Red Hat Satellite 6.8

Managing Hosts

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

Last Updated: 2022-01-19

Red Hat Satellite 6.8 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

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. OVERVIEW OF HOSTS IN RED HAT SATELLITE 6

A host is any Linux client that Red Hat Satellite manages. Hosts can be physical or virtual. Virtual hosts can be deployed on any platform supported by Red Hat Satellite, such as KVM, VMware vSphere, OpenStack, Amazon EC2, Rackspace Cloud Services or Google Compute Engine.

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 from the Satellite web UI or from the command line.

In the Satellite web UI, you can browse all hosts recognized by Satellite Server, grouped by type:

- **All Hosts** - a list of all hosts recognized by 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 that 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 and type **dev-node*** in the **Search** field. Alternatively, ***node*** will also find the content host **dev-node.example.com.**

WARNING

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

CHAPTER 2. ADMINISTERING HOSTS

This chapter describes creating, registering, administering, and removing hosts.

2.1. CREATING A HOST IN RED HAT SATELLITE

Use this procedure to create a host in Red Hat Satellite.

Procedure

1. In the Satellite web UI, click **Hosts** > **Create Host**.
2. On the **Host** tab, enter the required details.
3. Click the **Ansible Roles** tab, and from the **Ansible Roles** list, select one or more roles that you want to add to the host. Use the **arrow** icon to manage the roles that you add or remove.
4. On the **Puppet Classes** tab, select the Puppet classes you want to include.
5. 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 mange 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 Chapter 4, *Adding Network Interfaces* for details.

d. Click **Submit** to apply the changes and exit.

6. On the **Operating System** tab, enter the required details. For Red Hat operating systems, select **Synced Content** for **Media Selection**. If you want to use non Red Hat operating systems, select **All Media**, then select the installation media from the **Media Selection** list. You can select a partition table from the list or enter a custom partition table in the **Custom partition table** field. You cannot specify both.

7. On the **Parameters** tab, click **Add Parameter** to add any parameter variables that you want to pass to job templates at run time. This includes all Puppet Class, Ansible playbook parameters and host parameters that you want to associate with the host. To use a parameter variable with an Ansible job template, you must add a **Host Parameter**.
   When you create a Red Hat Enterprise Linux 8 host, you can set system purpose attributes. System purpose attributes define what subscriptions to attach automatically on host creation. In the **Host Parameters** area, enter the following parameter names with the corresponding values. For the list of values, see Configuring system purpose in the *Performing a standard RHEL installation* guide.
   - **syspurpose_role**
   - **syspurpose_sla**
   - **syspurpose_usage**
   - **syspurpose_addons**

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

9. Click **Submit** to complete your provisioning request.

For CLI Users

To create a host associated to a host group, enter the following command:

```
# hammer host create \
--name "host_name" \
--hostgroup "hostgroup_name" \
--interface="primary=true, \
          provision=true, \
          mac=mac_address, \
          ip=ip_address" \
```

```
--organization "Your_Organization" \
--location "Your_Location" \
--ask-root-password yes
```

This command prompts you to specify the root password. It is required to specify the host's IP and MAC address. Other properties of the primary network interface can be inherited from the host group or set using the **--subnet**, and **--domain** parameters. You can set additional interfaces using the **--interface** option, which accepts a list of key-value pairs. For the list of available interface settings, enter the **hammer host create --help** command.

2.2. EDITING THE SYSTEM PURPOSE OF A HOST

You can edit the system purpose attributes of Red Hat Enterprise Linux 8 hosts. System purpose attributes define which subscriptions to attach automatically to hosts. For more information about system purpose, see Configuring system purpose in the *Performing a standard RHEL installation* guide.

Procedure

1. In the Satellite web UI, navigate to **Hosts** > **Content Hosts** and click the name of the Red Hat Enterprise Linux 8 host that you want to edit.
2. In the **System Purpose** area, click the **Edit** or **Remove** icon for the system purpose attributes that you want to edit, add, or remove.
3. Click **Save**.
4. Click the **Subscriptions** tab and select **Subscriptions**.
5. Click **Run Auto-Attach** to attach subscriptions to your host automatically.
6. Refresh the page to verify that the subscriptions list contains the correct subscriptions.

For CLI Users

1. Log in to the host and edit the required system purpose attributes. For example, set the usage type to **Production**, the role to **Red Hat Enterprise Linux Server**, and add the **addon** add on. For the list of values, see Configuring system purpose in the *Performing a standard RHEL installation* guide.

```
# syspurpose set-usage Production
# syspurpose set-role Red Hat Enterprise Linux Server
# syspurpose add-addons 'your_addon'
```

2. Verify the system purpose attributes for this host:

```
# syspurpose show
```

3. Automatically attach subscriptions to this host:

```
# subscription-manager attach --auto
```

4. Verify the system purpose status for this host:

```
# subscription-manager status
```

2.3. CHANGING A MODULE STREAM FOR A HOST

If you have a Red Hat Enterprise Linux 8 host, you can modify the module stream for the repositories you install.

After you create the host, you can enable, disable, install, update, and remove module streams from your host in the Satellite web UI.

Procedure

1. In the Satellite web UI, navigate to **Hosts** > **Content Hosts** and click the name of the host that contains the modules you want to change.
2. Click the **Module Streams** tab.
3. From the **Available Module Streams** list, locate the module that you want to change. You can use the **Filter** field to refine the list entries. You can also use the **Filter Status** list to search for modules with a specific status.
4. From the **Actions** list, select the change that you want to make to the module.
5. In the **Job Invocation** window, ensure that the job information is accurate. Change any details that you require, and then click **Submit**.

2.4. CREATING A HOST GROUP

If you create a high volume of hosts, many of the hosts can have common settings and attributes. Adding these settings and attributes for every new host is time consuming. If you use host groups, you can apply common attributes to hosts that you create.

A host group functions as a template for common host settings, containing many of the same details that you provide to hosts. When you create a host with a host group, the host inherits the defined settings from the host group. You can then provide additional details to individualize the host.

Host Group Hierarchy

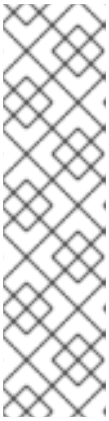
You can create a hierarchy of host groups. Aim to have one base level host group that represents all hosts in your organization and provide general settings, and then nested groups to provide specific settings. For example, you can have a base level host group that defines the operating system, and two nested host groups that inherit the base level host group:

- **Hostgroup: Base** (Red Hat Enterprise Linux 7.6)
  - **Hostgroup: Webserver** (applies the **httpd** Puppet class)
    - **Host: webserver1.example.com** (web server)
    - **Host: webserver2.example.com** (web server)
  - **Hostgroup: Storage** (applies the **nfs** Puppet class)
    - **Host: storage1.example.com** (storage server)
    - **Host: storage2.example.com** (storage server)
  - **Host: custom.example.com** (custom host)

In this example, all hosts use Red Hat Enterprise Linux 7.6 as their operating system because of their inheritance of the **Base** host group. The two web server hosts inherit the settings from the **Webserver** host group, which includes the **httpd** Puppet class and the settings from the **Base** host group. The two storage servers inherit the settings from the **Storage** host group, which includes the **nfs** Puppet class and the settings from the **Base** host group. The custom host only inherits the settings from the **Base** host group.

Procedure

1. In the Satellite web UI, navigate to **Configure** > **Host Groups** and click **Create Host Group**.
2. If you have an existing host group that you want to inherit attributes from, you can select a host group from the **Parent** list. If you do not, leave this field blank.
3. Enter a **Name** for the new host group.
4. Enter any further information that you want future hosts to inherit.
5. Click the **Ansible Roles** tab, and from the **Ansible Roles** list, select one or more roles that you want to add to the host. Use the **arrow** icon to manage the roles that you add or remove.
6. Click the additional tabs and add any details that you want to attribute to the host group.



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, manually create a directory and change the owner:

```
# mkdir /etc/puppetlabs/code/environments/example_environment
# chown apache /etc/puppetlabs/code/environments/example_environment
```

7. Click **Submit** to save the host group.

For CLI Users

Create the host group with the **hammer hostgroup create** command. For example:

```
# hammer hostgroup create --name "Base" \
--lifecycle-environment "Production" --content-view "Base" \
--puppet-environment "production" --content-source-id 1 \
--puppet-ca-proxy-id 1 --puppet-proxy-id 1 --domain "example.com" \
--subnet `ACME's Internal Network` --architecture "x86_64" \
--operatingsystem "RedHat 7.2" --medium-id 9 \
--partition-table "Kickstart default" --root-pass "p@55w0rd!" \
--locations "New York" --organizations "ACME"
```

2.5. CREATING A HOST GROUP FOR EACH LIFECYCLE ENVIRONMENT

Use this procedure to create a host group for the Library lifecycle environment and add nested host groups for other lifecycle environments.

Procedure

To create a host group for each life cycle environment, run the following Bash script:

```
MAJOR="7"
ARCH="x86_64"
ORG="Your Organization"
LOCATIONS="Your Location"
PTABLE_NAME="Kickstart default"
DOMAIN="example.com"

hammer --output csv --no-headers lifecycle-environment list --organization "${ORG}" | cut -d ',' -f 2 |
while read LC_ENV; do
  [[ ${LC_ENV} == "Library" ]] && continue

  hammer hostgroup create --name "rhel-${MAJOR}server-${ARCH}-${LC_ENV}" \
    --architecture "${ARCH}" \
    --partition-table "${PTABLE_NAME}" \
    --domain "${DOMAIN}" \
    --organizations "${ORG}" \
    --query-organization "${ORG}" \
    --locations "${LOCATIONS}" \
    --lifecycle-environment "${LC_ENV}"
done
```

2.6. CHANGING THE GROUP OF A HOST

Use this procedure to change the group of a host.

Procedure

1. In the Satellite web UI, navigate to **Hosts** > **All hosts**.
2. Select the check box of the host you want to change.
3. From the **Select Action** list, select **Change Group**. A new option window opens.
4. From the **Host Group** list, select the group that you want for your host.
5. Click **Submit**.

2.7. CHANGING THE ENVIRONMENT OF A HOST

Use this procedure to change the environment of a host.

Procedure

1. In the Satellite web UI, navigate to **Hosts** > **All hosts**.
2. Select the check box of the host you want to change.
3. From the **Select Action** list, select **Change Environment**. A new option window opens.
4. From the **Environment** list, select the new environment for your host.

5. Click **Submit**.

2.8. CHANGING THE MANAGED STATUS OF A HOST

Hosts provisioned by Satellite are Managed by default. When a host is set to Managed, you can 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 you need to obtain reports about configuration management on systems using an operating system not supported by Satellite, set the host to Unmanaged.

Use this procedure to switch a host between Managed and Unmanaged status.

Procedure

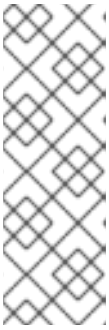
1. In the Satellite web UI, navigate to **Hosts** > **All hosts**.
2. Select the host.
3. Click **Edit**.
4. Click **Manage host** or **Unmanage host** to change the host's status.
5. Click **Submit**.

2.9. ASSIGNING A HOST TO A SPECIFIC ORGANIZATION

Use this procedure to assign a host to a specific organization. For general information about organizations and how to configure them, see Managing Organizations in the *Content Management Guide*.

Procedure

1. In the Satellite web UI, navigate to **Hosts** > **All hosts**.
2. Select the check box of the host you want to change.
3. From the **Select Action** list, select **Assign Organization**. A new option window opens.
4. From the **Select Organization** list, select the organization that you want to assign your host to. 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** 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**.

2.10. ASSIGNING A HOST TO A SPECIFIC LOCATION

Use this procedure to assign a host to a specific location. For general information about locations and how to configure them, see Creating a Location in the *Content Management Guide*.

Procedure

1. In the Satellite web UI, navigate to **Hosts** > **All hosts**.
2. Select the check box of the host you want to change.
3. From the **Select Action** list, select **Assign Location**. A new option window opens.
4. Navigate to the **Select Location** list 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 location you want to assign the host to. The option **Fix Location on Mismatch** will add such a resource to the location, and is therefore the recommended choice. The option **Fail on Mismatch** will always result in an error message. For example, reassigning a host from one location to another will fail, even if there is no actual mismatch in settings.

5. Click **Submit**.

2.11. REMOVING A HOST FROM RED HAT SATELLITE

Use this procedure to remove a host from Red Hat Satellite.

Procedure

1. In the Satellite web UI, navigate to **Hosts** > **All hosts** or **Hosts** > **Content Hosts**. Note that there is no difference from what page you remove a host, from **All hosts** or **Content Hosts**. In both cases, Satellite removes a host completely.
2. Select the hosts that you want to remove.
3. From the **Select Action** list, select **Delete Hosts**.
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.

Disassociating 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** list, select the **Disassociate Hosts** button.
3. Optionally, select the check box to keep the hosts for future action.
4. Click **Submit**.

CHAPTER 3. REGISTERING HOSTS

There are two main methods for registering a host to Satellite Server or Capsule Server:

- 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.
- 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.

You can also register Atomic Hosts to Satellite Server or Capsule Server.

Use one of the following procedures to register a host:

- Section 3.1, "Registering a Host to Red Hat Satellite"
- Section 3.2, "Registering an Atomic Host to Red Hat Satellite"
- Section 3.3, "Registering a Host to Red Hat Satellite Using The Bootstrap Script"

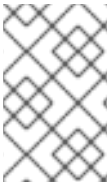
Use the following procedures to install and configure host tools:

- Section 3.4, "Installing the Katello Agent"
- Section 3.5, "Installing Tracer"
- Section 3.6, "Installing and Configuring the Puppet Agent"

Supported Host Operating Systems

Hosts must use one of the following Red Hat Enterprise Linux versions:

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



NOTE

Red Hat Enterprise Linux versions 6.1, 6.2, and 6.3 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

3.1. REGISTERING A HOST TO RED HAT SATELLITE

Use the following procedure to register a host to Red Hat Satellite 6.

Prerequisites

- Satellite Server, any Capsule Servers, and all hosts must be synchronized with the same NTP server, and have a time synchronization tool enabled and running.
- The daemon **rhsmcertd** must be running on the hosts.
- An activation key must be available for the host. For more information, see Managing Activation Keys in the *Content Management Guide*.
- Subscription Manager must be version 1.10 or later. The package is available in the standard Red Hat Enterprise Linux repository.

Procedure

Red Hat Enterprise Linux hosts register to the Red Hat Content Delivery Network by default.

Update each host configuration so that they receive updates from the correct Satellite Server or Capsule Server:

1. Note the fully qualified domain name (FQDN) of the Satellite Server or Capsule Server, for example *server.example.com*.
2. Log in to the host as the **root** user and download the **katello-ca-consumer-latest.noarch.rpm** package from Satellite Server or Capsule Server to which the host is to be registered. The consumer RPM configures the host to download content from the content source that is specified in Red Hat Satellite.

```
# curl --insecure --output katello-ca-consumer-latest.noarch.rpm
https://satellite.example.com/pub/katello-ca-consumer-latest.noarch.rpm
```

3. Install the **katello-ca-consumer-latest.noarch.rpm** package:

```
# yum localinstall katello-ca-consumer-latest.noarch.rpm
```

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** package is an additional **katello-ca-consumer** RPM that contains the server's host name. The **katello-ca-consumer-latest.noarch.rpm** package always reflects the most recent version. Both serve the same purpose.

4. Clear any old host data related to Red Hat Subscription Manager (RHSM):

```
# subscription-manager clean
```

5. 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:

1. In the Satellite web UI, navigate to **Hosts** > **Content Hosts**.
2. Select the check box next to the host that needs to be changed.
3. From the **Select Action** list, select **Set Release Version**.
4. From the **Release Version** list, select **6.3**.
5. Click **Done**.

3.2. REGISTERING AN ATOMIC HOST TO RED HAT SATELLITE

Use the following procedure to register an Atomic Host to Red Hat Satellite 6.

Procedure

1. Log in to the Atomic Host as the **root** user.
2. Retrieve **katello-rhsm-consumer** from Satellite Server:

```
# wget http://satellite.example.com/pub/katello-rhsm-consumer
```

3. Change the mode of **katello-rhsm-consumer** to make it executable:

```
# chmod +x katello-rhsm-consumer
```

4. Run **katello-rhsm-consumer**:

```
# ./katello-rhsm-consumer
```

Register with Red Hat Subscription Manager:

```
# subscription-manager register
```

NOTE

The Katello agent is not supported on Atomic Hosts.

3.3. REGISTERING A HOST TO RED HAT SATELLITE USING THE BOOTSTRAP SCRIPT

Use the bootstrap script to automate content registration and Puppet configuration. You can use the bootstrap script to register new hosts, or to migrate existing hosts to Red Hat Satellite 6 from Satellite 5, RHN, SAM, or RHSM.

The **katello-client-bootstrap** package is installed by default on Satellite Server's base operating system. The **bootstrap.py** script is installed in the **/var/www/html/pub/** directory to make it available to hosts at ***satellite.example.com*/pub/bootstrap.py**. The script includes documentation in the **/usr/share/doc/katello-client-bootstrap-*version*/README.md** file.

To use the bootstrap script, you must install it on the host. As the script is only required once, and only for the **root** user, you can place it in **/root** or **/usr/local/sbin** and remove it after use. This procedure uses **/root**.

Prerequisites

- You have a Satellite user with the permissions required to run the bootstrap script. The examples in this procedure specify the **admin** user. If this is not acceptable to your security policy, create a new role with the minimum permissions required and add it to the user that will run the script. For more information, see Section 3.3.1, "Setting Permissions for the Bootstrap Script".
- You have an activation key for your hosts with the Satellite Tools 6.8 repository enabled. For information on configuring activation keys, see Managing Activation Keys in the *Content Management Guide*.
- You have created a host group. For more information about creating host groups, see Section 2.4, "Creating a Host Group" .

Puppet Considerations

If a host group is associated with a Puppet environment created inside a **Production** environment, Puppet fails to retrieve the Puppet CA certificate while registering a host from that host group.

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/puppetlabs/code/environments/example_environment
# chown apache /etc/puppetlabs/code/environments/example_environment
```

2. Navigate to **Configure** > **Environments** and click **Import environment from**. The button name includes the FQDN of the internal or external Capsule.

3. Choose the created directory and click **Update**.

Procedure

1. Log in to the host as the **root** user.

2. Download the script:

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

3. Make the script executable:

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

4. Confirm that the script is executable by viewing the help text:

   - On Red Hat Enterprise Linux 8:

```
# /usr/libexec/platform-python bootstrap.py -h
```

   - On other Red Hat Enterprise Linux versions:

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

5. Enter the bootstrap command with values suitable for your environment.
   For the **--server** option, specify the FQDN of Satellite Server or a Capsule Server. For the **--location**, **--organization**, and **--hostgroup** options, use quoted names, not labels, as arguments to the options. See Section 3.3.2, “Advanced Bootstrap Script Configuration” for advanced use cases.

   - On Red Hat Enterprise Linux 8, enter the following command:

```
# /usr/libexec/platform-python bootstrap.py \
--login=admin \
--server satellite.example.com \
--location="Example Location" \
--organization="Example Organization" \
--hostgroup="Example Host Group" \
--activationkey=activation_key
```

   - On Red Hat Enterprise Linux 5, 6, or 7, enter the following command:

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

6. Enter the password of the Satellite user you specified with the **--login** option.
   The script sends notices of progress to **stdout**.

7. When prompted by the script, approve the host's Puppet certificate. In the Satellite web UI, navigate to **Infrastructure** > **Capsules** and find the Satellite or Capsule Server you specified with the **--server** option.

8. From the list in the **Actions** column, select **Certificates**.

9. In the **Actions** column, click **Sign** to approve the host's Puppet certificate.

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

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

12. Optional: After the host registration is complete, remove the script:

```
# rm bootstrap.py
```

3.3.1. Setting Permissions for the Bootstrap Script

Use this procedure to configure a Satellite user with the permissions required to run the bootstrap script.

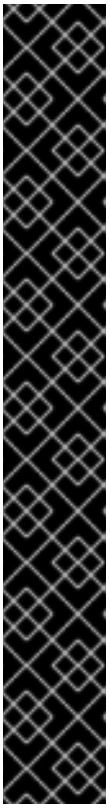
Procedure

1. In the Satellite web UI, navigate to **Administer** > **Users**.

2. Select an existing user by clicking the required **Username**. A new pane opens with tabs to modify information about the selected user. Alternatively, create a new user specifically for the purpose of running this script.

3. Click the **Roles** tab.

4. Select **Edit hosts** and **Viewer** from the **Roles** list.



IMPORTANT

The **Edit hosts** role 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 following permissions and assign it to the user:

- **view_organizations**
- **view_locations**
- **view_domains**
- **view_hostgroups**
- **view_hosts**
- **view_architectures**
- **view_ptables**
- **view_operatingsystems**
- **create_hosts**

5. Click **Submit**.

For CLI Users

1. Create a role with the minimum permissions required by the bootstrap script. This example creates a role with the name *Bootstrap*:

```
# 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
```

Alternatively, you can create a new user and assign this new role to them. For more information on creating users with Hammer, see Managing Users and Roles in the *Administering Red Hat Satellite* guide.

3.3.2. Advanced Bootstrap Script Configuration

This section has more examples for using the bootstrap script to register or migrate a host.

WARNING

These examples specify the **admin** Satellite user. If this is not acceptable to your security policy, create a new role with the minimum permissions required by the bootstrap script. For more information, see Section 3.3.1, "Setting Permissions for the Bootstrap Script".

Migrating a host from one Satellite 6 to another Satellite 6

Use the script with **--force** to remove the **katello-ca-consumer-*** packages from the old Satellite and install the **katello-ca-consumer-*** packages on the new Satellite.

- On Red Hat Enterprise Linux 8, enter the following command:

```
# /usr/libexec/platform-python bootstrap.py \
--login=admin \
--server satellite.example.com \
--location="Example Location" \
--organization="Example Organization" \
--hostgroup="Example Host Group" \
--activationkey=activation_key \
--force
```

- On Red Hat Enterprise Linux 5, 6, or 7, enter the following command:

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

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

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

- On Red Hat Enterprise Linux 8, enter the following command:

```
# /usr/libexec/platform-python bootstrap.py \
--login=admin \
--server satellite.example.com \
--location="Example Location" \
--organization="Example Organization" \
--hostgroup="Example Host Group" \
```

```
--activationkey=activation_key \
--legacy-purge \
--legacy-login rhn-user
```

- On Red Hat Enterprise Linux 5, 6, or 7, enter the following command:

```
# bootstrap.py --login=admin \
--server satellite.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**.

- On Red Hat Enterprise Linux 8, enter the following command:

```
# /usr/libexec/platform-python bootstrap.py \
--login=admin \
--server satellite.example.com \
--location="Example Location" \
--organization="Example Organization" \
--hostgroup="Example Host Group" \
--activationkey=activation_key \
--skip-puppet
```

- On Red Hat Enterprise Linux 5, 6, or 7, enter the following command:

```
# bootstrap.py --login=admin \
--server satellite.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 omit the provisioning and configuration management functions, use **--skip-foreman**.

- On Red Hat Enterprise Linux 8, enter the following command:

```
# /usr/libexec/platform-python bootstrap.py \
--server satellite.example.com \
--organization="Example Organization" \
--activationkey=activation_key \
--skip-foreman
```

- On Red Hat Enterprise Linux 5, 6, or 7, enter the following command:

```
# bootstrap.py --server satellite.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.

- On Red Hat Enterprise Linux 8, enter the following command:

```
# /usr/libexec/platform-python bootstrap.py \
--login=admin \
--server satellite.example.com \
--location="Example Location" \
--organization="Example Organization" \
--hostgroup="Example Host Group" \
--activationkey=activation_key \
--download-method https
```

- On Red Hat Enterprise Linux 5, 6, or 7, enter the following command:

```
# bootstrap.py --login=admin \
--server satellite.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 might need to override the auto-detection of the IP address and provide a specific IP address to Satellite. Use **--ip**.

- On Red Hat Enterprise Linux 8, enter the following command:

```
# /usr/libexec/platform-python bootstrap.py \
--login=admin \
--server satellite.example.com \
--location="Example Location" \
--organization="Example Organization" \
--hostgroup="Example Host Group" \
--activationkey=activation_key \
--ip 192.x.x.x
```

- On Red Hat Enterprise Linux 5, 6, or 7, enter the following command:

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

- On Red Hat Enterprise Linux 8, enter the following command:

```
# /usr/libexec/platform-python bootstrap.py \
--login=admin \
--server satellite.example.com \
--location="Example Location" \
--organization="Example Organization" \
--hostgroup="Example Host Group" \
--activationkey=activation_key \
--rex \
--rex-user root
```

- On Red Hat Enterprise Linux 5, 6, or 7, enter the following command:

```
# bootstrap.py --login=admin \
--server satellite.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 during registration

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

- On Red Hat Enterprise Linux 8, enter the following command:

```
# /usr/libexec/platform-python bootstrap.py \
--login=admin \
--server satellite.example.com \
--location="Example Location" \
--organization="Example Organization" \
--hostgroup="Example Host Group" \
--activationkey=activation_key \
--add-domain
```

- On Red Hat Enterprise Linux 5, 6, or 7, enter the following command:

```
# bootstrap.py --login=admin \
--server satellite.example.com \
--location="Example Location" \
```

```
--organization="Example Organization" \
--hostgroup="Example Host Group" \
--activationkey=activation_key \
--add-domain
```

Providing an alternative 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. If you cannot update the host to use an FQDN that is accepted by Satellite, you can use the bootstrap script to specify an alternative FQDN.

1. Set **create_new_host_when_facts_are_uploaded** and **create_new_host_when_report_is_uploaded** to false using Hammer:

```
# 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
```

2. Use **--fqdn** to specify the FQDN that will be reported to Satellite:

   - On Red Hat Enterprise Linux 8, enter the following command:

```
# /usr/libexec/platform-python bootstrap.py --login=admin \
--server satellite.example.com \
--location="Example Location" \
--organization="Example Organization" \
--hostgroup="Example Host Group" \
--activationkey=activation_key \
--fqdn node100.example.com
```

   - On Red Hat Enterprise Linux 5, 6, or 7, enter the following command:

```
# bootstrap.py --login=admin \
--server satellite.example.com \
--location="Example Location" \
--organization="Example Organization" \
--hostgroup="Example Host Group" \
--activationkey=activation_key \
--fqdn node100.example.com
```

3.4. INSTALLING THE KATELLO AGENT

You can install the Katello agent to remotely update Satellite clients.

NOTE

The Katello agent is deprecated and will be removed in a future Satellite version. Migrate your processes to use the remote execution feature to update clients remotely. For more information, see Host Management Without Goferd and Katello Agent in the *Managing Hosts Guide*.

The **katello-agent** package depends on the **gofer** package that provides the **goferd** service. This service must be enabled so that Satellite Server or Capsule Server can provide information about errata that are applicable for content hosts.

Prerequisites

Before installing the Katello agent, ensure the following conditions are met:

- You have enabled the Satellite Tools 6.8 repository on Satellite Server. For more information, see Enabling the Satellite Tools 6.8 Repository in *Installing Satellite Server from a Connected Network*.
- You have synchronized the Satellite Tools 6.8 repository on Satellite Server. For more information, see Synchronizing the Satellite Tools 6.8 Repository in *Installing Satellite Server from a Connected Network*.
- You have enabled the Satellite Tools 6.8 repository on the client.

Procedure

To install the Katello agent, complete the following steps:

1. Install the **katello-agent** package:

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

2. Start the **goferd** service :

```
# systemctl start goferd
```

3.5. INSTALLING TRACER

Use this procedure to install Tracer on Red Hat Satellite 6.8, and access Traces. Tracer displays a list of services and applications that are outdated and need to be restarted. Traces is the output generated by Tracer in the Satellite web UI.

Prerequisites

- The host must be registered to Red Hat Satellite.
- The Red Hat Satellite Tools 6.8 repository must be enabled and synchronized on Satellite Server, and enabled on the host.

Procedure

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. In the Satellite web UI, navigate to **Hosts** > **All hosts**, then click the required host name.

4. In the **Properties** tab, examine the **Properties** table and find the Traces item. If you cannot find a Traces item in the **Properties** table, then Tracer is not installed.
5. Navigate to **Hosts** > **Content Hosts**, then click the required host name.
6. Click the **Traces** tab to view Traces.

3.6. INSTALLING AND CONFIGURING THE PUPPET AGENT

Use this procedure to install and configure the Puppet agent on a host. For more information about Puppet, see the *Puppet Guide*.

Prerequisites

- The host must be registered to Red Hat Satellite.
- The host must have a Puppet environment assigned to it.
- The Red Hat Satellite Tools 6.8 repository must be enabled and synchronized on Satellite Server, and enabled on the host.

Procedure

1. Log in to the host as the **root** user.
2. Install the Puppet agent package:

```
# yum install puppet-agent
```

3. Configure the Puppet agent to start on boot:
   - On Red Hat Enterprise Linux 6:

```
# chkconfig puppet on
```

   - On Red Hat Enterprise Linux 7:

```
# systemctl enable puppet
```

4. Append the following server and environment settings to the **/etc/puppetlabs/puppet/puppet.conf** file. Set the **environment** parameter to the name of the Puppet environment to which the host belongs:

```
environment = My_Example_Org_Library
server = satellite.example.com
ca_server = satellite.example.com
```

5. Run the Puppet agent on the host:

```
# puppet agent -t
```

6. In the Satellite web UI, navigate to **Infrastructure** > **Capsules**.
7. From the list in the **Actions** column for the required Capsule Server, select **Certificates**.

8. Click **Sign** to the right of the required host to sign the SSL certificate for the Puppet client.
9. Enter the **puppet agent** command again:

```
# puppet agent -t
```

CHAPTER 4. ADDING NETWORK INTERFACES

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 2.1, "Creating a Host in Red Hat Satellite" 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. Use **alias** to add an additional IP address to an existing interface.
  For more information about adding a physical interface, see Section 4.1, "Adding a Physical Interface".

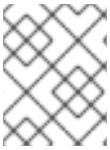
  For more information about adding a virtual interface, see Section 4.2, "Adding a Virtual Interface".

- **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, increasing the bandwidth and providing redundancy. See Section 4.3, "Adding a Bonded Interface" for details.

- **BMC**: Baseboard Management Controller (BMC) allows you to remotely monitor and manage the physical state of machines. For more information about BMC, see Enabling Power Management on Managed Hosts in *Installing Satellite Server from a Connected Network* . For more information about configuring BMC interfaces, see Section 4.4, "Adding a Baseboard Management Controller (BMC) Interface".

NOTE

Additional interfaces have the **Managed** flag enabled by default, 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.

4.1. ADDING A PHYSICAL INTERFACE

Use this procedure to add an additional physical interface to a host.

Procedure

1. In the Satellite web UI, navigate to **Hosts** > **All 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** list.
5. Specify a **MAC address**. This setting is required.
6. Specify the **Device Identifier**, for example **eth0**. The identifier is used to specify this physical interface when creating bonded interfaces, VLANs, and aliases.
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** list. To create and manage domains, navigate to **Infrastructure** > **Domains**.
9. Select a subnet from the **Subnet** list. To create and manage subnets, navigate to **Infrastructure** > **Subnets**.
10. Specify the **IP address**. Managed interfaces with an assigned DHCP Capsule Server require this setting for creating a DHCP lease. DHCP-enabled managed interfaces are automatically provided with a suggested IP address.
11. Select whether the interface is **Managed**. If the interface is managed, 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 whether this is the **Primary** interface for the host. The DNS name from the primary interface is used as the host portion of the FQDN.
13. Select whether this is the **Provision** interface for the host. TFTP boot takes place using the provisioning interface. For image-based provisioning, the script to complete the provisioning is executed through the provisioning interface.
14. Select whether to use the interface for **Remote execution**.
15. Leave the **Virtual NIC** check box clear.
16. Click **OK** to save the interface configuration.
17. Click **Submit** to apply the changes to the host.

4.2. ADDING A VIRTUAL INTERFACE

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

Procedure

1. In the Satellite web UI, navigate to **Hosts** > **All 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** list.
5. Specify the general interface settings. The applicable configuration options are the same as for the physical interfaces described in Section 4.1, "Adding a Physical Interface" .
   Specify a **MAC address** for managed virtual interfaces so that the configuration files for provisioning are generated correctly. However, a **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 **Device 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**: Optionally set a VLAN tag to trunk a network segment from the physical network through to the virtual interface. If you do not specify a tag, managed interfaces inherit the VLAN tag of the associated subnet. User-specified entries from this field are not applied to 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.
8. Click **Submit** to apply the changes to the host.

4.3. ADDING A BONDED INTERFACE

Use this procedure to configure a bonded interface for a host.

Procedure

1. In the Satellite web UI, navigate to **Hosts** > **All 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** list. 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 4.1, "Adding a Physical Interface" .
   Bonded interfaces use IDs in the form of **bond0** in the **Device Identifier** field.

   A single **MAC address** is sufficient.
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 4.1, "Bonding Modes Available in Red Hat Satellite" for a brief description of each bonding mode.

- **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**. See the Red Hat Enterprise Linux 7 Networking Guide for details of the configuration options you can specify for the bonded interface.

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

For CLI Users

To create a host with a bonded interface, enter the following command:

```
# hammer host create --name bonded_interface \
--hostgroup-id 1 \
--ip=192.168.100.123 \
--mac=52:54:00:14:92:2a \
--subnet-id=1 \
--managed true \
  --interface="identifier=eth1, \
              mac=52:54:00:62:43:06, \
              managed=true, \
              type=Nic::Managed, \
              domain_id=1, \
              subnet_id=1" \
  --interface="identifier=eth2, \
              mac=52:54:00:d3:87:8f, \
              managed=true, \
              type=Nic::Managed, \
              domain_id=1, \
              subnet_id=1" \
  --interface="identifier=bond0, \
              ip=172.25.18.123, \
              type=Nic::Bond, \
              mode=active-backup, \
              attached_devices=[eth1,eth2], \
              managed=true, \
              domain_id=1, \
              subnet_id=1" \
--organization "Your_Organization" \
--location "Your_Location" \
--ask-root-password yes
```

Table 4.1. Bonding Modes Available in Red Hat Satellite

Bonding Mode	Description
balance-rr	Transmissions are received and sent sequentially on each bonded interface.

Bonding Mode	Description
active-backup	Transmissions are received and sent through 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 is always 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.

4.4. ADDING A BASEBOARD MANAGEMENT CONTROLLER (BMC) INTERFACE

Use this procedure to configure a baseboard management controller (BMC) interface for a host that supports this feature.

Prerequisites

- 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, so that it can create a DHCP reservation.

Procedure

1. Enable BMC on the Capsule server if it is not already enabled:
   a. Configure BMC power management on the Capsule Server by running the **satellite-installer** script with the following options:

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

b. In the Satellite web UI, navigate to **Infrastructure** > **Capsules**.

c. From the list in the **Actions** column, click **Refresh**. The list in the **Features** column should now include BMC.

2. In the Satellite web UI, navigate to **Hosts** > **All hosts**.
3. Click **Edit** next to the host you want to edit.
4. On the **Interfaces** tab, click **Add Interface**.
5. Select **BMC** from the **Type** list. Type-specific configuration options are appended to the form.
6. Specify the general interface settings. The applicable configuration options are the same as for the physical interfaces described in Section 4.1, “Adding a Physical Interface” .
7. Specify the configuration options specific to BMC interfaces:
   - **Username** and **Password**: Specify any authentication credentials required by BMC.
   - **Provider**: Specify the BMC provider.
8. Click **OK** to save the interface configuration.
9. Click **Submit** to apply the changes to the host.

CHAPTER 5. UPGRADING HOSTS FROM RHEL 7 TO RHEL 8

You can use a job template to upgrade your Red Hat Enterprise Linux 7 hosts to Red Hat Enterprise Linux 8.

Prerequisites

- Ensure that your RHEL 7 hosts meet the requirements for the upgrade to RHEL 8. For more information, see Planning an upgrade in the *Upgrading from RHEL 7 to RHEL 8* guide.
- Prepare your hosts for the upgrade. For more information, see Preparing a RHEL 7 system for the upgrade in the *Upgrading from RHEL 7 to RHEL 8* guide.
- Enable remote execution feature on Satellite. For more information, see Chapter 10, *Configuring and Setting up Remote Jobs*.
- Distribute Satellite SSH keys to the hosts that you want to upgrade. For more information, see Section 10.8, "Distributing SSH Keys for Remote Execution".

Procedure

1. On Satellite, enable the **foreman_plugin_leapp** puppet module:

```
# satellite-installer --enable-foreman-plugin-leapp
```

2. In the Satellite web UI, navigate to **Hosts** > **All Hosts**.
3. Select the hosts that you want to upgrade to Red Hat Enterprise Linux 8.
4. In the upper right of the Hosts window, from the **Select Action** list, select **Preupgrade check with Leapp**.
5. Click **Submit** to start the pre-upgrade check.
6. When the check is finished, click the **Leapp preupgrade report** tab to see if LEAPP has found any issues on RHEL 7 hosts. Issues that have the **Inhibitor** flag are considered crucial and are likely to break the upgrade procedure. Some issues might have documentation linked that describe how to fix them.
7. Optional: If you have issues that have commands associated with them, you can fix them with a remote job. To do that, select these issues, click the **Fix Selected** button, and submit the job.
8. After you fixed the issues, click the **Rerun** button, and then click **Submit** to run the pre-upgrade check again to verify that your RHEL 7 hosts do not have any issues and are ready to be upgraded.
9. When your systems are ready for the upgrade, click the **Run Upgrade** button and click submit to start the upgrade.

CHAPTER 6. HOST MANAGEMENT AND MONITORING USING RED HAT WEB CONSOLE

Red Hat web console is an interactive web interface that you can use to perform actions and monitor Red Hat Enterprise Linux hosts. You can enable a remote-execution feature to integrate Satellite with Red Hat web console. When you install Red Hat web console on a host that you manage with Satellite, you can view the Red Hat web console dashboards of that host from within the Satellite web UI. You can also use the features that are integrated with Red Hat web console, for example, Red Hat Image Builder.

6.1. INTEGRATING SATELLITE WITH RED HAT WEB CONSOLE

By default, Red Hat web console integration is disabled in Satellite. If you want to access Red Hat web console features for your hosts from within Satellite, you must first enable Red Hat web console integration on Satellite Server.

Procedure

- On Satellite Server, run **satellite-installer** with the **--enable-foreman-plugin-remote-execution-cockpit** option:

```
# satellite-installer --enable-foreman-plugin-remote-execution-cockpit
```

6.2. MANAGING AND MONITORING HOSTS USING RED HAT WEB CONSOLE

You can access the Red Hat web console web UI through the Satellite web UI and use the functionality to manage and monitor hosts in Satellite.

Prerequisites

- Red Hat web console is enabled in Satellite.
- Red Hat web console is installed on the host that you want to view:
  - For Red Hat Enterprise Linux 8, see Installing the web console in the *Managing systems using the RHEL 8 web console* guide.
  - For Red Hat Enterprise Linux 7, see Installing the web console in the *Managing systems using the RHEL 7 web console* guide.
- Satellite or Capsule can authenticate to the host with SSH keys. For more information, Section 10.8, “Distributing SSH Keys for Remote Execution” .

Procedure

1. In the Satellite web UI, navigate to **Hosts** > **All Hosts** and select the host that you want to manage and monitor with Red Hat web console.
2. In the upper right of the host window, click **Web Console**.

You can now access the full range of features available for host monitoring and management, for example, Red Hat Image Builder, through the Red Hat web console.

For more information about getting started with Red Hat web console, see the *Managing systems using the RHEL 8 web console* guide or the *Managing systems using the RHEL 7 web console* guide.

For more information about using Red Hat Image Builder through Red Hat web console, see Accessing Image Builder GUI in the RHEL 8 web console or Accessing Image Builder GUI in the RHEL 7 web console.

CHAPTER 7. MONITORING HOSTS USING RED HAT INSIGHTS

In this chapter, you can find information about creating host monitoring reports and monitoring your hosts using Red Hat Insights and creating an Insights plan.

7.1. USING RED HAT INSIGHTS WITH HOSTS IN SATELLITE

You can use Red Hat Insights to diagnose systems and downtime related to security exploits, performance degradation and stability failures. You can use the dashboard to quickly identify key risks to stability, security, and performance. You can sort by category, view details of the impact and resolution, and then determine what systems are affected.

To use Red Hat Insights to monitor hosts that you manage with Satellite, you must first install Red Hat Insights on your hosts and register your hosts with Red Hat Insights.

To install and register your host using Puppet, or manually, see Red Hat Insights Getting Started.

Deploying Red Hat Insights using the Ansible Role

You can automate the installation and registration of hosts with Red Hat Insights using the **RedHatInsights.insights-client** Ansible role. For more information about adding this role to your Satellite, see Getting Started with Ansible in Satellite in *Configuring Satellite to use Ansible* .

1. Add the **RedHatInsights.insights-client** role to the hosts.
   For new hosts, see Section 2.1, “Creating a Host in Red Hat Satellite” .

   For existing hosts, see Using Ansible Roles to Automate Repetitive Tasks on Satellite Hosts in *Configuring Satellite to use Ansible* .

2. To run the **RedHatInsights.insights-client** role on your host, navigate to **Hosts** > **All Hosts** and click the name of the host that you want to use.

3. Click the **Run Ansible roles** button.

When the role completes, you can view and work with the host that you add on the **Insights** > **Overview** page of the Satellite web UI.

Additional Information

- To apply any system updates to the Red Hat Insights plug-in, enter **httpd restart** after updating.
- To view the logs for Red Hat Insights and all plug-ins, go to **/var/log/foreman/production.log**.
- If you have problems connecting to Red Hat Insights, ensure that your certificates are up-to-date. Refresh your subscription manifest to update your certificates.
- You can change the default schedule for running **insights-client** by configuring **insights-client.timer** on a host. For more information, see Changing the insights-client schedule in the *Client Configuration Guide for Red Hat Insights* .

7.2. CREATING AN INSIGHTS PLAN FOR HOSTS

With Satellite 6, you can create a Red Hat Insights remediation plan and run the plan on Satellite hosts.

Procedure

To create the plan, complete the following steps:

1. In the Satellite web UI, navigate to **Insights** > **Inventory**, and select the hosts that you want to include in an Insights plan.
2. From the **Actions** list, select **Create a new Plan/Playbook**
3. From the Plan/Playbook Builder window, select **Create new plan** and enter a name for the plan.
4. Select whether you want the rule to apply to a specific system, group, or all systems.
5. Select one or more rules that you want to add to the plan. Use the **Filter** field to search for specific keywords.
6. Click **Save**.
7. In the Planner window, select **Run Playbook**.

In the Jobs window, you can view the progress of your plan as the playbook runs.

You can view the Insights plan by navigating to **Insights** > **Planner**.

CHAPTER 8. USING REPORT TEMPLATES TO MONITOR HOSTS

You can use report templates to query Satellite data to obtain information about, for example, host status, registered hosts, applicable errata, applied errata, subscription details, and user activity. You can use the report templates that ship with Satellite or write your own custom report templates to suit your requirements. The reporting engine uses the embedded Ruby (ERB) syntax. For more information about writing templates and ERB syntax, see Appendix A, *Template Writing Reference*.

You can create a template, or clone a template and edit the clone. For help with the template syntax, click a template and click the **Help** tab.

8.1. GENERATING HOST MONITORING REPORTS

To view the report templates, in the Satellite web UI, navigate to **Monitor** > **Report Templates**.

To schedule reports, you can configure a cron job or use the Satellite web UI.

Procedure

1. In the Satellite web UI, navigate to **Monitor** > **Report Templates**.
2. To the right of the report template that you want to use, click **Generate**.
3. Optional: To schedule a report, to the right of the **Generate at** field, click the icon to select the date and time you want to generate the report at.
4. Optional: To send a report to an e-mail address, select the **Send report via e-mail** check box, and in the **Deliver to e-mail addresses** field, enter the required e-mail address.
5. Optional: Apply search query filters. To view all available results, do not populate the filter field with any values.
6. Click **Submit**. A CSV file that contains the report is downloaded. If you have selected the **Send report via e-mail** check box, the host monitoring report is sent to your e-mail address.

For CLI Users

To generate a report, complete the following steps:

1. List all available report templates:

```
# hammer report-template list
```

2. Generate a report:

```
# hammer report-template generate --id template ID
```

This command waits until the report fully generates before completing. If you want to generate the report as a background task, you can use the **hammer report-template schedule** command.

8.2. CREATING A REPORT TEMPLATE

In Satellite, you can create a report template and customize the template to suit your requirements. You can import existing report templates and further customize them with snippets and template macros.

Report templates use Embedded Ruby (ERB) syntax. To view information about working with ERB syntax and macros, in the Satellite web UI, navigate to **Monitor** > **Report Templates**, and click **Create Template**, and then click the **Help** tab.

When you create a report template in Satellite, safe mode is enabled by default. For more information about safe mode, see Section 8.8, "Report Template Safe Mode" .

For more information about writing templates, see the Appendix A, *Template Writing Reference*.

To view a step by step example of populating a template, see Section 8.7, "Creating a Report Template to Monitor Entitlements".

For more information about macros you can use in report templates, see Section A.4, "Templates Macros".

Procedure

1. In the Satellite web UI, navigate to **Monitor** > **Report Templates**, and click **Create Template**.
2. In the **Name** field, enter a unique name for your report template.
3. If you want the template to be available to all locations and organizations, select **Default**.
4. Create the template directly in the template editor or import a template from a text file by clicking **Import**. For more information about importing templates, see Section 8.5, "Importing Report Templates".
5. Optional: In the **Audit Comment** field, you can add any useful information about this template.
6. Click the **Input** tab, and in the **Name** field, enter a name for the input that you can reference in the template in the following format: **input('name')**. Note that you must save the template before you can reference this input value in the template body.
7. Select whether the input value is mandatory. If the input value is mandatory, select the **Required** check box.
8. From the **Value Type** list, select the type of input value that the user must input.
9. Optional: If you want to use facts for template input, select the **Advanced** check box.
10. Optional: In the **Options** field, define the options that the user can select from. If this field remains undefined, the users receive a free-text field in which they can enter the value they want.
11. Optional: In the **Default** field, enter a value, for example, a host name, that you want to set as the default template input.
12. Optional: In the **Description** field, you can enter information that you want to display as inline help about the input when you generate the report.
13. Optional: Click the **Type** tab, and select whether this template is a snippet to be included in other templates.
14. Click the **Location** tab and add the locations where you want to use the template.

15. Click the **Organizations** tab and add the organizations where you want to use the template.
16. Click **Submit** to save your changes.

8.3. EXPORTING REPORT TEMPLATES

You can export report templates that you create in Satellite.

Procedure

1. In the Satellite web UI, navigate to **Monitor** > **Report Templates**.
2. Locate the template that you want to export, and from the list in the **Actions** column, select **Export**.
3. Repeat this action for every report template that you want to download.

An **.erb** file that contains the template downloads.

For CLI Users

1. To view the report templates available for export, enter the following command:

```
# hammer report-template list
```

   Note the template ID of the template that you want to export in the output of this command.

2. To export a report template, enter the following command:

```
# hammer report-template dump --id template_ID > example_export.erb
```

8.4. EXPORTING REPORT TEMPLATES USING THE SATELLITE API

You can use the Satellite **report_templates** API to export report templates from Satellite. For more information about using the Satellite API, see the API Guide.

Procedure

1. Use the following request to retrieve a list of available report templates:

   Example request:

```
$ curl --insecure --user admin:redhat \
--request GET \
--config https://satellite.example.com/api/report_templates \
| json_reformat
```

   In this example, the **json_reformat** tool is used to format the JSON output.

   Example response:

```
{
    "total": 6,
```

```
    "subtotal": 6,
    "page": 1,
    "per_page": 20,
    "search": null,
    "sort": {
        "by": null,
        "order": null
    },
    "results": [
        {
            "created_at": "2019-11-20 17:49:52 UTC",
            "updated_at": "2019-11-20 17:49:52 UTC",
            "name": "Applicable errata",
            "id": 112
        },
        {
            "created_at": "2019-11-20 17:49:52 UTC",
            "updated_at": "2019-11-20 17:49:52 UTC",
            "name": "Applied Errata",
            "id": 113
        },
        {
            "created_at": "2019-11-30 16:15:24 UTC",
            "updated_at": "2019-11-30 16:15:24 UTC",
            "name": "Hosts - complete list",
            "id": 158
        },
        {
            "created_at": "2019-11-20 17:49:52 UTC",
            "updated_at": "2019-11-20 17:49:52 UTC",
            "name": "Host statuses",
            "id": 114
        },
        {
            "created_at": "2019-11-20 17:49:52 UTC",
            "updated_at": "2019-11-20 17:49:52 UTC",
            "name": "Registered hosts",
            "id": 115
        },
        {
            "created_at": "2019-11-20 17:49:52 UTC",
            "updated_at": "2019-11-20 17:49:52 UTC",
            "name": "Subscriptions",
            "id": 116
        }
    ]
}
```


2. Note the **id** of the template that you want to export, and use the following request to export the template:

Example request:

```
$ curl --insecure --output /tmp/_Example_Export_Template.erb_ \
--user admin:password --request GET --config \
https://satellite.example.com/api/report_templates/158/export
```

Note that **158** is an example ID of the template to export.

In this example, the exported template is redirected to **host_complete_list.erb**.

8.5. IMPORTING REPORT TEMPLATES

You can import a report template into the body of a new template that you want to create. Note that using the web UI, you can only import templates individually. For bulk actions, use the Satellite API. For more information, see Section 8.6, "Importing Report Templates Using the Satellite API" .

Prerequisite

- You must have exported templates from Satellite to import them to use in new templates. For more information see Section 8.3, "Exporting Report Templates".

Procedure

1. In the Satellite web UI, navigate to **Monitor** > **Report Templates**, and in the upper right of the Report Templates window, click **Create Template**.
2. On the upper right of the **Editor** tab, click the folder icon, and select the **.erb** file that you want to import.
3. Edit the template to suit your requirements, and click **Submit**.

For more information about customizing your new template, see Appendix A, *Template Writing Reference*.

8.6. IMPORTING REPORT TEMPLATES USING THE SATELLITE API

You can use the Satellite API to import report templates into Satellite. Importing report templates using the Satellite API automatically parses the report template metadata and assigns organizations and locations. For more information about using the Satellite API, see the API Guide.

Prerequisite

- Create a template using **.erb** syntax or export a template from another Satellite.
  For more information about writing templates, see Appendix A, *Template Writing Reference*.

  For more information about exporting templates from Satellite, see Section 8.4, "Exporting Report Templates Using the Satellite API".

Procedure

1. Use the following example to format the template that you want to import to a **.json** file:

```
# cat Example_Template.json
{
   "name": "Example Template Name",
   "template": "
   Enter ERB Code Here
"
}
```

Example JSON File with ERB Template:

```
{
   "name": "Hosts - complete list",
   "template": "
<%#
name: Hosts - complete list
snippet: false
template_inputs:
- name: host
  required: false
  input_type: user
  advanced: false
  value_type: plain
  resource_type: Katello::ActivationKey
model: ReportTemplate
-%>
<% load_hosts(search: input('host')).each_record do |host| -%>
<%
     report_row(
        'Server FQND': host.name
     )
-%>
<%  end -%>
<%= report_render %>
"
}
```

2. Use the following request to import the template:

```
$ curl --insecure --user admin:redhat \
--data @Example_Template.json --header "Content-Type:application/json" \
--request POST --config https://satellite.example.com/api/report_templates/import
```

3. Use the following request to retrieve a list of report templates and validate that you can view the template in Satellite:

```
$ curl --insecure --user admin:redhat \
 --request GET --config https://satellite.example.com/api/report_templates | json_reformat
```

8.7. CREATING A REPORT TEMPLATE TO MONITOR ENTITLEMENTS

You can use a report template to return a list of hosts with a certain subscription and to display the number of cores for those hosts.

For more information about writing templates, see Appendix A, *Template Writing Reference*.

Procedure

1. In the Satellite web UI, navigate to **Monitor** > **Report Templates**, and click **Create Template**.
2. Optional: In the **Editor** field, use the **<%# >** tags to add a comment with information that might be useful for later reference. For example:

```
<%#
name: Entitlements
snippet: false
model: ReportTemplate
require:
- plugin: katello
  version: 3.14.0
-%>
```

3. Add a line with the **load_hosts()** macro and populate the macro with the following method and variables:

```
<%- load_hosts(includes: [:lifecycle_environment, :operatingsystem, :architecture,
:content_view, :organization, :reported_data, :subscription_facet, :pools =>
[:subscription]]).each_record do |host| -%>
```

To view a list of variables you can use, click the **Help** tab and in the **Safe mode methods and variables** table, find the **Host::Managed** row.

4. Add a line with the **host.pools** variable with the **each** method, for example:

```
<%- host.pools.each do |pool| -%>
```

5. Add a line with the **report_row()** method to create a report and add the variables that you want to target as part of the report:

```
<%-    report_row(
        'Name': host.name,
        'Organization': host.organization,
        'Lifecycle Environment': host.lifecycle_environment,
        'Content View': host.content_view,
        'Host Collections': host.host_collections,
        'Virtual': host.virtual,
        'Guest of Host': host.hypervisor_host,
        'OS': host.operatingsystem,
        'Arch': host.architecture,
        'Sockets': host.sockets,
        'RAM': host.ram,
        'Cores': host.cores,
        'SLA': host_sla(host),
        'Products': host_products(host),
        'Subscription Name': sub_name(pool),
        'Subscription Type': pool.type,
        'Subscription Quantity': pool.quantity,
        'Subscription SKU': sub_sku(pool),
        'Subscription Contract': pool.contract_number,
        'Subscription Account': pool.account_number,
        'Subscription Start': pool.start_date,
        'Subscription End': pool.end_date,
        'Subscription Guest': registered_through(host)
        ) -%>
```

6. Add end statements to the template:

```
<%-   end -%>
<%- end -%>
```

7. To generate a report, you must add the <%= report_render -%> macro:

```
<%= report_render -%>
```

8. Click **Submit** to save the template.

8.8. REPORT TEMPLATE SAFE MODE

When you create report templates in Satellite, safe mode is enabled by default. Safe mode limits the macros and variables that you can use in the report template. Safe mode prevents rendering problems and enforces best practices in report templates. The list of supported macros and variables is available in the Satellite web UI.

To view the macros and variables that are available, in the Satellite web UI, navigate to **Monitor** > **Report Templates** and click **Create Template**. In the Create Template window, click the **Help** tab and expand **Safe mode methods**.

While safe mode is enabled, if you try to use a macro or variable that is not listed in **Safe mode methods**, the template editor displays an error message.

To view the status of safe mode in Satellite, in the Satellite web UI, navigate to **Administer** > **Settings** and click the **Provisioning** tab. Locate the **Safemode rendering** row to check the value.

CHAPTER 9. 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.

9.1. 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**.

For CLI Users

To create a host collection, enter the following command:

```
# hammer host-collection create \
--organization "Your_Organization" \
--name hc_name
```

9.2. 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**.

9.3. 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**.

9.4. 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 to add it to a Host Collection. For more information about registering hosts, Chapter 3, *Registering Hosts*.

Note that if you add a host to a host collection, the Satellite auditing system does not log the change.

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**.

For CLI Users

To add hosts to a host collection, enter the following command:

```
# hammer host-collection add-host \
--id hc_ID \
--host-ids host_ID1,host_ID2...
```

9.5. REMOVING A HOST FROM A HOST COLLECTION

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

Note that if you remove a host from a host collection, the host collection record in the database is not modified so the Satellite auditing system does not log the change.

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**.

9.6. ADDING CONTENT TO A HOST COLLECTION

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

9.6.1. 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.

9.6.2. 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.

9.7. 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.

9.8. 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 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 10. CONFIGURING AND SETTING UP REMOTE JOBS

Use this section as a guide to configuring Satellite to execute jobs on remote hosts.

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

10.1. ABOUT RUNNING JOBS ON HOSTS

You can run jobs on hosts remotely from Capsules using shell scripts or Ansible tasks and playbooks. This is referred to as remote execution.

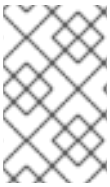
For custom Ansible roles that you create, or roles that you download, you must install the package containing the roles on the Capsule base operating system. Before you can use Ansible roles, you must import the roles into Satellite from the Capsule where they are installed.

Communication occurs through Capsule Server, which means that Satellite Server does not require direct access to the target host, and can scale to manage many hosts. Remote execution uses the SSH service that must be enabled and running on the target host. Ensure that the remote execution Capsule has access to port 22 on the target hosts.

Satellite uses ERB syntax job templates. For more information, see Template Writing Reference in the *Managing Hosts* guide.

Several job templates for shell scripts and Ansible are included by default. For more information, see Section 10.14, “Setting up Job Templates”.

By default, Satellite Server is configured to use the Katello Agent rather than remote execution. To change this setting, navigate to **Administer** > **Settings**, click **Content**, and change the **Use remote execution by default** setting.



NOTE

Any Capsule Server base operating 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 Capsules.

You can run jobs on multiple hosts at once, and you can use variables in your commands for more granular control over the jobs you run. You can use host facts and parameters to populate the variable values.

In addition, you can specify custom values for templates when you run the command.

For more information, see Section 10.15, “Executing a Remote Job” .

10.2. REMOTE EXECUTION WORKFLOW

When you run a remote job on hosts, for every host, Satellite performs the following actions to find a remote execution Capsule to use.

Satellite searches only for Capsules that have the remote execution feature enabled.

1. Satellite finds the host’s interfaces that have the **Remote execution** check box selected.

2. Satellite finds the subnets of these interfaces.
3. Satellite finds remote execution Capsules assigned to these subnets.
4. From this set of Capsules, Satellite selects the Capsule that has the least number of running jobs. By doing this, Satellite ensures that the jobs load is balanced between remote execution Capsules.
5. If Satellite does not find a remote execution Capsule at this stage, and if the **Fallback to Any Capsule** setting is enabled, Satellite adds another set of Capsules to select the remote execution Capsule from. Satellite selects the most lightly loaded Capsule from the following types of Capsules that are assigned to the host:
    - DHCP, DNS and TFTP Capsules assigned to the host's subnets
    - DNS Capsule assigned to the host's domain
    - Realm Capsule assigned to the host's realm
    - Puppet Master Capsule
    - Puppet CA Capsule
    - OpenSCAP Capsule
6. If Satellite does not find a remote execution Capsule at this stage, and if the **Enable Global Capsule** setting is enabled, Satellite selects the most lightly loaded remote execution Capsule from the set of all Capsules in the host's organization and location to execute a remote job.

10.3. 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.

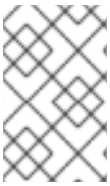
You can clone the Remote Execution User role and customize its filter for increased granularity. If you adjust the filter with the **view_job_templates** permission, the user can only see and trigger jobs based on matching job templates. You can use the **view_hosts** and **view_smart_proxies** permissions 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 = webservers
```

The first line in this example permits the user to apply the **Reboot** template to 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.

10.4. CREATING A JOB TEMPLATE:

1. Navigate to **Hosts** > **Job templates**.
2. Click **New Job Template**.
3. Click the **Template** tab, and in the **Name** field, enter a unique name for your job template.
4. Select **Default** to make the template available for all organizations and locations.
5. Create the template directly in the template editor or upload it from a text file by clicking **Import**.
6. Optional: In the **Audit Comment** field, add information about the change.
7. Click the **Job** tab, and in the **Job category** field, enter your own category or select from the default categories listed in Section B.2, "Default Job Template Categories".
8. Optional: In the **Description Format** field, enter a description template. For example, **Install package %{package_name}**. You can also use **%{template_name}** and **%{job_category}** in your template.
9. From the **Provider Type** list, select **SSH** for shell scripts and **Ansible** for Ansible tasks or playbooks.
10. Optional: In the **Timeout to kill** field, enter a timeout value to terminate the job if it does not complete.
11. Optional: Click **Add Input** to define an input parameter. Parameters are requested when executing the job and do not have to be defined in the template. For examples, see the **Help** tab.
12. Optional: Click **Foreign input set** to include other templates in this job.
13. Optional: In the **Effective user** area, configure a user if the command cannot use the default **remote_execution_effective_user** setting.
14. Optional: If this template is a snippet to be included in other templates, click the **Type** tab and select **Snippet**.
15. Click the **Location** tab and add the locations where you want to use the template.
16. Click the **Organizations** tab and add the organizations where you want to use the template.
17. Click **Submit** to save your changes.

You can extend and customize job templates by including other templates in the template syntax. For more information, see the appendices in the *Managing Hosts* guide.

For CLI Users

To create a job template using a template-definition file, enter the following command:

```
# hammer job-template create \
--file "path_to_template_file" \
--name "template_name" \
--provider-type SSH \
--job-category "category_name"
```

10.5. CONFIGURING THE FALLBACK TO ANY CAPSULE REMOTE EXECUTION SETTING IN SATELLITE

You can enable the **Fallback to Any Capsule** setting to configure Satellite to search for remote execution Capsules from the list of Capsules that are assigned to hosts. This can be useful if you need to run remote jobs on hosts that have no subnets configured or if the hosts' subnets are assigned to Capsules that do not have the remote execution feature enabled.

If the **Fallback to Any Capsule** setting is enabled, Satellite adds another set of Capsules to select the remote execution Capsule from. Satellite also selects the most lightly loaded Capsule from the set of all Capsules assigned to the host, such as the following:

- DHCP, DNS and TFTP Capsules assigned to the host's subnets
- DNS Capsule assigned to the host's domain
- Realm Capsule assigned to the host's realm
- Puppet Master Capsule
- Puppet CA Capsule
- OpenSCAP Capsule

Procedure

1. In the Satellite web UI, navigate to **Administer** > **Settings**.
2. Click **RemoteExecution**.
3. Configure the **Fallback to Any Capsule** setting.

For CLI Users

Enter the **hammer settings set** command on Satellite to configure the **Fallback to Any Capsule** setting. For example, to set the value to **true**, enter the following command:

```
# hammer settings set --name=remote_execution_fallback_proxy --value=true
```

10.6. CONFIGURING THE GLOBAL CAPSULE REMOTE EXECUTION SETTING IN SATELLITE

By default, Satellite searches for remote execution Capsules in hosts' organizations and locations regardless of whether Capsules are assigned to hosts' subnets or not. You can disable the **Enable Global Capsule** setting if you want to limit the search to the Capsules that are assigned to hosts' subnets.

If the **Enable Global Capsule** setting is enabled, Satellite adds another set of Capsules to select the remote execution Capsule from. Satellite also selects the most lightly loaded remote execution Capsule from the set of all Capsules in the host's organization and location to execute a remote job.

Procedure

1. In the Satellite web UI, navigate to **Administer** > **Settings**.
2. Click **RemoteExection**.
3. Configure the **Enable Global Capsule** setting.

For CLI Users

Enter the **hammer settings set** command on Satellite to configure the **Enable Global Capsule** setting. For example, to set the value to **true**, enter the following command:

```
# hammer settings set --name=remote_execution_global_proxy --value=true
```

10.7. CONFIGURING SATELLITE TO USE AN ALTERNATIVE DIRECTORY TO EXECUTE REMOTE JOBS ON HOSTS

By default, Satellite uses the **/var/tmp** directory on the client system to execute the remote execution jobs. If the client system has **noexec** set for the **/var/** volume or file system, you must configure Satellite to use an alternative directory because otherwise the remote execution job fails since the script cannot be run.

Procedure

To use an alternative directory, complete this procedure.

1. Create a new directory, for example *new_place*:

```
# mkdir /remote_working_dir
```

2. Copy the SELinux context from the default **var** directory:

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

3. Edit the **remote_working_dir** setting in the **/etc/foreman-proxy/settings.d/remote_execution_ssh.yml** file to point to the required directory, for example:

```
:remote_working_dir: /remote_working_dir
```

10.8. DISTRIBUTING SSH KEYS FOR REMOTE EXECUTION

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

Use one of the following methods to distribute the public SSH key from Capsule to target hosts:

1. Section 10.9, "Distributing SSH Keys for Remote Execution Manually" .
2. Section 10.10, "Using the Satellite API to Obtain SSH Keys for Remote Execution" .
3. Section 10.11, "Configuring a Kickstart Template to Distribute SSH Keys during Provisioning" .

Satellite distributes SSH keys for the remote execution feature to the hosts provisioned from Satellite by default.

If the hosts are running on Amazon Web Services, enable password authentication. For more information, see https://aws.amazon.com/premiumsupport/knowledge-center/new-user-accounts-linux-instance.

10.9. DISTRIBUTING SSH KEYS FOR REMOTE EXECUTION MANUALLY

To distribute SSH keys manually, complete the following steps:

Procedure

1. Enter the following command on Capsule. Repeat for each target host you want to manage:

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

2. To confirm that the key was successfully copied to the target host, enter the following command on Capsule:

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

10.10. USING THE SATELLITE API TO OBTAIN SSH KEYS FOR REMOTE EXECUTION

To use the Satellite API to download the public key from Capsule, complete this procedure on each target host.

Procedure

1. On the target host, create the **~/.ssh** directory to store the SSH key:

```
# mkdir ~/.ssh
```

2. Download the SSH key from Capsule:

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

3. Configure permissions for the **~/.ssh** directory:

```
# chmod 700 ~/.ssh
```

4. Configure permissions for the **authorized_keys** file:

```
# chmod 600 ~/.ssh/authorized_keys
```

10.11. CONFIGURING A KICKSTART TEMPLATE TO DISTRIBUTE SSH KEYS DURING PROVISIONING

You can add a **remote_execution_ssh_keys** snippet to your custom kickstart template to deploy SSH Keys to hosts during provisioning. Kickstart templates that Satellite ships include this snippet by default. Therefore, Satellite copies the SSH key for remote execution to the systems during provisioning.

Procedure

- To include the public key in newly-provisioned hosts, add the following snippet to the Kickstart template that you use:

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

10.12. CONFIGURING A KEYTAB FOR KERBEROS TICKET GRANTING TICKETS

Use this procedure to configure Satellite to use a keytab to obtain Kerberos ticket granting tickets. If you do not set up a keytab, you must manually retrieve tickets.

Procedure

To ensure that the **foreman-proxy** user on Satellite can obtain Kerberos ticket granting tickets, complete the following steps:

1. Find the ID of the **foreman-proxy** user:

```
# id -u foreman-proxy
```

2. Modify the **umask** value so that new files have the permissions **600**:

```
# umask 077
```

3. Create the directory for the keytab:

```
# mkdir -p "/var/kerberos/krb5/user/USER_ID"
```

4. Create a keytab or copy an existing keytab to the directory:

```
# cp your_client.keytab /var/kerberos/krb5/user/USER_ID/client.keytab
```

5. Change the directory owner to the **foreman-proxy** user:

```
# chown -R foreman-proxy:foreman-proxy "/var/kerberos/krb5/user/USER_ID"
```

6. Ensure that the keytab file is read-only:

```
# chmod -wx "/var/kerberos/krb5/user/USER_ID/client.keytab"
```

7. Restore the SELinux context:

```
# restorecon -RvF /var/kerberos/krb5
```

10.13. CONFIGURING KERBEROS AUTHENTICATION FOR REMOTE EXECUTION

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

Procedure

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

1. 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
```

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

3. 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.

10.14. SETTING UP JOB TEMPLATES

Satellite provides default job templates that you can use for executing jobs. To view the list of job templates, navigate to **Hosts** > **Job templates**. If you want to use a template without making changes, proceed to Section 10.15, “Executing a Remote Job” .

You can use default templates as a base for developing your own. Default job templates are locked for editing. Clone the template and edit the clone.

1. To clone a template, in the **Actions** column, select **Clone**.

2. Enter a unique name for the clone and click **Submit** to save the changes.

Job templates use the Embedded Ruby (ERB) syntax. For more information about writing templates, see the Template Writing Reference in the *Managing Hosts* guide.

Ansible Considerations

To create an Ansible job template, use the following procedure and instead of ERB syntax, use YAML syntax. Begin the template with **---**. You can embed an Ansible playbook YAML file into the job template body. You can also add ERB syntax to customize your YAML Ansible template. You can also import Ansible playbooks in Satellite. For more information, see Synchronizing Repository Templates in the *Managing Hosts* guide.

Parameter Variables

At run time, job templates can accept parameter variables that you define for a host. 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. If you do not want your Ansible job template to accept parameter variables at run time, in the Satellite web UI, navigate to **Administer** > **Settings** and click the **Ansible** tab. In the **Top level Ansible variables** row, change the **Value** parameter to **No**.

10.15. EXECUTING A REMOTE JOB

You can execute a job that is based on a job template against one or more hosts.

Procedure

1. Navigate to **Hosts** > **All Hosts** and select the target hosts on which you want to execute a remote job. You can use the search field to filter the host list.
2. From the **Select Action** list, select **Schedule Remote Job**.
3. On the **Job invocation** page, define the main job settings:
4. Select the **Job category** and the **Job template** you want to use.
5. Optional: Select a stored search string in the **Bookmark** list to specify the target hosts.
6. Optional: Further limit the targeted hosts by entering 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 lists the targeted hosts.
7. The remaining settings depend on the selected job template. See Section 10.4, "Creating a Job Template:" for information on adding custom parameters to a template.
8. Optional: To configure advanced settings for the job, click **Display advanced fields**. 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 the 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.
    - **Timeout to kill** 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.
- **Execution ordering** determines the order in which the job is executed on hosts: alphabetical or randomized.
  Concurrency level and **Timeout to kill** settings enable you to tailor job execution to fit your infrastructure hardware and needs.

9. To run 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*.
10. Click **Submit**. This displays the **Job Overview** page, and when the job completes, also displays the status of the job.

For CLI Users

Enter the following command on Satellite:

```
# hammer settings set --name=remote_execution_global_proxy --value=false
```

To execute a remote job with custom parameters, complete the following steps:

1. Find the ID of the job template you want to use:

   ```
   # hammer job-template list
   ```

2. Show the template details to see parameters required by your template:

   ```
   # hammer job-template info --id template_ID
   ```

3. Execute a remote job with custom parameters:

   ```
   # hammer job-invocation create \
   --job-template "template_name" \
   --inputs key1="value",key2="value",... \
   --search-query "query"
   ```

   Replace *query* with the filter expression that defines hosts, for example **"name ~ rex01"**. For more information about executing remote commands with hammer, enter **hammer job-template --help** and **hammer job-invocation --help**.

10.16. MONITORING JOBS

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

Ansible jobs run on batches of 100 hosts, so you cannot cancel a job running on a specific host. A job completes only after the Ansible playbook runs on all hosts in the batch.

Procedure

1. Navigate to the Job page. This page is automatically displayed if you triggered the job with the **Execute now** setting. To monitor scheduled jobs, navigate to **Monitor** > **Jobs** and select the job run you wish to inspect.

2. On the Job page, click the **Hosts** tab. This displays the list of hosts on which the job is running.

3. In the **Host** column, click the name of the host that you want to inspect. This displays the **Detail of Commands** page where you can monitor the job execution in real time.

4. Click **Back to Job** at any time to return to the **Job Details** page.

For CLI Users

To monitor the progress of a job while it is running, complete the following steps:

1. Find the ID of a job:

```
# hammer job-invocation list
```

2. Monitor the job output:

```
# hammer job-invocation output \
--id job_ID \
--host host_name
```

3. Optional: to cancel a job, enter the following command:

```
# hammer job-invocation cancel \
--id job_ID
```

CHAPTER 11. HOST MANAGEMENT WITHOUT GOFERD AND KATELLO AGENT

The **goferd** service that is used by the Katello agent to manage packages on content hosts consumes large amount of resources. To reduce memory and CPU load on content hosts, you can manage packages through remote execution.

Note that the Katello agent is deprecated and will be removed in a future Satellite version; therefore, using remote execution will be the only way to manage packages on hosts.

Prerequisites

- You have enabled the Satellite Tools 6.8 repository on Satellite Server. For more information, see Enabling the Satellite Tools 6.8 Repository in *Installing Satellite Server from a Connected Network*.
- You have synchronized the Satellite Tools 6.8 repository on Satellite Server. For more information, see Synchronizing the Satellite Tools 6.8 Repository in *Installing Satellite Server from a Connected Network*.
- You have enabled the Satellite Tools 6.8 repository on content hosts.

Procedure

1. Install the **katello-host-tools** package on content hosts:

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

2. Stop the goferd service on content hosts:

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

3. Disable the goferd service on content hosts:

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

4. Remove the Katello agent on content hosts:

WARNING

If your host is installed on Red Hat Virtualization version 4.4 or lower, do not remove the **katello-agent** package because the removed dependencies corrupt the host.

```
# yum remove katello-agent
```

5. Distribute the SSH keys to the content hosts. For more information, see Section 10.8, “Distributing SSH Keys for Remote Execution”.

6. In the Satellite web UI, navigate to **Administer** > **Settings**.
7. Select the **Content** tab.
8. Set the **Use remote execution by default** parameter to **Yes**.

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

Hammer Limitations

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 the Satellite remote execution feature to apply errata.

For example, enter the following command to 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
```

CHAPTER 12. SYNCHRONIZING TEMPLATE REPOSITORIES

In Satellite, you can synchronize repositories of job templates, provisioning templates, report templates, and partition table templates between Satellite Server and a version control system or local directory. In this chapter, a Git repository is used for demonstration purposes.

This section details the workflow for:

- installing and configuring the TemplateSync plug-in
- performing exporting and importing tasks

12.1. ENABLING THE TEMPLATESYNC PLUG-IN

1. To enable the plug-in on your Satellite Server, enter the following command:

```
# satellite-installer --enable-foreman-plugin-templates
```

2. To verify that the plug-in is installed correctly, ensure **Administer** > **Settings** includes the **TemplateSync** menu.

12.2. CONFIGURING THE TEMPLATESYNC PLUG-IN

In the Satellite web UI, navigate to **Administer** > **Settings** > **TemplateSync** to configure the plug-in. The following table explains the attributes behavior. Note that some attributes are used only for importing or exporting tasks.

Table 12.1. Synchronizing Templates Plug-in configuration

Parameter	API parameter name	Meaning on importing	Meaning on exporting
Associate	**associate** Accepted values: **always**, **new**, **never**	Associates templates with OS, Organization, and Location based on metadata.	N/A
Branch	**branch**	Specifies the default branch in Git repository to read from.	Specifies the default branch in Git repository to write to.
Dirname	**dirname**	Specifies the subdirectory under the repository to read from.	Specifies the subdirectory under the repository to write to.
Filter	**filter**	Imports only templates with names that match this regular expression.	Exports only templates with names that match this regular expression.
Force import	**force**	Imported templates overwrite locked templates with the same name.	N/A

Parameter	API parameter name	Meaning on importing	Meaning on exporting
Lock templates	**lock**	Do not overwrite existing templates when you import a new template with the same name, unless **Force import** is enabled.	N/A
Metadata export mode	**metadata_export_mode** Accepted values: **refresh**, **keep**, **remove**	N/A	Defines how metadata is handled when exporting: • **Refresh** – remove existing metadata from the template content and generate new metadata based on current assignments and attributes. • **Keep** – retain the existing metadata. • **Remove** – export template without metadata. Useful if you want to add metadata manually.
Negate	**negate** Accepted values: **true**, **false**	Imports templates ignoring the filter attribute.	Exports templates ignoring the filter attribute.
Prefix	**prefix**	Adds specified string to the beginning of the template if the template name does not start with the prefix already.	N/A
Repo	**repo**	Defines the path to the repository to synchronize from.	Defines the path to a repository to export to.

Parameter	API parameter name	Meaning on importing	Meaning on exporting
Verbosity	**verbose** Accepted values: **true**, **false**	Enables writing verbose messages to the logs for this action.	N/A

12.3. IMPORTING AND EXPORTING TEMPLATES

You can import and export templates using the Satellite web UI, Hammer CLI, or Satellite API. Satellite API calls use the role-based access control system, which enables the tasks to be executed as any user. You can synchronize templates with a version control system, such as Git, or a local directory.

12.3.1. Importing Templates

You can import templates from a repository of your choice. You can use different protocols to point to your repository, for example **/tmp/dir**, **git://example.com**, **https://example.com**, and **ssh://example.com**.

Prerequisites

- Each template must contain the location and organization that the template belongs to. This applies to all template types. Before you import a template, ensure that you add the following section to the template:

```
<%#
kind: provision
name: My Kickstart File
oses:
- RedHat 7
- RedHat 6
locations:
- First Location
- Second Location
organizations:
- Default Organization
- Extra Organization
%>
```

Procedure

1. In the Satellite web UI, navigate to **Hosts** > **Sync Templates**.
2. Click **Import**.
3. Each field is populated with values configured in **Administer** > **Settings** > **TemplateSync**. Change the values as required for the templates you want to import. For more information about each field, see Section 12.2, “Configuring the TemplateSync plug-in”.
4. Click **Submit**.

The Satellite web UI displays the status of the import. The status is not persistent; if you leave the status page, you cannot return to it.

For CLI Users

- To import a template from a repository, enter the following command:

```
$ hammer import-templates \
    --prefix '[Custom Index] ' \
    --filter '.*Template Name$' \
    --repo https://github.com/examplerepo/exampledirectory \
    --branch my_branch \
    --organization 'Default Organization'
```

For better indexing and management of your templates, use **--prefix** to set a category for your templates. To select certain templates from a large repository, use **--filter** to define the title of the templates that you want to import. For example **--filter '.*Ansible Default$'** imports various Ansible Default templates.

12.3.2. Exporting Templates

You can export templates to a version control server, such as a Git repository.

Procedure

1. In the Satellite web UI, navigate to **Hosts** > **Sync Templates**.
2. Click **Export**.
3. Each field is populated with values configured in **Administer** > **Settings** > **TemplateSync**. Change the values as required for the templates you want to export. For more information about each field, see Section 12.2, “Configuring the TemplateSync plug-in”.
4. Click **Submit**.

The Satellite web UI displays the status of the export. The status is not persistent; if you leave the status page, you cannot return to it.

For CLI Users

1. Clone a local copy of your Git repository:

```
$ git clone https://github.com/theforeman/community-templates /custom/templates
```

2. Change the owner of your local directory to the **foreman** user, and change the SELinux context with the following commands:

```
# chown -R foreman:foreman /custom/templates
# chcon -R -t httpd_sys_rw_content_t /custom/templates
```

3. To export the templates to your local repository, enter the following command:

```
hammer export-templates --organization 'Default Organization' --repo /custom/templates
```

12.3.3. Synchronizing Templates Using the Satellite API

Prerequisites

- Each template must contain the location and organization that the template belongs to. This applies to all template types. Before you import a template, ensure that you add the following section to the template:

```
<%#
kind: provision
name: My Kickstart File
oses:
- RedHat 7
- RedHat 6
locations:
- First Location
- Second Location
organizations:
- Default Organization
- Extra Organization
%>
```

Procedure

1. Configure a version control system that uses SSH authorization, for example gitosis, gitolite, or git daemon.

2. Configure the TemplateSync plug-in settings on a **TemplateSync** tab.

   a. Change the **Branch** setting to match the target branch on a Git server.

   b. Change the **Repo** setting to match the Git repository. For example, for the repository located in **git@git.example.com/templates.git** set the setting into **ssh://git@git.example.com/templates.git**.

3. Accept Git SSH host key as the **foreman** user:

```
# sudo -u foreman ssh git.example.com
```

   You can see the **Permission denied, please try again.** message in the output, which is expected, because the SSH connection cannot succeed yet.

4. Create an SSH key pair if you do not already have it. Do not specify a passphrase.

```
# sudo -u foreman ssh-keygen
```

5. Configure your version control server with the public key from your Satellite, which resides in **/usr/share/foreman/.ssh/id_rsa.pub**.

6. Export templates from your Satellite Server to the version control repository specified in the **TemplateSync** menu:

```
$ curl -H "Accept:application/json,version=2" \
-H "Content-Type:application/json" \
-u login:password \
-k https://_satellite.example.com/api/v2/templates/export \
```

```
-X POST

{"message":"Success"}
```

7. Import templates to Satellite Server after their content was changed:

```
$ curl -H "Accept:application/json,version=2" \
-H "Content-Type:application/json" \
-u login:password \
-k https://_satellite.example.com/api/v2/templates/import \
-X POST

{“message”:”Success”}
```

Note that templates provided by Satellite are locked and you cannot import them by default. To overwrite this behavior, change the **Force import** setting in the **TemplateSync** menu to **yes** or add the **force** parameter **-d '{ "force": "true" }'** to the import command.

12.3.4. Synchronizing Templates with a Local Directory Using the Satellite API

Synchronizing templates with a local directory is useful if you have configured a version control repository in the local directory. That way, you can edit templates and track the history of edits in the directory. You can also synchronize changes to Satellite Server after editing the templates.

Prerequisites

- Each template must contain the location and organization that the template belongs to. This applies to all template types. Before you import a template, ensure that you add the following section to the template:

```
<%#
kind: provision
name: My Kickstart File
oses:
- RedHat 7
- RedHat 6
locations:
- First Location
- Second Location
organizations:
- Default Organization
- Extra Organization
%>
```

Procedure

1. Create the directory where templates are stored and apply appropriate permissions and SELinux context:

```
# mkdir -p /usr/share/templates_dir/
# chown foreman /usr/share/templates_dir/
# chcon -t httpd_sys_rw_content_t /usr/share/templates_dir/ -R
```

2. Change the **Repo** setting on the **TemplateSync** tab to match the export directory **/usr/share/templates_dir/**.

3. Export templates from your Satellite Server to a local directory:

```
$ curl -H "Accept:application/json,version=2" \
-H "Content-Type:application/json" \
-u login:password \
-k https://_satellite.example.com/api/v2/templates/export \
-X POST \

{"message":"Success"}
```

4. Import templates to Satellite Server after their content was changed:

```
$ curl -H "Accept:application/json,version=2" \
-H "Content-Type:application/json" \
-u login:password \
-k https://_satellite.example.com/api/v2/templates/import \
-X POST

{“message”:”Success”}
```

Note that templates provided by Satellite are locked and you cannot import them by default. To overwrite this behavior, change the **Force import** setting in the **TemplateSync** menu to **yes** or add the **force** parameter **-d '{ "force": "true" }'** to the import command.

NOTE

You can override default API settings by specifying them in the request with the **-d** parameter. The following example exports templates to the **git.example.com/templates** repository:

```
$ curl -H "Accept:application/json,version=2" \
-H "Content-Type:application/json" \
-u login:password \
-k https://satellite.example.com/api/v2/templates/export \
-X POST \
-d "{\"repo\":\"git.example.com/templates\"}"
```

12.4. ADVANCED GIT CONFIGURATION

You can perform additional Git configuration for the TemplateSync plug-in using the command line or editing the **.gitconfig** file.

Accepting a self-signed Git certificate

If you are using a self-signed certificate authentication on your Git server, validate the certificate with the **git config http.sslCAPath** command.

For example, the following command verifies a self-signed certificate stored in **/cert/cert.pem**:

```
# sudo -u foreman git config --global http.sslCAPath cert/cert.pem
```

For a complete list of advanced options, see the **git-config** manual page.

12.5. UNINSTALLING THE PLUG-IN

To avoid errors after removing the foreman_templates plugin:

1. Disable the plug-in using the Satellite installer:

```
# satellite-installer --no-enable-foreman-plugin-templates
```

2. Clean custom data of the plug-in. The command does not affect any templates that you created.

```
# foreman-rake templates:cleanup
```

3. Uninstall the plug-in:

```
# satellite-maintain packages remove foreman-plugin-templates
```

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 the following cases:

Provisioning templates

For more information, see Creating Provisioning Templates in the *Provisioning Guide*.

Remote execution job templates

For more information, see Chapter 10, *Configuring and Setting up Remote Jobs*.

Report templates

For more information, see Chapter 8, *Using Report Templates to Monitor Hosts*.

Templates for partition tables

For more information, see Creating Partition Tables in the *Provisioning Guide*.

Smart Variables

For more information, see Configuring Smart Variables in the *Puppet Guide*.

Smart Class Parameters

For more information, see Configuring Smart Class Parameters in the *Puppet Guide*.

This section provides an overview of Satellite-specific macros 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**, **Monitor** > **Report 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 tags are the most important and commonly used in ERB templates:

<% %>

All Ruby code is enclosed within **<% %>** in an ERB template. The code is executed when the template is rendered. It can contain Ruby control flow structures as well as Satellite-specific macros 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 -%>
```

Note that this template silently performs an action with a service and returns nothing at the output.

<%= %>

This provides the same functionality as **<% %>** but when the template is executed, the code output is inserted into the template. This is useful for variable substitution, for example:

Example input:

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

Example rendering:

```
host.example.com
```

Example input:

```
<% server_name = @host.fqdn %>
<%= server_name %>
```

Example rendering:

```
host.example.com
```

Note that if you enter an incorrect variable, no output is returned. However, if you try to call a method on an incorrect variable, the following error message returns:

Example input:

```
<%= @example_incorrect_variable.fqdn -%>
```

Example rendering:

```
undefined method `fqdn' for nil:NilClass
```

<% -%>, <%= -%>

By default, a newline character is inserted after a Ruby block if it is closed at the end of a line:

Example input:

```
<%= "line1" %>
<%= "line2" %>
```

Example rendering:

```
line1
line2
```

To change the default behavior, modify the enclosing mark with **-%>**:

Example input:

```
<%= "line1" -%>
<%= "line2" %>
```

Example rendering:

```
line1line2
```

This is used to reduce the number of lines, where Ruby syntax permits, in rendered templates. White spaces in ERB tags are ignored.

An example of how this would be used in a report template to remove unnecessary newlines between a FQDN and IP address:

Example input:

```
<%= @host.fqdn -%>
<%= @host.ip -%>
```

Example rendering:

```
host.example.com10.10.181.216
```

<%# %>

Encloses a comment that is ignored during template rendering:

Example input:

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

This generates no output.

Indentation in ERB templates

Because of the varying lengths of the ERB tags, indenting the ERB syntax might seem messy. ERB syntax ignore white space. One method of handling the indentation is to declare the ERB tag at the beginning of each new line and then use white space within the ERB tag to outline the relationships within the syntax, for example:

```
<%- load_hosts.each do |host| -%>
<%-   if host.build? %>
<%=     host.name %> build is in progress
<%-   end %>
<%- end %>
```

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 list. 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 list next to the selected template to view it's rendered version.

A.3. GENERIC SATELLITE-SPECIFIC MACROS

This section lists Satellite-specific macros for ERB templates.

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

Table A.1. Generic Macros

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.

A.4. TEMPLATES MACROS

If you want to write custom templates, you can use some of the following macros.

Depending on the template type, some of the following macros have different requirements.

For more information about the available macros for report templates, in the Satellite web UI, navigate to **Monitor** > **Report Templates**, and click **Create Template**. In the Create Template window, click the **Help** tab.

For more information about the available macros for job templates, in the Satellite web UI, navigate to **Hosts** > **Job Templates**, and click the **New Job Template**. In the New Job Template window, click the **Help** tab.

input

Using the **input** macro, you can customize the input data that the template can work with. You can define the input name, type, and the options that are available for users. For report templates, you can only use user inputs. When you define a new input and save the template, you can then reference the input in the ERB syntax of the template body.

```
<%= input('cpus') %>
```

This loads the value from user input **cpus**.

load_hosts

Using the **load_hosts** macro, you can generate a complete list of hosts.

```
<%- load_hosts().each_record do |host| -%>
<%=     host.name %>
```

Use the **load_hosts** macro with the **each_record** macro to load records in batches of 1000 to reduce memory consumption.

If you want to filter the list of hosts for the report, you can add the option **search: input('*Example_Host*')**:

```
<% load_hosts(search: input('Example_Host')).each_record do |host| -%>
<%=  host.name %>
<% end -%>
```

In this example, you first create an input that you then use to refine the search criteria that the **load_hosts** macro retrieves.

report_row

Using the **report_row** macro, you can create a formatted report for ease of analysis. The **report_row** macro requires the **report_render** macro to generate the output.

Example input:

```
<%- load_hosts(search: input('Example_Host')).each_record do |host| -%>
<%-   report_row(
        'Server FQDN': host.name
      ) -%>
<%- end -%>
<%= report_render -%>
```

Example rendering:

```
Server FQDN
host1.example.com
host2.example.com
host3.example.com
host4.example.com
host5.example.com
host6.example.com
```

You can add extra columns to the report by adding another header. The following example adds IP addresses to the report:

Example input:

```
<%- load_hosts(search: input('host')).each_record do |host| -%>
<%-   report_row(
      'Server FQDN': host.name,
          'IP': host.ip
      ) -%>
<%- end -%>
<%= report_render -%>
```

Example rendering:

```
Server FQDN,IP
host1.example.com,10.8.30.228
host2.example.com,10.8.30.227
host3.example.com,10.8.30.226
host4.example.com,10.8.30.225
host5.example.com,10.8.30.224
host6.example.com,10.8.30.223
```

report_render

This macro is available only for report templates.
Using the **report_render** macro, you create the output for the report. During the template rendering process, you can select the format that you want for the report. YAML, JSON, HTML, and CSV formats are supported.

```
<%= report_render -%>
```

render_template()

This macro is available only for job templates.
Using this macro, you can render a specific template. You can also enable and define arguments that you want to pass to the template.

A.5. HOST-SPECIFIC VARIABLES

The following variables enable using host data within templates. Note that job templates accept only **@host** variables.

Table A.2. Host Specific Variables and Macros

Name	Description
@host.architecture	The architecture of the host.
@host.bond_interfaces	Returns an array of all bonded interfaces. See Section A.8, "Parsing Arrays".
@host.capabilities	The method of system provisioning, can be either build (for example kickstart) or image.
@host.certname	The SSL certificate name of the host.
@host.diskLayout	The disk layout of the host. Can be inherited from the operating system.
@host.domain	The domain of the host.
@host.environment	The Puppet environment of the host.

Name	Description
@host.facts	Returns a Ruby hash of facts from Facter. For example to access the 'ipaddress' fact from the output, specify @host.facts['ipaddress'].
@host.grub_pass	Returns the host's GRUB password.
@host.hostgroup	The host group of the host.
host_enc['parameters']	Returns a Ruby hash containing information on host parameters. For example, use host_enc['parameters']['lifecycle_environment'] to get the life cycle environment of a host.
@host.image_build?	Returns **true** if the host is provisioned using an image.
@host.interfaces	Contains an array of all available host interfaces including the primary interface. See Section A.8, "Parsing Arrays".
@host.interfaces_with_identifier('IDs')	Returns array of interfaces with given identifier. You can pass an array of multiple identifiers as an input, for example @host.interfaces_with_identifier(['eth0', 'eth1']). See Section A.8, "Parsing Arrays".
@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 Section A.8, "Parsing Arrays".
@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.

Name	Description
@host.operatingsystem.name	The assigned operating system name.
@host.operatingsystem.boot_files_uri(medium_provider)	Full path to the kernel and initrd, returns an array.
@host.os.medium_uri(@host)	The URI used for provisioning (path configured in installation media).
host_param('parameter_name')	Returns the value of the specified host parameter.
host_param_false?('parameter_name')	Returns **false** if the specified host parameter evaluates to false.
host_param_true?('parameter_name')	Returns **true** if the specified host parameter evaluates to true.
@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.puppet_ca_server	The Puppet CA server the host must use.
@host.puppetmaster	The Puppet master the host must 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.

Name	Description
@host.subnet.dns_primary	The primary DNS server of the host.
@host.subnet.dns_secondary	The secondary DNS server of the host.
@host.subnet.gateway	The gateway of the host.
@host.subnet.mask	The subnet mask of the host.
@host.url_for_boot(:initrd)	Full path to the initrd image associated with this host. Not recommended, as it does not interpolate variables.
@host.url_for_boot(:kernel)	Full path to the kernel associated with this host. Not recommended, as it does not interpolate variables, prefer boot_files_uri.
@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.

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 %>
```

A.6. KICKSTART-SPECIFIC VARIABLES

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

Table A.3. Kickstart Specific Variables

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

A.7. CONDITIONAL STATEMENTS

In your templates, you might perform different actions depending on which value exists. To achieve this, you can use conditional statements in your ERB syntax.

In the following example, the ERB syntax searches for a specific host name and returns an output depending on the value it finds:

Example input:

```
<% load_hosts().each_record do |host| -%>
<% if @host.name == "host1.example.com" -%>
<%      result="positive" -%>
<%  else -%>
<%      result="negative" -%>
<%  end -%>
<%= result -%>
```

Example rendering:

```
host1.example.com
positive
```

A.8. PARSING ARRAYS

While writing or modifying templates, you might encounter variables that return arrays. For example, 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.

Finding the Correct Method to Parse an Array

The following procedure is an example that you can use to find the relevant methods to parse arrays in your template. In this example, a report template is used, but the steps are applicable to other templates.

1. To retrieve the NIC of a content host, in this example, using the **@host.interfaces** variable returns class values that you can then use to find methods to parse the array.

   Example input:

   ```
   <%= @host.interfaces -%>
   ```

   Example rendering:

   ```
   <Nic::Base::ActiveRecord_Associations_CollectionProxy:0x00007f734036fbe0>
   ```

2. In the Create Template window, click the **Help** tab and search for the **ActiveRecord_Associations_CollectionProxy** and **Nic::Base** classes.

3. For **ActiveRecord_Associations_CollectionProxy**, in the **Allowed methods or members** column, you can view the following methods to parse the array:

   ```
   [] each find_in_batches first map size to_a
   ```

4. For **Nic::Base**, in the **Allowed methods or members** column, you can view the following method to parse the array:

   ```
   alias? attached_devices attached_devices_identifiers attached_to bond_options
   children_mac_addresses domain fqdn identifier inheriting_mac ip ip6 link mac managed?
   mode mtu nic_delay physical? primary provision shortname subnet subnet6 tag virtual?
   vlanid
   ```

5. To iterate through an interface array, add the relevant methods to the ERB syntax:

   Example input:

   ```
   <% load_hosts().each_record do |host| -%>
   <%    host.interfaces.each do |iface| -%>
     iface.alias?: <%= iface.alias? %>
     iface.attached_to: <%= iface.attached_to %>
     iface.bond_options: <%= iface.bond_options %>
     iface.children_mac_addresses: <%= iface.children_mac_addresses %>
     iface.domain: <%= iface.domain %>
     iface.fqdn: <%= iface.fqdn %>
     iface.identifier: <%= iface.identifier %>
     iface.inheriting_mac: <%= iface.inheriting_mac %>
     iface.ip: <%= iface.ip %>
     iface.ip6: <%= iface.ip6 %>
     iface.link: <%= iface.link %>
     iface.mac: <%= iface.mac %>
     iface.managed?: <%= iface.managed? %>
     iface.mode: <%= iface.mode %>
     iface.mtu: <%= iface.mtu %>
     iface.physical?: <%= iface.physical? %>
     iface.primary: <%= iface.primary %>
     iface.provision: <%= iface.provision %>
   ```

```
  iface.shortname: <%= iface.shortname %>
  iface.subnet: <%= iface.subnet %>
  iface.subnet6: <%= iface.subnet6 %>
  iface.tag: <%= iface.tag %>
  iface.virtual?: <%= iface.virtual? %>
  iface.vlanid: <%= iface.vlanid %>
<%- end -%>
```

Example rendering:

```
host1.example.com
  iface.alias?: false
  iface.attached_to:
  iface.bond_options:
  iface.children_mac_addresses: []
  iface.domain:
  iface.fqdn: host1.example.com
  iface.identifier: ens192
  iface.inheriting_mac: 00:50:56:8d:4c:cf
  iface.ip: 10.10.181.13
  iface.ip6:
  iface.link: true
  iface.mac: 00:50:56:8d:4c:cf
  iface.managed?: true
  iface.mode: balance-rr
  iface.mtu:
  iface.physical?: true
  iface.primary: true
  iface.provision: true
  iface.shortname: host1.example.com
  iface.subnet:
  iface.subnet6:
  iface.tag:
  iface.virtual?: false
  iface.vlanid:
```

A.9. EXAMPLE TEMPLATE SNIPPETS

Checking if a Host Has Puppet and Puppetlabs Enabled

The following example checks if the host has the Puppet and Puppetlabs repositories 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')
%>
```

Capturing Major and Minor Versions of a Host's Operating System

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 -%>
```

Importing Snippets to a Template

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

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

Conditionally Importing a Kickstart Snippet

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 -%>
```

Parsing Values from Host Custom Facts

You can use the **host.facts** variable to parse values from a host's facts and custom facts.

In this example **luks_stat** is a custom fact that you can parse in the same manner as **dmi::system::serial_number**, which is a host fact:

```
'Serial': host.facts['dmi::system::serial_number'],
'Encrypted': host.facts['luks_stat'],
```

In this example, you can customize the Applicable Errata report template to parse for custom information about the kernel version of each host:

```
<%-    report_row(
        'Host': host.name,
        'Operating System': host.operatingsystem,
        'Kernel': host.facts['uname::release'],
        'Environment': host.lifecycle_environment,
        'Erratum': erratum.errata_id,
        'Type': erratum.errata_type,
        'Published': erratum.issued,
        'Applicable since': erratum.created_at,
        'Severity': erratum.severity,
        'Packages': erratum.package_names,
        'CVEs': erratum.cves,
        'Reboot suggested': erratum.reboot_suggested,
      ) -%>
```

APPENDIX B. JOB TEMPLATE EXAMPLES AND EXTENSIONS

Use this section as a reference to help modify, customize, and extend your job templates to suit your requirements.

B.1. CUSTOMIZING JOB TEMPLATES

When creating a job template, you can include an existing template in the template editor field. 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** list. You can import all parameters or specify a comma separated list.

B.2. 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.

Job template category	Description
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.

B.3. EXAMPLE RESTORECON TEMPLATE

This example shows how to create a template called **Run Command - restorecon** that restores 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. Enter **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 is replaced by a user-defined directory during job invocation.
3. On the **Job** tab, set **Job category** to **Commands**.
4. Click **Add Input** to allow job customization. Enter **directory** to the **Name** field. The input name must match the value specified in the template editor.
5. Click **Required** so that the command cannot be executed without the user specified parameter.
6. Select **User input** from the **Input type** list. Enter a description to be shown during job invocation, for example **Target directory for restorecon**.
7. Click **Submit**.

See Section B.5, “Executing a restorecon Template on Multiple Hosts” for information on how to execute a job based on this template.

B.4. RENDERING A RESTORECON TEMPLATE

This example shows how to create a template derived from the **Run command - restorecon** template created in Section B.3, “Example 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 10.14, “Setting up Job Templates”, and specify the following string in the template editor:

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

B.5. EXECUTING A RESTORECON TEMPLATE ON MULTIPLE HOSTS

This example shows how to run a job based on the template created in Section B.3, "Example restorecon Template" on multiple hosts. The job restores the SELinux context in all files under the **/home/** directory.

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

B.6. 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 10.14, "Setting up Job Templates", and specify the following string in the template editor:

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