Red Hat Satellite 6.8

Configuring Satellite to use Ansible

Configure Ansible in Satellite to help automate repetitive tasks

Last Updated: 2022-08-05

Red Hat Satellite 6.8 Configuring Satellite to use Ansible

Configure Ansible in Satellite to help automate repetitive tasks

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

Legal Notice

Abstract

This guide describes how to set up and configure Red Hat Satellite to use Ansible to perform remote execution and automate repetitive tasks.

Table of Contents

CHAPTER 1. GETTING STARTED WITH ANSIBLE IN SATELLITE

Use this guide to configure Satellite to use Ansible, and then information about how to use Ansible for remote execution.

1.1. CONFIGURING YOUR DEPLOYMENT TO RUN ANSIBLE ROLES

In Satellite, you can import Ansible roles to help with automation of routine tasks. Ansible is enabled by default only on Satellite.

Complete this procedure to configure your Satellite deployment to run Ansible roles.

Procedure

1. Add the roles to the **/etc/ansible/roles** directory on the Satellite and all Capsules from where you want to use the roles. If you want to use custom or third party Ansible roles, ensure to configure an external version control system to synchronize roles between Satellite and Capsules.

2. On all Capsules that you want to use to run Ansible roles on hosts, enable the Ansible plug-in:

```
# satellite-installer --scenario capsule \
--enable-foreman-proxy-plugin-ansible
```

3. Distribute SSH keys to enable Capsules to connect to hosts using SSH. For more information, see Distributing SSH Keys for Remote Execution in the *Managing Hosts* guide. Satellite runs Ansible roles the same way it runs remote execution jobs.

4. Import the Ansible roles into Satellite.

5. Proceed to Using Ansible Roles to Automate Repetitive Tasks on Satellite Hosts in *Configuring Satellite To Use Ansible*.

1.2. IMPORTING ANSIBLE ROLES

You can import Ansible roles from the **/etc/ansible/roles** directory on Satellite or on a Capsule that has Ansible enabled. Ensure that the roles that you import are located in the **/etc/ansible/roles** directory on all Capsules from where you want to use the roles.

To import Ansible roles, complete the following steps:

1. In the Satellite web UI, navigate to **Configure** > **Roles** and click the Capsule that contains the roles that you want to import.

2. From the list of Ansible roles, select the check box of the roles you want to import, and then click **Update**.

1.3. IMPORTING ANSIBLE VARIABLES

Ansible roles use variables to help refine the configuration of systems that have specific requirements. For example, you might need to identify the IP address of a system and use it as a configuration value on another system.

If you want to use Ansible variables in your Ansible playbooks, you must import the Ansible variables from Capsule.

Procedure

To import Ansible variables, complete the following steps:

1. In the Satellite web UI, navigate to **Configure** > **Variables**.
2. In the upper right of the window, select the Capsule that contains the Ansible variables that you want to import.
3. Select the Ansible variables that you want to import, and click **Update**.

1.4. CREATING ANSIBLE VARIABLES

Ansible roles use variables to help refine the configuration of systems that have specific requirements.

Usually, you can import Ansible variables for the Ansible roles that you are using. If you require further refinement of your system configuration, you can also create Ansible variables in Satellite.

Procedure

1. In the Satellite web UI, navigate to **Configure** > **Variables**.
2. In the upper right of the window, click **New Ansible Variable**.
3. In the **Key** field, enter a name for the variable. Ensure that the name references the Ansible role name.
4. In the **Description** field, add a description for the variable.
5. From the **Ansible role** list, select the Ansible role to associate with the variable.
6. Optional: To override the Ansible variable with Satellite, see the Section 1.5, “Overriding Ansible Variables in Satellite”.
7. To save the Ansible variable, click **Submit**.

1.5. OVERRIDING ANSIBLE VARIABLES IN SATELLITE

If you run Ansible roles in Satellite, you can use Satellite to override Ansible variables for those roles.

Precedence in Overriding Variables

If you use an Ansible role to run a task as a user that is not the **Effective User**, there is a strict order of precedence for overriding Ansible variables. To ensure that the variable that you override follows the correct order of precedence, see Variable precedence: Where should I put a variable? in the *Ansible User Guide*.

Prerequisite

You must have Ansible variables in Satellite.

To import Ansible variables, see Section 1.3, “Importing Ansible Variables”

To create Ansible variables, see Section 1.4, “Creating Ansible Variables”

The following procedure makes reference to hosts and host groups. For more information about hosts and host groups, see the Managing Hosts guide.

Procedure

To override an Ansible variable, complete the following steps:

1. In the Satellite web UI, navigate to **Configure** > **Variables**.
2. Select the Ansible variable that you want to override and manage with Satellite.
3. Navigate to the **Default Behavior** area, and select the **Override** check box.
4. From the **Parameter Type** select the value type for validation. For example, a **string** or **boolean** variable.
5. In the **Default Value** field, enter the default value that you want to use if there is no match for the variable.
6. Optional: If you do not want to display the Ansible variable in plain text, select the **Hidden Values** check box to display the content of the variable as asterisks in the Satellite web UI.
7. To save the override settings, click **Submit**.

To use the Ansible variable, add the variable as a parameter to your host or host group, or add the variable as a global parameter.

For a Host Group:

1. In the Satellite web UI, navigate to **Configure** > **Host Groups**, and select the host group that you want to use.
2. Click the **Parameters** tab, and in the **Host Group Parameters** area, click **Add Parameter**.
3. In the **Name** field, add the Ansible variable name.
4. From the **Type** list, select the type of the variable for validation.
5. In the **Value** field, enter the value for the variable.

For a Host:

1. In the Satellite web UI, navigate to **Hosts** > **All Hosts**, and on the host that you want to use, click the **Edit** button.
2. Click the **Parameters** tab, and in the **Host Parameters** area, click **Add Parameter**.
3. In the **Name** field, add the Ansible variable name.
4. From the **Type** list, select the type of the variable for validation.
5. In the **Value** field, enter the value for the variable.

To add as a Global Parameter:

1. In the Satellite web UI, navigate to **Configure** > **Global Parameters**, and click **Create Parameter**.

2. In the **Name** field, add the Ansible variable name.
3. From the **Type** list, select the type of the variable for validation.
4. In the **Value** field, enter the value for the variable.
5. Optional: If you do not want to display the Ansible variable in plain text, select the **Hidden Values** check box to display the content of the variable as asterisks in the Satellite web UI.

1.6. ADDING RED HAT ENTERPRISE LINUX SYSTEM ROLES

Red Hat Enterprise Linux System Roles is a configuration interface to remotely manage Red Hat Enterprise Linux servers. You can use Red Hat Enterprise Linux System Roles to add Ansible roles in Satellite. Using Ansible Roles in Satellite can make configuration faster and easier.

Support levels for some of the Red Hat Enterprise Linux System Roles might be in Technology Preview. For up-to-date information about support levels and general information about Red Hat Enterprise Linux System Roles, see Red Hat Enterprise Linux System Roles .

Before subscribing to the Extras channels, see the Red Hat Enterprise Linux Extras Product Life Cycle article.

Procedure

1. Ensure that the **rhel-7-server-extras-rpms** repository is enabled.

```
# subscription-manager repos --enable=rhel-7-server-extras-rpms
```

2. Install the **rhel-system-roles** package.

```
# satellite-maintain packages install rhel-system-roles
```

   The **rhel-system-roles** package downloads to **/usr/share/ansible/roles/**. You can view and make any modifications that you want to the files before you import.

3. In the Satellite web UI, navigate to **Configure** > **Roles** and click the Capsule that contains the roles that you want to import.
4. From the list of Ansible roles, select the check box of the roles you want to import, and then click **Update**.

You can now assign Ansible roles to hosts or host groups. For more information, see Assigning Ansible Roles to an Existing Host in *Configuring Satellite to Use Ansible* .

You can also add the modules contained in these roles to your Ansible playbooks by adding them to Ansible Job Templates. You must include the **hosts:all** line in the job template. For more information, see Red Hat Enterprise Linux (RHEL) System Roles .

CHAPTER 2. USING ANSIBLE ROLES TO AUTOMATE REPETITIVE TASKS ON SATELLITE HOSTS

2.1. ASSIGNING ANSIBLE ROLES TO AN EXISTING HOST

You can use Ansible roles for remote management of Red Hat Enterprise Linux versions 8, 7, and 6.9 or later.

Prerequisites

- Ensure that you have configured and imported Ansible roles.

Procedure

1. In the Satellite web UI, navigate to **Hosts** > **All Hosts**.
2. On the host you want to assign an Ansible role to, click **Edit**.
3. Select the **Ansible Roles** tab, and in the **All items** list, search for the roles that you want to add.
4. Select the roles that you want to add, and click the arrow icon to move the roles to the **Selected items** list.
5. Click **Submit**.

After you assign Ansible roles to hosts, you can use Ansible for remote execution. For more information, see Section 3.8, “Distributing SSH Keys for Remote Execution” .

Overiding Parameter Variables

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 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**.

2.2. RUNNING ANSIBLE ROLES ON A HOST

You can run Ansible roles on a host through the Satellite web UI.

Prerequisites

- You must configure your deployment to run Ansible roles. For more information, see Configuring your Deployment to Run Ansible Roles in *Configuring Satellite to use Ansible* .
- You must have assigned the Ansible roles to the host.

Procedure

1. In the Satellite web UI, navigate to **Hosts** > **All Hosts**.
2. Select the check box of the host that contains the Ansible role you want to run.
3. From the **Select Action** list, select **Play Ansible roles**.

You can view the status of your Ansible job on the **Run Ansible roles** page. To rerun a job, click the **Rerun** button.

2.3. ASSIGNING AN ANSIBLE ROLE TO A HOST GROUP

You can use Ansible roles for remote management of Red Hat Enterprise Linux versions 8, 7, and 6.9 or later.

Prerequisites

- You must configure your deployment to run Ansible roles. For more information, see Configuring your Deployment to Run Ansible Roles in *Configuring Satellite to use Ansible* .

Procedure

1. In the Satellite web UI, navigate to **Configure** > **Host Groups**.
2. From the list of host groups, click the host group name that you want to add an Ansible Role to.
3. Select the **Ansible Roles** tab, and in the **All items** list, search for the roles that you want to add.
4. Select the roles that you want to add, and click the arrow icon to move the roles to the **Selected items** list.
5. Click **Submit**.

2.4. RUNNING ANSIBLE ROLES ON A HOST GROUP

You can run Ansible roles on a host group through the Satellite web UI.

Prerequisites

- You must configure your deployment to run Ansible roles. For more information, see Configuring your Deployment to Run Ansible Roles in *Configuring Satellite to use Ansible* .
- You must have assigned the Ansible roles to the host group.
- You must have at least one host in your host group.

Procedure

1. In the Satellite web UI, navigate to **Configure** > **Host Groups**.
2. From the list in the **Actions** column for the host group, select **Run all Ansible roles**.

You can view the status of your Ansible job on the **Run Ansible roles** page. To rerun a job, click the **Rerun** button.

CHAPTER 3. 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.

3.1. 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 3.14, “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 3.15, “Executing a Remote Job” .

3.2. 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 Ansible 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.

3.3. 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.

3.4. 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 A.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"
```

3.5. 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
```

3.6. 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
```

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

Ansible puts its own files it requires into the **$HOME/.ansible/tmp** directory, where **$HOME** is the home directory of the remote user. You have the option to set a different directory if required.

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 **ansible_working_dir** setting in the **/etc/foreman-proxy/settings.d/ansible.yml** file to point to the required directory, for example:

```
:ansible_working_dir: /remote_working_dir
```

3.8. 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 3.9, “Distributing SSH Keys for Remote Execution Manually” .
2. Section 3.10, “Using the Satellite API to Obtain SSH Keys for Remote Execution” .
3. Section 3.11, “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.

3.9. 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
```

3.10. 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
```

3.11. 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' %>
```

3.12. 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
```

3.13. 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.

3.14. 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 3.15, “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**.

3.15. 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 3.4, "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**.

3.16. 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 4. INTEGRATING RED HAT SATELLITE AND ANSIBLE TOWER

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

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

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

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

Prerequisites

- If your Satellite deployment is large, for example, managing tens of thousands of hosts, using a non-admin user can negatively impact performance because of time penalties that accrue during authorization checks. For large deployments, consider using an admin user.
- For non-admin users, you must assign the **Ansible Tower Inventory Reader** role to your Satellite Server user. For more information about managing users, roles, and permission filters, see Creating and Managing Roles in *Administering Red Hat Satellite*.
- You must host your Satellite Server and Ansible Tower on the same network or subnet.

Procedure

To add Satellite Server to Ansible Tower as a Dynamic Inventory Item, complete the following procedure:

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

   Table 4.1. Satellite Credentials

Credential Type:	Red Hat Satellite 6
Satellite 6 URL:	https://*satellite.example.com*
Username:	The username of the Satellite user with the integration role.
Password:	The password of the Satellite user.

2. Add an Ansible Tower user to the new credential. For more information about adding a user to a credential, see Getting Started with Credentials in the *Ansible Tower User Guide*.
3. Add a new inventory. For more information, see Add a new inventory in the *Ansible Tower User Guide*.

4. In the new inventory, add Satellite Server as the inventory source, specifying the following inventory source options. For more information, see Add Source in the *Ansible Tower User Guide*.

Table 4.2. Inventory Source Options

Source	Red Hat Satellite 6
Credential	The credential you create for Satellite Server.
Overwrite	Select
Overwrite Variables	Select
Update on Launch	Select
Cache Timeout	*90*

5. Ensure that you synchronize the source that you add.

4.2. CONFIGURING PROVISIONING CALLBACK FOR A HOST

When you create hosts in Satellite, you can use Ansible Tower to run playbooks to configure your newly created hosts. This is called *provisioning callback* in Ansible Tower.

The provisioning callback function triggers a playbook run from Ansible Tower as part of the provisioning process. The playbook configures the host after Kickstart deployment.

For more information about provisioning callbacks, see Provisioning Callbacks in the *Ansible Tower User Guide*.

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

1. **ansible_provisioning_callback**
2. **ansible_tower_callback_script**
3. **ansible_tower_callback_service**

You can add parameters to hosts or host groups to provide the credentials that these snippets can use to run Ansible playbooks on your newly created hosts.

Prerequisites

Before you can configure provisioning callbacks, you must add Satellite as a dynamic inventory in Ansible Tower. For more information, see Integrating Satellite and Ansible Tower .

In the Ansible Tower web UI, you must complete the following tasks:

1. Create a machine credential for your new host. Ensure that you enter the same password in the credential that you plan to assign to the host that you create in Satellite. For more information, see Add a New Credential in the *Ansible Tower User Guide*.

2. Create a project. For more information, see Projects in the *Ansible Tower User Guide*.
3. Add a job template to your project. For more information, see Job Templates in the *Ansible Tower User Guide*.
4. In your job template, you must enable provisioning callbacks, generate the host configuration key, and note the *template_ID* of your job template. For more information about job templates, see Job Templates in the *Ansible Tower User Guide*.

Procedure

To configure provisioning callback for a new host in Satellite, complete the following steps:

1. In the Red Hat Satellite web UI, navigate to **Configure** > **Host Group**.
2. Create a host group or edit an existing host group.
3. In the Host Group window, click the **Parameters** tab.
4. Click **Add Parameter**.
5. Enter the following information for each new parameter:

Table 4.3. Host Parameters

Name	Value	Description
ansible_tower_provisioning	true	Enables Provisioning Callback.
ansible_tower_fqdn	*tower.example.com*	The fully qualified domain name (FQDN) of your Ansible Tower. Do not add **https** because this is appended by Ansible Tower.
ansible_job_template_id	*template_ID*	The ID of your provisioning template that you can find in the URL of the template: **/templates/job_template/*5***.
ansible_host_config_key	*config_KEY*	The host configuration key that your job template generates in Ansible Tower.

6. Click **Submit**.
7. Create a host using the host group.
8. On the new host, enter the following command to start the **ansible-callback** service:

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

9. On the new host, enter the following command to output the status of the **ansible-callback** service:

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

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

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

Manual Provisioning Callback

You can use the provisioning callback URL and the host configuration key from a host to call Ansible Tower. For example:

```
# curl -k -s --data curl --insecure --data host_config_key=my_config_key \
 https://tower.example.com/api/v2/job_templates/8/callback/
```

Ensure that you use **https** when you enter the provisioning callback URL.

This triggers the playbook run specified in the template against the host.

APPENDIX A. JOB TEMPLATE EXAMPLES AND EXTENSIONS

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

A.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.

A.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.

A.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 A.5, "Executing a restorecon Template on Multiple Hosts" for information on how to execute a job based on this template.

A.4. RENDERING A RESTORECON TEMPLATE

This example shows how to create a template derived from the **Run command - restorecon** template created in Section A.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 3.14, "Setting up Job Templates", and specify the following string in the template editor:

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

A.5. EXECUTING A RESTORECON TEMPLATE ON MULTIPLE HOSTS

This example shows how to run a job based on the template created in Section A.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.

A.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 3.14, "Setting up Job Templates", and specify the following string in the template editor:

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