



Red Hat Satellite 6.3

Managing Hosts

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

Red Hat Satellite 6.3 Managing Hosts

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

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

Legal Notice

Copyright © 2018 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, JBoss, OpenShift, 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 trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL ® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js ® is an official trademark of Joyent. Red Hat Software Collections is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

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.

Abstract

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

Table of Contents

CHAPTER 1. INTRODUCTION	4
1.1. UNDERSTANDING HOSTS	4
1.2. UNDERSTANDING HOST MANAGEMENT	4
CHAPTER 2. MANAGING CONTAINERS	5
2.1. PREPARING CONTAINER HOSTS	5
2.2. CREATING CONTAINERS	6
2.3. MONITORING CONTAINERS	8
2.4. STARTING, COMMITTING, AND REMOVING CONTAINERS	9
CHAPTER 3. MANAGING HOSTS	11
3.1. BROWSING HOSTS	11
3.2. HOST STATUS TYPES	11
3.3. HOST OVERVIEW	12
3.4. CREATING A HOST	13
3.5. REGISTRATION	14
3.5.1. Configuring a Host for Registration	15
3.5.2. Registering a Host	16
3.5.3. Installing the Katello Agent	18
3.5.4. Installing Tracer	19
3.5.5. Installing and Configuring the Puppet Agent	20
3.5.6. Registering Hosts to Satellite 6 Using The Bootstrap Script	23
3.5.7. Advanced Bootstrap Script Configuration	27
3.6. CHANGING THE GROUP OF A HOST	30
3.7. CHANGING THE ENVIRONMENT OF A HOST	30
3.8. MANAGING HOSTS	30
3.9. ASSIGNING A HOST TO A SPECIFIC ORGANIZATION	31
3.10. ASSIGNING A HOST TO A SPECIFIC LOCATION	31
3.11. CONFIGURING AN ADDITIONAL NETWORK INTERFACE	32
3.11.1. Adding a Physical Interface	32
3.11.2. Adding a Virtual Interface	33
3.11.3. Adding a Bonded Interface	34
3.11.4. Adding a Baseboard Management Controller (BMC) Interface	35
3.12. REMOVING A HOST	36
CHAPTER 4. CONFIGURING HOST COLLECTIONS	38
4.1. CREATING A HOST COLLECTION	38
4.2. CLONING A HOST COLLECTION	38
4.3. REMOVING A HOST COLLECTION	38
4.4. ADDING A HOST TO A HOST COLLECTION	39
4.5. REMOVING A HOST FROM A HOST COLLECTION	39
4.6. ADDING CONTENT TO A HOST COLLECTION	39
4.6.1. Adding Packages to a Host Collection	39
4.6.2. Adding Errata to a Host Collection	40
4.7. REMOVING CONTENT FROM A HOST COLLECTION	40
4.8. CHANGING THE LIFE CYCLE ENVIRONMENT OR CONTENT VIEW OF A HOST COLLECTION	41
CHAPTER 5. RUNNING JOBS ON HOSTS	42
5.1. ESTABLISHING A SECURE CONNECTION FOR REMOTE COMMANDS	42
5.2. SETTING UP KERBEROS AUTHENTICATION FOR REMOTE EXECUTION	44
5.3. CONFIGURING AND RUNNING REMOTE COMMANDS	45
5.3.1. Setting up Job Templates	45

5.3.2. Executing Jobs	47
5.3.3. Monitoring Jobs	48
5.3.4. Creating Advanced Templates	48
5.4. CONFIGURING GLOBAL SETTINGS	49
5.4.1. Choosing a Capsule for Remote Execution	50
5.5. DELEGATING PERMISSIONS FOR REMOTE EXECUTION	51
CHAPTER 6. DISCOVERING BARE-METAL HOSTS ON SATELLITE	52
6.1. NETWORK CONFIGURATION FOR PXE-BASED DISCOVERY	52
6.2. CONFIGURING THE SATELLITE DISCOVERY PLUG-IN	53
6.2.1. Deploying the Satellite Discovery Image	53
6.2.2. Configuring PXE-booting	53
6.2.3. Reviewing Global Discovery Settings	54
6.3. CONFIGURING THE SATELLITE CAPSULE SERVER DISCOVERY PLUG-IN	55
6.3.1. Configuring Discovery Subnets	55
6.3.2. Using Hammer with the Discovery Plug-in	55
6.3.3. Reviewing User Permissions	55
6.4. PROVISIONING DISCOVERED HOSTS	55
6.4.1. Manually Provisioning Hosts	56
6.4.2. Decommissioning Discovered Hosts	56
6.4.3. Automatically Provisioning Hosts	56
6.4.4. Scoped Search Syntax	57
6.4.5. Host Name Patterns	58
6.4.6. Using the Discovery Plug-in on the Command Line	58
6.5. EXTENDING THE DISCOVERY IMAGE	59
6.6. TROUBLESHOOTING SATELLITE DISCOVERY	60
CHAPTER 7. INTEGRATING RED HAT SATELLITE AND ANSIBLE TOWER	61
7.1. ADDING SATELLITE SERVER TO ANSIBLE TOWER AS A DYNAMIC INVENTORY ITEM	61
7.2. CONFIGURING PROVISIONING CALLBACK FOR A HOST	62
CHAPTER 8. SAMPLE SCENARIOS	66
8.1. SIMPLE SCENARIO	66
8.1.1. Creating the Host	66
8.1.2. Registering the Host	67
8.1.3. Running a Job on the Host	69
APPENDIX A. TEMPLATE WRITING REFERENCE	70
A.1. WRITING ERB TEMPLATES	70
A.2. TROUBLESHOOTING ERB TEMPLATES	71
A.3. SATELLITE SPECIFIC FUNCTIONS AND VARIABLES	71
APPENDIX B. HOST MANAGEMENT WITHOUT GOFERD	78
B.1. PREREQUISITES	78
B.2. CONFIGURING HOST MANAGEMENT WITHOUT GOFERD AS THE SYSTEM DEFAULT	78
B.3. LIMITATIONS WITH HAMMER	78

CHAPTER 1. INTRODUCTION

This chapter describes what a host is, types of hosts, and actions you can perform on them.

1.1. UNDERSTANDING HOSTS

A host is any Red Hat Enterprise Linux system that Red Hat Satellite manages. Hosts may be physical or virtual. Virtual hosts may be deployed on any platform supported by Red Hat Satellite, including: KVM, VMware vSphere, OpenStack, Amazon EC2, Rackspace Cloud Services, Google Compute Engine, or in a Docker container.

1.2. UNDERSTANDING HOST MANAGEMENT

Red Hat Satellite enables host management at scale, including monitoring, provisioning, remote execution, configuration management, software management, and subscription management. You can manage your hosts via the Red Hat Satellite web UI, or the command line. For more information on managing hosts, see [Chapter 3, *Managing Hosts*](#).

CHAPTER 2. MANAGING CONTAINERS

This chapter outlines how to manage containers. For instructions on setting up container content, see [Managing Container Images](#) in the *Content Management Guide*.

2.1. PREPARING CONTAINER HOSTS

Prerequisites

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

To Prepare a Container Host:

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



NOTE

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

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

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

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

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

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

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

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

- ii. Modify the **OPTIONS** variable in the **/etc/sysconfig/docker** file as follows:

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

- iii. Restart the affected services to apply the changes:

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

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

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

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

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

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

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

To Create a Docker Compute Resource:

1. Make sure the port 5000 is enabled on the Satellite Server. The container host uses this port to pull images from Content Views on the Satellite Server.
2. Create the compute resource as described in [Adding an Atomic Host Connection to the Satellite Server](#) in the *Provisioning Guide*. Specify the resource **URL** according to the location of the container host:
 - a. If the container host is on the same machine as the Satellite Server, set **unix:///var/run/docker.sock** as the resource URL.
3. If the container host is on a different machine than the Satellite Server, specify the URL in the form of:

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

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

4. Click **Test Connection** to test if the container host is available.
5. Click **Submit** to create the compute resource.

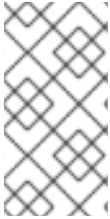
2.2. CREATING CONTAINERS

When there is at least one Docker compute resource present in your Satellite, you can create containers. To create a new container, follow the steps described in [To Create a Container](#). For instructions on how to monitor existing containers, see [Section 2.3, “Monitoring Containers”](#).

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

- **Local content:** represented by the **Content View** option when creating a container. This option allows you to import an image from a repository that is already present on a Capsule Server in a certain Content View and life cycle environment. For more information on how to create and populate a local registry, see [Importing Container Images from the Red Hat Container Catalog](#) in the *Content Management Guide*.

- **Docker Hub:** allows you to search the Docker Hub registry and pull images from there. Make sure that you pull only trusted images with verified content.
- **External Registry:** allows you to import images from a previously created external registry. For more information on creating registries in Red Hat Satellite, see [Importing Container Images from Other Image Registries](#) in the *Content Management Guide*.



NOTE

You cannot change the configuration of an existing container. To alter the configuration, you have to create a replacement container with modified settings as described in [To Create a Container](#). Therefore, make sure that containers can be replaced in your workflow.

To Create a Container:

1. Navigate to **Containers > New Container**. Alternatively, navigate to **Containers > All Containers** and click **New container**.
2. In the **Preliminary** stage of container creation, configure the following settings:
 - On the **Compute resource** tab, select the compute resource from the **Deployed on** drop-down menu. For more information on compute resources, see [Defining the Provisioning Workflow](#) in the *Provisioning Guide*.
 - On the **Locations** tab, select the locations where the new container will be available.
 - On the **Organizations** tab, select the organizations where the new container will be available.
Click **Next** to proceed.
3. In the **Image** stage of container creation, import an image that will act as a base for your container. This can be a platform image, or a previously created layered image. Select from one of the following options:
 - Select the **Content View** tab to import the image from a life cycle environment. Specify the life cycle environment, Content View, repository, tag, and Capsule Server.
 - Select the **Docker hub** tab to import the image from the Docker Hub registry. After you type the image name to the **Search** field, Satellite automatically searches the compute resource. Click the looking glass icon to search the Docker Hub. Select the image from the list of search results and pick a tag from the drop-down list.
 - Select the **External registry** tab to import the image from an existing registry. Select the registry from the drop-down menu, and search it by the image name. Satellite populates the **Tag** field with tags available for the selected image name. For more information, see [Importing Container Images from Other Image Registries](#) in the *Content Management Guide*.
Click **Next** to proceed.
4. In the **Configuration** stage of container creation, set the following parameters:
 - Provide the container name.
 - Specify a command to run inside the container.

- Specify an entrypoint, which is a command that is executed automatically as soon as the container starts. The default entrypoint is `/bin/sh -c`.
 - Assign CPUs to the container. For example, `0-2,16` represents CPUs 0, 1, 2, and 16.
 - Define the relative share of CPU time for the container.
 - Specify a memory limit for the container. For example, `512m` limits the container memory usage to 512 MB.
Click **Next** to proceed.
5. In the final stage of container creation named **Environment**, select if you want to allocate a pseudo-tty, attach STDIN, STDOUT, and STDERR to the container. Click **Add environment variable** to create a custom environment variable for the container. Select the **Run?** check box to start the container automatically after it is created.
 6. Click **Submit** to create the container.

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

Example 2.1. Creating a Red Hat Enterprise Linux Container in Satellite

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

1. Create a custom registry as described in [Importing Container Images from the Red Hat Container Catalog](#) in the *Content Management Guide*. Specify `registry.access.redhat.com` as the registry URL.
2. Create a new container as described in [Section 2.2, “Creating Containers”](#). In the **Image** stage of container creation, navigate to the **External registry** tab and select the registry created in the previous step. Use the search field to find the desired version of the Red Hat Enterprise Linux image. Proceed through the **Configuration** and **Environment** stages to finalize the container.

2.3. MONITORING CONTAINERS

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

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

To Investigate a Container:

1. Navigate to **Containers > All Containers**.
2. On the **Containers** page, every Docker compute resource has a dedicated tab. Each of these tabs contains the table of available containers together with selected parameters of each container. Select the tab of the compute resource you want to inspect.
3. To view the container metadata, click the name of the container you want to inspect. Satellite displays the table of container properties.

4. On the **Processes** tab, you can view processes that are currently running in the container. Click on the process name to view the metadata of the process.
5. If the container is running, you can view its standard output in the **Logs** tab. If you selected the **allocate a pseudo-tty** check box when creating a container, the console is interactive. Otherwise, it displays the initial standard output produced when the container started.

2.4. STARTING, COMMITTING, AND REMOVING CONTAINERS

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

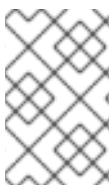
To Start or Stop a Container:

1. Navigate to **Containers > All Containers** to view the list of available containers.
2. Click **Power On** next to the container you want to start. After starting the container, the button changes to **Power Off**, which allows for stopping the container. These actions are equivalent to the **docker start** and **docker stop** commands.

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

To Commit a Container:

1. Navigate to **Containers > All Containers** to view the list of available containers.
2. Click the name of the container you want to commit.
3. Click **Commit**. Satellite prompts you to:
 - Specify a repository name. This can be a single name or combined with the user name, for example **user/my-rhel-image**.
 - Assign a tag to the image.
 - Provide your contact information.
 - Provide an informative comment about the image.
4. Click **Submit**.



NOTE

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

To Remove a Container:

1. Navigate to **Containers > All Containers** to view the list of available containers.
2. Click the name of the container you want to delete.
3. Click **Delete**.

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

CHAPTER 3. MANAGING HOSTS

This chapter describes creating, registering, managing, and removing hosts. This includes changing the group, and environment of a host, configuring an additional network interface, assigning the host to a specific organization, and location.

3.1. BROWSING HOSTS

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

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

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



WARNING




Satellite Server will be listed as a host itself even if it is not self-registered. Do not delete the Satellite Server from the list of hosts.

3.2. HOST STATUS TYPES

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

Table 3.1. Host Status Types

Icon	Status	Description
------	--------	-------------

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

3.3. HOST OVERVIEW

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

Details

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

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

Host Actions

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

- **Schedule Remote Job:** allows running jobs on the host. For more information on running jobs see [Chapter 5, Running Jobs on Hosts](#).
- **Boot disk:** a menu that allows you to select the boot disk for the host. For more information on creating a boot ISO for a host see [Creating New Hosts with PXE-less Provisioning](#) in the *Red Hat Satellite Provisioning Guide*.
- **Edit:** opens the host details page which allows you to configure settings for the host. Note that the installer configures all the settings automatically and normally no manual configurations are required.
- **Build:** flags the host to be provisioned on the next host boot. Note that the installer manages all aspects of the provisioning process and normally there is no need to provision hosts manually.
- **Delete:** deletes the host from the user interface.

Host Graphs

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

- **Runtime:** tracks two data points: **Config Retrieval** and **Runtime**. The **Config Retrieval** data point represents the amount of time taken to collect information about the host during a given Puppet run, and the **Runtime** data point represents the amount of time required to execute the Puppet run. Both data points are measured in seconds.
- **Resources:** tracks the number of actions performed on the host during a Puppet run. The categories displayed in this graph are identical to those displayed in the **Reports** page, and are measured using the number of actions in each category.

3.4. CREATING A HOST

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

To Create a Host:

1. Click **Hosts > Create Host**.
2. On the **Host** tab, enter the required details.
3. On the **Puppet Classes** tab, select the Puppet classes you want to include.
4. On the **Interfaces** tab:
 - a. For each interface, click **Edit** in the **Actions** column and configure the following settings as required:
 - **Type** — For a Bond or BMC interface, use the **Type** list and select the interface type.
 - **MAC address** — Enter the MAC address.
 - **DNS name** — Enter the DNS name that is known to the DNS server. This is used for the host part of the FQDN.

- **Domain** — Select the domain name of the provisioning network. This automatically updates the **Subnet** list with a selection of suitable subnets.
 - **IPv4 Subnet** — Select an IPv4 subnet for the host from the list.
 - **IPv6 Subnet** — Select an IPv6 subnet for the host from the list.
 - **IPv4 address** — If IP address management (IPAM) is enabled for the subnet, the IP address is automatically suggested. Alternatively, you can enter an address. The address can be omitted if provisioning tokens are enabled, if the domain does not manage DNS, if the subnet does not manage reverse DNS, or if the subnet does not manage DHCP reservations.
 - **IPv6 address** — If IP address management (IPAM) is enabled for the subnet, the IP address is automatically suggested. Alternatively, you can enter an address.
 - **Managed** — Select this check box to configure the interface during provisioning to use the Capsule provided DHCP and DNS services.
 - **Primary** — Select this check box to use the DNS name from this interface as the host portion of the FQDN.
 - **Provision** — Select this check box to use this interface for provisioning. This means TFTP boot will take place using this interface, or in case of image based provisioning, the script to complete the provisioning will be executed through this interface. Note that many provisioning tasks, such as downloading RPMs by **anaconda**, Puppet setup in a **%post** script, will use the primary interface.
 - **Virtual NIC** — Select this check box if this interface is not a physical device. This setting has two options:
 - **Tag** — Optionally set a VLAN tag. If unset, the tag will be the VLAN ID of the subnet.
 - **Attached to** — Enter the device name of the interface this virtual interface is attached to.
- b. Click **OK** to save the interface configuration.
 - c. Optionally, click **Add Interface** to include an additional network interface. See [Section 3.11, “Configuring an Additional Network Interface”](#) for details.
 - d. Press **Submit** to apply the changes and exit.
5. On the **Operating System** tab, enter the required details. You can select a partition table from the drop-down list or enter a custom partition table in the **Custom partition table** field. You cannot specify both.
 6. On the **Parameters** tab, click **Add Parameter** to add any required parameters. This includes all Puppet Class Parameters and Host Parameters associated with the host.
 7. On the **Additional Information** tab, enter additional information about the host.
 8. Click **Submit** to complete your provisioning request.

3.5. REGISTRATION

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

- Download and install the consumer RPM (*server.example.com/pub/katello-ca-consumer-latest.noarch.rpm*) and then run subscription manager. This method is suited for freshly installed hosts. See [Section 3.5.1, “Configuring a Host for Registration”](#) and [Section 3.5.2, “Registering a Host”](#) for more information.
- Download and run the bootstrap script (*server.example.com/pub/bootstrap.py*). This method is suited for both freshly installed hosts and hosts that have been previously registered, for example, to Satellite 5 or another Satellite 6. See [Section 3.5.6, “Registering Hosts to Satellite 6 Using The Bootstrap Script”](#) for more information.

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

3.5.1. Configuring a Host for Registration

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

Supported Host Operating Systems

Hosts must use the following Red Hat Enterprise Linux version:

- 5.7 or later
- 6.1 or later*
- 7.0 or later



NOTE

Red Hat Enterprise Linux versions 6.1, 6.2 and 6.3 GA require **subscription-manager** and related packages to be updated manually. For more information, see <https://access.redhat.com/solutions/1256763>

Supported Architectures

All architectures of Red Hat Enterprise Linux are supported:

- i386
- x86_64
- s390x
- ppc_64

Prerequisites

- Ensure that the Satellite Servers, any Capsule Servers, and all hosts are synchronized with the same NTP server.

- Ensure that a time synchronization tool is enabled and running on the Satellite Servers, any Capsule Servers, and the hosts.
 - For Red Hat Enterprise Linux 6:


```
# chkconfig ntpd on; service ntpd start
```
 - For Red Hat Enterprise Linux 7:


```
# systemctl start chronyd; systemctl enable chronyd
```
- Ensure that the daemon **rhsmcertd** is running on the hosts.
 - For Red Hat Enterprise Linux 6:


```
# service rhsmcertd start
```
 - For Red Hat Enterprise Linux 7:

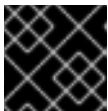

```
# systemctl start rhsmcertd
```

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

To Configure a Host for Registration:

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

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



IMPORTANT

Any running Docker Daemons will be restarted.



NOTE

The RPM package is not signed, if required, you can add the **--nosignature** option to install the package. The **katello-ca-consumer-hostname-1.0-1.noarch.rpm** is an additional katello-ca-consumer RPM available that contains the server's host name. The **katello-ca-consumer-latest.noarch.rpm** rpm will always reflect the most recent version. Both serve the same purpose.

3.5.2. Registering a Host

Prerequisites

- Complete all steps in [Section 3.5.1, “Configuring a Host for Registration”](#).
- Ensure that an activation key associated with the appropriate Content View and environment exists for the host. If not, see [Managing Activation Keys](#) in the *Content Management Guide* for more information. By default, an activation key has the auto-attach function enabled. The feature is commonly used with hosts used as hypervisors.
- Ensure that the version of the **subscription-manager** utility installed is 1.10 or higher. The package is available in the standard Red Hat Enterprise Linux repository.

To Register Hosts:

1. In a terminal, connect to the host as the root user.
2. Clear any old host data related to Red Hat Subscription Manager (RHSM):

```
# subscription-manager clean
```

3. Register the host using RHSM:

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

Example 3.1. Command Output after Registration:

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

NOTE

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

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





NOTE

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

To Point Red Hat Enterprise Linux 6.3 to the Repository:

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

3.5.3. Installing the Katello Agent

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

Note that **yum** is the only package management command that will update Satellite by triggering a Katello package upload. Avoid using **rpm** commands to install or update packages.

Prerequisites

Satellite version 6.1 and later require that you enable the **Red Hat Satellite Tools 6** repository. The **Red Hat Common** repositories are no longer used and are not compatible with Satellite version 6.1 and later.

The **Red Hat Satellite Tools 6** repository must be enabled, synchronized to the Red Hat Satellite Server and made available to your hosts as it provides the required packages.

To Verify the Red Hat Satellite Tools 6 Repository is Enabled:

1. Open the Satellite web UI, navigate to **Content** > **Red Hat Repositories** and click the **RPMS** tab.
2. Find and expand the **Red Hat Enterprise Linux Server** item.
3. Find and expand the **Red Hat Satellite Tools 6 (for RHEL VERSION Server) (RPMS)** item.
If the **Red Hat Satellite Tools 6** items are not visible, it may be because they are not included in the subscription manifest obtained from the Customer Portal. To correct that, log in to the Customer Portal, add these repositories, download the subscription manifest and import it into Satellite.
4. Ensure the **Enabled** check box beside the repository's name is selected. If not, select it.

Enable the **Red Hat Satellite Tools 6** repository for every supported major version of Red Hat Enterprise Linux running on your hosts.

To Install Katello Agent:

1. On the host, verify that the **rhel-version-server-satellite-tools-6-rpms** repository is enabled. If you registered the host using an activation key with auto-attach enabled, the repository is enabled automatically already.

```
# yum repolist enabled | grep -i rhel-version-server-satellite-
tools-6-rpms
```

If the **rhel-version-server-satellite-tools-6-rpms** is not enabled, enable it using the following command:

```
# subscription-manager repos --enable=rhel-version-server-satellite-
tools-6-rpms
```

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

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

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

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

```
# service goferd start
```

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

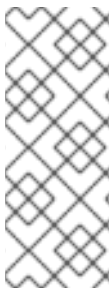
```
# systemctl start goferd
```

3.5.4. Installing Tracer

This section describes how you can integrate Red Hat Satellite 6.3 with Tracer, and provides instructions on how to install Tracer and access Traces.

Tracer is a monitoring tool that identifies if any running processes on a system must be restarted.

Traces displays a list of applications that are outdated and require to be restarted. Traces is accessible through the Satellite web UI.



NOTE

The integration of Tracer with Satellite Server is a Technology Preview feature. Technology Preview features are not fully supported under Red Hat Subscription Service Level Agreements (SLAs), may not be functionally complete, and are not intended for production use. However, these features provide early access to upcoming product innovations, enabling customers to test functionality and provide feedback during the development process.

Prerequisite

The Red Hat Satellite Tools 6.3 repository must be enabled, synchronized to Red Hat Satellite Server and made available to your hosts.

To Install Tracer:

Tracer is an optional component, therefore you must install it separately from the rest of the Katello host tools.

1. On the content host, install the **katello-host-tools-tracer** RPM package:

```
# yum install katello-host-tools-tracer
```

2. Enter the following command:

```
# katello-tracer-upload
```

3. Verify that Tracer is installed.
 - a. In the Satellite web UI, navigate to **Hosts > All hosts**, and then click on the required host name.
 - b. In the **Properties** tab examine the **Properties** table, and then find the Traces item. If you cannot find a Traces item in the **Properties** table, then Tracer is not installed.

To Access Traces:

Traces is the output generated by Tracer in the Satellite web UI.

1. In the Satellite web UI, navigate to **Hosts > Content Hosts**, and then click on the required host name.
2. Click on the **Traces** tab to view any displayed Traces.

For more information about the Tracer tool, see Red Hat Knowledgebase article [Satellite 6.3 Feature Overview: Tracer \[Tech Preview\]](#).

3.5.5. Installing and Configuring the Puppet Agent

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

Prerequisites

The **Red Hat Satellite Tools 6** repository must be enabled, synchronized to the Red Hat Satellite Server and made available to your hosts as it provides the required packages.

To Verify the Red Hat Satellite Tools 6 Repository is Enabled:

1. Open the Satellite web UI, navigate to **Content > Red Hat Repositories** and click the **RPMS** tab.
2. Find and expand the **Red Hat Enterprise Linux Server** item.
3. Find and expand the **Red Hat Satellite Tools 6 (for RHEL *VERSION* Server) (RPMS)** item.
If the **Red Hat Satellite Tools 6** items are not visible, it may be because they are not included in the subscription manifest obtained from the Customer Portal. To correct that, log in to the Customer Portal, add these repositories, download the subscription manifest and import it into Satellite.
4. Ensure the **Enabled** check box beside the repository's name is selected. If not, select it.

To Install and Enable the Puppet Agent:

1. On the host, open a terminal console and log in as the **root** user.
2. Verify that the **rhel-version-server-satellite-tools-6-rpms** repository is enabled, using the following command:

```
# yum repolist enabled | grep -i rhel-version-server-satellite-  
tools-6-rpms
```

If the **rhel-version-server-satellite-tools-6-rpms** is not enabled, enable it using the following command:

```
# subscription-manager repos --enable=rhel-version-server-satellite-  
tools-6-rpms
```

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

```
# yum install puppet
```

4. Configure the puppet agent to start at boot:

- a. On Red Hat Enterprise Linux 6:

```
# chkconfig puppet on
```

- b. On Red Hat Enterprise Linux 7:

```
# systemctl enable puppet
```

Prerequisites

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

- The host must be registered to the Red Hat Satellite Server.
- The Red Hat Satellite Tools 6 repository must be enabled.
- Puppet packages must be installed on the host.

To Configure the Puppet Agent:

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

```
# vi /etc/puppet/puppet.conf  
  
[main]  
# The Puppet log directory.  
# The default value is '$vardir/log'.  
logdir = /var/log/puppet  
  
# Where Puppet PID files are kept.  
# The default value is '$vardir/run'.
```

```

rundir = /var/run/puppet

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

...

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

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

...

```

IMPORTANT

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

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

For more information see the [Red Hat Satellite Puppet Guide](#).

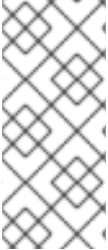
2. Run the Puppet agent on the host:

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

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

- a. Log in to the Satellite Server through the web UI.
- b. Select **Infrastructure > Capsules**.
- c. Select **Certificates** from the drop-down menu to the right of the required Capsule.
- d. Click **Sign** to the right of the required host.
- e. Enter the **puppet agent** command again:

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



NOTE

When the Puppet agent is configured on the host it will be listed under **All Hosts** but only when **Any Organization** is selected as the host will not be assigned to an organization or location. To assign the host to an organization, see [Section 3.9, “Assigning a Host to a Specific Organization”](#) or to assign the host to a location, see [Section 3.10, “Assigning a Host to a Specific Location”](#).

3.5.6. Registering Hosts to Satellite 6 Using The Bootstrap Script

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

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

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

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

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

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

Installing the Bootstrap Script on the Host:

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

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

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

```
# cd
```

2. Download the script:

```
# curl -O http://satellite6.example.com/pub/bootstrap.py
```

This installs the script in the current directory.

3. Make the script executable:

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

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

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

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

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

Required Permissions The user requires certain permissions in order to run the bootstrap script. You can set these permissions using roles through the web UI or using the Hammer command line tool.

Using the web UI to set permissions for the bootstrap script:

1. Navigate to **Administer > Users**.
2. Select an existing user or create a new user especially for the purpose of running this script. Users are located under the **Username** field. A new pane will open with tabs to modify information about the selected user.
3. Click the **Roles** tab.
4. Select **Viewer** and **Edit hosts** from the **Roles** list. The selected roles should appear on the **Selected Items** list confirming that they have been selected.



WARNING

Be aware that the following role **Edit hosts** allows the user to edit and delete hosts as well as being able to add hosts. If this is not acceptable to your security policy, create a new role with the minimal permissions required by the bootstrap script. Create the *bootstrap* role using the Hammer command line tool, see [Using Hammer to set permissions for the bootstrap script](#), then assign this role alone, to the user who will be running the bootstrap script. Alternatively, create an appropriate role using the web UI, navigate to **Administer > Roles**. Subsequently, use the web UI to create a role and set up the appropriate filters.

5. Click **Submit**. The permissions required for the bootstrap script to run will now be set for the user you have specified.

Using Hammer to set permissions for the bootstrap script:

1. Create a role with the minimum permissions required by the bootstrap script. This example creates a role with the name *Bootstrap*. Modify this to provide a more specific name if you wish.

```
$ ROLE='Bootstrap'
$ hammer role create --name "$ROLE"
$ hammer filter create --role "$ROLE" --permissions
view_organizations
$ hammer filter create --role "$ROLE" --permissions view_locations
$ hammer filter create --role "$ROLE" --permissions view_domains
$ hammer filter create --role "$ROLE" --permissions view_hostgroups
$ hammer filter create --role "$ROLE" --permissions view_hosts
$ hammer filter create --role "$ROLE" --permissions
view_architectures
$ hammer filter create --role "$ROLE" --permissions view_ptables
$ hammer filter create --role "$ROLE" --permissions
view_operatingsystems
$ hammer filter create --role "$ROLE" --permissions create_hosts
```

2. Assign the new role to an existing user.

```
$ hammer user add-role --id user_id --role Bootstrap
```

There is also the option to create a new user and assign this new role to them. For more information on how to create users through Hammer see [Creating Users](#) in the Red Hat Satellite Hammer CLI Guide.

Running the Bootstrap Script

Prerequisites

- The bootstrap script is installed as described previously.
- You have an activation key for your hosts. For configuring activation keys, see [Managing Activation Keys](#) in the *Content Management Guide*.
- You have created a host group. For creating host groups, see [Creating a Host Group on Satellite Server](#) in the *Provisioning Guide*.



NOTE

Puppet fails to retrieve the Puppet CA certificate while registering a host with a host group associated with a Puppet environment created inside a **Production** environment. To create a suitable Puppet environment to be associated with a host group, follow these steps:

1. Manually create a directory and change the owner:

```
# mkdir /etc/puppet/environments/example_environment
# chown apache
/etc/puppet/environments/example_environment
```

2. Navigate to **Configure** → **Environments** and click **Import environment from**. The button name will include the FQDN of the internal or external Capsule.
3. Choose the created directory and click **Update**.

To run the bootstrap script:

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



WARNING

The example in this section specifies the admin user. If this is not acceptable to your security policy, create a new role with the minimal permissions required by the bootstrap script that can be added to the appropriate user. This is achieved using the web UI or Hammer, see [Using the web UI to set permissions for the bootstrap script:](#) and [Using Hammer to set permissions for the bootstrap script:](#) for more information.

For the **--server** option, specify the FQDN name of Satellite Server or Capsule Server. For **--location**, **--organization**, and **--hostgroup** options, use quoted names, not labels, as arguments to the options. See [Section 3.5.7, “Advanced Bootstrap Script Configuration”](#) for advanced use cases.

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

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

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

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

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

- a. In the web UI, navigate to **Infrastructure > Capsules**.
 - b. Select **Certificates** to the right of the name of the Capsule corresponding to the FQDN given with **--server** option.
 - c. In the **Actions** column select **Sign** to approve the host's Puppet certificate.
 - d. Return to the host to see the remainder of the bootstrap process completing.
3. In the web UI, navigate to **Hosts > All hosts** and ensure that the host is connected to the correct host group.

If the Katello agent is not installed on the host, proceed to [Section 3.5.3, "Installing the Katello Agent"](#).

3.5.7. Advanced Bootstrap Script Configuration

The standard workflow of using the bootstrap script has been outlined in [Running the Bootstrap Script](#). This section covers some more examples.



WARNING

The examples in this section specify the admin user. If this is not acceptable to your security policy, create a new role with the minimal permissions required by the bootstrap script. See [Using the web UI to set permissions for the bootstrap script:](#) and [Using Hammer to set permissions for the bootstrap script:](#) for more information.

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

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

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

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

The bootstrap script detects the presence of `/etc/syconfig/rhn/systemid` and a valid connection to

RHN as an indicator that the system is registered to a legacy platform. The script then calls **rhnclassic-migrate-to-rhsm** to migrate the system from RHN. By default, the script does not delete the system's legacy profile due to auditing reasons. To remove the legacy profile, use **--legacy-purge** and use **--legacy-login** to supply an user account that has appropriate permissions to remove a profile. Enter the user account password when prompted. For example:

```
# bootstrap.py --login=admin \
--server satelite6.example.com \
--location="Example Location" \
--organization="Example Organization" \
--hostgroup="Example Host Group" \
--activationkey=activation_key \
--legacy-purge \
--legacy-login rhn-user
```

Registering a host to Satellite 6, omitting Puppet setup.

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

```
# bootstrap.py --login=admin \
--server satelite6.example.com \
--location="Example Location" \
--organization="Example Organization" \
--hostgroup="Example Host Group" \
--activationkey=activation_key \
--skip-puppet
```

Registering a host to Satellite 6 for content management only.

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

```
# bootstrap.py --server satelite6.example.com \
--organization="Example Organization" \
--activationkey=activation_key \
--skip-foreman
```

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

By default, the bootstrap script uses HTTP to download the consumer RPM (*server.example.com/pub/katello-ca-consumer-latest.noarch.rpm*). In some environments, you might want to allow HTTPS only between the host and Satellite. Use **--download-method** to change the download method from HTTP to HTTPS. For example:

```
# bootstrap.py --login=admin \
--server satelite6.example.com \
--location="Example Location" \
--organization="Example Organization" \
--hostgroup="Example Host Group" \
--activationkey=activation_key \
--download-method https
```

Providing the host's IP address to Satellite

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

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

Enabling Remote Execution on the host.

Use `--rex` and `--rex-user` to enable remote execution and add the required SSH keys for the specified user. For example:

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

Creating a domain for a host at registration time.

To create a host record, the DNS domain of a host needs to exist in Satellite prior to running script. If the domain does not exist, add it using `--add-domain`. For example:

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

Providing an arbitrary Fully Qualified Domain Name (FQDN) for the host.

If the host's host name is not an FQDN, or is not RFC compliant (containing a character such as an underscore), the script will fail at the host name validation stage. Use `--fqdn` to specify the FQDN that will be reported to Satellite. To do so, you will need to set `create_new_host_when_facts_are_uploaded` and `create_new_host_when_report_is_uploaded` to false using `hammer`. For example:

```
# hammer settings set \
--name create_new_host_when_facts_are_uploaded \
--value false
# hammer settings set \
--name create_new_host_when_report_is_uploaded \
--value false

# bootstrap.py --login=admin \
```

```
--server satellite6.example.com \  
--location="Example Location" \  
--organization="Example Organization" \  
--hostgroup="Example Host Group" \  
--activationkey=activation_key \  
--fqdn node100.example.com
```

3.6. CHANGING THE GROUP OF A HOST

The following steps show you how to change the group of a host.

1. Navigate to **Hosts > All hosts**.
2. Select the check box of the host you want to change.
3. From the **Select Action** menu at the upper right of the screen, select **Change Group**. A new option window will open.
4. From the **Host Group** menu, select the group that you want for your host.
5. Click **Submit**.

3.7. CHANGING THE ENVIRONMENT OF A HOST

The following steps show you how to change the environment of a host.

1. Navigate to **Hosts > All hosts**.
2. Select the check box of the host you want to change.
3. From the **Select Action** menu at the upper right of the screen, select **Change Environment**. A new option window will open.
4. From the **Environment** menu, select the new environment for your host.
5. Click **Submit**.

3.8. MANAGING HOSTS

Hosts provisioned by Satellite are managed by default. When the host is set to managed, it is possible to configure additional host parameters from Satellite Server. These additional parameters are listed on the **Operating System** tab.

If you change any settings on the **Operating System** tab they will not take effect until you set the host to build and reboot it.

If there is a necessity to obtain reports about configuration management in systems using an operating system not supported by Satellite it is recommended to unmanage the host.

The following procedure shows how to switch a host between Managed and Unmanaged status.

1. Navigate to **Hosts > All hosts**.
2. Select the host.

3. Click **Edit**.
4. Click **Manage host** or **Unmanage host** to change the host's status.
5. Click **Submit** to save the changes.

3.9. ASSIGNING A HOST TO A SPECIFIC ORGANIZATION

The following steps show you how to assign a host to a specific organization. For general information about organizations and how to configure them, see [Creating Organizations](#) in the *Content Management Guide*.

1. Navigate to **Hosts > All hosts**.
2. Select the check box of the host you want to change.
3. From the **Select Action** menu at the upper right of the screen, select **Assign Organization**. A new option window will open.
4. Navigate to the **Select Organization** menu and choose the organization that you want to assign for your host. Select the check box **Fix Organization on Mismatch**.



NOTE

A mismatch happens if there is a resource associated with a host, such as a domain or subnet, and at the same time not associated with the organization you want to assign the host to. The option **Fix Organization on Mismatch** will add such a resource to the organization, and is therefore the recommended choice. The option **Fail on Mismatch**, on the other hand, will always result in an error message. For example, reassigning a host from one organization to another will fail, even if there is no actual mismatch in settings.

5. Click **Submit**.

3.10. ASSIGNING A HOST TO A SPECIFIC LOCATION

The following steps show you how to assign a host to a specific location. For general information about locations and how to configure them, see [Creating a Location](#) in the *Provisioning Guide*.

1. Navigate to **Hosts > All hosts**.
2. Select the check box of the host you want to change.
3. From the **Select Action** menu at the upper right of the screen, select **Assign Location**. A new option window will open.
4. Navigate to the **Select Location** menu and choose the location that you want for your host. Select the check box **Fix Location on Mismatch**.



NOTE

A mismatch happens if there is a resource associated with a host, such as a domain or subnet, and at the same time not associated with the organization you want to assign the host to. The option **Fix Organization on Mismatch** will add such a resource to the organization, and is therefore the recommended choice. The option **Fail on Mismatch**, on the other hand, will always result in an error message. For example, reassigning a host from one organization to another will fail, even if there is no actual mismatch in settings.

5. Click **Submit** to complete the assigning of the location to your host.

3.11. CONFIGURING AN ADDITIONAL NETWORK INTERFACE

Red Hat Satellite supports specifying multiple network interfaces for a single host. You can configure these interfaces when creating a new host as described in [Section 3.4, “Creating a Host”](#) or when editing an existing host.

There are several types of network interfaces that you can attach to a host. When adding a new interface, select one of:

- **Interface:** Allows you to specify an additional physical or virtual interface. There are two types of virtual interfaces you can create. Use **VLAN** when the host needs to communicate with several (virtual) networks using a single interface, while these networks are not accessible to each other. Another type of virtual interface is **alias**, which is an additional IP address attached to an existing interface. See [Section 3.11.2, “Adding a Virtual Interface”](#), or [Section 3.11.1, “Adding a Physical Interface”](#) for details.
- **Bond:** Creates a bonded interface. NIC bonding is a way to bind multiple network interfaces together into a single interface that appears as a single device and has a single MAC address. This enables two or more network interfaces to act as one, simultaneously increasing the bandwidth and providing redundancy. See [Section 3.11.3, “Adding a Bonded Interface”](#) for details.
- **BMC:** Baseboard Management Controller (BMC) allows you to remotely monitor and manage physical state of machines. See [Enabling Power Management on Managed Hosts](#) in the *Red Hat Satellite Installation Guide* for more information on BMC, and [Section 3.11.4, “Adding a Baseboard Management Controller \(BMC\) Interface”](#) for details on configuring a BMC interface.



NOTE

Additional interfaces have by default the **Managed** flag enabled, which means the new interface is configured automatically during provisioning by the DNS and DHCP Capsule Servers associated with the selected subnet. This requires a subnet with correctly configured DNS and DHCP Capsule Servers. If you use a kickstart method for host provisioning, configuration files are automatically created for managed interfaces in the post-installation phase at `/etc/sysconfig/network-scripts/ifcfg-$interface_id`.



NOTE

Virtual and bonded interfaces currently require a MAC address of a physical device. Therefore, the configuration of these interfaces works only on bare-metal hosts.

3.11.1. Adding a Physical Interface

The following steps show how to add an additional physical interface to a host.

To Add a Physical Interface:

1. Navigate to **Hosts** > **All hosts** to view available hosts.
2. Click **Edit** next to the host you want to edit.
3. On the **Interfaces** tab, click **Add Interface**.
4. Keep the **Interface** option selected in the **Type** menu.
5. Specify a **MAC address** of the additional interface. This setting is required.
6. Specify the device **Identifier**, for example **eth0** or **eth1.1**. Identifier is used for bonded interfaces (in the **Attached devices** field, see [To Add a Bonded Interface](#)), VLANs and aliases (in the **Attached to** field, see [To Add a Virtual Interface](#)).
7. Specify the **DNS name** associated with the host's IP address. Satellite saves this name in the Capsule Server associated with the selected domain (the "DNS A" field) and the Capsule Server associated with the selected subnet (the "DNS PTR" field). A single host can therefore have several DNS entries.
8. Select a domain from the **Domain** drop-down menu. To create and manage domains, navigate to **Infrastructure** > **Domains**.
9. Select a subnet from the **Subnet** drop-down menu. To create and manage subnets, navigate to **Infrastructure** > **Subnets**.
10. Specify the interface **IP address**. Managed interfaces with assigned DHCP Capsule Server require this setting for creating a DHCP lease. DHCP-enabled managed interfaces provide an automatic suggestion of IP address.
11. Decide if the interface will be managed. If the **Managed** check box is selected, the interface configuration is pulled from the associated Capsule Server during provisioning, and DNS and DHCP entries are created. If using kickstart provisioning, a configuration file is automatically created for the interface.
12. Select the **Virtual NIC** check box to create a virtual interface. See [Section 3.11.2, "Adding a Virtual Interface"](#) for details.
13. Click **OK** to save the interface configuration, and then click **Submit** to apply the changes to the host.

3.11.2. Adding a Virtual Interface

The following procedure shows how to configure an additional virtual interface for a host. This can be either a VLAN or an alias interface.

An alias interface is an additional IP address attached to an existing interface. Note that:

- An alias interface automatically inherits a MAC address from the interface it is attached to, therefore you can create an alias without specifying a MAC address.
- The interface must be specified in a subnet with boot mode set to **static**.

To Add a Virtual Interface:

1. Navigate to **Hosts** > **All hosts** to view available hosts.
2. Click **Edit** next to the host you want to edit.
3. On the **Interfaces** tab, click **Add Interface**.
4. Keep the **Interface** option selected in the **Type** menu.
5. Specify the general interface settings. The applicable configuration options are the same as for the physical interfaces described in [Section 3.11.1, “Adding a Physical Interface”](#).
Specify **MAC address** for managed virtual interfaces so that the configuration files for provisioning are generated correctly. However, **MAC address** is not required for virtual interfaces that are not managed.

If creating a VLAN, specify ID in the form of **eth1.10** in the **Identifier** field. If creating an alias, use ID in the form of **eth1:10**.

6. Select the **Virtual NIC** check box. Additional configuration options specific to virtual interfaces are appended to the form:
 - **Tag**: You can specify tags per interface to provide a higher-level segmentation of the network. If left blank, managed interfaces inherit the tag from the VLAN ID of the associated subnet, given that this subnet has the VLAN ID specified. User-specified entries from this field are not applied on alias interfaces.
 - **Attached to**: Specify the identifier of the physical interface to which the virtual interface belongs, for example eth1. This setting is required.
7. Click **OK** to save the interface configuration. Then click **Submit** to apply the changes to the host.

3.11.3. Adding a Bonded Interface

The following steps show how to configure a bonded interface for a host.

To Add a Bonded Interface:

1. Navigate to **Hosts** > **All hosts** to view available hosts.
2. Click **Edit** next to the host you want to edit.
3. On the **Interfaces** tab, click **Add Interface**.
4. Select **Bond** from the **Type** menu. Additional type-specific configuration options are appended to the form.
5. Specify the general interface settings. The applicable configuration options are the same as for the physical interfaces described in [Section 3.11.1, “Adding a Physical Interface”](#). Bonded interfaces use IDs in the form of **bond0** in the **Identifier** field. It is sufficient if you specify a single MAC address in the **MAC address** field.
6. Specify the configuration options specific to bonded interfaces:
 - **Mode**: Select the bonding mode that defines a policy for fault tolerance and load balancing. See [Table 3.2, “Bonding Modes Available in Red Hat Satellite”](#) for a brief description of individual bonding modes.

- **Attached devices:** Specify a comma separated list of identifiers of attached devices. These can be physical interfaces or VLANs.
- **Bond options:** Specify a space separated list of configuration options, for example **miimon=100**. There are several configuration options you can specify for the bonded interface, see [Red Hat Enterprise Linux 7 Networking Guide](#) for details.

7. Click **OK** to save the interface configuration. Then click **Submit** to apply the changes to the host.

Table 3.2. Bonding Modes Available in Red Hat Satellite

Bonding Mode	Description
balance-rr	Transmissions are received and sent out sequentially on each bonded interface.
active-backup	Transmissions are received and sent out via the first available bonded interface. Another bonded interface is only used if the active bonded interface fails.
balance-xor	Transmissions are based on the selected hash policy. In this mode, traffic destined for specific peers will always be sent over the same interface.
broadcast	All transmissions are sent on all bonded interfaces.
802.a3	Creates aggregation groups that share the same settings. Transmits and receives on all interfaces in the active group.
balance-tlb	The outgoing traffic is distributed according to the current load on each bonded interface.
balance-alb	Receive load balancing is achieved through Address Resolution Protocol (ARP) negotiation.

3.11.4. Adding a Baseboard Management Controller (BMC) Interface

This section describes how to configure a baseboard management controller (BMC) interface for a host that supports this feature.

Prerequisites

Ensure the following prerequisites are satisfied before proceeding:

- BMC is enabled on the Capsule Server. If required, see [To Enable BMC Power Management on an Existing Capsule Server](#).
- The **ipmitool** package is installed.
- You know the MAC address, IP address, and other details of the BMC interface on the host, and the appropriate credentials for that interface.

**NOTE**

You only need the MAC address for the BMC interface if the BMC interface is managed. This is so that it can create a DHCP reservation.

To Enable BMC Power Management on an Existing Capsule Server:

1. Use the **satellite-installer** routine to configure BMC power management on the Capsule Server by running the following command with the following options:

```
# satellite-installer --foreman-proxy-bmc=true \ --foreman-proxy-bmc-default-provider=ipmitool
```

2. Refresh the features for the Capsule Server.
 - a. Log in to the Satellite web UI, and navigate to **Infrastructure > Capsules**.
 - b. Identify the Capsule Server whose features you need to refresh. In the drop-down list on the right, click **Refresh**. The list of features in the **Features** column should now include BMC.

To Add a BMC Interface:

1. Navigate to **Hosts > All hosts** to view available hosts.
2. Click **Edit** next to the host you want to edit.
3. On the **Interfaces** tab, click **Add Interface**.
4. Select **BMC** from the **Type** menu. Type-specific configuration options are appended to the form.
5. Specify the general interface settings. The applicable configuration options are the same as for the physical interfaces described in [Section 3.11.1, “Adding a Physical Interface”](#).
6. Specify the configuration options specific to BMC interfaces:
 - **Username, Password:** Specify any authentication credentials required by BMC.
 - **Provider:** Specify the BMC provider.
7. Click **OK** to save the interface configuration, and then click **Submit** to apply the changes to the host.

3.12. REMOVING A HOST

The following procedure shows how to remove a host from Red Hat Satellite.

To Remove a Host:

1. Click **Hosts > All hosts** or **Hosts > Content Hosts**.
2. Select the hosts that you want to remove.
3. Click **Select Action** and select **Delete Hosts** from the drop-down menu.
4. Click **Submit** to remove the host from Red Hat Satellite permanently.

**WARNING**

If a host record that is associated with a virtual machine is deleted, the virtual machine will be deleted as well. To avoid deleting the virtual machine in this situation, disassociate the virtual machine from Satellite without removing it from the hypervisor.

To Disassociate A Virtual Machine from Satellite without Removing it from a Hypervisor

1. In the Satellite web UI, navigate to **Hosts > All Hosts** and select the check box to the left of the hosts to be disassociated.
2. From the **Select Action** drop-down menu, select the **Disassociate Hosts** button.
3. In the confirmation window:
 - a. Optionally, select the check box to keep the hosts for future action.
 - b. Click **Submit** to save your changes.

CHAPTER 4. CONFIGURING HOST COLLECTIONS

A host collection is a group of multiple content hosts. This feature enables you to perform the same action on multiple hosts at once. These actions can include the installation, removal, and update of packages and errata, change of assigned life cycle environment, and change of Content View. You can create host collections to suit your requirements, and those of your company. For example, group hosts in host collections by function, department, or business unit.

4.1. CREATING A HOST COLLECTION

The following procedure shows how to create host collections.

To Create a Host Collection:

1. Click **Hosts > Host Collections**.
2. Click **New Host Collection**.
3. Add the Name of the host collection.
4. Clear **Unlimited Content Hosts**, and enter the desired maximum number of hosts in the **Limit** field.
5. Add the Description of the host collection.
6. Click **Save**.

4.2. CLONING A HOST COLLECTION

The following procedure shows how to clone a host collection.

To Clone a Host Collection:

1. Click **Hosts > Host Collections**.
2. On the left hand panel, click the host collection you want to clone.
3. Click **Copy Collection**.
4. Specify a name for the cloned collection.
5. Click **Create**.

4.3. REMOVING A HOST COLLECTION

The following procedure shows how to remove a host collection.

To Remove a Host Collection:

1. Click **Hosts > Host Collections**.
2. Choose the host collection to be removed.
3. Click **Remove**. An alert box appears:

Are you sure you want to remove host collection *Host Collection Name*?

4. Click **Remove**.

4.4. ADDING A HOST TO A HOST COLLECTION

The following procedure shows how to add hosts to host collections.

Prerequisites

A host must be registered to Red Hat Satellite in order to add it to a Host Collection. Refer to [Section 3.5, “Registration”](#) for information on how to register a host.

To Add Hosts to a Host Collection:

1. Click **Hosts > Host Collections**.
2. Click the host collection where the host should be added.
3. On the **Hosts** tab, select the **Add** subtab.
4. Select the hosts to be added from the table and click **Add Selected**.

4.5. REMOVING A HOST FROM A HOST COLLECTION

The following procedure shows how to remove hosts from host collections.

To Remove Hosts from a Host Collection:

1. Click **Hosts > Host Collections**.
2. Choose the desired host collection.
3. On the **Hosts** tab, select the **List/Remove** subtab.
4. Select the hosts you want to remove from the host collection and click **Remove Selected**.

4.6. ADDING CONTENT TO A HOST COLLECTION

These steps show how to add content to host collections in Red Hat Satellite.

4.6.1. Adding Packages to a Host Collection

The following procedure shows how to add packages to host collections.

Prerequisites

- The content to be added should be available in one of the existing repositories or added prior to this procedure.
- Content should be promoted to the environment where the hosts are assigned.

To Add Packages to Host Collections:

1. Click **Hosts > Host Collections**.
2. Click the host collection where the package should be added.
3. On the **Collection Actions** tab, click **Package Installation, Removal, and Update**.
4. To update all packages, click the **Update All Packages** button to use the default method. Alternatively, select the drop-down icon to the right of the button to select a method to use. Selecting the **via remote execution - customize first** menu entry will take you to the **Job invocation** page where you can customize the action.
5. Select the **Package** or **Package Group** radio button as required.
6. In the field provided, specify the package or package group name. Then click:
 - **Install** — to install a new package using the default method. Alternatively, select the drop-down icon to the right of the button and select a method to use. Selecting the **via remote execution - customize first** menu entry will take you to the **Job invocation** page where you can customize the action.
 - **Update** — to update an existing package in the host collection using the default method. Alternatively, select the drop-down icon to the right of the button and select a method to use. Selecting the **via remote execution - customize first** menu entry will take you to the **Job invocation** page where you can customize the action.

4.6.2. Adding Errata to a Host Collection

The following procedure shows how to add errata to host collections.

Prerequisites

- The errata to be added should be available in one of the existing repositories or added prior to this procedure.
- Errata should be promoted to the environment where the hosts are assigned.

To Add Errata to a Host Collection:

1. Click **Hosts > Host Collections**.
2. Select the host collection where the errata should be added.
3. On the **Collection Actions** tab, click **Errata Installation**.
4. Select the errata you want to add to the host collection and click the **Install Selected** button to use the default method. Alternatively, select the drop-down icon to the right of the button to select a method to use. Selecting the **via remote execution - customize first** menu entry will take you to the **Job invocation** page where you can customize the action.

4.7. REMOVING CONTENT FROM A HOST COLLECTION

The following procedure shows how to remove packages from host collections.

To Remove Content from a Host Collection:

1. Click **Hosts > Host Collections**.
2. Click the host collection where the package should be removed.
3. On the **Collection Actions** tab, click **Package Installation, Removal, and Update**.
4. Select the **Package** or **Package Group** radio button as required.
5. In the field provided, specify the package or package group name.
6. Click the **Remove** button to remove the package or package group using the default method. Alternatively, select the drop-down icon to the right of the button and select a method to use. Selecting the **via remote execution - customize first** menu entry will take you to the **Job invocation** page where you can customize the action.

4.8. CHANGING THE LIFE CYCLE ENVIRONMENT OR CONTENT VIEW OF A HOST COLLECTION

The following procedure shows how to change the assigned life cycle environment or Content View of host collections.

To Change the Life Cycle Environment or Content View of a Host Collection:

1. Click **Hosts > Host Collection**.
2. Selection the host collection where the life cycle environment or Content View should be changed.
3. On the **Collection Actions** tab, click **Change assigned Life Cycle Environment or Content View**.
4. Select the life cycle environment to be assigned to the host collection.
5. Select the required Content View from the drop-down list.
6. Click **Assign**.



NOTE

The changes take effect in approximately 4 hours. To make the changes take effect immediately, on the host, enter the following command:

```
# subscription-manager refresh
```

You can use remote execution to run this command on multiple hosts at the same time.

CHAPTER 5. RUNNING JOBS ON HOSTS

Red Hat Satellite supports the ability to run arbitrary commands on hosts. This is referred to as remote execution. Remote execution is enabled by default on the Satellite Server, but must be enabled manually on all desired Capsule Servers. Communication occurs through the Capsule Server which means that the Satellite Server does not require direct access to the target host, and can scale to control many hosts. Remote execution uses the SSH service which must be enabled and running on the target host. Ensure the Capsule has access to port 22 on the target hosts.

Commands can be customized in a similar fashion to provisioning templates or partition tables. Several job templates are included by default, that you can use to run commands. See [Section 5.3.1, “Setting up Job Templates”](#).



NOTE

Any Capsule Server’s base system is a client of Satellite Server’s internal Capsule, and therefore this section applies to any type of host connected to Satellite Server, including Capsule Servers.

You can execute commands on multiple hosts at once, and you can use variables in your commands to suit your deployment. Variable values can be filled by host fact, Smart Class Parameter, Smart Variable, or even host parameter. In addition, you can specify custom values for templates when you run the command. See [Section 5.3.2, “Executing Jobs”](#).

The following list provides some examples of how you can use remote execution:

- Install, update, or remove software packages
- Bootstrap a configuration management agent
- Trigger a Puppet, Salt, or Chef run

By default, each Capsule is installed with the remote execution feature disabled. To use remote execution on a Capsule Server you need to enable it. To enable, run the following command:

```
# satellite-installer --scenario capsule \
--enable-foreman-proxy-plugin-remote-execution-ssh
```

To verify that remote execution is running on the Capsule Server and in the web UI navigate to **Infrastructure > Capsules**. The Capsule Server should now list in the **Features** column that **SSH** is running.

By default, Satellite Server is configured to use remote execution rather than Katello Agent. If required, these settings can be changed by first creating custom job templates and then selecting these new templates in the web UI by going to **Administer > Remote Execution Features**. For each action you want to change, select the label and then select the job template to use.

5.1. ESTABLISHING A SECURE CONNECTION FOR REMOTE COMMANDS

The SSH keys used for remote execution are created automatically when installing a Capsule and the settings are in the `/etc/foreman-proxy/settings.d/remote_execution_ssh.yml` file. They include the following options:

ssh_identity_file

File to load the SSH key from. By default, set to **/usr/share/foreman-proxy/.ssh/id_rsa_foreman_proxy**.

local_working_dir

Directory used on the Satellite or Capsule to run the scripts necessary for remote execution. By default, set to **/var/tmp**.

remote_working_dir

Directory on the client system that is used to execute the remote execution jobs. By default, set to **/var/tmp**.

**NOTE**

If the client system has **noexec** set for the **/var/** volume or file system, change the **remote_working_dir** as otherwise the remote execution job will fail since the script cannot be executed.

If required to use an alternative directory, create the new directory, for example *new_place*, and then copy the SELinux context from the default directory. For example:

```
# chcon --reference=/var new_place
```

See [Maintaining SELinux Labels](#) in the *SELinux User's and Administrator's Guide* for more information on working with SELinux labels.

Distributing the SSH Keys for Remote Execution

To enable remote execution, distribute the public SSH key from a Capsule to the hosts that you want to manage. Ensure the SSH service is enabled and running on the hosts. Configure any network or host-based firewalls to enable access to port 22.

There are three ways to distribute the public key from a Capsule to target hosts:

- To distribute keys manually, execute the following command on the Capsule:

```
# ssh-copy-id -i ~foreman-proxy/.ssh/id_rsa_foreman_proxy.pub  
root@target.example.com
```

Here *target.example.com* is the host name of the target host. Repeat for each target host you want to manage.

To confirm the key was successfully copied to the target host, execute the following command on the Capsule:

```
# ssh -i ~foreman-proxy/.ssh/id_rsa_foreman_proxy  
root@target.example.com
```

- To use the Satellite API to download the public key directly from the Capsule, execute the following command on each target host:

```
# curl https://myproxy.example.com:9090/ssh/pubkey >>  
~/.ssh/authorized_keys
```

Here *myproxy.example.com* stands for the host name of the Capsule.

- To include the public key in newly-provisioned hosts, modify for example the **Kickstart default finish** template to include the following line:

```
<%= snippet 'remote_execution_ssh_keys' %>
```

5.2. SETTING UP KERBEROS AUTHENTICATION FOR REMOTE EXECUTION

From Satellite 6.3, you can use Kerberos authentication to establish an SSH connection for remote execution on Satellite hosts.

Prerequisites

Before you can use Kerberos authentication for remote execution on Red Hat Satellite, you must set up a Kerberos server for identity management and ensure that you complete the following prerequisites:

- Enroll Satellite Server on the Kerberos server
- Enroll the Satellite target host on the Kerberos server
- Configure and initialize a Kerberos user account for remote execution
- Ensure that the **foreman-proxy** user on Satellite has a valid Kerberos ticket granting ticket

To set up Satellite to use Kerberos authentication for remote execution on hosts, complete the following steps:

1. Before installing the **tfm-rubygem-net-ssh-krb** package, you must temporarily set SELinux to **permissive** until [Red Hat bug 1541481](#) is resolved:

```
# setenforce 0
```

2. To install the **tfm-rubygem-net-ssh-krb** package, enter the following command:

```
# yum install tfm-rubygem-net-ssh-krb
```

3. To install and enable Kerberos authentication for remote execution, enter the following command:

```
# satellite-installer --scenario satellite \
  --foreman-proxy-plugin-remote-execution-ssh-ssh-kerberos-auth true
```

4. Set SELinux to **enforcing**:

```
# setenforce 1
```

5. To edit the default user for remote execution, in the Satellite web UI, navigate to **Administer > Settings** and click the **RemoteExecution** tab. In the **remote_execution_ssh_user** row, edit the second column and add the user name for the Kerberos account.
6. Navigate to **remote_execution_effective_user** and edit the second column to add the user name for the Kerberos account.

To confirm that Kerberos authentication is ready to use, run a remote job on the host.

5.3. CONFIGURING AND RUNNING REMOTE COMMANDS

Any command that you want to execute on a remote host must be defined as a job template. After you have defined a job template you can execute it multiple times.

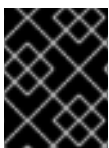
5.3.1. Setting up Job Templates

Satellite provides a number of default job templates that you can use for executing jobs, find them under **Hosts > Job templates**. If you find a template fitting your needs amongst the default templates, proceed to [Section 5.3.2, “Executing Jobs”](#).

You can also use default templates as a basis for developing your own. Default job templates are locked for editing, therefore you have to first clone the template to be able to modify it. Job templates use the Embedded Ruby (ERB) syntax, for more information see [Appendix A, Template Writing Reference](#).

To Create a Job Template:

1. Navigate to **Hosts > Job templates**.
2. Click **New Job Template**. As an alternative, you can modify an existing template – in the **Actions** column, select **Clone** from the drop-down menu.
3. Configure the job template:
 - a. On the **Template** tab, enter a unique name for your job template. Select **Default** to make the template available for all organizations and locations. You can insert the template manually using **Template editor** or upload it from a text file by clicking **Browse**. Templates use Embedded Ruby (ERB) template syntax, see [Section 5.3.4, “Creating Advanced Templates”](#) for more information. An advanced template is required, for example, for executing jobs that perform power actions; see [Example 5.4, “Including Power Actions in Templates”](#) for information on how to include the **Power Action - SSH Default** template in a custom template.
 - b. On the **Job** tab, you can define the job category (define your own or select from the default categories listed in [Table 5.1, “Default Job Template Categories”](#)) as well as the effective user; these settings can be configured also when invoking the job (see [To Execute a Remote Job](#)). You can also define input parameters for template commands. These parameters are then requested when executing the job.
 - c. Remaining tabs enable setting the template type, organizations and locations as well as viewing the template history.
4. Click **Submit**. When the page refreshes, your new template should appear in the list of job templates.



IMPORTANT

Note that only the parameters visible on the **Parameters** tab at the host’s edit page can be used as input parameters for job templates.

Table 5.1. Default Job Template Categories

Job template category	Description
Packages	Templates for performing package related actions. Install, update, and remove actions are included by default.
Puppet	Templates for executing Puppet runs on target hosts.
Power	Templates for performing power related actions. Restart and shutdown actions are included by default.
Commands	Templates for executing custom commands on remote hosts.
Services	Templates for performing service related actions. Start, stop, restart, and status actions are included by default.
Katello	Templates for performing content related actions. These templates are used mainly from different parts of the Satellite web UI (for example bulk actions UI for content hosts), but can be used separately to perform operations such as errata installation.

Example 5.1. Creating a restorecon Template

This example shows how to create a template called **Run Command - restorecon** that will restore the default **SELinux** context for all files in the selected directory on target hosts.

1. Navigate to **Hosts > Job templates**. Click **New Job Template**.
2. Insert **Run Command - restorecon** in the **Name** field. Select **Default** to make the template available to all organizations. Add the following text to the **Template editor**:

```
restorecon -RvF <%= input("directory") %>
```

The `<%= input("directory") %>` string will be replaced by a user-defined directory during job invocation.

3. On the **Job** tab, perform the following actions:
 - a. Set **Job category** to **Commands**.
 - b. Click **Add Input** to allow job customization. Insert **directory** to the **Name** field. The input name must match the value specified in the **Template editor**.
 - c. Click **Required** so that the command cannot be executed without the user specified parameter.
 - d. Select **User input** from the **Input type** drop-down list. Also provide a **Description** to be shown during job invocation, for example **Target directory for restorecon**.

4. Click **Submit**.

See [Example 5.2, “Executing a restorecon Template on Multiple Hosts”](#) for information on how to execute a job based on this template.

5.3.2. Executing Jobs

This section shows how to run a job based on a job template against one or more hosts.

To Execute a Remote Job:

1. Navigate to **Hosts > All hosts** and select the target hosts for your job. You can use the search field to narrow down the host list.
2. From the **Select Action** menu at the upper right of the screen select **Schedule Remote Job**. This will take you to the **Job invocation** page. Alternatively, if you target just one host, click its name and click **Schedule Remote Job** on the host information page. Note that you can invoke jobs also from the **Job Templates** page by using the **Run** button.
3. On the **Job invocation** page, define the main job settings:
 - a. Select the **Job category** and the **Job template** you want to use. These settings are required.
 - b. Optionally, select a stored search string in the **Bookmark** list to specify the target hosts.
 - c. Optionally, further limit the targeted hosts by inserting a **Search query**. The **Resolves to** line displays the number of hosts affected by your query. Use the refresh button to recalculate the number after changing the query. The preview icon will list the targeted hosts.
 - d. The remaining settings depend on the selected job template. See [To Create a Job Template](#): for information on adding custom parameters to a template.
4. Clicking **Display advanced fields** will show advanced setting for the job. Some of the advanced settings depend on the job template, the following settings are general:
 - **Effective user** defines the user for executing the job, by default it is the SSH user.
 - **Concurrency level** defines maximum number of jobs executed at once, which can prevent overload of systems' resources in a case of executing the job on a large number of hosts.
 - **Time span** defines time interval in seconds after which the job should be killed, if it is not finished already. A task which could not be started during the defined interval, for example, if the previous task took too long to finish, is canceled.
 - **Type of query** defines when the search query is evaluated. This helps to keep the query up to date for scheduled tasks.

Concurrency level and **Time span** settings enable you to tailor job execution to fit your infrastructure hardware and needs.
5. If you want to execute the job immediately, ensure that **Schedule** is set to **Execute now**. You can also define a one-time future job, or set up a recurring job. For recurring tasks, you can define start and end dates, number and frequency of runs. You can also use cron syntax to define repetition. For more information about cron, see the [Automating System Tasks](#) section of the Red Hat Enterprise Linux 7 *System Administrator's Guide*.

- Click **Submit**. This displays the **Job Overview** page, and when the job completes, also displays the status of the job.

Example 5.2. Executing a restorecon Template on Multiple Hosts

This example shows how to execute a job based on the template created in [Example 5.1, “Creating a restorecon Template”](#) on multiple hosts. The job will restore the SELinux context in all files under the **/home/** directory.

- Navigate to **Hosts > All hosts** and select target hosts. Select **Schedule Remote Job** from the **Select Action** drop-down list.
- In the **Job invocation** page, select the **Commands** job category and the **Run Command - restorecon** job template.
- Type **/home** in the **directory** field.
- Set **Schedule** to **Execute now**.
- Click **Submit**. You are taken to the **Job invocation** page where you can monitor the status of job execution.

5.3.3. Monitoring Jobs

You can monitor the progress of the job while it is running. This can help in any troubleshooting that may be required.

To Monitor a Job:

- Navigate to the Job page. This page is automatically displayed if you triggered the job with the **Execute now** setting. To monitor scheduled jobs, navigate to **Monitor > Jobs** and select the job run you wish to inspect.
- On the Job page, click the **Hosts** tab. This displays the list of hosts on which the job is running.
- In the **Host** column, click the name of the host that you want to inspect. This displays the **Detail of Commands** page where you can monitor the job execution in real time.
- Click **Back to Job** at any time to return to the **Job Details** page.

5.3.4. Creating Advanced Templates

When creating a job template, you can import an existing template in the **Template editor** field – this is referred to as rendering. This way you can combine templates, or create more specific templates from the general ones.

The following template combines default templates to install and start the **httpd** service on Red Hat Enterprise Linux systems:

```
<%= render_template 'Package Action - SSH Default', :action => 'install',
:package => 'httpd' %>
<%= render_template 'Service Action - SSH Default', :action => 'start',
:service_name => 'httpd' %>
```

The above template specifies parameter values for the rendered template directly. It is also possible to use the **input()** method to allow users to define input for the rendered template on job execution. For example, you can use the following syntax:

```
<%= render_template 'Package Action - SSH Default', :action => 'install',
:package => input("package") %>
```

With the above template, you have to import the parameter definition from the rendered template. To do so, navigate to the **Jobs** tab, click **Add Foreign Input Set**, and select the rendered template from the **Target template** drop-down list. You can import all parameters or specify a comma separated list.

Example 5.3. Rendering a restorecon Template

This example shows how to create a template derived from the **Run command - restorecon** template created in [Example 5.1, “Creating a restorecon Template”](#). This template does not require user input on job execution, it will restore the SELinux context in all files under the **/home/** directory on target hosts.

Create a new template as described in [Section 5.3.1, “Setting up Job Templates”](#), and specify the following string in the **Template editor** screen:

```
<%= render_template("Run Command - restorecon", :directory => "/home")
%>
```

Example 5.4. Including Power Actions in Templates

This example shows how to set up a job template for performing power actions, such as reboot. This procedure prevents Satellite from interpreting the disconnect exception upon reboot as an error, and consequently, remote execution of the job works correctly.

Create a new template as described in [Section 5.3.1, “Setting up Job Templates”](#), and specify the following string in the **Template editor** screen:

```
<%= render_template("Power Action - SSH Default", :action => "restart")
%>
```

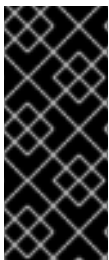
5.4. CONFIGURING GLOBAL SETTINGS

The Satellite remote execution feature provides numerous global settings that you can use to configure its behavior. These are listed in [Table 5.2, “Global Settings for Remote Execution”](#). To review and update these settings, navigate to **Administer > Settings** and click the **Remote Execution** tab.

Table 5.2. Global Settings for Remote Execution

Parameter Name	Description
----------------	-------------

Parameter Name	Description
<code>remote_execution_effective_user</code>	This is the default effective user for any job. When the job is executed the effective user of the process is changed accordingly (for example, by <code>sudo</code>). This option can be overridden per job template and job invocation.
<code>remote_execution_effective_user_method</code>	Specifies which method to use to set the effective user on the target host. Currently only <code>su</code> and <code>sudo</code> are supported.
<code>remote_execution_fallback_proxy</code>	Search the host for any Capsule with remote execution configured. This is useful when the host has no subnet or if the subnet does not have a Capsule with remote execution enabled.
<code>remote_execution_global_proxy</code>	Search for a remote execution Capsule outside of the Capsules assigned to the host. If Locations or Organizations are enabled, the search will be limited to the host's Organization or Location.
<code>remote_execution_ssh_user</code>	<p>The default user to use while the Capsule connects to the target using SSH. You can set the <code>remote_execution_ssh_user</code> variable to override this on a per-host basis.</p> <p>You can set this by Host, Host Group, Operating System, Domain, Location, or Organization. This can also be a different user from the <code>remote_execution_effective_user</code>.</p>
<code>remote_execution_sync_templates</code>	Defines whether job templates should be synchronized from disk when seeding a database.



IMPORTANT

It is possible to set global parameters in the `/etc/foreman/settings.yaml` configuration file, but any manual changes that you make to this file are overwritten the next time you run **`satellite-installer`**. Consequently, Red Hat recommends that you modify these parameters in the web UI. Alternatively, use the **`foreman-rake config`** command from a console.

5.4.1. Choosing a Capsule for Remote Execution

Remote execution requires a Capsule Server to perform any specified job on a host. By default, any Capsule within the host's organization and location with the **`remote execution provider`** feature enabled is considered available to perform these jobs. You can set the **`remote_execution_global_proxy`** variable to **`false`** to disable this behavior. This may be necessary in more complex environments, where not all Capsules can be used due to possible network isolation. In this configuration, you can assign a pool of Capsules to each subnet, and jobs are load balanced across them.

Alternatively, you can set the `remote_execution_fallback_proxy` variable to `true` to enable fallback mode. In this configuration, remote execution will use any Capsule associated with the host, such as its Puppet Master, provided that Capsule also has remote execution configured.

5.5. DELEGATING PERMISSIONS FOR REMOTE EXECUTION

You can control which users can run which jobs within your infrastructure, including which hosts they can target. The remote execution feature provides two built-in roles:

- **Remote Execution Manager:** This role allows access to all remote execution features and functionality.
- **Remote Execution User:** This role only allows running jobs; it does not provide permission to modify job templates.

You can clone the Remote Execution User role and customize its filter for increased granularity. If you adjust the filter with the `view_job_templates` permission, the user can only see and trigger jobs based on matching job templates. You can use the `view_hosts` and `view_smart_proxies` permissions to limit which hosts or Capsules are visible to the role.

The `execute_template_invocation` permission is a special permission that is checked immediately before execution of a job begins. This permission defines which job template you can run on a particular host. This allows for even more granularity when specifying permissions. For more information on working with roles and permissions see [Creating and Managing Roles](#) in the *Administering Red Hat Satellite*.

The following example shows filters for the `execute_template_invocation` permission:

```
name = Reboot and host.name = staging.example.com
name = Reboot and host.name ~ *.staging.example.com
name = "Restart service" and host_group.name = webserver
```

The first line in the above example permits the user to execute the **Reboot** template on one selected host. The second line defines a pool of hosts with names ending with `.staging.example.com`. The third line binds the template with a host group.



NOTE

Permissions assigned to users can change over time. If a user has already scheduled some jobs to run in the future, and the permissions have changed, this can result in execution failure because the permissions are checked immediately before job execution.

CHAPTER 6. DISCOVERING BARE-METAL HOSTS ON SATELLITE

Red Hat Satellite 6.3 includes the Discovery plug-in. The Discovery plug-in enables automatic bare-metal discovery of unknown hosts on the provisioning network. These new hosts are registered to the Satellite Server and the Puppet agent on the client uploads system facts collected by Facter, such as serial ID, network interface, memory, and disk information. After registration, the hosts are displayed on the **Discovered Hosts** page in the Satellite web UI. You can then initiate provisioning either manually (using the web UI, CLI, or API) or automatically, using predefined discovery rules.

The Discovery plug-in communicates through the Satellite Capsule Server, which has direct access both to the provisioning network and the Satellite Server instance. It is possible to discover hosts directly from the Satellite Server, but Red Hat recommends the following scheme be used:

```
Satellite Server (Satellite Server Discovery plug-in) <--> Satellite Capsule (Satellite Capsule Discovery plug-in) <--> Discovered Host (Satellite Discovery image)
```

The Satellite Discovery plug-in consists of three different components:

The Satellite Server Discovery plug-in

This runs on the Satellite Server and provides API and UI functionality for working with discovered hosts. The **tfm-rubygem-foreman_discovery** package contains this plug-in.

The Satellite Capsule Server Discovery plug-in

This is a communication proxy between discovered hosts on a provisioning network and the Satellite Server. The **rubygem-smart_proxy_discovery** package contains this plug-in.

The Satellite Discovery image

This is the minimal operating system based on Red Hat Enterprise Linux that is PXE-booted on hosts to acquire initial hardware information and to check in to the Satellite Server. Discovered hosts keep running the Satellite Discovery image until they are rebooted into Anaconda, which then initiates the provisioning process. The **foreman-discovery-image** package contains this image. It must be installed on the Satellite Capsule Server that provides TFTP services.

6.1. NETWORK CONFIGURATION FOR PXE-BASED DISCOVERY

The discovery process is based on PXE: Systems must boot from the network using a single Ethernet connection to the LAN or VLAN. All other network interface configurations are not supported (bonding, teaming, bridging, DSL, Wi-Fi and others).

You must have a separate LAN or VLAN for discovery and PXE provisioning. You can configure systems to use VLAN trunks, but you must also configure the provisioning interface with the correct VLAN tag for the provisioning VLAN, and then change the tag to the production VLAN using a post-installation script.

Although using special network configurations is technically possible in PXE-less mode, where discovered systems use kexec to load a new kernel with Anaconda which avoids PXE booting completely, the discovery image currently does not allow such a configuration. While it is possible to use discovery extensions or a script to re-configure the network, Satellite 6 discovery plug-in cannot work with such a configuration.

Because the discovery process currently has limited possibilities for configuring network interfaces, and because the provisioning interface is also the primary interface, to simplify the configuration, have a

separate primary interface from the interface used in production. Satellite 6 template features can be used to deploy post-installation scripts to configure interfaces if required.

6.2. CONFIGURING THE SATELLITE DISCOVERY PLUG-IN

The following sections describe how to configure the Satellite Discovery plug-in and how to prepare the PXE-boot template on the Satellite Server.

6.2.1. Deploying the Satellite Discovery Image

Install the package containing the Satellite Discovery image on the Satellite Capsule Server that provides TFTP services (not on the Satellite Server itself):

```
# yum install foreman-discovery-image
```

This package contains the Linux kernel and initial RAM disk image as a bootable ISO file which is used for PXE-booting discovered hosts. You can run the following command to investigate the contents of the package. This produces output similar to the following:

```
$ rpm -ql foreman-discovery-image
/usr/share/foreman-discovery-image
/usr/share/foreman-discovery-image/fdi-image-rhel_7-2.1.0-20150212.1.iso
```

When you install this package, it extracts the kernel and image from the ISO file into the TFTP directory and creates symbolic links to the latest versions of the image and kernel. Use the symbolic links in the PXE-boot provisioning template to make sure that you do not need to change the version in the template every time the **foreman-discovery-image** package is upgraded. For example:

```
$ find /var/lib/tftpboot/boot
/var/lib/tftpboot/boot
/var/lib/tftpboot/boot/fdi-image-rhel_7-2.1.0-20150212.1-img
/var/lib/tftpboot/boot/fdi-image-rhel_7-2.1.0-20150212.1-vmlinuz
/var/lib/tftpboot/boot/fdi-image-rhel_7-img
/var/lib/tftpboot/boot/fdi-image-rhel_7-vmlinuz
```



NOTE

Currently, only Red Hat Enterprise Linux 7 Discovery images are provided, even for Satellite 6 installations on Red Hat Enterprise Linux 6. If there are discovered hosts running during the upgrade of the **foreman-discovery-image** package, reboot them all to load the updated version of the image as soon as possible. This can be done through the Satellite 6 web UI, CLI, or API.

6.2.2. Configuring PXE-booting

When an unknown host is booted on the provisioning network, Satellite Server provides a PXELinux boot menu with a single option: to boot from the local hard drive. You can use following procedure to build a default PXE template in Satellite to enable hardware discovery.

To Configure PXE-booting for host discovery:

1. In the Satellite web UI, navigate to **Hosts > Provisioning Templates**.

2. In the upper-right of the **Provisioning Templates** page, click **Build PXE Default**, and click **OK**.

The template becomes the default template on all TFTP servers. Every new unknown host that is in the provisioning subnet uses this configuration and uses the Foreman Discovery Image as the default.

6.2.3. Reviewing Global Discovery Settings

You can review global settings related to the Discovery plug-in in the Satellite web UI. Navigate to **Administer > Settings** and open the **Discovered** tab. Notable settings are:

Discovery organization, Discovery location

These variables specify where to place the discovered hosts. By default, the discovered hosts are automatically placed under the first organization and location created.

Interface fact

This variable specifies which incoming fact to use to determine the MAC address of the discovered host. By default, the PXELinux BOOTIF kernel command line option is used.

Hostname facts

This variable allows you to list facts to use for the host name. These are separated by commas, and the first fact in the list takes precedence.

Auto provisioning

This variable enables automatic provisioning according to specified rules. Set to false by default. Red Hat recommends that you test the configuration with manual provisioning before enabling Auto provisioning. See [Section 6.4, “Provisioning Discovered Hosts”](#) for more information.

Reboot

This variable enables automatic reboot of a host discovered by PXE, or the use of kexec for a host booted from local media, during provisioning. This is set to true by default.

Hostname prefix

This variable specifies the default prefix to use for the host name. Set to "mac" by default. The variable must start with a letter.

Fact columns

This variable allows you to add any fact reported by Facter as an additional column in discovered host lists.

Highlighted facts

This variable uses regular expressions to organize facts for the highlights section.

Storage facts

This variable uses regular expressions to organize facts for the storage section.

Hardware facts

This variable uses regular expressions to organize facts for the hardware section.

Network facts

This variable uses regular expressions to organize facts for the network section.

IPMI facts

This variable uses regular expressions to organize facts for the IPMI section.

6.3. CONFIGURING THE SATELLITE CAPSULE SERVER DISCOVERY PLUG-IN

Ensure the `foreman_url` setting exists in the Satellite Capsule Server configuration file. The setting can appear as follows:

```
# grep foreman_url /etc/foreman-proxy/settings.yml
:foreman_url: https://satellite.example.com
```

The **satellite-installer** command configures this variable automatically, but Red Hat recommends that you check that the host responds correctly and there are no firewall rules blocking communication.

6.3.1. Configuring Discovery Subnets

You need to configure all subnets with discovered hosts to communicate through the Satellite Capsule Server. In the Satellite web UI, navigate to **Infrastructure > Subnets** and select the required Capsule Server for each subnet that needs to perform host discovery and ensure it is connected to the Discovery Capsule Server.

To verify that a Capsule Server has the Discovery plug-in enabled, navigate to **Infrastructure > Capsules**. The Discovery plug-in should appear in the list of features associated with the Capsule Server. Click **Refresh features** to ensure that the list is up-to-date.

6.3.2. Using Hammer with the Discovery Plug-in

To use the **hammer** command with the Discovery plug-in, you need to enable the Discovery plug-in in `/etc/hammer/cli.modules.d/foreman_discovery.yml` as follows:

```
:foreman_discovery:
:enable_module: true
```

See [hammer configuration directories](#) for more information about the files and directories that **hammer** uses.

6.3.3. Reviewing User Permissions

When it first starts, the Satellite Capsule Server Discovery plug-in creates a role called **Discovery**. You can assign this role to non-administrative users to allow them to use the Discovery plug-in. Alternatively, assign the **perform_discovery** permission to an existing role. For more information on roles and permissions, see [Creating and Managing Users](#) in *Administering Red Hat Satellite*.

6.4. PROVISIONING DISCOVERED HOSTS

After you have correctly configured Discovery plug-ins on both Satellite Server and Capsule Server, you can automatically detect bare-metal hosts. To do so, boot a machine in any provisioning network that was configured with the PXE configuration template described in [Section 6.2.2, “Configuring PXE-booting”](#). The machine is automatically registered with the Satellite Server and appears in the **Hosts > Discovered Hosts** list in the Satellite web UI.

You can either provision the discovered host manually, or you can configure automatic provisioning.

6.4.1. Manually Provisioning Hosts

The following procedure describes how to manually provision discovered hosts from the Satellite web UI.

To Manually Provision a Discovered Host:

1. Navigate to **Hosts > Discovered Hosts**.
2. Select the host you want to provision and click **Provision**.
3. On the host's **Edit** page, complete the necessary details, and then click **Save**.

When the host configuration is saved, Satellite modifies the host's PXELinux file on the TFTP server and reboots the discovered host. It then boots into an installer for the chosen operating system, and finally into the installed operating system.

If you decide to re-provision an existing discovered host, delete the operating system from the machine and reboot it. The host then reappears on the **Discovered Hosts** page.

6.4.2. Decommissioning Discovered Hosts

If you no longer require Red Hat Satellite to manage a specific host, you need to decommission that host to prevent it from being discovered.

To Decommission a Discovered Host:

1. Shut down the host.
2. Navigate to **Hosts > Discovered Hosts**.
3. In the **Name** column find the host you want to decommission and then select **Delete** from the corresponding **Edit** drop-down menu.

6.4.3. Automatically Provisioning Hosts

With Satellite 6.3, it is possible to define provisioning rules that will assign a host group to provisioned hosts and trigger provisioning automatically.

To Create a Provisioning Rule:

1. Navigate to **Configure > Discovery rules**.
2. Click **New Rule**. Specify the following parameters of the provisioning rule:
 - **Name** is the name of the rule displayed in the list of rules. This name must not contain spaces or non-alphanumeric characters.
 - **Search** is the search statement used to match discovered hosts for the particular rule. You can use scoped search syntax to define it. See [Section 6.4.4, "Scoped Search Syntax"](#) for examples of using scoped search.
 - **Host Group** is the host group to be assigned to a matching host before starting the provisioning process. Make sure that the selected host group has all the required parameters set; required parameters are marked with an asterisk (*).
 - **Hostname** defines a pattern for assigning human-readable host names to the matching

hosts. When left blank, the host name is assigned in the format "macMACADDRESS" by default. The same syntax used for provisioning templates is used in this instance. See [Section 6.4.5, "Host Name Patterns"](#) for more information and examples.

- **Hosts limit** is the maximum number of provisioned hosts per rule. If the limit is reached, the rule will not take effect until one or more hosts are deleted. Typical use cases are rules per server rack or row when it is necessary to change provisioning parameters such as host name or host group per entry. You can set this value to zero (0) to specify no limit.
- **Priority** specifies the order of execution of rules. The value must be greater than or equal to zero. A lower value indicates a higher priority. If two rules have the same priority, the first rule encountered is applied.
- **Enabled** provides the option to temporarily enable or disable rules.

3. Click **Submit** to save the rule.

By default, Satellite does not enable automatic discovery of hosts. The following procedure describes how to enable the Auto provisioning variable to provide automatic provisioning according to specified rules.

To Enable Automatic Provisioning:

1. Navigate to **Administer > Settings > Discovered** in the Satellite web UI.
2. Locate Auto provisioning in the **Name** column, and set its value to **true**.
3. Click **Save**.

After you have defined some rules, Red Hat recommends that you discover a host and apply the rules using the **Auto discover** button on the host. This triggers auto-provisioning without the need to enable the global option.

6.4.4. Scoped Search Syntax

This section shows how to use scoped search syntax to filter the discovered hosts according to selected parameters. This is useful when creating a rule for automatic provisioning (see [Section 6.4.3, "Automatically Provisioning Hosts"](#)).

The search fields in the Satellite web UI support automatic completion to make building search strings easier. For example, you can test search patterns on the **Hosts > Discovered Hosts** page. The following are examples of typical search queries:

- `facts.architecture = x86_64`
- `facts.bios_vendor ~ 'Dell'`
- `facts.macaddress = "aa:bb:cc:dd:ee:ff"`
- `facts.macaddress_eth0 = "aa:bb:cc:dd:ee:ff"`
- `facts.ipaddress_eth1 ~ "192.168.*"`
- `facts.architecture ^ (x86_64,i386)`

**NOTE**

The caret symbol (^) in scoped searches means "in" (the same usage as in SQL) and not "starts with" as it is used in regular expressions. You can review the full list of scoped search operators at https://github.com/wvanbergen/scoped_search/blob/master/lib/scoped_search/query_language.md

In Satellite 6.3, all facts are strings, so it is not possible to do numeric comparisons. However, three important facts are extracted and converted to numbers. These are described in Table 6.1, “Facts that Allow Numerical Comparison”.

Table 6.1. Facts that Allow Numerical Comparison

Search Parameter	Description	Example Usage
cpu_count	The number of CPUs	cpu_count >= 8
disk_count	The number of disks attached	disk_count < 10
disks_size	The total amount of disk space (in MiB)	disks_size > 1000000

6.4.5. Host Name Patterns

This section lists the host name patterns that you can use when creating a rule for automatic provisioning (see Section 6.4.3, “Automatically Provisioning Hosts”).

The target host name template pattern has the same syntax as the provisioning templates (ERB). The domain is appended automatically. In addition to the **@host** attribute, the **rand()** function for random integers is available. For example:

- application-server-<%= rand(99999) %>
- load-balancer-<%= @host.facts['bios_vendor'] + '-' + rand(99999) %>
- wwwsrv-<%= @host.hostgroup.name %>
- minion-<%= @host.discovery_rule.name %>
- db-server-<%= @host.ip.gsub('.', '-') + '-' + @host.hostgroup.subnet.name %>

**IMPORTANT**

When creating host name patterns, ensure the resulting host names are unique. Host names must not start with numbers. A good approach is to use unique information provided by Facter (for example, the MAC address, BIOS or serial ID) or to otherwise randomize the host name.

6.4.6. Using the Discovery Plug-in on the Command Line

You can use the **hammer** command to perform certain tasks related to discovery. Run the **hammer -h** command to verify your configuration:

```
$ hammer -h | grep discovery
discovery           Manipulate discovered hosts.
discovery_rule      Manipulate discovered rules.
```

Use the **hammer discovery -h** command to view the available options. For example, you can use the following command to reboot a discovered host (assuming its ID is 130):

```
$ hammer discovery reboot -id 130
Host reboot started
```

6.5. EXTENDING THE DISCOVERY IMAGE

It is possible to extend the Satellite Discovery image with custom facts, software, or device drivers. You can also provide a compressed archive file containing extra code for the image to use.

First, create the following directory structure:

```
.
├── autostart.d
│   └── 01_zip.sh
├── bin
│   └── ntpdate
├── facts
│   └── test.rb
└── lib
    ├── libcrypto.so.1.0.0
    ├── ruby
    └── test.rb
```

Where:

- The **autostart.d** directory contains scripts that are executed in POSIX order by the image when it starts, but before the host is registered to Satellite.
- The **bin** directory is added to the `$PATH` variable; you can place binary files here and use them in the autostart scripts.
- The **facts** directory is added to the `FACTERLIB` variable so that custom facts can be configured and sent to Satellite.
- The **lib** directory is added to the `LD_LIBRARY_PATH` variable and **lib/ruby** is added to the `RUBYLIB` variable, so that binary files in `/bin` can be executed correctly.

New directives and options are appended to the existing environment variables (`PATH`, `LD_LIBRARY_PATH`, `RUBYLIB` and `FACTERLIB`). If you need to specify the path to something explicitly in your scripts, the zip contents are extracted to the **/opt/extension** directory on the image.

After creating the above directory structure, package it into a zip archive with the following command:

```
zip -r my_extension.zip .
```

You can create multiple zip files but be aware they will be extracted to the same place on the Discovery image, so files in later zips will overwrite earlier ones if they have the same file name.

To inform the Discovery image of the extensions it should use, place your zip files on your TFTP server with the Discovery image, and then update the APPEND line of the PXELinux template with the **fdi.zips** option where the paths are relative to the TFTP root. For example, if you have two archives at **\$TFTP/zip1.zip** and **\$TFTP/boot/zip2.zip**, use the following syntax:

```
fdi.zips=zip1.zip,boot/zip2.zip
```

See [Section 6.2.2, “Configuring PXE-booting”](#) for more information on updating the PXE template.

6.6. TROUBLESHOOTING SATELLITE DISCOVERY

If a machine is not listed in the Satellite web UI under **Hosts > Discovered Hosts**, inspect the following configuration areas to help isolate the error:

- Navigate to **Hosts > Provisioning Templates** and redeploy the default PXELinux template using the **Build PXE Default** button.
- Verify the **pxelinux.cfg/default** configuration file on the TFTP Capsule Server.
- Ensure adequate network connectivity between hosts, Capsule Server, and Satellite Server.
- Check the PXELinux template in use and determine the PXE discovery snippet it includes. Snippets are named as follows: **pxelinux_discovery**, **pxegrub_discovery**, or **pxegrub2_discovery**. Verify the **proxy.url** and **proxy.type** options in the PXE discovery snippet.
- Ensure that DNS is working correctly for the discovered nodes, or use an IP address in the **proxy.url** option in the PXE discovery snippet included in the PXELinux template you are using.
- Ensure that the DHCP server is delivering IP addresses to the booted image correctly.
- Ensure the discovered host or virtual machine has at least 1200 MB of memory. Less memory can lead to various random kernel panic errors as the image needs to be extracted in-memory.

For gathering important system facts, use the **discovery-debug** command. It prints out system logs, network configuration, list of facts, and other information on the standard output. The typical use case is to redirect this output and copy it with the **scp** command for further investigation.

The first virtual console on the discovered host is reserved for **systemd** logs. Particularly useful system logs are tagged as follows:

- **discover-host** — initial facts upload
- **foreman-discovery** — facts refresh, reboot remote commands
- **nm-prepare** — boot script which pre-configures NetworkManager
- **NetworkManager** — networking information

Use TTY2 or higher to log in to a discovered host. The root account and SSH access are disabled by default, but you can enable SSH and set the root password using the following kernel command-line options in the Default PXELinux template on the APPEND line:

```
fdi.ssh=1 fdi.rootpw=redhat
```


CHAPTER 7. INTEGRATING RED HAT SATELLITE AND ANSIBLE TOWER

You can integrate Red Hat Satellite 6.3 and Ansible Tower to use Satellite Server as a dynamic inventory source for Ansible Tower.

You can also use the provisioning callback function to run playbooks on hosts managed by Satellite, from either the host or Ansible Tower. When provisioning new hosts from Satellite Server, you can use the provisioning callback function to trigger playbook runs from Ansible Tower. The playbook configures the host following Kickstart deployment.

7.1. ADDING SATELLITE SERVER TO ANSIBLE TOWER AS A DYNAMIC INVENTORY ITEM

To add Satellite Server to Ansible Tower as a dynamic inventory item, you must create a credential for a Satellite Server user on Ansible Tower, add an Ansible Tower user to the credential, and then configure an inventory source.

Prerequisites

Satellite Server and Ansible Tower communicate by using credentials and callbacks.

- You must have a Satellite Server user with an integration role that includes the necessary permission filters. For more information about managing users, roles, and permission filters, see [Managing Users and Roles](#) and [Creating and Managing Roles](#) in the *Administering Red Hat Satellite Guide*.
- You must specify the following permission filters and assign the role to the user.

Table 7.1. Permission Filters

Resource	Permissions	Access Description
Host	<code>view_hosts</code>	To view Satellite Server hosts.
Host Group	<code>view_hostgroups</code>	To view Satellite Server host groups.
Fact value	<code>view_facts</code>	To view Satellite Server Facts.

To Add Satellite Server to Ansible Tower as a Dynamic Inventory Item:

1. In the Ansible Tower Web UI, create a credential for your Satellite. For more information about creating credentials, see [Add a New Credential](#) and [Red Hat Satellite 6 Credentials](#) in the *Ansible Tower User Guide*.

Table 7.2. Satellite Credentials

Credential Type:	Red Hat Satellite 6
Satellite 6 URL:	<code>https://satellite.example.com</code>

Username:	The username of the Satellite user with the integration role.
Password:	The password of the Satellite user.

2. Add an Ansible Tower user to the new credential. For more information about adding a user to a credential, see [Getting Started with Credentials](#) in the *Ansible Tower User Guide*.
3. Add Satellite Server as a new inventory source, specifying the following inventory source options. For more information about adding inventories, see [Add a new inventory](#) in the *Ansible Tower User Guide*.

Table 7.3. Inventory Source Options

Source:	Red Hat Satellite 6
Credential:	The credential you created for Satellite Server.
Overwrite:	Selected
Update on Launch:	Selected
Cache Timeout:	90

For more information about managing inventories, see [Inventories](#) in the *Ansible Tower User Guide*.

7.2. CONFIGURING PROVISIONING CALLBACK FOR A HOST

You can configure Provisioning Callback for an Ansible Tower template. You can then call a specific URL on the Ansible Tower server, pass variables to it, and trigger a playbook run on the calling system.

You can also use this feature to trigger playbook runs on newly deployed hosts. For more information about provisioning callbacks, see [Provisioning Callbacks](#) in the *Ansible Tower User Guide*.

In Satellite the **Satellite Kickstart Default** and **Satellite Kickstart Default Finish** templates include three snippets:

```
ansible_provisioning_callback
ansible_tower_callback_script
ansible_tower_callback_service
```

To configure provisioning callback for a host or host group, you must create and define parameters for each snippet.

Prerequisites

- Red Hat Satellite 6.3 and Ansible Tower must be integrated before configuring Provisioning Callback for a host. For more information, see, [Integrating Satellite and Ansible Tower](#).
- In the Ansible Tower Web UI, you must enable provisioning callbacks, generate the host configuration key, and have the *template_ID* of your job template. For more information about job templates, see [Job Templates](#) in the *Ansible Tower User Guide*.

To Configure Provisioning Callback for a Host:

- In the Red Hat Satellite Web UI, navigate to **Hosts** > **All hosts**.
- On the Hosts page, select the host you want to edit from the **Hosts** list.
- In the Host Group window, click the **Parameters** tab.
- In the Host parameters window, click **Add Parameter**.
- In the **Name** field, enter **ansible_tower_provisioning**.
- In the **Value** field, enter **true**.
- Repeat Step 4 to Step 6 to create each of the following parameters:

Table 7.4. Host Parameters

Name	Value	Description
ansible_tower_provisioning	true	Enables Provisioning Callback.
ansible_tower_fqdn	<i>tower.example.com</i>	The fully qualified domain name (FQDN) of your Ansible Tower.
ansible_job_template_id	<i>template_ID</i>	The ID of your provisioning template found in the URL of the template: /templates/job_template/5.
ansible_host_config_key	<i>config_KEY</i>	The host configuration key generated by your job template in Ansible Tower.

- Click **Submit** when you have created all the necessary parameters.
- To verify that the provisioning callback is configured correctly, start the **ansible-callback** service and then check the status of the service:
 - On the command line, enter the following command to start the **ansible-callback** service:

```
# systemctl start ansible-callback
```

- On the command line, enter the following command to output the status of the **ansible-callback** service:

```
# systemctl status ansible-callback
```

Provisioning callback is configured correctly if the command returns the following output:

```
SAT_host systemd[1]: Started Provisioning callback to Ansible Tower...
```

To Configure Provisioning Callback for a Host Group:

1. In the Red Hat Satellite Web UI, navigate to **Configure > Host groups**.
2. On the Host Groups page, select the host group you want to edit from the **Host Group** list.
3. In the Host Group window, click the **Parameters** tab.
4. In the Host group parameters window, click **Add Parameter**.
5. In the **Name** field, enter **ansible_tower_provisioning**.
6. In the **Value** field, enter **true**.
7. Repeat Step 4 to Step 6 to create each of the following parameters:

Table 7.5. Host Group Parameters

Name	Value	Description
ansible_tower_provisioning	true	Enables Provisioning Callback.
ansible_tower_fqdn	<i>tower.example.com</i>	The fully qualified domain name (FQDN) of your Ansible Tower.
ansible_job_template_id	<i>template_ID</i>	The ID of your provisioning template found in the URL of the template: /templates/job_template/5.
ansible_host_config_key	<i>config_KEY</i>	The host configuration key generated by your job template in Ansible Tower.

8. Click **Submit** when you have created all the necessary parameters.
9. To verify that the provisioning callback is configured correctly, start the **ansible-callback** service and then check the status of the service:
 - a. On the command line, enter the following command to start the **ansible-callback** service:

```
# systemctl start ansible-callback
```

- b. On the command line, enter the following command to output the status of the **ansible-callback** service:

```
# systemctl status ansible-callback
```

Provisioning callback is configured correctly if the command returns the following output:

```
SAT_host systemd[1]: Started Provisioning callback to Ansible  
Tower...
```

You can use the provisioning callback URL and the host configuration key from a host to call Ansible Tower. This triggers the playbook run specified in the template against the host.

You can also use the provisioning callback function to trigger a playbook run from Ansible Tower as part of the provisioning process. The playbook configures the host after Kickstart deployment.

CHAPTER 8. SAMPLE SCENARIOS

8.1. SIMPLE SCENARIO

The simple scenario shows how to add a single host, register it, set up, and run a job on it.

8.1.1. Creating the Host

The following procedure provides an example on how to create a host in Red Hat Satellite.

To Create a Host:

1. Click **Hosts** > **Create Host**.
2. On the **Host** tab, enter the required details:
 - a. In the **Name** field, enter the host name, for example *host1.example.com*.
 - b. In the **Organization** field, enter the organization name, for example *MyOrg*.
 - c. In the **Location** field, enter the location name, for example *MyLoc*.
3. Optionally, on the **Puppet Classes** tab, select the Puppet classes you want to include.
4. On the **Interfaces** tab, edit the primary interface:
 - a. Click the **Edit** button in the **Actions** column.
 - b. Select a type from the **Type** drop-down menu, for example *Interface*.
 - c. In the **MAC address** field, enter the MAC address of your host.
 - d. In the **Device Identifier** field, specify the device identifier for this interface, for example *eth0*.
 - e. In the **DNS name** field, specify the DNS name of your host. For the primary interface, this host name is used with the domain name to form the FQDN.
 - f. Select a domain from the **Domain** drop-down menu, for example *satellite.example.com*. This automatically updates the **IPv4 Subnet** and **IPv6 Subnet** lists with a selection of available subnets. Optionally, select the subnets.
 - g. In the **IPv4 address** field, specify the IPv4 address of your host.
 - h. Click **Ok**.
5. On the **Operating System** tab, enter the required details:
 - a. Select an architecture from the **Architecture** drop-down menu, for example *x86_64*.
 - b. Select an operating system from the **Operating system** drop-down menu, for example *RHEL Server 7.4*.
 - c. Select a partition table from the **Partition table** drop-down menu, for example *Kickstart default*.
 - d. In the **Root password** field, enter the root password for your host.

6. Optionally, on the **Parameters** tab, select the parameters you want to supply to the Puppet master to override the default values.
7. Optionally, on the **Additional Information** tab, enter additional information about the host.
8. Click **Submit**.

8.1.2. Registering the Host

After you have created *host1.example.com*, you must register it so that it can receive updates. The following procedure assumes the host is running Red Hat Enterprise Linux 7.

To Register the Host:

1. In a terminal, connect to the host as the root user.
2. Ensure that a time synchronization tool is enabled and running on the host:

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

3. Install the consumer RPM from the Satellite Server or Capsule Server to which the host is to be registered. The consumer RPM updates the content source location of the host and allows the host to download content from the content source specified in Red Hat Satellite.

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

4. Ensure that an activation key associated with the appropriate Content View and environment exists for the host. If not, see [Managing Activation Keys](#) in the *Content Management Guide* for more information.
5. Clear any old host data related to Red Hat Subscription Manager (RHSM):

```
# subscription-manager clean
```

6. Register the host using RHSM:

```
# subscription-manager register --org=MyOrg \
--activationkey=my_activation_key
```

Command output after registration:

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

7. Enable the **Red Hat Satellite Tools 6** repository:

```
# subscription-manager repos --enable=rhel-version-server-satellite-
tools-6-rpms
```

8. Install the **katello-agent**:

—

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

9. Ensure the **goferd** service is running:

```
# systemctl start goferd
```

10. Install and configure the Puppet agent:

- a. Install the Puppet agent:

```
# yum install puppet
```

- b. Configure the Puppet agent to start at boot:

```
# systemctl enable puppet
```

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

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

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

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

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

...

[agent]
# The file in which puppetd stores a list of the classes
# associated with the retrieved configuration. Can be
loaded in
# the separate ``puppet`` executable using the ``--
loadclasses``
# option.
# The default value is '$confdir/classes.txt'.
classfile = $vardir/classes.txt
pluginsync = true
report = true
ignoreschedules = true
daemon = false
ca_server = satellite.example.com
server = satellite.example.com
environment = KT_Example_Org_Library_RHEL7Server

# Where puppetd caches the local configuration. An
```



```
# extension indicating the cache format is added
automatically.
# The default value is '$confdir/localconfig'.
localconfig = $vardir/localconfig

...
```

- d. Run the Puppet agent on the host:

```
# puppet agent -t --waitforcert 10 --server satellite.example.com
```

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

- i. Log in to the Satellite Server through the web UI.
- ii. Select **Infrastructure > Capsules**.
- iii. Select **Certificates** from the drop-down menu to the right of the required Capsule.
- iv. Click **Sign** to the right of the required host.
- v. Enter the **puppet agent** command again:

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

8.1.3. Running a Job on the Host

The following procedure shows how to run a job template on the previously created and registered host *host1.example.com*.

To Execute a Remote Job:

1. Navigate to **Hosts > All hosts** and select the target host, in our example *host1.example.com*.
2. From the **Select Action** menu at the upper right of the screen select **Schedule Remote Job**.
3. On the **Job invocation** page, define the main job settings:
 - a. Select a job category from the **Job category** drop-down menu, for example *Commands*.
 - b. Select a job template from the **Job template** drop-down menu, for example *Run Command - SSH Default*.
 - c. In the **command** field, enter the command you want to run on the host. For example, *timedatectl set-timezone Europe/Prague* will set the time zone to Prague in Europe.
4. Click **Submit**.

APPENDIX A. TEMPLATE WRITING REFERENCE

Embedded Ruby (ERB) is a tool for generating text files based on templates that combine plain text with Ruby code. Red Hat Satellite uses ERB syntax in **provisioning templates** ([Creating Provisioning Templates](#) in the *Provisioning Guide*), remote execution **job templates** ([Chapter 5, Running Jobs on Hosts](#)), templates for **partition tables** ([Creating Partition Tables](#) in the *Provisioning Guide*), **Smart Variables** ([Configuring Smart Variables](#) in the *Puppet Guide*), and **Smart Class Parameters** ([Configuring Smart Class Parameters](#) in the *Puppet Guide*). This section provides an overview of Satellite specific functions and variables that can be used in ERB templates along with some usage examples. Note that the default templates provided by Red Hat Satellite (**Hosts > Provisioning templates**, **Hosts > Job templates**) also provide a good source of ERB syntax examples.

When provisioning a host or running a remote job, the code in the ERB is executed and the variables are replaced with the host specific values. This process is referred to as **rendering**. The Satellite Server has the safemode rendering option enabled by default, which prevents any harmful code being executed from templates.

A.1. WRITING ERB TEMPLATES

The following points summarize the ERB syntax:

- `<% %>` – marks enclosing Ruby code within the ERB template. The code is executed when the template is rendered. It can contain Ruby control flow structures as well as Satellite specific functions and variables. For example:

```
<% if @host.operatingsystem.family == "Redhat" &&
@host.operatingsystem.major.to_i > 6 %>
systemctl <%= input("action") %> <%= input("service") %>
<% else %>
service <%= input("service") %> <%= input("action") %>
<% end -%>
```

- `<%= %>` – the code output is inserted into the template. This is useful for variable substitution, for example:

```
echo <%= @host.name %>
```

- `<% -%>`, `<%= -%>` – by default, a newline character is inserted after a Ruby block if it is closed at the end of a line. To suppress this behavior, modify the enclosing mark. For example, the following template:

```
curl <%= @host.ip -%>
/mydir
```

is rendered the same as:

```
curl <%= @host.ip %>/mydir
```

In practice, this is used to reduce the number of lines in rendered templates (where Ruby syntax permits).

- `<%# %>` – marks enclosing a comment that will be ignored during template rendering:

```
<%# A comment %>
```

A.2. TROUBLESHOOTING ERB TEMPLATES

The Satellite web UI provides two ways to verify the template rendering for a specific host:

- **Directly in the template editor** – when editing a template (under **Hosts > Partition tables**, **Hosts > Provisioning templates**, or **Hosts > Job templates**), on the **Template** tab click **Preview** and select a host from the drop-down menu. The template then renders in the text field using the selected host's parameters. Preview failures can help to identify issues in your template.
- **At the host's details page** – select a host at **Hosts > All hosts** and click the **Templates** tab to list templates associated with the host. Select **Review** from the drop-down menu next to the selected template to view it's rendered version.

A.3. SATELLITE SPECIFIC FUNCTIONS AND VARIABLES

This section lists Satellite specific functions and variables for ERB templates. Note that some of them can be used in any kind of template, others are limited, for example job templates accept only @host variables and variables from [Table A.4, "Kickstart Specific Variables"](#) are only applicable in Kickstart templates.

You can use the functions listed in the following table across all kinds of templates.

Table A.1. Generic Functions

Name	Description
indent(n)	Indents the block of code by n spaces, useful when using a snippet template that is not indented.
foreman_url(kind)	Returns the full URL to host-rendered templates of the given kind. For example, templates of the "provision" type usually reside at http://HOST/unattended/provision .
snippet(name)	Renders the specified snippet template. Useful for nesting provisioning templates.
snippets(file)	Renders the specified snippet found in the Foreman database, attempts to load it from the unattended/snippets/ directory if it is not found in the database.
snippet_if_exists(name)	Renders the specified snippet, skips if no snippet with the specified name is found.

Example A.1. Using the snippet and indent Functions

The following syntax imports the **subscription_manager_registration** snippet to the template and indents it by four spaces:

```
<%= indent 4 do
  snippet 'subscription_manager_registration'
end %>
```

The following functions can be used in job templates. See [Section 5.3.4, “Creating Advanced Templates”](#) for usage examples.

Table A.2. Functions Specific to Job Templates

Name	Description
<code>input(input_name)</code>	Returns the value of the specified input on the job execution.
<code>render_template(name, parameters)</code>	Renders the specified template, similar to the generic <code>snippet()</code> function but enables passing arguments to the template.

The following variables enable using host data within templates.

Table A.3. Host Specific Variables and Functions

Name	Description
<code>@host.architecture</code>	The architecture of the host.
<code>@host.bond_interfaces</code>	Returns an array of all bonded interfaces. See Note .
<code>@host.capabilities</code>	The method of system provisioning, can be either build (for example kickstart) or image.
<code>@host.certname</code>	The SSL certificate name of the host.
<code>@host.diskLayout</code>	The disk layout of the host. Can be inherited from the operating system.
<code>@host.domain</code>	The domain of the host.
<code>@host.environment</code>	The Puppet environment of the host.
<code>@host.facts</code>	Returns a Ruby hash of facts from Facter. For example to access the 'ipaddress' fact from the output, specify <code>@host.facts['ipaddress']</code> .
<code>@host.grub_pass</code>	Returns the host's GRUB password.
<code>@host.hostgroup</code>	The host group of the host.

Name	Description
@host.info['parameters']	Returns a Ruby hash containing information on host parameters. For example, use <code>@host.info['parameters']['lifecycle_environment']</code> to get the life cycle environment of a host.
@host.image_build?	Returns true if the host is provisioned using an image.
@host.interfaces	Contains an array of all available host interfaces including the primary interface. See Note .
@host.interfaces_with_identifier('IDs')	Returns array of interfaces with given identifier. You can pass an array of multiple identifiers as an input, for example <code>@host.interfaces_with_identifier(['eth0', 'eth1'])</code> . See Note .
@host.ip	The IP address of the host.
@host.location	The location of the host.
@host.mac	The MAC address of the host.
@host.managed_interfaces	Returns an array of managed interfaces (excluding BMC and bonded interfaces). See Note .
@host.medium	The assigned operating system installation medium.
@host.name	The full name of the host.
@host.operatingsystem.family	The operating system family.
@host.operatingsystem.major	The major version number of the assigned operating system.
@host.operatingsystem.minor	The minor version number of the assigned operating system.
@host.operatingsystem.name	The assigned operating system name.
@host.operatingsystem.boot_files_uri(@host.medium,@host.architecture)	Full path to the kernel and initrd, returns an array.
@host.os.medium_uri(@host)	The URI used for provisioning (path configured in installation media).

Name	Description
@host.param_false?(name)	Returns false if host parameter of a given name evaluates to false.
@host.param_true?(name)	Returns true if host parameter of a given name evaluates to true.
@host.params['parameter_name']	Returns the value of specified parameters.
@host.primary_interface	Returns the primary interface of the host.
@host.provider	The compute resource provider.
@host.provision_interface	Returns the provisioning interface of the host. Returns an interface object.
@host.ptable	The partition table name.
@host.puppetmaster	The Puppet master the host should use.
@host.pxe_build?	Returns true if the host is provisioned using the network or PXE.
@host.shortname	The short name of the host.
@host.sp_ip	The IP address of the BMC interface.
@host.sp_mac	The MAC address of the BMC interface.
@host.sp_name	The name of the BMC interface.
@host.sp_subnet	The subnet of the BMC network.
@host.subnet.dhcp	Returns true if a DHCP proxy is configured for this host.
@host.subnet.dns_primary	The primary DNS server of the host.
@host.subnet.dns_secondary	The secondary DNS server of the host.
@host.subnet.gateway	The gateway of the host.
@host.subnet.mask	The subnet mask of the host.
@host.url_for_boot(:initrd)	Full path to the initrd image associated with this host. Not recommended, as it does not interpolate variables.

Name	Description
@host.url_for_boot(:kernel)	Full path to the kernel associated with this host. Not recommended, as it does not interpolate variables, prefer <code>boot_files_uri</code> .
@provisioning_type	Equals to 'host' or 'hostgroup' depending on type of provisioning.
@static	Returns true if the network configuration is static.
@template_name	Name of the template being rendered.
grub_pass	Returns the GRUB password wrapped in md5pass argument, that is --md5pass=#{@host.grub_pass} .
ks_console	Returns a string assembled using the port and the baud rate of the host which can be added to a kernel line. For example console=ttyS1,9600 .
root_pass	Returns the root password configured for the system.

NOTE

Host variables related to network interfaces, such as `@host.interfaces` or `@host.bond_interfaces` return interface data grouped in an array. To extract a parameter value of a specific interface, use Ruby methods to parse the array. For example, to get information about the first interface from an array and use it in a kickstart template:

```
<% myinterface = @host.interfaces.first %>
IPADDR="<%= myinterface.ip %>"
NETMASK="<%= myinterface.subnet.mask %>"
GATEWAY="<%= myinterface.subnet.gateway %>"
```

You can iterate over the interface array, for example to extract an array of interface names use:

```
<% ifnames = []
@host.interfaces.each do |i|
  ifnames.push(i.name)
end %>
```

Example A.2. Using Host Specific Variables

The following example checks if the host has Puppet and the Puppetlabs repository enabled:

```
<%
pm_set = @host.puppetmaster.empty? ? false : true
```

```
puppet_enabled = pm_set || @host.param_true?('force-puppet')
puppetlabs_enabled = @host.param_true?('enable-puppetlabs-repo')
%>
```

The following example shows how to capture the minor and major version of the host's operating system, which can be used for package related decisions:

```
<%
os_major = @host.operatingsystem.major.to_i
os_minor = @host.operatingsystem.minor.to_i
%>

<% if ((os_minor < 2) && (os_major < 14)) -%>
...
<% end -%>
```

The following example imports the 'kickstart_networking_setup' snippet if the host's subnet has the DHCP boot mode enabled:

```
<% subnet = @host.subnet %>
<% if subnet.respond_to?(:dhcp_boot_mode?) -%>
<%= snippet 'kickstart_networking_setup' %>
<% end -%>
```

The majority of common Ruby methods can be applied on host specific variables. For example, to extract the last segment of the host's IP address, you can use:

```
<% @host.ip.split('.').last %>
```

The following variables are designed to be used within kickstart provisioning templates.

Table A.4. Kickstart Specific Variables

Name	Description
@arch	The host architecture name, same as @host.architecture.name.
@dynamic	Returns true if the partition table being used is a %pre script (has the #Dynamic option as the first line of the table).
@epel	A command which will automatically install the correct version of the epel-release rpm. Use in a %post script.
@mediapath	The full kickstart line to provide the URL command.

Name	Description
@osver	The operating system major version number, same as @host.operatingsystem.major.

APPENDIX B. HOST MANAGEMENT WITHOUT GOFERD

From **Satellite 6.2.11** onward, errata and package management via remote execution is available using a **yum** plugin. This allows the goferd service daemon to be disabled, thereby reducing memory and CPU load on content hosts.

The **yum** plugin is included with the **katello-host-tools**. This shipped with the Satellite 6.2.11 client update.

B.1. PREREQUISITES

The following must be made on all content hosts to allow host management by remote execution.

- Ensure **katello-agent** is installed on the content host by following [Section 3.5.3, “Installing the Katello Agent”](#).

- Stop the goferd service:

```
# systemctl stop goferd.service
```

- Disable the goferd service:

```
# systemctl disable goferd.service
```

- Distribute the SSH keys to the content hosts by following [Section 5.1, “Establishing a Secure Connection for Remote Commands”](#).

B.2. CONFIGURING HOST MANAGEMENT WITHOUT GOFERD AS THE SYSTEM DEFAULT

These steps configure host management to use remote execution as the system default for future package deployments.

To Configure Host Management Without Goferd as the System Default:

1. Log in to the Satellite web UI.
2. Navigate to **Administer > Settings**.
3. Select the **Content** tab.
4. Set the **Use remote execution by default** parameter to **Yes**.

The Satellite server now uses host management by remote execution instead of goferd.

B.3. LIMITATIONS WITH HAMMER

The following applies if you are using the **hammer** command to push errata. The **hammer** command is dependent on goferd to manage errata on content hosts. As a workaround, use Satellite’s remote execution feature to apply errata.

To Use Hammer Remote Execution Commands:

For example, perform a **yum -y update** on *host123.example.org*:

```
# hammer job-invocation create \  
--job-template "Run Command - SSH Default" \  
--inputs command="yum -y update" \  
--search-query "name ~ host123"  
Job invocation 24 created  
[.....] [100%]  
1 task(s), 1 success, 0 fail
```