



# **Red Hat**

## **Training and Certification**

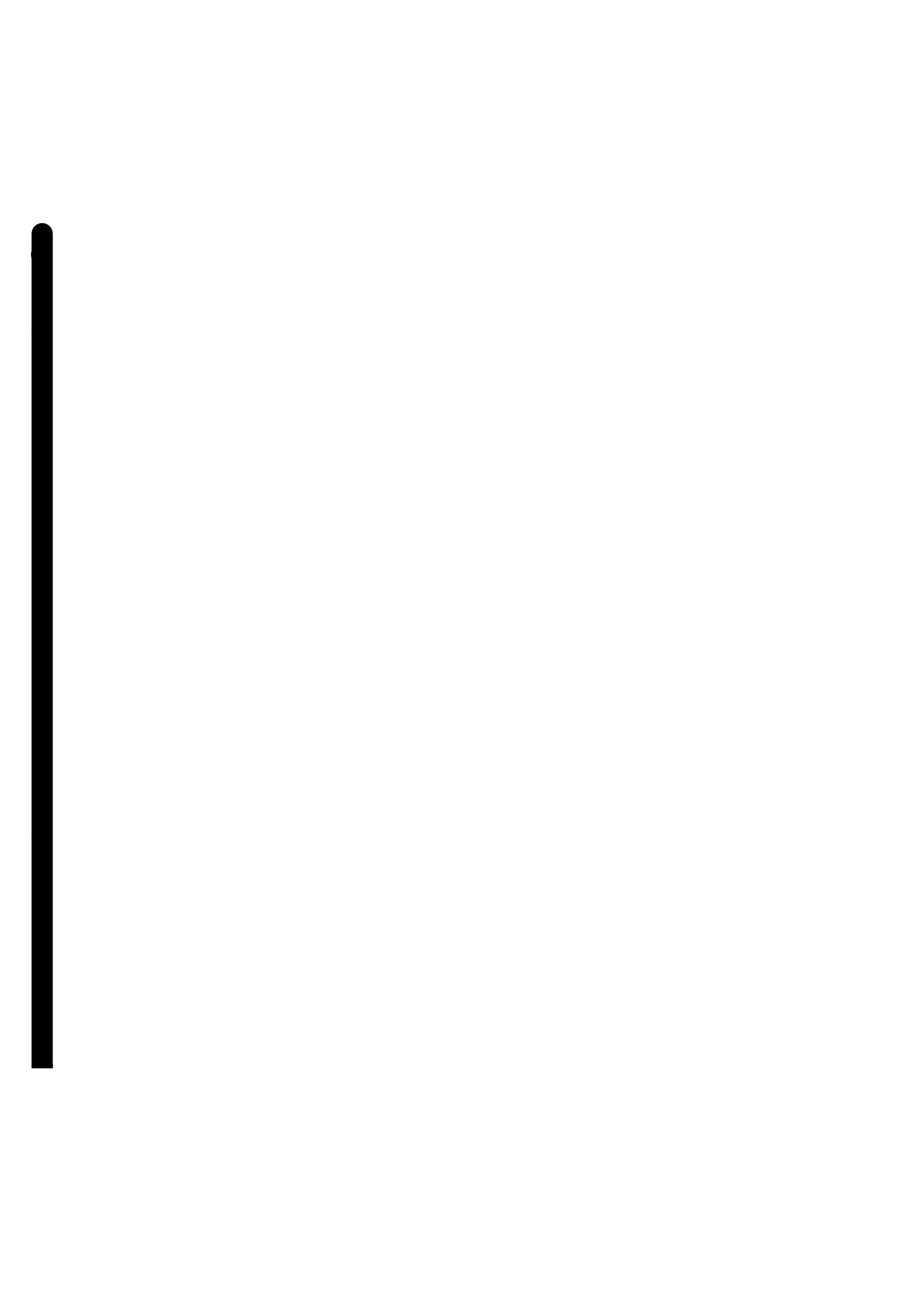
### **Student Workbook (ROLE)**

Red Hat Satellite 6.6 RH403

### **Red Hat Satellite 6 Administration**

Edition 1





# **Red Hat Satellite 6 Administration**



**Red Hat Satellite 6.6 RH403**  
**Red Hat Satellite 6 Administration**  
**Edition 120200108**  
**Publication date 20200108**

Authors: Victor Costea, Trey Feagle, Artur Glogowski, George Hacker,  
Snehangshu Karmakar, Saumik Paul, Herve Quatremain, Adolfo Vazquez,  
Morgan Weetman  
Editor: Steve Bonneville, Phil Sweany

Copyright © 2020 Red Hat, Inc.

The contents of this course and all its modules and related materials, including handouts to audience members, are  
Copyright © 2020 Red Hat, Inc.

No part of this publication may be stored in a retrieval system, transmitted or reproduced in any way, including, but  
not limited to, photocopy, photograph, magnetic, electronic or other record, without the prior written permission of  
Red Hat, Inc.

This instructional program, including all material provided herein, is supplied without any guarantees from Red Hat,  
Inc. Red Hat, Inc. assumes no liability for damages or legal action arising from the use or misuse of contents or details  
contained herein.

If you believe Red Hat training materials are being used, copied, or otherwise improperly distributed please e-mail  
[training@redhat.com](mailto:training@redhat.com) or phone toll-free (USA) +1 (866) 626-2994 or +1 (919) 754-3700.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, JBoss, Hibernate, Fedora, the Infinity Logo, and RHCE are  
trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux® is the registered trademark of Linus Torvalds in the United States and other countries.

Java® is a registered trademark of Oracle and/or its affiliates.

XFS® is a registered trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or  
other countries.

The OpenStack® Word Mark and OpenStack Logo are either registered trademarks/service marks or trademarks/  
service marks of the OpenStack Foundation, in the United States and other countries and are used with the  
OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack  
Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

Contributors: Dana Singletary, Rich Jerrido

<b>Document Conventions</b>	<b>ix</b>
<b>Introduction</b>	<b>xi</b>
Red Hat Satellite 6 Administration .....	xi
Orientation to the Classroom Environment .....	xii
Internationalization .....	xvii
<b>1. Planning and Deploying Red Hat Satellite</b>	<b>1</b>
Describing Red Hat Satellite .....	2
Quiz: Describing Red Hat Satellite .....	4
Planning a Red Hat Satellite Deployment .....	6
Quiz: Planning a Red Hat Satellite Deployment .....	10
Installing Red Hat Satellite .....	12
Guided Exercise: Verifying a Red Hat Satellite Installation .....	21
Configuring Organizations and Content Manifests .....	25
Guided Exercise: Configuring Organizations and Content Manifests .....	29
Lab: Planning and Deploying Red Hat Satellite .....	33
Summary .....	40
<b>2. Managing Software Life Cycles</b>	<b>41</b>
Synchronizing Red Hat Content .....	42
Guided Exercise: Synchronizing Red Hat Content .....	46
Creating Software Life Cycles .....	49
Guided Exercise: Creating Software Life Cycles .....	54
Publishing and Promoting Content Views .....	57
Guided Exercise: Publishing and Promoting Content Views .....	62
Lab: Managing Software Life Cycles .....	65
Summary .....	72
<b>3. Registering Hosts</b>	<b>73</b>
Registering and Configuring Content Hosts .....	74
Guided Exercise: Registering and Configuring Content Hosts .....	77
Managing Hosts with Host Collections .....	80
Guided Exercise: Managing Hosts with Host Collections .....	82
Automating Registration of Content Hosts .....	84
Guided Exercise: Automating Registration of Content Hosts .....	88
Lab: Registering Hosts .....	91
Summary .....	96
<b>4. Deploying Software to Hosts</b>	<b>97</b>
Controlling Software with Content Views .....	98
Guided Exercise: Controlling Software with Content Views .....	100
Creating Content View Filters .....	103
Guided Exercise: Creating Content View Filters .....	106
Managing and Applying Errata to Hosts .....	111
Guided Exercise: Managing and Applying Errata to Hosts .....	113
Managing Module Streams for RHEL 8 Hosts .....	116
Guided Exercise: Managing Module Streams for RHEL 8 Hosts .....	118
Lab: Deploying Software to Hosts .....	120
Summary .....	130
<b>5. Deploying Custom Software</b>	<b>131</b>
Creating Custom Products and Repositories .....	132
Guided Exercise: Creating Custom Products and Repositories .....	135
Creating Products Using Repository Discovery .....	137
Guided Exercise: Creating Products Using Repository Discovery .....	139
Administering Custom Products and Repositories .....	141
Guided Exercise: Administering Custom Products and Repositories .....	147

Lab: Deploying Custom Software .....	151
Summary .....	158
<b>6. Deploying Satellite Capsule Servers</b>	<b>159</b>
Installing a Satellite Capsule Server .....	160
Guided Exercise: Installing a Satellite Capsule Servers .....	165
Configuring Satellite Capsule Server Services .....	170
Guided Exercise: Configuring Satellite Capsule Server Services .....	173
Publishing Content Views to a Satellite Capsule Server .....	175
Guided Exercise: Publishing Content Views to a Satellite Capsule Server .....	178
Quiz: Deploying Satellite Capsule Servers .....	180
Summary .....	182
<b>7. Running Remote Execution</b>	<b>183</b>
Running Remote Jobs on Managed Hosts .....	184
Guided Exercise: Running Remote Jobs on Managed Hosts .....	192
Configuring Ansible Remote Execution .....	197
Guided Exercise: Configuring Ansible Remote Execution .....	205
Running Remote Puppet Jobs on Managed Hosts .....	208
Guided Exercise: Running Remote Puppet Jobs on Managed Hosts .....	211
Lab: Running Remote Execution .....	215
Summary .....	220
<b>8. Provisioning Hosts</b>	<b>221</b>
Configuring Satellite Server for Host Provisioning .....	222
Guided Exercise: Configuring Satellite Server for Host Provisioning .....	227
Preparing Network Configuration for Provisioning .....	231
Guided Exercise: Preparing Network Configuration for Provisioning .....	236
Performing Host Provisioning .....	240
Guided Exercise: Performing Host Provisioning .....	247
Lab: Provisioning Hosts .....	250
Summary .....	264
<b>9. Managing Red Hat Satellite Using the API</b>	<b>265</b>
Querying the Red Hat Satellite API .....	266
Guided Exercise: Querying the Red Hat Satellite API .....	270
Integrating Red Hat Satellite Functionality in Applications .....	278
Guided Exercise: Integrating Red Hat Satellite Functionality in Applications .....	285
Using the Hammer CLI as an API Interface .....	289
Guided Exercise: Using the Hammer CLI as an API Interface .....	292
Lab: Managing Red Hat Satellite Using the API .....	295
Summary .....	302
<b>10. Deploying Red Hat Satellite to a Cloud Platform</b>	<b>303</b>
Running Red Hat Satellite Server on a Cloud Platform .....	304
Quiz: Running Red Hat Satellite Server on a Cloud Platform .....	308
Managing Content Hosts on a Cloud Platform .....	310
Quiz: Managing Content Hosts on a Cloud Platform .....	314
Summary .....	316
<b>11. Performing Red Hat Satellite Server Maintenance</b>	<b>317</b>
Configuring Users and Roles for Task Delegation .....	318
Guided Exercise: Configuring Users and Roles for Task Delegation .....	321
Configuring Backup and Restore Operations .....	324
Guided Exercise: Configure Backup and Restore Operations .....	326
Managing Red Hat Satellite Databases .....	329
Guided Exercise: Managing Red Hat Satellite Databases .....	331
Exporting and Importing Content Views .....	336

Guided Exercise: Exporting and Importing Content Views .....	338
Lab: Performing Red Hat Satellite Server Maintenance .....	341
Summary .....	349
<b>12. Comprehensive Review</b>	<b>351</b>
Course Objectives Listing .....	352
Lab: Configuring Satellite Server .....	355
Lab: Installing and Configuring Satellite Capsule Server .....	360
Lab: Provisioning a Host .....	369
Lab: Performing Remote Execution .....	380
Lab: Signing RPM Packages .....	385



# Document Conventions

---



## References

"References" describe where to find external documentation relevant to a subject.



## Note

"Notes" are tips, shortcuts or alternative approaches to the task at hand. Ignoring a note should have no negative consequences, but you might miss out on a trick that makes your life easier.



## Important

"Important" boxes detail things that are easily missed: configuration changes that only apply to the current session, or services that need restarting before an update will apply. Ignoring a box labeled "Important" will not cause data loss, but may cause irritation and frustration.



## Warning

"Warnings" should not be ignored. Ignoring warnings will most likely cause data loss.



# Introduction

## Red Hat Satellite 6 Administration

*Red Hat Satellite 6 Administration* (RH403) is a four-day lab-based course that explores the concepts and methods necessary for successful large-scale management of Red Hat Enterprise Linux systems. Course participants will learn how to install a Red Hat Satellite 6 server and populate it with software packages. This course is based on Red Hat Enterprise Linux 8 and Red Hat Satellite 6.6. Students will use Red Hat Satellite to manage the software development life cycle of a subscribed host and its configuration, and learn how to provision hosts integrated with software and Ansible configuration management upon deployment.

### Course Objectives

- Verifying a Red Hat Satellite 6.6 installation.
- Regulating Red Hat Satellite with organizations, locations, users, and roles.
- Managing software with Red Hat Satellite environments and content views.
- Configuring Red Hat Satellite hosts with Ansible Playbooks and roles.
- Provisioning hosts with integrated software and configuration management.
- Implementing Metal-as-a-Service (MaaS) with Satellite discovery and provisioning of unprovisioned hosts.

### Audience

- Senior Red Hat Enterprise Linux system administrators responsible for the management of multiple servers.

### Prerequisites

- The prerequisites for this class are RHCE certification or equivalent experience, and Red Hat Satellite 6 experience.

# Orientation to the Classroom Environment

Red Hat Training courses are delivered using virtual machines running on either physical hypervisor systems in a local training environment, or a cloud-based platform remotely accessed from an Internet-enabled system. Training events are designated as one of these modalities:

## Instructor-led training (ILT)

A scheduled event with students, systems and an instructor together in a physical training environment.

## Red Hat Online Learning (ROL)

A remote, self-paced student has on-demand access to the Red Hat Online Learning Environment using a Red Hat Learning Subscription with online support.

## Virtual training (VT)

A scheduled event during which students remotely access an instructor-led course delivered through the Red Hat Online Learning Environment.

## Virtual instructor-led training (vILT)

A scheduled event with a local instructor and training environment, like ILT, but with course materials remotely delivered through the Red Hat Online Learning Environment.

This course architecture, VM configuration, and content are the same in all delivery modalities. The following diagram displays the virtual machines in this course:

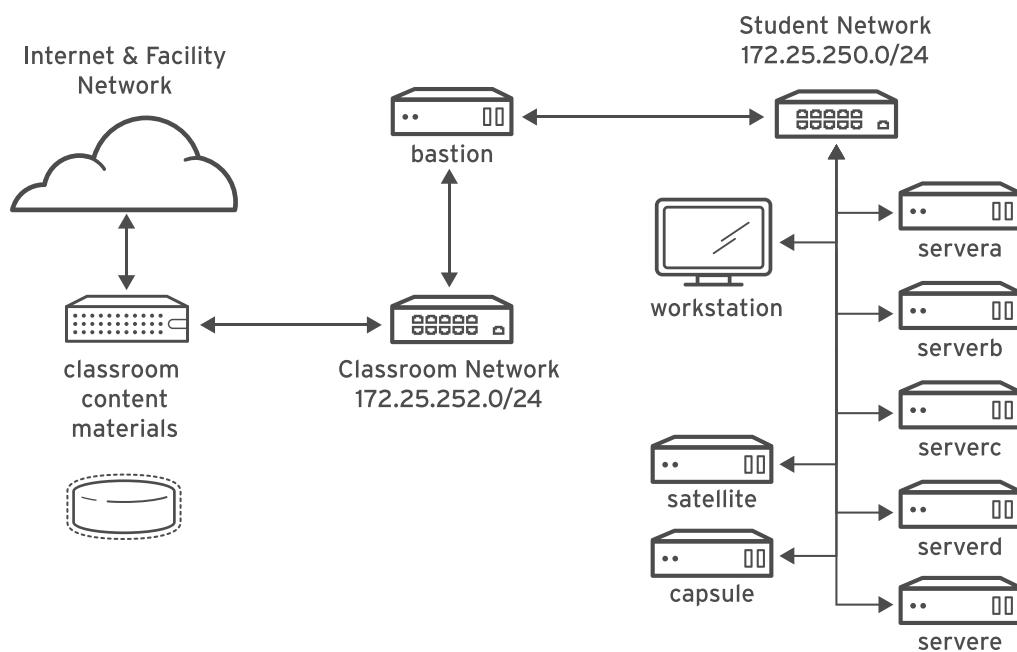


Figure 0.1: Classroom environment

## Classroom Virtual Machines

Machine name	IP addresses	Role
classroom.example.com content.example.com materials.example.com	172.25.252.254	Course support system providing DHCP, DNS and file-sharing services
bastion.lab.example.com	172.25.250.254	Router linking this student network to the classroom
workstation.lab.example.com	172.25.250.9	Graphical workstation that is used as a system administration desktop
servera.lab.example.com	172.25.250.10	Managed content host "A"
serverb.lab.example.com	172.25.250.11	Managed content host "B"
serverc.lab.example.com	172.25.250.12	Managed content host "C"
serverd.lab.example.com	172.25.250.13	Managed content host "D"
servere.lab.example.com	172.25.250.14	Managed content host "E"
satellite.lab.example.com	172.25.250.15	Satellite Server host
capsule.lab.example.com	172.25.250.16	Capsule Server host

## Support Systems

Within every course environment, specific virtual machines provide access or support services for student systems. The **bastion** VM is a network router to safely segregate the student network and virtual machines from the wider classroom and Internet, and may also supply DNS services. This system must remain running for the student environment to function properly. You do not have login access to this system.

The **classroom** VM is another support-only system with no student login access. This system provides network services, such as DHCP and DNS, and two HTTP file sharing aliases. The first alias, **content.example.com**, provides product repositories. The second alias, **materials.example.com**, shares course-specific materials needed for hands-on activities, as referenced in activity instructions. This system must also remain running for the student environment to function properly.

Some courses include a **utility** VM that provides course-specific services, such as identity management, a container repository, file storage, databases, IP address management, or custom network routing. Students are provided access to this system only in specific courses, when necessary.

## Student Systems

The remaining course VMs are for direct student use. The system naming and user accounts for these VMs are course-specific. Typically, students have access to all student VMs using a standard user account, **student**, with the password **student**, and the privileged account **root** with the password **redhat**. However, some specialized VMs, such as compute nodes or similar infrastructure, might restrict the **root** account or require unique access methods. The coursebook provides the necessary access information in relevant lectures and activities.

## The Workstation as the Student Workspace

The **workstation** VM is the primary system for student access to the course. You log in through the graphical console on **workstation**, using the **student** account. To perform privileged tasks, you can log in directly as **root**, or use the **sudo** command to switch to **root**.

Many Red Hat products use a web interface that presents a significant visual information, requiring that the **workstation** screen be set to the highest practical resolution. On **workstation**, use a setting equal to, or preferably slightly lower than the screen resolution of the physical system from which you access this course environment. First, configure the physical system for the highest acceptable resolution that displays correctly. Standard High Definition (HD) resolution (1920 x 1080) is suggested if your physical system supports it. Then, click **Activities** → **Settings** → **Displays** → **Resolution** on **workstation**. Set the resolution to between a minimum of **1280 x 960 (4:3)** and a maximum that is slightly lower than the resolution set for your physical system.

## Course-specific Systems

This course uses seven additional VMs for student activities: **satellite**, **capsule**, **servera**, **serverb**, **serverc**, **serverd**, and **servere**. The first two are for building Satellite services, and the remainder are used as practice content hosts, as shown in the table titled “Classroom Virtual Machines”. All VMs have fully qualified names in the **lab.example.com** DNS domain.

As activities progress in this Satellite course, systems will develop dependencies on each other. Activities include building a Capsule server using information obtained from the Satellite Server, and registering content hosts with a Satellite or Capsule Server. Resetting individual servers will affect those dependencies, such as a Satellite Server that has records for content hosts that have been reset, or a content host previously registered on a Capsule server that is initialized.

It is recommended that proper Satellite procedures be followed to unregister content hosts and uninstall Capsule Servers before they are reset. Accordingly, if you are experiencing a non-working Satellite environment, or wish to simply start again at the course beginning, it is recommended that you reprovision the classroom or reset all VMs in the classroom at the same time.

## Recommendation for Accessing Course Systems

Each student VM can be accessed through a console, or through an SSH remote terminal. Red Hat Training recommends that you access **workstation** first, then use SSH from **workstation** for command line access to each other course systems. Although you can open a graphics-capable console window to a course VM, this is not necessary because a server is usually headless, meaning that its console is text-based, and also because many Red Hat products are managed through browser-based web interfaces. VM console access is required only for VM boot process interaction, or to resolve SSH connection issues.

In a ROL course environment, open VM console windows from the **Online Lab** tab for the course. Each console opens as a new tab in your local browser on your physical system. For environment security, cut and paste activity is limited between your external access system and the online course environment, making this method cumbersome for VM activity. Instead, it is recommended to open a multi-tabbed terminal window on **workstation**, and use SSH to access each needed course system in a separate tab or terminal, as shown in Figure 0.2. Using this terminal-based access method, cut and paste between VMs works efficiently.

```

root@serverc:~ 
File Edit View Search Terminal Tabs Help
student@workstation:~ × student@servera:~ × student@serverb:~ × root@serverc:~ ×
[student@workstation ~]$ ssh serverc
Activate the web console with: systemctl enable --now cockpit.socket

Last login: Sun Dec 15 15:56:17 2019 from 172.25.250.9
[student@serverc ~]$ sudo -i
[sudo] password for student:
[root@serverc ~]#

```

Figure 0.2: Accessing course VMs using terminal tabs and SSH.

## Controlling Your Systems

Students are assigned remote computers in a Red Hat Online Learning classroom. They are accessed through a web application hosted at [rol.redhat.com](http://rol.redhat.com) [<http://rol.redhat.com>]. Students should log in to this site using their Red Hat Customer Portal user credentials.

### Controlling the Virtual Machines

The virtual machines in your classroom environment are controlled through a web page. The state of each virtual machine in the classroom is displayed on the page under the **Online Lab** tab.

#### Machine States

Virtual Machine State	Description
STARTING	The virtual machine is in the process of booting.
STARTED	The virtual machine is running and available (or, when booting, soon will be).
STOPPING	The virtual machine is in the process of shutting down.
STOPPED	The virtual machine is completely shut down. Upon starting, the virtual machine boots into the same state as when it was shut down (the disk will have been preserved).
PUBLISHING	The initial creation of the virtual machine is being performed.
WAITING_TO_START	The virtual machine is waiting for other virtual machines to start.

Depending on the state of a machine, a selection of the following actions is available.

#### Classroom/Machine Actions

Button or Action	Description
<b>PROVISION LAB</b>	Create the ROL classroom. Creates all of the virtual machines needed for the classroom and starts them. Can take several minutes to complete.
<b>DELETE LAB</b>	Delete the ROL classroom. Destroys all virtual machines in the classroom. <b>Caution: Any work generated on the disks is lost.</b>

Button or Action	Description
<b>START LAB</b>	Start all virtual machines in the classroom.
<b>SHUTDOWN LAB</b>	Stop all virtual machines in the classroom.
<b>OPEN CONSOLE</b>	Open a new tab in the browser and connect to the console of the virtual machine. Students can log in directly to the virtual machine and run commands. In most cases, students should log in to the <b>workstation</b> virtual machine and use <b>ssh</b> to connect to the other virtual machines.
<b>ACTION → Start</b>	Start (power on) the virtual machine.
<b>ACTION → Shutdown</b>	Gracefully shut down the virtual machine, preserving the contents of its disk.
<b>ACTION → Power Off</b>	Forcefully shut down the virtual machine, preserving the contents of its disk. This is equivalent to removing the power from a physical machine.
<b>ACTION → Reset</b>	Forcefully shut down the virtual machine and reset the disk to its initial state. <b>Caution: Any work generated on the disk is lost.</b>

At the start of an exercise, if instructed to reset a single virtual machine node, click **ACTION → Reset** for only the specific virtual machine.

At the start of an exercise, if instructed to reset all virtual machines, click **ACTION → Reset**

If you want to return the classroom environment to its original state at the start of the course, you can click **DELETE LAB** to remove the entire classroom environment. After the lab has been deleted, you can click **PROVISION LAB** to provision a new set of classroom systems.



### Warning

The **DELETE LAB** operation cannot be undone. Any work you have completed in the classroom environment up to that point will be lost.

## The Autostop Timer

The Red Hat Online Learning enrollment entitles students to a certain amount of computer time. To help conserve allotted computer time, the ROL classroom has an associated countdown timer, which shuts down the classroom environment when the timer expires.

To adjust the timer, click **MODIFY** to display the **New Autostop Time** dialog box. Set the number of hours until the classroom should automatically stop. Note that there is a maximum time of ten hours. Click **ADJUST TIME** to apply this change to the timer settings.

# Internationalization

## Per-user Language Selection

Your users might prefer to use a different language for their desktop environment than the system-wide default. They might also want to use a different keyboard layout or input method for their account.

### Language Settings

In the GNOME desktop environment, the user might be prompted to set their preferred language and input method on first login. If not, then the easiest way for an individual user to adjust their preferred language and input method settings is to use the Region & Language application.

You can start this application in two ways. You can run the command **gnome-control-center region** from a terminal window, or on the top bar, from the system menu in the right corner, select the settings button (which has a crossed screwdriver and wrench for an icon) from the bottom left of the menu.

In the window that opens, select Region & Language. Click the **Language** box and select the preferred language from the list that appears. This also updates the **Formats** setting to the default for that language. The next time you log in, these changes will take full effect.

These settings affect the GNOME desktop environment and any applications such as **gnome-terminal** that are started inside it. However, by default they do not apply to that account if accessed through an **ssh** login from a remote system or a text-based login on a virtual console (such as **tty5**).



#### Note

You can make your shell environment use the same **LANG** setting as your graphical environment, even when you log in through a text-based virtual console or over **ssh**. One way to do this is to place code similar to the following in your `~/.bashrc` file. This example code will set the language used on a text login to match the one currently set for the user's GNOME desktop environment:

```
i=$(grep 'Language=' /var/lib/AccountsService/users/${USER} \
| sed 's/Language=/'')
if [ "$i" != "" ]; then
    export LANG=$i
fi
```

Japanese, Korean, Chinese, and other languages with a non-Latin character set might not display properly on text-based virtual consoles.

Individual commands can be made to use another language by setting the **LANG** variable on the command line:

```
[user@host ~]$ LANG=fr_FR.utf8 date  
jeu. avril 25 17:55:01 CET 2019
```

Subsequent commands will revert to using the system's default language for output. The **locale** command can be used to determine the current value of **LANG** and other related environment variables.

## Input Method Settings

GNOME 3 in Red Hat Enterprise Linux 7 or later automatically uses the IBus input method selection system, which makes it easy to change keyboard layouts and input methods quickly.

The Region & Language application can also be used to enable alternative input methods. In the Region & Language application window, the **Input Sources** box shows what input methods are currently available. By default, **English (US)** may be the only available method. Highlight **English (US)** and click the **keyboard** icon to see the current keyboard layout.

To add another input method, click the **+** button at the bottom left of the **Input Sources** window. An **Add an Input Source** window will open. Select your language, and then your preferred input method or keyboard layout.

When more than one input method is configured, the user can switch between them quickly by typing **Super+Space** (sometimes called **Windows+Space**). A *status indicator* will also appear in the GNOME top bar, which has two functions: It indicates which input method is active, and acts as a menu that can be used to switch between input methods or select advanced features of more complex input methods.

Some of the methods are marked with gears, which indicate that those methods have advanced configuration options and capabilities. For example, the Japanese **Japanese (Kana Kanji)** input method allows the user to pre-edit text in Latin and use **Down Arrow** and **Up Arrow** keys to select the correct characters to use.

US English speakers may also find this useful. For example, under **English (United States)** is the keyboard layout **English (international AltGr dead keys)**, which treats **AltGr** (or the right **Alt**) on a PC 104/105-key keyboard as a "secondary shift" modifier key and dead key activation key for typing additional characters. There are also Dvorak and other alternative layouts available.



### Note

Any Unicode character can be entered in the GNOME desktop environment if you know the character's Unicode code point. Type **Ctrl+Shift+U**, followed by the code point. After **Ctrl+Shift+U** has been typed, an underlined **u** will be displayed to indicate that the system is waiting for Unicode code point entry.

For example, the lowercase Greek letter lambda has the code point U+03BB, and can be entered by typing **Ctrl+Shift+U**, then **03BB**, then **Enter**.

## System-wide Default Language Settings

The system's default language is set to US English, using the UTF-8 encoding of Unicode as its character set (**en\_US.utf8**), but this can be changed during or after installation.

From the command line, the **root** user can change the system-wide locale settings with the **localectl** command. If **localectl** is run with no arguments, it displays the current system-wide locale settings.

To set the system-wide default language, run the command **localectl set-locale** **LANG=locale**, where *locale* is the appropriate value for the **LANG** environment variable from the "Language Codes Reference" table in this chapter. The change will take effect for users on their next login, and is stored in **/etc/locale.conf**.

```
[root@host ~]# localectl set-locale LANG=fr_FR.utf8
```

In GNOME, an administrative user can change this setting from Region & Language by clicking the **Login Screen** button at the upper-right corner of the window. Changing the **Language** of the graphical login screen will also adjust the system-wide default language setting stored in the **/etc/locale.conf** configuration file.



### Important

Text-based virtual consoles such as **tty4** are more limited in the fonts they can display than terminals in a virtual console running a graphical environment, or pseudoterminals for **ssh** sessions. For example, Japanese, Korean, and Chinese characters may not display as expected on a text-based virtual console. For this reason, you should consider using English or another language with a Latin character set for the system-wide default.

Likewise, text-based virtual consoles are more limited in the input methods they support, and this is managed separately from the graphical desktop environment. The available global input settings can be configured through **localectl** for both text-based virtual consoles and the graphical environment. See the **localectl(1)** and **vconsole.conf(5)** man pages for more information.

## Language Packs

Special RPM packages called *langpacks* install language packages that add support for specific languages. These langpacks use dependencies to automatically install additional RPM packages containing localizations, dictionaries, and translations for other software packages on your system.

To list the langpacks that are installed and that may be installed, use **yum list langpacks-\***:

```
[root@host ~]# yum list langpacks-*
Updating Subscription Management repositories.
Updating Subscription Management repositories.
Installed Packages
langpacks-en.noarch      1.0-12.el8      @AppStream
Available Packages
langpacks-af.noarch       1.0-12.el8      rhel-8-for-x86_64-appstream-rpms
langpacks-am.noarch       1.0-12.el8      rhel-8-for-x86_64-appstream-rpms
langpacks-ar.noarch       1.0-12.el8      rhel-8-for-x86_64-appstream-rpms
langpacks-as.noarch       1.0-12.el8      rhel-8-for-x86_64-appstream-rpms
langpacks-ast.noarch      1.0-12.el8      rhel-8-for-x86_64-appstream-rpms
...output omitted...
```

To add language support, install the appropriate langpacks package. For example, the following command adds support for French:

```
[root@host ~]# yum install langpacks-fr
```

Use **yum repoquery --whatsonplements** to determine what RPM packages may be installed by a langpack:

```
[root@host ~]# yum repoquery --whatsonplements langpacks-fr
Updating Subscription Management repositories.
Updating Subscription Management repositories.
Last metadata expiration check: 0:01:33 ago on Wed 06 Feb 2019 10:47:24 AM CST.
glibc-langpack-fr-0:2.28-18.el8.x86_64
gnome-getting-started-docs-fr-0:3.28.2-1.el8.noarch
 hunspell-fr-0:6.2-1.el8.noarch
 hyphen-fr-0:3.0-1.el8.noarch
 libreoffice-langpack-fr-1:6.0.6.1-9.el8.x86_64
 man-pages-fr-0:3.70-16.el8.noarch
 mythes-fr-0:2.3-10.el8.noarch
```



### Important

Langpacks packages use RPM *weak dependencies* in order to install supplementary packages only when the core package that needs it is also installed.

For example, when installing *langpacks-fr* as shown in the preceding examples, the *mythes-fr* package will only be installed if the *mythes* thesaurus is also installed on the system.

If *mythes* is subsequently installed on that system, the *mythes-fr* package will also automatically be installed due to the weak dependency from the already installed *langpacks-fr* package.



### References

**locale(7), localectl(1), locale.conf(5), vconsole.conf(5), unicode(7),** and **utf-8(7)** man pages

Conversions between the names of the graphical desktop environment's X11 layouts and their names in **localectl** can be found in the file **/usr/share/X11/xkb/rules/base.lst**.

## Language Codes Reference



### Note

This table might not reflect all langpacks available on your system. Use **yum info langpacks-SUFFIX** to get more information about any particular langpacks package.

### Language Codes

Language	Langpacks Suffix	\$LANG value
English (US)	en	en_US.utf8

<b>Language</b>	<b>Langpacks Suffix</b>	<b>\$LANG value</b>
Assamese	as	as_IN.utf8
Bengali	bn	bn_IN.utf8
Chinese (Simplified)	zh_CN	zh_CN.utf8
Chinese (Traditional)	zh_TW	zh_TW.utf8
French	fr	fr_FR.utf8
German	de	de_DE.utf8
Gujarati	gu	gu_IN.utf8
Hindi	hi	hi_IN.utf8
Italian	it	it_IT.utf8
Japanese	ja	ja_JP.utf8
Kannada	kn	kn_IN.utf8
Korean	ko	ko_KR.utf8
Malayalam	ml	ml_IN.utf8
Marathi	mr	mr_IN.utf8
Odia	or	or_IN.utf8
Portuguese (Brazilian)	pt_BR	pt_BR.utf8
Punjabi	pa	pa_IN.utf8
Russian	ru	ru_RU.utf8
Spanish	es	es_ES.utf8
Tamil	ta	ta_IN.utf8
Telugu	te	te_IN.utf8



## Chapter 1

# Planning and Deploying Red Hat Satellite

### Goal

Plan a Red Hat Satellite deployment, installation, and initial configuration of Red Hat Satellite servers.

### Objectives

- Describe the purpose, architecture, and components of Red Hat Satellite.
- Discuss planning a distributed Red Hat Satellite with Satellite Capsule Servers deployment, to meet multiple requirements and scenarios.
- Describe how to perform an initial installation of Red Hat Satellite.
- Describe and configure organizations in Red Hat Satellite, and create and install each organization's content manifests.

### Sections

- Describing Red Hat Satellite (and Quiz)
- Planning a Red Hat Satellite Deployment (and Quiz)
- Installing Red Hat Satellite (and Guided Exercise)
- Configuring Organizations and Content Manifests (and Guided Exercise)

### Lab

Planning and Deploying Red Hat Satellite

# Describing Red Hat Satellite

## Objectives

After completing this section, you should be able to describe the purpose, architecture, and components of Red Hat Satellite.

## Satellite Server Components

Red Hat Satellite 6 is a systems management tool for configuring systems and providing software updates from the Red Hat Customer Portal. It serves as a local software content repository and as central management for Red Hat product entitlements. Red Hat Satellite performs provisioning and configuration management to adhere to predefined standard operating environments.

With the addition of Capsule Servers to its architecture, Red Hat Satellite 6 scales effectively to meet the demands of large enterprises. With a correctly designed deployment of Capsule Servers, Satellite delivers solid performance as workloads increase, even across a geographically distributed environment.

Satellite Server and Capsule Server are additional software installed on top of Red Hat Enterprise Linux. Hosts must have the Red Hat Subscription Manager software installed to receive software updates from Red Hat Satellite Server.

Satellite Server has several administration choices. A web browser can manage Satellite Server through the web UI. A command-line interface, named **hammer**, is available for Satellite Server administration. An API is available for programmatic interacting with Satellite Server to create custom workflows or task automation.

Red Hat Satellite 6.6 is the latest release for Red Hat Satellite, and includes new features such as a redesigned web UI and support for Red Hat Ansible Engine 2.8. The following figure illustrates the architecture of Red Hat Satellite 6:

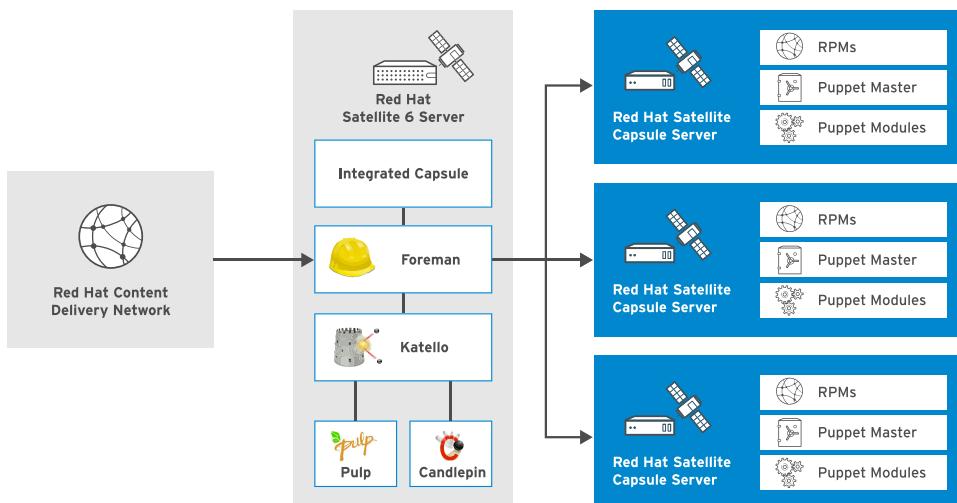


Figure 1.1: Red Hat Satellite 6 system architecture

### Foreman

Foreman is an open source application used for provisioning and life cycle management of physical and virtual hosts. Foreman can automatically configure systems using various methods, such as kickstart and Puppet modules. Foreman also provides historical data for reporting, auditing, and troubleshooting.

### Katello

Katello is a subscription and repository management application. It provides a means to subscribe to Red Hat repositories, and to download content. Different content versions can be applied to specific hosts to match their stage in a *Software Development Life Cycle (SDLC)*.

### Candlepin

Candlepin is a service that handles subscription management.

### Pulp

Pulp is a service that handles repository and content management. Pulp manages content views, sync plans, and the synchronous or delayed content transfer to Capsule Servers.

### Hammer

Hammer is a CLI tool providing command-line and shell equivalents for most functions available through the Satellite Server web UI. Hammer uses environment variables, aliases, and redirection to other CLI tools to expedite interaction with Satellite Server.

### REST API

Red Hat Satellite 6 includes a RESTful API service for system administrators and developers to write custom scripts and third-party applications that interface with Red Hat Satellite.

## Introducing Satellite Capsule Server

Red Hat Satellite Capsule Server provides a proxy for many Satellite functions, including repository storage, DNS, DHCP, and Puppet master configuration. The Satellite Server installation contains an integrated Capsule Server instance. For distributed environments, additional Capsule Servers can be deployed on separate hosts. These Capsule Servers provide redundancy for Satellite services, to offer high availability and to scale to greater workloads as the number of managed hosts increases.



### References

For more information, refer to the *Satellite 6 Architecture* chapter in the *Planning for Red Hat Satellite 6* guide at  
[https://access.redhat.com/documentation/en-us/red\\_hat\\_satellite/6.6/html-single/planning\\_for\\_red\\_hat\\_satellite\\_6](https://access.redhat.com/documentation/en-us/red_hat_satellite/6.6/html-single/planning_for_red_hat_satellite_6)

## ► Quiz

# Describing Red Hat Satellite

Choose the correct answers to the following questions:

- ▶ **1. Which two of the following are new features of Red Hat Satellite 6.6? (Choose two.)**
  - a. Redesigned web UI
  - b. Red Hat Insights integration
  - c. Red Hat Enterprise Linux system roles
  - d. Red Hat Ansible Engine 2.8 support
  
- ▶ **2. Which three of the following services are components of Red Hat Satellite? (Choose three.)**
  - a. Foreman
  - b. Katello
  - c. Pulp
  - d. Undercloud
  - e. Content Delivery Network (CDN)
  
- ▶ **3. What is the key functionality provided by Red Hat Satellite Capsule Server?**
  - a. A separate service to synchronize content from Red Hat Customer Portal.
  - b. A high availability service for Red Hat Satellite.
  - c. A proxy for some of the main Red Hat Satellite functions.
  - d. An API service to access Red Hat Satellite.
  
- ▶ **4. Which of the following statements better defines the hammer component for Red Hat Satellite?**
  - a. A CLI tool that supports most of the functions available through the Satellite Server web UI.
  - b. A service that handles subscription management.
  - c. A service that handles repository and content management.
  - d. A utility for provisioning and life-cycle management of physical and virtual hosts.

## ► Solution

# Describing Red Hat Satellite

Choose the correct answers to the following questions:

- ▶ 1. **Which two of the following are new features of Red Hat Satellite 6.6? (Choose two.)**
  - a. Redesigned web UI
  - b. Red Hat Insights integration
  - c. Red Hat Enterprise Linux system roles
  - d. Red Hat Ansible Engine 2.8 support
  
- ▶ 2. **Which three of the following services are components of Red Hat Satellite? (Choose three.)**
  - a. Foreman
  - b. Katello
  - c. Pulp
  - d. Undercloud
  - e. Content Delivery Network (CDN)
  
- ▶ 3. **What is the key functionality provided by Red Hat Satellite Capsule Server?**
  - a. A separate service to synchronize content from Red Hat Customer Portal.
  - b. A high availability service for Red Hat Satellite.
  - c. A proxy for some of the main Red Hat Satellite functions.
  - d. An API service to access Red Hat Satellite.
  
- ▶ 4. **Which of the following statements better defines the hammer component for Red Hat Satellite?**
  - a. A CLI tool that supports most of the functions available through the Satellite Server web UI.
  - b. A service that handles subscription management.
  - c. A service that handles repository and content management.
  - d. A utility for provisioning and life-cycle management of physical and virtual hosts.

# Planning a Red Hat Satellite Deployment

## Objectives

After completing this section, you should be able to discuss planning a distributed Red Hat Satellite with Satellite Capsule Servers deployment, to meet multiple requirements and scenarios.

## Satellite Server Configuration

A Red Hat Satellite deployment starts with the installation of a Red Hat Satellite Server. This installation also includes an integrated Capsule Server. The procedure depends on whether you choose a connected or disconnected installation. By default, the Red Hat Satellite Server installation creates the minimum resources required to run Red Hat Satellite, including a default organization and location. This typical installation does not include importing manifests or configuring the content repositories.



### Note

Use predefined **tuned** profiles to improve Red Hat Satellite performance.

## Satellite Server with External Database

Red Hat Satellite server uses PostgreSQL and MongoDB databases for its back end. A default Red Hat Satellite Server installation configures these databases to run on the same host with the other Red Hat Satellite Server components. Satellite also supports using external databases to distribute the workload and improve response times for database operations. Use an external database in any of the following scenarios:

- Large numbers of remote execution tasks
- Heavy disk I/O workload from frequent repository synchronization or content view publishing
- Large numbers of hosts
- Large volumes of synchronized content

## Satellite Deployment Scenarios

With the addition of Capsule Server to the Satellite 6 architecture, a Satellite infrastructure layout requires design considerations prior to the Satellite Server installation. A proper design aligns the Satellite Server and Capsule Servers to best serve the environment requirements. There are several options for Satellite infrastructure design.

## Standalone Satellite Server

The basic topology involves a single Satellite Server. In the following example, there are five pools of hosts registered to Satellite Server. The pools are categorized into three locations: United States, United Kingdom, and Japan. In addition, each department uses a unique organization: Finance, Marketing, and Sales. Satellite Server functions are shared among these locations and organizations. All systems are managed through the single Satellite Server instance.

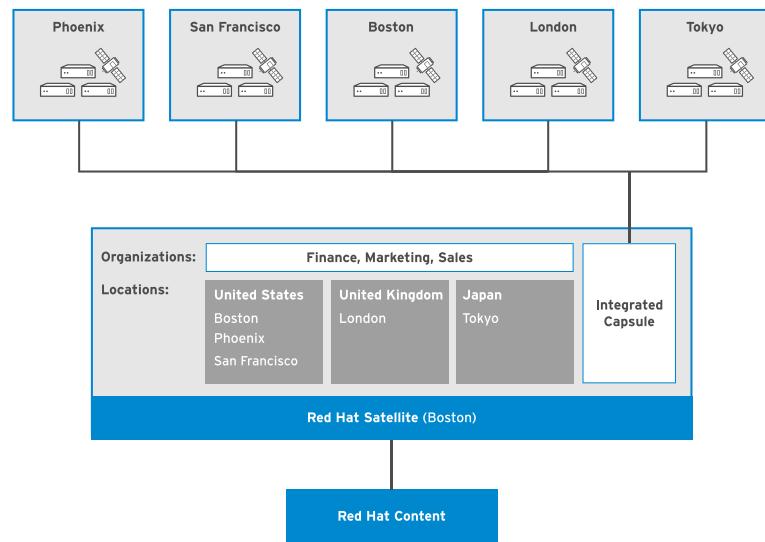


Figure 1.2: Single Satellite Server with integrated Satellite Capsule Server

## Satellite Server with Local Satellite Capsule Servers

In larger environments, additional Capsule Servers can be added to ease the workload on Satellite Server. This example topology includes two Capsule Servers collocated with the Satellite Server. One Capsule Server serves the three U.S.-based offices: Boston, Phoenix, and San Francisco, while the other serves the international offices: London and Tokyo. The workload handled by the two Capsule Servers eases the demand on the main Satellite Server and its integrated Capsule Server. As new offices are created in the locations and added to the corresponding location categories on the Satellite Server, they are served by the Capsule Server for that location.

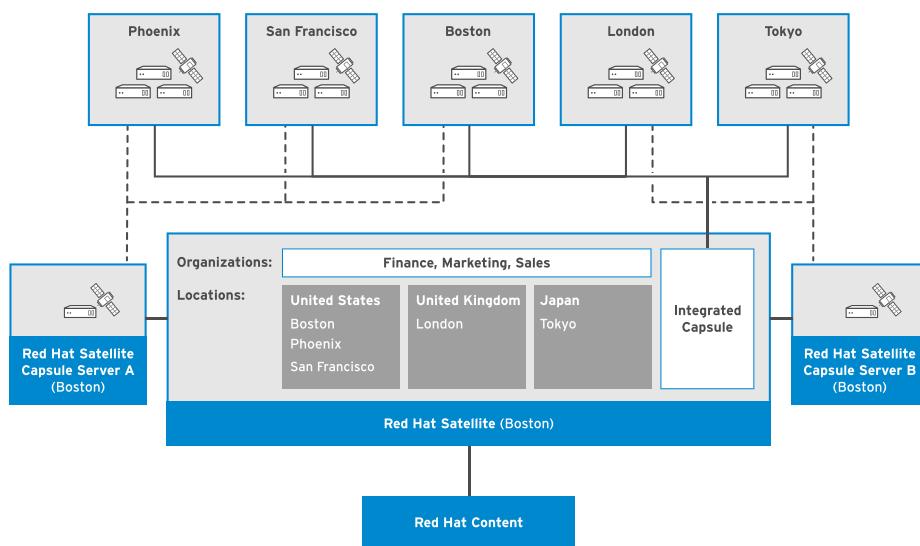
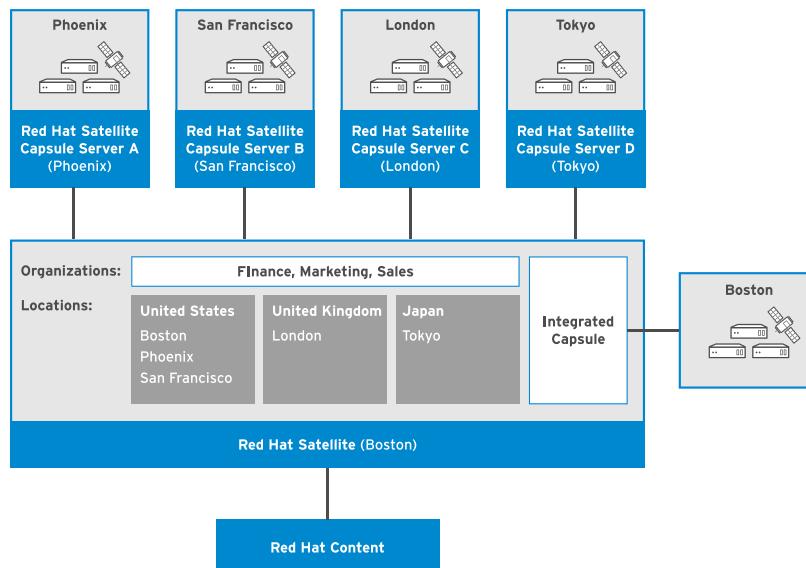


Figure 1.3: Single Satellite Server with integrated capsule and local capsules

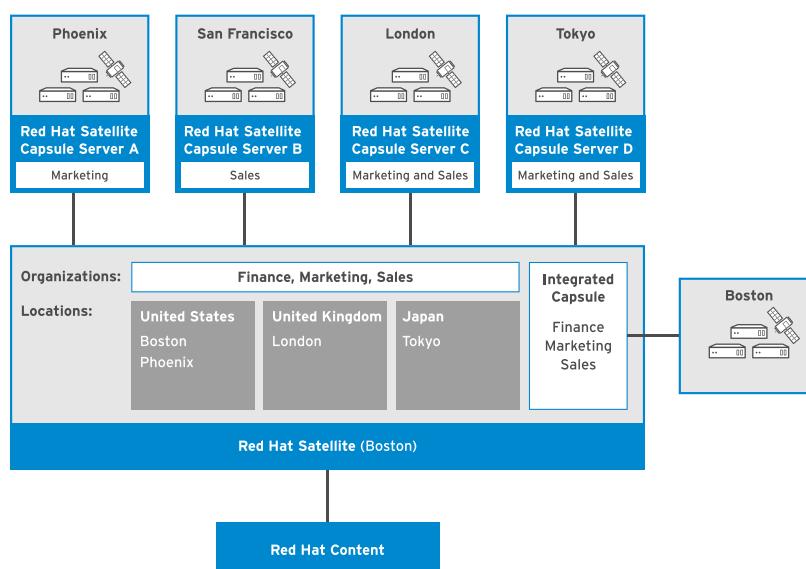
## Satellite Server with Remote Satellite Capsule Servers

An alternative is to collocate Capsule Servers at the remote locations. In this example, a Capsule Server is placed in each remote location, to service the systems within its location. In addition to easing the Satellite Server workload, Capsule Servers serve host content more quickly because the data transfer occurs over a local area network.



**Figure 1.4: Single Satellite with integrated capsule and remote location-based capsules**

In another approach, the Capsule Servers are assigned to organizations. For example, two Capsule Servers are assigned to the Phoenix location, but to different organizations: one for Marketing and the other for Sales. Similarly, two Satellite Capsule Servers are assigned to two organizations: one in London handling both the Sales and Marketing organizations, and the other in Tokyo for the same two organizations. Satellite Server and Capsule Servers can manage any layout of multiple organizations in multiple locations.



**Figure 1.5: Single Satellite with integrated capsule and remote organization-based capsules**

## Adding Subscriptions to Satellite Server

Red Hat Satellite use subscription permissions to provide access to repositories, and their content. A manifest contain one or more subscriptions, enabling access to a collection of repositories. Create a unique manifest for each organization configured in Red Hat Satellite. Manifests do not need to be owned by the same Red Hat Network account.



### Note

Red Hat Satellite 6.6 allows the use of future-dated subscriptions in the manifest.

Before you create your manifest, consider the following:

- Include the Satellite Server subscription in the manifest if planning a disconnected Red Hat Satellite installation.
- Include subscriptions for all Red Hat Satellite Capsule Servers.
- Include subscriptions for all Red Hat products to manage with Red Hat Satellite.
- Plan in advance the manifest renewals based on their expiration dates, and use the future-dated subscriptions to ease this process.

Modify and update the manifest for any organization to include additional infrastructure subscriptions. Do not delete the active manifests in your Satellite installation from the Red Hat Customer Portal, because doing so will unregister all your hosts.

## Configuring the Satellite Content Delivery Network

By default, Red Hat Satellite uses the Red Hat Content Delivery Network (CDN), available at <https://cdn.redhat.com>, to retrieve the latest versions of Red Hat repositories. This CDN is geographically distributed, and guarantees your Satellite installation manages the latest available updates for Red Hat products.

Access to the CDN requires port 443/TCP to be open on your Satellite Server host. To support security policy scenarios in which access to the CDN cannot be enabled, Satellite includes a collection of content ISOs, containing the repositories available in the Red Hat CDN. These content ISOs are located in the Red Hat Customer Portal **Downloads** section for each Red Hat Satellite release. Each ISO includes the packages for a Red Hat product release for a specific architecture. There are two types of content ISOs:

- Base Satellite content ISOs, containing content required to initially populate a new Satellite.
- Incremental Satellite content ISOs, containing content to update repositories already configured in Satellite.

To configure Red Hat Satellite without access to the Red Hat CDN, use these content ISOs to create a custom CDN in your local environment. Content ISOs follow a directory structure which contains the repositories. The directory structure is described in the referenced documentation. Share your custom local CDN directory structure using a local HTTP or HTTPS web server.



### References

For more information, refer to the *Satellite 6 Deployment Planning* chapter in the *Planning for Red Hat Satellite 6* guide at  
[https://access.redhat.com/documentation/en-us/red\\_hat\\_satellite/6.6/html-single/planning\\_for\\_red\\_hat\\_satellite\\_6](https://access.redhat.com/documentation/en-us/red_hat_satellite/6.6/html-single/planning_for_red_hat_satellite_6)

## ► Quiz

# Planning a Red Hat Satellite Deployment

Choose the correct answers to the following questions:

- ▶ **1. Which two of the following statements related to manifests are true? (Choose two.)**
  - a. A manifest file contains your subscription information.
  - b. Red Hat Satellite 6 requires a single manifest for all organizations configured in the Satellite deployment.
  - c. A manifest includes subscriptions for a single Red Hat product.
  - d. Satellite Server allows access to the repositories available in the CDN for the associated subscription.
  
- ▶ **2. Which two of the following need to be considered when creating a subscription manifest? (Choose two.)**
  - a. Add subscriptions for all Capsule Servers you want to create.
  - b. Do not add future-dated subscriptions.
  - c. Create one manifest per organization.
  - d. Create one manifest per location.
  
- ▶ **3. Which two of the following statements related to CDN are valid? (Choose two.)**
  - a. The default CDN is cdn.redhat.com.
  - b. A CDN structure can only contain one repository.
  - c. You can configure a local CDN with content ISO.
  - d. You can associate one or more CDN to an organization.
  
- ▶ **4. Which three of the following deployment scenarios are supported by Red Hat Satellite? (Choose three.)**
  - a. Single location
  - b. Disconnected Satellite with content ISO
  - c. Standalone Capsule Server
  - d. Multiple locations with a Satellite Server and Capsule Servers

## ► Solution

# Planning a Red Hat Satellite Deployment

Choose the correct answers to the following questions:

- ▶ 1. **Which two of the following statements related to manifests are true? (Choose two.)**
  - a. A manifest file contains your subscription information.
  - b. Red Hat Satellite 6 requires a single manifest for all organizations configured in the Satellite deployment.
  - c. A manifest includes subscriptions for a single Red Hat product.
  - d. Satellite Server allows access to the repositories available in the CDN for the associated subscription.
  
- ▶ 2. **Which two of the following need to be considered when creating a subscription manifest? (Choose two.)**
  - a. Add subscriptions for all Capsule Servers you want to create.
  - b. Do not add future-dated subscriptions.
  - c. Create one manifest per organization.
  - d. Create one manifest per location.
  
- ▶ 3. **Which two of the following statements related to CDN are valid? (Choose two.)**
  - a. The default CDN is cdn.redhat.com.
  - b. A CDN structure can only contain one repository.
  - c. You can configure a local CDN with content ISO.
  - d. You can associate one or more CDN to an organization.
  
- ▶ 4. **Which three of the following deployment scenarios are supported by Red Hat Satellite? (Choose three.)**
  - a. Single location
  - b. Disconnected Satellite with content ISO
  - c. Standalone Capsule Server
  - d. Multiple locations with a Satellite Server and Capsule Servers

# Installing Red Hat Satellite

---

## Objectives

After completing this section, you should be able to describe how to perform an initial installation of Red Hat Satellite.

## Red Hat Satellite 6 Requirements

Meet or exceed the hardware and software requirements to ensure a successful Satellite Server installation and maintain adequate performance as the managed environment grows and workloads increase. Administrators should understand how Satellite Server repository content is managed and stored to understand initial space requirements and to forecast future growth.

The Satellite Server system should ideally be a freshly provisioned system dedicated only to hosting Satellite Server. This ensures that no other functions compete for system resources and possibly degrade the Satellite Server performance.

Red Hat Satellite supports the installation of Satellite Server on a host with either a connected or disconnected network.

## Hardware Requirements

The Red Hat Satellite 6 host must meet the following minimum hardware specifications:

- x86\_64 architecture.
- Minimum of four 2.0 GHz CPU cores.
- Minimum of 20 GB of physical memory, and a minimum of 4 GB of swap space.

## Operating System Requirements

Red Hat Satellite 6.6 is supported on the latest version of Red Hat Enterprise Linux 7 Server. When installing RHEL 7 from disc or ISO image, use the default base installation and do not add extra packages. If installing using the kickstart method, only install the @Base package group.

## Firewall, SELinux, and NTP Configuration

To perform software deployment, configuration management, and provisioning, Satellite Server requires that the following traffic be allowed:

Port	Protocol	Service
80	TCP	HTTP
443	TCP	HTTPS
5000	TCP	qpid/Katello
5646	TCP	qpid/Katello

Port	Protocol	Service
5647	TCP	Docker registry/Katello
5671	TCP	amqp
8000	TCP	Anaconda
8140	TCP	Puppet
9090	TCP	Foreman Smart Proxy
7	UDP/TCP	ICMP
53	UDP/TCP	DNS
67	UDP	DHCP
68	UDP	DHCP
69	UDP	TFTP

Use the **firewall-cmd** command to configure access to these network ports. You can configure the ports with the following command:

```
[root@satellite ~]# firewall-cmd --permanent \
--add-port="53/udp" --add-port="53/tcp" \
--add-port="67/udp" --add-port="69/udp" \
--add-port="80/tcp" --add-port="443/tcp" \
--add-port="5000/tcp" --add-port="5647/tcp" \
--add-port="8000/tcp" --add-port="8140/tcp" \
--add-port="9090/tcp"
success
[root@satellite ~]# firewall-cmd --reload
success
```

A **firewalld** service configuration named **RH-Satellite-6** is predefined to simplify management of Satellite's port access requirements. Add the **RH-Satellite-6** service configuration to grant access for the default Satellite service ports.

```
[root@satellite ~]# firewall-cmd --permanent --add-service=RH-Satellite-6
success
[root@satellite ~]# firewall-cmd --reload
success
```

A Red Hat Satellite 6 host must have SELinux permissions set to **enforcing**. This is a requirement for Red Hat support, and is the default SELinux status for Red Hat Enterprise Linux 7 installations.

To ensure accurate time on the Red Hat Satellite 6 host, Red Hat recommends that *chrony* is installed and enabled on all RHEL 7 and RHEL 8 hosts.

## Storage Configuration

The Satellite Server host requires a minimum of 6 GB of storage for the base operating system. In addition, the host requires the following storage space for Satellite Server components and content:

- Minimum of 2 GB for Red Hat Satellite 6 software installation for disconnected installations.
- Minimum of 1 MB (up to 20 GB) for **/var/cache/pulp**, which temporarily stores content during synchronization by Satellite Server.
- Minimum of 1 MB (up to 500 GB) for **/var/lib/pulp**, which stores content synchronized by Satellite Server.
- Minimum of 25 MB in **/var/lib/qpidd** for each content host to be registered with Satellite Server.
- Minimum of 3.5 GB (up to 50 GB) for **/var/lib/mongodb**, which contains Satellite Server's MongoDB database.
- Minimum of 100 MB (up to 10 GB) for **/var/lib/pgsql**, which contains the Satellite Server PostgreSQL database.

Identical packages that exist in multiple repositories are stored only once in Satellite. New repositories containing packages already in existing repositories will use less disk space.

Because most of the data storage on Satellite Server resides under the **/var** directory, Red Hat highly recommends that **/var** be allocated as a separate partition on LVM storage. This configuration makes it easy to allocate additional space as **/var** storage needs increase over time.

The bulk of the storage used by software repositories resides under the **/var/lib/mongodb** and **/var/lib/pulp** directories. Because of the I/O intensive nature of many operations performed in these two directories, Red Hat recommends that they reside on high-bandwidth, low-latency storage, such as solid-state drives (SSD).

## DNS Resolution

After the operating system is installed, administrators require root access to execute the Satellite installation program. Forward and reverse DNS resolution of the fully qualified domain name of the Satellite host is required. Verify that the system's host name and **localhost** resolve properly by using the following commands:

```
[root@satellite ~]# ping -c1 localhost
PING localhost (127.0.0.1) 56(84) bytes of data.
64 bytes from localhost (127.0.0.1): icmp_seq=1 ttl=64 time=0.019 ms

--- localhost ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.019/0.019/0.019/0.000 ms

[root@satellite ~]# ping -c1 $(hostname -s)
PING satellite.lab.example.com (172.25.1.12) 56(84) bytes of data.
64 bytes from satellite.lab.example.com (172.25.1.12): icmp_seq=1 ttl=64
time=0.013 ms

--- satellite.lab.example.com ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
```

```
rtt min/avg/max/mdev = 0.013/0.013/0.013/0.000 ms

[root@satellite ~]# ping -c1 $(hostname -f)
PING satellite.lab.example.com (172.25.1.12) 56(84) bytes of data.
64 bytes from satellite.lab.example.com (172.25.1.12): icmp_seq=1 ttl=64
time=0.009 ms

--- satellite.lab.example.com ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.009/0.009/0.009/0.000 ms
```

Verify reverse DNS resolution.

```
[root@satellite ~]# dig -x 172.25.1.12
...output omitted...
;; ANSWER SECTION:
12.1.25.172.in-addr.arpa. 3600 IN PTR satellite.lab.example.com.
...output omitted...
```

## Browser Support

For the best user experience with the Satellite Server web UI, consult the latest list of supported browsers in the *Red Hat Satellite 6.6 Installation Guide*, which lists the support level and known issues for various browsers.

## Satellite Server Installation Methods

When the operating system is installed on the host system, Satellite Server software can be installed directly from repositories on Red Hat Content Delivery Network or from an ISO installation image.

### Installing from the Red Hat Content Delivery Network

Installing Satellite Server and associated software from Red Hat Content Delivery Network repositories consists of the following steps, executed as the **root** user. The Satellite Server host must be previously registered with the Red Hat Content Delivery Network.

1. Use the **subscription-manager** command to determine if there are subscriptions available for Red Hat Satellite and Red Hat Enterprise Linux. You may get a different output based on your available subscriptions. The Red Hat Satellite subscription also provides an installation of Red Hat Enterprise Linux.

```
[root@satellite ~]# subscription-manager list --available --all
+-----+
 Available Subscriptions
+-----+

Subscription Name: Red Hat Satellite Infrastructure Subscription
Provides:          Red Hat Satellite
                  Red Hat Software Collections (for RHEL Server)
                  Red Hat CodeReady Linux Builder for x86_64
                  Red Hat Ansible Engine
                  Red Hat Enterprise Linux Load Balancer (for RHEL Server)
                  Red Hat
```

```
Red Hat Software Collections (for RHEL Server)
Red Hat Enterprise Linux Server
Red Hat Satellite Capsule
Red Hat Enterprise Linux for x86_64
Red Hat Enterprise Linux High Availability for x86_64
Red Hat Satellite
Red Hat Satellite 5 Managed DB
Red Hat Satellite 6
Red Hat Discovery
SKU: MCT3719
Contract: 11878983
Pool ID: 8a85f99968b92c3701694ee998cf03b8
Provides Management: No
Available: 1
Suggested: 1
Service Level: Premium
Service Type: L1-L3
Subscription Type: Standard
Ends: 03/04/2020
System Type: Physical
```

2. Enable a subscription that provides Red Hat Satellite 6, Red Hat Enterprise Linux, and Red Hat Software Collections, using the pool ID displayed earlier for the subscription.

```
[root@satellite ~]# subscription-manager attach \
--pool=8a85f99968b92c3701694ee998cf03b8
Successfully attached a subscription for: Red Hat Satellite Infrastructure
Subscription
```

3. Disable all existing repositories to ensure that no unexpected software will be installed.

```
[root@satellite ~]# subscription-manager repos --disable "*"
```

4. Enable the Red Hat Satellite, Red Hat Enterprise Linux, and Red Hat Software Collections repositories. Ensure that the Red Hat Enterprise Linux and Red Hat Software Collections repositories being enabled match the operating system version on the Satellite Server host.

```
[root@satellite ~]# subscription-manager repos --enable=rhel-7-server-rpms \
--enable=rhel-server-rhsc1-7-rpms \
--enable=rhel-server-7-satellite-6-rpms \
--enable=rhel-7-server-satellite-maintenance-6-rpms \
--enable=rhel-7-server-ansible-2.8-rpms
```

5. Update all packages.

```
[root@satellite ~]# yum update
```

6. Install the *satellite* package.

```
[root@satellite ~]# yum install satellite
```

## Installation from an ISO Image

In environments with low network bandwidth, it is recommended to download the Satellite Server software as an ISO image so that the installation can be performed from local media. Ensure that your base system is configured with updated packages for the latest version of RHEL 7.

Installing Satellite Server from an ISO consists of the following steps, executed as the **root** user.

1. Import the Red Hat GPG key.

```
[root@satellite ~]# rpm --import /etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release
```

2. Log in to the Red Hat Customer Portal and download the Red Hat Satellite 6 ISO image from [Downloads](#).
3. Create a mount point directory and mount the ISO image.

```
[root@satellite ~]# mkdir /media/iso  
[root@satellite ~]# mount -o loop \  
/tmp/satellite-6.6-rhel-7-x86_64-dvd.iso /media/iso
```

4. Run the Satellite Server installer script in the mounted directory.

```
[root@satellite ~]# cd /media/iso; ./install_packages  
This script will install the foreman packages on the current machine.  
- Ensuring we are in an expected directory.  
- Copying installation files.  
- Creating a Repository File  
- Creating RHSCL Repository File  
- Checking to see if Foreman is already installed.  
- Importing the gpg key.  
- Foreman is not yet installed, installing it.  
- Installation repository will remain configured for future package installs.  
- Installation media can now be safely unmounted.  
  
Install is complete. Please run satellite-installer --scenario satellite.
```

## Satellite Server Initial Configuration

When *katello* is installed, initial configuration must be performed to prepare the Satellite Server for use. The **satellite-installer** command is used to perform this initial configuration.

### Performing Satellite Server Initial Configuration Manually

Before starting the Satellite Server configuration with the **satellite-installer** command, ensure that the **chrony** service is installed and enabled on the Satellite Server host.

```
[root@satellite ~]# yum install chrony  
[root@satellite ~]# systemctl start chronyd  
[root@satellite ~]# systemctl enable chronyd
```

You also need to install the **sos** package on that host.

```
[root@satellite ~]# yum install sos
```

When done, you can use the **satellite-installer** command to configure Satellite Server. Running the command without options configures Satellite Server with default settings. The script provides various options to override default settings. You can display available options to the **satellite-installer** command with the **--help** option.

```
[root@satellite ~]# satellite-installer --help
```

Execute the **satellite-installer** command as the **root** user. You can use the **--foreman-initial-admin-username** and **--foreman-initial-admin-password** options to configure the initial password for the **admin** user in Satellite Server. Upon completion, the command provides the following output:

```
[root@satellite ~]# satellite-installer --scenario satellite \
--foreman-initial-admin-username admin \
--foreman-initial-admin-password redhat
Installing      Done      [100%] [.....]
Success!
* Satellite is running at https://satellite.lab.example.com
  Initial credentials are admin / redhat
* To install additional capsule on separate machine continue by running:

  capsule-certs-generate --capsule-fqdn "$CAPSULE" --certs-tar "~/
$CAPSULE-certs.tar"

The full log is at /var/log/foreman-installer/satellite.log
```

In addition to logging the installation results in **/var/log/foreman-installer/satellite.log**, **satellite-installer** also saves installation parameters to an answer file located at **/etc/foreman-installer/scenarios.d/satellite-answers.yaml**. This file can be referenced to determine the parameters used in the Satellite Server initial configuration.

## Configuring Satellite Server Installation with an Answer File

A copy of the **/etc/foreman-installer/scenarios.d/satellite-answers.yaml** answer file can be modified and used to automate an initial configuration of Satellite Server. This is useful when using custom configuration options with the installer. Configuring Satellite Server with an installer answer file requires the following steps:

1. Make a copy of the default answer file, **/etc/foreman-installer/scenarios.d/satellite-answers.yaml**.

```
[root@satellite ~]# cp \
/etc/foreman-installer/scenarios.d/satellite-answers.yaml \
/etc/foreman-installer/scenarios.d/satellite-answers.custom.yaml
```

2. Use a text editor to modify the custom answer file parameters. You can reference module parameters in each module's **params.pp** file. The following command displays a list of module parameter files:

```
[root@satellite ~]# rpm -ql foreman-installer | grep params.pp
```

3. Edit the following line in `/etc/foreman-installer/scenarios.d/satellite.yaml` to use your custom answer file.

```
:answer_file: /etc/foreman-installer/scenarios.d/satellite-answers.custom.yaml
```

4. Execute the `satellite-installer` command.

```
[root@satellite ~]# satellite-installer --scenario satellite \
--foreman-initial-admin-username admin \
--foreman-initial-admin-password redhat
    Installing      Done      [100%] [.....]
    Success!
    * Satellite is running at https://satellite.lab.example.com
        Initial credentials are admin / redhat
    * To install additional capsule on separate machine continue by running:

        capsule-certs-generate --capsule-fqdn "$CAPSULE" --certs-tar "~/
$CAPSULE-certs.tar"

The full log is at /var/log/foreman-installer/satellite.log
```

## Accessing the Satellite Server Web UI

When the Satellite Server has been installed and configured, log in to the web UI. In a web browser, enter the Satellite Server host name; for example, `https://satellite.lab.example.com`.

When logging in for the first time, a certificate warning displays. Accept the self-signed certificate and add the Satellite Server URL to the security exception list to avoid future warnings.

Enter the username and password specified in the initial Satellite Server configuration. If these were not customized, then use the default username, `admin`, and the randomized initial password, to log in for the first time. When logged in successfully, the Satellite welcome screen appears.



### Note

If you forget the username or password for the Satellite Server `admin` account, use this command to reset the default user `admin` with a new randomized password:

```
[root@satellite ~]# foreman-rake permissions:reset
Reset to user: admin, password: NEW_RANDOMIZED_PASSWORD
```

## Verifying the Status of the Satellite Services

The Satellite Server installation includes the `satellite-maintain` utility to verify your Satellite Server installation health. Use `--help` to display available options. The `satellite-maintain service list` command displays the status of Satellite Server services.

```
[root@satellite ~]# satellite-maintain service list
Running Service List
=====
List applicable services:
dynflowd.service                         enabled
foreman-proxy.service                      enabled
```

```
httpd.service                         enabled
postgresql.service                    enabled
pulp_celerybeat.service              enabled
pulp_resource_manager.service        enabled
pulp_streamer.service                enabled
pulp_workers.service                 enabled
puppetserver.service                 enabled
qdrouterd.service                   enabled
qpidd.service                        enabled
rh-mongodb34-mongod.service         enabled
smart_proxy_dynflow_core.service    enabled
squid.service                        enabled
tomcat.service                       enabled
```

```
All services listed                      [OK]
```

The **satellite-maintain health check** command runs various health checks in the Satellite Server installation.

```
[root@satellite ~]# satellite-maintain health check
Running ForemanMaintain::Scenario::FilteredScenario
=====
Check for verifying syntax for ISP DHCP configurations:          [SKIPPED]
DHCP feature is not enabled
-----
Check whether all services are running:                          [OK]
-----
Check whether all services are running using the ping call:     [OK]
-----
Check for paused tasks:                                         [OK]
-----
Check to verify no empty CA cert requests exist:               [OK]
-----
```



## References

For more information, refer to the *Installing Satellite Server* chapters in the *Quick Start Guide* at  
[https://access.redhat.com/documentation/en-us/red\\_hat\\_satellite/6.6/html-single/quick\\_start\\_guide](https://access.redhat.com/documentation/en-us/red_hat_satellite/6.6/html-single/quick_start_guide)

For more information, refer to the *Installing Satellite Server* chapters in the *Installing Satellite Server from a disconnected network* guide at  
[https://access.redhat.com/documentation/en-us/red\\_hat\\_satellite/6.6/html-single/installing\\_satellite\\_server\\_from\\_a\\_disconnected\\_network](https://access.redhat.com/documentation/en-us/red_hat_satellite/6.6/html-single/installing_satellite_server_from_a_disconnected_network)

For more information, refer to the *Installing Satellite Server* chapters in the *Installing Satellite Server from a connected network* guide at  
[https://access.redhat.com/documentation/en-us/red\\_hat\\_satellite/6.6/html-single/installing\\_satellite\\_server\\_from\\_a\\_connected\\_network](https://access.redhat.com/documentation/en-us/red_hat_satellite/6.6/html-single/installing_satellite_server_from_a_connected_network)

## ► Guided Exercise

# Verifying a Red Hat Satellite Installation

In this exercise, you will verify that the Red Hat Satellite server is installed correctly, including initial environment configuration.

### Outcomes

You should be able to verify that the Red Hat Satellite initial installation meets the prerequisites.

### Before You Begin

Log in as the **student** user on **workstation** using **student** as the password.

Run the **lab deploy-install start** command. This command determines if the **satellite** host is reachable on the network and verifies that Red Hat Satellite is available.

```
[student@workstation ~]$ lab deploy-install start
```

- ▶ 1. Log in to the Satellite web UI as **admin** using **redhat** as the password.
  - 1.1. Use your browser to navigate to <https://satellite.lab.example.com>.
  - 1.2. Log in as **admin** with **redhat** as a password.
- ▶ 2. Verify that the **Default Organization** has no subscriptions available. When done, log out from the Satellite web UI.
  - 2.1. Navigate to **Content** → **Subscriptions**.
  - 2.2. Verify that the **There are no Subscriptions to display** message appears.
  - 2.3. In the upper-right corner, click **Admin User** → **Log Out** to log out from the Satellite web UI.
- ▶ 3. Verify the status of the Red Hat Satellite services.
  - 3.1. On **workstation**, use **ssh** to log in to **satellite** as **student**. Use **sudo -i** to switch to **root**.

```
[student@workstation ~]$ ssh student@satellite  
[student@satellite ~]$ sudo -i  
[sudo] password for student: student  
[root@satellite ~]#
```

- 3.2. Use the **satellite-maintain service list** command to verify the status of the Red Hat Satellite services.

```
[root@satellite ~]# satellite-maintain service list
Running Service List
=====
List applicable services:
dynflowd.service           enabled
foreman-proxy.service       enabled
httpd.service               enabled
postgresql.service          enabled
pulp_celerybeat.service    enabled
pulp_resource_manager.service enabled
pulp_streamer.service      enabled
pulp_workers.service        enabled
puppetserver.service        enabled
qdrouterd.service          enabled
qpidd.service               enabled
rh-mongodb34-mongod.service enabled
smart_proxy_dynflow_core.service enabled
squid.service               enabled
tomcat.service              enabled

All services listed [OK]
-----
```

- 4. Run health checks for Red Hat Satellite. You may need to provide the password for the **admin** user (**redhat**).

```
[root@satellite ~]# satellite-maintain health check
Running ForemanMaintain::Scenario::FilteredScenario
=====
Check for verifying syntax for ISP DHCP configurations: [SKIPPED]
DHCP feature is not enabled
-----
Check whether all services are running: [OK]
-----
Check whether all services are running using the ping call: [OK]
-----
Check for paused tasks: [OK]
-----
Check to verify no empty CA cert requests exist: [OK]
-----
```

- 5. Verify that all services and ports required by Red Hat Satellite are enabled in **firewalld**.

```
[root@satellite ~]# firewall-cmd --list-all
public (active)
  target: default
  icmp-block-inversion: no
  interfaces: eth0
  sources:
  services: dhcpcv6-client ssh
  ports: 53/udp 53/tcp 67/udp 69/udp 80/tcp 443/tcp 5000/tcp 5647/tcp 8000/tcp
         8140/tcp 9090/tcp
```

```
protocols:  
masquerade: no  
forward-ports:  
source-ports:  
icmp-blocks:  
rich rules:
```

► 6. Verify that **chronyd** is active and enabled.

```
[root@satellite ~]# systemctl status chronyd  
● chronyd.service - NTP client/server  
  Loaded: loaded (/usr/lib/systemd/system/chronyd.service; enabled; vendor  
  preset: enabled)  
  Active: active (running) since jue 2019-09-26 08:08:04 UTC; 3h 7min ago  
    ...output omitted...
```

► 7. Ensure that **localhost** and the host name for the **satellite** server resolve correctly.

7.1. Verify **localhost** resolution:

```
[root@satellite ~]# ping -c1 localhost  
PING localhost (127.0.0.1) 56(84) bytes of data.  
64 bytes from localhost (127.0.0.1): icmp_seq=1 ttl=64 time=0.087 ms  
  
--- localhost ping statistics ---  
1 packets transmitted, 1 received, 0% packet loss, time 0ms  
rtt min/avg/max/mdev = 0.087/0.087/0.087/0.000 ms
```

7.2. Verify the Satellite Server host name resolution.

```
[root@satellite ~]# ping -c1 satellite.lab.example.com  
PING satellite.lab.example.com (172.25.250.15) 56(84) bytes of data.  
64 bytes from satellite.lab.example.com (172.25.250.15): icmp_seq=1 ttl=64  
time=0.066 ms  
  
--- satellite.lab.example.com ping statistics ---  
1 packets transmitted, 1 received, 0% packet loss, time 0ms  
rtt min/avg/max/mdev = 0.066/0.066/0.066/0.000 ms
```

7.3. Verify the Satellite Server reverse DNS resolution.

```
[root@satellite ~]# host 172.25.250.15  
15.250.25.172.in-addr.arpa domain name pointer satellite.lab.example.com.
```

► 8. Log off from the **satellite** host.

```
[root@satellite ~]# exit  
logout  
[student@satellite ~]$ exit  
logout  
Connection to satellite closed.
```

## Finish

As the **student** user on the **workstation** machine, use the **lab deploy-install finish** command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab deploy-install finish
```

This concludes the guided exercise.

# Configuring Organizations and Content Manifests

---

## Objectives

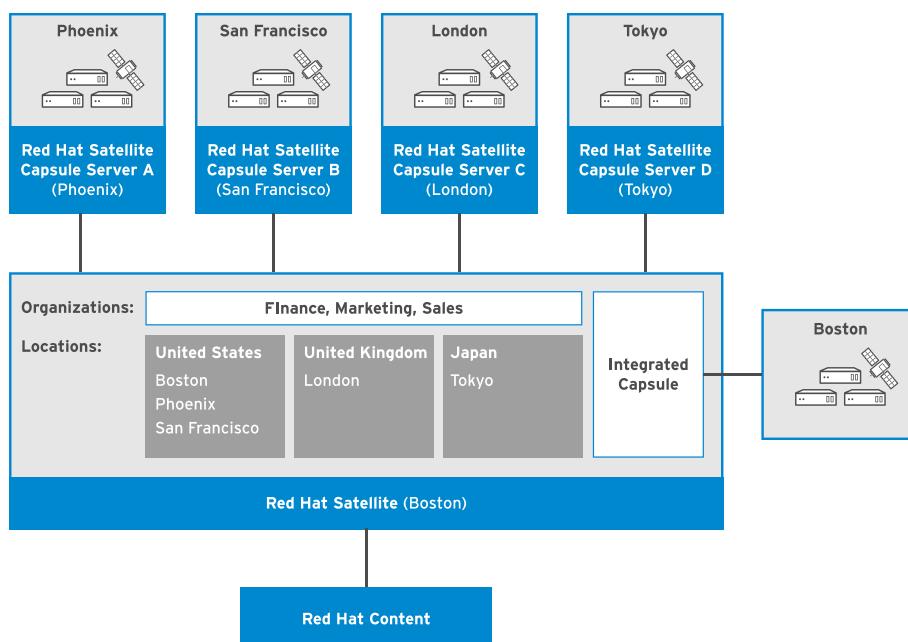
After completing this section, you should be able to describe and configure organizations in Red Hat Satellite, and create and install content manifests for organizations.

## Managing Systems by Organization and Location

Red Hat Satellite 5 supported the creation and management of multiple organizations within one Satellite installation. This allowed for the division of systems, content, and subscriptions across different organizations or groups. Satellite 6 provides yet another context for managing systems with the introduction of *locations*.

Within a single Satellite installation, administrators can define multiple organizations and multiple locations. Organizations can represent departments such as Finance, Marketing, and Sales. Locations are physical places of system installation. Administrators can create high-level locations, such as countries. In addition, administrators can create even more specific locations, such as cities, and nest these specific locations within high-level locations to create a hierarchical location tree.

As mentioned previously, the Satellite infrastructure can be extended with the addition of either local or remote Satellite Capsule Servers. Satellite Capsule Servers can be assigned either by organization or by location. In the following diagram, a single Satellite Server instance manages four organizations across five locations, thereby creating nine contexts for the management of the systems.



**Figure 1.6: Satellite 6 organizations and locations topology**

This Satellite infrastructure also includes four Satellite Capsule Servers, each assigned to a different geographic location. The management function resides only on the Satellite Server; content and configuration is synchronized between the Satellite Server and the Satellite Capsule Servers assigned to the various locations. The managed systems in each location are managed by Satellite Server, but fetch content and configuration from the Satellite Capsule Server assigned to their location.

## Managing Organizations

Within Satellite Server, content-related functions, such as the management of products, repositories, and content views, are specific to organizations. Therefore, the management of organizations is a task that administrators must complete early in the implementation of their Satellite infrastructure. The Satellite Server installation, by default, includes an organization called **Default Organization**. In Red Hat Satellite 6, you can create and manage multiple organizations and then divide and assign your Red Hat subscriptions to each individual organization.

To create an organization, navigate to **Administer** → **Organizations** and click **New Organization**. Create a name for the organization and a unique identifier as the organization's label. The label is used for creating and mapping certain assets, such as directories for content storage. When creating a label, use letters, numbers, underscores, and dashes, but not spaces. Additionally, you have the option to enter a description for the organization.

You can edit existing values, or assign infrastructure resources that you want to add to an organization, such as networking, installation media, and kickstart templates.

To edit a organization, navigate to **Administer** → **Organizations** and click the name of the organization you want to edit. Alternatively, you can click **Edit** in the **Actions** column in the same row as the organization you want to edit. Choose the category you want to edit from the left side of the **Organizations** page. For example, click **Primary** to edit basic values such as the organization's name, label, or description.

To delete an organization, navigate to **Administer** → **Organizations** and select **Delete** from the **Edit** list to the right of the name of the organization you want to delete.

## Managing Locations

Organizations divide Red Hat Satellite 6 resources into logical groups based on ownership, purpose, content, security level, or other divisions. Locations divide organizations into logical groups based on geographical location.

To create a location, navigate to **Administer** → **Locations** and click **New Location**. If this is to be a sublocation of another location, select that other location in the **Parent** field. Nested locations are convenient for creating management groups based on regional asset authority in your organization. Create a meaningful name for the location.

You can edit existing values, or assign infrastructure resources that you want to add to a location, such as networking, installation media, and kickstart templates.

To edit a location, navigate to **Administer** → **Locations** and click the name of the location you want to edit. Alternatively, you can click **Edit** in the **Actions** column in the same row as the location you want to edit. Choose the category you want to edit from the left side of the **Locations** page. For example, click **Primary** to edit the location's parent, name, or description.

You can delete a location if the location is not associated with any life-cycle environments or host groups. If there are any life-cycle environments or host groups associated with the location you are about to delete, remove them prior to deleting the location.



### Important

Do not delete the default location created during installation because the default location is a placeholder for any unassociated hosts in the Satellite environment. There must be at least one location in the environment at any given time.

To delete a location, navigate to **Administer** → **Locations** and select **Delete** from the **Edit** list to the right of the name of the location you want to delete.

## Managing Subscriptions and Content

Subscription management provides organizations a method to manage their Red Hat subscription information. Content management provides organizations a method to manage the software installed on systems.

One of the major roles of Satellite Server is to synchronize content from the Red Hat Content Delivery Network (CDN) to provide a local software repository of Red Hat content to subscribed hosts. Content includes packages, errata updates, kickstart trees, and installation ISO images. Satellite Server access to Red Hat Network content is governed by a subscription manifest. Users can obtain a subscription manifest through the Red Hat Customer Portal.

### Creating a Subscription Manifest

To create a subscription manifest, access the Red Hat Customer Portal and log in to the Red Hat account that you used to register the system to Red Hat Subscription Management. Navigate to **Subscriptions** in the upper-left corner of the Customer Portal. Click **Subscription Allocations** and then click **Create New subscription allocation**. Create a name for your manifest and select a type and version of the subscription management application you are using, such as Satellite 6.6.

To add subscriptions to your manifest, navigate to **Subscription Allocations** on the Customer Portal and click the name of the manifest you want to add subscriptions to. Click **Subscriptions** and then click **Add Subscriptions**.

A list of your Red Hat product subscriptions displays. Enter the required number of entitlements for each product you want in the manifest in the **Entitlement Quantity** field.

### Exporting a Subscription Manifest from the Customer Portal

While viewing a subscription allocation that has at least one subscription, you can export a manifest from either of two places:

On the Customer Portal, navigate to **Subscription Allocations** and click the name of the subscription manifest that you want to export. On either the **Subscriptions** or **Details** tab, click **Export Manifest**. The subscription manifest will download to your local system.

When the subscription manifest is exported, the Customer Portal encodes the selected subscription certificates and creates a compressed file which can be uploaded into the Satellite Server.

## Importing a Subscription Manifest into Satellite Server

A Satellite Server with access to Red Hat Network will be able to synchronize Red Hat content only after a subscription manifest is installed. A subscription manifest must be assigned to an organization within the Satellite Server.

In the Satellite web UI, ensure the context is set to the organization you want to associate with the subscription manifest. To import the subscription manifest, navigate to **Content** → **Subscriptions** and click **Manage Manifest** to display the manifest page for the organization.

In the **Red Hat Provider Details** section, set the correct location for **Red Hat CDN URL**. The default location is <https://cdn.redhat.com>, but for a disconnected Satellite set the location to where you are hosting your local content ISO.

In the **Subscription Manifest** section, click **Browse** to locate the manifest downloaded from the Customer Portal.



### References

For more information, refer to the *Creating a Subscription Allocation in Customer Portal* section in the *Red Hat Satellite 6.6 Installing Satellite Server from a Disconnected Network* at  
[https://access.redhat.com/documentation/en-us/red\\_hat\\_satellite/6.6/html-single/installing\\_satellite\\_server\\_from\\_a\\_disconnected\\_network/index#Managing\\_Subscriptions-Creating\\_a\\_Subscription\\_Manifest](https://access.redhat.com/documentation/en-us/red_hat_satellite/6.6/html-single/installing_satellite_server_from_a_disconnected_network/index#Managing_Subscriptions-Creating_a_Subscription_Manifest)

## ► Guided Exercise

# Configuring Organizations and Content Manifests

In this exercise, you will install a content manifest for an organization in Red Hat Satellite Server.

### Outcomes

You should be able to:

- Add an organization.
- Add a location.
- Install a content manifest for an organization.

### Before You Begin

Log in as the **student** user on the **workstation** VM, using **student** as the password.

Run the **lab deploy-organizations start** command. This command determines if the **satellite** host is reachable on the network and prepares it for this exercise.

```
[student@workstation ~]$ lab deploy-organizations start
```

- ▶ 1. Log in to the Satellite Server web interface, <https://satellite.lab.example.com>, as the **admin** user using **redhat** as the password.
- ▶ 2. Create the **Operations** organization.
  - 2.1. Click **Administer** → **Organizations**.
  - 2.2. On the **Organizations** page, click **New Organization**.
  - 2.3. Enter **Operations** in the **Name** and **Label** fields.
  - 2.4. Enter **Operations Department** in the **Description** field, and then click **Submit**.
- ▶ 3. Create the **Boston** location.
  - 3.1. Click **Administer** → **Locations**.
  - 3.2. On the **Locations** page, click **New Location**.
  - 3.3. Enter **Boston** in the **Name** field, and then click **Submit**.
- ▶ 4. Associate the **Boston** location with the **Operations** organization.
  - 4.1. Click **Administer** → **Locations**.

- 4.2. On the **Locations** page, click the **Boston** link.
  - 4.3. Click **Organizations** in the left navigation bar to associate the location **Boston** with an organization.
  - 4.4. Click the organization **Operations** to add it to the **Selected items** list. You may need to change the screen resolution to avoid rendering issues.
  - 4.5. Click **Submit**.
- ▶ 5. Configure the **Operations** organization to use the offline CDN available at `http://content.example.com/rhs6.6/x86_64/cdn`, and the manifest available at `http://materials.example.com/manifest_operations.zip`.
- 5.1. Click **Content** → **Subscriptions** and then click **Import a Manifest**.
  - 5.2. Update the **Red Hat CDN URL** field to `http://content.example.com/rhs6.6/x86_64/cdn`, and click **Update**. Do not click **Close**.
  - 5.3. Open a terminal, and download the manifest available at `http://materials.example.com/manifest_operations.zip`.

```
[student@workstation ~]$ wget \
http://materials.example.com/manifest_operations.zip
...output omitted...
```

- 5.4. Return to the Satellite web UI, click **Browse** and select the **manifest\_operations.zip** file from your **/home/student** directory.
  - 5.5. Verify that subscriptions attached to the manifest are now listed under **Content** → **Subscriptions**.
- ▶ 6. Use the **hammer** command to verify that the **Operations** organization and **Boston** location exist and are associated with one another.
- 6.1. On **workstation**, use **ssh** to log in to **satellite** as **student**. Use **sudo -i** to switch to **root**.

```
[student@workstation ~]$ ssh student@satellite
[student@satellite ~]$ sudo -i
[sudo] password for student: student
[root@satellite ~]#
```

- 6.2. Use the **hammer organization list** command to verify the **Operations** organization.
- ```
[root@satellite ~]# hammer --output json organization list
...output omitted...
[
  {
    "Id": 1,
    "Title": "Default Organization",
    "Name": "Default Organization",
```

```
        "Description": null,
        "Label": "Default_Organization"
    },
    {
        "Id": 3,
        "Title": "Operations",
        "Name": "Operations",
        "Description": "Operations Department",
        "Label": "Operations"
    }
]
```

- 6.3. Use the **hammer location list** command to verify that the **Boston** location is available.

```
[root@satellite ~]# hammer --output json location list
[
{
    {
        "Id": 4,
        "Title": "Boston",
        "Name": "Boston",
        "Description": ""
    },
    {
        "Id": 2,
        "Title": "Default Location",
        "Name": "Default Location",
        "Description": null
    }
]
```

- 6.4. Use the **hammer location list** command to verify that only the **Boston** location is associated with the **Operations** organization.

```
[root@satellite ~]# hammer --output json location list \
--organization "Operations"
[
{
    {
        "Id": 4,
        "Title": "Boston",
        "Name": "Boston",
        "Description": ""
    }
]
```

- 6.5. Log off from the **satellite** host.

```
[root@satellite ~]# exit
logout
[student@satellite ~]$ exit
logout
Connection to satellite closed.
```

## Finish

As the **student** user on the **workstation** machine, use the **lab deploy-organizations finish** command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab deploy-organizations finish
```

This concludes the guided exercise.

## ▶ Lab

# Planning and Deploying Red Hat Satellite

### Performance Checklist

In this lab, you will confirm that your Red Hat Satellite server is properly installed and operating, with a correctly named organization and subscription manifest.

### Outcomes

You should be able to:

- Configure an organization and a location.
- Associate a location with an organization.
- Verify the status of Red Hat Satellite services.
- Verify the prerequisites for Red Hat Satellite.

### Before You Begin

Log in to **workstation** as **student** using **student** as the password.

On **workstation**, run the **lab deploy-review start** command. This command determines if the **satellite** host is reachable on the network and verifies that Red Hat Satellite is available.

```
[student@workstation ~]$ lab deploy-review start
```

1. Log in to the Satellite web UI at <https://satellite.lab.example.com> as **admin** using **redhat** as the password.
2. Create an organization named **Finance** with the label also named **Finance** and **Finance Department** as the description.
3. Create the **San Francisco** and **Tokyo** locations.
4. Associate the **San Francisco** and **Tokyo** locations with the **Finance** organization.
5. Configure the **Finance** organization to use the offline CDN available at [http://content.example.com/rhs6.6/x86\\_64/cdn](http://content.example.com/rhs6.6/x86_64/cdn), and the manifest available at [http://materials.example.com/manifest\\_finance.zip](http://materials.example.com/manifest_finance.zip).
6. Verify the status of the Red Hat Satellite services.
7. Verify that the **satellite** server supports the system prerequisites for Red Hat Satellite on firewall ports, time synchronization, and DNS resolution.

### Evaluation

As the **student** user on the **workstation** machine, use the **lab deploy-review grade** command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab deploy-review grade
```

## Finish

As the **student** user on the **workstation** machine, use the **lab deploy-review finish** command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab deploy-review finish
```

This concludes the lab.

## ► Solution

# Planning and Deploying Red Hat Satellite

### Performance Checklist

In this lab, you will confirm that your Red Hat Satellite server is properly installed and operating, with a correctly named organization and subscription manifest.

### Outcomes

You should be able to:

- Configure an organization and a location.
- Associate a location with an organization.
- Verify the status of Red Hat Satellite services.
- Verify the prerequisites for Red Hat Satellite.

### Before You Begin

Log in to **workstation** as **student** using **student** as the password.

On **workstation**, run the **lab deploy-review start** command. This command determines if the **satellite** host is reachable on the network and verifies that Red Hat Satellite is available.

```
[student@workstation ~]$ lab deploy-review start
```

1. Log in to the Satellite web UI at <https://satellite.lab.example.com> as **admin** using **redhat** as the password.
  - 1.1. Use your browser to navigate to <https://satellite.lab.example.com>.
  - 1.2. Log in as **admin** with **redhat** as a password.
2. Create an organization named **Finance** with the label also named **Finance** and **Finance Department** as the description.
  - 2.1. Click **Administer** → **Organizations**.
  - 2.2. On the **Organizations** page, click **New Organization**.
  - 2.3. Enter **Finance** in the **Name** and **Label** fields.
  - 2.4. Enter **Finance Department** in the **Description** field, and then click **Submit**.
3. Create the **San Francisco** and **Tokyo** locations.
  - 3.1. Click **Administer** → **Locations**.
  - 3.2. On the **Locations** page, click **New Location**.
  - 3.3. Enter **San Francisco** in the **Name** field, and then click **Submit**.

- 3.4. Repeat the above steps to create the **Tokyo** location.
4. Associate the **San Francisco** and **Tokyo** locations with the **Finance** organization.
  - 4.1. Click **Administer** → **Locations**.
  - 4.2. On the **Locations** page, click the **San Francisco** link.
  - 4.3. Click the **Organizations** tab in the left navigation bar to associate the location **San Francisco** with an organization.
  - 4.4. Click the **Finance** organization to add it to the **Selected items** list, and then click **Submit**.
  - 4.5. Repeat the above steps to associate the **Tokyo** location with the **Finance** organization.
5. Configure the **Finance** organization to use the offline CDN available at `http://content.example.com/rhs6.6/x86_64/cdn`, and the manifest available at `http://materials.example.com/manifest_finance.zip`.
  - 5.1. Verify that the **Finance** organization is selected in the top navigation bar.
  - 5.2. Navigate to **Content** → **Subscriptions** and click **Import a Manifest**.
  - 5.3. Update the **Red Hat CDN URL** field to `http://content.example.com/rhs6.6/x86_64/cdn`, and click **Update**. Do not click **Close**.
  - 5.4. Open a terminal, and download the manifest available at `http://materials.example.com/manifest_finance.zip`.

```
[student@workstation ~]$ wget \
http://materials.example.com/manifest_finance.zip
...output omitted...
```

- 5.5. Return to the Satellite web UI, click **Browse** and select the **manifest\_finance.zip** file from your **/home/student** directory.
- 5.6. Verify that the subscriptions attached to the manifest are now listed under **Content** → **Subscriptions**.
6. Verify the status of the Red Hat Satellite services.

- 6.1. Go back to the terminal, and use **ssh** to log in to **satellite** as **student**. Use **sudo -i** to switch to **root**.

```
[student@workstation ~]$ ssh student@satellite
[student@satellite ~]$ sudo -i
[sudo] password for student: student
[root@satellite ~]#
```

- 6.2. Use the **satellite-maintain service list** command to verify the status for Red Hat Satellite services.

```
[root@satellite ~]# satellite-maintain service list
Running Service List
=====
```

```
List applicable services:  
dynflowd.service                         enabled  
foreman-proxy.service                     enabled  
httpd.service                            enabled  
postgresql.service                       enabled  
pulp_celerybeat.service                  enabled  
pulp_resource_manager.service            enabled  
pulp_streamer.service                   enabled  
pulp_workers.service                    enabled  
puppetserver.service                   enabled  
qdrouterd.service                      enabled  
qpidd.service                           enabled  
rh-mongodb34-mongod.service             enabled  
smart_proxy_dynflow_core.service        enabled  
squid.service                           enabled  
tomcat.service                          enabled
```

```
All services listed [OK]  
-----
```

- 6.3. Use the **satellite-maintain health check** command to check the health of Red Hat Satellite.

```
[root@satellite ~]# satellite-maintain health check  
Running ForemanMaintain::Scenario::FilteredScenario  
=====  
Check for verifying syntax for ISP DHCP configurations: [SKIPPED]  
DHCP feature is not enabled  
-----  
Check whether all services are running: [OK]  
-----  
Check whether all services are running using the ping call: [OK]  
-----  
Check for paused tasks: [OK]  
-----  
Check to verify no empty CA cert requests exist: [OK]  
-----
```

7. Verify that the **satellite** server supports the system prerequisites for Red Hat Satellite on firewall ports, time synchronization, and DNS resolution.

- 7.1. Verify that both services and ports required by Red Hat Satellite are enabled in FirewallD.

```
[root@satellite ~]# firewall-cmd --list-all  
public (active)  
  target: default  
  icmp-block-inversion: no  
  interfaces: eth0  
  sources:  
    services: dhcpcv6-client http https ssh  
    ports: 53/udp 53/tcp 67/udp 69/udp 80/tcp 443/tcp 5000/tcp 5647/tcp 8000/tcp  
    8140/tcp 9090/tcp  
  protocols:  
  masquerade: no
```

```
forward-ports:  
source-ports:  
icmp-blocks:  
rich rules:
```

7.2. Verify that **chrony** is active and enabled.

```
[root@satellite ~]# systemctl status chronyd  
● chronyd.service - NTP client/server  
   Loaded: loaded (/usr/lib/systemd/system/chronyd.service; enabled; vendor  
   preset: enabled)  
     Active: active (running) since jue 2019-09-26 08:08:04 UTC; 3h 7min ago  
       ...output omitted...
```

7.3. Ensure that **localhost** and the host name for the **satellite** server resolve correctly.

```
[root@satellite ~]# ping -c1 localhost  
PING localhost (127.0.0.1) 56(84) bytes of data.  
64 bytes from localhost (127.0.0.1): icmp_seq=1 ttl=64 time=0.087 ms  
  
--- localhost ping statistics ---  
1 packets transmitted, 1 received, 0% packet loss, time 0ms  
rtt min/avg/max/mdev = 0.087/0.087/0.087/0.000 ms  
[root@satellite ~]# ping -c1 satellite.lab.example.com  
PING satellite.lab.example.com (172.25.250.15) 56(84) bytes of data.  
64 bytes from satellite.lab.example.com (172.25.250.15): icmp_seq=1 ttl=64  
time=0.066 ms  
  
--- satellite.lab.example.com ping statistics ---  
1 packets transmitted, 1 received, 0% packet loss, time 0ms  
rtt min/avg/max/mdev = 0.066/0.066/0.066/0.000 ms
```

7.4. Ensure reverse DNS is operational.

```
[root@satellite ~]# host 172.25.250.15  
15.250.25.172.in-addr.arpa domain name pointer satellite.lab.example.com.
```

7.5. Log off from the **satellite** host.

```
[root@satellite ~]# exit  
logout  
[student@satellite ~]$ exit  
logout  
Connection to satellite closed.
```

This concludes the lab.

## Evaluation

As the **student** user on the **workstation** machine, use the **lab deploy-review grade** command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab deploy-review grade
```

## Finish

As the **student** user on the **workstation** machine, use the **lab deploy-review finish** command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab deploy-review finish
```

# Summary

---

In this chapter, you learned:

- Red Hat Satellite is a systems management tool that can be used to configure new systems and provide software updates from the Red Hat Customer Portal.
- Satellite components include Satellite Server and Capsule Server, which provides a proxy for many Satellite functions.
- By default, Satellite supports both Red Hat CDN, located at <https://cdn.redhat.com>, and local offline CDN following the appropriate directory structure.
- Satellite supports both connected and disconnected installation methods.
- Organizations and locations support content-related functions, such as the management of products, repositories, and content views.

## Chapter 2

# Managing Software Life Cycles

### Goal

Create and manage Red Hat software deployment life-cycle environments.

### Objectives

- Enable repositories and create products in the Red Hat Satellite library, and configure sync plans to keep the library current.
- Define a workflow for software promotion by creating life-cycle environments and organizing them into an environment path.
- Create and publish content views and promote them to life-cycle environments on an environment path.

### Sections

- Synchronizing Red Hat Content (and Guided Exercise)
- Creating Software Life Cycles (and Guided Exercise)
- Publishing and Promoting Content Views (and Guided Exercise)

### Lab

Managing Software Life Cycles

# Synchronizing Red Hat Content

---

## Objectives

After completing this section, you should be able to:

- Enable repositories and products in the Red Hat Satellite library.
- Configure sync plans to keep the library up-to-date.

## Describing Red Hat Content

In Red Hat Satellite Server, *content* refers to the software on hosts that Satellite Server manages. Content includes base operating system packages, middleware services, and user applications. Satellite Server manages subscriptions and content for Red Hat Enterprise Linux hosts.

Satellite Server stores Red Hat content to associate with different organizations to serve their varying business needs. For example, a company with three organizations, Development, Finance, and Operations, purchases a Red Hat subscription that includes Red Hat Enterprise Linux (RHEL), Red Hat OpenShift Container Platform (RHOCOP), and Red Hat Satellite Server. Based on the business needs of each organization, the company only requires RHOCOP for Development and Finance, but requires RHEL and Satellite Server for Operations.

The company uses the Red Hat Customer Portal to create three unique subscription manifests, one for each organization with the assigned products, and imports each manifest into Satellite Server in the appropriate organization context. After the manifests are successfully imported, the available subscriptions for the enabled products is listed under **Content → Subscriptions** in the Satellite Server web UI in each organization context.

For each product in an organization, a set of repositories is enabled. Hosts in different organizations receive package updates aligned to their organization's assigned products. For example, the Development and Finance organization hosts receive package updates from the RHOCOP repositories. The hosts in the Operations organization only receive package updates from the RHEL and Satellite Server repositories. This method of content management allows organizations in the same company to better manage systems using different products to support varying business needs.

## Managing Red Hat Products and Repositories

A product in Satellite Server groups related repositories together. The product's repositories consist of different versions, architectures, and add-ons. The correct product is automatically created when you enable a Red Hat source repository. For example, if you enable the **Red Hat Enterprise Linux 8 for x86\_64 - BaseOS RPMs x86\_64 8.1** and the **Red Hat Enterprise Linux 8 for x86\_64 - AppStream RPMs x86\_64 8.1** repositories, the **Red Hat Enterprise Linux for x86\_64** product is automatically created and contains the two repositories. Products in Satellite Server enable synchronization between repositories that are dependent on each other. Managing Satellite Server products and repositories requires administrative privileges.

Within a particular organization and location context, navigate to **Content → Products** to view the available products. Click the name of any product to list the product's repositories.

## Adding Red Hat Product Repositories

You can register multiple hosts with different versions of RHEL to Satellite Server, and use the Satellite system management services to manage all of those hosts. Enabling the appropriate repositories in Satellite Server addresses the requirement of specific repositories for the RHEL hosts. Enabling the repositories results in the automatic creation of the new product in the Satellite Server if the repositories belong to a product different from those listed. If the repositories belong to one of the already listed products, however, the repositories are added to that product.

### To Add a Repository:

- Choose the required organization and location from the main menu.
- Navigate to **Content → Red Hat Repositories** to list the available and enabled repositories for the current organization.
- Enter the name of the required repository and then click **Search**. Any matched repositories appear under **Available Repositories**.
- Expand the required repository and click the plus sign (+) next to the repository architecture information to enable it.

## Removing Red Hat Product Repositories

When a repository is no longer needed, you should remove it to free up disk space.

### To Remove a Repository:

- Choose the required organization and location from the main menu.
- Navigate to **Content → Red Hat Repositories**.
- In the **Enabled Repositories** list, click the minus sign (-) next to the repository to remove.

## Describing Content Synchronization

Satellite Server uses *content synchronization* to maintain an exact copy of the repositories in the Red Hat Content Delivery Network (CDN). Satellite Server stores the retrieved content on its local file system. Initial content synchronization is performed manually, after which you create a sync plan to perform periodic synchronization.

### Synchronizing Red Hat Product Repositories

- To manually synchronize repositories, navigate to **Content → Products**.
- Click the name of the product whose repositories you want to synchronize.
- Select the check box for the repositories to synchronize, and click **Sync Now**.



#### Note

Synchronization time depends on the repository size and your environment's network speed.

## Creating Sync Plans

Local Red Hat product repositories on Satellite Server should be kept synchronized with their respective source repositories. This keeps the local repositories current and makes security advisories, bug fixes, and product enhancements readily available for deployment soon after their release. A **sync plan** can help automate repository synchronization. You can use the Satellite Server web UI or the **hammer** command line interface to create sync plans. Only Satellite administrators can create and manage sync plans.

### To Create a Sync Plan:

- Navigate to **Content** → **Sync Plans** and then click **Create Sync Plan**.
- On the **New Sync Plan** page, complete the following details:

| Field              | Description                                                                                                                                          |
|--------------------|------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Name</b>        | A name for the sync plan.                                                                                                                            |
| <b>Description</b> | A meaningful description of the sync plan.                                                                                                           |
| <b>Interval</b>    | The interval for the sync plan to run at. Choose from hourly, daily, and weekly, or choose <b>custom cron</b> to create your own sync plan schedule. |
| <b>Start Date</b>  | The date the sync plan starts.                                                                                                                       |
| <b>Start Time</b>  | Time of the day the sync plan starts.                                                                                                                |

After you have specified all the required fields, click **Save** to create the sync plan.

## Automating Product Synchronization with Sync Plans

After you have created a sync plan, you can apply it to a product to automate repository synchronization.

### To Automate Product Synchronization with a Sync Plan:

- Choose the required organization and location from the main menu.
- Navigate to **Content** → **Sync Plans** to view the existing sync plans.
- Click the name of the required sync plan and then click the **Products** tab.
- Click the **Add** tab and select the check box for the intended product, and then click **Add Selected**.

### To Remove a Product from a Sync Plan:

- Navigate to **Content** → **Sync Plans** to view the existing sync plans.
- Click the name of the required sync plan and then click the **Products** tab.
- Click the **List/Remove** tab, select the check box for the product you want to remove, and then click **Remove Selected**.

## Describing Download Policies

Satellite Server uses download policies to control downloading content during content synchronization. The **On Demand** download policy downloads only the metadata during the synchronization. After the initial synchronization, packages are downloaded only when hosts request them.

The **Background** download policy runs a background task after the synchronization to download all packages. This download policy is applicable only to the content of **yum** repositories.

The **Immediate** download policy downloads all the metadata and the packages immediately during the synchronization.

### To Set the Default Repository Download Policy:

- Navigate to **Administer** → **Settings** and then click the **Content** tab.
- In the **Name** column, locate the **Default Repository download policy** entry and click the pencil icon in the **Value** column.
- Select the required download policy from the list, and confirm your selection.

## Synchronizing Content Using the CLI

You can use the following **hammer** command to synchronize a repository using a unique identifier.

```
[root@server]# hammer repository list  
[root@server]# hammer repository synchronize --id 5
```

To specify the organization context while synchronizing the repository, use the **--organization** option with the **hammer** command.

```
[root@server]# hammer repository synchronize --id 5 --organization myorg
```



### References

For more information, refer to *Synchronizing Red Hat Repositories* in the *Red Hat Satellite 6.6 Content Management Guide* at  
[https://access.redhat.com/documentation/en-us/red\\_hat\\_satellite/6.6/html-single/content\\_management\\_guide/importing\\_red\\_hat\\_content#Importing\\_Red\\_Hat\\_Content-Synchronizing\\_Red\\_Hat.Repositories](https://access.redhat.com/documentation/en-us/red_hat_satellite/6.6/html-single/content_management_guide/importing_red_hat_content#Importing_Red_Hat_Content-Synchronizing_Red_Hat.Repositories)

## ► Guided Exercise

# Synchronizing Red Hat Content

In this exercise, you will create a sync plan, enable software repositories, create products, and ensure that the products are current in the library.

## Outcomes

You should be able to:

- Enable and synchronize available repositories.
- Set the default download policy to **on\_demand**.
- Create a sync plan to provide control over the frequency of content synchronization.

## Before You Begin

Log in to **workstation** as **student** using **student** as the password.

Run the **lab lifecycles-sync start** command to prepare your system for the exercise. This command determines if the **satellite** host is reachable on the network and verifies that the required repositories are available.

```
[student@workstation ~]$ lab lifecycles-sync start
```

- ▶ 1. Log in to the Satellite Server web UI, <https://satellite.lab.example.com>, as **admin** using **redhat** as the password.
- ▶ 2. Choose the **Operations** organization and **Boston** location from the main menu.
- ▶ 3. Enable synchronization of the desired repository from the local CDN.
  - 3.1. Click **Content** → **Red Hat Repositories** to access the **Red Hat Repositories** page.
  - 3.2. Enter **RHEL 8 x86\_64** in the **Search** field, and then click **Search**. Toggle the **Recommended Repositories** button to **ON** to list only the recommended repositories.
  - 3.3. Expand the **Red Hat Enterprise Linux 8 for x86\_64 - AppStream (RPMs)**, **Red Hat Enterprise Linux 8 for x86\_64 - BaseOS (RPMs)**, and **Red Hat Satellite Tools 6.6 for RHEL 8 x86\_64 (RPMs)** repositories. Click the plus sign (+) next to **x86\_64 8.1** of each of the repositories to enable them.
- ▶ 4. Ensure that the default download policy for repositories is set to **on\_demand** so that Satellite Server downloads only the metadata during synchronization.
  - 4.1. Click **Administer** → **Settings** and then click the **Content** tab.
  - 4.2. In the **Name** column, locate the **Default Repository download policy** entry and ensure that **Default Repository download policy** is set to **on\_demand**.

- ▶ 5. Use the Satellite Server web UI to synchronize **Red Hat Enterprise Linux 8 for x86\_64 - AppStream RPMs x86\_64 8.1** and **Red Hat Enterprise Linux 8 for x86\_64 - BaseOS RPMs x86\_64 8.1** repositories.
- 5.1. Click **Content → Products** to open the **Products** page.
  - 5.2. Click **Red Hat Enterprise Linux for x86\_64**. The **Repositories** tab displays.
  - 5.3. Select the check boxes next to **Red Hat Enterprise Linux 8 for x86\_64 - AppStream (RPMs)** and **Red Hat Enterprise Linux 8 for x86\_64 - BaseOS (RPMs)** repositories that were enabled earlier. Click **Sync Now** to begin the repository synchronization.



**Note**

The synchronization task requires approximately 30 minutes to complete. You do not need to wait for the synchronization to complete before continuing.

- ▶ 6. Use the **hammer** command to synchronize the **Red Hat Satellite Tools 6.6 for RHEL 8 x86\_64 RPMs x86\_64** repository.
- 6.1. On **workstation**, use **ssh** to log in to **satellite** as **student**. Use **sudo -i** to switch to **root**.

```
[student@workstation ~]$ ssh student@satellite
[student@satellite ~]$ sudo -i
[sudo] password for student: student
[root@satellite ~]#
```

- 6.2. Use the **hammer repository list** command to list all the enabled repositories. Note the numeric identifier of the **Red Hat Satellite Tools 6.6 for RHEL 8 x86\_64 RPMs x86\_64** repository in the command output.

```
[root@satellite ~]# hammer --output json repository list
[
  ...
  ...
  {
    "Id": 2,
    "Name": "Red Hat Enterprise Linux 8 for x86_64 - BaseOS RPMs x86_64 8.1",
    "Product": "Red Hat Enterprise Linux for x86_64",
    "Content Type": "yum",
    "URL": "http://content.example.com/rhs6.6/...rhel8/8.1/x86_64/baseos/os"
  },
  {
    "Id": 3,
    "Name": "Red Hat Satellite Tools 6.6 for RHEL 8 x86_64 RPMs x86_64",
    "Product": "Red Hat Enterprise Linux for x86_64",
    "Content Type": "yum",
    "URL": "http://content.example.com/rhs6.6/...rhel8/x86_64/sat-tools/6.6/os"
  }
]
```

- 6.3. Use the `hammer repository synchronize` command to synchronize the **Red Hat Satellite Tools 6.6 for RHEL 8 x86\_64 RPMs x86\_64** repository.

```
[root@satellite ~]# hammer repository synchronize --id 3
[.....] [100%]
New packages: 16 (32.6 MB).
```

- 6.4. Log off from the **satellite** host.

```
[root@satellite ~]# exit
logout
[student@satellite ~]$ exit
logout
Connection to satellite closed.
```

- 7. Use the Satellite Server web UI to create a new sync plan that checks and updates the content every day at a scheduled time. Assign the sync plan to the **Red Hat Enterprise Linux for x86\_64** product.

- 7.1. Click **Content** → **Sync Plans**, and then click **Create Sync Plan**.
- 7.2. Enter **Red Hat Product Sync** in the **Name** field.
- 7.3. Select **custom cron** in the **Interval** field.
- 7.4. Enter the current time in the **Custom Cron** field to synchronize at that time every day. For example, enter **15 18 \* \* \*** in the **Custom Cron** field to synchronize at 18:15 every day. Select the current date for the **Start Date** field.
- 7.5. Click **Save** to create the sync plan. The **Details** tab displays.
- 7.6. Click the **Products** tab and then click the **Add** tab. Select the check box next to **Red Hat Enterprise Linux for x86\_64**, and then click **Add Selected**.

► 8. Click **Monitor** → **Recurring Logics** to verify the recurring task you created.

► 9. Click **Content** → **Products** and then click **Red Hat Enterprise Linux for x86\_64**.

► 10. Click the **Tasks** tab to verify that the synchronization task is started on the assigned product at the scheduled time.

## Finish

On the **workstation** machine, use the `lab lifecycles-sync finish` command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab lifecycles-sync finish
```

This concludes the guided exercise.

# Creating Software Life Cycles

---

## Objectives

After completing this section, you should be able to define a workflow for software promotion by creating life-cycle environments and organizing them into an environment path.

## Describing the Software Development Life Cycle

A *software development life cycle (SDLC)* is composed of a number of clearly defined and distinct work phases, which are used by engineers and developers to plan, design, build, test, and deliver software releases. These work phases are called *environments*. An SDLC aims to produce better results, staging software releases through each life-cycle environment so that unforeseen issues are resolved prior to the release of software to production systems.

Computer systems and operational environments have increased in complexity, often due to the interrelationships of software supplied by different vendors. To manage complexity, the SDLC model has been widely adopted for software deployment release management.

Host compute resource requirements depend on the activities within each life-cycle environment. To align the compute resources to environments, specify the life-cycle environment when registering each host to Satellite Server. Registering hosts is covered later in this course.

Associate a host with a particular life-cycle environment, ensuring that the host's compute resources match the requirements of the life-cycle environment. For example, register hosts used for writing code to the development environment. Register hosts used for testing code to the testing environment. When a host is associated with a life-cycle environment, it is uncommon for that host to change life-cycle environments unless the host is reinstalled for a different set of requirements. Managing Satellite Server life-cycle environments requires administrative privileges.

## Creating an Environment Path

With *life-cycle environments*, Red Hat Satellite Server 6 offers a way to release software packages and errata following the SDLC model. You can create life-cycle environments to match each stage of the SDLC. A sequence of life-cycle environments form an *environment path*. You can create multiple environment paths in Satellite Server.

Each environment path begins with the **Library** environment that syncs content from available sources. Avoid associating hosts directly with the **Library** environment, because the **Library** is constantly being synced to newer source content. Creating one or more life-cycle environments in a series extends the environment path to align to the workflow of your organization.

To create a life-cycle path within the current organization and location context:

- Navigate to **Content** → **Lifecycle Environments**.
- Click **Create Environment Path** to display the **New Environment** page.
- The **Name** acts as the environment identifier for users. The **Label** is autogenerated from the name, replacing white space with underscores. The **Description** could include the environment purpose or activities. Set a meaningful name and description for the environment, and click **Save** to create the environment.

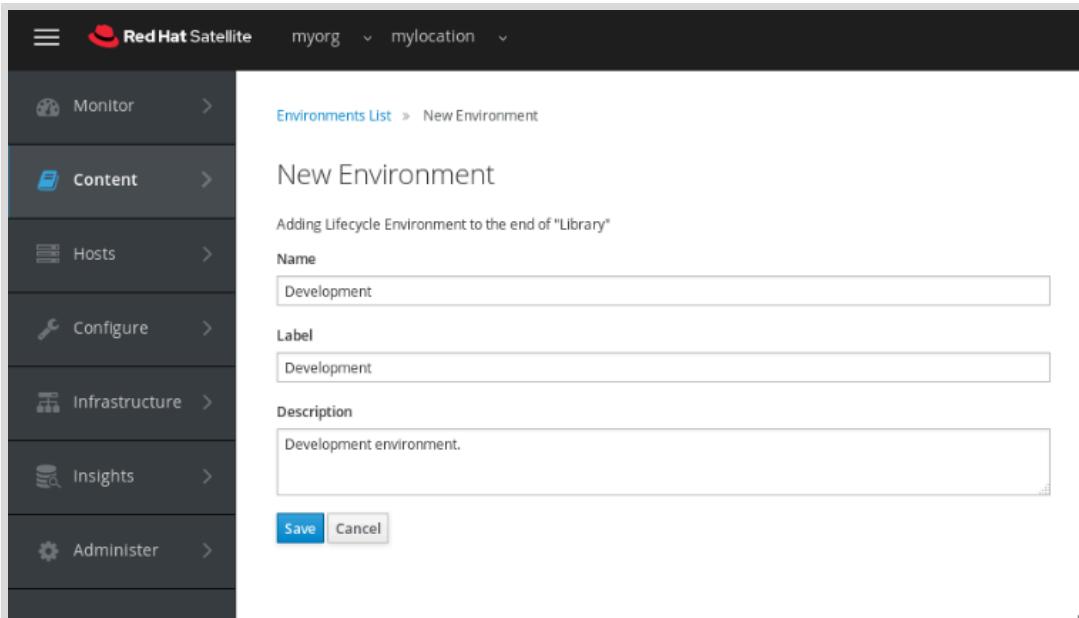


Figure 2.1: Creating an environment path

## Extending an Environment Path

After you have created an environment path, extend it by adding more life-cycle environments. Extending an environment path with environments implements the SDLC model according to your organization's policies. To extend an environment path with another environment:

- Navigate to **Lifecycle Environment Paths** and click **Add New Environment**.
- On the **New Environment** page, enter the **Name** and **Description**, then select an environment from **Prior Environment**.

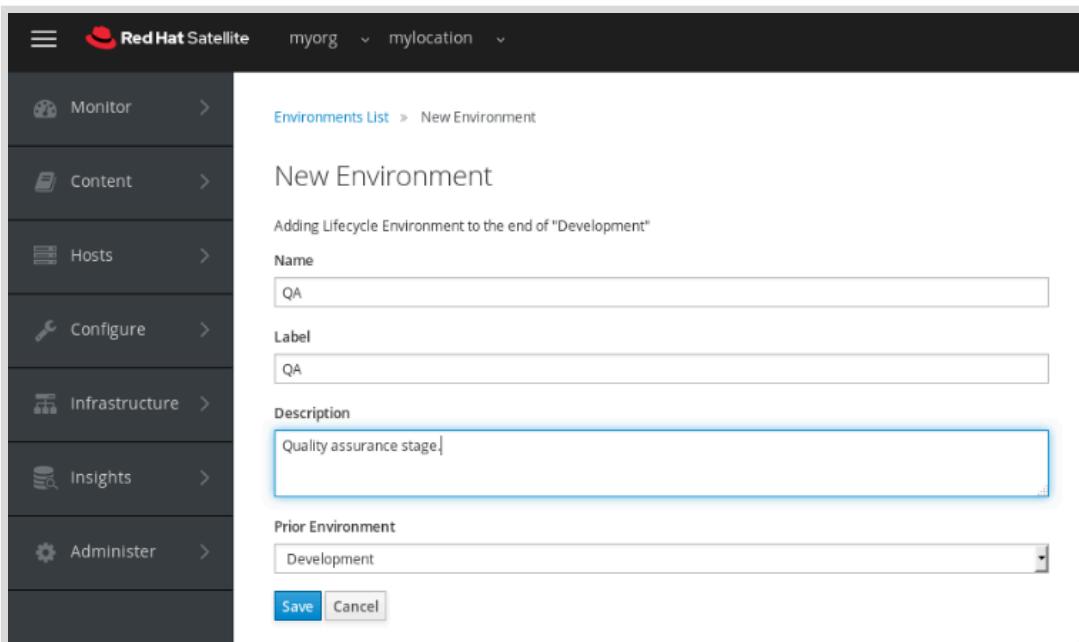


Figure 2.2: Extending an environment path

## Removing a Life-cycle Environment

You can remove a life-cycle environment that is no longer needed from an environment path. Click the name of the life-cycle environment in the **Lifecycle Environment Paths** page, and click **Remove Environment**.

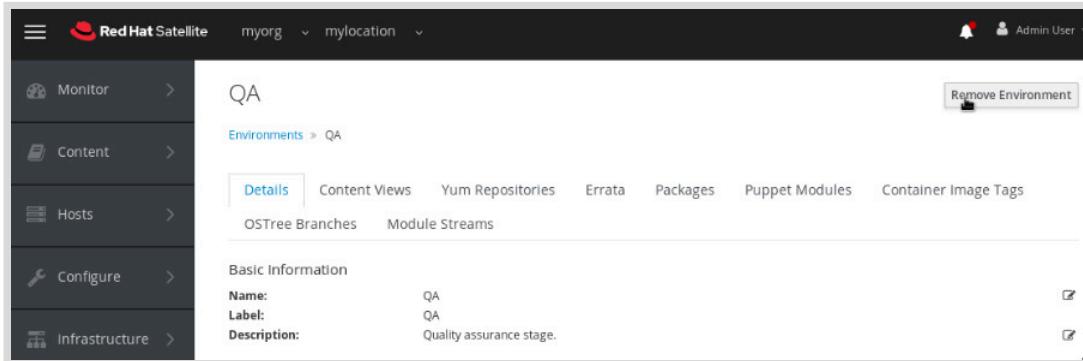


Figure 2.3: Removing a life-cycle environment

## Managing Life-cycle Environments from the CLI

This section describes the use of the **hammer** command to manage life-cycle environments.

During host registration, you specify to which organization the host belongs. Setting the organization for a life-cycle environment defines the scope of hosts that the environment can manage. Use the **--organization** option with **hammer** to specify the organization for the life-cycle environment. Use the **--name** and **--description** options to set the name and description of the life-cycle environment.

Every life-cycle environment follows either **Library** or another environment. A life-cycle environment following **Library** initiates a new life-cycle environment path. The life-cycle environment that follows an environment other than **Library** extends the existing life-cycle environment path. Use the **--prior** option with **hammer** to specify the preceding life-cycle environment that the new life-cycle environment follows.

```
[root@server]# hammer lifecycle-environment create \
--organization myorg --name Dev \
--description "Development Environment" \
--prior Library
Environment created.
```

The following **hammer** command lists the life-cycle environments in the **myorg** organization.

```
[root@server]# hammer lifecycle-environment list --organization myorg
---|-----|-----
ID	NAME	PRIOR
11 | Dev     | Library
10 | Library |
---|-----|-----
```

To produce output in JSON format, use the **hammer** command's **--output json** option.

The following **hammer** command lists the environment paths with every life-cycle environment for the **myorg** organization context.

```
[root@server]# hammer --output json lifecycle-environment paths \
--organization myorg
-----
LIFECYCLE PATH
-----
Library >> Build >> Deploy
Library >> Dev >> Test
-----
```

The following **hammer** command deletes the **Deploy** life-cycle environment from the **myorg** organization.

```
[root@server]# hammer lifecycle-environment delete \
--name Deploy --organization myorg
Environment deleted.
```

## Describing Content Life-cycle Scenarios

The following are among the common scenarios of content workflow.

The content is collected in the **Library** environment and then distributed directly to the production environment. This approach is simple and works well with the content that is ready for consumption, for example, the package updates from the Red Hat Content Delivery Network (CDN). Even with fewer environments, this content life-cycle design has the provision to test the content within the **Library** environment before making it available for the production environment.



Figure 2.4: A single life-cycle environment scenario

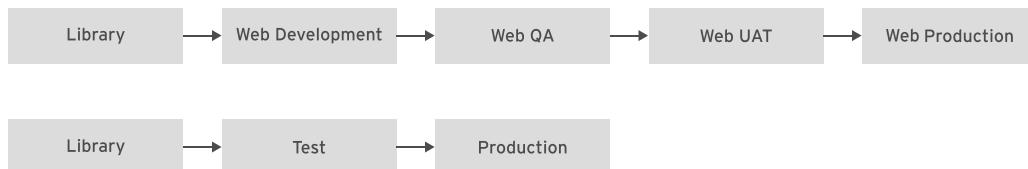
All content flows through a single life-cycle environment path, consisting of multiple life-cycle environments. This approach requires additional effort as compared to the single life-cycle environment approach. This content life-cycle design suits the organization that deals with a particular type of content and requires their content to pass through different stages for thorough testing prior to the production use. In this content life-cycle design, all content goes through the same life-cycle environment path, containing different life-cycle environments (for example, **Development**, **Testing**, **Production**).



Figure 2.5: A single life-cycle environment path containing multiple environments

Each specific type of content flows through a particular life-cycle environment path containing multiple life-cycle environments. This approach allows you to maintain separate release cycles for each type of content. In this content life-cycle design, you can associate specific compute resources with the life-cycle environments to facilitate testing. This approach increases the maintenance complexity but enables you to deal with multiple types of content, maintaining separate workflows for each content type. For example, you can create separate life-cycle

environment paths for the web development content and the base operating system packages content, segregating the workflow of both the content types.



**Figure 2.6: Content-specific life-cycle environment paths**

The previous figure shows the segregation between the web development content and the base operating system packages content using two separate life-cycle environment paths. With the first life-cycle environment path, the web development content undergoes programming changes in the **Web Development** phase and moves on to **Web QA**, where the quality of the content is verified. After passing the **Web QA** phase, the content is handed off to its intended users to test if the functional capabilities of the content meet the business requirements of the real world. This specific phase is called the *User Acceptance Testing (UAT)* phase. One strategy that the software vendors use to gain feedback from the UAT phase is to release the *beta* versions of the software before making the software generally available. In the previous figure, the UAT phase for the web content is represented as **Web UAT**. After successfully passing the UAT phase, the content is published to **Web Production** to be generally available for production use.

With the second life-cycle environment path in the previous figure, the base operating system packages content undergoes general testing in the **Test** phase to ensure that the content operates successfully with the intended environment. After successful testing of the content, it is published to the **Production** environment.



## References

For more information, refer to the *Creating an Application Life Cycle* chapter in the *Red Hat Satellite 6.6 Content Management Guide* at  
[https://access.redhat.com/documentation/en-us/red\\_hat\\_satellite/6.6/html-single/content\\_management\\_guide/creating\\_an\\_application\\_life\\_cycle#Creating\\_an\\_Application\\_Life\\_Cycle-Creating\\_a\\_New\\_Application\\_Life\\_Cycle](https://access.redhat.com/documentation/en-us/red_hat_satellite/6.6/html-single/content_management_guide/creating_an_application_life_cycle#Creating_an_Application_Life_Cycle-Creating_a_New_Application_Life_Cycle)

## ► Guided Exercise

# Creating Software Life Cycles

In this exercise, you will create an environment path made up of several life-cycle environments that represent different stages of a software development and deployment workflow.

## Outcomes

You should be able to create a new life-cycle environment path for the **Operations** organization that includes life-cycle environments named Development, QA, and Production.

## Before You Begin

Log in to **workstation** as **student** using **student** as the password.

Run the **lab lifecycles-define start** command to prepare your system for the exercise. This command determines if the **satellite** host is reachable on the network and verifies that the required resources exist.

```
[student@workstation ~]$ lab lifecycles-define start
```

- ▶ 1. Log in to the Satellite Server web UI, <https://satellite.lab.example.com>, as **admin** using **redhat** as the password.
- ▶ 2. Choose the **Operations** organization and **Boston** location from the main menu.
- ▶ 3. Click **Content → Lifecycle Environments** and then click **Create Environment Path**. The **New Environment** page displays.
- ▶ 4. Specify the appropriate details in the **New Environment** page to create the **Development** environment.
  - 4.1. Enter **Development** in the **Name** field. Notice that the **Label** field is automatically populated from the **Name** field.
  - 4.2. Enter **Development** in the **Description** field, and click **Save** to create the new environment.
- ▶ 5. Add the **QA** environment to the same life-cycle environment path as **Development** environment.
  - 5.1. Click **Add New Environment** to create the **QA** environment in the same environment path as **Development**.
  - 5.2. Enter **QA** in the **Name** field.
  - 5.3. Enter **Quality Assurance** in the **Description** field.
  - 5.4. Ensure that **Development** is selected for the **Prior Environment** field, and click **Save** to create the new environment.

- 6. Use the **hammer** command to add the **Production** environment to the same life-cycle environment path as **Development** and **QA**.

- 6.1. On **workstation**, use **ssh** to log in to **satellite** as **student**. Use **sudo -i** to switch to **root**.

```
[student@workstation ~]$ ssh student@satellite
[student@satellite ~]$ sudo -i
[sudo] password for student: student
[root@satellite ~]#
```

- 6.2. Use the **hammer lifecycle-environment list** command to list life-cycle environments in the **Operations** organization.

```
[root@satellite ~]# hammer lifecycle-environment list --organization Operations
---|-----|-----
ID	NAME	PRIOR
3  | Development | Library
2  | Library    |
4  | QA         | Development
---|-----|-----
```

- 6.3. Use the **hammer lifecycle-environment create** command to create the **Production** life-cycle environment in the **Operations** organization. Ensure that the **Production** environment follows the **QA** environment.

```
[root@satellite ~]# hammer lifecycle-environment create \
--organization Operations --name Production --label Production \
--description Production --prior QA
Environment created.
```

- 6.4. Use the **hammer lifecycle-environment paths** command to view the available life-cycle environment paths.

```
[root@satellite ~]# hammer lifecycle-environment paths --organization Operations
-----
LIFECYCLE PATH
-----
Library >> Development >> QA >> Production
-----
```

- 6.5. Log off from the **satellite** host.

```
[root@satellite ~]# exit
logout
[student@satellite ~]$ exit
logout
Connection to satellite closed.
```

## Finish

On the **workstation** machine, use the **lab lifecycles-define finish** command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab lifecycles-define finish
```

This concludes the guided exercise.

# Publishing and Promoting Content Views

## Objectives

After completing this section, you should be able to create and publish content views, and promote them to life-cycle environments on an environment path.

## Describing Content Views

A *content view* is a defined subset of content, to be made available to environments. You can create content view filters for curating content sets and then associate those sets with different environments. Deploying content from Red Hat Satellite Server begins with the publication of content into the library. Content views dictate what content is published into the repository, and control what is made available to environment paths and their life-cycle environments. Managing Satellite Server content views requires administrative privileges. To create a content view within the current organization and location context:

- Navigate to **Content** → **Content Views** and click **Create New View**.
- On the **Create Content View** page, enter a name and description, and then click **Save**. The label is autogenerated from the **Name**, replacing white space with underscores.

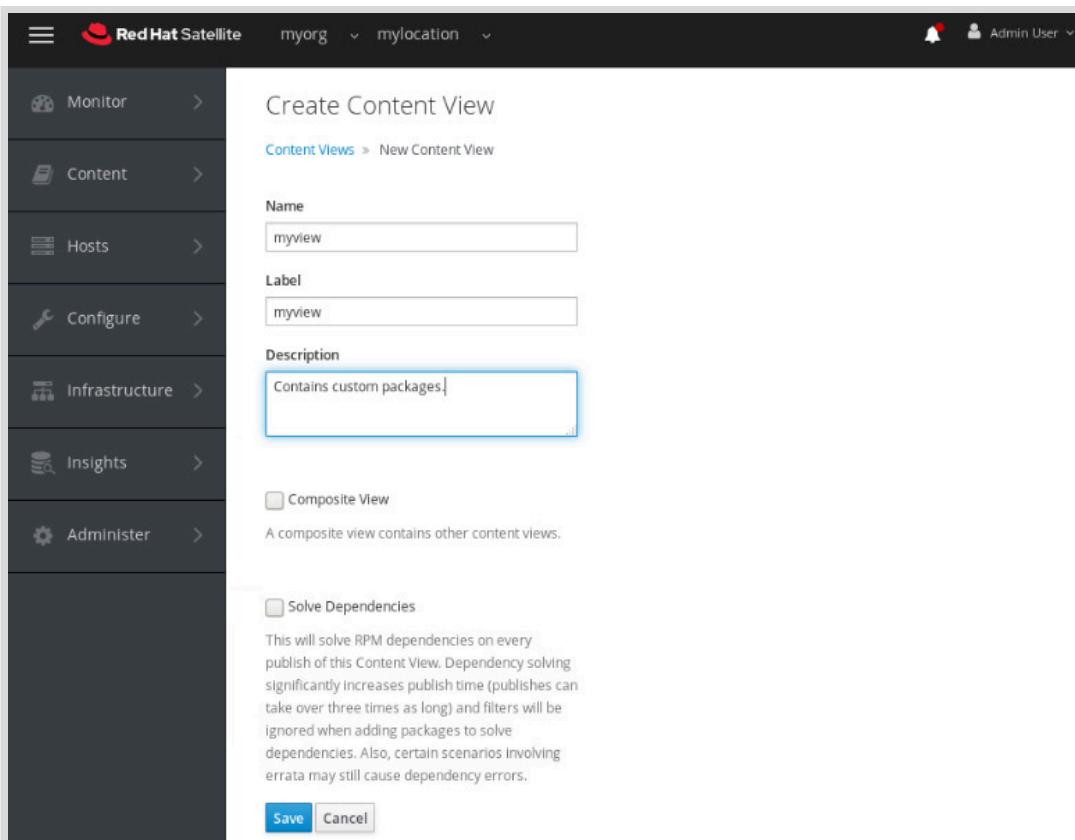


Figure 2.7: Creating content views

## Managing Repositories in Content Views

When you first create a content view it has no content associated with it. To populate the content view, associate it with the repositories that contain the required content. You can associate more than one repository with a content view.

### To Add Repositories to a Content View:

- For the required organization and location, click **Content** → **Content Views**, and then click the name of the content view.
- Click **Yum Content** → **Repositories** and then click the **Add** tab.
- Select the check box for the repositories that you want to add, and then click **Add Repositories**.

| <input type="checkbox"/>            | Name                                                            | Product                             | Last Sync        | Sync State | Content                                           |
|-------------------------------------|-----------------------------------------------------------------|-------------------------------------|------------------|------------|---------------------------------------------------|
| <input type="checkbox"/>            | Red Hat Enterprise Linux 8 for x86_64 - AppStream RPMs x86_64 8 | Red Hat Enterprise Linux for x86_64 | Nov 01, 11:07 AM | Success    | 8367 Packages<br>273 Errata<br>132 Module Streams |
| <input checked="" type="checkbox"/> | Red Hat Enterprise Linux 8 for x86_64 - BaseOS RPMs x86_64 8    | Red Hat Enterprise Linux for x86_64 | Nov 01, 10:50 AM | Success    | 2625 Packages<br>161 Errata<br>0 Module Streams   |
| <input type="checkbox"/>            | Red Hat Satellite Tools 6.6 for RHEL 8 x86_64 RPMs x86_64       | Red Hat Enterprise Linux for x86_64 | Nov 01, 10:42 AM | Success    | 16 Packages<br>0 Errata<br>0 Module Streams       |

Figure 2.8: Adding a repository to the content view

## Removing Repositories from a Content View

The **List/Remove** tab lists the repositories associated with the content view. Select the check box for the repository you want to remove, and click **Remove Repositories** to remove the repository from the content view.

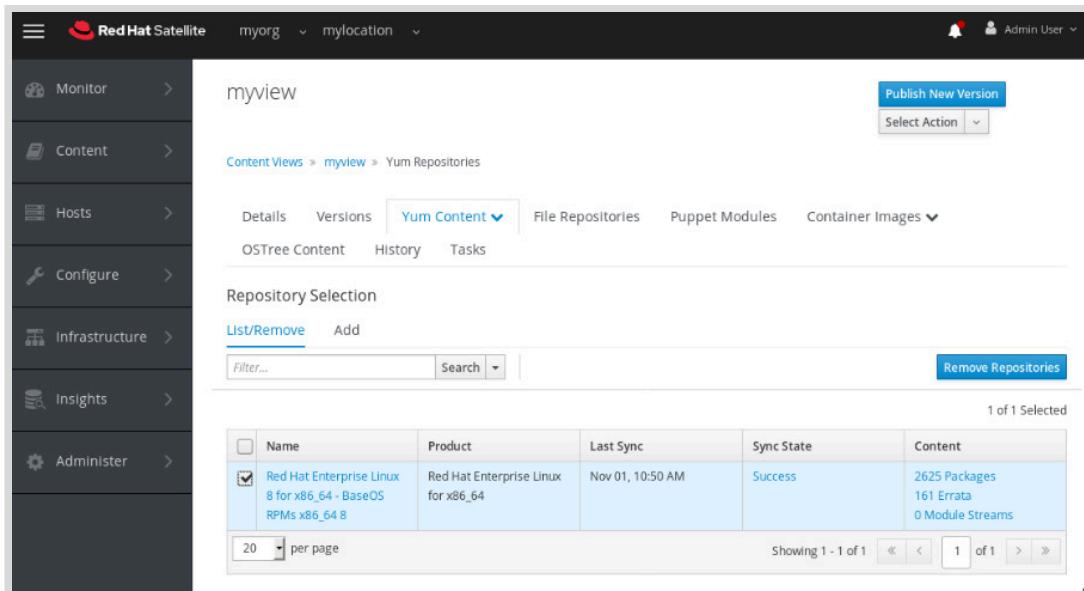


Figure 2.9: Removing a repository from a content view

## Using Filters

Content views include repositories and filters that define the content of the content view. Filter rules include target packages, package groups, and errata. These rules also control which package versions a particular version of content view distributes. Filtering content views is discussed later in this course.

Any customization to the content view repositories or filters requires the publication of a new version of the content view to make the change available. This new version of the content view is promoted to the intended life-cycle environment to distribute the changed content to the hosts of that life-cycle environment. Promoting a content view version to a life-cycle environment makes it available to the hosts in that environment. For example, the hosts in the Production environment might be using earlier package versions, but the hosts in the Development environment require later package versions, included in the latest version of the content view. The impact of publishing and promoting content views after you register the hosts is discussed later in the course. Publishing and promoting content views are discussed next in this section.

## Publishing Content Views

After you create a content view, you need to publish it to the library. Every time you publish a content view it creates a new version of that content view. Every version is numbered sequentially for identification, as well as for version control purposes.

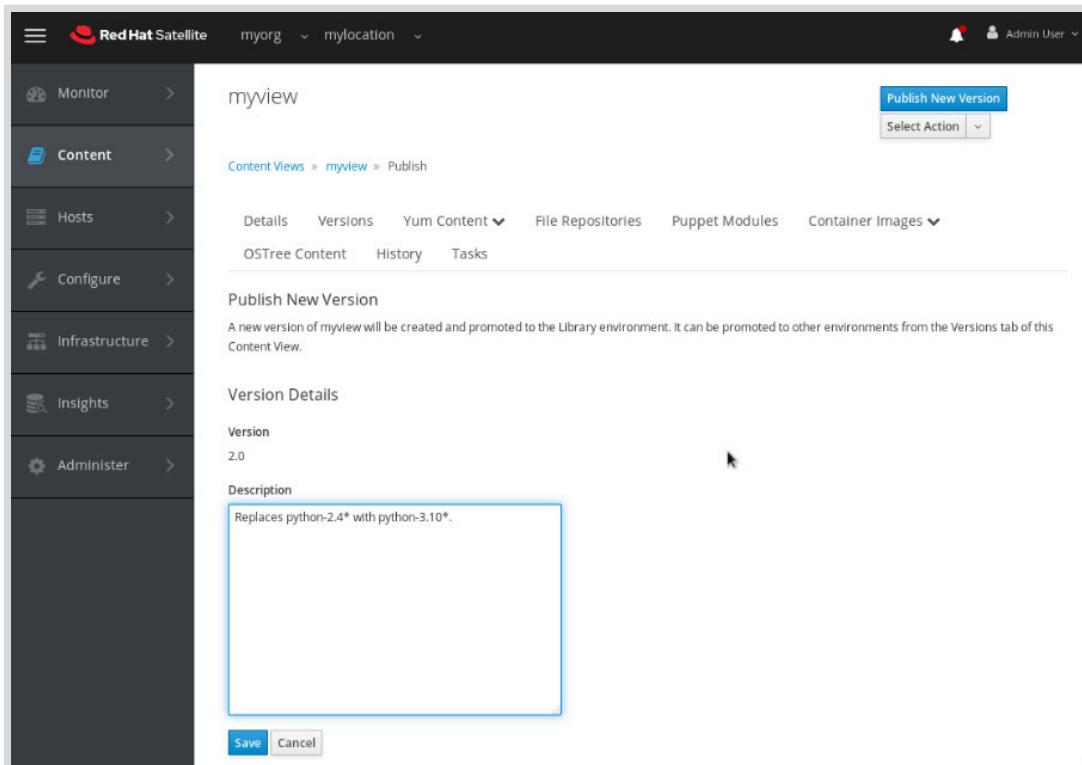
### To Publish a Content View:

- For the required organization and location, click **Content** → **Content Views**, and then click the name of the content view that you want to publish.
- After you have made changes such as updating packages in the content view, click **Publish New Version**.
- Note the version of the content view increases, add a meaningful description.

You may want to mention the difference between the new content view version and other versions as part of the description of the content view update. A meaningful description will

help you identify the purpose of that version at a later date. The version number itself does not provide any information about the content of the content view.

- After you have entered a suitable description, click **Save**.



**Figure 2.10: Publishing a content view**

## Promoting Content Views

After you have published a content view version to the library, you can deploy it to the first life-cycle environment in an environment path. This deployment of a content view is known as *promotion*.

After you have promoted a content view to an environment, you can release content (software packages and errata) to each life-cycle environment in an environment path. In accordance with the SDLC model, content views are promoted sequentially through an environment path. If you skip a particular life-cycle environment while promoting the content view, Satellite prompts you to confirm the forced promotion. Satellite Server maintains separate repositories across each life-cycle environment for each content view. When you promote a content view from one environment to the next in an environment path, the respective repository on Satellite Server updates and publishes the packages.

### To Promote a Content View:

- For the required organization and location, click **Content** → **Content Views**, and then click the name of the content view that you want to promote.
- The **Versions** tab lists the published versions of the content view. Each row in this tab represents a specific content view version.

In the **Actions** column, click **Promote** to display the **Promotion** page.

- Select the life-cycle environment that you want to promote to, and click **Promote Version**.

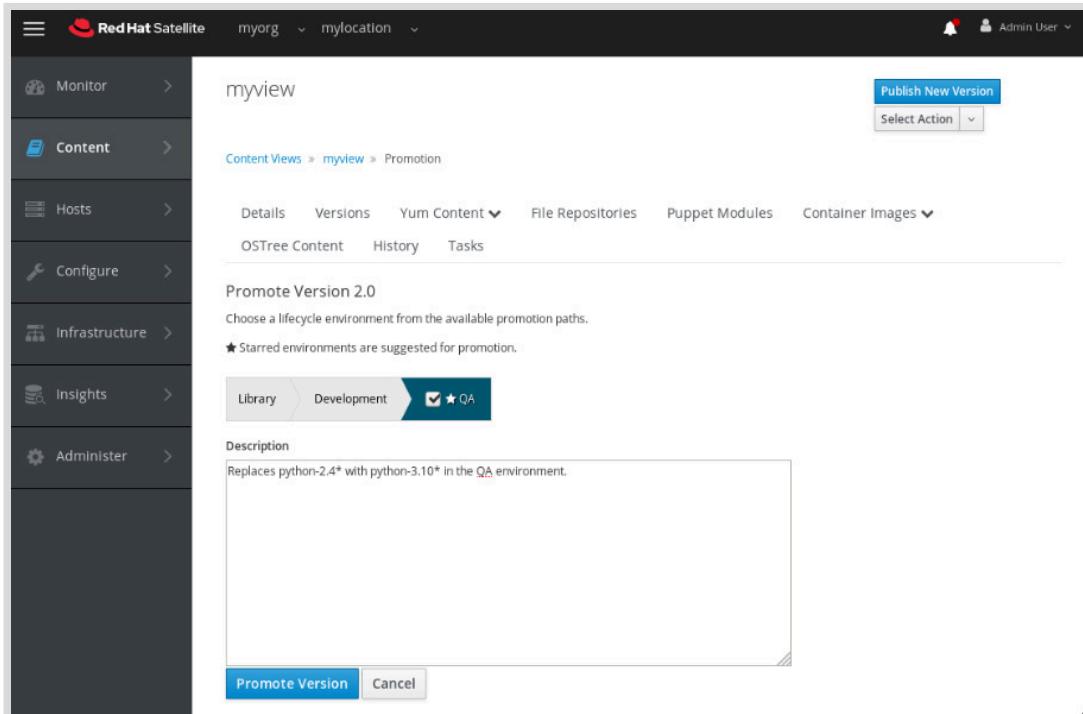


Figure 2.11: Promoting a content view

## Describing Content View Scenarios

In the *all-in-one content view* scenario, the content view contains all the desired content for all of your hosts. In this scenario, the reduced number of content views proves beneficial for an environment with uniform types of host or resource constraints, such as storage space. The trade-off with this scenario is the limitation on intelligent filtering that the content views are capable of.

In the *host-specific content view* scenario, dedicated content views exist for every type of host. This scenario is useful for an environment with a limited number of host types. In this scenario, the separation of hosts is strictly based on the host types. This scenario also prevents sharing content across host types.

Use a content view scenario based on the nature of your host environment. Avoid creating a large number of content views, but keep in mind that the size of a content view affects the speed of related operations, such as publishing and promoting. When creating a subset of packages for the content view, ensure that all dependencies are included. Note that kickstart repositories should not be added to content views, because they are only used for host provisioning.



### References

For more information, refer to the *Managing Content Views* chapter in the *Red Hat Satellite 6.6 Content Management Guide* at  
[https://access.redhat.com/documentation/en-us/red\\_hat\\_satellite/6.6/html-single/content\\_management\\_guide/managing\\_content\\_views#Managing\\_Content\\_VIEWS](https://access.redhat.com/documentation/en-us/red_hat_satellite/6.6/html-single/content_management_guide/managing_content_views#Managing_Content_VIEWS)

## ► Guided Exercise

# Publishing and Promoting Content Views

In this exercise, you will publish a new or modified content view and promote it to a specific life-cycle environment.

## Outcomes

You should be able to:

- Create and publish content views.
- Modify content views.
- Promote content views.

## Before You Begin

Log in to **workstation** as **student** using **student** as the password.

Run the **lab lifecycles-views start** command to prepare your system for the exercise. This command determines if the **satellite** host is reachable on the network and verifies that the required repositories are available.

```
[student@workstation ~]$ lab lifecycles-views start
```

- ▶ 1. Log in to the Satellite Server web UI, <https://satellite.lab.example.com>, as **admin** using **redhat** as the password.
- ▶ 2. Choose the **Operations** organization and **Boston** location from the main menu.
- ▶ 3. Create the **Base** content view.
  - 3.1. Click **Content** → **Content Views** and then click **Create New View** to display the **Create Content View** page.
  - 3.2. Enter **Base** in the **Name** field. Notice that the **Label** field is automatically populated from the **Name** field.
  - 3.3. Enter **Base Packages** in the **Description** field, and then click **Save** to create the new content view. The **Add Yum Repositories** page displays.
- ▶ 4. Select repositories and publish the **Base** content view to the **Development** life-cycle environment.
  - 4.1. Select the check box for the **Red Hat Enterprise Linux 8 for x86\_64 - BaseOS RPMs x86\_64 8.1** repository, and then click **Add Repositories** to add the selected repository to the content view.
  - 4.2. Click **Publish New Version** to display the **Publish** page.
  - 4.3. Enter **Base Repositories** in the **Description** field, and then click **Save** to publish the new content view version. Wait for the publishing process to complete.



### Note

This process may take up to 7 minutes to complete because of the repository size. You can monitor the publication progress in the **Status** column. At the end of this process, the **Base** content view is published to the **Library** environment.

- 4.4. In the **Actions** column, click **Promote** to display the **Promotion** page.
  - 4.5. Select the check box for **Development** life-cycle environment, and enter **Base Repositories** in the **Description** field.
  - 4.6. Click **Promote Version** to promote the new content view version. Wait for the promoting process to complete. At the end of this process the **Base** content view is published to the **Development** environment.
- 5. Modify the **Base** content view to include the following repositories. When done, promote the new version to the **Development** life-cycle.
- Red Hat Enterprise Linux 8 for x86\_64 - AppStream (RPMs)
  - Red Hat Satellite Tools 6.6 for RHEL 8 x86\_64 (RPMs)
- 5.1. Click **Yum Content** → **Repositories** to display the **Yum Repositories** page.  
In the **Repository Selection** section, click the **Add** tab to view the available repositories.
  - 5.2. Select the check boxes for the **Red Hat Enterprise Linux 8 for x86\_64**  
- **AppStream RPMs x86\_64 8.1** and **Red Hat Satellite Tools 6.6 for RHEL 8 x86\_64 RPMs x86\_64 8** repositories, and click **Add Repositories** to add the selected repositories to the content view.
  - 5.3. Click **Publish New Version** to display the **Publish** page. Notice that the **Version** is now 2.0.
  - 5.4. Enter **Base Repositories Version 2** in the **Description** field, and click **Save** to publish the new version. Wait for the publishing process to complete.



### Note

This process may take up to 10 minutes to complete because of the repository size. You can monitor the publication progress in the **Status** column. At the end of this process, the **Base** content view is published to the **Library** environment.

- 5.5. In the **Actions** column, click **Promote** for **Version 2.0** to display the **Promotion** page.
  - 5.6. Select the check box for **Development**, and enter **Base Repositories 2** in the **Description** field.
  - 5.7. Click **Promote Version** to promote the new content view version.  
The promotion process takes some time to complete. You can monitor the promotion progress in the **Status** column.
- 6. Promote Version 2 of the **Base** content view to the **QA** environment.

- 6.1. In the **Actions** column for **Version 2.0**, click **Promote** to display the **Promotion** page.
  - 6.2. Select the check box for **QA**, and enter **Base Repositories v2** in the **Description** field.
  - 6.3. Click **Promote Version** to promote the new content view version. You can monitor the promotion progress in the **Status** column.
- ▶ 7. Explore the Satellite Server web UI and inspect some of the differences between the two content view versions.
- 7.1. Click **Content** → **Content Views** to display the **Content Views** page. Inspect the **Environments** column of the **Base** content view to verify the life-cycle environments that are associated with the content view.
  - 7.2. Inspect the **Repositories** column of the **Base** content view to verify the number of repositories that are currently active in the latest version of the content view.
  - 7.3. In the **Name** column, click the **Base** content view to display the **Versions** tab.
  - 7.4. Notice the different content view versions, and the difference in number of packages between them. The number of packages in the **Version 2.0** content view version should be more, compared to **Version 1.0** because you added two additional repositories to the former.
  - 7.5. Click the **Version 1.0** content view version to display the **Versions** tab. Click **Yum Repositories** to verify the repositories of the content view version, and confirm that only the **Red Hat Enterprise Linux 8 for x86\_64 - BaseOS RPMs x86\_64 8** repository displays.
  - 7.6. Repeat the previous step for the **Version 2.0** content view version, and confirm that all the **Red Hat Enterprise Linux 8 for x86\_64 - BaseOS RPMs x86\_64 8**, **Red Hat Enterprise Linux 8 for x86\_64 - AppStream (RPMs)**, and **Red Hat Satellite Tools 6.6 for RHEL 8 x86\_64 (RPMs)** repositories are associated with that content view version.

## Finish

On the **workstation** machine, use the **lab lifecycles-views finish** command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab lifecycles-views finish
```

This concludes the guided exercise.

## ► Lab

# Managing Software Life Cycles

### Performance Checklist

In this lab, you will enable software repositories, create a sync plan, create life-cycle environments, and associate a new content view with a life-cycle environment.

### Outcomes

You should be able to:

- Enable software repositories.
- Create a sync plan.
- Create life-cycle environments.
- Create content views.
- Associate a content view with a life-cycle environment.

### Before You Begin

Log in to **workstation** as **student** using **student** as the password.

Run the **lab lifecycles-review start** command. This command verifies that the Red Hat Satellite Server is available. This command also ensures that the **Finance** organization, and the **San Francisco** and **Tokyo** locations exist. It also imports a subscription manifest for the **Finance** organization.

```
[student@workstation ~]$ lab lifecycles-review start
```

1. Enable the following software repositories for any location in the **Finance** organization.
  - Red Hat Enterprise Linux 8.1 for x86\_64 - AppStream (RPMs)
  - Red Hat Enterprise Linux 8.1 for x86\_64 - BaseOS (RPMs)
  - Red Hat Satellite Tools 6.6 for RHEL 8 x86\_64 (RPMs)
2. Use **hammer** to manually synchronize the enabled repositories with the local Content Delivery Network (CDN). Use the **Finance** organization and **Any Location** while synchronizing the repositories.
3. Create a new sync plan named **Sync Plan for Red Hat Products** in the **Finance** organization. This sync plan should update the content every day from the current day onward at 6 p.m. Use the **Finance** organization and **Any Location** location while creating the sync plan.
4. Assign **Sync Plan for Red Hat Products** to the following repositories you enabled within the **Finance** organization in the preceding steps.
  - Red Hat Enterprise Linux 8.1 for x86\_64 - AppStream RPMs x86\_64 8.1
  - Red Hat Enterprise Linux 8.1 for x86\_64 - BaseOS RPMs x86\_64 8.1
  - Red Hat Satellite Tools 6.6 for RHEL 8 x86\_64 RPMs x86\_64
5. Run the sync plan manually to implement immediate content synchronization.

6. Create the **Build**, **Test**, and **Deploy** life-cycle environments.
7. Create a new content view called **Base** and associate it with the **Red Hat Enterprise Linux 8 for x86\_64 - BaseOS (RPMs)** repository.
8. Publish the **Base** content view.
9. Promote the content view to the **Build** life-cycle environment.

## Evaluation

As the **student** user on the **workstation** machine, use the **lab lifecycles-review grade** command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab lifecycles-review grade
```

## Finish

As the **student** user on the **workstation** machine, use the **lab lifecycles-review finish** command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab lifecycles-review finish
```

This concludes the lab.

## ► Solution

# Managing Software Life Cycles

### Performance Checklist

In this lab, you will enable software repositories, create a sync plan, create life-cycle environments, and associate a new content view with a life-cycle environment.

### Outcomes

You should be able to:

- Enable software repositories.
- Create a sync plan.
- Create life-cycle environments.
- Create content views.
- Associate a content view with a life-cycle environment.

### Before You Begin

Log in to **workstation** as **student** using **student** as the password.

Run the **lab lifecycles-review start** command. This command verifies that the Red Hat Satellite Server is available. This command also ensures that the **Finance** organization, and the **San Francisco** and **Tokyo** locations exist. It also imports a subscription manifest for the **Finance** organization.

```
[student@workstation ~]$ lab lifecycles-review start
```

1. Enable the following software repositories for any location in the **Finance** organization.
  - Red Hat Enterprise Linux 8.1 for x86\_64 - AppStream (RPMs)
  - Red Hat Enterprise Linux 8.1 for x86\_64 - BaseOS (RPMs)
  - Red Hat Satellite Tools 6.6 for RHEL 8 x86\_64 (RPMs)
  - 1.1. On **workstation**, launch Firefox and navigate to `https://satellite.lab.example.com` to access the Red Hat Satellite Server web UI.
  - 1.2. Log in to the web UI as **admin** using **redhat** as the password.
  - 1.3. Choose the **Finance** organization and **Any Location** location, and then click **Content → Red Hat Repositories** to display the list of available repositories.
  - 1.4. Locate and expand the **Red Hat Enterprise Linux 8 for x86\_64 - AppStream (RPMs)**. Click the plus sign (+) next to **x86\_64 8.1** to enable this repository. Similarly, enable the **Red Hat Enterprise Linux 8 for x86\_64 - BaseOS (RPMs)** repository.
  - 1.5. Locate and expand the **Red Hat Satellite Tools 6.6 for RHEL 8 x86\_64 (RPMs)**. Click the plus sign (+) next to **x86\_64** to enable this repository.
  - 1.6. Confirm that the enabled repositories display on the right side of the page.

2. Use **hammer** to manually synchronize the enabled repositories with the local Content Delivery Network (CDN). Use the **Finance** organization and **Any Location** while synchronizing the repositories.

- 2.1. On **workstation**, use **ssh** to log in to **satellite** as **student**. Use **sudo -i** to switch to **root**. If prompted, enter **student** as the password.

```
[student@workstation ~]$ ssh student@satellite
[student@satellite ~]$ sudo -i
[sudo] password for student: student
[root@satellite ~]#
```

- 2.2. Use the **hammer** command to list the enabled repositories within the **Finance** organization. Use the **json** output format with the **hammer** command and note the numeric identifier of each of the repositories.

```
[root@satellite ~]# hammer --output json repository list --organization Finance
[
  {
    "Id": 7,
    "Name": "Red Hat Enterprise Linux 8 for x86_64 - AppStream RPMs x86_64 8.1",
    "Product": "Red Hat Enterprise Linux for x86_64",
    "Content Type": "yum",
    "URL": "http://content.example.com/rhs6.6/x86_64/cdn/content/dist/rhel8/8.1/x86_64/appstream/os"
  },
  {
    "Id": 8,
    "Name": "Red Hat Enterprise Linux 8 for x86_64 - BaseOS RPMs x86_64 8.1",
    "Product": "Red Hat Enterprise Linux for x86_64",
    "Content Type": "yum",
    "URL": "http://content.example.com/rhs6.6/x86_64/cdn/content/dist/rhel8/8.1/x86_64/baseos/os"
  },
  {
    "Id": 9,
    "Name": "Red Hat Satellite Tools 6.6 for RHEL 8 x86_64 RPMs x86_64",
    "Product": "Red Hat Enterprise Linux for x86_64",
    "Content Type": "yum",
    "URL": "http://content.example.com/rhs6.6/x86_64/cdn/content/dist/layered/rhel8/x86_64/sat-tools/6.6/os"
  }
]
```

- 2.3. Use the **hammer** command to synchronize each of the enabled repositories in the **Finance** organization with the Content Delivery Network (CDN). Use the **--id** option to specify the numeric identifier of each repository noted in the preceding step.

```
[root@satellite ~]# hammer repository synchronize --id 7 --organization Finance
[.....] [100%]
No new packages.
[root@satellite ~]# hammer repository synchronize --id 8 --organization Finance
[.....] [100%]
No new packages.
```

```
[root@satellite ~]# hammer repository synchronize --id 9 --organization Finance  
[.....] [100%]  
No new packages.
```

It takes about 14 minutes for the AppStream repository to synchronize.

It takes about 4 minutes for the BaseOS repository to synchronize.

It takes a few seconds for the Tools repository to synchronize.

- 2.4. In the web UI, click **Monitor** → **Tasks**, and search for the tasks named **Synchronize repository repo\_name**. Confirm that the **Result** column displays **success** for these actions.
- 2.5. In the terminal, log off from the **satellite** system and return to **workstation**.
3. Create a new sync plan named **Sync Plan for Red Hat Products** in the **Finance** organization. This sync plan should update the content every day from the current day onward at 6 p.m. Use the **Finance** organization and **Any Location** location while creating the sync plan.
  - 3.1. In the web UI, choose the **Finance** organization and **Any Location** location from the main menu, and click **Content** → **Sync Plans** to display the **Sync Plans** page.
  - 3.2. Click **Create Sync Plan** to display the **New Sync Plan** page.
  - 3.3. Enter **Sync Plan for Red Hat Products** in the **Name** field. In the **Interval** field, click **daily**. Specify the current date as the start date. Enter 18 and 00 in the **HH** and **MM** fields, respectively, and then click **Save**.
4. Assign **Sync Plan for Red Hat Products** to the following repositories you enabled within the **Finance** organization in the preceding steps.
  - Red Hat Enterprise Linux 8.1 for x86\_64 - AppStream RPMs x86\_64 8.1
  - Red Hat Enterprise Linux 8.1 for x86\_64 - BaseOS RPMs x86\_64 8.1
  - Red Hat Satellite Tools 6.6 for RHEL 8 x86\_64 RPMs x86\_64
  - 4.1. In the web UI, choose the **Finance** organization and **Any Location** location, and then click **Content** → **Products** to display the **Products** page.
  - 4.2. Click **Red Hat Enterprise Linux for x86\_64** to display the **Repositories** tab for this product.
  - 4.3. Confirm that the **Red Hat Enterprise Linux 8 for x86\_64 - AppStream RPMs x86\_64 8.1**, **Red Hat Enterprise Linux 8 for x86\_64 - BaseOS RPMs x86\_64 8.1**, and **Red Hat Satellite Tools 6.6 for RHEL 8 x86\_64 RPMs x86\_64** repositories belong to the **Red Hat Enterprise Linux for x86\_64** product.
  - 4.4. Click **Content** → **Sync Plans**.
  - 4.5. Click **Sync Plan for Red Hat Products** to display the **Details** tab.
  - 4.6. Navigate to the **Products** tab and click the **Add** tab.
  - 4.7. Select the check box next to **Red Hat Enterprise Linux for x86\_64**, and click **Add Selected**. This associates the product repositories with the **Sync Plan for Red Hat Products** sync plan.

- 4.8. On the **List/Remove** tab, confirm that the **Red Hat Enterprise Linux for x86\_64** product displays.
- 4.9. Click the **Details** tab and view the details of the sync plan.
5. Run the sync plan manually to implement immediate content synchronization.
  - 5.1. On the **Sync Plan for Red Hat Products** page, click **Select Action** and then click **Run Sync Plan**.
  - 5.2. Click **Monitor → Tasks** and search for the tasks named **Synchronize repository repo\_name**. Confirm that the **Result** column displays **success** for these actions.
6. Create the **Build**, **Test**, and **Deploy** life-cycle environments.
  - 6.1. Click **Content → Lifecycle Environments** and then click **Create Environment Path** to display the **New Environment** page.
  - 6.2. Enter **Build** in the **Name** field, and click **Save** to display the **Lifecycle Environment Paths** page.
  - 6.3. Click **Add New Environment** to display the **New Environment** page.
  - 6.4. Enter **Test** in the **Name** field, and ensure that the **Prior Environment** field is set to **Build**. Click **Save**.
  - 6.5. On the **Lifecycle Environment Paths** page, click **Add New Environment** to display the **New Environment** page.
  - 6.6. Enter **Deploy** in the **Name** field, and ensure that the **Prior Environment** field is set to **Test**. Click **Save**.
7. Create a new content view called **Base** and associate it with the **Red Hat Enterprise Linux 8 for x86\_64 - BaseOS (RPMs)** repository.
  - 7.1. Click **Content → Content Views** and then click **Create New View**.
  - 7.2. On the **New Content View** page, enter **Base** in the **Name** field, and click **Save**.
  - 7.3. On the **Yum Content** tab, click the **Add** tab and select the check box next to **Red Hat Enterprise Linux 8 for x86\_64 - BaseOS RPMs x86\_64 8.1**, and then click **Add Repositories**.
  - 7.4. On the **List/Remove** tab, confirm that **Red Hat Enterprise Linux 8 for x86\_64 - BaseOS RPMs x86\_64 8** displays as the added repository.
8. Publish the **Base** content view.
  - 8.1. Click **Content → Content Views** to display the **Content Views** page.
  - 8.2. Click the **Base** content view to display the **Base** page.
  - 8.3. Click **Publish New Version** and then click **Save**. On the **versions** tab, the first version of **Base** appears as **Version 1.0**
  - 8.4. Wait for the version of the content view to be published successfully. This takes a couple of minutes. You can monitor the progress in the **Status** column.

9. Promote the content view to the **Build** life-cycle environment.
  - 9.1. On the **Versions** tab, click **Promote** for the **Version 1.0** version of the **Base** content view.
  - 9.2. Select the check box for **Build** and click **Promote Version**.

This concludes the lab.

## Evaluation

As the **student** user on the **workstation** machine, use the **lab lifecycles-review grade** command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab lifecycles-review grade
```

## Finish

As the **student** user on the **workstation** machine, use the **lab lifecycles-review finish** command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab lifecycles-review finish
```

## Summary

---

In this chapter, you learned:

- Enabling a Red Hat repository in Satellite Server automatically creates the relevant product.
- Use sync plans to automate content synchronization in Satellite server.
- The software development life cycle consists of different stages called environments in an environment path.
- Content views in Satellite Server allow selective content deployment on different hosts.

## Chapter 3

# Registering Hosts

### Goal

Register and configure your Red Hat Enterprise Linux systems to use Red Hat Satellite, and organize those systems into groups for easier management.

### Objectives

- Register a system with Red Hat Satellite and configure it to use a particular organization, location, subscription, and life-cycle environment.
- Organize registered hosts into host collections to manage them concurrently.
- Create and use activation keys to automatically register and configure a system for management by Satellite Server.

### Sections

- Registering and Configuring Content Hosts (and Guided Exercise)
- Managing Hosts with Host Collections (and Guided Exercise)
- Automating Registration of Content Hosts (and Guided Exercise)

### Lab

Registering Hosts

# Registering and Configuring Content Hosts

---

## Objectives

After completing this section, you should be able to register a system with Red Hat Satellite and configure it to use a particular organization, location, subscription, and life-cycle environment.

## Registering Content Hosts

A major difference between the previous version of Satellite and Satellite 6 is the content host tooling. Satellite 5 used the various RHN tools, such as **rhn\_register** and **rhnreg\_ks**, whereas Satellite 6 uses **subscription-manager**.

- In Satellite 5, the tools are content driven. They provide access to content but attaching to a subscription happens asynchronously in the background.
- In Satellite 6, the tools are certificate-based and subscription driven. Without a proper subscription, the host has no access to content.

The **subscription-manager** tool is useful for manually registering a freshly installed host to Satellite, or for troubleshooting registration. Although suitable for basic content host registration, manual registration is not typically used in production because it does not scale well.

Use an *activation key* with **subscription-manager** to apply additional configuration settings during registration, or with a bootstrap script designed to standardize a host's software configuration after registration. Using activation keys to automate registration is covered in a later section of this chapter.

## Preparing Hosts for Satellite Registration

To ensure proper interaction between Satellite Server and the host, the date and time must be correctly set for both systems. Red Hat recommends using **chrony** for time synchronization. After an accurate time configuration has been put in place, perform the following steps as the **root** user on the host to prepare it for Satellite registration.

- Update the *subscription-manager* and *yum* packages to the latest versions.

```
[root@host ~]# yum update subscription-manager yum
```

- Download and install the consumer certificate RPM from Satellite Server. The RPM is generated as part of the Satellite Server installation and provides correctly signed certificates for secure communication between the content host and Satellite Server. The consumer certificate RPM updates the content source location of the host and allows the host to download content from the content source specified in Satellite.

```
[root@host ~]# yum localinstall \
http://satellite.example.com/pub/katello-ca-consumer-latest.noarch.rpm
```

## Registering a Host to Satellite Server

When preparation is complete, perform these steps as the **root** user on the host to complete its registration to Satellite Server.

- Clear old registration data from the host.

```
[root@host ~]# subscription-manager clean
```

- Register the host to a chosen organization on Satellite Server. You will be prompted for a username and password with sufficient privileges. For example, enter the authentication information for the **admin** user on Satellite Server.



### Important

The argument to the **--org** option is the organization's label, not the organization name.

If an organization has environments configured, you are prompted for an environment during host registration.

```
[root@host ~]# subscription-manager register --org organization_label
Registering to: satellite.example.com:443/rhsm
Username: admin
Password: password
The system has been registered with ID: f134704d-3b8d-4a03-b0c1-e2b8491c4bdc
The registered system name is: host.example.com
```

- After registering to Satellite Server, a host gains access to repository contents. To perform package and errata management, however, you must install the Katello Agent on the managed host.

```
[root@host ~]# yum install katello-agent
```

- To verify the successful registration of a host to Satellite Server, set the desired **Organization** from the context menu at the upper-left of the Satellite web UI. Then, navigate to **Hosts** → **Content Hosts** to verify that the host is listed in the **Content Hosts** table.

## Configuring Content Hosts

After a host has been registered to Satellite Server, you can use the Satellite web UI to modify various aspects of the host's profile, such as its properties, subscription, and access to product content.

## Configuring Content Host Properties

To modify a content host's properties, perform the following steps as the **admin** user.

- Use the context menu to select the registered host's organization.
- Click **Hosts** → **Content Hosts** and click the name of the host to modify.
- Click the **Details** tab, make the changes, and click **Save**.

## Configuring Content Host Subscriptions

Subscriptions originate from the subscription manifest uploaded to an organization. Hosts must be attached to subscriptions to gain access to product content.

To modify a content host's subscription, perform the following steps as the **admin** user.

- Use the context menu to select the registered host's organization.
- Click **Hosts → Content Hosts** and click the name of the host to modify.
- Click the **Subscriptions** tab and make the desired changes to the host's subscription.
  - To remove a subscription, select its check box on the **List/Remove** tab and click **Remove Selected**.
  - To add a subscription, select its check box on the **Add** tab and click **Add Selected**.

## Modifying Product Content Access

Subscriptions grant hosts access to Red Hat products and their repositories, but not all repositories are enabled by default. Select which repositories to enable for a host by modifying product content settings in the Satellite web UI.

To modify a content host's product content settings, perform the following steps as the **admin** user.

- Use the context menu to select the registered host's organization.
- Navigate to **Hosts → Content Hosts** and click the name of the host to modify.
- Click the **Repository Sets** tab.

The available product and repositories are displayed. Make the changes to the status entry.

- Select the check box next to the repository to modify.
- Select the desired setting from the **Select Action** list.



### Important

When you change a host's configuration using the Satellite web UI, such as to subscription and product content, the changes may take time to be reflected on the managed host. You can use the **subscription-manager** command on the host to force a refresh to see the changes:

```
[root@host ~]# subscription-manager refresh  
All local data refreshed
```



### References

For more information, refer to the *Registering Hosts* chapter in the *Red Hat Satellite 6.6 Managing Hosts Guide* at  
[https://access.redhat.com/documentation/en-us/red\\_hat\\_satellite/6.6/html-single/managing\\_hosts/index#Registering\\_Hosts](https://access.redhat.com/documentation/en-us/red_hat_satellite/6.6/html-single/managing_hosts/index#Registering_Hosts)

## ► Guided Exercise

# Registering and Configuring Content Hosts

In this exercise, you will register a system with Red Hat Satellite Server and manually configure it for ongoing management by Satellite Server.

## Outcomes

You should be able to:

- Register a system with Red Hat Satellite Server.
- Attach the registered system to a subscription.

## Before You Begin

Log in to **workstation** as **student** using **student** as the password.

Run the **lab clients-register start** command. This command determines if the **satellite.lab.example.com** host is reachable on the network and verifies that the required resources exist. In the Satellite UI, ensure that the **Base** content view belongs to the **Operations** organization and is promoted to the **Development** life-cycle environment.

```
[student@workstation ~]$ lab clients-register start
```

- 1. On **workstation**, use **ssh** to log in to **servera** as **student**. Use **sudo -i** to switch to **root**.

```
[student@workstation ~]$ ssh student@servera
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

- 2. Use **subscription-manager clean** to clear any old registration data from **servera**. Ignore warnings related to the activation of the **yum** plug-ins.

```
[root@servera ~]# subscription-manager clean
All local data removed
```

### WARNING

The yum/dnf plugins: /etc/dnf/plugins/subscription-manager.conf, /etc/dnf/plugins/product-id.conf were automatically enabled for the benefit of Red Hat Subscription Management. If not desired, use "subscription-manager config --rhsm.auto\_enable\_yum\_plugins=0" to block this behavior.

- ▶ 3. Install the *katello-ca-consumer-latest* package from Red Hat Satellite Server (**satellite.lab.example.com**).

```
[root@servera ~]# yum localinstall \
http://satellite.lab.example.com/pub/katello-ca-consumer-latest.noarch.rpm
...output omitted...
Is this ok [y/N]: y
...output omitted...
Complete!
```

- ▶ 4. Use **subscription-manager register** to register **servera** to Satellite Server. Specify **Operations** for the organization and **Development/Base** as the environment. Register using the Satellite **admin** user with **redhat** as the password.

```
[root@servera ~]# subscription-manager register \
--org Operations --environment Development/Base
Registering to: satellite.lab.example.com:443/rhsm
Username: admin
Password: redhat
The system has been registered with ID: f134704d-3b8d-4a03-b0c1-e2b8491c4bdc
The registered system name is: servera.lab.example.com
```

In the preceding command, **Base** refers to the content view that was promoted to the **Development** life-cycle environment. The registration process takes a few seconds to complete.

- ▶ 5. Verify that the **servera** system was successfully registered to Satellite Server, and set its release version to **8.1**.

- 5.1. On **workstation**, launch Firefox and navigate to <https://satellite.lab.example.com> to access the Satellite Server web UI.
- 5.2. Log in to the web UI as **admin** using **redhat** as the password.
- 5.3. Change to the **Operations** organization and the **Any Location** context, and navigate to **Hosts** → **Content Hosts**.
- 5.4. Confirm that the **servera** system is listed as the registered host in the **Content Hosts** page.
- 5.5. On the **Content Hosts** page, click **servera.lab.example.com**, and select **8.1** on the **Release Version** drop-down menu. When done, click **Save**.
- 5.6. Confirm the host details, such as the **Development** environment and the **Base** content view.

- ▶ 6. Add the available subscriptions to **servera**.

- 6.1. On the **servera.lab.example.com** page, click **Subscriptions** → **Subscriptions**.
- 6.2. Click the **Add** tab below **Subscription Details**.
- 6.3. Select the check boxes for **Red Hat Enterprise Linux Server, Standard (Physical or Virtual Nodes)**, and **Red Hat Satellite Infrastructure Subscription**, and then click **Add Selected**.

- 6.4. Click the **List/Remove** tab below **Subscription Details** and confirm that the added subscriptions display.

## Finish

On the **workstation** machine, use the **lab clients-register finish** command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab clients-register finish
```

This concludes the guided exercise.

# Managing Hosts with Host Collections

## Objectives

After completing this section, you should be able to organize registered hosts into host collections to manage them concurrently.

## Configuring Host Collections

In Red Hat Satellite 6, you can create a logical group of hosts called *host collections* for multiple content hosts with similar characteristics. Host collections simplify various aspects of host administration. Rather than performing actions separately on individual hosts, you can instead perform them on a host collection, which applies those actions to all hosts in that host collection. Actions performed on host collections include the installation and update of packages, package groups, and errata.

You can create any number of host collections to organize hosts for repetitive actions that apply to various sets of hosts. Content hosts can simultaneously belong to multiple host collections.

## Creating a Host Collection

Host collections are defined at the organization level; you need to select an organization before you create a host collection.

### To Create a Host Collection:

- Navigate to **Hosts** → **Host Collections**.
- Click **Create Host Collection** and enter a name for your host collection. You can optionally add a brief description of the characteristics shared by the members of the host collection. The default number of hosts in a host collection is unlimited, however, you can clear the **Unlimited Hosts** check box and set a maximum number of hosts allowed in the host collection.



### Note

Using an activation key that assigns hosts to a host collection will fail to assign the host if the host collection has already reached its maximum limit. Activation keys are covered later in this chapter.

## Managing Host Collection Membership

A host must be registered to Red Hat Satellite in order to add it to a host collection.

- Navigate to **Hosts** → **Host Collections**.
- Click the name of the required host collection to display the details of that collection. The **Details** tab displays information about the host collection, including the number of content hosts, and actions that can be performed on content hosts. The **Hosts** tab lists the hosts that are members of the host collection and allows you to remove or add hosts to the host collection.

## Cloning a Host Collection

Cloning is an easy way to create a new host collection that is similar to an existing collection.

After cloning, the new host collection has the same hosts as the source host collection. Because they are now separate collections, either can be modified to add or remove hosts without affecting the other host collection.

### To Clone a Host Collection:

- Navigate to **Hosts → Host Collections** and click the name of the host collection you want to clone.
- In the **Select Action** list, select **Copy Host Collection**.
- Enter a name for the new host collection and click **Create**.

## Removing a Host Collection

Removing a host collection does not modify any content or registration configuration for the host members. The only change is that the hosts will no longer belong to that removed collection.

### To Remove a Host Collection:

- Navigate to **Hosts → Host Collections** and click the name of the host collection you want to remove.
- In the **Select Action** list, select **Remove**.
- Click **Remove** in the confirmation dialog box.



### References

For more information, refer to the *Configuring Host Collections* chapter in the *Red Hat Satellite 6.6 Managing Hosts Guide* at

[https://access.redhat.com/documentation/en-us/red\\_hat\\_satellite/6.6/html-single/managing\\_hosts/index#chap-Red\\_Hat\\_Satellite-Managing\\_Hosts-Configuring\\_Host\\_Collections](https://access.redhat.com/documentation/en-us/red_hat_satellite/6.6/html-single/managing_hosts/index#chap-Red_Hat_Satellite-Managing_Hosts-Configuring_Host_Collections)

## ► Guided Exercise

# Managing Hosts with Host Collections

In this exercise, you will create a host collection and assign registered hosts to that host collection.

### Outcomes

You should be able to:

- Create a host collection.
- Add a host to a host collection.

### Before You Begin

Log in as the **student** user on the **workstation** VM, using **student** as the password.

Run the **lab clients-collections start** command. This command determines if the **satellite** system is reachable on the network and prepares it for this exercise.

```
[student@workstation ~]$ lab clients-collections start
```

- ▶ 1. Log in to the Satellite Server web UI, <https://satellite.lab.example.com>, as **admin** using **redhat** as the password.
- ▶ 2. Use the organization and location context menu to select the **Operations** organization and **Any Location** location context.
- ▶ 3. Create the **OpsServers** host collection.
  - 3.1. Click **Hosts** → **Host Collections** and then click **Create Host Collection**.
  - 3.2. Enter **OpsServers** in the **Name** field.
  - 3.3. Enter **Operations server systems** in the **Description** field.
  - 3.4. Ensure the **Unlimited Hosts** check box is selected, and then click **Save**.
- ▶ 4. Add a registered host to the **OpsServers** host collection.
  - 4.1. Click the **Hosts** tab and then click the **Add** tab.
  - 4.2. Select the **servera.lab.example.com** check box, and then click **Add Selected**.
  - 4.3. Click the **List/Remove** tab to verify that **servera.lab.example.com** is listed as a member of the **OpsServers** host collection.

### Finish

On the **workstation** machine, use the **lab clients-collections finish** command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab clients-collections finish
```

This concludes the guided exercise.

# Automating Registration of Content Hosts

---

## Objectives

After completing this section, you should be able to create and use activation keys to automatically register and configure a system for management by Satellite Server.

## Automating Content Host Registration

Activation keys provide a method for automating system registration and attaching subscriptions. You can create multiple keys and associate them with different environments and Content Views. For example, you might create a basic activation key with a subscription for Red Hat Enterprise Linux workstations and associate it with Content Views from a particular environment.

Activation keys can define the following properties for content hosts:

- Associated subscriptions and subscription attachment behavior
- Available products and repositories
- A life-cycle environment and a Content View
- Host collection membership



### Note

Activation keys are used only when hosts are registered. If changes are made to an activation key, they apply only to hosts that are registered with the amended activation key in the future. The changes are not made to existing hosts.

## Creating Activation Keys

Activation keys are defined within an organizational context. You need to select an organization to access the **Activation Keys** page in the Satellite web UI.

### To Create an Activation Key:

- Log in as the **admin** user and navigate to **Content** → **Activation Keys**.
- If prompted, select the appropriate organization and then click **Select**.
- Click **Create Activation Key** and enter a name for your activation key. Optionally add a brief description for the activation key. The default number of systems that you can register with the activation key is unlimited, but you can clear the **Unlimited Hosts** check box and set a maximum number of systems that you can register using this activation key.
- In the **Environment** section, select the environment you want to use with this activation key. After you select an environment, a list displays beneath the **Content View** section. Select a content view to use and then click **Save**.

Alternatively, use the **hammer** command to create an activation key:

```
[root@host ~]# hammer activation-key create \
--name "Activation_Key_Name" \
--unlimited-hosts \
--description "Example description" \
--lifecycle-environment "Example environment" \
--content-view "Example content view" \
--organization "Organization_Name"
```

## Adding Subscriptions to Activation Keys

You can associate subscriptions with activation keys so that hosts that use the activation keys are automatically attached to the associated subscriptions when they register with Satellite Server.

Navigate to **Content** → **Activation Keys**. Click the name of an activation key to configure. Click the **Subscriptions** tab to list current subscriptions associated with the activation key.

The **Activation Key Type** is set to **Auto-Attach** by default. 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, such as, 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 Red Hat Enterprise Linux subscription.

To add a subscription, click the **Add** tab, select the checkbox for the subscription to add, then click **Add Selected**. To remove a subscription, click the **List/Remove** tab, select the checkbox for the subscription to remove, then click **Remove Selected**.

Alternatively, use the **hammer** command to add a subscription to an activation key:

- Obtain a list of an organization's subscription IDs:

```
[root@host ~]# hammer --output json subscription list \
--organization "Organization Name"
[
  {
    "ID": 7,
    "UUID": "2c997a8f6da8fa20016dad2d79d90063",
    "Name": "Red Hat Enterprise Linux Server, Standard (Physical or Virtual Nodes)",
    ...output omitted...
    "Quantity": 5,
    "Consumed": 1
  }
]
```

- Add the Red Hat Enterprise Linux subscription UUID to the activation key:

```
[root@host ~]# hammer activation-key add-subscription \
--name "Activation_Key_Name" \
--subscription-id 2c997a8f6da8fa20016dad2d79d90063 \
--organization "Organization Name"
```

## Adding Host Collections to Activation Keys

You can associate host collections with activation keys so that hosts that use those activation keys are automatically added to the associated host collections when they register with Satellite Server.

In the Satellite web UI, navigate to **Content** → **Activation Keys**. Click the name of an activation key that you want to configure. Click the **Host Collections** tab to list current host collections associated with the activation key.

To add a host collection, click the **Add** tab, select the checkbox for the host collection to add, then click **Add Selected**. To remove a host collection, click the **List/Remove** tab, select the checkbox for the host collection to remove, then click **Remove Selected**.

## Using Multiple Activation Keys with a Content Host

You can associate a content host with multiple activation keys, which are combined to define the host settings. In case of conflicting settings, the last specified activation key takes precedence. You can specify the order of precedence by setting a host group parameter as follows:

```
[root@host ~]# hammer hostgroup set-parameter \
--name kt_activation_keys \
--value name_of_first_key, name_of_second_key, ... \
--hostgroup hostgroup_name
```

## Registering Hosts with an Activation Key

After you have created an activation key, you can use it as an argument to the **subscription-manager** command when you register a host to Satellite Server.

## Preparing a Host for Registration to Satellite Server

Before you register a host to Satellite Server, verify that the *subscription-manager* and *yum* packages are installed on the host system.

```
[root@host ~]# rpm -q subscription-manager yum
subscription-manager-1.23.8-35.el8.x86_64
yum-4.0.9.2-5.el8.noarch
```

Download the consumer certificate RPM for your Satellite Server. This RPM installs the necessary certificates for accessing repositories on Satellite Server and configures Red Hat Subscription Manager to use the Satellite Server URL. The RPM is located in the **/pub** directory on the Satellite's web server. For example, for a Satellite Server with the host name **satellite.example.com**, enter the following command on the host:

```
[root@host ~]# yum localinstall \
http://satellite.example.com/pub/katello-ca-consumer-latest.noarch.rpm
```

## Using an Activation Key to Register a Host to Satellite Server

The following command registers the host to the default organization on Satellite Server and uses the **--activationkey** option to reference the **TestKey** activation key:

```
[root@host ~]# subscription-manager register --org "Default_Organization" \
--activationkey TestKey
The system has been registered with ID: d56b875c-3017-450f-a438-d5b8db35276e
Installed Product Current Status:
Product Name: Red Hat Enterprise Linux Server
Status: Subscribed
```

After you have registered a host to Satellite Server, it gains access to repository contents. However, before you can perform package and errata management you need to install the **Katello Agent** on the host system.

```
[root@host ~]# yum install katello-agent
```

In the Satellite web UI, navigate to **Hosts** → **Content Hosts** to verify successful registration of a host.



### References

For more information, refer to the *Managing Activation Keys* chapter in the *Red Hat Satellite 6.6 Content Management Guide* at  
[https://access.redhat.com/documentation/en-us/red\\_hat\\_satellite/6.6/html-single/content\\_management\\_guide/index#Managing\\_Activation\\_Keys](https://access.redhat.com/documentation/en-us/red_hat_satellite/6.6/html-single/content_management_guide/index#Managing_Activation_Keys)

## ► Guided Exercise

# Automating Registration of Content Hosts

In this exercise, you will register a system with Red Hat Satellite Server and configure it for management by Satellite Server using an activation key.

## Outcomes

You should be able to:

- Create an activation key.
- Use an activation key to register a host.

## Before You Begin

Log in as the **student** user on the **workstation** VM, using **student** as the password.

Run the **lab clients-automate start** command. This command determines if the **satellite** host is reachable on the network and prepares it for this exercise.

```
[student@workstation ~]$ lab clients-automate start
```

Create an activation key called **OperationsServers** in the **Operations** organization that can automatically attach a content host to the provided subscription, and add it to the **OpsServers** host collection when the content host registers to Satellite Server. In addition, ensure that all the repositories provided by the subscription are enabled by default.

- ▶ 1. Log in to the Satellite Server web UI, <https://satellite.lab.example.com>, as **admin** using **redhat** as the password.
- ▶ 2. Use the organization and location context menu to select the **Operations** organization and **Any Location** location context.
- ▶ 3. Create the activation key.
  - 3.1. Click **Content** → **Activation Keys** and then click **Create Activation Key**.
  - 3.2. Enter **OperationsServers** in the **Name** field.
  - 3.3. Select the **Unlimited Hosts** check box.
  - 3.4. Select the **Development** environment.
  - 3.5. Select the **Base** content view, and then click **Save**.
- ▶ 4. Set the release version for the content host.
  - 4.1. On the **OperationsServers** page, click the **Details** tab.

- 4.2. In the **Activation Key Content** section, click the pencil icon for **Release Version**.
- 4.3. Select **8.1** from the list, then click **Save**.
- ▶ 5. Attach subscriptions to the activation key.
  - 5.1. On the **OperationsServers** page, click the **Subscriptions** tab, then click the **Add** tab.
  - 5.2. Select the check boxes for the **Red Hat Enterprise Linux Server, Standard (Physical or Virtual Nodes)** and the **Red Hat Satellite Infrastructure Subscription** subscriptions.
  - 5.3. Click **Add Selected**.
- ▶ 6. Assign the **OpsServers** host collection to the activation key.
  - 6.1. On the **OperationsServers** page, click the **Host Collections** tab.
  - 6.2. Click the **Add** tab.
  - 6.3. Select the **OpsServers** check box, then click **Add Selected**.
- ▶ 7. Configure product content and enable repositories.
  - 7.1. On the **OperationsServers** page, click the **Repository Sets** tab.
  - 7.2. Select the **Red Hat Satellite Tools 6.6 for RHEL 8 x86\_64 (RPMs)** repository check box.
  - 7.3. From the **Select Action** list, select **Override to Enabled**.
- ▶ 8. Prepare the host for registration to Satellite Server.
  - 8.1. On **workstation**, use **ssh** to log in to **servera** as **student**. Use **sudo -i** to switch to **root**.

```
[student@workstation ~]$ ssh student@servera
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```
  - 8.2. Verify that the *subscription-manager* and *yum* packages are installed on the host system.

```
[root@servera ~]# rpm -q subscription-manager yum
subscription-manager-1.23.8-35.el8.x86_64
yum-4.0.9.2-5.el8.noarch
```
  - 8.3. Verify that the latest consumer RPM, *katello-ca-consumer-latest.noarch.rpm*, is installed from the following URL:

```
[root@servera ~]# yum localinstall \
http://satellite.lab.example.com/pub/katello-ca-consumer-latest.noarch.rpm
```

- ▶ 9. Register the host to Satellite Server using the activation key.

9.1. Register the system to the **Operations** organization. Reference the activation key with the **--activationkey** option.



**Note**

If the system is already registered, append the **--force** option to the **subscription-manager register** command. The command will unregister the system from Satellite, clear all local data, then register the system using the activation key.

```
[root@servera ~]# subscription-manager register --org 'Operations' \
--activationkey 'OperationsServers'
The system has been registered with ID: b5d099b3-44ba-4634-ad73-dd5b72dfbf02
Installed Product Current Status:
Product Name: Red Hat Enterprise Linux Server
Status: Subscribed
```

- 9.2. Install *Katello Agent* on the host system.

```
[root@servera ~]# yum install katello-agent
```

- ▶ 10. After the host has been registered, verify that it appears in the Satellite web UI.

10.1. Click **Hosts** → **Content Hosts**. Your newly registered host should appear in the table of content hosts.

10.2. Click the **servera.lab.example.com** link.

10.3. On the **Details** tab, in the **Content Host Content** section, confirm that **Release Version** is set to **8.1**. In the **Subscriptions** section, confirm that **Subscription Status** is set to **Fully entitled**.

On the **Host Collections** tab, confirm that the system belongs to the **OpsServers** host collection.

On the **Repository Sets** tab, confirm that the **Red Hat Satellite Tools 6.6 for RHEL 8 x86\_64 (RPMs)** repository is enabled.

## Finish

On the **workstation** machine, use the **lab clients-automate finish** command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab clients-automate finish
```

This concludes the guided exercise.

## ► Lab

# Registering Hosts

### Performance Checklist

In this lab, you will create a host collection and an activation key, and use these to register hosts.

### Outcomes

You should be able to:

- Create host collections.
- Create activation keys.
- Set the release version of a content host.
- Attach subscriptions to activation keys.
- Clear previous subscriptions and register servers using activation keys.

### Before You Begin

Log in to **workstation** as **student** using **student** as the password.

Run the **lab clients-review start** command. This command verifies that Red Hat Satellite Server is available. This command also ensures that the **Finance** organization and the **San Francisco** and **Tokyo** locations exist. This command also imports a subscription manifest for the **Finance** organization.

```
[student@workstation ~]$ lab clients-review start
```

1. Create the **FinanceServers** host collection with the description **Finance Servers**. Keep the **Unlimited Hosts** box selected.
2. Create the **FinanceKey** activation key. Use the **Library** environment, and the **Default Organization View** content view. Keep the **Unlimited Hosts** box selected.
3. Set the release version to **8.1** for any host that uses the **FinanceKey** activation key.
4. Attach the **Red Hat Enterprise Linux Server, Standard (Physical or Virtual Nodes)**, and the **Red Hat Satellite Infrastructure Subscription** subscriptions to the **FinanceKey** activation key.
5. Assign the **FinanceServers** host collection to the **FinanceKey** activation key.
6. Clean any previous registration from **serverb** and register using the **FinanceKey** activation key.

### Evaluation

As the **student** user on the **workstation** machine, use the **lab clients-review grade** command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab clients-review grade
```

## Finish

As the **student** user on the **workstation** machine, use the **lab clients-review finish** command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab clients-review finish
```

This concludes the lab.

## ► Solution

# Registering Hosts

### Performance Checklist

In this lab, you will create a host collection and an activation key, and use these to register hosts.

### Outcomes

You should be able to:

- Create host collections.
- Create activation keys.
- Set the release version of a content host.
- Attach subscriptions to activation keys.
- Clear previous subscriptions and register servers using activation keys.

### Before You Begin

Log in to **workstation** as **student** using **student** as the password.

Run the **lab clients-review start** command. This command verifies that Red Hat Satellite Server is available. This command also ensures that the **Finance** organization and the **San Francisco** and **Tokyo** locations exist. This command also imports a subscription manifest for the **Finance** organization.

```
[student@workstation ~]$ lab clients-review start
```

1. Create the **FinanceServers** host collection with the description **Finance Servers**. Keep the **Unlimited Hosts** box selected.
  - 1.1. On **workstation**, navigate to <https://satellite.lab.example.com>.
  - 1.2. Log in to the Satellite Server web UI as **admin** using **redhat** as the password.
  - 1.3. Use the Organization and Location context menus to select the **Finance** organization and **Any Location** context.
  - 1.4. Navigate to **Hosts → Host Collections** and then click **Create Host Collection**.
  - 1.5. Enter **FinanceServers** in the **Name** field and **Finance Servers** in the **Description** field.
  - 1.6. Ensure **Unlimited Hosts** is selected, and then click **Save**.
2. Create the **FinanceKey** activation key. Use the **Library** environment, and the **Default Organization View** content view. Keep the **Unlimited Hosts** box selected.
  - 2.1. Navigate to **Content → Activation Keys** and then click **Create Activation Key**.

- 2.2. Enter **FinanceKey** in the **Name** field.
- 2.3. Select **Unlimited Hosts**.
- 2.4. Select the **Library** environment.
- 2.5. Select the **Default Organization View** content view, and then click **Save**.
3. Set the release version to **8.1** for any host that uses the **FinanceKey** activation key.
  - 3.1. In the **Activation Key Content** section, click the pencil icon for **Release Version**.
  - 3.2. Select **8.1** from the list and then click **Save**.
4. Attach the **Red Hat Enterprise Linux Server, Standard (Physical or Virtual Nodes)**, and the **Red Hat Satellite Infrastructure Subscription** subscriptions to the **FinanceKey** activation key.
  - 4.1. On the **Activation Keys → FinanceKey** page, click the **Subscriptions** tab and then click the **Add** tab.
  - 4.2. Select the check boxes for the **Red Hat Enterprise Linux Server, Standard (Physical or Virtual Nodes)** and **Red Hat Satellite Infrastructure Subscription** subscriptions.
  - 4.3. Click **Add Selected**.
5. Assign the **FinanceServers** host collection to the **FinanceKey** activation key.
  - 5.1. Click the **Host Collections** tab and then click the **Add** tab.
  - 5.2. Select the checkbox for **FinanceServers** and then click **Add Selected**.
6. Clean any previous registration from **serverb** and register using the **FinanceKey** activation key.
  - 6.1. On **workstation**, use **ssh** to log in to **serverb** as **student**. Use **sudo -i** to switch to **root**.

```
[student@workstation ~]$ ssh student@serverb
[student@serverb ~]$ sudo -i
[sudo] password for student: student
[root@serverb ~]#
```

- 6.2. Use **subscription-manager clean** to clear any old registration data from **serverb**. Ignore any warnings related to the activation of the **yum** plug-ins.

```
[root@serverb ~]# subscription-manager clean
All local data removed
```

**WARNING**

The yum/dnf plugins: /etc/dnf/plugins/subscription-manager.conf, /etc/dnf/plugins/product-id.conf were automatically enabled for the benefit of Red Hat Subscription Management. If not desired, use "subscription-manager config --rhsm.auto\_enable\_yum\_plugins=0" to block this behavior.

- 6.3. Download and install a copy of the CA Certificate from the following URL on Satellite Server:

```
[root@serverb ~]# yum localinstall \
http://satellite.lab.example.com/pub/katello-ca-consumer-latest.noarch.rpm
...output omitted...
```

- 6.4. Register the system to the **Finance** organization. Reference the activation key with the **--activationkey** option.

```
[root@serverb ~]# subscription-manager register --org 'Finance' \
--activationkey 'FinanceKey'
The system has been registered with ID: b5d099b3-44ba-4634-ad73-dd5b72dfbf02
Installed Product Current Status:
Product Name: Red Hat Enterprise Linux Server
Status: Subscribed
```



#### Note

If the system is already registered, append the **--force** option to the **subscription-manager register** command. The command will unregister the system from Satellite, clear all local data, then register the system using the activation key.

- 6.5. Exit from the **serverb** system and return to **workstation**.

This concludes the lab.

## Evaluation

As the **student** user on the **workstation** machine, use the **lab clients-review grade** command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab clients-review grade
```

## Finish

As the **student** user on the **workstation** machine, use the **lab clients-review finish** command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab clients-review finish
```

# Summary

---

In this chapter, you learned:

- The **subscription-manager** command-line tool is used to register hosts to both Red Hat Customer Portal and Red Hat Satellite.
- The *katello-ca-consumer* package updates the content source location of the host and allows the host to download content from the content source specified in Satellite.
- Host collections simplify host administration by allowing administrators to perform tasks on a group of hosts defined in a host collection. Tasks include the installation and update of packages, package groups, and errata.
- Activation keys can help automate system registration and subscription attachment.
- Activation keys define properties for content hosts, such as available products and repositories, life-cycle environments, and content views.

## Chapter 4

# Deploying Software to Hosts

### Goal

Manage software deployment to registered hosts of your Red Hat Satellite infrastructure and practice managing environment paths, life-cycle environments, and content views.

### Objectives

- Manage software installed on specific registered hosts using content views and life-cycle environments.
- Create and manage content view filters and composite content views to provide content subsets or supersets to managed hosts.
- Inspect, filter, and apply Red Hat errata to content views for precise patch management.
- Inspect available module streams, use Satellite to install them on hosts, and apply module stream errata.

### Sections

- Controlling Software with Content Views (and Guided Exercise)
- Creating Content View Filters (and Guided Exercise)
- Managing and Applying Errata to Hosts (and Guided Exercise)
- Managing Module Streams for RHEL 8 Hosts (and Guided Exercise)

### Lab

Deploying Software to Hosts

# Controlling Software with Content Views

## Objectives

After completing this section, you should be able to manage software installed on specific registered hosts using content views and life-cycle environments.

## Managing Content Views

You can use content views to define which software versions a particular environment uses. For example, a production environment might use a content view containing packages tested for stability, but a development environment might use a content view containing later package versions.

Satellite Server stores and manages repositories in each content view across each environment. When you promote a content view from one environment to the next in the application life cycle, the respective repository updates and publishes the packages.



Figure 4.1: New software in the Development content view

The repositories for **Testing** and **Production** contain the *example\_software-1.0-0.noarch.rpm* package. If you promote version 2 of the content view from **Development** to **Testing**, the repository for **Testing** regenerates and subsequently contains the *example\_software-1.1-0.noarch.rpm* package.



Figure 4.2: New software promoted to the Testing content view

## Publishing a Content View

After you create a content view, you must publish it to the **Library** so that host systems can see and use it. If you make any changes to a content view, such as updating repositories or filters, you must republish that content view.

Each content view publication results in the creation of a new content view version. Versions are numbered sequentially for identification and version control purposes.

To publish a content view, log in to the Satellite web UI as an **admin** user. Choose the organization that hosts the content view you want to publish.

Navigate to **Content** → **Content Views** and select the content view to publish. Click **Publish New Version**. Optionally, add a description then click **Save**.

A new version of the selected content view is created and promoted to the **Library** environment.

## Promoting a Content View

Once a new content view has been published to the **Library**, it can then be *promoted* to the first life-cycle environment in an environment path.

When you promote a content view to an environment, you can systematically release software packages and errata to each life-cycle environment in an environment path. In accordance with the *Software Development Life Cycle (SDLC)* model, content views can only be promoted sequentially through an environment path. Satellite Server does not allow a life-cycle environment to be skipped during content view promotion.

### To Promote a Content View:

- Log in to the Satellite web UI as the **admin** user and choose the organization that contains the content view to promote.
- Navigate to **Content** → **Content Views** and select the content view to promote.
- On the **Versions** tab, click **Promote** for the version of the content view to promote.
- A list of applicable environment paths displays. For each environment path, only one life-cycle environment is available as the target for the content view promotion.
- Select the desired life-cycle environment and then click **Promote Version**.



### References

For more information, refer to the *Managing Content Views* chapter in the *Red Hat Satellite 6.6 Content Management Guide* at

[https://access.redhat.com/documentation/en-us/red\\_hat\\_satellite/6.6/html-single/content\\_management\\_guide/index#Managing\\_Content\\_VIEWS](https://access.redhat.com/documentation/en-us/red_hat_satellite/6.6/html-single/content_management_guide/index#Managing_Content_VIEWS)

## ► Guided Exercise

# Controlling Software with Content Views

In this exercise, you will work with hosts in two separate life-cycle environments for testing and production, and deploy different versions of the same software to hosts in the two environments.

## Outcomes

You should be able to:

- Register a host to a specific life-cycle environment and content view.
- Deploy software from different life-cycle environments and content views.

## Before You Begin

Log in as the **student** user on **workstation** using **student** as the password.

Run the **lab software-control start** command. This command determines if the **satellite** and **serverc** hosts are reachable on the network. It also verifies that Red Hat Satellite is running, the **servera** host is registered, and the **Base** content view and the **QA** life-cycle environment are available.

```
[student@workstation ~]$ lab software-control start
```

- ▶ 1. Log in to the Satellite web UI as **admin** using **redhat** as the password. Select the **Operations** organization and the **Any Location** location.
  - 1.1. Use your browser to navigate to <https://satellite.lab.example.com> and log in as **admin** using **redhat** as the password.
  - 1.2. Select the **Operations** organization and **Any Location** location.
- ▶ 2. Verify the two versions available for the **Base** content view.
  - 2.1. Click **Content** → **Content Views** and then click **Base**.
  - 2.2. Verify that both **Version 1.0** and **Version 2.0** are available.
  - 2.3. Click **Promote** for the **Version 1.0** version, select the **Development** life-cycle, and then click **Promote Version**.
  - 2.4. Click **Promote** in the **Force Promote** window.
- ▶ 3. Verify that the life-cycle and content view exist for **servera**.
  - 3.1. Click **Hosts** → **Content Hosts** and then click **servera.lab.example.com**.
  - 3.2. Verify that the value for the **Content View** field is **Base**.
  - 3.3. Verify that the selected value for the **Lifecycle Environment** field is **Development**.

- 4. Register **serverc** in the **Operations** organization and the **QA/Base** environment.
- 4.1. On **workstation**, use **ssh** to log in to **serverc** as **student**. Use **sudo -i** to switch to **root**.

```
[student@workstation ~]$ ssh student@serverc
[student@serverc ~]$ sudo -i
[sudo] password for student: student
[root@serverc ~]#
```

- 4.2. Use **subscription-manager clean** to clear existing registration data from **serverc**. Ignore any warnings related to the activation of the **yum** plug-ins.

```
[root@serverc ~]# subscription-manager clean
All local data removed
```

WARNING

The yum/dnf plugins: /etc/dnf/plugins/subscription-manager.conf, /etc/dnf/plugins/product-id.conf were automatically enabled for the benefit of Red Hat Subscription Management. If not desired, use "subscription-manager config --rhsm.auto\_enable\_yum\_plugins=0" to block this behavior.

- 4.3. Install the *katello-ca-consumer-latest* package from the Red Hat Satellite server (*satellite.lab.example.com*). Installing this package executes a script that loads the necessary configuration, including the CA certificate, on to the system to securely communicate with the appropriate Red Hat Satellite Server. Use the `http://satellite.lab.example.com/pub/katello-ca-consumer-latest.noarch.rpm` URL to install *katello-ca-consumer-latest*.

```
[root@serverc ~]# yum localinstall \
http://satellite.lab.example.com/pub/katello-ca-consumer-latest.noarch.rpm
...output omitted...
Is this ok [y/N]: y
...output omitted...
Complete!
```

- 4.4. Use **subscription-manager register** to register **serverc** to Red Hat Satellite Server. Use the **--org** option to specify the **Operations** organization. When prompted, enter **admin** as the username and **redhat** as the password.

```
[root@serverc ~]# subscription-manager register --org Operations \
--environment QA/Base
Registering to: satellite.lab.example.com:443/rhsm
Username: admin
Password: redhat
The system has been registered with ID: f134704d-3b8d-4a03-b0c1-e2b8491c4bdc
The registered system name is: serverc.lab.example.com
```

The registration process takes a few seconds to complete.

- 5. Add the available subscription to **serverc** and verify the enabled module streams for **serverc**.

- 5.1. Click **Hosts** → **Content Hosts**, and click the fully qualified domain name (FQDN) of **serverc**.
- 5.2. Select **8.1** on the **Release Version** drop-down menu. When done, click **Save**.
- 5.3. On the **serverc.lab.example.com** page, click **Subscriptions** → **Subscriptions**.
- 5.4. Click the **Add** tab below **Subscription Details**.
- 5.5. Select the check boxes for **Red Hat Enterprise Linux Server, Standard (Physical or Virtual Nodes)**, and **Red Hat Satellite Infrastructure Subscription**. When done, click **Add Selected**.
- 5.6. Click the **List/Remove** tab below **Subscription Details**, and confirm that the added subscription displays.

► 6. Install a package on **serverc**, and verify that this package is not available on **servera**.

- 6.1. Install the *ant* package on **serverc**.

```
[root@serverc ~]# yum install ant
...output omitted...
Is this ok [y/N]: yes
...output omitted...
```

- 6.2. Log off from **serverc**, and use **ssh** to log in to **servera** as **student**. Use **sudo -i** to switch to **root**.

```
[student@workstation ~]$ ssh student@servera
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

- 6.3. Verify that the *ant* package is not available on **servera**.

```
[root@servera ~]# yum install ant
...output omitted...
No match for argument: ant
...output omitted...
```

- 6.4. Log off from **servera**.

## Finish

On the **workstation** machine, use the **lab software-control finish** command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab software-control finish
```

This concludes the guided exercise.

# Creating Content View Filters

---

## Objectives

After completing this section, you should be able to create and manage content view filters and composite content views to provide content subsets or supersets to managed hosts.

## Filtering Content Views

Content views make content available through their associations with repositories. By default, content views display all repository contents. However, you can apply filters to limit the range of content that is displayed.

Satellite Server offers several types of filters. For example, you can filter by package name or package group name. This is useful when trying to maintain a consistent software profile on systems in an environment.

You can also filter by erratum ID, date, or type. This allows you to regulate errata availability so that you can release them in a controlled manner through life-cycle environments.

With all content types, you can filter using inclusion or exclusion logic. For example, when filtering a content view by package name, you can configure a filter to either include or exclude packages.

### To Create a Filter for a Content View:

- Click **Content** → **Content Views** and click the name of the content view you want to filter.
- Click **Yum Content** → **Filters** and then click **New Filter**.
- Add a name for the filter, and select which content type and inclusion type to use. Optionally add a description for the filter.

## Choosing a Content Type

The content type defines the type of content to be filtered by the content view filter. Red Hat Satellite supports four content type filters:

### Package

Filters packages based on their name and version number.

### Package Group

Filters packages based on their associated package group. The list of package groups includes those provided by the repositories included in the content view.

### Erratum (by ID)

Filters the errata available in the content view repositories by ID.

### Erratum (by Date and Type)

Filters errata based on a date range and an errata type (Bug fix, Enhancement, or Security).

## Choosing an Inclusion Type

The *inclusion type* defines the initial content your filter is going to use. There are two filter types available: **Include** and **Exclude**. If you use the **Include** type, you start with no content, and you use the filter to select the content you want to add. You can use the **Include** type to combine multiple content items.

The **Exclude** type filters start with all the content, and the filter rules select which content is removed. This filter type is useful when you want to exclude just some packages from the content view.



### Note

You can use a combination of **Include** and **Exclude** filters for a content view. When you publish such a content view, the **Include** filters run first and then the **Exclude** filters. Because of this order, you need to first think about the content you want to add to your filtered content view, and then define which content you want to exclude from that content.

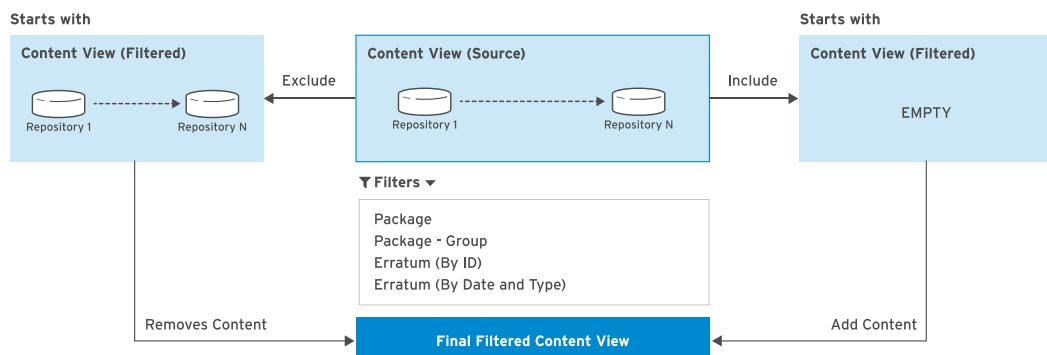


Figure 4.3: Filtering content views

## Composite Content Views

Composite content views are content views made up of other content views. Composite content views allow for the reuse of content views and simplify content view management.

For example, you can create a content view called **RHEL 7** which includes the **Red Hat Enterprise Linux 7 Server RPMs x86\_64 7Server** repository. You can create a separate content view named **Custom Application** to provide access to a repository that contains third-party application software. You can use a composite content view to grant content hosts access to both content views.

You can create a composite content view called **Application Stack** which consists of both the **RHEL 7** and **Custom Application** content views. Content hosts assigned to the **Application Stack** composite content view would have access to both the **Red Hat Enterprise Linux 7 Server RPMs x86\_64 7Server** repository, as well as the repository which contains third-party application software.



### Note

A composite content view can only include content views that do not have overlapping repositories. For example, if two content views both contained the **Red Hat Enterprise Linux 7 Server RPMs x86\_64 7Server** repository, they cannot both be included in a composite content view.

### To Create a Composite Content View:

- Click **Content** → **Content Views** and then click **Create New View**.
- Enter a name, label, and optionally a description for the composite content view.
- Select **Composite View** and then click **Save**.
- On the **Add Content Views** page, click **Add Content Views** to add available content views to the composite content view.



### References

For more information, refer to the *Managing Content Views* chapter in the *Content Management Guide* at  
[https://access.redhat.com/documentation/en-us/red\\_hat\\_satellite/6.6/html-single/content\\_management\\_guide/index](https://access.redhat.com/documentation/en-us/red_hat_satellite/6.6/html-single/content_management_guide/index)

## ► Guided Exercise

# Creating Content View Filters

In this exercise, you will create a composite content view, and use a content view filter to limit which packages are visible to hosts.

## Outcomes

You should be able to:

- Create a composite content view.
- Configure a content view filter.

## Before You Begin

Log in as the **student** user on **workstation** using **student** as the password.

Run the **lab software-filter start** command. This command determines if the **satellite** host is reachable on the network, verifies that Red Hat Satellite and the **Base** content view are available, and deletes **serverc** configuration in the Satellite Server.

```
[student@workstation ~]$ lab software-filter start
```

- ▶ 1. Log in to the Satellite web UI as **admin** with **redhat** as the password. Select the **Operations** organization and the **Any Location** location.
  - 1.1. Navigate to <https://satellite.lab.example.com>.
  - 1.2. Log in as **admin** with **redhat** as a password.
  - 1.3. Select the **Operations** organization and the **Any Location** location.
- ▶ 2. Enable synchronization of the **Red Hat Ansible Engine 2.8 for RHEL 8 x86\_64 (RPMs)** repository from the local CDN.
  - 2.1. Click **Content** → **Red Hat Repositories**.
  - 2.2. Enter **ansible-2.8-for-rhel-8-x86\_64-rpms** in the **Search** field, and click **Search**.
  - 2.3. Expand **Red Hat Ansible Engine 2.8 for RHEL 8 x86\_64 (RPMs)**. Click the enable icon next to this repository.
- ▶ 3. Use the Satellite Server web UI to synchronize the **Red Hat Ansible Engine 2.8 for RHEL 8 x86\_64 (RPMs)** repository.
  - 3.1. Click **Content** → **Products** and then click **Red Hat Ansible Engine**.
  - 3.2. On the **Repositories** tab, select the **Red Hat Ansible Engine 2.8 for RHEL 8 x86\_64 RPMs x86\_64** repository check box, and click **Sync Now**. Wait for the task to complete. Watch the progress of the synchronization on the **Tasks** tab.

- ▶ 4. Create a new content view named **Ansible**.
    - 4.1. Click **Content** → **Content Views** and click **Create New View**.
    - 4.2. On the **Create Content View** page, enter **Ansible** in the **Name** field. Notice that the **Label** field is automatically populated with the name provided in the **Name** field.
    - 4.3. Enter **Ansible Packages** in the **Description** field and click **Save** to display the **Repository Selection** page.
  - ▶ 5. Select the **Red Hat Ansible Engine 2.8 for RHEL 8 x86\_64 RPMs x86\_64** repository and click **Add Repositories**.
  - ▶ 6. Create a filter for the **Ansible** content view and configure it to exclude Ansible packages prior to version 2.8.4.
    - 6.1. Click **Yum Content** → **Filters** and click **New Filter**.
    - 6.2. Complete the **Add New Yum Filter** page with the following details:

| Field          | Value                                       |
|----------------|---------------------------------------------|
| Name           | Ansible-older-2.8.4                         |
| Content Type   | Package                                     |
| Inclusion Type | Exclude                                     |
| Description    | Excluding Ansible packages older than 2.8.4 |

Ensure all the details are correct and click **Save**.
  - 6.3. Click **Add Rule** to start adding a new rule to the filter.
  - 6.4. Create a new rule with the following details:

| Field           | Value     |
|-----------------|-----------|
| RPM Name        | ansible   |
| Architecture    | noarch    |
| Version         | Less Than |
| Maximum Version | 2.8.4     |

Ensure all the details are correct and click **Save**.
- ▶ 7. Publish a new version of the **ansible** content view, and promote this version to the **Development** life-cycle environment.
    - 7.1. Click **Publish New Version** to display the **Publish** page.
    - 7.2. Enter **Ansible Repository** in the **Description** field, and click **Save**. The **Versions** page displays, where you can monitor the publishing process. This process takes a

short time to complete. The **ansible** content view is now published in the **Library** environment.

- 7.3. In the **Actions** column, click **Promote**.
- 7.4. Select the check box for **Development**, and enter **Ansible Repository** in the **Description** field.
- 7.5. Click **Promote Version** to start the promotion and display the **Versions** page. The promotion process takes a short time to complete.
- ▶ 8. Create a new composite view named **BaseAnsibleComposite**, which includes both the **Base** and **Ansible** content views.
  - 8.1. Click **Content** → **Content Views** and click **Create New View**.
  - 8.2. On the **Create Content View** page, enter **BaseAnsibleComposite** in the **Name** field. The **Label** field is automatically populated from the **Name** field content.
  - 8.3. Enter **Base and Ansible Packages** in the **Description** field.
  - 8.4. Select the **Composite View** and **Auto Publish** check boxes, and click **Save**.
  - 8.5. Select the **Ansible** and **Base** content views and click **Add Content Views**.
  - 8.6. Click the **List/Remove** tab and verify that both content views are listed.
- ▶ 9. Publish a new version of the **BaseAnsibleComposite** composite view and promote this version to the **Development** life-cycle environment.
  - 9.1. Click **Publish New Version**.
  - 9.2. Enter **Base and Ansible Content Views Repositories** in the **Description** field, and click **Save**. The **Versions** page displays, where you can monitor the publishing process. This process takes a short time to complete. The **BaseAnsibleComposite** composite view is now published to the **Library** environment.
  - 9.3. In the **Actions** column, click **Promote**.
  - 9.4. Select the **Development** life-cycle, and enter **Base and Ansible Content Views Repository** in the **Description** field.
  - 9.5. Click **Promote Version** to start the promotion and display the **Versions** page. The promotion takes a short time to complete.
- ▶ 10. Register **serverc** in the **Operations** organization and the **Development/ BaseAnsibleComposite** environment.
  - 10.1. On **workstation**, use **ssh** to log in to **serverc** as **student**. Use **sudo -i** to switch to **root**.

```
[student@workstation ~]$ ssh student@serverc
[student@serverc ~]$ sudo -i
[sudo] password for student: student
[root@serverc ~]#
```

10.2. Use **subscription-manager clean** to clear existing registration data from **serverc**. Ignore warnings related to the activation of the **yum** plug-ins.

```
[root@serverc ~]# subscription-manager clean
All local data removed

WARNING

The yum/dnf plugins: /etc/dnf/plugins/subscription-manager.conf, /etc/dnf/
plugins/product-id.conf were automatically enabled for the benefit of Red Hat
Subscription Management. If not desired, use "subscription-manager config --
rhsm.auto_enable_yum_plugins=0" to block this behavior.
```

10.3. Use **subscription-manager register** to register **serverc** to Red Hat Satellite Server. Use the **--org** option to specify the **Operations** organization. When prompted, use **admin** as the username and **redhat** as the password.

```
[root@serverc ~]# subscription-manager register --org Operations \
--environment Development/BaseAnsibleComposite
Registering to: satellite.lab.example.com:443/rhsm
Username: admin
Password: redhat
The system has been registered with ID: f134704d-3b8d-4a03-b0c1-e2b8491c4bdc
The registered system name is: serverc.lab.example.com
```

The registration process takes a few seconds to complete.

- ▶ 11. Add the available subscriptions to **serverc**, and verify the enabled module streams for **serverc**.
  - 11.1. Click **Hosts** → **Content Hosts**, and click the fully qualified domain name (FQDN) of **serverc**.
  - 11.2. Select **8.1** on the **Release Version** drop-down menu. When done, click **Save**.
  - 11.3. Click **Subscriptions** → **Subscriptions**, and click the **Add** tab below **Subscription Details**.
  - 11.4. Select the **Red Hat Enterprise Linux Server, Standard (Physical or Virtual Nodes)** and **Red Hat Satellite Infrastructure Subscription** check boxes, and then click **Add Selected**.
  - 11.5. Click the **List/Remove** tab below **Subscription Details**, and confirm that the added subscriptions display.
- ▶ 12. Verify which **ansible** package versions are available on **serverc**.
  - 12.1. Return to the terminal and verify which repositories are available on **serverc**.

```
[root@serverc ~]# subscription-manager repos --list
+-----+
 Available Repositories in /etc/yum.repos.d/redhat.repo
+-----+
 Repo ID: satellite-tools-6.6-for-rhel-8-x86_64-rpms
```

```
Repo Name: Red Hat Satellite Tools 6.6 for RHEL 8 x86_64 (RPMs)
Repo URL: https://satellite.lab.example.com/pulp/repos/Operations/Development/B
aseAnsibleComposite/content/dist/layered/rhel8/x86_64/sat-tools/6.6/o
s
Enabled: 0

Repo ID: ansible-2.8-for-rhel-8-x86_64-rpms
Repo Name: Red Hat Ansible Engine 2.8 for RHEL 8 x86_64 (RPMs)
Repo URL: https://satellite.lab.example.com/pulp/repos/Operations/Development/B
aseAnsibleComposite/content/dist/layered/rhel8/x86_64/ansible/2/os
Enabled: 0

Repo ID: rhel-8-for-x86_64-appstream-rpms
Repo Name: Red Hat Enterprise Linux 8 for x86_64 - AppStream (RPMs)
Repo URL: https://satellite.lab.example.com/pulp/repos/Operations/Development/B
aseAnsibleComposite/content/dist/rhel8/$releasever/x86_64/appstream/o
s
Enabled: 1

Repo ID: rhel-8-for-x86_64-baseos-rpms
Repo Name: Red Hat Enterprise Linux 8 for x86_64 - BaseOS (RPMs)
Repo URL: https://satellite.lab.example.com/pulp/repos/Operations/Development/B
aseAnsibleComposite/content/dist/rhel8/$releasever/x86_64/baseos/os
Enabled: 1
```

12.2. Enable the **ansible-2.8-for-rhel-8-x86\_64-rpms** repository.

```
[root@serverc ~]# subscription-manager repos \
--enable=ansible-2.8-for-rhel-8-x86_64-rpms
```

12.3. Verify which versions of the *ansible* package are available.

```
[root@serverc ~]# yum search ansible --showduplicates
...output omitted...
===== Name Exactly Matched: ansible =====
ansible-2.8.5-2.el8ae.noarch : SSH-based configuration management, deployment, ...
ansible-2.8.6-1.el8ae.noarch : SSH-based configuration management, deployment, ...
ansible-2.8.4-1.el8ae.noarch : SSH-based configuration management, deployment, ...
ansible-2.8.7-1.el8ae.noarch : SSH-based configuration management, deployment, ...
...output omitted...
```

12.4. Log off from **serverc**.

## Finish

On the **workstation** machine, use the **lab software-filter finish** command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab software-filter finish
```

This concludes the guided exercise.

# Managing and Applying Errata to Hosts

## Objectives

After completing this section, you should be able to inspect, filter, and apply Red Hat errata to content views for precise patch management.

## Errata Management

After their initial release, Red Hat software packages may receive updates in the form of errata. Red Hat groups several packages into an erratum, with an advisory that includes a description of that erratum. There are three types of advisories provided, listed below in order of importance.

### Security advisory

Addresses security issues found in software packages. The security impact of the issue is categorized as Low, Moderate, Important, or Critical.

### Bug fix advisory

Provides fixes to bugs discovered in software packages.

### Product enhancement advisory

Provides enhancements and new features added to the package.

Content view errata filtering provides granular control over which updates are propagated to content views. The promotion of content views to life-cycle environments allows precise control over when and to which systems the updates are applied.



### Note

A single erratum can address multiple issue types, such as a security issue and a product enhancement. Errata are categorized according to the most important advisory type they contain. For example, errata categorized as product enhancement errata can contain only enhancement updates, but security errata can contain security and bug fixes, as well as product enhancements.

## Viewing Available Errata

In Red Hat Satellite, errata are further classified as either *applicable* or *installable* to indicate their status in relationship to specific content hosts,

Applicable errata apply to content hosts and update packages present on the content host.  
Applicable errata are not yet available at the content host.

Installable errata also apply to content hosts, are available to the content host's life-cycle environment and content view, but have not been installed. These errata can be installed by users who have permissions to manage content hosts, but not entitled to errata management.

To review the available errata, navigate to **Content → Errata**. You can filter the list by repository and errata type (applicable or installable). Use a query string in **parameter operator value** form to limit the errata displayed. Refer to the table titled "Summary of Query Parameters for Filtering Errata" for a summary of the query parameters for filtering errata.

## Summary of Query Parameters for Filtering Errata

| Parameter    | Description                                                                                                                                                                                  | Example                           |
|--------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------|
| bug          | Search by Bugzilla ID.                                                                                                                                                                       | bug = BZ#12345                    |
| cve          | Search by CVE number.                                                                                                                                                                        | cve = CVE-2015-0101               |
| id           | Search by errata ID.                                                                                                                                                                         | id = RHBA-2015:2000               |
| issued       | Search by errata issue date. Accepted values include exact date, "Feb 16, 2015", or keywords, "Yesterday" or "1 hour ago". Use less-than (<) and greater-than (>) operators for time ranges. | issued < "Jan 1, 2015"            |
| package      | Search by the full package build name.                                                                                                                                                       | package = glib2-2.22.5-6.el6.i686 |
| package_name | Search by the package name.                                                                                                                                                                  | package_name = glib2              |
| severity     | Search by severity level. This only applies to security errata.                                                                                                                              | severity = Important              |
| title        | Search by advisory title. The tilde allows case-insensitive matches.                                                                                                                         | title ~ apache                    |
| type         | Search by advisory type.                                                                                                                                                                     | type = bugfix                     |
| updated      | Search by last update date. Accepts the same search values as the <b>issued</b> parameter.                                                                                                   | updated = "2 days ago"            |

## Applying Errata

You can apply errata to a single host or to multiple hosts at the same time. Satellite Server can apply both installable errata and applicable errata. Although applicable errata are pertinent to content hosts, they are not available in the content host's content view and life-cycle environment. Satellite 6 still allows administrators to apply applicable errata to content hosts, by automatically creating a minor version of the affected content view to include the applicable errata, then promoting this minor version to the appropriate life-cycle environment.

### To Apply Errata to Specific Content Hosts:

- Navigate to **Content → Errata** to determine which errata are available.
- Select the check boxes for the errata you want to apply, and click **Apply Errata**. When done, you can select which content hosts to apply the errata.



### References

For more information, refer to the *Managing Errata* chapter in the *Content Management Guide* at  
[https://access.redhat.com/documentation/en-us/red\\_hat\\_satellite/6.6/html-single/content\\_management\\_guide/index](https://access.redhat.com/documentation/en-us/red_hat_satellite/6.6/html-single/content_management_guide/index)

## ► Guided Exercise

# Managing and Applying Errata to Hosts

In this exercise, you will create an errata content view filter, add errata to an incremental content view, and apply errata to managed hosts.

## Outcomes

You should be able to:

- Create an errata content view filter.
- Apply errata to managed hosts.

## Before You Begin

Log in as the **student** user on **workstation** using **student** as the password.

Run the **lab software-errata start** command. This command determines if the **satellite** host is reachable on the network and verifies that Red Hat Satellite and the **Base** content view are available, and that the **servera** host is registered.

```
[student@workstation ~]$ lab software-errata start
```

- ▶ 1. Log in to the Satellite web UI as **admin** using **redhat** as the password. Select the **Operations** organization and the **Any Location** location.
  - 1.1. Navigate to <https://satellite.lab.example.com> and log in as **admin** using **redhat** as the password.
  - 1.2. Select the **Operations** organization and the **Any Location** location.
- ▶ 2. Create a filter for the **Base** content view to exclude all non-security errata from August 1st onwards. Use this filter to create a new version of the **Base** content view, and promote this version to the **Development** life-cycle environment.
  - 2.1. Click **Content** → **Content Views** and then click the **Base** content view.
  - 2.2. Click **Yum Content** → **Filters** and then click **New Filter**.
  - 2.3. Create a new filter with the following details:

| Field          | Value                      |
|----------------|----------------------------|
| Name           | Non-security Errata Filter |
| Content Type   | Erratum - Date and Type    |
| Inclusion Type | Exclude                    |

| Field       | Value                                         |
|-------------|-----------------------------------------------|
| Description | Excluding non-security errata from August 1st |

Ensure all the details are correct and click **Save**.

- 2.4. Clear the **Security** check box in the **Errata Type** section. Select August 1st 2019 as **Start Date**, and click **Save**.
  - 2.5. Click **Publish New version** to publish a new version of the **Base** content view. Enter **Adding non-security errata filter** in the **Description** field, and click **Save**. Verify that the new version has fewer packages than the previous content view version.
  - 2.6. Click **Promote** for the version you just published, select the **Development** environment, and click **Promote Version**.
- 3. Apply the previous version of the **Base** content view with errata to the **servera** host.
- 3.1. Click **Hosts** → **Content Hosts** and click **servera.lab.example.com**.
  - 3.2. On the **Errata** tab, search for the errata with the **RHSA-2019:2692** ID using **id=RHSA-2019:2692** in the **Filter** field. Verify that this errata contains the *libnnghttp2-1.33.0-1.el8\_0.1* package.
  - 3.3. Go back to the **Errata** tab and select the errata with the **RHSA-2019:2692** ID. Click **via Katello Agent** in the **Apply Selected** drop-down menu. Wait until the task completes.
- 4. Open a terminal on **servera** and verify that the *libnnghttp2-1.33.0-1.el8\_0.1* package is available.
- 4.1. On **workstation**, use **ssh** to log in to **servera** as **student**. Use **sudo -i** to switch to **root**.

```
[student@workstation ~]$ ssh student@servera
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

- 4.2. Verify that the *libnnghttp2-1.33.0-1.el8\_0.1* package is installed.

```
[root@servera ~]# yum history info
Updating Subscription Management repositories.
Transaction ID : 7
Begin time   : Tue 17 Dec 2019 03:29:40 AM EST
Begin rpmdb  : 581:97dfb0746fde98b12cffd5612bd294930ab9069f
End time     : Tue 17 Dec 2019 03:29:41 AM EST (1 seconds)
End rpmdb    : 581:e380d5e74f197c62b59fa1546645a4ba353a45bc
User        : System <unset>
Return-Code  : Success
Releasever  : 8
Command Line :
```

```
Packages Altered:
```

```
Upgrade libnghhttp2-1.33.0-1.el8_0.1.x86_64 @rhel-8-for-x86_64-baseos-rpms
Upgraded libnghhttp2-1.33.0-1.el8.x86_64 @@System
```

4.3. Log off from **servera**.

## Finish

On the **workstation** machine, use the **lab software-errata finish** command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab software-errata finish
```

This concludes the guided exercise.

# Managing Module Streams for RHEL 8 Hosts

## Objectives

After completing this section, you should be able to inspect available module streams, use Red Hat Satellite to install them on hosts, and apply module stream errata.

## Package Modularity in RHEL 8

Red Hat Enterprise Linux 8 brings some additions in package management capabilities for Yum: *modules*, *streams*, and *profiles*.

A module is a collection of packages that are installed together, such as an application, a language stack, a database, or a set of tools. Modules resolve dependency issues between earlier and later package versions. A module can include one or more streams, which are different versions of the software provided by the module. A module has just one stream active at a time; by default, the stream containing the latest version of the software. Modules also include profiles, which are lists of packages that support a use case, for example, to deploy a minimal install.



### Note

The module feature replaces Red Hat Software Collections.

Installing a stream wholly replaces the previously installed stream, whether as an upgrade or downgrade. Contrast this with a legacy package upgrade that adds or removes only changed package files. For example, you could install the Perl 5.24 stream, and even though Perl 5.26 packages are available, no updates will be listed or applied. You could switch to the Perl 5.26 stream at any time.



### Note

Each stream receives updates independently.

## Module Naming Specification

The full module naming specification takes the form **name:stream:version:context:architecture/profile**. Shorter specifications can be used when the default values are acceptable.

- **name**
- **name:stream**
- **name:stream:version**

Any of the above specifications with **::architecture** appended.

- **name:stream:version:context:architecture**

Any of the above specifications with **/profile** appended.

## Managing Module Streams

RHEL 8 includes the **yum module** command to manage module streams.

For example, to install a new module stream you need to first determine the available streams for that module in your installation with the **yum module list** command.

```
[root@demo ~]# yum module list container-tools
...output omitted...
Name           Stream      Profiles
container-tools    1.0        common [d]  Common tools and dependencies
for container runtimes
container-tools    rhe18 [d][e]  common [d]  Common tools and dependencies
for container runtimes
```

To install a specific stream for a module use the **yum module install** command. For example, to install the **rhe18** stream for the *container-tools* module use the following command.

```
[root@serverc ~]# yum module install container-tools:rhe18
```

## Viewing Module Streams

You can view which module streams are available in the repositories in your content views. To see a list of available modules and module streams, navigate to **Content** → **Content Views** and click the **Module Streams** tab. You can use the **Filter** field to filter the list.

## Applying Module Stream Errata

Use the **yum update** command to update a RHEL 8 host with the latest errata available for a module stream. The following example applies the errata available for the **rhe18** stream of the *container-tools* module.

```
[root@demo ~]# yum update @container-tools:rhe18
```

You can find the module stream for an errata with the **hammer erratum info** command. The following example displays information for the erratum with the **RHSA-2019:45776** ID.

```
[root@demo ~]# hammer erratum info --id RHSA-2019:45776
```



### References

For more information, refer to the *Introduction to Modules* chapter in the *Installing, managing, and removing user space components Guide* at  
[https://access.redhat.com/documentation/en-us/red\\_hat\\_enterprise\\_linux/8/html-single/installing\\_managing\\_and\\_removing\\_user\\_space\\_components](https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/installing_managing_and_removing_user_space_components)

## ► Guided Exercise

# Managing Module Streams for RHEL 8 Hosts

In this exercise, you will inspect the Satellite Server library's available module streams, and apply specific module streams to a host.

## Outcomes

You should be able to:

- Determine which module streams are available in a content view.
- Install a specific stream for a module on a host.

## Before You Begin

Log in as the **student** user on **workstation** using **student** as the password.

Run the **lab software-modules start** command. This command determines if the **satellite** host is reachable on the network and verifies that Red Hat Satellite and the **Base** content view are available, and the **serverc** host is registered.

```
[student@workstation ~]$ lab software-modules start
```

- ▶ 1. Log in to the Satellite web UI as **admin** using **redhat** as the password. Select the **Operations** organization and the **Any Location** location.
  - 1.1. Navigate to <https://satellite.lab.example.com> and log in as **admin** using **redhat** as the password.
  - 1.2. Select the **Operations** organization and the **Any Location** location.
- ▶ 2. Verify which module streams are available for the **Base** content view.
  - 2.1. Click **Content** → **Content Views** and then click **Base**.
  - 2.2. Click the latest version of the **Base** content view, and then click the **Module Streams** tab.
  - 2.3. Search for the **container-tools** module stream to verify the availability of the **rhel8** stream.
- ▶ 3. Open a terminal on **serverc** and install the **rhel8** stream for the *container-tools* module.
  - 3.1. On **workstation**, use **ssh** to log in to **serverc** as **student**. Use **sudo -i** to switch to **root**.

```
[student@workstation ~]$ ssh student@serverc  
[student@serverc ~]$ sudo -i  
[sudo] password for student: student  
[root@serverc ~]#
```

3.2. List the streams available for the *container-tools* module.

```
[root@serverc ~]# yum module list container-tools  
...output omitted...  
Name Stream Profiles  
container-tools 1.0 common [d] Common tools and dependencies  
for container runtimes  
container-tools rhel8 [d][e] common [d] Common tools and dependencies  
for container runtimes
```

3.3. Install the **rhel8** stream for the *container-tools* module. Enter **y** when prompted during the installation.

```
[root@serverc ~]# yum module install container-tools:rhel8  
...output omitted...  
Is this ok [y/N]: y  
...output omitted...
```

3.4. Log off from **serverc**.

## Finish

On the **workstation** machine, use the **lab software-modules finish** command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab software-modules finish
```

This concludes the guided exercise.

## ► Lab

# Deploying Software to Hosts

### Performance Checklist

In this lab, you will work with hosts in two separate life-cycle environments for testing and production, and deploy different versions of the same software to hosts in the two environments.

### Outcomes

You should be able to:

- Register a host to a specific life-cycle environment and content view.
- Deploy software from different life-cycle environments and content views.
- Create a composite content view.
- Configure a content view filter.
- Create an errata content view filter.
- Apply errata to managed hosts.
- Verify the module streams available in a content view.
- Install a specific stream for a module in a host.

### Before You Begin

Log in as the **student** user on **workstation** using **student** as the password.

Run the **lab software-review start** command. This command determines if the **satellite**, **serverb**, **serverc**, and **serverd** hosts are reachable on the network. This command verifies that the Red Hat Satellite Server is running, and the **serverb** host is registered, and that the **Base** content view and the **QA** life-cycle environment are available. This exercise requires the **Finance** organization, the **Base** content view, and the **Build**, **Test**, and **Deploy** life-cycle environments, which you created in the preceding exercises.

```
[student@workstation ~]$ lab software-review start
```

1. Log in to the Satellite web UI as **admin** using **redhat** as the password. Select the **Finance** organization and the **Any Location** location.
2. Ensure that **Version 1.0** of the **Base** content view is available.
3. Ensure that the content view and life-cycle environment for **serverb** are set to **Base** and **Build** respectively.
4. Confirm that the **ant** package is not available on **serverb**.
5. Modify the **Base** content view to include the following repositories. When done, create a new version of the content view, and promote this version to the **Build** life-cycle environment.
  - Red Hat Enterprise Linux 8 for x86\_64 - AppStream (RPMs)
  - Red Hat Satellite Tools 6.6 for RHEL 8 x86\_64 (RPMs)
6. Install the **ant** package on **serverb**.

7. Enable the **Red Hat Ansible Engine 2.8 for RHEL 8 x86\_64 (RPMs)** repository from the local CDN.
8. Use the Satellite Server web UI to synchronize the **Red Hat Ansible Engine 2.8 for RHEL 8 x86\_64 (RPMs)** repository.
9. Create a new content view named **Ansible**, and add the **Red Hat Ansible Engine 2.8 for RHEL 8 x86\_64 RPMs x86\_64** repository to this content view.
10. Create a filter, named **Ansible-older-2.8.4**, for the **Ansible** content view. Configure the filter to exclude *ansible* packages prior to version **2.8.4**. Use the **noarch** architecture.
11. Create a version for the **Ansible** content view, and promote this version to the **Build** life-cycle environment.
12. Create a new composite content view named **BaseAnsibleComposite**, which includes both the **Base** and **Ansible** content views. Set this content view to auto publish.
13. Publish a new version of the **BaseAnsibleComposite** composite view, and promote this version to the **Test** life-cycle environment.
14. Register the **serverd** host in the **Finance** organization and the **Test/**  
**BaseAnsibleComposite** environment.
15. Add the available subscription to **serverd**.
16. Verify which *ansible* package versions are available on **serverd**.
17. Create a filter for the **Base** content view to exclude all non-security errata starting from August 1st, 2019. Use this filter to create a new version of the **Base** content view, and promote this version to the **Test** life-cycle environment.
18. Verify which module streams are available for the **Base** content view.
19. Open a terminal on **serverd**, and install the **rhel8** stream for the *container-tools* module.
20. Log out from **serverd**.

## Evaluation

As the **student** user on the **workstation** machine, use the **lab software-review grade** command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab software-review grade
```

## Finish

As the **student** user on the **workstation** machine, use the **lab software-review finish** command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab software-review finish
```

This concludes the lab.

## ► Solution

# Deploying Software to Hosts

### Performance Checklist

In this lab, you will work with hosts in two separate life-cycle environments for testing and production, and deploy different versions of the same software to hosts in the two environments.

### Outcomes

You should be able to:

- Register a host to a specific life-cycle environment and content view.
- Deploy software from different life-cycle environments and content views.
- Create a composite content view.
- Configure a content view filter.
- Create an errata content view filter.
- Apply errata to managed hosts.
- Verify the module streams available in a content view.
- Install a specific stream for a module in a host.

### Before You Begin

Log in as the **student** user on **workstation** using **student** as the password.

Run the **lab software-review start** command. This command determines if the **satellite**, **serverb**, **serverc**, and **serverd** hosts are reachable on the network. This command verifies that the Red Hat Satellite Server is running, and the **serverb** host is registered, and that the **Base** content view and the **QA** life-cycle environment are available. This exercise requires the **Finance** organization, the **Base** content view, and the **Build**, **Test**, and **Deploy** life-cycle environments, which you created in the preceding exercises.

```
[student@workstation ~]$ lab software-review start
```

1. Log in to the Satellite web UI as **admin** using **redhat** as the password. Select the **Finance** organization and the **Any Location** location.
  - 1.1. Navigate to <https://satellite.lab.example.com> and log in as **admin** using **redhat** as the password.
  - 1.2. Select the **Finance** organization and the **Any Location** location.
2. Ensure that **Version 1.0** of the **Base** content view is available.
  - 2.1. Click **Content → Content Views** and then click **Base**.
  - 2.2. Confirm that **Version 1.0** of the **Base** content view displays on the **Versions** page, and is available to the **Build** life-cycle environment.
3. Ensure that the content view and life-cycle environment for **serverb** are set to **Base** and **Build** respectively.

- 3.1. Click **Hosts** → **Content Hosts**, and then click **serverb.lab.example.com**.
- 3.2. On the **Details** tab of **serverb.lab.example.com**, click the pencil icon for the **Content View** field under **Content Host Content** to enable the menu for that field. Select **Base** from the menu, and click **Save**.
- 3.3. Select the check box for the **Build** life-cycle environment.
4. Confirm that the *ant* package is not available on **serverb**.
  - 4.1. Use **ssh** to log in to **serverb** as **student**. Use **sudo -i** to switch to **root**.

```
[student@workstation ~]$ ssh student@serverb
[student@serverb ~]$ sudo -i
[sudo] password for student: student
[root@serverb ~]#
```

- 4.2. Confirm that the *ant* package is not available on **serverb**.

```
[root@serverb ~]# yum list ant
Updating Subscription Management repositories.
Error: No matching Packages to list
```

5. Modify the **Base** content view to include the following repositories. When done, create a new version of the content view, and promote this version to the **Build** life-cycle environment.
  - Red Hat Enterprise Linux 8 for x86\_64 - AppStream (RPMs)
  - Red Hat Satellite Tools 6.6 for RHEL 8 x86\_64 (RPMs)
  - 5.1. Back into the Satellite web UI, click **Content** → **Content Views**, and click **Base**.
  - 5.2. Click **Yum Content** → **Repositories**. In the **Repository Selection** section, click the **Add** tab to view available repositories.
  - 5.3. Select the check boxes for the **Red Hat Enterprise Linux 8 for x86\_64 - AppStream RPMs x86\_64 8.1** and **Red Hat Satellite Tools 6.6 for RHEL 8 x86\_64 RPMs x86\_64 8** repositories, and click **Add Repositories** to add the selected repositories to the content view.
  - 5.4. Click **Publish New Version** to display the **Publish** page. The **Version** is now 2.0.
  - 5.5. Click **Save** to publish the new version.

The publication process takes some time to complete. You can monitor the publication progress in the **Status** column.
  - 5.6. In the **Actions** column, click **Promote** for **Version 2.0** to display the **Promotion** page.
  - 5.7. Select the **Build** environment.
  - 5.8. Click **Promote Version** to promote the new content view version.

The promotion process takes some time to complete. You can monitor the promotion progress in the **Status** column.
6. Install the *ant* package on **serverb**.
  - 6.1. In the **serverb** terminal, install the *ant* package on **serverb**.

```
[root@serverb ~]# yum install ant
...output omitted...
Complete!
```

7. Enable the **Red Hat Ansible Engine 2.8 for RHEL 8 x86\_64 (RPMs)** repository from the local CDN.
  - 7.1. Click **Content** → **Red Hat Repositories**.
  - 7.2. Enter **ansible-2.8-for-rhel-8-x86\_64-rpms** in the **Search** field, and click **Search**.
  - 7.3. Expand **Red Hat Ansible Engine 2.8 for RHEL 8 x86\_64 (RPMs)**. Click the + sign next to **x86\_64** to enable this repository.
8. Use the Satellite Server web UI to synchronize the **Red Hat Ansible Engine 2.8 for RHEL 8 x86\_64 (RPMs)** repository.
  - 8.1. Click **Content** → **Products** and then click **Red Hat Ansible Engine**.
  - 8.2. In the **Repositories** tab, click the check box next to the **Red Hat Ansible Engine 2.8 for RHEL 8 x86\_64 RPMs x86\_64** repository, and click **Sync Now** to begin synchronizing the repository.  
You can view the progress of the synchronization on the **Tasks** tab.
9. Create a new content view named **Ansible**, and add the **Red Hat Ansible Engine 2.8 for RHEL 8 x86\_64 RPMs x86\_64** repository to this content view.
  - 9.1. Click **Content** → **Content Views** and then click **Create New View**.
  - 9.2. On the **Create Content View** page, enter **Ansible** in the **Name** field. The **Label** field is automatically populated with the **Name** field value.
  - 9.3. Click **Save** to display the **Repository Selection** page.
  - 9.4. Select the check box for the **Red Hat Ansible Engine 2.8 for RHEL 8 x86\_64 RPMs x86\_64** repository, and click **Add Repositories**.
10. Create a filter, named **Ansible-older-2.8.4**, for the **Ansible** content view. Configure the filter to exclude *ansible* packages prior to version **2.8.4**. Use the **noarch** architecture.
  - 10.1. Click **Yum Content** → **Filters** for the **Ansible** content view, and click **New Filter**.
  - 10.2. Complete the **Add New Yum Filter** page with the following details:

| Field          | Value               |
|----------------|---------------------|
| Name           | Ansible-older-2.8.4 |
| Content Type   | Package             |
| Inclusion Type | Exclude             |

Ensure all the details are correct and click **Save**.

- 10.3. Click **Add Rule** to start adding a new rule to the filter.

10.4. Create a new rule with the following details:

| Field           | Value     |
|-----------------|-----------|
| RPM Name        | ansible   |
| Architecture    | noarch    |
| Version         | Less Than |
| Maximum Version | 2.8.4     |

Ensure all the details are correct and click **Save**.

11. Create a version for the **Ansible** content view, and promote this version to the **Build** life-cycle environment.
  - 11.1. Click **Content** → **Content Views** and then click **Ansible**.
  - 11.2. Click **Publish New Version** to display the **Publish** page. Click **Save**.  
The publication process takes a short time to complete, after which the **Ansible** content view is available in the **Library** environment.
  - 11.3. In the **Actions** column, click **Promote**.
  - 11.4. Select the check box for **Build**.
  - 11.5. Click **Promote Version** to start the promotion and display the **Versions** page. The promotion process takes a short time to complete.
12. Create a new composite content view named **BaseAnsibleComposite**, which includes both the **Base** and **Ansible** content views. Set this content view to auto publish.
  - 12.1. Click **Content** → **Content Views**, and click **Create New View**.
  - 12.2. In the **Create Content View** page, enter **BaseAnsibleComposite** in the **Name** field. The **Label** field is automatically populated with the **Name** field value.
  - 12.3. Select the **Composite View** and **Auto Publish** check boxes, and click **Save**.
  - 12.4. Click the **Add** tab, and select the **Ansible** and **Base** content views, and click **Add Content Views**.
  - 12.5. On the **List/Remove** tab, confirm that both content views are listed.
13. Publish a new version of the **BaseAnsibleComposite** composite view, and promote this version to the **Test** life-cycle environment.
  - 13.1. Click **Publish New Version**.
  - 13.2. Click **Save** to start the publication process, which takes a short time to complete. The **BaseAnsibleComposite** composite view is now published to the **Library** environment.
  - 13.3. In the **Actions** column, click **Promote** to open the **Promotion** page.
  - 13.4. Select **Test**.

13.5. Click **Promote Version**. If you are prompted to force the promotion of the content view, confirm the force promotion to continue. The promotion takes a short time to complete.

14. Register the **serverd** host in the **Finance** organization and the **Test/BaseAnsibleComposite** environment.

14.1. On **workstation**, use **ssh** to log in to **serverd** as **student**. Use **sudo -i** to switch to **root**.

```
[student@workstation ~]$ ssh student@serverd
[student@serverd ~]$ sudo -i
[sudo] password for student: student
[root@serverd ~]#
```

14.2. Use **subscription-manager clean** to clear existing registration data from **serverd**. Ignore warnings related to the activation of the **yum** plug-ins.

```
[root@serverd ~]# subscription-manager clean
All local data removed
```

**WARNING**

The yum/dnf plugins: /etc/dnf/plugins/subscription-manager.conf, /etc/dnf/plugins/product-id.conf were automatically enabled for the benefit of Red Hat Subscription Management. If not desired, use "subscription-manager config --rhsm.auto\_enable\_yum\_plugins=0" to block this behavior.

14.3. Install the *katello-ca-consumer-latest* package from the Red Hat Satellite server ([satellite.lab.example.com](http://satellite.lab.example.com)). Use the <http://satellite.lab.example.com/pub/katello-ca-consumer-latest.noarch.rpm> URL to install *katello-ca-consumer-latest*.

```
[root@serverd ~]# yum localinstall \
http://satellite.lab.example.com/pub/katello-ca-consumer-latest.noarch.rpm
...output omitted...
Is this ok [y/N]: y
...output omitted...
Complete!
```

14.4. Use **subscription-manager register** to register **serverd** to the Red Hat Satellite Server. Use the **--org** option to specify the **Finance** organization. Use the **--environment** option to specify the **Test/BaseAnsibleComposite** environment. When prompted, enter **admin** as the user name and **redhat** as the password.

```
[root@serverd ~]# subscription-manager register --org Finance \
--environment Test/BaseAnsibleComposite
Registering to: satellite.lab.example.com:443/rhsm
Username: admin
Password: redhat
The system has been registered with ID: 20fa8362-6e54-0b2a-221aad4923b6
The registered system name is: serverd.lab.example.com
```

The registration process takes a few seconds to complete.

15. Add the available subscription to **serverd**.

15.1. Click **Hosts** → **Content Hosts**, and click the fully qualified domain name (FQDN) of **serverd** host.

15.2. Select **8.1** on the **Release Version** drop-down menu. When done, click **Save**.

15.3. On the **serverd.lab.example.com** page, click **Subscriptions** → **Subscriptions**.

15.4. Click the **Add** tab below **Subscription Details**.

15.5. Select the check box for **Red Hat Enterprise Linux Server, Standard (Physical or Virtual Nodes)**, and **Red Hat Satellite Infrastructure Subscription**, and click **Add Selected**.

15.6. Click the **List/Remove** tab below **Subscription Details**, and confirm that the added subscriptions display.

16. Verify which *ansible* package versions are available on **serverd**.

16.1. Return to the terminal of **serverd**, and verify the available repositories.

```
[root@serverd ~]# subscription-manager repos --list
+-----+
 Available Repositories in /etc/yum.repos.d/redhat.repo
+-----+
Repo ID:    rhel-8-for-x86_64-baseos-rpms
Repo Name:  Red Hat Enterprise Linux 8 for x86_64 - BaseOS (RPMS)
Repo URL:   https://satellite.lab.example.com/pulp/repos/Finance/Test/BaseAnsible
Composite/content/dist/rhel8/8.1/x86_64/baseos/os
Enabled:    1

Repo ID:    rhel-8-for-x86_64-appstream-rpms
Repo Name:  Red Hat Enterprise Linux 8 for x86_64 - AppStream (RPMS)
Repo URL:   https://satellite.lab.example.com/pulp/repos/Finance/Test/BaseAnsible
Composite/content/dist/rhel8/8.1/x86_64/appstream/os
Enabled:    1

Repo ID:    ansible-2.8-for-rhel-8-x86_64-rpms
Repo Name:  Red Hat Ansible Engine 2.8 for RHEL 8 x86_64 (RPMS)
Repo URL:   https://satellite.lab.example.com/pulp/repos/Finance/Test/BaseAnsible
Composite/content/dist/layered/rhel8/x86_64/ansible/2.8/os
Enabled:    0

Repo ID:    satellite-tools-6.6-for-rhel-8-x86_64-rpms
Repo Name:  Red Hat Satellite Tools 6.6 for RHEL 8 x86_64 (RPMS)
Repo URL:   https://satellite.lab.example.com/pulp/repos/Finance/Test/BaseAnsible
Composite/content/dist/layered/rhel8/x86_64/sat-tools/6.6/os
Enabled:    0
```

16.2. Enable the **ansible-2.8-for-rhel-8-x86\_64-rpms** repository.

```
[root@serverd ~]# subscription-manager repos \
--enable ansible-2.8-for-rhel-8-x86_64-rpms
Repository 'ansible-2.8-for-rhel-8-x86_64-rpms' is enabled for this system.
```

16.3. Verify which versions of the *ansible* package are available.

```
[root@serverd ~]# yum search ansible --showduplicates
...output omitted...
=====
Name Exactly Matched: ansible =====
ansible-2.8.5-2.el8ae.noarch : SSH-based configuration management, deployment, ...
ansible-2.8.6-1.el8ae.noarch : SSH-based configuration management, deployment, ...
ansible-2.8.4-1.el8ae.noarch : SSH-based configuration management, deployment, ...
ansible-2.8.7-1.el8ae.noarch : SSH-based configuration management, deployment, ...
```

16.4. Log off from **serverd**.

17. Create a filter for the **Base** content view to exclude all non-security errata starting from August 1st, 2019. Use this filter to create a new version of the **Base** content view, and promote this version to the **Test** life-cycle environment.

17.1. Click **Content** → **Content Views**, and click **Base**.

17.2. Click **Yum Content** → **Filters** for the content view, and click **New Filter**.

17.3. Complete the **Add New Yum Filter** page with the following details:

| Field          | Value                      |
|----------------|----------------------------|
| Name           | Non-security Errata Filter |
| Content Type   | Erratum - Date and Type    |
| Inclusion Type | Exclude                    |

Ensure all the details are correct and click **Save**.

17.4. Clear the **Security** check box in the **Errata Type** section. Select August 1st, 2019 in the **Start Date** field, and click **Save**.

17.5. Click **Publish New version** to publish a new version of the **Base** content view. Click **Save**. Verify that the new version has fewer packages than the previous content view version.

17.6. Click **Promote** for the version you just published, select the check box for the **Test** life-cycle environment, and click **Promote Version**. If you are prompted to force the promotion of the content view, click **Promote**.

18. Verify which module streams are available for the **Base** content view.

18.1. From the menu, navigate to **Content** → **Content Views**, and click **Base**.

18.2. Click into the latest version for the **Base** content view, and go to the **Module Streams** tab.

18.3. Search for the **container-tools** module stream to verify it is available in the **rhel18** stream.

19. Open a terminal on **serverd**, and install the **rhel18** stream for the *container-tools* module.

19.1. On **workstation**, use **ssh** to log in to **serverd** as **student**. Use **sudo -i** to switch to **root**.

```
[student@workstation ~]$ ssh student@serverd  
[student@serverd ~]$ sudo -i  
[sudo] password for student: student  
[root@serverd ~]#
```

19.2. List the streams available for the *container-tools* module.

```
[root@serverd ~]# yum module list container-tools  
...output omitted...  
Name Stream Profiles  
container-tools 1.0 common [d] Common tools and dependencies  
for container runtimes  
container-tools rhel8 [d][e] common [d] Common tools and dependencies  
for container runtimes
```

19.3. Install the **rhel8** stream for the *container-tools* module.

```
[root@serverd ~]# yum module install container-tools:rhel8  
...output omitted...  
Is this ok [y/N]: y  
...output omitted...  
Is this ok [y/N]: y  
...output omitted...  
Is this ok [y/N]: y  
...output omitted...
```

20. Log out from **serverd**.

This concludes the lab.

## Evaluation

As the **student** user on the **workstation** machine, use the **lab software-review grade** command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab software-review grade
```

## Finish

As the **student** user on the **workstation** machine, use the **lab software-review finish** command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab software-review finish
```

# Summary

---

In this chapter, you learned:

- Content views define which repositories a particular life-cycle environment uses.
- You can use content view filters to limit the availability of software packages or errata.
- Composite content views are content views made up of other content views.
- Satellite supports module management on RHEL 8 hosts.

## Chapter 5

# Deploying Custom Software

### Goal

Create, manage, and deploy custom software products and repositories.

### Objectives

- Create products and repositories for non-Red Hat content in Red Hat Satellite.
- Use the repository discovery feature to search URLs for multiple repositories and use them to create custom products and repositories.
- Update custom products and repositories and use content views to make them available to hosts.

### Sections

- Creating Custom Products and Repositories (and Guided Exercise)
- Creating Products Using Repository Discovery (and Guided Exercise)
- Administering Custom Products and Repositories (and Guided Exercise)

### Lab

Deploying Custom Software

# Creating Custom Products and Repositories

---

## Objectives

After completing this section, you should be able to create products and repositories for non-Red Hat content in Red Hat Satellite.

## Describing Custom Products and Repositories

When you add Red Hat content to Red Hat Satellite Server, it automatically creates Red Hat repositories and their parent Red Hat products. Satellite supports storing and distributing non-Red Hat content, such as custom packages, or files. Satellite can manage repositories that contain the following content types:

- RPM packages
- Kickstart trees
- Puppet modules
- Files
- ISO and KVM images
- Containers
- OSTrees

You can secure repositories containing RPM packages, files, and Puppet modules with GPG keys. The files type supports provisioning configuration files created from a template without needing to deploy an RPM package. OSTrees is a deprecated Atomic Server technology.

To host custom and third-party content on Satellite Server, you must create new repositories and products. Products are collections of repositories grouped to suit software relationships. For example, you can use products to group software repositories from different software vendors.

As with Red Hat repositories, custom and third-party products and repositories must also be created and maintained in an organizational context. Products and repositories created within an organizational context are visible only to that organization.

There are two methods for creating repositories in Satellite Server for non-Red Hat content. This section discusses the first method; manually creating custom products and their associated repositories. The second method, repository discovery, is discussed later in this chapter.

## Creating Custom Products

Satellite Server contains custom and third-party software packages in repositories, and those repositories must be associated with a product. Therefore, to use Satellite Server to host non-Red Hat content, you must first create the products. Creating products in Satellite Server requires administrative privileges.

### To Create a Custom Product:

- Choose the required organization and location from the main menu.
- Navigate to **Content → Products** and click **Create Product**.

- Set meaningful values for the **Name** and **Description** fields. The value of **Name** acts as the identifier of the product. The **Description** field provides a human-friendly description of the repository names that the product includes. The **Label** field is automatically populated using the value of the **Name** field.
- Choose the required GPG key from the **GPG Key** list. You need to have already imported this key into Satellite Server. Satellite uses that GPG key to validate the origin of the packages in the repository of the product, and when installed on a content host.

## Creating Custom Repositories

After you create a product, you can create the repositories within the product that you intend to group together. Creating product repositories in Satellite Server requires administrative privileges.

### To Create a Custom Repository:

- Navigate to **Content** → **Products** and click the product name to which to add a repository.
- On the **Repositories** tab, click **New Repository**.
- On the **Repositories** tab, set meaningful values for the **Name** and **Description** fields. The **Label** field is auto-populated from the **Name** field. Use the **Description** field to specify a human-friendly version of the repository name.
- For example, if your custom repository is to be a software package repository, select **yum** from the **Type** field. Selecting the **yum** repository type enables additional fields under the **Sync Settings** section of the page. The parameters in this section control the custom repository synchronization with the upstream repository.

| Option                          | Description                                                                                                                                                                                                                                                                                                                                                                                                        |
|---------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Restrict to Architecture</b> | Enables the repository for a particular architecture. The <b>Default</b> value enables the repository for all architectures.                                                                                                                                                                                                                                                                                       |
| <b>Upstream URL</b>             | Sets the upstream repository URL for this repository to use as the content source. This field is optional. If the URL is not specified, the new repository acts as a standalone repository without source content synchronization. The empty repository can be populated by manually adding packages.                                                                                                              |
| <b>Upstream Username</b>        | Sets the user name for authentication at the upstream repository. Leave this blank if the upstream repository is not password protected.                                                                                                                                                                                                                                                                           |
| <b>Upstream Password</b>        | Sets the password for authentication at the upstream repository. Leave this blank if the upstream repository is not password protected.                                                                                                                                                                                                                                                                            |
| <b>Download Policy</b>          | Specifies how clients will retrieve packages from the source. The <b>On Demand</b> policy downloads only metadata during synchronization. After synchronization, the packages are downloaded only as clients request them. The <b>Background</b> policy runs a background task after synchronization to download all packages. The <b>Immediate</b> policy downloads metadata and packages during synchronization. |

- To enable the Katello component of Satellite Server to verify that the SSL certificates of the upstream repository are signed by a trusted Certificate Authority (CA), select **Verify SSL** in the **Sync Settings** section.

- To enable removal of packages from the custom repository that are no longer part of the upstream repository, select **Mirror on Sync**.
- To enable publication of the packages in the custom repository via HTTP, select **Publish via HTTP**.

## Adding Packages to Repositories

Packages are typically added to a custom repository during its initial creation but can also be added later. Adding packages to custom product repositories in Satellite Server requires administrative privileges.

### To Add Packages to a Repository:

- Choose the required organization and location from the main menu.
- Navigate to **Content → Products** and click the product name to which to add packages.
- On the **Repositories** tab, click the name of the repository to display the details of the repository.
- In the **Upload Package** section, click **Browse** to select the packages that are locally available, and click **Open** to mark the packages for uploading to the repository.
- Click **Upload** to upload the packages to the repository.
- In the **Content Counts** section, the **Content Type** table displays the number of packages in the repository. To view the packages in the repository, click the number that corresponds to the package count to open the **Packages** page of the repository.

## Removing Packages from Repositories

Removing packages from product repositories requires administrative privileges.

To remove repository packages, navigate to the **Packages** page of the custom product repository, select the check box for the package to remove, and click **Remove Packages**.



### References

For more information, refer to the *Creating a Custom Product* section of the *Importing Custom Content* chapter in the *Red Hat Satellite 6.6 Content Management Guide* at  
[https://access.redhat.com/documentation/en-us/red\\_hat\\_satellite/6.6/html-single/content\\_management\\_guide/importing\\_custom\\_content#Importing\\_Custom\\_Content-Creating\\_a\\_Custom\\_Product](https://access.redhat.com/documentation/en-us/red_hat_satellite/6.6/html-single/content_management_guide/importing_custom_content#Importing_Custom_Content-Creating_a_Custom_Product)

For more information, refer to the *Adding a Custom RPM Repository* section of the *Importing Custom Content* chapter in the *Red Hat Satellite 6.6 Content Management Guide* at  
[https://access.redhat.com/documentation/en-us/red\\_hat\\_satellite/6.6/html-single/content\\_management\\_guide/index#Importing\\_Custom\\_Content-Creating\\_a\\_Custom\\_RPM\\_Repository](https://access.redhat.com/documentation/en-us/red_hat_satellite/6.6/html-single/content_management_guide/index#Importing_Custom_Content-Creating_a_Custom_RPM_Repository)

## ► Guided Exercise

# Creating Custom Products and Repositories

In this exercise, you will create a new product with a custom repository on Red Hat Satellite Server.

### Outcomes

You should be able to:

- Create a custom product.
- Add repositories to the custom product.

### Before You Begin

Log in as the **student** user on the **workstation** VM, using **student** as the password.

Run the **lab custom-create start** command. This command determines if the **satellite** host is reachable on the network and prepares it for this exercise.

```
[student@workstation ~]$ lab custom-create start
```

- ▶ 1. Use the Red Hat Satellite Server web UI to create a custom product named **Custom Software**.
  - 1.1. Log in to the Satellite Server web UI at <https://satellite.lab.example.com> as the **admin** user, using **redhat** as the password.
  - 1.2. Choose the **Operations** organization and the **Any Location** location from the main menu.
  - 1.3. Click **Content → Products** and then click **Create Product** to open the **New Product** page.
  - 1.4. Enter **Custom Software** in the **Name** field. Notice that the **Label** field is automatically populated from the value of **Name**, replacing white space with underscores.
  - 1.5. Enter **Custom Software** in the **Description** field, and click **Save** to create the custom product.  
This displays the **Repositories** tab for your new product. In the following steps, you will use this tab to create a custom repository.
- ▶ 2. Add the **Admin Tools** custom repository to **Custom Software**.
  - 2.1. On the **Repositories** tab of **Custom Software**, click **New Repository** to open the **New Repository** page.

- 2.2. Enter **Admin Tools** in the **Name** field. Notice that the **Label** field gets automatically populated from the value of **Name**, replacing white space with underscores.
  - 2.3. Select **yum** from the **Type** menu.
  - 2.4. Ensure that the **Upstream Username** and the **Upstream Password** fields are empty because the **Admin Tools** custom repository is not intended to be password protected.
  - 2.5. Leave all the other fields with their default values, and click **Save**.
- ▶ 3. On **workstation**, download the following packages to include in the custom repository.

- *hwinfo-21.47-9.el8.x86\_64.rpm*
- *osinfo-db-20181011-8.el8\_0.1.noarch.rpm*

```
[student@workstation ~]$ wget -P ~/Downloads/ \
http://materials.example.com/hwinfo-21.47-9.el8.x86_64.rpm
...output omitted...
[student@workstation ~]$ wget -P ~/Downloads/ \
http://materials.example.com/osinfo-db-20181011-8.el8_0.1.noarch.rpm
...output omitted...
```

- ▶ 4. Return to the Satellite UI and upload the *hwinfo* and *osinfo-db* packages to the **Admin Tools** repository.
- 4.1. On the **Repositories** tab, click the name of the **Admin Tools** repository to display its details.
  - 4.2. On the **Admin Tools** page, in the **Upload Package** section, click **Browse**. Select the *hwinfo* and *osinfo-db* packages from the **/home/student/Downloads** directory, and click **Open**.



#### Note

To select multiple items, press the **Ctrl** key and click the required items.

- 4.3. Click **Upload** to upload the selected packages to the **Admin Tools** repository. Notice that the package count for the repo is now 2.

## Finish

On the **workstation** machine, use the **lab custom-create finish** command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab custom-create finish
```

This concludes the guide exercise.

# Creating Products Using Repository Discovery

---

## Objectives

After completing this section, you should be able to create custom products and repositories using the repository discovery feature to search URLs for multiple repositories.

## Describing Repository Discovery

Third-party software vendors often have multiple software package repositories available for their different software collections. Rather than manually creating each repository on Satellite Server by specifying their individual URLs, you can use *repository discovery* to expedite the process of adding multiple yum repositories from the same source.

When you use the repository discovery feature, you supply a URL to Satellite Server. Satellite Server scans through the contents of this URL to discover all the repositories that are accessible under the directory structure of the provided URL. You use the results to create the desired repositories all at once.

### To Create a New Product Using Repository Discovery:

- In the Satellite web UI, choose the required organization and location from the main menu. Navigate to **Content** → **Products**.
- Click **Repo Discovery** to open the **Discover Repositories** page. Use the following table to determine suitable values:

| Option                 | Description                                                                                                                                                                               |
|------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Repository Type</b> | Specifies the type of the repository. For software package repositories, select <b>Yum Repositories</b> .                                                                                 |
| <b>URL to Discover</b> | Specifies the URL to search for repositories. This URL points to the Red Hat CDN for a connected Satellite Server. For a disconnected Satellite Server, this URL points to the local CDN. |
| <b>Username</b>        | Sets the user name to enable authenticated access to the repository if you intend to protect the repository with a password. Leave it blank to enable anonymous access to the repository. |
| <b>Password</b>        | Sets the password of the user for authentication to the repository. Leave it blank to enable anonymous access to the repository.                                                          |

- Click **Discover** to start the repository discovery based on the specified settings. The discovered repositories display in the **Discovered Repository** section of the page. To create a product with the discovered repositories, select the check boxes for the intended repositories, and click **Create Selected**.
- On the **Create Repositories** page, in the **Product Options** section, select **New Product** in the **Product** field to add the selected repositories to a new product in Satellite Server.

The **Name** field sets the name of the new product. The **Label** field is automatically populated from the **Name** field.

- Select **Serve via HTTP** to publish the content of the repository via HTTP. Select **Verify SSL** to validate the SSL certificates of the external source of content against the trusted Certificate Authorities.
- Click **Run Repository Creation** to create the custom product with the selected repositories.

## Extending Existing Product Repositories Using Repository Discovery

To add or modify selections for an existing product repository navigate to **Content → Products** and select the check box next to the product you wish to modify, then click **Repo Discovery**. Select the desired **Repository Type** from the drop-down menu and enter a URL for the CDN to search in the **URL to Discover** field. Click the **Discover** button. Select the desired repositories from the **Discovered Repository** list then click **Create Selected**.

From the **Create Repositories** page, under **Product Options**, select **Existing Product** from the drop-down menu of the **Product** field to add the selected repositories to an existing product in the Satellite Server. Select the intended product from the drop-down menu of the **Name** field. Set the values of the other fields as appropriate, and click **Run Repository Creation** to add the newly discovered repository to the existing product. All the other fields retain the same significance as previously described.



### References

For more information, refer to the *Adding a Custom RPM Repository* section of the *Importing Custom Content* chapter in the *Red Hat Satellite 6.6 Content Management Guide* at  
[https://access.redhat.com/documentation/en-us/red\\_hat\\_satellite/6.6/html-single/content\\_management\\_guide/importing\\_custom\\_content#Importing\\_Custom\\_Content-Creating\\_a\\_Custom\\_RPM\\_Repository](https://access.redhat.com/documentation/en-us/red_hat_satellite/6.6/html-single/content_management_guide/importing_custom_content#Importing_Custom_Content-Creating_a_Custom_RPM_Repository)

## ► Guided Exercise

# Creating Products Using Repository Discovery

In this exercise, you will create a custom product containing repositories located using repository discovery.

### Outcomes

You should be able to use repository discovery to create a new product and repository on your Satellite Server from an existing Yum repository located in the local CDN.

### Before You Begin

Log in to **workstation** as **student** using **student** as the password.

Run the **lab custom-discovery start** command. This command determines if the **satellite** host is reachable on the network and prepares it for this exercise.

```
[student@workstation ~]$ lab custom-discovery start
```

The **Operations** team have existing Yum repositories that they want to make available through Satellite Server. They want to create a new product for DVDs, and would like to initially add the material from a Yum repository containing RHEL 8.1 for x86\_64 DVD content to this product.

You will use repository discovery to obtain a list of the URLs for the repositories located under <http://content.example.com>. You will create a new product called **DVDS** with repositories for RHEL 8.0 AppStream and BaseOS. The repository should be sourced from the [http://content.example.com/rhs6.6/x86\\_64/cdn](http://content.example.com/rhs6.6/x86_64/cdn) URL.

- ▶ 1. Log in to the Satellite Server web UI, <https://satellite.lab.example.com>, as the **admin** user with **redhat** as the password.
- ▶ 2. Choose **Operations** and **Any Location** from the main menu.
- ▶ 3. Use the repository discovery feature to locate and create repositories for RHEL 8.x AppStream and BaseOS.
  - 3.1. Click **Content → Products** and then click **Repo Discovery**.
  - 3.2. On the **Discover Repositories** page, set the **Repository Type** field to **Yum Repositories**.
  - 3.3. Enter [http://content.example.com/rhs6.6/x86\\_64/cdn](http://content.example.com/rhs6.6/x86_64/cdn) in the **URL to Discover** field.  
Click **Discover**.
  - 3.4. In the **Discovered Repository** results list, select the check box for the following repositories:

- /content/dist/rhel8/8.1/x86\_64/appstream/os/
- /content/dist/rhel8/8.1/x86\_64/baseos/os/

Click **Create Selected**.

- 3.5. You are redirected to the **Create Repositories** page. Enter the following information in the **Product Options** section.

| Option          | Value       |
|-----------------|-------------|
| Product         | New Product |
| Name            | DVDs        |
| Label           | DVDs        |
| Server via HTTP | Check       |
| Verify SSL      | Check       |

Click **Run Repository Creation** to create the repository.

- 4. Synchronize the new repository.

- 4.1. Click **Content** → **Products** and then click the **DVDs** product name.
- 4.2. On the **Repositories** tab, select the check box for the following repositories:

- **content dist rhel8 8.1 x86\_64 appstream os**
- **content dist rhel8 8.1 x86\_64 baseos os**

- 4.3. Click **Sync Now** to synchronize the repositories.

On the **Tasks** tab you can monitor state, results, and other synchronization details. Wait for the synchronization to complete. This may take several minutes.

## Finish

On the **workstation** machine, use the **lab custom-discovery finish** command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab custom-discovery finish
```

This concludes the guided exercise.

# Administering Custom Products and Repositories

---

## Objectives

After completing this section, you should be able to update custom products and repositories and use content views to make them available to hosts.

## Validating RPM Packages with GPG Keys

Red Hat Satellite supports using GPG keys to secure products and repositories. When you create non-Red Hat products and repositories, you can specify the GPG public key associated with the repository's RPM files. If no GPG key is specified, the repository configuration on the managed hosts has the **gpgcheck** option set to disabled.

However, if you specify a GPG public key, the repository configuration on the managed hosts has the **gpgcheck** option set to enabled. Satellite Server makes the GPG public key available to hosts, which then use the key to verify the origin and integrity of packages during package installation. Packages in a repository specified with a GPG key pair must be signed with the correct GPG private key to be able to be loaded to that repository.

If you configure a GPG key pair on a product, all repositories within that product use the same GPG public key. Additionally, a GPG key pair configured on a repository for a product overrides the GPG key pair configured for the product.

## Adding GPG Keys

To associate GPG keys during product or repository creation, you first need to add the GPG key to Satellite Server. A GPG key then appears in the **GPG Key** menu when you create products or repositories. Adding GPG keys require administrative privileges. The Satellite web UI displays these GPG keys as *content credentials*.

### To Add a GPG Key Pair:

- In the Satellite web UI, choose the required organization and location from the main menu.
- Click **Content** → **Content Credentials** to open the **Content Credentials** page.
- Click **Create Content Credentials** to open the **New Content Credential** page.
- Specify a meaningful name for the content credential, and select **GPG Key** as the value of the **Type** field.
- In the **Content Credential Contents** field, paste the content of a GPG key file, and click **Save** to create the content credential.

## Editing GPG Keys

To modify the name of a GPG key or to change its content requires administrative privileges.

### To Edit a GPG Key Pair:

- In the Satellite web UI, choose the required organization and location from the main menu.

- Click **Content** → **Content Credentials** to open the **Content Credentials** page.
- Click the name of the GPG key you want to edit.
  - To edit the name, click the pencil icon next to the name, and enter a new name in the **Name** field.
  - To modify the GPG key content, click the pencil icon next to the **Content** field and replace the content of the GPG key. Alternatively, click **Browse** and select the file containing the new GPG key.
- Click **Save** to update the GPG key.

## Removing GPG Keys

Optionally, you may choose to replace or remove a GPG key pair from a custom product and repository. Removing GPG keys requires administrative privileges.

### To Remove a GPG Key Pair:

- In the Satellite web UI, choose the required organization and location from the main menu.
- Click **Content** → **Content Credentials** to open the **Content Credentials** page.
- Click the name of the GPG key you want to remove.
- Click **Remove Content Credential** and then click **Remove**.

## Digitally Sign the RPM Packages

RPM packages are normally digitally signed so that you can verify that a package actually came from the declared source. This helps to block forged packages from being installed if a Yum repository is compromised in some way. The next few steps detail how to create a signing key.

- Install the supporting RPM packages.

```
[user@demo ~]$ sudo yum install rpm-sign  
...output omitted...
```

- Generate a GPG key for the user account that manages RPMs.



### Important

Generating enough entropy to create the keys can cause **gpg** to hang while waiting for more entropy. You can use the **rngd -r /dev/urandom** command prior to generating the GPG keys to avoid this issue.

You must have a graphical session open to run **gpg --full-generate-key**. It uses a graphical box to accept your input for the passphrase.

```
[user@demo ~]$ sudo rngd -r /dev/urandom  
[user@demo ~]$ gpg --full-generate-key  
gpg (GnuPG) 2.2.9; Copyright (C) 2018 Free Software Foundation, Inc.  
This is free software: you are free to change and redistribute it.
```

```
There is NO WARRANTY, to the extent permitted by law.
```

```
Please select what kind of key you want:
```

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

```
Your selection? Enter
```

```
RSA keys may be between 1024 and 4096 bits long.
```

```
What keysize do you want? (2048) Enter
```

```
Requested keysize is 2048 bits
```

```
Please specify how long the key should be valid.
```

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

```
Key is valid for? (0) Enter
```

```
Key does not expire at all
```

```
Is this correct? (y/N) y
```

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

```
Real name: demouser
```

```
Email address: demouser@demo.lab.example.com
```

```
Comment: Enter
```

```
You selected this USER-ID:
```

```
"demouser <demouser@demo.lab.example.com>"
```

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

```
We need to generate a lot of random bytes. It is a good idea to perform  
some other action (type on the keyboard, move the mouse, utilize the  
disks) during the prime generation; this gives the random number  
generator a better chance to gain enough entropy.
```

```
gpg: key 460C349387230ADD marked as ultimately trusted  
gpg: revocation certificate stored as '/home/student/.gnupg/openpgp-  
revocs.d/068F2E1C7A005074749E12C8460C349387230ADD.rev'  
public and secret key created and signed.
```

```
pub    rsa2048 2019-12-20 [SC]  
      068F2E1C7A005074749E12C8460C349387230ADD  
uid    demouser <demouser@demo.lab.example.com>  
sub    rsa2048 2019-12-20 [E]
```

- Find the username and email address from the output of **gpg --full-generate-key**, or with the **gpg --fingerprint** command, and then export the public key into an ASCII armored file.

```
[user@demo ~]$ gpg --fingerprint  
/home/user/.gnupg/pubring.kbx  
-----  
pub    rsa2048 2019-12-20 [SC]  
      068F 2E1C 7A00 5074 749E  12C8 460C 3493 8723 0ADD  
uid    [ultimate] demouser <demouser@demo.lab.example.com>  
sub    rsa2048 2019-12-20 [E]
```

Export an ASCII armor version of the public key to publish to client machines so they can verify RPM packages signed with the private key

```
[user@demo ~]$ gpg --armor \
--export demouser@demo.lab.example.com > /tmp/DEMO-GPG-KEY
```

- Instead of creating a new GPG keypair, you can import an existing key pair on a file with the **gpg --import** command. You need the passphrase to import that keypair.

```
[user@demo ~]$ gpg --import file.asc
gpg: key 5C0009F4E43EEF8D: "demouser <demouser@demo.lab.example.com>" not changed
gpg: key 5C0009F4E43EEF8D: secret key imported
gpg: Total number processed: 1
gpg:          unchanged: 1
gpg:      secret keys read: 1
gpg:  secret keys imported: 1
```

- Individual users can override select system-wide macros by creating the `~/.rpmmacros` file in their home directory. Adding your key's username and email address to this file allows the **rpm** and **rpmbuild** commands to scan the `~/.rpmmacros` file for the value of the `%_gpg_name` macro and use that key for signing RPM packages. Instead of using the key's username and email address, you can also use the GPG key ID, like **A7A051123**, the GPG pub key ID, like **068F2E1C7A005074749E12C8460C349387230ADD**, or just the email address.

Create or modify `~/.rpmmacros` so that `%_gpg_name` is set to the username, and email address of the key previously created. For example:

```
[user@demo ~]$ echo '%_gpg_name demouser demouser@demo.lab.example.com' >>
~/rpmmacros
```

- You can now sign packages using this new GPG key.

```
[user@demo ~]$ rpmsign --addsign rpm_file_name.rpm
```

- You can check the signature for a package.

```
[user@demo ~]$ rpm --checksig rpm_file_name.rpm
rpm_file_name.rpm: size pgp md5 OK
```

## Synchronizing Custom and Third-party Software Repositories

Similarly to Red Hat content products and repositories, you can manually synchronize or associate an existing sync plan with the products and repositories of custom and third-party software for automated, routine synchronization. If the desired sync plans already exist, you can select them during the creation of custom and third-party products and repositories.

Specifying a URL for the source repository while you are creating a repository is optional. When no source repository is specified, the repository becomes an independent, standalone repository. Even though the controls for synchronization are still available for standalone repositories, any attempt to synchronize fails due to the absence of a synchronization source.

## Managing Subscriptions to non-Red Hat Products and Repositories

When you create a non-Red Hat product, a corresponding subscription automatically appears under **Content → Red Hat Subscriptions**. Because subscriptions control the content that hosts have access to, you must add desired subscriptions to a custom product which is then assigned to a host profile before the host can access its contents.

### Enabling Host Subscription to Custom Repositories

To activate a subscription for a host, you add the subscription to a host profile on Satellite Server on a per-host basis.

#### To Enable Host Subscription to a Custom Repository:

- In the Satellite web UI, choose the required organization and location from the main menu, and then click **Hosts → Content Hosts**, and click the name of the intended host.
- Click the **Subscriptions** tab, then the **Add** tab, and select the check box next to the desired subscription for the host.
- Click **Add Selected** to add the subscription.
- Click the **Product Content** tab, and then click the pencil icon to change the **Enabled by Default** setting for the repositories in the new product.
- Select the required enable setting from the menu and then click **Save**.

### Enabling Activation Key Subscription to Custom Repositories

An alternative to adding subscriptions to host profiles is to add subscriptions to activation keys so that hosts registered with the activation key automatically inherit the subscription.

#### To Enable Activation Key Subscription to a Custom Repository:

- In the Satellite web UI, choose the required organization and location from the main menu, and then click **Content → Activation Keys**.
- Click the name of the activation key that you want to add the subscription to.
- Click the **Subscriptions** tab, and then the **Add** tab.
- Select the check box next to the desired subscription, and then click **Add Selected**.
- On the **Product Content** tab, click the edit icon to change the **Enabled by Default** setting for the repositories in the new product.
- Select the required enable setting from the menu, and then click **Save**.



## References

For more information, refer to the *Creating a Custom Product* section of the *Importing Custom Content* chapter in the *Red Hat Satellite 6.6 Content Management Guide* at

[https://access.redhat.com/documentation/en-us/red\\_hat\\_satellite/6.6/html-single/content\\_management\\_guide/importing\\_custom\\_content#Importing\\_Custom\\_Content-Creating\\_a\\_Custom\\_Product](https://access.redhat.com/documentation/en-us/red_hat_satellite/6.6/html-single/content_management_guide/importing_custom_content#Importing_Custom_Content-Creating_a_Custom_Product)

For more information, refer to the *Adding a Custom RPM Repository* section of the *Importing Custom Content* chapter in the *Red Hat Satellite 6.6 Content Management Guide* at

[https://access.redhat.com/documentation/en-us/red\\_hat\\_satellite/6.6/html-single/content\\_management\\_guide/index#Importing\\_Custom\\_Content-Creating\\_a\\_Custom\\_RPM\\_Repository](https://access.redhat.com/documentation/en-us/red_hat_satellite/6.6/html-single/content_management_guide/index#Importing_Custom_Content-Creating_a_Custom_RPM_Repository)

## ► Guided Exercise

# Administering Custom Products and Repositories

In this exercise, you will create a custom product, add custom repositories, and use content views to apply packages from those repositories to hosts.

## Outcomes

You should be able to:

- Configure a custom product and repository to provide a newly registered host with a public GPG key that validates the origin and content of the packages in the repository.
- Register a host system to a custom product and repository, and have Satellite correctly configure the host system to verify packages with the GPG key.

## Before You Begin

You must finish the previous guided exercise before starting this exercise. Log in to **workstation** as **student** using **student** as the password.

Run the **lab custom-admin start** command. This command determines if the **satellite** host is reachable on the network and prepares it for this exercise.

```
[student@workstation ~]$ lab custom-admin start
```

The Operations team has created a product named **Custom Software**, which contains a repository named **Admin Tools**. They want to make sure that hosts that install packages from this repository are configured to verify the origin and integrity of those packages, using GPG keys.

To verify that the hosts have been configured correctly, you will modify the existing **OperationsServers** activation key so that:

- Hosts registered with this activation key automatically use the **Custom Software** product.
- Hosts are granted access to the **Admin Tools** repository.
- Hosts are configured with the public GPG package key.

You will reregister **servera** with the **OperationsServers** key and make sure that the **osinfo** package from the repository successfully installs.

- ▶ 1. Log in to the Satellite Server web UI at <https://satellite.lab.example.com> as the **admin** user, using **redhat** as the password.
- ▶ 2. Choose the **Operations** organization and **Any Location** location from the main menu.
- ▶ 3. Add the GPG public key.

- 3.1. Open a second tab in your browser. Display the contents of the GPG key by entering <http://materials.example.com/EXAMPLE-RPM-GPG-KEY> into your browser. Copy the contents of the GPG key.
  - 3.2. Click **Content** → **Content Credentials** and then click **Create Content Credential**.
  - 3.3. Enter **Example Software** in the **Name** field.
  - 3.4. Select **GPG Key** from the **Type** list.
  - 3.5. Paste the contents of the GPG key into the **Content Credential Contents** field, and then click **Save**.
- ▶ 4. Associate the GPG key with the repository.
- 4.1. Click **Content** → **Products**.
  - 4.2. Click **Custom Software**, and then click the **Admin Tools** repository.
  - 4.3. Click the edit icon for the **GPG Key** field, select **Example Software** from the list, and then click **Save**.
- ▶ 5. Add a subscription for the **Custom Software** product to the **OperationsServers** activation key and enable access to the **Admin Tools** repository.
- 5.1. Click **Content** → **Activation Keys** and then click the **OperationsServers** activation key.
  - 5.2. Click the **Subscriptions** tab, and then the **Add** tab.
  - 5.3. Select the check box next to the **Custom Software** subscription, and then click **Add Selected**.
  - 5.4. Click the **Repository Sets** tab.
  - 5.5. Select the check box for the **Admin Tools** repository. If the status of the **Admin Tools** repository is **Disabled**, select **Override to Enable** from the **Select Action** list.
- ▶ 6. Publish the **Admin Tools** repository and promote it to the **Development** life-cycle environment.
- 6.1. Click **Content** → **Content Views** and click the **Base** content view.
  - 6.2. Select **Repositories** from the **Yum Content** list, and then click the **Add** tab.
  - 6.3. Select the check box for **Admin Tools** repository, and then click **Add Repositories**.
  - 6.4. Click **Publish New Version**.
  - 6.5. Enter **Add Admin Tools Repository** in the **Description** field. Note the version number assigned to this publication.  
Click **Save**. On the **Versions** tab, monitor the progress in the **Status** column. Wait for the publication to complete.
  - 6.6. For the version that you just published, click **Promote** and select the **Development** life-cycle environment.

Enter **Admin Tools Repository** in the **Description** field, and then click **Promote Version**.

- 7. Prepare the host for registration to Satellite Server.

7.1. From **workstation**, **ssh** to **servera** as **student**. Use **sudo -i** to switch to **root**.

```
[student@workstation ~]$ ssh student@servera
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

7.2. Clean up from previous exercises by cleaning up the Yum repository and unregistering **servera** from Satellite Server:

```
[root@servera ~]# yum clean all
[root@servera ~]# subscription-manager unregister
```

7.3. Install a copy of the CA Certificate from Satellite Server:

```
[root@servera ~]# yum localinstall \
http://satellite.lab.example.com/pub/katello-ca-consumer-latest.noarch.rpm
```

- 8. Register the client to Satellite Server using the activation key.

8.1. Register the system to the **Operations** organization. Reference the activation key with the **--activationkey** option.

```
[root@servera ~]# subscription-manager register --org='Operations' \
--activationkey='OperationsServers'
The system has been registered with ID: 869836db-9a1e-4c01-8c14-6e6bad6e8c50
The registered system name is: servera.lab.example.com
Installed Product Current Status:
Product Name: Red Hat Enterprise Linux for x86_64
Status: Subscribed
```



**Note**

Depending on the state of your **servera** configuration it may be necessary to remove it from Satellite prior to registering. Click **Hosts** → **Content Hosts** and select the check box next to **servera.lab.example.com**. Choose **Remove Hosts** from the **Select Action** list.

8.2. Install **Katello Agent** on the client system.

```
[root@servera ~]# yum install katello-agent
```

- 9. Install the **osinfo** package. You should see output indicating the installation of the GPG key. The output shows that the package is signed with the GPG private key. The command also triggers the installation of the GPG public key.

```
[root@servera ~]# yum install osinfo
...output omitted...
Downloading packages:
warning: /var/cache/yum/x86_64/7Server/Operations_Custom_Software_Admin_Tools/
packages/osinfo-1.0-1.el7.x86_64.rpm: Header V4 RSA/SHA1 Signature, key ID
cf21d1a4: NOKEY
Public key for osinfo-1.0-1.el7.x86_64.rpm is not installed
osinfo-1.0-1.el7.x86_64.rpm   | 2.8 kB  00:00:00
Retrieving key from https://satellite.lab.example.com/katello/api/repositories/16/
gpg_key_content
Importing GPG key 0xCF21D1A4:
Userid      : "RPM Build <rpmbuild@example.com>"
Fingerprint: 6f21 1715 5720 99cc 44d0 acc7 bdca b663 cf21 d1a4
From        : https://satellite.lab.example.com/katello/api/repositories/16/
gpg_key_content
...output omitted...

Installed:
  osinfo.x86_64 0:1.0-1.el7

Complete!
```

- 10. Log off from **servera** and return to **workstation** as **student** user.

## Finish

On the **workstation** VM, run the **lab custom-admin finish** command to complete this exercise.

```
[student@workstation ~]$ lab custom-admin finish
```

This concludes the guided exercise.

## ► Lab

# Deploying Custom Software

### Performance Checklist

In this lab, you will make a custom software package available to hosts through a product and repository on Satellite Server.

### Outcomes

You should be able to create a new product and repository on Satellite Server for hosting custom software packages and register a content host using an activation key, and then install custom software on that content host.

### Before You Begin

Log in to **workstation** as **student** using **student** as the password.

Run the **lab custom-review start** command. This command determines if the **satellite** and **serverb** machines are reachable on the network, and verifies that Red Hat Satellite is available.

```
[student@workstation ~]$ lab custom-review start
```

The Finance team would like to begin using Satellite Server to deploy their custom software packages. They would like to make the signing GPG public key available so that hosts can verify the origin and integrity of these packages during installation. For the implementation, they have initially provided you with an RPM, [http://materials.example.com/bkp-1.0-1.el7.x86\\_64.rpm](http://materials.example.com/bkp-1.0-1.el7.x86_64.rpm), and the GPG key, <http://materials.example.com/EXAMPLE-RPM-GPG-KEY>.

The Finance team currently registers their hosts using the **FinanceServers** activation key. They would like hosts to have access to their custom software packages upon registration.

You need to add the GPG key named **Example Software** to Satellite Server, then create a new product named **Custom Software** and a repository named **Admin Tools** to contain the provided RPM. You need to add the product subscription to the **FinanceKey** activation key.

Verify your implementation by registering **serverb.lab.example.com** to the **Finance** organization using the **FinanceKey** activation key. Finally, install the *bkp* package on **serverb.lab.example.com**.

1. Add the GPG public key.
2. Create the **Custom Software** product.
3. Create the **Admin Tools** repository.
4. Add the *bkp-1.0-1.el7.x86\_64.rpm* RPM package to the **Admin Tools** custom repository.
5. Add a subscription for the **Custom Software** product to the **FinanceKey** activation key and enable access to the **Admin Tools** repository.

6. Ensure that the following repositories are included in the **Base** content view:
  - Red Hat Enterprise Linux 8 for x86\_64 - BaseOS RPMs x86\_64 8
  - Red Hat Enterprise Linux 8 for x86\_64 - AppStream RPMs x86\_64 8
  - Red Hat Satellite Tools 6.6 for RHEL 8 x86\_64 RPMs x86\_64 8
  - Admin Tools
- Publish and promote the repositories to the **Build** life-cycle environment.
7. Prepare **serverb.lab.example.com** for registration to Satellite Server.
8. Register **serverb.lab.example.com** using the **FinanceKey** activation key and install the *katello-agent* package.
9. Install the *bkp* package. The output shows that the package is signed with the GPG private key. The output should also indicate the installation of the GPG public key.
10. Return to **workstation** as **student** user.

## Evaluation

As the **student** user on the **workstation** machine, use the **lab** command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab custom-review grade
```

## Finish

As the **student** user on the **workstation** machine, use the **lab** command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab custom-review finish
```

This concludes the lab.

## ► Solution

# Deploying Custom Software

### Performance Checklist

In this lab, you will make a custom software package available to hosts through a product and repository on Satellite Server.

### Outcomes

You should be able to create a new product and repository on Satellite Server for hosting custom software packages and register a content host using an activation key, and then install custom software on that content host.

### Before You Begin

Log in to **workstation** as **student** using **student** as the password.

Run the **lab custom-review start** command. This command determines if the **satellite** and **serverb** machines are reachable on the network, and verifies that Red Hat Satellite is available.

```
[student@workstation ~]$ lab custom-review start
```

The Finance team would like to begin using Satellite Server to deploy their custom software packages. They would like to make the signing GPG public key available so that hosts can verify the origin and integrity of these packages during installation. For the implementation, they have initially provided you with an RPM, [http://materials.example.com/bkp-1.0-1.el7.x86\\_64.rpm](http://materials.example.com/bkp-1.0-1.el7.x86_64.rpm), and the GPG key, <http://materials.example.com/EXAMPLE-RPM-GPG-KEY>.

The Finance team currently registers their hosts using the **FinanceServers** activation key. They would like hosts to have access to their custom software packages upon registration.

You need to add the GPG key named **Example Software** to Satellite Server, then create a new product named **Custom Software** and a repository named **Admin Tools** to contain the provided RPM. You need to add the product subscription to the **FinanceKey** activation key.

Verify your implementation by registering **serverb.lab.example.com** to the **Finance** organization using the **FinanceKey** activation key. Finally, install the *bkp* package on **serverb.lab.example.com**.

1. Add the GPG public key.
  1. Choose the **Finance** organization and **Any Location** location from the main menu.
  2. Open a second tab in your browser. Navigate to <http://materials.example.com/EXAMPLE-RPM-GPG-KEY> to open the GPG key file in your browser. Copy the contents of the GPG key.
  3. Click **Content** → **Content Credentials** and then click **Create Content Credential**.

- 1.4. Enter **Example Software** in the **Name** field.
- 1.5. Select **GPG Key** from the **Type** list.
- 1.6. Paste the contents of the GPG key into the **Content Credential Contents** field, and then click **Save**.
2. Create the **Custom Software** product.
  - 2.1. Click **Content → Products** and then click **Create Product**.
  - 2.2. Enter the following product information, and then click **Save**:

| Field          | Value                                  |
|----------------|----------------------------------------|
| <b>Name</b>    | <b>Custom Software</b>                 |
| <b>Label</b>   | Automatically populated from the name. |
| <b>GPG Key</b> | <b>Example Software</b>                |

3. Create the **Admin Tools** repository.
  - 3.1. On the **Custom Software** page, click **New Repository**.
  - 3.2. Enter the following repository information:
4. Add the *bkp-1.0-1.el7.x86\_64.rpm* RPM package to the **Admin Tools** custom repository.
  - 4.1. On **workstation**, download the *bkp-1.0-1.el7.x86\_64.rpm* RPM package.

```
[student@workstation ~]$ wget -P ~/Downloads/ \
http://materials.example.com/bkp-1.0-1.el7.x86_64.rpm
...output omitted...
```

- 4.2. In the Satellite web UI, on the **Custom Software** page, click the **Repositories** tab, and then click **Admin Tools**.
- 4.3. In the **Upload Package** section, click **Browse**, and select *bkp* package file that you downloaded.
- 4.4. Click **Upload**. Notice that the package count for the repo is now 1.
5. Add a subscription for the **Custom Software** product to the **FinanceKey** activation key and enable access to the **Admin Tools** repository.

- 5.1. Click **Content** → **Activation Keys** and then click the **FinanceKey** activation key.
  - 5.2. Click the **Subscriptions** tab, and then the **Add** tab.
  - 5.3. Select the check box for the **Custom Software** subscription.
  - 5.4. Click **Add Selected**.
  - 5.5. Click the **Repository Sets** tab.
  - 5.6. Select the check box for the **Admin Tools** repository. If the status of the **Admin Tools** repository is **Disabled**, select **Override to Enable** from the **Select Action** list.
6. Ensure that the following repositories are included in the **Base** content view:
    - **Red Hat Enterprise Linux 8 for x86\_64 - BaseOS RPMs x86\_64 8**
    - **Red Hat Enterprise Linux 8 for x86\_64 - AppStream RPMs x86\_64 8**
    - **Red Hat Satellite Tools 6.6 for RHEL 8 x86\_64 RPMs x86\_64 8**
    - **Admin Tools**
- Publish and promote the repositories to the **Build** life-cycle environment.
- 6.1. Click **Content** → **Content Views** and click the **Base** content view.
  - 6.2. Select **Repositories** from the **Yum Content** list.
  - 6.3. On the **Add** tab, select the check box for any of the repositories listed above, and then click **Add Repositories**.
  - 6.4. Click **Publish New Version**.
  - 6.5. Note the version number assigned to this publication.

Click **Save**. You are redirected to the **Versions** tab. On the **Versions** tab, monitor the progress in the **Status** column. Wait for the publication to complete.
  - 6.6. For the version that you just published, click **Promote** and select the **Build** life-cycle environment.

Click **Promote Version**.
7. Prepare **serverb.lab.example.com** for registration to Satellite Server.

- 7.1. From **workstation**, ssh to **serverb** as **student**. Use **sudo -i** to switch to **root**.

```
[student@workstation ~]$ ssh student@serverb
[student@serverb ~]$ sudo -i
[sudo] password for student: student
[root@serverb ~]#
```

- 7.2. Download and install a copy of the CA Certificate from the following URL on Satellite Server:

```
[root@serverb ~]# yum localinstall \
http://satellite.lab.example.com/pub/katello-ca-consumer-latest.noarch.rpm
```

8. Register **serverb.lab.example.com** using the **FinanceKey** activation key and install the **katello-agent** package.

- 8.1. Clear any old registration data from the system.

```
[root@serverb ~]# subscription-manager clean
```

- 8.2. Register the system to the **Finance** organization. Reference the activation key with the **--activationkey** option.

```
[root@serverb ~]# subscription-manager register --org='Finance' \
--activationkey='FinanceKey'
The system has been registered with ID: f03ca761-9a8f-4c07-9468-f697520ad127
The registered system name is: serverb.lab.example.com
Installed Product Current Status:
Product Name: Red Hat Enterprise Linux for x86_64
Status: Subscribed
```



#### Note

Depending on the state of your **serverb** configuration it may be necessary to remove it from Satellite prior to registering.

Click **Hosts** → **Content Hosts** and select the check box next to **serverb.lab.example.com**. Choose **Remove Hosts** from the **Select Action** list.

- 8.3. Check for disabled repositories. The activation key makes these repositories available.

```
[root@serverb ~]# subscription-manager repos --list-disabled
+-----+
 Available Repositories in /etc/yum.repos.d/redhat.repo
+-----+
Repo ID: satellite-tools-6.6-for-rhel-8-x86_64-rpms
Repo Name: Red Hat Satellite Tools 6.6 for RHEL 8 x86_64 (RPMS)
Repo URL: https://satellite.lab.example.com/pulp/repos/Finance/Library/content/
dist/layered/rhel8/x86_64/sat-tools/6.6/os
Enabled: 0

Repo ID: ansible-2-for-rhel-8-x86_64-rpms
Repo Name: Red Hat Ansible Engine 2 for RHEL 8 x86_64 (RPMS)
Repo URL: https://satellite.lab.example.com/pulp/repos/Finance/Library/content/
dist/layered/rhel8/x86_64/ansible/2/os
Enabled: 0
```

- 8.4. Enable repositories.

```
[root@serverb ~]# subscription-manager repos --enable='*'
Repository 'Finance_Custom_Software_Admin_Tools' is enabled for this system.
Repository 'rhel-8-for-x86_64-baseos-rpms' is enabled for this system.
Repository 'satellite-tools-6.6-for-rhel-8-x86_64-rpms' is enabled for this
system.
Repository 'ansible-2.8-for-rhel-8-x86_64-rpms' is enabled for this system.
Repository 'rhel-8-for-x86_64-appstream-rpms' is enabled for this system.
```

- 8.5. Install **Katello Agent** on **serverb**.

```
[root@serverb ~]# yum install katello-agent
```

9. Install the **bkp** package. The output shows that the package is signed with the GPG private key. The output should also indicate the installation of the GPG public key.

```
[root@serverb ~]# yum install bkp  
...output omitted...  
Complete!
```

10. Return to **workstation** as **student** user.

This concludes the lab.

## Evaluation

As the **student** user on the **workstation** machine, use the **lab** command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab custom-review grade
```

## Finish

As the **student** user on the **workstation** machine, use the **lab** command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab custom-review finish
```

This concludes the lab.

## Summary

---

In this chapter, you learned:

- Red Hat Satellite can store and distribute non-Red Hat content, such as custom or third-party packages.
- You must create custom repositories and products to host custom and third-party content on Satellite Server.
- The repository discovery feature supports the automated discovery of all third-party software repositories available through a URL.
- The creation of a non-Red Hat product automatically associates a subscription. This subscription enables access to that product.

## Chapter 6

# Deploying Satellite Capsule Servers

### Goal

Perform installation and initial configuration of Red Hat Satellite Capsule Servers as components of a deployment plan.

### Objectives

- Install an external Satellite Capsule Server to provide Satellite Server support to a distributed location.
- Configure a Satellite Capsule Server for a defined purpose by enabling content features, infrastructure management services, and host management services.
- Create and publish content views, and promote them to life-cycle environments on an environment path.

### Sections

- Installing a Satellite Capsule Server (and Guided Exercise)
- Configuring Satellite Capsule Server Services (and Guided Exercise)
- Publishing Content Views to a Satellite Capsule Server (and Guided Exercise)

### Lab

Deploying Satellite Capsule Servers

# Installing a Satellite Capsule Server

---

## Objectives

After completing this section, you should be able to install an external Satellite Capsule Server to provide Satellite Server support to a distributed location.

## Satellite Capsule Server Requirements

Satellite Capsule Server should be provisioned on a clean system dedicated to only hosting Capsule. This ensures that no other function will compete for resources, or cause dependency conflicts during installation and upgrades.

Specific hardware and software requirements are identical to Satellite Server and will be required to scale based on workloads.

Satellite Capsule Server requires multiple ports for traffic.

### Capsule to Satellite

| <b>Port</b> | <b>Protocol</b> | <b>Service</b> |
|-------------|-----------------|----------------|
| 5646        | TCP             | amqp           |

### Host to Capsule:

| <b>Port</b> | <b>Protocol</b> | <b>Service</b> |
|-------------|-----------------|----------------|
| 80          | TCP             | HTTP           |
| 443         | TCP             | HTTPS          |
| 5647        | TCP             | amqp           |
| 8000        | TCP             | HTTPS          |
| 8140        | TCP             | HTTPS          |
| 8443        | TCP             | HTTPS          |
| 9090        | TCP             | HTTPS          |
| 53          | TCP and UDP     | DNS            |
| 67          | UDP             | DHCP           |
| 69          | UDP             | TFTP           |
| 5000        | TCP             | HTTPS          |

### To add required firewall rules:

- To allow Capsule server to access Satellite server, use the following command:

```
[root@demo_satellite ~]# firewall-cmd --permanent --add-port="5646/tcp"
```

- To allow Host access to Capsule server, use the following command:

```
[root@demo_capsule ~]# firewall-cmd --permanent --add-port="80/tcp" \
> --add-port="443/tcp" --add-port="5647/tcp" --add-port="8000/tcp" \
> --add-port="8140/tcp" --add-port="8443/tcp" --add-port="9090/tcp" \
> --add-port="53/tcp" --add-port="53/udp" --add-port="67/udp" \
> --add-port="69/udp" --add-port="5000/tcp"
```

- On both servers use the following command to apply changes:

```
[root@demo_capsule ~]# firewall-cmd --reload
success
```

## Installing Satellite Capsule Server

You must register the server that will host Capsule to the Satellite Server in order to access Red Hat Satellite Server products and subscriptions.

- Use the following command **yum** to download and install the CA certificate on the Capsule server.

```
[root@demo_capsule ~]# yum localinstall \
> http://satellite.lab.example.com/pub/katello-ca-consumer-latest.noarch.rpm
...output omitted...
Is this ok [y/N]: y
...output omitted...
Complete!
```

- Register the Capsule Server under the desired organization. The following example registers the Capsule Server under the **Default Organization** organization, using an activation key, however **bootstrap.py** is another option to simplify the process.

```
[root@demo_capsule ~]# subscription-manager register \
> --org="Default Organization" --activationkey=example_activation_key
The system has been registered with ID: ea19f207-b2dc-4269-bef2-8c39e76ca253
The registered system name is: capsule.example.com
Installed Product Current Status:
Product Name: Red Hat Enterprise Linux Server
Status: Subscribed
```

- Locate the *Pool ID* for the Satellite Infrastructure subscription.

```
[root@demo_capsule ~]# subscription-manager list --all --available \
> --matches 'Red Hat Satellite Infrastructure Subscription'
+-----+
Available Subscriptions
```

```
+-----+
Subscription Name: Red Hat Satellite Infrastructure Subscription
Provides: Red Hat Satellite
           Red Hat Software Collections (for RHEL Server)
           Red Hat CodeReady Linux Builder for x86_64
           Red Hat Satellite Capsule
           Red Hat Ansible Engine
           Red Hat Satellite with Embedded Oracle
           Red Hat Satellite 5 Managed DB
           Red Hat Enterprise Linux Load Balancer (for RHEL Server)
           Red Hat Beta
           Red Hat Software Collections Beta (for RHEL Server)
           Red Hat Enterprise Linux Server
           Red Hat Enterprise Linux for x86_64
           Red Hat Satellite Proxy
           Red Hat Enterprise Linux High Availability for x86_64
           Red Hat Discovery
SKU: MCT3718
Contract:
Pool ID: 40282d846e21fc1d016e2228d1fb2fff
Provides Management: Yes
Available: 20
Suggested: 1
Service Level: Premium
Service Type: L1-L3
Subscription Type: Standard
Starts: 06/28/2018
Ends: 03/01/2021
System Type: Physical
```

- Use the Pool ID to attach the subscription to Capsule Server.

```
[root@demo_capsule ~]# subscription-manager attach \
> --pool=40282d846e21fc1d016e2228d1fb2fff
Successfully attached a subscription for: Red Hat Satellite Infrastructure
Subscription
```

- Verify that all previous repositories are disabled.

```
[root@demo_capsule ~]# subscription-manager repos --disable "*"
...output omitted...
```

- Enable the repositories for Red Hat Server, Satellite Capsule, Satellite maintenance, Satellite tools, software collections, and Ansible repositories.

```
[root@demo_capsule ~]# subscription-manager repos --enable=rhel-7-server-rpms \
> --enable=rhel-7-server-satellite-capsule-6.6-rpms \
> --enable=rhel-7-server-satellite-maintenance-6-rpms \
> --enable=rhel-7-server-satellite-tools-6.6-rpms \
> --enable=rhel-server-rhscl-7-rpms \
> --enable=rhel-7-server-ansible-2.8-rpms
...output omitted...
Repository 'rhel-7-server-satellite-tools-6.6-rpms' is enabled for this system.
```

- Clear the repository cache and verify that all packages are up to date.

```
[root@demo_capsule ~]# yum clean all  
...output omitted...  
[root@demo_capsule ~]# yum update  
...output omitted...
```

- Install the **satellite-capsule** package.

```
[root@demo_capsule ~]# yum install satellite-capsule  
...output omitted...  
Total download size: 143 M  
Is this ok [y/d/N]: y  
...output omitted...
```

- On the Satellite Server use the **capsule-certs-generate** command to generate the needed SSL certificates. Output will include an **satellite-installer** command that you will use to complete installation.

```
[root@demo_satellite ~]# capsule-certs-generate \  
--foreman-proxy-fqdn capsule.example.com --certs-tar /root/capsule_SSL.tar  
Installing      Done          [100%]  
[.....]  
Success!  
...output omitted...  
satellite-installer \  
>   --scenario capsule \  
>   --certs-tar-file           "/root/capsule_SSL.tar"\ \  
>   --foreman-proxy-content-parent-fqdn "satellite.example.com"\ \  
>   --foreman-proxy-register-in-foreman "true"\ \  
>   --foreman-proxy-foreman-base-url  "https://satellite.example.com"\ \  
>   --foreman-proxy-trusted-hosts    "satellite.example.com"\ \  
>   --foreman-proxy-trusted-hosts    "capsule.example.com"\ \  
>   --foreman-proxy-oauth-consumer-key "mf6vLfWWpAE4fkMcTS3xx5AQMU2oQ4cf"\ \  
>   --foreman-proxy-oauth-consumer-secret "YP64QCHRBCUtaFTG5L8s7bevgTixBegL"\ \  
>   --puppet-server-foreman-url     "https://satellite.example.com"
```

- Copy the newly generated SSL tar file to Capsule Server.

```
[root@demo_satellite ~]# scp /root/capsule_SSL.tar \  
root@capsule.example.com:/root/capsule_SSL.tar  
...output omitted...
```

- On the Capsule Server use the entire **satellite-installer** command provided by the **capsule-certs-generate** command from previous steps. This command includes the path to the tar file which has the copied SSL certificates and configuration options based on your Satellite Server installation.

```
[root@demo_capsule ~]# satellite-installer \
>   --scenario capsule \
>   --certs-tar-file          "/root/capsule_SSL.tar" \
>   --foreman-proxy-content-parent-fqdn    "satellite.example.com" \
>   --foreman-proxy-register-in-foreman     "true" \
>   --foreman-proxy-foreman-base-url        "https://satellite.example.com" \
>   --foreman-proxy-trusted-hosts           "satellite.example.com" \
>   --foreman-proxy-trusted-hosts           "capsule.example.com" \
>   --foreman-proxy-oauth-consumer-key      "mf6vLfwWpAE4fkMcTS3xx5AQMU2oQ4Cf" \
>   --foreman-proxy-oauth-consumer-secret   "YP64QCHRBCUtaFTG5L8s7bevgTixBegL" \
>   --puppet-server-foreman-url            "https://satellite.example.com"
...output omitted...
Installing
Done
[100%]
Success!
* Capsule is running at https://capsule.example.com:9090
The full log is at /var/log/foreman-installer/capsule.log
```

Once Capsule Server is installed, verify that the capsule is assigned to the correct organization within the Satellite web UI.

- Choose the **Any Organization** organization context and the **Any Location** location context from the main menu.
- Navigate to **Infrastructure → Capsules**, and click **Edit** in the same row as the new Capsule Server.
- Click the **Organizations** tab and select the organization that the Capsule Server should be attached to. Click **Submit** to complete this process.



## References

For more information, refer to the *Installing Capsule Server* at  
[https://access.redhat.com/documentation/en-us/red\\_hat\\_satellite/6.6/html-single/installing\\_capsule\\_server/index](https://access.redhat.com/documentation/en-us/red_hat_satellite/6.6/html-single/installing_capsule_server/index)

## ► Guided Exercise

# Installing a Satellite Capsule Servers

In this exercise, you will install a Satellite Capsule Server using a selection of internal and external support services.

### Outcomes

You should be able to:

- Prepare a system for Satellite Capsule Server installation.
- Install and configure a Satellite Capsule Server.

### Before You Begin

Log in to **workstation** as **student** using **student** as the password.

Run the **lab capsule-install start** command to prepare your system for the exercise. The command determines if the **satellite** host is reachable on the network and verifies that the required repositories are available.

```
[student@workstation ~]$ lab capsule-install start
```



#### Important

The lab script takes approximately 30 minutes to create the required resources, including content views and an activation key, and to synchronize RHEL 7 repositories for the Capsule installation.

- 1. On the Capsule Server, execute the following commands to register to the Satellite Server under the **Operations** organization and attach the Satellite Infrastructure subscription.

- 1.1. On **workstation**, use **ssh** to log in to **capsule** as **student**. Use **sudo -i** to switch to **root**.

```
[student@workstation ~]$ ssh student@capsule
[student@capsule ~]$ sudo -i
[sudo] password for student: student
[root@capsule ~]#
```

- 1.2. Download the **bootstrap.py** script and use it to register the host. Use the **Operations** organization and the **Capsule\_Key** activation key. The activation key contains all repositories and subscriptions needed for installing Capsule.

```
[root@capsule ~]# wget http://satellite.lab.example.com/pub/bootstrap.py
...output omitted...
2019-11-27 10:42:42 (64.9 MB/s) - 'bootstrap.py' saved [72893/72893]
[root@capsule ~]# python bootstrap.py \
```

```
--login=admin --server=satellite.lab.example.com \
--organization=Operations --location="Default Location" \
--activationkey=Capsule_Key --skip foreman
Foreman Bootstrap Script
This script is designed to register new systems or to migrate an existing system
to a Foreman server with Katello
...output omitted...

[SUCCESS], [2019-11-27 10:56:58],...output omitted...completed successfully.
```

► 2. Configure and verify the firewall rules on the **capsule** server.

- 2.1. Use **firewall-cmd** to configure the correct ports for Capsule Server.

```
[root@capsule ~]# firewall-cmd --add-port="53/udp" --add-port="53/tcp" \
--add-port="67/udp" --add-port="69/udp" \
--add-port="80/tcp" --add-port="443/tcp" \
--add-port="5000/tcp" --add-port="5647/tcp" \
--add-port="8000/tcp" --add-port="8140/tcp" \
--add-port="8443/tcp" --add-port="9090/tcp" --permanent
success
[root@capsule ~]# firewall-cmd --reload
success
[root@capsule ~]# firewall-cmd --list-ports
53/udp 53/tcp 67/udp 69/udp 80/tcp 443/tcp 5000/tcp 5647/tcp 8000/tcp 8140/tcp
8443/tcp 9090/tcp
```

► 3. Configure the needed repositories for Capsule installation.

- 3.1. Use **subscription-manager repos --disable "\*"** to disable all currently configured repositories.

```
[root@capsule ~]# subscription-manager repos --disable "*"
...output omitted...
```

- 3.2. Use **subscription-manager** to enable the required repositories for the Capsule Server installation.

```
[root@capsule ~]# subscription-manager repos --enable=rhel-7-server-rpms \
--enable=rhel-server-rhscl-7-rpms \
--enable=rhel-7-server-satellite-capsule-6.6-rpms \
--enable=rhel-7-server-satellite-maintenance-6-rpms \
--enable=rhel-7-server-satellite-tools-6.6-rpms
...output omitted...
Repository 'rhel-7-server-satellite-tools-6.6-rpms' is enabled for this system.
```

- 3.3. Use **yum clean all** to flush the Yum cache.

```
[root@capsule ~]# yum clean all
...output omitted...
Uploading Enabled Repositories Report
Loaded plugins: langpacks, product-id, subscription-manager
```

► 4. Install the `satellite-capsule` package.

```
[root@capsule ~]# yum install satellite-capsule
...output omitted...
Total download size: 143 M
Is this ok [y/d/N]: y
...output omitted...
Complete!
Uploading Enabled Repositories Report
Loaded plugins: langpacks, product-id, subscription-manager
```

► 5. On Satellite Server, generate a certificate and copy it to Capsule Server.

- 5.1. On **workstation**, use `ssh` to log in to **satellite** as **student**. Use `sudo -i` to switch to **root**.

```
[student@workstation ~]$ ssh student@satellite
[student@satellite ~]$ sudo -i
[sudo] password for student: student
[root@satellite ~]#
```

- 5.2. Use `mkdir /root/capsule_SSL` to create a directory to save the generated SSL certificate.

```
[root@satellite ~]# mkdir /root/capsule_SSL
```

- 5.3. Use `capsule-certs-generate --foreman-proxy-fqdn capsule.lab.example.com --certs-tar /root/capsule_SSL/capsule_SSL.tar` to generate the SSL certificate. Copy the **satellite-installer** command with all the options, you will need it in the next steps.



**Important**

The `capsule-certs-generate` command generates not only the SSL certificates but also the correct command for you to use and install the capsule. Be cautious and copy the installation command from the output, you will need this in the next step of this exercise.

```
[root@satellite ~]# capsule-certs-generate --foreman-proxy-fqdn \
capsule.lab.example.com --certs-tar \
/root/capsule_SSL/capsule_SSL.tar
...output omitted...
satellite-installer \
--scenario capsule \
--certs-tar-file          "/root/capsule_SSL.tar"\ \
--foreman-proxy-content-parent-fqdn  "satellite.lab.example.com"\ \
--foreman-proxy-register-in-foreman  "true"\ \
--foreman-proxy-foreman-base-url    "https://satellite.lab.example.com"\ \
--foreman-proxy-trusted-hosts      "satellite.lab.example.com"\ \
--foreman-proxy-trusted-hosts      "capsule.lab.example.com"\ \
```

```
--foreman-proxy-oauth-consumer-key      "dacaNSsRnZyFJ6domy4wVd7YnpMJKYLU"\"  
--foreman-proxy-oauth-consumer-secret   "8sFNbsqVykgRrqR4dUCekWuCNLhh6V8P"\"  
--puppet-server-foreman-url            "https://satellite.lab.example.com"
```

5.4. Use **scp** command to copy the newly generated SSL certificate to Capsule Server.

```
[root@satellite ~]# scp /root/capsule_SSL/capsule_SSL.tar \  
root@capsule.lab.example.com:~  
...output omitted...  
Are you sure you want to continue connecting (yes/no)? yes  
...output omitted...  
root@capsule.lab.example.com's password: redhat  
capsule_SSL.tar      100%   61KB  31.9MB/s  00:00
```

5.5. On **workstation**, use **ssh** to log in to **capsule** as **student**. Use **sudo -i** to switch to **root**.

```
[student@workstation ~]$ ssh student@capsule  
...output omitted...  
[student@capsule ~]$ sudo -i  
[sudo] password for student: student  
[root@capsule ~]#
```

5.6. Configure Capsule Server with the SSL certificates created and copied in previous steps. Paste the Capsule installation command from the previous step and use it to install the Capsule. This may take up to 15 minutes.

```
[root@capsule ~]# satellite-installer \  
--scenario capsule \  
--certs-tar-file          "/root/capsule_SSL.tar" \  
--foreman-proxy-content-parent-fqdn  "satellite.lab.example.com" \  
--foreman-proxy-register-in-foreman "true" \  
--foreman-proxy-foreman-base-url    "https://satellite.lab.example.com" \  
--foreman-proxy-trusted-hosts      "satellite.lab.example.com" \  
--foreman-proxy-trusted-hosts      "capsule.lab.example.com" \  
--foreman-proxy-oauth-consumer-key "dacaNSsRnZyFJ6domy4wVd7YnpMJKYLU"\"  
--foreman-proxy-oauth-consumer-secret "8sFNbsqVykgRrqR4dUCekWuCNLhh6V8P"\"  
--puppet-server-foreman-url       "https://satellite.lab.example.com"  
...output omitted...  
Installing Done [100%]  
Success!  
* Capsule is running at https://capsule.lab.example.com:9090  
The full log is at /var/log/foreman-installer/capsule.log
```

► 6. Move Capsule Server to the **Boston** location and the **Operations** organization.

6.1. Use your browser to navigate to <https://satellite.lab.example.com>. Log in as **admin** using **redhat** as the password.

6.2. Choose the **Any Organization** organization and **Any Location** location context.

6.3. Navigate to **Infrastructure → Capsules**.

- 6.4. Click **Edit** in the same row as **capsule.lab.example.com**.
  - 6.5. Click the **Locations** tab. Click **Boston** on the **Locations** list.
  - 6.6. Click the **Organizations** tab. Verify that the **Operations** organization has been selected and click **Submit**.
- ▶ 7. Log off from the **capsule** and **satellite** hosts and return to **workstation**.
- ▶ 8. Log out from the Satellite web UI.

## Finish

On the **workstation** machine, use the **lab capsule-install finish** command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab capsule-install finish
```

This concludes the guided exercise.

# Configuring Satellite Capsule Server Services

---

## Objectives

After completing this section, you should be able to configure a Satellite Capsule Server for a defined purpose by enabling content features, infrastructure management services, and host management services.

## Satellite Capsule Configuration

Satellite Capsule Server requires review and modification of configuration settings to match the environment's needs.

- In the Satellite web UI, navigate to **Infrastructure → Capsules**. Notice that only the Capsules that are configured for the currently selected organization and location are displayed. Click **Edit** to start editing the Capsule settings.
  - Select the **Capsule** tab.
  - The **Name\*** field is the display name for the Capsule Server within the Satellite UI.
  - The **Uri\*** field provides the URL and port of Capsule Server, which is used for communication..
  - The **Download Policy** sets when Capsule Server downloads packages. The options function the same as for Satellite Server with the exception that Immediate will cause Satellite to download all packages as well, regardless of its settings.

## Selecting Lifecycle Environments

When Capsule has content functionality enabled, you must assign a **Lifecycle Environment** in order to synchronize repositories. If the **Library** life cycle is assigned then a synchronization is triggered each time the CDN updates a repository. This will consume large amounts of resources and should be avoided.

- To enable Lifecycle Environments, select the **Lifecycle Environments** tab.
- The table on the left provides a list of available lifecycle environments, click the lifecycle name to assign it to the capsule. Selecting one that is already assigned in the table on the right will remove the entry.
- Click **Submit** to accept the changes.

## Location and Organization Management

- Select the **Locations** tab. The table on the left provides a list of available locations. Clicking the location name will assign it to the capsule. Selecting one that is already assigned in the table on the right will remove the entry. Click **Submit** to accept the changes.
- Select the **Organizations** tab. The table on the left provides a list of available organizations. Clicking the organization name will assign it to the capsule. Selecting one that is already assigned in the table on the right will remove the entry. Click **Submit** to accept the changes.

## Capsule Features and Services

To view features, services, and their related information:

- Within the Satellite UI Navigate to **Infrastructure** → **Capsules**, click on the **Name** for the Capsule. Click **Overview**.
- Active features** lists active features currently available to the Capsule.
- Click **Services** to display active services as well as their respective version.

## Configuring Satellite Services

Configuring services is handled on the command line of the Capsule Server. Use **satellite-installer --scenario capsule --help** to view all available options.

```
[root@demo_capsule ~]# satellite-installer --scenario capsule --help
...output omitted...
```

To configure TFTP, DNS, and DHCP on Capsule Server, use the **satellite-installer** command.

```
root@demo_capsule ~]# satellite-installer --scenario capsule \
> --foreman-proxy-tftp true \
> --foreman-proxy-tftp-managed true \
> --foreman-proxy-tftp-servername capsule.lab.example.com \
> --foreman-proxy-dns true \
> --foreman-proxy-dns-managed true \
> --foreman-proxy-dns-interface eth1 \
> --foreman-proxy-dns-zone lab.example.com \
> --foreman-proxy-dns-forwarders 172.25.251.1 \
> --foreman-proxy-dns-reverse 251.25.172.in-addr.arpa \
> --foreman-proxy-dhcp true \
> --foreman-proxy-dhcp-managed true \
> --foreman-proxy-dhcp-interface eth1 \
> --foreman-proxy-dhcp-range "172.25.251.100 172.25.251.150" \
> --foreman-proxy-dhcp-gateway 172.25.251.1 \
> --foreman-proxy-dhcp-nameservers 172.25.251.2
...output omitted
Preparing installation Debug: /File[/etc/foreman-proxy/settings.d/puppet_
Preparing installation Debug: /File[authn_core.load]/seluser: Found selus
Preparing installation Debug: /File[/etc/systemd/system/smart_proxy_dynfl
Installing     Package[dhcp]                                [2%] [...]
...output omitted
Installing     Service[smart_proxy_dynflow_core]          [100%] [...]
Installing     Done                                     [100%] [...]
The full log is at /var/log/foreman-installer/capsule.log
```

After additional services have been configured, they display in the Satellite UI within the **Overview** and **Services** tabs, respectively.



## References

For more information, refer to the *Installing Capsule Server* at  
[https://access.redhat.com/documentation/en-us/red\\_hat\\_satellite/6.6/html-single/installing\\_capsule\\_server/index](https://access.redhat.com/documentation/en-us/red_hat_satellite/6.6/html-single/installing_capsule_server/index)

## ► Guided Exercise

# Configuring Satellite Capsule Server Services

In this exercise, you will verify that your Satellite Capsule Server installed correctly.

### Outcomes

You should be able to verify the current status for the Satellite Capsule Server.

### Before You Begin

Log in as the **student** user on **workstation** using **student** as the password.

Run the **lab capsule-enable start** command. This command determines if the **capsule** host is reachable on the network and verifies that Satellite Capsule Server is available.

```
[student@workstation ~]$ lab capsule-enable start
```

To run this exercise, you need a Satellite Capsule Server to be available on **capsule** and registered on the Satellite Server on **satellite**.

- ▶ 1. Log in to the Satellite web UI as **admin** using **redhat** as the password. Select the **Operations** organization and the **Any Location** location.
  - 1.1. Use your browser to navigate to <https://satellite.lab.example.com> and log in as **admin** using **redhat** as the password.
  - 1.2. Select the **Operations** organization and **Any Location** location.
- ▶ 2. Verify the status of the Satellite Capsule Server services.
  - 2.1. Click **Infrastructure → Capsules** and then click **capsule.lab.example.com**.
  - 2.2. Click the **Services** tab, and notice that the capsule is running the following services: **Dynflow**, **HTTPBoot**, **Pulp node**, **SSH**, **TFTP**, and **Templates**.
  - 2.3. Click the **Logs** tab, and notice the most recent log messages about Satellite Capsule Server services initialization.
- ▶ 3. Install the **pulp-admin** utility on the Satellite Capsule server.
  - 3.1. On **workstation**, use **ssh** to log in to **capsule** as **student**. Use **sudo -i** to switch to **root**.

```
[student@workstation ~]$ ssh student@capsule
[student@capsule ~]$ sudo -i
[root@capsule ~]#
```

3.2. Install the `pulp-admin-client` package on **capsule**.

```
[root@capsule ~]# yum install pulp-admin-client  
...output omitted...  
Is this ok [y/N]: yes  
...output omitted...
```

3.3. Determine the password for the **admin** user on the Satellite Capsule Server.

```
[root@capsule ~]# grep default_password /etc/pulp/server.conf  
...output omitted...  
default_password: MRBMaU8P4X6pD4jAQAtXAjwijtvARMMB
```

► 4. Verify that no repository is available into the Satellite Capsule Server.

```
[root@capsule ~]# pulp-admin -u admin -p MRBMaU8P4X6pD4jAQAtXAjwijtvARMMB \  
repo list  
+-----+  
| Repositories |  
+-----+
```

## Finish

On the **workstation** machine, use the **lab** command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab capsule-enable finish
```

This concludes the guided exercise.

# Publishing Content Views to a Satellite Capsule Server

---

## Objectives

After completing this section, you should be able to create and publish content views, and promote them to life-cycle environments on an environment path.

## Creating Content Views

To provide content to hosts from a Capsule Server you must first create a content view on Satellite Server.

### To Create a Content View:

- For the required organization and location, click **Content** → **Content Views**, and then click **Create New View**.
- On the **Create Content View** page, enter the required name and description, and then click **Save**. The label is automatically set to the value of **Name**, replacing all white space and punctuation with underscores.

## Adding Repositories to a Content View

When you first create a content view it has no content associated with it. To populate a content view, associate it with the repositories that contain the required content. You can associate more than one repository with a content view.

### To Add Repositories to a Content View:

- For the required organization and location, click **Content** → **Content Views**, and then click the name of the content view.
- Click **Yum Content** → **Repositories** and then click the **Add** tab.
- Select the check box for the repositories that you want to add, and then click **Add Repositories**.

## Publishing a Content View

After you create a content view, you need to publish it to the library. Every time you publish a content view it creates a new version of that content view. Every version is numbered sequentially for identification, as well as for version control purposes.

### To Publish a Content View:

- For the required organization and location, click **Content** → **Content Views**, and then click the name of the content view that you want to publish.
- After you have made changes, such as updating packages in the content view, click **Publish New Version**. Note that the version number of the content view increases.

- Add a meaningful description. You may want to mention the difference between content view versions as part of the description of the content view update. A meaningful description will help you identify the purpose of that version at a later date. The version number itself does not provide any information about the content of the content view.
- After you have entered a suitable description, click **Save**.

## Promoting a Content View

After you publish a content view version to the library, you promote it to a life-cycle environment in an environment path.

### To Promote a Content View:

- For the required organization and location, click **Content** → **Content Views**, and then click the name of the content view that you want to promote.
- The **Versions** tab lists the published versions of the content view. Each row on this tab represents a specific content view version.

In the **Actions** column, click **Promote** to display the **Promotion** page.

- Select the life-cycle environment that you want to promote to, and click **Promote Version**.

## Synchronizing a Life-cycle Environment with Capsule Server

To provide content to a host from Capsule Server you must add a life-cycle environment so it can synchronize its content from the Satellite Server.

### To Add a Life-cycle Environment to Capsule Server:

- From the Satellite web UI, navigate to **Infrastructure** → **Capsules**, and select the Capsule Server that you want to add a life cycle to.
- Click **Edit** and then click the **Life Cycle Environments** tab.
- Select the life-cycle environments that you want to add to Capsule and then click **Submit**.

### To Synchronize Content to Capsule Server:

- Click the **Overview** tab.
- From the **Synchronize** list, there are two options:
  - **Optimized Sync** for a focus on speed by bypassing unneeded steps such as syncing unchanged metadata.
  - **Complete Sync** which will sync repositories even if upstream metadata has not changed.

## Describing Capsule Server Download Policies

Red Hat Satellite provides multiple download policies for synchronizing RPM content for repositories. For example, you might want to download only the content metadata while deferring the actual content download for later.

The **Download Policy** sets when Capsule Server downloads packages. The options function the same as for Satellite Server with the exception that **Immediate** will cause Satellite to download all packages as well, regardless of its settings.

**To Change the Download Policy for a Repository:**

- Navigate to **Content** → **Products**, and then click the product name.
- On the **Repositories** tab, click the required repository name, locate the **Download Policy** field, and click the **edit** icon.
- Select the required download policy and then click **Save**.



### References

For more information, refer to the *Managing Content Views* chapter in the *Red Hat Satellite 6.6 Content Management Guide* at  
[https://access.redhat.com/documentation/en-us/red\\_hat\\_satellite/6.6/html-single/content\\_management\\_guide/managing\\_content\\_views#Managing\\_Content\\_Views](https://access.redhat.com/documentation/en-us/red_hat_satellite/6.6/html-single/content_management_guide/managing_content_views#Managing_Content_Views)

## ► Guided Exercise

# Publishing Content Views to a Satellite Capsule Server

In this exercise, you will publish a new or modified content view and promote it to a specific life-cycle environment.

## Outcomes

You should be able to publish a new content view, promote it to a specific life-cycle environment, and synchronize it with your local Capsule.

## Before You Begin

This exercise requires a running Satellite Capsule Server on the **capsule** host, which you installed in a preceding exercise.

Log in to **workstation** as **student** using **student** as the password.

Run the **lab capsule-views start** command to prepare your system for the exercise. The command determines if the **satellite** host is reachable on the network and verifies that the required repositories are available.

```
[student@workstation ~]$ lab capsule-views start
```



### Note

The lab script verifies and if needed creates the required resources, including the content views, repositories, and the activation key.

- 1. Log in to the Satellite web UI as **admin** using **redhat** as the password.
  - 1.1. Use your browser to navigate to <https://satellite.lab.example.com>.
  - 1.2. Log in as **admin** using **redhat** as the password.
- 2. Choose the **Operations** organization and **Boston** location from the main menu.
- 3. Enable synchronization of the **Red Hat Ansible Engine 2.8 RPMs for RHEL 8** repository from the local CDN.
  - 3.1. Click **Content** → **Red Hat Repositories** to access the **Red Hat Repositories** page.
  - 3.2. Expand **Red Hat Ansible Engine 2.8 RPMs for RHEL 8**. Click the plus sign (+) next to **x86\_64** to enable it. This repository may already be enabled in your environment.
- 4. Use the Satellite Server web UI to synchronize the new repository.

- 4.1. Click **Content** → **Products** to open the **Products** page.
- 4.2. Click **Red Hat Ansible Engine**. The **Repositories** tab displays.
- 4.3. Select the check box next to the **Red Hat Ansible Engine 2.8 RPMs for RHEL 8 x86\_64 RPMs x86\_64** repository. Click **Sync Now** to begin the repository synchronization.

Wait for the synchronization task to complete.
- ▶ 5. Add the new repository to the **Base** content view. When done, publish and promote the content view to the **Development** life-cycle environment.
  - 5.1. Click **Content** → **Content Views** to access the **Content Views** page. Click the **Base** content view.
  - 5.2. Click **Yum Content** → **Repositories**, and then click the **Add** tab.
  - 5.3. Select the check boxes next to the **Red Hat Ansible Engine 2.8 RPMs for RHEL 8 x86\_64 RPMs x86\_64** and then click **Add Repositories**.
  - 5.4. Click **Publish New Version**, and then click **Save**. Wait for the publishing process to complete. The task can take up to 10 minutes to finish.
  - 5.5. When the publishing process is complete, click **Promote**. Select the **Development** life-cycle environment, and then click **Promote Version**.
- ▶ 6. For the new Ansible Engine repository to be available from Satellite Capsule Server in Boston, synchronize the **Development** life-cycle environment with Satellite Capsule Server.
  - 6.1. Click **Infrastructure** → **Capsules** to access the **Capsules** page. Click **Edit** at the end of the **capsule.lab.example.com** row.

Click the **Lifecycle Environments** tab, and then click **Development** to add it to the **Selected items** list.

Click **Submit**.
  - 6.2. On the **Capsules** page, click the **capsule.lab.example.com** link. Click **Synchronize** → **Optimized Sync** to start a new synchronization.

You have just added a new repository, synchronized that to the Satellite from your local CDN, published a new version of content view containing the new repository, and at the end synchronized that new content view version with your local capsule.

## Finish

On the **workstation** machine, use the **lab** command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab capsule-views finish
```

This concludes the guided exercise.

## ► Quiz

# Deploying Satellite Capsule Servers

Choose the correct answers to the following questions:

- ▶ 1. Which three of the following features can be provided by Satellite Capsule Server? (Choose three).
  - a. Discovery
  - b. Windows errata management
  - c. DNS and DHCP servers
  - d. Repository synchronization
  
- ▶ 2. Which port is required for communications between Satellite Server and Satellite Capsule Server?
  - a. 5000
  - b. 80
  - c. 5646
  - d. 443
  
- ▶ 3. Which five of the following repositories are required for capsule installation? (Choose five).
  - a. `rhel-7-server-satellite-tools-6.6-rpms`
  - b. `rhel-8-server-ansible-1.8-rpms`
  - c. `rhel-server-rhscl-7-rpms`
  - d. `rhel-7-server-satellite-capsule-6.6-rpms`
  - e. `rhel-7-server-satellite-maintenance-6-rpms`
  - f. `rhel-7-server-rpms`
  
- ▶ 4. Which two of the following utilities can be used to retrieve information about the status of the Satellite Capsule Server? (Choose two).
  - a. `pulp-admin`
  - b. `capsule-certs-generate`
  - c. `hammer`
  - d. `satellite-installer`

## ► Solution

# Deploying Satellite Capsule Servers

Choose the correct answers to the following questions:

- ▶ 1. Which three of the following features can be provided by Satellite Capsule Server? (Choose three).
  - a. Discovery
  - b. Windows errata management
  - c. DNS and DHCP servers
  - d. Repository synchronization
  
- ▶ 2. Which port is required for communications between Satellite Server and Satellite Capsule Server?
  - a. 5000
  - b. 80
  - c. 5646
  - d. 443
  
- ▶ 3. Which five of the following repositories are required for capsule installation? (Choose five).
  - a. `rhel-7-server-satellite-tools-6.6-rpms`
  - b. `rhel-8-server-ansible-1.8-rpms`
  - c. `rhel-server-rhscl-7-rpms`
  - d. `rhel-7-server-satellite-capsule-6.6-rpms`
  - e. `rhel-7-server-satellite-maintenance-6-rpms`
  - f. `rhel-7-server-rpms`
  
- ▶ 4. Which two of the following utilities can be used to retrieve information about the status of the Satellite Capsule Server? (Choose two).
  - a. `pulp-admin`
  - b. `capsule-certs-generate`
  - c. `hammer`
  - d. `satellite-installer`

## Summary

---

In this chapter, you learned:

- There are many prerequisites to satisfy prior to installing a Red Hat Capsule Server.
- Capsule requires a life-cycle environment to provide content to hosts.
- Many Capsule services are not installed and configured by default and require setting specifically for the environment.
- Content views can be handled differently between Capsule Servers.

## Chapter 7

# Running Remote Execution

### Goal

Configure the ability to run ad hoc and scheduled tasks on managed hosts using a variety of configuration management tools.

### Objectives

- Prepare for remote execution by establishing a secure connection to hosts and creating job templates, and then run remote jobs and view job results.
- Configure and enable remote execution and install additional Ansible Roles on a Satellite Capsule Server.
- Create a content view that supports Puppet configuration and register a Puppet agent to a Red Hat Satellite server.

### Sections

- Running Remote Jobs on Managed Hosts (and Guided Exercise)
- Configuring Ansible Remote Execution (and Guided Exercise)
- Running Remote Puppet Jobs on Managed Hosts (and Guided Exercise)

### Lab

Running Remote Execution

# Running Remote Jobs on Managed Hosts

## Objectives

After completing this section, you should be able to prepare for remote execution by establishing a secure connection to hosts and creating job templates, and then run remote jobs and view job results.

## Running Remote Jobs on Hosts

Satellite allows users to run remote jobs on hosts using shell commands and scripts, and Ansible ad hoc commands, playbooks, and roles. This feature is called *remote execution*.

The remote execution feature is enabled by default on Satellite Server, but must be specifically enabled on Capsule Server. With the feature enabled, Capsule Servers can handle remote execution for their managed hosts, allowing Satellite Server to scale to control many hosts without requiring direct access to all target hosts.

For remote execution to work, SSH must be enabled and running on the target hosts. Satellite and Capsule must have access to port 22 on target hosts.

Jobs can be run on multiple hosts at the same time, and command variables can be used to simplify multiple target scripts.

## Enabling Remote Execution

Capsules by default are installed with the remote execution feature disabled. To enable remote execution on your Capsule, use the following command:

```
[root@demo_capsule ~]# satellite-installer --scenario capsule \
--enable-foreman-proxy-plugin-remote-execution-ssh
```

### To Verify if Capsule has Remote Execution Enabled:

- Navigate to **Infrastructure → Capsules**.
- View the **Features** column for the Capsule. Remote execution is enabled when **SSH** is listed.

| Capsules                  |                             |                                        |                                                                                          |                           |
|---------------------------|-----------------------------|----------------------------------------|------------------------------------------------------------------------------------------|---------------------------|
| Name                      | Locations                   | Organizations                          | Features                                                                                 | Actions                   |
| capsule.lab.example.com   | Boston and Default Location | Operations                             | Ansible, Dynflow, HTTPBoot, Logs, Pulp Node, Puppet, Puppet CA, SSH, TFTP, and Templates | <a href="#">Edit</a>      |
| satellite.lab.example.com | Default Location            | Default Organization, Finance, and ... | Ansible, Discovery, Dynflow, HTTPBoot, Logs, Openscap, ...                               | <a href="#">Edit</a>      |
| 20 ^ per page             |                             |                                        |                                                                                          | 1-2 of 2 << < 1 of 1 > >> |

Figure 7.1: Remote execution enabled

By default, Satellite uses remote execution to run the remote jobs, but you can change this behavior to use the Katello agent if required. To use the Katello agent, navigate to **Administer → Remote Execution Features** and change the remote execution type.

## Distributing SSH Keys for Remote Execution

During Capsule installation the SSH keys for remote execution are created automatically. Keys are not automatically distributed. You must manually distribute the keys to each host where you want to use remote execution.

The remote execution settings for SSH are stored in the `/etc/foreman-proxy/settings.d/remote_execution_ssh.yml` file on each of your Capsules.

To use SSH keys for remote execution authentication, you must distribute the Capsule public SSH key to each registered host to manage. Verify that the SSH service is enabled and running on all the hosts, and that the firewall enables access to port 22.

There are multiple ways to distribute the public key from your Capsule to the target hosts:

- Distribute the SSH keys manually using the `ssh-copy-id` command. Repeat for each target host to manage.

```
[root@demo_capsule ~]# ssh-copy-id -i \
~foreman-proxy/.ssh/id_rsa_foreman_proxy.pub \
root@target_host.example.com
```

- Use the Satellite API to download the public key from the Capsule. On each target host use the following command:

```
[root@target_host ~]# curl \
https://demo_capsule.example.com:9090/ssh/pubkey >> \
~/ssh/authorized_keys
```

- Distribute the public key when you provision a new system using Kickstart. To include the public key in the Kickstart configuration, modify the **Kickstart default finish** template to include the following line:

```
<%= snippet 'remote_execution_ssh_keys' %>
```



### Note

You can use Kerberos authentication to establish an SSH connection for remote execution. For more information on how to configure your environment using Kerberos, refer to the *Setting Up Kerberos Authentication for Remote Execution* section in the *Managing Hosts* guide at [https://access.redhat.com/documentation/en-us/red\\_hat\\_satellite/6.6/html/managing\\_hosts/chap-managing\\_hosts-running\\_remote\\_jobs\\_on\\_hosts#setting\\_up\\_kerberos\\_authentication\\_for\\_remote\\_execution](https://access.redhat.com/documentation/en-us/red_hat_satellite/6.6/html/managing_hosts/chap-managing_hosts-running_remote_jobs_on_hosts#setting_up_kerberos_authentication_for_remote_execution)

## Using Job Templates to Define Remote Jobs

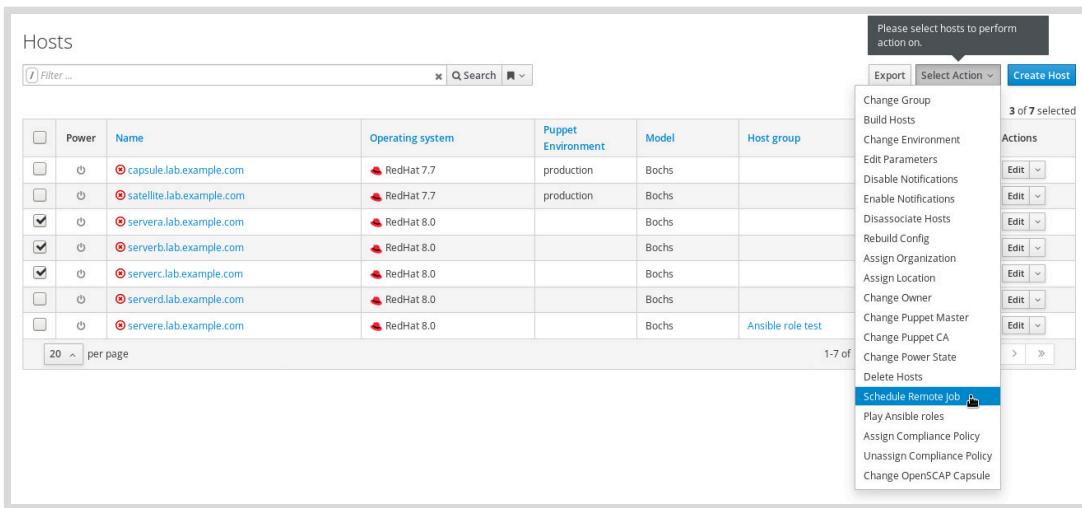
The commands to run on a remote host must exist on Satellite as a job template. Several job templates have been included on your Satellite, but you can create custom job templates. You can manage and execute any job templates multiple times.

### Invoking a Job Template

With the SSH keys distributed on the target hosts, you can now run jobs based on existing job templates against one or more of the hosts.

This procedure shows how to execute a remote job based on an existing template:

- Log in to the Satellite web UI as a user with the appropriate access level.
- Typically, choose the Organization and Location where the target host is registered. To access the full list of hosts, set the organization to **Any Organization** and the location to **Any Location**.
- Navigate to **Hosts** → **All hosts** and select the required target hosts for the job.
- Click **Select Action** and choose **Schedule Remote Job**.



**Figure 7.2: Remote Job Scheduling**

- On the **Job invocation** page, define the job settings:
  - Select the **Job category** and the **Job template** to use.
  - Enter a search query to limit the list of targeted hosts (optional).
  - The remaining settings depend on the job template you have selected. Choosing a **Run Command** template allows running a Linux command or script.
- To execute the job as a user other than the SSH user, click **Display advanced fields**. Advanced settings depend on the job template type. In this example, **Effective user** defines the user for executing the job.
- Specify when to run this job:

| Option                            | Description                                                                                          |
|-----------------------------------|------------------------------------------------------------------------------------------------------|
| <b>Execute now</b>                | Execute the job on submission.                                                                       |
| <b>Schedule future execution</b>  | Specify a future date and time to execute the job.                                                   |
| <b>Set up recurring execution</b> | Create a recurring job where you can specify the start and end dates, number, and frequency of runs. |

- Click **Submit** to display the **Job Overview** page, showing the status of the job.

Job category \* Commands

Job template \* Run Command - SSH Default

Bookmark

Search Query name ^ (serverb.lab.example.com, servera.lab.example.com, serverc.lab.example.com)

Resolves to 3 hosts

command \* uptime

Schedule  Execute now  Schedule future execution  Set up recurring execution

Start at 2019-11-06 07:08

Start before YYYY-mm-dd HH:MM

[Display advanced fields](#)

Submit Cancel

Figure 7.3: Remote job scheduling

## Monitoring Jobs

While executing, monitor the job progress from the **Job invocation** page. For finished jobs or those scheduled to run, access the job output on the **Job** page.

To view currently running job output on the **Job invocation** page, click the name of the target host. This opens a new page with the output of your job.

### To Monitor Scheduled Jobs:

- Navigate to **Monitor** → **Jobs** and select the job to inspect.
- On the **Job invocations** page, click the job to inspect its output.

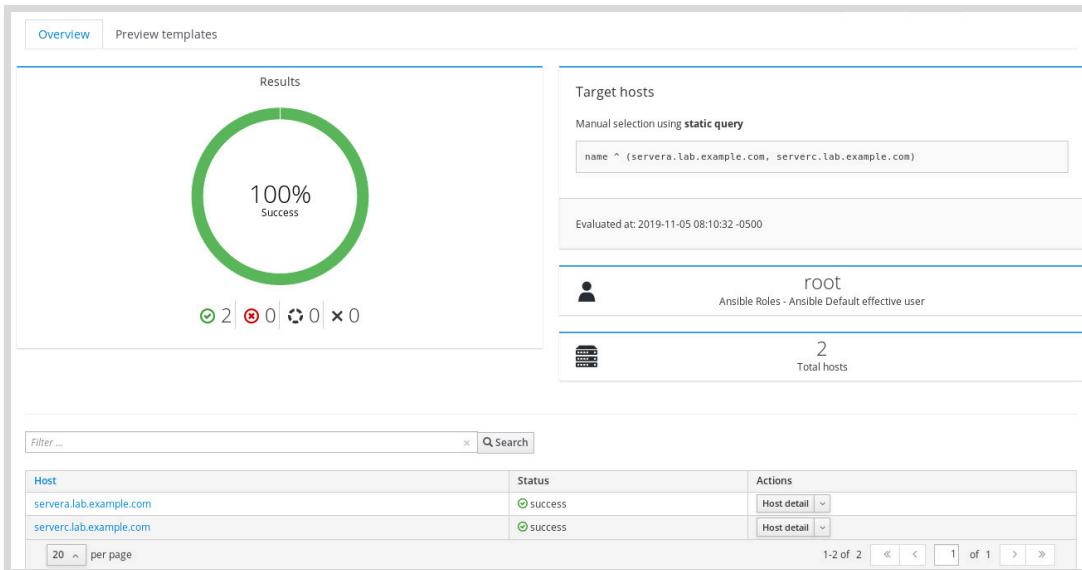


Figure 7.4: Job details page

- On the **Overview** tab, scroll down and click the host to inspect. This displays a new page where you can monitor the job execution results.

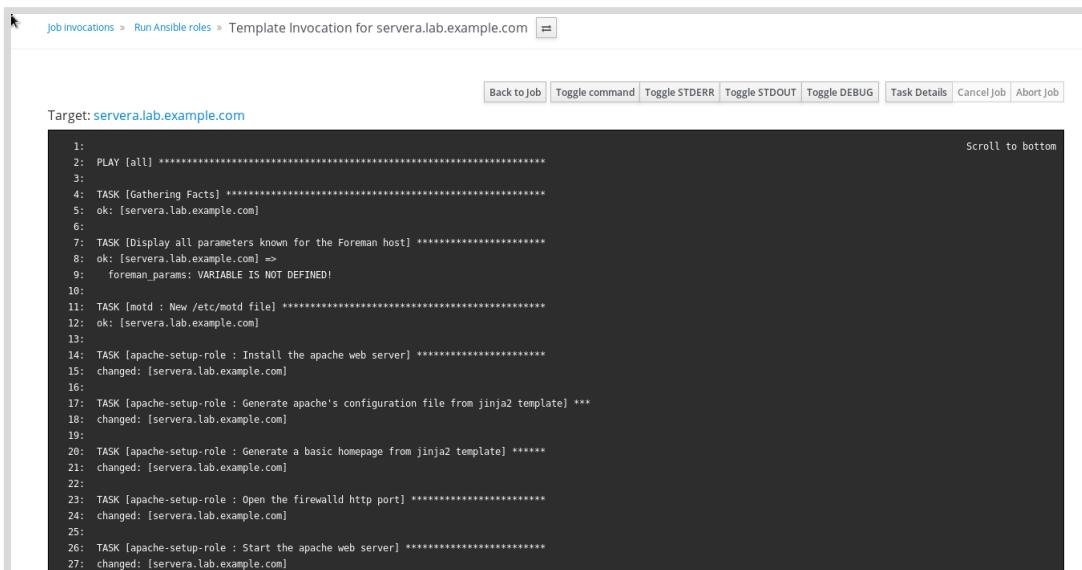


Figure 7.5: Monitoring a remote job

## Executing and Monitoring Jobs from CLI

You can execute remote jobs and monitor them from the CLI using the **hammer** command. You can create shell scripts to interact with Satellite and automate your administrative tasks.

### To Execute a Remote Job from the CLI:

- To access template details and parameters, find and use the job template's ID.

```
[root@demo_capsule ~]# hammer job-template list
[root@demo_capsule ~]# hammer job-template info --id your_template_ID
```

- Execute a remote job with custom parameters. Replace the example parameters and query details with your job template details, and create a filter that specifies the target hosts (for example "name=server1.example.com").

```
[root@demo_capsule ~]# hammer job-invocation create \
--job-template template_name --inputs key1=value,key2=value \
--search-query query
```

#### To Monitor a Running Job or Inspect the Output of a Finished Remote Job:

- Use the ID to access the job output. Replace **host\_name** with the name of the host the job was executed on.

```
[root@demo_capsule ~]# hammer job-invocation \
output --id your_job_ID \
--host host_name
```

#### To Cancel a Running Job:

- Use the following **hammer** command:

```
[root@demo_capsule ~]# hammer \
job-invocation cancel \
--id job_ID
```

## Satellite Remote Execution Global Settings

Satellite provides global settings on the **Administer** → **Settings** page to customize remote execution.

The following table explains some of the most important global settings that you can use:

| Parameter name                    | Description                                                                                                                                     |
|-----------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Default SSH key passphrase</b> | Specifies the default passphrase to use for SSH.                                                                                                |
| <b>Effective User Method</b>      | Specifies the method to use to set the effective user on the target hosts.                                                                      |
| <b>Effective User</b>             | Specifies the effective user for any job. This setting can be overridden for each job template and job invocation.                              |
| <b>SSH User</b>                   | Specifies the default user that the Capsule will use while connecting to the target host. This can be a different user from the effective user. |
| <b>Sudo password</b>              | Specifies the sudo password.                                                                                                                    |
| <b>Default SSH password</b>       | Specifies the default password to use for SSH.                                                                                                  |

**Important**

Only change these settings using the web UI. Any manual change to the `/etc/foreman/settings.yml` file are overwritten the next time you run the `satellite-installer` command. Alternatively, use the `foreman-rake config` command from a console.

## Creating a New Job Template

A job template is a powerful tool for easily and consistently running remote commands. It also makes it possible to add additional logic for handling more complex things such as different operating system versions. Satellite ships with a variety of job templates that you can use, for example to run simple remote commands, use Puppet for configuration changes, or Ansible ad hoc commands as well as Ansible Playbooks or roles. Depending on your environment configuration, you might need to create a new job template. You can modify or combine existing job templates or create completely new job templates that suit your needs.

### To Create a New Job Template in the Satellite Web UI:

- Navigate to **Hosts** → **Job templates** and click **New Job Template**.
- Click the **Template** tab and enter a unique name in the **Name** field.
- Select the appropriate organizations and locations that will be able to access your template, or use **Default** to make it available to all your organizations.
- Use the template editor to create the template, or upload it from a file using the **Import** button.
- Click the **Job** tab and select a category from the **Job category** field. Alternatively, create your own category.
- Select the required **Provider Type** from the list. Use **SSH** for shell scripts, and **Ansible** for Ansible-related tasks or playbooks.
- Optionally click **Add Input** to define an input parameter. These parameters are requested when executing the job and they do not have to be defined in the template.
- Click the **Location** tab and choose the locations where this template will be used.
- Click the **Organizations** tab and select the organizations where this template will be used.
- Click **Submit** to create the new job template.

## Using ERB in Job Templates

Red Hat Satellite uses *ERB (Embedded Ruby)* syntax among others in job templates. The default templates included in Satellite provide a good source for the ERB syntax.

When the remote job is running the code in the ERB is executed and the variables are replaced with specific values. This process is called *rendering*. Satellite uses a safe-mode rendering mechanism, which prevents any harmful code from being executed using the templates.

The following summarize the ERB syntax:

`<% %>`

Encloses Ruby code within the ERB template. It is rendered when the template is rendered. It can contain Ruby structure as well as Satellite-specific functions and variables. This example shows how to restart a service depending on the operating system in use:

```
<% if @host.operatingsystem.family == "Redhat" && @host.operatingsystem.major.to_i  
> 6 %>  
systemctl <%= input("action") %> <%= input("service") %>  
<% else %>  
service <%= input("service") %> <%= input("action") %>  
<% end -%>
```

If the host is running a version of RHEL greater than version 6, it should run the **systemctl** command, followed by variables that will be replaced with the specified action and service name. For all other hosts, it will run the **service** command, followed by variables that will be replaced with the specified service name and action.

```
<%= %>
```

The code output is inserted into the template. For example, for variable substitution:

```
echo <%= @host.name %>
```

```
<% -%>, <%= -%>
```

A newline character is inserted after a Ruby block if it is closed at the end of a line.

```
<%# %>
```

Indicates a comment that will be ignored when the template is rendered.

```
<%# This is a comment %>
```



## References

For more information, refer to the *Running Jobs on Hosts* section in the *Managing Hosts* guide at

[https://access.redhat.com/documentation/en-us/red\\_hat\\_satellite/6.6/html/managing\\_hosts](https://access.redhat.com/documentation/en-us/red_hat_satellite/6.6/html/managing_hosts)

## ► Guided Exercise

# Running Remote Jobs on Managed Hosts

In this exercise, you will configure a host for remote execution, prepare an ad hoc job, run it, and confirm a successful completion.

## Outcomes

You should be able to execute commands remotely on a host.

## Before You Begin

Log in as the **student** user on the **workstation** VM, using **student** as the password.

Run the **lab remote-run start** command. This command determines if the **satellite** host is reachable on the network and prepares it for this exercise.

```
[student@workstation ~]$ lab remote-run start
```

- ▶ 1. Distribute SSH keys for remote execution. Depending on your environments configuration you will distribute the SSH keys from your satellite or from your capsule. In the current configuration you must distribute the keys from both the **satellite** and the **capsule** servers.
  - 1.1. On **workstation**, use **ssh** to log in to **capsule** as **student**. Use **sudo -i** to switch to **root**.

```
[student@workstation ~]$ ssh student@capsule
[student@capsule ~]$ sudo -i
[sudo] password for student: student
[root@capsule ~]#
```

- 1.2. Distribute the SSH key manually to the **servera.lab.example.com** host.

Use the **ssh-copy-id** command. The key is located at **~foreman-proxy/.ssh/id\_rsa\_foreman\_proxy.pub**

```
[root@capsule ~]# ssh-copy-id -i \
~foreman-proxy/.ssh/id_rsa_foreman_proxy.pub \
root@servera.lab.example.com
...output omitted...
Are you sure you want to continue connecting (yes/no)? yes
...output omitted...
root@servera.lab.example.com's password: redhat

Number of key(s) added: 1
...output omitted...
```

- ▶ 2. To save significant sync time for this exercise, your Capsule Server's Ansible repositories are not enabled, requiring you to distribute the SSH keys from the Satellite Server. Distribute the SSH key manually to the **servera.lab.example.com** host.
  - 2.1. On **workstation**, use **ssh** to log in to **satellite** as **student**. Use **sudo -i** to switch to **root**.

```
[student@workstation ~]$ ssh student@satellite
[student@satellite ~]$ sudo -i
[sudo] password for student: student
[root@satellite ~]#
```

- 2.2. Use the **ssh-copy-id** command to distribute the SSH key to the **servera.lab.example.com** host.

The key is located at **~foreman-proxy/.ssh/id\_rsa\_foreman\_proxy.pub**.

```
[root@satellite ~]# ssh-copy-id -i \
~foreman-proxy/.ssh/id_rsa_foreman_proxy.pub \
root@servera.lab.example.com
...output omitted...
Are you sure you want to continue connecting (yes/no)? yes
...output omitted...
root@servera.lab.example.com's password: redhat

Number of key(s) added: 1
...output omitted...
```

- ▶ 3. Log in to the Satellite Server web UI located at **https://satellite.lab.example.com** as the **admin** user with the password of **redhat**.
- ▶ 4. Choose the Operations organization and Any Location location from the main menu.
- ▶ 5. Run a remote command on the **servera.lab.example.com** machine.
  - 5.1. Click **Hosts** → **All Hosts**, and click **servera.lab.example.com**.



#### Note

You can run remote jobs on multiple hosts concurrently. Select the check box for each host, and click **Select Action** → **Schedule Remote Job**.

- 5.2. Click **Schedule Remote Job**.
- 5.3. Set the **Job category** to **Commands**, and set **Job template** to **Run Command - SSH Default**.
- 5.4. In the **command** field, enter **uptime; hostname; whoami**.
- 5.5. The **Schedule** defaults to **Execute now**. Click **Submit** to execute the remote command. On the **Overview** tab, monitor the remote execution status. Wait for it to successfully finish.
- ▶ 6. To view the remote command output, scroll down and click the **servera.lab.example.com** link. Output will be similar to the following:

```
1: 15:30:22 up 2 days, 1 user, load average: 0.00, 0.00, 0.00
2: servera.lab.example.com
3: root
4: Exit status: 0
```

- 7. Run the same remote execution from a command line on **satellite.lab.example.com**.

- 7.1. On **workstation**, **ssh** to **satellite** as **student**. Use **sudo -i** to switch to **root**.

```
[student@workstation ~]$ ssh student@satellite
[student@satellite ~]$ sudo -i
[sudo] password for student: student
[root@satellite ~]#
```

- 7.2. Use the **hammer job-invocation create** command to run a new remote execution command.

```
[root@satellite ~]# hammer job-invocation create \
--job-template "Run Command - SSH Default" \
--search-query "name = servera.lab.example.com" \
--inputs command="uptime; hostname; whoami"
...output omitted...
```

- 8. Use the **hammer job-invocation output** command to display the output. Replace **ID NUMBER** with the ID shown by the output of the previous command.

```
[root@satellite ~]# hammer job-invocation output \
--id ID NUMBER --host servera.lab.example.com
```

- 9. Run a remote Ansible ad hoc command on **servera.lab.example.com**.

- 9.1. Click **Hosts** → **All Hosts**, and click the **servera.lab.example.com** link.

- 9.2. Click **Schedule Remote Job**.

- 9.3. Ensure that **Job category** is set to **Ansible Commands**, and **Job template** is set to **Run Command - Ansible Default**.

- 9.4. Enter **df -h** in the **command** field. This command displays the storage usage on **servera.lab.example.com**.



**Note**

You can execute the command immediately, schedule it to run in the future, or set up recurring execution.

- 9.5. Click **Submit** to execute the remote command. A page displays to show the remote execution status. Wait for it to successfully finish.

- 10. To see the output of the remotely executed Ansible ad hoc command, click the **servera.lab.example.com** link. You should see output similar to the following:

```
PLAY [all] *****
TASK [Gathering Facts] *****
TASK [shell] *****
...output omitted...
```

- 11. Run the same Ansible remote execution from a command line on the **satellite.lab.example.com** server.

11.1. On **workstation**, ssh to **satellite** as **student**. Use **sudo -i** to switch to **root**.

```
[student@workstation ~]$ ssh student@satellite
[student@satellite ~]$ sudo -i
[sudo] password for student: student
[root@satellite ~]#
```

11.2. Use the **hammer job-invocation create** command to start a run a new Ansible remote execution command.

```
[root@satellite ~]# hammer job-invocation create \
--job-template "Run Command - Ansible Default" \
--search-query "name = servera.lab.example.com" \
--inputs command="df -h"
...output omitted...
```

- 12. Use the **hammer job-invocation output** command to see the output. Replace the ID NUMBER with the ID shown by the output of the previous command.

```
[root@satellite ~]# hammer job-invocation output \
--id ID NUMBER --host servera.lab.example.com
```

- 13. Use the provided playbook example to create a new job template.

13.1. Click **Hosts** → **Job templates** and click **New Job Template**.

13.2. Enter **My new custom banner** in the **Name** field.

13.3. In the template editor, copy and paste the contents of the **/home/student/playbook-example.yml** file on **workstation**.



#### Note

This playbook line replaces the default banner message with a custom message when the job template is executed:

```
...output omitted...
<%= input('banner_var') %>
...output omitted...
```

- 13.4. Click the **Inputs** tab and then click **Add Input**.
  - 13.5. Enter **banner\_var** in the **Name** field. This field corresponds to the variable name in your playbook.
  - 13.6. Click the **Job** tab. Remove the existing job category and choose **Ansible Playbook**.
  - 13.7. Change the **Provider Type** to **Ansible**, and then click **Submit** to save the job template.
- 14. Use the new job template to customize the SSH banner on **servera.lab.example.com** host.
- 14.1. Click **Hosts** → **All Hosts**, and then click the **servera.lab.example.com** link.
  - 14.2. Click **Schedule Remote Job**.
  - 14.3. Change the **Job category** to **Ansible Playbook** and the **Job template** to the new **My new custom banner** template.
  - 14.4. Enter **Welcome to my new customized server** in the **banner\_var** field, and then click **Submit**.  
On the next page, click **servera.lab.example.com** and observe the Ansible Playbook execution output.

- 15. Verify that the SSH banner has been replaced with your custom message.
- 15.1. On **workstation**, use **ssh** to log in to **servera** as **student**.

```
[student@workstation ~]$ ssh student@servera
Welcome to my new customized server
...output omitted...
[student@servera ~]$
```

- 16. Log off from **satellite** and **servera** hosts and return to **workstation**.

## Finish

On the **workstation** machine, use the **lab remote-run finish** command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab remote-run finish
```

This concludes the guided exercise.

# Configuring Ansible Remote Execution

## Objectives

After completing this section, you should be able to configure and enable remote execution and install additional Ansible Roles on a Satellite Capsule Server.

## Customizing Roles with Ansible in Satellite

Satellite supports both Ansible Playbooks and Ansible Roles. There are multiple ways to use Ansible with Satellite. You can create custom roles or use external roles available from the Ansible Galaxy website. You can also use Red Hat Enterprise Linux system roles to automate common RHEL subsystems.

To use Ansible with Satellite to manage and use custom Ansible Roles requires a valid Ansible subscription.



### Important

To use Ansible with Satellite, you must enable and configure the remote execution feature, and distribute the correct SSH key to each managed host. This is discussed in the Running Remote Jobs on Managed Hosts section.

## Characteristics of Ansible Roles

Ansible Roles make it easier to reuse generic Ansible code. You can package the tasks, variables, files, templates, and other resources to provision infrastructure or deploy applications into a standardized directory structure. You can copy roles from project to project by copying their directory. You can then execute that role from a playbook.

Reusable, modular roles use passed-in playbook variables. Variables modify role behavior by setting site-specific host names, IP addresses, user names, secrets, and other local-specific details. For example, a role to deploy a database could use variables for the host name, database admin user and password, and other environment-specific parameters. The role author can set reasonable default variables values to work with playbooks that do not set the variables.

Ansible Roles have the following benefits:

- Allows easy sharing of code by grouping content.
- Defines essential elements of a system type: web server, database server, Git repository, or other purpose.
- Makes larger projects more manageable.
- Allows parallel development by different administrators.

## Ansible Role Structure

An Ansible Role is defined by a standardized structure of subdirectories and files. The top-level directory defines the name of the role itself. Files are organized into subdirectories that are named according to each file's purpose in the role, such as **tasks** and **handlers**. The **files** and **templates** subdirectories contain files referenced by tasks in other YAML files.

The following `tree` command displays the directory structure of the `user.example` role.

```
[user@host roles]$ tree user.example
user.example/
├── defaults
│   └── main.yml
├── files
├── handlers
│   └── main.yml
├── meta
│   └── main.yml
├── README.md
├── tasks
│   └── main.yml
├── templates
├── tests
│   ├── inventory
│   └── test.yml
└── vars
    └── main.yml
```

### Ansible Role Subdirectories

| Subdirectory     | Function                                                                                                                                                                                                                                            |
|------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>defaults</b>  | The <code>main.yml</code> file in this directory contains the default values of role variables that can be overwritten when the role is used. These variables have low precedence and are intended to be changed and customized in plays.           |
| <b>files</b>     | This directory contains static files that are referenced by role tasks.                                                                                                                                                                             |
| <b>handlers</b>  | The <code>main.yml</code> file in this directory contains the role's handler definitions.                                                                                                                                                           |
| <b>meta</b>      | The <code>main.yml</code> file in this directory contains information about the role, including author, license, platforms, and optional role dependencies.                                                                                         |
| <b>tasks</b>     | The <code>main.yml</code> file in this directory contains the role's task definitions.                                                                                                                                                              |
| <b>templates</b> | This directory contains Jinja2 templates that are referenced by role tasks.                                                                                                                                                                         |
| <b>tests</b>     | This directory can contain an inventory and <code>test.yml</code> playbook that can be used to test the role.                                                                                                                                       |
| <b>vars</b>      | The <code>main.yml</code> file in this directory defines the role's variable values. Often these variables are used for internal role purposes. These variables have high precedence, and are intended to remain unchanged when used in a playbook. |

Not every role will have all of these directories.

## Defining Variables and Defaults

Role variables are defined in a `vars/main.yml` file using key-value pairs, in the role directory hierarchy. They are referenced in the role YAML file like other variables: `{{ VAR_NAME }}`. These variables have a high precedence and can not be overridden by inventory variables. The intent of these variables is to be used as internal role functioning.

Creating *default variables* sets initial values for variables that configures a role or customizes its behavior. They are defined in the **defaults/main.yml** file with key: value pairs in the role directory hierarchy. Default variables have the lowest precedence of available variables. They can be overridden by other variables, including inventory variables. These variables are intended for a playbook author to customize or control how a role or playbook behaves. Variables can be used to provide information to the role to configure or deploy objects correctly.

Use either **vars/main.yml** or **defaults/main.yml** to define a specific variable, but not both. Default variables are used when it is intended that their values will be overridden.



### Important

Roles should not contain site-specific data. They should not contain secrets such as passwords or private keys. Roles are intended to be generic, reusable, and freely shareable. Site-specific details should not be hard-coded into roles.

Secrets should be provided to the role through other means, such as passed-in role variables. Role variables set in the playbook could provide the secret, or point to an Ansible Vault-encrypted file containing the secret.

## Red Hat Enterprise Linux System Roles

Beginning with Red Hat Enterprise Linux 7.4, a number of Ansible Roles have been provided with the operating system as part of the *rhel-system-roles* package. In Red Hat Enterprise Linux 8 the package is available in the AppStream channel. A brief description of each role is provided below:

### RHEL System Roles

| Name                              | State              | Description                                                                                                                        |
|-----------------------------------|--------------------|------------------------------------------------------------------------------------------------------------------------------------|
| <b>rhel-system-roles.kdump</b>    | Fully Supported    | Configures the <b>kdump</b> crash recovery service.                                                                                |
| <b>rhel-system-roles.network</b>  | Fully Supported    | Configures network interfaces.                                                                                                     |
| <b>rhel-system-roles.selinux</b>  | Fully Supported    | Configures and manages SELinux customization, including SELinux mode, file and port contexts, Boolean settings, and SELinux users. |
| <b>rhel-system-roles.timesync</b> | Fully Supported    | Configures time synchronization using Network Time Protocol or Precision Time Protocol.                                            |
| <b>rhel-system-roles.postfix</b>  | Technology Preview | Configures each host as a Mail Transfer Agent using the Postfix service.                                                           |
| <b>rhel-system-roles.firewall</b> | In Development     | Configures a host's firewall.                                                                                                      |
| <b>rhel-system-roles.tuned</b>    | In Development     | Configures the <b>tuned</b> service to tune system performance.                                                                    |

System roles aim to standardize the configuration of Red Hat Enterprise Linux subsystems across multiple versions. System roles are available for Red Hat Enterprise Linux 6.10 and later.

## Simplified Configuration Management

As an example, the recommended time synchronization service for Red Hat Enterprise Linux 7 is the **chrony** service. In Red Hat Enterprise Linux 6, however, the recommended service is the **ntpd** service. In an environment with a mixture of Red Hat Enterprise Linux 6 and 7 hosts, you need to manage the configuration files for both services.

With RHEL system roles, you no longer need to maintain configuration files for both services. You can use the **rhel-system-roles.timesync** role to configure time synchronization for both Red Hat Enterprise Linux 6 and 7 hosts. A simplified YAML file containing role variables defines the configuration of time synchronization for both types of hosts.

## Support for RHEL System Roles

RHEL system roles are derived from the open source Linux System Roles project, found on Ansible Galaxy. Unlike Linux system roles, RHEL system roles are supported by Red Hat as part of a standard Red Hat Enterprise Linux subscription. RHEL system roles have the same life cycle support benefits that come with a Red Hat Enterprise Linux subscription.

Every system role is tested and stable. The **Fully Supported** system roles also have stable interfaces. For any **Fully Supported** system role, Red Hat will endeavor to ensure that role variables are unchanged in future versions. Playbook refactoring due to system role changes should be minimal.

The **Technology Preview** system roles may utilize different role variables in future versions. Integration testing is recommended for playbooks that incorporate any **Technology Preview** role. Playbooks may require refactoring if role variables change in a future version of the role.

Other roles are in development in the upstream Linux System Roles project, but are not yet available through a RHEL subscription. These roles are available through Ansible Galaxy.

## Importing Ansible Roles into Satellite

Ansible is enabled by default on Satellite and Capsule. You can import Ansible Roles and Red Hat Enterprise Linux system roles to help you automate the routine tasks.



### Important

All of your customized or third-party Ansible Roles must be stored in the **/etc/ansible/roles** directory of the Satellite or Capsule, depending on where you want to use them. Before you can use them, you must import the roles into Satellite Server from the **/etc/ansible/roles** directory.

### To Import Ansible Roles:

- Copy the custom Ansible Role to the **/etc/ansible/roles** directory on your Satellite or Capsule server.
- Navigate to **Configure** → **Roles** and click the Capsule or Satellite Server that contains the roles you want to import.
- From the list of available Ansible Roles, select those that you want to import, and click **Update**.

| <input checked="" type="checkbox"/> | Name                           | Hosts count | Hostgroups count | Operation |
|-------------------------------------|--------------------------------|-------------|------------------|-----------|
| <input type="checkbox"/>            | RedHatInsights.insights-client | 0           | 0                | Add       |
| <input type="checkbox"/>            | theforeman.foreman_scap_client | 0           | 0                | Add       |

Cancel **Update**

Figure 7.6: Importing Ansible Roles

## Importing Red Hat Enterprise Linux System Roles

You can install and import Red Hat Enterprise Linux system roles in Satellite to make the configuration of managed hosts faster and easier.



### Note

For more information about Red Hat Enterprise Linux system roles refer to this article at <https://access.redhat.com/articles/3050101>

Satellite 6.6 runs on Red Hat Enterprise Linux 7. Consequently, you need to enable the **Extras** repository and install the *rhel-system-roles* package from there.

#### To Install the RHEL System Roles Package on Satellite or Capsule:

- Enable the **rhel-7-server-extras-rpms** repository.

```
[root@demo_capsule ~]# subscription-manager repos \
--enable=rhel-7-server-extras-rpms
```

- Install the *rhel-system-roles* package.

```
[root@demo_capsule ~]# satellite-maintain packages install rhel-system-roles
```

This installs the roles to the **/usr/share/ansible/roles** directory. You can review and modify the roles there before you import them.

- Navigate to **Configure** → **Roles** and click the Capsule or Satellite Server that contains the roles you want to import.
- From the list of available Ansible Roles, select those that you want to import, and click **Update**.

## Importing Ansible Variables

To refine the configuration of systems that have specific requirements, Ansible Roles use variables. To use variables in your Ansible Playbooks or roles, first import the variables to your Satellite or Capsule.

You can import variables from Ansible Roles that you have already imported into your Satellite.

### To Import Variables from Existing Ansible Roles:

- Navigate to **Configure** → **Variables** and click the Capsule or Satellite Server that contains the roles with variables to import.
- Select the Ansible variable to import, and click **Update**.

| <input checked="" type="checkbox"/> | Name                           | Ansible role      | Hosts count | Hostgroups count | Operation |
|-------------------------------------|--------------------------------|-------------------|-------------|------------------|-----------|
| <input checked="" type="checkbox"/> | apache_max_keep_alive_requests | apache-setup-role | 0           | 0                | Add       |

Cancel **Update**

Figure 7.7: Importing Ansible variables

## Creating Ansible Variables

Typically, you can import Ansible variables from the Ansible Roles that you are using. You might want to further refine the configuration of a system. To do this, you can create Ansible variables directly in Satellite.

### To Create an Ansible Variable in Satellite:

- Navigate to **Configure** → **Variables** and click **New Ansible Variable**.
- In the **Key** field, enter the name of the variable. If desired, add a description in the **Description** field.
- From the **Ansible Role** list, select the appropriate Ansible Role to associate with the variable.
- Optionally, you can override the variable with Satellite. If required, select **Override** to allow Satellite to manage the variable. When overriding, select the value parameter for validation from the **Parameter Type**.
- If required, enter a default value to use for the variable in the **Default Value** field.

- Click **Submit** to create the variable.

The screenshot shows the 'Ansible Variables > Create Ansible Variable' page. It has several sections:

- Ansible Variable Details:** Fields include 'Key \*' (variable\_name), 'Description' (empty), and 'Ansible Role \*' (apache-setup-role) with a dropdown menu.
- Default Behavior:** Includes 'Override' (checked), 'Parameter Type' (string), 'Default Value' (empty), and 'Hidden Value' (unchecked).
- Optional Input Validator:** A link to a separate section.
- Prioritize Attribute Order:** A note about resolution order followed by a 'Order' dropdown containing 'fqdn', 'hostgroup', 'os', and 'domain'.

Figure 7.8: Creating Ansible variables

## Assigning Ansible Roles

After importing all the Ansible Roles and variables into your Satellite, use the roles for remote management of your RHEL systems. RHEL versions starting with 6.9 support this feature.

Ansible Roles can be assigned to a single host or to a host group. Creating and using host groups is covered later in this course. After you assign an Ansible Role to a host, you can use Ansible for remote execution.

### To Assign an Imported Ansible Role to a Host:

- Navigate to **Hosts** → **All Hosts**.
- From the list of available hosts click the name of the host to which to assign the Ansible Role. On the details page for the host, click **Edit**.
- Click the **Ansible Roles** tab to display all available Satellite Ansible Roles.
- To assign a new role to your host, click the plus sign (+) next to the Ansible Role to assign. This moves the role from the **Available Ansible Roles** list to the **Assigned Ansible Roles** list for that host. Similarly, if you want to remove a role from a host, click the minus sign (-) next to the Ansible Role you want to remove.
- Click **Submit**.

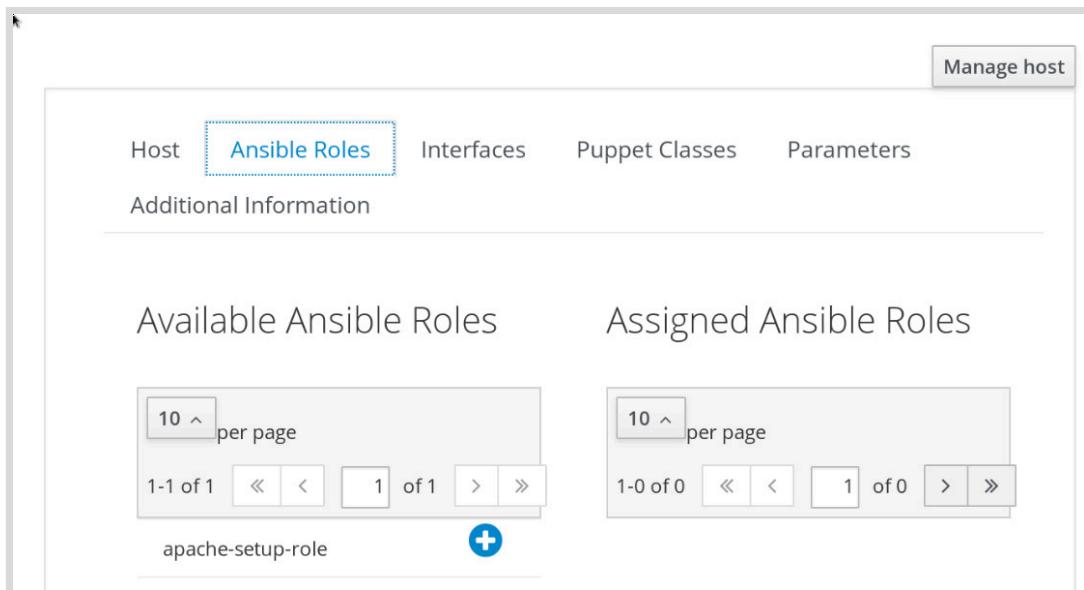


Figure 7.9: Assigning Ansible Roles

## Running Ansible Roles on a Host

With a role imported into Satellite and assigned to a host, it is available to run on that host.

To be able to run an Ansible Role on a host you must ensure that:

- The Ansible Role has been imported into Satellite.
- The imported Ansible Role has been assigned to a host.

### To Run an Ansible Role on a Host:

- Navigate to **Hosts → All Hosts**.
- Select the check box for the host that contains the Ansible Role to run.
- Click **Select Action** and choose **Play Ansible roles**. On the next page, you can monitor progress of the remote execution of the Ansible Role on that host.



### References

For more information, refer to the *Managing Ansible Roles* chapters in the *Administering Red Hat Satellite* guide at  
[https://access.redhat.com/documentation/en-us/red\\_hat\\_satellite/6.6/html-single/administering\\_red\\_hat\\_satellite](https://access.redhat.com/documentation/en-us/red_hat_satellite/6.6/html-single/administering_red_hat_satellite)

For more information, refer to the *Using Ansible Roles* chapters in the *Managing Hosts* guide at  
[https://access.redhat.com/documentation/en-us/red\\_hat\\_satellite/6.6/html-single/managing\\_hosts](https://access.redhat.com/documentation/en-us/red_hat_satellite/6.6/html-single/managing_hosts)

## ► Guided Exercise

# Configuring Ansible Remote Execution

In this exercise, you will enable the remote execution feature, install Ansible Roles on Satellite Capsule Server, and import the roles into Satellite Server.

### Outcomes

You should be able to enable the remote execution feature and install additional Ansible Roles on Satellite Capsule Server.

### Before You Begin

Log in as the **student** user on **workstation** using **student** as the password.

Run the **lab remote-config start** command. This command determines if the **satellite** host is reachable on the network and verifies that Red Hat Satellite is available.

```
[student@workstation ~]$ lab remote-config start
```

- 1. On **workstation**, use **ssh** to log in to **satellite** as **student**. Use **sudo -i** to switch to **root**.

```
[student@workstation ~]$ ssh student@satellite
[student@satellite ~]$ sudo -i
[sudo] password for student: student
[root@satellite ~]#
```

- 2. In the following steps, you create a new custom Ansible Role on Satellite Server. You will import this role and use it later to customize a server in your environment.

- 2.1. Extract the precreated **ansible-role.tgz** file.

```
[root@satellite ~]# tar xzvf ansible-role.tgz
```

- 2.2. Create the **motd** role directory and subdirectories in the **/etc/ansible/roles/** directory:

```
[root@satellite ~]# mkdir -p /etc/ansible/roles/motd/tasks
[root@satellite ~]# mkdir -p /etc/ansible/roles/motd/templates
```

- 2.3. Copy the **main.yaml** file to the **/etc/ansible/roles/motd/tasks** directory.  
Copy the **motd.j2** Jinja2 template to the **/etc/ansible/roles/motd/templates** directory.

```
[root@satellite ~]# cp main.yaml /etc/ansible/roles/motd/tasks
[root@satellite ~]# cp motd.j2 /etc/ansible/roles/motd/templates
```

- ▶ 3. Log in to the Satellite web UI as **admin** using **redhat** as the password.
  - 3.1. Navigate to <https://satellite.lab.example.com> and log in as **admin** using **redhat** as the password.
  - 3.2. Choose the **Operations** organization and **Any Location** location from the main menu.
- ▶ 4. Import the **motd** Ansible Role into Satellite.
  - 4.1. Click **Configure → Roles** and click **Import from satellite.lab.example.com** to import your new custom role.
  - 4.2. From the list of available Ansible Roles, select the check box for the **motd** role and click **Update** to import the role.
- ▶ 5. Create a new Ansible variable that will be used by your **motd** role.
  - 5.1. Click **Configure → Variables** and then click **New Ansible Variable** to create a new variable.
  - 5.2. Enter **new\_var** in the **Key** field. This is the name of the variable, which is defined in the **motd.j2** Jinja2 template inside your role.
  - 5.3. Click the **Ansible Role** list and choose the **motd** role.
  - 5.4. Select **Override**
  - 5.5. Enter **Hello from the RH403 course** in the **Default Value** field.
  - 5.6. Click **Submit** to create the variable.
- ▶ 6. Assign the new role to **servera.lab.example.com** server.
  - 6.1. Click **Hosts → All Hosts** and then click the **servera.lab.example.com** host.
  - 6.2. Click **Edit**, and then click the **Ansible Roles** tab.
  - 6.3. Click the plus sign (+) next to the **motd** role to assign the role to the **servera.lab.example.com** server.
  - 6.4. Click **Submit**.
- ▶ 7. Use the new **motd** role to customize the **servera.lab.example.com** server.
  - 7.1. Click **Hosts → All Hosts** and select the check box for **servera.lab.example.com**.
  - 7.2. Click **Select Action** and choose **Play Ansible roles**.  
This takes you to a page that shows the status of the remote execution. Wait for it to successfully finish.
- ▶ 8. From **workstation**, log in to the **servera.lab.example.com** and verify that the **/etc/motd** file was customized by Satellite using your new role.

```
[student@workstation ~]$ ssh student@servera
Welcome to my new customized server.
This is an example /etc/motd created during the RH403 course using Ansible/
Satellite.

Here is an example of the test_var variable being passed from Satellite: Hello
from the RH403 course
...output omitted...
```

- ▶ **9.** Log off from the **servera** and **satellite** hosts and return to **workstation**.
- ▶ **10.** Log out from the Satellite web UI.

## Finish

On the **workstation** machine, use the **lab remote-config finish** command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab remote-config finish
```

This concludes the guided exercise.

# Running Remote Puppet Jobs on Managed Hosts

## Objectives

After completing this section, you should be able to:

- Create a content view that supports Puppet configuration.
- Create an activation key that enables Puppet configuration.
- Register a Puppet agent to a Red Hat Satellite Server.

## Installing and Configuring the Puppet Agent

Red Hat Satellite 6.6 also supports Puppet. You can upload Puppet modules to Red Hat Satellite Server or Capsule. The Puppet agent software needs to be deployed and configured on host systems to work with Satellite Server, which acts as the Puppet master.

### To Configure the Puppet Agent:

1. Define a content view that provides the necessary packages and Puppet modules.
2. Register the host with Red Hat Satellite.
3. Install the Puppet agent software.
4. Sign the agent host certificate.
5. Launch the Puppet agent.

## Define a Content View with Puppet Modules

Satellite content views determine which software packages are available for installation on hosts. Because Puppet may install software packages, the content view associated with a system should provide the necessary packages, including dependencies, that Puppet will use to configure services on a system. A content view does not determine how a system is configured; instead it provides access to the packages needed by Puppet to configure a system. The key word concerning content views is availability.

To use Puppet modules in Satellite, you need to create a new **Product** and associate that product with a Puppet repository into which you upload the needed Puppet modules.

Another chapter in this course describes how to create a new product, and how to define a content view that provides access to software packages. This section will look at how to define content views so that they can support Puppet configuration. Such a content view must provide access to the **puppet** package and its dependencies. These packages are provided by the **Red Hat Satellite Tools 6.x** and the **Red Hat Enterprise Linux** Yum repositories at a minimum.

You should also consider the additional software needed by Puppet classes. Puppet classes can include manifests that install additional software. For example, a Puppet class that configures a web server could install the *httpd* package and its dependencies.



### Important

Be careful when creating content filters. Puppet classes must have access to required packages to configure managed hosts to provide functionality.

Content views determine which Puppet modules hosts can access. Modules are made available but not activated when defined in a content view. When editing a content view selected from the **Content → Content View** page, click the **Puppet Modules** tab. Click **Add New Module** to publish previously loaded modules in Satellite Server to hosts. After creating a content view, publish it and promote it to the software environment where the managed host is registered.

After a content view has been created, published, and promoted, Red Hat Satellite creates a Puppet environment for that content view in each of the life-cycle environments in the organization. Puppet environment names have the following structure, where *ORG* is the organization name, *ENV* is the life-cycle environment, *VIEW* is the content view name, and *#* is an internal sequence number:

```
KT_ORG_ENV_VIEW_#
```

Some menu items will not be active when assigning a host group to a host if it has not been assigned a Puppet environment.

## Register a Host with Red Hat Satellite

After creating a content view for the Puppet host, register the host to associate it with that content view. Use an activation key to automate this process. When creating an activation key, associate the hosts to the environment and content view providing the required Puppet modules.

### To Register a Managed Host to Satellite with an Activation Key:

- Choose the appropriate organization, and navigate to **Content → Activation Keys**.
- Click an existing activation key or click **New Activation Key**. This displays a page where you can choose the required software environment and content view.
- Install the CA certificate RPM that signs the Satellite Server host certificate. Then, use **subscription-manager** and the activation key to register the system to an organization.

```
[root@demo_host ~]# yum localinstall \
http://SATELLITE.FQDN/pub/katello-ca-consumer-latest.noarch.rpm
[root@demo_host ~]# subscription-manager register --org=ORG --activationkey='KEY'
```

## Install the Puppet Agent Software

Install the Puppet agent software after the host is registered and can access appropriate repositories. The packages could be installed in advance when building VM base images.

### To Install the Puppet Agent:

- Use the **yum** command to install the Puppet agent. The *katello-agent* package is usually installed when you register the managed host.

```
[root@demo_host ~]# yum install puppet katello-agent
```

- Modify the primary agent configuration file so the Puppet agent uses the correct server as a Puppet master. Add the following line with your server FQDN to **/etc/puppet/puppet.conf**:

```
server=satellite.FQDN
```

## Sign the Agent Host Certificate

Finishing the host registration transaction takes two additional steps: connect with the Puppet master, and sign the Puppet agent host certificate.

### To Sign the Agent Host Certificate:

- Use the **puppet** command have the Puppet agent contact the Puppet master.

```
[root@demo_host ~]# puppet agent --test --noop
```

The Puppet agent sends a host certificate to the Puppet master to allow for secure communications. The **--noop** option keeps the agent from trying to apply changes suggested by the Puppet master. The **puppet --waitforcert=DELAY** option is also useful. It causes the Puppet agent to connect to the master every *DELAY* seconds and ask it to sign its host certificate request. The default delay value is 120 seconds.

- Log in to the Satellite web UI and sign the host certificate presented by the agent. Set the organization context to **Any Organization**, and navigate to **Infrastructure → Capsules**.
- In the **Actions** column for the Capsule Server host name, click the down arrow next to the **Edit** button and then click **Certificates** to display a list of host certificates.
- Click **Sign** at the right of the host name to allow it to connect to Satellite Server.

You can configure Satellite to auto-sign Puppet host certificates. When the list of host certificates for a Capsule Server displays, click **Autosign Entries** that appears above the list. You can use this feature to create host name entries (which can include wildcards) that will have their host certificates signed automatically when they first connect to Satellite Server.



### Warning

Configuring Satellite to auto-sign Puppet host certificates creates a security risk. In this case, any host can connect and request Puppet manifests (which may contain privileged information, such as passwords, shared keys, and certificates).

## Launch the Puppet Agent

Use the **systemctl** command to start daemons and services immediately and persistently. The following commands start and enable the Puppet agent as a daemon on a Red Hat Enterprise Linux 8 system:

```
[root@demo_host ~]# systemctl start puppet.service
[root@demo_host ~]# systemctl enable puppet.service
ln -s '/usr/lib/systemd/system/puppet.service' '/etc/systemd/multi-
user.target.wants/puppet.service'
```



### References

For more information, refer to the *Puppet Guide* at  
[https://access.redhat.com/documentation/en-us/red\\_hat\\_satellite/6.6/html/puppet\\_guide](https://access.redhat.com/documentation/en-us/red_hat_satellite/6.6/html/puppet_guide)

## ► Guided Exercise

# Running Remote Puppet Jobs on Managed Hosts

In this exercise, you will remotely install the Puppet agent on a managed host, and use it to apply the latest configuration version.

## Outcomes

You should be able to execute commands remotely on a host and install the Puppet agent.

## Before You Begin

Log in as the **student** user on the **workstation** VM, using **student** as the password.

Run the **lab remote-puppet start** command. This command determines if the **satellite** host is reachable on the network and prepares it for this exercise.

```
[student@workstation ~]$ lab remote-puppet start
```

- 1. Distribute SSH keys for remote execution to the **serverb.lab.example.com** host.

- 1.1. On **workstation**, ssh to **capsule** as **student**. Use **sudo -i** to switch to **root**.

```
[student@workstation ~]$ ssh student@capsule
[student@capsule ~]$ sudo -i
[sudo] password for student: student
[root@capsule ~]#
```

- 1.2. Distribute the SSH key manually to the **serverb.lab.example.com** host. Use the **ssh-copy-id** command. The key is located at **~foreman-proxy/.ssh/id\_rsa\_foreman\_proxy.pub**

```
[root@capsule ~]# ssh-copy-id -i \
~foreman-proxy/.ssh/id_rsa_foreman_proxy.pub \
root@serverb.lab.example.com
...output omitted...
Are you sure you want to continue connecting (yes/no)? yes
...output omitted...
root@serverb.lab.example.com's password: redhat

Number of key(s) added: 1
...output omitted...
```

- 1.3. Log out from **capsule**.

- 2. Move **serverb.lab.example.com** to the **QA** life-cycle.

- 2.1. Log in to the Satellite Server web UI located at `https://satellite.lab.example.com` as **admin** using **redhat** as the password.
- 2.2. Choose the **Operations** organization and **Any Location** location.
- 2.3. Click **Hosts → Content Hosts** and then click the **serverb.lab.example.com** link.
- 2.4. On the hosts **Details** page mark the checkbox next to the **QA** life-cycle environment. Click **Save**.

- 3. Update the **serverb.lab.example.com** server to RHEL 8.1.
- 3.1. On **workstation**, use **ssh** to log in to **serverb** as **student**. Use **sudo -i** to switch to **root**.

```
[student@workstation ~]$ ssh student@serverb
[student@serverb ~]$ sudo -i
[sudo] password for student: student
[root@serverb ~]#
```

- 3.2. Use the **yum** command to update the operating system. Wait for the update to finish.

```
[root@serverb ~]# yum update
...output omitted...
Is this ok [y/N]: y
...output omitted...
```

- 4. Remotely install the puppet agent on the **serverb.lab.example.com** server.
- 4.1. Click **Hosts → All Hosts** and then click the **serverb.lab.example.com** link.
  - 4.2. Click **Schedule Remote Job**.
  - 4.3. Set **Job category** to **Packages** and set **Job template** to **Package Action - SSH Default**.
  - 4.4. Verify that **action** is set to **install**.
  - 4.5. In the **package** field, enter **puppet**, and then click **Submit** to execute the remote command.  
Monitor the status of the remote execution on the **Overview** page that displays. Wait for it to successfully finish.
- 5. To see the output of the remote command, click the **serverb.lab.example.com** link.

```
...output omitted...
10: Installing:
11:   puppet-agent
...output omitted...
```

- 6. Remotely enable the Puppet agent on **serverb.lab.example.com** server.

- 6.1. Click **Hosts → All Hosts** and then click the **serverb.lab.example.com** link.

6.2. Click **Schedule Remote Job**.

6.3. Set **Job category** to **Puppet** and set **Job template** to **Puppet Agent Enable - SSH Default**.

6.4. Click **Submit** to execute the remote command.

Monitor the status of the remote execution on the **Overview** tab that displays. Wait for it to successfully finish.

- 7. Remotely apply the latest Puppet configuration version on **serverb.lab.example.com** server. This job will fail because you must first sign the Puppet CA certificate for **serverb.lab.example.com** host.

7.1. Click **Hosts → All Hosts**, and then click the **serverb.lab.example.com** link.

7.2. Click **Schedule Remote Job**.

7.3. Set **Job category** to **Puppet** and set **Job template** to **Puppet Run Once - SSH Default**.

7.4. In the **puppet\_options** field, enter **-t --server satellite.lab.example.com**.

```
-t --server satellite.lab.example.com
```

7.5. Click **Submit** to execute the remote command.

Monitor the status of the remote execution on the **Overview** tab that displays. Wait for it to fail because of the certificate not being signed.

```
1: Info Creating a new SSL key for serverb.lab.example.com
...output omitted...
7: Exiting: no certificate found and waitforcert is disabled
8: Exit status: 1
```

- 8. Sign the **serverb.lab.example.com** puppet certificate.

8.1. Click **Infrastructure → Capsules** and then click the **satellite.lab.example.com** link.

8.2. Click **Puppet CA** and then click **Certificates**.

8.3. In the **Actions** column, click **Sign** in the line with the **serverb.lab.example.com** host.

- 9. Remotely apply the latest Puppet configuration version on the **serverb.lab.example.com** server. With the signed certificate, the job will successfully finish this time.

9.1. Click **Hosts → All Hosts** and then click the **serverb.lab.example.com** link.

9.2. Click **Schedule Remote Job**.

9.3. Set **Job category** to **Puppet** and set **Job template** to **Puppet Run Once - SSH Default**.

9.4. In the **puppet\_options** field, enter **-t --server satellite.lab.example.com**.

```
-t --server satellite.lab.example.com
```

- 9.5. Click **Submit** to execute the remote command.

Monitor the status of the remote execution on the **Overview** tab that displays. Wait for it to successfully finish.

- 10. To see the output of the remote command, click the **serverb.lab.example.com** link.

```
1: Info: Caching certificate for serverb.lab.example.com
...output omitted...
200: Notice: Applied catalog in 0.03 seconds
201: Exit status: 0
```

- 11. Log off from the **satellite** host and return to **workstation**.

## Finish

On the **workstation** machine, use the **lab remote-config finish** command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab remote-puppet finish
```

This concludes the guided exercise.

## ► Lab

# Running Remote Execution

### Performance Checklist

In this lab, you will configure a host for remote execution, import a new Ansible Role, create a new variable in Satellite, and use the new role to install and configure Apache remotely.

### Outcomes

You should be able to:

- Configure a host for remote execution.
- Import a new Ansible Role.
- Create new variables in Satellite.
- Play the new role remotely on two servers to install and configure Apache.

### Before You Begin

Log in to **workstation** as **student** using **student** as the password.

On **workstation**, run the **lab remote-review start** command. The command determines if the host, **satellite**, is reachable on the network and verifies Red Hat Satellite is available.

```
[student@workstation ~]$ lab remote-review start
```

1. Distribute the SSH keys for remote execution from **satellite.lab.example.com** to **serverb.lab.example.com**.
2. Create a new custom Ansible Role on Satellite Server. Extract and copy the content from the precreated **/root/apache-setup-role.tgz** archive to the **/etc/ansible/roles** directory to create the new role on Satellite Server.
3. Import the **apache-setup-role** Ansible Role into Satellite. Use the **Operations** organization and make it available for any location.
4. Create a new **apache\_test\_message** Ansible variable in the **apache-setup-role** Ansible role, that overrides the default value for that variable with **This variable has been set by Satellite**.
5. Assign the new role to the **servera.lab.example.com** and **serverb.lab.example.com** servers.
6. Use the new **apache-setup-role** role to customize the **servera.lab.example.com** and **serverb.lab.example.com** servers.
7. Use your browser on **workstation** to verify that Apache has been successfully deployed on both **servera.lab.example.com** and **serverb.lab.example.com**.

## Evaluation

As the **student** user on the **workstation** machine, use the **lab remote-review grade** command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab remote-review grade
```

## Finish

As the **student** user on the **workstation** machine, use the **lab remote-review finish** command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab remote-review finish
```

This concludes the lab.

## ► Solution

# Running Remote Execution

### Performance Checklist

In this lab, you will configure a host for remote execution, import a new Ansible Role, create a new variable in Satellite, and use the new role to install and configure Apache remotely.

### Outcomes

You should be able to:

- Configure a host for remote execution.
- Import a new Ansible Role.
- Create new variables in Satellite.
- Play the new role remotely on two servers to install and configure Apache.

### Before You Begin

Log in to **workstation** as **student** using **student** as the password.

On **workstation**, run the **lab remote-review start** command. The command determines if the host, **satellite**, is reachable on the network and verifies Red Hat Satellite is available.

```
[student@workstation ~]$ lab remote-review start
```

1. Distribute the SSH keys for remote execution from **satellite.lab.example.com** to **serverb.lab.example.com**.

- 1.1. On **workstation**, **ssh** to **satellite** as **student**. Use **sudo -i** to switch to **root**.

```
[student@workstation ~]$ ssh student@satellite
[student@satellite ~]$ sudo -i
[sudo] password for student: student
[root@satellite ~]#
```

- 1.2. Distribute the key to **serverb.lab.example.com**. Use the **ssh-copy-id** command to copy the **~foreman-proxy/.ssh/id\_rsa\_foreman\_proxy.pub** key.

```
[root@satellite ~]# ssh-copy-id -i \
~foreman-proxy/.ssh/id_rsa_foreman_proxy.pub \
root@serverb.lab.example.com
...output omitted...
Are you sure you want to continue connecting (yes/no)? yes
...output omitted...
root@serverb.lab.example.com's password: redhat
Number of key(s) added: 1
...output omitted...
```

2. Create a new custom Ansible Role on Satellite Server. Extract and copy the content from the precreated **/root/apache-setup-role.tgz** archive to the **/etc/ansible/roles** directory to create the new role on Satellite Server.

2.1. On **workstation**, **ssh** to **satellite** as **student**. Use **sudo -i** to switch to **root**.

```
[student@workstation ~]$ ssh student@satellite
[student@satellite ~]$ sudo -i
[root@satellite ~]#
```

2.2. Extract the precreated **apache-setup-role.tgz** file.

```
[root@satellite ~]# tar xzvf apache-setup-role.tgz
```

2.3. Copy the **apache-setup-role** directory to **/etc/ansible/roles/**:

```
[root@satellite ~]# cp -R apache-setup-role /etc/ansible/roles/
```

3. Import the **apache-setup-role** Ansible Role into Satellite. Use the **Operations** organization and make it available for any location.

3.1. Log in to the Satellite web UI at <https://satellite.lab.example.com> as **admin** using **redhat** as the password.

1. Use your browser to navigate to <https://satellite.lab.example.com>.
2. Log in as **admin** with **redhat** as a password.

3.2. Choose the **Operations** organization and **Any Location** location.

3.3. Click **Configure → Roles** and then click **Import from satellite.lab.example.com** to import your new custom role.

3.4. From the list of available Ansible Roles, select the check box for the **apache-setup-role** role and click **Update** to import the roles.

4. Create a new **apache\_test\_message** Ansible variable in the **apache-setup-role** Ansible role, that overrides the default value for that variable with **This variable has been set by Satellite**.

4.1. Click **Configure → Variables** and then click **New Ansible Variable** to create a new variable.

4.2. In the **Key** field, enter **apache\_test\_message** as the name for the variable. This variable is defined in the **defaults/main.yml** file inside your role.

4.3. Click the **Ansible Role** list and choose the **apache-setup-role** role.

4.4. Select the check box for **Override**.

4.5. In the **Default Value** field, enter **This variable has been set by Satellite** as the variable value.

4.6. Click **Submit** to create the variable.

5. Assign the new role to the **servera.lab.example.com** and **serverb.lab.example.com** servers.
  - 5.1. Click **Hosts** → **All Hosts** and then click the **servera.lab.example.com** host.
  - 5.2. Click **Edit** and then click the **Ansible Roles** tab.
  - 5.3. Click the plus sign (+) next to the **apache-setup-role** role to assign the role to the **servera.lab.example.com** server, and then click **Submit**.
  - 5.4. Click **Hosts** → **All Hosts** and then click the **serverb.lab.example.com** host.
  - 5.5. Click **Edit** and then click the **Ansible Roles** tab.
  - 5.6. Click the plus sign (+) next to the **apache-setup-role** role to assign the role to the **serverb.lab.example.com** server, and then click **Submit**.
6. Use the new **apache-setup-role** role to customize the **servera.lab.example.com** and **serverb.lab.example.com** servers.
  - 6.1. Click **Hosts** → **All Hosts** and then select the checkbox for the **servera.lab.example.com** and **serverb.lab.example.com** hosts.
  - 6.2. Click the **Select Action** list and choose **Play Ansible Roles**.

Monitor the status of the remote execution on the **Overview** tab that displays. Wait for it to successfully finish.
7. Use your browser on **workstation** to verify that Apache has been successfully deployed on both **servera.lab.example.com** and **serverb.lab.example.com**.
  - 7.1. Use your browser to navigate to `http://servera.lab.example.com` and `http://serverb.lab.example.com`.

This concludes the lab.

## Evaluation

As the **student** user on the **workstation** machine, use the **lab remote-review grade** command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab remote-review grade
```

## Finish

As the **student** user on the **workstation** machine, use the **lab remote-review finish** command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab remote-review finish
```

This concludes the lab.

## Summary

---

In this chapter, you learned:

- Red Hat Satellite allows you to run remote jobs on hosts using shell commands and scripts, and Ansible ad hoc commands, playbooks, and roles.
- Any command you want to run on a remote host must exist on Satellite as a job template.
- Satellite supports both Ansible Playbooks and Ansible Roles.
- When you are using variables in your Ansible Playbooks or roles, you must first import the variables to Satellite or Capsule.
- Red Hat Satellite 6.6 also supports Puppet.

## Chapter 8

# Provisioning Hosts

### Goal

Configure Satellite Server for host deployment and perform host provisioning.

### Objectives

- Describe provisioning types and configure provisioning contexts and resources on Satellite Server and Satellite Capsule Servers.
- Describe networking requirements for different provisioning types, and configure network services to support provisioning.
- Deploy new hosts using the configured provisioning resources, network services, and installation parameters.

### Sections

- Configuring Satellite Server for Host Provisioning (and Guided Exercise)
- Preparing Network Configuration for Provisioning (and Guided Exercise)
- Performing Host Provisioning (and Guided Exercise)

### Lab

Provisioning Hosts

# Configuring Satellite Server for Host Provisioning

---

## Objectives

After completing this section, you should be able to describe provisioning types and configure provisioning contexts and resources on Satellite Server and Satellite Capsule Servers.

## Defining Provisioning

*Provisioning* is the process of converting a system from bare metal to a state where it is deployed with an operating system. This includes adding baseline software and any configuration necessary for it to be placed into active duty.

A provisioning environment usually includes the following components:

### DHCP server

Dispenses configuration information, such as IP addresses, PXE images, and other information, including the addresses of network file servers.

### Network installation server

Stores and shares all files that make up the OS and applications installation to the network.

### PXE-capable hardware

These hosts can boot from the network. This means that local, physical installation media is not required to start installing an operating system. The ability to boot off of an image across the network relieves administrators of the burden of making physical installation media available locally to the hosts in order to initiate operating system installation.

### Kickstart file

The Kickstart file can be thought of as the complete set of instructions to install Red Hat Enterprise Linux on a new machine and bring it to a full state of readiness. This text file includes install settings, options, and scripts.

Red Hat Satellite integrates all those components. It can also provide supporting network services needed for system provisioning, such as DHCP, TFTP, and DNS.

## Provisioning with Red Hat Satellite

Red Hat Satellite can provision not only on bare-metal systems but also on virtualized infrastructures, as well as on public and private clouds. With Red Hat Satellite, you can deploy systems on all these platforms using the same tool and following the same provisioning process.

Red Hat Satellite supports the following virtualization infrastructures:

- Servers running Kernel-based Virtual Machines (KVMs).
- Red Hat Virtualization
- VMware vSphere

On those infrastructures, Red Hat Satellite can provision virtual machines from existing templates, or images, or from PXE.

Red Hat Satellite supports the following cloud providers:

- Red Hat OpenStack Platform
- Amazon EC2
- Google Compute Engine

All those platforms provide an API that Red Hat Satellite uses for provisioning.

## Provisioning with PXE

With PXE provisioning, you can deploy PXE-capable systems across the network without having to provide local installation media.

Even though PXE provisioning is widely used, it is not an option in many data center environments where PXE, TFTP, or DHCP is not allowed or cannot be under the control of Satellite Server due to the heterogeneous nature of some network environments. For these scenarios, Red Hat Satellite offers a boot disk plug-in which allows for the creation of boot ISO images. By booting off of these images, you can provision systems on networks where DHCP and TFTP services are not available.

In addition to PXE and boot disk provisioning, Red Hat Satellite can also search for and discover non-provisioned hosts on the network, so that they can be prepared for deployment. After they have been discovered, you can easily deploy those systems using either PXE or boot-disk provisioning.

## Describing Provisioning Templates

When preparing a system for deployment, you often need to create configuration files specific to that system. For example, the Kickstart file must indicate the partition scheme, the configuration details of the network interfaces, and other items specific to the new machine.

To avoid the creation of such static, one-off files to suit specific system and application requirements, Red Hat Satellite uses *provisioning templates*.

Provisioning templates use the *Embedded Ruby (ERB)* syntax. ERB is a feature of Ruby that enables the convenient generation of any kind of text from templates. With ERB, templates combine plain text with Ruby code for flow control and variable substitution.

You can find provisioning templates in the Satellite Server web UI under **Hosts → Provisioning Templates**. Red Hat Satellite supports various template types. The most commonly used provisioning template types are as follows:

### Provisioning Template Types

| Template     | Description                                                                                                                                                                                                  |
|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Provisioning | These templates provide the Kickstart files that Satellite Server uses for unattended operating system installation.                                                                                         |
| PXE Linux    | Satellite Server deploys the contents of these templates to the TFTP server. PXELINUX provisioning templates ensure that the provisioned system boots the correct installer with the correct kernel options. |
| Finish       | These templates contain scripts used to customize hosts after the main provisioning process is complete. You can only use those templates for hosts you provision from images in virtual environments.       |

| Template  | Description                                                                                                                                                                                            |
|-----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| User Data | The content of these templates is for cloud images that accept user data. They often provide data for the <b>cloud-init</b> program running inside images in Red Hat OpenStack Platform or Amazon EC2. |

Each template type addresses a different stage of the provisioning workflow. For example, when you deploy a new virtual machine, the PXE provisioning process makes use of the PXELINUX, provision, and finish provisioning templates:

1. Satellite Server deploys the PXELINUX template to the TFTP server.
2. The new virtual machine boots in PXE mode, and the PXELINUX template directs the machine to the provisioning template.
3. The provisioning template automates the unattended installation of the operating system.
4. At the end to the installation, the provisioning template directs the installer to retrieve and execute the contents of the finish template.

Satellite Server comes preinstalled with several provisioning templates. For the deployment of Red Hat Enterprise Linux, you usually use the following provisioning templates:

#### Default Provisioning Template for Red Hat Enterprise Linux

| Template type | Template name                      |
|---------------|------------------------------------|
| Provisioning  | <b>Kickstart default</b>           |
| PXELinux      | <b>Kickstart default PXELinux</b>  |
| Finish        | <b>Kickstart default finish</b>    |
| User Data     | <b>Kickstart default user data</b> |

Satellite Server also ships with *snippet* templates, which are reusable ERB code that other templates include.

## Describing Partition Table Templates

*Partition table templates* provide the instructions to configure the disks of new hosts. For example, the **Kickstart default** partition table template contains the kickstart instructions to partition and format the disks of new machines.

Provisioning templates call those partition table templates. By having the disk configurations outside the provisioning templates, you can use the same provisioning template for several models of machines but call a different partition table template depending on the disk size.

You can find the partition table templates in the Satellite Server web UI under **Hosts** → **Partition Tables**.

## Editing Templates

You can use the Satellite Server web UI to create new templates, but you cannot edit the default templates. Satellite Server locks those templates for protection, but you can clone them and then edit the copies.

The default templates are available in all organizations and locations. The templates you create or clone, however, belong by default to your current organization and location.

## Describing the Operating System Resource

An operating system resource groups the templates specific to an operating system type. For example, the default **Red Hat 8.1** operating system resource provides the **provisioning**, **PXElinux**, **finish**, **user data**, and **partition table** templates to use when deploying Red Hat Enterprise Linux 8.1 hosts.

When you prepare a new machine for provisioning, you associate it with an operating system for Satellite Server to know which templates to use during the provisioning workflow.

In the Satellite Server web UI, click **Hosts → Operating Systems** to access the operating systems configuration page.

## Enabling Kickstart Repositories

Before installing new hosts, you need to enable and synchronize the kickstart repositories from the Red Hat Content Delivery Network (CDN). A kickstart repository differs from standard repositories in that it contains boot images. Satellite uses the kickstart repository to generate installation media for provisioning.

To install new Red Hat Enterprise Linux 8 hosts, synchronize the following two repositories:

- **Red Hat Enterprise Linux 8 for x86\_64 - BaseOS (Kickstart)**
- **Red Hat Enterprise Linux 8 for x86\_64 - AppStream (Kickstart)**

The packages from those kickstart repositories are only used during operating system installations. After the hosts are installed and registered, the RPM repositories are used.

## Describing Hardware Models

When Satellite registers a host it did not build, it will generate model information based on the facts obtained by Facter. This can result in incorrect models or several models that are very similar. Satellite uses the **foreman-rake models:consolidate** command to attempt to consolidate similar models where possible.

You can define a hardware model to ensure that the correct DHCP options, such as **vendor class** and **hardware model**, are configured for your specific server hardware. For example, Sun SPARC systems can use the Jumpstart method to perform operating system installations, and having the precise model defined allows the DHCP server to allocate leases correctly.

Generic hardware models for Intel-based servers are already configured, so you should not need to define custom hardware models if you only use this class of hardware.



## References

For more information, refer to the *Introduction* and *Configuring Provisioning Resources* chapters in the *Red Hat Satellite 6.6 Provisioning Guide* at  
[https://access.redhat.com/documentation/en-us/red\\_hat\\_satellite/6.6/html-single/provisioning\\_guide/index](https://access.redhat.com/documentation/en-us/red_hat_satellite/6.6/html-single/provisioning_guide/index)

For more information on ERB, refer to the *Template Writing Reference* appendix in the *Red Hat Satellite 6.6 Managing Hosts* guide at  
[https://access.redhat.com/documentation/en-us/red\\_hat\\_satellite/6.6/html-single/managing\\_hosts/index#appe-Red\\_Hat\\_Satellite-Managing\\_Hosts-Template\\_Writing\\_Reference](https://access.redhat.com/documentation/en-us/red_hat_satellite/6.6/html-single/managing_hosts/index#appe-Red_Hat_Satellite-Managing_Hosts-Template_Writing_Reference)

## ► Guided Exercise

# Configuring Satellite Server for Host Provisioning

In this exercise, you will configure a Satellite Capsule Server with context and resources to support the bare-metal provisioning type.

## Outcomes

You should be able to:

- Enable and synchronize the kickstart repositories required for bare-metal provisioning.
- Configure Satellite Capsule Server to provide the resources used during provisioning.
- Prepare the provisioning templates, partition tables, and operating system resources.

## Before You Begin

This exercise requires a running Satellite Capsule Server on the **capsule** host, which you installed in a preceding exercise.

As the **student** user on the **workstation** machine, use the **lab provision-configure start** command to prepare your system for this exercise.

This command determines if the **satellite** and **capsule** hosts are reachable on the network and verifies that the required resources exist.

```
[student@workstation ~]$ lab provision-configure start
```

- ▶ 1. Log in to the Satellite web UI as **admin** using **redhat** as the password.
  - 1.1. Use your browser to navigate to <https://satellite.lab.example.com>.
  - 1.2. Log in as **admin** using **redhat** as the password.
- ▶ 2. Choose the **Operations** organization and **Boston** location from the main menu.
- ▶ 3. Enable synchronization of the Red Hat Enterprise Linux 8 kickstart repositories from the local CDN. Remember that the installation of new systems requires kickstart repositories and not the RPM repositories you enabled in a preceding exercise.
  - 3.1. Click **Content → Red Hat Repositories** to access the **Red Hat Repositories** page.
  - 3.2. In the drop-down menu under the search bar, select **Kickstart** and clear the check mark near **RPM**.

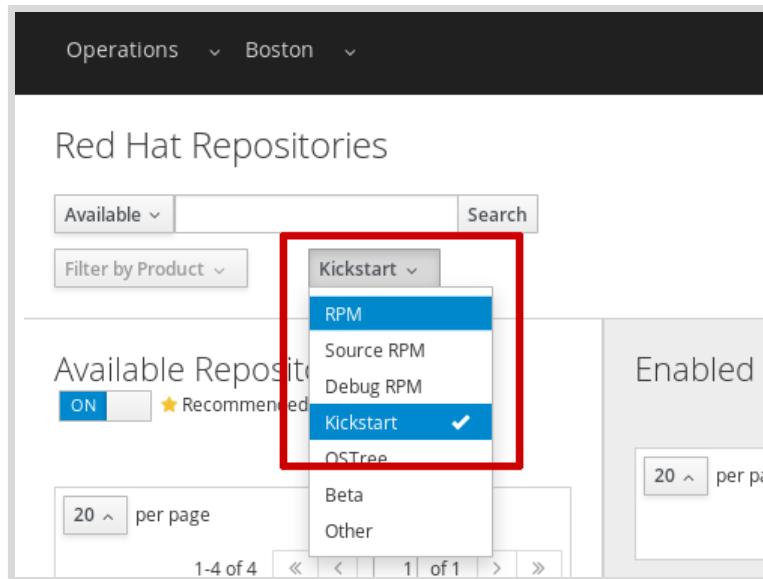


Figure 8.1: Listing Kickstart Repositories

Toggle the **Recommended Repositories** button to **ON** to list only the recommended repositories.

- 3.3. Expand **Red Hat Enterprise Linux 8 for x86\_64 - AppStream (Kickstart)** and **Red Hat Enterprise Linux 8 for x86\_64 - BaseOS (Kickstart)**. Click the plus sign (+) next to **x86\_64 8.1** of each of the repositories to enable them.
4. Use the Satellite Server web UI to synchronize the two kickstart repositories.
  - 4.1. Click **Content** → **Products**, and then click **Red Hat Enterprise Linux for x86\_64**.
  - 4.2. Select the check boxes for the **Red Hat Enterprise Linux 8 for x86\_64 - AppStream Kickstart 8.1** and **Red Hat Enterprise Linux 8 for x86\_64 - BaseOS Kickstart 8.1** repositories. Make sure that you select the kickstart repositories and not the RPM repositories. Click **Sync Now** to begin the repository synchronization.

Wait for the synchronization task to complete. The task can take up to 10 minutes to finish.
5. Add the two repositories to the **Base** content view. When done, publish and promote the content view to the **Development** life-cycle environment.

In a following exercise, you use that life-cycle environment to provision a new bare-metal system.

  - 5.1. Click **Content** → **Content Views**, and then click the **Base** content view.
  - 5.2. Click **Yum Content** → **Repositories**, and then click the **Add** tab.
  - 5.3. Select the check boxes next to the **Red Hat Enterprise Linux 8 for x86\_64 - AppStream Kickstart 8.1** and **Red Hat Enterprise Linux 8 for x86\_64 - BaseOS Kickstart 8.1** repositories, and then click **Add Repositories**.

- 5.4. Click **Publish New Version**, and then click **Save**. Wait for the publishing process to complete. The task can take up to 10 minutes to finish.
  - 5.5. When the publishing process is complete, click **Promote**. Select the **Development** life-cycle environment, and then click **Promote Version**.
- ▶ 6. Confirm that Satellite Capsule Server installed on the **capsule** host belongs to the **Operations** organization in **Boston**. If not, change the Capsule Server organization to **Operations** and its location to **Boston**.
- 6.1. Choose the **Operations** organization and **Boston** location from the main menu. Click **Infrastructure → Capsules** to access the **Capsules** page.

The page should list a capsule named **capsule.lab.example.com** which belongs to the **Operations** organization in **Boston**. If not, complete the following substeps:

    - Choose the **Any Organization** organization and **Any Location** location from the main menu to display all Satellite Capsule Servers.
    - Click **Edit** at the end of the **capsule.lab.example.com** row.
    - Click the **Locations** tab, and then click **Boston** to add it to the **Selected items** list.
    - Click the **Organizations** tab, and then click **Operations** to add it to the **Selected items** list.
    - Click **Submit**.
  - 6.2. Choose the **Operations** organization and **Boston** location from the main menu.
- ▶ 7. For the two Red Hat Enterprise Linux 8 kickstart repositories to be available from Satellite Capsule Server in Boston, synchronize the **Development** life-cycle environment with Satellite Capsule Server. If you already have added the **Development** life-cycle environment to Satellite Capsule Server in a previous exercise, then that synchronization is automatic.
- 7.1. Click **Infrastructure → Capsules** to access the **Capsules** page. Click **Edit** at the end of the **capsule.lab.example.com** row.

Click the **Lifecycle Environments** tab, and then click **Development** to add it to the **Selected items** list.

Click **Submit**.
  - 7.2. On the **Capsules** page, click the **capsule.lab.example.com** link.
  - 7.3. In the **Content Sync** section, select **Optimized Sync** from the **Synchronize** list to start a new synchronization. Do not wait for the synchronization to complete; proceed with the next step.
- ▶ 8. Confirm that the default provisioning templates are available to the **Boston** location and the **RedHat 8.1** operating system.
- 8.1. Click **Hosts → Provisioning Templates** to access the **Provisioning Templates** page.
  - 8.2. In the filter field, enter **kind = PXELinux**, and then click **Search**.
  - 8.3. Click the **Kickstart default PXELinux** link in the **Name** field.

- 8.4. Click the **Association** tab and make sure that the **RedHat 8.1** operating system displays in the **Selected items** list.
  - 8.5. Click the **Locations** tab and make sure that the **Boston** location displays in the **Selected items** list. If not, click **Boston** to move it to the **Selected items** list, and then click **Submit**.
  - 8.6. Repeat the previous substeps for the **Kickstart default** provisioning template. Use the **kind = provision** search filter to find that template.
- ▶ 9. Confirm that the default partition table template is available to the **Boston** location.
- 9.1. Click **Hosts → Partition Tables** to access the **Partition Tables** page.
  - 9.2. Click the **Kickstart default** link.
  - 9.3. Click the **Locations** tab and make sure that the **Boston** location displays in the **Selected items** list.
- ▶ 10. Confirm that the **RedHat 8.1** operating system references the correct provisioning templates and partition table.
- 10.1. Click **Hosts → Operating Systems** to access the **Operating Systems** page. The page displays the available operating systems.  
Satellite Server automatically created the **RedHat 8.1** operating system entry when you enabled the **Red Hat Enterprise Linux 8 for x86\_64 - AppStream (Kickstart)** and **Red Hat Enterprise Linux 8 for x86\_64 - BaseOS (Kickstart)** repositories.
  - 10.2. Click the **RedHat 8.1** link.
  - 10.3. On the **Partition Table** tab, confirm that the **Kickstart default** partition table displays in the **Selected items** list.
  - 10.4. On the **Templates** tab, confirm that **PXE Linux template** is set to **Kickstart default PXE Linux** and that **Provisioning template** is set to **Kickstart default**.

## Finish

On the **workstation** machine, use the **lab provision-configure finish** command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab provision-configure finish
```

This concludes the guided exercise.

# Preparing Network Configuration for Provisioning

---

## Objectives

After completing this section, you should be able to describe networking requirements for different provisioning types, and configure network services to support provisioning.

## Providing Network Services

To fully automate host provisioning, Red Hat Satellite Server requires access to properly configured DHCP and DNS services. For PXE provisioning, Satellite also requires a TFTP service. Each of these network services can either be installed directly on an integrated or external Capsule Server, or may be supplied by supported network services from existing, external servers in your organization. The following external services are supported:

### ISC BIND

The original BIND DNS service from the Internet Systems Consortium.

### ISC DHCP

A DHCP service from the Internet Systems Consortium, mounted to the capsule using NFS.

### Red Hat IdM

The DNS service of the Identity Management bundled with every RHEL distribution.

### Infoblox

An appliance or service applications for DNS and DHCP services.

### TFTP

An external TFTP server directory mounted to the capsule using NFS.

For typical host deployments to bare metal and virtual machines, a Capsule Server reserves an IP address from the DHCP server for the new host or instance, and registers the address and host name with the DNS service. Because cloud providers integrate managed DHCP and DNS services into their infrastructure, cloud-based deployments requested by Satellite are based on images and network configured by that provider, and managed by Satellite only when the instance becomes accessible. Cloud-based deployments are covered in a later chapter.

## Configuring a Capsule Server with Network Services

Capsule Server installation was introduced in a previous chapter, using the **satellite-installer** command to manage features for both Satellite and Capsule scenarios. The **satellite-installer** command is run repetitively to install, enable, revert, or modify features on each Capsule Server. The same command, with selected service options and parameters, is used to install specific services on a Capsule Server, or to configure a Capsule Server to provide a secure connection and access to an external service.

In general, Capsule Servers configured to use external network services are given privileged account access to allow both read and update capabilities for the external service. Network services can be reverted or reconfigured by re-executing **satellite-installer**.

To list and learn available command option and parameter syntax, use the command's help option.

```
[root@satellite ~]# satellite-installer --help
```

Network services can be installed directly on a Capsule Server. The following example configures and enables DNS, DHCP, and TFTP on the local Capsule Server. Each service is configured as being managed, and the included parameter values are for the local server.

```
[root@demo_capsule ~]# satellite-installer --scenario capsule \
--foreman-proxy-dns true \
--foreman-proxy-dns-managed true \
--foreman-proxy-dns-interface eth0 \
--foreman-proxy-dns-zone boston.lab.example.com \
--foreman-proxy-dns-forwarders 172.25.250.254 \
--foreman-proxy-dns-reverse 250.25.172.in-addr.arpa \
--foreman-proxy-dhcp true \
--foreman-proxy-dhcp-managed true \
--foreman-proxy-dhcp-interface eth0 \
--foreman-proxy-dhcp-range "172.25.250.50 172.25.250.100" \
--foreman-proxy-dhcp-gateway 172.25.250.254 \
--foreman-proxy-dhcp-nameservers 172.25.250.16 \
--foreman-proxy-tftp true \
--foreman-proxy-tftp-managed true \
--foreman-proxy-tftp servername $(hostname)
```

## Configuring a Capsule Server to Use External Services

For comparison, the following examples illustrate the syntax for configuring external services. For some external services, Capsule Server uses NFS to access the external service's configuration and data files. The service must already be installed on the external server and shared using NFS, and Capsule Server must be configured as an NFS client.



### Note

Using external services requires creating site-dependent network, service, and security configurations prior to configuring Capsule Server. These examples demonstrate only the **satellite-installer** syntax. To properly configure your Satellite environment, follow the complete procedures found in the *Installing Capsule Server and Provisioning Guide* product documentation.

The following syntax configures the capsule to use an ISC DHCP server, with the configuration and lease files made available locally through NFS mounts.

```
[root@demo_capsule ~]# satellite-installer --scenario capsule \
--foreman-proxy-dhcp true \
--foreman-proxy-dhcp-provider=remote_isc \
--foreman-proxy-plugin-dhcp-remote-isc-dhcp-config /mnt/nfs/etc/dhcp/dhcpd.conf \
--foreman-proxy-plugin-dhcp-remote-isc-dhcp-leases \
/mnt/nfs/var/lib/dhcpd/dhcpd.leases \
--foreman-proxy-plugin-dhcp-remote-isc-key-name omapi_key \
--foreman-proxy-plugin-dhcp-remote-isc-key-secret \
jNSE5YI3H1A80j/tkV4...A2Z0Hb6zv315CkNAY7DMYYCj48Umw== \
```

```
--foreman-proxy-plugin-dhcp-remote-isc-omapi-port=7911 \
--enable-foreman-proxy-plugin-dhcp-remote-isc \
--foreman-proxy-dhcp-server=dhcp.lab.example.com
```

Similarly, the following syntax configures the capsule to use a remote TFTP server, with the configuration files available locally through an NFS mount.

```
[root@demo_capsule ~]# satellite-installer --scenario capsule \
--foreman-proxy-tftp=true \
--foreman-proxy-tftp-root /mnt/nfs/var/lib/tftpboot \
--foreman-proxy-tftp-servername=tftp.lab.example.com
```

For an external DNS server, Capsule Server can perform necessary read and update tasks using a correctly configured DNS client. In this example, Capsule Server uses **nsupdate** to interact with an ISC DNS server. Capsule Server does not manage the DNS service, but still requires the correct RNDC keys to be allowed read and update access for the DNS service records.

```
[root@demo_capsule ~]# satellite-installer --scenario capsule \
--foreman-proxy-dns true \
--foreman-proxy-dns-managed=false \
--foreman-proxy-dns-provider=nsupdate \
--foreman-proxy-dns-server="172.25.250.16" \
--foreman-proxy-keyfile=/etc/rndc.key \
--foreman-proxy-dns-ttl=86400
```

## Configuring Provisioning Contexts

After configuring the Capsule network services, configure the provisioning contexts to be supported for deployments by adding organizations and locations to the Capsule. When configuring and initiating deployments, select a provision context as a combination of a single organization and location.

### To Configure Provisioning Contexts:

- In the Satellite web UI, choose the **Any Organization** organization and **Any Location** location, and then navigate to **Infrastructure → Capsules**.
- Select the desired Capsule Server and click **Edit**.
- Set the locations and organizations using the **Locations** and **Organizations** tabs, respectively.

You can use the **hammer** command to perform the same task:

```
[root@server ~]# hammer capsule update --name capsule \
--locations location1,location2,location3 \
--organizations organization1,organization2,organization3
```

## Creating Domains and Subnets

Each Satellite-deployed host is configured to reside in a single DNS domain and a network subnet. The domains and subnets must be preconfigured in the DNS and DHCP services defined earlier in this section. They are then created in Satellite as infrastructure resources before provisioning can be attempted.

Before creating infrastructure resources, set the Satellite provisioning context so that the resources are created for the correct organization and location context. In the Satellite UI, first choose the appropriate organization and location. Although it is not typical for subdomains and subnets to be used in multiple provisioning contexts, to do so requires resetting the organization and location before configuring infrastructure resources for that context.

## Configuring Domains

Provisioned hosts obtain their fully qualified domain name (FQDN) by combining their designated host name with the name of the domain to which they are assigned. During provisioning, Satellite updates the DNS server to add or modify the address records for the provisioned host's FQDN.

### To Configure Domains in the Satellite Web UI:

- Navigate to **Infrastructure → Capsules** and click **Create Domain**.
- Enter the previously configured domain name in the **DNS Domain** field.
- Indicate which Capsule Server will manage this domain in the **DNS Capsule** field.
- Use the **Locations** and **Organizations** tabs to verify the domain's locations and organizations. The available locations and organizations are limited to those configured for the selected Capsule.

You can use the **hammer** command to perform the same task:

```
[root@server ~]# hammer domain create --name domain.example.com \
--dns capsule.lab.example.com \
--locations location1,location2,location3 \
--organizations organization1,organization2,organization3
```

## Importing and Configuring Subnets

Satellite Server can import the details of the subnets that were declared with the **satellite-installer** command when the DHCP service was installed or configured on the Capsule Server.

### To Use the Satellite Web UI to Import Subnet Information:

- Navigate to **Infrastructure → Capsules**.
- For the required Capsule Server, click the arrow on the **Actions** list, and select **Import IPv4 subnets**.



#### Note

If the **Import IPv4 subnets** entry is not available, then the Capsule does not have a properly configured DHCP service.

- Enter a subnet name, for later resource recognition, and provide the IP address, in the **Primary DNS Server** field, for the DNS server that resolves hosts for this subnet.
- Select **DHCP** from the **IPAM** list to specify the import source, and then click **Submit**.

### To Configure Previously Imported Subnets:

- Navigate to **Infrastructure → Subnets** and select an imported subnet.

- On the **Domains** and **Capsules** tabs, select the domain to associate with the subnet and the capsule to manage the subnet.
- Use the **Locations** and **Organizations** tabs to verify the subnet locations and organizations. The available locations and organizations are limited to those configured for the selected Capsule.

To create new subnets, navigate to **Infrastructure** → **Subnets** and click **Create Subnet**.

You can use the **hammer** command to perform the same task:

```
[root@server ~]# hammer subnet create --name subnet_name \
--locations location1,location2,location3 \
--organizations organization1,organization2,organization3 \
--domains domain.example.com \
--network 172.25.250.0 \
--mask 255.255.255.0 \
--dns-primary 172.25.250.16 \
--from 172.25.250.50 \
--to 172.25.250.100 \
--dns capsule.lab.example.com \
--dhcp capsule.lab.example.com \
--tftp capsule.lab.example.com \
--boot-mode DHCP \
--ipam DHCP
```

The domains and subnets are available for provisioning only in the organizations and locations in which they are configured.



## References

For more information, refer to the *Configuring Networking* chapter in the *Red Hat Satellite 6.6 Provisioning Guide* at  
[https://access.redhat.com/documentation/en-us/red\\_hat\\_satellite/6.6/html-single/provisioning\\_guide/index#Configuring\\_Networking](https://access.redhat.com/documentation/en-us/red_hat_satellite/6.6/html-single/provisioning_guide/index#Configuring_Networking)

For more information, refer to *Red Hat Satellite 6.6 Installing Capsule Server* at  
[https://access.redhat.com/documentation/en-us/red\\_hat\\_satellite/6.6/html-single/installing\\_capsule\\_server/index](https://access.redhat.com/documentation/en-us/red_hat_satellite/6.6/html-single/installing_capsule_server/index)

## ► Guided Exercise

# Preparing Network Configuration for Provisioning

In this exercise, you will configure network services and resource data to prepare to deploy a host.

### Outcomes

You should be able to:

- Configure Satellite Capsule Server to provide the DNS, DHCP, and TFTP services.
- Create domain and subnet resources with the Satellite web UI.

### Before You Begin

This exercise requires a running Satellite Capsule Server on the **capsule** host, which you installed in a preceding exercise.

As the **student** user on the **workstation** machine, use the **lab provision-network start** command to prepare your system for this exercise.

This command determines if the **satellite** and **capsule** hosts are reachable on the network and verifies that the required resources exist.

```
[student@workstation ~]$ lab provision-network start
```

The Operations team would like to use Satellite Capsule Server in **capsule.lab.example.com** to provision systems on the **172.25.250.0/24** subnet in the Boston office. The hosts in Boston belong to the **boston.lab.example.com** DNS domain.

In this exercise, you configure Satellite Capsule Server to provide the DNS, DHCP, and TFTP services.

Satellite Capsule Server must be authoritative for the **boston.lab.example.com** DNS zone. The DHCP service must provide IP addresses from the range **172.25.250.50** through **172.25.250.100**.

- ▶ 1. Use the **satellite-installer** command on the **capsule.lab.example.com** machine to enable and configure the DNS, DHCP, and TFTP services.
  - 1.1. On **workstation**, use **ssh** to log in to **capsule** as **student**. Use **sudo -i** to switch to **root**.

```
[student@workstation ~]$ ssh student@capsule
[student@capsule ~]$ sudo -i
[root@capsule ~]#
```

- 1.2. Use the **satellite-installer** command to enable the DNS, DHCP, and TFTP services. The following table details the configuration parameters.

#### Service Parameters

| Parameter                    | Value                                  |
|------------------------------|----------------------------------------|
| DNS network interface        | <b>eth0</b>                            |
| DNS forwarder                | <b>172.25.250.254</b>                  |
| DNS zone                     | <b>boston.lab.example.com</b>          |
| DNS reverse                  | <b>250.25.172.in-addr.arpa</b>         |
| DHCP network interface       | <b>eth0</b>                            |
| DHCP range                   | <b>172.25.250.50 to 172.25.250.100</b> |
| Name server provided by DHCP | <b>172.25.250.16</b>                   |
| Gateway provided by DHCP     | <b>172.25.250.254</b>                  |

In the preceding table, the **172.25.250.16** IP address is the address of the **capsule.lab.example.com** server, which is the DNS server for the **boston.lab.example.com** zone.

For your convenience, you can copy and paste the following command from the **/root/satellite-installer-example.txt** file.

```
[root@capsule ~]# satellite-installer --scenario capsule \
--foreman-proxy-dns true \
--foreman-proxy-dns-interface eth0 \
--foreman-proxy-dns-forwarders 172.25.250.254 \
--foreman-proxy-dns-zone boston.lab.example.com \
--foreman-proxy-dns-reverse 250.25.172.in-addr.arpa \
--foreman-proxy-dhcp true \
--foreman-proxy-dhcp-interface eth0 \
--foreman-proxy-dhcp-range "172.25.250.50 172.25.250.100" \
--foreman-proxy-dhcp-nameservers 172.25.250.16 \
--foreman-proxy-dhcp-gateway 172.25.250.254 \
--foreman-proxy-tftp true
...output omitted...
Installing          Done      [100%] [.....]
Success!
* Capsule is running at https://capsule.lab.example.com:9090
The full log is at /var/log/foreman-installer/capsule.log
```

- ▶ 2. Log in to the Satellite web UI as **admin** using **redhat** as the password.
  - 2.1. Use your browser to navigate to <https://satellite.lab.example.com>.
  - 2.2. Log in as **admin** using **redhat** as the password.
- ▶ 3. Choose the **Operations** organization and **Boston** location from the main menu.

- ▶ 4. Confirm that the DNS, DHCP, and TFTP features are active on Satellite Capsule Server.
- 4.1. Click **Infrastructure → Capsules** to access the **Capsules** page, and then click the **capsule.lab.example.com** link.
  - 4.2. On the **Overview** tab, confirm that the **Active features** section lists the DNS, DHCP, and TFTP features. If not, click **Refresh features**.
- ▶ 5. Create the **boston.lab.example.com** DNS domain and make it available for use by the **Operations** organization at the **Boston** location. This is the domain that Satellite Capsule Server manages.
- 5.1. Click **Infrastructure → Domains**, and then click **Create Domain**.
  - 5.2. Enter **boston.lab.example.com** in the **DNS Domain** field.
  - 5.3. Choose **capsule.lab.example.com** from the **DNS Capsule** list.
  - 5.4. Click the **Locations** tab and verify that the **Boston** location is already associated with the new domain. If not, click **Boston** to add it to the **Selected items** list.
  - 5.5. Click the **Organizations** tab and verify that the **Operations** organization is already associated with the new domain. If not, click **Operations** to add it to the **Selected items** list.
  - 5.6. Click **Submit**.
- ▶ 6. Create the **Boston Data Center** subnet, **172.25.250.0/24**, and define its DHCP address range and DNS server information. Because this is the subnet that you used when you ran the **satellite-installer** command in a previous step, Satellite Server can help you create that subnet resource.
- 6.1. Click **Infrastructure → Capsules** to access the **Capsules** page.
  - 6.2. Select **Import IPv4 subnets** from the **Actions** list for the **capsule.lab.example.com** row to import the **172.25.250.0/24** subnet that you used with the **satellite-installer** command.
  - 6.3. Complete the page with the following details.
- Boston Subnet Details**
- | Field                     | Value                     |
|---------------------------|---------------------------|
| <b>Name</b>               | <b>Boston Data Center</b> |
| <b>Primary DNS Server</b> | <b>172.25.250.16</b>      |
| <b>IPAM</b>               | <b>DHCP</b>               |
- Do not modify any other field.
- 6.4. Click **Submit**.
- ▶ 7. Define the IP range for the new **Boston Data Center (172.25.250.0/24)** subnet and associate it with the **boston.lab.example.com** domain. Confirm that

it belongs to the **Operations** organization at the **Boston** location. Also, assign the DHCP, TFTP, and DNS services for the subnet to Satellite Capsule Server running on **capsule.lab.example.com**.

- 7.1. Click **Infrastructure** → **Subnets** to access the **Subnets** page, and then click the **Boston Data Center** link.
- 7.2. Set **Start Of Ip Range** to **172.25.250.50** and **End Of Ip Range** to **172.25.250.100**.
- 7.3. Click the **Domains** tab, and then click the **boston.lab.example.com** domain to associate it with the subnet.
- 7.4. Click the **Capsules** tab, and then select **capsule.lab.example.com** in all the lists.
- 7.5. Click the **Locations** tab and confirm that the **Boston** location is in the **Selected items** list.
- 7.6. Click the **Organizations** tab and confirm that the **Operations** organization is in the **Selected items** list.
- 7.7. Click **Submit**.

## Finish

On the **workstation** machine, use the **lab provision-network finish** command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab provision-network finish
```

This concludes the guided exercise.

# Performing Host Provisioning

---

## Objectives

After completing this section, you should be able to deploy new hosts using configured provisioning resources, network services, and installation parameters.

## Supported Provisioning Platforms

Red Hat Satellite 6 supports host provisioning on numerous compute platforms, including Red Hat products and other vendor platforms, including both public and private clouds. Provisioning methods support both bare metal and virtual machine instances, and can initiate provisioning using network booting, boot images, and discovery or introspection services. This section describes the platforms currently supported in Red Hat Satellite 6.6.

## Introducing Compute Resources and Profiles

Satellite 6 refers to provisioning provider platforms as *compute resources*, which can be categorized into three provisioning types: bare metal, virtualization, and cloud. Each provisioning type has unique technical implementation features that determine the tools and methods available for provisioning hosts on each platform.

Most platforms also use their own form of a *compute profile*, which allows users to predefine hardware such as CPUs, memory, and storage. The following narrative describes the provisioning types, lists the supported compute resources for each, and provides a brief summary of the contrasting provisioning characteristics:

### Bare metal

Bare-metal servers can be provisioned using either manual or unattended methods, initiated by a PXE network boot process or various forms of buildable boot disks. The installation process then uses kickstart-based RPM package methods or third party installation media.

For PXELinux provisioning, Satellite is capable of working with both BIOS and UEFI firmware and interfaces. Available blank hosts can be initially discovered and have their hardware detected and stored in Satellite as a ready-to-provision host list.

### Virtualization

Virtualization platforms support PXE network and kickstart-based package installations, but administrators may prefer image-based provisioning. Virtualization platforms create hosts using profiles that predefine physical characteristics, eliminating the need for host discovery. Satellite uses privileged account access to request a VM instance, then continues with PXE or image-based provisioning. These platforms are supported in Satellite 6.6:

- libvirt - A Red Hat Enterprise Linux hypervisor use the Kernel-based Virtual Machine (KVM) virtualization daemon, accessed through the **libvirt** API. Satellite connects to the **libvirt** API to provision hosts on the hypervisor and control many virtualization functions. Hosts are instantiated using compute profiles configured in Satellite.
- Red Hat Virtualization - RHV is an enterprise server and desktop virtualization platform built on Red Hat Enterprise Linux and managed through a REST API. Satellite connects to the RHV API to create virtual machines and control their power states. Hosts are created using Satellite compute profiles that refer to templates and images configured and stored in RHV.

- VMware – VMware vSphere is an enterprise virtualization platform from VMware. Satellite connects with the vSphere platform to create new virtual machines and control their power management states. Similar to RHV behavior, hosts are created using Satellite compute profiles using templates and images configured and stored in VSphere.

#### Cloud

Satellite Server can interact with supported cloud providers, similar to the description for virtualization platforms, but with important differences. Typically, cloud platforms include DHCP services in their infrastructure and provide Satellite limited access to DNS. Satellite connects to a cloud management API to request a new host instance, which is then configured and registered by the cloud platform, then booted using a cloud-based image.

Satellite can connect to and manage instances after they are deployed. Package-based installations are not recommended for cloud, due to high cost and bandwidth requirements. Provisioning on supported cloud platforms is discussed in a later chapter. These platforms are supported in Satellite 6.6:

- Red Hat OpenStack Platform - RHOSP is an Infrastructure-as-a-Service hybrid cloud platform for the development of cloud-enabled workloads. Satellite connects to the RHOSP REST API to create new instances and control their power management states.
- Amazon Elastic Compute Cloud – Amazon EC2 is a web service that provides public cloud compute resources. Satellite connects with the EC2 public API to create instances and control their power management states.
- Google Compute Engine - Satellite 6 connects to GCE to create new virtual machines and control their power management states. Only image-based provisioning is supported for creating GCE hosts.

## Configuring Compute Resources and Profiles

To configure Satellite to work with a provisioning provider, navigate to **Infrastructure** → **Compute Resources** and click **Create Compute Resource** to create a provider connection. Enter a name to identify the resource, and select the provider type from the **Provider** list.

All remaining fields on this screen are determined by the selected platform type. Typically, a compute resource connection requires a URL for the platform location, a user name and password with sufficient privileges, and security and structure information needed to deploy systems to the correct data centers, clusters, or projects on the platform.

The information stored in compute profiles is specific to the platform for which it is created.

To configure a profile, navigate to **Infrastructure** → **Compute Profiles** and click **Create Compute Profile**. Choose an existing compute resource that defines the provider connection where this profile will be used. The remaining fields are determined by the selected compute resource. Typically, a compute profile specifies the CPU, memory, storage allocation, network interface configuration, an image to use, and additional platform-specific parameters.



#### Note

The available platforms, technologies, and capabilities supported by Red Hat Satellite Server are continuously being expanded. Covering all provisioning configurations is outside of the scope of this course. The remainder of this narrative section focuses on bare-metal provisioning. To properly configure each of your supported compute platforms, follow the detailed procedures located in the *Provisioning Guide* product documentation.

## PXE Provisioning on Bare Metal

There are four primary ways to provision bare-metal instances in Satellite 6.6. Provisioning can be initiated using either a PXE network boot or a boot disk. Provisioning can continue with either an installation or, when using the **Discovery** service, listing the host as available for later manual or automated selection and installation.

### Unattended Provisioning (PXE boot and installation)

New hosts are identified by a MAC address and Satellite provisions the host using a PXE boot process. The host obtains the kickstart tree and installation packages from Satellite.

### Unattended Provisioning with Discovery (PXE boot and hardware detection)

New hosts use PXE boot to load the Satellite **Discovery** service. This service identifies hardware information about the host and lists it as an available host to provision by a later, manual user selection. If sufficient discovery rules are configured, the host may automatically begin installing the OS when listed.

### PXE-less Provisioning (boot disk and installation)

New hosts are provisioned with a boot disk or PXE-less discovery image that Satellite generates. Boot disks can be boot-only ISOs that point to an installation media URL, full distribution installation ISOs, or custom built ISOs for a defined use case. Boot disk types are described later in this section.

### PXE-less Provisioning with Discovery (boot disk and hardware detection)

New hosts use an ISO boot disk that loads the Satellite **Discovery** service. This service identifies hardware information about the host and lists it as an available host to provision. As described earlier, this results in the host being listed as available for manual or automated selection and installation.

## Provisioning Resources

Throughout this course, you have created resources that are now necessary for successful provisioning. You have created synchronized content with repositories, subscriptions, and kickstart trees to make installable operating systems available. You have created the provisioning templates, partition tables, and activation keys to simplify the host configuration process, and configured Capsule Servers to provide or proxy DNS, DHCP and TFTP services.

Previously in this chapter, you defined domains and imported or created subnets to configure where the provisioned hosts will reside. You ensured that those provisioning resources are available in the correct organization and location provisioning contexts. Finally, you created compute resources and compute profiles to define the compute platforms and parameters required to initialize hosts at the start of provisioning. Now that the resources are ready, host provision begins by creating a new host in the Satellite web UI.

## Creating a Host Group

To provision a new host, you first create a host resource in Satellite Server for that host. The host resource groups all the resources that you prepared previously, such as the domain, the subnet, and the operating system, which defines the provisioning templates to use for provisioning. The host resource can also provide an activation key for registration.

When you have similar hosts to create, Red Hat recommends that you first create a *host group* resource. A host group provides default values for all the above parameters. Creating a host resource becomes a matter of selecting the appropriate host group and setting the few remaining specific parameters.

Host groups simplify the host creation process and allow for consistent provisioning.

To create a host group, navigate to **Configure → Host Groups**, and then click **Create Host Group**. Choose a descriptive name for the host group, fill the form, and then click **Submit**.

You can do the same from the command line with the **hammer hostgroup create** command. Run the **hammer hostgroup create --help** command to list all the parameters you can define in a host group.

You can nest host groups in hierarchies. The idea is to create a base, or parent, host group that contains all the generic parameters for your organization, then create nested, or child, host groups under that parent, to provide more specific settings (for example, to configure web servers or database servers).

## Creating a Host

Unattended provisioning is the simplest form of host provisioning. Enter the host details on Satellite and network boot your host. Satellite automatically manages the PXE configuration, organizes networking services, and provides the operating system and configuration for the host.

To create a host for unattended provisioning, navigate to **Hosts → Create Host** and enter a name for the new host. Ensure that the organizations and locations are set to your provisioning context requirements. Select a host group from the **Host Group** list for information to populate this screen. Configure networking parameters by editing the interface settings, including a fully qualified domain name, a MAC address, and ensure that the **Managed**, **Primary**, and **Provision** options are selected for the first interface.

Verify that the **Operating System** tab's fields contain proper values. Click **Resolve** in the **Provisioning** template to confirm that the new host can identify the right provisioning templates to use. Click the **Parameters** tab, and ensure that a parameter exists that provides an activation key. If not, add an activation key. Click **Submit** to save the host details.

The same task can be accomplished from the command line:

```
[root@demo_satellite]# hammer host create --name unattended_hostname \
--organization example --location location1 \
--hostgroup host_group1 --mac aa:bb:cc:dd:ee:ff \
--build true --enabled true --managed true
```

The created host entry includes all necessary provisioning settings and parameters for PXE booting the bare-metal host. When network-booted, the host will use PXE to interact with the relevant Capsule Server's DHCP service. From the DHCP service, the host will obtain its assigned IP address and the information needed to load the boot image from the configured TFTP service. After that, it will start installing the operating system from the assigned Kickstart tree.

As part of the provisioning process, DNS forward and reverse resolution records will be created in the DNS service's zones. When the installation completes, the host uses the activation key to register to Satellite and install the necessary configuration and management tools from the Red Hat Satellite Tools repository. The host is now running and accessible on the network.

## Boot Disk Provisioning

PXE provisioning may not be allowed in some environments, or available in the firmware for some hardware. Satellite can perform PXE-less host provisioning by generating a boot ISO that hosts can use. Using this ISO, the host can connect to Satellite, boot the installation media, and complete an installation. There are four types of boot ISOs:

#### Host image

A boot ISO for a specific host. This image contains only the boot files that are necessary to access the installation media on Satellite. You define the host and subnet data in Satellite and the image is created with static networking. The boot image will dynamically chain-load the operating system boot loader from Satellite Server.

#### Full host image

A boot ISO that contains the kernel and initial RAM disk image for the specific host. This image is useful if the host fails to correctly chain-load the operating system boot loader from Satellite Server due to hardware or firmware issues. The provisioning template still downloads from Satellite and defines the configuration. These images take significantly longer to generate, and become out-of-date quickly as the operating system and associated templates change.

#### Generic image

A boot ISO that is not associated with any specific host, and can be used with all hosts configured with Satellite Server. After it has booted, the host attempts to retrieve a provisioning template from Satellite by matching the MAC address or IP address against host provisioning configurations on Satellite. The image does not store IP address details, and requires access to a network DHCP server to bootstrap. This image is available at the **/bootdisk/disks/generic** URL on your Satellite Server, for example, <https://satellite.example.com/bootdisk/disks/generic>, and does not need to be dynamically generated.

#### Subnet image

A boot ISO that is similar to the generic image but is configured with the address of a Capsule Server. This image is generic to all hosts with a provisioning NIC on the same subnet.

## Generating a Boot Disk

When the host configuration screen is completed in the Satellite web UI, the host detail page displays. On this page, click **Boot disk** to select the desired boot disk type to generate an iPXE boot image. The generated image can then be written to CD, DVD, USB stick, or memory cards and then booted locally from the host to initiate bare-metal provisioning. Because the **Host** image uses user-defined static networking information, it does not require DHCP or TFTP services. This method is then suitable for network environments where administrators who perform provisioning do not have necessary access or control over network infrastructure.

## Detecting Hosts with the Discovery Service

Red Hat Satellite provides a method to automatically detect hosts on a network that are not in your Satellite inventory. These hosts boot the discovery image that performs hardware detection and relays this information back to Satellite. This method creates a list of ready-to-provision hosts in Satellite without needing to enter the MAC address of each host. The **Discovery** service is enabled by default on Satellite.

The discovery image is installed to the **/usr/share/foreman-discovery-image/** directory, and a PXE boot image is generated and saved in the **/var/lib/tftpboot/boot directory**. Capsule Server is configured as a proxy for the **Discovery** service. After the discovery configuration is complete, a new menu option is located in the Satellite web UI at **Hosts** → **Discovered Hosts**.

To use the discovery service, boot a blank bare-metal host, and choose **discovery** from the resulting boot menu to boot the Discovery image. When finished booting, a status screen displays. In the Satellite web UI, navigate to **Hosts** → **Discovered Hosts** to view the newly discovered host. The discovered hosts automatically define their host name based on their MAC address.

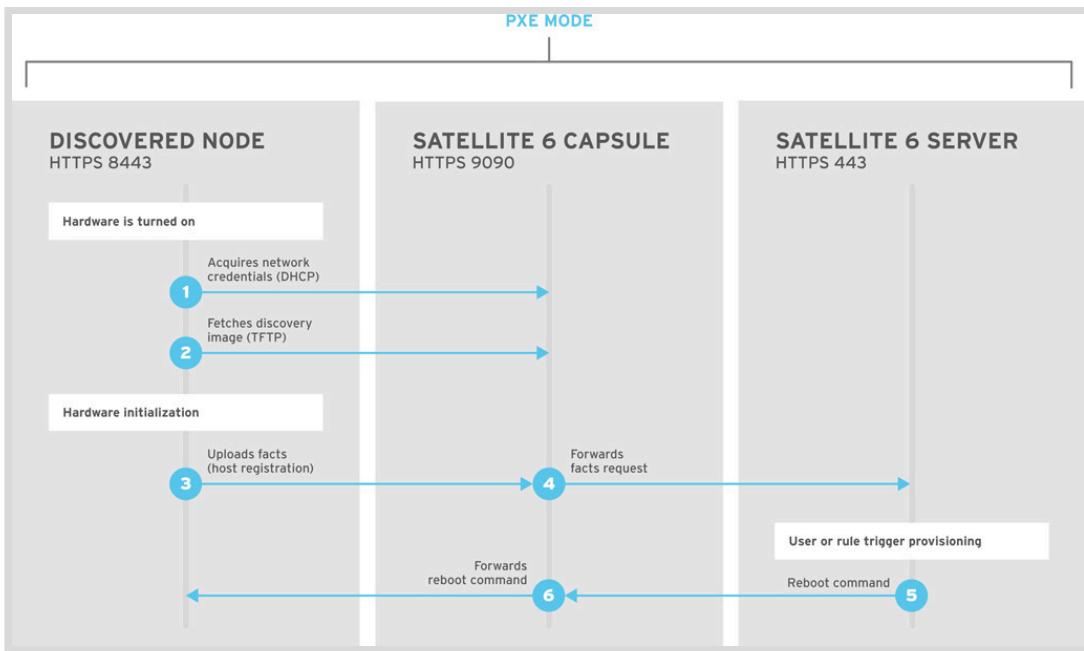


Figure 8.2: Discovery service with PXE boot

Satellite also provides a PXE-less Discovery service that operates without the need for PXE-based services (DHCP and TFTP). The discovery ISO acts as bootable media. The ISO can be written to CD, DVD, USB sticks or memory cards and then booted locally from the host to initiate discovery.

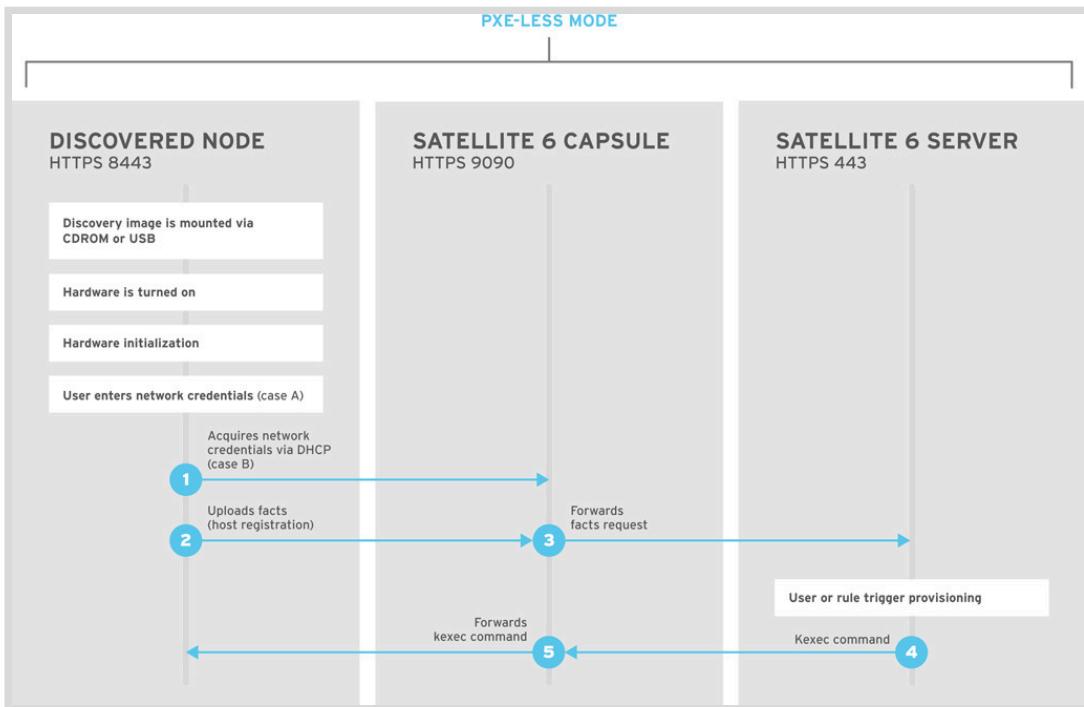


Figure 8.3: Discovery service with image boot

Insert the Discovery boot media into a bare-metal host and boot from the media. The Discovery image displays an option menu with two choices:

- **Manual network setup** – the Discovery image requests a set of network options, including the primary network interface that connects to Satellite Server, and the **IPv4 Address**, **IPv4 Gateway**, and **IPv4 DNS** server configuration parameters.
- **Discovery with DHCP** – the Discovery image requests only the primary network interface that connects to Satellite Server. It attempts to automatically configure the network interface using the configured DHCP server.

After the primary interface is configured, the Discovery image requests the URL of the Capsule Server offering the Discovery service. The Discovery image provides fields to input custom facts for the Facter tool to relay back to Satellite, entered in a name-value format.

When Satellite reports a successful communication with the Discovery service, navigate to **Hosts** → **Discovered Hosts** to view the newly discovered host.

Provisioning discovered hosts follows a provisioning process that is similar to PXE provisioning. The main difference is that instead of manually entering the host's MAC address, you select the host to provision from the list of discovered hosts.

As a method of automating the provisioning process for discovered hosts, Red Hat Satellite 6 provides a feature to create discovery rules. These rules define how discovered hosts automatically provision themselves, based on the assigned host group. For example, you can automatically provision hosts with a high CPU count as hypervisors or provision hosts with large hard disks as storage servers.



### References

For more information, refer to the *Red Hat Satellite 6.6 Provisioning Guide* at  
[https://access.redhat.com/documentation/en-us/red\\_hat\\_satellite/6.6/html-single/provisioning\\_guide/index](https://access.redhat.com/documentation/en-us/red_hat_satellite/6.6/html-single/provisioning_guide/index)

## ► Guided Exercise

# Performing Host Provisioning

In this exercise, you will deploy a host using the bare-metal provisioning type configuration.

### Outcomes

You should be able to:

- Create host group and host resources with the Satellite web UI to prepare for bare-metal provisioning.
- Perform PXE provisioning of a new host.

### Before You Begin

This exercise requires a running Satellite Capsule Server on the **capsule** host, which you installed in a preceding exercise.

As the **student** user on the **workstation** machine, use the **lab provision-host start** command to prepare your machine for this exercise.

This command determines if the **satellite** and **capsule** hosts are reachable on the network and verifies that the required resources exist.

```
[student@workstation ~]$ lab provision-host start
```

- ▶ 1. Log in to the Satellite web UI as **admin** using **redhat** as the password.
  - 1.1. Use your browser to navigate to <https://satellite.lab.example.com>.
  - 1.2. Log in to the Satellite web UI.
- ▶ 2. Choose the **Operations** organization and **Boston** location from the main menu.
- ▶ 3. To simplify new host provisioning in the **Operations** organization and **Boston** location, create a host group named **Operations Host Group**. This group contains the common settings needed when creating new hosts in the **Operations** organization.
  - 3.1. Click **Configure → Host Groups**, and then click **Create Host Group**. Complete the page with the following details.

Do not modify any other field, and do not submit the form yet.

#### Host Group Details

| Field                 | Value                        |
|-----------------------|------------------------------|
| Name                  | <b>Operations Host Group</b> |
| Lifecycle Environment | <b>Development</b>           |

| Field                 | Value                   |
|-----------------------|-------------------------|
| <b>Content View</b>   | Base                    |
| <b>Content Source</b> | capsule.lab.example.com |

- 3.2. Click the **Network** tab, and then set **Domain** to **boston.lab.example.com**. Set **IPv4 Subnet** to **Boston Data Center (172.25.250.0/24)**.
- 3.3. Click the **Operating System** tab, and complete the page with the following details.  
Do not modify any other field, and do not submit the form yet.

#### Host Group Operating System Details

| Field                   | Value             |
|-------------------------|-------------------|
| <b>Architecture</b>     | x86_64            |
| <b>Operating system</b> | RedHat 8.1        |
| <b>Media Selection</b>  | Synced Content    |
| <b>Partition Table</b>  | Kickstart default |
| <b>PXE loader</b>       | PXELinux BIOS     |

- 3.4. Click the **Locations** tab and confirm that the **Boston** location is in the **Selected items** list.
  - 3.5. Click the **Organizations** tab and confirm that the **Operations** organization is in the **Selected items** list.
  - 3.6. Click the **Activation Keys** tab, and then set **Activation Keys** to **OperationsServers**.
  - 3.7. Click **Submit**.
- 4. Create a host resource for **servere.boston.lab.example.com**, which is the system to provision.

- 4.1. Click **Hosts** → **Create Host**, and then complete the page with the following details.  
When you set the host group, the system automatically fills the form with the parameters from that group. Do not modify any other field, and do not submit the form yet.

#### New Host Details

| Field               | Value      |
|---------------------|------------|
| <b>Name</b>         | servere    |
| <b>Organization</b> | Operations |
| <b>Location</b>     | Boston     |

| Field             | Value                        |
|-------------------|------------------------------|
| <b>Host Group</b> | <b>Operations Host Group</b> |

- 4.2. Click the **Operating System** tab, and then set **Root Password** to **redhat123**. Click **Resolve** to confirm that the system can correctly retrieve the templates for provisioning.
- 4.3. Click the **Interfaces** tab, and then click **Edit** at the end of the interface row. Set the **MAC Address** to **52:54:00:00:fa:0e**. This is the address of the **servere** network interface. Click **Ok**.
- 4.4. Click **Submit**.
- 5. Access the **servere** console and initiate a PXE boot. Provisioning should proceed automatically. When the installation is complete, the host boots the newly provisioned operating system. This may take up to 15 minutes.
- 5.1. Locate the icon for the **servere** console, as appropriate for your classroom environment. Open the console.
- 5.2. To reboot, send a **Ctrl+Alt+Del** to your system using the relevant keyboard, virtual, or menu entry.
- 5.3. Press the **Spacebar** at the **Boot options** screen to stop the countdown.
- 5.4. Select **Network boot from device net0**, and press **Enter**. The provisioning process starts automatically.
- 5.5. Wait for the installation to complete, it can take more than 10 minutes, and the system to reboot. From the console, log in as **root** using **redhat123** as the password. Use the **hostname** command to verify the system name.

```
[root@servere ~]# hostname
servere.boston.lab.example.com
```

When done, log off from **servere** and close the console.

## Finish

On the **workstation** machine, use the **lab provision-host finish** command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab provision-host finish
```

This concludes the guided exercise.

## ► Lab

# Provisioning Hosts

In this lab, you will configure the discovery service, and detect and deploy a host using the discovery method.

## Outcomes

You should be able to:

- Create the Satellite Server resources for provisioning.
- Configure the Discovery feature on a Satellite Capsule Server.
- Discover new hosts on the network.
- Provision discovered hosts.

## Before You Begin

This exercise requires a running Satellite Capsule Server on the **capsule** host, which you installed in a preceding exercise.

As the **student** user on the **workstation** machine, use the **lab provision-review start** command to prepare your system for this exercise.

This command determines if the **satellite** and **capsule** hosts are reachable on the network and verifies that the required resources exist.

```
[student@workstation ~]$ lab provision-review start
```

1. Enable and synchronize the following Red Hat Enterprise Linux 8 kickstart repositories for the **Tokyo** location in the **Finance** organization:
  - **Red Hat Enterprise Linux 8 for x86\_64 - AppStream (Kickstart)**
  - **Red Hat Enterprise Linux 8 for x86\_64 - BaseOS (Kickstart)**
2. Add the two kickstart repositories to the **Base** content view. When done, publish and promote the content view to the **Build** life-cycle environment.
3. Set the location of Satellite Capsule Server installed on the **capsule** host to **Tokyo**. Set the organization to **Finance**. Associate the **Build** life-cycle environment with Satellite Capsule Server and initiate a synchronization.  
Remove the other organizations, locations, and life-cycle environments from the Satellite Capsule Server configuration.
4. Confirm that the Finance team in Tokyo can use the following resources for provisioning Red Hat Enterprise Linux 8.1 systems.
  - **Kickstart default PXELinux** template
  - **Kickstart default** provisioning template
  - **Kickstart default** partition table

Make sure that the **RedHat 8.1** operating system resource uses those templates by default.

5. Enable the DNS, DHCP, and TFTP services in Satellite Capsule Server on **capsule.lab.example.com**. Use the following settings.

#### Service Parameters

| Parameter                    | Value                                  |
|------------------------------|----------------------------------------|
| DNS network interface        | <b>eth0</b>                            |
| DNS forwarder                | <b>172.25.250.254</b>                  |
| DNS zone                     | <b>tokyo.lab.example.com</b>           |
| DNS reverse                  | <b>250.25.172.in-addr.arpa</b>         |
| DHCP network interface       | <b>eth0</b>                            |
| DHCP range                   | <b>172.25.250.50 to 172.25.250.100</b> |
| Name server provided by DHCP | <b>172.25.250.16</b>                   |
| Gateway provided by DHCP     | <b>172.25.250.254</b>                  |

6. Create the **tokyo.lab.example.com** DNS domain and make it available for use by the **Finance** organization at the **Tokyo** location. Satellite Capsule in Tokyo manages that domain.
7. Create the new **Tokyo Data Center** subnet, **172.25.250.0/24**, as follows:

#### Tokyo Subnet Details

| Field                     | Value                                               |
|---------------------------|-----------------------------------------------------|
| <b>Name</b>               | <b>Tokyo Data Center</b>                            |
| <b>Network Address</b>    | <b>172.25.250.0</b>                                 |
| <b>Network Prefix</b>     | <b>24</b>                                           |
| <b>Gateway Address</b>    | <b>172.25.250.254</b>                               |
| <b>Primary DNS Server</b> | <b>172.25.250.16</b>                                |
| <b>IPAM</b>               | <b>DHCP</b>                                         |
| <b>Start Of Ip Range</b>  | <b>172.25.250.50</b>                                |
| <b>End Of Ip Range</b>    | <b>172.25.250.100</b>                               |
| <b>Boot Mode</b>          | <b>DHCP</b>                                         |
| <b>Domains</b>            | <b>tokyo.lab.example.com</b>                        |
| <b>Capsules</b>           | <b>capsule.lab.example.com</b> for all the features |

8. Create a host group as follows:

#### Host Group Details

| Parameter                             | Value                                          |
|---------------------------------------|------------------------------------------------|
| Name                                  | <b>Finance Host Group</b>                      |
| Life-cycle Environment                | <b>Build</b>                                   |
| Content View                          | <b>Base</b>                                    |
| Satellite Capsule as a content source | <b>capsule.lab.example.com</b>                 |
| Network domain                        | <b>tokyo.lab.example.com</b>                   |
| IPv4 Subnet                           | <b>Tokyo Data Center<br/>(172.25.250.0/24)</b> |
| Operating system architecture         | <b>x86_64</b>                                  |
| Operating System                      | <b>RedHat 8.1</b>                              |
| Partition table                       | <b>Kickstart default</b>                       |
| PXE loader                            | <b>PXELinux BIOS</b>                           |
| Location                              | <b>Tokyo</b>                                   |
| Organization                          | <b>Finance</b>                                 |
| Activation key                        | <b>FinanceKey</b>                              |

9. Use the **satellite-installer** command on **capsule.lab.example.com** to activate the Discovery feature on Satellite Capsule Server. When done, install the package that provides the Discovery boot image and the package to configure Satellite Capsule Server as a proxy for the Discovery service. Restart the Satellite Capsule Server services.
10. Configure the **Tokyo Data Center** subnet as the discovery network.
11. Clone the following provisioning templates and adjust their contents to use the Discovery image from the **capsule.lab.example.com** Satellite Capsule Server.

#### Provisioning Templates to Clone

| Provisioning template name     | Clone name                             |
|--------------------------------|----------------------------------------|
| <b>PXELinux global default</b> | <b>Finance PXELinux global default</b> |
| <b>pxelinux_discovery</b>      | <b>finance_pxelinux_discovery</b>      |

12. Modify the Satellite Server settings to use the **Finance PXELinux global default** provisioning template for Discovery. Set **discovery** as the default PXE entry to use by default. Set the default organization of discovered hosts to **Finance** and the location to **Tokyo**.
13. Build and deploy your **Finance PXELinux global default** provisioning template to Satellite Capsule Server.

14. Access the **servere** console and initiate a PXE boot. Wait for the discovery process to complete.
15. In the Satellite Server web UI, create a **servere** host resource for the newly discovered system. Use the **Finance Host Group** host group. Set the **root** password to **redhat123**.



### Note

As soon as you submit your request to create the **servere** host resource, the system reboots **servere**. To be able to interrupt the default boot process on time, open **servere** console in advance.

16. Boot **servere** in PXE, and then let the provisioning process complete.

## Evaluation

As the **student** user on the **workstation** machine, use the **lab provision-review grade** command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab provision-review grade
```

## Finish

As the **student** user on the **workstation** machine, use the **lab** command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab provision-review finish
```

This concludes the lab.

## ► Solution

# Provisioning Hosts

In this lab, you will configure the discovery service, and detect and deploy a host using the discovery method.

## Outcomes

You should be able to:

- Create the Satellite Server resources for provisioning.
- Configure the Discovery feature on a Satellite Capsule Server.
- Discover new hosts on the network.
- Provision discovered hosts.

## Before You Begin

This exercise requires a running Satellite Capsule Server on the **capsule** host, which you installed in a preceding exercise.

As the **student** user on the **workstation** machine, use the **lab provision-review start** command to prepare your system for this exercise.

This command determines if the **satellite** and **capsule** hosts are reachable on the network and verifies that the required resources exist.

```
[student@workstation ~]$ lab provision-review start
```

1. Enable and synchronize the following Red Hat Enterprise Linux 8 kickstart repositories for the **Tokyo** location in the **Finance** organization:
  - **Red Hat Enterprise Linux 8 for x86\_64 - AppStream (Kickstart)**
  - **Red Hat Enterprise Linux 8 for x86\_64 - BaseOS (Kickstart)**
  - 1.1. On **workstation**, launch Firefox and navigate to <https://satellite.lab.example.com> to access the Red Hat Satellite Server web UI.
  - 1.2. Log in to the Satellite Server web UI as **admin** using **redhat** as the password.
  - 1.3. Choose the **Finance** organization and **Tokyo** location from the main menu.
  - 1.4. Click **Content → Red Hat Repositories** to access the **Red Hat Repositories** page.
  - 1.5. From the **RPM** list, select **Kickstart** and clear **RPM**. Toggle the **Recommended Repositories** button to **ON** to list only the recommended repositories.
  - 1.6. Expand **Red Hat Enterprise Linux 8 for x86\_64 - AppStream (Kickstart)** and **Red Hat Enterprise Linux 8 for x86\_64 - BaseOS (Kickstart)**. Click the plus sign (+) next to **x86\_64 8.1** for each of the repositories to enable them.

- 1.7. Click **Content** → **Products** to open the **Products** page.
- 1.8. Click **Red Hat Enterprise Linux for x86\_64**. The **Repositories** tab displays.
- 1.9. Select the check boxes for the **Red Hat Enterprise Linux 8 for x86\_64 - AppStream Kickstart x86\_64 8.1** and **Red Hat Enterprise Linux 8 for x86\_64 - BaseOS Kickstart x86\_64 8.1** repositories. Make sure that you select the kickstart repositories and not the RPM repositories. Click **Sync Now** to begin the repository synchronization.

Wait for the synchronization task to complete. The task can take up to 15 minutes to finish.
2. Add the two kickstart repositories to the **Base** content view. When done, publish and promote the content view to the **Build** life-cycle environment.
  - 2.1. Click **Content** → **Content Views** to access the **Content Views** page. Click the **Base** content view.
  - 2.2. Click **Yum Content** → **Repositories**, and then click the **Add** tab.
  - 2.3. Select the check boxes for the **Red Hat Enterprise Linux 8 for x86\_64 - AppStream Kickstart x86\_64 8.1** and **Red Hat Enterprise Linux 8 for x86\_64 - BaseOS Kickstart x86\_64 8.1** repositories, and then click **Add Repositories**.
  - 2.4. Click **Publish New Version**, and then click **Save**. Wait for the publishing process to complete. The task can take up to 15 minutes to finish.
  - 2.5. When the publishing process is complete, click **Promote**. Select the **Build** life-cycle environment, and then click **Promote Version**.
3. Set the location of Satellite Capsule Server installed on the **capsule** host to **Tokyo**. Set the organization to **Finance**. Associate the **Build** life-cycle environment with Satellite Capsule Server and initiate a synchronization.

Remove the other organizations, locations, and life-cycle environments from the Satellite Capsule Server configuration.

  - 3.1. Choose the **Any Organization** organization and **Any Location** location from the main menu.
  - 3.2. Click **Infrastructure** → **Capsules** to access the **Capsules** page, and then click **Edit** at the end of the **capsule.lab.example.com** row.
  - 3.3. Click the **Locations** tab, and then click **Tokyo** to add it to the **Selected items** list.

Click the **Organizations** tab, and then click **Finance** to add it to the **Selected items** list.

Click the **Lifecycle Environments** tab, and then click **Build** to add it to the **Selected items** list.
  - 3.4. Click **Submit**.
  - 3.5. On the **Capsules** page, click the **capsule.lab.example.com** link. Click **Synchronize** → **Optimized Sync** to start a new synchronization. Do not wait for the synchronization to complete, but proceed with the next step.

- 3.6. Choose the **Finance** organization and **Tokyo** location from the main menu.
4. Confirm that the Finance team in Tokyo can use the following resources for provisioning Red Hat Enterprise Linux 8.1 systems.
- **Kickstart default PXELinux** template
  - **Kickstart default** provisioning template
  - **Kickstart default** partition table
- Make sure that the **RedHat 8.1** operating system resource uses those templates by default.
- 4.1. Click **Hosts** → **Provisioning Templates** to access the **Provisioning Templates** page.
  - 4.2. In the filter field, enter **kind = PXELinux**, and then click **Search**.
  - 4.3. Click the **Kickstart default PXELinux** link.
  - 4.4. Click the **Association** tab and make sure that the **RedHat 8.1** operating system displays in the **Selected items** list.
  - 4.5. Click the **Locations** tab and make sure that the **Tokyo** location displays in the **Selected items** list.
  - 4.6. Repeat the previous substeps for the **Kickstart default** provisioning template. Use the **kind = provision** search filter to find that template more easily.
  - 4.7. Click **Hosts** → **Partition Tables** to access the **Partition Tables** page, and then click the **Kickstart default** link.
  - 4.8. Click the **Locations** tab and make sure that the **Tokyo** location displays in the **Selected items** list.
  - 4.9. Click **Hosts** → **Operating Systems** to access the **Operating Systems** page, and then click the **RedHat 8.1** link.
  - 4.10. On the **Partition Table** tab, confirm that the **Kickstart default** partition table displays in the **Selected items** list.
  - 4.11. On the **Templates** tab, confirm that **PXELinux template** is set to **Kickstart default PXELinux** and that **Provisioning template** is set to **Kickstart default**.
5. Enable the DNS, DHCP, and TFTP services in Satellite Capsule Server on **capsule.lab.example.com**. Use the following settings.

#### Service Parameters

| Parameter             | Value                          |
|-----------------------|--------------------------------|
| DNS network interface | <b>eth0</b>                    |
| DNS forwarder         | <b>172.25.250.254</b>          |
| DNS zone              | <b>tokyo.lab.example.com</b>   |
| DNS reverse           | <b>250.25.172.in-addr.arpa</b> |

| Parameter                    | Value                                  |
|------------------------------|----------------------------------------|
| DHCP network interface       | <b>eth0</b>                            |
| DHCP range                   | <b>172.25.250.50 to 172.25.250.100</b> |
| Name server provided by DHCP | <b>172.25.250.16</b>                   |
| Gateway provided by DHCP     | <b>172.25.250.254</b>                  |

- 5.1. On **workstation**, use **ssh** to log in to **capsule** as **student**. Use **sudo -i** to switch to **root**.

```
[student@workstation ~]$ ssh student@capsule
[student@capsule ~]$ sudo -i
[root@capsule ~]#
```

- 5.2. Use the **satellite-installer** command to enable and configure the DNS, DHCP, and TFTP services on **capsule.lab.example.com**.

For your convenience, you can copy and paste the following command from the **/root/satellite-installer-review.txt** file.

```
[root@capsule ~]# satellite-installer --scenario capsule \
--foreman-proxy-dns true \
--foreman-proxy-dns-interface eth0 \
--foreman-proxy-dns-forwarders 172.25.250.254 \
--foreman-proxy-dns-zone tokyo.lab.example.com \
--foreman-proxy-dns-reverse 250.25.172.in-addr.arpa \
--foreman-proxy-dhcp true \
--foreman-proxy-dhcp-interface eth0 \
--foreman-proxy-dhcp-range "172.25.250.50 172.25.250.100" \
--foreman-proxy-dhcp-nameservers 172.25.250.16 \
--foreman-proxy-dhcp-gateway 172.25.250.254 \
--foreman-proxy-tftp true
...output omitted...
Installing          Done    [100%] [.....]
Success!
* Capsule is running at https://capsule.lab.example.com:9090
The full log is at /var/log/foreman-installer/capsule.log
```

- 5.3. Verify that the DNS, DHCP, and TFTP features are active on Satellite Capsule Server. In the Satellite web UI, click **Infrastructure** → **Capsules**, and then click the **capsule.lab.example.com** link.

Confirm that the **Active features** section lists the DNS, DHCP, and TFTP features. If not, click **Refresh features**.

6. Create the **tokyo.lab.example.com** DNS domain and make it available for use by the **Finance** organization at the **Tokyo** location. Satellite Capsule in Tokyo manages that domain.

- 6.1. Click **Infrastructure** → **Domains**, and then click **Create Domain**.

- 6.2. Enter **tokyo.lab.example.com** in the **DNS Domain** field.

- 6.3. Choose **capsule.lab.example.com** as the **DNS Capsule**.
  - 6.4. Click the **Locations** tab and verify that the **Tokyo** location is already associated with the new domain. If not, click **Tokyo** to add it to the **Selected items** list.
  - 6.5. Click the **Organizations** tab and verify that the **Finance** organization is already associated with the new domain. If not, click **Finance** to add it to the **Selected items** list.
  - 6.6. Click **Submit**.
7. Create the new **Tokyo Data Center** subnet, **172.25.250.0/24**, as follows:

#### Tokyo Subnet Details

| Field                     | Value                                               |
|---------------------------|-----------------------------------------------------|
| <b>Name</b>               | <b>Tokyo Data Center</b>                            |
| <b>Network Address</b>    | <b>172.25.250.0</b>                                 |
| <b>Network Prefix</b>     | <b>24</b>                                           |
| <b>Gateway Address</b>    | <b>172.25.250.254</b>                               |
| <b>Primary DNS Server</b> | <b>172.25.250.16</b>                                |
| <b>IPAM</b>               | <b>DHCP</b>                                         |
| <b>Start Of Ip Range</b>  | <b>172.25.250.50</b>                                |
| <b>End Of Ip Range</b>    | <b>172.25.250.100</b>                               |
| <b>Boot Mode</b>          | <b>DHCP</b>                                         |
| <b>Domains</b>            | <b>tokyo.lab.example.com</b>                        |
| <b>Capsules</b>           | <b>capsule.lab.example.com</b> for all the features |

- 7.1. Click **Infrastructure** → **Subnets** to access the **Subnets** page, and then click **Create Subnet**.
- 7.2. Complete the page according to the preceding table. Do not modify any other field, and do not submit the form yet.
- 7.3. Click the **Domains** tab, and then add the **tokyo.lab.example.com** domain to the **Selected items** list.
- 7.4. Click the **Capsules** tab and set all the fields to **capsule.lab.example.com**.
- 7.5. Click the **Locations** tab and verify that the **Tokyo** location is already associated with the new subnet. If not, click **Tokyo** to add it to the **Selected items** list.
- 7.6. Click the **Organizations** tab and verify that the **Finance** organization is already associated with the new subnet. If not, click **Finance** to add it to the **Selected items** list.

- 7.7. Click **Submit**.
8. Create a host group as follows:

#### Host Group Details

| Parameter                             | Value                                          |
|---------------------------------------|------------------------------------------------|
| Name                                  | <b>Finance Host Group</b>                      |
| Life-cycle Environment                | <b>Build</b>                                   |
| Content View                          | <b>Base</b>                                    |
| Satellite Capsule as a content source | <b>capsule.lab.example.com</b>                 |
| Network domain                        | <b>tokyo.lab.example.com</b>                   |
| IPv4 Subnet                           | <b>Tokyo Data Center<br/>(172.25.250.0/24)</b> |
| Operating system architecture         | <b>x86_64</b>                                  |
| Operating System                      | <b>RedHat 8.1</b>                              |
| Partition table                       | <b>Kickstart default</b>                       |
| PXE loader                            | <b>PXElinux BIOS</b>                           |
| Location                              | <b>Tokyo</b>                                   |
| Organization                          | <b>Finance</b>                                 |
| Activation key                        | <b>FinanceKey</b>                              |

- 8.1. Click **Configure → Host Groups**, and then click **Create Host Group**.
- 8.2. Complete the page according to the preceding table. You have to navigate between the different tabs to provide all the parameters.
- 8.3. Click **Submit**.
9. Use the **satellite-installer** command on **capsule.lab.example.com** to activate the Discovery feature on Satellite Capsule Server. When done, install the package that provides the Discovery boot image and the package to configure Satellite Capsule Server as a proxy for the Discovery service. Restart the Satellite Capsule Server services.

- 9.1. On **workstation**, use **ssh** to log in to **capsule** as **student**. Use **sudo -i** to switch to **root**.

```
[student@workstation ~]$ ssh student@capsule
[student@capsule ~]$ sudo -i
[root@capsule ~]#
```

- 9.2. Use the **satellite-installer** command to enable the Discovery plug-in.

```
[root@capsule ~]# satellite-installer --scenario capsule \
--enable-foreman-proxy-plugin-discovery
...output omitted...
```

9.3. Install the *foreman-discovery-image* and *rubygem-smart\_proxy\_discovery* packages.

```
[root@capsule ~]# yum install foreman-discovery-image \
rubygem-smart_proxy_discovery
...output omitted...
```

9.4. Restart the Satellite Capsule Server services.

```
[root@capsule ~]# satellite-maintain service restart
...output omitted...
```

9.5. Verify that the Discovery feature is active on Satellite Capsule Server. In the Satellite web UI, click **Infrastructure** → **Capsules**, and then click the **capsule.lab.example.com** link.

Confirm that the **Active features** section lists the Discovery feature. If not, click **Refresh features**.

**10.** Configure the **Tokyo Data Center** subnet as the discovery network.

10.1. Click **Infrastructure** → **Subnets** to access the **Subnets** page, and then click the **Tokyo Data Center** link.

10.2. Click the **Capsules** tab, and then set **capsule.lab.example.com** as the **Discovery Capsule**.

10.3. Click **Submit**.

**11.** Clone the following provisioning templates and adjust their contents to use the Discovery image from the **capsule.lab.example.com** Satellite Capsule Server.

#### Provisioning Templates to Clone

| Provisioning template name | Clone name                      |
|----------------------------|---------------------------------|
| PXElinux global default    | Finance PXElinux global default |
| pxelinux_discovery         | finance_pxelinux_discovery      |

11.1. Click **Hosts** → **Provisioning Templates** to access the **Provisioning Templates** page.

11.2. In the filter field, enter **kind = PXElinux**, and then click **Search**.

11.3. Click **Clone** at the end of the **PXElinux global default** row, and set the name of the cloned template to **Finance PXElinux global default**.

11.4. In the template, replace the reference to the **pxelinux\_discovery** snippet with **finance\_pxelinux\_discovery**. You clone that second template in a following step.

```

<##
kind: PXELinux
name: PXELinux global default
model: ProvisioningTemplate
%>
<%# Used to boot unknown hosts, do not associate or change the name. %>

DEFAULT menu
MENU TITLE Booting unknown host (ESC to stop)
TIMEOUT 200
ONTIMEOUT <%= global_setting("default_pxe_item_global", "local") %>

<%= snippet "pxelinux_chainload" %>

<%= snippet "finance_pxelinux_discovery" %>

<% unless @profiles.nil? -%>
...output omitted...

```

The **ONTIMEOUT** line specifies the default PXE entry to use after the timeout expires. The preceding ERB template sets that value to **default\_pxe\_item\_global**. That **default\_pxe\_item\_global** parameter corresponds to the **Default PXE global template entry** Satellite Server setting, in **Administer** → **Settings**, on the **Provisioning** tab. In a following step, you modify that setting for new systems to automatically boot into the Discovery image.

- 11.5. Click the **Locations** tab, and then add **Tokyo** to the **Selected items** list.
- 11.6. Click the **Organizations** tab, and then add **Finance** to the **Selected items** list.
- 11.7. Click **Submit**.
- 11.8. In the **Provisioning Templates** page, in the filter field, enter **pxelinux\_discovery**, and then click **Search**.
- 11.9. Click **Clone** at the end of the **pxelinux\_discovery** row, and set the name of the cloned template to **finance\_pxelinux\_discovery**.
- 11.10. In the template, set the **proxy.url** variable to **https://capsule.lab.example.com:9090** and the **proxy.type** variable to **proxy**.

```

...output omitted...
MENU LABEL Foreman Discovery Image
KERNEL boot/fdi-image/vmlinuz0
APPEND initrd=boot/fdi-image/initrd0.img rootflags=loop root=/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 nokaslr nomodeset proxy.url=https://
capsule.lab.example.com:9090 proxy.type=proxy
IPAPPEND 2
...output omitted...

```

Notice that the label for this PXE entry is **discovery**. This is the name you set in the **Default PXE global template entry** Satellite Server setting in a following step.

- 11.11. Click the **Locations** tab, and then add **Tokyo** to the **Selected items** list.
- 11.12. Click the **Organizations** tab, and then add **Finance** to the **Selected items** list.
- 11.13. Click **Submit**.
12. Modify the Satellite Server settings to use the **Finance PXELinux global default** provisioning template for Discovery. Set **discovery** as the default PXE entry to use by default. Set the default organization of discovered hosts to **Finance** and the location to **Tokyo**.
  - 12.1. Click **Administer** → **Settings**, and then click the **Provisioning** tab.
  - 12.2. Set the **Global default PXELinux template** setting to **Finance PXELinux global default** and set the **Default PXE global template entry** to **discovery**.
  - 12.3. Click the **Discovered** tab. Set **Discovery organization** to **Finance** and **Discovery location** to **Tokyo**.
13. Build and deploy your **Finance PXELinux global default** provisioning template to Satellite Capsule Server.
  - 13.1. Click **Hosts** → **Provisioning Templates**, and then click **Build PXE Default**.
  - 13.2. Click **OK** to confirm the update of the default PXE menu.
14. Access the **servere** console and initiate a PXE boot. Wait for the discovery process to complete.
  - 14.1. Locate the icon for **servere** console, as appropriate for your classroom environment. Open the console.
  - 14.2. To reboot, send a **Ctrl+Alt+Del** to your system using the relevant keyboard, virtual, or menu entry.
  - 14.3. Press the **Spacebar** at the **Boot options** screen to stop the countdown.
  - 14.4. Select **Network boot from device net0**, and press **Enter**.
  - 14.5. Let the timeout expire; the system boots in the Discovery image.



#### Note

The PXE menu always highlights the first entry, **Default local boot**, even though PXE boots the **Foreman Discovery Image** entry after the timeout.

- 14.6. At the **Welcome to Foreman Discovery** screen, let the timeout expire. Do not interrupt that timeout. Wait for the **Discovery status** screen to display.
15. In the Satellite Server web UI, create a **servere** host resource for the newly discovered system. Use the **Finance Host Group** host group. Set the **root** password to **redhat123**.



#### Note

As soon as you submit your request to create the **servere** host resource, the system reboots **servere**. To be able to interrupt the default boot process on time, open **servere** console in advance.

- 15.1. Click **Hosts** → **Discovered Hosts**, and then click **Provision** at the right of the discovered host.
  - 15.2. Set **Host Group** to **Finance Host Group**, **Organization** to **Finance**, and **Location** to **Tokyo**. Click **Create Host**.
  - 15.3. On the **Operating System** tab, click the **Change the password** pencil icon at the right of the **Root Password** field, and then set the password to **redhat123**. Click **Resolve** to confirm that the system can correctly retrieve the templates for provisioning.
  - 15.4. Click the **Host** tab, and then set the name to **servere**.
  - 15.5. Click **Submit**.
16. Boot **servere** in PXE, and then let the provisioning process complete.
    - 16.1. **servere** should reboot automatically. If not, from **servere** console, send a **Ctrl+Alt+Del** to reboot.
    - 16.2. Press the **Spacebar** at the **Boot options** screen to stop the countdown.
    - 16.3. Select **Network boot from device net0**, and press **Enter**. The provisioning process starts automatically.
    - 16.4. Wait for the installation to complete and the system to reboot. From the console, log in as **root** using **redhat123** as the password.  
Use the **hostname** command to verify the system name.

```
[root@servere ~]# hostname  
servere.tokyo.lab.example.com
```

When done, log off from **servere** and close the console.

This concludes the lab.

## Evaluation

As the **student** user on the **workstation** machine, use the **lab provision-review grade** command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab provision-review grade
```

## Finish

As the **student** user on the **workstation** machine, use the **lab** command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab provision-review finish
```

# Summary

---

In this chapter, you learned:

- Satellite Server provides several provisioning methods, and can deploy bare physical systems, virtual machines, and cloud instances.
- Provisioning templates utilize Embedded Ruby (ERB) syntax to combine plain text with Ruby code.
- Capsule services, such as DNS, DHCP, and TFTP, can be enabled and configured on Satellite Server using the **satellite-installer** command.
- Satellite Server offers domain and subnet resources for managing network configuration.
- Host groups expedite host provisioning by acting as a template for provisioning resource configurations.
- The Foreman Discovery feature uses a discovery image to detect unprovisioned systems on the network.

## Chapter 9

# Managing Red Hat Satellite Using the API

### Goal

Integrate Red Hat Satellite functionality with custom scripts or external applications that access the API over HTTP

### Objectives

- Query the Red Hat Satellite REST API to observe Satellite functionality, syntax, and authentication methods.
- Integrate common Satellite tasks and object queries into custom scripts and external applications, including popular languages such as curl, Ruby, and Python.
- Describe the Hammer CLI syntax, and compare the Hammer and scripted API use cases.

### Sections

- Querying the Red Hat Satellite API (and Guided Exercise)
- Integrating Red Hat Satellite Functionality in Applications (and Guided Exercise)
- Using the Hammer CLI as an API Interface (and Guided Exercise)

### Lab

Managing Red Hat Satellite Using the API

# Querying the Red Hat Satellite API

## Objectives

After completing this section, you should be able to query the Red Hat Satellite REST API to observe Satellite functionality, syntax, and authentication methods.

## Describing Red Hat Satellite API

Red Hat Satellite includes a Representational State Transfer (REST) API. You can use this REST API to control a Satellite environment without using the Satellite web UI. The Satellite REST API makes it easy to combine Satellite functionality with any third-party application in your environment that can communicate over HTTP.

The Satellite REST API supports the following use cases:

- Integration of Satellite functionality with production IT systems or third-party applications.
- Automation of maintenance tasks to ensure an operational Satellite environment.
- Automation of recurring tasks using scripts.

## Describing Satellite API Syntax

Use the following `curl` command synopsis to make a Satellite REST API request.

```
[user@host]$ curl --request HTTP_VERB \ ①
  --insecure \ ②
  --user sat_username:sat_password \ ③
  --data @file.json \ ④
  --header "Content-Type:application/json" \ ⑤
  API_ROUTE \ ⑥
| python -m json.tool ⑦
```

- ➊ The `HTTP_VERB` argument signifies the API call that you intend to make with the `curl` command. The API call determines the action to perform on the Satellite environment. Use the `--request` option to specify an appropriate HTTP verb. For example, the `GET` HTTP verb retrieves data from the Satellite API. The `POST` HTTP verb submits data to the Satellite API to create a resource. The `PUT` HTTP verb submits data to the Satellite API to modify a resource. The `DELETE` HTTP verb instructs the Satellite API to delete a resource.
- ➋ Use the `--insecure` option to skip validating SSL certificates.
- ➌ Use the `--user` option to specify user authentication credentials. The `sat_username` argument represents the Satellite user name. The `sat_password` argument represents the password of the specified Satellite user.
- ➍ Satellite API accepts data in JSON format only. Use the `--data` option to pass JSON data to the Satellite API. You can pass the JSON data from either standard input (`stdin`), or from a file. The `file.json` argument represents a file containing data in JSON format. This option is only applicable to the `POST` and `PUT` HTTP verbs.
- ➎ Use the `--header` option to pass extra headers to the API request. For JSON data, use the `Content-Type:application/json` extra header.
- ➏ Signifies the API route of the request. For example, the `GET` HTTP verb with the `https://satellite.example.com/katello/api/activation_keys` API route lists the

activation keys in the Satellite server. In Satellite 6, the default API version is **2**. You can ignore adding **v2** to the URL for API calls if you are using the default version.

- ⑦ For better readability, feed the output from the **curl** command to the **json.tool** Python module.

## Retrieving Data from Satellite API

Use the **GET** HTTP verb with the API request to retrieve information about a particular resource from the Satellite API.

```
[user@host]$ curl --request GET \
--insecure \
--user sat_username:sat_password \
API_ROUTE \
| python -m json.tool
```

For example, the following **curl** command retrieves the list of organizations in your Satellite Server. Each returned entry includes various details about the organization.

```
[user@host]$ curl --request GET \
--insecure \
--user admin:redhat \
https://satellite.lab.example.com/katello/api/organizations \
| python -m json.tool
% Total    % Received % Xferd  Average Speed   Time     Time      Current
   Dload  Upload   Total Spent  Left  Speed
100    713     0    713     0       0  2951      0  --::-- --::-- --::-- 2958
{
  "page": 1,
  "per_page": 20,
  "results": [
    {
      "created_at": "2019-11-08 12:15:18 UTC",
      "description": null,
      "id": 1,
      "label": "Default_Organization",
      "name": "Default Organization",
      "title": "Default Organization",
      "updated_at": "2019-11-08 12:15:18 UTC"
    },
    ...output omitted...
  ],
  "search": null,
  "sort": {
    "by": null,
    "order": null
  },
  "subtotal": 3,
  "total": 3
}
```

## Creating a Resource from Satellite API

Use the **POST** HTTP verb with an API request to create a resource in the Satellite Server. Ensure that you submit the data to the API in JSON format while running a **POST** HTTP request to the Satellite Server. To submit the JSON data from a file, create the file with its content in JSON format. This JSON file contains values for different attributes of the particular resource that you are creating.

```
{"parameter1": "value1", "parameter2": "value2", "parameter3": "value3"}
```

After creating the JSON file, run the **curl** command, passing the JSON data to the Satellite API to create the resource.

```
[user@host]$ curl --request POST \
--insecure \
--user sat_username:sat_password \
--header "Content-Type:application/json" \
--data file.json \
API_ROUTE \
| python -m json.tool
```

## Modifying a Resource from Satellite API

Use the **PUT** HTTP verb with the API request to modify an existing resource in the Satellite Server. Ensure that you are submitting the data to the API in JSON format while running a **PUT** HTTP request to the Satellite Server. To submit the JSON data from a file, create the file with its content in JSON format. This JSON file contains values for different attributes of the particular resource that you are modifying. Based on those values, the parameters of the intended resource are modified.

```
{"parameter1": "value1", "parameter2": "value2", "parameter3": "value3"}
```

After creating the JSON file, run the **curl** command, passing the JSON data to the Satellite API to modify the existing resource.

```
[user@host]$ curl --request PUT \
--insecure \
--user sat_username:sat_password \
--header "Content-Type:application/json" \
--data file.json \
API_ROUTE \
| python -m json.tool
```

## Deleting a Resource from Satellite API

Use the **DELETE** HTTP verb with the API request to delete an existing resource from the Satellite Server. To delete the resource, specify an API route that includes the numeric identifier of the intended resource. Based on this numeric identifier, the API command instructs the Satellite Server to delete a particular resource.

```
[user@host]$ curl --request DELETE \  
--insecure \  
--user sat_username:sat_password \  
--header "Content-Type:application/json" \  
API_ROUTE \  
| python -m json.tool
```

For example, the following **curl** command deletes the **Test** organization, which has an ID of **15**.

```
[user@host]$ curl --request DELETE \  
--insecure \  
--user admin:redhat \  
--header "Content-Type:application/json" \  
https://satellite.lab.example.com/katello/api/organizations/15 \  
| python -m json.tool
```



## References

### Red Hat Satellite 6 REST API documentation

<https://access.redhat.com/solutions/2191451>

For more information, refer to the *API Requests in Different Languages* chapter in the *Red Hat Satellite 6.6 API Guide* at

[https://access.redhat.com/documentation/en-us/red\\_hat\\_satellite/6.6/html-single/api\\_guide/index#chap-Red\\_Hat\\_Satellite-API\\_Guide-API\\_Requests\\_in\\_Different\\_Languages](https://access.redhat.com/documentation/en-us/red_hat_satellite/6.6/html-single/api_guide/index#chap-Red_Hat_Satellite-API_Guide-API_Requests_in_Different_Languages)

## ► Guided Exercise

# Querying the Red Hat Satellite API

In this exercise, you will query Red Hat Satellite through the REST API using authenticated curl commands.

### Outcomes

You should be able to use the Red Hat Satellite API to perform common administrative queries.

### Before You Begin

Log in as the **student** user on **workstation** using **student** as the password.

Run the **lab api-overview start** command. The command verifies that the **satellite** host is available.

```
[student@workstation ~]$ lab api-overview start
```

- 1. On **workstation**, ssh to **satellite** as **student**.

```
[student@workstation ~]$ ssh student@satellite
[student@satellite ~]$
```

- 2. List host names:

```
[student@satellite ~]$ curl --request GET --user admin:redhat \
https://satellite.lab.example.com/api/v2/hosts \
| python -m json.tool | grep '"name":' \
...output omitted...
{
    "name": "satellite.lab.example.com",
    "name": "satellite.lab.example.com",
    "name": "satellite.lab.example.com",
    "name": "servera.lab.example.com",
    "name": "Base"
    "name": "Development"
    "name": "serverb.lab.example.com",
    "name": "Default Organization View"
    "name": "Library"
    "name": "serverc.lab.example.com",
    "name": "Base"
    "name": "Build"
    "name": "serverd.lab.example.com",
```

- 3. List the name, ID, and various organization information. Results will vary depending on your environment.

```
[student@satellite ~]$ curl --request GET --user admin:redhat \
https://satellite.lab.example.com/katello/api/organizations \
| python -m json.tool
...output omitted...
{
  ...
  "results": [
    {
      "created_at": "2019-10-22 13:38:32 UTC",
      "description": null,
      "id": 1,
      "label": "Default_Organization",
      "name": "Default Organization",
      "title": "Default Organization",
      "updated_at": "2019-10-22 13:38:32 UTC"
    },
    {
      "created_at": "2019-10-29 22:54:43 UTC",
      "description": "Finance Department",
      "id": 5,
      "label": "Finance",
      "name": "Finance",
      "title": "Finance",
      "updated_at": "2019-10-29 22:58:35 UTC"
    },
    {
      "created_at": "2019-10-29 22:44:01 UTC",
      "description": "Operations Department",
      "id": 3,
      "label": "Operations",
      "name": "Operations",
      "title": "Operations",
      "updated_at": "2019-10-29 22:47:52 UTC"
    }
  ...
}
```

- 4. Select the ID value assigned to the **Finance** organization to list its related life-cycle environments in the API query.

```
[student@satellite ~]$ curl --request GET --user admin:redhat \
https://satellite.lab.example.com/katello/api/organizations/5/environments \
| python -m json.tool
...output omitted...
{
  ...
  "results": [
    {
      ...
    }
  ...
  "name": "Build",
  "organization": {
    "id": 5,
    "label": "Finance",
    ...
  }
}
```

```
        "name": "Finance"
    },
...output omitted...
    "name": "Deploy",
    "organization": {
        "id": 5,
        "label": "Finance",
        "name": "Finance"
    },
...output omitted...
    "name": "Library",
    "organization": {
        "id": 5,
        "label": "Finance",
        "name": "Finance"
    },
...output omitted...
    "name": "Test",
    "organization": {
        "id": 5,
        "label": "Finance",
        "name": "Finance"
    },
...output omitted...
}
```

- ▶ 5. Create a new **Training** organization with **Global product training** as the description. Output will vary depending on your environment.

- 5.1. Create a **/home/student/create-training-organization.json** data file with organization parameters.

```
[student@satellite ~]$ vi /home/student/create-training-organization.json
{"name": "Training", "description": "Global product training"}
```

- 5.2. Create the **Training** organization.

Note the **"id":** and **"library\_id":** values to use in an upcoming step.

```
[student@satellite ~]$ curl --header "Content-Type:application/json" \
--request POST --user admin:redhat \
--data @create-training-organization.json \
https://satellite.lab.example.com/katello/api/organizations \
| python -m json.tool
...output omitted...
],
"created_at": "2019-11-14 02:40:19 UTC",
"default_content_view_id": 21,
"description": "Global product training",
"domains": [],
"environments": [],
"hostgroups": [],
"hosts_count": 0,
"id": 17,
```

```
"label": "Training",
"library_id": 40,
"media": [],
"name": "Training",
"owner_details": {
...output omitted...
```

- 6. Create a life-cycle environment named **Devel** in the **Training** organization.

- 6.1. Create a `/home/student/create-devel-environment.json` data file with environment parameters. Use the organization ID and Library ID noted earlier in the create **Training** API results.

```
[student@satellite ~]$ vi /home/student/create-devel-environment.json
>{"organization_id":17, "name":"Devel", "prior":40}
```

- 6.2. Create the **Devel** environment.

Note the `"id"`: number assigned to the `"name": "Devel"` environment to use in an upcoming step.

```
[student@satellite ~]$ curl --header "Content-Type:application/json" \
--request POST --user admin:redhat \
--data @create-devel-environment.json \
https://satellite.lab.example.com/katello/api/environments \
| python -m json.tool
...output omitted...
{
  "counts": {
    "content_hosts": 0,
    "content_views": 0
  },
  "created_at": "2019-11-14 04:20:00 UTC",
  "description": null,
  "id": 44,
  "label": "Devel",
  "library": false,
  "name": "Devel",
  "organization": {
    "id": 17,
    "label": "Training",
    "name": "Training"
  },
  "organization_id": 17,
  "permissions": {
    "create_lifecycle_environments": true,
    "destroy_lifecycle_environments": true,
    "edit_lifecycle_environments": true,
    "promote_or_remove_content_views_to_environments": true,
    "view_lifecycle_environments": true
  },
  "prior": {
    "id": 40,
    "name": "Library"
  }
}
```

```
},
"registry_name_pattern": null,
"registry_unauthenticated_pull": false,
"successor": null,
"updated_at": "2019-11-14 04:20:00 UTC"
}
...output omitted...
```

- ▶ 7. Update the **Devel** environment adding **Pre-testing development** as the environment description.

- 7.1. Create a **/home/student/update-devel-environment.json** data file with the environment parameters to update.

```
[student@satellite ~]$ vi /home/student/update-devel-environment.json
{"description":"Pre-testing development"}
```

- 7.2. Modify the **curl** command to use the **PUT** verb with the **update-devel-environment.json** data file and the **Devel** environment ID.

```
[student@satellite ~]$ curl --header "Content-Type:application/json" \
--request PUT --user admin:redhat \
--data @update-devel-environment.json \
https://satellite.lab.example.com/katello/api/environments/44 \
| python -m json.tool
...output omitted...
{
...output omitted...
    "created_at": "2019-11-14 04:20:00 UTC",
    "description": "Pre-testing development",
    "id": 44,
    "label": "Devel",
    "library": false,
    "name": "Devel",
    "organization": {
        "id": 17,
        "label": "Training",
        "name": "Training"
    },
...output omitted...
    "prior": {
        "id": 40,
        "name": "Library"
    },
...output omitted...
    "registry_name_pattern": null,
    "registry_unauthenticated_pull": false,
    "successor": null,
    "updated_at": "2019-11-14 04:27:19 UTC"
}
...output omitted...
```

- 8. Add a successor to the **Devel** environment named **Test** with **Pre-production testing** as the description.
- 8.1. Create the **/home/student/create-test-environment.json** data file with environment parameters, including a description. Use the ID value for the **Devel** environment to identify the prior environment value for **Test**.

```
[student@satellite ~]$ vi /home/student/create-test-environment.json
>{"organization_id": "17", "name": "Test", "description": "Pre-production testing",
 "prior": 44}
```

- 8.2. After the command executes, note the **"id"**: number assigned to the **"name"**: **"Test"** environment to use in an upcoming step.

```
[student@satellite ~]$ curl --header "Content-Type:application/json" \
--request POST --user admin:redhat \
--data @create-test-environment.json \
https://satellite.lab.example.com/katello/api/environments \
| python -m json.tool
...output omitted...
{
    "created_at": "2019-11-17 23:53:26 UTC",
    "description": null,
    "id": 45,
    "label": "Test",
    "library": false,
    "name": "Test",
    "organization": {
        "id": 17,
        "label": "Training",
        "name": "Training"
    },
    ...output omitted...
    "prior": {
        "id": 44,
        "name": "Devel"
    },
    "registry_name_pattern": null,
    "registry_unauthenticated_pull": false,
    "successor": null,
    "updated_at": "2019-11-17 23:53:26 UTC"
}
...output omitted...
```

- 9. Add a successor to the **Test** environment named **Prod** with **Production environment** as the description.
- 9.1. Create the **/home/student/create-prod-environment.json** data file with environment parameters, including a description. Use the ID value for the **Test** environment to identify the prior environment value for **Prod**.

```
[student@satellite ~]$ vi /home/student/create-prod-environment.json
>{"organization_id": 17, "name": "Prod", "description": "Production environment",
 "prior": 45}
```

9.2. Create the **Prod** environment.

```
[student@satellite ~]$ curl --header "Content-Type:application/json" \
--request POST --user admin:redhat \
--data @create-prod-environment.json \
https://satellite.lab.example.com/katello/api/environments \
| python -m json.tool
...output omitted...
  "created_at": "2019-11-18 00:07:37 UTC",
  "description": "Production environment",
  "id": 46,
  "label": "Prod",
  "library": false,
  "name": "Prod",
  "organization": {
    "id": 17,
    "label": "Training",
    "name": "Training"
  },
...output omitted...
  "prior": {
    "id": 45,
    "name": "Test"
  },
  "registry_name_pattern": null,
  "registry_unauthenticated_pull": false,
  "successor": null,
  "updated_at": "2019-11-18 00:07:37 UTC"
}
...output omitted...
```

► 10. Delete the **Training** organization and respective life-cycle environments.

10.1. Modify the **curl** command to use the **DELETE** verb, and append the organization ID for **Training** to the end of the query.

```
[student@satellite ~]$ curl --header "Content-Type:application/json" \
--request DELETE --user admin:redhat \
https://satellite.lab.example.com/katello/api/organizations/17 \
| python -m json.tool
...output omitted...
{
  "action": "Destroy organization 'Training'",
  "cli_example": null,
  "ended_at": null,
  "humanized": {
    "action": "Destroy",
    "errors": [],
  ...output omitted...
  "organization": {
    "id": 17,
    "label": "Training",
    "name": "Training"
```

```
    },  
...output omitted...
```

- 11. Log out from the **satellite** SSH session and return to **workstation** as the **student** user.

## Finish

On the **workstation** machine, use the **lab** command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab api-overview finish
```

This concludes the guided exercise.

# Integrating Red Hat Satellite Functionality in Applications

## Objectives

After completing this section, you should be able to integrate common Satellite tasks and object queries into custom scripts and external applications, including popular languages such as curl, Ruby, and Python.

## Creating Custom API Scripts

The **curl** command is used with the Satellite API to perform various tasks from the command-line, as previously discussed. For example, the following command is a quick way to list organizations in Satellite.

```
[student@satellite ~]$ curl --request GET --insecure --user admin:redhat \
https://satellite.lab.example.com/api/v2/organizations \
| python -m json.tool | grep '"name":'
...output omitted...
    "name": "Default Organization",
    "name": "Finance",
    "name": "Operations",
```

Languages such as Ruby and Python are used as a front end to the Satellite API to automate routine tasks, such as those to create, modify, or delete objects in Satellite. However, if you are not familiar with the language syntax, then starting with a simple example will help pave the way for more complex scripts that automate administrative tasks.

## Making API Requests with Ruby

The following code is an example of a simple Ruby script that lists all organizations in Satellite. This code is a starting point to observe and learn fundamental syntax that when extended can automate more complex tasks, such as creating or modifying objects in Satellite.

```
#!/usr/bin/env ruby
require 'json'
require 'rest-client'

url = 'https://satellite.lab.example.com/api/v2/'
$username = 'admin'
$password = 'ADD THE PASSWORD HERE'

def get_json(location)
  response = RestClient::Request.new(
    :method => :get,
    :url => location,
    :user => $username,
    :password => $password,
    :headers => { :accept => :json,
    :content_type => :json }
```

```
    ).execute
    results = JSON.parse(response.to_str)
end

hosts = get_json(url+"hosts/")
#puts JSON.pretty_generate(hosts)

puts "Hosts within Satellite are:"
hosts['results'].each do |name|
    puts name['name']
end

exit()
```

The Ruby script returns the names of all organizations in Satellite.

```
[student@satellite ~]$ list-organizations.rb
Organizations within Satellite are:
Default Organization
Finance
Operations
```

## Creating Objects Using Ruby

The following code is an example of a more complex Ruby script that connects to the Red Hat Satellite 6 API and creates an organization, and then creates three environments in the organization. If the organization already exists, then the script uses that organization. If any of the environments already exist in the organization, then the script raises an error and quits.

```
#!/usr/bin/ruby

require 'rest-client'
require 'json'

# MODIFY BOTH URLs TO MATCH YOUR SATELLITE ENVIRONMENT
url = 'https://satellite.example.com/api/v2/'
katello_url = "#{url}/katello/api/v2/"

$username = 'admin'
$password = 'ADD THE PASSWORD HERE'

org_name = "ADD THE ORGANIZATION NAME HERE"
environments = [ "Development", "Testing", "Production" ]

# Performs a GET using the passed URL location
def get_json(location)
    response = RestClient::Request.new(
        :method => :get,
        :url => location,
        :user => $username,
        :password => $password,
        :headers => { :accept => :json,
        :content_type => :json }
```

```
.execute
JSON.parse(response.to_str)
end

# Performs a POST and passes the data to the URL location
def post_json(location, json_data)
  response = RestClient::Request.new(
    :method => :post,
    :url => location,
    :user => $username,
    :password => $password,
    :headers => { :accept => :json,
      :content_type => :json},
    :payload => json_data
  ).execute
  JSON.parse(response.to_str)
end

# Creates a hash with ids mapping to names for an array of records
def id_name_map(records)
  records.inject({}) do |map, record|
    map.update(record['id'] => record['name'])
  end
end

# Get list of existing organizations
orgs = get_json("#{katello_url}/organizations")
org_list = id_name_map(orgs['results'])

if !org_list.has_value?(org_name)
  # If our organization is not found, create it
  puts "Creating organization: \t#{org_name}"
  org_id = post_json("#{katello_url}/organizations", JSON.generate({"name"=>
  org_name}))["id"]
else
  # Our organization exists, so let's grab it
  org_id = org_list.key(org_name)
  puts "Organization \"#{org_name}\" exists"
end

# Get list of organization lifecycle environments
envs = get_json("#{katello_url}/organizations/#{org_id}/environments")
env_list = id_name_map(envs['results'])
prior_env_id = env_list.key("Library")

# Exit the script if at least one life cycle environment already exists
environments.each do |e|
  if env_list.has_value?(e)
    puts "ERROR: One of the Environments is not unique to organization"
    exit
  end
end

# Create life cycle environments
environments.each do |environment|
```

```
    puts "Creating environment: \t#{environment}"
    prior_env_id = post_json("#{katello_url}/organizations/#{org_id}/environments",
      JSON.generate({"name" => environment, "organization_id" => org_id, "prior_id" =>
      prior_env_id}))["id"]
  end
```

## Making API Requests with Python

The following Python script is a simple example that lists all organizations in Satellite. This script does not create or modify objects, but it is a good starting point to learn fundamental syntax used in scripts that automate tasks.

```
#!/usr/bin/python
import json
import requests
import sys
import urllib3
import logging
logging.captureWarnings(True)

# Define Satellite location and login details
SAT_API = "https://satellite.lab.example.com/katello/api/v2/"
USERNAME = "admin"
PASSWORD = "ADD THE PASSWORD HERE"
SSL_VERIFY = False

def get_json(location):
    """
    Performs a GET using the passed URL location
    """

    r = requests.get(location, auth=(USERNAME, PASSWORD), verify=SSL_VERIFY)

    return r.json()

def main():
    # List all organizations available to the user
    orgs = get_json(SAT_API + "organizations/")

    # Pretty Print the returned JSON of Organizations
    print json.dumps(orgs, sort_keys=True, indent=4)

if __name__ == "__main__":
    main()
```

The Python script returns the names, in JSON format, of all organizations in Satellite.

```
[student@satellite ~]$ list-organizations.py
{
...output omitted...
{
  "created_at": "2019-10-22 13:38:32 UTC",
```

```
        "description": null,
        "id": 1,
        "label": "Default_Organization",
        "name": "Default Organization",
        "title": "Default Organization",
        "updated_at": "2019-10-22 13:38:32 UTC"
    },
    {
        "created_at": "2019-10-29 22:54:43 UTC",
        "description": "Finance Department",
        "id": 5,
        "label": "Finance",
        "name": "Finance",
        "title": "Finance",
        "updated_at": "2019-10-29 22:58:35 UTC"
    },
    {
        "created_at": "2019-10-29 22:44:01 UTC",
        "description": "Operations Department",
        "id": 3,
        "label": "Operations",
        "name": "Operations",
        "title": "Operations",
        "updated_at": "2019-10-29 22:47:52 UTC"
    }
],
...output omitted...
}
```

## Creating Objects Using Python

The following code is an example of a more complex Python script that connects to the Red Hat Satellite 6 API and creates an organization, and then creates three environments in the organization. If the organization already exists, then the script uses that organization. If any of the environments already exist in the organization, then the script raises an error and quits.

```
#!/usr/bin/python

import json
import sys
import urllib3
import logging
logging.captureWarnings(True)

try:
    import requests
except ImportError:
    print ("Please install the python-requests module.")
    sys.exit(-1)

# URL to your Satellite 6 server
URL = "https://satellite.lab.example.com"
# URL for the API to your deployed Satellite 6 server
SAT_API = "%s/katello/api/v2/" % URL
```

```
# Katello-specific API
KATELLO_API = "%s/katello/api/" % URL
POST_HEADERS = {'content-type': 'application/json'}
# Default credentials to login to Satellite 6
USERNAME = "ADD USER NAME"
PASSWORD = "ADD PASSWORD"
# Ignore SSL for now
SSL_VERIFY = False

# Name of the organization to be either created or used
ORG_NAME = "ADD ORGANIZATION"
# Name for life cycle environments to be either created or used
ENVIRONMENTS = ["ADD ENV 1", "ADD ENV 2", "ADD ENV 3"]

def get_json(location):
    """
    Performs a GET using the passed URL location
    """
    r = requests.get(location, auth=(USERNAME, PASSWORD), verify=SSL_VERIFY)

    return r.json()

def post_json(location, json_data):
    """
    Performs a POST and passes the data to the URL location
    """

    result = requests.post(
        location,
        data=json_data,
        auth=(USERNAME, PASSWORD),
        verify=SSL_VERIFY,
        headers=POST_HEADERS)

    return result.json()

def main():
    """
    Main routine that creates or re-uses an organization and
    life cycle environments. If life cycle environments already
    exist, exit out.
    """

    # Check if our organization already exists
    org = get_json(SAT_API + "organizations/" + ORG_NAME)

    # If our organization is not found, create it
    if org.get('error', None):
        org_id = post_json(
            SAT_API + "organizations/",
            json.dumps({"name": ORG_NAME}))["id"]
```

```
print ("Creating organization: \t" + ORG_NAME)
else:
    # Our organization exists, so let's grab it
    org_id = org['id']
    print ("Organization '%s' exists." % ORG_NAME)

# Now, let's fetch all available life cycle environments for this org...
envs = get_json(
    SAT_API + "organizations/" + str(org_id) + "/environments/")

# ... and add them to a dictionary, with respective 'Prior' environment
prior_env_id = 0
env_list = {}
for env in envs['results']:
    env_list[env['id']] = env['name']
    prior_env_id = env['id'] if env['name'] == "Library" else prior_env_id

# Exit the script if at least one life cycle environment already exists
if all(environment in env_list.values() for environment in ENVIRONMENTS):
    print ("ERROR: One of the Environments is not unique to organization")
    sys.exit(-1)

# Create life cycle environments
for environment in ENVIRONMENTS:
    new_env_id = post_json(
        SAT_API + "organizations/" + str(org_id) + "/environments/",
        json.dumps(
            {
                "name": environment,
                "organization_id": org_id,
                "prior": prior_env_id
            })["id"])

    print ("Creating environment: \t" + environment)
    prior_env_id = new_env_id

if __name__ == "__main__":
    main()
```



## References

For more information, refer to the *API Requests in Different Languages* chapter in the *Red Hat Satellite 6.6 API Guide* at  
[https://access.redhat.com/documentation/en-us/red\\_hat\\_satellite/6.6/html-single/api\\_guide/index#chap-Red\\_Hat\\_Satellite-API\\_Guide-API\\_Requests\\_in\\_Different\\_Languages](https://access.redhat.com/documentation/en-us/red_hat_satellite/6.6/html-single/api_guide/index#chap-Red_Hat_Satellite-API_Guide-API_Requests_in_Different_Languages)

## ► Guided Exercise

# Integrating Red Hat Satellite Functionality in Applications

In this exercise, you will perform common tasks and object queries using multiple custom scripts.

### Outcomes

You should be able to create new scripts and modify existing ones to perform specific tasks, such as creating or updating objects in Satellite.

### Before You Begin

Log in as the **student** user on **workstation** using **student** as the password.

Run the **lab api-use start** command. This command determines if the host, **satellite**, is reachable on the network and prepares it for this exercise.

```
[student@workstation ~]$ lab api-use start
```

- 1. Prepare **satellite** to execute custom scripts from the **/home/student** directory.

- 1.1. On **workstation**, ssh to **satellite** as **student**.

```
[student@workstation ~]$ ssh student@satellite  
[student@satellite ~]$
```

- 1.2. Create the **/home/student/bin** directory.

```
[student@satellite ~]$ mkdir /home/student/bin
```

- 2. Modify an existing Python template script to create a new organization in Satellite named **Sales**, and three life-cycle environments named **Development**, **Testing**, and **Production**. The life-cycle environments must belong to the **Sales** organization.
- 2.1. Download the <https://materials.lab.example.com/create-objects-template.py> file to the **/home/student/bin** directory, and then rename the template file to **create-objects.py**.

```
[student@satellite ~]$ cd /home/student/bin  
[student@satellite bin]$ wget http://materials.example.com/create-objects-template.py  
[student@satellite bin]$ mv create-objects-template.py create-objects.py
```

Ensure that the script is executable.

```
[student@satellite bin]$ chmod +x /home/student/bin/create-objects.py
```

- 3. Modify the following variables in the **create-objects.py** file.

| Variable Name | Value                                                                                                |
|---------------|------------------------------------------------------------------------------------------------------|
| USERNAME      | admin                                                                                                |
| PASSWORD      | redhat                                                                                               |
| ORG_NAME      | Sales                                                                                                |
| ENVIRONMENTS  | <ul style="list-style-type: none"><li>• Development</li><li>• Testing</li><li>• Production</li></ul> |

Variable modifications in the Python file, **create-objects.py**, should match the following:

```
[student@satellite bin]$ cat create-objects.py
#!/usr/bin/python3

import json
import sys
import urllib3
urllib3.disable_warnings()

try:
    import requests
except ImportError:
    print ("Please install the python-requests module.")
    sys.exit(-1)

# URL to your Satellite 6 server
URL = "https://satellite.lab.example.com"
# URL for the API to your deployed Satellite 6 server
SAT_API = "%s/katello/api/v2/" % URL
# Katello-specific API
KATELLO_API = "%s/katello/api/" % URL
POST_HEADERS = {'content-type': 'application/json'}
# Default credentials to login to Satellite 6
USERNAME = "admin"
PASSWORD = "redhat"
# Ignore SSL for now
SSL_VERIFY = False

# Name of the organization to be either created or used
ORG_NAME = "Sales"
# Name for life cycle environments to be either created or used
ENVIRONMENTS = ["Development", "Testing", "Production"]

...output omitted...
```

- ▶ 4. Execute the **create-objects.py** Python script.

```
[student@satellite bin]$ create-objects.py
Creating organization: Sales
Creating environment: Development
Creating environment: Testing
Creating environment: Production
```

- ▶ 5. Use the **hammer** command to verify the life-cycle environment paths for the **Sales** organization.

```
[student@satellite bin]$ sudo hammer lifecycle-environment paths \
--organization Sales
[sudo] password for student: student
-----
LIFECYCLE PATH
-----
Library >> Development >> Testing >> Production
-----
```

- ▶ 6. Create an additional set of objects using a Ruby script instead of Python.

Modify an existing Ruby script to create a new organization in Satellite named **Research** and three life-cycle environments named **Development**, **Testing**, and **Production**. The life-cycle environments must belong to the **Research** organization.

- 6.1. Download the <https://materials.lab.example.com/create-objects-template.rb> file to the **/home/student/bin** directory, and then rename the template file to **create-objects.rb**.

```
[student@satellite bin]$ wget http://materials.example.com/create-objects-
template.rb
[student@satellite bin]$ mv create-objects-template.rb create-objects.rb
```

Ensure that the script is executable.

```
[student@satellite bin]$ chmod +x create-objects.rb
```

- ▶ 7. Modify the following variables in the **create-objects.rb** file.

| Variable Name | Value                                                                                                                         |
|---------------|-------------------------------------------------------------------------------------------------------------------------------|
| USERNAME      | <b>admin</b>                                                                                                                  |
| PASSWORD      | <b>redhat</b>                                                                                                                 |
| ORG_NAME      | <b>Research</b>                                                                                                               |
| ENVIRONMENTS  | <ul style="list-style-type: none"> <li>• <b>Development</b></li> <li>• <b>Testing</b></li> <li>• <b>Production</b></li> </ul> |

Variable modifications in the Ruby file, **create-objects.rb**, should match the following:

```
[student@satellite bin]$ cat create-objects.rb
#!/usr/bin/ruby

require 'rest-client'
require 'json'

url = 'https://satellite.lab.example.com/'
katello_url = "#{url}/katello/api/v2/"

$username = 'admin'
$password = 'redhat'

$org_name = "Research"
environments = [ "Development", "Testing", "Production" ]

...output omitted...
```

- 8. Execute the **create-objects.rb** Ruby script.

```
[student@satellite bin]$ ./create-objects.rb
Creating organization: Research
Creating environment: Development
Creating environment: Testing
Creating environment: Production
```

- 9. Use the **hammer** command to verify the life-cycle environment paths for the **Research** organization.

```
[student@satellite bin]$ sudo hammer lifecycle-environment paths \
--organization Research
[sudo] password for student: student
-----
LIFECYCLE PATH
-----
Library >> Development >> Testing >> Production
-----
```

- 10. Log out from the **satellite** SSH session and return to **workstation** as the **student** user.

## Finish

On the **workstation** machine, use the **lab** command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab api-use finish
```

This concludes the guided exercise.

# Using the Hammer CLI as an API Interface

---

## Objectives

After completing this section, you should be able to:

- Describe the Hammer CLI tool syntax.
- Compare Hammer and scripted API use cases.

## Describing the Hammer CLI Tool

The *Hammer* utility is provided with Satellite as a command-line interface to the Satellite API. You might find that using Hammer to perform common tasks is quicker than navigating through the Satellite web UI. Hammer also has options to generate parseable output in formats such as CSV, YAML, or JSON, making it suitable for scripting.

The Satellite web UI has a higher priority during development, so new features might not be supported immediately in Hammer.

## Comparing Hammer with Scripted API Use

There are reasons for and against writing custom scripts to interact with the Satellite API. The following table compares features of custom scripts with the features of Hammer.

| Hammer                                            | Custom Script                                         |
|---------------------------------------------------|-------------------------------------------------------|
| Performs a single task when executed              | Might perform several tasks when executed             |
| Authenticates every time                          | Might store an authentication token for future use    |
| Tracks changes and additions to the Satellite API | Requires maintenance to support Satellite API updates |
| Provides several output formats by default        | Support for each output format must be added          |

The Hammer utility has a **--debug** option that provides help when developing custom scripts. Enabling this option displays the API requests and responses made, and the headers used by the command.

The following example shows the API response when Hammer is listing organizations:

```
[root@satellite ~]# hammer --debug organization list
...output omitted...
[DEBUG 2019-11-19T03:44:33 SSLOptions] SSL options: {
    :ssl_ca_file => "/etc/pki/katello/certs/katello-server-ca.crt",
    :verify_ssl => true
}
[DEBUG 2019-11-19T03:44:33 API] Global headers: {
    :content_type => "application/json",
```

```
:accept => "application/json;version=2",
"Accept-Language" => "en"
}
...output omitted...
[DEBUG 2019-11-19T03:44:34 API] Using authenticator:
HammerCLIForeman::Api::InteractiveBasicAuth
[DEBUG 2019-11-19T03:44:35 API] Response: {
    "total" => 3,
    "subtotal" => 3,
    "page" => 1,
    "per_page" => 1000,
    "search" => nil,
    "sort" => {
        "by" => nil,
        "order" => nil
    },
    "results" => [
        [0] {
            "label" => "Default_Organization",
            "created_at" => "2019-10-22 13:38:32 UTC",
            "updated_at" => "2019-10-22 13:38:32 UTC",
            "id" => 1,
            "name" => "Default Organization",
            "title" => "Default Organization",
            "description" => nil
        },
        [1] {
            "label" => "Finance",
            "created_at" => "2019-11-06 08:42:16 UTC",
            "updated_at" => "2019-11-06 08:46:54 UTC",
            "id" => 5,
            "name" => "Finance",
            "title" => "Finance",
            "description" => "Finance Department"
        },
        [2] {
            "label" => "Operations",
            "created_at" => "2019-11-06 08:21:03 UTC",
            "updated_at" => "2019-11-06 08:27:54 UTC",
            "id" => 3,
            "name" => "Operations",
            "title" => "Operations",
            "description" => "Operations Department"
        }
    ]
}
[DEBUG 2019-11-19T03:44:35 API] Response headers: {
    :date => "Tue, 19 Nov 2019 03:44:34 GMT",
    :server => "Apache",
    :foreman_version => "1.22.0.10",
    :foreman_api_version => "2",
    :apipie_checksum =>
"b7b03d397d043236a88a4f46f5b1d18d401a9e3d",
    :cache_control => "max-age=0, private, must-revalidate",
    :x_request_id => "9689b918-484f-4f4b-ac8e-6e3d11f97f02",
}
```

```
        :x_runtime => "0.061372",
:strict_transport_security => "max-age=631139040; includeSubdomains",
        :x_frame_options => "sameorigin",
        :x_content_type_options => "nosniff",
        :x_xss_protection => "1; mode=block",
        :x_download_options => "noopen",
:x_permitted_cross_domain_policies => "none",
        :content_security_policy => "default-src 'self'; child-src 'self';
connect-src 'self' ws: wss: img-src 'self' data: *.gravatar.com; script-src
'unsafe-eval' 'unsafe-inline' 'self'; style-src 'unsafe-inline' 'self''",
        :x_powered_by => "Phusion Passenger 4.0.18",
        :set_cookie => [
[0] "_session_id=b0adbe438d23d85f4a754769c9e7f650; path=/; secure;
HttpOnly; SameSite=Lax"
],
        :etag => "W/\"8ac3205d0eef9280e94726e742d538e4-
gzip\",
        :status => "200 OK",
        :vary => "Accept-Encoding",
:content_encoding => "gzip",
:content_length => "291",
        :content_type => "application/json; charset=utf-8"
}
...output omitted...
```

From the output you can see that the response contains an array (list) of hashes (key-value pairs), which might help you determine how to process it.



## References

For more information, refer to the *Hammer CLI Guide* at  
[https://access.redhat.com/documentation/en-us/red\\_hat\\_satellite/6.6/html-single/hammer\\_cli\\_guide/index](https://access.redhat.com/documentation/en-us/red_hat_satellite/6.6/html-single/hammer_cli_guide/index)

For more information, refer to the *Hammer Cheat Sheet* at  
[https://access.redhat.com/documentation/en-us/red\\_hat\\_satellite/6.6/html-single/hammer\\_cheat\\_sheet/index](https://access.redhat.com/documentation/en-us/red_hat_satellite/6.6/html-single/hammer_cheat_sheet/index)

## ► Guided Exercise

# Using the Hammer CLI as an API Interface

In this exercise, you will perform common tasks and object queries using the Hammer CLI.

## Outcomes

You should be able to:

- Check hammer authentication status.
- Create organizations, users, user groups, and host collections using hammer.

## Before You Begin

Log in as the **student** user on **workstation** using **student** as the password.

Run the **lab api-hammer start** command. The command verifies that **satellite** is available.

```
[student@workstation ~]$ lab api-hammer start
```

- 1. On **satellite**, use **hammer** to create the following objects:

| Object Type     | Name                                                |
|-----------------|-----------------------------------------------------|
| Organization    | <b>SecOps</b>                                       |
| Host Collection | <b>Firewalls</b><br><b>IDS</b><br><b>LogServers</b> |
| User            | <b>SecOpsAdmin</b>                                  |
| User group      | <b>SecOperators</b>                                 |

- 1.1. On **workstation**, ssh to **satellite** as **student**. Use **sudo -i** to switch to **root**.

```
[student@workstation ~]$ ssh satellite
[student@satellite ~]$ sudo -i
[sudo] password for student: student
[root@satellite ~]#
```

- 1.2. View the **/root/.hammer/cli.modules.d/foreman.yml** configuration file for hammer. This file was created by the **satellite-installer** command during the Satellite installation. Note that the **admin** user credentials for Satellite are stored in this file.

```
[root@satellite ~]# cat ~/.hammer/cli.modules.d/foreman.yml
:foreman:
  :username: 'admin'
  :password: 'redhat'
  :refresh_cache: false
  :request_timeout: 120
```

- 1.3. Check the authentication status for hammer.

```
[root@satellite ~]# hammer auth status
Using configured credentials for user 'admin'.
```

- 1.4. Create the SecOps organization.

```
[root@satellite ~]# hammer organization create \
--name SecOps \
--description 'Security Operations Organization'
Organization created.
```

- 1.5. Create the Firewalls, IDS, and LogServers host collections.

```
[root@satellite ~]# hammer host-collection create \
--name Firewalls \
--organization SecOps
Host collection created.
[root@satellite ~]# hammer host-collection create \
--name IDS \
--organization SecOps
Host collection created.
[root@satellite ~]# hammer host-collection create \
--name LogServers \
--organization SecOps
Host collection created.
```

- 1.6. Create the SecOpsAdmin user. The **--auth-source-id 1** option specifies that this user will be authenticated from the internal database.

```
[root@satellite ~]# hammer user create \
--login SecOpsAdmin \
--password redhat \
--mail secalert@example.com \
--auth-source-id 1 \
--organization SecOps
User [SecOpsAdmin] created.
```

- 1.7. Assign the SecOpsAdmin user the Organization admin role for the SecOps organization.

```
[root@satellite ~]# hammer user add-role \
--login SecOpsAdmin \
--role 'Organization admin'
User role has been assigned.
```

- 1.8. Create the SecOperators user group, and then log out of the **satellite** host.

```
[root@satellite ~]# hammer user-group create \
--name SecOperators \
--organization SecOps
User group [SecOperators] created.
[root@satellite ~]# logout
[student@satellite ~]$ logout
[student@workstation ~]$
```

- 2. Log in to the Satellite web UI as **SecOpsAdmin** using **redhat** as the password. Verify that the objects created using hammer are present.

- 2.1. Use your browser to navigate to <https://satellite.lab.example.com>.

- 2.2. Log in as **SecOpsAdmin** using **redhat** as the password.

Note the organization at the upper-left corner has defaulted to **SecOps**. Select **Any Location** and note that there are no locations created for this organization.

- 2.3. Click **SecOps** → **Manage Organizations** and note that the SecOpsAdmin user has no permissions to view or manage other organizations.

- 2.4. Navigate to **Hosts** → **Host Collections** and then verify that the collections created with hammer are present.

- 2.5. Navigate to **Administer** → **User Groups** and then verify that the SecOperators group is present.

Log out of the Satellite web UI when finished.

## Finish

On the **workstation** machine, use the **lab** command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab api-hammer finish
```

This concludes the guided exercise.

## ► Lab

# Managing Red Hat Satellite Using the API

### Performance Checklist

In this lab, you will perform common tasks and object queries using the curl command, custom application scripts, and the Hammer CLI.

### Outcomes

You should be able to:

- Create Satellite objects with **curl** REST API commands.
- Create Satellite objects with a Ruby application script.
- Query, create, or modify Satellite objects with the Hammer CLI.

### Before You Begin

Log in as the **student** user on **workstation** using **student** as the password.

Run the **lab api-review start** command. The command verifies that the **satellite** host is available.

The **lab api-review start** command sets all steps in this lab to a *clean* start position.

```
[student@workstation ~]$ lab api-review start
```

1. Use the **curl** command with the Red Hat Satellite REST API to create a new **Global Sales** organization with **Global product sales** as its description.
2. Modify an existing Ruby script to add three life-cycle environments named **Development**, **Testing**, and **Production** to the **Global Sales** organization. Use the **admin** user with **redhat** as password.  
Download and run the available at <http://materials.example.com/create-objects-template.rb> Ruby script.
3. Use the Hammer CLI to update the description for each of the life-cycle environments in the **Global Sales** organization.

| Environment Name | Description            |
|------------------|------------------------|
| Development      | Sales dev environment  |
| Testing          | Sales test environment |
| Production       | Sales prod environment |

4. Use the Hammer CLI to create the following objects in the **Global Sales** organization.

| Object Type     | Name               |
|-----------------|--------------------|
| Host Collection | GlobalSalesServers |
| User            | SalesAdmin         |
| User role       | Organization admin |
| User group      | SalesStaff         |

5. Log out from the **satellite** SSH session and return to **workstation** as the **student** user.

## Evaluation

As the **student** user on the **workstation** machine, use the **lab** command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab api-review grade
```

## Finish

As the **student** user on the **workstation** machine, use the **lab** command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab api-review finish
```

This concludes the lab.

## ► Solution

# Managing Red Hat Satellite Using the API

### Performance Checklist

In this lab, you will perform common tasks and object queries using the curl command, custom application scripts, and the Hammer CLI.

### Outcomes

You should be able to:

- Create Satellite objects with **curl** REST API commands.
- Create Satellite objects with a Ruby application script.
- Query, create, or modify Satellite objects with the Hammer CLI.

### Before You Begin

Log in as the **student** user on **workstation** using **student** as the password.

Run the **lab api-review start** command. The command verifies that the **satellite** host is available.

The **lab api-review start** command sets all steps in this lab to a *clean* start position.

```
[student@workstation ~]$ lab api-review start
```

1. Use the **curl** command with the Red Hat Satellite REST API to create a new **Global Sales** organization with **Global product sales** as its description.

- 1.1. On **workstation**, ssh to **satellite** as **student**.

```
[student@workstation ~]$ ssh student@satellite  
[student@satellite ~]$
```

- 1.2. Create a **/home/student/create-globalsales-organization.json** data file with organization parameters.

```
[student@satellite ~]$ vi /home/student/create-globalsales-organization.json  
{"name": "Global Sales", "description": "Global product sales"}
```

- 1.3. Create the **Global Sales** organization.

```
[student@satellite ~]$ curl --header "Content-Type:application/json" \
--request POST --user admin:redhat \
--data @create-globalsales-organization.json \
https://satellite.lab.example.com/katello/api/organizations \
| python -m json.tool
...output omitted...
  "created_at": "2019-11-20 00:39:44 UTC",
  "default_content_view_id": 14,
  "description": "Global product sales",
  "domains": [],
  "environments": [],
  "hostgroups": [],
  "hosts_count": 0,
  "id": 10,
  "label": "Global_Sales",
  "library_id": 18,
  "media": [],
  "name": "Global Sales",
...output omitted...
```

2. Modify an existing Ruby script to add three life-cycle environments named **Development**, **Testing**, and **Production** to the **Global Sales** organization. Use the **admin** user with **redhat** as password.

Download and run the available at <http://materials.example.com/create-objects-template.rb> Ruby script.

- 2.1. Create the **/home/student/bin** directory.

```
[student@satellite ~]$ mkdir /home/student/bin
```

- 2.2. Download the <https://materials.lab.example.com/create-objects-template.rb> file to the **/home/student/bin** directory and rename the template file to **create-lifecycle-envs.rb**.

```
[student@satellite ~]$ cd /home/student/bin
[student@satellite bin]$ wget http://materials.example.com/create-objects-
template.rb
[student@satellite bin]$ mv create-objects-template.rb create-lifecycle-envs.rb
```

Ensure that the script is executable.

```
[student@satellite bin]$ chmod +x create-lifecycle-envs.rb
```

- 2.3. Modify the following variables in the **create-lifecycle-envs.rb** file.

| Variable Name | Value  |
|---------------|--------|
| username      | admin  |
| password      | redhat |

| Variable Name | Value                                                                                                    |
|---------------|----------------------------------------------------------------------------------------------------------|
| org_name      | Global Sales                                                                                             |
| environments  | <ul style="list-style-type: none"> <li>• Development</li> <li>• Testing</li> <li>• Production</li> </ul> |

Variable modifications in the Ruby file, **create-lifecycle-envs.rb**, should match the following:

```
[student@satellite bin]$ cat create-lifecycle-envs.rb
#!/usr/bin/ruby

require 'rest-client'
require 'json'

url = 'https://satellite.lab.example.com/'
katello_url = "#{url}/katello/api/v2/"

$username = 'admin'
$password = 'redhat'

$org_name = "Global Sales"
environments = [ "Development", "Testing", "Production" ]

...output omitted...
```

#### 2.4. Execute the **create-lifecycle-envs.rb** Ruby script.

```
[student@satellite bin]$ ./create-lifecycle-envs.rb
Organization "Global Sales" exists
Creating environment: Development
Creating environment: Testing
Creating environment: Production
```

3. Use the Hammer CLI to update the description for each of the life-cycle environments in the **Global Sales** organization.

| Environment Name | Description            |
|------------------|------------------------|
| Development      | Sales dev environment  |
| Testing          | Sales test environment |
| Production       | Sales prod environment |

#### 3.1. Use **sudo -i** to switch to **root**.

```
[student@satellite ~]$ sudo -i
[sudo] password for student: student
[root@satellite ~]#
```

3.2. Check the authentication status for hammer.

```
[root@satellite ~]# hammer auth status  
Using configured credentials for user 'admin'.
```

3.3. Update each of the life-cycle environment descriptions.

```
[root@satellite ~]# hammer lifecycle-environment update \  
--name Development \  
--organization "Global Sales" \  
--description "Sales dev environment"  
Environment updated.
```

```
[root@satellite ~]# hammer lifecycle-environment update \  
--name Testing \  
--organization "Global Sales" \  
--description "Sales test environment"  
Environment updated.
```

```
[root@satellite ~]# hammer lifecycle-environment update \  
--name Production \  
--organization "Global Sales" \  
--description "Sales prod environment"  
Environment updated.
```

4. Use the Hammer CLI to create the following objects in the **Global Sales** organization.

| Object Type     | Name               |
|-----------------|--------------------|
| Host Collection | GlobalSalesServers |
| User            | SalesAdmin         |
| User role       | Organization admin |
| User group      | SalesStaff         |

4.1. Create **GlobalSalesServers** host collection.

```
[root@satellite ~]# hammer host-collection create \  
--name GlobalSalesServers \  
--organization "Global Sales"  
Host collection created.
```

4.2. Create the **SalesAdmin** user.

```
[root@satellite ~]# hammer user create \
--login SalesAdmin \
--password redhat \
--mail salesadm@example.com \
--auth-source-id 1 \
--organization "Global Sales"
User [SalesAdmin] created.
```

Note that the **--auth-source-id 1** option specifies that this user will be authenticated internally rather than externally.

- 4.3. Assign the **SalesAdmin** user the **Organization admin** role for the **Global Sales** organization.

```
[root@satellite ~]# hammer user add-role \
--login SalesAdmin \
--role 'Organization admin'
User role has been assigned.
```

- 4.4. Create the **SalesStaff** user group.

```
[root@satellite ~]# hammer user-group create \
--name SalesStaff \
--organization "Global Sales"
User group [SalesStaff] created.
```

5. Log out from the **satellite** SSH session and return to **workstation** as the **student** user.

This concludes the lab.

## Evaluation

As the **student** user on the **workstation** machine, use the **lab** command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab api-review grade
```

## Finish

As the **student** user on the **workstation** machine, use the **lab** command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab api-review finish
```

This concludes the lab.

# Summary

---

In this chapter, you learned:

- Red Hat Satellite includes a REpresentational State Transfer (REST) API to control your Satellite environment without using the Satellite web UI.
- You can query the Satellite REST API from the command line using the **curl** command, or by using programming languages such as Ruby and Python.
- The **Hammer** utility provides a command-line interface to the Satellite REST API.

## Chapter 10

# Deploying Red Hat Satellite to a Cloud Platform

### Goal

Plan a Red Hat Satellite deployment on a cloud platform, including managed content hosts.

### Objectives

- Prepare for installing Red Hat Satellite Server and Capsules on selected cloud platforms.
- Manage content host cloud instances using Red Hat Satellite to interact with the cloud platform.

### Sections

- Running Red Hat Satellite Server on a Cloud Platform (and Quiz)
- Managing Content Hosts on a Cloud Platform (and Quiz)

# Running Red Hat Satellite Server on a Cloud Platform

## Objectives

After completing this section, you should be able to prepare for installing Red Hat Satellite Server and Capsules on selected cloud platforms.

## Deploying Satellite Server on the Cloud

This course uses Red Hat Satellite Server as an on-premise software delivery and management platform. Local content hosts are managed by Satellite Server, and content hosts in additional geographic locations are managed by remote Capsule Servers. An expanded architecture is required to use cloud platforms as additional compute resources while remaining cost efficient.

Using an on-premise Satellite Server to directly manage content hosts running on cloud platforms can incur high bandwidth costs, which are typically charged at the cloud vendor's external bandwidth rate. One option is to install Satellite Server directly on the cloud platform to manage instances in a single cloud region, as shown in the following diagram.

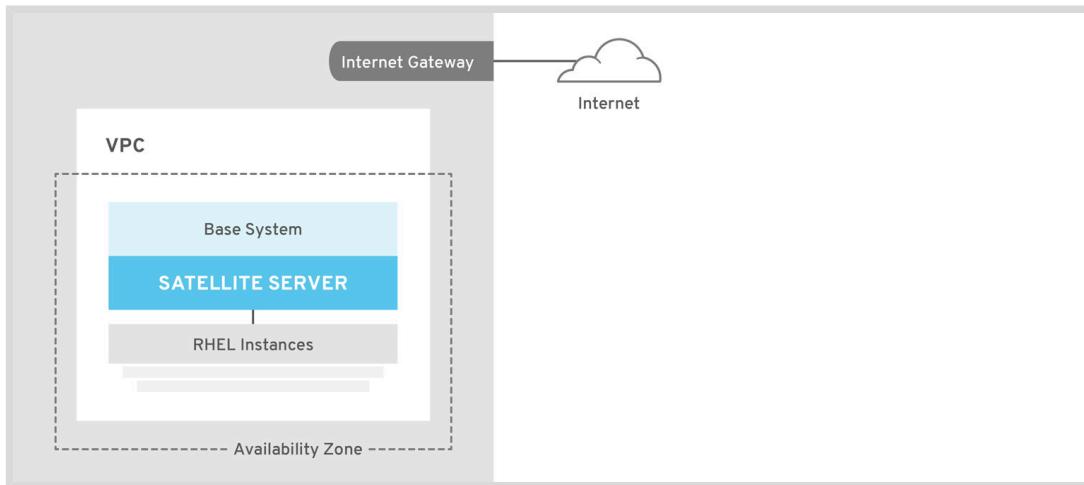
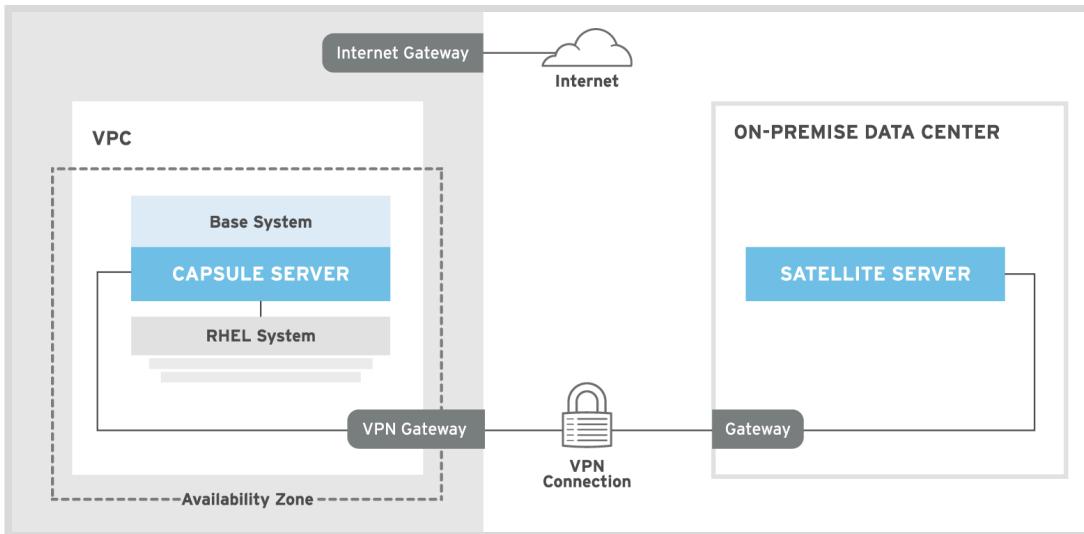


Figure 10.1: Satellite Server deployed in a single region

Another option is to deploy a Capsule Server in the cloud region, managed and synchronized by an on-premise Satellite Server. Additional Capsule Servers can be configured to manage multiple cloud regions or even multiple cloud providers using sufficient VPN or internet gateways. Similarly, both the Satellite Server and one or more Capsule Servers can be installed in individual Virtual Private Cloud (VPC) regions with VPN gateways between them for synchronizing content. This option is shown in the following diagram.



**Figure 10.2: Cloud-based Capsule Server securely connected to an on-premise Satellite Server**

## Comparing Red Hat Cloud Access and On-demand Images

Before choosing a Satellite architecture, determine how you intend to obtain Red Hat product images and support. There is an important support difference between providing your own images through the "bring your own subscription" model known as Red Hat Cloud Access, and using on-demand images obtained directly from supported cloud providers in a pay-as-you-go model.

### On-demand Images

On-demand images are purchased from cloud service providers, such as Amazon Web Services (AWS), Microsoft Azure, Google Cloud Platform, or IBM Cloud. These images can only be used within the cloud environment and are for cloud instances. Red Hat supplies updates to these images to the provider, who makes them available to their customers via Red Hat Update Infrastructure. On-demand customers should not register these images with Red Hat Subscription Management or with Satellite Server.

### Red Hat Cloud Access

If you want to host images in the cloud using your Red Hat subscriptions, purchase the necessary subscriptions directly from Red Hat and enable them for Red Hat Cloud Access. Enabling Red Hat Cloud Access for your eligible Red Hat product subscriptions allows you to use those products on supported public cloud providers. In this model, the terms of your subscription with Red Hat remain the same. You work directly with Red Hat. With Cloud Access, you can:

- Upload images of Red Hat products, to which you are subscribed, to a cloud provider.
- Access your images through the applicable web console.
- Maintain your support services with Red Hat directly.
- Move subscriptions to certified public cloud providers.
- Access all errata and updates to help improve security and quality.
- Maintain the consistency and help improve the security of your applications.

Red Hat Cloud Access is specifically for virtualized environments. For bare metal services, Cloud Access is not required. For example, clients can bring their Red Hat subscriptions using certified hardware on IBM Cloud Bare Metal Servers. For more information on hardware certified by Red Hat, refer to the hardware certification catalog for Red Hat Enterprise Linux.

## Use-case Scenarios on Cloud Providers

Because cloud providers are image-only services, most but not all Satellite use cases are available on cloud platforms. You can perform these Satellite activities on supported cloud providers:

- Managing subscriptions and errata
- Managing host content
- Managing host configuration
- Using Red Hat Insights with Satellite
- Using IdM for realm integration and external authentication
- Managing security compliance with OpenSCAP
- Running remote job execution on hosts

### Unusable Services in the Cloud

Cloud providers typically do not offer client-manageable DHCP services, making all PXE, iPXE, and kickstart provisioning methods unusable or unavailable, including:

- PXE provisioning
- Discovery and discovery rules
- ISO provisioning using iPXE
- PXE-less discovery (iPXE)

## Supported Cloud Providers for Running Satellite

Red Hat supports running Red Hat Satellite Server 6.5 or later Satellite or Capsule instances on the following supported cloud providers. Red Hat will not assist with configuring cloud provider networking, availability zones, or other cloud provider resources needed to deploy Satellite or Capsule Servers on cloud provider platforms, but that support is available through the cloud provider. Customers using Cloud Access no longer require a Red Hat support exception when installing on the following cloud providers:

- Alibaba Cloud
- Amazon Web Services
- Google Cloud Platform
- IBM Cloud
- Microsoft Azure

## Installing Satellite on Amazon Web Services

Using the AWS web management interface, request a new instance that meets the following requirements, and deploy a Red Hat Enterprise Linux virtual machine to the instance.

- The latest version of Red Hat Enterprise Linux 7 Server with a current subscription.
- 4-core 2.0 GHz CPU at a minimum.
- 20 GB memory at a minimum with 4 GB of swap space minimum. Satisfactory performance may require more CPU and memory resources.
- Forward and reverse DNS host name resolution using a fully qualified domain name.
- 600 GB of storage, preferably on solid state drives (SSD).
- AWS Storage Optimized instance.
- Elastic Block Store (EBS) volume for storing synchronized content other than the boot volume.
- Separate EBS volumes for storing other data, such as the MongoDB directory.

If you want a Satellite Server and Capsule Server to communicate using external DNS host names, open the required ports for communication in the AWS security group that is associated with each instance. Connect to the newly created EC2 instance when it becomes available. Install

Satellite Server or Capsule Server, as required, using the same procedures used for an on-premise installation.

When configuring Satellite on a cloud provider with integrated DNS and DHCP services, configure Satellite to disable management of these services by Puppet. When you install and configure Satellite for the first time, use **satellite-installer** with the **--foreman-proxy-dns-managed=false** and **--foreman-proxy-dhcp-managed=false** options. If these options are not specified during the initial installer run, rerunning the installer overwrites all manual changes.

For the complete steps and recommendations for installing Satellite Server, see the *Installing Satellite Server from a Connected Network* product documentation.



## References

### **Red Hat Enterprise Linux on Amazon EC2 - FAQs**

<https://aws.amazon.com/partners/redhat/faqs/>

### **What is the difference between Red Hat Cloud Access and Red Hat Enterprise Linux on-demand subscriptions in a public cloud?**

<https://access.redhat.com/articles/2041283>

### **Deploying Red Hat Satellite on IBM Cloud**

<https://access.redhat.com/articles/4277261>

### **Red Hat Cloud Access Frequently Asked Questions**

<https://access.redhat.com/articles/3664231>



## References

For more information, refer to the *Installing Satellite Server from a Connected Network* guide at

[https://access.redhat.com/documentation/en-us/red\\_hat\\_satellite/6.6/html-single/installing\\_satellite\\_server\\_from\\_a\\_connected\\_network/index](https://access.redhat.com/documentation/en-us/red_hat_satellite/6.6/html-single/installing_satellite_server_from_a_connected_network/index)

For more information, refer to the *Installing Capsule Server* guide at

[https://access.redhat.com/documentation/en-us/red\\_hat\\_satellite/6.6/html-single/installing\\_capsule\\_server/index](https://access.redhat.com/documentation/en-us/red_hat_satellite/6.6/html-single/installing_capsule_server/index)

For more information, refer to the *Planning for Red Hat Satellite 6* guide at

[https://access.redhat.com/documentation/en-us/red\\_hat\\_satellite/6.6/html-single/planning\\_for\\_red\\_hat\\_satellite\\_6/index](https://access.redhat.com/documentation/en-us/red_hat_satellite/6.6/html-single/planning_for_red_hat_satellite_6/index)

## ► Quiz

# Running Red Hat Satellite Server on a Cloud Platform

Choose the correct answers to the following questions:

- ▶ **1. Which four Satellite use cases will work on supported cloud providers? (Choose four.)**
  - a. Managing host configuration
  - b. Discovery and discovery rules
  - c. Managing subscriptions and errata
  - d. Running remote job execution on hosts
  - e. PXE-less discovery (iPXE)
  - f. Managing security compliance with OpenSCAP
  
- ▶ **2. Which six of the following Red Hat Cloud Access features are allowed for enabled products? (Choose six.)**
  - a. Access all errata and updates to ensure security and quality.
  - b. Access your images through the applicable web console.
  - c. Enable subscriptions for bare metal services on a certified public cloud provider.
  - d. Maintain the consistency and security of your applications.
  - e. Directly maintain your support services with Red Hat.
  - f. Move subscriptions to certified public cloud providers.
  - g. Upload images of Red Hat products, to which you are subscribed, to a cloud provider.

## ► Solution

# Running Red Hat Satellite Server on a Cloud Platform

Choose the correct answers to the following questions:

- ▶ **1. Which four Satellite use cases will work on supported cloud providers? (Choose four.)**
  - a. Managing host configuration
  - b. Discovery and discovery rules
  - c. Managing subscriptions and errata
  - d. Running remote job execution on hosts
  - e. PXE-less discovery (iPXE)
  - f. Managing security compliance with OpenSCAP
  
- ▶ **2. Which six of the following Red Hat Cloud Access features are allowed for enabled products? (Choose six.)**
  - a. Access all errata and updates to ensure security and quality.
  - b. Access your images through the applicable web console.
  - c. Enable subscriptions for bare metal services on a certified public cloud provider.
  - d. Maintain the consistency and security of your applications.
  - e. Directly maintain your support services with Red Hat.
  - f. Move subscriptions to certified public cloud providers.
  - g. Upload images of Red Hat products, to which you are subscribed, to a cloud provider.

# Managing Content Hosts on a Cloud Platform

---

## Objectives

After completing this section, you should be able to manage content host cloud instances using Red Hat Satellite to interact with the cloud platform.

Red Hat Satellite Server can deploy and manage content hosts on both private and public cloud platforms, in addition to the bare-metal and enterprise virtualization provisioning discussed in an earlier chapter. Similar to how Satellite interacts with enterprise virtualization platforms, Satellite connects to a cloud platform API to create instances that can then be managed. Because cloud platform deployments are image-based, Satellite does not support performing standard software package-based installations on cloud platforms. Satellite can manage host content and configuration after an instance is deployed in the cloud and becomes accessible.

Provisioning on a cloud platform occurs by requesting a new instance using the cloud provider's API and cloud-based resources. Configuring Satellite to use a cloud provider is a series of tasks that is almost identical for all providers:

- Define how to connect to the cloud provider API and compute region using a valid account and security keys.
- Upload supported images to the cloud provider region, and define the image details.
- Define the necessary virtualized hardware settings for the image in a compute profile.
- Create a new host from the Satellite web UI with similar information as for provisioning on bare metal, but instead selecting a cloud provider compute resource connection to cause the host to build in the cloud.

## Provisioning Cloud Instances in Amazon EC2

Amazon Elastic Compute Cloud (EC2) is a web service providing a public cloud platform. Satellite connects to the EC2 public API to create cloud instances and control their power management states. To prepare your on-premise Satellite Server to manage EC2 instances, install a Capsule Server in each Amazon region where you will request compute resources. Connect to each Capsule Server using a Virtual Private Cloud (VPC) to secure your content and communications.

Synchronize the appropriate content repositories for Red Hat Enterprise Linux and custom products to the Capsule Servers. Use this content for post-provisioning instance configuration and application installation. For image-based provisioning, upload your Cloud Access-enabled RHEL images as Amazon Machine Images (AMIs). Create activation keys for host registrations, configuring each with subscriptions, a life-cycle environment, content view, repositories, and host collection membership to support the content host's applications.

## Creating a Connection for the Amazon EC2 Compute Resource

Before configuring the EC2 connection, ensure that your Satellite Server's system time is correctly synchronized, using **ntpd** or **chrony** configured with authoritative public NTP peers. Having an inaccurate clock causes authentication protocols to suspect that communications are being tampered with, resulting in an inability to access Amazon Web Services.

To create an Amazon EC2 connection, navigate to **Infrastructure → Compute Resources**, and click **Create Compute Resource**. When naming EC2 compute resources, use a convention that includes the AWS region, and the provisioning context if you require separate connections for different Satellite locations and organizations. From the **Provider** list, select EC2. The remaining page refreshes to display fields for EC2-specific parameters.

Use the **Description** field to enter information to recognize the resource later. From the **HTTP proxy** list, select a configured HTTP smart proxy to connect to external API services. Currently, Amazon Web Services is the only cloud provider properly implementing Pulp content access through a connection-configured HTTP proxy. Satellite also supports using global HTTP proxies for all HTTP communications between Satellite and any cloud provider.

Amazon EC2 accounts are secured using access keys created during account setup. Enter your access key ID and secret key for the connection to use for all communications. With your keys, Satellite can connect to Amazon to obtain a list of available regions. Click **Load Regions** to populate the **Regions** list, and select the EC2 region or data center for this connection.

Navigate to the **Locations** and **Organizations** tabs to assign provisioning contexts to this EC2 connection. Deployments using this connection will be limited to those locations and organizations. Finally, save the Amazon EC2 connection. You can use the **hammer** command to perform the same task. Specify the access key ID and secret key with the **--user** and **--password** options:

```
[root@satellite ~]# hammer compute-resource create --name "EC2_region_context" \
--provider "EC2" --description "EC2 US-East-1 for Finance" \
--user "my_ami_id" --password "secret_key" --region "us-east-1" \
--locations "Boston" --organizations "Finance"
```

## Identifying Amazon EC2 Images to Satellite Server

Amazon EC2 uses image-based provisioning to create hosts. Load your Cloud Access-enabled images to the desired Amazon regions. Alternatively, Red Hat provides Gold Amazon Machine Images (AMI) to Amazon for use by Red Hat Cloud Access clients. The AMIs that are shared with your enrolled AWS account will be for Red Hat Enterprise Linux, Red Hat Enterprise Linux Atomic Host, Red Hat Gluster Storage, or Red Hat JBoss Enterprise Application Platform depending on the subscription you specified during Cloud Access enrollment. For all other Red Hat products, start with a base RHEL AMI and follow the product's recommended installation guidelines.

To add image details for your uploaded or existing Amazon images to the Satellite Server, including access details and image location, navigate to **Infrastructure → Compute Resources** and select an Amazon EC2 connection. Use the **Images** tab to attach a new image definition to this connection. Supply a name to identify the image for future use.

Select the operating system and architecture that corresponds with the image. Enter the username and password for an account already configured on that image that will allow access for post-configuration and ongoing management. Identify the image by entering its AMI ID in the format **ami-xxxxxxxx**.

Deployed instances are configured using either kickstart finish scripts or user-data input such as **cloud-init** data. Enabling one of these choices disables the other. For user-data input, the image must be built with **cloud-init**, or similar configuration tooling, properly configured and able to reach the Satellite or Capsule Server from the Amazon region where the image is deployed. Similarly, you must create and test a kickstart finish script if that choice is enabled. Using a finish script requires the use of a remote execution SSH key.

Finally, configure the image definition with the Amazon security role that will be used when creating an instance using this image. Click **Submit** to save the image details. You can use the **hammer** command to perform the same task:

```
[root@satellite ~]# hammer compute-resource image create --name "RHEL7 EC2 AMI" \
--operatingsystem "RHEL 7.7" --architecture "x86_64" --username root \
--user-data true --uuid "my_ami_id" --compute-resource "EC2_region_context"
```

## Creating a Compute Profile with EC2 Parameters

As discussed in a previous chapter, a compute profile specifies the hardware settings to use when provisioning content hosts. For the Amazon cloud provider, a compute profile is also configured with EC2 resource parameters to be used when this EC2 connection is selected.

To create a Compute Profile, navigate to **Infrastructure** → **Compute Profiles**, click the name of your profile, and then click the EC2 connection to configure it with an image. From the lists, select the EC2 flavor, availability zone with target cluster, and subnet to use when requesting an instance deployment. Select an image from the image definitions created earlier.

From the **Security Groups** list, select the cloud-based access rules for ports and IP addresses to control the host's network access. From the **Managed IP** list, select either a **Public IP** or a **Private IP**, depending on whether this host needs to be accessed by external users, or only by other region-based hosts. Click **Submit** to save the EC2 compute profile. This task cannot currently be accomplished using **hammer**, because the compute profile CLI commands are not yet implemented in Red Hat Satellite 6.6. As an alternative, you can include the same settings directly during the host creation process.

## Deploying a Host on Amazon EC2

The Amazon EC2 provisioning process creates hosts from existing images on the Amazon EC2 server. Navigate to **Hosts** → **New Host**. After entering a name for the new host, select a host group from the **Host Group** list to populate most of the new host's fields. Select an EC2 connection and compute profile to automatically populate the virtual-machine-based settings.

On the **Interface** tab, verify that the host's interface information is already populated with values. Satellite Server will automatically select an IP address and set the **Managed**, **Primary**, and **Provision** options for the first interface on the host. Leave the **MAC address** field blank to allow it to be autogenerated by the cloud provider.

Verify that the fields on the **Operating System** and **Virtual Machine** tabs are populated with values. On the **Parameters** tab, ensure that a parameter exists that provides an activation key. If not, add an activation key. Click **Submit** to save your host entry, which immediately triggers the EC2 service to create the instance, using the specified image as the new host's boot volume. You can use the **hammer** command to perform the same task:

```
[root@satellite ~]# hammer host create --name "ec2-test1" \
--organization "Finance" --location "Boston" --hostgroup "Base" \
--compute-resource "EC2_region_context" --provision-method image \
--image "RHEL7 EC2 AMI" --enabled true --managed true \
--interface "managed=true,primary=true,provision=true,subnet_id=EC2" \
--compute-attributes="flavor_id=m1.small,image_id=TestImage,availability_zones=us-east-1a,security_group_ids=Default,managed_ip=Public"
```

## Connecting to an Amazon EC2 instance using SSH

To connect remotely to an Amazon EC2 instance provisioned from Satellite Server, use the compute resource's private key to authenticate to the EC2 API and access the deployed instance.

The compute resource private key is located in the Foreman database in PostgreSQL. The key must be retrieved from the database and saved in a key-file format. On Satellite Server, first determine the ID of the compute resource for which you need the access key. Then switch to the **postgres** user, initiate a **postgres** shell, and connect to the Foreman database. After you have connected, use an SQL command to display the key. For **compute\_resource\_id**, use the compute resource ID located with the first **hammer** command.

```
[root@satellite ~]# hammer compute-resource list
[root@satellite ~]# su - postgres
[postgres@satellite ~]$ psql
# postgres=# \c foreman
# select secret from key_pairs where compute_resource_id = X; secret
```

Copy the key between **-----BEGIN RSA PRIVATE KEY-----** and **-----END RSA PRIVATE KEY-----**. Do not include the begin and end marking text in the created key file. Create a **.pem** file, paste the key into the file, and set file permissions to allow only the owner to use or read the file. Without correct permissions, SSH will not allow the key to work. Any user can access the EC2 instance with this key.

```
[root@satellite ~]# vim keyname.pem
[root@satellite ~]# chmod 600 keyname.pem
[root@satellite ~]# ssh -i keyname.pem ec2-user@example.aws.com
```

The deployed host is now available for management by the cloud-based Satellite or Capsule Server in similar fashion to procedures discussed in this course for on-premise content hosts.



### References

#### Red Hat in the Public Cloud

<https://access.redhat.com/public-cloud>



### References

For more information, refer to the *Provisioning Guide* guide at

[https://access.redhat.com/documentation/en-us/red\\_hat\\_satellite/6.6/html-single/provisioning\\_guide/index](https://access.redhat.com/documentation/en-us/red_hat_satellite/6.6/html-single/provisioning_guide/index)

For more information, refer to the *Managing Hosts* guide at

[https://access.redhat.com/documentation/en-us/red\\_hat\\_satellite/6.6/html-single/managing\\_hosts/index](https://access.redhat.com/documentation/en-us/red_hat_satellite/6.6/html-single/managing_hosts/index)

## ► Quiz

# Managing Content Hosts on a Cloud Platform

Choose the correct answers to the following questions:

► 1. **Which four common provisioning tasks are the basic steps to provision on a cloud platform? (Choose four.)**

- a. Define the necessary virtualized hardware settings.
- b. Configure the DNS server domains for forward and reverse records for deployed hosts.
- c. Create disk partitions and file systems on Elastic Block Storage volumes.
- d. Define how to connect to the cloud provider API and compute region.
- e. Create a new host from the Satellite web console.
- f. Upload supported images to the cloud provider region.

► 2. **Which cloud provider requires you to include an HTTP proxy for Pulp synchronization in the connection configuration?**

- a. Alibaba Cloud
- b. Amazon Web Services
- c. Google Cloud Platform
- d. IBM Cloud
- e. Microsoft Azure

## ► Solution

# Managing Content Hosts on a Cloud Platform

Choose the correct answers to the following questions:

- ▶ 1. **Which four common provisioning tasks are the basic steps to provision on a cloud platform? (Choose four.)**
  - a. Define the necessary virtualized hardware settings.
  - b. Configure the DNS server domains for forward and reverse records for deployed hosts.
  - c. Create disk partitions and file systems on Elastic Block Storage volumes.
  - d. Define how to connect to the cloud provider API and compute region.
  - e. Create a new host from the Satellite web console.
  - f. Upload supported images to the cloud provider region.
  
- ▶ 2. **Which cloud provider requires you to include an HTTP proxy for Pulp synchronization in the connection configuration?**
  - a. Alibaba Cloud
  - b. Amazon Web Services
  - c. Google Cloud Platform
  - d. IBM Cloud
  - e. Microsoft Azure

## Summary

---

In this chapter, you learned:

- Satellite and Capsule Servers can be deployed on supported cloud providers to manage cloud-based content hosts.
- Cloud-based content hosts are image-deployed using the cloud provider's native services.
- Deployed content hosts can be Satellite-managed similar to on-premise content hosts.
- Provisioning using Kickstart and PXE methods or DHCP-managed services are not workable in the cloud.
- Cloud Provider resources, such as images, profiles and templates, reside in the cloud but are configured in Satellite to make cloud provisioning of content hosts simple.

## Chapter 11

# Performing Red Hat Satellite Server Maintenance

### Goal

Maintain Red Hat Satellite for security, recoverability, and growth.

### Objectives

- Create users and groups, and assign roles and permissions, to securely delegate Red Hat Satellite tasks.
- Perform backup and restore operations on Red Hat Satellite Servers and Satellite Capsule Servers, including databases and content stores.
- Describe the distributed architecture and management features of Red Hat Satellite databases.
- Perform maintenance tasks on a Red Hat Satellite Server database.
- Describe the package content export structure, and perform content exports and imports for content migration.

### Sections

- Configuring Users and Roles for Task Delegation (and Guided Exercise)
- Configuring Backup and Restore Operations (and Guided Exercise)
- Managing Red Hat Satellite Databases (and Guided Exercise)
- Exporting and Importing Content Views (and Guided Exercise)

### Lab

Performing Red Hat Satellite Server Maintenance

# Configuring Users and Roles for Task Delegation

---

## Objectives

After completing this section, you should be able to create users and groups, and assign roles and permissions, to securely delegate Red Hat Satellite tasks.

## Managing Satellite Users

In Red Hat Satellite 6, a user is a unique individual who can access and use the system. Every user has a profile, which contains information such as their name, email address, and password. The locations, organizations, and roles assigned to a Satellite user determine which objects they can view or manipulate within Satellite Server.

Navigate to **Administer** → **Users** to manage Satellite users. The **Create User** button creates a new Satellite user. Specify the **Username**, **Email Address**, and **Authorized by** fields in the initial **User** tab that displays. The **Authorized by** field points to the service that provides user storage and authentication, such as an external LDAP server that has been integrated. Choosing **INTERNAL** in this field creates a new user record in Satellite's internal database, requiring that a new password is also entered.

The values selected in the **Locations** and **Organizations** tabs limit the scope of which Satellite objects the user can access. Set the **Default on login** field to the most commonly used location and organization for that user. That default context is set each time the user logs in, but the user can switch to any other assigned location or organization after they log in.

Roles determine the Satellite resources a user can access and manage within Satellite. A user can be assigned multiple roles, as specified in the **Roles** tab in the user's profile. The special **Administrator** role grants a user full access and permissions to Satellite Server.

SSH public keys can be assigned to a user on the **SSH Keys** tab. These keys can be deployed on systems provisioned by the user, allowing the user to log in without a password. Click **Submit** after all the tabs have been completed and reviewed.

Administrators can edit or delete existing users using the **Administer** → **Users** page. To edit a user, click the hyperlink in the **Username** field to access the user's profile. To delete a user account, click **Delete** in the **Actions** column.

## Managing Satellite Roles

Red Hat Satellite uses *Role Based Access Control (RBAC)* to control which Satellite resources users can access and what actions they can perform upon those resources. Satellite is configured with predefined roles for standard Satellite tasks. Because predefined roles are used for configuration management integration, and can also be used by external tools, those roles are locked to prevent changes. Satellite administrators can create new roles manually or by cloning and customizing any existing role, including predefined roles. Assigning roles to users and groups facilitates user access and privilege management.

Manage roles using the **Administer** → **Roles** page. The **Create Role** button initiates a new role. Enter a unique name, the click **Submit** to create the role without filters. You must create the new role before you can manage its filters.

An alternative way to create a role is to clone an existing one. Navigate to **Administer** → **Roles** and select **Clone** from the **Actions** list on the existing role's row. Enter a unique name, and then click **Submit**. The new role initially has the same filters as the original role.

Role filters grant permissions to Satellite resources. To manage role filters, click the role name on the **Administer** → **Roles** page, and then click the **Filters** tab. Existing filters for cloned roles appear as a list, sorted by resource type.

Existing role filters can be deleted or edited using the menu in the **Actions** column. Click **New Filter** to create a new filter for the selected role. Grant access to a resource type by selecting it from the **Resource Type** menu. A list of available permissions for that resource displays. Each resource type has a list of valid user permissions for that type. Typically, permissions for viewing, creating, editing, and deleting are available.

By default, role filters apply to all resources of the selected type. Clearing the **Unlimited?** check box can provide more granular control. Clicking the **Search** field generates a menu of field names and operators to use to select resources. The syntax of this search expression is:

```
field_name operator value
```

This expression limits the resources this filter will match for this role.

## Managing Satellite User Groups

Individual Satellite users are assigned roles that grant privileges to Satellite resources. Creating Satellite user groups allows Satellite administrators to manage collections of Satellite users. Roles can be assigned to user groups, providing additional privileges to group members.

Manage user groups on the **Administer** → **User Groups** page. The **Create User Group** button creates a new user group. Provide a unique name, then select users to be members of the group. User groups can also contain other user groups. Roles are assigned to the user group using the **Roles** tab.

Administrators can edit or delete existing user groups on the **Administer** → **User Groups** page. To edit a user group, click the hyperlink in the **Name** field to access the user group's profile. To delete a user group, click **Delete** in the **Actions** column.

## Authenticating LDAP Users

Red Hat Satellite supports the use of LDAP servers for authenticating users. Satellite user groups can also be mapped to external user groups provided by an LDAP server. Satellite supports POSIX-compliant LDAP, Red Hat Identity Manager (FreeIPA), and Microsoft Active Directory servers as authentication sources.

Navigate to **Administer** → **LDAP Authentication** to manage external authentication sources. Click **Create Authentication Source** to begin configuring a new authentication source.

The **LDAP server** tab is where LDAP host connections are configured. Required information includes the server's fully qualified host name, the network port, whether to use encrypted communication, and the LDAP server type (POSIX, FreeIPA, or Active Directory). Click **Test Connection** to verify that the specified settings communicate with the LDAP server.

The **Account** tab specifies the privileged account configured to perform queries on the LDAP server. The top-level domain name of the LDAP directory is specified in the **Base DN** field. If desired, enter a custom LDAP search filter in the **LDAP filter** field to limit LDAP queries which

can improve server efficiency and response times. Select the **Automatically Create Accounts in Satellite** to have the Satellite Server create a corresponding Satellite user the first time an LDAP user authenticates. The Satellite user account stores authorization information, such as role, group, and permission assignments. The LDAP server continues to store and validate the user authentication and password.

The **Attribute mappings** tab is used to map LDAP attributes to Satellite user profile data elements. Attributes that can be mapped include the login name, first name, surname, email address, and photo.



### References

For more information, refer to the *Managing Users and Roles* chapter in the *Administering Red Hat Satellite* Guide at  
[https://access.redhat.com/documentation/en-us/red\\_hat\\_satellite/6.6/html-single/administering\\_red\\_hat\\_satellite/index#chap-Red\\_Hat\\_Satellite-Administering\\_Red\\_Hat\\_Satellite-Users\\_and\\_Roles](https://access.redhat.com/documentation/en-us/red_hat_satellite/6.6/html-single/administering_red_hat_satellite/index#chap-Red_Hat_Satellite-Administering_Red_Hat_Satellite-Users_and_Roles)

## ► Guided Exercise

# Configuring Users and Roles for Task Delegation

In this exercise, you will create a new user with a granular administrative role.

### Outcomes

You should be able to:

- Create a new Satellite role.
- Create a new Satellite user.

### Before You Begin

Log in as the **student** user on **workstation** using **student** as the password.

Run the **lab maintain-user start** command. This command verifies that **satellite** is available and an organization named **Operations** exists.

```
[student@workstation ~]$ lab maintain-user start
```

The Satellite administrator would like to delegate user administration of the **Operations** department to another user. The new administrator account should be able to view existing users and create new users, but not delete users. This new account should only have visibility and control of the users in the **Operations** organization.

Using the existing Red Hat Satellite Server on **satellite.lab.example.com**, you will create a new role called **useradmin**. You will create a user in the **Operations** organization called **userboss** and assign them the **useradmin** role. You will log in to the **userboss** account and create a new user called **userptest** with default permissions. The **userboss** account should not be able to delete users.

- 1. On the Satellite Server, create the **useradmin** role.
  - 1.1. Open a browser and navigate to <https://satellite.lab.example.com>. Log in to the Satellite web UI as **admin** using **redhat** as the password.
  - 1.2. Click **Administer** → **Roles**. Note that only the first 20 predefined roles display. They display with “lock” icons because they cannot be changed.
  - 1.3. Go to the second page of the list of roles to display the **System admin** role. That predefined role manages users, roles, organizations, locations, and other resources. Click the icon to the right of the **Filters** button in the **System admin** row and select **Clone** from the menu that appears.
  - 1.4. Enter **useradmin** in the **Name** field of the **Role** page. Leave the other fields blank, and then click **Submit**.
- 2. Restrict the **useradmin** role so that it can create and edit users, but not delete them.

- 2.1. Click **Filters** at the end of the row for **useradmin**.
  - 2.2. In the **Actions** column, select **Delete** for all of the resources except (**Miscellaneous**), **Filter**, **Location**, **Organization**, **Role**, and **User**.
  - 2.3. For the **Location** resource, click **Edit**. Clear all other selected items except **assign\_locations** and **view\_locations**. Click **Submit** to confirm your changes to the resource filters.
  - 2.4. Edit the **Organization** resource so that only **assign\_organizations** and **view\_organizations** remain selected.
  - 2.5. Edit the **Role** resource so that only **view\_roles** remains selected.
  - 2.6. Edit the **User** resource so that only **create\_users**, **edit\_users**, and **view\_users** remain selected.
- 3. Create the **userboss** user. Limit the account so that they can only access the **Operations** organization. Assign the **useradmin** role to this new account.
- 3.1. Click **Administer → Users**.
  - 3.2. Click **Create User**. Complete the fields on the **User** tab with the following details:
- | Field                        | Value                                 |
|------------------------------|---------------------------------------|
| <b>Username</b>              | <b>userboss</b>                       |
| <b>Email Address</b>         | <b>root@satellite.lab.example.com</b> |
| <b>Authorized by</b>         | <b>INTERNAL</b>                       |
| <b>Password (and Verify)</b> | <b>redhat</b>                         |
- Leave all other fields blank because they are not required.
- 3.3. Click the **Organizations** tab. Click **Operations** to move it to the **Selected items** column. Click **Default Organization** to move it to the **All items** column.  
Set the **Default on login** organization to **Operations**.
  - 3.4. Click the **Roles** tab. Click **useradmin** to move it to the **Selected items** column.
  - 3.5. Click **Submit** after you have confirmed all your selections are correct.
  - 3.6. Log out of the **admin** account.
- 4. Log in as **userboss** and create the **usertest** user.
- 4.1. Log in to the Satellite web UI as **userboss** using **redhat** as the password. A permission denied screen displays because this user cannot manage hosts. Note that only the **Monitor** and **Administrator** tabs are available.
  - 4.2. Click **Administer → Users**. Only **userboss** appears in the list of users because this user can only see users in the **Operations** organization.
  - 4.3. Click **Create User**. Complete the fields in the **User** tab with the following details:

| Field                 | Value                          |
|-----------------------|--------------------------------|
| Username              | usertest                       |
| Email Address         | root@satellite.lab.example.com |
| Authorized by         | INTERNAL                       |
| Password (and Verify) | redhat                         |

Leave all other fields blank because they are not required.

- 4.4. Click the **Roles** tab. Click **Remote Execution User** to move it to the **Selected items** column.
- 4.5. Click **Submit** after you have confirmed all your selections are correct.
- 4.6. Log out of the **userboss** account.
- ▶ 5. Log in as **usertest** to see the access the new user has.
  - 5.1. Log in to the Satellite web UI as **usertest** using **redhat** as the password.
  - 5.2. The main menu that **usertest** sees is limited. It only displays the **Monitor**, **Hosts**, **Infrastructure**, and **Administer** tabs.
  - 5.3. Log out of the **usertest** account.

## Finish

On the **workstation** machine, use the **lab** command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab maintain-user finish
```

This concludes the guided exercise.

# Configuring Backup and Restore Operations

## Objectives

After completing this section, you should be able to perform backup and restore operations on Red Hat Satellite Servers and Satellite Capsule Servers, including databases and content stores.

## Backing Up Red Hat Satellite Server

Having backups available is an essential part of a disaster recovery plan. Red Hat Satellite Server provides the **satellite-maintain backup** utility to save content externally for later access.

The **root** user on the Satellite Server executes the **satellite-maintain backup** command. The command has the following syntax:

```
[root@satellite ~]# satellite-maintain backup {online|offline} target-directory
```

The **offline** and **online** subcommands determine whether Satellite Server remains available during the backup. Specify the target directory where a subdirectory called **satellite-backup-YYYY-MM-DD-HH-MM-SS** will automatically be created to contain the backup files. These files should be stored offline for future retrieval.

Before starting the backup, ensure there is enough storage available for the backup to complete. The backup utility compresses the Satellite database information as it is extracted and saved. The *Administering Red Hat Satellite Guide* provides a detailed method for estimating the amount of space required.

The Pulp content, which is all of the repository software packages managed by Satellite Server, constitutes the largest amount of data that is backed up. The packages are compressed files on disk so this archive data is not compressed again during the backup. The **--skip-pulp-content** option excludes the Pulp content when performing the backup. Creating and restoring a full system requires backing up Pulp content, so use the **--skip-pulp-content** option only for testing and specialized backups.



### Note

Backups contain sensitive data, such as SSH keys and SSL certificates. Store backups securely to avoid unauthorized access to hosts managed by Satellite Server.

The **offline** subcommand initiates a cold backup that requires the database to be offline during the backup. When an offline backup is performed, Satellite Server is not available for use. The **online** subcommand performs a backup while Satellite Server remains available, but some operations should be avoided during the backup to ensure that the backup is consistent for later restore. Operations involving content views, sync plans, and repository administration should not be performed.

## Creating Incremental Backups

Full backups can take a long time to run. The **--incremental** option initiates a partial backup of Satellite Server with only the changes made since an earlier full or incremental backup. This option requires including an additional directory argument to specify the previous backup directory. An incremental backup command has the following syntax:

```
[root@satellite ~]# satellite-maintain backup {online|offline} \
--incremental previous-backup-dir target-directory
```

Incremental backups can be performed online or offline. The restrictions and limitations for the backup subcommand also apply when an incremental backup is performed.



### References

For more information, refer to the *Backing Up Satellite Server and Capsule Server* chapter in the *Administering Red Hat Satellite Guide* at  
[https://access.redhat.com/documentation/en-us/red\\_hat\\_satellite/6.6/html-single/administering\\_red\\_hat\\_satellite/index#chap-Red\\_Hat\\_Satellite-Administering\\_Red\\_Hat\\_Satellite-Backup\\_and\\_Disaster\\_Recovery](https://access.redhat.com/documentation/en-us/red_hat_satellite/6.6/html-single/administering_red_hat_satellite/index#chap-Red_Hat_Satellite-Administering_Red_Hat_Satellite-Backup_and_Disaster_Recovery)

## Restoring Red Hat Satellite Server

The **satellite-maintain restore** command restores Red Hat Satellite Server backup data. To start a full disaster recovery process, start with a freshly installed Satellite Server. Previous backups can also be restored onto an existing system, which can overwrite existing data and content. The host name and other system configuration should be the same as the original system that was backed up. The **satellite-maintain restore** command will validate this information, and will refuse to restore backups to a differently named or configured server. The following command restores from a previous backup:

```
[root@satellite ~]# satellite-maintain restore backup-directory
```

When restoring incremental backups, follow the complete chronological backup sequence. First, restore the most recent full backup, then restore each incremental backup in the same order they were created.



### References

For more information, refer to the *Backing Up Satellite Server and Capsule Server* chapter in the *Administering Red Hat Satellite Guide* at  
[https://access.redhat.com/documentation/en-us/red\\_hat\\_satellite/6.6/html-single/administering\\_red\\_hat\\_satellite/index#sect-Red\\_Hat\\_Satellite-Administering\\_Red\\_Hat\\_Satellite-Backup\\_and\\_Disaster\\_Recovery-Restoring\\_Satellite\\_Server\\_or\\_Capsule\\_Server\\_from\\_a\\_Backup](https://access.redhat.com/documentation/en-us/red_hat_satellite/6.6/html-single/administering_red_hat_satellite/index#sect-Red_Hat_Satellite-Administering_Red_Hat_Satellite-Backup_and_Disaster_Recovery-Restoring_Satellite_Server_or_Capsule_Server_from_a_Backup)

## ► Guided Exercise

# Configure Backup and Restore Operations

In this exercise, you will perform backup operations on a Satellite Server and verify that the server is operational again.

### Outcomes

You should be able to:

- Perform a full backup of a Satellite Server.
- Locate and identify the backup files that are created.
- Restore a Satellite Server from backup.

### Before You Begin

Log in as the **student** user on **workstation** using **student** as the password.

Run the **lab maintain-backup start** command. This command verifies that **satellite** is available.

```
[student@workstation ~]$ lab maintain-backup start
```

Because of recent changes, it necessary to perform a backup. Assume that the package content has been saved previously. Back up only the other Satellite data and configuration.

► 1. On **satellite**, create a **/var/tmp/backup** directory to store backup data.

- 1.1. On **workstation**, use **ssh** to log in to **satellite** as **student**. Use **sudo -i** to switch to **root**.

```
[student@workstation ~]$ ssh satellite
[student@satellite ~]$ sudo -i
[sudo] password for student: student
[root@satellite ~]#
```

- 1.2. Create the directory for the backup files.

```
[root@satellite ~]# mkdir /var/tmp/backup
```

► 2. Back up the Satellite Server offline, excluding Pulp content.

- 2.1. Initiate the backup using the **satellite-maintain backup** command.

```
[root@satellite ~]# satellite-maintain backup offline --skip-pulp-content \
/var/tmp/backup
...output omitted...
Do you want to proceed?, [y(yes), q(quit)] y
```

2.2. Note the name of the subdirectory created in **/var/tmp/backup**. The **satellite-maintain backup** command displayed a name beginning with **satellite-backup** and includes a time stamp. The following is an example of the output:

```
Done with backup: 2019-11-25 15:43:34 +0000
**** BACKUP Complete, contents can be found in: /var/tmp/backup/satellite-
backup-2019-11-25-15-33-21 ****
```

- ▶ 3. Explore the contents of the created backup subdirectory. Use the name provided by **satellite-maintain backup**. Your file sizes and time stamps will vary from this example.

```
[root@satellite ~]# ls -l /var/tmp/backup
total 0
drwxrwx---. 2 root postgres 164 Nov 25 15:43 satellite-backup-YYYY-MM-DD-HH-MM-SS
[root@satellite ~]# cd /var/tmp/backup/satellite-backup-YYYY-MM-DD-HH-MM-SS
[root@satellite satellite-backup-YYYY-MM-DD-HH-MM-SS]# du -sh .
591M .
[root@satellite satellite-backup-YYYY-MM-DD-HH-MM-SS]# ls -l
total 603580
-rw-r--r--. 1 root root 51619061 Nov 25 15:40 config_files.tar.gz
-rw-r--r--. 1 root root 45077 Nov 25 15:39 metadata.yml
-rw-r--r--. 1 root root 509343314 Nov 25 15:40 mongo_data.tar.gz
-rw-r--r--. 1 root root 57046763 Nov 25 15:40 pgsql_data.tar.gz
```

- ▶ 4. On Satellite Server, update the **admin** description, and create a user named **backtest**.
  - 4.1. Navigate to <https://satellite.lab.example.com> and log in to Satellite as **admin** with the **redhat** password.
  - 4.2. Click **Administer → Users** and then click the **admin** user. Modify the **Description** field by adding text of your choosing. Click **Submit** to save the changes.
  - 4.3. Click **Create User** and complete the **User** tab with the following details. Leave all other fields unchanged as they are not required.

| Field                        | Value                                 |
|------------------------------|---------------------------------------|
| <b>Username</b>              | <b>backtest</b>                       |
| <b>Email Address</b>         | <b>root@satellite.lab.example.com</b> |
| <b>Authorized by</b>         | <b>INTERNAL</b>                       |
| <b>Password (and Verify)</b> | <b>redhat</b>                         |

Ensure the details are correct and then click **Submit** to create the new user.

- 4.4. Log out of the **admin** account.
- ▶ 5. Restore the Satellite Server to the state that it was in when you performed the backup.



**Note**

The restoration process takes approximately 20 minutes.

Return to the terminal running on **satellite**. As the **root** user, use the **satellite-maintain restore** command to restore Satellite to its previous state. Provide the name of the subdirectory created earlier.

```
[root@satellite ~]# satellite-maintain restore /var/tmp/backup/satellite-
backup-YYYY-MM-DD-HH-MM-SS
...output omitted...
Do you want to proceed?, [y(yes), q(quit)] y
```

- ▶ 6. When the restoration is complete, verify that Satellite Server has been returned to the original state before the backup. The **admin** user description is reverted and the **backtest** user does not exist.
  - 6.1. Log in to Satellite as **admin** using **redhat** as the password.
  - 6.2. Navigate to **Administer** → **Users**. Note the **backtest** user does not exist.
  - 6.3. Select the **admin** user. The **Description** field is restored to its original value.
  - 6.4. Log out of the **admin** account.
- ▶ 7. Reclaim the disk space taken up by the backup files.

```
[root@satellite ~]# rm -rf /var/tmp/backup
```

- ▶ 8. Log off from the **satellite** host and return to **workstation**.

## Finish

On the **workstation** machine, use the **lab** command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab maintain-backup finish
```

This concludes the guided exercise.

# Managing Red Hat Satellite Databases

## Objectives

After completing this section, you should be able to:

- Describe the distributed architecture and management features of Red Hat Satellite databases.
- Perform maintenance tasks on a Red Hat Satellite Server database.

## Reclaiming Disk Space from MongoDB

The MongoDB database on a Red Hat Satellite Server stores all of the Satellite resource records. In a heavily loaded Satellite Server deployment, this database can consume a large amount of disk space.

Before manipulating the MongoDB database, perform a full offline backup. Verify that there is enough free space for a database repair to take place. The procedure requires enough free space for the size of the MongoDB database plus 2 GB of additional temporary disk storage.

- Stop all Pulp services as **root** on the Satellite Server:

```
[root@satellite ~]# systemctl stop goferd httpd pulp_workers pulp_celerybeat \
    pulp_resource_manager pulp_streamer
```

- Start a MongoDB shell using the Pulp database on the Satellite Server:

```
[root@satellite ~]# mongo pulp_database
```

- Query the disk space used to ensure there is enough free space:

```
> db.stats()
```

- Start the Pulp database repair. It can take a long time to complete. The database will be offline during the procedure:

```
> db.repairDatabase()
```

- Verify that disk space has been reclaimed as unused, and then exit the MongoDB shell:

```
> db.stats()
...output omitted...
> exit
```

- Restart the Pulp services:

```
[root@satellite ~]# systemctl start goferd httpd pulp_workers pulp_celerybeat \
    pulp_resource_manager pulp_streamer
```

## Managing Satellite Audit Records

Satellite Server keeps a record of commands that are performed on the server. The **hammer audit** command queries the audit records. The **list** subcommand displays audit records, using options to select the types of audit records to list. The **info** subcommand displays more information about individual audit records.

The **foreman-rake** command can manage Satellite Server audit records. The **audits:expire** subcommand removes old audit records. The following command removes all audit records that are more than 30 days old:

```
[root@satellite ~]# foreman-rake audits:expire days=30
```

When the **days=N** argument is omitted, the default retention period is 90 days.

The **foreman-rake** command also supports the **audits:anonymize** subcommand. Although this command does not save space by keeping all of the audit records, it removes user account and IP information for old audit records. This subcommand also has a **days=N** option that specifies how old records must be to be anonymized.

## Cleaning Unused Tasks

Status information about scheduled tasks remains on Satellite Server after tasks complete, both for successful tasks and tasks that exit with an error. A **cron** job to automatically remove old tasks can be enabled, but it is disabled by default. The following command enables the **cron** job to automatically remove old tasks:

```
[root@satellite ~]# satellite-installer --foreman-plugin-tasks-automatic-cleanup \
    true
```

When it is enabled, the **cron** job runs every day at 19:45 by default. The following command changes the cleanup time to be 23:30 every day:

```
[root@satellite ~]# satellite-installer --foreman-plugin-tasks-cron-line \
    '30 23 * * *'
```

The syntax for the time specification uses standard **cron** syntax.

More advanced cleanup task configuration can be made by adjusting settings found in the **/etc/foreman/plugins/foreman-tasks.yaml** file. Rules are defined in this file that define the age of tasks that are deleted each time the **cron** job runs.



### References

For more information, refer to the *Managing Satellite Server* chapter in the *Administering Red Hat Satellite* Guide at  
[https://access.redhat.com/documentation/en-us/red\\_hat\\_satellite/6.6/html-single/administering\\_red\\_hat\\_satellite/index#chap-Red\\_Hat\\_Satellite-Administering\\_Red\\_Hat\\_Satellite-Maintaining\\_a\\_Red\\_Hat\\_Satellite\\_Server](https://access.redhat.com/documentation/en-us/red_hat_satellite/6.6/html-single/administering_red_hat_satellite/index#chap-Red_Hat_Satellite-Administering_Red_Hat_Satellite-Maintaining_a_Red_Hat_Satellite_Server)

## ► Guided Exercise

# Managing Red Hat Satellite Databases

In this exercise, you will perform maintenance tasks on a Red Hat Satellite Server database.

### Outcomes

You should be able to:

- Clean up audit records.
- Clean up unused tasks.
- Reclaim space from MongoDB.

### Before You Begin

Log in as the **student** user on **workstation** using **student** as the password.

Run the **lab maintain-db start** command. This command verifies that **satellite** is available.

```
[student@workstation ~]$ lab maintain-db start
```

- 1. On **satellite**, clean up audit records older than one day.

- 1.1. On **workstation**, use **ssh** to log in to **satellite** as **student**. Use **sudo -i** to switch to **root**.

```
[student@workstation ~]$ ssh satellite
[student@satellite ~]$ sudo -i
[sudo] password for student: student
[root@satellite ~]#
```

- 1.2. Use the **hammer** command to count the number of audit records.

```
[root@satellite ~]# hammer audit list | wc -l
142
```



#### Note

The number of audit records will vary based on your Satellite usage.

- 1.3. Use the **foreman-rake** command to clean up audit records older than one day.

```
[root@satellite ~]# foreman-rake audits:expire days=1
Deleting audits older than 2019-11-19 06:00:22 UTC. This might take a few
minutes...
Successfully deleted 138 audits!
```

► 2. Enable the task clean-up **cron** job.

- 2.1. The **cron** job configuration can be modified by editing the **/etc/foreman/plugins/foreman-tasks.yaml** file. Examine this file to view the configuration settings and their default values. Do not make any changes.

```
:foreman-tasks:  
#  
# Logging configuration can be changed by uncommenting the loggers  
# section and the logger configuration desired.  
#  
#   :loggers:  
#     :dynflow:  
#       :enabled: true  
#       :action:  
#         :enabled: true  
  
# Task backup configuration can be changed by altering the values in  
# the backup section  
#  
:backup:  
#  
# Whether to back up tasks when they are removed  
#  
:backup_deleted_tasks: true  
#  
# Where to put the tasks which were backed up  
#  
:backup_dir: /var/lib/foreman/tasks-backup  
  
# Cleaning configuration: how long should the actions be kept before deleted  
# by `rake foreman_tasks:clean` task  
#  
:cleanup:  
#  
# the period after which to delete all the tasks (by default all tasks are not  
# being deleted after some period)  
# will be deprecated in Foreman 1.18 and the use of rules is recommended.  
#  
#   :after: 30d  
#  
# per action settings to override the default defined in the actions  
(self.cleanup_after method)  
#  
#   :actions:  
#     - :name: Actions::Foreman::Host::ImportFacts  
#       :after: 10d  
#  
# Rules defined in this section by default don't operate  
# on tasks specified in the actions section. This behavior  
# can be overridden by setting the override_actions to true  
:rules:  
# Delete successful tasks after a month  
- :filter: result = success  
:after: 30d
```

```
# Delete everything (any action, any state) after one year
- :states: all # Either list of state names or all
  :after: 1y
  :override_actions: true
```

2.2. Use the **satellite-installer** command to enable the task clean up **cron** job.

```
[root@satellite ~]# satellite-installer \
--foreman-plugin-tasks-automatic-cleanup true
Package versions are locked. Continuing with unlock.
Installing           Done          [100%] [.....]
Package versions are being locked.
Success!
* Satellite is running at https://satellite.lab.example.com

* To install an additional Capsule on separate machine continue by running:

  capsule-certs-generate --foreman-proxy-fqdn "$CAPSULE" --certs-tar "/root/
$CAPSULE-certs.tar"

* To upgrade an existing 6.5 Capsule to 6.6:
  Please see official documentation for steps and parameters to use when
upgrading a 6.5 Capsule to 6.6.

The full log is at /var/log/foreman-installer/satellite.log
```

2.3. View the resulting **cron** job. Note the time specification, the command that is executed, and the file where the command output is saved.

```
[root@satellite ~]# cat /etc/cron.d/foreman-tasks
SHELL=/bin/sh
RAILS_ENV=production
FOREMAN_HOME=/usr/share/foreman

# Clean up expired tasks from the database

45 19 * * *    foreman    /usr/sbin/foreman-rake foreman_tasks:cleanup >>/var/log/
foreman/cron.log 2>&1
```

► 3. Reclaim unused space in the MongoDB Pulp database.

3.1. Stop all Pulp services.

```
[root@satellite ~]# systemctl stop goferd httpd pulp_workers pulp_celerybeat \
pulp_resource_manager pulp_streamer
Failed to stop goferd.service: Unit goferd.service not loaded.
```



**Note**

The **goferd** service will only be running if your Satellite is registered to itself as a host. In this classroom that is not the case, so the error above is safe to ignore.

3.2. Enter the MongoDB shell.

```
[root@satellite ~]# mongo pulp_database
MongoDB shell version v3.4.9
connecting to: mongodb://127.0.0.1:27017/pulp_database
MongoDB server version: 3.4.9
Welcome to the MongoDB shell.
For interactive help, type "help".
For more comprehensive documentation, see
  http://docs.mongodb.org/
Questions? Try the support group
  http://groups.google.com/group/mongodb-user
>
```

3.3. Verify the current index size.

```
> db.stats()
{
  "db" : "pulp_database",
  "collections" : 47,
  "views" : 0,
  "objects" : 35124,
  "avgObjSize" : 14300.054948183579,
  "dataSize" : 502275130,
  "storageSize" : 377286656,
  "numExtents" : 0,
  "indexes" : 161,
  "indexSize" : 16715776,
  "ok" : 1
}
```

3.4. Reclaim unused space in the database.

```
> db.repairDatabase()
{ "ok" : 1 }
```

3.5. Determine the new index size, then exit from the MongoDB shell.

```
> db.stats()
{
  "db" : "pulp_database",
  "collections" : 47,
  "views" : 0,
  "objects" : 35124,
  "avgObjSize" : 14300.054948183579,
  "dataSize" : 502275130,
  "storageSize" : 377286656,
  "numExtents" : 0,
  "indexes" : 161,
  "indexSize" : 8937472,
  "ok" : 1
}
```

```
}
```

> **exit**  
bye

3.6. Restart all Pulp services, then exit from the SSH session.

```
[root@satellite ~]# systemctl start goferd httpd pulp_workers pulp_celerybeat \
pulp_resource_manager pulp_streamer
Failed to start goferd.service: Unit not found.
[root@satellite ~]# logout
[student@satellite ~]$ logout
[student@workstation ~]$
```

## Finish

On the **workstation** machine, use the **lab** command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab maintain-db finish
```

This concludes the guided exercise.

# Exporting and Importing Content Views

## Objectives

After completing this section, you should be able to:

- Describe situations where transferring content views is desirable.
- Export a content view.
- Import a content view archive.
- Describe the layout of a content view archive.

## Describing Import and Export Use Cases

There are many use cases for exporting and importing content with a Satellite Server. The following are examples where export and import are required:

- Your organization has more than one Satellite Server, and one or more may not have internet access. Updated content is exported from the connected Satellite to import to the disconnected one.
- Your organization has a central Satellite Server to host all content, and uses only content subsets on other Satellite Servers.
- Your organization has a staging Satellite host used to test upgrades to Satellite itself before upgrading the production Satellite Server.
- You test content views against representative systems, before duplicating the content views to other organizations on the same Satellite Server.

## Exporting Content Views

Verify that prerequisites are satisfied before exporting a content view. The following list specifies the settings for each repository included in the content view:

- The download policy must be set to **immediate**.
- The **Mirror on Sync** setting must be cleared (shown as **No**).
- The repository must be synchronized up to the required date.

With the prerequisites verified, use this process to export a content view:

- Determine the ID of the content view version to export. This example uses the Demo content view version 1.0.

```
[root@demo ~]# hammer content-view version list --organization Demo
-----|-----|-----|-----|-----|-----|-----|-----|
ID | NAME           | VERSION | LIFECYCLE ENVIRONMENTS
-----|-----|-----|-----|-----|-----|-----|-----|
3  | Demo 1.0       | 1.0     | Development
2  | Default Organization View 1.0 | 1.0     | Library
-----|-----|-----|-----|-----|-----|-----|-----|
```

- Export the content view to a directory. The archive name will match the format **export-cv\_name-cv\_version.tar**.

```
[root@demo ~]# hammer content-view version export --export-dir /tmp/ --id 3
```

## Describing the Layout of a Content View Archive

The archive file generated by the export process contains two files. The first is in JSON format and contains all the metadata for the content view version. The second is an archive containing all the packages and repository metadata exported from Pulp.

```
[root@satellite ~]# tar tvf /var/tmp/export-Demo-1.0.tar
drwxr-xr-x root/root          0 2019-11-27 04:16 export-Demo-1.0/
-rw-r--r-- root/root 3398000640 2019-11-27 04:16 export-Demo-1.0/export-Demo-1.0-
repos.tar
-rw-r--r-- root/root      168954 2019-11-27 04:16 export-Demo-1.0/export-
Demo-1.0.json
```

## Importing Content Views

The import process for a content view archive also has prerequisites:

- A content view with the same name and label must exist.
- The repositories included in the content view must be enabled.
- At the time of writing, you must be in the **/var/lib/pulp/katello-export** directory when performing the import.

Use the following command to import the archive:"

```
[root@demo katello-export]# hammer content-view version import \
--export-tar /tmp/export-Demo-1.0.tar --organization DemoOps
```



### Note

The **--export-tar** option must use an absolute path until bug BZ#1745081 is resolved.



### References

For more information, refer to the *Synchronizing Content Between Satellite Servers* chapter in the *Content Management Guide* at  
[https://access.redhat.com/documentation/en-us/red\\_hat\\_satellite/6.6/html-single/content\\_management\\_guide/index#Using\\_ISS](https://access.redhat.com/documentation/en-us/red_hat_satellite/6.6/html-single/content_management_guide/index#Using_ISS)

**Bug 1745081 - Hammer content view import failing to import when using relative path.**

[https://bugzilla.redhat.com/show\\_bug.cgi?id=1745081](https://bugzilla.redhat.com/show_bug.cgi?id=1745081)

## ► Guided Exercise

# Exporting and Importing Content Views

In this exercise, you will export life-cycle content to disk, and query the resulting package storage structure.

### Outcomes

You should be able to:

- Export a content view to disk.
- Describe the layout of the content view archive.

### Before You Begin

Log in as the **student** user on **workstation** using **student** as the password.

Run the **lab maintain-views start** command. This command verifies that **satellite** is available.

```
[student@workstation ~]$ lab maintain-views start
```

- 1. Navigate to <https://satellite.lab.example.com> and log in to the Satellite web UI as **admin** using **redhat** as the password. Ensure your organization is set to **Operations**, and verify that the prerequisites for exporting the **Base v1.0** content view are all present.

- 1.1. Navigate to **Content** → **Content Views**, and then click **Base**.
- 1.2. Click **Version 1.0**, and then click **Yum Content** → **Repositories**.
- 1.3. Click **Red Hat Enterprise Linux 8 for x86\_64 - BaseOS RPMs x86\_64 8.1**, and ensure that the **Sync Settings** are set to the following:  
The **Mirror on Sync** setting should be **No**. The **Download Policy** setting should be **Immediate**.
- 1.4. Click **Select Action** and then **Sync Now** to ensure the repository is up-to-date. This may take up to 15 minutes.

- 2. Log in to Satellite over SSH, then identify and export the **Base 1.0** content view.

- 2.1. On **workstation**, use **ssh** to log in to **satellite** as **student**. Use **sudo -i** to switch to **root**.

```
[student@workstation ~]$ ssh satellite
[student@satellite ~]$ sudo -i
[sudo] password for student: student
[root@satellite ~]#
```

- 2.2. List the content views for the **Operations** organization. Note the ID for the **Base 1.0** content view.

```
[root@satellite ~]# hammer content-view version list \
--organization Operations
---|-----|-----|-----|-----|-----|
ID | NAME | VERSION | LIFECYCLE ENVIRONMENTS
---|-----|-----|-----|-----|
5 | Base 2.0 | 2.0 | Library, Development, QA
4 | Base 1.0 | 1.0 |
2 | Default Organization View 1.0 | 1.0 | Library
---|-----|-----|-----|-----|
```

2.3. Export the **Base 1.0** content view to the **/var/tmp** directory.

```
[root@satellite ~]# hammer content-view version export \
--export-dir /var/tmp/ --id 4
```

► 3. On **satellite**, examine the layout of the content view archive.

- 3.1. Verify the size of the content view archive. The name of the file will follow the format **export-cv\_name-cv\_version.tar**

```
[root@satellite ~]# ls -lh /var/tmp/export*
-rw-r--r--. 1 root root 3.2G Nov 27 04:16 /var/tmp/export-Base-1.0.tar
```

3.2. Change into the **/var/tmp** directory and extract the archive.

```
[root@satellite ~]# cd /var/tmp
[root@satellite tmp]# tar xf export-Base-1.0.tar
```

- 3.3. The tar file contains metadata about the content view in JSON format, and another archive containing the packages and repository metadata managed by Pulp.

```
-rw-r--r--. 1 root root 168954 Nov 27 04:16 export-Base-1.0.json
-rw-r--r--. 1 root root 3398000640 Nov 27 04:16 export-Base-1.0-repos.tar
```

- 3.4. View the **export-Base-1.0.json** file content. Note that the content view name and version are specified, as well as the path on disk where the packages were originally located. The errata resolved by the packages in the content view are also documented.

```
{
  "name": "Base",
  "major": 1,
  "minor": 0,
  "composite_components": null,
  "repositories": [
    {
      "id": 4,
      "label": "Red_Hat_Enterprise_Linux_8_for_x86_64_-_BaseOS_RPMs_x86_64_8_1",
      "content_type": "yum",
      "backend_identifier": "3-Base-v1_0-cee47e8c-aae9-4efb-a552-d2243e7c97b1",
      "relative_path": "Operations/content_views/Base/1.0/content/dist/rhel8/8.1/x86_64/baseos/os",
      "errata": [
        {
          "id": 1234567890,
          "severity": "medium",
          "description": "A security update for the libxml2 package in Red Hat Enterprise Linux 8.1. This update addresses multiple security issues in libxml2. It is recommended that users apply this update as soon as possible. For more information, see the advisory at https://access.redhat.com/advisories/RA-2020-0010."
        }
      ]
    }
  ]
}
```

```
"on_disk_path": "/var/lib/pulp/published/yum/https/repos//Operations/
content_views/Base/1.0/content/dist/rhel8/8.1/x86_64/baseos/os",
"rpm_filenames": [
    "aaohan-comfortaa-fonts-3.001-2.el8.noarch.rpm",
    "acl-2.2.53-1.el8.x86_64.rpm",
    "acpica-tools-20180629-3.el8.x86_64.rpm",
    "adcli-0.8.2-2.el8.x86_64.rpm",
    "adcli-0.8.2-3.el8.x86_64.rpm",
    "adcli-doc-0.8.2-2.el8.noarch.rpm",
    ...output omitted...
    "zlib-devel-1.2.11-10.el8.i686.rpm",
    "zsh-5.5.1-6.el8.x86_64.rpm"
],
"errata_ids": [
    "RHBA-2019:3778",
    "RHEA-2019:3779",
    "RHBA-2019:3737",
    ...output omitted...
]
```

3.5. View the content of the **export-Base-1.0-repos.tar** archive.

```
[root@satellite tmp]# tar tvf export-Base-1.0/export-Base-1.0-repos.tar
...output omitted...
Operations/content_views/Base/1.0/content/dist/rhel8/8.1/x86_64/baseos/os/
Packages/y/yum-4.2.7-7.el8_1.noarch.rpm
Operations/content_views/Base/1.0/content/dist/rhel8/8.1/x86_64/baseos/os/
Packages/y/yum-4.2.7-6.el8.noarch.rpm
Operations/content_views/Base/1.0/content/dist/rhel8/8.1/x86_64/baseos/os/
Packages/y/yum-utils-4.0.8-3.el8.noarch.rpm
```

Note that the structure of the archive matches the relative path shown in the previous step. Exit from the Satellite SSH session.

## Finish

On the **workstation** machine, use the **lab** command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab maintain-views finish
```

This concludes the guided exercise.

## ► Lab

# Performing Red Hat Satellite Server Maintenance

In this lab, you will perform maintenance tasks on a Red Hat Satellite Server.

### Outcomes

You should be able to:

- Clean up audit records.
- Reclaim space from MongoDB.

### Before You Begin

Log in as the **student** user on **workstation** using **student** as the password.

Run the **lab maintain-review start** command. This command verifies that **satellite** is available.

```
[student@workstation ~]$ lab maintain-review start
```

1. Navigate to <https://satellite.lab.example.com> and log in to the Satellite web UI as **admin** using **redhat** as the password. Create the **Compliance Auditor** role by combining the **Viewer** and **Auditor** role permissions. The role should have access to the **Finance** and **Operations** organizations.
2. Create the **auditor** user, assign it the **Compliance Auditor** role, and associate it with the **Finance** and **Operations** organizations. Log in as the **auditor** user, and confirm that you have view access to all organizations and audit records.
3. Clean up audit records older than one day.
4. Reclaim unused space in the MongoDB Pulp database.
5. Back up Satellite to **/var/tmp** but exclude Pulp content.
6. Ensure your organization is set to **Operations**, and ensure that the prerequisites for exporting the **Base v2.0** content view have all been met.
7. Export the **Base 2.0** content view to the **/var/tmp** directory.

### Evaluation

As the **student** user on the **workstation** machine, use the **lab** command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab maintain-review grade
```

## Finish

As the **student** user on the **workstation** machine, use the **lab** command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab maintain-db finish
```

This concludes the lab.

## ► Solution

# Performing Red Hat Satellite Server Maintenance

In this lab, you will perform maintenance tasks on a Red Hat Satellite Server.

### Outcomes

You should be able to:

- Clean up audit records.
- Reclaim space from MongoDB.

### Before You Begin

Log in as the **student** user on **workstation** using **student** as the password.

Run the **lab maintain-review start** command. This command verifies that **satellite** is available.

```
[student@workstation ~]$ lab maintain-review start
```

1. Navigate to <https://satellite.lab.example.com> and log in to the Satellite web UI as **admin** using **redhat** as the password. Create the **Compliance Auditor** role by combining the **Viewer** and **Auditor** role permissions. The role should have access to the **Finance** and **Operations** organizations.
  - 1.1. Navigate to **Administer** → **Roles**, locate the **Viewer** role, and then click **Clone** from the **Actions** list.
  - 1.2. Enter **Compliance Auditor** as the name, select the **Finance** and **Operations** organizations, and then click **Submit**.
  - 1.3. Select **Add filter** from the **Actions** list for the **Compliance Auditor** role, select the **Audit** resource type. Select the **view\_audit\_logs** permission and then click **Submit**.
2. Create the **auditor** user, assign it the **Compliance Auditor** role, and associate it with the **Finance** and **Operations** organizations. Log in as the **auditor** user, and confirm that you have view access to all organizations and audit records.
  - 2.1. Navigate to **Administer** → **Users**, and then click **Create User**.
  - 2.2. On the **User** tab, enter **auditor** in the **Name** field and **root@satellite.lab.example.com** in the **Email** field. Select **INTERNAL** for the **Authorized by** field and specify **redhat** for the password.  
On the **Organizations** tab, select **Finance** and **Operations**. On the **Roles** tab, select **Compliance Auditor**. Click **Submit**.
- 2.3. Log out of the Satellite web UI, then log back in as the **auditor** user.

Navigate to **Monitor** → **Audits** and click on an audit log entry to ensure you can view the details.

Switch between organizations and note that you are unable to change the state of any objects such as content views or life-cycle environments. Menus and other controls that are normally present will not show up when permissions do not allow.

**3.** Clean up audit records older than one day.

- 3.1. On **workstation**, use **ssh** to log in to **satellite** as **student**. Use **sudo -i** to switch to **root**.

```
[student@workstation ~]$ ssh satellite
[student@satellite ~]$ sudo -i
[sudo] password for student: student
[root@satellite ~]#
```

- 3.2. Use the **hammer** command to count the number of audit log entries. The number of audit entries varies based on your Satellite usage.

```
[root@satellite ~]# hammer audit list | wc -l
142
```

- 3.3. Use the **foreman-rake** command to clean up audit records older than one day.

```
[root@satellite ~]# foreman-rake audits:expire days=1
Deleting audits older than 2019-11-19 06:00:22 UTC. This might take a few
minutes...
Successfully deleted 138 audits!
```

**4.** Reclaim unused space in the MongoDB Pulp database.

- 4.1. Stop all Pulp services.

```
[root@satellite ~]# systemctl stop goferd httpd pulp_workers pulp_celerybeat \
pulp_resource_manager pulp_streamer
Failed to stop goferd.service: Unit goferd.service not loaded.
```



**Note**

The **goferd** service will only be running if your Satellite is registered to itself as a host. In this classroom that is not the case, so the error above is safe to ignore.

- 4.2. Enter the MongoDB shell.

```
[root@satellite ~]# mongo pulp_database
MongoDB shell version v3.4.9
connecting to: mongodb://127.0.0.1:27017/pulp_database
MongoDB server version: 3.4.9
Welcome to the MongoDB shell.
For interactive help, type "help".
For more comprehensive documentation, see
http://docs.mongodb.org/
```

```
Questions? Try the support group  
http://groups.google.com/group/mongodb-user  
>
```

4.3. Determine the current index size.

```
> db.stats()  
{  
  "db" : "pulp_database",  
  "collections" : 47,  
  "views" : 0,  
  "objects" : 35124,  
  "avgObjSize" : 14300.054948183579,  
  "dataSize" : 502275130,  
  "storageSize" : 377286656,  
  "numExtents" : 0,  
  "indexes" : 161,  
  "indexSize" : 16715776,  
  "ok" : 1  
}
```

4.4. Reclaim unused space in the database.

```
> db.repairDatabase()  
{ "ok" : 1 }
```

4.5. Determine the new index size, then exit from the MongoDB shell.

```
> db.stats()  
{  
  "db" : "pulp_database",  
  "collections" : 47,  
  "views" : 0,  
  "objects" : 35124,  
  "avgObjSize" : 14300.054948183579,  
  "dataSize" : 502275130,  
  "storageSize" : 377286656,  
  "numExtents" : 0,  
  "indexes" : 161,  
  "indexSize" : 8937472,  
  "ok" : 1  
}  
> exit  
bye
```

4.6. Restart all Pulp services.

```
[root@satellite ~]# systemctl start goferd httpd pulp_workers pulp_celerybeat \  
pulp_resource_manager pulp_streamer  
Failed to start goferd.service: Unit not found.
```

5. Back up Satellite to **/var/tmp** but exclude Pulp content.

5.1. Determine the amount of disk space currently used by Satellite.

```
[root@satellite ~]# du -csh /var/lib/mongodb \
/var/lib/pgsql/data /var/lib/pulp /var/lib/qpidd \
/var/lib/tftpboot /etc /root/ssl-build /var/www/html/pub /opt/puppetlabs
661M /var/lib/mongodb
160M /var/lib/pgsql/data
4.0G /var/lib/pulp
31M /var/lib/qpidd
17M /var/lib/tftpboot
39M /etc
604K /root/ssl-build
116K /var/www/html/pub
178M /opt/puppetlabs
5.0G total
```

5.2. Back up the Satellite database to **/var/tmp** but exclude Pulp content.

```
[root@satellite ~]# satellite-maintain backup offline \
--skip-pulp-content --assumeyes /var/tmp/
Starting backup: 2019-12-03 02:34:09 +0000
Running preparation steps required to run the next scenarios
=====
Make sure Foreman DB is up:
/ Checking connection to the Foreman DB [OK]
-----

Running Backup
=====
Confirm turning off services is allowed:
WARNING: This script will stop your services.

[OK]
-----

Prepare backup Directory:
Creating backup folder /var/tmp/satellite-backup-2019-12-03-02-34-09 [OK]
-----
Check if the directory exists and is writable: [OK]
-----
Generate metadata:
/ Saving metadata to metadata.yml [OK]
-----
Detect features available in the local proxy: [OK]
-----
disable active sync plans:
- Total 2 sync plans are now disabled. [OK]
-----
Add maintenance_mode chain to iptables: [OK]
-----
Stop applicable services:
Stopping the following service(s):
```

```
rh-mongodb34-mongod, postgresql, qdrouterd, qpidd, squid, pulp_celerybeat,
pulp_resource_manager, pulp_streamer, pulp_workers, smart_proxy_dynflow_core,
tomcat, dynflowd, httpd, puppetserver, foreman-proxy
- All services stopped [OK]
-----
Backup config files:
/ Collecting config files to backup [OK]
-----
Backup Pulp data: [SKIPPED]
-----
Backup mongo offline:
Collecting Mongo data [OK]
Backup Candlepin DB offline:
/ Collecting data from /var/lib/pgsql/data/ [OK]
-----
Backup Foreman DB offline:
Already done [OK]
-----
Start applicable services:
Starting the following service(s):

rh-mongodb34-mongod, postgresql, qdrouterd, qpidd, squid, pulp_celerybeat,
pulp_resource_manager, pulp_streamer, pulp_workers, smart_proxy_dynflow_core,
tomcat, dynflowd, httpd, puppetserver, foreman-proxy
\ All services started [OK]
-----
re-enable sync plans:
\ Total 2 sync plans are now enabled. [OK]
-----
Remove maintenance_mode chain from iptables: [OK]
-----
Compress backup data to save space:
/ Compressing backup of Postgress DB
\ Compressing backup of Mongo DB [OK]
-----
Done with backup: 2019-12-03 02:38:36 +0000
**** BACKUP Complete, contents can be found in: /var/tmp/satellite-
backup-2019-12-03-02-34-09 ****
```

6. Ensure your organization is set to **Operations**, and ensure that the prerequisites for exporting the **Base v2.0** content view have all been met.
  - 6.1. Navigate to **Content** → **Content Views**, and then click **Base**.
  - 6.2. Click **Version 2.0**, and then click **Yum Content** → **Repositories**.
  - 6.3. Click each repository and ensure that the **Sync Settings** are correct.  
The **Mirror on Sync** setting should be **No**. The **Download Policy** setting should be **Immediate**.



### Note

This exercise skips synchronizing the repositories to save time.

7. Export the **Base 2.0** content view to the **/var/tmp** directory.

- 7.1. List the content views for the **Operations** organization. Note the ID for the **Base 2.0** content view.

```
[root@satellite ~]# hammer content-view version list \
--organization Operations
----|-----|-----|-----|-----|-----|
ID | NAME           | VERSION | LIFECYCLE ENVIRONMENTS
----|-----|-----|-----|-----|
5  | Base 2.0       | 2.0     | Library, Development, QA
4  | Base 1.0       | 1.0     |
2  | Default Organization View 1.0 | 1.0     | Library
----|-----|-----|-----|-----|
```

- 7.2. Export the Base 2.0 content view to the **/var/tmp** directory.

```
[root@satellite ~]# hammer content-view version export \
--export-dir /var/tmp/ --id 5
```

- 7.3. Determine the size of the content view archive, then log out of the Satellite host.

```
[root@satellite ~]# ls -lh /var/tmp/export*
-rw-r--r--. 1 root root 18G Dec  3 04:28 /var/tmp/export-Base-2.0.tar
[root@satellite ~]# logout
[student@satellite ~]$ logout
[student@workstation ~]$
```

This concludes the lab.

## Evaluation

As the **student** user on the **workstation** machine, use the **lab** command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab maintain-review grade
```

## Finish

As the **student** user on the **workstation** machine, use the **lab** command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab maintain-db finish
```

This concludes the lab.

# Summary

---

In this chapter, you learned:

- Custom roles can provide granular sets of permissions specific to your organization.
- It is a good practice to assign roles to user groups, and then manage the members of the group as required.
- Satellite includes the **satellite-maintain** utility which can back up the configuration, databases, and RPM packages of Satellite Server.
- There are several database maintenance tasks which should be performed regularly, including cleaning old audit records, cleaning unused tasks, and compressing MongoDB indexes.
- Satellite supports the import and export of content views between Satellite Servers, or organizations within the same Satellite.



# Comprehensive Review

### Goal

Verify and configure a Red Hat Satellite Server installation and provision content hosts.

### Objectives

- Review tasks from *Red Hat Satellite 6 Administration*

### Sections

- Course Objectives Listing

### Lab

- Lab: Configuring Satellite Server
- Lab: Installing and Configuring Satellite Capsule Server
- Lab: Provisioning a Host
- Lab: Performing Remote Execution
- Lab: Signing RPM Packages

# Course Objectives Listing

---

## Objectives

After completing this section, you should have reviewed and refreshed the knowledge and skills learned in *Red Hat Satellite 6 Administration*.

### Reviewing Red Hat Satellite 6 Administration

Before beginning the comprehensive review for this course, you should be comfortable with the topics covered in each chapter.

You can refer to earlier sections in the textbook for extra study.

### Chapter 1, *Planning and Deploying Red Hat Satellite*

Plan a Red Hat Satellite deployment, installation, and initial configuration of Red Hat Satellite servers.

- Describe the purpose, architecture, and components of Red Hat Satellite.
- Discuss planning a distributed Red Hat Satellite with Satellite Capsule Servers deployment, to meet multiple requirements and scenarios.
- Describe how to perform an initial installation of Red Hat Satellite.
- Describe and configure organizations in Red Hat Satellite, and create and install each organization's content manifests.

### Chapter 2, *Managing Software Life Cycles*

Create and manage Red Hat software deployment life-cycle environments.

- Enable repositories and create products in the Red Hat Satellite library, and configure sync plans to keep the library current.
- Define a workflow for software promotion by creating life-cycle environments and organizing them into an environment path.
- Create and publish content views and promote them to life-cycle environments on an environment path.

### Chapter 3, *Registering Hosts*

Register and configure your Red Hat Enterprise Linux systems to use Red Hat Satellite, and organize those systems into groups for easier management.

- Register a system with Red Hat Satellite and configure it to use a particular organization, location, subscription, and life-cycle environment.
- Organize registered hosts into host collections to manage them concurrently.
- Create and use activation keys to automatically register and configure a system for management by Satellite Server.

## **Chapter 4, Deploying Software to Hosts**

Manage software deployment to registered hosts of your Red Hat Satellite infrastructure and practice managing environment paths, life-cycle environments, and content views.

- Manage software installed on specific registered hosts using content views and life-cycle environments.
- Create and manage content view filters and composite content views to provide content subsets or supersets to managed hosts.
- Inspect, filter, and apply Red Hat errata to content views for precise patch management.
- Inspect available module streams, use Satellite to install them on hosts, and apply module stream errata.

## **Chapter 5, Deploying Custom Software**

Create, manage, and deploy custom software products and repositories.

- Create products and repositories for non-Red Hat content in Red Hat Satellite.
- Use the repository discovery feature to search URLs for multiple repositories and use them to create custom products and repositories.
- Update custom products and repositories and use content views to make them available to hosts.

## **Chapter 6, Deploying Satellite Capsule Servers**

Perform installation and initial configuration of Red Hat Satellite Capsule Servers as components of a deployment plan.

- Install an external Satellite Capsule Server to provide Satellite Server support to a distributed location.
- Configure a Satellite Capsule Server for a defined purpose by enabling content features, infrastructure management services, and host management services.
- Create and publish content views, and promote them to life-cycle environments on an environment path.

## **Chapter 7, Running Remote Execution**

Configure the ability to run ad hoc and scheduled tasks on managed hosts using a variety of configuration management tools.

- Prepare for remote execution by establishing a secure connection to hosts and creating job templates, and then run remote jobs and view job results.
- Configure and enable remote execution and install additional Ansible Roles on a Satellite Capsule Server.
- Create a content view that supports Puppet configuration and register a Puppet agent to a Red Hat Satellite server.

## **Chapter 8, Provisioning Hosts**

Configure Satellite Server for host deployment and perform host provisioning.

- Describe provisioning types and configure provisioning contexts and resources on Satellite Server and Satellite Capsule Servers.
- Describe networking requirements for different provisioning types, and configure network services to support provisioning.
- Deploy new hosts using the configured provisioning resources, network services, and installation parameters.

## **Chapter 9, *Managing Red Hat Satellite Using the API***

Integrate Red Hat Satellite functionality with custom scripts or external applications that access the API over HTTP

- Query the Red Hat Satellite REST API to observe Satellite functionality, syntax, and authentication methods.
- Integrate common Satellite tasks and object queries into custom scripts and external applications, including popular languages such as curl, Ruby, and Python.
- Describe the Hammer CLI syntax, and compare the Hammer and scripted API use cases.

## **Chapter 10, *Deploying Red Hat Satellite to a Cloud Platform***

Plan a Red Hat Satellite deployment on a cloud platform, including managed content hosts.

- Prepare for installing Red Hat Satellite Server and Capsules on selected cloud platforms.
- Manage content host cloud instances using Red Hat Satellite to interact with the cloud platform.

## **Chapter 11, *Performing Red Hat Satellite Server Maintenance***

Maintain Red Hat Satellite for security, recoverability, and growth.

- Create users and groups, and assign roles and permissions, to securely delegate Red Hat Satellite tasks.
- Perform backup and restore operations on Red Hat Satellite Servers and Satellite Capsule Servers, including databases and content stores.
- Describe the distributed architecture and management features of Red Hat Satellite databases.
- Perform maintenance tasks on a Red Hat Satellite Server database.
- Describe the package content export structure, and perform content exports and imports for content migration.

## ► Lab

# Configuring Satellite Server

In this review, you will perform initial configuration, organization and location creation, and life-cycle creation for an installed Satellite Server.

### Outcomes

You should be able to:

- Create an organization.
- Add a location to the organization.
- Create a life-cycle environment path.
- Create a content view.

### Before You Begin



#### Warning

Save any work you want to keep from earlier exercises, and reset your lab environment now.

To set up your computers for this exercise, log in to **workstation** as **student** and run the following command:

```
[student@workstation ~]$ lab configure-cr start
```

### Instructions

Accomplish the following tasks.

- Ensure that the **Finance** organization exists with the **San Francisco** location.
- Define an application life-cycle with a life-cycle environment path consisting of the **Development**, **QA**, and **Production** environments. In this application life-cycle, the content is developed in the **Development** environment, is tested in **QA**, and is delivered to the users via **Production**. In this application life-cycle, the **QA** environment should follow **Development**, and the **Production** environment should follow **QA**.
- Define a content view called **Base**.

### Evaluation

As the **student** user on the **workstation** machine, use the **lab** command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab configure-cr grade
```

## Finish

As the **student** user on the **workstation** machine, use the **lab** command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab configure-cr finish
```

This concludes the lab.

## ► Solution

# Configuring Satellite Server

In this review, you will perform initial configuration, organization and location creation, and life-cycle creation for an installed Satellite Server.

### Outcomes

You should be able to:

- Create an organization.
- Add a location to the organization.
- Create a life-cycle environment path.
- Create a content view.

### Before You Begin



#### Warning

Save any work you want to keep from earlier exercises, and reset your lab environment now.

To set up your computers for this exercise, log in to **workstation** as **student** and run the following command:

```
[student@workstation ~]$ lab configure-cr start
```

### Instructions

Accomplish the following tasks.

- Ensure that the **Finance** organization exists with the **San Francisco** location.
- Define an application life-cycle with a life-cycle environment path consisting of the **Development**, **QA**, and **Production** environments. In this application life-cycle, the content is developed in the **Development** environment, is tested in **QA**, and is delivered to the users via **Production**. In this application life-cycle, the **QA** environment should follow **Development**, and the **Production** environment should follow **QA**.
- Define a content view called **Base**.

#### 1. Create the **Finance** organization.

- 1.1. On **workstation**, launch Firefox and navigate to `https://satellite.lab.example.com` to access the Satellite Server web UI.
- 1.2. Log in to the web UI as **admin** using **redhat** as the password.
- 1.3. Click **Administer** → **Organizations** to access the **Organizations** page.

- 1.4. Click **New Organization** to open the **New Organization** page.
- 1.5. Enter **Finance** in the **Name** field. Enter **Finance** in the **Label** field.
- 1.6. Click **Submit** to create the organization.
2. Add the **San Francisco** location to the **Finance** organization.
  - 2.1. From the Satellite Server web UI, click **Administer** → **Locations** to access the **Locations** page.
  - 2.2. Click **New Location** to open the **New Location** page.
  - 2.3. Enter **San Francisco** in the **Name** field.
  - 2.4. Click **Submit** to create the location. The **Edit San Francisco** page displays.
  - 2.5. From the **Edit San Francisco** page, click **Organizations**.
  - 2.6. From the **Select organizations** window, click **Finance** under **All items** to mark it as the selected organization. The **Finance** organization displays as the selected organization under **Selected items**.
  - 2.7. Click **Submit** to associate the **Finance** organization with the **San Francisco** location.
3. Create a life-cycle environment called **Development**, and then extend it with **QA** and **Production** environments. The **Development** environment should precede **QA**, and the **QA** environment should precede **Production**.
  - 3.1. From the Satellite Server web UI, change to the **Finance** organization and the **Any Location** context.
  - 3.2. Click **Content** → **Lifecycle Environments** to access the **Lifecycle Environment Paths** page.
  - 3.3. Click **Create Environment Path** to open the **New Environment** page.
  - 3.4. Enter **Development** in the **Name** field.
  - 3.5. Click **Save** to create the life-cycle environment.
  - 3.6. Click **Add New Environment** to extend the **Development** environment with a new environment called **QA**. The **New Environment** page displays.
  - 3.7. Enter **QA** in the **Name** field. Select **Development** in the **Prior Environment** field.
  - 3.8. Click **Save** to create the life-cycle environment.
4. Add a new life-cycle environment called **Production** to the same environment path as **Development** and **QA**.
  - 4.1. From the **Lifecycle Environment Paths** page, click **Add New Environment** to extend the **QA** environment with a new environment called **Production**. The **New Environment** page displays.
  - 4.2. Enter **Production** in the **Name** field. Select **QA** in the **Prior Environment** field.

- 4.3. Click **Save** to create the life-cycle environment.
5. Create a content view called **Base**.
  - 5.1. Click **Content** → **Content Views** to access the **Content Views** page.
  - 5.2. Click **Create New View** to open the **New Content View** page.
  - 5.3. Enter **Base** in the **Name** field.
  - 5.4. Click **Save** to create the content view.

This concludes the lab.

## Evaluation

As the **student** user on the **workstation** machine, use the **lab** command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab configure-cr grade
```

## Finish

As the **student** user on the **workstation** machine, use the **lab** command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab configure-cr finish
```

## ► Lab

# Installing and Configuring Satellite Capsule Server

In this review, you will install and perform initial configuration for an external Satellite Capsule Server.

## Outcomes

You should be able to:

- Configure Satellite Server to provide the resources required by Satellite Capsule Server.
- Prepare a system for installing Satellite Capsule Server.
- Deploy Satellite Capsule Server.

## Before You Begin

As the **student** user on the **workstation** machine, use the **lab** command to prepare your system for this exercise.

This command determines if the **satellite.lab.example.com** and **capsule.lab.example.com** hosts are reachable on the network and verifies that the required resources exist.

```
[student@workstation ~]$ lab capsule-cr start
```

## Instructions

Perform the following tasks to complete the comprehensive review:

- Configure the **Finance** organization and the **San Francisco** location to use the offline CDN available at [http://content.example.com/rhs6.6/x86\\_64/cdn](http://content.example.com/rhs6.6/x86_64/cdn), and the manifest available at [http://materials.example.com/manifest\\_finance.zip](http://materials.example.com/manifest_finance.zip).
- Enable and synchronize the repositories that the Satellite Capsule Server installation requires:
  - **Red Hat Enterprise Linux 7 Server (RPMs)**
  - **Red Hat Satellite Maintenance 6 (for RHEL 7 Server) (RPMs)**
  - **Red Hat Ansible Engine 2.8 RPMs for Red Hat Enterprise Linux 7 Server**
  - **Red Hat Software Collections RPMs for Red Hat Enterprise Linux 7 Server**
  - **Red Hat Satellite Capsule 6.6 (for RHEL 7 Server) (RPMs)**
- Create the **CapsuleServers** activation key. Ensure that the activation key is configured to be able to register at least three hosts. Associate the activation key to the **Library** life-cycle environment and the **Default Organization View** content view. Attach all the available subscriptions to the activation key.



### Warning

The content must be fully synchronized before you can continue with the next steps. If the repositories are still synchronizing, then wait for them to finish.

- Register the **capsule.lab.example.com** system with Satellite Server, in the **Finance** organization, using the **CapsuleServers** activation key. Subscribe the **capsule.lab.example.com** system to all the repositories you previously enabled.
- Configure the firewall rules on the **capsule.lab.example.com** system.
- On Satellite Server, generate a certificate for **capsule.lab.example.com**.
- Deploy Satellite Capsule Server on **capsule.lab.example.com**.
- Set Satellite Capsule Server organization to **Finance** and location to **San Francisco**.

## Evaluation

As the **student** user on the **workstation** machine, use the **lab** command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab capsule-cr grade
```

## Finish

As the **student** user on the **workstation** machine, use the **lab** command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab capsule-cr finish
```

This concludes the lab.

## ► Solution

# Installing and Configuring Satellite Capsule Server

In this review, you will install and perform initial configuration for an external Satellite Capsule Server.

## Outcomes

You should be able to:

- Configure Satellite Server to provide the resources required by Satellite Capsule Server.
- Prepare a system for installing Satellite Capsule Server.
- Deploy Satellite Capsule Server.

## Before You Begin

As the **student** user on the **workstation** machine, use the **lab** command to prepare your system for this exercise.

This command determines if the **satellite.lab.example.com** and **capsule.lab.example.com** hosts are reachable on the network and verifies that the required resources exist.

```
[student@workstation ~]$ lab capsule-cr start
```

## Instructions

Perform the following tasks to complete the comprehensive review:

- Configure the **Finance** organization and the **San Francisco** location to use the offline CDN available at [http://content.example.com/rhs6.6/x86\\_64/cdn](http://content.example.com/rhs6.6/x86_64/cdn), and the manifest available at [http://materials.example.com/manifest\\_finance.zip](http://materials.example.com/manifest_finance.zip).
- Enable and synchronize the repositories that the Satellite Capsule Server installation requires:
  - **Red Hat Enterprise Linux 7 Server (RPMs)**
  - **Red Hat Satellite Maintenance 6 (for RHEL 7 Server) (RPMs)**
  - **Red Hat Ansible Engine 2.8 RPMs for Red Hat Enterprise Linux 7 Server**
  - **Red Hat Software Collections RPMs for Red Hat Enterprise Linux 7 Server**
  - **Red Hat Satellite Capsule 6.6 (for RHEL 7 Server) (RPMs)**
- Create the **CapsuleServers** activation key. Ensure that the activation key is configured to be able to register at least three hosts. Associate the activation key to the **Library** life-cycle environment and the **Default Organization View** content view. Attach all the available subscriptions to the activation key.

**Warning**

The content must be fully synchronized before you can continue with the next steps. If the repositories are still synchronizing, then wait for them to finish.

- Register the **capsule.lab.example.com** system with Satellite Server, in the **Finance** organization, using the **CapsuleServers** activation key. Subscribe the **capsule.lab.example.com** system to all the repositories you previously enabled.
- Configure the firewall rules on the **capsule.lab.example.com** system.
- On Satellite Server, generate a certificate for **capsule.lab.example.com**.
- Deploy Satellite Capsule Server on **capsule.lab.example.com**.
- Set Satellite Capsule Server organization to **Finance** and location to **San Francisco**.

1. Configure the **Finance** organization to use the offline CDN available at `http://content.example.com/rhs6.6/x86_64/cdn`, and the manifest available at `http://materials.example.com/manifest_finance.zip`.
  - 1.1. Log in to the Satellite web UI. To do so, use your browser to navigate to `https://satellite.lab.example.com`.
  - 1.2. Log in as **admin** using **redhat** as a password.
  - 1.3. Choose the **Finance** organization, and the **San Francisco** location from the main menu.
  - 1.4. Click **Content** → **Subscriptions**, and then click **Import a Manifest**.
  - 1.5. Update the **Red Hat CDN URL** field to `http://content.example.com/rhs6.6/x86_64/cdn`, and then click **Update**.
  - 1.6. Open a terminal, and then download the manifest available at `http://materials.example.com/manifest_finance.zip`.

```
[student@workstation ~]$ wget \
http://materials.example.com/manifest_finance.zip
...output omitted...
```

- 1.7. Return to the Satellite web UI, click **Browse**, and then select the **manifest\_finance.zip** file from your `/home/student/` directory.
- 1.8. Verify that the subscriptions attached to the manifest are displayed from **Content** → **Subscriptions**.
2. Enable and synchronize the following repositories:
  - **Red Hat Enterprise Linux 7 Server (RPMs)**
  - **Red Hat Satellite Maintenance 6 (for RHEL 7 Server) (RPMs)**
  - **Red Hat Ansible Engine 2.8 RPMs for Red Hat Enterprise Linux 7 Server**

- Red Hat Software Collections RPMs for Red Hat Enterprise Linux 7 Server
- Red Hat Satellite Capsule 6.6 (for RHEL 7 Server) (RPMs)

- 2.1. Click **Content** → **Red Hat Repositories** to access the **Red Hat Repositories** page.
- 2.2. Toggle **Recommended Repositories** to **ON** to list only the recommended repositories.
- 2.3. Find and expand the repositories from the preceding list, and then click the plus sign (+) for each of them.
- 2.4. Click **Content** → **Products** to open the **Products** page.
- 2.5. Select the check boxes for all the products, and then click **Select Action** → **Sync Selected**. Do not wait for the synchronization to complete; continue the exercise.
3. Create the **CapsuleServers** activation key.
  - 3.1. Click **Content** → **Activation Keys**, and then click **Create Activation Key**.
  - 3.2. Enter **CapsuleServers** in the **Name** field.
  - 3.3. Clear the check box for **Unlimited Hosts**, and then set the limit to **3**.
  - 3.4. Select the **Library** environment.
  - 3.5. Select the **Default Organization View** content view, and then click **Save**.
  - 3.6. On the **Activation Keys** → **CapsuleServers** page, click the **Subscriptions** tab, and then click the **Add** tab.
  - 3.7. Select the check boxes for all the subscriptions, and then click **Add Selected**.
4. Register the **capsule.lab.example.com** machine with Satellite Server.



### Warning

The content must be fully synchronized before you can continue with the next steps. If the repositories are still synchronizing, then wait for them to finish.

- 4.1. On **workstation**, ssh to **capsule** as **student**. Use **sudo -i** to switch to **root**.

```
[student@workstation ~]$ ssh student@capsule
[student@capsule ~]$ sudo -i
[sudo] password for student: student
[root@capsule ~]#
```

- 4.2. Install the consumer RPM, *katello-ca-consumer-latest.noarch.rpm*.

```
[root@capsule ~]# yum localinstall \
http://satellite.lab.example.com/pub/katello-ca-consumer-latest.noarch.rpm
```

- 4.3. Register the system to the **Finance** organization. Reference the **CapsuleServers** activation key with the **--activationkey** option.

```
[root@capsule ~]# subscription-manager register --org=Finance \
--activationkey=CapsuleServers
The system has been registered with ID: 95424d51-69a1-4142-ad18-4453dc96364e
The registered system name is: capsule.lab.example.com
Installed Product Current Status:
Product Name: Red Hat Enterprise Linux Server
Status: Subscribed
```

5. Disable all repositories on **capsule**, and then enable only the repositories that the Satellite Capsule Server installation requires.

- 5.1. Disable all the repositories.

```
[root@capsule ~]# subscription-manager repos --disable "*"
```

- 5.2. List all the available repositories.

```
[root@capsule ~]# subscription-manager repos --list
-----
Available Repositories in /etc/yum.repos.d/redhat.repo
-----
Repo ID: rhel-7-server-rpms
Repo Name: Red Hat Enterprise Linux 7 Server (RPMS)
Repo URL: https://satellite.lab.example.com/pulp/repos/...
Enabled: 0

Repo ID: rhel-7-server-satellite-capsule-6.6-rpms
Repo Name: Red Hat Satellite Capsule 6.6 (for RHEL 7 Server) (RPMS)
Repo URL: https://satellite.lab.example.com/pulp/repos/...
Enabled: 0

Repo ID: rhel-server-rhscl-7-rpms
Repo Name: Red Hat Software Collections RPMS for Red Hat Enterprise Linux 7 Server
Repo URL: https://satellite.lab.example.com/pulp/repos/...
Enabled: 0

Repo ID: rhel-7-server-ansible-2.8-rpms
Repo Name: Red Hat Ansible Engine 2.8 RPMS for Red Hat Enterprise Linux 7 Server
Repo URL: https://satellite.lab.example.com/pulp/repos/...
Enabled: 0

Repo ID: rhel-7-server-satellite-maintenance-6-rpms
Repo Name: Red Hat Satellite Maintenance 6 (for RHEL 7 Server) (RPMS)
Repo URL: https://satellite.lab.example.com/pulp/repos/...
Enabled: 0

Repo ID: rhel-7-server-satellite-tools-6.6-rpms
Repo Name: Red Hat Satellite Tools 6.6 (for RHEL 7 Server) (RPMS)
Repo URL: https://satellite.lab.example.com/pulp/repos/...
Enabled: 0
```

- 5.3. Enable the repositories that the Satellite Capsule Server installation requires.

```
[root@capsule ~]# subscription-manager repos \
--enable=rhel-7-server-rpms \
--enable=rhel-7-server-satellite-capsule-6.6-rpms \
--enable=rhel-server-rhscl-7-rpms \
--enable=rhel-7-server-satellite-maintenance-6-rpms \
--enable=rhel-7-server-ansible-2.8-rpms
Repository 'rhel-7-server-rpms' is enabled for this system.
Repository 'rhel-7-server-satellite-capsule-6.6-rpms' is enabled for this system.
Repository 'rhel-7-server-satellite-maintenance-6-rpms' is enabled for this
system.
Repository 'rhel-7-server-ansible-2.8-rpms' is enabled for this system.
Repository 'rhel-server-rhscl-7-rpms' is enabled for this system.
```

6. Update the **capsule** system, and then install the *satellite-capsule* package.

6.1. Update the system.

```
[root@capsule ~]# yum update
```

6.2. Install the *satellite-capsule* package and its dependencies.

```
[root@capsule ~]# yum install satellite-capsule
```

7. Configure the firewall rules on the **capsule** system to allow communications from hosts and Satellite Server.

7.1. Use the **firewall-cmd** command to configure the firewall.

```
[root@capsule ~]# firewall-cmd --add-port="53/udp" --add-port="53/tcp" \
--add-port="67/udp" --add-port="69/udp" \
--add-port="80/tcp" --add-port="443/tcp" \
--add-port="5000/tcp" --add-port="5647/tcp" \
--add-port="8000/tcp" --add-port="8140/tcp" \
--add-port="8443/tcp" --add-port="9090/tcp"
success
```

7.2. Make the configuration persistent.

```
[root@capsule ~]# firewall-cmd --runtime-to-permanent
success
```

8. On Satellite Server, create a directory to store the Satellite Capsule Server certificate, and then generate that certificate. When done, use the **scp** command to copy the certificate archive to the **capsule** machine.

8.1. Create the **/root/capsule\_cert/** directory to store the certificate.

```
[root@satellite ~]# mkdir /root/capsule_cert
```

8.2. Use the **capsule-certs-generate** command to generate a certificate for **capsule.lab.example.com**.

```
[root@satellite ~]# capsule-certs-generate \
--foreman-proxy-fqdn capsule.lab.example.com \
--certs-tar /root/capsule_cert/capsule_certs.tar
Installing      Done      [100%]      [.....]
Success!
...output omitted...
1. Ensure that the satellite-capsule package is installed on the system.
2. Copy the following file /root/capsule_cert/capsule_certs.tar to the
   system capsule.lab.example.com at the following location
   /root/capsule_certs.tar
   scp /root/capsule_cert/capsule_certs.tar root@capsule.lab.example.com:/root/
capsule_certs.tar
3. Run the following commands on the Capsule (possibly with the customized
   parameters, see satellite-installer --scenario capsule --help and
   documentation for more info on setting up additional services):

satellite-installer \
--scenario capsule \
--certs-tar-file          "/root/capsule_certs.tar"\ \
--foreman-proxy-content-parent-fqdn "satellite.lab.example.com"\ \
--foreman-proxy-register-in-foreman "true"\ \
--foreman-proxy-foreman-base-url  "https://satellite.lab.example.com"\ \
--foreman-proxy-trusted-hosts    "satellite.lab.example.com"\ \
--foreman-proxy-trusted-hosts    "capsule.lab.example.com"\ \
--foreman-proxy-oauth-consumer-key "mf6vLfwWpAE4fkMcTS3xx5AQMU2oQ4Cf"\ \
--foreman-proxy-oauth-consumer-secret "YP64QCHRBCUtaFTG5L8s7bevgTixBegL"\ \
--puppet-server-foreman-url     "https://satellite.lab.example.com"
```

- 8.3. Copy the archive to the **capsule** machine. Copy and paste the **scp** command from the preceding output.

```
[root@satellite ~]# scp /root/capsule_cert/capsule_certs.tar \
root@capsule.lab.example.com:/root/capsule_certs.tar
root@capsule.lab.example.com's password: redhat
capsule_certs.tar          100%   61KB   3.3MB/S   00:00
```

## 9. Install Satellite Capsule Server on **capsule.lab.example.com**.

- 9.1. From the **capsule** machine command line, run the **satellite-installer** command. Copy and paste that command from the output of the **capsule-certs-generate** command you ran in the preceding step.

```
[root@capsule ~]# satellite-installer \
--scenario capsule \
--certs-tar-file          "/root/capsule_certs.tar"\ \
--foreman-proxy-content-parent-fqdn "satellite.lab.example.com"\ \
--foreman-proxy-register-in-foreman "true"\ \
--foreman-proxy-foreman-base-url  "https://satellite.lab.example.com"\ \
--foreman-proxy-trusted-hosts    "satellite.lab.example.com"\ \
--foreman-proxy-trusted-hosts    "capsule.lab.example.com"\ \
--foreman-proxy-oauth-consumer-key "mf6vLfwWpAE4fkMcTS3xx5AQMU2oQ4Cf"\ \
--foreman-proxy-oauth-consumer-secret "YP64QCHRBCUtaFTG5L8s7bevgTixBegL"\ \
--puppet-server-foreman-url     "https://satellite.lab.example.com"
```

```
Installing      Done      [100%]      [.....]
Success!
* Capsule is running at https://capsule.lab.example.com:9090
The full log is at /var/log/foreman-installer/capsule.log
```

10. Use the Satellite Server web UI to confirm the success of the installation, and then set Satellite Capsule Server organization to **Finance** and location to **San Francisco**.
- 10.1. Choose the **Any Organization** organization and **Any Location** location from the main menu.
  - 10.2. Click **Infrastructure → Capsules** to access the **Capsules** page, and then click the **capsule.lab.example.com** link.
  - 10.3. Confirm that the **Communication status** mark is green, which means Satellite Server can communicate with Satellite Capsule Server.
  - 10.4. Click **Edit**, and then click the **Locations** tab. Click **San Francisco** to move it to the **Selected items** list.
  - 10.5. Click the **Organizations** tab. Click **Finance** to move it to the **Selected items** list.
  - 10.6. Click **Submit**

This concludes the lab.

## Evaluation

As the **student** user on the **workstation** machine, use the **lab** command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab capsule-cr grade
```

## Finish

As the **student** user on the **workstation** machine, use the **lab** command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab capsule-cr finish
```

## ▶ Lab

# Provisioning a Host

In this review, you will configure provisioning features and services on a Satellite Server, and deploy and provision a new host.

### Outcomes

You should be able to:

- Configure Satellite Capsule Server to provide DNS, DHCP, and TFTP services.
- Prepare resources for provisioning using the Satellite web UI.
- Perform PXE provisioning of a new system.

### Before You Begin

This exercise requires a running Satellite Capsule Server on the **capsule.lab.example.com** host, which you installed in a preceding exercise.

As the **student** user on the **workstation** machine, use the **lab** command to prepare your system for this exercise.

This command determines if the **satellite** and **capsule** hosts are reachable on the network and verifies that the required resources exist.

```
[student@workstation ~]$ lab host-cr start
```

### Instructions

Perform the following tasks to complete the comprehensive review:

- Enable and synchronize the repositories required to install new Red Hat Enterprise Linux 8 systems in the **Finance** organization, in the **San Francisco** location:
  - **Red Hat Enterprise Linux 8 for x86\_64 - AppStream (RPMS)**
  - **Red Hat Enterprise Linux 8 for x86\_64 - BaseOS (RPMS)**
  - **Red Hat Satellite Tools 6.6 for RHEL 8 x86\_64 (RPMS)**
  - **Red Hat Enterprise Linux 8 for x86\_64 - AppStream (Kickstart)**
  - **Red Hat Enterprise Linux 8 for x86\_64 - BaseOS (Kickstart)**
- Add the preceding repositories to the **Base** content view, and then publish and promote the content view to the **Development** life-cycle.
- Associate and synchronize the **Development** life-cycle with Satellite Capsule Server.
- Enable the DNS, DHCP, and TFTP features on Satellite Capsule Server.

Satellite Capsule Server must manage the **sf.lab.example.com** and **250.25.172.in-addr.arpa** DNS zones, and forward all other requests to the DNS server at **172.25.250.254**.

The DHCP service must provide IP addresses in the **172.25.250.70** to **172.25.250.90** range. It must provide to hosts the **172.25.250.254** IP address as the network gateway, and the IP address of **capsule.lab.example.com (172.25.250.16)** as the DNS server.

The network interface name of the `capsule.lab.example.com` system is `eth0`.

- In Satellite Server, create the `sf.lab.example.com` domain resource.
- Create a subnet resource named **San Francisco Data Center** for the **172.25.250.0/24** network that Satellite Capsule Server manages. The associated domain is `sf.lab.example.com`.
- Create the **FinanceServers** activation key. Set the release version to 8.1, and associate the activation key to the **Development** life-cycle environment and the **Base** content view. Attach all the available subscriptions to the activation key. Make sure that the Red Hat Satellite Tools repository is enabled by default on new systems.
- Create the **Finance Host Group** host group with the following details:

#### Host Group Details

| Field                      | Value                                |
|----------------------------|--------------------------------------|
| Name                       | <b>Finance Host Group</b>            |
| Life-cycle environment     | <b>Development</b>                   |
| Content view               | <b>Base</b>                          |
| Capsule Server for content | <code>capsule.lab.example.com</code> |
| DNS domain                 | <code>sf.lab.example.com</code>      |
| Subnet                     | <b>San Francisco Data Center</b>     |
| Operating system           | <b>RedHat 8.1</b>                    |
| Partition table template   | <b>Kickstart default</b>             |
| Activation key             | <b>FinanceServers</b>                |

- Create a host resource for the system to provision, `servere`. Use the **Finance Host Group** host group as a base. Set the `root` password to `redhat123`. `servere` MAC address is **52:54:00:00:fa:0e**.
- Reboot and provision the `servere` machine.

## Evaluation

As the **student** user on the **workstation** machine, use the **lab** command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab host-cr grade
```

## Finish

As the **student** user on the **workstation** machine, use the **lab** command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab host-cr finish
```

This concludes the lab.

## ► Solution

# Provisioning a Host

In this review, you will configure provisioning features and services on a Satellite Server, and deploy and provision a new host.

## Outcomes

You should be able to:

- Configure Satellite Capsule Server to provide DNS, DHCP, and TFTP services.
- Prepare resources for provisioning using the Satellite web UI.
- Perform PXE provisioning of a new system.

## Before You Begin

This exercise requires a running Satellite Capsule Server on the **capsule.lab.example.com** host, which you installed in a preceding exercise.

As the **student** user on the **workstation** machine, use the **lab** command to prepare your system for this exercise.

This command determines if the **satellite** and **capsule** hosts are reachable on the network and verifies that the required resources exist.

```
[student@workstation ~]$ lab host-cr start
```

## Instructions

Perform the following tasks to complete the comprehensive review:

- Enable and synchronize the repositories required to install new Red Hat Enterprise Linux 8 systems in the **Finance** organization, in the **San Francisco** location:
  - **Red Hat Enterprise Linux 8 for x86\_64 - AppStream (RPMS)**
  - **Red Hat Enterprise Linux 8 for x86\_64 - BaseOS (RPMS)**
  - **Red Hat Satellite Tools 6.6 for RHEL 8 x86\_64 (RPMS)**
  - **Red Hat Enterprise Linux 8 for x86\_64 - AppStream (Kickstart)**
  - **Red Hat Enterprise Linux 8 for x86\_64 - BaseOS (Kickstart)**
- Add the preceding repositories to the **Base** content view, and then publish and promote the content view to the **Development** life-cycle.
- Associate and synchronize the **Development** life-cycle with Satellite Capsule Server.
- Enable the DNS, DHCP, and TFTP features on Satellite Capsule Server.

Satellite Capsule Server must manage the **sf.lab.example.com** and **250.25.172.in-addr.arpa** DNS zones, and forward all other requests to the DNS server at **172.25.250.254**.

The DHCP service must provide IP addresses in the **172.25.250.70** to **172.25.250.90** range. It must provide to hosts the **172.25.250.254** IP address as the network gateway, and the IP address of **capsule.lab.example.com (172.25.250.16)** as the DNS server.

The network interface name of the `capsule.lab.example.com` system is `eth0`.

- In Satellite Server, create the `sf.lab.example.com` domain resource.
- Create a subnet resource named **San Francisco Data Center** for the **172.25.250.0/24** network that Satellite Capsule Server manages. The associated domain is `sf.lab.example.com`.
- Create the **FinanceServers** activation key. Set the release version to 8.1, and associate the activation key to the **Development** life-cycle environment and the **Base** content view. Attach all the available subscriptions to the activation key. Make sure that the Red Hat Satellite Tools repository is enabled by default on new systems.
- Create the **Finance Host Group** host group with the following details:

#### Host Group Details

| Field                      | Value                                |
|----------------------------|--------------------------------------|
| Name                       | <b>Finance Host Group</b>            |
| Life-cycle environment     | <b>Development</b>                   |
| Content view               | <b>Base</b>                          |
| Capsule Server for content | <code>capsule.lab.example.com</code> |
| DNS domain                 | <code>sf.lab.example.com</code>      |
| Subnet                     | <b>San Francisco Data Center</b>     |
| Operating system           | <b>RedHat 8.1</b>                    |
| Partition table template   | <b>Kickstart default</b>             |
| Activation key             | <b>FinanceServers</b>                |

- Create a host resource for the system to provision, `servere`. Use the **Finance Host Group** host group as a base. Set the `root` password to `redhat123`. `servere` MAC address is **52:54:00:00:fa:0e**.
- Reboot and provision the `servere` machine.

1. Enable and synchronize the following repositories:

- **Red Hat Enterprise Linux 8 for x86\_64 - AppStream (RPMs)**
- **Red Hat Enterprise Linux 8 for x86\_64 - BaseOS (RPMs)**
- **Red Hat Satellite Tools 6.6 for RHEL 8 x86\_64 (RPMs)**
- **Red Hat Enterprise Linux 8 for x86\_64 - AppStream (Kickstart)**
- **Red Hat Enterprise Linux 8 for x86\_64 - BaseOS (Kickstart)**

- 1.1. Log in to the Satellite web UI. To do so, use your browser to navigate to `https://satellite.lab.example.com`.

- 1.2. Log in as **admin** using **redhat** as a password.
  - 1.3. Choose the **Finance** organization and **San Francisco** location from the main menu.
  - 1.4. Click **Content → Red Hat Repositories** to access the **Red Hat Repositories** page.
  - 1.5. Toggle **Recommended Repositories** to **ON** to list only the recommended repositories.
  - 1.6. Find and expand the repositories from the preceding list, and then click the plus sign (+) for each of them. To find the two kickstart repositories, in the **RPM** list, select **Kickstart** and clear the check mark from **RPM**.
  - 1.7. Click **Content → Products** to open the **Products** page. Select the **Red Hat Enterprise Linux for x86\_64** product, and then click **Select Action → Sync Selected**.
  - 1.8. Click **Content → Sync Status**, and then click **Expand All**. Wait for all the repositories to obtain a result status of **Syncing Complete**. This operation can take up to 20 minutes to complete.
2. Add the repositories to the **Base** content view. When done, publish and promote the content view to the **Development** life-cycle environment.
    - 2.1. Click **Content → Content Views** to access the **Content Views** page. Click the **Base** content view.
    - 2.2. Click **Yum Content → Repositories**, and then click the **Add** tab. Select the check boxes next to the following repositories:
      - **Red Hat Enterprise Linux 8 for x86\_64 - AppStream RPMs x86\_64 8.1**
      - **Red Hat Enterprise Linux 8 for x86\_64 - BaseOS RPMs x86\_64 8.1**
      - **Red Hat Satellite Tools 6.6 for RHEL 8 x86\_64 RPMs x86\_64**
      - **Red Hat Enterprise Linux 8 for x86\_64 - AppStream Kickstart x86\_64 8.1**
      - **Red Hat Enterprise Linux 8 for x86\_64 - BaseOS Kickstart x86\_64 8.1**
    - 2.3. Click **Add Repositories**.
    - 2.4. Click **Publish New Version**, and then click **Save**. Wait for the publishing process to complete. The task can take up to 15 minutes to finish.
  3. Synchronize the **Development** life-cycle environment with Satellite Capsule Server.
    - 3.1. Click **Infrastructure → Capsules** to access the **Capsules** page. Click **Edit** at the end of the **capsule.lab.example.com** row.
    - 3.2. Click the **Lifecycle Environments** tab, and then click **Development** to add it to the **Selected items** list.

Click **Submit**.

- 3.3. On the **Capsules** page, click the **capsule.lab.example.com** link. Click **Synchronize** → **Optimized Sync** to start a new sync. Wait for the sync to complete.
4. Use the **satellite-installer** command on the **capsule.lab.example.com** machine to enable and configure the DNS, DHCP, and TFTP services.
- 4.1. On **workstation**, **ssh** to **capsule** as **student**. Use **sudo -i** to switch to **root**.

```
[student@workstation ~]$ ssh student@capsule
[student@capsule ~]$ sudo -i
[root@capsule ~]#
```

- 4.2. Use the **satellite-installer** command to enable the DNS, DHCP, and TFTP services. The following table details the configuration parameters.

#### Service Parameters

| Parameter                    | Value                                 |
|------------------------------|---------------------------------------|
| DNS network interface        | <b>eth0</b>                           |
| DNS forwarder                | <b>172.25.250.254</b>                 |
| DNS zone                     | <b>sf.lab.example.com</b>             |
| DNS reverse                  | <b>250.25.172.in-addr.arpa</b>        |
| DHCP network interface       | <b>eth0</b>                           |
| DHCP range                   | <b>172.25.250.70 to 172.25.250.90</b> |
| Name server provided by DHCP | <b>172.25.250.16</b>                  |
| Gateway provided by DHCP     | <b>172.25.250.254</b>                 |

For your convenience, you can copy and paste the following command from the **/root/satellite-installer-review.txt** file.

```
[root@capsule ~]# satellite-installer --scenario capsule \
--foreman-proxy-dns true \
--foreman-proxy-dns-interface eth0 \
--foreman-proxy-dns-forwarders 172.25.250.254 \
--foreman-proxy-dns-zone sf.lab.example.com \
--foreman-proxy-dns-reverse 250.25.172.in-addr.arpa \
--foreman-proxy-dhcp true \
--foreman-proxy-dhcp-interface eth0 \
--foreman-proxy-dhcp-range "172.25.250.70 172.25.250.90" \
--foreman-proxy-dhcp-nameservers 172.25.250.16 \
--foreman-proxy-dhcp-gateway 172.25.250.254 \
--foreman-proxy-tftp true
Installing      Done    [100%] [.....]
Success!
* Capsule is running at https://capsule.lab.example.com:9090
The full log is at /var/log/foreman-installer/capsule.log
```

5. Create the **sf.lab.example.com** DNS domain and make it available for use by the **Finance** organization at the **San Francisco** location. This is the domain that Satellite Capsule Server manages.
  - 5.1. Click **Infrastructure → Domains**, and then click **Create Domain**.
  - 5.2. Enter **sf.lab.example.com** for the **DNS Domain** field.
  - 5.3. Choose **capsule.lab.example.com** as the **DNS Capsule**.
  - 5.4. Click the **Locations** tab and verify that the **San Francisco** location is already associated with the new domain. If not, click **San Francisco** to add it to the **Selected items** list.
  - 5.5. Click the **Organizations** tab and verify that the **Finance** organization is already associated with the new domain. If not, click **Finance** to add it to the **Selected items** list.
  - 5.6. Click **Submit**.
6. Create the new **San Francisco Data Center** subnet, **172.25.250.0/24**, as follows:

#### San Francisco Subnet Details

| Field                     | Value                                               |
|---------------------------|-----------------------------------------------------|
| <b>Name</b>               | <b>San Francisco Data Center</b>                    |
| <b>Network Address</b>    | <b>172.25.250.0</b>                                 |
| <b>Network Prefix</b>     | <b>24</b>                                           |
| <b>Gateway Address</b>    | <b>172.25.250.254</b>                               |
| <b>Primary DNS Server</b> | <b>172.25.250.16</b>                                |
| <b>IPAM</b>               | <b>DHCP</b>                                         |
| <b>Start Of Ip Range</b>  | <b>172.25.250.70</b>                                |
| <b>End Of Ip Range</b>    | <b>172.25.250.90</b>                                |
| <b>Boot Mode</b>          | <b>DHCP</b>                                         |
| <b>Domains</b>            | <b>sf.lab.example.com</b>                           |
| <b>Capsules</b>           | <b>capsule.lab.example.com</b> for all the features |

- 6.1. Click **Infrastructure → Subnets** to access the **Subnets** page, and then click **Create Subnet**.
- 6.2. Complete the page according to the preceding table. Do not modify any other field, and do not submit the form yet.
- 6.3. Click the **Domains** tab, and then add the **sf.lab.example.com** domain to the **Selected items** list.

- 6.4. Click the **Capsules** tab, and then set all the fields to **capsule.lab.example.com**.
  - 6.5. Click the **Locations** tab and verify that the **San Francisco** location is already associated with the new subnet. If not, click **San Francisco** to add it to the **Selected items** list.
  - 6.6. Click the **Organizations** tab and verify that the **Finance** organization is already associated with the new subnet. If not, click **Finance** to add it to the **Selected items** list.
  - 6.7. Click **Submit**.
7. Create the **FinanceServers** activation key.
- 7.1. Click **Content** → **Activation Keys**, and then click **Create Activation Key**.
  - 7.2. Enter **FinanceServers** in the Name field.
  - 7.3. Select the **Development** life-cycle environment.
  - 7.4. Select the **Base** content view, and then click **Save**.
  - 7.5. Click the pencil icon for **Release Version**, select **8.1** from the list, and then click **Save**.
  - 7.6. On the **Activation Keys** → **CapsuleServers** page, click the **Subscriptions** tab, and then click the **Add** tab.
  - 7.7. Select the check boxes for all the subscriptions, and then click **Add Selected**.
  - 7.8. Click the **Repository Sets** tab, and then select **Red Hat Satellite Tools 6.6 for RHEL 8 x86\_64 (RPMs)**. Click **Select Action** → **Override to Enabled**.
8. Create a host group called **Finance Host Group**.
- 8.1. Click **Configure** → **Host Groups**, and then click **Create Host Group**.
  - 8.2. Complete the page with the following details.

#### Host Group Details

| Field                        | Value                          |
|------------------------------|--------------------------------|
| <b>Name</b>                  | <b>Finance Host Group</b>      |
| <b>Lifecycle Environment</b> | <b>Development</b>             |
| <b>Content View</b>          | <b>Base</b>                    |
| <b>Content Source</b>        | <b>capsule.lab.example.com</b> |

Do not modify any other field, and do not submit the form yet.

- 8.3. Click the **Network** tab, and then set **Domain** to **sf.lab.example.com**. Set **IPv4 Subnet** to **San Francisco Data Center (172.25.250.0/24)**.
- 8.4. Click the **Operating System** tab, and then complete the page with the following details.

### Host Group Operating System Details

| Field                   | Value                    |
|-------------------------|--------------------------|
| <b>Architecture</b>     | <b>x86_64</b>            |
| <b>Operating system</b> | <b>RedHat 8.1</b>        |
| <b>Media Selection</b>  | <b>Synced Content</b>    |
| <b>Partition Table</b>  | <b>Kickstart default</b> |
| <b>PXE loader</b>       | <b>PXELinux BIOS</b>     |

Do not modify any other field, and do not submit the form yet.

- 8.5. Click the **Locations** tab and confirm that the **San Francisco** location is in the **Selected items** list.
- 8.6. Click the **Organizations** tab and confirm that the **Finance** organization is in the **Selected items** list.
- 8.7. Click the **Activation Keys** tab, and then set **Activation Keys** to **FinanceServers**.
- 8.8. Click **Submit**.
9. Create a host resource for the system to provision, **servere**.
- 9.1. Click **Hosts** → **Create Host**, and then complete the page with the following details.

### New Host Details

| Field               | Value                     |
|---------------------|---------------------------|
| <b>Name</b>         | <b>servere</b>            |
| <b>Organization</b> | <b>Finance</b>            |
| <b>Location</b>     | <b>San Francisco</b>      |
| <b>Host Group</b>   | <b>Finance Host Group</b> |

Do not modify any other field, and do not submit the form yet.

- 9.2. Click the **Operating System** tab, and then set **Root Password** to **redhat123**. Click **Resolve** to confirm that the system can correctly retrieve the templates for provisioning.
- 9.3. Click the **Interfaces** tab, and then click **Edit** at the end of the interface row. Set the **MAC Address** to **52:54:00:00:fa:0e**. This is the address of **servere** network interface. Click **Ok**.
- 9.4. Click **Submit**.
10. Access **servere** console and initiate a PXE boot. Provisioning should proceed automatically. After the installation is complete, the host boots the newly provisioned operating system.

- 10.1. Locate the icon for **servere** console, as appropriate for your classroom environment.  
Open the console.
- 10.2. To reboot, send a **Ctrl+Alt+Del** to your system using the relevant keyboard, virtual, or menu entry.
- 10.3. Press the space bar at the **Boot options** screen to stop the countdown.
- 10.4. Select **Network boot from device net0**. The provisioning process starts automatically.
- 10.5. Wait for the installation to complete and the system to reboot. From the console, log in as **root** with **redhat123** as a password.  
Use the **hostname** command to verify the system name.

```
[root@servere ~]# hostname  
servere.sf.lab.example.com
```

When done, log off from **servere** and close the console.

This concludes the lab.

## Evaluation

As the **student** user on the **workstation** machine, use the **lab** command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab host-cr grade
```

## Finish

As the **student** user on the **workstation** machine, use the **lab** command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab host-cr finish
```

## ► Lab

# Performing Remote Execution

In this review, you will enable and configure remote execution jobs, and then perform remote execution using Ansible Playbooks on a managed host.

## Outcomes

You should be able to:

- Configure a host for remote execution.
- Run remote jobs.
- Create job templates and execute them on a host.

## Before You Begin

To set up your computers for this exercise, log in to **workstation** as **student** and run the following command:

```
[student@workstation ~]$ lab remote-cr start
```

## Instructions

Configure **servera.lab.example.com** for remote execution.

- Register **servera.lab.example.com** to the **Finance** organization. Use the **FinanceServers** activation key.
- Use a remote job to determine the uptime for **servera.lab.example.com**.
- Use the provided **playbook-example-cr.yml** Ansible Playbook to create a job template called "My new custom banner". As job category choose Ansible Playbook, and as Provider choose Ansible. Execute the job on **servera.lab.example.com**.

## Evaluation

As the **student** user on the **workstation** machine, use the **lab** command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab remote-cr grade
```

## Finish

As the **student** user on the **workstation** machine, use the **lab** command to complete this exercise. This ensures that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab remote-cr finish
```

## ► Solution

# Performing Remote Execution

In this review, you will enable and configure remote execution jobs, and then perform remote execution using Ansible Playbooks on a managed host.

### Outcomes

You should be able to:

- Configure a host for remote execution.
- Run remote jobs.
- Create job templates and execute them on a host.

### Before You Begin

To set up your computers for this exercise, log in to **workstation** as **student** and run the following command:

```
[student@workstation ~]$ lab remote-cr start
```

### Instructions

Configure **servera.lab.example.com** for remote execution.

- Register **servera.lab.example.com** to the **Finance** organization. Use the **FinanceServers** activation key.
- Use a remote job to determine the uptime for **servera.lab.example.com**.
- Use the provided **playbook-example-cr.yml** Ansible Playbook to create a job template called "My new custom banner". As job category choose Ansible Playbook, and as Provider choose Ansible. Execute the job on **servera.lab.example.com**.

1. Register **servera.lab.example.com** to the **Finance** organization. Use the **FinanceServers** activation key.

- 1.1. On **workstation**, ssh to **servera** as **student**. Use **sudo -i** to switch to **root**.

```
[student@workstation ~]$ ssh student@servera
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

- 1.2. Install the consumer RPM, *katello-ca-consumer-latest.noarch.rpm*.

```
[root@servera ~]# yum localinstall \
http://satellite.lab.example.com/pub/katello-ca-consumer-latest.noarch.rpm
```

- 1.3. Register the system to the **Finance** organization. Reference the **FinanceServers** activation key with the **--activationkey** option.

```
[root@servera ~]# subscription-manager register --org=Finance \
--activationkey=FinanceServers
The system has been registered with ID: 95424d51-69a1-4142-ad18-4453dc96364e
The registered system name is: servera.lab.example.com
Installed Product Current Status:
Product Name: Red Hat Enterprise Linux Server
Status: Subscribed
```

2. Distribute SSH keys for remote execution.

- 2.1. On **workstation**, use **ssh** to log in to **satellite** as **student**. Use **sudo -i** to switch to **root**.

```
[student@workstation ~]$ ssh student@satellite
[student@satellite ~]$ sudo -i
[sudo] password for student: student
[root@satellite ~]#
```

- 2.2. Distribute the SSH key manually to the **servera.lab.example.com** host.

Use the **ssh-copy-id** command. The key is located at **~foreman-proxy/.ssh/id\_rsa\_foreman\_proxy.pub**

```
[root@satellite ~]# ssh-copy-id -i \
~foreman-proxy/.ssh/id_rsa_foreman_proxy.pub \
root@servera.lab.example.com
...output omitted...
Are you sure you want to continue connecting (yes/no)? yes
...output omitted...
root@servera.lab.example.com's password: redhat

Number of key(s) added: 1
...output omitted...
```

3. Log into the Satellite Server web interface located at **https://satellite.lab.example.com** as the **admin** user with **redhat** as the password.
4. Using the organization and location context menu in the upper-left, select the **Finance** organization and the **Any Location** location context.
5. Run a remote command on **servera.lab.example.com** server.

- 5.1. Click **Hosts** → **All Hosts**.

- 5.2. On the **Hosts** page, click the **servera.lab.example.com** link.

- 5.3. Click **Schedule Remote Job**.

- 5.4. Set the **Job category** to **Commands**.

- 5.5. Verify that the **Job template** is set to **Run Command - SSH Default**.

- 5.6. In the **command** text field, type **uptime**. This will display the uptime for the host, the host name, and the user who ran the command.

```
uptime
```

- 5.7. Click **Submit** to execute the remote command.

A page displays the status of the remote execution. Wait for it to successfully finish.

- 6. To see the output of the remote command, click the **servera.lab.example.com** link. You should see an output similar to the following:

```
1: 15:30:22 up 2 days, 1 user, load average: 0.00, 0.00, 0.00
2: Exit status: 0
```

- 7. Use the provided **playbook-example-cr.yml** Ansible Playbook to create a job template called "My new custom banner". As job category choose Ansible Playbook, and as Provider choose Ansible.

- 7.1. Click **Hosts** → **Job templates**.

- 7.2. On the **Job Templates** page, click **New Job Template**.

- 7.3. Type **My new custom banner** as the name for the new job template.

- 7.4. In the new template editor window, copy and paste the content of the **playbook-example-cr.yml** file that is located in the home directory of the student user on **workstation**.

- 7.5. Click **Job**.

- 7.6. Remove the existing job category and from the list choose **Ansible Playbook**.

- 7.7. Change the **Provider Type** to **Ansible**.

- 7.8. Click **Submit** to save the new job template.

- 8. Execute the new job template on **servera.lab.example.com**.

- 8.1. Click **Hosts** → **All Hosts**.

- 8.2. On the **Hosts** page, click the **servera.lab.example.com** link.

- 8.3. Click **Schedule Remote Job**.

- 8.4. Change the **Job category** to **Ansible Playbook**.

- 8.5. Change the **Job category** to the new **My new custom banner** template.

- 8.6. Click **Submit** to execute the remote command.

On the next page, click the **servera.lab.example.com** link and observe the Ansible Playbook execution output.

- 9. Verify that the SSH banner is replaced by your custom message.

- 9.1. On **workstation**, use **ssh** to log in to **servera** as **student**.

```
[student@workstation ~]$ ssh student@servera
"Welcome to the RH403 course,
Comprehensive Review chapter"
...output omitted...
[student@servera ~]$
```

10. Exit from the **satellite** and **servera** hosts and return to **workstation**.

This concludes the lab.

## Evaluation

As the **student** user on the **workstation** machine, use the **lab** command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab remote-cr grade
```

## Finish

As the **student** user on the **workstation** machine, use the **lab** command to complete this exercise. This ensures that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab remote-cr finish
```

## ► Lab

# Signing RPM Packages

In this review, you will import a GPG key pair, and sign an RPM package.

### Outcomes

You should be able to:

- Import a GPG key pair.
- Sign an RPM package.

### Before You Begin

To set up your computers for this exercise, log in to **workstation** as **student** and run the following command:

```
[student@workstation ~]$ lab sign-cr start
```

### Instructions

Perform the following tasks to complete the comprehensive review:

- Import the GPG key pair from the file at <http://materials.example.com/key.asc>. Use **testing123** as the passphrase.
- Use that GPG key pair to sign the RPM package available at [http://materials.example.com/sm-practice-1.0-1.el7.x86\\_64.rpm](http://materials.example.com/sm-practice-1.0-1.el7.x86_64.rpm).

## Evaluation

As the **student** user on the **workstation** machine, use the **lab** command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab sign-cr grade
```

## Finish

As the **student** user on the **workstation** machine, use the **lab** command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab sign-cr finish
```

This concludes the comprehensive review.

## ► Solution

# Signing RPM Packages

In this review, you will import a GPG key pair, and sign an RPM package.

### Outcomes

You should be able to:

- Import a GPG key pair.
- Sign an RPM package.

### Before You Begin

To set up your computers for this exercise, log in to **workstation** as **student** and run the following command:

```
[student@workstation ~]$ lab sign-cr start
```

### Instructions

Perform the following tasks to complete the comprehensive review:

- Import the GPG key pair from the file at <http://materials.example.com/key.asc>. Use **testing123** as the passphrase.
- Use that GPG key pair to sign the RPM package available at [http://materials.example.com/sm-practice-1.0-1.el7.x86\\_64.rpm](http://materials.example.com/sm-practice-1.0-1.el7.x86_64.rpm).

1. Import the GPG key pair available at <http://materials.example.com/key.asc>. Use **testing123** as the passphrase.
  - 1.1. Download the GPG key pair file available at <http://materials.example.com/key.asc>.

```
[student@workstation ~]$ wget \
http://materials.example.com/key.asc
```

- 1.2. Import the GPG key pair available at the **key.asc** file. Enter the pass phrase, **testing123**, when prompted.

```
[student@workstation ~]$ gpg --import key.asc
gpg: directory '/home/student/.gnupg' created
gpg: keybox '/home/student/.gnupg/pubring.kbx' created
gpg: /home/student/.gnupg/trustdb.gpg: trustdb created
gpg: key 5C0009F4E43EEF8D: public key "student
<student@workstation.lab.example.com>" imported
gpg: key 5C0009F4E43EEF8D: secret key imported
gpg: Total number processed: 1
```

```
gpg:           imported: 1
gpg:    secret keys read: 1
gpg:    secret keys imported: 1
```

- 1.3. List the key information by using the command **gpg --fingerprint**. You use the name and email address later in this lab.

```
[student@workstation ~]$ gpg --fingerprint
/home/student/.gnupg/pubring.kbx
-----
pub   rsa2048 2019-12-20 [SC]
EEC4 91E4 3835 696D 255A BAEC 5C00 09F4 E43E EF8D
uid   [ultimate] student <student@workstation.lab.example.com>
sub   rsa2048 2019-12-20 [E]
```

2. Sign the RPM package available at [http://materials.example.com/sm-practice-1.0-1.el7.x86\\_64.rpm](http://materials.example.com/sm-practice-1.0-1.el7.x86_64.rpm) with the GPG key pair.

- 2.1. Install the *rpm-sign* RPM package.

```
[student@workstation ~]$ sudo yum install -y rpm-sign
...output omitted...
```

- 2.2. Create or modify the **.rpmmacros** file in **/home/student**. Set the **%\_gpg\_name** macro value to the previously created GPG key ID for **student**.

```
[student@workstation ~]$ echo \
'%_gpg_name student <student@workstation.lab.example.com>' >> ~/.rpmmacros
```

- 2.3. Download the *sm-practice-1.0-1.el7.x86\_64.rpm* RPM package available at [http://materials.example.com/sm-practice-1.0-1.el7.x86\\_64.rpm](http://materials.example.com/sm-practice-1.0-1.el7.x86_64.rpm).

```
[student@workstation ~]$ wget \
http://materials.example.com/sm-practice-1.0-1.el7.x86_64.rpm
```

- 2.4. Sign the *sm-practice-1.0-1.el7.x86\_64.rpm* RPM package. Use **testing123** as the pass phrase.

```
[student@workstation ~]$ rpmsign --addsign \
sm-practice-1.0-1.el7.x86_64.rpm
Enter pass phrase: testing123
Pass phrase is good.
sm-practice-1.0-1.el7.x86_64.rpm:
```

- 2.5. Verify the signature for *sm-practice-1.0-1.el7.x86\_64.rpm* RPM package.



### Note

The NOKEY warning message can be ignored for this exercise. It means that although the package has been signed, the signing key has not been imported into the local RPM database.

```
[student@workstation ~]$ rpm -qip sm-practice-1.0-1.el7.x86_64.rpm
warning: sm-practice-1.0-1.el7.x86_64.rpm: Header V4 RSA/SHA256 Signature, key ID
e43eef8d: NOKEY
Name        : sm-practice
Version     : 1.0
Release     : 1.el7
Architecture: x86_64
Install Date: (not installed)
Group       : Unspecified
Size        : 1300
License      : BSD
Signature    : RSA/SHA256, vie 20 dic 2019 06:44:56 EST, Key ID 5c0009f4e43eef8d
Source RPM   : sm-practice-1.0-1.el7.src.rpm
Build Date   : jue 19 dic 2019 06:38:10 EST
Build Host   : workstation.lab.example.com
Relocations  : (not relocatable)
URL         : http://workstation.lab.example.com
Summary      : A simple program to practice building RPM packages
Description  :
This program will function as a practice exercise for building an RPM package.
```

This concludes the lab.

## Evaluation

As the **student** user on the **workstation** machine, use the **lab** command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab sign-cr grade
```

## Finish

As the **student** user on the **workstation** machine, use the **lab** command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab sign-cr finish
```

This concludes the comprehensive review.