Red Hat Satellite 6.2

Host Configuration Guide

A guide to managing hosts in a Red Hat Satellite 6 environment.
Edition 1.0

Last Updated: 2018-06-25

Red Hat Satellite 6.2 Host Configuration Guide

A guide to managing hosts in a Red Hat Satellite 6 environment.
Edition 1.0

Red Hat Satellite Documentation Team
satellite-doc-list@redhat.com

Legal Notice

Abstract

The Red Hat Satellite 6 Host Configuration Guide describes how to configure and work with hosts in a Red Hat Satellite environment. Before continuing with this workflow you must have successfully installed a Red Hat Satellite 6 Server and any required Capsule Servers.

Table of Contents

CHAPTER 1. USING THE RED HAT SATELLITE CONTENT DASHBOARD

The Red Hat Satellite content dashboard provides a status overview of the subscriptions and hosts currently registered, an overview of promotions and synchronization, and a list of the latest notifications.

Navigate to **Monitor** → **Dashboard** to access the content dashboard. The dashboard can be rearranged by clicking on a section title and dragging the section to a different position. The following sections are available:

Content Host Subscription Status

An overview of the subscriptions currently consumed by the hosts registered to Satellite. A subscription is a purchased certificate that unlocks access to software, upgrades, and security fixes for hosts. The following table shows the possible states of subscriptions.

Table 1.1. Host Subscription States

State	Description	Icon
`Invalid`	Hosts that have products installed, but are not correctly subscribed. These hosts need attention immediately.	🟥
`Partial`	Hosts that have a subscription and a valid entitlement, but are not using their full entitlements. These hosts should be monitored to ensure they are configured as expected.	🟧
`Valid`	Hosts that have a valid entitlement and are using their full entitlements.	🟩

Click the subscription type to view content hosts associated with subscriptions of the selected type.

Latest Events

A list of messages produced by hosts including administration information, product and subscription changes, and any errors.

Monitor this section for global notifications sent to all users and to detect any unusual activity or errors.

Sync Overview

An overview of all products or repositories enabled in Satellite and their Synchronization status. All products that are in the queue for synchronization, are unsynchronized or have been previously synchronized are listed in this section. Click a product name to view the synchronization status.

Host Collections

A list of all host collections in Satellite and their status, including the number of content hosts in each host collection. Click a host collection name to view that host collection.

Current Subscription Totals

An overview of the current subscription totals that shows the number of active subscriptions, the number of subscriptions that expire in the next 120 days, and the number of subscriptions that have recently expired. Click the number to list subscriptions of the selected type.

Content Views Overview

A list of all Content Views in Satellite and their publish status.

Errata Overview

A list of all errata available for hosts registered to Satellite.

Task Status

A summary of all current tasks, grouped by their state and result. Click the number to go to the list of corresponding tasks.

Latest Warning/Error Tasks

A list of the latest tasks that have been stopped due to a warning or error. Click a task to see more details.

NOTE

It is not possible to change the date format displayed in the Satellite web UI.

1.1. MANAGING TASKS

Red Hat Satellite keeps a complete log of all planned or performed tasks, such as repositories synchronised, errata applied, Content Views published, and so on. To review the log, navigate to **Monitor** → **Tasks**. The page enables you to search for specific tasks, view their status and details, and resume those that resulted in an error, if applicable.

The tasks are managed using the Dynflow engine. Remote tasks have a timeout which can be adjusted as needed.

Procedure 1.1. To Adjust Timeout Settings:

1. Navigate to **Administer** → **Settings**.
2. Enter *%_timeout* in the search box and click **`Search`**. The search should return four settings, including a description.
3. In the **`Value`** column, click the icon next to a number to edit it.
4. Enter the desired value in seconds, and click **Save**.

NOTE

Adjusting the *%_finish_timeout* values might help in case of low bandwith. Adjusting the *%_accept_timeout* values might help in case of high latency.

When a task is initialized, any back-end service that will be used in the task, such as Candlepin or Pulp, will be checked for correct functioning. If the check fails, you will receive an error similar to the following one:

```
There was an issue with the backend service candlepin: Connection refused
– connect(2).
```

If the back-end service checking feature turns out to be causing any trouble, it can be disabled as follows.

Procedure 1.2. To Disable Checking for Services:

1. Navigate to **Administer** → **Settings**.
2. Enter *check_services_before_actions* in the search box and click**`Search`**.
3. In the **`Value`** column, click the icon to edit the value.
4. From the drop-down menu, select **`false`**.
5. Click **Save**.

CHAPTER 2. SEARCHING AND BOOKMARKING

The Satellite web UI features powerful search functionality which is available on most pages of the web UI. It enables you to search all kinds of resources that Satellite Server manages. Searches accept both free text and syntax-based queries, which can be built using extensive input prediction. Search queries can be saved as bookmarks for future reuse.

2.1. BUILDING SEARCH QUERIES

As you start typing a search query, a list of valid options to complete the current part of the query appears. You can either select an option from the list and keep building the query using the prediction, or continue typing. To learn how free text is interpreted by the search engine, see Section 2.2, “Using Free Text Search”.

2.1.1. Query Syntax

```
parameter operator value
```

Available fields, resources to search, and the way the query is interpreted all depend on context, that is, the page where you perform the search. For example, the field "hostgroup" on the Hosts page is equivalent to the field "name" on the Host Groups page. The field type also determines available operators and accepted values. For a list of all operators, see Section 2.1.2, “Operators”. For descriptions of value formats, seeSection 2.1.3, “Values”.

2.1.2. Operators

All operators that can be used between *parameter* and *value* are listed in the following table. Other symbols and special characters that might appear in a prediction-built query, such as colons, do not have special meaning and are treated as free text.

Table 2.1. Comparison Operators Accepted by Search

Operator	Short Name	Description	Example
=	EQUALS	Accepts numerical, temporal, or text values. For text, exact case sensitive matches are returned.	`hostgroup = RHEL7`
!=	NOT EQUALS		
~	LIKE	Accepts text or temporal values. Returns case insensitive matches. Accepts the following wildcards: _ for a single character, % or * for any number of characters including zero. If no wildcard is specified, the string is treated as if surrounded by wildcards: %rhel7%	`hostgroup ~ rhel%`
!~	NOT LIKE		

Operator	Short Name	Description	Example
>	GREATER THAN	Accepts numerical or temporal values. For temporal values, the operator > is interpreted as "later than", and < as "earlier than". Both operators can be combined with EQUALS: >= <=	**`registered_at > 10-January-2017`** The search will return hosts that have been registered after the given date, that is, between 10th January 2017 and now. **`registered_at <= Yesterday`** The search will return hosts that have been registered yesterday or earlier.
<	LESS THAN		
^	IN	Compares an expression against a list of values, as in SQL. Returns matches that contain or not contain the values, respectively.	**`release_version !^ 7`**
!^	NOT IN		
HAS		Returns values that are present or not present, respectively.	**`has hostgroup`** On the Puppet Classes page, the search will return classes that are assigned to at least one host group. **`not has hostgroup`** On the Dashboard with an overview of hosts, the search will return all hosts that have no assigned host group.
NOT HAS			

Simple queries that follow the described syntax can be combined into more complex ones using logical operators AND, OR, and NOT. Alternative notations of the operators are also accepted:

Table 2.2. Logical Operators Accepted by Search

Operator	Alternative Notations			Example
and	&	&&	<whitespace>	**`class = motd AND environment ~ production`**
or	|	||		**`errata_status = errata_needed || errata_status = security_needed`**
not	–	!		**`hostgroup ~ rhel7 not status.failed`**

2.1.3. Values

Text Values

Text containing whitespaces must be enclosed in quotes. A whitespace is otherwise interpreted as the AND operator.

Examples:

```
hostgroup = "Web servers"
```

The search will return hosts with assigned host group named "Web servers".

```
hostgroup = Web servers
```

The search will return hosts in the host group Web with any field matching %servers%.

Temporal Values

Many date and time formats are accepted, including the following:

- "10 January 2017"
- "10 Jan 2017"
- 10-January-2017
- 10/January/2017
- "January 10, 2017"
- Today, Yesterday, and the like.

WARNING

Avoid ambiguous date formats, such as 02/10/2017 or 10-02-2017.

2.2. USING FREE TEXT SEARCH

When you enter free text, it will be searched for across multiple fields. For example, if you type "64", the search will return all hosts that have that number in their name, IP address, MAC address, and architecture.

NOTE

Multi-word queries must be enclosed in quotes, otherwise the whitespace is interpreted as the AND operator.

Because of searching across all fields, free text search results are not very accurate and searching can be slow, especially on a large number of hosts. For this reason, we recommend that you avoid free text and use more specific, syntax-based queries whenever possible.

2.3. BOOKMARKING

To save your search query, use the **Bookmark this search** item in the menu next to the **`Search`** button. Saved bookmarks will be available in the same menu under a name of your choice. Upon creation, a bookmark can be marked as public, which will make it visible to all users, or as private, which will make it visible only to the user who created it.

Each page has its own bookmarks. On some pages, there are default bookmarks available for the most common searches, for example, all active hosts, disabled hosts, or hosts that have recently reported errors. To view and manage all existing bookmarks, navigate to **Administer** → **Bookmarks**.

CHAPTER 3. USING CONTENT VIEWS

Content views are managed selections of content, which contain one or more repositories (yum, puppet, or containers) with optional filtering. These filters can be either inclusive or exclusive, and tailor a system view of content for life cycle management. They are used to customize content to be made available to client systems.

Keeping repositories for yum, Puppet, and containers in separate Content Views has the advantage that any updates to one repository only requires republishing the relevant Content View. You can use Composite Content Views to combine published Content Views for ease of management.

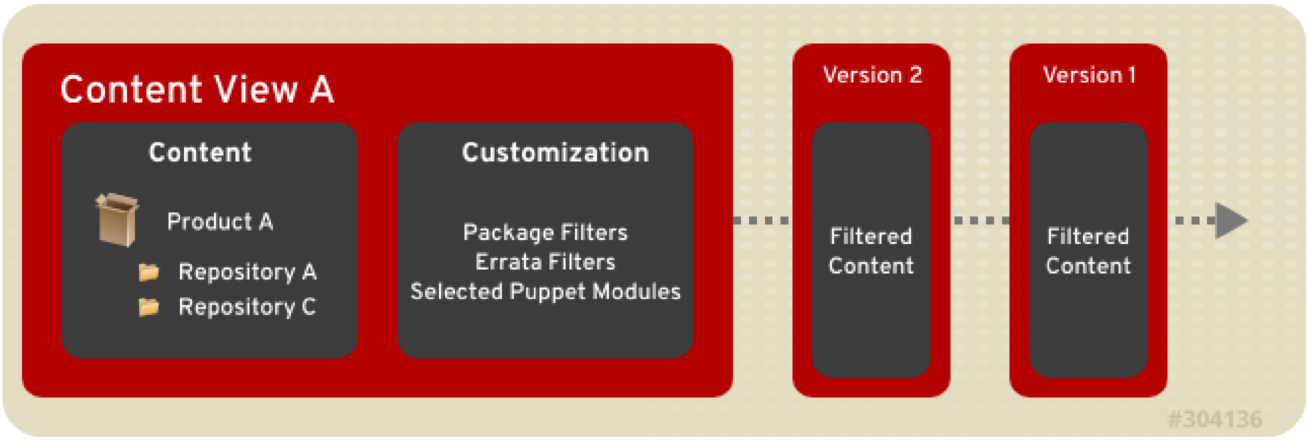


Figure 3.1. This diagram details the creation of new versions of a Content View. These content view versions are promoted along an environment path during the application life cycle.

Published content views are used with life cycle environments.

3.1. CREATING A CONTENT VIEW

A user with administrator privileges can create content views for use within the life cycle environments.

Procedure 3.1. To Create a Content View:

1. Log in as a Satellite administrator.
2. Click **Content** → **Content Views**.
3. Click **Create New View**.
4. Specify the **Name** of the content view. The **Label** field is automatically populated when the **Name** field is filled out. Optionally, provide a description of the content view.
5. Select the **Composite Content View** check box to combine a series of published content views into one and choose which content view.

NOTE

If you select **`Composite Content View`** it will override any filtering and allow you to choose a group of published content views and bundle those views into a composite one.

6. Click **`Save`**.

3.2. ADDING REPOSITORIES TO THE CONTENT VIEW

A repository provides storage for content. For example, a YUM repository, Puppet repository, or a Docker repository.

The **`Content View`** page contains a searchable list of content views. When searching for content views using the **`Search`** field, use an asterisk (*) to perform a partial string search.

For example, if searching for a content view named **`RHEL7_Base`**, entering **`RHEL7`** will not return any results, instead enter **`RHEL7*`**. Alternatively, **`*Base*`** also retrieves the content view **`RHEL7_Base`**.

Procedure 3.2. To Associate a Repository with a Content View:

1. Click **Content** → **Content Views** and choose the Content View to add repositories to.

2. Depending on the type of content you want to store:

   - To add a Yum repository, click **`Yum Content`** and select **Repositories** from the drop-down menu. From the submenu, click **Add**. For example, to be able to install Katello agent on your host, you need to enable the **`Satellite Tools`** repository.

   - To add a Puppet repository, click **`Puppet Modules`** and click **`Add New Module`**.

   - To add a Docker repository, click **`Docker Content`** and click **Add** in the submenu.

3. Select the repositories to add and click **`Add Repositories`**.

3.3. FILTERING CONTENT

Filters provide a mechanism to prevent packages from being promoted to subsequent environments. You can use package names or regular expressions in the filter to create the rules to blacklist packages. Then you can associate the filter to entire products or individual repositories within any product.

3.3.1. Creating a Filter

The following procedure shows how to create a filter for packages.

Procedure 3.3. To Create a Filter:

1. Navigate to **Content** → **Content Views** and select the Content View you want to filter.

2. Click **Yum Content** → **Filters** and click `New Filter`.

3. In the `Name` field, specify the name of the new filter and choose a content type from the `Content Type` drop-down menu. Choose whether the filter includes or excludes the selected content type by selecting the `Type` drop-down menu. Optionally, insert a description in the `Description` field.

4. Click `Save` to save your new filter.

3.3.2. Adding Content to a Filter

The following procedure shows how to add content to a package filter.

Procedure 3.4. To Add Content to a Filter:

1. Navigate to **Content** → **Content Views** and select the Content View you want to filter.

2. Click **Yum Content** → **Filters** and click the name of the filter you want to edit. Depending on the type of filter selected, perform the following actions:

   a. If the filter is made for packages, specify a package name on the `Packages` subtab, and select a **Detail** value from the drop-down menu. Click`Add` to add the package to the filter.

   b. If the filter is made for package groups, click the `Add` subtab, and choose the desired package group. Click `Add Package Group`.

   c. If the filter is made for errata, click the `Add` subtab. Select the errata type (`Security`, `Enhancement`, or `Bugfix`), and specify a start date and end date. Click `Add Errata`.

   d. If the filter is made for errata - date and type, on the `Erratum Date Range` subtab, select the errata type (`Security`, `Enhancement`, or `Bugfix`) and specify a start date and end date. Click `Save`.

3. On the `Affected Repositories` subtab, choose whether the filter will affect all or a subset of repositories. If you choose a subset of repositories, select the desired repositories and click `Update Repositories`.

4. Click `Publish New Version`. Insert a comment if desired, then click`Save`.

3.3.3. Removing Content from a Filter

The following procedure shows how to remove content from a package filter.

Procedure 3.5. To Remove Content from a Filter:

1. Navigate to **Content** → **Content Views** and select the Content View you want to filter.

2. Click **Yum Content** → **Filters** and click the name of the filter you want to edit. Depending on the type of filter selected, perform the following actions:

   a. If the filter is made for packages, click the **Packages** subtab and select the **Package Name** check box next to the package to be removed. Click **Remove Packages** to remove the package from the filter.

   b. If the filter is made for package groups, click the **List/Remove** subtab and select the **Name** check box next to the package group to be removed. Click **Remove Package Group** to remove the package group from the filter.

   c. If the filter is made for errata, click the **List/Remove** subtab select the **Errata ID** check box next to the errata to be removed. Click **Remove Errata** to remove the errata from the filter.

   d. If the filter is made for errata - date and type, on the **Erratum Date Range** subtab, check the errata type (**Security**, **Enhancement**, or **Bugfix**). Specify the start date and end date. Click **Save**.

3. On the **Affected Repositories** subtab, choose whether the filter will affect all or a subset of repositories. If you choose a subset of repositories, select the desired repositories and click **Update Repositories**.

4. Click **Publish New Version**. Insert a comment if desired, and click **Save**.

3.3.4. Removing a Filter

The following procedure shows how to remove a filter.

Procedure 3.6. To Remove a Filter:

1. Navigate to **Content** → **Content Views** and select the Content View you want to filter.

2. Click **Yum Content** → **Filters** and select the check box next to the name of the package filter you want to remove.

3. Click **Remove Filters**.

3.4. PUBLISHING A CONTENT VIEW

After a content view has been created, it needs to be published in order for it to be visible and usable by hosts. Before publishing the content view definition, make sure that the content view definition has the necessary products, repositories and filters.

Procedure 3.7. To Publish a Content View Definition:

1. Click **Content** → **Content Views**.

2. Click on the content view to be published.

3. Click **Publish New Version**.

4. Fill in a comment.

5. Click **Save**.

CHAPTER 4. VIEWING AND APPLYING ERRATA

Software packages in Red Hat products are subject to updates, referred to as *errata*, that are released at regular intervals as well as asynchronously. Red Hat Satellite provides tools to inspect and filter errata, allowing for precise update management. This way, you can select relevant updates and propagate them through content views to selected content hosts. See Chapter 3, *Using Content Views* for more information on content views.

IMPORTANT

Install the katello-agent package on the Satellite Server as described in Section 10.5.3, “Installing the Katello Agent”. This package provides the necessary services for errata management.

Before applying the latest updates, make sure you have correctly synchronized the Satellite content. Navigate to **Monitor** → **Content Dashboard** to see the overview of errata synchronization.

Errata contain advisories that describe the changes introduced by the update. There are three types of advisories (in order of importance):

- **Security Advisory** describes fixed security issues found in the package. The security impact of the issue can be *Low*, *Moderate*, *Important*, or *Critical*.
- **Bug Fix Advisory** describes bug fixes for the package.
- **Product Enhancement Advisory** describes enhancements and new features added to the package.

NOTE

Errata are labeled according to the most important advisory type they contain. Therefore, errata labeled as *Product Enhancement Advisory* can contain only enhancement updates, while *Bug Fix Advisory* errata can contain both bug fixes and enhancements, and *Security Advisory* can contain all three types.

In Red Hat Satellite, there are two keywords that describe an erratum's relationship to the available content hosts:

- **Applicable**: erratum applies to one or more content hosts, which means it updates packages present on the content host. Applicable errata are not yet accessible by the content host.
- **Installable**: erratum applies to one or more content hosts and it has been made available to the content host. Installable errata are present in the content host’s life cycle environment and content view, but are not yet installed. This way, errata can be installed by users who have permissions to manage content hosts, but are not entitled for errata management at higher levels.

4.1. INSPECTING AVAILABLE ERRATA

The following procedure describes how to view and filter the available errata and how to display metadata of the selected advisory.

Procedure 4.1. To Inspect Available Errata:

1. Navigate to **Content** → **Errata** to view the list of available errata.

2. Use the filtering tools at the top of the page to limit the number of displayed errata:

   - Select the repository to be inspected from the drop-down list. **All Repositories** is selected by default.

   - The **Applicable** check box is selected by default to view only errata applicable to the selected repository. Select the **Installable** check box to view only errata marked as installable.

   - To search the table of errata, type the query in the **Search** field in the form of:

     ```
     parameter operator value
     ```

     See Table 4.1, “Parameters Available for Errata Search” for the list of parameters available for search. Find the list of applicable operators in Supported Operators for Granular Search in the *Server Administration Guide*. Automatic suggestion works as you type. You can also combine queries with the use of *and* and *or* operators. For example, to display only security advisories related to the kernel package, type:

     ```
     type = security and package_name = kernel
     ```

     Press **Enter** to start the search.

3. Click the **Errata ID** of the erratum you want to inspect:

   - The **Details** tab contains the description of the updated package as well as documentation of important fixes and enhancements provided by the update.

   - On the **Content Hosts** tab, you can apply the erratum to selected content hosts as described in Section 4.2, “Applying Errata to Content Hosts”.

   - The **Repositories** tab lists repositories that already contain the erratum. You can filter repositories by the environment and content view, and search for them by the repository name.

Table 4.1. Parameters Available for Errata Search

Parameter	Description	Example
bug	Search by the Bugzilla number.	**bug = 1172165**
cve	Search by the CVE number.	**cve = CVE-2015-0235**
id	Search by the errata ID. The auto-suggest system displays a list of available IDs as you type.	**id = RHBA-2014:2004**

Parameter	Description	Example
issued	Search by the issue date. You can specify the exact date, like "*Feb16,2015*", or use keywords, for example "*Yesterday*", or "*1 hour ago*". The time range can be specified with the use of the "<" and ">" operators.	**`issued < "Jan 12,2015"`**
package	Search by the full package build name. The auto-suggest system displays a list of available packages as you type.	**`package = glib2-2.22.5-6.el6.i686`**
package_name	Search by the package name. The auto-suggest system displays a list of available packages as you type.	**`package_name = glib2`**
severity	Search by the severity of the issue fixed by the security update. Specify *Critical*, *Important*, or *Moderate.*	**`severity = Critical`**
title	Search by the advisory title.	**`title ~ openssl`**
type	Search by the advisory type. Specify *security*, *bugfix*, or *enhancement*.	**`type = bugfix`**
updated	Search by the date of the last update. You can use the same formats as with the *issued* parameter.	**`updated = "6 days ago"`**

4.2. APPLYING ERRATA TO CONTENT HOSTS

The following procedures show how to apply one or more errata to content hosts.

Procedure 4.2. To Apply a Single Erratum to Content Hosts:

1. Navigate to **Content** → **Errata** to view the list of available errata.
2. Click the **`Errata ID`** of the erratum you want to apply.
3. On the **`Content Hosts`** tab, select one or more content hosts to be updated. You can filter the available content hosts by the environment, and search for them by name. If you select the check box at the top of the page, only the content hosts that already have the installable erratum in their life cycle environment are displayed.
4. Click **`Apply to Hosts`**.
   - If the erratum is *applicable*, a new minor version of the content view is created. If you select **`Apply Errata to Content Hosts Immediately after publishing`**, Satellite will automatically install the erratum on the content host when promoting the updated content view. Otherwise, the erratum will be made

available for installation on the content host. Installable errata can be applied later using the same procedure, or manually per content host as described in Procedure 4.4, “To Apply Installable Errata to a Content Host:”.

- If the erratum is *installable*, which means it is already present in the selected content host's life cycle environment but is not installed yet, no new content view version is created.

5. Click **`Confirm`**.

Procedure 4.3. To Apply Multiple Errata to Content Hosts:

1. Navigate to **Content** → **Errata** to view the list of available errata.
2. Select errata you want to apply by selecting the check box to the left of the **`Errata ID`** field.
3. Click **`Apply Errata`** to apply all selected errata.
4. Select one or more content hosts to be updated. You can filter the available content hosts by the environment, and search for them by name. If you select the check box at the top of the page, only content hosts that already have the installable errata in their life cycle environment are displayed.
5. Click **`Next`**. If some of the selected errata are*applicable*, a new minor version of the content view is created. If you select **`Apply Errata to Content Hosts Immediately after publishing`**, Satellite will automatically install errata on the content host when promoting the updated content view. If only installable errata are selected, they are installed without creating a new content view version.

If the content host's life cycle environment contains installable errata, you can install them from the **`Content Hosts`** page as described inProcedure 4.4, “To Apply Installable Errata to a Content Host:” This way, errata can be applied by users who have permissions to manage content hosts, but are not entitled for errata management at higher levels. Similarly, you can apply installable errata to host collections as described in Section 13.3.2, “Adding Errata to a Host Collection”.

Procedure 4.4. To Apply Installable Errata to a Content Host:

1. Navigate to **Hosts** → **Content Hosts**.
2. Click the name of the content host you want to manage.
3. On the **`Errata`** tab, select errata you want to install.
4. Click **`Apply Selected`** to install the selected updates.

4.3. SUBSCRIBING TO ERRATA NOTIFICATIONS

You can configure email notifications for Satellite users as described in Configuring Email Notifications in the *Server Administration Guide*. Users can receive a summary of applicable and installable errata, notifications on content view promotion or after synchronizing a repository.

CHAPTER 5. SECURITY COMPLIANCE MANAGEMENT

Security compliance management is the ongoing process of defining security policies, auditing for compliance with those policies and resolving instances of non-compliance. Once a security policy is defined, an audit is conducted to verify compliance with the policy. Any non-compliance is managed according to the organization's configuration management policies. Security policies vary in their scope, from being host-specific to industry-wide, so there is a need for flexibility in their definition.

The Security Content Automation Protocol (SCAP) enables the definition of security configuration policies. For example, a security policy might specify that for hosts running Red Hat Enterprise Linux, login via SSH is not permitted for the **`root`** account. In Satellite 6, tools provided by the *OpenSCAP* project are used to implement security compliance auditing. For more information about OpenSCAP see the Red Hat Enterprise Linux 7 Security Guide. The Satellite web UI enables scheduled compliance auditing and reporting on all hosts under management by Red Hat Satellite.

The following specifications are supported by OpenSCAP:

- XCCDF: The Extensible Configuration Checklist Description Format (version 1.2)
- OVAL: Open Vulnerability and Assessment Language (version 5.11)
- Asset Identification (version 1.1)
- ARF: Asset Reporting Format (version 1.1)
- CCE: Common Configuration Enumeration (version 5.0)
- CPE: Common Platform Enumeration (version 2.3)
- CVE: Common Vulnerabilities and Exposures
- CVSS: Common Vulnerability Scoring System (version 2.0)

5.1. WHAT IS SCAP

5.1.1. SCAP Content

SCAP content is a datastream format containing the configuration and security baseline against which hosts are checked. Checklists are described in the *extensible checklist configuration description format* (XCCDF) and vulnerabilities in the*open vulnerability and assessment language* (OVAL). Checklist items, also known as*rules* express the desired configuration of a system item. For example, you may specify that no one can log in to a host over SSH using the **`root`** user account. Rules can be grouped into one or more*profiles*, allowing multiple profiles to share a rule. SCAP content consists of both rules and profiles.

You can either create SCAP content or obtain it from a vendor. Supported profiles are provided for Red Hat Enterprise Linux in the scap-security-guide package. The creation of SCAP content is outside the scope of this guide, but see the Red Hat Enterprise Linux 7 Security Guide or Red Hat Enterprise Linux 6 Security Guide for information on how to download, deploy, modify, and create your own content. The SCAP content provided with Red Hat Enterprise Linux is compliant with SCAP specification 1.2.

The default SCAP content provided with the OpenSCAP components of Satellite 6 depends on the version of Red Hat Enterprise Linux:

- On Red Hat Enterprise Linux 6, content for Red Hat Enterprise Linux 6 is installed.
- On Red Hat Enterprise Linux 7, content for both Red Hat Enterprise Linux 6 and Red Hat Enterprise Linux 7 is installed.

5.1.2. XCCDF Profile

An XCCDF profile is a checklist against which a host or host group is evaluated. Profiles are generally created to verify compliance with a standard, whether that be an industry standard or a custom standard.

To list all available profiles, open the Satellite web UI, navigate to **Hosts** → **Policies**, select **Edit** from the drop-down list next to the policy of interest and select the **`SCAP Content`** tab. Select the **`SCAP Content`** of interest and browse the available profiles in the **`XCCDF Profile`** drop-down list.

The profiles provided with Satellite 6 are obtained from the OpenSCAP project.

5.1.3. Compliance Policy

A scheduled audit, also known as a *compliance policy*, is a scheduled task which checks the specified hosts for compliance against an XCCDF profile. The schedule on which a scan is run is specified by the Satellite Server but the scan itself occurs on the host. When the scan is complete, an *Asset Reporting File* (ARF) is generated in XML format and uploaded to the Satellite Server. You can see the results of the scan in the compliance policy dashboard. A compliance policy does not make any changes to the scanned hosts. The OpenSCAP content includes several profiles and their associated rules but no policies are included by default.

5.2. INSTALLATION

5.2.1. Install OpenSCAP Packages

Procedure 5.1. Installing OpenSCAP Packages

Install the OpenSCAP plugin and content on the Satellite Server and all external Capsule Servers.

1. On the Satellite Server, install the OpenSCAP plug-in and content.

   a.
   ```
   # satellite-installer --enable-foreman-plugin-openscap
   ```
   Successful installation is indicated by a progress indicator, and the word **`Success!`**. The OpenSCAP plugin adds to the Satellite web UI a **`Compliance`** section, under the **Hosts** menu, containing the following pages:

   - **Policies**
   - **SCAP Contents**
   - **Reports**

   b.
   ```
   # yum install puppet-foreman_scap_client
   ```

2. On all external Capsule Servers, install the OpenSCAP plug-in and content.

   NOTE

   If OpenSCAP functionality is to be enabled on a Capsule Server, Puppet must already have been enabled on that server.

   ```
   # satellite-installer --enable-foreman-proxy-plugin-openscap
   ```

   Successful installation is indicated by a progress indicator, and the word **`Success!`**. This provides the Puppet classes required to set up hosts to perform OpenSCAP scans and creates the Cron jobs for automated compliance scanning.

3. On external Capsule Servers with the Puppet master role, install the OpenSCAP client.

   ```
   # yum install puppet-foreman_scap_client
   ```

   To identify the relevant external Capsule Servers, open the Satellite web UI, navigate to **Infrastructure** → **Capsules** and identify those external Capsule Servers with **`Puppet`** listed in the **`Features`** column.

5.2.2. Loading Default OpenSCAP Content

Procedure 5.2. Load the Default OpenSCAP Content

- Load the OpenSCAP content on the Satellite Server.

  ```
  # foreman-rake foreman_openscap:bulk_upload:default
  ```

5.2.3. Importing OpenSCAP Puppet Modules

Procedure 5.3. Import OpenSCAP Puppet Modules

1. OpenSCAP requires a Puppet environment, but by default they are only created for Content Views which contain Puppet modules. To list available Puppet environments, open the Satellite web UI and navigate to **Configure** → **Environments**.

   If there are no Puppet environments, open a CLI session on the Satellite Server and create a directory for the **`production`** Puppet environment.

   ```
   # mkdir -p /etc/puppet/environments/production/modules
   ```

2. Import the OpenSCAP content into selected Puppet environments. Each host which is to be audited with OpenSCAP must be associated with a Puppet environment.

   a. In the Satellite web UI, select from the context menu **`Any Organization`** and **`Any Location`**.

   b. Navigate to **Configure** → **Environments**.

c. Click **Import**, then **Import from *satellite.example.com***.

d. For each Puppet environment associated with hosts to be audited using OpenSCAP, select the check box, then click **Update**. If no other Puppet environment exists, select the **production** environment.

   The **foreman_scap_client** Puppet module, amongst others, will be added to the selected environments.

e. Verify that the **foreman_scap_client** Puppet module has been added.

   Navigate to **Configure** → **Environments**, then click **Classes** in the Puppet environment's row. The procedure has been successful if the **foreman_scap_client** Puppet class is listed.

5.2.4. Uploading Extra SCAP Content

You can upload extra SCAP content into the Satellite Server, either content created by yourself or obtained elsewhere. SCAP content must be imported into the Satellite Server before being applied in a policy. For example, the scap-security-guide RPM package available in the Red Hat Enterprise Linux 7.2 repositories includes a profile for the *Payment Card Industry Data Security Standard* (PCI-DSS) version 3. You can upload this content into a Satellite Server even if it is not running Red Hat Enterprise Linux 7.2 as the content is not specific to an operating system version.

Procedure 5.4. Upload Extra SCAP Content

1. Log in to the Satellite web UI.

2. Navigate to **Hosts** → **SCAP contents** and click **Upload New SCAP Content**.

3. Enter a title in the **Title** text box. For example: **RHEL 7.2 SCAP Content**.

4. Click **Choose file**, navigate to the location containing the SCAP content file and select **Open**.

5. Click **Submit**.

If the SCAP content file is loaded successfully, a message similar to **Successfully created RHEL 7.2 SCAP Content** will be shown and the list of**SCAP Contents** will include the new title.

5.3. MANAGING COMPLIANCE POLICIES

5.3.1. Creating a Policy

Follow these steps to create a compliance policy, which specifies the SCAP content and profile to be applied to a location and either a host or host group at a specified time.

Prerequisites

- Section 5.2.1, “Install OpenSCAP Packages”.

- Section 5.2.3, “Importing OpenSCAP Puppet Modules”.

Procedure 5.5. To Create a Policy:

1. In the Satellite web UI, navigate to **Hosts** → **Policies**, click **`New Policy`** and follow the wizard's steps.

2. Enter a name for this policy, a description (optional), then click **`Next`**.

3. Select the SCAP Content and XCCDF Profile to be applied, then click **`Next`**.

4. Specify the scheduled time when the policy is to be applied, then click **`Next`**.

   Select **Weekly**, **Monthly**, or **Custom** from the **`Period`** drop-down list.

   - If you select **Weekly**, also select the desired day of the week from the **Weekday** drop-down list.

   - If you select **Monthly**, also specify the desired day of the month in the **`Day of month`** field.

   - If you select **Custom**, enter a valid Cron expression in the **`Cron line`** field.

   The **Custom** option allows for greater flexibility in the policy's schedule than either the **Weekly** or **Monthly** options.

5. Select the locations to which the policy is to be applied, then click **`Next`**.

6. Select the organizations to which the policy is to be applied, then click **`Next`**.

7. Select the host groups to which the policy is to be applied, then click **`Next`**.

8. Click **`Submit`**.

When the Puppet agent runs on the hosts which belong to the selected host group, or hosts to which the policy has been applied, the OpenSCAP client will be installed and a Cron job added with the policy's specified schedule. The **`SCAP Content`** tab provides the name of the SCAP content file which will be distributed to the directory **`/var/lib/openscap/content/`** on all target hosts.

5.3.2. Viewing a Policy

Follow these steps to preview the rules which will be applied by specific OpenSCAP content and profile combination. This is useful when planning policies.

Procedure 5.6. To View a Policy:

1. In the Satellite web UI, navigate to **Hosts** → **Policies**.

2. Click **Show Guide**.

5.3.3. Editing a Policy

Follow these steps to edit a policy. An edited policy is applied to the host when its Puppet agent next checks with the Satellite Server for updates. By default this occurs every 30 minutes.

Procedure 5.7. To Edit a Policy:

1. In the Satellite web UI, navigate to **Hosts** → **Policies**.
2. From the drop-down list to the right of the policy's name, select **Edit**.
3. Edit the necessary attributes.
4. Click **`Submit`**.

An edited policy is applied to the host when its Puppet agent next checks with the Satellite Server for updates. By default this occurs every 30 minutes.

5.3.4. Deleting a Policy

Follow these steps to delete an existing policy.

1. In the Satellite web UI, navigate to **Hosts** → **Policies**.
2. From the drop-down list to the right of the policy's name, select **Delete**.
3. Click **`OK`** in the confirmation message.

5.3.5. Adding a Policy to a Host

Follow these steps to add a policy to one or more hosts.

1. In the Satellite web UI, navigate to **Hosts** → **All hosts**.
2. Select the host or hosts to which you want to add the policy.
3. Click **`Select Action`**.
4. In the new panel that opens, select the appropriate policy from the list of available policies and click **`Submit`**.

5.4. MONITORING COMPLIANCE

Monitoring compliance is an ongoing task of ensuring that audits are conducted and that non-compliance is identified. Red Hat Satellite 6 enables centralized compliance monitoring and management. Hosts under Satellite management are checked for compliance according to your custom schedule and details are collated by the Satellite Server. A compliance dashboard provides an overview of hosts' compliance and the ability to view details for each host within the scope of that policy. Compliance reports provide a detailed analysis of each host's compliance with the applicable policy. With this information you can evaluate the risks presented by each host and better manage the resources required to bring hosts into compliance.

Common objectives when monitoring compliance using SCAP include the following:

- Verifying policy compliance.
- Detecting changes in compliance.

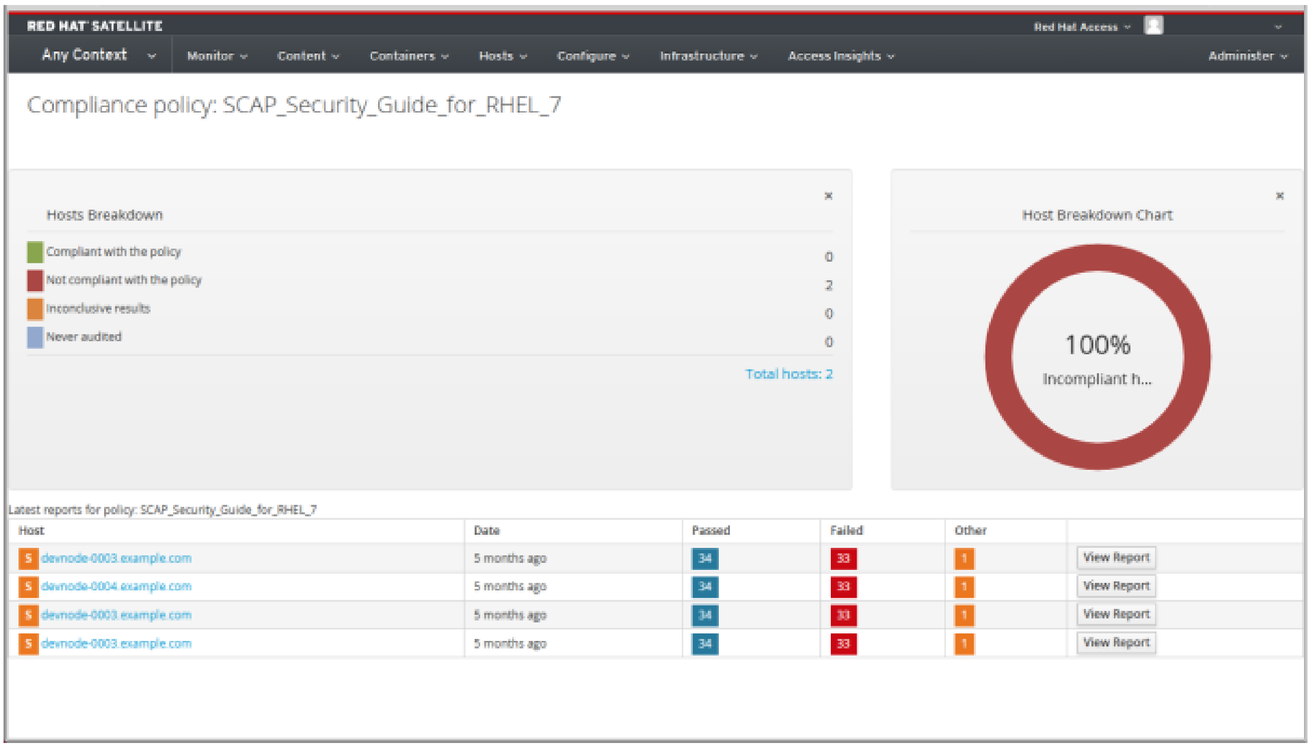
The Satellite web UI provides all the necessary information to achieve these objectives. Verify policy compliance with the compliance policy dashboard. Detect changes in policy compliance by either viewing a compliance report's history or subscribing to notification of changes by email.

5.4.1. Compliance Policy Dashboard

The compliance policy dashboard provides an overview of hosts' compliance with a policy. To view a compliance policy's dashboard, open the Satellite web UI and navigate to **Hosts** → **Policies**, then click the policy's name. The dashboard provides the following information:

- A ring chart illustrating a high-level view of hosts' compliance with the policy.
- A statistical breakdown of hosts' compliance with the policy, in tabular format.
- Links to the policy's latest report for each host.

The dashboard view provides a statistical summary of hosts' compliance and is a good starting point for compliance management. For all hosts which were evaluated as non-compliant, the **`Failed`** statistic provides a useful metric for prioritizing compliance effort. Those hosts detected as **`Never audited`** should also be a priority, since their status is unknown.



Compliance policy: SCAP_Security_Guide_for_RHEL_7

Hosts Breakdown

🟩 Compliant with the policy	0
🟥 Not compliant with the policy	2
🟧 Inconclusive results	0
🟦 Never audited	0

Total hosts: 2

Latest reports for policy: SCAP_Security_Guide_for_RHEL_7

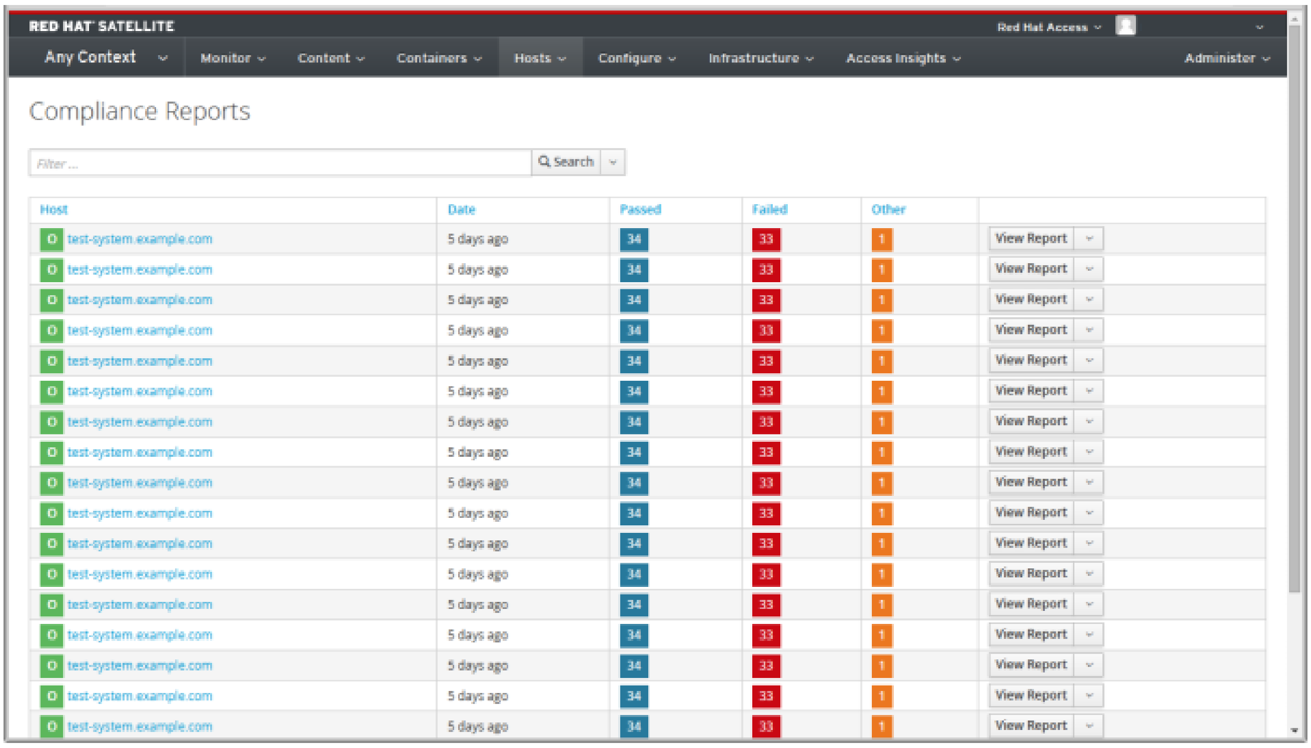
Host	Date	Passed	Failed	Other	
S devnode-0003.example.com	5 months ago	34	33	1	View Report
S devnode-0004.example.com	5 months ago	34	33	1	View Report
S devnode-0003.example.com	5 months ago	34	33	1	View Report
S devnode-0003.example.com	5 months ago	34	33	1	View Report

Figure 5.1. Compliance Policy Dashboard

5.4.2. Compliance Reports Overview

A compliance report is the output of a policy run against a host. To list all available compliance reports, open the Satellite web UI and navigate **Hosts** → **Reports**. For each report the total number of rules passed or failed per policy are listed. By default, all reports are listed in descending date order. To change the sort order, click on the label of the column by which you want it sorted. Click on the same label again to change to either descending or ascending order. From the **`Compliance Reports`** page, click **`View Report`** to view an individual report or use the **`Search`** field to narrow the list of reports to a host or a subset of hosts. To delete a compliance report, select **Delete** from the drop-down list beside **`View Report`**.

When managing the policy compliance of hosts, it is useful to monitor compliance changes over time. Satellite 6 provides the information necessary to monitor compliance changes manually, also notification via email. Use the **`Search`** field to narrow the list of reports to one or more hosts and evaluate changes manually, as detailed in Section 5.4.3, “Searching Compliance Reports”. Subscribe to compliance change email messages, as detailed in Configuring Email Notifications in the *Server Administration Guide*.



Host	Date	Passed	Failed	Other	
O test-system.example.com	5 days ago	34	33	1	View Report
O test-system.example.com	5 days ago	34	33	1	View Report
O test-system.example.com	5 days ago	34	33	1	View Report
O test-system.example.com	5 days ago	34	33	1	View Report
O test-system.example.com	5 days ago	34	33	1	View Report
O test-system.example.com	5 days ago	34	33	1	View Report
O test-system.example.com	5 days ago	34	33	1	View Report
O test-system.example.com	5 days ago	34	33	1	View Report
O test-system.example.com	5 days ago	34	33	1	View Report
O test-system.example.com	5 days ago	34	33	1	View Report
O test-system.example.com	5 days ago	34	33	1	View Report
O test-system.example.com	5 days ago	34	33	1	View Report
O test-system.example.com	5 days ago	34	33	1	View Report
O test-system.example.com	5 days ago	34	33	1	View Report
O test-system.example.com	5 days ago	34	33	1	View Report
O test-system.example.com	5 days ago	34	33	1	View Report
O test-system.example.com	5 days ago	34	33	1	View Report

Figure 5.2. Compliance Reports Overview

5.4.3. Searching Compliance Reports

The Compliance Reports search field allows you to narrow the list of reports. Narrowing your attention on a subset of hosts allows you to focus resources where they are most needed. To apply a filter, enter search criteria in the **Search** field and either press Enter or click **Search**. The search performed is case-insensitive. Click on the empty **Search** field to see a list of available search parameters.

See Supported Operators for Granular Search in the *Server Administration Guide* for details of all available search operators. You can create complex queries with the logical operators: **and**, **not** and **has**. Regular expressions are not valid search criteria, however multiple fields can be used in a single search expression.

Logical Operators

- **not**: Negates an expression.
- **has**: Object must have a specified property.
- **and**: Combines search criteria.

Search Use Cases

The following search criteria finds all compliance reports for which more than five rules failed.

```
failed > 5
```

The following search criteria finds all compliance reports created after November 5, 2015, for hosts whose host name contains the string **`prod-`**.

```
host ~ prod- AND date > "Nov 5, 2015"
```

The following search criteria finds all reports generated by the compliance_policy **`rhel7_audit`** from an hour ago.

```
"1 hour ago" AND compliance_policy = date = "1 hour ago" AND
compliance_policy = rhel7_audit
```

To again list *all* available compliance reports, delete the **`Search`** criteria and press Enter or click **`Search`**.

Bookmarking Your Searches

You can bookmark a search, allowing you to apply the same search criteria again.

Procedure 5.8. To Bookmark a Search:

1. Apply your search criteria.
2. From the **`Search`** list select **`Bookmark this search`**.
3. Complete the **`Name`** field.

   If you want the bookmark available to other users of this Satellite instance, select the **`Public`** check box.
4. Click **`Submit`**.

To use a bookmark, navigate to **Hosts** → **Reports**, click the drop-down item beside the **`Search`** button and click the bookmark.

5.4.4. Viewing a Compliance Report

Navigate to **Hosts** → **Reports** and click **`View Report`** in the row of the specific host.

A compliance report consists of the following sections:

- Introduction
- Evaluation Characteristics
- Compliance and Scoring
- Rule Overview

5.4.4.1. Evaluation Characteristics

This section provides details about an evaluation against a specific profile, including the host that was evaluated, the profile used in the evaluation, and when the evaluation started and finished. For reference, the IPv4, IPv6, and MAC addresses of the host are also listed.

Evaluation Characteristics

`Target machine`

The fully-qualified domain name (FQDN) of the evaluated host. Example: **`test-system.example.com`**.

`Benchmark URL`

The URL of the SCAP content against which the host was evaluated. Example: **`/var/lib/openscap/content/1fbdc87d24db51ca184419a2b6f`**.

`Benchmark ID`

The identifier of the benchmark against which the host was evaluated. A benchmark is a set of profiles. Example: **`xccdf_org.ssgproject.content_benchmark_RHEL_7`**.

`Profile ID`

The identifier of the profile against which the host was evaluated. Example: **`xccdf_org.ssgproject_content_profile_rht-ccp`**.

`Started at`

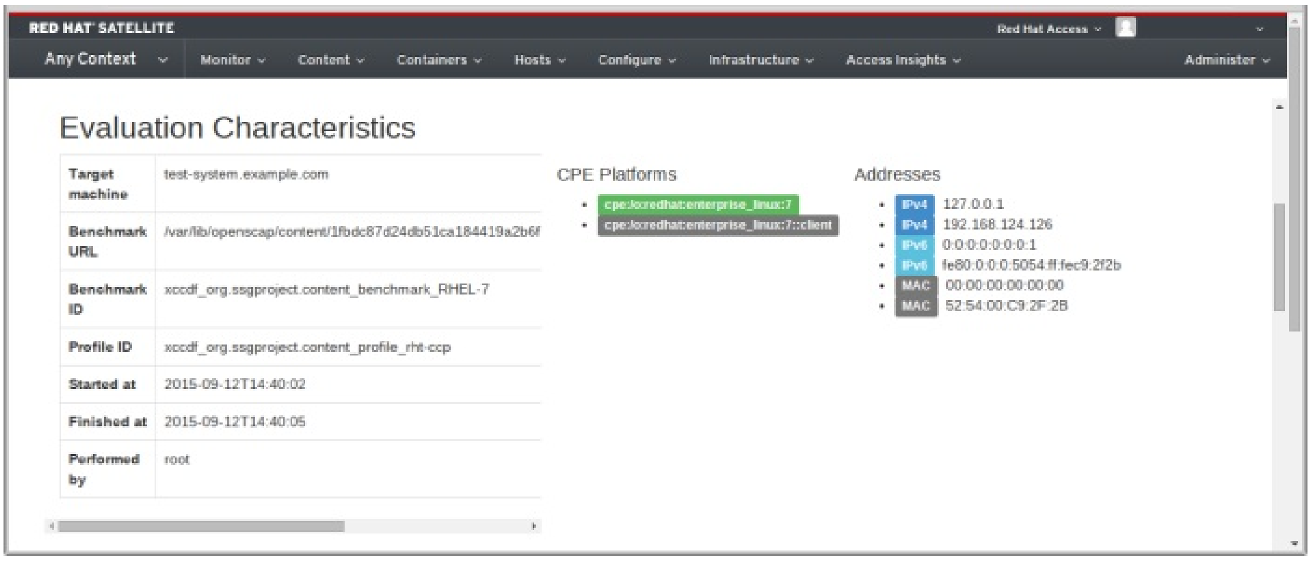
The date and time at which the evaluation started, in ISO 8601 format. Example: **`2015-09-12T14:40:02`**.

`Finished at`

The date and time at which the evaluation finished, in ISO 8601 format. Example: **`2015-09-12T14:40:05`**.

`Performed by`

The local account name under which the evaluation was performed on the host. Example: **`root`**.



Evaluation Characteristics

Target machine	test-system.example.com
Benchmark URL	/var/lib/openscap/content/1fbdc87d24db51ca184419a2b6f
Benchmark ID	xccdf_org.ssgproject.content_benchmark_RHEL-7
Profile ID	xccdf_org.ssgproject.content_profile_rht-ccp
Started at	2015-09-12T14:40:02
Finished at	2015-09-12T14:40:05
Performed by	root

CPE Platforms

- cpe:/o:redhat:enterprise_linux:7
- cpe:/o:redhat:enterprise_linux:7::client

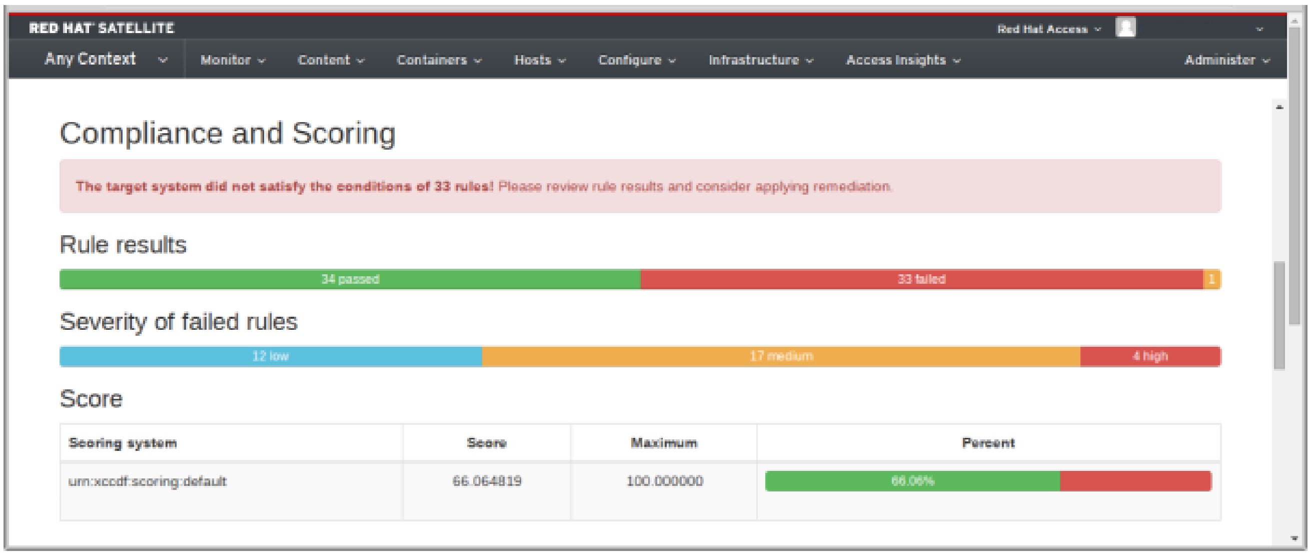
Addresses

- IPv4 127.0.0.1
- IPv4 192.168.124.126
- IPv6 0:0:0:0:0:0:0:1
- IPv6 fe80:0:0:0:5054:ff:fec9:2f2b
- MAC 00:00:00:00:00:00
- MAC 52:54:00:C9:2F:2B

Figure 5.3. Evaluation Characteristics

5.4.4.2. Compliance and Scoring

This section provides an overview of whether or not the host is in compliance with the profile’s rules, a breakdown of compliance failures by severity, and an overall compliance score as a percentage. If compliance with a rule was not checked, this is categorized in the **`Rule results`** as **`Other`**.



Compliance and Scoring

The target system did not satisfy the conditions of 33 rules! Please review rule results and consider applying remediation.

Rule results

Severity of failed rules

Score

Scoring system	Score	Maximum	Percent
urn:xccdf:scoring:default	66.064819	100.000000	66.06%

Figure 5.4. Compliance and Scoring

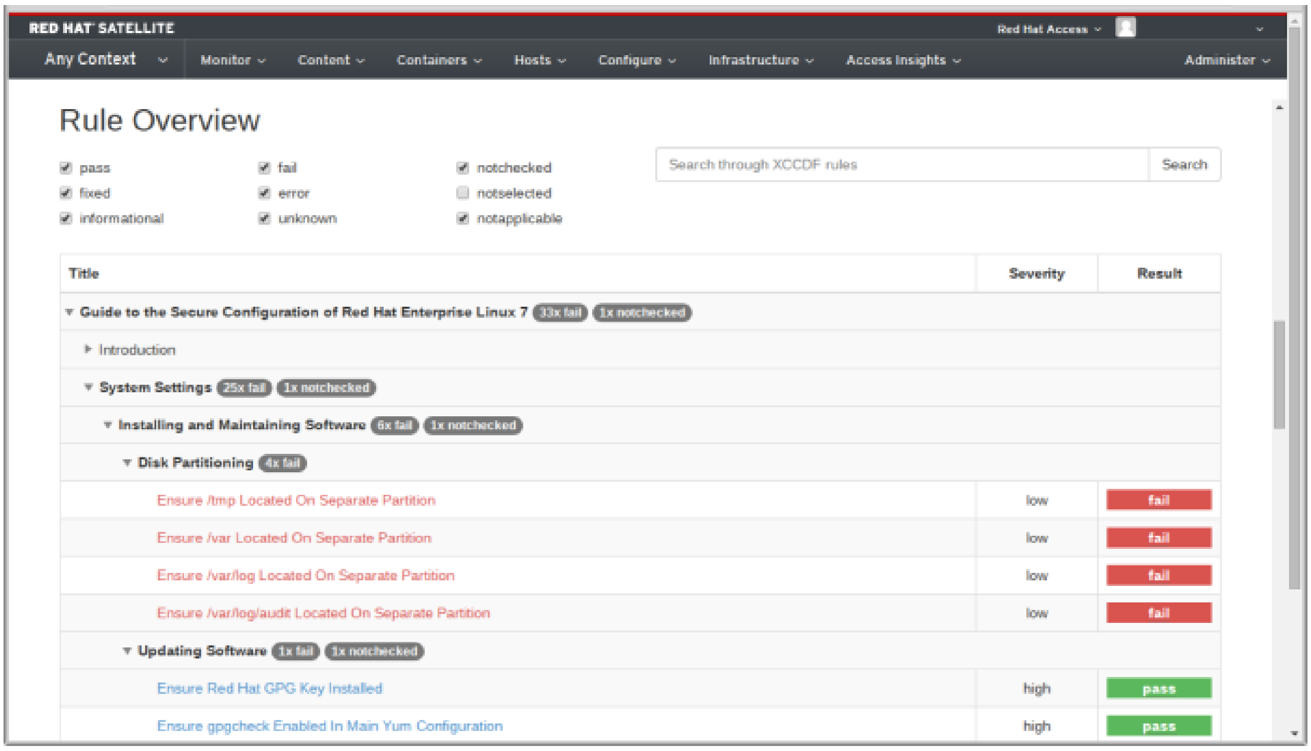
5.4.4.3. Rule Overview

This section provides details of every rule and the compliance result, with the rules presented in a hierarchical layout.

Select or clear the check boxes to narrow the list of rules included in the compliance report. For example, if the focus of your review is any non-compliance, clear the **`pass`** and **`informational`** check boxes.

To search all rules, enter a criterion in the **`Search`** field. The search is dynamically applied as you type. The **`Search`** field only accepts a single plain-text search term and it is applied as a case-insensitive search. When you perform a search, only those rules whose descriptions match the search criterion will be listed. To remove the search filter, delete the search criterion.

For an explanation of each result, hover the cursor over the status shown in the **`Result`** column.



Rule Overview

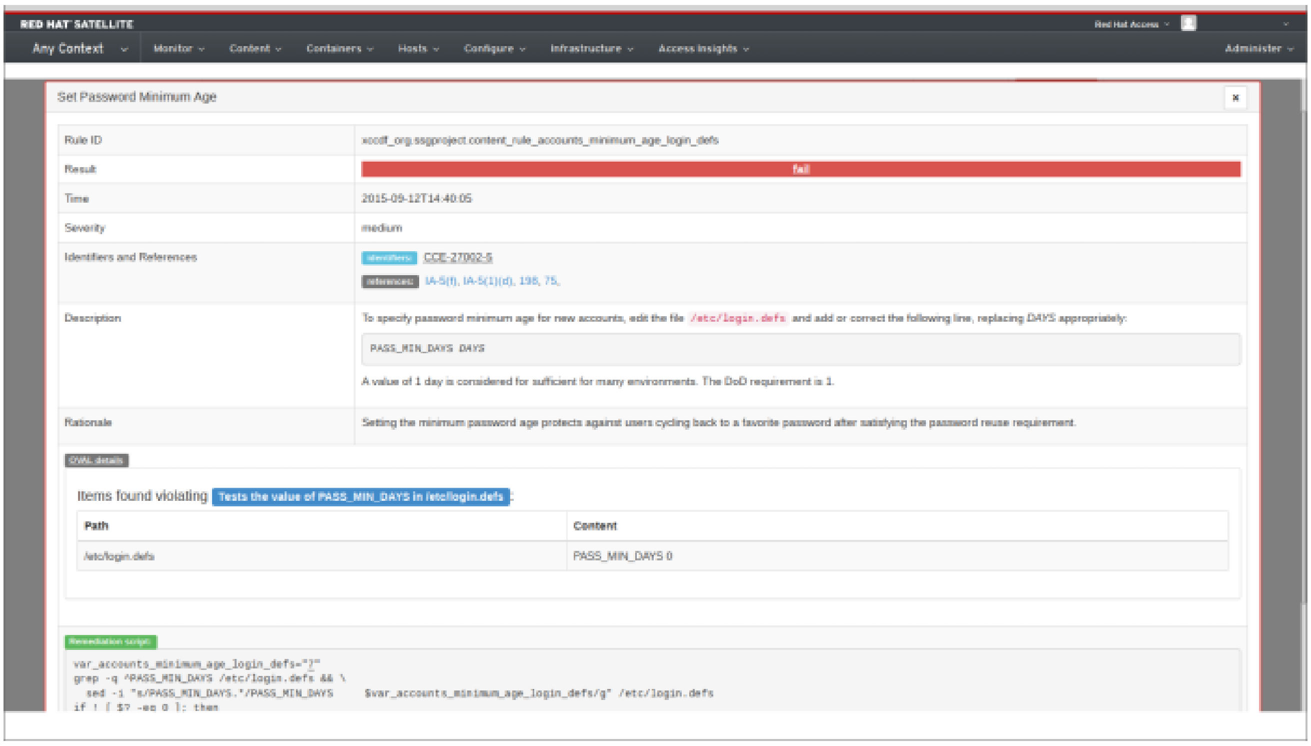
- ✓ pass
- ✓ fail
- ✓ notchecked
- ✓ fixed
- ✓ error
- notselected
- ✓ informational
- ✓ unknown
- ✓ notapplicable

Title	Severity	Result
▾ Guide to the Secure Configuration of Red Hat Enterprise Linux 7 33x fail 1x notchecked		
▸ Introduction		
▾ System Settings 25x fail 1x notchecked		
▾ Installing and Maintaining Software 6x fail 1x notchecked		
▾ Disk Partitioning 4x fail		
Ensure /tmp Located On Separate Partition	low	fail
Ensure /var Located On Separate Partition	low	fail
Ensure /var/log Located On Separate Partition	low	fail
Ensure /var/log/audit Located On Separate Partition	low	fail
▾ Updating Software 1x fail 1x notchecked		
Ensure Red Hat GPG Key Installed	high	pass
Ensure gpgcheck Enabled In Main Yum Configuration	high	pass

Figure 5.5. Rule Overview

5.4.4.4. Examining Rule Results

To determine why a host failed compliance on a rule, click on the rule's title. The window which then opens provides further details, including: a description of the rule (with instructions for bringing the host into compliance if available), the rationale for the rule, and in some cases a remediation script.



Set Password Minimum Age

Rule ID	xccdf_org.ssgproject.content_rule_accounts_minimum_age_login_defs
Result	fail
Time	2015-09-12T14:40:05
Severity	medium
Identifiers and References	identifiers: CCE-27002-5 references: IA-5(f), IA-5(1)(d), 198, 75,
Description	To specify password minimum age for new accounts, edit the file `/etc/login.defs` and add or correct the following line, replacing *DAYS* appropriately: `PASS_MIN_DAYS DAYS` A value of 1 day is considered for sufficient for many environments. The DoD requirement is 1.
Rationale	Setting the minimum password age protects against users cycling back to a favorite password after satisfying the password reuse requirement.

OVAL details

Items found violating Tests the value of PASS_MIN_DAYS in /etc/login.defs:

Path	Content
/etc/login.defs	PASS_MIN_DAYS 0

Remediation script:

```
var_accounts_minimum_age_login_defs="7"
grep -q ^PASS_MIN_DAYS /etc/login.defs && \
  sed -i "s/PASS_MIN_DAYS.*/PASS_MIN_DAYS     $var_accounts_minimum_age_login_defs/g" /etc/login.defs
if ! [ $? -eq 0 ]; then
```

Figure 5.6. Rule Evaluation Result

WARNING

Do not implement any of the recommended remedial actions or scripts without first testing them in a non-production environment.

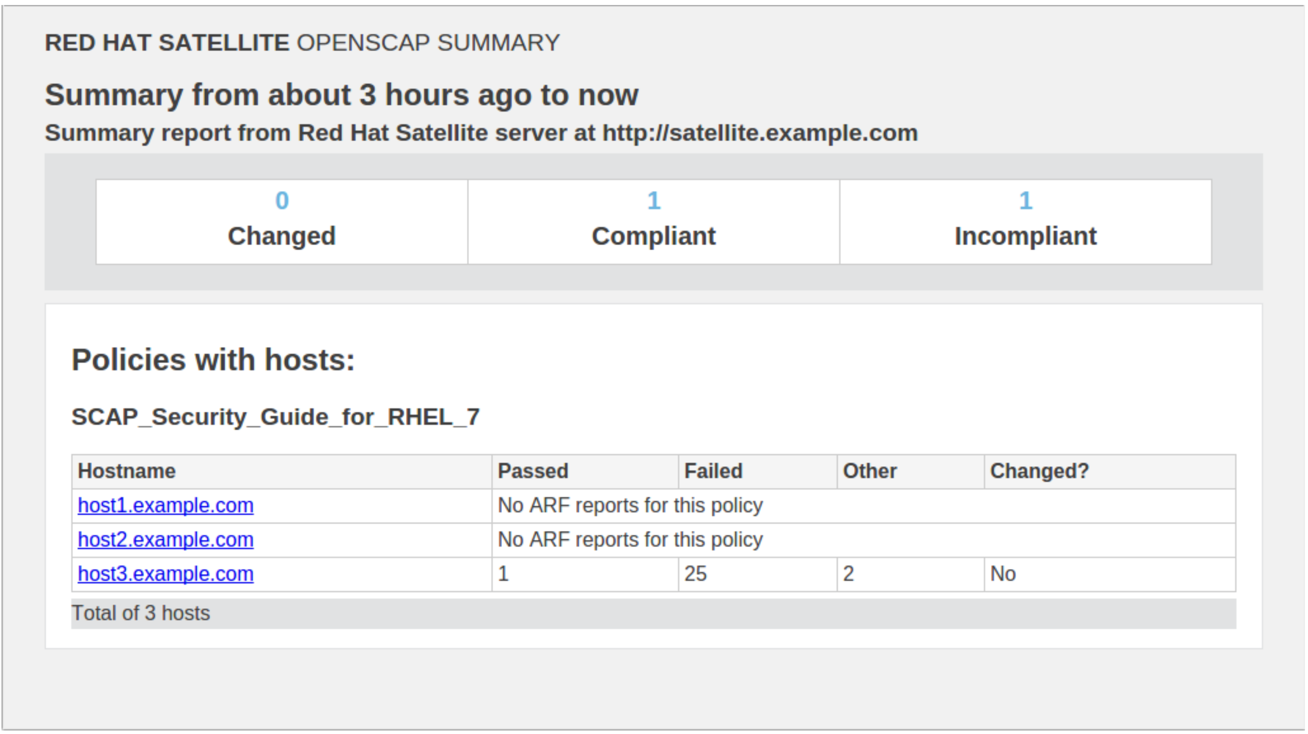
5.4.5. Compliance Email Notifications

The Satellite Server sends an OpenSCAP Summary email to all users who subscribe to the **`Openscap policy summary`** email notifications (refer to Configuring Email Notifications in the *Server Administration Guide*). Each time a policy is run, Satellite checks the results against the previous run, noting any changes between them. The email is sent according to the frequency requested by each subscriber, providing a summary of each policy and its most recent result.

An **`OpenSCAP Summary`** email message contains the following information:

- Details of the time period it covers.
- Totals for all hosts by status: changed, compliant, and incompliant.
- A tabular breakdown of each host and the result of its latest policy, including totals of the rules that passed, failed, changed, or where results were unknown.

The following is an example OpenSCAP Summary email message.



RED HAT SATELLITE OPENSCAP SUMMARY

Summary from about 3 hours ago to now

Summary report from Red Hat Satellite server at http://satellite.example.com

0	1	1
Changed	**Compliant**	**Incompliant**

Policies with hosts:

SCAP_Security_Guide_for_RHEL_7

Hostname	Passed	Failed	Other	Changed?
host1.example.com	No ARF reports for this policy			
host2.example.com	No ARF reports for this policy			
host3.example.com	1	25	2	No

Total of 3 hosts

CHAPTER 6. WORKING WITH CONTAINERS

Docker is an open source project that automates the deployment of applications inside *Linux containers*, and provides the capability to package an application with its runtime dependencies into a container. Linux containers enable rapid application deployment, simpler testing, maintenance, and troubleshooting while improving security. For more information, see the Get Started with Docker Formatted Container Images on Red Hat Systems article on the Red Hat Customer Portal[1].

A container in the Docker format is composed of the following parts:

- **`Container`**: 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 a new image layer is added to store your changes.

- **`Image`**: A static snapshot of the container's configuration that is never modified. Any changes made to the container can be saved only by creating a new image layer. Each image depends on one or more parent images.

- **`Platform image`**: An image that has no parent. Platform images define the runtime environment, packages and utilities necessary for containerized applications to run. The platform image is not writable, so any changes are reflected in the copied images stacked on top of it. For information on how to access Red Hat Enterprise Linux platform images from Red Hat Satellite see Example 6.1, “Creating a Red Hat Enterprise Linux Container in Satellite”.

- **`Registry`**: A public or private archive that contains images available for download. Some registries allow users to upload images to make them available to others. Red Hat Satellite allows you to import images from local and external registries. Satellite itself can act as an image registry for hosts, however, hosts cannot push changes back to the registry. For more information, see Section 6.1.1, “Creating Containers”

- **`Tag`**: A mark used to differentiate images in a repository, typically by the version of the application stored in the image. Repositories are used to group similar images in a container registry. Images only have unique alphanumeric identifiers, so naming in form or *repository*:*tag* provides a human-readable way of identifying images. For more information, see Section 6.5, “Using Container Tags” and Section 6.2, “Managing Repositories”.

With Red Hat Satellite, you can create an on-premise registry, import images from various sources and distribute them to containers using content views (see Section 3.2, “Adding Repositories to the Content View” for more information on loading images to a content view). Satellite supports creating one or more Docker compute resources that act as servers for running containers. This way, you can import an image, start a container based on this image, monitor the container's activity, and commit its state to a new image layer that can be further propagated.

6.1. MANAGING CONTAINERS

The following sections show how to create, view, start, stop, and commit a container.

Prerequisites

In Red Hat Satellite, you can deploy containers only on a compute resource of the Docker provider type. Therefore, when you attempt to view or create containers for the first time, Satellite prompts you to create a Docker compute resource. To do so, first create a container host, then specify this host as a compute resource.

Procedure 6.1. To Prepare a Container Host:

1. Prepare a Red Hat Enterprise Linux 7 server for hosting images and enable the **`docker`** service on this server as described in the*Getting Docker in RHEL 7* section of the Get Started with Docker Formatted Container Images on Red Hat Systems guide on the Red Hat Customer Portal[2]. You can deploy the container host either on the same machine as the Satellite Server or independently.

   NOTE

   Red Hat Enterprise Linux 7 is currently the only supported system for a container host. The docker package is available in the rhel-7-server-extras-rpms repository. Red Hat Enterprise Linux 6 systems are currently not supported to host containers.

2. Run the following command on the container host to install the Satellite Server's CA certificate:

   ```
   rpm -Uvh https://satellite.example.com/pub/katello-ca-consumer-
   latest.noarch.rpm
   ```

   Here, *satellite.example.com* is the fully qualified domain name of your Satellite Server. Skip this step if the container host is already registered as a Satellite host.

3. Depending on the location of the container host, perform the following tasks:

   - If the container host is on the same machine as the Satellite Server:

     a. Create a docker user group and add the foreman user to it:

     ```
     # groupadd docker
     # usermod -aG docker foreman
     ```

     b. Modify the OPTIONS variable in the **`/etc/sysconfig/docker`** file as follows:

     ```
     OPTIONS='--selinux-enabled -G docker'
     ```

     c. Restart the affected services to apply the changes:

     ```
     # systemctl restart docker.service
     # katello-service restart
     ```

   - If the container host is on a different machine than the Satellite Server:

     a. Open a port on the container host to communicate with the Satellite Server. To do so, modify the OPTIONS variable in the **`/etc/sysconfig/docker`** file as follows:

     ```
     OPTIONS='--selinux-enabled -H tcp://0.0.0.0:2375 -H
     unix:///var/run/docker.sock'
     ```

-

  You can use port **2376** if TLS is enabled.

b. Restart the docker service and verify your settings as follows:

```
# systemctl restart docker.service
# systemctl status docker.service
```

Procedure 6.2. To Create a Docker Compute Resource:

1. Make sure the port 5000 is enabled on the Satellite Server. The container host uses this port to pull images from Content Views on the Satellite Server.

2. Create the compute resource as described in Section 9.3.4, “Compute Resources”. Specify the resource **URL** according to the location of the container host:

   a. If the container host is on the same machine as the Satellite Server, set *unix://var/run/docker.sock* as the resource URL.

   b. If the container host is on a different machine than the Satellite Server, specify the URL in the form of:

   ```
   http://container_host_fqdn:2375
   ```

   Here, *container_host_fqdn* stands for the fully qualified domain name of the container host, and the port number opened on the container host for communication with Satellite can be either **2375** or, if using TLS, **2376**.

3. Click **Test Connection** to test if the container host is available.

4. Click **Submit** to create the compute resource.

6.1.1. Creating Containers

When there is at least one Docker compute resource present in your Satellite, you can create containers. To create a new container, follow the steps described in Procedure 6.3, “To Create a Container:”. For instructions on how to monitor existing containers, see Section 6.1.2, “Monitoring Containers”.

To create a container, you must first import an image, which can be a platform image or a previously created layered image. Satellite supports the following image sources:

- **Local content**: represented by the **Content View** option when creating a container. This option allows you to import an image from a repository that is already present on a Capsule Server in a certain content view and life cycle environment. For more information on how to create and populate a local registry, see Section 6.2, “Managing Repositories”.

- **Docker Hub**: allows you to search the Docker Hub registry and pull images from there. Make sure that you pull only trusted images with verified content.

- **External Registry**: allows you to import images from a previously created external registry. For more information on creating registries in Red Hat Satellite, see Section 6.3, “Importing External Registries”.

NOTE

You cannot change the configuration of an existing container. To alter the configuration, you have to create a replacement container with modified settings as described in Procedure 6.3, “To Create a Container:”. Therefore, make sure that containers can be replaced in your workflow.

Procedure 6.3. To Create a Container:

1. Navigate to **Containers** → **New Container**. Alternatively, navigate to **Containers** → **All Containers** and click **`New container`**.

2. In the **`Preliminary`** stage of container creation, configure the following settings:

    - On the **`Compute resource`** tab, select the compute resource from the **`Deployed on`** drop-down menu. For more information on compute resources, see Section 9.3.4, “Compute Resources”.

    - On the **`Locations`** tab, select the locations where the new container will be available.

    - On the **`Organizations`** tab, select the organizations where the new container will be available.

    Click **`Next`** to proceed.

3. In the **`Image`** stage of container creation, import an image that will act as a base for your container. This can be a platform image, or a previously created layered image. Select from one of the following options:

    - Select the **`Content View`** tab to import the image from a life cycle environment. Specify the life cycle environment, content view, repository, tag, and Capsule Server.

    - Select the **`Docker hub`** tab to import the image from the Docker Hub registry. After you type the image name to the **`Search`** field, Satellite automatically searches the compute resource. Click the looking glass icon to search the Docker Hub. Select the image from the list of search results and pick a tag from the drop-down list.

    - Select the **`External registry`** tab to import the image from an existing registry. Select the registry from the drop-down menu, and search it by the image name. Satellite populates the **`Tag`** field with tags available for the selected image name. For more information, see Section 6.3, “Importing External Registries”.

    Click **`Next`** to proceed.

4. In the **`Configuration`** stage of container creation, set the following parameters:

    - Provide the container name.

    - Specify a command to run inside the container.

    - Specify an entrypoint, which is a command that is executed automatically as soon as the container starts. The default entrypoint is **`/bin/sh -c`**.

- Assign CPUs to the container. For example, **`0-2,16`** represents CPUs 0, 1, 2, and 16.
- Define the relative share of CPU time for the container.
- Specify a memory limit for the container. For example, **`512m`** limits the container memory usage to 512 MB.

Click **`Next`** to proceed.

5. In the final stage of container creation named **`Environment`**, select if you want to allocate a pseudo-tty, attach STDIN, STDOUT, and STDERR to the container. Click **`Add environment variable`** to create a custom environment variable for the container. Select the **`Run?`** check box to start the container automatically after it is created.

6. Click **`Submit`** to create the container.

After creating a container, Satellite displays a summary of container metadata. By default, new containers are disabled (unless you selected the **`Run?`** check box when creating the container). For instructions how to start containers see Procedure 6.5, “To Start or Stop a Container:”.

Example 6.1. Creating a Red Hat Enterprise Linux Container in Satellite

To enable a Red Hat Enterprise Linux container in Red Hat Satellite, perform the following actions:

1. Create a custom registry as described in Section 6.3, “Importing External Registries”. Specify *registry.access.redhat.com* as the registry URL.
2. Create a new container as described in Section 6.1.1, “Creating Containers”. In the **`Image`** stage of container creation, navigate to the **`External registry`** tab and select the registry created in the previous step. Use the search field to find the desired version of the Red Hat Enterprise Linux image. Proceed through the **`Configuration`** and **`Environment`** stages to finalize the container.

6.1.2. Monitoring Containers

Red Hat Satellite provides the means to monitor the status of containers as well as processes running inside them. Some containers can be marked as *managed*, which means they were created and provisioned inside the Satellite environment.

The following procedure shows how to list containers of a selected organization and how to monitor the container metadata.

Procedure 6.4. To Investigate a Container:

1. Navigate to **Containers** → **All Containers**.
2. On the **`Containers`** page, every Docker compute resource has a dedicated tab. Each of these tabs contains the table of available containers together with selected parameters of each container. Select the tab of the compute resource you want to inspect.

3. To view the container metadata, click the name of the container you want to inspect. Satellite displays the table of container properties.

4. On the **`Processes`** tab, you can view processes that are currently running in the container. Click on the process name to view the metadata of the process.

5. If the container is running, you can view its standard output in the **`Logs`** tab. If you selected the **`allocate a pseudo-tty`** check box when creating a container, the console is interactive. Otherwise, it displays the initial standard output produced when the container started.

6.1.3. Starting, Committing, and Removing Containers

New containers are by default disabled. By enabling a container, you start the processes of the containerized application in the compute resource. Hosts are then able to communicate with the container as with a web application. The following procedure shows how to start and stop a container:

Procedure 6.5. To Start or Stop a Container:

1. Navigate to **Containers** → **All Containers** to view the list of available containers.

2. Click **`Power On`** next to the container you want to start. After starting the container, the button changes to **`Power Off`**, which allows for stopping the container. These actions are equivalent to the **`docker start`** and **`docker stop`** commands.

The following procedure shows how to commit a container to create a new image layer that stores the status of the container.

Procedure 6.6. To Commit a Container:

1. Navigate to **Containers** → **All Containers** to view the list of available containers.

2. Click the name of the container you want to commit.

3. Click **`Commit`**. Satellite prompts you to:

    - Specify a repository name. This can be a single name or combined with the user name, for example *user/my-rhel-image*.

    - Assign a tag to the image.

    - Provide your contact information.

    - Provide an informative comment about the image.

4. Click **`Submit`**.

NOTE

The container is committed to the repository of the original image. For example, if the container is based on an image pulled from the Docker Hub, the committed changes are pushed back to the Docker Hub.

Procedure 6.7. To Remove a Container:

1. Navigate to **Containers** → **All Containers** to view the list of available containers.

2. Click the name of the container you want to delete.

3. Click **`Delete`**.

4. In the alert box, click **`OK`** to remove the container.

6.2. MANAGING REPOSITORIES

This section shows how to create an internal repository for container images. You can use internal repositories to create containers as described in Section 6.1.1, “Creating Containers”.

6.2.1. Creating Repositories

Repositories provide a way to synchronize the container content either from the Red Hat Content Delivery Network or from other sources.

Procedure 6.8. To Create a Docker Repository:

1. Navigate to **Content** → **Products**. Click **`New Product`**, specify the product name and click **`Save`**.

2. Select the product you created in the previous step and navigate to the **`Repositories`** tab. Click **`Create Repository`**.

3. Specify the repository name and select **`docker`** from the **`Type`** drop-down menu. This unlocks additional fields where you specify the **`URL`** of the content source you want to synchronize in this registry. Specify which repository you want to pull from the content source in the **`Upstream Repository Name`** field.

   NOTE

   In the previous version of Red Hat Satellite it was possible to upload locally stored container images to the repository. With Red Hat Satellite 6.2, this is no longer possible.

4. Click **`Save`** to create the repository. First the URL and repository name is validated, then the repository is created.

6.3. IMPORTING EXTERNAL REGISTRIES

The following procedure shows how to import an external registry to the Satellite Server.

Procedure 6.9. To Import an External Registry:

1. Navigate to **Containers** → **Registries**. Click **`New Registry`**.

2. On the **`Registry`** tab, specify the name and URL of the registry. These settings are required. Optionally, provide a brief description of the registry. Specify a user name and password if required for accessing the registry.

3. On the **`Locations`** tab, select the locations where the new registry will be available.

4. On the **Organizations** tab, select the organizations where the new registry will be available.

5. Click **Submit** to create the registry.

6.4. IMPORTING IMAGES TO COMPUTE RESOURCES

Importing an image is a necessary step when creating a container. You can also import images to a compute resource before creating a container as described in the following procedure.

Procedure 6.10. To Import an Image to a Compute Resource:

1. Navigate to **Infrastructure** → **Compute resources** to view a list of compute resources.

2. Select the docker compute resource you want to edit.

3. Click **New image**.

4. Specify the image details including the image name, operating system, architecture, user credentials, and a parent image. Select **User data** to enable user input for this image.

5. Click **Submit**.

6.5. USING CONTAINER TAGS

Tags are convenient for organizing images, especially when you operate with several versions of a containerized application. The following procedure shows how to use tags to search for images:

Procedure 6.11. To Search Registries by Tags:

1. Navigate to **Content** → **Docker tags**.

2. Use the search field to filter tags by the image name, tag, or repository name. Automatic suggestion works as you type. For example, the following query searches for tags applied on images from the repository named test_repo:

   ```
   repository = test_repo
   ```

3. Click the name of the tag you want to view. Satellite displays a list of images that use this tag.

4. Select an image to view its environment and content view version. The **Published At** field shows the URL that you can use to pull the image from the command line.

For the list of alternative comparison operators, see Supported Operators for Granular Search in the *Server Administration Guide*. By default, the search field recognizes the input string as a tag name. For example, type **centos** to search for all centos tags.

[1] https://access.redhat.com/articles/881893

[2] https://access.redhat.com/articles/881893#get

CHAPTER 7. CONFIGURING ACTIVATION KEYS

Activation keys define selected properties of content hosts. You can use activation keys during content host registration to improve the speed, simplicity and consistency of the process. Activation can specify:

- Associated subscriptions and subscription attach behavior.
- Available products and repositories.
- A life cycle environment and a content view.
- Host collection membership.

The same activation key can be applied to multiple content hosts, as long as it contains enough subscriptions. However, activation keys only set the initial configuration for a content host. When it is registered to an organization, other content which that organization possesses can be attached to the content host manually.

A content host can be associated with multiple activation keys that are combined to define the host settings. In case of conflicting settings, the last specified activation key takes precedence.

Note that activation keys are only used when hosts are registered. If changes are made to an activation key, it is only applicable to hosts that are registered with the amended activation key in the future. The changes are not made to existing hosts.

7.1. CREATING AN ACTIVATION KEY

This section describes how to create an activation key.

Procedure 7.1. To Create an Activation Key:

1. Click **Content** → **Activation keys**.
2. Click **`New Activation Key`**. Perform the following actions:
   a. Specify the activation key name. This setting is required.
   b. Optionally, clear the **`Unlimited Hosts`** check box if you want to limit the number of host that can be associated with the activation key. Specify the number in the **`Limit`** field. .
   c. Optionally, enter a suitable description in the **`Description`** field. You can also select the **`Environment`** and **`Content View`** to which this key should apply. For host registration, select a content view that has the **`Satellite Tools`** repository enabled.
3. Click **`Save`** to create the activation key.

7.2. DEFINING SUBSCRIPTION PROPERTIES OF AN ACTIVATION KEY

With activation keys you can define how the content host is subscribed during registration. The subscription behavior defined by the activation key depends on two factors:

1. Are there any subscriptions associated with the activation key?
2. Is the auto-attach option enabled?

Based on the above factors, there are three possible scenarios for subscribing with activation keys:

- **Activation key with no subscriptions specified**. With no subscriptions specified and auto-attach enabled, hosts using the activation key search for the best fitting subscription from the ones provided by the Satellite Server. This is akin to running the **`subscription-manager --auto-attach`** command.
- **Activation key providing a custom subscription pool for auto-attach** . If there are subscriptions specified and auto-attach is enabled, hosts using the activation key select the best fitting subscription from the list specified in the activation key.
- **Activation key with the exact set of subscriptions**. If there are subscriptions specified and auto-attach is disabled, hosts using the activation key are associated with all subscriptions specified in the activation key.

For instructions on how to add subscriptions see Procedure 7.2, “To Add a Subscription to an Activation Key:”, to learn how to enable auto-attach seeProcedure 7.4, “To Enable or Disable Auto-Attach on an Activation Key:”.

NOTE

If a custom product (typically containing content not provided by Red Hat) is assigned to an activation key, this product is always enabled for the registered content host regardless of the auto-attach setting.

7.2.1. Adding and Removing Subscriptions

This section describes how to add subscriptions to an activation key as well as how to remove them.

Procedure 7.2. To Add a Subscription to an Activation Key:

1. Click **Content** → **Activation keys**.
2. Click the activation key name you want to edit.
3. On the **`Subscriptions`** tab, select the **`Add`** subtab.
4. From the list of available subscriptions, select the subscriptions you want to add.
5. Click **`Add Selected`**.

Procedure 7.3. To Remove Subscriptions from an Activation Key:

1. Click **Content** → **Activation keys**.
2. A list of activation keys is displayed. Click the activation key you want to remove subscriptions from.
3. Click the **Subscriptions** tab.

4. Under the **`List/Remove`** subtab, a list of attached subscriptions is displayed. Select the subscriptions to be removed.

5. Click **`Remove Selected`**.

7.2.2. Enabling Auto-Attach

When auto-attach is enabled on an activation key and there are subscriptions associated with the key, the subscription management service selects and attaches the best-matched associated subscriptions based on a set of criteria like currently-installed products, architecture, and preferences like service level.

You can enable auto-attach and have no subscriptions associated with the key. This type of key is commonly used to register virtual machines when you do not want the virtual machine to consume a RHEL subscription but to inherit a RHEL Virtual Data Center (VDC) subscription from the hypervisor.

Auto-attach is enabled by default. Disable the option if the desired behavior is to force attach all subscriptions associated with the activation key.

Procedure 7.4. To Enable or Disable Auto-Attach on an Activation Key:

1. Click **Content** → **Activation keys**.

2. Click the activation key name that you want to edit.

3. Click the **`Subscriptions`** tab.

4. Click the edit icon next to **`Auto-Attach`**.

5. Select or deselect the check box to enable or disable auto-attach.

6. Click **Save**.

NOTE

To register virtual content hosts to the Satellite Server using an auto-attach activation key, first use the **virt-who** utility to map those hosts to a hypervisor entitled with the Virtual Datacenter (VDC) subscription. Without this prerequisite, virtual hosts will be registered only with a temporary virtual subscription for 24 hours. For more information see the Red Hat Satellite Virtual Instances Guide.

7.2.3. Setting the Service Level

An activation key can be configured to define a default service level for the new host created with the activation key. Setting a default service level will select only the matching subscriptions to be attached to the host. For example, if the default service level on an activation key is set to Premium, only subscriptions with premium service levels will be attached to the host upon registration.

Procedure 7.5. To Set the Service Level on an Activation Key:

1. Click **Content** → **Activation keys**.

2. Click the activation key name you want to edit.
3. Click the **`Details`** tab.
4. Click the edit icon next to **`Service Level`**.
5. Select the required service level from the drop-down list. The drop-down list only contains service levels available to the activation key.
6. Click **Save**.

7.3. EDITING ACTIVATION KEYS

This section describes how to edit activation key properties such as host collections, product content, and life cycle environment.

7.3.1. Adding and Removing Host Collections

These steps show how to add host collections to an activation key. Host collections can be associated with activation keys so that hosts utilizing the activation keys will automatically be added to the associated host collections upon registering with the Satellite Server.

Procedure 7.6. To Add Host Collections to an Activation Key:

1. Click **Content** → **Activation keys**.
2. Click the activation key that you want to add a host collection to.
3. On the **`Host Collections`** tab click the **Add** subtab to display the list of available host collections.
4. Select the host collections you want to add, and then click **`Add Selected`**.

Procedure 7.7. To Remove Host Collections from the Activation Key:

1. Click **Content** → **Activation keys**.
2. A list of activation keys is displayed. Click the activation key you want to remove host collections from.
3. Click the **Host Collections** tab.
4. Under **`List/Remove`** subtab, a list of host collections attached to the activation key is displayed. Select the check box of the host collections you want to remove.
5. Click **`Remove Selected`** to remove host collections from the activation key.

7.3.2. Editing Product Content

The number of products available for the activation key is determined by associated subscriptions. You can change which repositories in products are enabled on the **`Product Content`** tab.

Procedure 7.8. To Edit Product Content on an Activation Key:

1. Click **Content** → **Activation keys**.
2. Click the activation key name that you want to edit.
3. Click the **`Product Content`** tab to view the products and repositories associated with the activation key through subscriptions.
4. Click the edit icon next to the repository you want to edit.
5. From the drop-down menu, select if the repository will be enabled or disabled. Click **`Save`** to apply the change.

7.3.3. Setting a Life Cycle Environment and a Content View

You can set a life cycle environment and a content view for an activation key when creating it. It is also possible to modify these settings afterwards using the following procedure.

Procedure 7.9. To Set a Life Cycle Environment and a Content View for an Activation Key:

1. Click **Content** → **Activation keys**.
2. Click the activation key name that you want to edit.
3. Click the check box next to the environment you want to associate with the activation key. Select a content view from the drop-down menu.
4. Click **`Save`**.

7.4. REMOVING AN ACTIVATION KEY

This section describes how to remove an activation key.

Procedure 7.10. To Remove an Activation Key:

1. Click **Content** → **Activation keys**.
2. Click the activation key name that you want to remove.
3. In the upper right of the **`Activation Key`** details panel, click **`Remove`**.
4. In the alert box, click **`Remove`** to confirm that you want to remove the key.

7.5. AUTOMATED HOST REGISTRATION WITH ACTIVATION KEYS

The following steps show how to automatically register a host using an activation key. When the activation key has been created, you can apply it by using the **`subscription-manager`** utility during host registration on the Satellite Server. Note that the version of the **`subscription-manager`** utility installed must be 1.10 or higher. Prepare the host as described in Section 10.5.1, "Configuring a Host for Registration", then follow the steps outlined in Procedure 7.11, "To Automatically Register a Host with an Activation Key:".

Procedure 7.11. To Automatically Register a Host with an Activation Key:

1. Clear any old registration data from the system:

   ```
   # subscription-manager clean
   ```

2. Download and install a copy of the CA Certificate for the host from the Satellite Server:

   ```
   # rpm -Uvh http://satellite.example.com/pub/katello-ca-consumer-
   latest.noarch.rpm
   ```

3. Register the system to the required organization on the Satellite Server. Use the --*activationkey* flag to register the system using the activation key. Enter the user authentication details when prompted.

   ```
   # subscription-manager register --org "Default_Organization" --
   activationkey "Test_Key"
   ```

4. When the system is registered, it gains access to repository content but administrators will not be able perform package and errata management until the Katello agent has been installed on the client system. To do so, execute:

   ```
   # yum install katello-agent
   ```

Combining Multiple Activation Keys for Host Registration

You can use multiple activation keys when registering a content host. This allows you to create activation keys for specific subscription sets and then combine them according to content host requirements. For example, the following command registers a content host to the ACME organization with both VDC and OpenShift subscriptions:

```
# subscription-manager register --org "ACME" --activationkey "ak-VDC,ak-
OpenShift"
```

If there are conflicting settings in activation keys, the rightmost key takes precedence.

- Settings that conflict: Service Level, Release Version, Environment, Content View, and Product Content.

- Settings that do not conflict and the host will get the union of them: Subscriptions and Host Collections.

- Settings that influence the behavior of the key itself and not the host configuration: Content Host Limit and Auto-Attach.

CHAPTER 8. CONFIGURING GPG KEYS

GPG keys allow you to add your existing GPG keys to Red Hat Satellite Server products and repositories to enable pairing with your repositories.

8.1. CREATING A GPG KEY

This section describes how to add a GPG key to Red Hat Satellite.

Procedure 8.1. To Add a GPG Key to Satellite:

1. Click **Content** → **GPG Keys** and then click **New GPG Key**.
2. In the **Name** field enter a name for the GPG key.
3. Either upload the GPG key file or paste the GPG key contents into the text box.
4. Click **Save** to add the GPG key to Satellite.

8.2. REMOVING A GPG KEY

This section describes how to remove a GPG from Red Hat Satellite.

Procedure 8.2. To Remove a GPG Key:

1. Click **Content** → **GPG Keys**.
2. Click the GPG key that you want to remove, and then click **Remove GPG Key**.
3. In the confirmation box, click **Remove** to confirm that you want to remove the selected key.

CHAPTER 9. CONFIGURING THE PROVISIONING ENVIRONMENT

Provisioning refers to a process that starts with a bare physical or virtual compute resource and ends with a fully configured, ready-to-use operating system. Red Hat Satellite provides an ability to define and automate fine-grained provisioning for a large number of hosts. This section shows how to configure the necessary components needed for provisioning. For details on the provisioning process itself, refer to the Red Hat Satellite Provisioning Guide.

9.1. CREATING A HOST GROUP

A host group defines a set of default values that hosts inherit when placed in that group. Hosts can belong to only one host group, but host groups can be nested in hierarchies. You can create a "base" or "parent" host group that represents all hosts in your organization, and then create nested or "child" host groups under that parent to provide specific settings. This section describes how to create a host group.

Procedure 9.1. To Add a Host Group to Satellite:

1. Click **Configure** → **Host Groups** and then click **`New Host Group`**.

2. Enter the required details for the Host Group, and then click **`Submit`**.

Host Group Attributes

The following table describes the attributes that apply to Satellite Host Groups.

Table 9.1. Table of Host Group Attributes

Submenu	Options	Description
Host Group	Parent	The parent Host Group for the new Host Group.
	Name	The name of the Host Group.
	Life Cycle Environment	The environment containing this Host Group.
	Puppet CA	The Red Hat Satellite Capsule Server to use for the Puppet CA server.
	Puppet Master	The Red Hat Satellite Capsule Server to use as the Puppet Master.
Puppet Classes	Included Classes	The Puppet Classes included with the Host Group.
	Available Classes	The Puppet Classes available to use with the Host Group.
Network	Domain	The domain for hosts in the Host Group.
	Subnet	The subnet for hosts in the Host Group.

Submenu	Options	Description
Operating System	Architecture	The default architecture for systems in the Host Group.
	Operating Systems	The default operating system for systems in the Host Group.
	Media	The location of the installation media for the operating system.
	Partition Table	A file system partition layout for the operating system installation.
	Root Password	The root password for the operating system.
Parameters	Add Parameter	Provides a Name and Value pair to set parameters for the Host Group.
Organizations	Organizations	The organizations that own this host group.
Activation Keys	Content Environment	Defines the activation keys made available in templates as **`@host.params['kt_activation_keys']`**.

9.2. PARAMETERS

Red Hat Satellite parameters define key-value pairs to use when provisioning hosts. These are similar to Puppet's concept of a default scope parameter. You can define parameters when setting up a host with Puppet.

Types of Parameters

Red Hat Satellite has two types of parameters:

Simple Parameters

String parameters that define a relationship between a key and value pair. They cannot be overridden by user configuration, but they are overridden according to Satellite's parameter hierarchy. The following parameters are simple parameters in Red Hat Satellite: Global, organization-level, location-level, domain-level, operating system level, host group, and host parameters.

Smart Parameters

Complex parameters that define a value for a key but allow conditional arguments, validation, and overrides for specific object types. Smart parameters enable a Puppet class to get external data. They are used in Puppet Classes called *parameterized classes* in Puppet terminology. The hierarchy for these parameters can be configured the in the web UI.

The following parameter hierarchy applies for simple parameters:

Global Parameters

Default parameters that apply to every host in Satellite. Configured in **Configure → Global parameters**.

Organization-level parameters

Parameters that affect all hosts in a given organization. Organization-level parameters override Global parameters. Configured in **Administer → Organizations → Edit → Parameters**.

Location-level parameters

Parameters that affect all hosts in a given location. Location-level parameters override Organization-level and Global parameters. Configured in **Administer → Locations → Edit → Parameters**

Domain Parameters

Parameters that affect all hosts in a given domain. Domain parameters override Location-level and higher parameters. Configured in **Infrastructure → Domains → [choose_a_domain] → Parameters**.

Operating System Level Parameters

Parameters that affect all hosts with a given operating system. Operating system level parameters override Domain and higher parameters. Configured in **Hosts → Operating systems → [choose_an_operating_system] → Parameters**.

Host Group Parameters

Parameters that affect all hosts in a given Host Group. Host Group parameters override Operating system level and higher parameters. Configured in **Configure → Host Groups → [choose_a_host_group] → Parameters**.

Host Parameters

Parameters that affect a specific host. All previously inherited parameters are visible on the Parameters subtab and can be overridden. Configured in **Hosts → All hosts → Edit → Parameters**.

Using Parameters with Puppet Classes

Red Hat Satellite has two ways to supply values to a Puppet Master for a host to use with a Puppet class:

Smart Variables

A tool to provide global parameters to the Puppet Master, in key-value form, for classes that do not have Smart parameters. They enable overriding parameter values in a Puppet manifest. They are intended for use when a class does not have Smart parameters or in special cases when a global parameter is desired. They can have multiple possible values, all depending on hierarchical context or various conditions a user can apply. They existed before Puppet had parameterized classes and today are kept either for backward compatibility or for the use of global parameters where you want validations, to use only with specific Puppet classes, and for types other than string (because otherwise you could just use the simple parameters).

Parameterized Classes

Puppet classes containing Smart parameters. The classes are imported from the Puppet Master and the name of the parameter, for example **`$::`*`name`*** (preferred) or **`$`*`name`***, is defined by the person who wrote the class and cannot be changed. They enable you to decide the value of the variable for a specific class rather than globally.

Configured parameters are included in each host's corresponding YAML file and sent to the Puppet Master. The YAML file can be viewed in the web UI on the page for a specific host. You should not manually change the **`/etc/foreman/settings.yaml`** configuration file because they are overwritten the next time you run the **`satellite-installer`** command.

IMPORTANT

Parameterized class support is enabled by default in Satellite 6. If required to ensure that it is, navigate to **Administer** → **Settings**, select the **`Puppet`** tab, and ensure the **`Parametrized_Classes_in_ENC`** is set to **`True`**.

9.2.1. Creating a Global Simple Parameter

This procedure shows how to add a new global parameter to Satellite.

Procedure 9.2. To Create a Global Simple Parameter:

1. Click **Configure** → **Global Parameters**.
2. Click **`New Parameter`**.
3. Type a **`Name`** for the parameter's key.
4. Enter a **`Value`** for the parameter.
5. Optionally select if you want the value to be hidden in the web UI.
6. Click **`Submit`**.

9.2.2. Configuring Smart Variables

The following procedure configures Smart Variables to override a value in a Puppet class.

Procedure 9.3. To Configure Smart Variables:

1. Click **Configure** → **Puppet Classes**.
2. Select a class from the list.
3. Click the **`Smart Variables`** tab. This displays a new screen. The left section contains a list of possible parameters the class supports. The right section contains the configuration options for the parameter selected. Click the **`Add Variable`** to add a new parameter. Otherwise, select a parameter from the left-hand list.
4. Type a name for the parameter in the **`Key`** field.
5. Edit the **`Description`** text box to add any plain text notes.
6. Select the **`Key type`** of data to pass. This is most commonly a string, but other data types are supported.
7. Enter a **`Default Value`** for the parameter to be sent to the Puppet Master if no host

match occurs.

8. Optionally select **Hidden value** if the field contains data you do not want to be displayed while you are working.

9. Use the **Optional Input Validator** section to restrict the allowed values for the parameter. Choose a **Validator type** (either a **list** of comma separated values or a regular expression, **regexp**) and input the allowed values or regular expression code in the **Validator rule** field.

10. The **Prioritize attribute order** section provides options for overriding values for specific hosts based upon conditional arguments. The attribute type and its value is known as a *matcher*.

    a. Set the **Order** of precedence in which the host attributes or Facts are to be evaluated against the matchers by arranging the entries in the list. You can add to the default list. To create a logical AND condition between matchers, arrange them on one line as a comma separated list.

    b. Click **Add Matcher** to add a conditional argument. The attributes to match against should correspond to the entries in the **Order** list. If no matcher is configured then only the default value can be used for the override feature.

       For example, if the desired value of the parameter to be supplied to the Puppet Master is **test** for any host with a fully qualified domain name of **server1.example.com**, then specify the matcher as **fqdn=server1.example.com** and the **Value** as **test**.

       The precedence for matching is as follows:

       1. If the matcher is a host attribute, use that.

       2. If there are no attributes with that name, look for a matching host parameter (which is inherited according to the parameter hierarchy).

       3. If there is still no match, check the host Facts.

       It is recommend to use an attribute that is present in Facter and cannot be confused with a host attribute. Host attributes can be either host parameters or associations to the host, such as host group, domain, and organization. The matcher must only be something the host has one of, for example config group cannot be used because the host can have many config groups but a host only has one location so location is a valid matcher.

       Dynamic data is possible by using parameters and Puppet Facts in the **Value** field in *Embedded Ruby* (ERB) template syntax. For example, to use a Puppet Fact as part of the value:

       ```
       <%= @host.facts['network_eth0'] %>
       ```

       To list available Puppet Facts navigate to **Monitor** → **Facts**.

11. Click **Submit** to save your changes.

For further information on working with Puppet modules see Adding Puppet Modules to Red Hat Satellite 6. For more information on ERB syntax seeAppendix A, *Template Writing Reference*.

9.2.3. Importing Parameterized Classes from a Puppet Master

The following procedure imports parameterized classes from your Puppet Master.

NOTE

The import of parameterized classes happens automatically if your Puppet modules are managed via a Product and a Content View.

Procedure 9.4. To Import Parameterized Classes:

1. In the Satellite web UI, select from the context menu **`Any Organization`** and **`Any Location`**.
2. Click **Configure** → **Puppet Classes**.
3. Click **`Import from`** ***`Host Name`*** to import parameterized classes from your Puppet Master.
4. The **`Puppet Classes`** page displays with the new classes listed.

9.2.4. Configuring Smart Class Parameters

The following procedure configures parameters within a class. Classes that contain parameters are referred to as *parameterized classes*.

Procedure 9.5. To Configure Smart Class Parameters:

1. Click **Configure** → **Puppet Classes**.
2. Select a class from the list that has parameters as indicated in the **`Parameters`** column.
3. Click the **`Smart Class Parameter`** tab. This displays a new screen. The left section contains a list of possible parameters the class supports. The right section contains the configuration options for the parameter selected.
4. Select a parameter from the left-hand list.
5. Edit the **`Description`** text box to add any plain text notes.
6. Select **`Override`** to allow Satellite control over this variable. If the check box is not selected, Satellite does not pass the new variable to Puppet.
7. Select the **`Key type`** of data to pass. This is most commonly a string, but other data types are supported.
8. Enter a **`Default Value`** for the parameter to be sent to the Puppet Master if no host match occurs.
9. Optionally select **`Use Puppet Default`** to **not** send a value to the Puppet Master unless an override match occurs.
10. Optionally select **`Hidden value`** if the field contains data you do not want to be displayed while you are working.

11. Use the **`Optional input validator`** section to restrict the allowed values for the parameter. Choose a **`Validator type`** (either a **`list`** of comma separated values or a regular expression, **`regexp`**) and input the allowed values or regular expression code in the **`Validator rule`** field.

12. The **`Prioritize attribute order`** section will appear if the **`Override`** option is selected. This provides options for overriding values for specific hosts based upon conditional arguments. The attribute type and its value is known as a *matcher*.

    a. Set the **`Order`** of precedence in which the host attributes or Facts are to be evaluated against the matchers by arranging the entries in the list. You can add to the default list. To create a logical AND condition between matchers, arrange them on one line as a comma separated list.

    b. Click **`Add Matcher`** to add a conditional argument. The attributes to match against should correspond to the entries in the **`Order`** list. If no matcher is configured then only the default value can be used for the override feature.

       For example, if the desired value of the parameter to be supplied to the Puppet Master is **`test`** for any host with a fully qualified domain name of **`server1.example.com`**, then specify the matcher as **`fqdn=server1.example.com`** and the **`Value`** as **`test`**.

       The precedence for matching is as follows:

       1. If the matcher is a host attribute, use that.

       2. If there are no attributes with that name, look for a matching host parameter (which is inherited according to the parameter hierarchy).

       3. If there is still no match, check the host Facts.

       It is recommend to use an attribute that is present in Facter and cannot be confused with a host attribute.

       Dynamic data is possible by using parameters and Puppet Facts in the **`Value`** field in *Embedded Ruby* (ERB) template syntax. For example, to use a Puppet Fact as part of the value:

       ```
       <%= @host.facts['network_eth0'] %>
       ```

       To list available Puppet Facts navigate to **Monitor** → **Facts**.

13. Click **`Submit`** to save your changes.

For further information on working with Puppet modules see Adding Puppet Modules to Red Hat Satellite 6. For more information on ERB syntax seeAppendix A, *Template Writing Reference*.

9.3. CONFIGURING PROVISIONING SETTINGS

This section shows how to create and configure elements of a provisioning environment.

9.3.1. Domains

Satellite has the ability to assign domain names with Red Hat Satellite Capsule Server DNS. This provides users with a means to group and name hosts within a particular domain.

Procedure 9.6. To Create a Domain:

1. Click **Infrastructure** → **Domains**.
2. Click **New Domain**. On the **Domain** tab, specify the following settings:
   a. Specify a **Name** for the Domain. This is the required DNS domain name.
   b. Type a **Description** for the Domain.
   c. Select a DNS-enabled Capsule Server.
3. On the **Parameters** tab, specify domain parameters.
4. On the **Locations** tab, select locations for the domain.
5. On the **Organizations** tab, select organizations for the domain.

IMPORTANT

Ensure that the Locations and Organizations are configured as they will help with future debugging.

6. Click **Submit**.

9.3.2. Subnets

Satellite has the ability to create networks for groups of systems. Subnets use standard IP address settings to define the network and use the Red Hat Satellite Capsule Server's DHCP features to assign IP addresses to systems within the subnet.

9.3.2.1. Creating a Subnet

The following procedure shows how to create a subnet.

Procedure 9.7. To Create a Subnet:

1. Click **Infrastructure** → **Subnets**.
2. Click **New Subnet**. On the **Subnet** tab, specify the following settings:
   a. Specify a **Name**, **Network address** (IP address), and **Network mask** for the subnet. These settings are required.
   b. Optionally, specify the **Gateway address**, **Primary DNS server**, **Secondary DNS server**, and **VLAN ID**. Note that the gateway address and DNS server settings are optional only with IPAM and Boot modes set to DHCP (default). If you decide to change these default modes, you also have to specify gateway and DNS.

      You can also select the **IPAM** mode (DHCP, Internal DB, or None) and define the IP assignment range with the **Start of IP range** and **End of IP range** fields.

   c. Select the default **Boot mode** for the subnet (DHCP or Static).

3. On the **Domains** tab, select the applicable domains for the subnet.

4. On the **Capsules** tab, select the Capsule Servers to be used for hosting the **DHCP Proxy**, **TFTP Proxy**, **DNS Proxy**, and **Discovery Proxy** services.

5. On the **Locations** tab, select locations for the subnet.

6. On the **Organizations** tab, select organizations for the subnet.

IMPORTANT

Ensure that the Locations and Organizations are configured as they will help with future debugging.

7. Click **Submit**.

9.3.3. Architectures

An architecture in Satellite represents a logical grouping of hosts and operating systems. Architectures are created by Satellite automatically when hosts check in with Puppet. However, none exist with a default installation and require creation.

Procedure 9.8. To Create an Architecture:

1. Click **Hosts** → **Architectures** and then click **New Architecture**.

2. Specify a **Name** for the architecture.

3. Select any **Operating Systems** that include this architecture. If none are available, you can create and assign them under **Hosts** → **Operating Systems**.

4. Click **Submit**.

9.3.4. Compute Resources

Compute resources are hardware abstractions from virtualization and cloud providers. Satellite uses compute resources to provision virtual machines and containers. Supported private providers include Red Hat Enterprise Virtualization, oVirt, OpenStack, VMware, Libvirt, and Docker. Supported public cloud providers include Amazon EC2, Google Compute Engine, and Rackspace.

Procedure 9.9. To Add a Compute Resource:

1. Navigate to **Infrastructure** → **Compute Resources**.

2. Click **New Compute Resource**. On the **Compute Resource** tab, specify the following settings:

   a. Specify a **Name** and a **Provider** type for the Compute Resource. Optionally, insert a **Description**.

b. Depending on the provider type chosen, the next few fields ask for authentication and datacenter details. Refer to the following table for more information about each provider type.

Table 9.2. Provider Settings

Type	Description
RHEV	Suits Red Hat Enterprise Virtualization environments. Requires the **URL** of the Manager API, a valid **Username** and **Password**, and a **Datacenter** on the system to abstract compute resources. Click **Load Datacenters** to populate the drop-down menu. Optionally, you can specify a **Quota ID** and provide one or more certificate authorities in the **X509 Certification Authorities** field.
Libvirt	Suits Libvirt-based environments. Requires the **URL** of the virtual machine. Select the **Display type**. Click **Test Connection** to test if the virtual machine is available. Select **Console passwords** to set a randomly generated password on the display connection.
VMware	Suits VMware-based environments. Requires the host name of the **VCenter/Server**, a valid VMware **Username** and **Password**, and a **Datacenter** to abstract compute resources. Click **Load Datacenters** to populate the drop-down menu. You can specify a certificate **Fingerprint** and select **Console passwords** to set a randomly generated password on the display connection.
RHEL OpenStack Platform	Suits OpenStack-based environments. Requires the **URL** of the OpenStack server, a valid OpenStack **Username** and **Password**, and a **Tenant** to abstract compute resources. Click **Load Tenants** to populate the drop-down menu.
Rackspace	Suits Rackspace public cloud accounts. Requires the **URL** of the Rackspace API, a valid Rackspace **Username** and **API Key**, and a **Region** to abstract compute resources. Click **Test Connection** to make sure your connection to the chosen region is valid.
EC2	Suits Amazon EC2 public cloud accounts. Requires the **Access Key** and **Secret Key** available from any valid Amazon EC2 account. Requires a **Region** to act as a Datacenter for resource abstraction. Click **Load Regions** to populate the selection drop-down menu.
Google	Suits Google Compute Engine public cloud accounts. Requires the **Google Project ID**, a valid **Client Email** and a **Certificate path** to the p12 file. You can also specify a**Zone** to abstract compute resources. Click **Load zones** to populate the drop-down menu.
Docker	Suits container registries. Requires the **URL** of the internal or external compute resource. Optionally, specify a **Username**, **Password**, and a contact **Email**. Click **Test Connection** to test if the connection is available.

3. On the **Locations** tab, select desired locations to add them to the**Selected Items** list.

4. On the **Organizations** tab, select the desired organizations to add them to the **Selected Items** list.

    IMPORTANT

    Ensure that the Locations and Organizations are configured as they will help with future debugging.

5. Click **Submit**.

9.3.5. Configuring Libvirt as a Compute Resource

On the system where the Libvirt hypervisor is to be used, ensure the following packages are installed:

```
# yum install qemu-kvm libvirt virt-manager
```

Ensure the FQDN of the hypervisor host resolves correctly on the hypervisor machine and on the base system where Satellite Server is running.

- If the web UI browser is running on separate system to **virt-manager**, proceed to Procedure 9.11, “To Configure SSH Access to Libvirt:”.

- If the web UI browser is running on the same system as **virt-manager**, complete the following procedure to add a Libvirt compute resource.

Procedure 9.10. To Add a Libvirt Compute Resource:

1. Navigate to **Infrastructure → Compute resources**.

2. Click **New Compute Resource**. On the **Compute Resource** tab, specify the following settings:

    a. Specify a **Name** and from the **Provider** drop-down menu, select Libvirt as the type for the Compute Resource. Optionally, insert a **Description**.

    b. In the **URL** field, enter a string as follows:

    ```
    qemu:///system
    ```

    c. From the **Display Type** drop-down menu, select **VNC**.

    d. Optionally select the **Console passwords** check box if this compute resource will only be used for new Libvirt guests. This option cannot be used together with previously configured Libvirt guests.

    e. Click **Test Connection**. If the connection is successful the button turns green.

    f. Click **Submit** to save the configuration.

    g. Select the **Virtual Machines** tab. Previously configured Libvirt guests will be shown.

Procedure 9.11. To Configure SSH Access to Libvirt:

Perform the following steps on the system where Red Hat Satellite is running unless otherwise directed.

1. Ensure the SSH file permissions and SELinux context for the **`foreman`** user are correct:

    ```
    # ls -Zd /usr/share/foreman/.ssh
    drwx------. foreman foreman system_u:object_r:ssh_home_t:s0
    /usr/share/foreman/.ssh
    ```

2. Create SSH keys for the **`foreman`** user:

    ```
    # su - foreman -s /bin/bash
    -bash-4.2$ ssh-keygen
    Generating public/private rsa key pair.
    Enter file in which to save the key
    (/usr/share/foreman/.ssh/id_rsa):
    Enter passphrase (empty for no passphrase):
    Enter same passphrase again:
    Your identification has been saved in
    /usr/share/foreman/.ssh/id_rsa.
    Your public key has been saved in
    /usr/share/foreman/.ssh/id_rsa.pub.
    The key fingerprint is:
    07:47:a9:23:d2:fe:2f:07:fb:55:75:46:3e:8e:6e:69
    foreman@satellite.example.com
    The key's randomart image is:
    +--[ RSA 2048]----+
    |          ..    .|
    |         ..    o |
    |     .  ...    .=|
    |    . o oo    ooo|
    |     o .S..  ... |
    |      . ..  ...  |
    |       . o  .E   |
    |        + ..o    |
    |         =o      |
    +-----------------+
    ```

3. Copy the **SSH** public key to the remote hypervisor system. For example, if your Libvirt host is *kvm.example.com*:

    ```
    -bash-4.2$ ssh-copy-id root@kvm.example.com
    The authenticity of host 'kvm.example.com (192.168.1.2)' can't be
    established.
    ECDSA key fingerprint is
    78:79:41:d0:b8:40:d5:4a:6d:7f:22:03:bd:cd:a0:dd.
    Are you sure you want to continue connecting (yes/no)? yes
    /usr/bin/ssh-copy-id: INFO: attempting to log in with the new
    key(s), to filter out any that are already installed
    /usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if
    you are prompted now it is to install the new keys
    foreman@192.168.1.2's password:

    Number of key(s) added: 1
    ```

```
Now try logging into the machine, with:   "ssh
'root@kvm.example.com'"
and check to make sure that only the key(s) you wanted were added.
```

4. Make an **SSH** connection to the remote system to confirm that no password prompt appears:

   ```
   -bash-4.2$ ssh root@kvm.example.com
   ```

   You should **not** be prompted for the password. The public key can be found in the **.ssh/authorized_keys** file on the remote system. Exit after successfully logging in and checking the keys:

   ```
   -bash-4.2$ exit
   ```

5. In the web UI, navigate to **Infrastructure** → **Compute resources** and click **New Compute Resource**.

6. In the **Name** field enter a suitable name.

7. From the **Provider** drop-down menu, select **Libvirt**.

8. In the **URL** field, enter a string in the following format:

   ```
   qemu+ssh://root@kvm.example.com/system
   ```

   Where *kvm.example.com* is the FQDN of your Libvirt host.

9. From the **Display Type** drop-down menu, select **VNC**.

10. Optionally select the **Console passwords** check box if this compute resource will only be used for new Libvirt guests. This option cannot be used together with previously configured Libvirt guests.

11. Click **Test Connection**. If the connection is successful the button turns green.

12. Click **Submit** to save the configuration.

13. Select the **Virtual Machines** tab. Previously configured Libvirt guests will be shown.

Procedure 9.12. To Configure the noVNC Console:

Prerequisites:

- SSH keys must be configured for the **foreman** user on the Satellite Server's base system (as explained previously).

- Existing Libvirt guests must be configured to use VNC server as the display type, the port settings set to **Auto**, and no VNC password selected.

1. On the hypervisor host system, configure the firewall to allow **VNC** service on ports **5900 to 5930**:

   - On Red Hat Enterprise Linux 6:

```
# iptables -A INPUT -p tcp --dport 5900:5930 -j ACCEPT
# service iptables save
```

- On Red Hat Enterprise Linux 7:

```
# firewall-cmd --add-port=5900-5930/tcp
# firewall-cmd --add-port=5900-5930/tcp --permanent
```

2. In the browser used for the web UI, trust the Satellite Server certificate as follows:

   a. Visit the public downloads page of the Satellite Server, for example **`https://satellite.example.com/pub/`**, and click the certificate file**`katello-server-ca.crt`**.

   b. Select to trust the certificate for identifying websites.

3. In the browser used for the web UI, disable *HTTP strict transport security* (HSTS). HSTS is described in RFC 6797. For example, in Firefox, enter**`About:Config`** in the browser address bar and set the following boolean to **`True`**:

```
network.websocket.allowInsecureFromHTTPS
```

4. Ensure you are using the FQDN in the browser for the web UI. NoVNC will not work if the domain name in the URL does not match the CN value in the certificate, which should be the same as the FQDN. Use a command as follows to check the CN value:

```
# openssl x509 -text -noout -in /etc/pki/katello/certs/katello-
apache.crt | grep CN
Issuer: C=US, ST=North Carolina, L=Raleigh, O=Katello,
OU=SomeOrgUnit, CN=satellite.example.com
Subject: C=US, ST=North Carolina, O=Katello, OU=SomeOrgUnit,
CN=satellite.example.com
output truncated
```

5. Navigate to **Infrastructure**+**Compute Resources**. Select the name of a Libvirt resource. On the **Virtual Machines** tab, select the name of a Libvirt guest. Ensure the machine is powered on and then select **Console**. The console window appears after the noVNC handshake completes.

9.3.6. Hardware Models

Hardware models help run unattended installations on systems based on the Scalable Processor Architecture (SPARC).

Procedure 9.13. To Create a Hardware Model:

1. Click **Hosts** → **Hardware Models**.

2. Click **`New Model`**.

3. Specify a **`Name`** for the Hardware Model.

4. For SPARC builds, insert the CPU **`Hardware model`** and **`Vendor class`**. Other architectures do not require values in these fields.

5. Type a description of the Hardware Model in the **`Information`** field.

6. Click **`Submit`**.

9.3.7. Installation Media

Red Hat Satellite uses installation media (ISO images) as content for kickstart trees and new host installations.

Procedure 9.14. To Add an Installation Medium:

1. Click **Hosts** → **Installation Media**.

2. Click **`New Medium`**. On the **`Medium`** tab, specify the following settings:

   a. Type a **`Name`** for the Installation Media. This setting is required.

   b. Type a **`Path`** to the Installation Medium. Options include either a URL or a valid NFS server. This setting is required.

   c. Select an **`Operating System Family`** to define the type of the Installation Medium.

3. On the **`Locations`** tab, select the desired locations to add them to the **`Selected Items`** list.

4. On the **`Organizations`** tab, select the desired organizations to add them to the **`Selected Items`** list.

   IMPORTANT

   Ensure that the Locations and Organizations are configured as they will help with future debugging.

5. Click **`Submit`**.

9.3.8. Partition Tables

Partition tables define the partitions and file system layout for new installations when provisioning systems. Satellite users specify the host's disk layout as an explicit sequence of partitions or use a dynamic disk layout script.

Procedure 9.15. To Create a Partition Table:

1. Click **Hosts** → **Partition Tables**.

2. Click **`New Partition Table`**.

3. Type a **`Name`** for the partition table.

4. Optionally select **`Default`**. This check box defines if the partition is automatically associated with new organizations or locations.

5. Optionally select **`Snippet`**. This check box defines if the partition is a reusable snippet for other partition table layouts.

6. Select the operating system from the **`Operating system family`** drop-down list.

7. Specify the Layout of the partition table. You can enter the layout in the text area under **`Template editor`** or click **`Choose File`** to upload a template file.

   NOTE

   The format of the layout must match that for the intended operating system. For example, Red Hat Enterprise Linux 7.2 requires a layout that matches a kickstart file.

8. Use the **`Audit Comment`** field to add a summary of changes to the partition layout.

9. Click **`Submit`**.

New partition table has to be associated with an operating system as described in Section 9.3.11, “Operating Systems”

9.3.9. Provisioning Templates

Provisioning templates provide the systematic means to run unattended installations. Provisioning templates can be executed via several methods including bash scripts, kickstart scripts, and PXE-based installations.

Procedure 9.16. To Create a Provisioning Template:

1. Click **Hosts** → **Provisioning Templates**.

2. Click **`New Template`**. On the **`Provisioning Template`** tab, specify the following settings:

   a. Specify a **`Name`** for the template.

   b. Insert your template in the **`Template editor`** field. Alternatively, click **`Browse`** to upload the template. This replaces the content in the **`Template editor`** field with the content of your chosen file.

   c. Optionally, type a comment in the **`Audit Comment`** field. Satellite adds the comment to the template history to track changes. View the template history under the **`History`** tab.

3. On the **`Type`** tab, select **`Snippet`** to store the template code without defining it as particular script or template type, or select the type from the **`Type`** drop-down menu.

4. On the **`Association`** tab, select host groups, environments and operating systems to be associated with the template. Select the operating systems from the **`Applicable Operating Systems`** list. Click **`Add Combination`** and select a **`Hostgroup`** and **`Environment`** to limit the template's use. Note that associations are not available for templates of type snippet.

5. On the **`Association`** tab, you can view the history of existing templates. No history is available when creating a new template.

6. On the **`Locations`** tab, select locations for the template.

7. On the **`Organizations`** tab, select organizations for the template.

IMPORTANT

Ensure that the Locations and Organizations are configured as they will help with future debugging.

8. Click **`Submit`**.

For more information on provisioning templates, see Creating Provisioning Templates in the *Red Hat Satellite Provisioning Guide*.

9.3.10. Configuring gPXE to Reduce Provisioning Times

To reduce provisioning time when downloading PXE boot files, gPXE enables the use of additional protocols such as **`HTTP`** to reduce download time. To make use of gPXE, proceed as follows:

- On systems configured to be a **`TFTP`** server, copy **`/usr/share/syslinux/gpxelinuxk.0`** to **`/var/lib/tftpboot`**.
- In the **`PXE Handoff`** section of **`/etc/dhcp/dhcpd.conf`**, change the **`DHCP filename`** option from **`pxelinux.0`** to **`gpxelinuxk.0`**.
- Create provisioning templates as follows and then assign them, together with the default template, to the operating systems.

Procedure 9.17. To Configure a gPXE Provisioning Template:

1. Click **Hosts** → **Provisioning templates**.
2. Find the template **Kickstart default PXELinux** and select **Clone**.
3. Enter a name, for example, **`Kickstart default gPXELinux`**.
4. In the Template editor, search and replace **`@initrd`** with **`@host.url_for_boot(:initrd)`**
5. In the Template editor, search and replace **`@kernel`** with **`@host.url_for_boot(:kernel)`**
6. Select the **Type** tab. From the **Type** drop-down menu, select **PXELinux**.
7. On the **`Association`** tab, select host groups, environments and operating systems to be associated with the template. Select the operating systems from the **`Applicable Operating Systems`** list. Click **`Add Combination`** and select a **`Hostgroup`** and **`Environment`** to limit the template's use.
8. Click **`Submit`**.

9.3.11. Operating Systems

Operating Systems define combinations of installation methods and media and are grouped within families. As a default, Red Hat Satellite uses a **`RedHat`** family. Families allow Satellite to change certain behaviors when provisioning hosts.

Procedure 9.18. To Add an Operating System:

1. Click **Hosts** → **Operating Systems**.

2. Click **`New Operating system`**. On the **`Operating System`** tab, specify the following settings:

   a. Type the **`Name`** of the Operating System and its **`Major Version`**. These settings are required.

   b. Optionally, define the **`Minor Version`**, select the **`OS Family`**, and add a **`Description`** of the operating system.

   c. Select a **`Root password hash`** (MD5, SHA256, of SHA512).

   d. Select the **`Architectures`** from the list of available Architectures. If none are available, create and assign them under **Hosts** → **Architectures** as described in Section 9.3.3, "Architectures".

3. On the **`Partition tables`** tab, select the applicable file system layouts from the list. For more information on creating partition tables, see Section 9.3.8, "Partition Tables".

4. On the **`Installation Media`** tab, select the applicable installation media from the list. For more information on adding installation media, see Section 9.3.7, "Installation Media".

5. On the **`Templates`** tab, you can assign provisioning templates when editing an existing operating system. This option is not available when creating a new operating system. For more information on creating provisioning templates, see Section 9.3.9, "Provisioning Templates".

6. On the **`Parameters`** tab, you can add parameters for the operating system.

7. Click **`Submit`**.

9.4. STORING AND MAINTAINING HOST INFORMATION

Red Hat Satellite 6 uses a combination of applications to gather information about managed hosts and to ensure that those hosts are maintained in the desired state. These applications include:

- Foreman: Provides for the provisioning and life cycle management of physical and virtual systems. Foreman automatically configures these systems using various methods, including kickstart and Puppet modules.

- Puppet: A client/server architecture for configuring hosts, consisting of the Puppet Master (server) and the Puppet Agent (client).

- Facter: Puppet's system inventory tool. Facter gathers basic information (facts) about hosts such as hardware details, network settings, OS type and version, IP addresses, MAC addresses, SSH keys, and more. These facts are then made available in Puppet manifests as variables.

The use of Puppet, Facter, and facts is discussed in more detail below.

9.4.1. The Puppet Architecture

Puppet usually runs in an agent/master (also known as a client/server) architecture, where a Puppet server controls important configuration information, and managed hosts (clients) request only their own configuration catalogs. Puppet configures hosts in two steps:

- It compiles a catalog
- It applies that catalog to the appropriate host

In the agent/master setup, the Puppet client sends facts gathered by Facter and other information to the Puppet Master. The Puppet Master compiles a catalog based on these facts, and then sends this catalog to the client. The client sends a report of all the changes it made, or would have made if the ***`--noop`*** parameter had been used, to the Puppet Master, which in turn sends the results to Foreman. This catalog describes the desired state for one specific host. It lists the resources to manage on that host, including any dependencies between those resources. The agent applies the catalog to the host.

This communication between master and agent occurs every 30 minutes by default. You can specify a different value in the **`/etc/puppet/puppet.conf`** file using the ***`runinterval`*** parameter. You can also run **`puppet agent apply`** to initiate communication manually.

9.4.2. Using Facter and Facts

Facter is Puppet's system inventory tool, and includes a large number of built-in facts. You can run Facter at the command line on a local host to display fact names and values. You can extend Facter with custom facts, and then use these to expose site-specific details of your hosts to your Puppet manifests. You can also use the facts provided by Facter to inform conditional expressions in Puppet.

Puppet determines a system state based on resources; for example, you can tell Puppet that the **`httpd`** service should always be running and Puppet knows how to handle that. If you are managing different operating systems, you can use the ***`osfamily`*** fact to create conditional expressions to tell Puppet which service to watch or which package to install. You can use the ***`operatingsystemmajrelease`*** and ***`versioncmp`*** parameters to create conditional expressions based on different versions of the same operating system. See Example 9.1, “Using Conditional Expressions with Facts” for an example of using conditional expressions.

Example 9.1. Using Conditional Expressions with Facts

```
if $:: osfamily == 'RedHat' {
  if $::operatingsystemmajrelease == '6' {
   $ntp_service_name = 'ntpd'
   }

  elseif versioncmp($::operatingsystemmajrelease, '7') >= 0 {
   $ntp_service_name = 'chrony'
   }
 }
```

NOTE

This example uses the expression "versioncmp($::operatingsystemmajrelease, '7') >= 0" to test for version 7 or later of Red Hat Enterprise Linux. Do not use the expression "$::operatingsystemmajrelease >= '7'" to perform this test. See https://docs.puppetlabs.com/references/latest/function.html#versioncmp for more information about this and other Puppet functions.

Puppet also sets other special variables that behave a lot like facts. See Special Variables Added by Puppet[3] and Core Facts[4] for more information.

9.4.2.1. Displaying Facts for a Particular Host

Puppet can access Facter's built-in core facts as well as any custom or external facts present in your Puppet modules. You can view available facts from the command line (**`facter -p`**) and also from the web UI (**Monitor** → **Facts**). You can browse the list of facts or use the **`Search`** box to search for specific facts. For example, type "**`facts.`**" to display a list of available facts.

NOTE

The list of available facts is very long. The UI only displays 20 facts at a time. The list of facts gradually filters as you enter more details. For example, type "facts.e" to display all facts that begin with the letter "e."

Procedure 9.19. To View Facts for a Particular Host:

1. On the main menu, click **Hosts** → **All Hosts** and then click the name of the host that you want to inspect.

2. In the **`Details`** pane, click **`Facts`** to display all known facts about the host.

NOTE

- For any fact listed on this page, you can click **`Chart`** to display a chart of the distribution of this fact name over all managed hosts.

- You can bookmark a search to make it easier to use in the future. When you have refined your search, click the drop-down arrow next to the **`Search`** button, and click **`Bookmark this search`**. Bookmarked searches appear in the **`Search`** drop-down list, and also under **Administer** → **Bookmarks** on the main menu.

9.4.2.2. Searching for Hosts based on Facts

You can use Facter information to search for specific hosts. This means that you can search for all hosts that match specific fact criteria, such as **`facts.architecture = x86_64`**.

Procedure 9.20. To Search for Hosts Based on Facts:

1. On the main menu, click **Monitor** → **Facts** to display the **`Fact Values`** page.

2. In the **`Search`** field, start typing the name of the fact that you want to filter by. You can search by specific name, name/value pairs, and so on.

3. Click **Search** to retrieve the list of matching hosts.

9.4.2.3. Custom Fact Reporting

Obtaining custom information from managed hosts is fully supported with Red Hat Satellite 6. This section illustrates using a Puppet module obtained from Puppet Forge, but the principle applies equally for other sources of Puppet modules.

The number of facts reported via the standard Facter interface can be extended. For example, to gather a fact for use as a variable in modules. If a fact that describes the packages installed was available, you could search this data and make informed configuration management decisions based on the information.

To obtain a report on the packages installed on a host the process is as follows:

- The manifest **pkginventory** is obtained from Puppet Forge and saved to the base system.
- The Puppet module is added to a content view and then this is promoted to a system and deployed to that system.
- The facts for the system are then queried using a package name. In this example, for a host called *hostname* and using a Satellite user with credentials*username* and *password*, the following API query would return the facts that matched the search string "bash":

  ```
  curl -u username:password -X GET
  http://localhost/api/hosts/:hostname/facts?search=bash
  {"hostname":{"pkg_bash":"4.2.45-5.el7_0.4"}}
  ```

  The search returns the package version. This could then be used to populate an external database.

Adding the pkginventory Puppet Module

To add the **pkginventory** Puppet module to the Red Hat Satellite Server application, download the module from https://forge.puppetlabs.com/ody/pkginventory to the base system where the Satellite Server application is installed and then follow the procedures below.

Puppet modules are usually stored in a custom repository named Puppet Modules. The following procedure assumes you have made a custom repository with that name. If you have not yet made a custom repository for Puppet Modules, see Creating Custom Products and Enabling Repositories in the *Red Hat Satellite Quick Start Guide*.

Procedure 9.21. To Upload a Puppet Module to a Repository:

1. Download the Puppet module to the base system. Modules that are downloaded will have a **.tar.gz** extension.
2. Click **Content** → **Products** and then click the product name in the**Name** field associated with the Puppet module repository. For example, **Custom Products**.
3. On the **Repositories** tab, select the Puppet Modules repository you want to modify. For example, **Puppet Modules**.

4. In the **`Upload Puppet Module`** section, click **`Browse`**, and navigate to the module that you downloaded.

5. Click **`Upload`**.

To distribute a Puppet module to clients, content hosts, the module must be applied to a Content View and published. Follow this procedure to add a module to a Content View.

Procedure 9.22. To Add a Module to a Content View:

1. Click **Content** → **Content Views** and then select a content view from the**Name** menu.

2. On the **`Puppet Modules`** tab, click **`Add New Module`**. A list of installed modules appears.

3. From the **`Actions`** column, click **`Select a Version`** to select the module you want to add. A table of available versions appears.

4. Click **`Select Version`** next to the version of the module that you want to add.

5. Click **`Publish New Version`** to create the new Content View.

6. Optionally add a description and click **`Save`**.

[3] https://docs.puppetlabs.com/puppet/3.7/reference/lang_facts_and_builtin_vars.html#special-variables-added-by-puppet

[4] https://docs.puppetlabs.com/facter/latest/core_facts.html

CHAPTER 10. CONFIGURING HOSTS

In Red Hat Satellite, hosts are client systems which have Red Hat Subscription Manager installed. Red Hat Subscription Manager sends updates to Red Hat Satellite and Red Hat Satellite provides updates to these client systems.

Hosts must be registered in order to be managed. After a host has been registered, it can be viewed and edited in the **`Hosts`** tab. This enables a user to add and manage subscriptions, add and remove software packages, and apply updates.

10.1. BROWSING HOSTS

The Satellite Server web UI provides an opportunity to browse all hosts recognized by the Satellite Server. Navigate to **`Hosts`** tab at the top of the screen to open the drop-down menu with the following items:

- **`All Hosts`** - a list of all hosts recognized by the Satellite Server.
- **`Discovered Hosts`** - a list of bare-metal hosts detected on the provisioning network by the Discovery plug-in.
- **`Content Hosts`** - a list of hosts which manage tasks related to content and subscriptions.
- **`Host Collections`** - a list of user-defined collections of hosts used for bulk actions such as Errata Installation.

To search for a host, type in the **`Search`** field, and use an asterisk (*) to perform a partial string search. For example, if searching for a content host named **`dev-node.example.com`**, click the **`Content Hosts`** page, type **`dev-node*`** in the **`Search`** field. Alternatively, **`*node*`** will also find the content host **`dev-node.example.com.`**

10.2. HOST STATUS TYPES

Each host recognized by the Satellite Server is assigned a status type in accordance with the most recent action performed on or upcoming changes to be applied to that host. Navigate to **`Hosts → All hosts`** to view the status of each host. The following table outlines the status types to which hosts can be assigned:

Table 10.1. Host Status Types

Icon	Status	Description
⦿	Error	An error has been detected on the host. If you hover the mouse over the error icon, a tooltip showing the actual reason of the error will appear. You can see a more detailed report of issues by clicking on the host.
ⓘ	Warning	The host has been configured, but no reports have been collected for that host over the last reporting interval.
✓	OK	There are no pending actions on the host, no pending changes, and no errors over the last reporting interval.

10.3. HOST OVERVIEW

The host overview page displays information about a given host and the connection between the host and the installer. To view the host overview page, select **`Hosts → All hosts`**, then click the name of a host.

Details

The details bar contains a row of buttons that provide shortcuts to more information about the host, and tabs that display summaries of important details and events.

- **`Audits`**: a page containing audit entries for the current host.
- **`Facts`**: a page containing a list of facts for the current host. This button is only available after the installer has collected facts from the host.
- **`Reports`**: a page containing a list of reports for the current host. This button is only available after the installer has collected reports from the host.
- **`YAML`**: a page containing details about the host in YAML format, such as its IP address, MAC address, name, and values of parameters that have been applied to the host.
- **`Properties`**: a list of general details about the host, such as its IP address, MAC address, and the operating system entry that has been applied to the host.
- **`Metrics`**: a table showing a summary of all events reported for the host.
- **`Templates`**: a list of all provisioning templates currently accessible by the host. The provisioning templates include in this list are automatically configured in accordance with the operating system entry applied to the host.
- **`NICs`**: a table showing detailed information on NICs configured for the host.

Host Actions

Click each of these buttons to perform common actions on the host.

- **`Run Job`**: allows running jobs on the host. For more information on running jobs see Chapter 12, *Running Jobs on Satellite Hosts*.
- **`Boot disk`**: a menu that allows you to select the boot disk for the host. For more information on creating a boot ISO for a host see Creating New Hosts with PXE-less Provisioning in the *Red Hat Satellite Provisioning Guide*.
- **`Edit`**: opens the host details page which allows you to configure settings for the host. Note that the installer configures all the settings automatically and normally no manual configurations are required.
- **`Build`**: flags the host to be provisioned on the next host boot. Note that the installer manages all aspects of the provisioning process and normally there is no need to provision hosts manually.
- **`Delete`**: deletes the host from the user interface.

Host Graphs

The host overview page contains two graphs that display the status of recent Puppet runs executed on the host.

- **Runtime**: tracks two data points: **Config Retrieval** and **Runtime**. The **Config Retrieval** data point represents the amount of time taken to collect information about the host during a given Puppet run, and the **Runtime** data point represents the amount of time required to execute the Puppet run. Both data points are measured in seconds.

- **Resources**: tracks the number of actions performed on the host during a Puppet run. The categories displayed in this graph are identical to those displayed in the **Reports** page, and are measured using the number of actions in each category.

10.4. CREATING A HOST

The following procedure describes how to create a host in Red Hat Satellite.

Procedure 10.1. To Create a Host:

1. Click **Hosts** → **New Host**.

2. On the **Host** tab, enter the required details.

3. On the **Puppet Classes** tab, select the Puppet classes you want to include.

4. On the **Interfaces** tab:

   a. For each interface, click **Edit** in the **Actions** column and configure the following settings as required:

      - **Type** — For a Bond or BMC interface, use the**Type** list and select the interface type.

      - **MAC address** — Enter the MAC address.

      - **Identifier** — Enter the device name as a device identifier.

      - **DNS name** — Enter the DNS name that is known to the DNS server. This is used for the host part of the FQDN.

      - **Domain** — Select the domain name of the provisioning network. This automatically updates the **Subnet** list with a selection of suitable subnets.

      - **Subnet** — Select the subnet for the host from the list.

      - **IP address** — If there is a DHCP-enabled Capsule Server on the selected subnet, the IP address is automatically suggested. If required, click **Suggest new** to generate a different address. Alternatively, you can enter an IP address. The address can be omitted if provisioning tokens are enabled, if the domain does not mange DNS, if the subnet does not manage reverse DNS, or if the subnet does not manage DHCP reservations.

      - **Managed** — Select this check box to configure the interface during provisioning to use the Capsule provided DHCP and DNS services.

      - **Primary** — Select this check box to use the DNS name from this interface as the host portion of the FQDN.

      - **Provision** — Select this check box to use this interface for provisioning. This means TFTP boot will take place using this interface, or in case of image based provisioning, the script to complete the provisioning will be executed through this interface. Note that many provisioning tasks, such as downloading RPMs by **anaconda**, Puppet setup in a `%post` script, will use the primary interface.

- **`Virtual NIC`** — Select this check box if this interface is not a physical device. This setting has two options:
  - **`Tag`** — Optionally set a VLAN tag. If unset, the tag will be the VLAN ID of the subnet.
  - **`Attached to`** — Enter the device name of the interface this virtual interface is attached to.

b. Click **`OK`** to save the interface configuration.

c. Optionally, click **`Add Interface`** to include an additional network interface. See Section 10.11, "Configuring an Additional Network Interface" for details.

d. Press **`Submit`** to apply the changes and exit.

5. On the **`Operating System`** tab, enter the required details. You can select a partition table from the drop-down list or enter a custom partition table in the **`Custom partition table`** field. You cannot specify both.

6. On the **`Parameters`** tab, click **`Add Parameter`** to add any required parameters. This includes all Puppet Class Parameters and Host Parameters associated with the host.

7. On the **`Additional Information`** tab, enter additional information about the host.

8. Click **`Submit`** to complete your provisioning request.

10.5. REGISTRATION

This section shows you how to register hosts to Satellite Server or Capsule Server. There are two main methods for registering a host:

- Download and install the consumer RPM (*server.example.com*/pub/katello-ca-consumer-latest.noarch.rpm) and then run subscription manager. This method is suited for freshly installed hosts. See Section 10.5.1, "Configuring a Host for Registration" and Section 10.5.2, "Registering a Host" for more information.

- Download and run the bootstrap script (*server.example.com*/pub/bootstrap.py). This method is suited for both freshly installed hosts and hosts that have been previously registered, for example, to Satellite 5 or another Satellite 6. See Section 10.5.5, "Registering Hosts to Satellite 6 Using The Bootstrap Script" for more information.

Hosts registered to the Satellite Server via Red Hat Subscription Manager, which can occur either during the post phase of a kickstart or through the terminal, will appear on the **Content Hosts** page accessible through **Hosts** → **Content Hosts**. Hosts provisioned by Satellite Server appear on the **Hosts** page accessible through **Hosts** → **All hosts**.

10.5.1. Configuring a Host for Registration

Red Hat Enterprise Linux hosts register to the Customer Portal Subscription Management by default. You must update each host configuration so that they receive updates from the correct Satellite Server or Capsule Server.

Prerequisites

- Hosts must be using the following Red Hat Enterprise Linux version:
  - 5.7 or later
  - 6.4 or later
  - 7.0 or later
- All architectures of Red Hat Enterprise Linux are supported (i386, x86_64, s390x, ppc_64).
- Ensure that the Satellite Servers, any Capsule Servers, and all hosts are synchronized with the same NTP server.
- Ensure that a time synchronization tool is up and running on the Satellite Servers, any Capsule Servers, and the hosts.
  - For Red Hat Enterprise Linux 6:

    ```
    # chkconfig ntpd on; service ntpd start
    ```

  - For Red Hat Enterprise Linux 7:

    ```
    # systemctl start chronyd; systemctl enable chronyd
    ```

- Ensure that the daemon **`rhsmcertd`** is running on the hosts.
  - For Red Hat Enterprise Linux 6:

    ```
    # service rhsmcertd start
    ```

  - For Red Hat Enterprise Linux 7:

    ```
    # systemctl start rhsmcertd
    ```

The following procedure shows how to configure your host to register to Red Hat Satellite.

Procedure 10.2. To Configure a Host for Registration:

1. Take note of the fully qualified domain name (FQDN) of the Satellite Server or Capsule Server, for example *server.example.com*.
2. On the host, open a terminal and log in as root.
3. Install the consumer RPM from the Satellite Server or Capsule Server to which the host is to be registered. The consumer RPM updates the content source location of the host and allows the host to download content from the content source specified in Red Hat Satellite.
   -

```
# rpm -Uvh http://server.example.com/pub/katello-ca-consumer-
latest.noarch.rpm
```

IMPORTANT

Any running Docker Daemons will be restarted.

NOTE

katello-ca-consumer-*hostname*-1.0-1.noarch.rpm is an additional katello-ca-consumer RPM available that contains the server's host name. The katello-ca-consumer-latest.noarch.rpm rpm will always reflect the most updated version. Both serve the same purpose.

10.5.2. Registering a Host

Prerequisites

- Complete all steps in Section 10.5.1, “Configuring a Host for Registration”.
- Ensure that an activation key associated with the appropriate content view and environment exists for the host. If not, see Chapter 7, *Configuring Activation Keys* for more information. By default, an activation key has the auto-attach function enabled. The feature is commonly used with hosts used as hypervisors.
- Ensure that the version of the **`subscription-manager`** utility installed is 1.10 or higher. The package is available in the standard Red Hat Enterprise Linux repository.

Procedure 10.3. To Register Hosts:

1. On the host, open a terminal and log in as root.
2. Clear any old host data related to Red Hat Subscription Manager (RHSM):

   ```
   # subscription-manager clean
   ```

3. Register the host using RHSM:

   ```
   # subscription-manager register --org your_org_name --activationkey
   your_activation_key
   ```

   Example 10.1. Command Output after Registration:

   ```
   # subscription-manager register --org MyOrg --activationkey
   TestKey-1
   The system has been registered with id: 62edc0f8-855b-4184-b1b8-
   72a9dc793b96
   ```

NOTE

You can use the **`--environment`** option to override the content view and life cycle environment defined by the activation key. For example, to register a host to the content view "MyView" in a "Development" life cycle environment:

```
 # subscription-manager register --org your_org_name --
environment Development/MyView --activationkey
your_activation_key
```

NOTE

For Red Hat Enterprise Linux 6.3 hosts, the release version defaults to Red Hat Enterprise Linux 6 Server and needs to be pointed to the 6.3 repository.

Procedure 10.4. To Point Red Hat Enterprise Linux 6.3 to the Repository:

1. On Red Hat Satellite, select **Hosts** → **Content Hosts**.
2. Click the name of the host that needs to be changed.
3. In the **`Content Host Content`** section click the edit icon to the right of **`Release Version`**.
4. Select "6.3" from the **Release Version** drop-down menu.
5. Click **Save**.

10.5.3. Installing the Katello Agent

The following procedure shows how to install the Katello agent on a host registered to Satellite 6. The katello-agent package depends on the gofer package that provides the **`goferd`** service. This service must be enabled so that the Red Hat Satellite Server or Capsule Server can provide information about errata that are applicable for content hosts.

Prerequisites

Satellite version 6.1 and later require that you enable the **`Satellite Tools`** repository. The **`Red Hat Common`** repositories are no longer used and are not compatible with Satellite version 6.1 and later.

The **`Satellite Tools`** repository must be enabled, synchronized to the Red Hat Satellite Server and made available to your hosts as it provides the required packages.

Procedure 10.5. To Verify the Satellite Tools Repository is Enabled:

1. Open the Satellite web UI, navigate to **Content** → **Red Hat Repositories** and click on the **RPMs** tab.
2. Find and expand the **`Red Hat Enterprise Linux Server`** item.
3. Find and expand the **`Red Hat Satellite Tools 6.2 (for RHEL`** ***`VERSION`*** **`Server) (RPMs)`** item.

   If the **`Red Hat Satellite Tools 6.2`** items are not visible, it may be because they are not included in the subscription manifest obtained from the Customer Portal. To correct that, log in to the Customer Portal, add these repositories, download the subscription manifest and import it into Satellite.

4. Ensure the **`Enabled`** check box beside the repository's name is selected. If not, select it.

Enable the **`Satellite Tools`** repository for every supported major version of Red Hat Enterprise Linux running on your hosts.

Procedure 10.6. To Install Katello Agent:

1. On the host, verify that the **`satellite-tools`** repository is enabled. If you registered the host using an activation key with auto-attache enabled, the repository is enabled automatically already.

   ```
   # yum repolist enabled | grep -i satellite-tools
   ```

   If the **`satellite-tools`** is not enabled, enable it using the following command:

   ```
   # subscription-manager repos --enable=rhel-version-server-satellite-
   tools-6.2-rpms
   ```

2. Install the **`katello-agent`** RPM package using the following command:

   ```
   # yum install katello-agent
   ```

3. Ensure the **`goferd`** service is running.

   - On Red Hat Enterprise Linux 6, enter the following command:

     ```
     # service goferd start
     ```

   - On Red Hat Enterprise Linux 7, enter the following command:

     ```
     # systemctl start goferd
     ```

10.5.4. Installing and Configuring the Puppet Agent

This section describes how to install and configure the Puppet agent on a host. When you have correctly installed and configured the Puppet agent, you can navigate to **Hosts** → **All hosts** to list all hosts visible to Red Hat Satellite Server.

Prerequisites

The **`Satellite Tools`** repository must be enabled, synchronized to the Red Hat Satellite Server and made available to your hosts as it provides the required packages.

Procedure 10.7. To Verify the Satellite Tools Repository is Enabled:

1. Open the Satellite web UI, navigate to **Content** → **Red Hat Repositories** and click on the **`RPMs`** tab.

2. Find and expand the **`Red Hat Enterprise Linux Server`** item.

3. Find and expand the **Red Hat Satellite Tools 6.2 (for RHEL *VERSION* Server) (RPMs)** item.

   If the **Red Hat Satellite Tools 6.2** items are not visible, it may be because they are not included in the subscription manifest obtained from the Customer Portal. To correct that, log in to the Customer Portal, add these repositories, download the subscription manifest and import it into Satellite.

4. Ensure the **Enabled** check box beside the repository's name is selected. If not, select it.

Procedure 10.8. To Install and Enable the Puppet Agent:

1. On the host, open a terminal console and log in as the **root** user.

2. Verify that the **satellite-tools** repository is enabled, using the following command:

   ```
   # yum repolist enabled | grep -i satellite-tools
   ```

   If the **satellite-tools** is not enabled, enable it using the following command:

   ```
   # subscription-manager repos --enable=rhel-version-server-satellite-
   tools-6.2-rpms
   ```

3. Install the Puppet agent RPM package using the following command:

   ```
   # yum install puppet
   ```

4. Configure the puppet agent to start at boot:

   - On Red Hat Enterprise Linux 6:

     ```
     # chkconfig puppet on
     ```

   - On Red Hat Enterprise Linux 7:

     ```
     # systemctl enable puppet
     ```

Prerequisites

The following conditions must be met before configuring the Puppet Agent:

- The host must be registered to the Red Hat Satellite Server.
- The Satellite Tools repository must be enabled.
- Puppet packages must be installed on the host.

Procedure 10.9. To Configure the Puppet Agent:

1. Configure the Puppet agent by specifying the server and environment settings in the **/etc/puppet/puppet.conf** file:

```
# vi /etc/puppet/puppet.conf
```

```
[main]
    # The Puppet log directory.
    # The default value is '$vardir/log'.
    logdir = /var/log/puppet

    # Where Puppet PID files are kept.
    # The default value is '$vardir/run'.
    rundir = /var/run/puppet

    # Where SSL certificates are kept.
    # The default value is '$confdir/ssl'.
    ssldir = /var/lib/puppet/ssl

...

[agent]
    # The file in which puppetd stores a list of the classes
    # associated with the retrieved configuratiion.  Can be loaded
in
    # the separate ``puppet`` executable using the ``--loadclasses``
    # option.
    # The default value is '$confdir/classes.txt'.
    classfile = $vardir/classes.txt
    pluginsync = true
    report = true
    ignoreschedules = true
    daemon = false
    ca_server = satellite.example.com
    server = satellite.example.com
    environment = KT_Example_Org_Library_RHEL6Server_3

    # Where puppetd caches the local configuration.  An
    # extension indicating the cache format is added automatically.
    # The default value is '$confdir/localconfig'.
    localconfig = $vardir/localconfig

...
```

IMPORTANT

Set the **`environment`** parameter to the name of the Puppet environment to which the host belongs. A Puppet environment is a collection of Puppet modules that can be associated with a host or a host group.

- To find the host's Puppet environment, navigate to **Hosts** → **All Hosts** and inspect the **`Environment`** column in the host table.
- To assign a Puppet environment to a host, navigate to **Hosts** → **All Hosts** and click **`Edit`** next to the selected host.
- To list Puppet environments enabled on the Satellite Server, navigate to **Configure** → **Environments**. You can also inspect the **`/etc/puppet/environments/`** directory on the Satellite Server to find what Puppet modules and manifests are associated with Puppet environments.

For more information see the Red Hat Satellite Puppet Guide.

2. Run the Puppet agent on the host:

```
# puppet agent -t --server satellite.example.com
```

3. Sign the SSL certificate for the Puppet client through the Satellite Server web UI:

   a. Log in to the Satellite Server through the web UI.

   b. Select **Infrastructure** → **Capsules**.

   c. Select **`Certificates`** from the drop-down menu to the right of the required Capsule.

   d. Click **`Sign`** to the right of the required host.

   e. Enter the **`puppet agent`** command again:

   ```
   # puppet agent -t --server satellite.example.com
   ```

NOTE

When the Puppet agent is configured on the host it will be listed under **`All Hosts`** but only when **`Any Organization`** is selected as the host will not be assigned to an organization or location. To assign the host to an organization, see Section 10.9, "Assigning a Host to a Specific Organization" or to assign the host to a location, see Section 10.10, "Assigning a Host to a Specific Location".

10.5.5. Registering Hosts to Satellite 6 Using The Bootstrap Script

The bootstrap script, which is included in 6.2 and higher, can be used to register new hosts or migrate existing hosts to Satellite 6.

The bootstrap script handles content registration, product certificates, and Puppet configuration. The bootstrap script has the advantage of taking a Red Hat Enterprise Linux system, regardless of where it is registered (RHN, Satellite 5, SAM, RHSM), or if it is registered at all, and subscribing it to Satellite 6.

The bootstrap script package, katello-client-bootstrap, is installed by default on the Satellite Server's base system and the script itself is installed in the **`/var/www/html/pub/`** directory to make it available to hosts. It can be accessed using a URL in the following form:

```
satellite6.example.com/pub/bootstrap.py
```

The script includes documentation in a readme file. To view the file on the Satellite CLI:

```
$ less /usr/share/doc/katello-client-bootstrap-version/README.md
```

Procedure 10.10. Installing the Bootstrap Script on the Host:

As the script is only required once, and only for the **`root`** user, you can place it in**`/root`** and remove it after use, or place it in **`/usr/local/sbin`**. This example will use**`/root`**.

As **`root`**, install the bootstrap script on the host as follows:

1. Ensure you are in the correct directory. For example, to change to **`/root`**:

   ```
   # cd
   ```

2. Download the script:

   ```
   # wget http://satellite6.example.com/pub/bootstrap.py
   ```

   This will install the script to the current directory.

3. Make the script executable:

   ```
   # chmod +x bootstrap.py
   ```

4. To confirm that the script can now be run, view the usage statement as follows:

   ```
   # ./bootstrap.py -h
   ```

5. Optionally, when the transition process is complete, remove the script:

   ```
   # cd
   # rm bootstrap.py
   ```

Procedure 10.11. Running the Bootstrap Script

Prerequisites

- The bootstrap script is installed as described previously.
- You have an activation key for your desired hosts. For configuring activation keys, see Chapter 7, *Configuring Activation Keys*.

- You have created a host group. For creating host groups, see Section 9.1, “Creating a Host Group”.

1. Enter the bootstrap command as follows with values suitable for your environment.

   For the **`--server`** option, specify the FQDN name of Satellite Server or Capsule Server. For **`--location`**, **`--organization`**, and **`--hostgroup`** options, use quoted names, not labels, as arguments to the options. See Section 10.5.6, “Advanced Bootstrap Script Configuration” for advanced use cases.

   ```
   # bootstrap.py --login=admin \
   --server satellite6.example.com \
   --location="Example Location" \
   --organization="Example Organization" \
   --hostgroup="Example Host Group" \
   --activationkey=activation_key
   ```

   The script will prompt you for the password corresponding to the Satellite user name you entered with the **`--login`** option.

2. The script will run and send notices of progress to **`stdout`**. Watch for output prompting you to approve the certificate. For example:

   ```
   [NOTIFICATION], [2016-04-26 10:16:00], [Visit the UI and approve
   this certificate via Infrastructure->Capsules]
   [NOTIFICATION], [2016-04-26 10:16:00], [if auto-signing is disabled]
   [RUNNING], [2016-04-26 10:16:00], [/usr/bin/puppet agent --test --
   noop --tags no_such_tag --waitforcert 10]
   ```

   The host will wait indefinitely until an administrator approves the Puppet certificate.

   a. In the web UI, navigate to **Infrastructure** → **Capsules**.

   b. Select **Certificates** to the right of the name of the Capsule corresponding to the FQDN given with **`--server`** option.

   c. In the **Actions** column select **Sign** to approve the host's Puppet certificate.

   d. Return to the host to see the remainder of the bootstrap process completing.

3. In the web UI, navigate to **Hosts** → **All hosts** and ensure that the host is connected to the correct host group.

If the Katello agent is not installed on the host, proceed to Section 10.5.3, “Installing the Katello Agent”.

10.5.6. Advanced Bootstrap Script Configuration

The standard workflow of using the bootstrap script has been outlined in Procedure 10.11, “Running the Bootstrap Script”. This section covers some more examples.

Migrating a host from one Satellite 6 to another Satellite 6.

Use the script with **`--force`**, and the script will remove the katello-ca-consumer-* packages from the old Satellite and install the katello-ca-consumer-* packages from the new Satellite. For example:

-

```
# bootstrap.py --login=admin \
--server satellite6.example.com \
--location="Example Location" \
--organization="Example Organization" \
--hostgroup="Example Host Group" \
--activationkey=activation_key \
--force
```

Migrating a host from Red Hat Network (RHN) or Satellite 5 to Satellite 6.

The bootstrap script detects the presence of **`/etc/syconfig/rhn/systemid`** and a valid connection to RHN as an indicator that the system is registered to a legacy platform. The script then calls **`rhn-classic-migrate-to-rhsm`** to migrate the system from RHN. By default, the script does not delete the system's legacy profile due to auditing reasons. To remove the legacy profile, use **`--legacy-purge`** and use **`--legacy-login`** to supply an user account that has appropriate permissions to remove a profile. Enter the user account password when prompted. For example:

```
# bootstrap.py --login=admin \
--server satellite6.example.com \
--location="Example Location" \
--organization="Example Organization" \
--hostgroup="Example Host Group" \
--activationkey=activation_key \
--legacy-purge \
--legacy-login rhn-user
```

Registering a host to Satellite 6, omitting Puppet setup.

By default, the bootstrap script configures the host for content management and configuration management. If you have an existing configuration management system and do not want to install puppet on the host, use **`--skip-puppet`**. For example:

```
# bootstrap.py --login=admin \
--server satellite6.example.com \
--location="Example Location" \
--organization="Example Organization" \
--hostgroup="Example Host Group" \
--activationkey=activation_key \
--skip-puppet
```

Registering a host to Satellite 6 for content management only.

To register a system as a content host, and leave out the provisioning and configuration management functions, use **`--skip-foreman`**. For example:

```
# bootstrap.py --server satellite6.example.com \
--organization="Example Organization" \
--activationkey=activation_key \
--skip-foreman
```

Changing the method the bootstrap script uses to download the consumer RPM.

By default, the bootstrap script uses HTTP to download the consumer RPM (*server.example.com*/pub/katello-ca-consumer-latest.noarch.rpm). In some environments, it is desired to only allow HTTPS between the host and Satellite. Use **`--`**

download-method to change the download method from HTTP to HTTPS. For example:

```
# bootstrap.py --login=admin \
--server satellite6.example.com \
--location="Example Location" \
--organization="Example Organization" \
--hostgroup="Example Host Group" \
--activationkey=activation_key \
--download-method https
```

Providing the host's IP address to Satellite

On hosts with multiple interfaces or multiple IP addresses on one interface, you may need to override the auto-detection of the IP address and provide a specific IP address to Satellite. Use **--ip**. For example:

```
# bootstrap.py --login=admin \
--server satellite6.example.com \
--location="Example Location" \
--organization="Example Organization" \
--hostgroup="Example Host Group" \
--activationkey=activation_key \
--ip 192.x.x.x
```

Enabling Remote Execution on the host.

Use **--rex** and **--rex-user** to enable remote execution and add the required SSH keys for the specified user. For example:

```
# bootstrap.py --login=admin \
--server satellite6.example.com \
--location="Example Location" \
--organization="Example Organization" \
--hostgroup="Example Host Group" \
--activationkey=activation_key \
--rex \
--rex-user root
```

Creating a domain for a host at registration time.

To create a host record, the DNS domain of a host needs to exist in Satellite prior to running script. If the domain does not exist, add it using **--add-domain**. For example:

```
# bootstrap.py --login=admin \
--server satellite6.example.com \
--location="Example Location" \
--organization="Example Organization" \
--hostgroup="Example Host Group" \
--activationkey=activation_key \
--add-domain
```

Providing an arbitrary Fully Qualified Domain Name (FQDN) for the host.

If the host's host name is not an FQDN, or is not RFC compliant (containing a character such as an underscore), the script will fail at the host name validation stage. Use **--fqdn** to specify the FQDN that will be reported to Satellite. To do so, you will need to set

`create_new_host_when_facts_are_uploaded` and **`create_new_host_when_report_is_uploaded`** to false using **`hammer`**. For example,

```
# hammer settings set \
--name  create_new_host_when_facts_are_uploaded \
--value false
# hammer settings set \
--name  create_new_host_when_report_is_uploaded \
--value false
```

```
# bootstrap.py --login=admin \
--server satellite6.example.com \
--location="Example Location" \
--organization="Example Organization" \
--hostgroup="Example Host Group" \
--activationkey=activation_key \
--fqdn node100.example.com
```

10.6. CHANGING THE GROUP OF A HOST

The following steps show you how to change the group of a host.

1. Navigate to **Hosts** → **All hosts**.
2. Select the check box of the host you want to change.
3. From the **`Select Action`** menu at the upper right of the screen, select **`Change Group`**. A new option window will open.
4. From the **`Select Action`** menu, select the desired group for your host.
5. Click **`Submit`**.

10.7. CHANGING THE ENVIRONMENT OF A HOST

The following steps show you how to change the environment of a host.

1. Navigate to **Hosts** → **All hosts**.
2. Select the check box of the host you want to change.
3. From the **`Select Action`** menu at the upper right of the screen, select **`Change Environment`**. A new option window will open.
4. From the **`Select Action`** menu, select the desired environment for your host.
5. Click **`Submit`**.

10.8. MANAGING HOSTS

The following procedure shows how to switch a host between Managed and Unmanaged status. If the host is set to be managed, it is possible to adjust network configuration, storage, and resource allocation parameters from Satellite Server.

Hosts provisioned by Satellite are managed by default. If there is a necessity to obtain reports about configuration management in systems using an operating system not supported by Satellite it is recommended to unmanage the host.

1. Navigate to **Hosts** → **All hosts**.
2. Select the host.
3. Click **`Edit`**.
4. Click **`Manage host`** or **`Unmanage host`** to change the host's status.
5. Click **`Submit`** to save the changes.

> **NOTE**
>
> If you change a host's status from unmanaged to managed, you must rebuild the host by clicking **`Build`** on the Host page.

10.9. ASSIGNING A HOST TO A SPECIFIC ORGANIZATION

The following steps show you how to assign a host to a specific organization. For general information about organizations and how to configure them, see Configuring Organizations, Locations and Life Cycle Environments in the *Server Administration Guide*.

1. Navigate to **Hosts** → **All hosts**.
2. Select the check box of the host you want to change.
3. From the **`Select Action`** menu at the upper right of the screen, select **`Assign Organization`**. A new option window will open.
4. Navigate to the **`Select Organization`** menu and choose the desired organization for your host. Select the check box **`Fix Organization on Mismatch`**.

   > **NOTE**
   >
   > A mismatch happens if there is a resource associated with a host, such as a domain or subnet, and at the same time not associated with the organization you want to assign the host to. The option **`Fix Organization on Mismatch`** will add such a resource to the organization, and is therefore the recommended choice. The option **`Fail on Mismatch`**, on the other hand, will always result in an error message. For example, reassigning a host from one organization to another will fail, even if there is no actual mismatch in settings.

5. Click **`Submit`**.

10.10. ASSIGNING A HOST TO A SPECIFIC LOCATION

The following steps show you how to assign a host to a specific location. For general information about locations and how to configure them, see Locations in the *Server Administration Guide*.

1. Navigate to **Hosts** → **All hosts**.

2. Select the check box of the host you want to change.

3. From the **`Select Action`** menu at the upper right of the screen, select **`Assign Location`**. A new option window will open.

4. Navigate to the **`Select Location`** menu and choose the desired location for your host. Select the check box **`Fix Location on Mismatch`**.

    NOTE

    A mismatch happens if there is a resource associated with a host, such as a domain or subnet, and at the same time not associated with the organization you want to assign the host to. The option **`Fix Organization on Mismatch`** will add such a resource to the organization, and is therefore the recommended choice. The option **`Fail on Mismatch`**, on the other hand, will always result in an error message. For example, reassigning a host from one organization to another will fail, even if there is no actual mismatch in settings.

5. Click **`Submit`** to complete the assigning of the location to your host.

10.11. CONFIGURING AN ADDITIONAL NETWORK INTERFACE

Red Hat Satellite supports specifying multiple network interfaces for a single host. You can configure these interfaces when creating a new host as described in Section 10.4, “Creating a Host” or when editing an existing host.

There are several types of network interfaces that you can attach to a host. When adding a new interface, select one of:

- **Interface**: Allows you to specify an additional physical or virtual interface. There are two types of virtual interfaces you can create. Use *VLAN* when the host needs to communicate with several (virtual) networks using a single interface, while these networks are not accessible to each other. Another type of virtual interface is *alias*, which is an additional IP address attached to an existing interface. See Section 10.11.2, “Adding a Virtual Interface”, or Section 10.11.1, “Adding a Physical Interface” for details.

- **Bond**: Creates a bonded interface. NIC bonding is a way to bind multiple network interfaces together into a single interface that appears as a single device and has a single MAC address. This enables two or more network interfaces to act as one, simultaneously increasing the bandwidth and providing redundancy. See Section 10.11.3, “Adding a Bonded Interface” for details.

- **BMC**: *Baseboard Management Controller* (BMC) allows you to remotely monitor and manage physical state of machines. See Enabling Power Management on Managed Hosts in the *Red Hat Satellite Installation Guide* for more information on BMC, and Section 10.11.4, “Adding a Baseboard Management Controller (BMC) Interface” for details on configuring a BMC interface.

NOTE

Additional interfaces have by default the **Managed** flag enabled, which means the new interface is configured automatically during provisioning by the DNS and DHCP Capsule Servers associated with the selected subnet. This requires a subnet with correctly configured DNS and DHCP Capsule Servers. If you use a kickstart method for host provisioning, configuration files are automatically created for managed interfaces in the post-installation phase at **`/etc/sysconfig/network-scripts/ifcfg-$`*`interface_id`***.

NOTE

Virtual and bonded interfaces currently require a MAC address of a physical device. Therefore, the configuration of these interfaces works only on bare-metal hosts.

10.11.1. Adding a Physical Interface

The following steps show how to add and additional physical interface to a host.

Procedure 10.12. To Add a Physical Interface:

1. Navigate to **Hosts** → **All hosts** to view available hosts.
2. Click **`Edit`** next to the host you want to edit.
3. On the **`Network`** tab, click **`Add Interface`**.
4. Keep the *Interface* option selected in the **Type** menu.
5. Specify a **`MAC address`** of the additional interface. This setting is required.
6. Specify the device **`Identifier`**, for example *eth0* or *eth1.1*. Identifier is used for bonded interfaces (in the **`Attached devices`** field, see Procedure 10.14, “To Add a Bonded Interface:”), VLANs and aliases (in the **`Attached to`** field, see Procedure 10.13, “To Add a Virtual Interface:”).
7. Specify the **`DNS name`** associated with the host's IP address. Satellite saves this name in the Capsule Server associated with the selected domain (the "DNS A" field) and the Capsule Server associated with the selected subnet (the "DNS PTR" field). A single host can therefore have several DNS entries.
8. Select a domain from the **`Domain`** drop-down menu. To create and manage domains, navigate to **Infrastructure** → **Domains**.
9. Select a subnet from the **`Subnet`** drop-down menu. To create and manage subnets, navigate to **Infrastructure** → **Subnets**.
10. Specify the interface **`IP address`**. Managed interfaces with assigned DHCP Capsule Server require this setting for creating a DHCP lease. DHCP-enabled managed interfaces provide an automatic suggestion of IP address.
11. Decide if the interface will be managed. If the **`Managed`** check box is selected, the interface configuration is pulled from the associated Capsule Server during provisioning, and DNS and DHCP entries are created. If using kickstart provisioning, a configuration file is automatically created for the interface.

12. Select the **`Virtual NIC`** check box to create a virtual interface. SeeSection 10.11.2, “Adding a Virtual Interface” for details.

13. Click **`OK`** to save the interface configuration, and then click**`Submit`** to apply the changes to the host.

10.11.2. Adding a Virtual Interface

The following procedure shows how to configure an additional virtual interface for a host. This can be either a VLAN or an alias interface.

An alias interface is an additional IP address attached to an existing interface. Note that:

- An alias interface automatically inherits a MAC address from the interface it is attached to, therefore you can create an alias without specifying a MAC address.

- The interface must be specified in a subnet with boot mode set to **`static`**.

Procedure 10.13. To Add a Virtual Interface:

1. Navigate to **Hosts** → **All hosts** to view available hosts.

2. Click **`Edit`** next to the host you want to edit.

3. On the **`Network`** tab, click **`Add Interface`**.

4. Keep the *Interface* option selected in the**Type** menu.

5. Specify the general interface settings. The applicable configuration options are the same as for the physical interfaces described in Section 10.11.1, “Adding a Physical Interface”.

   Specify **`MAC address`** for managed virtual interfaces so that the configuration files for provisioning are generated correctly. However, **`MAC address`** is not required for virtual interfaces that are not managed.

   If creating a VLAN, specify ID in the form of *eth1.10* in the**`Identifier`** field. If creating an alias, use ID in the form of *eth1:10*.

6. Select the **`Virtual NIC`** check box. Additional configuration options specific to virtual interfaces are appended to the form:

   - **`Tag`**: You can specify tags per interface to provide a higher-level segmentation of the network. If left blank, managed interfaces inherit the tag form the VLAN ID of the associated subnet, given that this subnet has the VLAN ID specified. User-specified entries from this field are not applied on alias interfaces.

   - **`Attached to`**: Specify the identifier of the physical interface to which the virtual interface belongs, for example eth1. This setting is required.

7. Click **`OK`** to save the interface configuration. Then click**`Submit`** to apply the changes to the host.

10.11.3. Adding a Bonded Interface

The following steps show how to configure a bonded interface for a host.

Procedure 10.14. To Add a Bonded Interface:

1. Navigate to **Hosts** → **All hosts** to view available hosts.
2. Click **`Edit`** next to the host you want to edit.
3. On the **`Network`** tab, click **`Add Interface`**.
4. Select *Bond* from the **Type** menu. Additional type-specific configuration options are appended to the form.
5. Specify the general interface settings. The applicable configuration options are the same as for the physical interfaces described in Section 10.11.1, "Adding a Physical Interface". Bonded interfaces use IDs in the form of *bond0* in the **`Identifier`** field. It is sufficient if you specify a single MAC address in the **`MAC address`** field.
6. Specify the configuration options specific to bonded interfaces:
   - **`Mode`**: Select the *bonding mode* that defines a policy for fault tolerance and load balancing. See Table 10.2, "Bonding Modes Available in Red Hat Satellite" for a brief description of individual bonding modes.
   - **`Attached devices`**: Specify a comma separated list of identifiers of attached devices. These can be physical interfaces or VLANs.
   - **`Bond options`**: Specify a space separated list of configuration options, for example *miimon=100*. There are several configuration options you can specify for the bonded interface, see Red Hat Enterprise Linux 7 Networking Guide for details.
7. Click **`OK`** to save the interface configuration. Then click **`Submit`** to apply the changes to the host.

Table 10.2. Bonding Modes Available in Red Hat Satellite

Bonding Mode	Description
balance-rr	Transmissions are received and sent out sequentially on each bonded interface.
active-backup	Transmissions are received and sent out via the first available bonded interface. Another bonded interface is only used if the active bonded interface fails.
balance-xor	Transmissions are based on the selected hash policy. In this mode, traffic destined for specific peers will always be sent over the same interface.
broadcast	All transmissions are sent on all bonded interfaces.
802.a3	Creates aggregation groups that share the same settings. Transmits and receives on all interfaces in the active group.
balance-tlb	The outgoing traffic is distributed according to the current load on each bonded interface.

Bonding Mode	Description
balance-alb	Receive load balancing is achieved through Address Resolution Protocol (ARP) negotiation.

10.11.4. Adding a Baseboard Management Controller (BMC) Interface

This section describes how to configure a *baseboard management controller* (BMC) interface for a host that supports this feature.

Prerequisites

Ensure the following prerequisites are satisfied before proceeding:

- BMC is enabled on the Capsule Server. If required, see Procedure 10.15, "To Enable BMC Power Management on an Existing Capsule Server:".
- The ipmitool package is installed.
- You know the MAC address, IP address, and other details of the BMC interface on the host, and the appropriate credentials for that interface.

NOTE

You only need the MAC address for the BMC interface if the BMC interface is managed. This is so that it can create a DHCP reservation.

Procedure 10.15. To Enable BMC Power Management on an Existing Capsule Server:

1. Use the **satellite-installer** routine to configure BMC power management on the Capsule Server by running the following command with the following options:

   ```
   # satellite-installer --foreman-proxy-bmc=true --foreman-proxy-bmc-
   default-provider=ipmitool
   ```

2. Refresh the features for the Capsule Server.

   a. Log in to the Satellite web UI, and navigate to **Infrastructure** → **Capsules**.

   b. Identify the Capsule Sever whose features you need to refresh. In the drop-down list on the right, click **`Refresh features`**. The list of features in the**`Features`** column should now include BMC.

Procedure 10.16. To Add a BMC Interface:

1. Navigate to **Hosts** → **All hosts** to view available hosts.
2. Click **`Edit`** next to the host you want to edit.
3. On the **`Network`** tab, click **`Add Interface`**.

4. Select *BMC* from the **Type** menu. Type-specific configuration options are appended to the form.

5. Specify the general interface settings. The applicable configuration options are the same as for the physical interfaces described in Section 10.11.1, “Adding a Physical Interface”.

6. Specify the configuration options specific to BMC interfaces:

   - **`Username`**, **`Password`**: Specify any authentication credentials required by BMC.

   - **`Provider`**: Specify the BMC provider.

7. Click **`OK`** to save the interface configuration, and then click**`Submit`** to apply the changes to the host.

10.12. REMOVING A HOST

The following procedure shows how to remove a host from Red Hat Satellite.

Procedure 10.17. To Remove a Host:

1. Click **Hosts** → **All hosts** or **Hosts** → **Content Hosts**.

2. Choose the hosts to be removed.

3. Click **`Select Action`** and choose **`Delete Hosts`** from the drop-down menu.

4. A confirmation pop-up box will appear. Select **`Yes`** to remove the host from Red Hat Satellite permanently.

WARNING

If a host record that is associated with a virtual machine is deleted, the virtual machine will be deleted as well. To avoid deleting the virtual machine in this situation, disassociate the virtual machine from Satellite without removing it from the hypervisor.

Procedure 10.18. To Disassociate A Virtual Machine from Satellite without Removing it from a Hypervisor

1. In the Satellite web UI, navigate to **Hosts** → **All Hosts** and select the check box to the left of the hosts to be disassociated.

2. In the confirmation window:

   a. Optionally, select the check box to keep the hosts for future action.

   b. Click **`Submit`** to save your changes.

CHAPTER 11. DISCOVERING BARE-METAL HOSTS ON SATELLITE

Red Hat Satellite 6.2 ships with the Discovery plug-in already installed. The Discovery plug-in enables automatic bare-metal discovery of unknown hosts on the provisioning network. These new hosts are registered to the Satellite Server and the Puppet agent on the client uploads system facts collected by Facter, such as serial ID, network interface, memory, and disk information. After registration, the hosts are displayed on the **`Discovered Hosts`** page in the Satellite web UI. You can then initiate provisioning either manually (using the web UI, CLI, or API) or automatically, using predefined discovery rules.

The Discovery plug-in communicates through the Satellite Capsule Server, which has direct access both to the provisioning network and the Satellite Server instance. It is possible to discover hosts directly from the Satellite Server, but Red Hat recommends the following scheme be used:

```
Satellite Server (Satellite Server Discovery plug-in) <--> Satellite
Capsule (Satellite Capsule Discovery plug-in) <--> Discovered Host
(Satellite Discovery image)
```

The Satellite Discovery plug-in consists of three different components:

The Satellite Server Discovery plug-in

This runs on the Satellite Server and provides API and UI functionality for working with discovered hosts. The tfm-rubygem-foreman_discovery package contains this plug-in.

The Satellite Capsule Server Discovery plug-in

This is a communication proxy between discovered hosts on a provisioning network and the Satellite Server. The rubygem-smart_proxy_discovery package contains this plug-in.

The Satellite Discovery image

This is the minimal operating system based on Red Hat Enterprise Linux that is PXE-booted on hosts to acquire initial hardware information and to check in to the Satellite Server. Discovered hosts keep running the Satellite Discovery image until they are rebooted into Anaconda, which then initiates the provisioning process. The foreman-discovery-image package contains this image. It must be installed on the Satellite Capsule Server that provides TFTP services.

11.1. CONFIGURING THE SATELLITE DISCOVERY PLUG-IN

The following sections describe how to configure the Satellite Discovery plug-in and how to prepare the PXE-boot template on the Satellite Server.

11.1.1. Deploying the Satellite Discovery Image

Install the package containing the Satellite Discovery image on the Satellite Capsule Server that provides TFTP services (not on the Satellite Server itself):

```
# yum install foreman-discovery-image
```

This package contains the Linux kernel and initial RAM disk image as a bootable ISO file which is used for PXE-booting discovered hosts. You can run the following command to investigate the contents of the package. This produces output similar to the following:

```
$ rpm -ql foreman-discovery-image
/usr/share/foreman-discovery-image
/usr/share/foreman-discovery-image/fdi-image-rhel_7-2.1.0-20150212.1.iso
```

When you install this package, it extracts the kernel and image from the ISO file into the TFTP directory and creates symbolic links to the latest versions of the image and kernel. Use the symbolic links in the PXE-boot provisioning template to make sure that you do not need to change the version in the template every time the foreman-discovery-image package is upgraded. For example:

```
$ find /var/lib/tftpboot/boot
/var/lib/tftpboot/boot
/var/lib/tftpboot/boot/fdi-image-rhel_7-2.1.0-20150212.1-img
/var/lib/tftpboot/boot/fdi-image-rhel_7-2.1.0-20150212.1-vmlinuz
/var/lib/tftpboot/boot/fdi-image-rhel_7-img
/var/lib/tftpboot/boot/fdi-image-rhel_7-vmlinuz
```

The last two lines in the above output contain symbolic links to be used in the PXE-boot provisioning template. For more information see Section 11.1.2, “Configuring PXE-booting”.

NOTE

Currently, only Red Hat Enterprise Linux 7 Discovery images are provided, even for Satellite 6 installations on Red Hat Enterprise Linux 6. If there are discovered hosts running during the upgrade of the foreman-discovery-image package, reboot them all to load the updated version of the image as soon as possible. This can be done through the Satellite 6 web UI, CLI, or API.

11.1.2. Configuring PXE-booting

When an unknown host is booted on the provisioning network, the Satellite Server provides a PXELinux boot menu with a single option; to boot from the local hard drive. The following procedure describes how to change this behavior in order to enable hardware discovery. This requires changing several variables in the PXE Linux global default template. These variables are described below:

- The KERNEL and APPEND lines in the template use symbolic links, created when installing the foreman-discovery-image package (see Section 11.1.1, “Deploying the Satellite Discovery Image”). The URLs are relative to the **/var/lib/tftpboot/** directory. Ensure the ***APPEND*** parameters are specified on a single line.

- The **proxy.type** variable can be set to either **proxy** (recommended) or **foreman**. When the variable is set to **proxy**, all communication goes through the Satellite Capsule Server. When the variable is set to **foreman**, the communication goes directly to Satellite Server. Examples in this chapter assume **proxy.type** is set to **proxy**.

- The **proxy.url** variable specifies the URL of the Satellite Capsule Server or Satellite Server, depending on the **proxy.type** setting. Both **HTTP** and **HTTPS** schemes are supported. The default port is 9090 for accessing the Satellite Capsule Server (**proxy.type=proxy**), and 80 for for direct communication with the Satellite Server (**proxy.type=foreman**).

- The **IPAPPEND 2** setting detects interfaces connected to the provisioning network. The image will not boot correctly if this option is removed or modified.

Procedure 11.1. To Configure PXE-booting:

1. In the Satellite web UI, navigate to **Hosts** → **Provisioning Templates**.
2. Edit the *PXELinux global default* template. Add the following menu entry to the template:

```
LABEL discovery
MENU LABEL Foreman Discovery
MENU DEFAULT
KERNEL boot/fdi-image-rhel_7-vmlinuz
APPEND initrd=boot/fdi-image-rhel_7-img rootflags=loop
root=live:/fdi.iso rootfstype=auto ro rd.live.image acpi=force
rd.luks=0 rd.md=0 rd.dm=0 rd.lvm=0 rd.bootif=0 rd.neednet=0
nomodeset proxy.url=https://SATELLITE_CAPSULE_URL:9090
proxy.type=proxy
IPAPPEND 2
```

3. Set the new menu entry to be the default by modifying the **ONTIMEOUT** variable:

```
ONTIMEOUT discovery
```

4. Click **Build PXE Default** at the top of the**Provisioning Templates** page. This instructs the TFTP proxy to rewrite the **pxelinux.cfg/default** file. Repeat this step every time a change is made to the default template to ensure that the changes are deployed on the TFTP Satellite Capsule Server.

As an alternative to the above procedure, you can omit the **proxy.url** variable from the PXE-boot template. In this case, the Discovery image searches the DNS configuration file for an SRV record named **x-foreman.tcp**. The**proxy.url** variable must be set to**proxy** in this case. The DNS server must also be suitably configured. For example, the following configuration statement specifies the Capsule Server to be used with HTTPS:

```
_x-foreman._tcp SRV 0 5 9090 capsule
```

Here, *capsule* is the name of the Capsule Server that is included in the DNS configuration.

NOTE

Satellite 6.2 only allows you to specify one Capsule Server URL for all subnets where hosts can be discovered. Because templates cannot be used per subnet, use a DNS alias name on all networks. Alternatively, use an SRV record.

IMPORTANT

The DNS servers from the DHCP settings are taken into account only for the interface that is specified via the **`BOOTIF`** variable. **`BOOTIF`** is set automatically by the **`IPAPPEND`** variable in the PXE template. This means that when a system has multiple NICs, DNS will only work for the interface that it was booted from.

11.1.3. Reviewing Global Discovery Settings

You can review global settings related to the Discovery plug-in in the Satellite web UI. Navigate to **Administer** → **Settings** and open the **`Discovered`** tab. Notable settings are:

`discovery_organization`, `discovery_location`

These variables specify where to place the discovered hosts. By default, the discovered hosts are automatically placed under the first organization and location created.

`discovery_fact`

This variable specifies which incoming fact to use to determine the MAC address of the discovered host. By default, the PXELinux BOOTIF kernel command line option is used.

`discovery_auto`

This variable enables automatic provisioning according to specified rules. Set to false by default. Red Hat recommends that you test the configuration with manual provisioning before enabling **`discovery_auto`**. See Section 11.3, "Provisioning Discovered Hosts" for more information.

`discovery_fact_column`

This variable allows you to add any fact reported by Facter as an additional column in the list of discovered hosts.

11.2. CONFIGURING THE SATELLITE CAPSULE SERVER DISCOVERY PLUG-IN

Ensure the **`foreman_url`** setting exists in the Satellite Capsule Server configuration file. The setting can appear as follows:

```
# grep foreman_url /etc/foreman-proxy/settings.yml
:foreman_url: https://satellite.example.com
```

The **`satellite-installer`** command configures this variable automatically, but Red Hat recommends that you check that the host responds correctly and there are no firewall rules blocking communication.

11.2.1. Configuring Discovery Subnets

You need to configure all subnets with discovered hosts to communicate through the Satellite Capsule Server. In the Satellite web UI, navigate to **Infrastructure** → **Subnets** and select the required Capsule Server for each subnet that needs to perform host discovery and ensure it is connected to the Discovery Capsule Server.

To verify that a Capsule Server has the Discovery plug-in enabled, navigate to

Infrastructure → **Capsules**. The Discovery plug-in should appear in the list of features associated with the Capsule Server. Click **`Refresh features`** to ensure that the list is up-to-date.

11.2.2. Using Hammer with the Discovery Plug-in

To use the **`hammer`** command with the Discovery plug-in, you need to enable the Discovery plug-in in **`/etc/hammer/cli.modules.d/foreman_discovery.yml`** as follows:

```
:foreman_discovery:
  :enable_module: true
```

See hammer configuration directories[5] for more information about the files and directories that **`hammer`** uses.

11.2.3. Reviewing User Permissions

When it first starts, the Satellite Capsule Server Discovery plug-in creates a role called **`Discovery`**. You can assign this role to non-administrative users to allow them to use the Discovery plug-in. Alternatively, assign the **`perform_discovery`** permission to an existing role. For more information on roles and permissions, see Creating and Managing Users in the *Server Administration Guide*.

11.3. PROVISIONING DISCOVERED HOSTS

After you have correctly configured Discovery plug-ins on both the Satellite Server and the Capsule Server, you can automatically detect bare-metal hosts. To do so, boot a machine in any provisioning network that was configured with the PXE configuration template described in Section 11.1.2, “Configuring PXE-booting”. The machine is automatically registered with the Satellite Server and appears in the **Hosts** → **Discovered Hosts** list in the Satellite web UI.

You can either provision the discovered host manually, or you can configure automatic provisioning.

11.3.1. Manually Provisioning Hosts

The following procedure describes how to manually provision discovered hosts from the Satellite web UI.

Procedure 11.2. To Manually Provision a Discovered Host:

1. Navigate to **Hosts** → **Discovered Hosts**.
2. Select the host you want to provision and click **`Provision`**.
3. On the host's **`Edit`** page, complete the necessary details, and then click**`Save`**.

When the host configuration is saved, Satellite modifies the host's PXELinux file on the TFTP server and reboots the discovered host. It then boots into an installer for the chosen operating system, and finally into the installed operating system.

If you decide to re-provision an existing discovered host, delete the operating system from the machine and reboot it. The host then reappears on the **`Discovered Hosts`** page.

11.3.2. Decommissioning Discovered Hosts

If you no longer require Red Hat Satellite to manage a specific host, you need to decommission that host to prevent it from being discovered.

Procedure 11.3. To Decommission a Discovered Host:

1. Shut down the host.
2. Navigate to **Hosts** → **Discovered Hosts**.
3. In the **Name** column find the host you want to decommission and then select **Delete** from the corresponding **Edit** drop-down menu.

11.3.3. Automatically Provisioning Hosts

With Satellite 6.2, it is possible to define provisioning rules that will assign a host group to provisioned hosts and trigger provisioning automatically.

Procedure 11.4. To Create a Provisioning Rule:

1. Navigate to **Configure** → **Discovery rules**.
2. Click **`New Rule`**. Specify the following parameters of the provisioning rule:
    - **`Name`** is the name of the rule displayed in the list of rules. This name must not contain spaces or non-alphanumeric characters.
    - **`Search`** is the search statement used to match discovered hosts for the particular rule. You can use scoped search syntax to define it. See Section 11.3.4, “Scoped Search Syntax” for examples of using scoped search.
    - **`Host Group`** is the host group to be assigned to a matching host before starting the provisioning process. Make sure that the selected host group has all the required parameters set; required parameters are marked with an asterisk (*).
    - **`Hostname`** defines a pattern for assigning human-readable host names to the matching hosts. When left blank, the host name is assigned in the format "macMACADDRESS" by default. The same syntax used for provisioning templates is used in this instance. See Section 11.3.5, “Host Name Patterns” for more information and examples.
    - **`Hosts limit`** is the maximum number of provisioned hosts per rule. If the limit is reached, the rule will not take effect until one or more hosts are deleted. Typical use cases are rules per server rack or row when it is necessary to change provisioning parameters such as host name or host group per entry. You can set this value to zero (0) to specify no limit.
    - **`Priority`** specifies the order of execution of rules. The value must be greater than or equal to zero. A lower value indicates a higher priority. If two rules have the same priority, the first rule encountered is applied.
    - **`Enabled`** provides the option to temporarily enable or disable rules.
3. Click **`Submit`** to save the rule.

By default, Satellite does not enable automatic discovery of hosts. The following procedure describes how to enable the `discovery_auto` variable to provide automatic provisioning according to specified rules.

Procedure 11.5. To Enable Automatic Provisioning:

1. Navigate to **Administer** → **Settings** → **Discovered** in the Satellite web UI.
2. Locate `discovery_auto` in the `Name` column, and set its value to `true`.
3. Click **Save**.

After you have defined some rules, Red Hat recommends that you discover a host and apply the rules using the `Auto discover` button on the host. This triggers auto-provisioning without the need to enable the global option.

11.3.4. Scoped Search Syntax

This section shows how to use scoped search syntax to filter the discovered hosts according to selected parameters. This is useful when creating a rule for automatic provisioning (see Section 11.3.3, “Automatically Provisioning Hosts”).

The search fields in the Satellite web UI support automatic completion to make building search strings easier. For example, you can test search patterns on the **Hosts** → **Discovered Hosts** page. The following are examples of typical search queries:

- facts.architecture = x86_64
- facts.bios_vendor ~ 'Dell*'
- facts.macaddress = "aa:bb:cc:dd:ee:ff"
- facts.macaddress_eth0 = "aa:bb:cc:dd:ee:ff"
- facts.ipaddress_eth1 ~ "192.168.*"
- facts.architecture ^ (x86_64,i386)

NOTE

The caret symbol (^) in scoped searches means "in" (the same usage as in SQL) and not "starts with" as it is used in regular expressions. You can review the full list of scoped search operators at https://github.com/wvanbergen/scoped_search/blob/master/lib/scoped_search/qu

In Satellite 6.2, all facts are strings, so it is not possible to do numeric comparisons. However, three important facts are extracted and converted to numbers. These are described in Table 11.1, “Facts that Allow Numerical Comparison”.

Table 11.1. Facts that Allow Numerical Comparison

Search Parameter	Description	Example Usage
cpu_count	The number of CPUs	cpu_count >= 8

Search Parameter	Description	Example Usage
disk_count	The number of disks attached	disk_count < 10
disks_size	The total amount of disk space (in MiB)	disks_size > 1000000

11.3.5. Host Name Patterns

This section lists the host name patterns that you can use when creating a rule for automatic provisioning (see Section 11.3.3, "Automatically Provisioning Hosts").

The target host name template pattern has the same syntax as the provisioning templates (ERB). The domain is appended automatically. In addition to the ***@host*** attribute, the **rand()** function for random integers is available. For example:

- application-server-<%= rand(99999) %>
- load-balancer-<%= @host.facts['bios_vendor'] + '-' + rand(99999) %>
- wwwsrv-<%= @host.hostgroup.name %>
- minion-<%= @host.discovery_rule.name %>
- db-server-<%= @host.ip.gsub('.','-') + '-' + @host.hostgroup.subnet.name %>>

IMPORTANT

When creating host name patterns, ensure the resulting host names are unique. Host names must not start with numbers. A good approach is to use unique information provided by Facter (for example, the MAC address, BIOS or serial ID) or to otherwise randomize the host name.

11.3.6. Using the Discovery Plug-in on the Command Line

You can use the **hammer** command to perform certain tasks related to discovery. Run the **hammer -h** command to verify your configuration:

```
$ hammer -h | grep discovery
 discovery                     Manipulate discovered hosts.
 discovery_rule                Manipulate discovered rules.
```

Use the **hammer discovery -h** command to view the available options. For example, you can use the following command to reboot a discovered host (assuming its ID is 130):

```
$ hammer discovery reboot -id 130
Host reboot started
```

11.4. EXTENDING THE DISCOVERY IMAGE

It is possible to extend the Satellite Discovery image with custom facts, software, or device drivers. You can also provide a compressed archive file containing extra code for the image to use.

First, create the following directory structure:

```
.
├── autostart.d
│   └── 01_zip.sh
├── bin
│   └── ntpdate
├── facts
│   └── test.rb
└── lib
    ├── libcrypto.so.1.0.0
    └── ruby
        └── test.rb
```

Where:

- The **`autostart.d`** directory contains scripts that are executed in POSIX order by the image when it starts, but before the host is registered to Satellite.
- The **`bin`** directory is added to the $PATH variable; you can place binary files here and use them in the autostart scripts.
- The **`facts`** directory is added to the FACTERLIB variable so that custom facts can be configured and sent to Satellite.
- The **`lib`** directory is added to the LD_LIBRARY_PATH variable and**`lib/ruby`** is added to the RUBYLIB variable, so that binary files in **`/bin`** can be executed correctly.

New directives and options are appended to the existing environment variables (PATH, LD_LIBRARY_PATH, RUBYLIB and FACTERLIB). If you need to specify the path to something explicitly in your scripts, the zip contents are extracted to the **`/opt/extension`** directory on the image.

After creating the above directory structure, package it into a zip archive with the following command:

```
zip -r my_extension.zip .
```

You can create multiple zip files but be aware they will be extracted to the same place on the Discovery image, so files in later zips will overwrite earlier ones if they have the same file name.

To inform the Discovery image of the extensions it should use, place your zip files on your TFTP server with the Discovery image, and then update the APPEND line of the PXELinux template with the **`fdi.zips`** option where the paths are relative to the TFTP root. For example, if you have two archives at **`$TFTP/zip1.zip`** and **`$TFTP/boot/zip2.zip`**, use the following syntax:

```
fdi.zips=zip1.zip,boot/zip2.zip
```

See Section 11.1.2, “Configuring PXE-booting” for more information on updating the PXE template.

11.5. TROUBLESHOOTING SATELLITE DISCOVERY

If a machine does not show up correctly in the Satellite web UI under **Hosts** → **Discovered Hosts**, inspect the following configuration areas to help isolate the error:

- Try redeploying the default PXE Linux template.
- Verify the **`pxelinux.cfg/default`** configuration file on the TFTP Capsule Server.
- Ensure adequate network connectivity between hosts, the Capsule Server, and the Satellite Server.
- Verify the **`proxy.url`** and **`proxy.type`** options in the default PXE Linux template.
- Ensure that the DNS is working correctly for that image, or use an IP address in the **`proxy.url`** option in the default PXE Linux template.
- Ensure that the DHCP server is delivering IP addresses to the booted image correctly.
- Ensure the discovered host (or virtual machine) has at least 500 MB of memory. Less memory can lead to various random kernel panic errors as the image needs to be extracted in-memory.

For gathering important system facts, use the **`discovery-debug`** command. It prints out system logs, network configuration, list of facts, and other information on the standard output. The typical use case is to redirect this output and copy it with the **`scp`** command for further investigation.

The first virtual console on the discovered host is reserved for systemd logs. Particularly useful system logs are tagged as follows:

- discover-host - initial facts upload
- foreman-discovery - facts refresh, reboot remote commands
- nm-prepare - boot script which pre-configures NetworkManager
- NetworkManager - networking information

Use TTY2 or higher to log in to a discovered host. The root account and SSH access are disabled by default, but you can enable SSH and set the root password using the following kernel command-line options in the Default PXELinux template on the APPEND line:

```
fdi.ssh=1 fdi.rootpw=redhat
```

[5] https://github.com/theforeman/hammer-cli/blob/master/doc/installation.md#locations

CHAPTER 12. RUNNING JOBS ON SATELLITE HOSTS

Red Hat Satellite supports the ability to run arbitrary commands on hosts. This is referred to as *remote execution*. Remote execution is enabled by default on the Satellite Server, but must be enabled manually on all desired Capsule Servers. Communication occurs through the Capsule Server which means that the Satellite Server does not require direct access to the target host, and can scale to control many hosts. Remote execution uses the SSH service which must be enabled and running on the target host. Ensure the Capsule has access to port 22 on the target hosts.

Commands can be customized in a similar fashion to provisioning templates or partition tables. Several job templates are included by default, that you can use to run commands. See Section 12.2.1, “Setting up Job Templates”.

NOTE

Any Capsule Server's base system is a client of Satellite Server's internal Capsule, and therefore this section applies to any type of host connected to Satellite Server, including Capsule Servers.

You can execute commands on multiple hosts at once, and you can use variables in your commands to suit your deployment. Variable values can be filled by host fact, smart class parameter, smart variable, or even host parameter. In addition, you can specify custom values for templates when you run the command. See Section 12.2.2, “Executing Jobs”.

The following list provides some examples of how you can use remote execution:

- Install, update, or remove software packages
- Bootstrap a configuration management agent
- Trigger a Puppet, Salt, or Chef run

By default, each Capsule is installed with the remote execution feature disabled. To use remote execution on a Capsule Server you need to enable it. To enable, run the following command:

```
# satellite-installer --scenario capsule --enable-foreman-proxy-plugin-
remote-execution-ssh
```

To verify that remote execution is running on the Capsule Server and in the web UI navigate to **Infrastructure** → **Capsules**. The Capsule Server should now list in the **Features** column that `SSH` is running.

By default, Satellite Server is configured to use remote execution rather than Katello Agent. If required, these settings can be changed by first creating custom job templates and then selecting these new templates in the web UI by going to **Administer** → **Remote Execution Features**. For each action you want to change, select the label and then select the job template to use.

12.1. ESTABLISHING A SECURE CONNECTION FOR REMOTE COMMANDS

The SSH keys used for remote execution are created automatically when installing a Capsule and the settings are in the **/etc/foreman-proxy/settings.d/remote_execution_ssh.yml** file. They include the following options:

ssh_identity_file

File to load the SSH key from. By default, set to **/usr/share/foreman-proxy/.ssh/id_rsa_foreman_proxy**.

local_working_dir

Directory used on the Satellite or Capsule to run the scripts necessary for remote execution. By default, set to **/var/tmp**.

remote_working_dir

Directory on the client system that is used to execute the remote execution jobs. By default, set to **/var/tmp**.

NOTE

If the client system has **noexec** set for the **/var/** volume or file system, change the **remote_working_dir** as otherwise the remote execution job will fail since the script cannot be executed.

If required to use an alternative directory, create the new directory, for example *new_place*, and then copy the SELinux context from the default directory. For example:

```
# chcon --reference=/var new_place
```

See the Maintaining SELinux Labels section of the *SELinux User's and Administrator's Guide* for more information on working with SELinux labels.

Distributing the SSH Keys for Remote Execution

To enable remote execution, distribute the public SSH key from a Capsule to the hosts that you want to manage. Ensure the SSH service is enabled and running on the hosts. Configure any network or host-based firewalls to enable access to port 22.

There are three ways to distribute the public key from a Capsule to target hosts:

- To distribute keys manually, execute the following command on the Capsule:

  ```
  # ssh-copy-id -i ~foreman-proxy/.ssh/id_rsa_foreman_proxy.pub
  root@target.example.com
  ```

  Here *target.example.com* is the host name of the target host. Repeat for each target host you want to manage.

  To confirm the key was successfully copied to the target host, execute the following command on the Capsule:

  ```
  # ssh -i ~foreman-proxy/.ssh/id_rsa_foreman_proxy
  root@target.example.com
  ```

- To use the Satellite API to download the public key directly from the Capsule, execute the following command on each target host:

  ```
  # curl https://myproxy.example.com:9090/ssh/pubkey >>
  ~/.ssh/authorized_keys
  ```

  Here *myproxy.example.com* stands for the host name of the Capsule.

- To include the public key in newly-provisioned hosts, modify for example the *Kickstart default finish* template to include the following line:

  ```
  <%= snippet 'remote_execution_ssh_keys' %>
  ```

12.2. CONFIGURING AND RUNNING REMOTE COMMANDS

Any command that you want to execute on a remote host must be defined as a job template. After you have defined a job template you can execute it multiple times.

12.2.1. Setting up Job Templates

Satellite provides a number of default job templates that you can use for executing jobs, find them under **Hosts** → **Job templates**. If you find a template fitting your needs amongst the default templates, proceed to Section 12.2.2, “Executing Jobs”.

You can also use default templates as a basis for developing your own. Default job templates are locked for editing, therefore you have to first clone the template to be able to modify it.

Procedure 12.1. To Create a Job Template:

1. Navigate to **Hosts** → **Job templates**.

2. Click **`New Job Template`**. As an alternative, you can modify an existing template – in the **`Actions`** column, select **`Clone`** from the drop-down menu.

3. Configure the job template:

   a. On the **`Template`** tab, enter a unique name for your job template. Select **`Default`** to make the template available for all organizations and locations. You can insert the template manually using **`Template editor`** or upload it from a text file by clicking **`Browse`**. Templates use Embedded Ruby (ERB) template syntax, see Section 12.2.4, “Creating Advanced Templates” for more information. An advanced template is required, for example, for executing jobs that perform power actions; see Example 12.4, “Including Power Actions in Templates” for information on how to include the *Power Action - SSH Default* template in a custom template.

   b. On the **`Job`** tab, you can define the job category (define your own or select from the default categories listed in Table 12.1, “Default Job Template Categories”) as well as the effective user; these settings can be configured also when invoking the job (see Procedure 12.2, “To Execute a Remote Job:”). You can also define input parameters for template commands. These parameters are then requested when executing the job.

   c. Remaining tabs enable setting the template type, organizations and locations as well as viewing the template history.

4. Click **Submit**. When the page refreshes, your new template should appear in the list of job templates.

IMPORTANT

Note that only the parameters visible on the **Parameters** tab at the host's edit page can be used as input parameters for job templates.

Table 12.1. Default Job Template Categories

Job template category	Description
Packages	Templates for performing package related actions. Install, update, and remove actions are included by default.
Puppet	Templates for executing Puppet runs on target hosts.
Power	Templates for performing power related actions. Restart and shutdown actions are included by default.
Commands	Templates for executing custom commands on remote hosts.
Services	Templates for performing service related actions. Start, stop, restart, and status actions are included by default.
Katello	Templates for performing content related actions. These templates are used mainly from different parts of the Satellite web UI (for example bulk actions UI for content hosts), but can be used separately to perform operations such as errata installation.

Example 12.1. Creating a restorecon Template

This example shows how to create a template called *Run Command - restorecon* that will restore the default **SELinux** context for all files in the selected directory on target hosts.

1. Navigate to **Hosts** → **Job templates**. Click **New Job Template**.

2. Insert *Run Command - restorecon* in the **Name** field. Select **Default** to make the template available to all organizations. Add the following text to the **Template editor**:

```
restorecon -RvF <%= input("directory") %>
```

   The **<%= input("directory") %>** string will be replaced by a user-defined directory during job invocation.

3. On the **Job** tab, perform the following actions:

a. Set **Job category** to **Commands**.

b. Click **Add Input** to allow job customization. Insert **directory** to the **Name** field. The input name must match the value specified in the **Template editor**.

c. Click **Required** so that the command cannot be executed without the user specified parameter.

d. Select **User input** from the **Input type** drop-down list. Also provide a **Description** to be shown during job invocation, for example *Target directory for restorecon*.

4. Click **Submit**.

See Example 12.2, "Executing a restorecon Template on Multiple Hosts" for information on how to execute a job based on this template.

12.2.2. Executing Jobs

This section shows how to run a job based on a job template against one or more hosts.

Procedure 12.2. To Execute a Remote Job:

1. Navigate to **Hosts** → **All hosts** and select the target hosts for your job. You can use the search field to narrow down the host list.

2. From the **Select Action** menu at the upper right of the screen select **Run Job**. This will take you to the **Job invocation** page. Alternatively, if you target just one host, click its name and click **Run Job** on the host information page. Note that you can invoke jobs also from the **Job Templates** page by using the **Run** button.

3. On the **Job invocation** page, define the main job settings:

   a. Select the **Job category** and the **Job template** you want to use. These settings are required.

   b. Optionally, select a stored search string in the **Bookmark** list to specify the target hosts.

   c. Optionally, further limit the targeted hosts by inserting a **Search query**. The **Resolves to** line displays the number of hosts affected by your query. Use the refresh button to recalculate the number after changing the query. The preview icon will list the targeted hosts.

   d. The remaining settings depend on the selected job template. See Procedure 12.1, "To Create a Job Template:" for information on adding custom parameters to a template.

4. Clicking **Display advanced fields** will show advanced setting for the job. Some of the advanced settings depend on the job template, the following settings are general:

   - **Effective user** defines the user for executing the job, by default it is the SSH user.

   - **Concurrency level** defines maximum number of jobs executed at once, which can prevent overload of systems' resources in a case of executing the job on a large number of hosts.
   - **Time span** defines time interval in seconds after which the job should be killed, if it is not finished already. A task which could not be started during the defined interval, for example, if the previous task took too long to finish, is canceled.
   - **Type of query** defines when the search query is evaluated. This helps to keep the query up to date for scheduled tasks.

   Concurrency level and **Time span** settings enable you to tailor job execution to fit your infrastructure hardware and needs.

5. If you want to execute the job immediately, ensure that **Schedule** is set to ***Execute now***. You can also define a one-time future job, or set up a recurring job. For recurring tasks, you can define start and end dates, number and frequency of runs. You can also use cron syntax to define repetition. For more information about cron, see Automating System Tasks section of the Red Hat Enterprise Linux 7 *System Administrator's Guide*.

6. Click **Submit**. This displays the **Job Overview** page, and when the job completes, also displays the status of the job.

Example 12.2. Executing a restorecon Template on Multiple Hosts

This example shows how to execute a job based on the template created in Example 12.1, “Creating a restorecon Template” on multiple hosts. The job will restore the SELinux context in all files under the **/home/** directory.

1. Navigate to **Hosts** → **All hosts** and select target hosts. Select **Run Job** from the **Select Action** drop-down list.
2. In the **Job invocation** page, select the **Commands** job category and the **Run Command - restorecon** job template.
3. Type **/home** in the **directory** field.
4. Set **Schedule** to **Execute now**.
5. Click **Submit**. You are taken to the **Job invocation** page where you can monitor the status of job execution.

12.2.3. Monitoring Jobs

You can monitor the progress of the job while it is running. This can help in any troubleshooting that may be required.

Procedure 12.3. To Monitor a Job:

1. Navigate to the Job page. This page is automatically displayed if you triggered the job with the **Execute now** setting. To monitor scheduled jobs, navigate to **Monitor** → **Jobs** and select the job run you wish to inspect.

2. On the Job page, click the **Hosts** tab. This displays the list of hosts on which the job is running.

3. In the **Host** column, click the name of the host that you want to inspect. This displays the **Detail of Commands** page where you can monitor the job execution in real time.

4. Click **Back to Job** at any time to return to the**Job Details** page.

12.2.4. Creating Advanced Templates

When creating a job template, you can import an existing template in the **Template editor** field – this is referred to as*rendering*. This way you can combine templates, or create more specific templates from the general ones.

The following template combines default templates to install and start the **httpd** service on Red Hat Enterprise Linux systems:

```
<%= render_template 'Package Action - SSH Default', :action => 'install',
:package => 'httpd' %>
<%= render_template 'Service Action - SSH Default', :action => 'start',
:service_name => 'httpd' %>
```

The above template specifies parameter values for the rendered template directly. It is also possible to use the **input()** method to allow users to define input for the rendered template on job execution. For example, you can use the following syntax:

```
<%= render_template 'Package Action - SSH Default', :action => 'install',
:package => input("package") %>
```

With the above template, you have to import the parameter definition from the rendered template. To do so, navigate to the **Jobs** tab, click **Add Foreign Input Set**, and select the rendered template from the **Target template** drop-down list. You can import all parameters or specify a comma separated list.

Example 12.3. Rendering a restorecon Template

This example shows how to create a template derived from the *Run command - restorecon* template created in Example 12.1, “Creating a restorecon Template”. This template does not require user input on job execution, it will restore the SELinux context in all files under the **/home/** directory on target hosts.

Create a new template as described in Section 12.2.1, “Setting up Job Templates”, and specify the following string in the **Template editor** screen:

```
<%= render_template("Run Command - restorecon", :directory => "/home")
%>
```

Example 12.4. Including Power Actions in Templates

This example shows how to set up a job template for performing power actions, such as reboot. This procedure prevents Satellite from interpreting the disconnect exception upon reboot as an error, and consequently, remote execution of the job works correctly.

> Create a new template as described in Section 12.2.1, “Setting up Job Templates”, and specify the following string in the **`Template editor`** screen:
>
> ```
> <%= render_template("Power Action - SSH Default", :action => "restart")
> %>
> ```

12.3. CONFIGURING GLOBAL SETTINGS

The Satellite remote execution feature provides numerous global settings that you can use to configure its behavior. These are listed in Table 12.2, “Global Settings for Remote Execution”. To review and update these settings, navigate to**Administer** → **Settings** and click the **`Remote Execution`** tab.

Table 12.2. Global Settings for Remote Execution

Parameter Name	Description
remote_execution_effective_user	This is the default effective user for any job. When the job is executed the effective user of the process is changed accordingly (for example, by sudo). This option can be overridden per job template and job invocation.
remote_execution_effective_user_method	Specifies which method to use to set the effective user on the target host. Currently only su and sudo are supported.
remote_execution_fallback_proxy	Search the host for any Capsule with remote execution configured. This is useful when the host has no subnet or if the subnet does not have a Capsule with remote execution enabled.
remote_execution_global_proxy	Search for a remote execution Capsule outside of the Capsules assigned to the host. If Locations or Organizations are enabled, the search will be limited to the host's Organization or Location.
remote_execution_ssh_user	The default user to use while the Capsule connects to the target using SSH. You can set the ***`remote_execution_ssh_user`*** variable to override this on a per-host basis. You can set this by Host, Host Group, Operating System, Domain, Location, or Organization. This can also be a different user from the ***`remote_execution_effective_user`***.
remote_execution_sync_templates	Defines whether job templates should be synchronized from disk when seeding a database.

IMPORTANT

It is possible to set global parameters in the **`/etc/foreman/settings.yaml`** configuration file, but any manual changes that you make to this file are overwritten the next time you run **`satellite-installer`**. Consequently, Red Hat recommends that you modify these parameters in the web UI. Alternatively, use the **`foreman-rake config`** command from a console.

12.3.1. Choosing a Capsule for Remote Execution

Remote execution requires a Capsule Server to perform any specified job on a host. By default, any Capsule within the host's organization and location with the *remote execution provider* feature enabled is considered available to perform these jobs. You can set the ***`remote_execution_global_proxy`*** variable to **`false`** to disable this behavior. This may be necessary in more complex environments, where not all Capsules can be used due to possible network isolation. In this configuration, you can assign a pool of Capsules to each subnet, and jobs are load balanced across them.

Alternatively, you can set the ***`remote_execution_fallback_proxy`*** variable to **`true`** to enable fallback mode. In this configuration, remote execution will use any Capsule associated with the host, such as its Puppet Master, provided that Capsule also has remote execution configured.

12.4. DELEGATING PERMISSIONS FOR REMOTE EXECUTION

You can control which users can run which jobs within your infrastructure, including which hosts they can target. The remote execution feature provides two built-in roles:

- *Remote Execution Manager*: This role allows access to all remote execution features and functionality.
- *Remote Execution User*: This role only allows running jobs; it does not provide permission to modify job templates.

You can clone the Remote Execution User role and customize its filter for increased granularity. If you adjust the filter with the ***`view_job_templates`*** permission, the user can only see and trigger jobs based on matching job templates. You can use the ***`view_hosts`*** and ***`view_smart_proxies`*** permissions to limit which hosts or Capsules are visible to the role.

The ***`execute_template_invocation`*** permission is a special permission that is checked immediately before execution of a job begins. This permission defines which job template you can run on a particular host. This allows for even more granularity when specifying permissions. For more information on working with roles and permissions see Creating and Managing Roles in the *Server Administration Guide*.

The following example shows filters for the ***`execute_template_invocation`*** permission:

```
name = Reboot and host.name = staging.example.com
name = Reboot and host.name ~ *.staging.example.com
name = "Restart service" and host_group.name = webservers
```

The first line in the above example permits the user to execute the *Reboot* template on one selected host. The second line defines a pool of hosts with names ending with *.staging.example.com*. The third line binds the template with a host group.

NOTE

Permissions assigned to users can change over time. If a user has already scheduled some jobs to run in the future, and the permissions have changed, this can result in execution failure because the permissions are checked immediately before job execution.

CHAPTER 13. CONFIGURING HOST COLLECTIONS

The *Host Collections* application tab is a system management tool that allows the administrator to:

- Add hosts to a collection.
- Apply a mass installation of packages, errata, or package groups to all host members of a host collection.
- Update specific packages, errata, or specific package groups to all host members.

13.1. CREATING A HOST COLLECTION

The following procedure shows how to create host collections.

Procedure 13.1. To Create a Host Collection:

1. Click **Hosts** → **Host Collections**.
2. Click **`New Host Collection`**.
3. Add the Name and Description of the host collection.
4. Deselect **`Unlimited Content Hosts`** to specify the maximum number of hosts that will be allowed to the group. Otherwise, leave it checked to allow unlimited hosts to join the host collection.
5. Click **`Save`**.

13.2. ADDING HOSTS TO A HOST COLLECTION

The following procedure shows how to add hosts to host collections.

Prerequisites

A host must be registered to Red Hat Satellite in order to add it to a Host Collection. Refer to Section 10.5, “Registration” for information on how to register a host.

Procedure 13.2. To Add Hosts to a Host Collection:

1. Click **Hosts** → **Host Collections**.
2. Click the host collection where the host should be added.
3. On the **`Hosts`** tab, select the **`Add`** subtab.
4. Select the hosts to be added from the table and click **`Add Selected`**.

13.3. ADDING CONTENT TO HOST COLLECTIONS

These steps show how to add content to host collections in Red Hat Satellite.

13.3.1. Adding Packages to a Host Collection

The following procedure shows how to add packages to host collections.

Prerequisites

- The content to be added should be available in one of the existing repositories or added prior to this procedure.
- Content should be promoted to the environment where the hosts are assigned.

Procedure 13.3. To Add Packages to Host Collections:

1. Click **Hosts** → **Host Collections**.
2. Click the host collection where the package should be added.
3. On the **`Collection Actions`** tab, click **`Package Installation, Removal, and Update`**.
4. To update all packages, click the **`Update All Packages`** button to use the default method. Alternatively, select the drop-down icon to the right of the button to select a method to use. Selecting the **via remote execution - customize first** menu entry will take you to the **`Job invocation`** page where you can customize the action.
5. Select the **`Package`** or **`Package Group`** radio button as required.
6. In the field provided, specify the package or package group name. Then click:
   - **`Install`** — to install a new package using the default method. Alternatively, select the drop-down icon to the right of the button and select a method to use. Selecting the **via remote execution - customize first** menu entry will take you to the **`Job invocation`** page where you can customize the action.
   - **`Update`** — to update an existing package in the host collection using the default method. Alternatively, select the drop-down icon to the right of the button and select a method to use. Selecting the **via remote execution - customize first** menu entry will take you to the **`Job invocation`** page where you can customize the action.

13.3.2. Adding Errata to a Host Collection

The following procedure shows how to add errata to host collections.

Prerequisites

- The errata to be added should be available in one of the existing repositories or added prior to this procedure.
- Errata should be promoted to the environment where the hosts are assigned.

Procedure 13.4. To Add Errata to a Host Collection:

1. Click **Hosts** → **Host Collections**.
2. Select the host collection where the errata should be added.
3. On the **`Collection Actions`** tab, click **`Errata Installation`**.

4. Select the errata you want to add to the host collection and click the **`Install Selected`** button to use the default method. Alternatively, select the drop-down icon to the right of the button to select a method to use. Selecting the **via remote execution - customize first** menu entry will take you to the**`Job invocation`** page where you can customize the action.

13.4. REMOVING CONTENT FROM A HOST COLLECTION

The following procedure shows how to remove packages from host collections.

Procedure 13.5. To Remove Content from a Host Collection:

1. Click **Hosts** → **Host Collections**.

2. Click the host collection where the package should be removed.

3. On the **`Collection Actions`** tab, click **`Package Installation, Removal, and Update`**.

4. Select the **`Package`** or **`Package Group`** radio button as required.

5. In the field provided, specify the package or package group name.

6. Click the **`Remove`** button to remove the package or package group using the default method. Alternatively, select the drop-down icon to the right of the button and select a method to use. Selecting the **via remote execution - customize first** menu entry will take you to the **`Job invocation`** page where you can customize the action.

13.5. CHANGING THE LIFE CYCLE ENVIRONMENT OR CONTENT VIEW OF A HOST COLLECTION

The following procedure shows how to change the assigned life cycle environment or content view of host collections.

Procedure 13.6. To Change the Life Cycle Environment or Content View of a Host Collection:

1. Click **Hosts** → **Host Collection**.

2. Selection the host collection where the life cycle environment or content view should be changed.

3. On the **`Collection Actions`** tab, click **`Change assigned Life Cycle Environment or Content View`**.

4. Select the life cycle environment to be assigned to the host collection.

5. Select the required content view from the drop-down list.

6. Click **`Assign`**.

NOTE

The changes take effect in approximately 4 hours. To make the changes take effect immediately, on the host, enter the following command:

```
# subscription-manager refresh
```

You can use remote execution to run this command on multiple hosts at the same time.

13.6. REMOVING A HOST FROM A HOST COLLECTION

The following procedure shows how to remove hosts from host collections.

Procedure 13.7. To Remove Hosts from a Host Collection:

1. Click **Hosts** → **Host Collections**.
2. Choose the desired host collection.
3. On the **Hosts** tab, select the **List/Remove** subtab.
4. Select the hosts you want to remove from the host collection and click **Remove Selected**.

13.7. REMOVING A HOST COLLECTION

The following procedure shows how to remove a host collection.

Procedure 13.8. To Remove a Host Collection:

1. Click **Hosts** → **Host Collections**.
2. Choose the host collection to be removed.
3. Click **Remove**. An alert box appears:

   ```
   Are you sure you want to remove host collection Host Collection
   Name?
   ```

4. Click **Remove**.

13.8. CLONING A HOST COLLECTION

The following procedure shows how to clone a host collection.

Procedure 13.9. To Clone a Host Collection:

1. Click **Hosts** → **Host Collections**.
2. On the left hand panel, click the host collection you want to clone.

3. Click **`Copy Collection`**.
4. Specify a name for the cloned collection.
5. Click **`Create`**.

13.9. REVIEWING HOST COLLECTION DETAILS

The following procedure shows how to review host collection details.

Procedure 13.10. To Reviewing Host Collection Details:

1. Click **Hosts** → **Host Collections**.
2. Select the host collection you want to review and navigate to the **`Details`** tab.

APPENDIX A. TEMPLATE WRITING REFERENCE

Embedded Ruby (ERB) is a tool for generating text files based on templates that combine plain text with Ruby code. Red Hat Satellite uses ERB syntax in *provisioning templates* (Section 9.3.9, “Provisioning Templates” and Creating Provisioning Templates in the *Red Hat Satellite Provisioning Guide*), remote execution *job templates* (Chapter 12, *Running Jobs on Satellite Hosts*), templates for *partition tables* (Section 9.3.8, “Partition Tables”), and *smart parameters* (Section 9.2.2, “Configuring Smart Variables”). This section provides an overview of Satellite specific functions and variables that can be used in ERB templates along with some usage examples. Note that the default templates provided by Red Hat Satellite (**Hosts** → **Provisioning templates**, **Hosts** → **Job templates**) also provide a good source of ERB syntax examples.

When provisioning a host or running a remote job, the code in the ERB is executed and the variables are replaced with the host specific values. This process is referred to as *rendering*. The Satellite Server has the safemode rendering option enabled by default, which prevents any harmful code being executed from templates.

A.1. WRITING ERB TEMPLATES

The following points summarize the ERB syntax:

- **<% %>** – marks enclosing Ruby code within the ERB template. The code is executed when the template is rendered. It can contain Ruby control flow structures as well as Satellite specific functions and variables. For example:

  ```
  <% if @host.operatingsystem.family == "Redhat" &&
  @host.operatingsystem.major.to_i > 6 %>
  systemctl <%= input("action") %> <%= input("service") %>
  <% else %>
  service <%= input("service") %> <%= input("action") %>
  <% end -%>
  ```

- **<%= %>** – the code output is inserted into the template. This is useful for variable substitution, for example:

  ```
  echo <%= @host.name %>
  ```

- **<% -%>**, **<%= -%>** – by default, a newline character is inserted after a Ruby block if it is closed at the end of a line. To suppress this behavior, modify the enclosing mark. For example, the following template:

  ```
  curl <%= @host.ip -%>
  /mydir
  ```

  is rendered the same as:

  ```
  curl <%= @host.ip %>/mydir
  ```

  In practice, this is used to reduce the number of lines in rendered templates (where Ruby syntax permits).

- **<%# %>** – marks enclosing a comment that will be ignored during template rendering:

```
<%# A comment %>
```

A.2. TROUBLESHOOTING ERB TEMPLATES

The Satellite web UI provides two ways to verify the template rendering for a specific host:

- *Directly in the template editor* – when editing a template (under **Hosts** → **Partition tables**, **Hosts** → **Provisioning templates**, or **Hosts** → **Job templates**), on the **`Template`** tab click **`Preview`** and select a host from the drop-down menu. The template then renders in the text field using the selected host's parameters. Preview failures can help to identify issues in your template.

- *At the host's details page* – select a host at **Hosts** → **All hosts** and click the **`Templates`** tab to list templates associated with the host. Select **`Review`** from the drop-down menu next to the selected template to view it's rendered version.

A.3. SATELLITE SPECIFIC FUNCTIONS AND VARIABLES

This section lists Satellite specific functions and variables for ERB templates. Note that some of them can be used in any kind of template, others are limited, for example job templates accept only @host variables and variables from Table A.4, “Kickstart Specific Variables” are only applicable in Kickstart templates.

You can use the functions listed in the following table across all kinds of templates.

Table A.1. Generic Functions

Name	Description
indent(n)	Indents the block of code by n spaces, useful when using a snippet template that is not indented.
foreman_url(kind)	Returns the full URL to host-rendered templates of the given kind. For example, templates of the "provision" type usually reside at *http://HOST/unattended/provision*.
snippet(name)	Renders the specified snippet template. Useful for nesting provisioning templates.
snippets(file)	Renders the specified snippet found in the Foreman database, attempts to load it from the **`unattended/snippets/`** directory if it is not found in the database.
snippet_if_exists(name)	Renders the specified snippet, skips if no snippet with the specified name is found.

Example A.1. Using the snippet and indent Functions

The following syntax imports the *subscription_manager_registration* snippet to the template and indents it by four spaces:

```
<%= indent 4 do
snippet 'subscription_manager_registration'
end %>
```

The following functions can be used in job templates. See Section 12.2.4, “Creating Advanced Templates” for usage examples.

Table A.2. Functions Specific to Job Templates

Name	Description
input(input_name)	Returns the value of the specified input on the job execution.
render_template(name, parameters)	Renders the specified template, similar to the generic snippet() function but enables passing arguments to the template.

The following variables enable using host data within templates.

Table A.3. Host Specific Variables and Functions

Name	Description
@host.architecture	The architecture of the host.
@host.bond_interfaces	Returns an array of all bonded interfaces. See Note.
@host.capabilities	The method of system provisioning, can be either build (for example kickstart) or image.
@host.certname	The SSL certificate name of the host.
@host.diskLayout	The disk layout of the host. Can be inherited from the operating system.
@host.domain	The domain of the host.
@host.environment	The Puppet environment of the host.
@host.facts	Returns a Ruby hash of facts from Facter. For example to access the 'ipaddress' fact from the output, specify @host.facts['ipaddress'].
@host.grub_pass	Returns the host's GRUB password.
@host.hostgroup	The host group of the host.

Name	Description
@host.info['parameters']	Returns a Ruby hash containing information on host parameters. For example, use @host.info['parameters']['lifecycle_environment'] to get the life cycle environment of a host.
@host.image_build?	Returns **`true`** if the host is provisioned using an image.
@host.interfaces	Contains an array of all available host interfaces including the primary interface. See Note.
@host.interfaces_with_identifier('IDs')	Returns array of interfaces with given identifier. You can pass an array of multiple identifiers as an input, for example @host.interfaces_with_identifier(['eth0', 'eth1']). See Note.
@host.ip	The IP address of the host.
@host.location	The location of the host.
@host.mac	The MAC address of the host.
@host.managed_interfaces	Returns an array of managed interfaces (excluding BMC and bonded interfaces). See Note.
@host.medium	The assigned operating system installation medium.
@host.name	The full name of the host.
@host.operatingsystem.family	The operating system family.
@host.operatingsystem.major	The major version number of the assigned operating system.
@host.operatingsystem.minor	The minor version number of the assigned operating system.
@host.operatingsystem.name	The assigned operating system name.
@host.operatingsystem.boot_files_uri(@host.medium,@host.architecture)	Full path to the kernel and initrd, returns an array.
@host.os.medium_uri(@host)	The URI used for provisioning (path configured in installation media).
@host.param_false?(name)	Returns **`false`** if host parameter of a given name evaluates to false.

Name	Description
@host.param_true?(name)	Returns `true` if host parameter of a given name evaluates to true.
@host.params['parameter_name']	Returns the value of specified parameters.
@host.primary_interface	Returns the primary interface of the host.
@host.provider	The compute resource provider.
@host.provision_interface	Returns the provisioning interface of the host. Returns an interface object.
@host.ptable	The partition table name.
@host.puppetmaster	The Puppet master the host should use.
@host.pxe_build?	Returns `true` if the host is provisioned using the network or PXE.
@host.shortname	The short name of the host.
@host.sp_ip	The IP address of the BMC interface.
@host.sp_mac	The MAC address of the BMC interface.
@host.sp_name	The name of the BMC interface.
@host.sp_subnet	The subnet of the BMC network.
@host.subnet.dhcp	Returns `true` if a DHCP proxy is configured for this host.
@host.subnet.dns_primary	The primary DNS server of the host.
@host.subnet.dns_secondary	The secondary DNS server of the host.
@host.subnet.gateway	The gateway of the host.
@host.subnet.mask	The subnet mask of the host.
@host.url_for_boot(:initrd)	Full path to the initrd image associated with this host. Not recommended, as it does not interpolate variables.
@host.url_for_boot(:kernel)	Full path to the kernel associated with this host. Not recommended, as it does not interpolate variables, prefer boot_files_uri.

Name	Description
@provisioning_type	Equals to 'host' or 'hostgroup' depending on type of provisioning.
@static	Returns **`true`** if the network configuration is static.
@template_name	Name of the template being rendered.
grub_pass	Returns the GRUB password wrapped in md5pass argument, that is **`--md5pass=#{@host.grub_pass}`**.
ks_console	Returns a string assembled using the port and the baud rate of the host which can be added to a kernel line. For example **`console=ttyS1,9600`**.
root_pass	Returns the root password configured for the system.

NOTE

Host variables related to network interfaces, such as **`@host.interfaces`** or **`@host.bond_interfaces`** return interface data grouped in an array. To extract a parameter value of a specific interface, use Ruby methods to parse the array. For example, to get information about the first interface from an array and use it in a kickstart template:

```
<% myinterface = @host.interfaces.first %>
IPADDR="<%= myinterface.ip %>"
NETMASK="<%= myinterface.subnet.mask %>"
GATEWAY="<%= myinterface.subnet.gateway %>"
```

You can iterate over the interface array, for example to extract an array of interface names use:

```
<% ifnames = []
@host.interfaces.each do |i|
  ifnames.push(i.name)
end %>
```

Example A.2. Using Host Specific Variables

The following example checks if the host has Puppet and the Puppetlabs repository enabled:

```
<%
pm_set = @host.puppetmaster.empty? ? false : true
puppet_enabled = pm_set || @host.param_true?('force-puppet')
puppetlabs_enabled = @host.param_true?('enable-puppetlabs-repo')
%>
```

The following example shows how to capture the minor and major version of the host's operating system, which can be used for package related decisions:

```
<%
os_major = @host.operatingsystem.major.to_i
os_minor = @host.operatingsystem.minor.to_i
%>

<% if ((os_minor < 2) && (os_major < 14)) -%>
...
<% end -%>
```

The following example imports the 'kickstart_networking_setup' snippet if the host's subnet has the DHCP boot mode enabled:

```
<% subnet = @host.subnet %>
<% if subnet.respond_to?(:dhcp_boot_mode?) -%>
<%= snippet 'kickstart_networking_setup' %>
<% end -%>
```

The majority of common Ruby methods can be applied on host specific variables. For example, to extract the last segment of the host's IP address, you can use:

```
<% @host.ip.split('.').last %>
```

The following variables are designed to be used within kickstart provisioning templates.

Table A.4. Kickstart Specific Variables

Name	Description
@arch	The host architecture name, same as @host.architecture.name.
@dynamic	Returns `true` if the partition table being used is a %pre script (has the #Dynamic option as the first line of the table).
@epel	A command which will automatically install the correct version of the epel-release rpm. Use in a %post script.
@mediapath	The full kickstart line to provide the URL command.
@osver	The operating system major version number, same as @host.operatingsystem.major.