Managing Configurations Using Ansible Integration in Red Hat Satellite

Configure Ansible integration in Satellite and use Ansible roles and playbooks to configure your hosts
Configure Ansible integration in Satellite and use Ansible roles and playbooks to configure your hosts

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

This guide describes how to integrate Red Hat Satellite with Ansible and how to perform remote execution and automate repetitive tasks using Ansible roles and playbooks in Satellite.
# Table of Contents

**PROVIDING FEEDBACK ON RED HAT DOCUMENTATION** .............................................. 4

**CHAPTER 1. GETTING STARTED WITH ANSIBLE IN SATELLITE** .................................. 5
  1.1. SUPPORTED ANSIBLE VERSIONS ........................................................................ 5
  1.2. CONFIGURING YOUR SATELLITE TO RUN ANSIBLE ROLES ......................... 5
  1.3. ENABLING ANSIBLE INTEGRATION WITH SATELLITE .................................... 6
  1.4. IMPORTING ANSIBLE ROLES AND VARIABLES ............................................ 6
  1.5. OVERRIDING ANSIBLE VARIABLES IN SATELLITE ....................................... 6
  1.6. ADDING RED HAT ENTERPRISE LINUX SYSTEM ROLES ............................... 8
  1.7. SYNCHRONIZING ANSIBLE COLLECTIONS .................................................. 9
  1.8. CUSTOMIZING ANSIBLE CONFIGURATION .................................................. 10
  1.9. USING ANSIBLE VAULT WITH SATELLITE .................................................. 11

**CHAPTER 2. USING ANSIBLE ROLES TO AUTOMATE REPETITIVE TASKS ON CLIENTS** .......................... 12
  2.1. ASSIGNING ANSIBLE ROLES TO AN EXISTING HOST ................................. 12
  2.2. REMOVING ANSIBLE ROLES FROM A HOST ............................................. 12
  2.3. CHANGING THE ORDER OF ANSIBLE ROLES ........................................... 12
  2.4. RUNNING ANSIBLE ROLES ON A HOST .................................................... 13
  2.5. ASSIGNING ANSIBLE ROLES TO A HOST GROUP ....................................... 13
  2.6. RUNNING ANSIBLE ROLES ON A HOST GROUP ....................................... 14
  2.7. RUNNING ANSIBLE ROLES IN CHECK MODE ............................................ 14

**CHAPTER 3. CONFIGURING AND SETTING UP REMOTE JOBS** ................................... 15
  3.1. ABOUT RUNNING JOBS ON HOSTS ............................................................ 15
  3.2. REMOTE EXECUTION WORKFLOW ................................................................ 15
  3.3. PERMISSIONS FOR REMOTE EXECUTION .................................................. 16
  3.4. TRANSPORT MODES FOR REMOTE EXECUTION ....................................... 17
  3.5. CONFIGURING A HOST TO USE THE PULL CLIENT ................................... 17
  3.6. CREATING A JOB TEMPLATE ....................................................................... 19
  3.7. IMPORTING AN ANSIBLE PLAYBOOK BY NAME ....................................... 20
  3.8. IMPORTING ALL AVAILABLE ANSIBLE PLAYBOOKS .................................. 20
  3.9. CONFIGURING THE FALLBACK TO ANY CAPSULE REMOTE EXECUTION SETTING IN SATELLITE ........................................................................................................... 21
  3.10. CONFIGURING THE GLOBAL CAPSULE REMOTE EXECUTION SETTING IN SATELLITE ......................................................................................................................... 22
  3.11. CONFIGURING SATELLITE TO USE AN ALTERNATIVE DIRECTORY TO EXECUTE REMOTE JOBS ON HOSTS ................................................................. 22
  3.12. ALTERING THE PRIVILEGE ELEVATION METHOD ..................................... 22
  3.13. DISTRIBUTING SSH KEYS FOR REMOTE EXECUTION ............................... 23
  3.14. DISTRIBUTING SSH KEYS FOR REMOTE EXECUTION MANUALLY ............. 23
  3.15. USING THE SATELLITE API TO OBTAIN SSH KEYS FOR REMOTE EXECUTION ................................................................................................................................. 24
  3.16. CONFIGURING A KICKSTART TEMPLATE TO DISTRIBUTE SSH KEYS DURING PROVISIONING .......................................................................................... 24
  3.17. CONFIGURING A KEYTAB FOR KERBEROS TICKET GRANTING TICKETS ........ 25
  3.18. CONFIGURING KERBEROS AUTHENTICATION FOR REMOTE EXECUTION ...... 25
  3.19. SETTING UP JOB TEMPLATES ..................................................................... 26
  3.20. EXECUTING A REMOTE JOB ...................................................................... 27
  3.21. ADVANCED SETTINGS IN THE JOB WIZARD ............................................ 28
  3.22. USING EXTENDED CRON LINES .................................................................. 29
  3.23. SCHEDULING A RECURRING ANSIBLE JOB FOR A HOST ......................... 30
  3.24. SCHEDULING A RECURRING ANSIBLE JOB FOR A HOST GROUP ............. 30
  3.25. MONITORING JOBS .................................................................................... 31
  3.26. SETTING THE JOB RATE LIMIT ON CAPSULE ......................................... 31

**CHAPTER 4. INTEGRATING RED HAT SATELLITE AND ANSIBLE AUTOMATION CONTROLLER** .......................... 33
4.1. ADDING SATELLITE SERVER TO ANSIBLE AUTOMATION CONTROLLER AS A DYNAMIC INVENTORY ITEM
4.2. CONFIGURING PROVISIONING CALLBACK FOR A HOST

CHAPTER 5. JOB TEMPLATE EXAMPLES AND EXTENSIONS .............................................. 37
5.1. CUSTOMIZING JOB TEMPLATES ................................................................. 37
5.2. DEFAULT JOB TEMPLATE CATEGORIES ...................................................... 37
5.3. EXAMPLE RESTORECON TEMPLATE ....................................................... 37
5.4. RENDERING A RESTORECON TEMPLATE .................................................. 38
5.5. EXECUTING A RESTORECON TEMPLATE ON MULTIPLE HOSTS ............... 38
5.6. INCLUDING POWER ACTIONS IN TEMPLATES .......................................... 39
PROVIDING FEEDBACK ON RED HAT DOCUMENTATION

We appreciate your input on our documentation. Please let us know how we could make it better.

You can submit feedback by filing a ticket in Bugzilla:

1. Navigate to the Bugzilla website.

2. In the Component field, use Documentation.

3. In the Description field, enter your suggestion for improvement. Include a link to the relevant parts of the documentation.

4. Click Submit Bug.
CHAPTER 1. GETTING STARTED WITH ANSIBLE IN SATELLITE

Use this guide to configure Satellite to use Ansible for remote execution.

1.1. SUPPORTED ANSIBLE VERSIONS

Satellite uses Ansible as provided by the base operating system of Satellite Server or any Capsules for remote execution. Therefore, the supported version of Ansible depends on your base OS configuration.

1.2. CONFIGURING YOUR SATELLITE TO RUN ANSIBLE ROLES

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

Ansible paths

Satellite imports and runs Ansible roles from the following paths:

- `/etc/ansible/roles`
- `/usr/share/ansible/roles`
- `/etc/ansible/collections`
- `/usr/share/ansible/collections`

Roles and collections from installed packages are placed under `/usr/share/ansible`. If you want to add custom roles or collections, place them under `/etc/ansible`.

Note that Red Hat provides support only for Ansible roles and collections obtained from Red Hat.

The paths are configured by Satellite. For more information, see Section 1.8, "Customizing Ansible Configuration".

Procedure

1. Add the roles to a directory in an Ansible path on Satellite Server and all Capsule Servers 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 Server and Capsule Servers.

2. On all Capsule Servers 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 Managing Hosts. Satellite runs Ansible roles the same way it runs remote execution jobs.

4. Import the Ansible roles into Satellite.

5. Proceed to Chapter 2, Using Ansible Roles to Automate Repetitive Tasks on Clients.
1.3. ENABLING ANSIBLE INTEGRATION WITH SATELLITE

Perform the following procedure to enable the Ansible plug-in on your Satellite Server.

Procedure

- Enable the Ansible plug-in on your Satellite Server:

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

1.4. IMPORTING ANSIBLE ROLES AND VARIABLES

You can import Ansible roles and variables from the Ansible paths on Satellite Server or Capsule that has Ansible enabled.

Note that some roles take longer to import than others.

Prerequisite

- Ensure that the roles and variables that you import are located in the Ansible paths on all Capsules from where you want to use the roles.

Procedure

1. In the Satellite web UI, navigate to Configure > Roles.
2. Click Import to select the Capsule from which you want to import.
3. Select the roles that you want to import.
4. 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.

The following procedure refers to hosts and host groups. For more information, see Managing Hosts.

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 the variable that you override follows the correct order of precedence, see Variable precedence: Where should I put a variable?

Prerequisite

- You must have Ansible variables in Satellite. For more information, see Section 1.4, “Importing Ansible Roles and Variables”

Procedure

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. In the Default Behavior area, select the Override checkbox.

4. In the Parameter Type field, select the value type for validation such as string or boolean. The types array and hash have further options for handling upon a variable match. For more information, see the Prioritize Attribute Order area below.

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 value of the variable as plain text in the Satellite web UI, select the Hidden Value checkbox to display the content of the variable as asterisks. This is useful for sensitive values such as passwords or secret tokens.

7. Optional: Expand the Optional Input Validator area and specify conditions that will be used to validate concrete values of the variable:
   - Select Required if you want to enforce users to fill in this variable.
   - In the Validator Type field, select how the value will be validated:
     - list – The value will be validated against an enumeration of allowed values.
     - regex – The value will be validated against a regular expression pattern.

8. Optional: In the Prioritize Attribute Order area, specify the order of priority to match an override with a host by host attributes. Order at the top takes higher precedence. The first match wins.
   You can combine multiple attributes into a single matcher key using a comma as the AND operation. For example, the matcher key of hostgroup, environment would expect matchers such as hostgroup = “web servers” AND environment = production.

   If you use the parameter type array or hash, you can further set:
   - Merge Overrides – Merges members of the arrays/hashes instead of replacing the whole array or hash. If the hashes contain the same key, the value is overwritten by the value of the host.
   - Merge Default – Adds the default value to the array or hash.
   - Avoid Duplicates – Ensures that the values in the array or hash are unique.

9. Optional: Expand the Specify Matchers area and specify criteria for selecting hosts on which the variable overrides.

10. 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.

**Adding the Variable to a Host**

1. In the Satellite web UI, navigate to Hosts > All Hosts and select the host that you want to use.

2. Click the Ansible tab, and in the Variables area, click the pencil icon to edit the value of the variable.
3. Click the tick icon to accept the value of the changed variable or the cross icon to cancel the change.

Adding the Variable to 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.

Adding the Variable 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 checkbox 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 following repository is enabled:

   - On Red Hat Enterprise Linux 8, ensure that the Appstream repository is enabled:
     
     ```bash
     # subscription-manager repos --enable=rhel-8-for-x86_64-appstream-rpms
     ```
     You must enable an Appstream repository that is designated for your architecture. For more information, see RHEL 8 repositories.

   - On Red Hat Enterprise Linux 7, ensure that the Extras repository is enabled:
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 checkbox 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 Section 2.1, “Assigning Ansible Roles to an Existing Host”.

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.

### 1.7. SYNCHRONIZING ANSIBLE COLLECTIONS

On Satellite, you can synchronize your Ansible Collections from Private Automation Hub, `console.redhat.com`, and other Satellite instances. Ansible Collections will appear on Satellite as a new repository type in the Satellite web UI menu under Content after the sync.

**Procedure**

1. In the Satellite web UI, navigate to Content > Products.

2. Select the required product name.

3. In the Products window, select the name of a product that you want to create a repository for.

4. Click the Repositories tab, and then click New Repository.

5. In the Name field, enter a name for the repository. The Label field is populated automatically based on the name.

6. From the Type list, select ansible collection.

7. In the Upstream URL field, enter the URL for the upstream collections repository. The URL can be any Ansible Galaxy endpoint. For example, `https://console.redhat.com/api/automation-hub/`.

8. Optional: In the Requirements.yml field, you can specify the list of collections you want to sync from the endpoint, as well as their versions.

   If you do not specify the list of collections, everything from the endpoint will be synced.

   ```
   ---
   collections:
   - name: my_namespace.my_collection
     version: 1.2.3
   ```
For more information, see Install multiple collections with a requirements file in Galaxy User Guide.

   a. To sync Satellite from Private Automation Hub, enter your token in the Auth Token field.
      For more information, see Connect Private Automation Hub in Connect to Hub.
   b. To sync Satellite from console.redhat.com, enter your token in the Auth Token field and enter your SSO URL in the Auth URL field.
      For more information, see Getting started with automation hub.
   c. To sync Satellite from Satellite, leave both authentication fields blank.

10. Click Save.

11. Navigate to the Ansible Collections repository.

12. From the Select Action menu, select Sync Now.

1.8. CUSTOMIZING ANSIBLE CONFIGURATION

Satellite manages essential Ansible configuration that is required for Ansible integration with Satellite. However, you can customize other Ansible configuration options as usual.

Satellite stores the essential Ansible configuration as environment variables in /etc/foreman-proxy/ansible.env. This file is managed by satellite-installer.

Ansible reads configuration from a configuration file and the environment provided by Capsule. If you need to customize Ansible configuration, you can do so in the system-wide /etc/ansible/ansible.cfg file or in the /usr/share/foreman-proxy/.ansible.cfg file in the home directory of the foreman-proxy user. Note that if you use /usr/share/foreman-proxy/.ansible.cfg, Ansible spawned by Satellite ignores configuration in /etc/ansible/ansible.cfg.

Note that environment variables take precedence over values in ansible.cfg, which ensures that the essential configuration required by Satellite is retained.

The following table lists the essential Ansible configuration options managed by Satellite.

### Table 1.1. Essential Ansible Configuration

<table>
<thead>
<tr>
<th>Environment Variable</th>
<th>Config Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANSIBLE_CALLBACKS_ENABLED</td>
<td>callbacks_enabled</td>
<td>Enables callback to Satellite; equivalent to callback_whitelist for cross-version compatibility</td>
</tr>
<tr>
<td>ANSIBLE_CALLBACK_WHITELIST</td>
<td>callback_whitelist</td>
<td>Enables callback to Satellite; equivalent to callbacks_enabled for cross-version compatibility</td>
</tr>
<tr>
<td>ANSIBLE_COLLECTIONS_PATHS</td>
<td>collections_paths</td>
<td>List of paths to Ansible collections</td>
</tr>
<tr>
<td>ANSIBLE_HOST_KEY_CHECKING</td>
<td>host_key_checking</td>
<td>Disables checking of host keys during SSH connection</td>
</tr>
</tbody>
</table>
### Environment Variable | Config Key | Description
--- | --- | ---
ANSIBLE_LOCAL_TMP | local_tmp | Temporary directory on Capsule
ANSIBLE_ROLES_PATH | roles_path | List of paths to Ansible roles
ANSIBLE_SSH_ARGS | ssh_args | Arguments passed to SSH connection

**Additional resources**

- Ansible Configuration Settings

### 1.9. USING ANSIBLE VAULT WITH SATELLITE

You can encrypt sensitive Ansible data files using the Ansible Vault tool and configure Ansible to access the encrypted files using a password stored in a file.

**Procedure**

1. If you customized `/etc/ansible/ansible.cfg`, copy your configuration from `/etc/ansible/ansible.cfg` to `/usr/share/foreman-proxy/.ansible.cfg`.

2. Encrypt the sensitive file using the `ansible-vault` command:
   
   ```
   # ansible-vault encrypt /etc/ansible/roles/Role_Name/vars/main.yml
   ```

   Note that `ansible-vault` changes the file permissions to **600**.

3. Change the group and permissions of the encrypted file to ensure that the `foreman-proxy` user can read it:
   
   ```
   # chgrp foreman-proxy /etc/ansible/roles/Role_Name/vars/main.yml
   # chmod 0640 /etc/ansible/roles/Role_Name/vars/main.yml
   ```

4. Create the `/usr/share/foreman-proxy/.ansible_vault_password` file and enter the Vault password into it.

5. Change the user and permissions of the `.ansible_vault_password` file to ensure that only the `foreman-proxy` user can read it:
   
   ```
   # chown foreman-proxy:foreman-proxy /usr/share/foreman-proxy/.ansible_vault_password
   # chmod 0400 /usr/share/foreman-proxy/.ansible_vault_password
   ```

6. Add the path of the Vault password file to the `[defaults]` section in `/usr/share/foreman-proxy/.ansible.cfg`:
   
   ```
   [defaults]
   vault_password_file = /usr/share/foreman-proxy/.ansible_vault_password
   ```

   The path to the Vault password file must be absolute.
CHAPTER 2. USING ANSIBLE ROLES TO AUTOMATE REPETITIVE TASKS ON CLIENTS

2.1. ASSIGNING ANSIBLE ROLES TO AN EXISTING HOST

You can use Ansible roles for remote management of Satellite clients.

Prerequisites

- Ensure that you have configured and imported Ansible roles.

Procedure

1. In the Satellite web UI, navigate to Hosts > All Hosts.
2. Select the host and click Edit.
3. On the Ansible Roles tab, select the role that you want to add from the Available Ansible Roles list.
4. Click the + icon to add the role to the host. You can add more than one role.
5. Click Submit.

After you assign Ansible roles to hosts, you can use Ansible for remote execution. For more information, see Section 3.13, “Distributing SSH Keys for Remote Execution”.

Overriding 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. REMOVING ANSIBLE ROLES FROM A HOST

Use the following procedure to remove Ansible roles from a host.

Procedure

1. In the Satellite web UI, navigate to Hosts > All Hosts.
2. Select the host and click Edit.
3. Select the Ansible Roles tab.
4. In the Assigned Ansible Roles area, click the - icon to remove the role from the host. Repeat to remove more roles.
5. Click Submit.

2.3. CHANGING THE ORDER OF ANSIBLE ROLES

Use the following procedure to change the order of Ansible roles applied to a host.
**Procedure**

1. In the Satellite web UI, navigate to **Hosts > All Hosts**.
2. Select a host.
3. Select the **Ansible Roles** tab.
4. In the **Assigned Ansible Roles** area, you can change the order of the roles by dragging and dropping the roles into the preferred position.
5. Click **Submit** to save the order of the Ansible roles.

**2.4. 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 Section 1.2, “Configuring Your Satellite to Run Ansible Roles”.
- 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 checkbox of the host that contains the Ansible role you want to run.
3. From the **Select Action** list, 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.

**2.5. ASSIGNING ANSIBLE ROLES TO A HOST GROUP**

You can use Ansible roles for remote management of Satellite clients.

**Prerequisite**

- You must configure your deployment to run Ansible roles. For more information, see Section 1.2, “Configuring Your Satellite to Run Ansible Roles”.

**Procedure**

1. In the Satellite web UI, navigate to **Configure > Host Groups**.
2. Click the host group name to which you want to assign an Ansible role.
3. On the **Ansible Roles** tab, select the role that you want to add from the **Available Ansible Roles** list.
4. Click the + icon to add the role to the host group. You can add more than one role.
5. Click Submit.

2.6. 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 Section 1.2, “Configuring Your Satellite to Run Ansible Roles”.
- 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.

2.7. RUNNING ANSIBLE ROLES IN CHECK MODE

You can run Ansible roles in check mode through the Satellite web UI.

Prerequisites

- You must configure your deployment to run Ansible roles. For more information, see Section 1.2, “Configuring Your Satellite to Run Ansible Roles”.
- 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 Hosts > All Hosts.
2. Click Edit for the host you want to enable check mode for.
3. In the Parameters tab, ensure that the host has a parameter named ansible_roles_check_mode with type boolean set to true.
4. Click Submit.
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. For more information, see Section 3.4, “Transport Modes for Remote Execution”.

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

Several job templates for shell scripts and Ansible are included by default. For more information, see Setting up Job Templates in Managing Hosts.

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 Executing a Remote Job in Managing Hosts.

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 checkbox 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.

If you have enabled **Prefer registered through Capsule for remote execution**, Satellite runs the REX job using the Capsule the host is registered to.

By default, **Prefer registered through Capsule for remote execution** is set to **No**. To enable it, in the Satellite web UI, navigate to **Administer > Settings**, and on the **Content** tab, set **Prefer registered through Capsule for remote execution** to **Yes**. This ensures that Satellite performs REX jobs on hosts by the Capsule to which they are registered to.

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 server Capsule
- Puppet CA Capsule
- OpenSCAP Capsule

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. PERMISSIONS FOR REMOTE EXECUTION

You can control which roles 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**: Can access all remote execution features and functionality.
- **Remote Execution User**: Can only run jobs.

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 on a customized role, you 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.

You can run remote execution jobs against Red Hat Satellite and Capsule registered as hosts to Red Hat Satellite with the **execute_jobs_on_infrastructure_hosts** permission. Standard **Manager** and **Site Manager** roles have this permission by default. If you use either the **Manager** or **Site Manager** role, or if you use a custom role with the **execute_jobs_on_infrastructure_hosts** permission, you can execute remote jobs against registered Red Hat Satellite and Capsule hosts.
For more information on working with roles and permissions, see Creating and Managing Roles in 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
```

Use the first line in this example to apply the Reboot template to one selected host. Use the second line to define a pool of hosts with names ending with .staging.example.com. Use the third line to bind the template with a host group.

**NOTE**

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

### 3.4. TRANSPORT MODES FOR REMOTE EXECUTION

You can configure your Satellite to use two different modes of transport for remote job execution.

On Capsules in ssh mode, remote execution uses the SSH service to transport job details. This is the default transport mode. The SSH service must be enabled and active on the target hosts. The remote execution Capsule must have access to the SSH port on the target hosts. Unless you have a different setting, the standard SSH port is 22.

On Capsules in pull-mqtt mode, remote execution uses Message Queueing Telemetry Transport (MQTT) to publish jobs it receives from Satellite Server. The host subscribes to the MQTT broker on Capsule for job notifications using the yggdrasil pull client. After the host receives a notification, it pulls job details from Capsule over HTTPS, runs the job, and reports results back to Capsule.

To use the pull-mqtt mode, you must enable it on Capsule Server and configure the pull client on the target hosts.

**Additional resources**

- To enable pull mode on Capsule Server, see Configuring Remote Execution for Pull Client in Installing Capsule Server.
- To enable pull mode on an existing host, continue with Section 3.5, “Configuring a Host to Use the Pull Client”.
- To migrate a host from Katello Agent, see Migrating from Katello Agent to Remote Execution in Managing Hosts.
- To enable pull mode on a new host, continue with either of the following procedures in Managing Hosts:
  - Creating a Host
  - Registering Hosts

### 3.5. CONFIGURING A HOST TO USE THE PULL CLIENT
For Capsules configured to use pull-mqtt mode, hosts can subscribe to remote jobs using the remote execution pull client. Managed hosts do not require an SSH connection to their Capsule Server.

Prerequisites

- You have registered the host to Satellite.
- The host’s Capsule is configured to use pull-mqtt mode. For more information, see Configuring Remote Execution for Pull Client in Installing Capsule Server.
- The Red Hat Satellite Client 6 repository is enabled and synchronized on Satellite Server, and enabled on the host.
- The host is able to communicate with its Capsule over MQTT using port 1883.
- The host is able to communicate with its Capsule over HTTPS.

**NOTE**

The katello-pull-transport-migrate package was created to help users migrate from Katello Agent to remote execution with the pull client. However, having Katello Agent installed on the host is not a requirement. You can use katello-pull-transport-migrate regardless of whether Katello Agent is installed.

Procedure

1. Install the katello-pull-transport-migrate package on your host:
   - On Red Hat Enterprise Linux 8 and Red Hat Enterprise Linux 9 hosts:
     ```bash
     # dnf install katello-pull-transport-migrate
     ```
   - On Red Hat Enterprise Linux 7 hosts:
     ```bash
     # yum install katello-pull-transport-migrate
     ```

   The package installs foreman_ygg_worker and yggdrasil as dependencies and enables the pull mode on the host. The host’s subscription-manager configuration and consumer certificates are used to configure the yggdrasil client on the host, and the pull mode client worker is started.

2. Optional: To verify that the pull client is running and configured properly, check the status of the yggdrasil service:
   ```bash
   # systemctl status yggdrasil
   ```

3. Optional: After the package is installed, you can remove katello-agent from the host.
3.6. CREATING A JOB TEMPLATE

Use this procedure to create a job template. To use the CLI instead of the Satellite web UI, see the CLI procedure.

Procedure

1. In the Satellite web UI, 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 **Default Job Template Categories** in *Managing Hosts*.
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. Optional: If you use the Ansible provider, click the **Ansible** tab. Select **Enable Ansible Callback** to allow hosts to send facts, which are used to create configuration reports, back to Satellite after a job finishes.

16. Click the **Location** tab and add the locations where you want to use the template.

17. Click the **Organizations** tab and add the organizations where you want to use the template.

18. 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 [Template Writing Reference](#) and [Job Template Examples and Extensions](#) in *Managing Hosts*.

**CLI procedure**

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

  ```bash
  # hammer job-template create \
  --file "Path_to_My_Template_File" \
  --job-category "My_Category_Name" \
  --name "My_Template_Name" \
  --provider-type SSH
  ```

**3.7. IMPORTING AN ANSIBLE PLAYBOOK BY NAME**

You can import Ansible playbooks by name to Satellite from collections installed on Capsule.

**Prerequisite**

- Ansible plug-in in Satellite is enabled

**Procedure**

1. Fetch the available Ansible playbooks using the following API request:

   ```bash
   # curl -X GET 'Content-Type: application/json' 
   https://satellite.example.com/ansible/api/v2/ansible_playbooks/fetch?proxy_id=My_capsule_ID
   ```

2. Select the Ansible playbook you want to import and note its name.

3. Import the Ansible playbook using its name:

   ```bash
   # curl -X PUT 'Content-Type: application/json' -d '{ "playbook_names": ["My_Playbook_Name"] }' 
   https://satellite.example.com/ansible/api/v2/ansible_playbooks/sync?proxy_id=My_capsule_ID
   ```

   You get a notification in the Satellite web UI after the import completes.

**3.8. IMPORTING ALL AVAILABLE ANSIBLE PLAYBOOKS**

You can import all the available Ansible playbooks to Satellite from collections installed on Capsule.
Prerequisite

- Ansible plug-in in Satellite is enabled

Procedure

- Import the Ansible playbooks using the following API request:

  ```
  # curl -X PUT 'Content-Type: application/json'
  https://satellite.example.com/ansible/api/v2/ansible_playbooks/sync?proxy_id=My-capsule-ID
  
  You get a notification in the Satellite web UI after the import completes.
  ```

3.9. CONFIGURING THE FALBACK 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 server Capsule
- Puppet CA Capsule
- OpenSCAP Capsule

Procedure

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

2. Click **Remote Execution**.

3. Configure the **Fallback to Any Capsule** setting.

CLI procedure

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

  ```
  # hammer settings set \ 
  --name=remote_execution_fallback_proxy \ 
  --value=true
  ```
3.10. 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 **Remote Execution**.
3. Configure the **Enable Global Capsule** setting.

**CLI procedure**

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

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

3.11. 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**

1. Create a new directory:

   ```bash
   # mkdir /My_Remote_Working_Directory
   ```

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

   ```bash
   # chcon --reference=/var /My_Remote_Working_Directory
   ```

3. Configure the system:

   ```bash
   # satellite-installer \n   --foreman-proxy-plugin-ansible-working-dir /My_Remote_Working_Directory
   ```

3.12. ALTERING THE PRIVILEGE ELEVATION METHOD
By default, push-based remote execution uses `sudo` to switch from the SSH user to the effective user that executes the script on your host. In some situations, you might require to use another method, such as `su` or `dzdo`. You can globally configure an alternative method in your Satellite settings.

**Prerequisites**

- Your user account has a role assigned that grants the `view_settings` and `edit_settings` permissions.
- If you want to use `dzdo` for Ansible jobs, ensure the `community.general` Ansible collection, which contains the required `dzdo` become plug-in, is installed. For more information, see [Installing collections](#) in Ansible documentation.

**Procedure**

1. Navigate to **Administer** > **Settings**.
2. Select the **Remote Execution** tab.
3. Click the value of the **Effective User Method** setting.
4. Select the new value.
5. Click **Submit**.

### 3.13. DISTRIBUTING SSH KEYS FOR REMOTE EXECUTION

For Capsules in `ssh` mode, remote execution connections are authenticated using SSH. The public SSH key from Capsule must be distributed 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:

2. Section 3.15, “Using the Satellite API to Obtain SSH Keys for Remote Execution”.
3. Section 3.16, “Configuring a Kickstart Template to Distribute SSH Keys During Provisioning”.
4. For new Satellite hosts, you can deploy SSH keys to Satellite hosts during registration using the global registration template. For more information, see Registering a Host to Red Hat Satellite Using the Global Registration Template in Managing Hosts.

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 New User Accounts.

### 3.14. DISTRIBUTING SSH KEYS FOR REMOTE EXECUTION MANUALLY

To distribute SSH keys manually, complete the following steps:
Procedure

- Copy the SSH pub key from your Capsule to your target host:

```bash
# ssh-copy-id -i ~/.ssh/id_rsa_foreman_proxy.pub root@client.example.com
```

Repeat this step for each target host you want to manage.

Verification

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

```bash
# ssh -i ~/.ssh/id_rsa_foreman_proxy root@client.example.com
```

3.15. 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:

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

2. Download the SSH key from Capsule:

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

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

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

4. Configure permissions for the `authorized_keys` file:

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

3.16. 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. 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:
3.17. 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

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/My_User_ID"
   ```

4. Create a keytab or copy an existing keytab to the directory:
   ```
   # cp My_Client.keytab /var/kerberos/krb5/user/My_User_ID/client.keytab
   ```

5. Change the directory owner to the *foreman-proxy* user:
   ```
   # chown -R foreman-proxy:foreman-proxy "/var/kerberos/krb5/user/My_User_ID"
   ```

6. Ensure that the keytab file is read-only:
   ```
   # chmod -wx "/var/kerberos/krb5/user/My_User_ID/client.keytab"
   ```

7. Restore the SELinux context:
   ```
   # restorecon -RvF /var/kerberos/krb5
   ```

3.18. CONFIGURING KERBEROS AUTHENTICATION FOR REMOTE EXECUTION

You can use Kerberos authentication to establish an SSH connection for remote execution on Satellite hosts.

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

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

   ```bash
   # satellite-installer --scenario satellite \
   --foreman-proxy-plugin-remote-execution-script-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 Remote Execution 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.

Verification

- To confirm that Kerberos authentication is ready to use, run a remote job on the host. For more information, see Executing a Remote Job in Managing Configurations Using Ansible Integration in Red Hat Satellite.

3.19. 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 Executing a Remote Job in Managing Hosts.

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.

Procedure

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 Managing Hosts.

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 Managing Hosts.

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.

### 3.20. EXECUTING A REMOTE JOB

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

To use the CLI instead of the Satellite web UI, see the CLI procedure.

**Procedure**

1. In the Satellite web UI, navigate to Monitor > Jobs and click Run job.

2. Select the Job category and the Job template you want to use, then click Next.

3. Select hosts on which you want to run the job. If you do not select any hosts, the job will run on all hosts you can see in the current context.

   **NOTE**

   If you want to select a host group and all of its subgroups, it is not sufficient to select the host group as the job would only run on hosts directly in that group and not on hosts in subgroups. Instead, you must either select the host group and all of its subgroups or use this search query:

   ```
   hostgroup_fullname ~ "My_Host_Group"
   ```

   Replace My_Host_Group with the name of the top-level host group.

4. If required, provide inputs for the job template. Different templates have different inputs and some templates do not have any inputs. After entering all the required inputs, click Next.

5. Optional: To configure advanced settings for the job, fill in the Advanced fields. To learn more about advanced settings, see Section 3.21, “Advanced Settings in the Job Wizard”.

6. Click Next.

7. Schedule time for the job.
   - To execute the job immediately, keep the pre-selected Immediate execution.
   - To execute the job in future time, select Future execution.
   - To execute the job on regular basis, select Recurring execution.

8. Optional: If you selected future or recurring execution, select the Query type, otherwise click Next.
   - Static query means that job executes on the exact list of hosts that you provided.
   - Dynamic query means that the list of hosts is evaluated just before the job is executed. If you entered the list of hosts based on some filter, the results can be different from when you first used that filter.

   Click Next after you have selected the query type.
9. Optional: If you selected future or recurring execution, provide additional details:
   - For **Future execution**, enter the **Starts at** date and time. You also have the option to select the **Starts before** date and time. If the job cannot start before that time, it will be canceled.
   - For **Recurring execution**, select the start date and time, frequency, and the condition for ending the recurring job. You can choose the recurrence to never end, end at a certain time, or end after a given number of repetitions. You can also add **Purpose** - a special label for tracking the job. There can only be one active job with a given purpose at a time.

   Click **Next** after you have entered the required information.

10. Review job details. You have the option to return to any part of the job wizard and edit the information.

11. Click **Submit** to schedule the job for execution.

**CLI procedure**

1. Enter the following command on Satellite:
   
   ```bash
   # hammer settings set \
   --name=remote_execution_global_proxy \
   --value=false
   ```

2. Find the ID of the job template you want to use:
   
   ```bash
   # hammer job-template list
   ```

3. Show the template details to see parameters required by your template:
   
   ```bash
   # hammer job-template info --id My_Template_ID
   ```

4. Execute a remote job with custom parameters:
   
   ```bash
   # hammer job-invocation create \
   --inputs My_Key_1="My_Value_1",My_Key_2="My_Value_2",... \
   --job-template "My_Template_Name" \
   --search-query "My_Search_Query"
   ```

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

**3.21. ADVANCED SETTINGS IN THE JOB WIZARD**

Some job templates require you to enter advanced settings. Some of the advanced settings are only visible to certain job templates. Below is the list of general advanced settings.

**SSH user**

A user to be used for connecting to the host through SSH.

**Effective user**
A user to be used for executing the job. By default it is the SSH user. If it differs from the SSH user, su or sudo, depending on your settings, is used to switch the accounts.

**Description**

A description template for the job.

**Timeout to kill**

Time in seconds from the start of the job after which the job should be killed if it is not finished already.

**Time to pickup**

Time in seconds after which the job is canceled if it is not picked up by a client. This setting only applies to hosts using **pull-mqtt** transport.

**Password**

Is used if SSH authentication method is a password instead of the SSH key.

**Private key passphrase**

Is used if SSH keys are protected by a passphrase.

**Effective user password**

Is used if effective user is different from the ssh user.

**Concurrency level**

Defines the maximum number of jobs executed at once. This can prevent overload of system resources in a case of executing the job on a large number of hosts.

**Time span**

Distributes the remote execution over the selected number of seconds. Jobs start one at a time in regular intervals to fit the given time window. Similarly to concurrency level, this can also prevent overload of system resources.

**Execution ordering**

Determines the order in which the job is executed on hosts. It can be alphabetical or randomized.

### 3.22. USING EXTENDED CRON LINES

When scheduling a cron job with remote execution, you can use an extended cron line to specify the cadence of the job. The standard cron line contains five fields that specify minute, hour, day of the month, month, and day of the week. For example, `0 5 * * *` stands for every day at 5 AM.

The extended cron line provides the following features:

**You can use # to specify a concrete week day in a month**

For example:

- `0 0 * * mon#1` specifies first Monday of the month
- `0 0 * * fri#3,fri#4` specifies 3rd and 4th Fridays of the month
- `0 7 * * fri#-1` specifies the last Friday of the month at 07:00
- `0 7 * * fri#L` also specifies the last Friday of the month at 07:00
- `0 23 * * mon#2,tue` specifies the 2nd Monday of the month and every Tuesday, at 23:00

**You can use % to specify every n-th day of the month**

For example:
You can use & to specify that the day of the month has to match the day of the week
For example:
- 0 0 30 * 1& specifies 30th day of the month, but only if it is Monday

3.23. SCHEDULING A RECURRING ANSIBLE JOB FOR A HOST
You can schedule a recurring job to run Ansible roles on hosts.

Prerequisite
- Ensure you have the view_foreman_tasks, view_job_invocations, and view_recurring_logics permissions.

Procedure
1. In the Satellite web UI, navigate to Hosts > All Hosts and select the target host on which you want to execute a remote job.
2. On the Ansible tab, select Jobs.
3. Click Schedule recurring job.
4. Define the repetition frequency, start time, and date of the first run in the Create New Recurring Ansible Run window.
5. Click Submit.
6. Optional: View the scheduled Ansible job in host overview or by navigating to Ansible > Jobs.

3.24. SCHEDULING A RECURRING ANSIBLE JOB FOR A HOST GROUP
You can schedule a recurring job to run Ansible roles on host groups.

Procedure
1. In the Satellite web UI, navigate to Configure > Host groups.
2. In the Actions column, select Configure Ansible Job for the host group you want to schedule an Ansible roles run for.
3. Click Schedule recurring job.
4. Define the repetition frequency, start time, and date of the first run in the Create New Recurring Ansible Run window.
5. Click Submit.
3.25. MONITORING JOBS

You can monitor the progress of a 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. In the Satellite web UI, navigate to Monitor > Jobs. 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.

CLI procedure

1. Find the ID of a job:

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

2. Monitor the job output:

   ```
   # hammer job-invocation output \\n   --host My_Host_Name \\n   --id My_Job_ID
   ```

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

   ```
   # hammer job-invocation cancel \\n   --id My_Job_ID
   ```

3.26. SETTING THE JOB RATE LIMIT ON CAPSULE

You can limit the maximum number of active jobs on a Capsule at a time to prevent performance spikes. The job is active from the time Capsule first tries to notify the host about the job until the job is finished on the host.

The job rate limit only applies to mqtt based jobs.

**NOTE**

The optimal maximum number of active jobs depends on the computing resources of your Capsule Server. By default, the maximum number of active jobs is unlimited.

Procedure
Set the maximum number of active jobs using `satellite-installer`:

```
# satellite-installer \
--foreman-proxy-plugin-remote-execution-script-mqtt-rate-limit MAX_JOBS_NUMBER
```

For example:

```
# satellite-installer \
--foreman-proxy-plugin-remote-execution-script-mqtt-rate-limit 200
```
CHAPTER 4. INTEGRATING RED HAT SATELLITE AND ANSIBLE AUTOMATION CONTROLLER

You can integrate Red Hat Satellite and Ansible Automation Controller to use Satellite Server as a dynamic inventory source for Ansible Automation Controller. Ansible Automation Controller is a component of the Red Hat Ansible Automation Platform.

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

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

To add Satellite Server to Ansible Automation Controller as a dynamic inventory item, you must create a credential for a Satellite Server user on Ansible Automation Controller, add an Ansible Automation Controller 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 Automation Controller on the same network or subnet.

Procedure

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

   Table 4.1. Satellite Credentials

<table>
<thead>
<tr>
<th>Credential Type</th>
<th></th>
<th>Red Hat Satellite 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Satellite URL</td>
<td></td>
<td><a href="https://satellite.example.com">https://satellite.example.com</a></td>
</tr>
<tr>
<td>Username</td>
<td></td>
<td>The username of the Satellite user with the integration role.</td>
</tr>
<tr>
<td>Password</td>
<td></td>
<td>The password of the Satellite user.</td>
</tr>
</tbody>
</table>

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

Table 4.2. Inventory Source Options

<table>
<thead>
<tr>
<th>Source</th>
<th>Red Hat Satellite 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Credential</td>
<td>The credential you create for Satellite Server.</td>
</tr>
<tr>
<td>Overwrite</td>
<td>Select</td>
</tr>
<tr>
<td>Overwrite Variables</td>
<td>Select</td>
</tr>
<tr>
<td>Update on Launch</td>
<td>Select</td>
</tr>
<tr>
<td>Cache Timeout</td>
<td>90</td>
</tr>
</tbody>
</table>

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 Automation Controller to run playbooks to configure your newly created hosts. This is called provisioning callback in Ansible Automation Controller.

The provisioning callback function triggers a playbook run from Ansible Automation Controller as part of the provisioning process. The playbook configures the host after the provisioning process.

For more information about provisioning callbacks, see Provisioning Callbacks in the Automation Controller 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 Automation Controller. For more information, see Integrating Satellite and Ansible Automation Controller.

In the Ansible Automation Controller 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 Automation Controller User Guide.

2. Create a project. For more information, see Projects in the Ansible Automation Controller User Guide.

3. Add a job template to your project. For more information, see Job Templates in the Automation Controller 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 Automation Controller User Guide.

Procedure

1. In the 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

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ansible_tower_provisioning</td>
<td>true</td>
<td>Enables Provisioning Callback.</td>
</tr>
<tr>
<td>ansible_tower_fqdn</td>
<td>controller.exampl.e.com</td>
<td>The fully qualified domain name (FQDN) of your Ansible Automation Controller. Do not add https because this is appended by Satellite.</td>
</tr>
<tr>
<td>ansible_job_template_id</td>
<td>template_ID</td>
<td>The ID of your provisioning template that you can find in the URL of the template: /templates/job_template/5.</td>
</tr>
<tr>
<td>ansible_host_config_key</td>
<td>config_KEY</td>
<td>The host configuration key that your job template generates in Ansible Automation Controller.</td>
</tr>
</tbody>
</table>

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 Automation Controller...
```

**Manual Provisioning Callback**

- You can use the provisioning callback URL and the host configuration key from a host to call Ansible Automation Controller:

```
# curl -k -s --data curl --insecure --data host_config_key=my_config_key
https://controller.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.
CHAPTER 5. JOB TEMPLATE EXAMPLES AND EXTENSIONS

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

5.1. 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 nginx service on clients:

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

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:

```erb
<%= 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.

5.2. DEFAULT JOB TEMPLATE CATEGORIES

<table>
<thead>
<tr>
<th>Job template category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Packages</td>
<td>Templates for performing package related actions. Install, update, and remove actions are included by default.</td>
</tr>
<tr>
<td>Puppet</td>
<td>Templates for executing Puppet runs on target hosts.</td>
</tr>
<tr>
<td>Power</td>
<td>Templates for performing power related actions. Restart and shutdown actions are included by default.</td>
</tr>
<tr>
<td>Commands</td>
<td>Templates for executing custom commands on remote hosts.</td>
</tr>
<tr>
<td>Services</td>
<td>Templates for performing service related actions. Start, stop, restart, and status actions are included by default.</td>
</tr>
<tr>
<td>Katello</td>
<td>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.</td>
</tr>
</tbody>
</table>

5.3. 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.

**Procedure**

1. In the Satellite web UI, navigate to **Hosts > Job templates**.

2. Click **New Job Template**

3. 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.

4. On the **Job** tab, set **Job category** to **Commands**.

5. 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.

6. Click **Required** so that the command cannot be executed without the user specified parameter.

7. Select **User input** from the **Input type** list. Enter a description to be shown during job invocation, for example **Target directory for restorecon**.

8. Click **Submit**. For more information, see **Executing a restorecon Template on Multiple Hosts** in **Managing Hosts**.

### 5.4. RENDERING A RESTORECON TEMPLATE

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

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

### 5.5. EXECUTING A RESTORECON TEMPLATE ON MULTIPLE HOSTS

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

**Procedure**

1. In the Satellite web UI, navigate to **Monitor > Jobs** and click **Run job**.

2. Select **Commands** as **Job category** and **Run Command – restorecon** as **Job template** and click **Next**.
3. Select the hosts on which you want to run the job. If you do not select any hosts, the job will run on all hosts you can see in the current context.

4. In the directory field, provide a directory, for example /home, and click Next.

5. Optional: To configure advanced settings for the job, fill in the Advanced fields. To learn more about advanced settings, see Section 3.21, “Advanced Settings in the Job Wizard”. When you are done entering the advanced settings or if it is not required, click Next.

6. Schedule time for the job.
   - To execute the job immediately, keep the pre-selected Immediate execution.
   - To execute the job in future time, select Future execution.
   - To execute the job on regular basis, select Recurring execution.

7. Optional: If you selected future or recurring execution, select the Query type, otherwise click Next.
   - Static query means that the job executes on the exact list of hosts that you provided.
   - Dynamic query means that the list of hosts is evaluated just before the job is executed. If you entered the list of hosts based on some filter, the results can be different from when you first used that filter. Click Next after you have selected the query type.

8. Optional: If you selected future or recurring execution, provide additional details:
   - For Future execution, enter the Starts at date and time. You also have the option to select the Starts before date and time. If the job cannot start before that time, it will be canceled.
   - For Recurring execution, select the start date and time, frequency, and condition for ending the recurring job. You can choose the recurrence to never end, end at a certain time, or end after a given number of repetitions. You can also add Purpose - a special label for tracking the job. There can only be one active job with a given purpose at a time. Click Next after you have entered the required information.

9. Review job details. You have the option to return to any part of the job wizard and edit the information.

10. Click Submit to schedule the job for execution.

## 5.6. 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 Setting up Job Templates, and specify the following string in the template editor:

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