadata and publish the repository.

```
# pulp-admin  --username admin --password "redhat" rpm repo update -
-repo-id "custom_repo1"
# pulp-admin --username admin --password "redhat"  rpm repo publish
run --repo-id "custom_repo1"
```

6. Remove the orphaned RPM disassociated from the repository and reclaim disk space as described in Section 14.3, “Orphaned Content Units”.

14.3. ORPHANED CONTENT UNITS

RHUI does not delete orphaned content units (also known as orphans) when the Red Hat Update Infrastructure Management Tool deletes a repository. See Section 14.2.4, “Delete a Red Hat Repository” for more details. Orphans are package files no longer referenced by any repository but remain on the file system and consume disk space. Package files can become orphans as a result of the configuration settings or repository deletion. If you are unsure about the deletion of these content units, consider enlarging disk space instead of removing the orphans.

You can delete orphans on the RHUA and CDSs to reclaim disk space. The following procedure deletes orphans from RHUI. Perform a complete backup before using these steps.

1. Run the following command from the RHUA to display orphaned packages.

```
[root@rhua ~]# pulp-admin -u admin -p admin orphan list
```

2. Run the following command to see available arguments.

```
[root@rhua ~]# pulp-admin -u admin -p admin orphan list --help
Command: list
Description: display a list of orphaned units

Available Arguments:

  --type    - restrict to one content type such as "rpm", "errata",
```

```
              "puppet_module", etc.
  --details - include a detailed list of the individual orphaned
units, ignored
              when content type is not specified
```

3. There are three flags for removing orphans.

```
        --type=<type> to remove all the orphaned content units of a
particular type
        --id=<id> to remove a particular orphaned content unit
        --all to remove all the orphaned content units on the server
```

Here is one example of how to delete an orphan.

```
[root@rhua ~]# pulp-admin orphan remove --all
```

4. Run the following command to see a list of arguments.

```
[root@rhua ~]# pulp-admin -u admin -p admin orphan remove --help
Command: remove
Description: remove one or more orphaned units

Available Arguments:

  --bg      - if specified, the client process will end immediately
(the task
              will continue to run on the server)
  --type    - restrict to one content type such as "rpm", "errata",
              "puppet_module", etc.
  --unit-id - ID of a content unit; if specified, you must also
specify a type
  --all     - remove all orphaned units, ignoring other options
```

14.4. MANAGE THE CONTENT DELIVERY SERVER NODES

CDS nodes are the main component of a content delivery network (CDN), offering high availability to the client. Running servers in a geographically dispersed manner can also improve response time.

The **Content Delivery Server (CDS) Management** screen is used to list, add, reinstall, and delete CDS nodes.

1. In the Red Hat Update Infrastructure Management Tool home screen, press **c** to access the **Content Delivery Server (CDS) Management** screen.

```
             -= Red Hat Update Infrastructure Management Tool =-

-= Home =-

   r   manage repositories
   c   manage content delivery servers (CDS)
   l   manage HAProxy load-balancer instances
   s   synchronization status and scheduling
   e   create entitlement certificates and client configuration RPMs
   n   manage Red Hat entitlement certificates
```

```
   sm  manage Red Hat subscriptions
   u   manage RHUI users

                                                        Connected:
rhua.example.com
```

2. From the **Content Delivery Server (CDS) Management** screen, press **l** at the prompt to list the CDS nodes that RHUI manages.

```
-------------------------------------------------------------
--------------
        -= Red Hat Update Infrastructure Management Tool =-

-= Content Delivery Server (CDS) Management =-

l   list all known CDS instances managed by the RHUI
a   register (add) a new CDS instance
r   reinstall and reapply configuration to an existing CDS instance
d   unregister (delete) a CDS instance from the RHUI

                                  Connected: ip-10-99-206-
124.ec2.internal
-------------------------------------------------------------
--------------
rhui (cds) =>l

-= RHUI Content Delivery Servers =-

Hostname:          cds1.example.com
SSH Username:      root
SSH Private Key:   /root/.ssh/cds.rsa

Hostname:          cds2.example.com
SSH Username:      root
SSH Private Key:   /root/.ssh/cds.rsa

Hostname:          cds3.example.com
SSH Username:      root
SSH Private Key:   /root/.ssh/cds.rsa
-------------------------------------------------------------
--------------
```

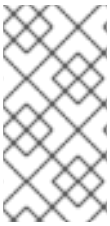
14.5. WORKING WITH CONTAINERS

Red Hat Update Infrastructure 3.0 in a Red Hat Enterprise Linux 7 system or Red Hat Atomic Host system uses Docker to automate the deployment of applications inside Linux containers. Using Docker offers the following advantages:

- Requires less storage and in-memory space than VMs: Because the containers hold only what is needed to run an application, saving and sharing is more efficient with Docker containers than it is with VMs that include entire operating systems.
- Improved performance: Because you are not running an entirely separate operating system, a container typically runs faster than an application that carries the overhead of a whole new VM.

- Secure: Because a Docker container typically has its own network interfaces, file system, and memory, the application running in that container can be isolated and secured from other activities on a host computer.
- Flexible: With an application's runtime requirements included with the application in the container, a Docker container can run in multiple environments.

Linux containers with docker format are supported running on hosts with SELinux enabled. SELinux is not supported when the **`/var/lib/docker`** directory is located on a volume using the B-tree file system (Btrfs).



NOTE

The docker API takes over the root folder (`/`) on the httpd instance and must run on a different port. Port 5000 is currently used, but this will be user-configurable in the future. The RHUA must know the port because the docker client uses the host name and port when finding the Certificate Authority to use for docker content.

See Get Started with Docker Formatted Container Images and Red Hat Enterprise Linux Atomic Host 7: Getting Started with Containers for more information about containers.

14.6. MANAGE THE CONTENT DELIVERY SERVER DOCKER CONTENT

14.6.1. Docker Content in Red Hat Update Infrastructure

Docker content includes containers, images, and platform images. Currently, docker content does not have entitlement enforcement available. To put such a feature in place, the docker client must first support X.509 certificates. The implication for RHUI is that downloaded or published docker content is available publicly on the CDS's registry.

A container is an application sandbox. Each container is based on an image that holds necessary configuration data. When you launch a container from an image, a writable layer is added on top of this image. Every time you commit a container (using the **`docker commit`** command), a new image layer is added to store your changes.

An image is a read-only layer that is never modified; all changes are made in the top-most writable layer, and it can be saved only by creating a new image. Each image depends on one or more parent images.

A platform image is an image that has no parent. Platform images define the runtime environment, packages, and utilities necessary for a containerized application to run. The platform image is read-only, so any changes are reflected in the copied images stacked on top of it.

14.6.2. Add a Container to Red Hat Update Infrastructure

The following steps describe how to add a Docker container to the client machine where docker via RHUI is going to be used. Access to docker requires access to the *Red Hat Enterprise Linux Extras* repository.

1. Register the client and get subscriptions using the instructions in Chapter 4, *Register Red Hat Update Infrastructure and Attach Subscriptions*.
2. Alternatively, you can register the system using Subscription Management tools and install the docker package. Also enable the software repositories needed. (Replace pool_id with the pool ID of your RHEL 7 subscription.) For example:

```
# subscription-manager register --username=rhnuser --
password=rhnpasswd
# subscription-manager list --available    Find valid RHEL pool ID
# subscription-manager attach --pool=pool_id
# subscription-manager repos --enable=rhel-7-server-extras-rpms
# subscription-manager repos --enable=rhel-7-server-optional-rpms
```

The current RHEL 7 release and RHEL 7 Atomic Host release each include two different versions of Docker.

- docker: This package includes the version of Docker that is the default for the current release of RHEL. Install this package if you want a more stable version of Docker that is compatible with the current versions of Kubernetes and OpenShift available with Red Hat Enterprise Linux.

- docker-latest: This package includes a later version of Docker that you can use if you want to work with newer features of Docker. This version is not compatible with the versions of Kubernetes and OpenShift that are available with the current release of Red Hat Enterprise Linux.
  See the Atomic Host and Containers section of the Red Hat Enterprise Linux Release Notes for more details on the contents of docker and docker-latest packages and how to enable the docker-latest package.

3. Install and use the default docker package (along with a couple of dependent packages if they are not yet installed).

```
# yum install docker device-mapper-libs device-mapper-event-libs
```

See Section 1.3. Getting Docker in RHEL 7 of the Getting Started with Containers document for more information about Docker and Red Hat Enterprise Linux and Atomic Host.

4. Add a docker container to RHUI.

```
[root@rhua ~]# rhui-manager
```

5. From the **Red Hat Update Infrastructure Management Tool**, press **r** to access the **Repository Management** screen.

```
-= Red Hat Update Infrastructure Management Tool =-

-= Repository Management =-

l   list repositories currently managed by the RHUI
i   display detailed information on a repository
a   add a new Red Hat content repository
ad  add a new Red Hat docker container
c   create a new custom repository (RPM content only)
d   delete a repository from the RHUI
u   upload content to a custom repository (RPM content only)
p   list packages in a repository (RPM content only)

                                        Connected: rhua.example.com
```

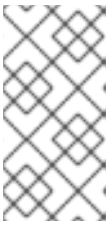
6. Press **ad** to add a new Red Hat docker container.

```
rhui (repo) => ad

Name of the container in the registry:
```

7. Enter the name of the container in the registry.

```
jboss-eap-6/eap64-openshift
```

8. Enter a unique ID for the container.



NOTE

The rhui-manager can convert the name of the container from the registry to the format that is usable in Pulp. It does so by replacing slashes and dots with underscores. You can accept such a converted name by pressing **Enter** or by entering a name of your choice.

```
jboss-eap-6_eap64-openshift
```

9. Enter a display name for the container.

```
jboss-eap-6_eap64-openshift
The following container will be added:
  Container Id:              jboss-eap-6_eap64-openshift
  Display Name:            jboss-eap-6_eap64-openshift
  Upstream Container Name:   jboss-eap-6/eap64-openshift
Proceed? (y/n)
```

10. Press **y** to proceed or **n** to cancel.

```
y
Successfully added container JBoss_EAP_Container
```

11. Press **^** to return to the Red Hat Update Infrastructure Management Tool home screen.

14.6.3. Synchronize the docker Repository

The following steps describe how to synchronize a docker repository.

1. Press **s** to access the **Synchronization Status** screen.
2. Press **sr** to synchronize an individual repository immediately.
3. Enter the number of the repository that you wish to synchronize.
4. Press **c** to confirm the selection. You can enter **?** for more commands.
5. Press **y** to proceed or **n** to cancel.

```
The following repositories will be scheduled for synchronization:
  jboss-eap-6_eap64-openshift
Proceed? (y/n) y
```

```
Scheduling sync for jboss-eap-6_eap64-openshift...
... successfully scheduled for the next available timeslot.
```

6. Press ^ to return to the Red Hat Update Infrastructure Management Tool home screen.

14.6.4. Generate the docker Client Configuration

The client configuration RPM is intended for RHUI clients that should pull docker containers from RHUI. The RPM contains the load balancer's certificate. When you install the RPM, it:

- adds the load balancer as a docker registry.
- modifies the docker configuration.

1. Press **e** to access the **Client Entitlement Management** screen.
2. Press **d** to create a docker client configuration RPM.
3. Enter the full path to the local directory where the client configuration files generated will be stored. This directory will be created if it does not exist.

```
/root/
```

4. Enter the name of the RPM.

```
dockertest
```

5. Enter the version number of the configuration RPM. The default is 2.0.
6. Enter the port that will serve docker content. The default is 5000.

```
Successfully created client configuration RPM.
Location: /root/dockertest-2.0/build/RPMS/noarch/dockertest-2.0-
1.noarch.rpm
```

14.6.5. Install a RPM on the Client

1. Navigate to the directory where the RPM is saved.

```
[root@rhua noarch]# cd /root/dockertest-2.0/build/RPMS/noarch/
```

2. Copy the RPM to the client.

```
# scp dockertest-2.0-1.noarch.rpm <hostname_of_cli:path_on_cli>
```

3. Switch to the client and install the RPM.

```
[root@cli01 ~]# yum install dockertest-2.0-1.noarch.rpm

Loaded plugins: amazon-id, rhui-lb, search-disabled-repos
Examining dockertest-2.0-1.noarch.rpm: dockertest-2.0-1.noarch
Marking dockertest-2.0-1.noarch.rpm to be installed
Resolving Dependencies
```

```
--> Running transaction check
---> Package dockertest.noarch 0:2.0-1 will be installed
--> Processing Dependency: docker-common for package: dockertest-
2.0-1.noarch
rhel-7-server-rhui-extras-rpms
| 3.4 kB

--> Running transaction check
---> Package docker-common.x86_64 2:1.10.3-59.el7 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

================================================================
================================================
 Package                  Arch              Version
Repository                                   Size
================================================================
================================================
Installing:
 dockertest               noarch             2.0-1
/dockertest-2.0-1.noarch                    1.7 k
Installing for dependencies:
 docker-common            x86_64             2:1.10.3-59.el7
rhel-7-server-rhui-extras-rpms              63 k

Transaction Summary
================================================================
================================================
Install  1 Package (+1 Dependent package)

Total size: 64 k
Total download size: 63 k
Installed size: 4.7 k
Is this ok [y/d/N]: y
Downloading packages:
docker-common-1.10.3-59.el7.x86_64.rpm
|  63 kB  00:00:01
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction

Installed:
  dockertest.noarch 0:2.0-1

Dependency Installed:
  docker-common.x86_64 2:1.10.3-59.el7

Complete!
```

14.6.6. Test the docker pull Command on the Client

The **`docker pull`** command consumes content from a container. The following steps describe how to test a **`docker pull`** command on the client.

1. Start the docker service.

```
[root@cli01 ~]# systemctl start docker
```

2. Run the **`docker pull`** command.

```
[root@cli01 ~]# docker pull jboss-eap-6_eap64-openshift

Using default tag: latest
Trying to pull repository cds.example.com:5000/jboss-eap-6_eap64-
openshift ...
latest: Pulling from cds.example.com:5000/jboss-eap-6_eap64-
openshift
30cf2e26a24f: Pull complete
99dd41655d8a: Pull complete
05d9aa366d71: Pull complete
39feddb214c9: Pull complete
76786100be04: Pull complete
d48e1afdcad8: Pull complete
Digest:
sha256:5331cae5edaeede56c7e14bede8608229a89f73067d7373af246cabe4b8d4
a24
Status: Downloaded newer image for cds.example.com:5000/jboss-eap-
6_eap64-openshift:latest
```

3. If the **`docker pull`** command fails, check the rhui-manager container synchronization status. The synchronization probably has not been performed yet and you have to wait until it synchronizes.

```
Using default tag: latest
Trying to pull repository cds.example.com:5000/jboss-eap-6_eap64-
openshift ...
unknown: Not Found
Trying to pull repository docker.io/library/jboss-eap-6_eap64-
openshift ...
Pulling repository docker.io/library/jboss-eap-6_eap64-openshift
Error: image library/jboss-eap-6_eap64-openshift not found
Error: image library/jboss-eap-6_eap64-openshift not found
```

14.7. ATOMIC HOST AND OSTREE CONTENT

Red Hat Enterprise Linux Atomic Host is a variation of Red Hat Enterprise Linux 7 optimized to run Linux containers. It has been built to be lightweight and efficient, making it a particularly optimal operating system to use as a container runtime system for cloud environments. RHEL Atomic Host comes with many tools for running containers preinstalled (docker, atomic, etcd, flannel). All-in-one kubernetes installs are still supported, but Red Hat no longer supports Kubernetes clusters.

RHEL Atomic Host uses an open source tool called rpm-OSTree to manage bootable, immutable, versioned file system trees made of RPM content. Red Hat composes these trees from packages, and the rpm-ostree tool replicates the trees atomically. This results in a strategy for upgrade and maintenance that centers around atomic updates. The use of rpm-ostree instead of Yum to upgrade and maintain software means that RHEL Atomic Host is managed differently than other RHEL 7 variants.

Specifically, when using RHEL Atomic Host, the operating system content is mounted in read-only

mode. There are only two writable directories for local system configuration: **/etc/** and **/var/**. Updates work in the following way: a new bootable file system tree is generated, which shares storage with the current file system tree. When you download the new system tree, the old one is retained in parallel with it. This means that the first, pre-upgrade version of the file system tree can be atomically restored when needed.

User files that are intended to persist across upgrades, including containers and data, should be placed in the /var/ directory. The operating system itself is stored in the **/usr/** directory and is read-only. If you perform a long file listing in the root directory using the command **ls -l /**, you will discover that many of the traditional root-level directories are symbolic links to one of these two locations. For example, the **/home/** directory is a symbolic link to the **/var/home/** directory. This directory will therefore persist across upgrades.

The default partitioning dedicates most of the available space for the containers, using direct LVM as the storage backend instead of the default loopback as it is on Red Hat Enterprise Linux. Storage is managed the docker-storage-setup daemon, which creates two Logical Volumes during installation, root for the file system content, and docker-pool for the images and containers.

RHEL Atomic Host uses SELinux to provide strong safeguards in multi-tenant environments. The iptables services are available as firewall; iptables is turned off by default.

See Red Hat Enterprise Linux Atomic Host 7 Installation and Configuration Guide for more information about Red Hat Atomic Host.

14.7.1. Add an Atomic Host Repository

1. Follow Steps 1 through 5 in Section 9.1, “Create a Repository” to add a new Red Hat content repository.

2. Select the **By Product** method by pressing **2**.

```
Import Repositories:
  1  - All in Certificate
  2  - By Product
  3  - By Repository
Enter value (1-3) or 'b' to abort:
```

3. Select the atomic repository from the list by entering the number beside the repository.

```
Red Hat Enterprise Linux Atomic Host (Trees) from RHUI
```

4. Press **c**. The **Red Hat Update Infrastructure Management Tool** displays the repository to be deployed and prompts for confirmation. Press **y** to proceed. A message prints as the repository is deployed.

5. Check that the repository has been installed by pressing **l** to access the **list repositories currently managed by the RHUI** screen.

14.7.2. Synchronize the OSTree Repository

The following steps describe how to synchronize an OSTree repository.

1. Press **s** to access the **Synchronization Status** screen.

2. Press **sr** to synchronize an individual repository immediately.

3. Enter the number of the repository that you wish to synchronize.

4. Press **c** to confirm the selection. You can enter **?** for more commands.

5. Press **y** to proceed or **n** to cancel.

```
The following repositories will be scheduled for synchronization:
  Red Hat Enterprise Linux Atomic Host (Trees) from RHUI (Version
7.3.4)
Proceed? (y/n) y
Scheduling sync for Red Hat Enterprise Linux Atomic Host (Trees)
from RHUI (Version 7.3.4)...
... successfully scheduled for the next available timeslot.
--------------------------------------------------------------------
--------------------------------------------------------------
rhui (sync) =>
```

6. Press **^** to return to the Red Hat Update Infrastructure Management Tool home screen.

14.7.3. Generate a Client Configuration Package on the RHUA

The following steps describe how to configure the Atomic Host client.

1. Generate an entitlement certificate for the OSTree repository by following the steps in Section 10.1, “Create an Entitlement Certificate”. Include the recently added OSTree repository in the certificate.

2. On the Red Hat Update Infrastructure Management Tool home screen, press **e** to select **create entitlement certificates and client configuration RPMs**.

3. On the **Client Entitlement Management** screen, press **o** to select **create an atomic client configuration package**.

```
-= Red Hat Update Infrastructure Management Tool =-

-= Client Entitlement Management =-

e   generate an entitlement certificate
c   create a client configuration RPM from an entitlement
certificate
d   create a docker client configuration RPM
o   create an atomic client configuration package

                                        Connected: rhua.example.com
```

4. Enter the full path of a local directory to save the configuration files to.

```
Full path to local directory in which the client configuration files
generated by this tool
should be stored (if this directory does not exist, it will be
created):

/tmp
```

5. Enter the name of the tar file.

```
Name of the tar file (excluding extension):
testcerttar
```

6. Enter the full path to the entitlement certificate authorizing the client to access specific channels.

```
Full path to the entitlement certificate authorizing the client to
access
specific channels:
/tmp/testcert.crt
```

7. Enter the full path to the private key for the entitlement certificate.

```
Full path to the private key for the above entitlement certificate:
/tmp/testcert.key
```

8. Enter the port to serve Docker content on. Port 5000 is the default.

```
Port to serve Docker content on (default 5000):

Successfully created client configuration package.
Location: /tmp/testcerttar.tar.gz
```

14.7.4. Configure Atomic Host

1. Copy the generated **`.tar.gz`** file to the Atomic Host.
2. Extract the tar file.
3. Run the install.sh script

```
[root@atomiccli01 ~]# ./install.sh
```

14.7.5. Test the ostree pull Command with Atomic Host

The **`ostree pull`** command consumes content from a container. The following steps describe how to test an **`ostree pull`** command on the client.

1. Run the **`ostree pull`** command.

```
[root@atomiccli01 ~] ostree pull rhui-rhel-atomic-host-rhui-
ostree:rhel-atomic-host/7/x86_64/standard

GPG: Verification enabled, found 1 signature:

  Signature made Mon 10 Apr 2017 04:46:45 PM UTC using RSA key ID
199E2F91FD431D51
  Good signature from "Red Hat, Inc. <security@redhat.com>"

809 metadata, 4395 content objects fetched; 308693 KiB transferred
in 108 second
```

2. If **`ostree pull`** returns an error, check the OSTree repository synchronization status. The synchronization probably has not been performed yet and you have to wait until it synchronizes.

Report a bug

CHAPTER 15. MANAGE CERTIFICATES AND KEYS

15.1. RED HAT UPDATE APPLIANCE CERTIFICATES

The Red Hat Update Appliance (RHUA) in Red Hat Update Infrastructure (RHUI) uses the following certificates and keys:

- Content certificate and private key
- Entitlement certificate and private key
- SSL certificate and private key
- Cloud provider's Certificate Authority (CA) certificate

The RHUA is configured with the content certificate and the entitlement certificate. The RHUA uses the content certificate to connect to the Red Hat Content Delivery Network (CDN). It also uses the Red Hat CA certificate to verify the connection to the Red Hat CDN. As the RHUA is the only component that connects to the Red Hat CDN, it will be the only RHUI component that has this certificate deployed. It should be noted that multiple RHUI installations can use the same content certificate. For instance, the Amazon EC2 cloud runs four RHUI installations (one per region), but each RHUI installation uses the same content certificate.

Clients use the entitlement certificate only to permit access to packages in RHUI. To perform an environment health check, RHUA attempts a yum request against each CDS. To succeed, the yum request must specify a valid entitlement certificate.

15.2. CONTENT DELIVERY SERVER CERTIFICATES

Each CDS node in RHUI uses the following certificates and keys:

- SSL certificate and private key
- Cloud provider's CA certificate

The only certificate necessary for the CDS is an SSL certificate, which permits HTTPS communications between the client and the CDS. The SSL certificates are scoped to a specific host name, so a unique SSL certificate is required for each CDS node. If SSL errors occur when connecting to a CDS, the certificate should be double-checked to make sure its common name is set to the fully qualified domain name of the CDS on which it is installed.

The CA certificate is used to verify that the entitlement certificate sent by the client as part of a yum request was signed by the cloud provider. This prevents a rogue instance from generating its own entitlement certificate for unauthorized use within RHUI.

15.3. CLIENT CERTIFICATES

Each client in the RHUI uses the following certificates and keys:

- Entitlement certificate and private key
- Cloud provider's CA certificate

The entitlement certificate and its private key enable information encyrption from the CDS back to the client. Each client uses the entitlement certificate when connecting to the CDS to prove it has permission to download its packages. All clients use a single entitlement certificate.

The cloud provider's CA certificate is used to verify the CDS's SSL certificate when connecting to it. This ensures that a rogue instance is not impersonating the CDS and introducing potentially malicious packages into the client.

The CA certificate is used to verify the CDS's SSL certificate, not the entitlement certificate itself. The reverse is true for the CDS node. The CDS's SSL certificate and private key are used for encrypting data from the client to the CDS. The CA certificate present on the CDS is used to verify that the CDS node should trust the entitlement certificate sent by the client.

15.4. DISPLAY AND MANAGE CERTIFICATES

When Red Hat issues the original entitlement certificate, it grants access to the repositories you requested. When you create client entitlement certificates, you need to decide how to subdivide your clients and create a separate certificate for each one. You can then use each certificate to create individual RPMs for installation on the appropriate guest images.

15.4.1. List the Entitled Products for a Certificate

The **Entitlements Manager** screen is used to list entitled products in the current Red Hat content certificates and to upload new certificates.

1. Navigate to the Red Hat Update Infrastructure Management Tool home screen.

```
[root@rhua ~]# rhui-manager
```

2. Press **n** at the prompt to access the **Entitlements Manager** screen.

3. From the **Entitlements Manager** screen, press **l** at the prompt to list data about the current content certificate. The Red Hat Update Infrastructure Management Tool displays the following information about the certificate.

```
rhui (entitlements) => l

Red Hat Entitlements

Valid
RHEL RHUI Atomic 7 Ostree Repo
Expiration: 08-04-2025  Certificate: content_cert.pem

RHEL RHUI Server 7 7server Extras Debug
Expiration: 08-04-2025  Certificate: content_cert.pem

RHEL RHUI Server 7 7server Extras OS
Expiration: 08-04-2025  Certificate: content_cert.pem

RHEL RHUI Server 7 7server Extras Source Srpms
Expiration: 08-04-2025  Certificate: content_cert.pem

RHEL RHUI Server 7 Containers
Expiration: 08-04-2025  Certificate: content_cert.pem
```

```
RHEL RHUI Server 7 Debug
Expiration: 08-04-2025  Certificate: content_cert.pem

RHEL RHUI Server 7 OS
Expiration: 08-04-2025  Certificate: content_cert.pem
-------------------------------------------------------------
```

15.4.2. List Custom Repository Entitlements

1. Navigate to the Red Hat Update Infrastructure Management Tool home screen.

```
[root@rhua ~]# rhui-manager
```

2. Press **n** at the prompt to access the **Entitlements Manager** screen.

3. From the **Entitlements Manager** screen, press **c** at the prompt to list data about the custom repository entitlements.

```
rhui (entitlements) => c

Custom Repository Entitlements
For each entitlement URL listed, the corresponding repositories that
are configured with that entitlement are listed.

/protected/$basearch/os

 Name: Repo 1
URL: protected/i386/os

Name: Repo 2
URL: protected/x86_64/os

-------------------------------------------------------------
--------------
```

15.4.3. Upload a Content Certificate

Red Hat might need to issue a new content certificate if your content certificate is about to expire, or they may need to change the certificate's entitlements. If Red Hat issues a new content certificate, it will need to be uploaded to RHUI.

When you upload a new content certificate, it will be updated in the RHUA and will be used for synchronizing Red Hat repositories. Do not upload a new content certificate before it becomes valid; it will cause your synchronizations to fail until the valid date is reached.

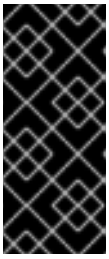
If two or more content certificates provide the same entitlements, the certificate with an expiration date furthest in the future will be used.

1. The Red Hat Update Infrastructure Management Tool expects that the content certificate and its private key are contained in the same file. If you have existing content certificates with separate keys, you can create the single file using the **cat** command at a shell prompt.

```
# cat file1 file2 > file3
```

2. From the **Entitlements Manager** screen, press **u** at the prompt to upload a new or updated Red Hat content certificate.

```
rhui (entitlements) => u
```



IMPORTANT

Content certificates are stored on the same system the Red Hat Update Infrastructure Management Tool is installed on at */etc/pki/rhui*. For security reasons, this directory requires root permissions. If you do not have the correct permissions, the Red Hat Update Infrastructure Management Tool will not allow you to proceed.

3. Enter the full path to the new content certificate; the details of the new certificate to be uploaded display.

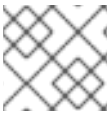
4. Press **y** at the prompt to confirm the information and upload the packages. The Red Hat Update Infrastructure Management Tool lists the current certificates.

Report a bug

CHAPTER 16. RED HAT UPDATE INFRASTRUCTURE 3.0 STATUS CODES, LOG FILES, AND CONFIGURATION FILES

Table 16.1. Status Codes

Status Code	Description
0	Success
1	Repository synchronization error
32	Entitlement CA certificate expiration warning
64	Entitlement CA certificate expiration error



NOTE

Status Code 1 (Repository synchronization error) is not currently returned due to a bug.

Table 16.2. Log Files

Component	File or Directory	Usage
Red Hat Update Appliance	**`/root/.rhui/rhui.log*`**, assuming **`rhui-manager`** is run as root	Red Hat Update Infrastructure Management Tool logs
	`/var/log/kafo/configuration.log`	RHUI installation logging information
	`/var/log/messages`	The most recent four versions are kept in addition to the currently written file.
	`/var/log/httpd/*`	Apache logs
	`/var/log/messages`	Qpid logs
	`/var/log/rhui-subscription-sync.log`	Subscription synchronization log
Content Delivery Server	**`/var/log/httpd/*`**	Apache logs
	`/var/log/httpd/ssl_access_log`	Sample Logging – Successful **`yum repolist`**: the CDS nodes log this client activity
Client	**`/var/log/yum.log`**	Yum command logs

Component	File or Directory	Usage
	/var/log/messages	Client syslog
	Console output from yum commands	

Table 16.3. Configuration Files

Component	File or Directory	Usage
Red Hat Update Appliance	**/etc/pulp/***	Pulp config files
	/etc/rhui/rhui-tools.conf	rhui-manager config files
	/etc/qpid/qpidd.conf	Configuration file
	/etc/pki/rhui/*	Certificates for Red Hat Update Infrastructure
	/etc/pki/pulp/*	Certificates for content
	/etc/rhui-installer/answers.yaml	Used to set up the RHUA
	/etc/rhui/rhui-subscription-sync.conf	Configuration for the subscription synchronization script
	/etc/pulp/admin/admin.conf/etc/pulp/server.conf/etc/pulp/server.conf/etc/pulp/server.conf/etc/pulp/vhosts80/ostree.conf/etc/pulp/vhosts80/rpm.conf	Pulp configuration files
Content Delivery Server	**/etc/pulp/repo_auth.conf etc/pulp/rhui_repo_auth.conf**	Pulp configuration files
Content Delivery Server	**/etc/pki/cds/***	Certificates for CDS
HAProxy	**/etc/haproxy/haproxy.cfg**	HAProxy configuration file

Report a bug

CHAPTER 17. UPGRADE RED HAT UPDATE INFRASTRUCTURE

The following steps describe how to migrate from Red Hat Update Infrastructure 2.1.3 to Red Hat Update Infrastructure 3.0. The current migration handles only Red Hat repositories; it does not handle custom repositories.

1. Mount the downloaded ISO to the *mnt* directory on the RHUA by running the following commands.

```
[root@rhua mnt]# mkdir rhui
[root@rhua mnt]# mount -o loop <ISO> rhui
```

2. Copy the migration script from *rhui/migrate/migrate.py* to the RHUI 2+ RHUA.

```
[root@rhua ~]# scp rhui/migrate/migrate.py rhua2:/directory/in/rhua2
```

3. Log in to the Red Hat Update Infrastructure 2.1.3 RHUA and run the following script to generate the rhui-export-config-{timestamp}.tar.

```
[root@rhua2 ~]# python ./migrate.py -password <PASSWORD>
```

NOTE

Replace <PASSWORD> with the RHUI administrator's password.

4. Copy the rhui-export tarball to a Red Hat Update Infrastructure 3.0 directory.

```
[root@rhua2 ~]# scp rhui-export-config-{timestamp}.tar rhui3:
```

5. Access a fully installed Red Hat Update Infrastructure 3.0.

6. Unpack the rhui-export tarball.

```
[root@rhua ~]# tar xvf rhui-export-config-{timestamp}.tar
```

7. Import data from Red Hat Update Infrastructure 2.1.3.

```
[root@rhua ~]# ./import-rhui2-data.sh
```

8. Enter the RHUI password when prompted.

There are several steps you must perform to complete the migration from Red Hat Update Infrastructure 2.1.3 to Red Hat Update Infrastructure 3.0. For instance, you have to:

- generate a new entitlement certificate for the migrated repositories on the Red Hat Update Infrastructure 3.0 RHUA.
- create a client configuration RPM from that certificate.
- copy and install the RPM on the client, replacing the previous Yum configuration for the Red Hat Update Infrastructure 2.1.3 RHUA.

To elaborate a little more, you must have created such an RPM on Red Hat Update Infrastructure 2.1.3 and installed it on a client, for example, the file name was **example-rpm-1-1.noarch.rpm**. On Red Hat Update Infrastructure 3.0, after migrating the repositories, you generate an entitlement and create an RPM named **example-rpm-2-1.noarch.rpm**.

The Red Hat Update Infrastructure Management Tool enables you specify the "Version of the configuration RPM." If you used "1" on Red Hat Update Infrastructure 2.1.3, use a higher number on Red Hat Update Infrastructure 3.0, for example, "2." Then you can use **yum update example-rpm-2-1.noarch.rpm**_ or **rpm -U example-rpm-2-1.noarch.rpm** on the client. The configuration gets replaced completely, and the client is ready to consume content from Red Hat Update Infrastructure 3.0. If you use a different name, you must remove the old configuration package (**yum remove example-rpm** or **rpm -e example-rpm**) and install the new one.

Report a bug

CHAPTER 18. BACK UP AND RESTORE RED HAT UPDATE INFRASTRUCTURE

This chapter explains the procedure of backing up and restoring your Red Hat Update Infrastructure. See Pulp Backups and MongoDB Backup Methods for more information on backing up Pulp and MongoDB.

18.1. BACK UP THE RED HAT UPDATE APPLIANCE

Follow these steps to back up the Red Hat Update Appliance server. Stopping services does not disable any client instances from updating or installing packages because clients are only connected to the content delivery servers (CDSs), not to the Red Hat Update Appliance server. If you have an automated monitoring solution in place, your monitoring may fail during the backup process.

1. The */var/lib/pulp* directory may be large, depending on how many repositories have been deployed on the Red Hat Update Appliance. See Section 1.4.3 Storage in the Red Hat Satellite 6.1 Installation Guide for specific storage requirements. or use the **du** command from the command-line interface to determine its size.

2. Stop the pulp-server services.

   a. For RHEL 6:

   ```
   # service pulp_workers stop; service pulp_resource_manager stop;
   service pulp_celerybeat stop
   ```

   b. For RHEL 7:

   ```
   # systemctl stop pulp_workers; systemctl stop
   pulp_resource_manager; systemctl stop pulp_celerybeat
   ```

3. Replace **`stop`** with **`status`** in the above commands to verify each service has stopped.

4. It is important that the following files retain their current attributes when backed up.

   - /etc/httpd/conf.d/05-pulp-https.conf
   - /etc/httpd/conf.d/pulp*
   - /etc/httpd/conf.d/ssl.conf
   - /etc/pki/katello-certs-tools/*
   - /etc/pki/pulp/*
   - /etc/pki/rhui/*
   - /etc/pulp/*
   - /etc/puppet/*
   - /etc/rhui/*
   - /etc/rhui/rhui-tools.conf

- /etc/rhui-installer/*
- /etc/qpid/qpidd.conf
- /var/lib/mongodb/pulp_database*
- /var/lib/pulp/*
- /var/log/pulp/*
- /var/log/httpd/*
  Use the following command to back up the files.

```
# cp -a source_files_path destination_files_path
```

5. You may want to back up any generated client entitlement certificates and client configuration RPMs.

6. Restart the pulp-server services.

   a. For RHEL 6:

```
# service pulp_workers start; service pulp_resource_manager
start; service pulp_celerybeat start
```

   b. For RHEL 7:

```
# systemctl start pulp_workers; systemctl start
pulp_resource_manager; systemctl start pulp_celerybeat
```

7. Replace **`start`** with **`status`** in the above commands to verify each service has started.

18.2. RESTORE THE RED HAT UPDATE APPLIANCE

1. Prepare a new Red Hat Update Appliance instance by following Section 4.2, “Register Red Hat Update Infrastructure” and Section 4.3, “Attach a Subscription to the Red Hat Update Appliance”. Once those steps are completed, proceed with the following restoration steps.

2. Stop the pulp-server services.

   a. For RHEL 6:

```
# service pulp_workers stop; service pulp_resource_manager stop;
service pulp_celerybeat stop
```

   b. For RHEL 7:

```
# systemctl stop pulp_workers; systemctl stop
pulp_resource_manager; systemctl stop pulp_celerybeat
```

3. Replace **`stop`** with **`status`** in the above commands to verify each service has stopped.

IMPORTANT

It is crucial that the files included in the restore retain their current attributes.

4. Use the following command to restore the files to their original locations.

```
# cp -a source_files_path destination_files_path
```

5. Restart the pulp-server services.

   a. For RHEL 6:

```
# service pulp_workers start; service pulp_resource_manager
start; service pulp_celerybeat start
```

   b. For RHEL 7:

```
# systemctl start pulp_workers; systemctl start
pulp_resource_manager; systemctl start pulp_celerybeat
```

6. Replace **`start`** with **`status`** in the above commands to verify each service has started.

18.3. BACK UP A CONTENT DELIVERY SERVER

Follow these steps to back up a CDS. To mitigate the outage, if you have more than one CDS, only back up a single CDS at a time. Client instances will automatically fail over to other running CDS instances.

1. The */var/lib/pulp* directory may be large, depending on how many repositories have been deployed on the Red Hat Update Appliance. See Section 1.4.3 Storage in the Red Hat Satellite 6.1 Installation Guide for specific storage requirements. or use the **du** command from the command-line interface to determine its size.

2. Stop the httpd service.

   a. For RHEL 6:

```
# service httpd stop
```

   b. For RHEL 7:

```
# systemctl stop httpd
```

3. Replace **`stop`** with **`status`** in the above commands to verify each service has stopped.

4. It is important that the following files retain their current attributes when backed up.

   - /etc/httpd/conf.d/*.conf
   - /var/lib/pulp/*
   - /var/log/pulp/*
   - /var/log/httpd/*

- /etc/pki/rhui/*
- /etc/pulp/*
  Use the following command to back up the files.

```
# cp -a source_files_path destination_files_path
```

5. In addition to the above files, you may want to back up any generated client entitlement certificates and client configuration RPMs.
6. Restart the service.
   a. For RHEL 6:

```
# service httpd start
```

   b. For RHEL 7:

```
# systemctl start httpd
```

7. Replace **`start`** with **`status`** in the above commands to verify each service has started.

18.4. RESTORE A CONTENT DELIVERY SERVER

1. Prepare a new CDS instance by following all steps in Chapter 7, *Add a Content Delivery Server*. Once those steps are completed, proceed with the following restoration steps.
2. Stop the httpd service.
   a. For RHEL 6:

```
# service httpd stop
```

   b. For RHEL 7:

```
# systemctl stop httpd
```

3. Replace **`stop`** with **`status`** in the above commands to verify each service has stopped.

IMPORTANT

It is crucial that the files included in the restore retain their current attributes.

4. Use the following command to restore the files to their original locations.

```
# cp -a source_files_path destination_files_path
```

5. Restart the httpd service.
   a. For RHEL 6:

```
# service httpd start
```

b. For RHEL 7:

```
# systemctl start httpd
```

6. Replace **start** with **status** in the above commands to verify each service has started.

18.5. BACK UP AN HAPROXY SERVER

Follow these steps to back up an HAProxy server.

1. It is important that the following files retain their current attributes when backed up.
   - /etc/haproxy/haproxy.cfg
   - /etc/pki/rhui/*
2. Use the following command to back up the files.

```
# cp -a source_files_path destination_files_path
```

3. In addition to the above files, you may wish to back up any generated client entitlement certificates and client configuration RPMs.

18.6. RESTORE AN HAPROXY SERVER

1. Prepare a new HAProxy instance by following all steps in Chapter 8, *Add an HAProxy Load Balancer*. Once those steps are completed, proceed with the following restoration steps.

IMPORTANT

It is crucial that the files included in the restore retain their current attributes.

2. Use the following command to restore the files to their original locations.

```
# cp -a source_files_path destination_files_path
```

Report a bug

CHAPTER 19. MIGRATE TO A NEW LOAD BALANCER, OR CHANGE THE NAME OF AN EXISTING LOAD BALANCER

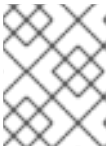
This procedure describes how to change the host name of the load balancer in an already running Red Hat Update Infrastructure (RHUI) environment.

1. Remove the content delivery server (CDS) certificate files from the Red Hat Update Appliance (RHUA).

```
# rm -f /etc/puppet/rhui-secrets/cds-cert.*
```

2. Run rhui-manager with the new load balancer host name, for example:

```
# rhui-installer --cds-lb-hostname=hap02.example.com
```



NOTE

Use any other necessary arguments if you do not want their initial/default values as indicated by the answers file.

3. Add the new load balancer to the RHUI environment, either in the interactive mode (rhui-manager - l - a - …) or in the command-line interface (CLI). For the CLI, use the following example.

```
# rhui haproxy add hap02.example.com ec2-user /root/.ssh/id_rsa_rhua
-u
```

NOTE

Consider unregistering the old load balancer at the same time.

4. Reapply the CDS configuration to all of your CDS nodes, either in the interactive mode (rhui-manager - c - r - …) or in the CLI. For the CLI, use the following example.

```
 # rhui cds reinstall cds01.example.com
```

5. Create an updated client configuration RPM. This can be done in the interactive mode (rhui-manager - e - c - …) or in the CLI. For the CLI, use the following example.

```
# rhui-manager client rpm --private_key /root/rhui.key --
entitlement_cert /root/rhui.crt --rpm_version 2.1 --rpm_name rhui-
repos --dir /root/
```

NOTE

Use the files and the RPM name that you created in Chapter 10, *Client Entitlement Certificate and Client Configuration RPM*.

6. Update the client configuration RPM on all your clients. Your clients should be all set now.

Report a bug

APPENDIX A. RED HAT UPDATE INFRASTRUCTURE MANAGEMENT TOOL MENUS AND COMMANDS

The seven screens within the Red Hat Update Infrastructure Management Tool, accessed from the Red Hat Update Appliance, provide menu options that allow you to configure and update the various components of the Red Hat Update Infrastructure. The following table presents the options available through the Red Hat Update Infrastructure Management Tool.

Table A.1. Red Hat Update Infrastructure Management Tool Menus and Commands

Screen	Screen Command	Menu Option	Menu Option Command
manage repositories	r		
Repository Management		list repositories currently managed by the RHUI	l
		display detailed information on a repository	i
		add a new Red Hat content repository	a
		add a new Red Hat docker container	ad
		create a new custom repository (RPM content only)	c
		delete a repository from the RHUI	d
		upload content to a custom repository (RPM content only)	u
		list packages in a repository (RPM content only)	p
manage content delivery servers (CDS)	c		
Content Delivery Server (CDS) Management		list all known CDS instances managed by the RHUI	l

Screen	Screen Command	Menu Option	Menu Option Command
		register (add) a new CDS instance	a
		reinstall and reapply configuration to an existing CDS instance	r
		unregister (delete) a CDS instance from the RHUi	d
manage HAProxy load-balancer instances	l		
Load-balancer (HAProxy) Management		list all known HAProxy Load-balancer instances managed by the RHUI	l
		register (add) a new HAProxy Load-balancer instance	a
		reinstall and reapply configuration to an existing HAProxy Load-balancer instance	r
		unregister (delete) a HAProxy Load-balancer instance from the RHUI	d
synchronization status and scheduling	s		
Synchronization Status		display repo sync summary	dr
		view the details of the last repository sync	vr
		sync an individual repository immediately	sr
create entitlement certificates and client configuration RPMs	e		

Screen	Screen Command	Menu Option	Menu Option Command
Client Entitlement Management		generate an entitlement certificate	e
		create a client configuration RPM from an entitlement certificate	c
		create a docker client configuration RPM	d
		create an atomic client configuration package	o
manage Red Hat entitlement certificates	n		
Entitlements Manager		list Red Hat content certificate entitlements	l
		list custom repository entitlements	c
		upload a new or updated Red Hat content certificate	u
manage RHUI users	u		
User Manager		change a user's password	p

Report a bug

APPENDIX B. RED HAT UPDATE INFRASTRUCTURE COMMAND-LINE INTERFACE

The majority of RHUI's administrative tasks lie in its installation. After installation, it runs on its own, periodically getting updated packages from the Red Hat CDN and automatically making those packages available to clients.

A command-line interface called Red Hat Update Infrastructure Management Tool (run with the **rhui-manager** command) facilitates the installation. This tool provides interactive prompts for the necessary configuration elements for each RHUI component: RHUA, CDS, and load balancer. This tool also provides a means for taking the content certificate provided by Red Hat for use when connecting to the Red Hat CDN and generating internal, cloud-specific certificates that clients will use to connect to RHUI. The Red Hat Update Infrastructure Management Tool allows the cloud provider to generate a client configuration bundle that will be installed on client RHEL instances. This bundle allows the clients to get updates from the RHUI installation.

Red Hat Update Infrastructure Management Tool is designed to be run using the interactive shell; some functions can also be run from a standard shell prompt. The Red Hat Update Infrastructure Management Tool uses five main commands. For each command's subcommand, a list of options is provided if the subcommand expects one or more options other than **-h** and **--help**.

View all options and commands.

```
[root@rhua ~]# rhui-manager --help
Usage: rhui-manager [options]

OPTIONS
   -h/--help  show this help message and exit
   --debug    enables debug logging
   --config   absolute path to the configuration file; defaults to
/etc/rhui/rhui-tools.conf
   --server   location of the RHUA server (overrides the config file)
   --username if specified, previously saved authentication credentials
are ignored and this username is used to login
   --password used in conjunction with --username

 COMMANDS
   cert          : Red Hat content certificate management
   packages      : package manipulation on repositories
   repo          : repository listing and manipulation
   status        : RHUI status and health information
   client        : Red Hat client management
   subscriptions : Red Hat subscriptions management
```

The following subsections provide more details about the subcommands and options for their associated command.

B.1. CERT

- info: display information about the current content certificate
  - (This subcommand has no options.)
- upload: uploads a new content certificate

- --cert - full path to the new content certificate (required)
- --key - full path to the new content certificate's key

Example

```
[root@rhua ~]# rhui-manager cert upload --cert
/tmp/extra_rhui_files/rhcert.pem
Red Hat Entitlements

  Valid
    Beta RHEL RHUI Everything 7 Debug
    Expiration: 05-17-2026     Certificate: rhcert.pem

    Beta RHEL RHUI Everything 7 OS
    Expiration: 05-17-2026     Certificate: rhcert.pem
```

B.2. PACKAGES

- list: lists all packages in a repository
  - --repo_id - id of the repository to list packages for (required)
- upload: uploads a package or directory of packages to a custom repository
  - --repo_id - id of the custom repository where the packages will be uploaded (required)
  - --packages - path to an .rpm file or directory of RPMs that will be uploaded (required)

Example

```
[root@rhua ~]# rhui-manager packages upload --repo_id my_custom_repo --
packages /tmp/webalizer-2.21-6.2.x86_64.rpm
Uploading /tmp/webalizer-2.21-6.2.x86_64.rpm...
/tmp/webalizer-2.21-6.2.x86_64.rpm successfully uploaded
```

B.3. REPO

- sync: sync a repository
  - --repo_id - identifies the repository to display (required)
- add_by_repo: add Red Hat repositories to the RHUA via repo ID
  - --repo_ids - repo IDs to add, comma-separated (required)
- unused: list of products available but not synced to the RHUA
  - -by_repo_id - List unused products by repo id
- list: lists all repositories in the RHUI
  - (this subcommand has no options)
- add: add a Red Hat repository to the RHUA

- --product_name - product to add the RHUA (required)

- info: displays information on an individual repo
  - --repo_id - identifies the repository to display (required)

Example

```
[root@rhua ~]# rhui-manager repo list
ID                             :: Repository Name

Red Hat Repositories
--------------------
rhel-rhui-server-7-rh-gluster-samba-3.1-os-7Server-x86_64 :: RHEL RHUI
Server 7 Rh-gluster-samba 3.1 OS (7Server-x86_64)

Custom Repositories
--------------------
my_custom_repo                 :: My Custom Repo
```

B.4. STATUS

- (This command has no subcommands.)
  - --code - if specified, only a numeric code for the result will be displayed

Example

```
[root@rhua ~]# rhui-manager status
RHEL RHUI Server 7 Rh-gluster-samba 3.1 OS (7Server-x86_64) ..........
Never


Entitlement CA certificate expiration date = 2038-01-18T10:30:32Z  .... [
OK  ]
0
```

B.5. CLIENT

- rpm: create a client config rpm
  - --private_key - entitlement private key (required)
  - --entitlement_cert - entitlement certificate (required)
  - --rpm_version - version number of the client config rpm (required)
  - --rpm_name - name of the client config rpm (required)
  - --dir - directory where the rpm will be created (required)
  - --unprotected_repos - comma-separated list of unprotected repos to include
- labels: list the labels required for client certificate creation

- (this subcommand has no options)

- cert: create a content certificate for a rhui client
  - --repo_label - identifies the repositories to add. Comma delimited string of repo labels (required)
  - --name - identifies the certificate name (required)
  - --days - number of days cert will be valid (required)
  - --dir - directory where the certificate will be stored (required)

Example

```
[root@rhua ~]# rhui-manager client cert --repo_label rhel-rhui-server-7-
rh-gluster-samba-3.1-os --name gluster-samba-31 --days 365 --dir /tmp
.....................................................................
..............+++
..............+++
Entitlement certificate created at /tmp/gluster-samba-31.crt
```

B.6. SUBSCRIPTIONS

- list: list the registered or available subscriptions
  - --available - list the subscriptions available to the system; this option and --registered are mutually exclusive
  - --pool-only - list only the Pool IDs of the subscriptions one per line
  - --registered - list the subscriptions registered with RHUI (default); this option and --available are mutually exclusive

- register : register the subscription to RHUI
  - --pool - the pool ID of the subscription to register (required)

- unregister : remove the subscription from RHUI
  - --pool - the pool ID of the subscription to unregister (required)

Example

```
[root@rhua ~]# rhui-manager subscriptions list --available
Available subscriptions

  Label: Red Hat Update Infrastructure and RHEL Add-Ons for Providers
  Pool ID: 8a85f98b61e217aa0161ed24079a583e
```

Report a bug

APPENDIX C. RESOLVE COMMON PROBLEMS IN RED HAT UPDATE INFRASTRUCTURE

The following table lists known issues with Red Hat Update Infrastructure 2.1.3. If you encounter any of these issues with Red Hat Update Infrastructure 3.0, report the problem through Bugzilla. See Troubleshooting Red Hat Update Infrastructure Issues for more details about common issues.

Table C.1. Common Problems in Red Hat Update Infrastructure

Category	Issue	Solution
ISO Download/Red Hat Update Infrastructure Certificate	You cannot download the Red Hat Update Infrastructure ISO from the Customer Portal with the RHUI Certificate.	Verify that RHUI entitlements are in place and enabled in your Red Hat Network account. Verify the credibility of the certificate being used to download the ISO. Make sure the certificate is a RHUI consumer certificate; follow the instructions in the Installation Guide for creating the appropriate RHUI consumer content certificate.
Red Hat Update Infrastructure Certificate	You see an error message while uploading Entitlement certificate using rhui-manager.	See https://access.redhat.com/solutions/363844 for more details.
Installation/Configuration	You experience communication issues between the Red Hat Upate Appliance and the CDSs.	Verify the fully qualified domain name (FQDN) is set for the RHUA and CDS and is resolvable. Configure the HTTP proxy properly as described in Bug 726420 – Quick note on proxy URL

Category	Issue	Solution
Synchronization	You cannot synchronize repositories with Red Hat.	Verify the RHUI SKUs are in your account. Verify the proper content certificates are loaded to the RHUA. Look for temporary CDN issues. Look for any HTTP proxy in your environment and make sure you are not hitting an error. The RHUA cannot synchronize to CDSs, typically due to expired qpid certificates: See Knowledgebase solution "CDS sync fails with error "sslv3 alert certificate expired" because of expired qpid CA certificates on RHUI 2.X.""

Category	Issue	Solution
Red Hat Update Appliance/Content Delivery Network Communication	The Red Hat Update Appliance is not communicating with the Content Delivery Network.	Use the content certificate in */etc/pki/rhui/redhat* (the .pem file) to test connectivity and access between the RHUA and the CDN. **`# cd /etc/pki/rhui/redhatwget --certificate=8a85f98146a087b80146afacb3362499.pem --ca-certificate=/etc/rhsm/ca/redhat-uep.pem`** https://cdn.redhat.com/content/dist/rhel/rhui/server/6/6Server/x86_64/os/repodata/repomd.xml Note from the curl (1) man page: If the NSS PEM PKCS#11 module (lib-nsspem.so) is available, then PEM files may be loaded. If you want to use a file from the current directory, precede it with "./" prefix to avoid confusion with a nickname. On each CDS, the entitlement certificate in **`/etc/pki/pulp/content`** can be used to test the availability of the RHUA content using **`# curl --cert ./rhui-ec2-20120619.pem`**. The URL for the repositories hosted on the RHUA always start with **`https://fqdn/pulp/repos`**. You can divulge the remaining URL by: -Looking at the file path on the RHUA under **`/var/lib/pulp/repos`** - Examining the content certificate directly using openssl commands because the OIDs ending in 1.6 contain the path

Category	Issue	Solution
Client/Content Delivery Server Communication	`curl` can be used to verify client communications with the content delivery server nodes as well.	# curl --cert /etc/pki/entitlement/product/content.crt --key /etc/pki/entitlement/key.pem https://ip-10-4-58-34.ec2.internal/pulp/repos/content/dist/rhel/rhui/server/6/6Server/x86_64 /rhui/2.1/os/repodata/repomd.xml -k <?xml version="1.0" encoding="UTF-8"?> <repomd xmlns="http://linux.duke.edu/metadata/repo" xmlns:rpm="http://linux.duke.edu/metadata/rpm"> <revision>1339940325</revision> <data type="other_db"> <location href="repodata/4f86b0ae203bba90d22a8363120c66ed6f37da81-other.sqlite.bz2"/> <checksum type="sha">4f86b0ae203bba90d22a8363120c66ed6f37da81</checksum> <timestamp>1339940328.43</timestamp>
Content Delivery Server Synchronization	The CDS synchronization fails with SSL errors because of expired Qpid certificates	CDS sync fails with error "sslv3 alert certificate expired" due to expired qpid CA certificates on RHUI. https://access.redhat.com/articles/523163
Client/HAProxy communication	All HAProxy nodes are down. Clients have lost access to RHUI repositories.	Add and configure at least one new HAProxy node. If you cannot do so for whatever reason, temporarily change the DNS configuration so that the main load balancer host name (cds.example.com in this guide) resolves to the IP address of one of your CDS nodes. This will allow the clients to avoid the unavailable HAProxy nodes and communicate with the CDS directly.

Report a bug

APPENDIX D. API REFERENCE IN RED HAT UPDATE INFRASTRUCTURE 3.0

NOTE

All material cited here was taken from the Pulp 2.8 repository.

D.1. REPOSITORY APIS

D.1.1. Creation, Deletion, and Configuration

D.1.1.1. Create a Repository

Creates a new repository in Pulp. This call accepts optional parameters for importer and distributor configuration. More detailed description of these parameters can be found below in the documentation of APIs to associate an importer or a distributor to an already existing repository. If these parameters are not passed, the call will only create the repository in Pulp. The real functionality of a repository isn't defined until importers and distributors are added. Repository IDs must be unique across all repositories in the server.

Method: POST

Path: `/pulp/api/v2/repositories/`

Permission: create

Request Body Contents:

- **id** (string) - unique identifier for the repository
- **display_name** (string) - *(optional)* user-friendly name for the repository
- **description** (string) - *(optional)* user-friendly text describing the repository's contents
- **notes** (object) - *(optional)* key-value pairs to programmatically tag the repository
- **importer_type_id** (string) - *(optional)* type id of importer being associated with the repository
- **importer_config** (object) - *(optional)* configuration the repository will use to drive the behavior of the importer. Note that *proxy_password* and *basic_auth_password* will be returned as '*' for security purposes.
- **distributors** (array) - *(optional)* array of objects containing values of distributor_type_id, repo_plugin_config, auto_publish, and distributor_id

Response Codes:

- **201**- the repository was successfully created
- **400**- if one or more of the parameters is invalid
- **409**- if there is already a repository with the given ID
- **500**- if the importer or distributor raises an error during initialization

Return: database representation of the created repository

Sample Request:

```
{
 "display_name": "Harness Repository: harness_repo_1",
 "id": "harness_repo_1",
 "importer_type_id": "harness_importer",
 "importer_config": {
   "num_units": "5",
   "write_files": "true"
 },
 "distributors": [{
              "distributor_id": "dist_1",
              "distributor_type_id": "harness_distributor",
              "distributor_config": {
              "publish_dir": "/tmp/harness-publish",
              "write_files": "true"
              },
              "auto_publish": false
       }],
}
```

Sample 201 Response Body:

```
{
 "scratchpad": {},
 "display_name": "Harness Repository: harness_repo_1",
 "description": null,
 "_ns": "repos",
 "notes": {},
 "content_unit_counts": {},
 "_id": {
   "$oid": "52280416e5e71041ad000066"
 },
 "id": "harness_repo_1",
 "_href": "/pulp/api/v2/repositories/harness_repo_1/"
}
```

D.1.1.2. Update a Repository

Much like create repository is simply related to the repository metadata (as compared to the associated importers/distributors), the update repository call is centered around updating only that metadata.

Method: PUT

Path: **`/pulp/api/v2/repositories/<repo_id>/`**

Permission: update

Request Body Contents: The body of the request is a JSON document with three possible root elements:

- **delta** (object) - *(optional)* object containing keys with values that should be updated on the repository

- **importer_config** (object) - *(optional)* object containing keys with values that should be updated on the repository's importer config
- **distributor_configs** (object) - *(optional)* object containing keys that are distributor ids, and values that are objects containing plugin specific keys/value pairs

Response Codes:

- **200** - if the update was executed and successful
- **202** - if the update was executed but additional tasks were created to update nested distributor configurations
- **400**- if one or more of the parameters is invalid
- **404** - if there is no repository with the give ID

Return: a Call Report containing the database representation of the repository (after changes made by the update) and any tasks spawned to apply the consumer bindings for the repository. See Bind a Consumer to a Repository for details on the bindings tasks that will be generated.

Sample Request:

```
{
 "delta": {
  "display_name" : "Updated"
 },
 "importer_config": {
  "demo_key": "demo_value"
 },
 "distributor_configs": {
  "demo_distributor": {
    "demo_key": "demo_value"
  }
 }
}
```

Sample result value: The result field of the Call Report contains the database representation of the repository

```
{
...
"result": {
   "display_name": "zoo",
   "description": "foo",
   "_ns": "repos",
   "notes": {
     "_repo-type": "rpm-repo"
   },
   "content_unit_counts": {
     "package_group": 2,
     "package_category": 1,
     "rpm": 32,
     "erratum": 4
   },
   "_id": {
```

```
      "$oid": "5328b2983738202945a3bb47"
    },
    "id": "zoo",
    "_href": "/pulp/api/v2/repositories/zoo/"

  },
  ...
}
```

D.1.1.3. Associate an Importer to a Repository

Configures an importer for a previously created Pulp repository. Each repository maintains its own configuration for the importer which is used to dictate how the importer will function when it synchronizes content. The possible configuration values are contingent on the type of importer being added; each importer type will support a different set of values relevant to how it functions.

Only one importer may be associated with a repository at a given time. If a repository already has an associated importer, the previous association is removed. The removal is performed before the new importer is initialized, thus there is the potential that if the new importer initialization fails the repository is left without an importer.

Adding an importer performs the following validation steps before confirming the addition:

- The importer plugin is contacted and asked to validate the supplied configuration for the importer. If the importer indicates its configuration is invalid, the importer is not added to the repository.
- The importer's importer_added method is invoked to allow the importer to do any initialization required for that repository. If the plugin raises an exception during this call, the importer is not added to the repository.
- The Pulp database is updated to store the importer's configuration and the knowledge that the repository is associated with the importer.

The details of the added importer are returned from the call.

Method: POST

Path: `/pulp/api/v2/repositories/<repo_id>/importers/`

Permission: create

Request Body Contents:

- importer_type_id (string) - indicates the type of importer being associated with the repository; there must be an importer installed in the Pulp server with this ID
- importer_config (object) - configuration the repository will use to drive the behavior of the importer

Response Codes:

- **202** - if the association was queued to be performed
- **400**- if one or more of the required parameters is missing, the importer type ID refers to a non-existent importer, or the importer indicates the supplied configuration is invalid

- **404** - if there is no repository with the given ID
- **500**- if the importer raises an error during initialization

Return: a Call Report containing the current state of the association task

Sample Request:

```
{
 "importer_type_id": "harness_importer",
 "importer_config": {
   "num_units": "5",
   "write_files": "true"
 }
}
```

Sample result value for the Task Report: The result field of the Task Report will contain the database representation of the importer (not the full repository details, just the importer)

```
{
 "scratchpad": null,
 "_ns": "repo_importers",
 "importer_type_id": "harness_importer",
 "last_sync": null,
 "repo_id": "harness_repo_1",
 "_id": "bab0f9d5-dfd1-45ef-bd1d-fd7ea8077d75",
 "config": {
   "num_units": "5",
   "write_files": "true"
 },
 "id": "harness_importer"
}
```

Tags: The task created will have the following tags: **`pulp:action:update_importer`**, **`pulp:repository:<repo_id>`**, **`pulp:repository_importer:<importer_type_id>`**

D.1.1.4. Associate a Distributor with a Repository

Configures a distributor for a previously created Pulp repository. Each repository maintains its own configuration for the distributor which is used to dictate how the distributor will function when it publishes content. The possible configuration values are contingent on the type of distributor being added; each distributor type will support a different set of values relevant to how it functions.

Multiple distributors may be associated with a repository at a given time. There may be more than one distributor with the same type. The only restriction is that the distributor ID must be unique across all distributors for a given repository.

Adding a distributor performs the following validation steps before confirming the addition:

- If provided, the distributor ID is checked for uniqueness in the context of the repository. If not provided, a unique ID is generated.
- The distributor plugin is contacted and asked to validate the supplied configuration for the distributor. If the distributor indicates its configuration is invalid, the distributor is not added to the repository.

- The distributor's distributor_added method is invoked to allow the distributor to do any initialization required for that repository. If the plugin raises an exception during this call, the distributor is not added to the repository.
- The Pulp database is updated to store the distributor's configuration and the knowledge that the repository is associated with the distributor.

The details of the added distributor are returned from the call.

Method: POST

Path: `/pulp/api/v2/repositories/<repo_id>/distributors/`

Permission: create

Request Body Contents:

- **distributor_type_id** (string) - indicates the type of distributor being associated with the repository; there must be a distributor installed in the Pulp server with this ID
- **distributor_config** (object) - plugin specific configuration the repository will use to drive the behavior of the distributor
- **distributor_id** (string) - *(optional)* if specified, this value will be used to refer to the distributor; if not specified, one will be randomly assigned to the distributor
- **auto_publish** (boolean) - *(optional)* if true, this distributor will automatically have its publish operation invoked after a successful repository sync. Defaults to false if unspecified

Response Codes:

- **201**- if the distributor was successfully added
- **400**- if one or more of the required parameters is missing, the distributor type ID refers to a non-existent distributor, or the distributor indicates the supplied configuration is invalid
- **404** - if there is no repository with the given ID
- **500**- if the distributor raises an error during initialization

Return: database representation of the distributor (not the full repository details, just the distributor)

Sample Request:

```
{
 "distributor_id": "dist_1",
 "distributor_type_id": "harness_distributor",
 "distributor_config": {
   "publish_dir": "/tmp/harness-publish",
   "write_files": "true"
 },
 "auto_publish": false
}
```

Sample 201 Response Body:

```
{
```

```
 "scratchpad": null,
 "_ns": "repo_distributors",
 "last_publish": null,
 "auto_publish": false,
 "distributor_type_id": "harness_distributor",
 "repo_id": "harness_repo_1",
 "publish_in_progress": false,
 "_id": "cfdd6ab9-6dbe-4192-bde2-d00db768f268",
 "config": {
   "publish_dir": "/tmp/harness-publish",
   "write_files": "true"
 },
 "id": "dist_1"
}
```

D.1.1.5. Update an Importer Associated with a Repository

Update the configuration for an importer that has already been associated with a repository.

Any importer configuration value that is not specified remains unchanged.

Note that the importer's proxy_password and basic_auth_password fields will be returned as * if they are populated. This is done for security purposes.

Method: PUT

Path: `/pulp/api/v2/repositories/<repo_id>/importers/<importer_id>/`

Permission: update

Request Body Contents:

- importer_config (object) - object containing keys with values that should be updated on the importer

Response Codes:

- **202** - if the request was accepted by the server to update the importer when the repository is available
- **400**- if request body parameters are invalid
- **404** - if there is no repository or importer with the specified IDs

Return: a Call Report which includes a spawned task that should be polled for aTask Report

Sample Request:

```
{
 "importer_config": {
   "demo_key": "demo_value"
 }
}
```

Sample result value for the Task Report: The result field of the Task Report contains the database representation of the importer. This does not include the full repository details.

```
{
  "scratchpad": null,
  "_ns": "repo_importers",
  "importer_type_id": "demo_importer",
  "last_sync": "2013-10-03T14:08:35Z",
  "scheduled_syncs": [],
  "repo_id": "demo_repo",
  "_id": {
    "$oid": "524db282dd01fb194283e53f"
  },
  "config": {
    "demo_key": "demo_value"
  },
  "id": "demo_importer"
}
```

Tags: The task created will have the following tags: **`pulp:action:update_importer`**, **`pulp:repository:<repo_id>`**, **`pulp:repository_importer:<importer_id>`**

D.1.1.6. Disassociate an Importer from a Repository

Method: DELETE

Path: **`/pulp/api/v2/repositories/<repo_id>/importers/<importer_id>/`**

Permission: delete

Response Codes:

- **202** - if the request was accepted by the server to disassociate when the repository is available
- **404** - if there is no repository or importer with the specified IDs

Return: a Call Report

Tags: The task created will have the following tags: **`pulp:action:delete_importer`**, **`pulp:repository:<repo_id>`**, **`pulp:repository_importer:<importer_id>`**

D.1.1.7. Update a Distributor Associated with a Repository

Update the configuration for a distributor that has already been associated with a repository. This performs the following actions:

1. Updates the distributor on the server.
2. Rebinds any bound consumers.

Any distributor configuration value that is not specified remains unchanged and any value that is set to explicitly to *None* will be removed from the config.

The first step is represented by a Call Report. Upon completion of step 1 the spawned_tasks field will be populated with links to any tasks required to complete step 2. Updating a distributor causes each binding associated with that repository to be updated as well. See Bind a Consumer to a Repository for details.

Method: PUT

Path: `/pulp/api/v2/repositories/<repo_id>/distributors/<distributor_id>/`

Permission: update

Request Body Contents:

- **distributor_config** (object) - *(optional)* object containing plugin specific keys with values that will update the distributor config
- **delta** (object) - *(optional)* object containing keys with values that will update the distributor object, currently only supports `auto_publish`

Response Codes:

- **202** - if the request was accepted by the server to update the distributor when the repository is available
- **404** - if there is no repository or distributor with the specified IDs

Return: a Call Report

Sample Request:

```
{
 "distributor_config": {
   "demo_key": "demo_value"
 },
 "delta": {
   "auto_publish": true
 }
}
```

Tags: The task created to update the distributor will have the following tags: `pulp:action:update_distributor`, `pulp:repository:<repo_id>`, `pulp:repository_distributor:<distributor_id>`. Information about the binding tasks can be found at Bind a Consumer to a Repository.

D.1.1.8. Disassociate a Distributor from a Repository

Disassociating a distributor performs the following actions:

1. Remove the association between the distributor and the repository.
2. Unbind all bound consumers.

The first step is represented by a Call Report. Upon completion of step 1 the spawned_tasks field will be populated with links to any tasks required complete step 2. The total number of spawned tasks depends on how many consumers are bound to the repository.

Method: DELETE

Path: `/pulp/api/v2/repositories/<repo_id>/distributors/<distributor_id>/`

Permission: delete

Response Codes:

- **202** - if the request was accepted by the server to disassociate when the repository is available
- **404** - if there is no repository or distributor with the specified IDs
- **500**- if the server raises an error during disassociation

Return: a Call Report

Tags: The task created to delete the distributor will have the following tags: **`pulp:action:remove_distributor`**, **`pulp:repository:<repo_id>`**, **`pulp:repository_distributor:<distributor_id>`**

D.1.1.9. Delete a Repository

When a repository is deleted, it is removed from the database and its local working directory is deleted. The content within the repository, however, is not deleted. Deleting content is handled through the orphaned unit process.

Deleting a repository is performed in the following major steps:

- Delete the repository.
- Unbind all bound consumers.

The first step is represented by a Call Report. Upon completion of step 1 the spawned_tasks field will be populated with links to any tasks required to complete step 2. The total number of spawned tasks depends on how many consumers are bound to the repository.

Method: DELETE

Path: **`/pulp/api/v2/repositories/<repo_id>/`**

Permission: delete

Response Codes:

- **202** - if the request was accepted by the server to delete the repository
- **404** - if the requested repository does not exist

Return: a Call Report

Tags: The task created to delete the repository will have the following tags: **`pulp:action:delete`**,**`pulp:repository:<repo_id>`**

D.1.2. Retrieval

D.1.2.1. Retrieve a Single Repository

Retrieves information on a single Pulp repository. The returned data includes general repository metadata, metadata describing any importers and distributors associated with it, and a count of how many content units have been stored locally for the repository.

Method: GET

Path: **`/pulp/api/v2/repositories/<repo_id>/`**

Permission: read

Query Parameters:

- details (boolean) - *(optional)* shortcut for including distributors, importers, and content unit counts
- importers (boolean) - *(optional)* include the “importers” attribute on each repository
- distributors (boolean) - *(optional)* include the “distributors” attribute on each repository

Response Codes:

- **200** - if the repository exists
- **404** - if no repository exists with the given ID

Return: database representation of the matching repository

Sample 200 Response Body:


```
{
 "display_name": "Harness Repository: harness_repo_1",
 "description": null,
 "distributors": [
   {
     "scratchpad": 1,
     "_ns": "repo_distributors",
     "last_publish": "2012-01-25T15:26:32Z",
     "auto_publish": false,
     "distributor_type_id": "harness_distributor",
     "repo_id": "harness_repo_1",
     "publish_in_progress": false,
     "_id": "addf9261-345e-4ce3-ad1e-436ba005287f",
     "config": {
       "publish_dir": "/tmp/harness-publish",
       "write_files": "true"
     },
     "id": "dist_1"
   }
 ],
 "notes": {},
 "scratchpad": {},
 "content_unit_counts": {},
 "last_unit_added": "2012-01-25T15:26:32Z",
 "last_unit_removed": "2012-01-25T15:26:32Z",
 "importers": [
   {
     "scratchpad": 1,
     "_ns": "repo_importers",
     "importer_type_id": "harness_importer",
     "last_sync": "2012-01-25T15:26:32Z",
     "repo_id": "harness_repo_1",
     "sync_in_progress": false,
     "_id": "bbe81308-ef7c-4c0c-b684-385fd627d99e",
     "config": {
       "num_units": "5",
       "write_files": "true"
```

```
      },
      "id": "harness_importer"
    }
  ],
  "id": "harness_repo_1",
  "total_repository_units": 5,
  "locally_stored_units": 3
}
```

D.1.2.2. Retrieve All Repositories

Returns information on all repositories in the Pulp server. It is worth noting that this call will never return a 404; an empty array is returned in the case where there are no repositories.

Method: GET

Path: `/pulp/api/v2/repositories/`

Permission: read

Query Parameters:

- **details** (boolean) - *(optional)* shortcut for including both distributors and importers
- **importers** (boolean) - *(optional)* include the “importers” attribute on each repository
- **distributors** (boolean) - *(optional)* include the “distributors” attribute on each repository

Response Codes:

- **200** - containing the array of repositories

Return: the same format as retrieving a single repository, except the base of the return value is an array of them

Sample 200 Response Body:

```
[
 {
   "display_name": "Harness Repository: harness_repo_1",
   "description": null,
   "last_unit_added": "2012-01-25T15:26:32Z",
   "last_unit_removed": null,
   "distributors": [
     {
       "scratchpad": 1,
       "_ns": "repo_distributors",
       "last_publish": "2012-01-25T15:26:32Z",
       "auto_publish": false,
       "distributor_type_id": "harness_distributor",
       "repo_id": "harness_repo_1",
       "publish_in_progress": false,
       "_id": "addf9261-345e-4ce3-ad1e-436ba005287f",
       "config": {
         "publish_dir": "/tmp/harness-publish",
         "write_files": "true"
```

```
        },
        "id": "dist_1"
      }
    ],
    "notes": {},
    "scratchpad": {},
    "content_unit_counts": {},
    "importers": [
      {
        "scratchpad": 1,
        "_ns": "repo_importers",
        "importer_type_id": "harness_importer",
        "last_sync": "2012-01-25T15:26:32Z",
        "repo_id": "harness_repo_1",
        "sync_in_progress": false,
        "_id": "bbe81308-ef7c-4c0c-b684-385fd627d99e",
        "config": {
          "num_units": "5",
          "write_files": "true"
        },
        "id": "harness_importer"
      }
    ],
    "id": "harness_repo_1"
  }
]
```


D.1.2.3. Advanced Search for Repositories

See Search API for more details on how to perform these searches.

Returns information on repositories in the Pulp server that match your search parameters. It is worth noting that this call will never return a 404; an empty array is returned in the case where there are no repositories.

Method: POST

Path: `/pulp/api/v2/repositories/search/`

Permission: read

Request Body Contents:

- **details** (boolean) - *(optional)* shortcut to include “importers” and “distributors”
- **importers** (boolean) - *(optional)* include the “importers” attribute on each repository
- **distributors** (boolean) - *(optional)* include the “distributors” attribute on each repository

Response Codes:

- **200** - containing the array of repositories

Return: the same format as retrieving a single repository, except the base of the return value is an array of them

Sample 200 Response Body:


```
[
 {
   "display_name": "Harness Repository: harness_repo_1",
   "description": null,
   "distributors": [
     {
       "scratchpad": 1,
       "_ns": "repo_distributors",
       "last_publish": "2012-01-25T15:26:32Z",
       "auto_publish": false,
       "distributor_type_id": "harness_distributor",
       "repo_id": "harness_repo_1",
       "publish_in_progress": false,
       "_id": "addf9261-345e-4ce3-ad1e-436ba005287f",
       "config": {
         "publish_dir": "/tmp/harness-publish",
         "write_files": "true"
       },
       "id": "dist_1"
     }
   ],
   "notes": {},
   "scratchpad": {},
   "content_unit_counts": {},
   "last_unit_added": null,
   "last_unit_removed": null,
   "importers": [
     {
       "scratchpad": 1,
       "_ns": "repo_importers",
       "importer_type_id": "harness_importer",
       "last_sync": "2012-01-25T15:26:32Z",
       "repo_id": "harness_repo_1",
       "sync_in_progress": false,
       "_id": "bbe81308-ef7c-4c0c-b684-385fd627d99e",
       "config": {
         "num_units": "5",
         "write_files": "true"
       },
       "id": "harness_importer"
     }
   ],
   "id": "harness_repo_1"
 }
]
```


Returns information on repositories in the Pulp server that match your search parameters. It is worth noting that this call will never return a 404; an empty array is returned in the case where there are no repositories.

This method is slightly more limiting than the POST alternative, because some filter expressions may not be serializable as query parameters.

Method: GET

Path: `/pulp/api/v2/repositories/search/`

Permission: read

Query Parameters: query params should match the attributes of a Criteria object as defined in Search Criteria. The exception is the 'fields' parameter, which should be specified in singular form as follows: For example: **`/v2/repositories/search/?field=id&field=display_name&limit=20'`**

- **details** (boolean) - *(optional)* shortcut for including both distributors and importers
- **importers** (boolean) - *(optional)* include the "importers" attribute on each repository
- **distributors** (boolean) - *(optional)* include the "distributors" attribute on each repository

Response Codes:

- **200** - containing the array of repositories

Return: the same format as retrieving a single repository, except the base of the return value is an array of them

Sample 200 Response Body:

```
[
 {
   "display_name": "Harness Repository: harness_repo_1",
   "description": null,
   "distributors": [
     {
       "scratchpad": 1,
       "_ns": "repo_distributors",
       "last_publish": "2012-01-25T15:26:32Z",
       "auto_publish": false,
       "distributor_type_id": "harness_distributor",
       "repo_id": "harness_repo_1",
       "publish_in_progress": false,
       "_id": "addf9261-345e-4ce3-ad1e-436ba005287f",
       "config": {
         "publish_dir": "/tmp/harness-publish",
         "write_files": "true"
       },
       "id": "dist_1"
     }
   ],
   "notes": {},
   "scratchpad": {},
   "content_unit_counts": {},
   "last_unit_added": null,
   "last_unit_removed": null,
   "importers": [
     {
       "scratchpad": 1,
       "_ns": "repo_importers",
       "importer_type_id": "harness_importer",
       "last_sync": "2012-01-25T15:26:32Z",
       "repo_id": "harness_repo_1",
       "sync_in_progress": false,
```

```
        "_id": "bbe81308-ef7c-4c0c-b684-385fd627d99e",
        "config": {
          "num_units": "5",
          "write_files": "true"
        },
        "id": "harness_importer"
      }
    ],
    "id": "harness_repo_1"
  }
]
```

D.1.2.4. Retrieve Importers Associated with a Repository

Retrieves the importer (if any) associated with a repository. The array will either be empty (no importer configured) or contain a single entry.

Method: GET

Path: `/pulp/api/v2/repositories/<repo_id>/importers/`

Permission: read

Query Parameters: None

Response Codes:

- **200** - containing an array of importers
- **404** - if there is no repository with the given ID; this will not occur if the repository exists but has no associated importers

Return: database representation of the repository's importer or an empty list

Sample 200 Response Body:

```
[
    {
        "_href": "/pulp/api/v2/repositories/zoo/importers/yum_importer/",
        "_id": {
            "$oid": "563c82fa45ef48043f026c32"
        },
        "_ns": "repo_importers",
        "config": {
            "feed": "http://example.com/repos/zoo/"
        },
        "id": "yum_importer",
        "importer_type_id": "yum_importer",
        "last_sync": "2015-11-06T10:38:23Z",
        "repo_id": "zoo",
        "scratchpad": {
            "previous_skip_list": [],
            "repomd_revision": 1331832478
        }
    }
]
```

-

D.1.2.5. Retrieve an Importer Associated with a Repository

Retrieves the given importer (if any) associated with a repository.

Method: GET

Path: `/pulp/api/v2/repositories/<repo_id>/importers/<importer_id>/`

Permission: read

Query Parameters: None

Response Codes:

- **200** - containing the details of the importer
- **404** - if there is either no repository or importer with a matching ID.

Return: database representation of the repository's importer

Sample 200 Response Body:

```
{
   "_href": "/pulp/api/v2/repositories/zoo/importers/yum_importer/",
   "_id": {
       "$oid": "563c82fa45ef48043f026c32"
   },
   "_ns": "repo_importers",
   "config": {
       "feed": "http://example.com/repos/zoo/"
   },
   "id": "yum_importer",
   "importer_type_id": "yum_importer",
   "last_sync": "2015-11-06T10:38:23Z",
   "repo_id": "zoo",
   "scratchpad": {
       "previous_skip_list": [],
       "repomd_revision": 1331832478
   }
}
```

D.1.2.6. Retrieve Distributors Associated with a Repository

Retrieves all distributors associated with a repository. If the repository has no associated distributors, an empty array is returned.

Method: GET

Path: `/pulp/api/v2/repositories/<repo_id>/distributors/`

Permission: read

Query Parameters: None

Response Codes:

- **200** - containing an array of distributors
- **404** - if there is no repository with the given ID; this will not occur if the repository exists but has no associated distributors

Return: database representations of all distributors on the repository

Sample 200 Response Body:

```
[
 {
   "scratchpad": 1,
   "_ns": "repo_distributors",
   "last_publish": "2012-01-25T15:26:32Z",
   "auto_publish": false,
   "distributor_type_id": "harness_distributor",
   "repo_id": "harness_repo_1",
   "publish_in_progress": false,
   "_id": "addf9261-345e-4ce3-ad1e-436ba005287f",
   "config": {
     "publish_dir": "/tmp/harness-publish",
     "write_files": "true"
   },
   "id": "dist_1"
 }
]
```

D.1.2.7. Retrieve a Distributor Associated with a Repository

Retrieves a single distributor associated with a repository.

Method: GET

Path: **`/pulp/api/v2/repositories/<repo_id>/distributors/<distributor_id>/`**

Permission: read

Query Parameters: None

Response Codes:

- **200** - containing the details of a distributors
- **404** - if there is either no repository or distributor with a matching ID.

Return: database representation of the distributor

Sample 200 Response Body:

```
{
  "scratchpad": 1,
  "_ns": "repo_distributors",
  "last_publish": "2012-01-25T15:26:32Z",
  "auto_publish": false,
  "distributor_type_id": "harness_distributor",
  "repo_id": "harness_repo_1",
```

```
  "publish_in_progress": false,
  "_id": {"$oid": "addf9261-345e-4ce3-ad1e-436ba005287f"},
  "config": {
    "publish_dir": "/tmp/harness-publish",
    "write_files": "true"
  },
  "id": "dist_1"
}
```

D.1.2.8. Advanced Search for Distributors

See Search API for more details on how to perform these searches.

Returns information on distributors in the Pulp server that match your search parameters. It is worth noting that this call will never return a 404; an empty array is returned in the case where there are no distributors.

Method: POST

Path: `/pulp/api/v2/distributors/search/`

Permission: read

Request Body Contents:

Response Codes:

- **200** - containing the array of distributors

Return: a list of distributor objects

Sample 200 Response Body:

```
[
  {
    "repo_id": "el7",
    "last_publish": "2015-04-28T18:19:01Z",
    "auto_publish": null,
    "scheduled_publishes": [],
    "distributor_type_id": "ostree_web_distributor",
    "scratchpad": null,
    "config": {
      "relative_path": "/opt/content/ostree/el7"
    },
    "id": "ostree_web_distributor_name_cli"
  },
  {
    "repo_id": "el6",
    "last_publish": "2015-5-28T18:18:01Z",
    "auto_publish": null,
    "scheduled_publishes": [],
    "distributor_type_id": "ostree_web_distributor",
    "scratchpad": null,
    "config": {
      "relative_path": "/opt/content/ostree/el6"
    },
```

```
    "id": "ostree_web_distributor_name_cli"
  }
]
```

D.1.3. Synchronization

D.1.3.1. Sync a Repository

Syncs content into a repository from a feed source using the repository's importer.

Method: POST

Path: `/pulp/api/v2/repositories/<repo_id>/actions/sync/`

Permission: execute

Request Body Contents:

- **override_config** (object) - *(optional)* importer configuration values that override the importer's default configuration for this sync

Response Codes:

- **202** - if the sync is set to be executed
- **404** - if repo does not exist

Return: a Call Report

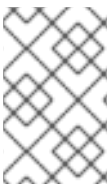
Sample Request:

```
{
  "override_config": {"verify_checksum": false,
                      "verify_size": false},
}
```

Tags: The task created will have the following tags: `pulp:action:sync`, `pulp:repository:<repo_id>`

D.1.3.2. Download a Repository

Downloads content into a repository that was deferred at sync time. This is useful for repositories with importers that are configured with download_policy=(background | on_demand). Content that has already been downloaded will not be downloaded again.



NOTE

This API requires that the Alternate Download Policies features must be installed and configured to work. If it has not been configured, the task dispatched by this API does nothing.

Method: POST

Path: `/pulp/api/v2/repositories/<repo_id>/actions/download/`

Permission: execute

Request Body Contents:

- **verify_all_units** (boolean) - *(optional)* check all units in the repository for corrupted or missing files and re-download files as necessary rather than just downloading files that are known to be missing (defaults to false)

Response Codes:

- **202** - if the download is set to be executed
- **404** - if the repository does not exist

Return: a Call Report

Sample Request:

```
{
  "verify_all_units": false
}
```

Tags: The task created will have the following tags: **`pulp:action:download_repo`**, **`pulp:repository:<repo_id>`**

D.1.3.3. Scheduling a Sync

A repository can be synced automatically using an iso8601 interval. To create a scheduled sync, the interval, sync override config, and other schedule options must be set on the repository's importer.

Method: POST

Path: **`/pulp/api/v2/repositories/<repo_id>/importers/<importer_id>/schedules/sync/`**

Permission: create

Request Body Contents:

- **schedule** (string) - the schedule as an iso8601 interval
- **override_config** (object) - *(optional)* the overridden configuration for the importer to be used on the scheduled sync
- **failure_threshold** (number) - *(optional)* consecutive failures allowed before this scheduled sync is disabled
- **enabled** (boolean) - *(optional)* whether the scheduled sync is initially enabled (defaults to true)

Response Codes:

- **201**- if the schedule was successfully created
- **400**- if one or more of the parameters are invalid
- **404** - if there is no repository or importer with the specified IDs

Return: schedule report representing the current state of the scheduled call

Sample Request:

```
{
 "override_config": {},
 "schedule": "00:00:00Z/P1DT",
 "failure_threshold": 3,
}
```

Sample 201 Response Body:

```
{
 "next_run": "2014-01-27T21:41:50Z",
 "task": "pulp.server.tasks.repository.sync_with_auto_publish",
 "last_updated": 1390858910.292712,
 "first_run": "2014-01-27T21:41:50Z",
 "schedule": "PT1H",
 "args": [
   "demo"
 ],
 "enabled": true,
 "last_run_at": null,
 "_id": "52e6d29edd01fb70bd0d9c37",
 "total_run_count": 0,
 "failure_threshold": 3,
 "kwargs": {
   "overrides": {}
 },
 "resource": "pulp:importer:demo:puppet_importer",
 "remaining_runs": null,
 "consecutive_failures": 0,
 "_href":
"/pulp/api/v2/repositories/demo/importers/puppet_importer/schedules/sync/5
2e6d29edd01fb70bd0d9c37/"
}
```

D.1.3.4. Updating a Scheduled Sync

The same parameters used to create a scheduled sync may be updated at any point.

Method: PUT

Path:
`/pulp/api/v2/repositories/<repo_id>/importers/<importer_id>/schedules/sync/<schedule_id>/`

Permission: create

Request Body Contents:

- **schedule** (string) - *(optional)* new schedule as an iso8601 interval
- **override_config** (object) - *(optional)* new overridden configuration for the importer to be used on the scheduled sync

- **failure_threshold** (number) - *(optional)* new consecutive failures allowed before this scheduled sync is disabled
- **enabled** (boolean) - *(optional)* whether the scheduled sync is enabled

Response Codes:

- **200** - if the schedule was successfully updated
- **400**- if one or more of the parameters are invalid
- **404** - if there is no repository, importer or schedule with the specified IDs

Return: schedule report representing the current state of the scheduled call (See sample response of Scheduling a Sync for details.)

D.1.3.5. Deleting a Scheduled Sync

Delete a scheduled sync to remove it permanently from the importer.

Method: DELETE

Path:
`/pulp/api/v2/repositories/<repo_id>/importers/<importer_id>/schedules/sync/<schedule_id>/`

Permission: delete

Response Codes:

- **200** - if the schedule was deleted successfully
- **404** - if there is no repository, importer or schedule with the specified IDs

Return: null

D.1.3.6. Listing All Scheduled Syncs

All of the scheduled syncs for a given importer may be listed.

Method: GET

Path:
`/pulp/api/v2/repositories/<repo_id>/importers/<importer_id>/schedules/sync/`

Permission: read

Response Codes:

- **200** - if repo, importer exist
- **404** - if there is no repository or importer with the specified IDs

Return: array of schedule reports for all scheduled syncs defined

Sample 200 Response Body:

```
[
   {
       "_href":
"/pulp/api/v2/repositories/test/importers/yum_importer/schedules/sync/54d8
852245ef4876fade7cc2/",
       "_id": "54d8852245ef4876fade7cc2",
       "args": [
           "test"
       ],
       "consecutive_failures": 0,
       "enabled": true,
       "failure_threshold": null,
       "first_run": "2015-02-09T10:00:02Z",
       "kwargs": {
           "overrides": {}
       },
       "last_run_at": "2015-02-09T10:00:23Z",
       "last_updated": 1423476133.825821,
       "next_run": "2015-02-10T10:00:02Z",
       "remaining_runs": null,
       "resource": "pulp:importer:test:yum_importer",
       "schedule": "P1DT",
       "task": "pulp.server.tasks.repository.sync_with_auto_publish",
       "total_run_count": 1
   }
]
```

D.1.3.7. Listing a Single Scheduled Sync

Each scheduled sync may be inspected.

Method: GET

Permission: read

Path: **`/pulp/api/v2/repositories/<repo_id>/importers/<importer_id>/schedules/sync/<schedule_id>/`**

Response Codes:

- **200** - if repo, importer, schedule exist
- **404** - if there is no repository, importer or schedule with the specified IDs

Return: a schedule report for the scheduled sync

Sample 200 Response Body:

```
{
   "_href":
"/pulp/api/v2/repositories/test/importers/yum_importer/schedules/sync/54d8
852245ef4876fade7cc2/",
   "_id": "54d8852245ef4876fade7cc2",
   "args": [
```

```
        "test"
    ],
    "consecutive_failures": 0,
    "enabled": true,
    "failure_threshold": null,
    "first_run": "2015-02-09T10:00:02Z",
    "kwargs": {
        "overrides": {}
    },
    "last_run_at": "2015-02-09T10:00:23Z",
    "last_updated": 1423476133.825821,
    "next_run": "2015-02-10T10:00:02Z",
    "remaining_runs": null,
    "resource": "pulp:importer:test:yum_importer",
    "schedule": "P1DT",
    "task": "pulp.server.tasks.repository.sync_with_auto_publish",
    "total_run_count": 1
}
```

D.1.3.8. Retrieving Sync History

Retrieve sync history for a repository. Each sync performed on a repository creates a history entry.

Method: GET

Permission: read

Path: **`/pulp/api/v2/repositories/<repo_id>/history/sync/`**

Query Parameters:

- **limit** (integer) - *(optional)* the maximum number of history entries to return; if not specified, the entire history is returned
- **sort** (string) - *(optional)* options are ‘ascending’ and ‘descending’; the array is sorted by the sync timestamp
- **start_date** (iso8601 datetime) - *(optional)* any entries with a timestamp prior to the given date are not returned
- **end_date** (iso8601 datetime) - *(optional)* any entries with a timestamp after the given date are not returned

Response Codes:

- **200** - if the history was successfully retrieved
- **404** - if the repository id given does not exist

Return: an array of sync history entries

Sample 200 Response Body:

```
[
 {
  "result": "success",
```

```
  "importer_id": "my_demo_importer",
  "exception": null,
  "repo_id": "demo_repo",
  "traceback": null,
  "started": "1970:00:00T00:00:00Z",
  "completed": "1970:00:00T00:00:01Z",
  "importer_type_id": "demo_importer",
  "error_message": null,
 }
]
```

D.1.4. Publication

D.1.4.1. Publish a Repository

Publish content from a repository using a repository's distributor. This call always executes asynchronously and will return a Call Report.

Method: POST

Path: **`/pulp/api/v2/repositories/<repo_id>/actions/publish/`**

Permission: execute

Request Body Contents:

- **id (string)** - identifies which distributor on the repository to publish
- **override_config** (object) - *(optional)* distributor configuration values that override the distributor's default configuration for this publish

Response Codes:

- **202** - if the publish is set to be executed
- **404** - if repo does not exist

Return: a Call Report representing the current state of the sync

Sample Request:

```
{
  "id": "distributor_1",
  "override_config": {},
}
```

Tags: The task created will have the following tags: **`pulp:action:publish`**, **`pulp:repository:<repo_id>`**

D.1.4.2. Scheduling a Publish

A repository can be published automatically using an iso8601 interval. To create a scheduled publish, the interval, publish override config, and other schedule options must be set on a repository's distributor.

Method: POST

Path: `/pulp/api/v2/repositories/<repo_id>/distributors/<distributor_id>/schedules/publish/`

Permission: create

Request Body Contents:

- **schedule** (string) - the schedule as an iso8601 interval
- **override_config** (object) - *(optional)* the overridden configuration for the distributor to be used on the scheduled publish
- **failure_threshold** (number) - *(optional)* consecutive failures allowed before this scheduled publish is disabled
- **enabled** (boolean) - *(optional)* whether the scheduled publish is initially enabled (defaults to true)

Response Codes:

- **201**- if the schedule was successfully created
- **400**- if one or more of the parameters are invalid
- **404** - if there is no repository or distributor with the specified IDs

Return: schedule report representing the current state of the scheduled call

Sample Request:

```
{
 "override_config": {},
 "schedule": "PT1H",
 "failure_threshold": 3,
}
```

Sample 201 Response Body:

```
{
 "next_run": "2014-01-27T21:27:56Z",
 "task": "pulp.server.tasks.repository.publish",
 "last_updated": 1390858076.682694,
 "first_run": "2014-01-27T21:27:56Z",
 "schedule": "PT1H",
 "args": [
   "demo",
   "puppet_distributor"
 ],
 "enabled": true,
 "last_run_at": null,
 "_id": "52e6cf5cdd01fb70bd0d9c34",
 "total_run_count": 0,
 "failure_threshold": 3,
 "kwargs": {
   "overrides": {}
```

```
 },
 "resource": "pulp:distributor:demo:puppet_distributor",
 "remaining_runs": null,
 "consecutive_failures": 0,
 "_href":
"/pulp/api/v2/repositories/demo/distributors/puppet_distributor/schedules/
publish/52e6cf5cdd01fb70bd0d9c34/"
}
```

D.1.4.3. Updating a Scheduled Publish

The same parameters used to create a scheduled publish may be updated at any point.

Method: PUT

Path: **`/pulp/api/v2/repositories/<repo_id>/distributors/<distributor_id>/schedules/publish/<schedule_id>/`**

Permission: create

Request Body Contents:

- **schedule** (string) - *(optional)* new schedule as an iso8601 interval
- **override_config** (object) - *(optional)* new overridden configuration for the importer to be used on the scheduled sync
- **failure_threshold** (number) - *(optional)* new consecutive failures allowed before this scheduled sync is disabled
- **enabled** (boolean) - *(optional)* whether the scheduled sync is enabled

Response Codes:

- **200** - if the schedule was successfully updated
- **400**- if one or more of the parameters are invalid
- **404** - if there is no repository, distributor or schedule with the specified IDs

Return: schedule report representing the current state of the scheduled call (See sample response of Scheduling a Publish for details.)

D.1.4.4. Deleting a Scheduled Publish

Delete a scheduled publish to remove it permanently from the distributor.

Method: DELETE

Path: **`/pulp/api/v2/repositories/<repo_id>/distributors/<distributor_id>/schedules/publish/<schedule_id>/`**

Permission: delete

Response Codes:

- **200** - if the schedule was deleted successfully
- **404** - if there is no repository, distributor or schedule with the specified IDs

Return: null

D.1.4.5. Listing All Scheduled Publishes

All of the scheduled publishes for a given distributor may be listed.

Method: GET

Path: **`/pulp/api/v2/repositories/<repo_id>/distributors/<distributor_id>/schedules/publish/`**

Permission: read

Response Codes:

- **200** - if repo, distributor exist
- **404** - if there is no repository or distributor with the specified IDs

Return: array of schedule reports for all scheduled publishes defined (See sample response of Scheduling a Publish for details.)

Sample 200 Response Body:

```
{
    "_href":
"/pulp/api/v2/repositories/test/distributors/yum_distributor/schedules/pub
lish/54d88df045ef4876fb50c994/",
    "_id": "54d88df045ef4876fb50c994",
    "args": [
        "test",
        "yum_distributor"
    ],
    "consecutive_failures": 0,
    "enabled": true,
    "failure_threshold": null,
    "first_run": "2015-02-09T10:37:36Z",
    "kwargs": {
        "overrides": {}
    },
    "last_run_at": "2015-02-09T10:38:23Z",
    "last_updated": 1423478256.805917,
    "next_run": "2015-02-10T10:37:36Z",
    "remaining_runs": null,
    "resource": "pulp:distributor:test:yum_distributor",
    "schedule": "P1DT",
    "task": "pulp.server.tasks.repository.publish",
    "total_run_count": 1
}
```

D.1.4.6. Listing a Single Scheduled Publish

Each scheduled publish may be inspected.

Method: GET

Permission: read

Path: **`/pulp/api/v2/repositories/<repo_id>/distributors/<distributor_id>/schedules/publish/<schedule_id>/`**

Response Codes:

- **200** - if repo, distributor or schedule exist
- **404** - if there is no repository, distributor or schedule with the specified IDs

Return: a schedule report for the scheduled publish (See sample response of Scheduling a Publish for details.)

Sample 200 Response Body:

```
{
   "_href":
"/pulp/api/v2/repositories/test/distributors/yum_distributor/schedules/pub
lish/54d88df045ef4876fb50c994/",
   "_id": "54d88df045ef4876fb50c994",
   "args": [
       "test",
       "yum_distributor"
   ],
   "consecutive_failures": 0,
   "enabled": true,
   "failure_threshold": null,
   "first_run": "2015-02-09T10:37:36Z",
   "kwargs": {
       "overrides": {}
   },
   "last_run_at": "2015-02-09T10:38:23Z",
   "last_updated": 1423478256.805917,
   "next_run": "2015-02-10T10:37:36Z",
   "remaining_runs": null,
   "resource": "pulp:distributor:test:yum_distributor",
   "schedule": "P1DT",
   "task": "pulp.server.tasks.repository.publish",
   "total_run_count": 1
}
```

D.1.4.7. Retrieving Publish History

Retrieve publish history for a repository. Each publish performed on a repository creates a history entry.

Method: GET

Permission: read

Path: `/pulp/api/v2/repositories/<repo_id>/history/publish/<distributor_id>/`

Query Parameters:

- **limit** (integer) - *(optional)* the maximum number of history entries to return; if not specified, the entire history is returned
- **sort** (string) - *(optional)* options are 'ascending' and 'descending'; the array is sorted by the publish timestamp
- **start_date** (iso8601 datetime) - *(optional)* any entries with a timestamp prior to the given date are not returned
- **end_date** (iso8601 datetime) - *(optional)* any entries with a timestamp after the given date are not returned

Response Codes:

- **200** - if the history was successfully retrieved
- **404** - if the repository id given does not exist

Return: an array of publish history entries

Sample 200 Response Body:

```
[
 {
  "result": "success",
  "distributor_id": "my_demo_distributor",
  "distributor_type_id": "demo_distributor",
  "exception": null,
  "repo_id": "demo_repo",
  "traceback": null,
  "started": "1970:00:00T00:00:00Z",
  "completed": "1970:00:00T00:00:01Z",
  "error_message": null,
 }
]
```

D.1.5. Content Retrieval

D.1.5.1. Advanced Unit Search

A Unit Association Criteria can be used to search for units within a repository.

Method: POST

Path: `/pulp/api/v2/repositories/<repo_id>/search/units/`

Permission: read

Request Body Contents:

- **criteria** (object) - a UnitAssociationCriteria

Response Codes:

- **200** - if the search executed
- **400**- if the criteria is missing or not valid
- **404** - if the repository is not found

Return: array of objects representing content unit associations

Sample Request:

```
{
  "criteria": {
    "fields": {
      "unit": [
        "name",
        "version"
      ]
    },
    "type_ids": [
      "rpm"
    ],
    "limit": 1
  }
}
```

Sample 200 Response Body:

```
[
  {
    "updated": "2013-09-04T22:12:05Z",
    "repo_id": "zoo",
    "created": "2013-09-04T22:12:05Z",
    "_ns": "repo_content_units",
    "unit_id": "4a928b95-7c4a-4d23-9df7-ac99978f361e",
    "metadata": {
      "_id": "4a928b95-7c4a-4d23-9df7-ac99978f361e",
      "version": "4.1",
      "name": "bear",
      "pulp_user_metadata": {}
    },
    "unit_type_id": "rpm",
    "owner_type": "importer",
    "_id": {
      "$oid": "522777f5e19a002faebebf79"
    },
    "id": "522777f5e19a002faebebf79",
    "owner_id": "yum_importer"
  }
]
```

D.2. TASK MANAGEMENT

Pulp can execute almost any call asynchronously and some calls are always executed asynchronously. Pulp provides REST APIs to inspect and manage the tasks executing these calls.

D.2.1. Task Report

The task information object is used to report information about any asynchronously executed task.

- **_href** (string) - uri path to retrieve this task report object.
- **state** (string) - the current state of the task. The possible values include: 'waiting', 'skipped', 'running', 'suspended', 'finished', 'error' and 'canceled'.
- **task_id** (string) - the unique id of the task that is executing the asynchronous call
- **task_type** (string) - deprecated the fully qualified (package/method) type of the task that is executing the asynchronous call. The field is empty for tasks performed by consumer agent.
- **progress_report** (object) - arbitrary progress information, usually in the form of an object
- **result** (any) - the return value of the call, if any
- **exception** (null or string) - deprecated the error exception value, if any
- **traceback** (null or array) - deprecated the resulting traceback if an exception was raised
- **start_time** (null or string) - the time the call started executing
- **finish_time** (null or string) - the time the call stopped executing
- **tags** (array) - arbitrary tags useful for looking up the Call Report
- **spawned_tasks** (array) - List of objects containing the uri and task id for any tasks that were spawned by this task.
- **worker_name** (string) - The worker associated with the task. This field is empty if a worker is not yet assigned.
- **queue** (string) - The queue associated with the task. This field is empty if a queue is not yet assigned.
- **error** (null or object) - Any, errors that occurred that did not cause the overall call to fail. See Error Details.

NOTE

The exception and traceback fields have been deprecated as of Pulp 2.4. The information about errors that have occurred will be contained in the error block. See Error Details for more information.

Example Task Report:

```
{
 "_href": "/pulp/api/v2/tasks/0fe4fcab-a040-11e1-a71c-00508d977dff/",
 "state": "running",
 "worker_name": "reserved_resource_worker-0@your.domain.com",
 "task_id": "0fe4fcab-a040-11e1-a71c-00508d977dff",
```

```
 "task_type": "pulp.server.tasks.repository.sync_with_auto_publish",
 "progress_report": {}, # contents depend on the operation
 "result": null,
 "start_time": "2012-05-17T16:48:00Z",
 "finish_time": null,
 "exception": null,
 "traceback": null,
 "tags": [
   "pulp:repository:f16",
   "pulp:action:sync"
 ],
 "spawned_tasks": [{"href": "/pulp/api/v2/tasks/7744e2df-39b9-46f0-bb10-
feffa2f7014b/",
                    "task_id": "7744e2df-39b9-46f0-bb10-feffa2f7014b" }],
 "error": null
}
```

D.2.2. Polling Task Progress

Poll a task for progress and result information for the asynchronous call it is executing. Polling returns a Task Report

Method: GET

Path: `/pulp/api/v2/tasks/<task_id>/`

Permission: read

Response Codes:

- **200** - if the task is found
- **404** - if the task is not found

Return: a Task Report representing the task queried

D.2.3. Cancelling a Task

Some asynchronous tasks may be canceled by the user before they complete. A task must be in the waiting or running states in order to be canceled.

NOTE

It is possible for a task to complete or experience an error before the cancellation request is processed, so it is not guaranteed that a task's final state will be 'canceled' as a result of this call. In these instances this method call will still return a response code of 200.

Method: DELETE

Path: `/pulp/api/v2/tasks/<task_id>/`

Permission: delete

Response Codes:

- **200** - if the task cancellation request was successfully received
- **404** - if the task is not found

Return: null

D.2.4. Listing Tasks

All currently running and waiting tasks may be listed. This returns an array of Task Report instances. the array can be filtered by tags.

Method: GET

Path: `/pulp/api/v2/tasks/`

Permission: read

Query Parameters:

- **tag** (string) - *(optional)* only return tasks tagged with all tag parameters

Response Codes:

- **200** - containing an array of tasks

Return: array of Task Report

D.2.5. Deleting Completed Tasks

All completed tasks with states finished, error, skipped may be deleted. This call returns response code 204 if successful or code 403 if the request is forbidden.

Method: DELETE

Path: `/pulp/api/v2/tasks/`

Permission: delete

- **state** (string) - *(optional)* only delete tasks currently in this state

For example:

`/pulp/api/v2/tasks/?state=finished&state=skipped`

Response Codes:

- **204** - if the tasks were successfully deleted
- **403** - if there was a forbidden request

Return: httpResponse or pulp Exception

D.2.6. Searching for Tasks

API callers may also search for tasks. This uses a search criteria document.

Method: POST

Path: `/pulp/api/v2/tasks/search/`

Permission: read

Request Body Contents: include the key “criteria” whose value is a mapping structure as defined in Search Criteria

Response Codes:

- **200** - containing the list of tasks

Return: the same format as retrieving a single task, except the base of the return value is a list. If no results are found, an empty list is returned.

Method: GET

Path: `/pulp/api/v2/tasks/search/`

Permission: read

Query Parameters: query params should match the attributes of a Criteria object as defined in Search Criteria. The exception is that field names should be specified in singular form with as many ‘field=foo’ pairs as needed.

For example:

`/pulp/api/v2/tasks/search/?field=id&field=task_type&limit=20`**Response Codes:**

- **200** - containing the array of tasks.

D.3. TASK GROUP MANAGEMENT

D.3.1. Cancelling Tasks in a Task Group

All asynchronous tasks in a particular task group may be canceled by the user before they complete. A task must be in the *waiting* or *running* state in order to be canceled.

NOTE

=== It is possible for a task to complete or experience an error before the cancellation request is processed, so it is not guaranteed that a task’s final state will be ‘canceled’ as a result of this call. In these instances this method call will still return a response code of 200. ===

Method: DELETE

Path: `/pulp/api/v2/task_groups/<group_id>/`

Permission: delete

Response Codes:

- **200** - if the task group cancellation request was successfully received

- **404** - if the task group is not found

Return: null

D.3.2. Task Group Summary

Task Group Summary object summarizes the state of all the tasks belonging to a task group.

- **accepted** (int) - number of tasks in 'accepted' state
- **finished** (int) - number of tasks in 'finished' state
- **running** (int) - number of tasks in 'running' state
- **canceled** (int) - number of tasks in 'canceled' state
- **waiting** (int) - number of tasks in 'waiting' state
- **skipped** (int) - number of tasks in 'skipped' state
- **suspended** (int) - number of tasks in 'suspended' state
- **error** (int) - number of tasks in 'error' state
- **total** (int) - total number of tasks in the task group

Example task group summary:

```
{
 "accepted": 0,
 "finished": 100,
 "running": 4,
 "canceled": 0,
 "waiting": 2,
 "skipped": 0,
 "suspended": 0,
 "error": 0,
 "total": 106
}
```

D.3.3. Polling Task Group Progress

Poll a group of tasks for progress summary. Polling returns a Cancelling Tasks in a Task Group

Method: GET

Path: `/pulp/api/v2/task_groups/<task_group_id>/state_summary/`

Permission: read

Response Codes:

- **200** - if the task group is found
- **404** - if the task group id is not found

Return: a Cancelling Tasks in a Task Group summarizing the state of all tasks belonging to queried task group id

Report a bug

APPENDIX E. GNU GENERAL PUBLIC LICENSES

E.1. GNU GENERAL PUBLIC LICENSE VERSION 2.0

GNU GENERAL PUBLIC LICENSE Version 2, June 1991

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software—to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation’s software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Lesser General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author’s protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone’s free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Progr am or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no

more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program 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.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public 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.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

- one line to give the program's name and an idea of what it does.
- Copyright © yyyy name of author
- This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.
- This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.
- You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA.
- Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

```
Gnomovision version 69, Copyright (C) year name of author
Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type `show w`.
This is free software, and you are welcome to redistribute it under
certain conditions; type `show c` for details.
```

The hypothetical commands **`show w`** and **`show c`** should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than **`show w`** and **`show c`**; they could even be mouse-clicks or menu items—whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

```
Yoyodyne, Inc., hereby disclaims all copyright interest in the program
`Gnomovision'
(which makes passes at compilers) written by James Hacker.

signature of Ty Coon, 1 April 1989
Ty Coon, President of Vice
```

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the [GNU Lesser General Public License] (http://www.gnu.org/licenses/lgpl.html) instead of this License.

E.2. GNU GENERAL PUBLIC LICENSE VERSION 3.0

GNU GENERAL PUBLIC LICENSE Version 3, 29 June 2007

Copyright © 2007 Free Software Foundation, Inc. http://fsf.org/

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program—to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS

0. Definitions

"This License" refers to version 3 of the GNU General Public License.

"Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

A "covered work" means either the unmodified Program or a work based on the Program.

To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code

The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work’s System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this

License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work’s users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies

You may convey verbatim copies of the Program’s source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

a) The work must carry prominent notices stating that you modified it, and giving a relevant date.

b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".

c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.

d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.

b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.

d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product

from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or

b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or

c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or

d) Limiting the use for publicity purposes of names of licensors or authors of the material; or

e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or

f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party’s predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License

The Free Software Foundation may publish revised and/or new versions of the GNU General Public 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.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy’s public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

- one line to give the program's name and a brief idea of what it does
- Copyright © <year> <name of author>
- This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.
- This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.
- You should have received a copy of the GNU General Public License along with this program. If not, see http://www.gnu.org/licenses/.
- Also add information on how to contact you by electronic and paper mail.

If the program does terminal interaction, make it output a short notice like this when it starts in an interactive mode:

```
<program>  Copyright (C) <year>  <name of author>
This program comes with ABSOLUTELY NO WARRANTY; for details type `show w`.
This is free software, and you are welcome to redistribute it under
certain conditions; type `show c` for details.
```

The hypothetical commands **show w** and **show c** should show the appropriate parts of the General Public License. Of course, your program's commands might be different; for a GUI interface, you would use an "about box".

You should also get your employer (if you work as a programmer) or school, if any, to sign a "copyright disclaimer" for the program, if necessary. For more information on this, and how to apply and follow the GNU GPL, see http://www.gnu.org/licenses.

The GNU General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License. But first, please read http://www.gnu.org/philosophy/why-not-lgpl.html.

Report a bug

APPENDIX F. RED HAT UPDATE INFRASTRUCTURE SYSTEM ADMINISTRATOR'S GUIDE DOCUMENT REVISION HISTORY

Version	Date	Change	Author
Beta	10/13/2016	Initial Release	Les Williams
Beta	12/21/2016	Revised for second beta release	Les Williams
Beta	01/5/2017	Revised to address BZ 1409575, BZ 1410330, and BZ 1410717	Les Williams
3.0	03/02/2017	General Availability	Les Williams
3.0	03/06/2017	Revised to address BZ 1406821, BZ 1429766, and add path for default password	Les Williams
3.0	03/07/2017	Revised Section 4.7 to include repository for Gluster Storage and RHEL 6	Les Williams
3.0	03/08/2017	Updated the link to the Red Hat Certified Cloud and Service Provider Certification Workflow Guide in Chapter 14	Les Williams
3.0	04/19/2017	Revised Section 12.6 to reflect how to restart new pulp services	Les Williams
3.0	04/20/2017	Revised Section 12.6 to reflect how to stop and start new pulp services	Les Williams
3.0	04/25/2017	Revised Section 15.7 to reflect how to synchronize the OSTree repository	Les Williams
3.0	05/02/2017	Revised Chapter 1 to clarify versions and RHUA setup; revised Chapters 1, 3, and 9 to add details about round-robin DNS	Les Williams
3.0	05/05/2017	Revised Section 15.1 to point to the complete list of available repositories	Les Williams

Version	Date	Change	Author
3.0	05/10/2017	Revised to include Gluster Storage working wth RHEL 6; revised password reset information in Chapter 6; minor revisions to Chapters 10, 11, and 12 for repositories	Les Williams
3.0	05/22/2017	Revised to include minimum number of CDS nodes; included information about quorum and split-brain management for Gluster Storage; added an issue in Appendix C regarding client/HAProxy communication; revised Figure 1.1	Les Williams
3.0	05/23/2017	Added an entry for the HAProxy configuration file in Table 17.2	Les Williams
3.0	05/25/2017	Added Section 10.3 for checking if a repository has synchronized	Les Williams
3.0	06/09/2017	Removed reference to an embedded load balancer on the RHUA in Section 1.3.3 and Section 16.1	Les Williams
3.0	06/22/2017	Added Chapter 19 to address how to back up and restore RHUI	Les Williams
3.0	06/26/2017	Revised Section 10.3 to address how to restart Pulp services after rebooting for synchronization errors	Les Williams
3.0	07/10/2017	Revised Sections 6.6 and 6.7 to clarifying initial password usage	Les Williams
3.0	07/28/2017	Added Sections 5.2 and 5.3 to explain how to extend the storage volume and how to expand block storage size	Les Williams
3.0	08/04/2017	Revised Section 4.6 to reflect new Subscription Management functionality on the Red Hat Customer Portal	Les Williams

Version	Date	Change	Author
3.0	08/08/2017	Revised Chapter 5 and Section 6.6 to reflect how to set up NFS storage	Les Williams
3.0	08/15/2017	Added Section 4.6.2 to reflect how to download an entitlement certificate for an unregistered system; added Section 4.6.3 to explain how entitlement certificates are downloaded; revised Sections 6.1.1 and 6.1.2 to leave the passphrase field blank when generating a key pair	Les Williams
3.0	08/25/2017	Revised Table 3.1 to show the correct port settings	Les Williams
3.0	08/29/2017	Revised Section 4.6.1 title to Download an Entitlement Certificate for a Registered System	Les Williams
3.0	09/06/2017	Revised title line and first line of Table 3.1 to show the correct usage	Les Williams
3.0	09/28/2017	Revised Section 5.1.1 to include information about port access if Gluster peer probe fails	Les Williams
3.0	10/06/2017	Revised Sections 10.3, 12.5, 19.1, 19.2, 19.3, and 19.4 to add steps for using systemctl with RHEL 7	Les Williams
3.0	10/25/2017	Added Chapter 20 to add steps for changing a load balancer's name or migrating to a new load balancer; added a note to Section 6.6 to point to Chapter 20	Les Williams
3.0.1	12/07/2017	Revised Chapters 1, 3, 4, 6, and 12 to reflect that using an ISO to install RHUI 3.0 is optional and to reflect how to enable required repositories; added status codes to Chapter 17	Les Williams
3.0.1	12/19/2017	Updated the repository names in Section 4.7, Step 4	Les Williams

Version	Date	Change	Author
3.0.2	04/11/2018	Added Section 15.2.8 to describe how to delete packages in a custom repo	Les Williams
3.0.3	05/16/2018	Revised to address the following bugs (BZ#1506872, BZ#1506875, BZ#1450430, BZ#1428756, BZ#1483311, BZ#1488613, BZ#1538430, BZ#1485725, and BZ#1199426) and enhancements (BZ#1510136, BZ#1563113, and BZ#1443286).	Les Williams
3.0.4	07/10/2018	Added a note to Section 10.3 regarding disabling rhn-plugin and subscription-manager plugin per BZ#1415681; revised Section 6.7 per BZ#1297538; added Section 10.4, Working with EUS per BZ#1504229	Les Williams
3.0.5	09/05/2018	Revised locations of client configuration RPMs and tar files	Les Williams
3.0.5	09/19/2018	Revised links in Chapter 5 and revised text to recommend using three nodes for storage	Les Williams
3.0.5	10/18/2018	Revised Step 3 in Chapter 17 to allow for using the correct administrator password when migrating from RHUI 2.1.3 to RHUI 3.0	Les Williams