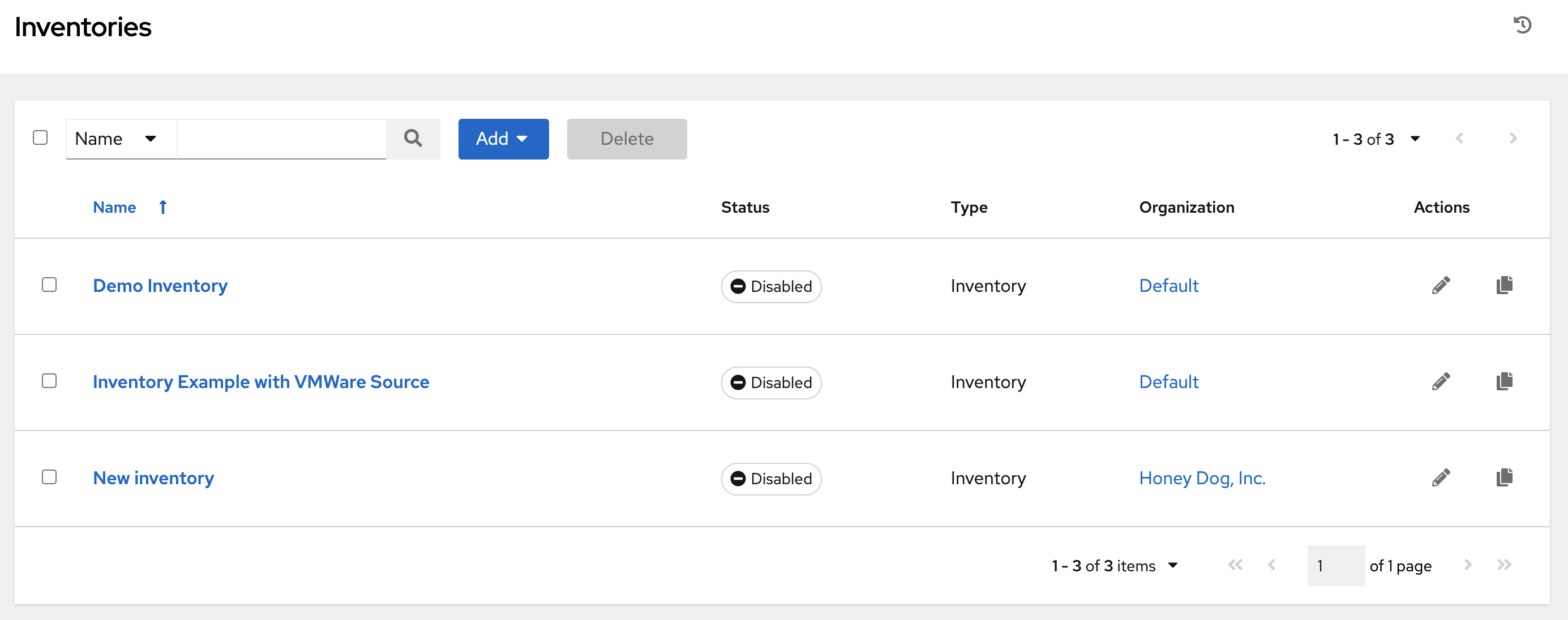
**Inventories**

An Inventory is a collection of hosts against which jobs may be launched, the same as an Ansible inventory file. Inventories are divided into groups and these groups contain the actual hosts. Groups may be sourced manually, by entering host names into the automation controller, or from one of its supported cloud providers.

**Note**

If you have a custom dynamic inventory script, or a cloud provider that is not yet supported natively in the controller, you can also import that into the controller. Refer to Inventory File Importing in the *Automation Controller Administration Guide*.

The Inventories window displays a list of the inventories that are currently available. The inventory list may be sorted by name and searched type, organization, description, owners and modifiers of the inventory, or additional criteria as needed.



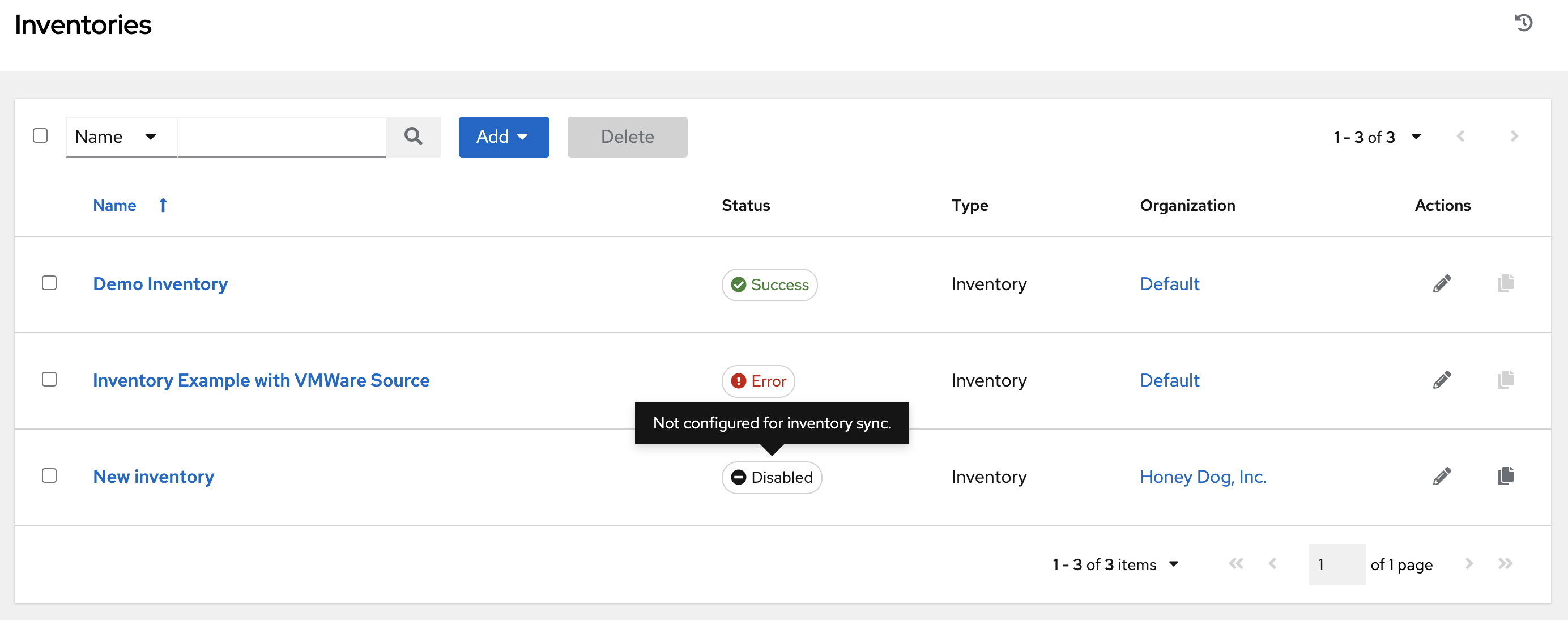
The list of Inventory details includes:

* **Name**: The inventory name. Clicking the Inventory name navigates to the properties screen for the selected inventory, which shows the inventory’s groups and hosts. (This view is also accessible from the edit button icon.)
* **Status**

The statuses are:

* **Success**: when the inventory source sync completed successfully
* **Disabled**: no inventory source added to the inventory
* **Error**: when the inventory source sync completed with error

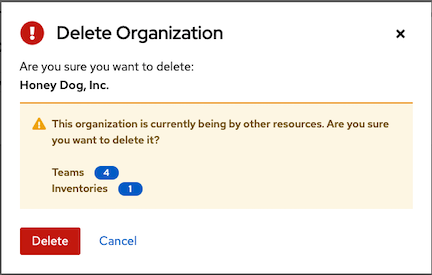
An example of inventories of various states, including one with detail for a disabled state:



* **Type**: Identifies whether it is a standard inventory or a Smart Inventory.
* **Organization**: The organization to which the inventory belongs.
* **Actions**: The following actions are available for the selected inventory:
  + **Edit** (edit button): Edit the properties for the selected inventory
  + **Copy** (copy): Makes a copy of an existing inventory as a template for creating a new one

**Note**

If deleting items that are used by other work items, a message opens listing the items are affected by the deletion and prompts you to confirm the deletion. Some screens will contain items that are invalid or previously deleted, so they will fail to run. Below is an example of such a message:



**18.1. Smart Inventories**

A Smart Inventory is a collection of hosts defined by a stored search that can be viewed like a standard inventory and made to be easily used with job runs. Organization administrators have admin permission to inventories in their organization and can create a Smart Inventories. A Smart Inventory is identified by KIND=smart. You can define a Smart Inventory using the same method being used with Search. InventorySource is directly associated with an Inventory.

The Inventory model has the following new fields that are blank by default but are set accordingly for Smart Inventories:

* kind is set to smart for Smart Inventories
* host\_filter is set AND kind is set to smart for Smart Inventories.

The host model has a related endpoint, smart\_inventories that identifies a set of all the Smart Inventory a host is associated with. The membership table is updated every time a job runs against a smart inventory.

**Note**

To update the memberships more frequently, you can change the file-based setting AWX\_REBUILD\_SMART\_MEMBERSHIP to **True** (default is False). This will update memberships in the following events:

* a new host is added
* an existing host is modified (updated or deleted)
* a new Smart Inventory is added
* an existing Smart Inventory is modified (updated or deleted)

You can view actual inventories without being editable:

* Names of Host and Group created as a result of an inventory source sync
* Group records cannot be edited or moved

You cannot create hosts from a Smart Inventory host endpoint (/inventories/N/hosts/) as with a normal inventory. The administrator of a Smart Inventory has permission to edit fields such as the name, description, variables, and the ability to delete, but does not have the permission to modify the host\_filter, because that will affect which hosts (that have a primary membership inside another inventory) are included in the smart inventory. Note, host\_filter only apply to hosts inside of inventories inside of the Smart Inventory’s organization.

In order to modify the host\_filter, you need to be the organization administrator of the inventory’s organization. Organization admins already have implicit “admin” access to all inventories inside the organization, therefore, this does not convey any permissions they did not already possess.

Administrators of the Smart Inventory can grant other users (who are not also admins of your organization) permissions like “use” “adhoc” to the smart inventory, and these will allow the actions indicate by the role, just like other standard inventories. However, this will not give them any special permissions to hosts (which live in a different inventory). It will not allow them direct read permission to hosts, or permit them to see additional hosts under /#/hosts/, although they can still view the hosts under the smart inventory host list.

In some situations, you can modify the following:

* A new Host manually created on Inventory w/ inventory sources
* In Groups that were created as a result of inventory source syncs
* Variables on Host and Group are changeable

Hosts associated with the Smart Inventory are manifested at view time. If the results of a Smart Inventory contains more than one host with identical hostnames, only one of the matching hosts will be included as part of the Smart Inventory, ordered by Host ID.

**18.1.1. Smart Host Filter**

You can use a search filter to populate hosts for an inventory. This feature utilized the capability of the fact searching feature.

Facts generated by an Ansible playbook during a Job Template run are stored by the automation controller into the database whenever use\_fact\_cache=True is set per-Job Template. New facts are merged with existing facts and are per-host. These stored facts can be used to filter hosts via the /api/v2/hosts endpoint, using the GET query parameter host\_filter For example: /api/v2/hosts?host\_filter=ansible\_facts\_\_ansible\_processor\_vcpus=8

The host\_filter parameter allows for:

* grouping via ()
* use of the boolean and operator:
  + \_\_ to reference related fields in relational fields
  + \_\_ is used on ansible\_facts to separate keys in a JSON key path
  + [] is used to denote a json array in the path specification
  + "" can be used in the value when spaces are wanted in the value
* “classic” Django queries may be embedded in the host\_filter

Examples:

/api/v2/hosts/?host\_filter=name=localhost

/api/v2/hosts/?host\_filter=ansible\_facts\_\_ansible\_date\_time\_\_weekday\_number="3"

/api/v2/hosts/?host\_filter=ansible\_facts\_\_ansible\_processor[]="GenuineIntel"

/api/v2/hosts/?host\_filter=ansible\_facts\_\_ansible\_lo\_\_ipv6[]\_\_scope="host"

/api/v2/hosts/?host\_filter=ansible\_facts\_\_ansible\_processor\_vcpus=8

/api/v2/hosts/?host\_filter=ansible\_facts\_\_ansible\_env\_\_PYTHONUNBUFFERED="true"

/api/v2/hosts/?host\_filter=(name=localhost or name=database) and (groups\_\_name=east or groups\_\_name="west coast") and ansible\_facts\_\_an

You can search host\_filter by host name, group name, and Ansible facts.

The format for a group search is:

groups**.**name:groupA

The format for a fact search is:

ansible\_facts**.**ansible\_fips:false

You can also perform Smart Search searches, which consist a host name and host description.

host\_filter**=**name**=**my\_host

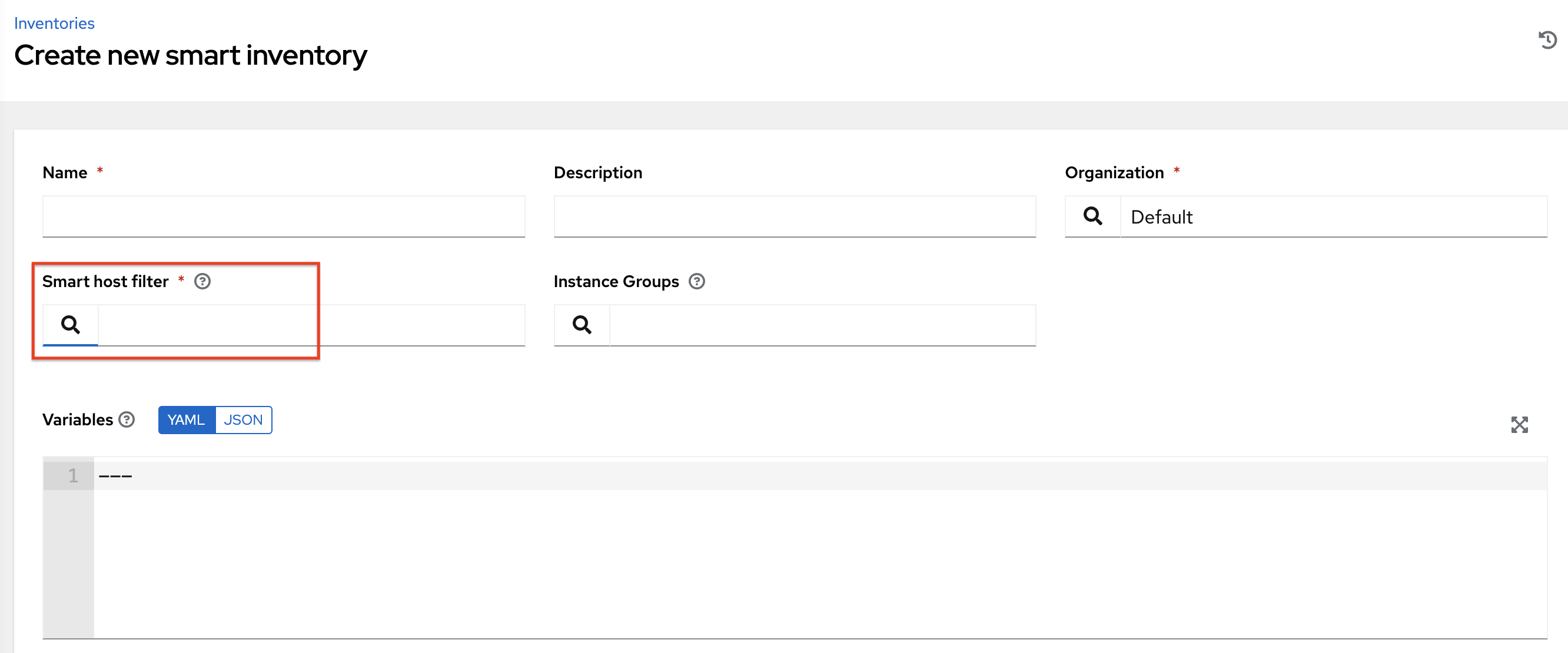
If a search term in host\_filter is of string type, to make the value a number (e.g. 2.66), or a JSON keyword (e.g. null, true or false) valid, add double quotations around the value to prevent the controller from mistakenly parsing it as a non-string:

host\_filter**=**ansible\_facts\_\_packages\_\_dnsmasq[]\_\_version**=**"2.66"

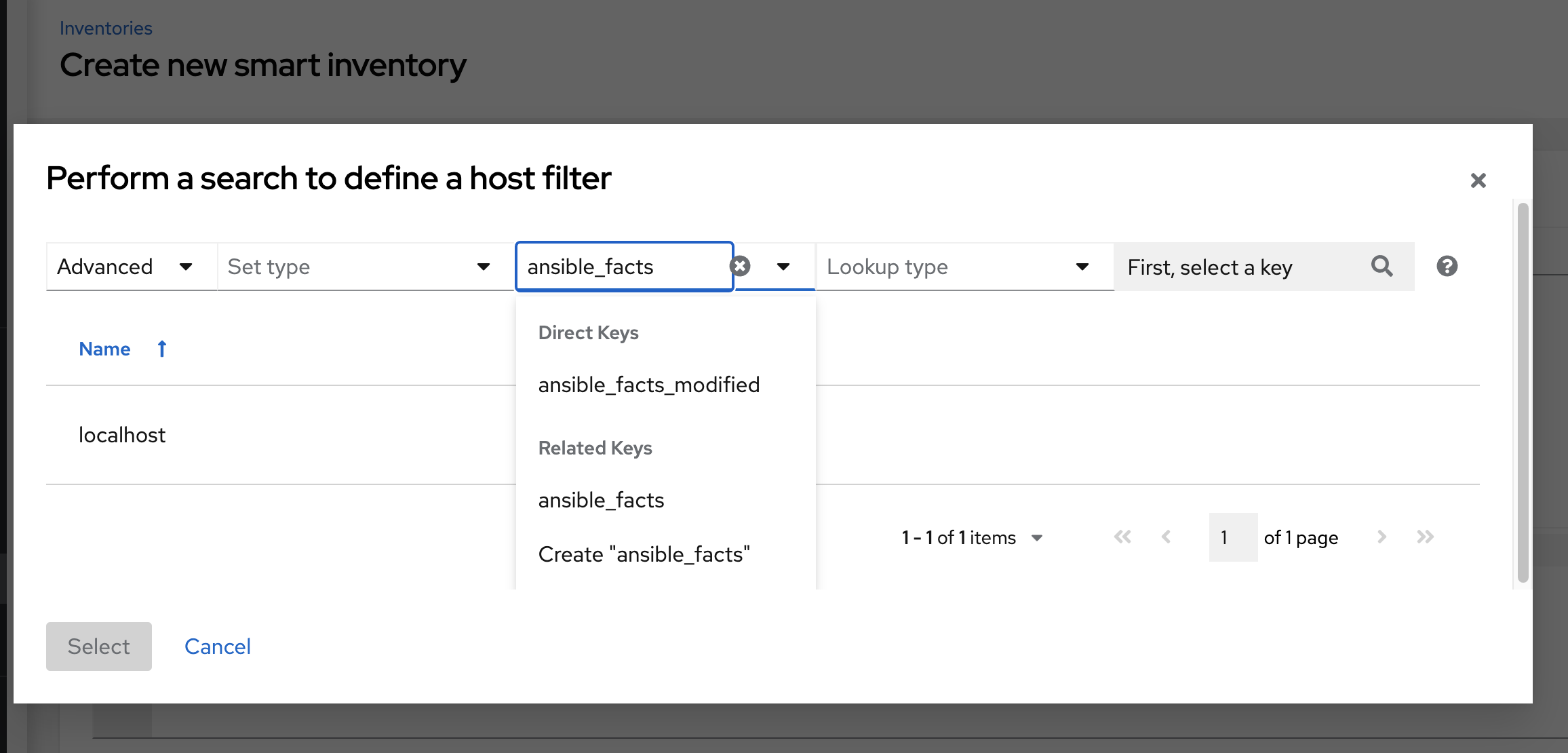
**18.1.2. Define host filter with ansible\_facts**

To use ansible\_facts to define the host filter when creating Smart Inventories, perform the following steps:

1. In the *Create new smart inventory screen*, click the search button next to the **Smart host filter** field to open a pop-up window to filter hosts for this inventory.



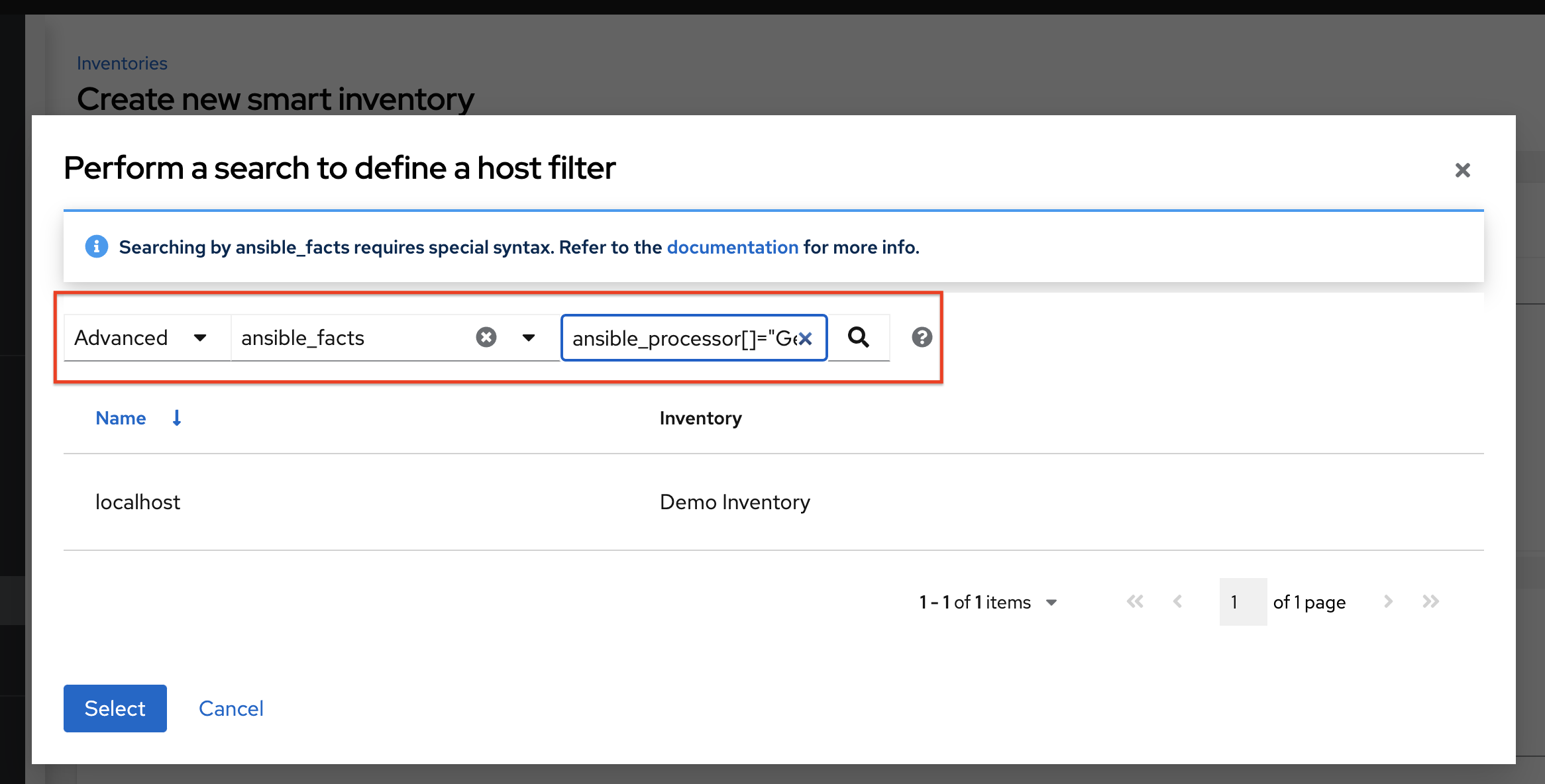
1. In the search pop-up window, change the search criteria from **Name** to **Advanced** and select **ansible\_facts** from the **Key** field.



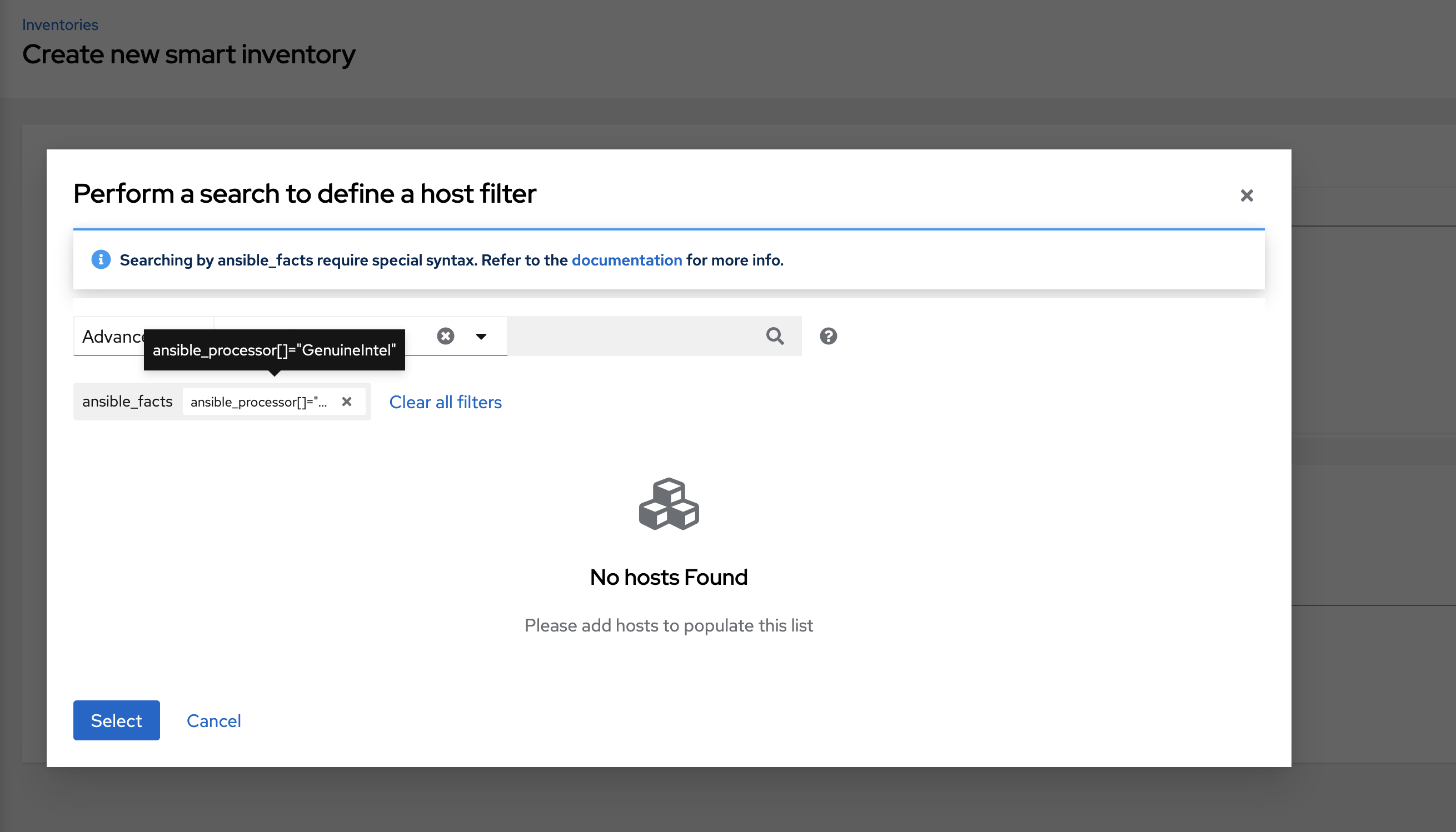
If you wanted to add an ansible fact of

/api/v2/hosts/?host\_filter=ansible\_facts\_\_ansible\_processor[]="GenuineIntel"

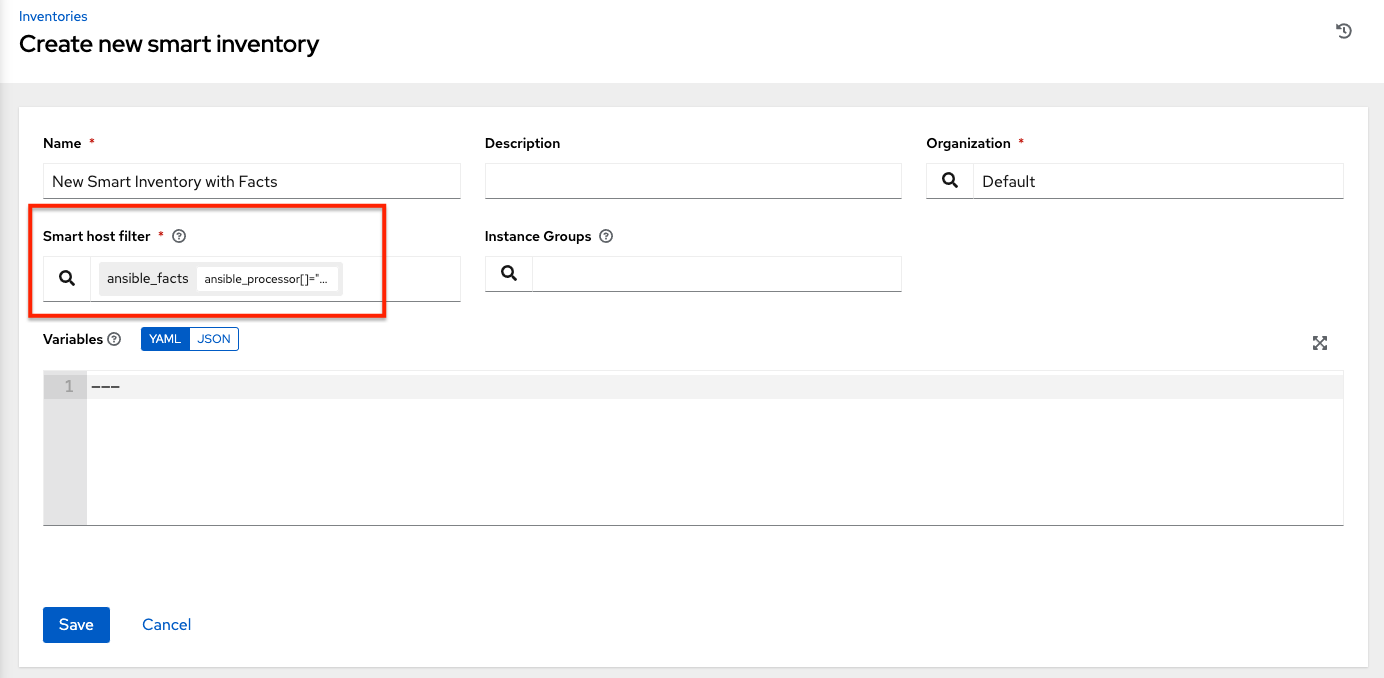
In the search field, enter ansible\_processor[]="GenuineIntel" (no extra spaces or \_\_ before the value) and press **[Enter]**.



The resulting search criteria for the specified ansible fact populates in the lower part of the window.

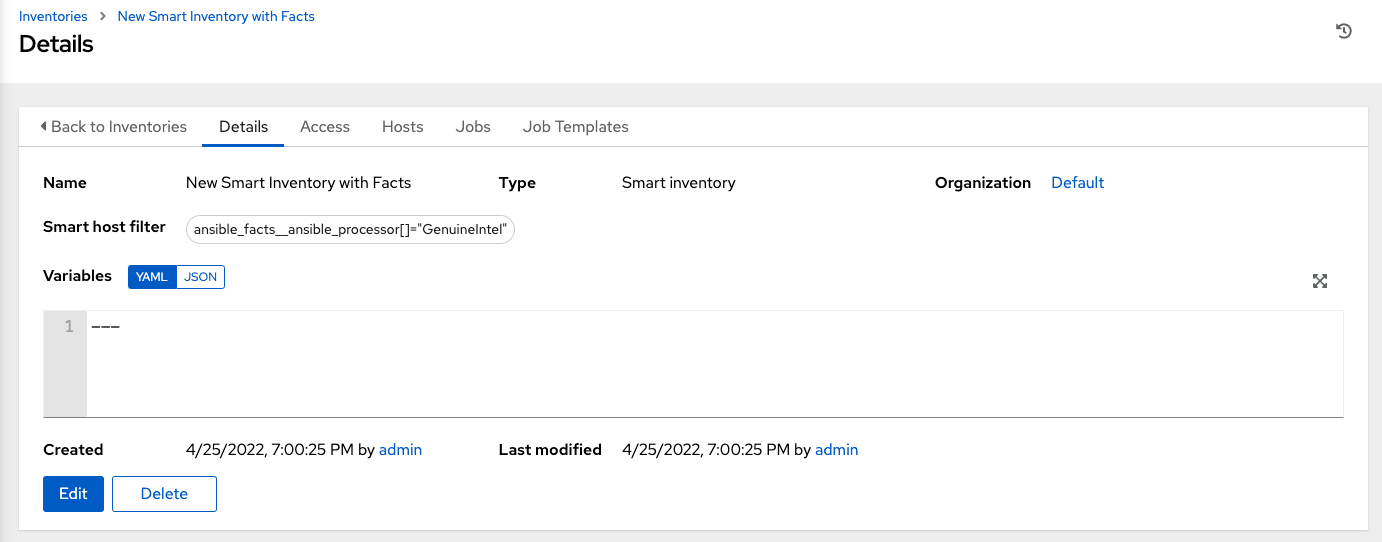


1. Click **Select** to add it to the **Smart host filter** field.

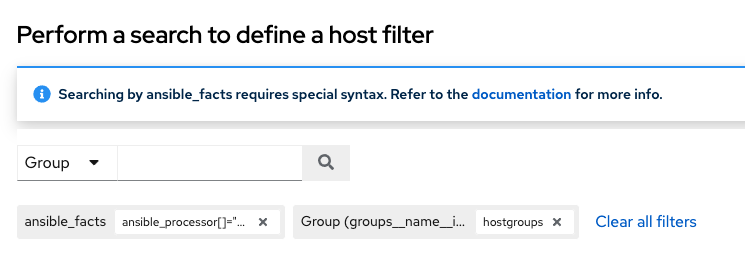


1. Click **Save** to save the new Smart Inventory.

The Details tab of the new Smart Inventory opens and displays the specified ansible facts in the **Smart host filter** field.



1. From the Details view, you can edit the **Smart host filter** field by clicking **Edit** and delete existing filter(s), clear all existing filters, or add new ones.



**18.2. Inventory Plugins**

Inventory updates use dynamically-generated YAML files which are parsed by their respective inventory plugin. In Automation Controller Version 4.3, users can provide the new style inventory plugin config directly to the controller via the inventory source source\_vars for all the following inventory sources:

* Amazon Web Services EC2
* Google Compute Engine
* Microsoft Azure Resource Manager
* VMware vCenter
* Red Hat Satellite 6
* Red Hat Insights
* OpenStack
* Red Hat Virtualization
* Red Hat Ansible Automation Platform

Newly created configurations for inventory sources will contain the default plugin configuration values. If you want your newly created inventory sources in 3.8 to match the output of a 3.7 source, you must apply a specific set of configuration values for that source. To ensure backward compatibility, the controller uses “templates” for each of these sources to force the output of inventory plugins into the legacy format. Refer to Supported Inventory Plugin Templates section of this guide for each source and their respective templates to help you migrate to the new style inventory plugin output.

source\_vars that contain plugin: foo.bar.baz as a top-level key will be replaced with the appropriate fully-qualified inventory plugin name at runtime based on the InventorySource source. For example, if ec2 is selected for the InventorySource then, at run-time, plugin will be set to amazon.aws.aws\_ec2.

**18.3. Add a new inventory**

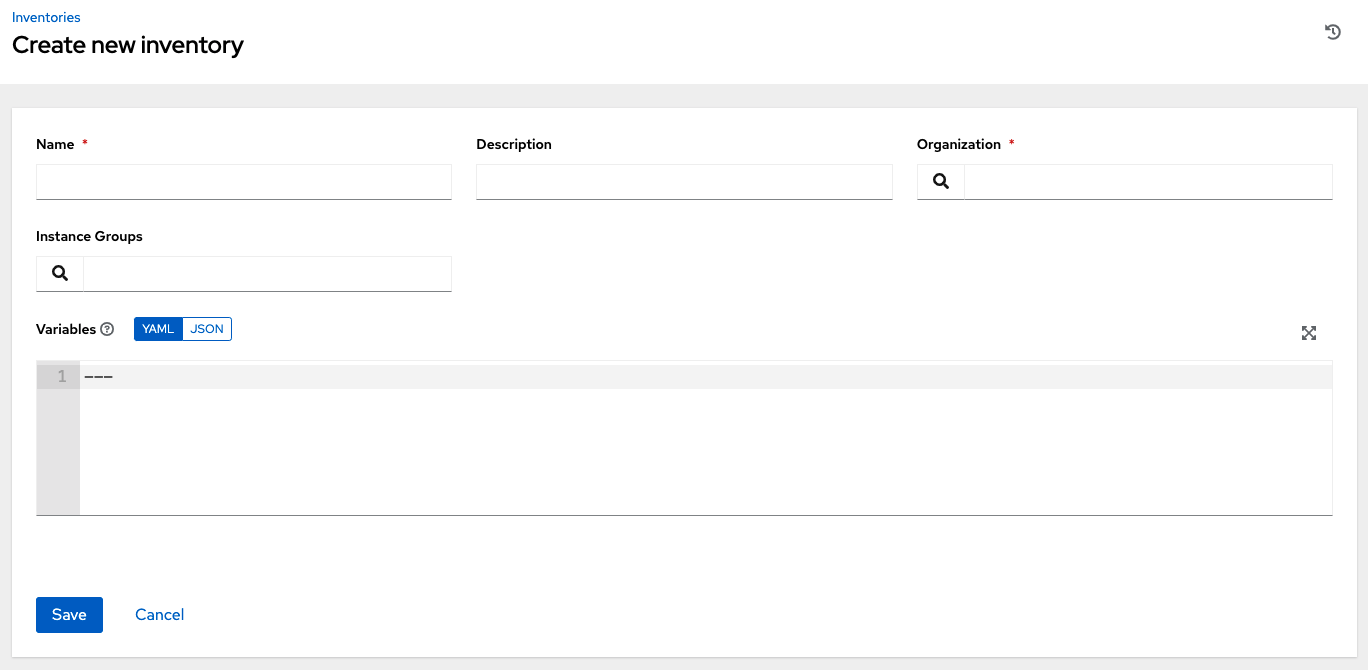
Adding a new inventory involves several components:

* Add permissions
* Add groups
* Add hosts
* Add source
* View completed jobs

To create a new inventory or Smart Inventory:

1. Click the **Add** button, and select the type of inventory to create.

The type of inventory is identified at the top of the create form.

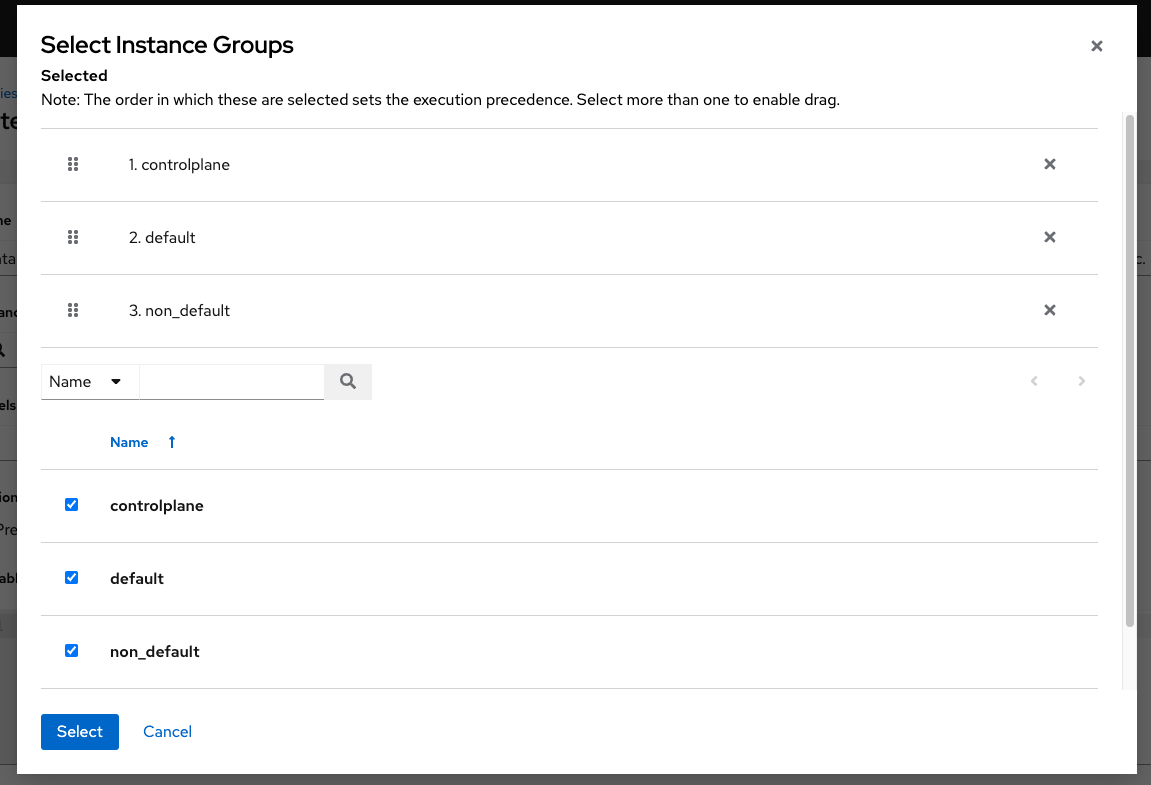


1. Enter the appropriate details into the following fields:

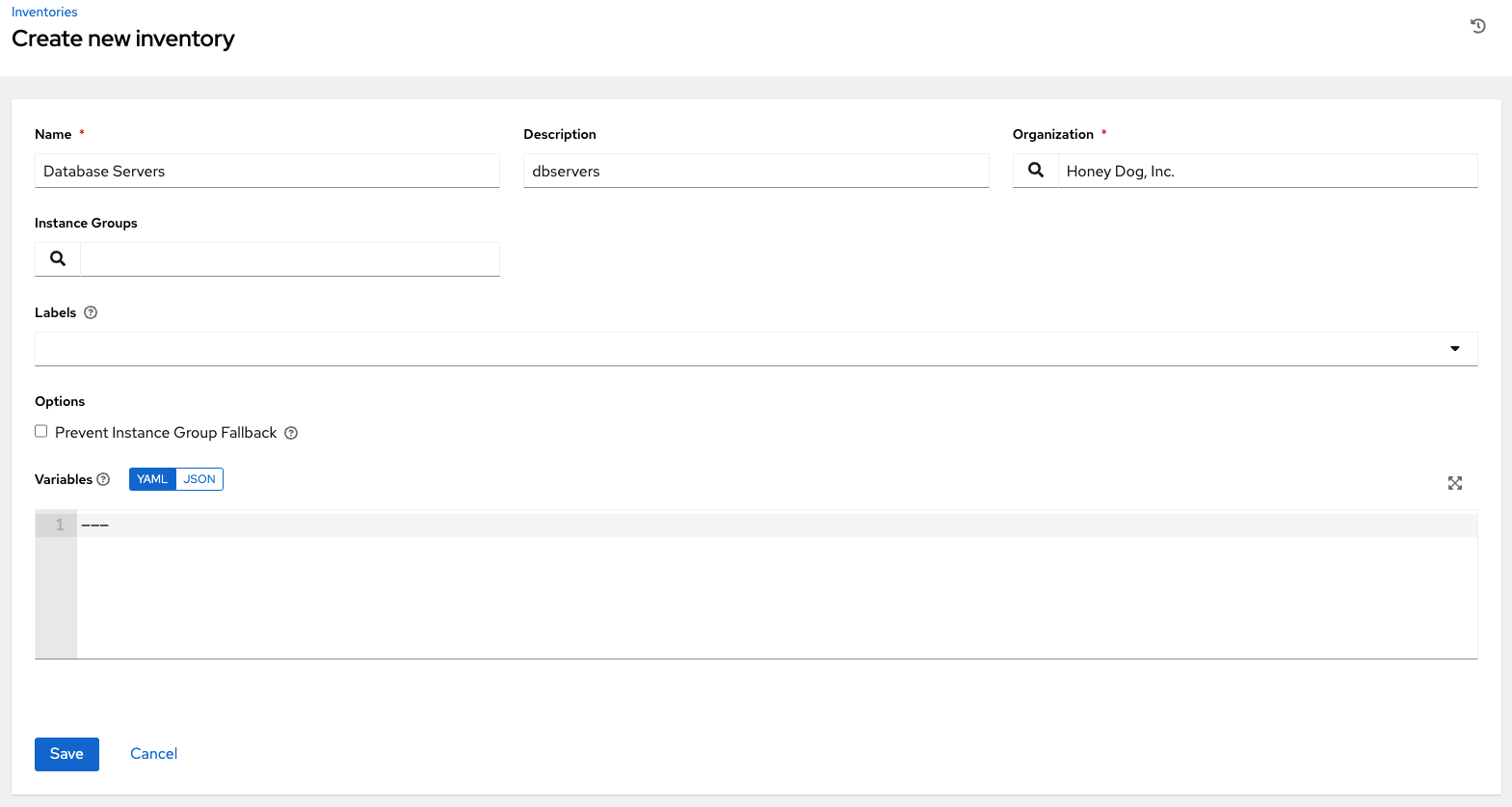
* **Name**: Enter a name appropriate for this inventory.
* **Description**: Enter an arbitrary description as appropriate (optional).
* **Organization**: Required. Choose among the available organizations.
* **Smart Host Filter**: (Only applicable to Smart Inventories) Click the search button to open a separate window to filter hosts for this inventory. These options are based on the organization you chose.

Filters are similar to tags in that tags are used to filter certain hosts that contain those names. Therefore, to populate the **Smart Host Filter** field, you are specifying a tag that contains the hosts you want, not actually selecting the hosts themselves. Enter the tag in the **Search** field and press [Enter]. Filters are case-sensitive. Refer to the Smart Host Filter section for more information.

* **Instance Groups**: Click the search button to open a separate window. Choose the instance group(s) for this inventory to run on. If the list is extensive, use the search to narrow the options. You may select multiple instance groups and sort them in the order you want them ran.



* **Options**: Check the **Prevent Instance Group Fallback** option to allow only the instance groups listed in the **Instance Groups** field above to execute the job. If unchecked, all available instances in the execution pool will be used based on the hierarchy described in Control Where a Job Runs. Click the help icon for additional information.
* **Variables**: Variable definitions and values to be applied to all hosts in this inventory. Enter variables using either JSON or YAML syntax. Use the radio button to toggle between the two.

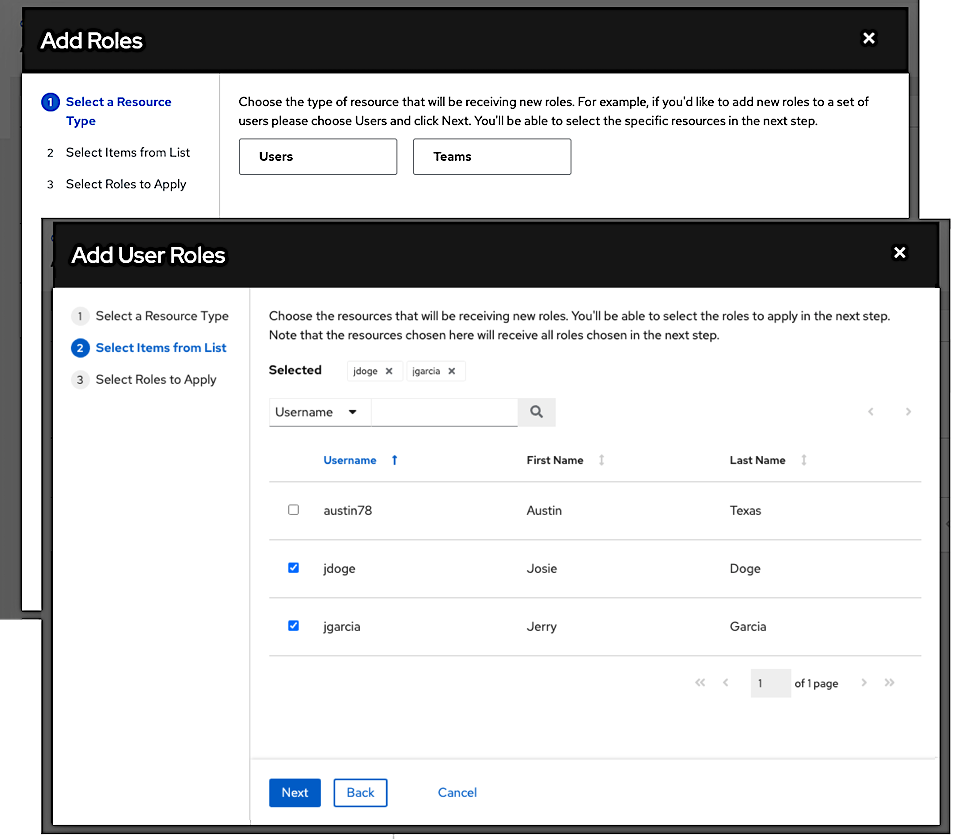


1. Click **Save** when done.

After saving the new inventory, you can proceed with configuring permissions, groups, hosts, sources, and view completed jobs, if applicable to the type of inventory. For more instructions, refer to the subsequent sections.

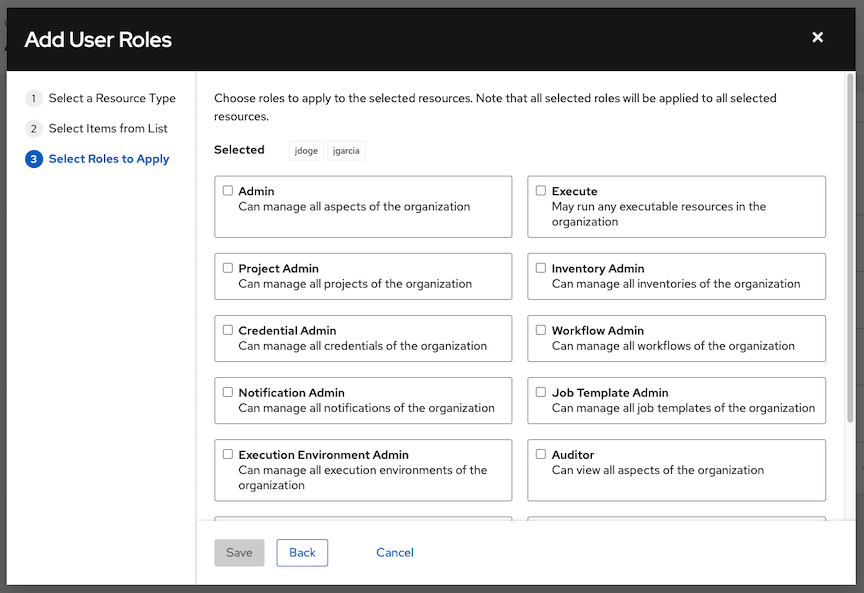
**18.3.1. Add permissions**

1. In the **Access** tab, click the **Add** button.
2. Select a user or team to add and click **Next**
3. Select one or more users or teams from the list by clicking the check box(es) next to the name(s) to add them as members and click **Next**.



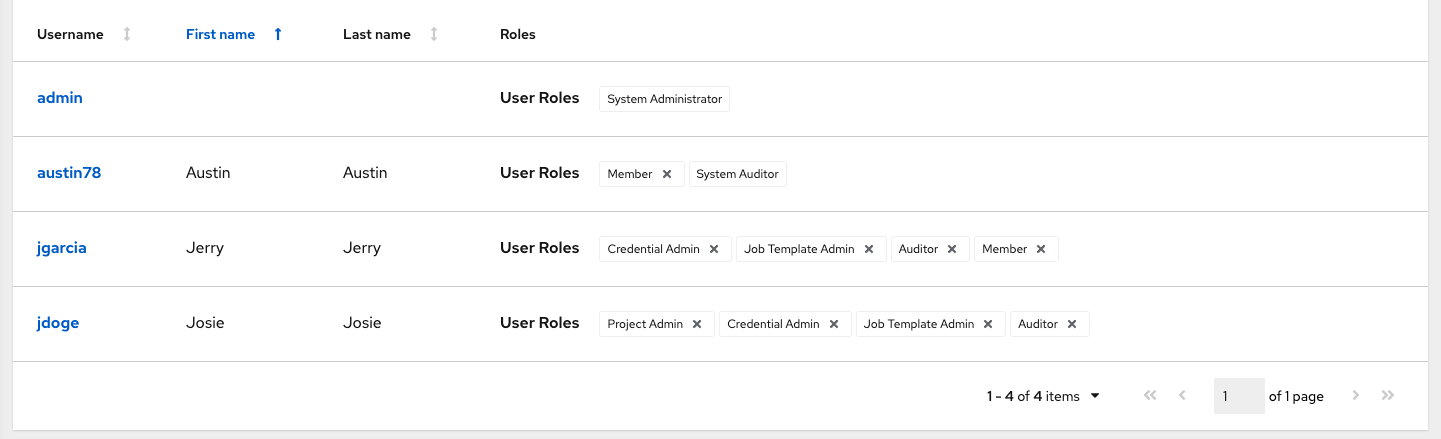
In this example, two users have been selected to be added.

1. Select the role(s) you want the selected user(s) or team(s) to have. Be sure to scroll down for a complete list of roles. Different resources have different options available.

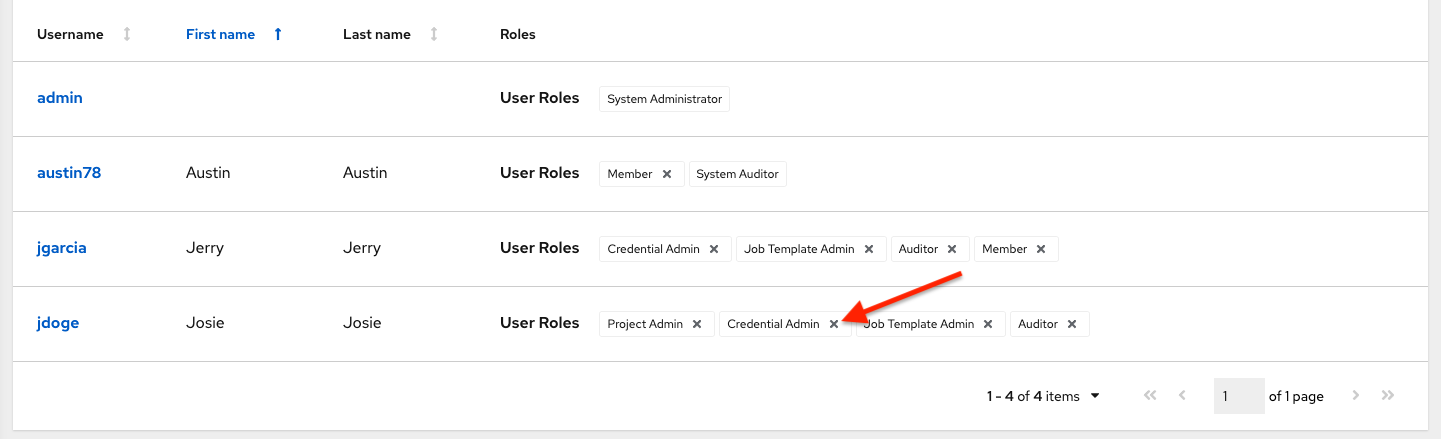


1. Click the **Save** button to apply the roles to the selected user(s) or team(s) and to add them as members.

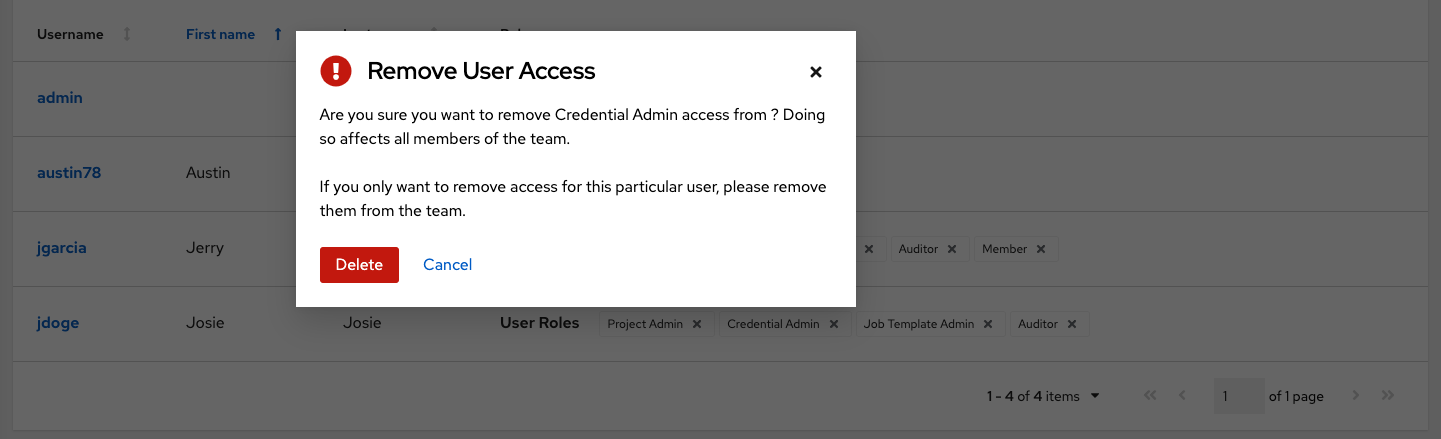
The Add Users/Teams window closes to display the updated roles assigned for each user and team.



To remove roles for a particular user, click the disassociate (x) button next to its resource.



This launches a confirmation dialog, asking you to confirm the disassociation.



**18.3.2. Add groups**

Inventories are divided into groups, which may contain hosts and other groups, and hosts. Groups are only applicable to standard inventories and is not a configurable directly through a Smart Inventory. You can associate an existing group through host(s) that are used with standard inventories. There are several actions available for standard inventories:

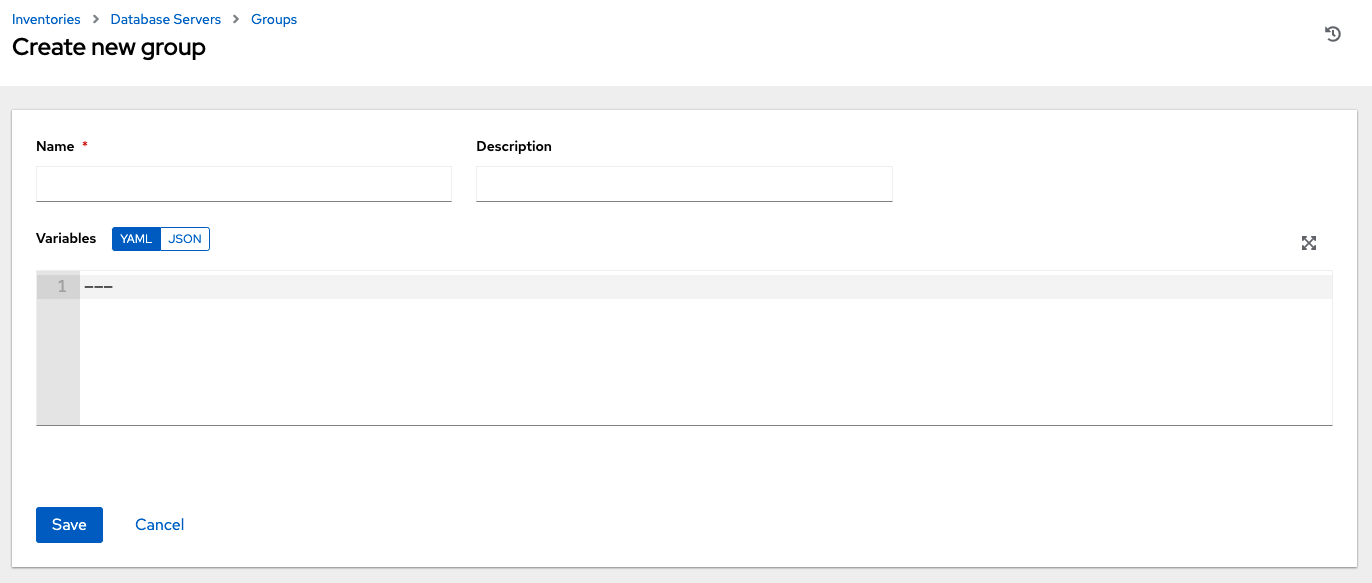
* Create a new Group
* Create a new Host
* Run a command on the selected Inventory
* Edit Inventory properties
* View activity streams for Groups and Hosts
* Obtain help building your Inventory

**Note**

Inventory sources are not associated with groups. Spawned groups are top-level and may still have child groups, and all of these spawned groups may have hosts.

To create a new group for an inventory:

1. Click the **Add** button to open the **Create Group** window.



1. Enter the appropriate details into the required and optional fields:

* **Name**: Required
* **Description**: Enter an arbitrary description as appropriate (optional)
* **Variables**: Enter definitions and values to be applied to all hosts in this group. Enter variables using either JSON or YAML syntax. Use the radio button to toggle between the two.

1. When done, click **Save**.

**18.3.2.1. Add groups within groups**

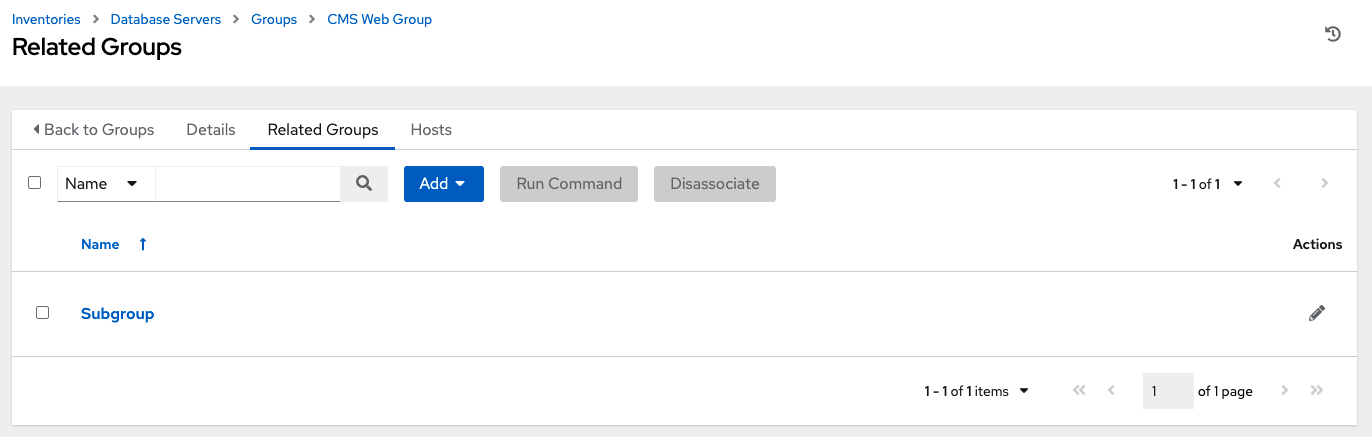
To add groups within groups:

1. Click the **Related Groups** tab.
2. Click the **Add** button, and select whether to add a group that already exists in your configuration or create a new group.
3. If creating a new group, enter the appropriate details into the required and optional fields:

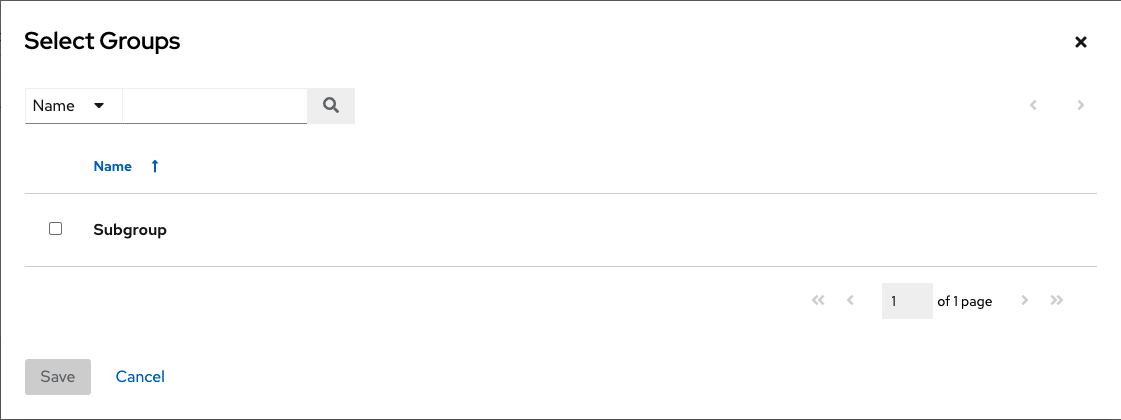
* **Name**: Required
* **Description**: Enter an arbitrary description as appropriate (optional)
* **Variables**: Enter definitions and values to be applied to all hosts in this group. Enter variables using either JSON or YAML syntax. Use the radio button to toggle between the two.

1. When done, click **Save**.

The **Create Group** window closes and the newly created group displays as an entry in the list of groups associated with the group that it was created for.



If you chose to add an existing group, available groups will appear in a separate selection window.



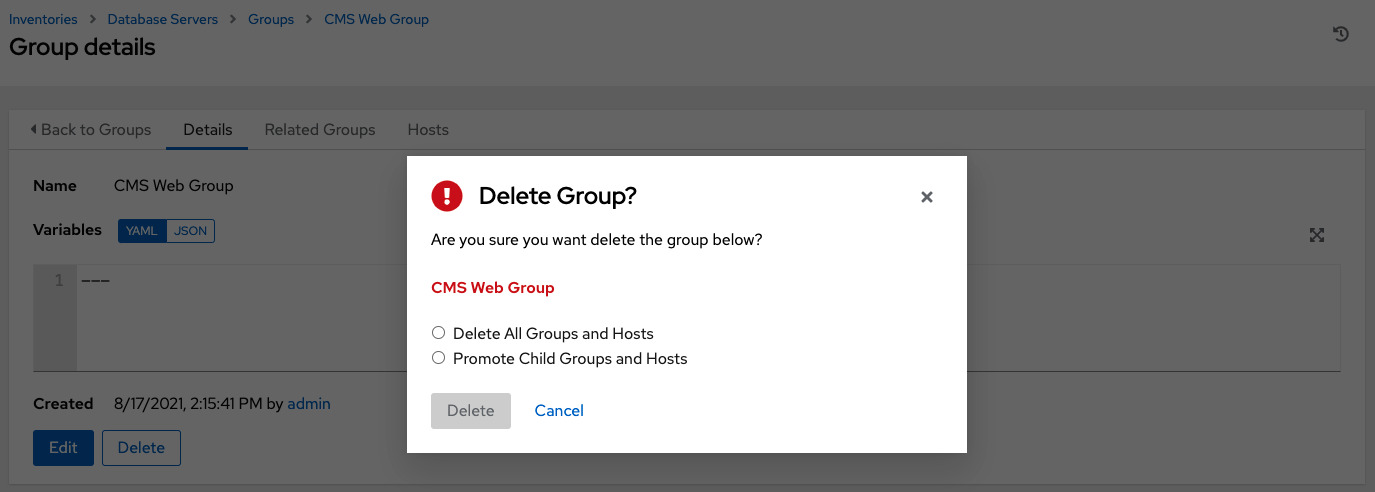
Once a group is selected, it displays as an entry in the list of groups associated with the group.

5. To configure additional groups and hosts under the subgroup, click on the name of the subgroup from the list of groups and repeat the same steps described in this section.

**18.3.2.2. View or edit inventory groups**

The list view displays all your inventory groups at once, or you can filter it to only display the root group(s). An inventory group is considered a root group if it is not a subset of another group.

You may be able to delete a subgroup without concern for dependencies, since the controller will look for dependencies such as any child groups or hosts. If any exists, a confirmation dialog displays for you to choose whether to delete the root group and all of its subgroups and hosts; or promote the subgroup(s) so they become the top-level inventory group(s), along with their host(s).



**18.3.3. Add hosts**

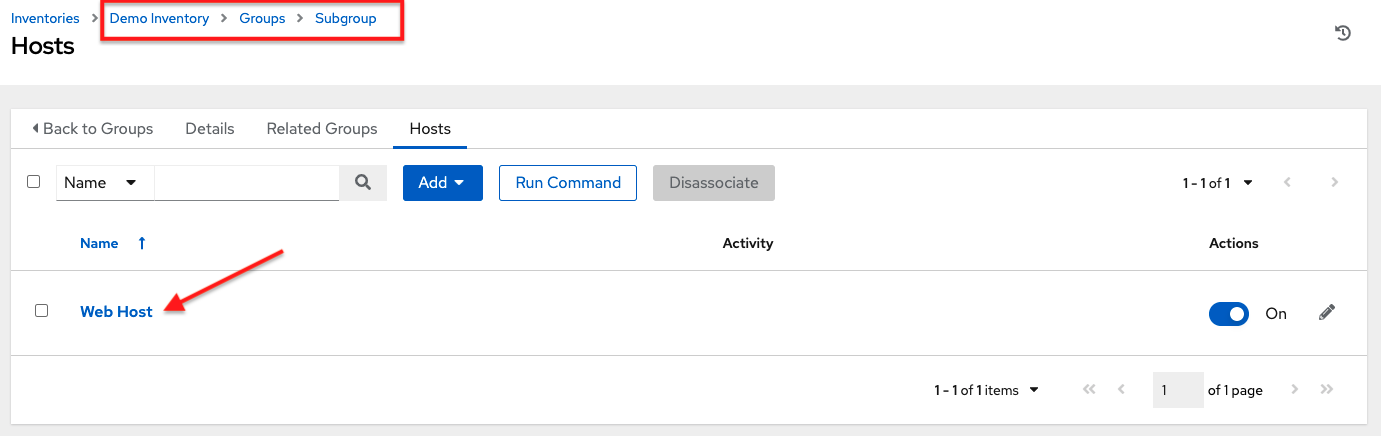
You can configure hosts for the inventory as well as for groups and groups within groups. To configure hosts:

1. Click the **Hosts** tab.
2. Click the **Add** button, and select whether to add a host that already exists in your configuration or create a new host.
3. If creating a new host, select the toggle button button to specify whether or not to include this host while running jobs.
4. Enter the appropriate details into the required and optional fields:

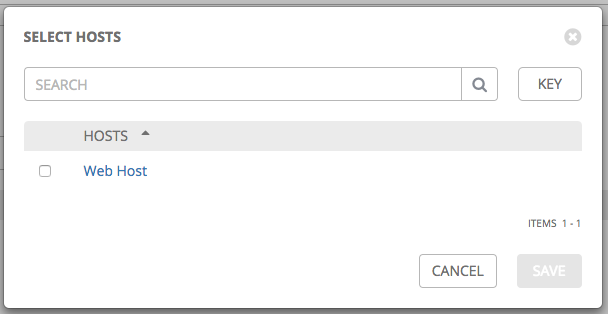
* **Host Name**: Required
* **Description**: Enter an arbitrary description as appropriate (optional)
* **Variables**: Enter definitions and values to be applied to all hosts in this group. Enter variables using either JSON or YAML syntax. Use the radio button to toggle between the two.

1. When done, click **Save**.

The **Create Host** window closes and the newly created host displays as an entry in the list of hosts associated with the group that it was created for.



If you chose to add an existing host, available hosts will appear in a separate selection window.

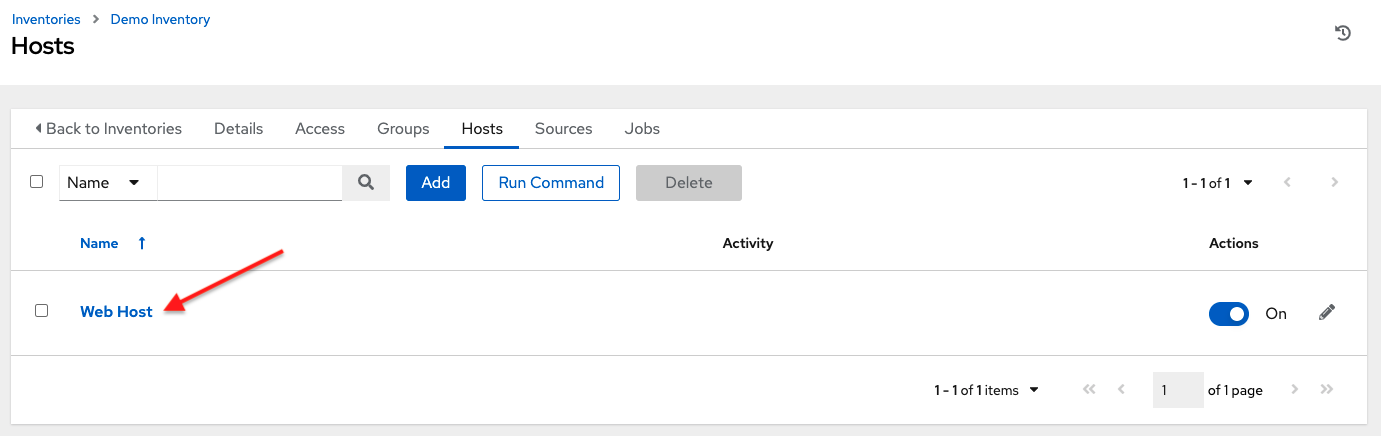


Once a host is selected, it displays as an entry in the list of hosts associated with the group. You can disassociate a host from this screen by selecting the host and click the **Disassociate** button.

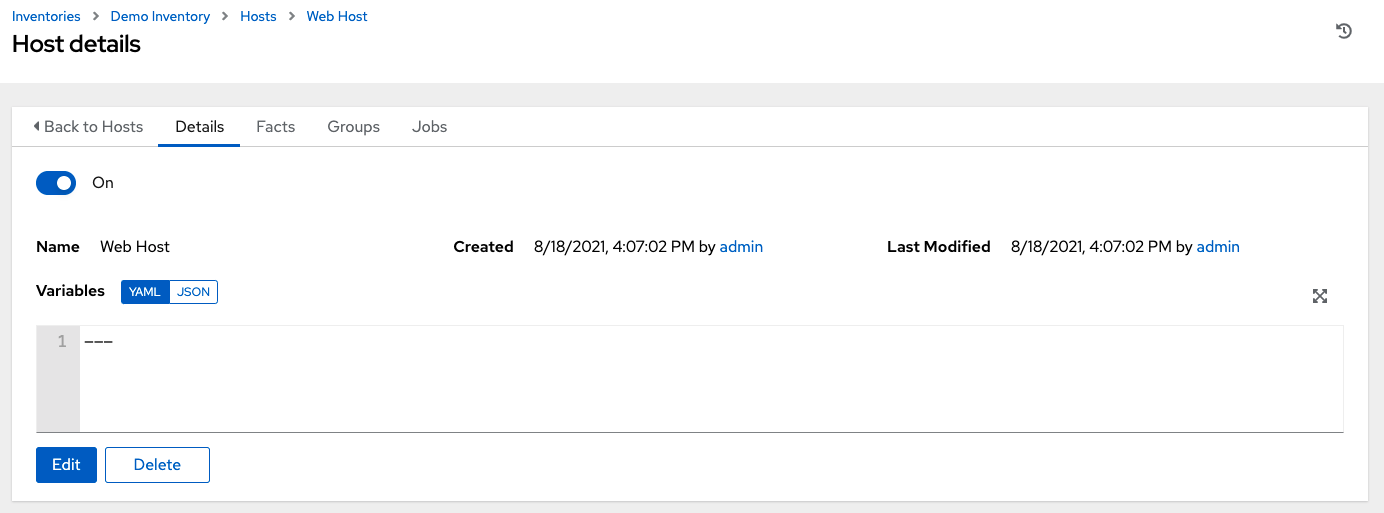
**Note**

You may also run ad hoc commands from this screen. Refer to Running Ad Hoc Commands for more detail.

6. To configure additional groups for the host, click on the name of the host from the list of hosts.

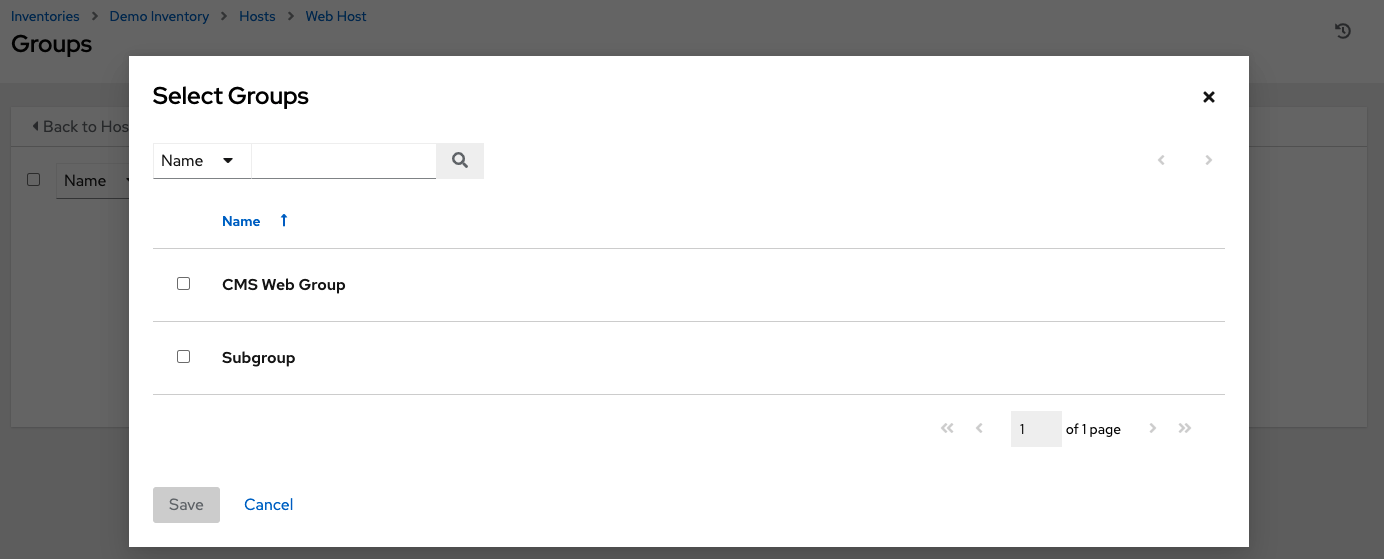


This opens the Details tab of the selected host.



1. Click the **Groups** tab to configure groups for the host.
2. Click the **Add** button to associate the host with an existing group.

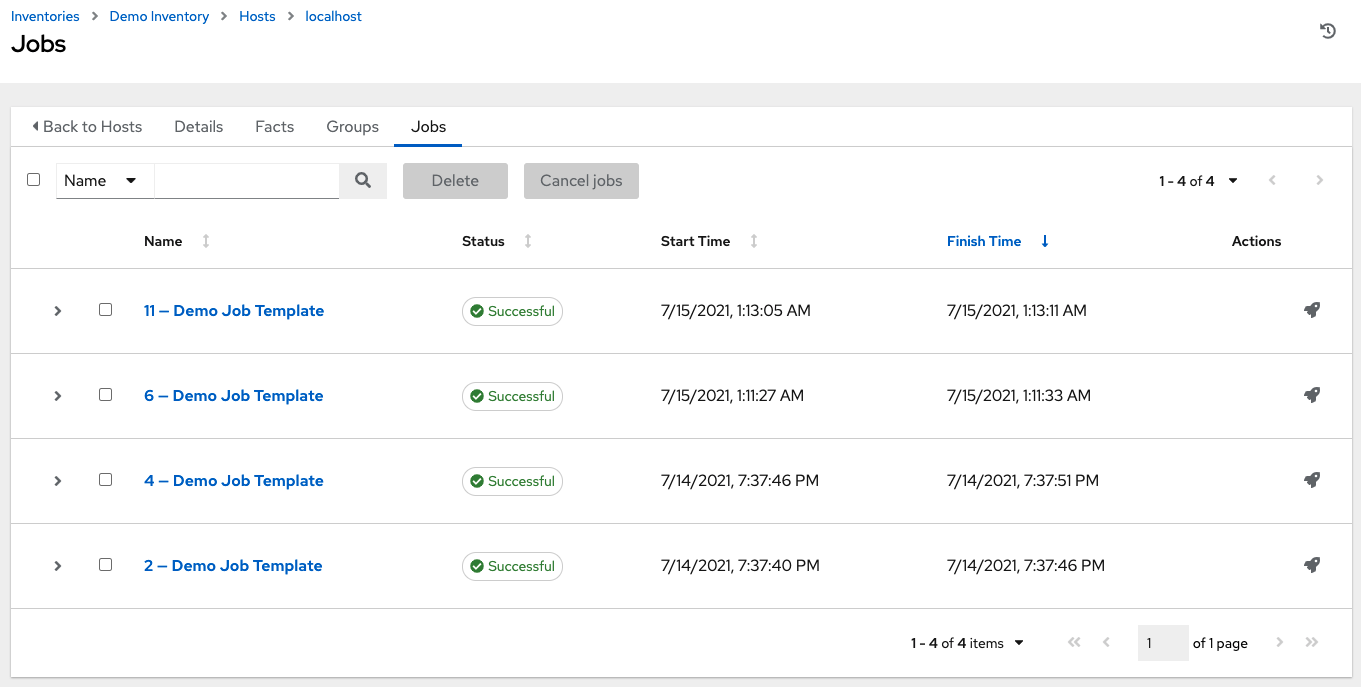
Available groups appear in a separate selection window.



1. Click to select the group(s) to associate with the host and click **Save**.

Once a group is associated, it displays as an entry in the list of groups associated with the host.

1. If a host was used to run a job, you can view details about those jobs in the **Completed Jobs** tab of the host and click **Expanded** to view details about each job.

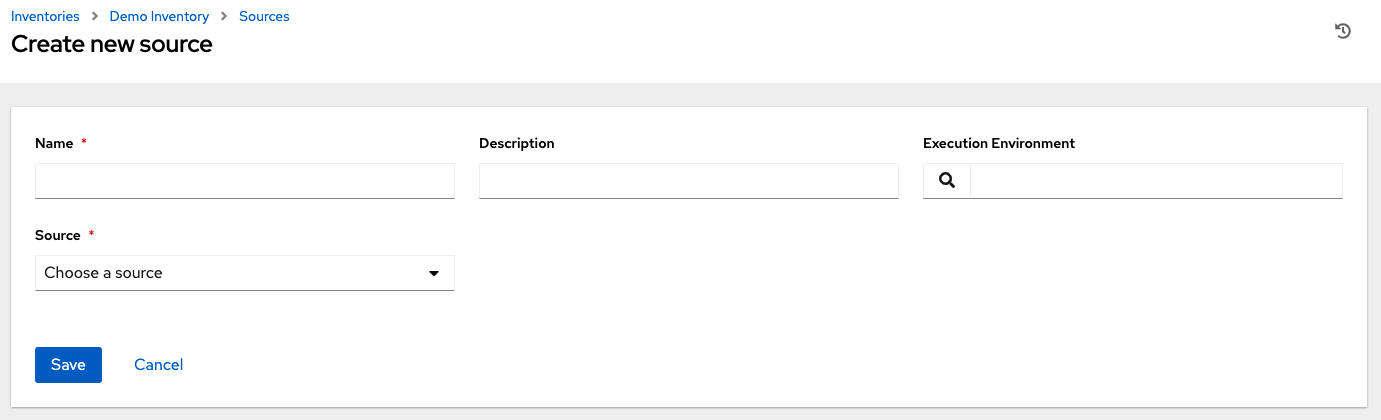


**18.3.4. Add source**

Inventory sources are not associated with groups. Spawned groups are top-level and may still have child groups, and all of these spawned groups may have hosts. Adding a source to an inventory only applies to standard inventories. Smart inventories inherit their source from the standard inventories they are associated with. To configure the source for the inventory:

1. In the inventory you want to add a source, click the **Sources** tab.
2. Click the **Add** button.

This opens the Create Source window.



1. Enter the appropriate details into the required and optional fields:

* **Name**: Required
* **Description**: Enter an arbitrary description as appropriate (optional)
* **Execution Environment**: Optionally search (search) or enter the name of the execution environment with which you want to run your inventory imports. Refer to the Execution Environments section for details on building an execution environment.
* **Source**: Choose a source for your inventory. Refer to the Inventory Sources section for more information about each source and details for entering the appropriate information.

1. After completing the required information for your chosen inventory source, you can continue to optionally specify other common parameters, such as verbosity, host filters, and variables.

**Note**

The **Regions**, **Instance Filters**, and **Only Group By** fields have been removed in automation controller 3.8.

1. Select the appropriate level of output on any inventory source’s update jobs from the **Verbosity** drop-down menu.
2. Use the **Host Filter** field to specify only matching host names to be imported into the controller.
3. In the **Enabled Variable**, specify the controller to retrieve the enabled state from the given dictionary of host variables. The enabled variable may be specified using dot notation as ‘foo.bar’, in which case the lookup will traverse into nested dicts, equivalent to: from\_dict.get('foo', {}).get('bar', default).
4. If you specified a dictionary of host variables in the **Enabled Variable** field, you can provide a value to enable on import. For example, if enabled\_var='status.power\_state' and enabled\_value='powered\_on' with the following host variables, the host would be marked enabled:

{

"status": {

"power\_state": "powered\_on",

"created": "2020-08-04T18:13:04+00:00",

"healthy": true

},

"name": "foobar",

"ip\_address": "192.168.2.1"

}

If power\_state were any value other than powered\_on, then the host would be disabled when imported into the controller. If the key is not found, then the host will be enabled.

1. All cloud inventory sources have the following update options:

* **Overwrite**: If checked, any hosts and groups that were previously present on the external source but are now removed, will be removed from the controller inventory. Hosts and groups that were not managed by the inventory source will be promoted to the next manually created group, or if there is no manually created group to promote them into, they will be left in the “all” default group for the inventory.

When not checked, local child hosts and groups not found on the external source will remain untouched by the inventory update process.

* **Overwrite Variables**: If checked, all variables for child groups and hosts will be removed and replaced by those found on the external source. When not checked, a merge will be performed, combining local variables with those found on the external source.
* **Update on Launch**: Each time a job runs using this inventory, refresh the inventory from the selected source before executing job tasks. To avoid job overflows if jobs are spawned faster than the inventory can sync, selecting this allows you to configure a **Cache Timeout** to cache prior inventory syncs for a certain number of seconds.

The “Update on Launch” setting refers to a dependency system for projects and inventory, and it will not specifically exclude two jobs from running at the same time. If a cache timeout is specified, then the dependencies for the second job is created and it uses the project and inventory update that the first job spawned. Both jobs then wait for that project and/or inventory update to finish before proceeding. If they are different job templates, they can then both start and run at the same time, if the system has the capacity to do so. If you intend to use the controller’s provisioning callback feature with a dynamic inventory source, **Update on Launch** should be set for the inventory group.

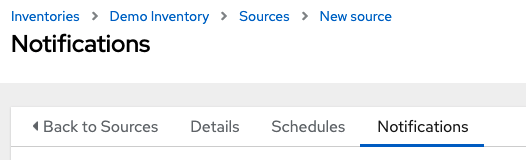
If you sync an inventory source that uses a project that has **Update On Launch** set, then the project may automatically update (according to cache timeout rules) before the inventory update starts.

You can create a job template that uses an inventory that sources from the same project that the template uses. In this case, the project will update and then the inventory will update (if updates are not already in-progress, or if the cache timeout has not already expired).

1. Review your entries and selections and click **Save** when done. This allows you to configure additional details, such as schedules and notifications.
2. To configure schedules associated with this inventory source, click the **Schedules** tab.
3. If schedules are already set up; review, edit, or enable/disable your schedule preferences.
4. if schedules have not been set up, refer to Schedules for more information.

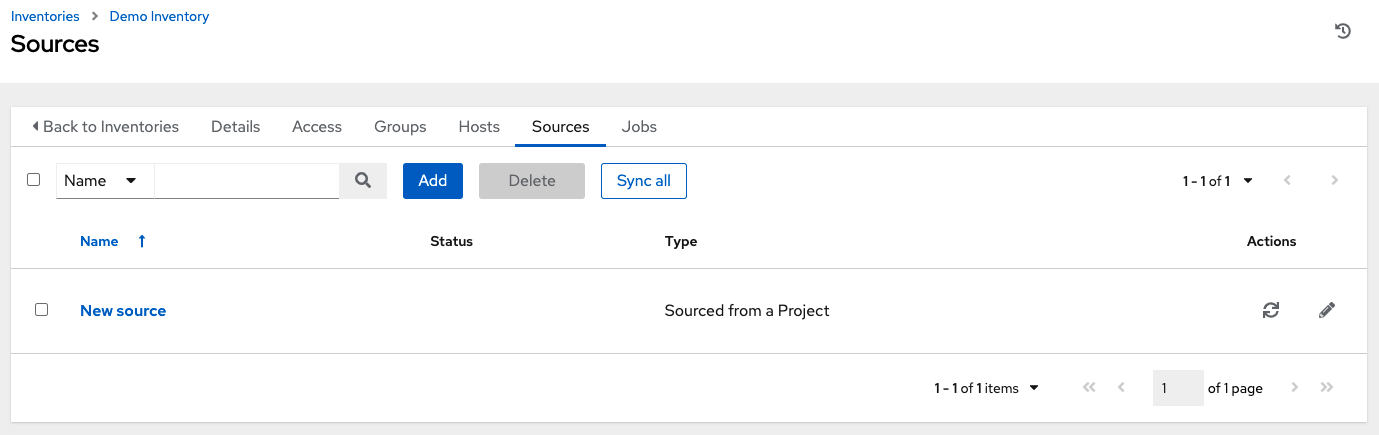
**Note**

The **Notifications** tab is only present after you save the newly-created source.



1. To configure notifications for the source, click the **Notifications** tab.
2. If notifications are already set up, use the toggles to enable or disable the notifications to use with your particular source. For more detail, see Enable and Disable Notifications.
3. if notifications have not been set up, refer to Notifications for more information.
4. Review your entries and selections and click **Save** when done.

Once a source is defined, it displays as an entry in the list of sources associated with the inventory. From the **Sources** tab you can perform a sync on a single source, or sync all of them at once. You can also perform additional actions such as scheduling a sync process, and edit or delete the source.



**18.3.4.1. Inventory Sources**

Choose a source which matches the inventory type against which a host can be entered:

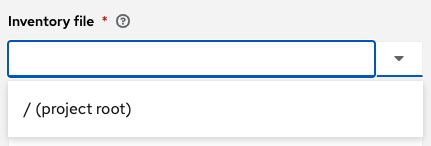
* Sourced from a Project
* Amazon Web Services EC2
* Google Compute Engine
* Microsoft Azure Resource Manager
* VMware vCenter
* Red Hat Satellite 6
* Red Hat Insights
* OpenStack
* Red Hat Virtualization
* Red Hat Ansible Automation Platform

**18.3.4.1.1. Sourced from a Project**

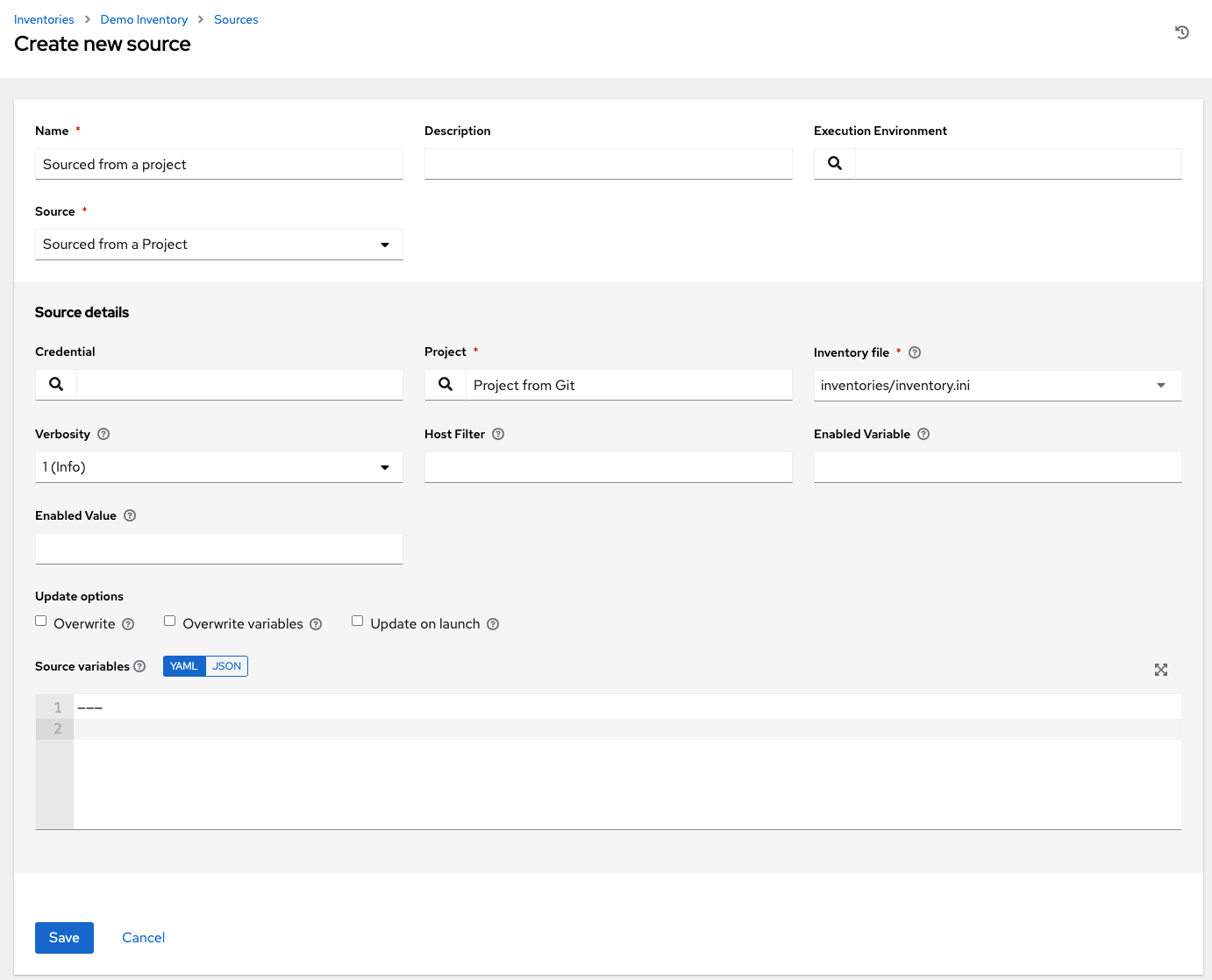
An inventory that is sourced from a project means that is uses the SCM type from the project it is tied to. For example, if the project’s source is from GitHub, then the inventory will use the same source.

1. To configure a project-sourced inventory, select **Sourced from a Project** from the Source field.
2. The Create Source window expands with additional fields. Enter the following details:

* **Credential**: Optionally specify the credential to use for this source.
* **Project**: Required. Specify the project this inventory is using as its source. Click the search button to choose from a list of projects. If the list is extensive, use the search to narrow the options.
* **Inventory File**: Required. Select an inventory file associated with the sourced project. If not already populated, you can type it into the text field within the drop down menu to filter the extraneous file types. In addition to a flat file inventory, you can point to a directory or an inventory script.



1. You can optionally specify the verbosity, host filter, enabled variable/value, and update options as described in the main procedure for adding a source.
2. In order to pass to the custom inventory script, you can optionally set environment variables in the **Environment Variables** field. You may also place inventory scripts in source control and then run it from a project. See Inventory File Importing in the *Automation Controller Administration Guide* for detail.



**Note**

If you are executing a custom inventory script from SCM, please make sure you set the execution bit (i.e. chmod +x) on the script in your upstream source control. If you do not, the controller will throw a [Errno 13] Permission denied error upon execution.

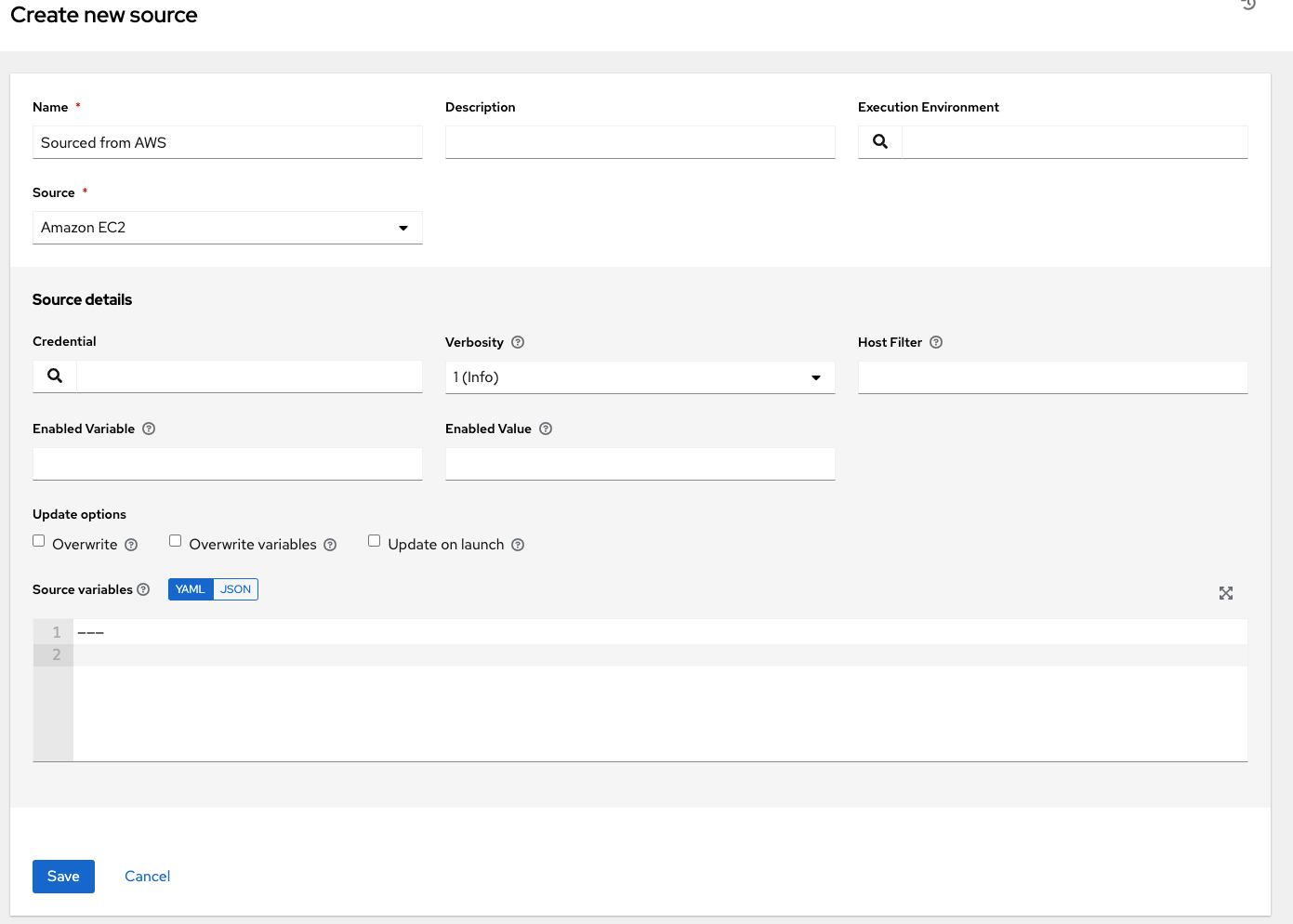
**18.3.4.1.2. Amazon Web Services EC2**

1. To configure an AWS EC2-sourced inventory, select **Amazon EC2** from the Source field.
2. The Create Source window expands with additional fields. Enter the following details:

* **Credential**: Optionally choose from an existing AWS credential (for more information, refer to Credentials).

If the controller is running on an EC2 instance with an assigned IAM Role, the credential may be omitted, and the security credentials from the instance metadata will be used instead. For more information on using IAM Roles, refer to the IAM\_Roles\_for\_Amazon\_EC2\_documentation\_at\_Amazon.

1. You can optionally specify the verbosity, host filter, enabled variable/value, and update options as described in the main procedure for adding a source.
2. Use the **Source Variables** field to override variables used by the aws\_ec2 inventory plugin. Enter variables using either JSON or YAML syntax. Use the radio button to toggle between the two. For a detailed description of these variables, view the aws\_ec2 inventory plugin documenation.

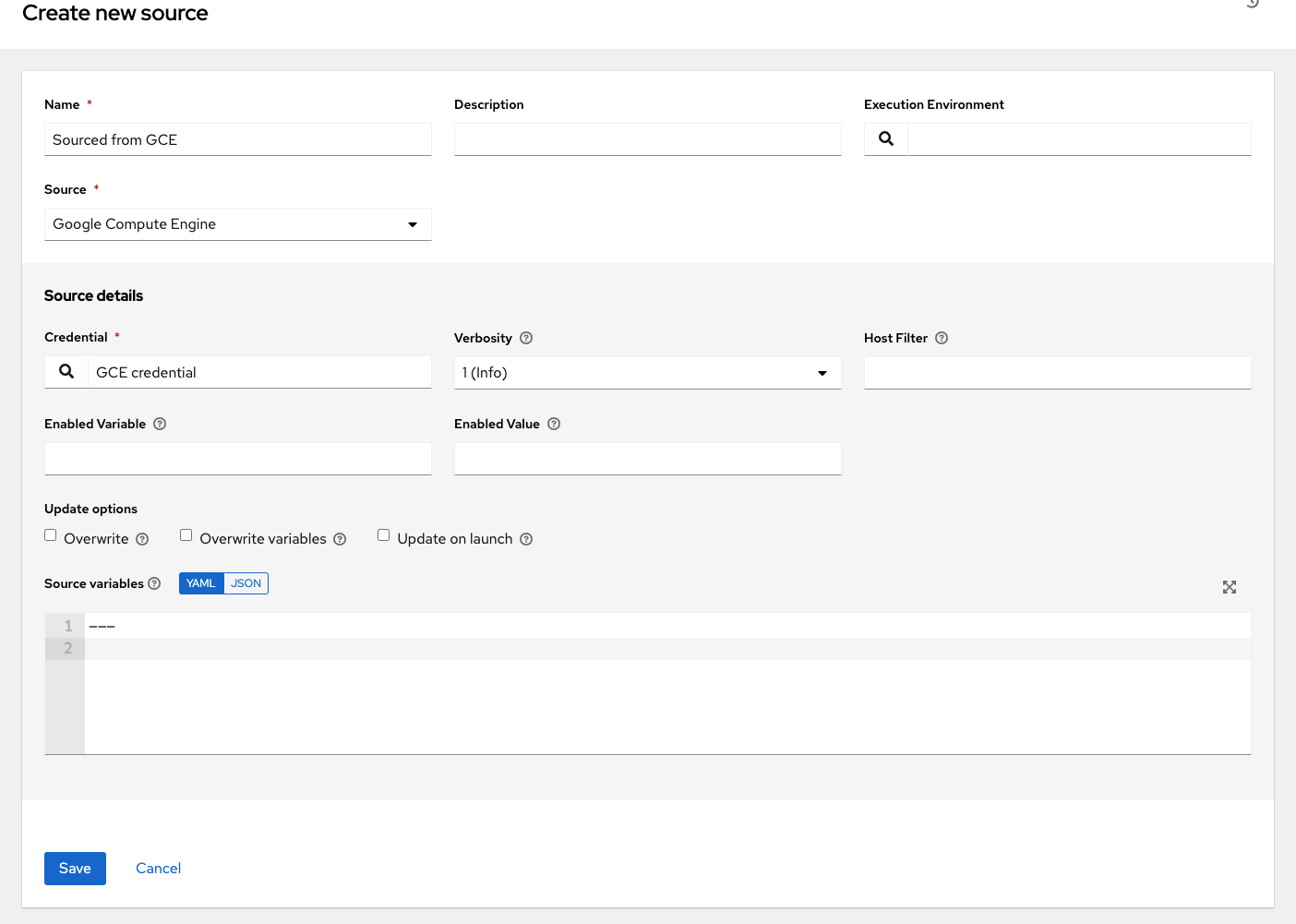


**Note**

If you only use include\_filters, the AWS plugin always returns all the hosts. To use this properly, the first condition on the or must be on filters and then build the rest of the OR conditions on a list of include\_filters.

**18.3.4.1.3. Google Compute Engine**

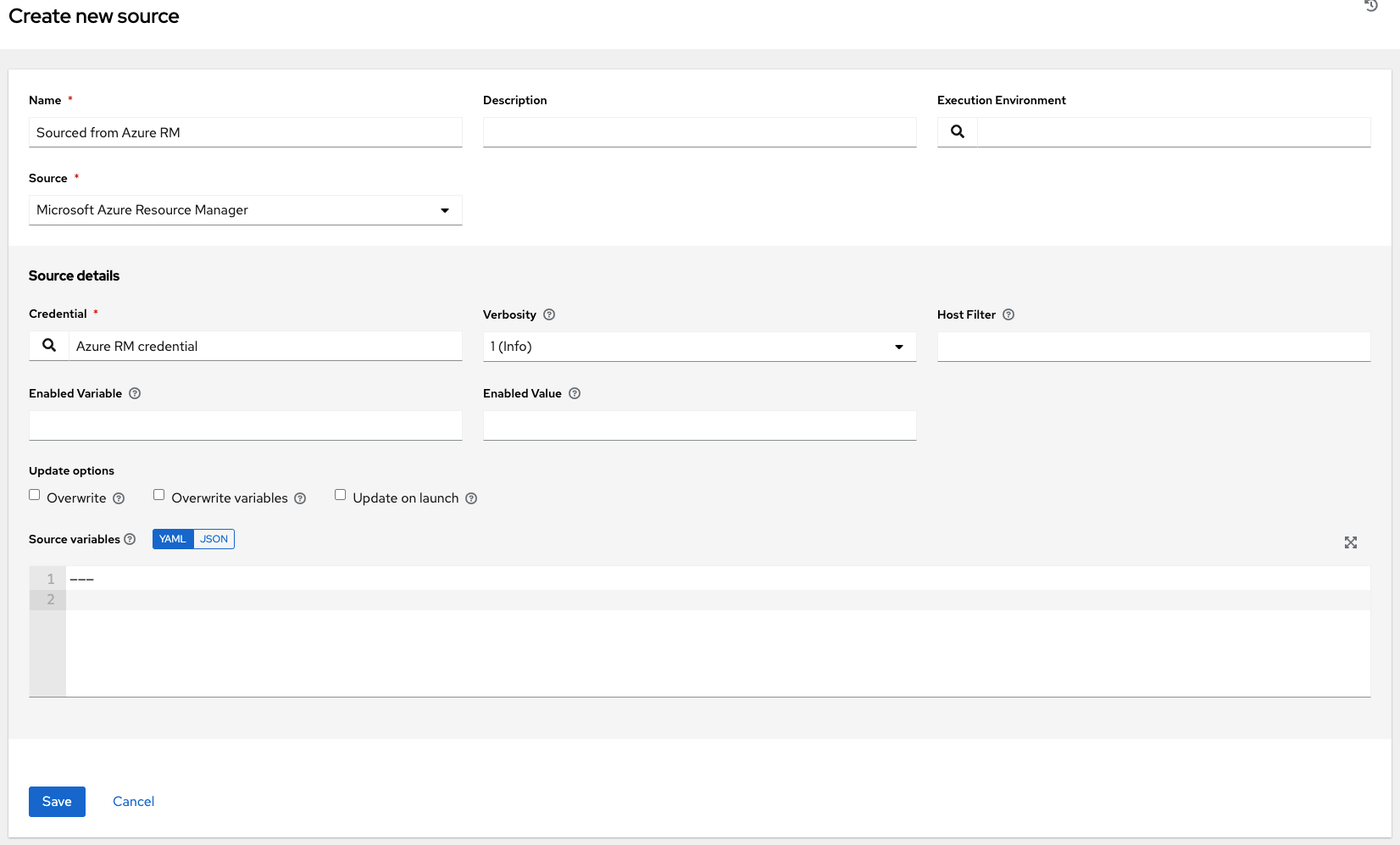
1. To configure a Google-sourced inventory, select **Google Compute Engine** from the Source field.
2. The Create Source window expands with the required **Credential** field. Choose from an existing GCE Credential. For more information, refer to Credentials.



1. You can optionally specify the verbosity, host filter, enabled variable/value, and update options as described in the main procedure for adding a source.
2. Use the **Source Variables** field to override variables used by the gcp\_compute inventory plugin. Enter variables using either JSON or YAML syntax. Use the radio button to toggle between the two. For a detailed description of these variables, view the gcp\_compute inventory plugin documenation.

**18.3.4.1.4. Microsoft Azure Resource Manager**

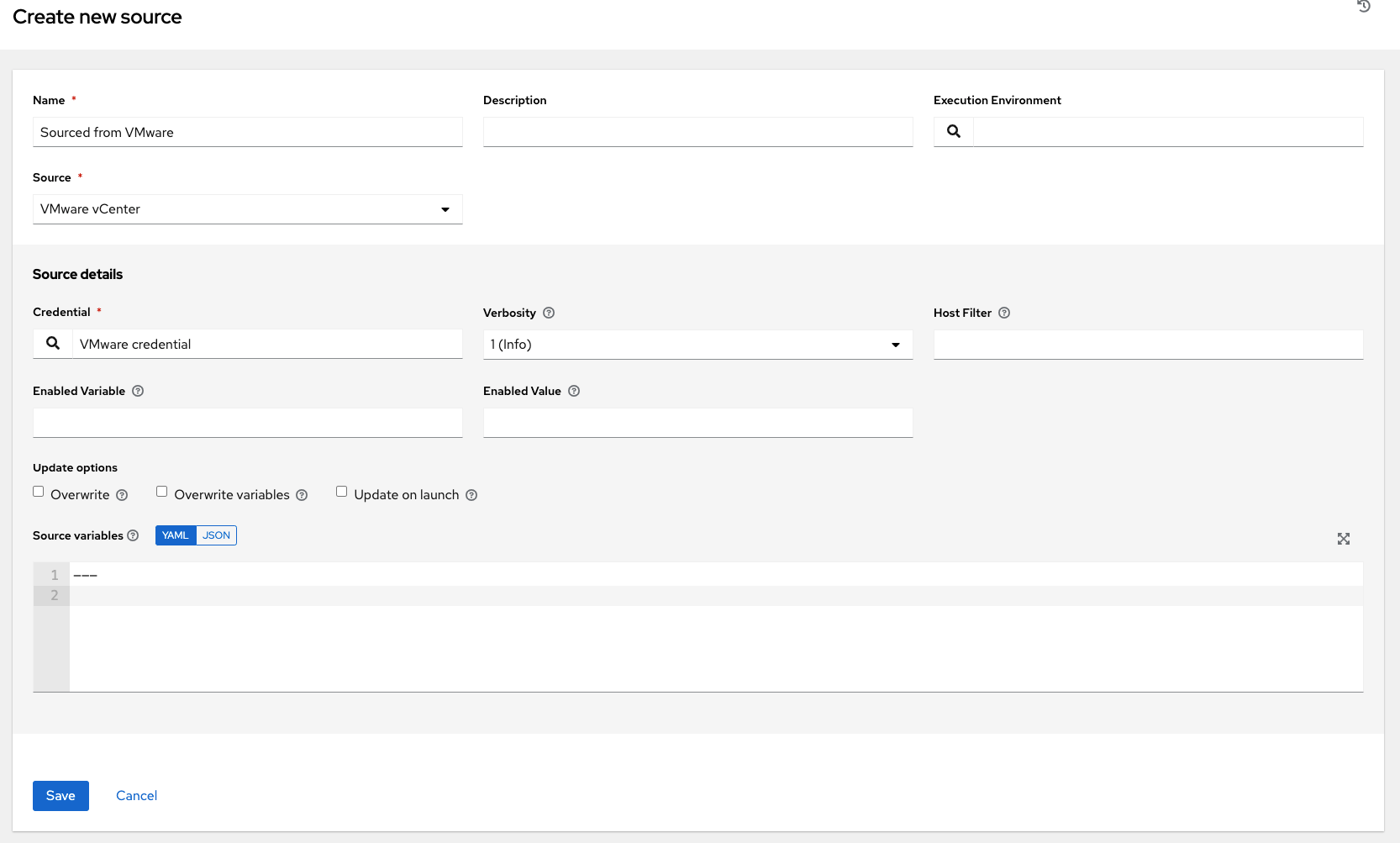
1. To configure a Azure Resource Manager-sourced inventory, select **Microsoft Azure Resource Manager** from the Source field.
2. The Create Source window expands with the required **Credential** field. Choose from an existing Azure Credential. For more information, refer to Credentials.
3. You can optionally specify the verbosity, host filter, enabled variable/value, and update options as described in the main procedure for adding a source.
4. Use the **Source Variables** field to override variables used by the azure\_rm inventory plugin. Enter variables using either JSON or YAML syntax. Use the radio button to toggle between the two. For a detailed description of these variables, view the azure\_rm inventory plugin documentation.



**18.3.4.1.5. VMware vCenter**

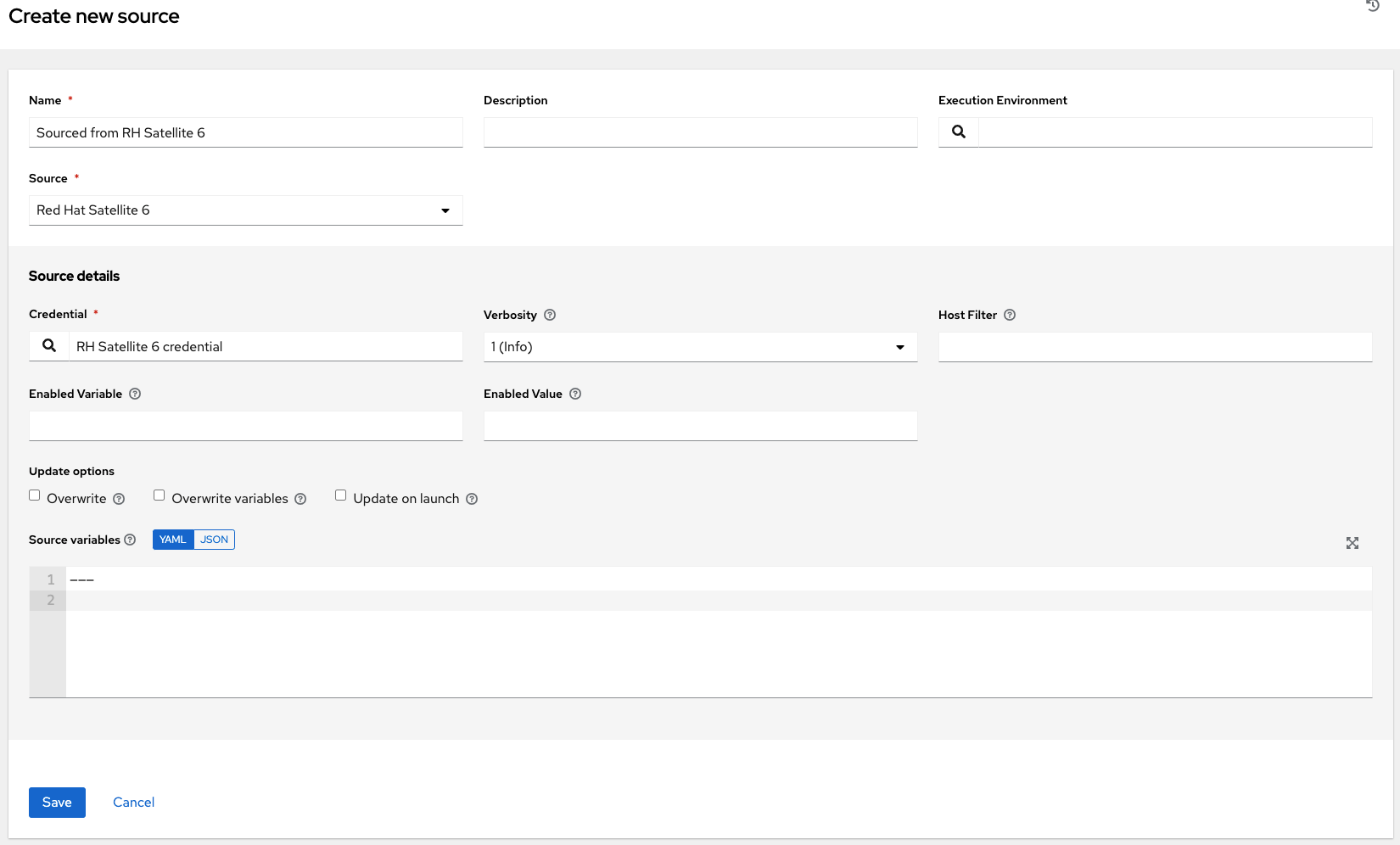
1. To configure a VMWare-sourced inventory, select **VMware vCenter** from the Source field.
2. The Create Source window expands with the required **Credential** field. Choose from an existing VMware Credential. For more information, refer to Credentials.
3. You can optionally specify the verbosity, host filter, enabled variable/value, and update options as described in the main procedure for adding a source.
4. Use the **Source Variables** field to override variables used by the vmware\_inventory inventory plugin. Enter variables using either JSON or YAML syntax. Use the radio button to toggle between the two. For a detailed description of these variables, view the vmware\_inventory inventory plugin.

Starting with Ansible 2.9, VMWare properties have changed from lower case to camelCase. The controller provides aliases for the top-level keys, but lower case keys in nested properties have been discontinued. For a list of valid and supported properties starting with Ansible 2.9, refer to virtual machine attributes in the VMware dynamic inventory plugin.



**18.3.4.1.6. Red Hat Satellite 6**

1. To configure a Red Hat Satellite-sourced inventory, select **Red Hat Satellite** from the Source field.
2. The Create Source window expands with the required **Credential** field. Choose from an existing Satellite Credential. For more information, refer to Credentials.
3. You can optionally specify the verbosity, host filter, enabled variable/value, and update options as described in the main procedure for adding a source.
4. Use the **Source Variables** field to specify parameters used by the foreman inventory source. Enter variables using either JSON or YAML syntax. Use the radio button to toggle between the two. For a detailed description of these variables, refer to the theforeman.foreman.foreman – Foreman inventory source in the Ansible documentation.

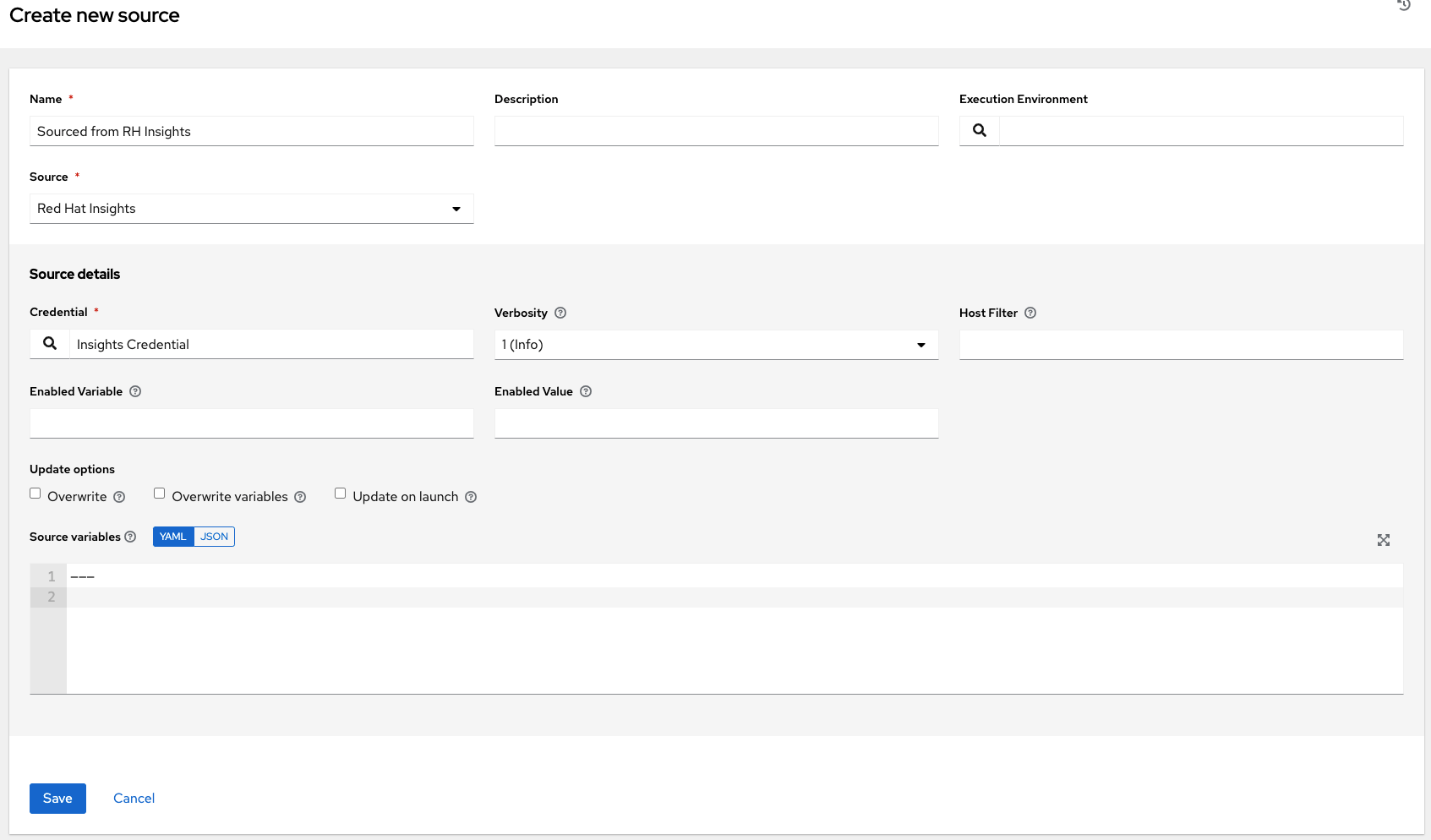


If you encounter an issue with the controller inventory not having the “related groups” from Satellite, you might need to define these variables in the inventory source. See the inventory plugins template example for Red Hat Satellite 6 in the *Ansible Automation Platform Installation and Reference Guide* for detail.

If you see the message, "no foreman.id" variable(s) when syncing the inventory, refer to the solution on the Red Hat Customer Portal at: https://access.redhat.com/solutions/5826451. Be sure to login with your customer credentials to access the full article.

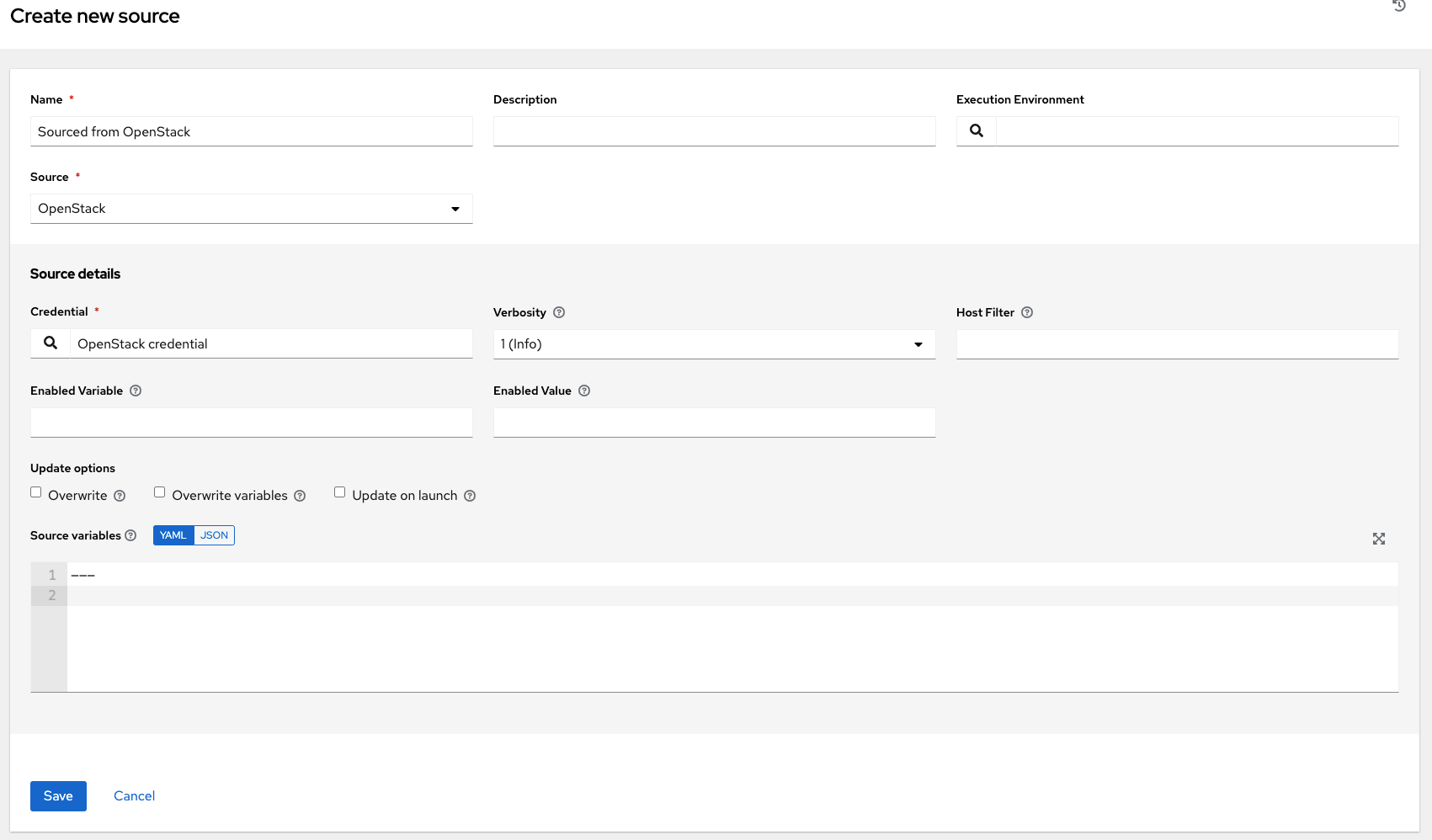
**18.3.4.1.7. Red Hat Insights**

1. To configure a Red Hat Insights-sourced inventory, select **Red Hat Insights** from the Source field.
2. The Create Source window expands with the required **Credential** field. Choose from an existing Insights Credential. For more information, refer to Credentials.
3. You can optionally specify the verbosity, host filter, enabled variable/value, and update options as described in the main procedure for adding a source.
4. Use the **Source Variables** field to override variables used by the insights inventory plugin. Enter variables using either JSON or YAML syntax. Use the radio button to toggle between the two. For a detailed description of these variables, view the insights inventory plugin.



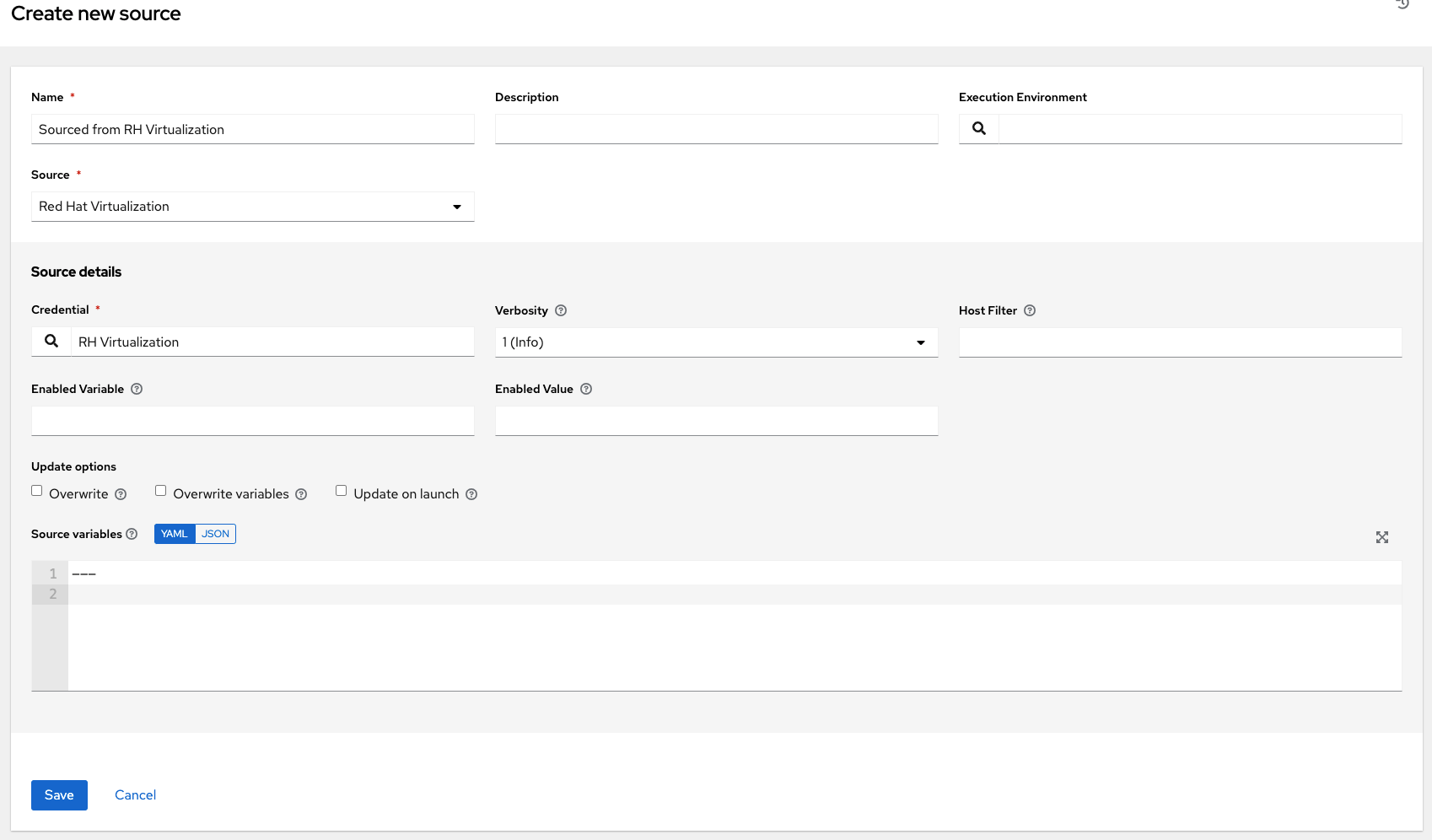
**18.3.4.1.8. OpenStack**

1. To configure an OpenStack-sourced inventory, select **OpenStack** from the Source field.
2. The Create Source window expands with the required **Credential** field. Choose from an existing OpenStack Credential. For more information, refer to Credentials.
3. You can optionally specify the verbosity, host filter, enabled variable/value, and update options as described in the main procedure for adding a source.
4. Use the **Source Variables** field to override variables used by the openstack inventory plugin. Enter variables using either JSON or YAML syntax. Use the radio button to toggle between the two. For a detailed description of these variables, view the openstack inventory plugin in the Ansible collections documentation.



**18.3.4.1.9. Red Hat Virtualization**

1. To configure a Red Hat Virtualization-sourced inventory, select **Red Hat Virtualization** from the Source field.
2. The Create Source window expands with the required **Credential** field. Choose from an existing Red Hat Virtualization Credential. For more information, refer to Credentials.
3. You can optionally specify the verbosity, host filter, enabled variable/value, and update options as described in the main procedure for adding a source.
4. Use the **Source Variables** field to override variables used by the ovirt inventory plugin. Enter variables using either JSON or YAML syntax. Use the radio button to toggle between the two. For a detailed description of these variables, view the ovirt inventory plugin.

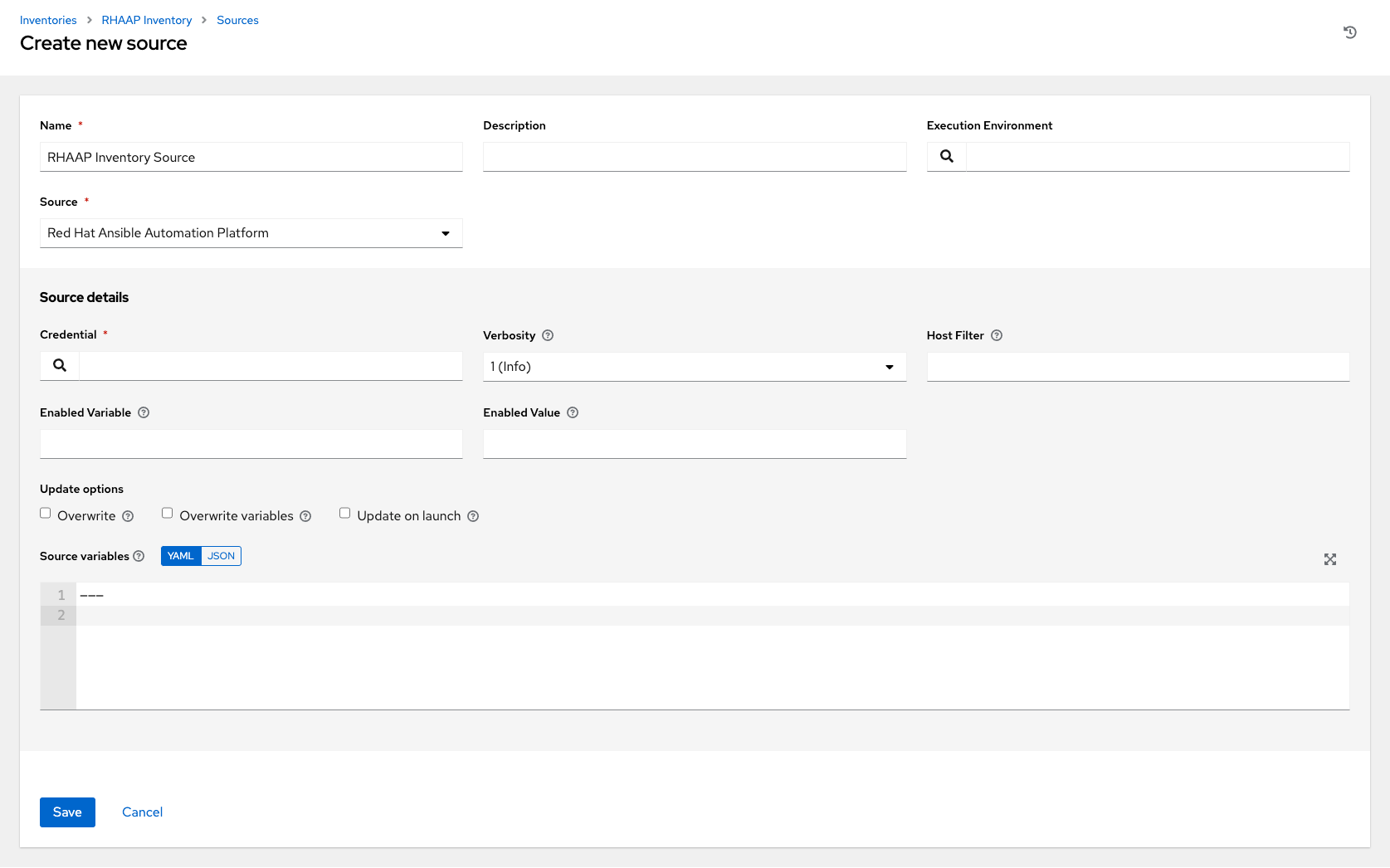


**Note**

Red Hat Virtualization (ovirt) inventory source requests are secure by default. To change this default setting, set the key ovirt\_insecure to **true** in source\_variables, which is only available from the API details of the inventory source at the /api/v2/inventory\_sources/N/ endpoint.

**18.3.4.1.10. Red Hat Ansible Automation Platform**

1. To configure a automation controller-sourced inventory, select **Red Hat Ansible Automation Platform** from the Source field.
2. The Create Source window expands with the required **Credential** field. Choose from an existing Ansible Automation Platform Credential. For more information, refer to Credentials.
3. You can optionally specify the verbosity, host filter, enabled variable/value, and update options as described in the main procedure for adding a source.



1. Use the **Source Variables** field to override variables used by the controller inventory plugin. Enter variables using either JSON or YAML syntax. Use the radio button to toggle between the two. For a detailed description of these variables, view the controller inventory plugin (requires your Red Hat Customer login).

**18.3.4.2. Export old inventory scripts**

Despite the removal of the custom inventory scripts API, the scripts are still saved in the database. The commands described in this section allows you to recover the scripts in a format that is suitable for you to subsequently check into source control. Usage looks like this:

$ awx-manage export\_custom\_scripts --filename=my\_scripts.tar

Dump of old custom inventory scripts at my\_scripts.tar

Making use of the output:

$ mkdir my\_scripts

$ tar -xf my\_scripts.tar -C my\_scripts

The naming of the scripts is \_<pk>\_\_<name>. This is the naming scheme used for project folders.

$ ls my\_scripts

\_10\_\_inventory\_script\_rawhook \_19\_\_ \_30\_\_inventory\_script\_listenhospital

\_11\_\_inventory\_script\_upperorder \_1\_\_inventory\_script\_commercialinternet45 \_4\_\_inventory\_script\_whitestring

\_12\_\_inventory\_script\_eastplant \_22\_\_inventory\_script\_pinexchange \_5\_\_inventory\_script\_literaturepossession

\_13\_\_inventory\_script\_governmentculture \_23\_\_inventory\_script\_brainluck \_6\_\_inventory\_script\_opportunitytelephone

\_14\_\_inventory\_script\_bottomguess \_25\_\_inventory\_script\_buyerleague \_7\_\_inventory\_script\_letjury

\_15\_\_inventory\_script\_wallisland \_26\_\_inventory\_script\_lifesport \_8\_\_random\_inventory\_script\_

\_16\_\_inventory\_script\_wallisland \_27\_\_inventory\_script\_exchangesomewhere \_9\_\_random\_inventory\_script\_

\_17\_\_inventory\_script\_bidstory \_28\_\_inventory\_script\_boxchild

\_18\_\_p \_29\_\_inventory\_script\_wearstress

Each file contains a script. Scripts can be bash/python/ruby/more, so the extension is not included. They are all directly executable (assuming the scripts worked). If you execute the script, it dumps the inventory data.

$ ./my\_scripts/\_11\_\_inventory\_script\_upperorder

{"group\_\ud801\udcb0\uc20e\u7b0e\ud81c\udfeb\ub12b\ub4d0\u9ac6\ud81e\udf07\u6ff9\uc17b": {"hosts":

["host\_\ud821\udcad\u68b6\u7a51\u93b4\u69cf\uc3c2\ud81f\uddbe\ud820\udc92\u3143\u62c7",

"host\_\u6057\u3985\u1f60\ufefb\u1b22\ubd2d\ua90c\ud81a\udc69\u1344\u9d15",

"host\_\u78a0\ud820\udef3\u925e\u69da\ua549\ud80c\ude7e\ud81e\udc91\ud808\uddd1\u57d6\ud801\ude57",

"host\_\ud83a\udc2d\ud7f7\ua18a\u779a\ud800\udf8b\u7903\ud820\udead\u4154\ud808\ude15\u9711",

"host\_\u18a1\u9d6f\u08ac\u74c2\u54e2\u740e\u5f02\ud81d\uddee\ufbd6\u4506"], "vars": {"ansible\_host": "127.0.0.1", "ansible\_connection":

"local"}}}

You can verify functionality with ansible-inventory. This should give the same data, but reformatted.

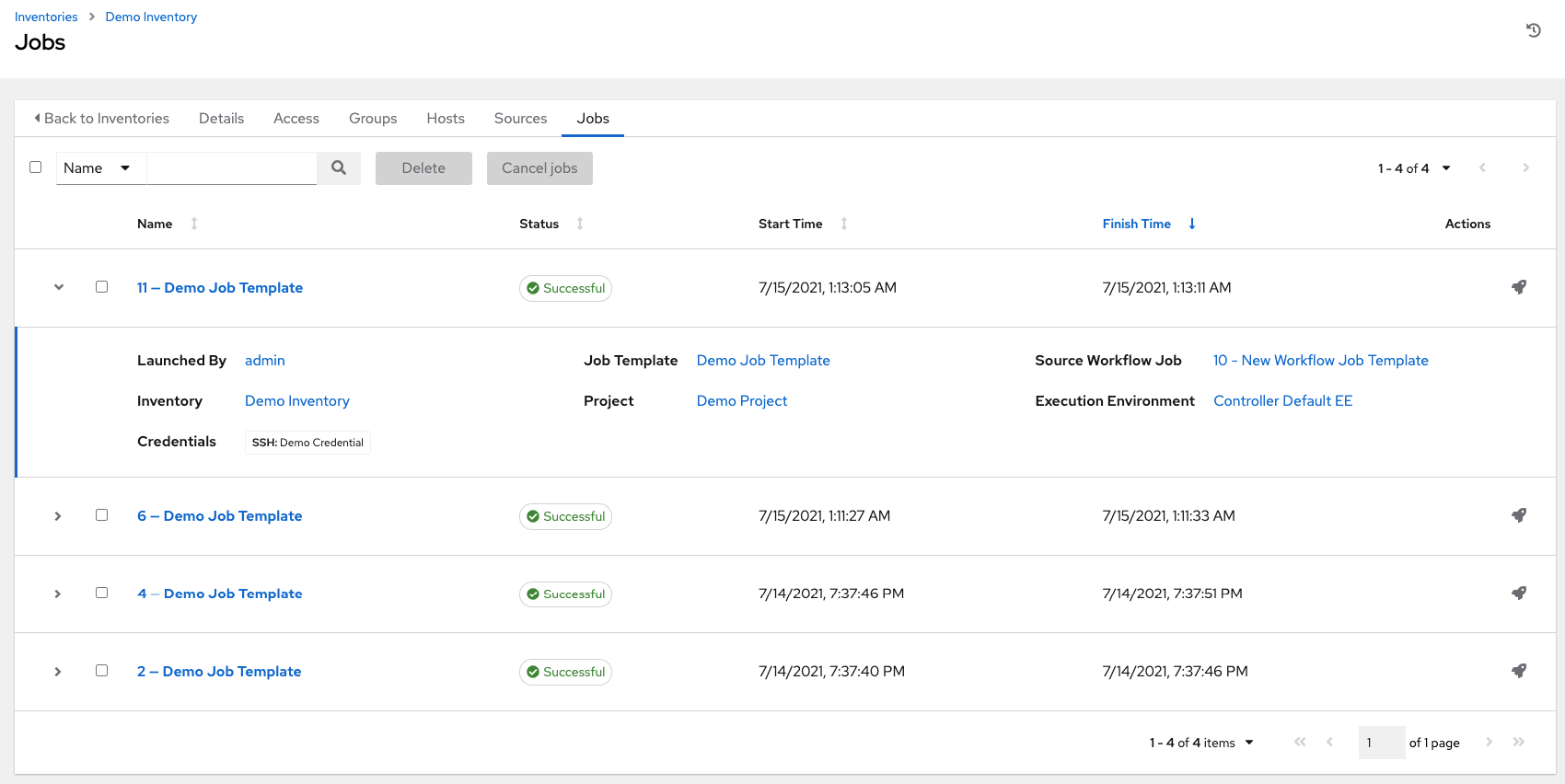
$ ansible-inventory -i ./my\_scripts/\_11\_\_inventory\_script\_upperorder --list --export

In the above example, you could cd into my\_scripts and then issue a git init command, add the scripts you want, push it to source control, and then create an SCM inventory source in the automation controller user interface.

For more information on syncing or using custom inventory scripts, refer to Inventory File Importing in the *Automation Controller Administration Guide*.

**18.3.5. View completed jobs**

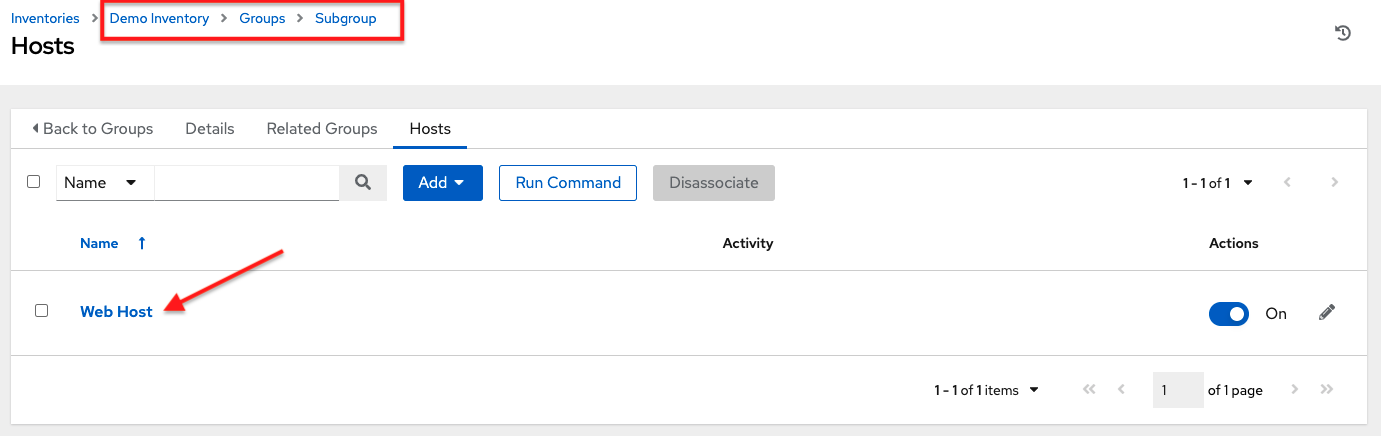
If an inventory was used to run a job, you can view details about those jobs in the **Completed Jobs** tab of the inventory and click **Expanded** to view details about each job.



**18.4. Running Ad Hoc Commands**

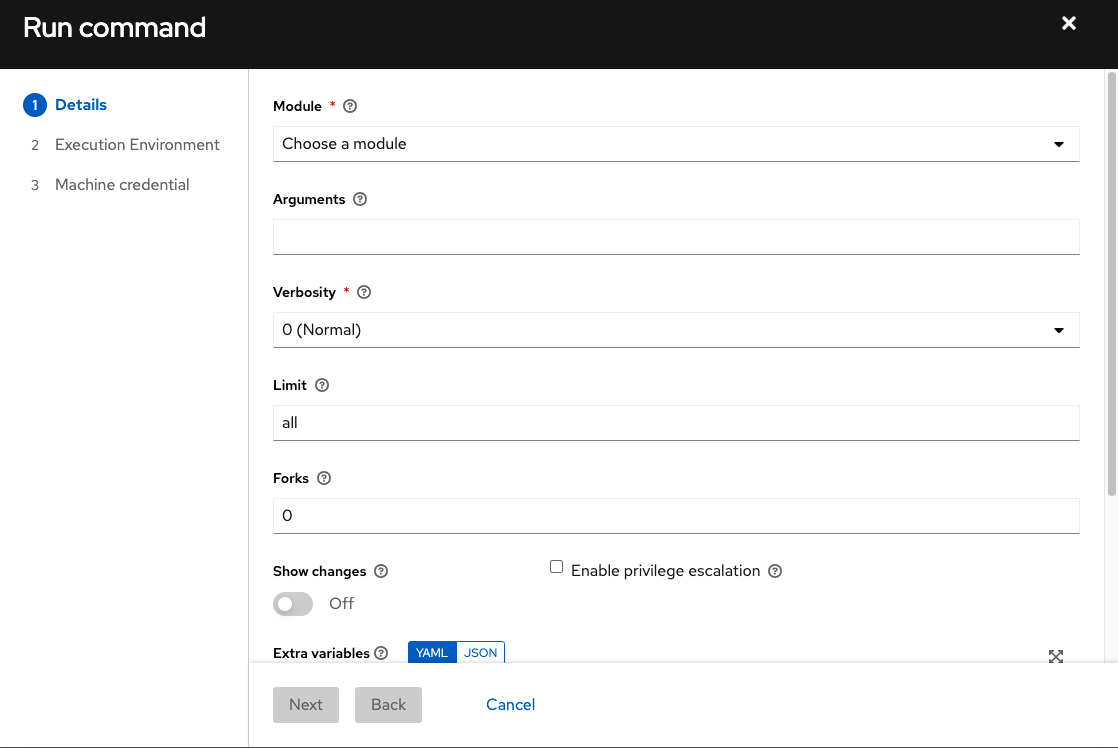
To run an ad hoc command:

1. Select an inventory source from the list of hosts or groups. The inventory source can be a single group or host, a selection of multiple hosts, or a selection of multiple groups.



1. Click the **Run Command** button.

The Run command window opens.

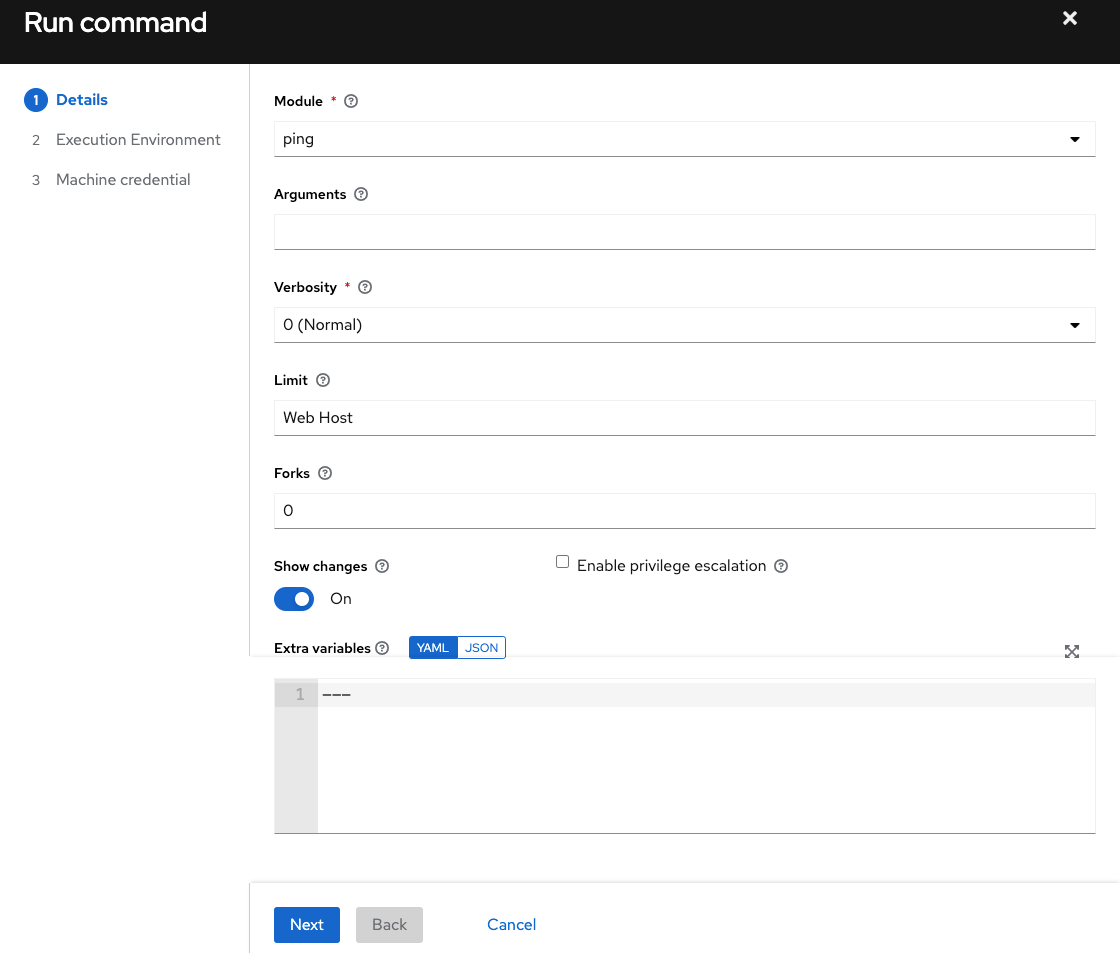


1. Enter the details for the following fields:

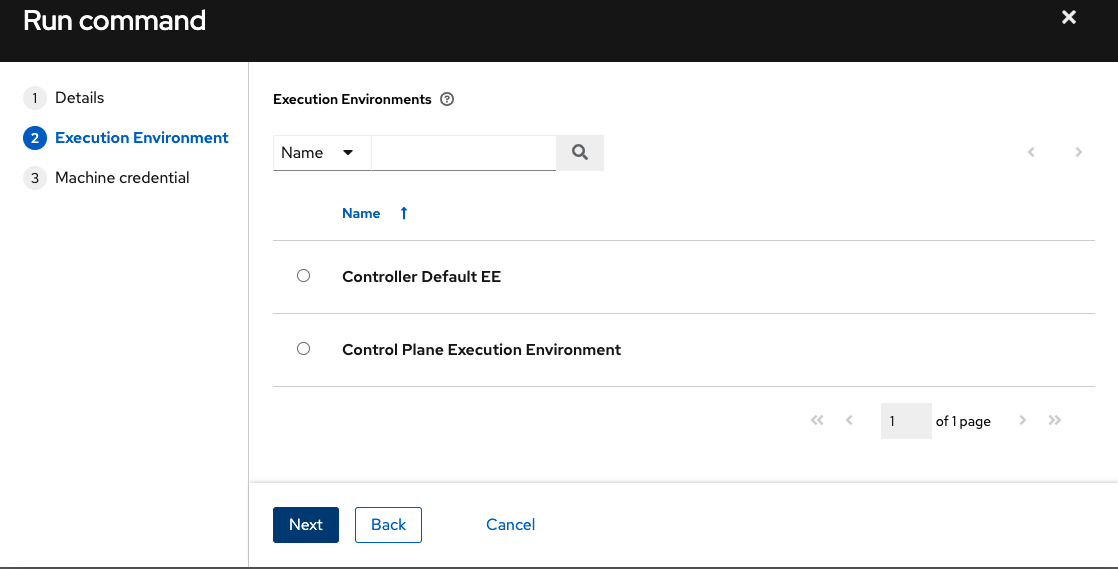
* **Module**: Select one of the modules that the automation controller supports running commands against.

|  |  |  |  |
| --- | --- | --- | --- |
| command | apt\_repository | mount | win\_service |
| shell | apt\_rpm | ping | win\_updates |
| yum | service | selinux | win\_group |
| apt | group | setup | win\_user |
| apt\_key | user | win\_ping |

* **Arguments**: Provide arguments to be used with the module you selected.
* **Limit**: Enter the limit used to target hosts in the inventory. To target all hosts in the inventory enter all or \*, or leave the field blank. This is automatically populated with whatever was selected in the previous view prior to clicking the launch button.
* **Machine Credential**: Select the credential to use when accessing the remote hosts to run the command. Choose the credential containing the username and SSH key or password that Ansbile needs to log into the remote hosts.
* **Verbosity**: Select a verbosity level for the standard output.
* **Forks**: If needed, select the number of parallel or simultaneous processes to use while executing the command.
* **Show Changes**: Select to enable the display of Ansible changes in the standard output. The default is OFF.
* **Enable Privilege Escalation**: If enabled, the playbook is run with administrator privileges. This is the equivalent of passing the --become option to the ansible command.
* **Extra Variables**: Provide extra command line variables to be applied when running this inventory. Enter variables using either JSON or YAML syntax. Use the radio button to toggle between the two.



1. Click **Next** to choose the execution environment you want the ad-hoc command to be run against.



1. Click **Next** to choose the credential you want to use and click the **Launch** button.

The results display in the **Output** tab of the module’s job window.

