Automation Controller User Guide

Release Automation Controller 4.3.0

Red Hat, Inc.

Dec 16, 2022

CONTENTS

Thank you for your interest in Red Hat Ansible Automation Platform controller. Automation controller helps teams manage complex multi-tier deployments by adding control, knowledge, and delegation to Ansible-powered environments.

The *Automation Controller User Guide* discusses all of the functionality available in automation controller and assumes moderate familiarity with Ansible, including concepts such as **Playbooks**, **Variables**, and **Tags**. For more information on these and other Ansible concepts, please see the Ansible documentation at http://docs.ansible.com/. This document has been updated to include information for the latest release of Automation Controller v4.3.0.

We Need Feedback!

If you spot a typo in this documentation, or if you have thought of a way to make this manual better, we would love to hear from you! Please send an email to: docs@ansible.com

If you have a suggestion, try to be as specific as possible when describing it. If you have found an error, please include the manual's title, chapter number/section number, and some of the surrounding text so we can find it easily. We may not be able to respond to every message sent to us, but you can be sure that we will be reading them all!

Automation Controller Version 4.3.0; November 29, 2022; https://access.redhat.com/

CHAPTER

ONE

OVERVIEW

Thank you for your interest in Red Hat Ansible Automation Platform. Ansible Automation Platform makes it possible for users across an organization to share, vet, and manage automation content by means of a simple, powerful, and agentless technical implementation. IT managers can provide guidelines on how automation is applied to individual teams. Meanwhile, automation developers retain the freedom to write tasks that use existing knowledge, without the operational overhead of conforming to complex tools and frameworks. It is a more secure and stable foundation for deploying end-to-end automation solutions, from hybrid cloud to the edge.

Ansible Automation Platform includes automation controller, which allows users to define, operate, scale, and delegate automation across their enterprise.

1.1 Real-time Playbook Output and Exploration

Watch playbooks run in real time, seeing each host as they check in. Easily go back and explore the results for specific tasks and hosts in great detail. Search for specific plays or hosts and see just those results, or quickly zero in on errors that need to be corrected.

1.2 "Push Button" Automation

Access your favorite projects and re-trigger execution from the web interface with a minimum of clicking. automation controller will ask for input variables, prompt for your credentials, kick off and monitor the job, and display results and host history over time.

1.3 Enhanced and Simplified Role-Based Access Control and Auditing

Automation controller allows for the granting of permissions to perform a specific task (such as to view, create, or modify a file) to different teams or explicit users through role-based access control (RBAC).

Keep some projects private, while allowing some users to edit inventory and others to run playbooks against only certain systems–either in check (dry run) or live mode. You can also allow certain users to use credentials without exposing the credentials to them. Regardless of what you do, automation controller records the history of operations and who made them–including objects edited and jobs launched.

Based on user feedback, automation controller both expands and simplifies its role-based access control. No longer is job template visibility configured via a combination of permissions on inventory, projects, and credentials. If you want to give any user or team permissions to use a job template, just assign permissions directly on the job template.

Similarly, credentials are now full objects in automation controller's RBAC system, and can be assigned to multiple users and/or teams for use.

Automation controller includes an 'Auditor' type, who can see all aspects of the systems automation, but has no permission to run or change automation, for those that need a system-level auditor. (This may also be useful for a service account that scrapes automation information from the REST API.) Refer to *Role-Based Access Controls* for more information.

Subsequent releases of automation controller provides more granular permissions, making it easier to delegate inside your organizations and remove automation bottlenecks.

1.4 Cloud & Autoscaling Flexibility

Automation controller features a powerful provisioning callback feature that allows nodes to request configuration on demand. While optional, this is an ideal solution for a cloud auto-scaling scenario, integrating with provisioning servers like Cobbler, or when dealing with managed systems with unpredictable uptimes. Requiring no management software to be installed on remote nodes, the callback solution can be triggered via a simple call to 'curl' or 'wget', and is easily embeddable in init scripts, kickstarts, or preseeds. Access is controlled such that only machines in inventory can request configuration.

1.5 The Ideal RESTful API

The automation controller REST API is the ideal RESTful API for a systems management application, with all resources fully discoverable, paginated, searchable, and well modeled. A styled API browser allows API exploration from the API root at `http://<server name>/api/`, showing off every resource and relation. Everything that can be done in the user interface can be done in the API - and more.

1.6 Backup and Restore

The ability to backup and restore your system(s) has been integrated into the Ansible Automation Platform setup playbook, making it easy for you to backup and replicate your instance as needed.

1.7 Ansible Galaxy Integration

When it comes to describing your automation, everyone repeats the DRY mantra–"Don't Repeat Yourself." Using centralized copies of Ansible roles, such as in Ansible Galaxy, allows you to bring that philosophy to your playbooks. By including an Ansible Galaxy requirements.yml file in your project directory, automation controller automatically fetches the roles your playbook needs from Galaxy, GitHub, or your local source control. Refer to *Ansible Galaxy Support* for more information.

1.8 Inventory Support for OpenStack

Ansible is committed to making OpenStack simple for everyone to use. As part of that, dynamic inventory support has been added for OpenStack. This allows you to easily target any of the virtual machines or images that you're running in your OpenStack cloud.

1.9 Remote Command Execution

Often times, you just need to do a simple task on a few hosts, whether it's add a single user, update a single security vulnerability, or restart a misbehaving service. Automation controller includes remote command execution–any task that you can describe as a single Ansible play can be run on a host or group of hosts in your inventory, allowing you to get managing your systems quickly and easily. Plus, it is all backed by an RBAC engine and detailed audit logging, removing any questions regarding who has done what to what machines.

1.10 System Tracking

You can collect facts by using the fact caching feature. Refer to *Fact Caching* for more detail.

1.11 Integrated Notifications

automation controller allows you to easily keep track of the status of your automation. You can configure stackable notifications for job templates, projects, or entire organizations, and configure different notifications for job start, job success, job failure, and job approval (for workflow nodes). The following notification sources are supported:

- Email
- Grafana
- IRC
- Mattermost
- PagerDuty
- Rocket.Chat
- Slack
- Twilio
- Webhook (post to an arbitrary webhook, for integration into other tools)

Additionally, you can *customize notification messages* for each of the above notification types.

1.12 Satellite Integration

Dynamic inventory sources for Red Hat Satellite 6 are supported.

1.13 Run-time Job Customization

Bringing the flexibility of the Ansible command line, you can now prompt for any of the following:

- inventory
- credential
- job tags
- limits

1.14 Red Hat Insights Integration

Automation controller supports integration with Red Hat Insights, which allows Insights playbooks to be used as a Ansible Automation Platform Project.

1.15 Enhanced User Interface

The layout of the user interface is organized with intuitive navigational elements. With information displayed at-a-glance, it is intuitive to find and use the automation you need. Compact and expanded viewing modes show and hide information as needed, and various built-in attributes make it easy to sort.

1.16 Custom Virtual Environments

Custom Ansible environment support allows you to have different Ansible environments and specify custom paths for different teams and jobs.

1.17 Authentication Enhancements

Automation controller supports LDAP, SAML, token-based authentication. Enhanced LDAP and SAML support allows you to integrate your enterprise account information in a more flexible manner. Token-based Authentication allows for easily authentication of third-party tools and services with automation controller via integrated OAuth 2 token support.

1.18 Cluster Management

Run-time management of cluster groups allows for easily configurable scaling.

1.19 Container Platform Support

Ansible Automation Platform is available as a containerized pod service for Red Hat OpenShift Container Platform that can be scaled up and down easily as needed.

1.20 Workflow Enhancements

In order to better model your complex provisioning, deployment, and orchestration workflows, automation controller expanded workflows in a number of ways:

- **Inventory overrides for Workflows**. You can now override an inventory across a workflow at workflow definition time, or even at launch time. Define your application deployment workflow, and then easily re-use them in multiple environments.
- **Convergence nodes for Workflows**. When modeling complex processes, you sometimes need to wait for multiple steps to finish before proceeding. Now automation controller workflows can easily replicate this; workflow steps can now wait for any number of prior workflow steps to complete properly before proceeding.
- **Workflow Nesting**. Re-use individual workflows as components of a larger workflow. Examples include combining provisioning and application deployment workflows into a single master workflow.
- **Workflow Pause and Approval**. You can build workflows containing approval nodes that require user intervention. This makes it possible to pause workflows in between playbooks so that a user can give approval (or denial) for continuing on to the next step in the workflow.

1.21 Job Distribution

As automation moves enterprise-wide, the need to automate at scale grows. Automation controller offer the ability to take a fact gathering or configuration job running across thousands of machines and slice it into individual job slices that can be distributed across your automation controller cluster for increased reliability, faster job completion, and better cluster utilization. If you need to change a parameter across 15,000 switches at scale, or gather information across your multi-thousand-node RHEL estate, you can now do so easily.

1.22 Support for deployment in a FIPS-enabled environment

If you require running your environment in restricted modes such as FIPS, automation controller deploys and runs in such environments.

1.23 Limit the number of hosts per organization

Lots of large organizations have instances shared among many organizations. They do not want any one organization to be able to use all the licensed hosts, this feature allows superusers to set a specified upper limit on how many licensed hosts may be allocated to each organization. The automation controller algorithm factors changes in the limit for an organization and the number of total hosts across all organizations. Any inventory updates will fail if an inventory sync brings an organization out of compliance with the policy. Additionally, superusers are able to 'over-allocate' their licenses, with a warning.

1.24 Inventory Plugins

Updated automation controller to use the following inventory plugins from upstream collections if inventory updates are run with Ansible 2.9:

- amazon.aws.aws_ec2
- community.vmware.vmware_vm_inventory
- azure.azcollection.azure_rm
- google.cloud.gcp_compute
- theforeman.foreman.foreman
- openstack.cloud.openstack
- ovirt.ovirt.ovirt
- awx.awx.tower

1.25 Secret Management System

With a secret management system, external credentials are stored and supplied for use in automation controller so you don't have to provide them directly.

1.26 Automation Hub Integration

Automation Hub will act as a content provider for automation controller, which requires both an automation controller deployment and an Automation Hub deployment running alongside each other.

CHAPTER

TWO

RED HAT ANSIBLE AUTOMATION PLATFORM CONTROLLER LICENSING, UPDATES, AND SUPPORT

Red Hat Ansible Automation Platform controller (“**Automation Controller**”) is a software product provided as part of an annual Red Hat Ansible Automation Platform subscription entered into between you and Red Hat, Inc. (“**Red Hat**”).

Ansible is an open source software project and is licensed under the GNU General Public License version 3, as detailed in the Ansible source code: https://github.com/ansible/ansible/blob/devel/COPYING

You **must** have valid subscriptions attached before installing the Ansible Automation Platform. See *Attaching Subscriptions* for detail.

2.1 Support

Red Hat offers support to paid Red Hat Ansible Automation Platform customers.

If you or your company has purchased a subscription for Ansible Automation Platform, you can contact the support team at https://access.redhat.com. To better understand the levels of support which match your Ansible Automation Platform subscription, refer to *Subscription Types*. For details of what is covered under an Ansible Automation Platform subscription, please see the Scopes of Support at: https://access.redhat.com/support/policy/updates/ansible-tower#scope-of-coverage-4 and https://access.redhat.com/support/policy/updates/ansible-engine.

2.2 Trial / Evaluation

While a license is required for automation controller to run, there is no fee for a trial license.

- Trial licenses for Red Hat Ansible Automation are available at: http://ansible.com/license
- Support is not included in a trial license or during an evaluation of the automation controller software.

2.3 Subscription Types

Red Hat Ansible Automation Platform is provided at various levels of support and number of machines as an annual Subscription.

- **Standard**
  - Manage any size environment
  - Enterprise 8x5 support and SLA

  - Maintenance and upgrades included
  - Review the SLA at: https://access.redhat.com/support/offerings/production/sla
  - Review the Red Hat Support Severity Level Definitions at: https://access.redhat.com/support/policy/severity

- **Premium**
  - Manage any size environment, including mission-critical environments
  - Premium 24x7 support and SLA
  - Maintenance and upgrades included
  - Review the SLA at: https://access.redhat.com/support/offerings/production/sla
  - Review the Red Hat Support Severity Level Definitions at: https://access.redhat.com/support/policy/severity

All Subscription levels include regular updates and releases of automation controller, Ansible, and any other components of the Platform.

For more information, contact Ansible via the Red Hat Customer portal at https://access.redhat.com/ or at http://www.ansible.com/contact-us/.

2.4 Node Counting in Licenses

The Red Hat Ansible Automation Platform controller license defines the number of Managed Nodes that can be managed as part of a Red Hat Ansible Automation Platform subscription. A typical license will say 'License Count: 500', which sets the maximum number of Managed Nodes at 500.

For more information on managed node requirements for licensing, please see https://access.redhat.com/articles/3331481.

2.5 Attaching Subscriptions

You **must** have valid subscriptions attached before installing the Ansible Automation Platform. Attaching an Ansible Automation Platform subscription enables Automation Hub repositories. A valid subscription needs to be attached to the Automation Hub node only. Other nodes do not need to have a valid subscription/pool attached, even if the **[automationhub]** group is blank, given this is done at the `repos_el` role level and that this role is run on both **[default]** and **[automationhub]** hosts.

Note: Attaching subscriptions is unnecessary if your Red Hat account enabled Simple Content Access Mode. But you still need to register to RHSM or Satellite before installing the Ansible Automation Platform.

To find out the `pool_id` of your Ansible Automation Platform subscription:

```
#subscription-manager list --available --all | grep "Ansible Automation Platform" -B 3 -A 6
```

The command returns the following:

```
Subscription Name: Red Hat Ansible Automation Platform, Premium (5000 Managed Nodes)
Provides: Red Hat Ansible Engine
Red Hat Single Sign-On
Red Hat Ansible Automation Platform
SKU: MCT3695
Contract: ********
Pool ID: ********************
Provides Management: No
Available: 4999
Suggested: 1
```

To attach this subscription:

```
#subscription-manager attach --pool=<pool_id>
```

If this is properly done, and all nodes have Red Hat Ansible Automation Platform attached, then it will find the Automation Hub repositories correctly.

To check whether the subscription was successfully attached:

```
#subscription-manager list --consumed
```

To remove this subscription:

```
#subscription-manager remove --pool=<pool_id>
```

2.6 Ansible Automation Platform Component Licenses

To view the license information for the components included within automation controller, refer to `/usr/share/doc/automation-controller-<version>/README` where `<version>` refers to the version of automation controller you have installed.

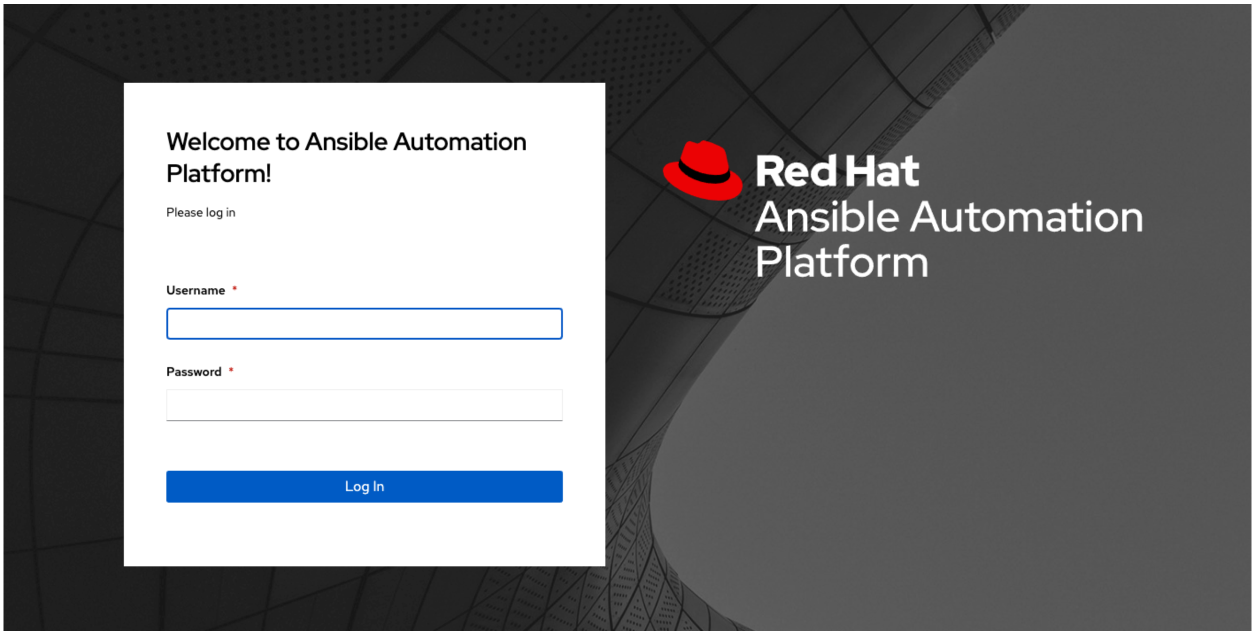
To view a specific license, refer to `/usr/share/doc/automation-controller-<version>/*.txt`, where `*` is replaced by the license file name to which you are referring.

CHAPTER

THREE

LOGGING IN

To log in, browse to the user interface at: `http://<server name>/`



Welcome to Ansible Automation Platform!

Please log in

Log in using a valid username and password.

The default username is ‘’admin”, and the default password is set during installation. To configure users and passwords, you can do so by accessing **Users** from the left navigation bar.

CHAPTER

FOUR

IMPORT A SUBSCRIPTION

Available subscriptions or a subscription manifest authorize the use of the automation controller. To obtain your automation controller subscription, you can either:

1. Provide your Red Hat or Satellite username and password on the license page.
2. Obtain a subscriptions manifest from your Subscription Allocations page on the customer portal. See *Obtaining a subscriptions manifest* in the *Automation Controller User Guide* for more detail.

If you **have** a Red Hat Ansible Automation Platform subscription, use your Red Hat customer credentials when you launch the controller to access your subscription information (see instructions below).

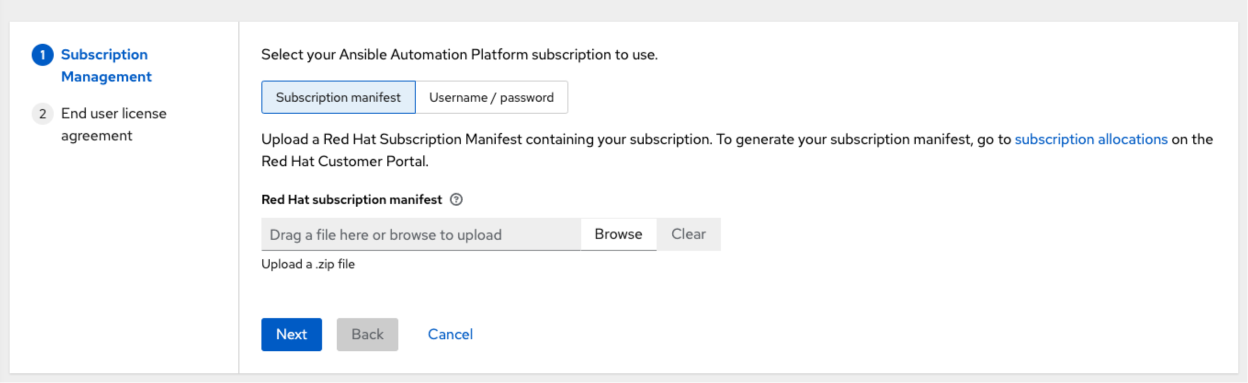
If you **do not** have a Red Hat Ansible Automation Platform subscription, you can request a trial subscription here or click **Request Subscription** and follow the instructions to request one.

Disconnected environments with Satellite will be able to use the login flow on vm-based installations if they have configured subscription manager on the controller instance to connect to their Satellite instance. Recommended workarounds for disconnected environments **without Satellite** include [1] downloading a manifest from access.redhat.com in a connected environment, then uploading it to the disconnected controller instance, or [2] connecting to the Internet through a proxy server.

Note: In order to use a disconnected environment, it is necessary to have a valid automation controller entitlement attached to your Satellite organization's manifest. This can be confirmed by using `hammer subscription list \--organization <org_name>`.

To understand what is supported with your subscription, see *Red Hat Ansible Automation Platform Controller Licensing, Updates, and Support* for more information. If you have issues with the subscription you have received, please contact your Sales Account Manager or Red Hat Customer Service at https://access.redhat.com/support/contact/customerService/.

When the controller launches for the first time, the Subscription screen automatically displays.



1 Subscription Management
2 End user license agreement

Select your Ansible Automation Platform subscription to use.

Subscription manifest
Username / password

Upload a Red Hat Subscription Manifest containing your subscription. To generate your subscription manifest, go to subscription allocations on the Red Hat Customer Portal.

Red Hat subscription manifest

Drag a file here or browse to upload
Browse
Clear

Upload a .zip file

1. By default, the option to retrieve and import your subscription is to upload a subscription manifest you generate from https://access.redhat.com/management/subscription_allocations. See *Obtaining a subscriptions manifest* for more detail. Once you have a subscription manifest, you can upload it by browsing to the location where the file is saved (the subscription manifest is the complete .zip file, not its component parts).

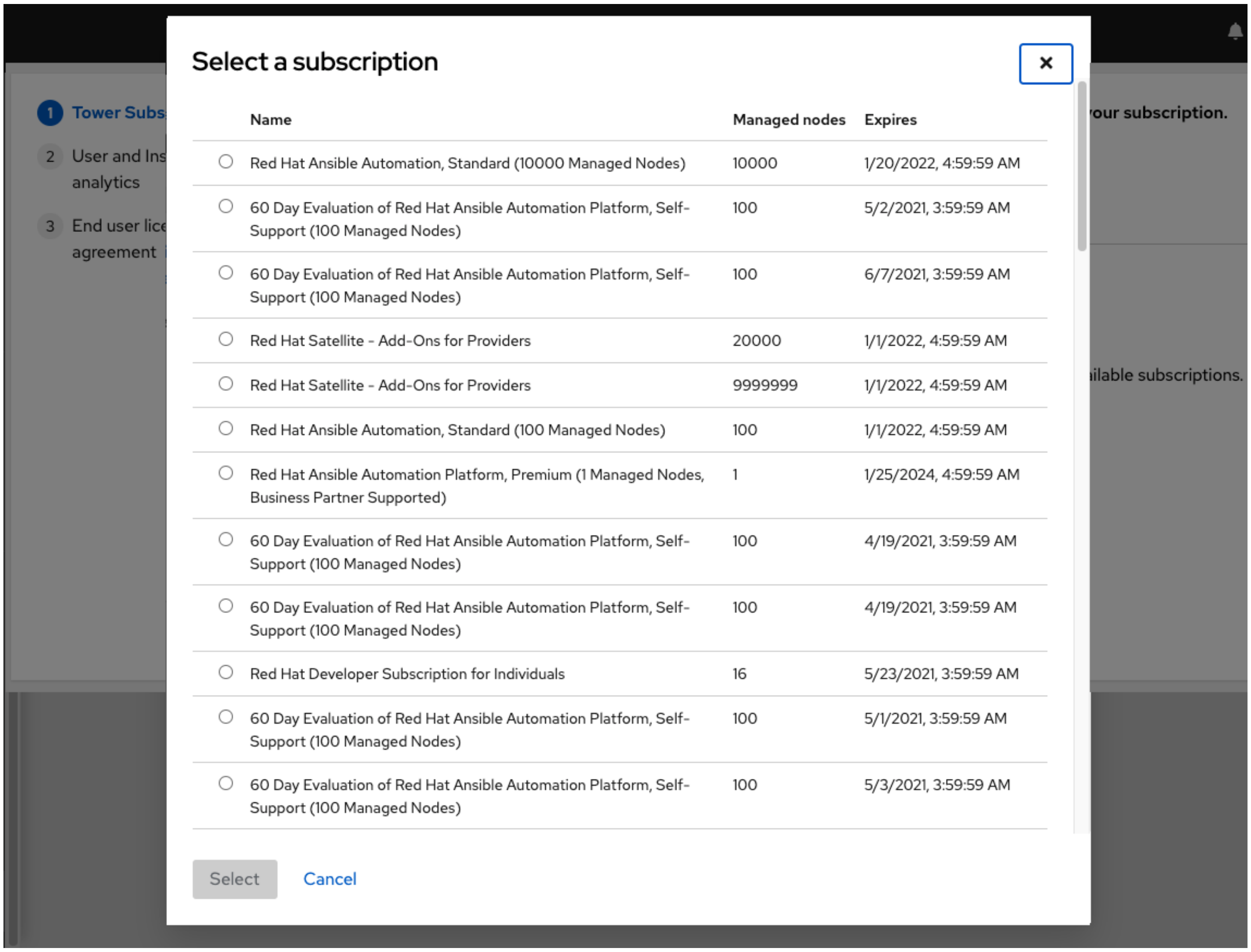
Note: If the **Browse** button in the subscription manifest option is grayed-out, clear the username and password fields to enable the **Browse** button.

Alternatively, you can choose the option to enter your Red Hat customer credentials using your username and password. Use your Satellite username/password if your controller cluster nodes are registered to Satellite via Subscription Manager. Once you entered your credentials, click **Get Subscriptions**.

2. The subscription metadata is then retrieved from the RHSM/Satellite API, or from the manifest provided.

- If it is a subscription manifest, and multiple subscription counts were applied in a single installation, the controller will combine the counts but use the earliest expiration date as the expiry (at which point you will need to refresh your subscription).
- If you entered your credential information (username/password), the controller retrieves your configured subscription service. Then it prompts you to choose the subscription you want to run (the example below shows multiple subscriptions) and entitles the controller with that metadata. You can log in over time and retrieve new subscriptions if you have renewed.

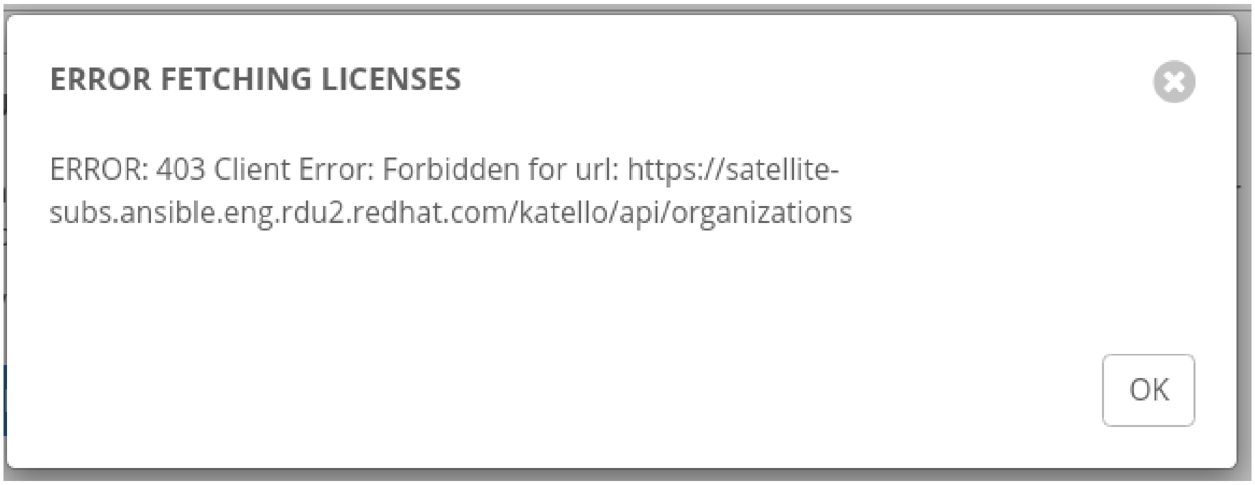
Note: When your subscription expires (you can check this in the Subscription details of the Subscription settings window), you will need to renew it in the controller by one of these two methods.



Select a subscription

	Name	Managed nodes	Expires
○	Red Hat Ansible Automation, Standard (10000 Managed Nodes)	10000	1/20/2022, 4:59:59 AM
○	60 Day Evaluation of Red Hat Ansible Automation Platform, Self-Support (100 Managed Nodes)	100	5/2/2021, 3:59:59 AM
○	60 Day Evaluation of Red Hat Ansible Automation Platform, Self-Support (100 Managed Nodes)	100	6/7/2021, 3:59:59 AM
○	Red Hat Satellite - Add-Ons for Providers	20000	1/1/2022, 4:59:59 AM
○	Red Hat Satellite - Add-Ons for Providers	9999999	1/1/2022, 4:59:59 AM
○	Red Hat Ansible Automation, Standard (100 Managed Nodes)	100	1/1/2022, 4:59:59 AM
○	Red Hat Ansible Automation Platform, Premium (1 Managed Nodes, Business Partner Supported)	1	1/25/2024, 4:59:59 AM
○	60 Day Evaluation of Red Hat Ansible Automation Platform, Self-Support (100 Managed Nodes)	100	4/19/2021, 3:59:59 AM
○	60 Day Evaluation of Red Hat Ansible Automation Platform, Self-Support (100 Managed Nodes)	100	4/19/2021, 3:59:59 AM
○	Red Hat Developer Subscription for Individuals	16	5/23/2021, 3:59:59 AM
○	60 Day Evaluation of Red Hat Ansible Automation Platform, Self-Support (100 Managed Nodes)	100	5/1/2021, 3:59:59 AM
○	60 Day Evaluation of Red Hat Ansible Automation Platform, Self-Support (100 Managed Nodes)	100	5/3/2021, 3:59:59 AM

If you encounter the following error message, you will need the proper permissions required for the Satellite user with which the controller admin uses to apply a subscription.



ERROR FETCHING LICENSES

ERROR: 403 Client Error: Forbidden for url: https://satellite-subs.ansible.eng.rdu2.redhat.com/katello/api/organizations

The Satellite username/password is used to query the Satellite API for existing subscriptions. From the Satellite API, the automation controller gets back some metadata about those subscriptions, then filter through to find valid subscriptions that you could apply, which are then displayed as valid subscription options in the UI.

The following Satellite roles grant proper access:

- Custom with `view_subscriptions` and `view_organizations` filter
- Viewer
- Administrator
- Organization Admin
- Manager

As the *Custom* role is the most restrictive of these, this is the recommend role to use for your controller integration. Refer to the Satellite documentation on managing users and roles for more detail.

Note: The System Administrator role is not equivalent to the Administrator user checkbox, and will not provide sufficient permissions to access the subscriptions API page.

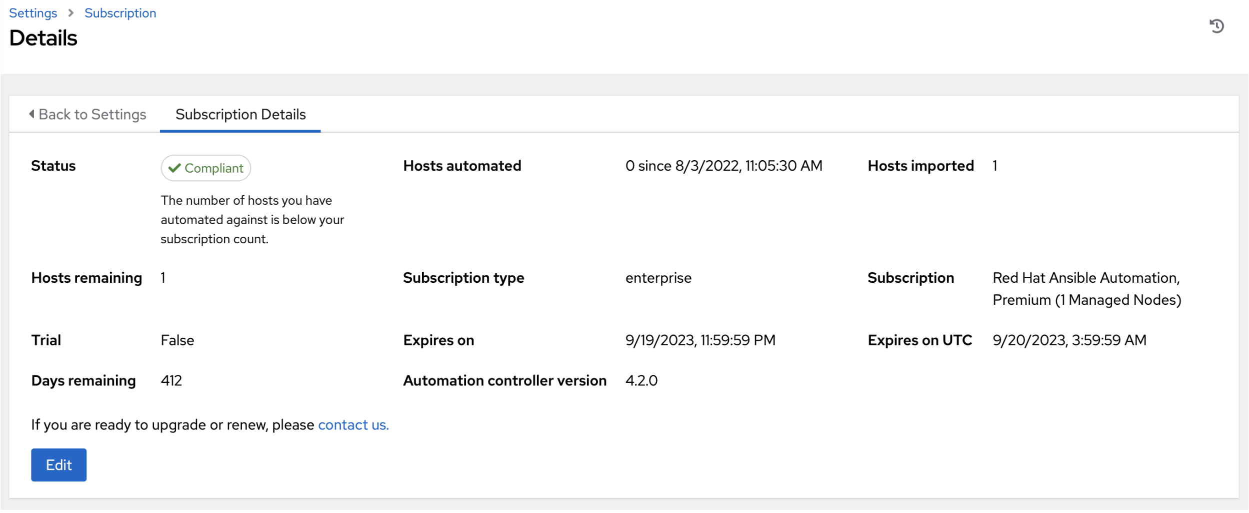
3. Click **Next** to proceed to **Tracking and Insights**. Tracking and insights collect data to help Red Hat improve the product by delivering you a much better user experience. For more information about data collection, refer to Usability Analytics and Data Collection. This option is checked by default, but you may opt out of any of the following:

- **User analytics** collects data from the controller User Interface.
- **Insights Analytics** provides a high level analysis of your automation with automation controller, which is used to help you identify trends and anomalous use of the controller. For opt-in of Automation Analytics to have any effect, your instance of automation controller **must** be running on Red Hat Enterprise Linux. See instructions described in the Automation Analytics section. If you select to opt-in for this option, the screen expands and prompts for a username and password to enable Insights, if applicable.

Note: You may change your analytics data collection preferences at any time, as described in the Usability Analytics and Data Collection section.

4. After you have specified your tracking and insights preferences, click **Next** to proceed to the End User Agreement.
5. Review and check the **I agree to the End User License Agreement** checkbox and click **Submit**.

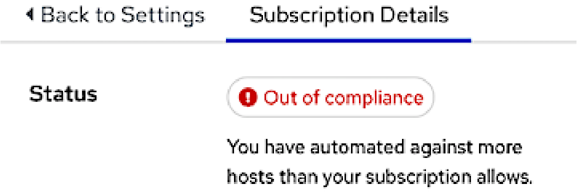
Once your subscription has been accepted, the controller briefly displays the subscription details and navigates you to the Dashboard of the automation controller interface. For later reference, you can return to this screen by clicking **Settings** from the left navigation bar and select **Subscription settings** from the Subscription option.



Status	✓ Compliant The number of hosts you have automated against is below your subscription count.	**Hosts automated**	0 since 8/3/2022, 11:05:30 AM	**Hosts imported**	1
Hosts remaining	1	**Subscription type**	enterprise	**Subscription**	Red Hat Ansible Automation, Premium (1 Managed Nodes)
Trial	False	**Expires on**	9/19/2023, 11:59:59 PM	**Expires on UTC**	9/20/2023, 3:59:59 AM
Days remaining	412	**Automation controller version**	4.2.0		

If you are ready to upgrade or renew, please contact us.

A status of *Compliant* indicates your subscription is in compliance with the number of hosts you have automated within your subscription count. Otherwise, your status will show an *Out of Compliance* status, indicating you have exceeded the number of hosts in your subscription.



You have automated against more hosts than your subscription allows.

Other important information displayed are:

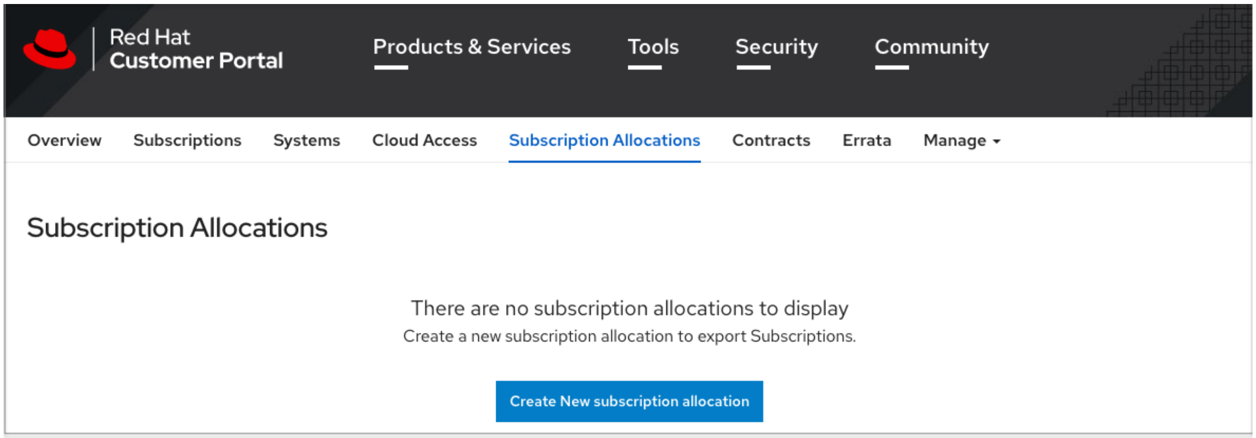
- **Hosts automated**: Host count automated by the job, which consumes the license count
- **Hosts imported**: Host count considering all inventory sources (does not impact hosts remaining)
- **Hosts remaining**: Total host count minus hosts automated

4.1 Obtaining a subscriptions manifest

In order to upload a subscriptions manifest, first set up your subscription allocations:

1. Go to https://access.redhat.com/management/subscription_allocations.

The Subscriptions Allocations page contains no subscriptions until you create one.



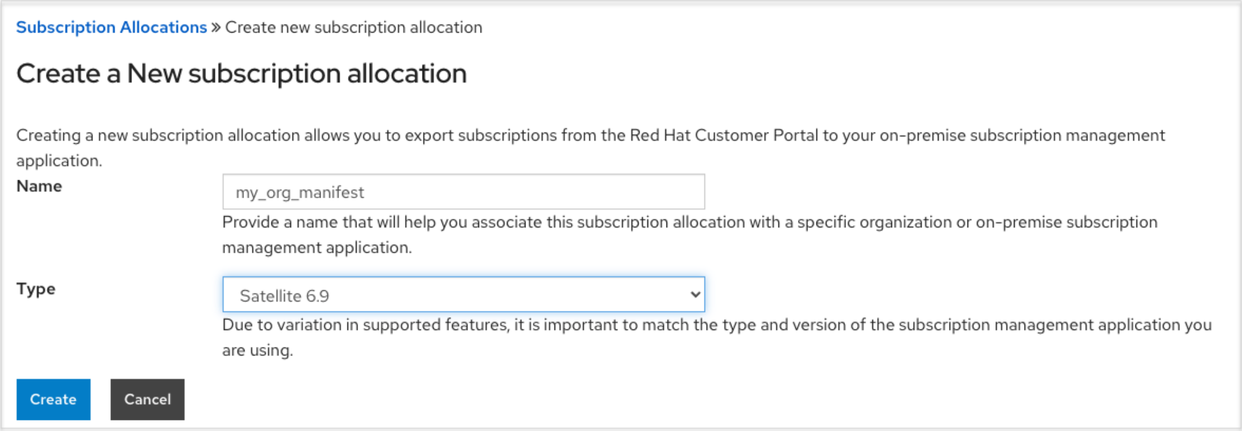
There are no subscription allocations to display

Create a new subscription allocation to export Subscriptions.

2. Click the **Create New subscription allocation** button to create a new subscription allocation.

Note: If this button is not present or disabled, you do not have the proper permissions to create subscription allocations. To create a subscription allocation, you must either be an *Administrator* on the Customer Portal, or have the *Manage Your Subscriptions* role. Contact an access.redhat.com administrator or organization administrator who will be able to grant you permission to manage subscriptions.

3. Enter a name for your subscription and select **Satellite 6.8** from the Type drop-down menu.



Subscription Allocations » Create new subscription allocation

Create a New subscription allocation

Creating a new subscription allocation allows you to export subscriptions from the Red Hat Customer Portal to your on-premise subscription management application.

Name: my_org_manifest

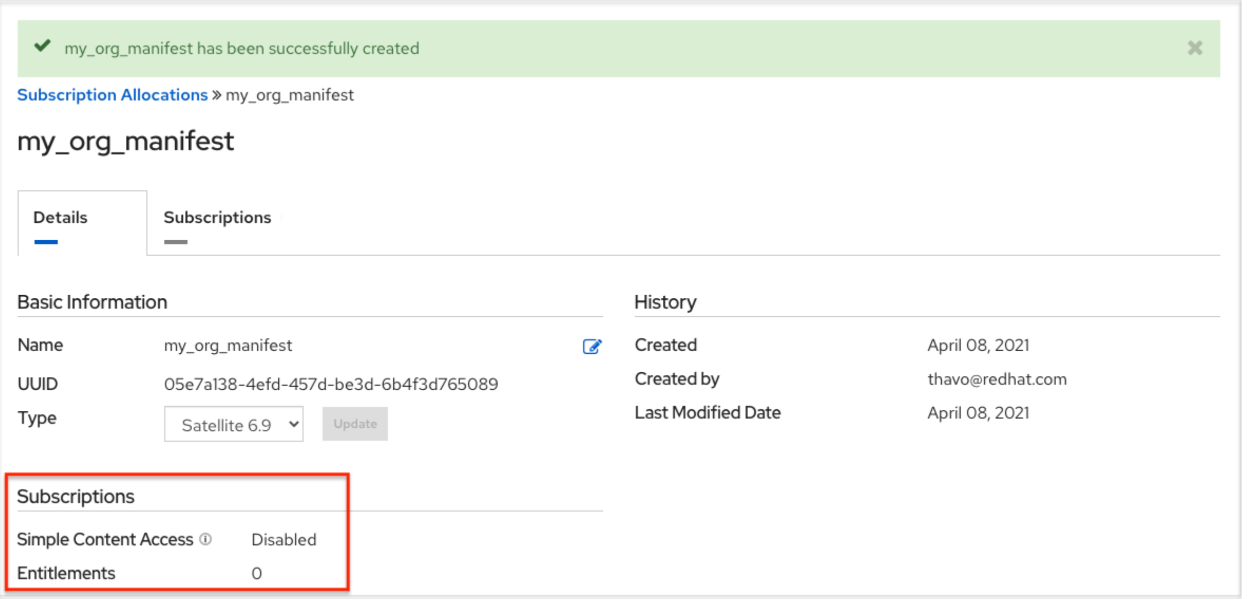
Provide a name that will help you associate this subscription allocation with a specific organization or on-premise subscription management application.

Type: Satellite 6.9

Due to variation in supported features, it is important to match the type and version of the subscription management application you are using.

4. Click **Create**.

5. Once your subscriptions manifest is successfully created, it displays various information including subscription information at the bottom of the **Details** tab. The number indicated next to Entitlements indicates the number of entitlements associated with your subscription.



my_org_manifest has been successfully created

Subscription Allocations » my_org_manifest

my_org_manifest

Basic Information

Name	my_org_manifest
UUID	05e7a138-4efd-457d-be3d-6b4f3d765089
Type	Satellite 6.9

History

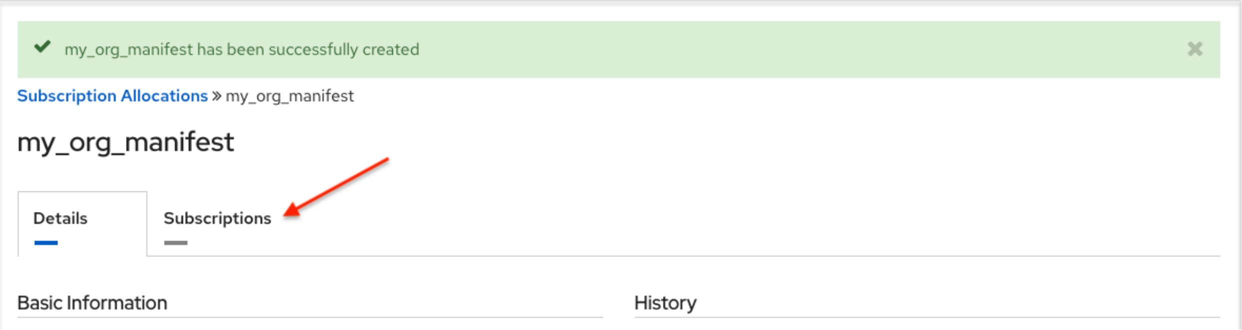
Created	April 08, 2021
Created by	thavo@redhat.com
Last Modified Date	April 08, 2021

Subscriptions

Simple Content Access	Disabled
Entitlements	0

In order to obtain a subscriptions manifest, you must add an entitlement to your subscriptions though the Subscriptions tab.

6. Click the **Subscriptions** tab.

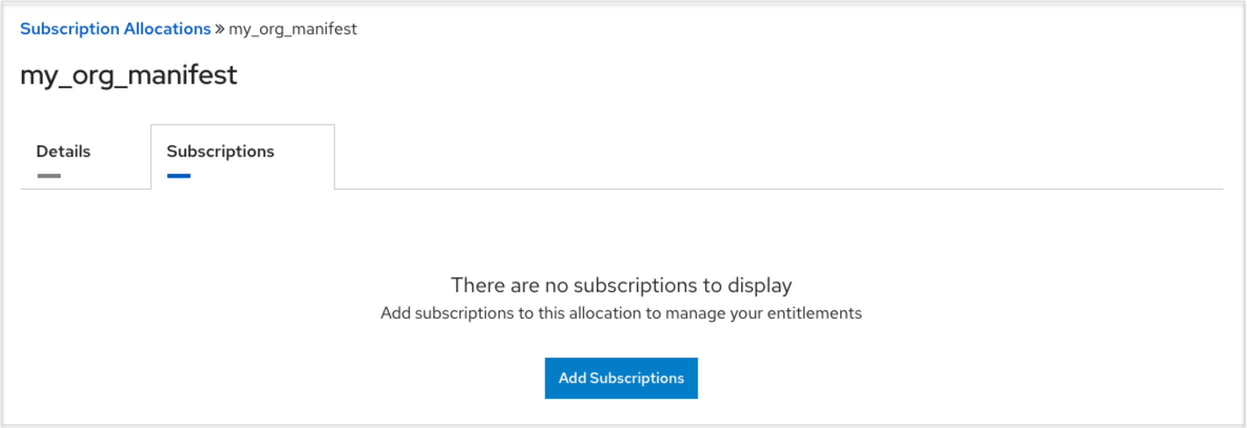


my_org_manifest has been successfully created

Subscription Allocations » my_org_manifest

my_org_manifest

7. In the Subscriptions tab, there are no subscriptions to display, click the **Add Subscriptions** button.



Subscription Allocations » my_org_manifest

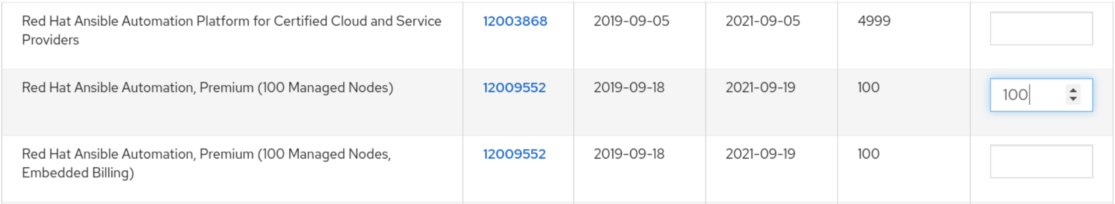
my_org_manifest

There are no subscriptions to display

Add subscriptions to this allocation to manage your entitlements

The next screen allows you to select and add entitlements to put in the manifest file. You may select multiple Ansible Automation Platform subscriptions (with the same SKU) in your subscription allocation. Valid Ansible Automation Platform subscriptions commonly go by the name "Red Hat Ansible Automation…".

8. Specify the number of entitlements/managed nodes to put in the manifest file. This allows you to split up a subscription (for example: 400 nodes on a development cluster and 600 nodes for the production cluster, out of a 1000 node subscription).



Red Hat Ansible Automation Platform for Certified Cloud and Service Providers	12003868	2019-09-05	2021-09-05	4999	
Red Hat Ansible Automation, Premium (100 Managed Nodes)	12009552	2019-09-18	2021-09-19	100	100
Red Hat Ansible Automation, Premium (100 Managed Nodes, Embedded Billing)	12009552	2019-09-18	2021-09-19	100	

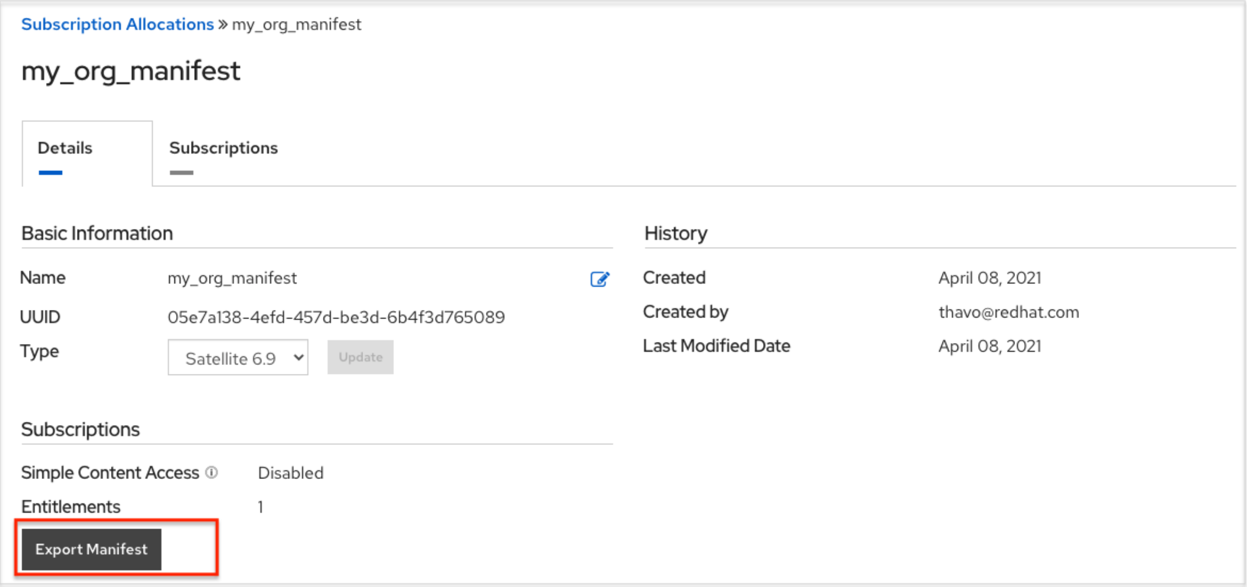
Note: You can apply multiple subscriptions to a single installation by adding multiple subscriptions of the same type to a manifest file and uploading them. Similarly, a subset of a subscription can be applied by only allocating a portion of the subscription when creating the manifest.

9. Click **Submit**.

The allocations you specified, once successfully added, are displayed in the **Subscriptions** tab.

10. Click the **Details** tab to access the subscription manifest file.

11. At the bottom of the details window under *Entitlements*, click the **Export Manifest** button to export the manifest file for this subscription.



Subscription Allocations » my_org_manifest

my_org_manifest

Basic Information

Name	my_org_manifest
UUID	05e7a138-4efd-457d-be3d-6b4f3d765089
Type	Satellite 6.9

History

Created	April 08, 2021
Created by	thavo@redhat.com
Last Modified Date	April 08, 2021

Subscriptions

Simple Content Access	Disabled
Entitlements	1

A folder pre-pended with `manifest_` in the name is downloaded to your local drive. Multiple subscriptions with the same SKU will be aggregated.

12. Now that you have a subscription manifest, proceed to the *Subscription screen*. Upload the entire manifest file (.zip) by clicking **Browse** and navigate to the location where the file is saved. Do not open it or upload individual parts of it.

4.2 Adding a subscription manually

If you are unable to apply or update the subscription info using the user interface, you can upload the subscriptions manifest manually in an Ansible playbook using the `license` module in the `ansible.controller` collection:

```
- name: Set the license using a file
  license:
  manifest: "/tmp/my_manifest.zip"
```

See the Ansible tower_license module for more information.

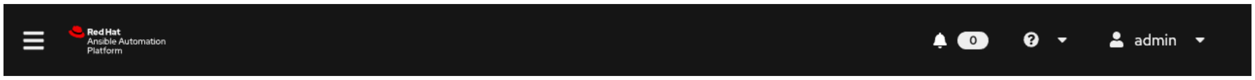
CHAPTER

FIVE

THE USER INTERFACE

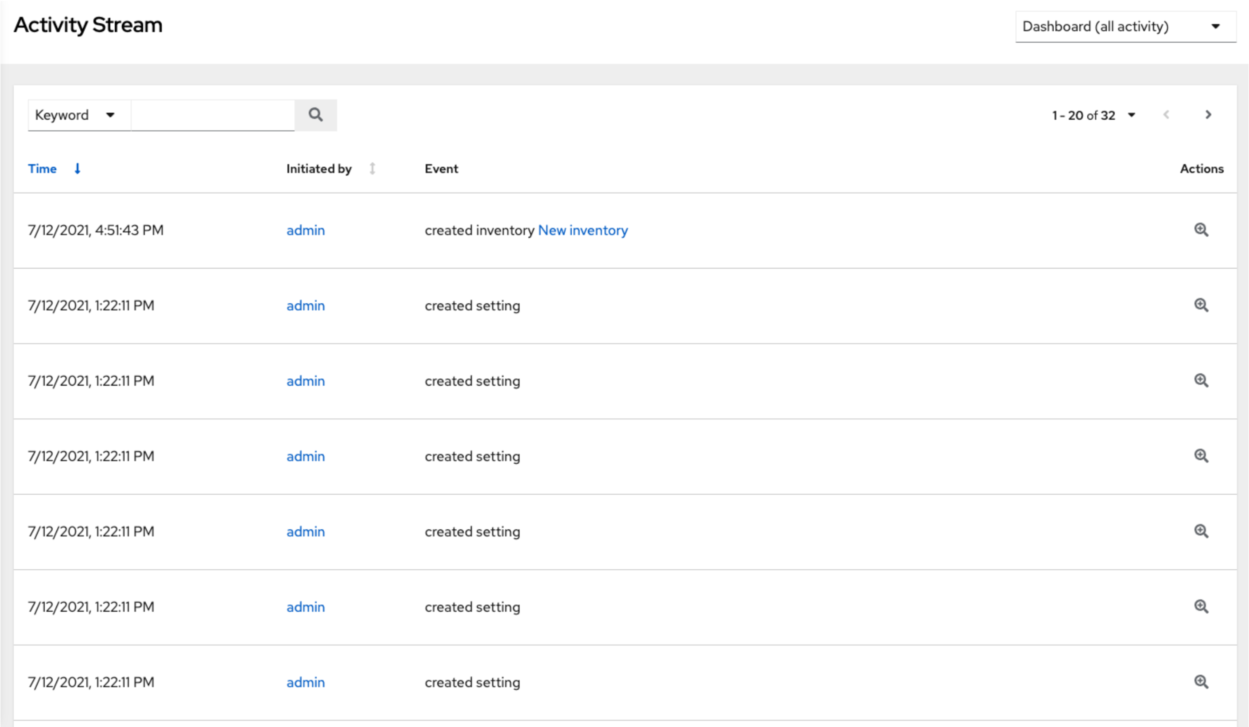
The User Interface offers a friendly graphical framework for your IT orchestration needs. The left navigation bar provides quick access to resources, such as **Projects**, **Inventories**, **Job Templates**, and **Jobs**.

Across the top-right side of the interface, you can access your user profile, the About page, view related documentation, and log out. Right below these options, you can view the activity stream for that user by clicking on the Activity Stream ⟲ button.



5.1 Activity Streams

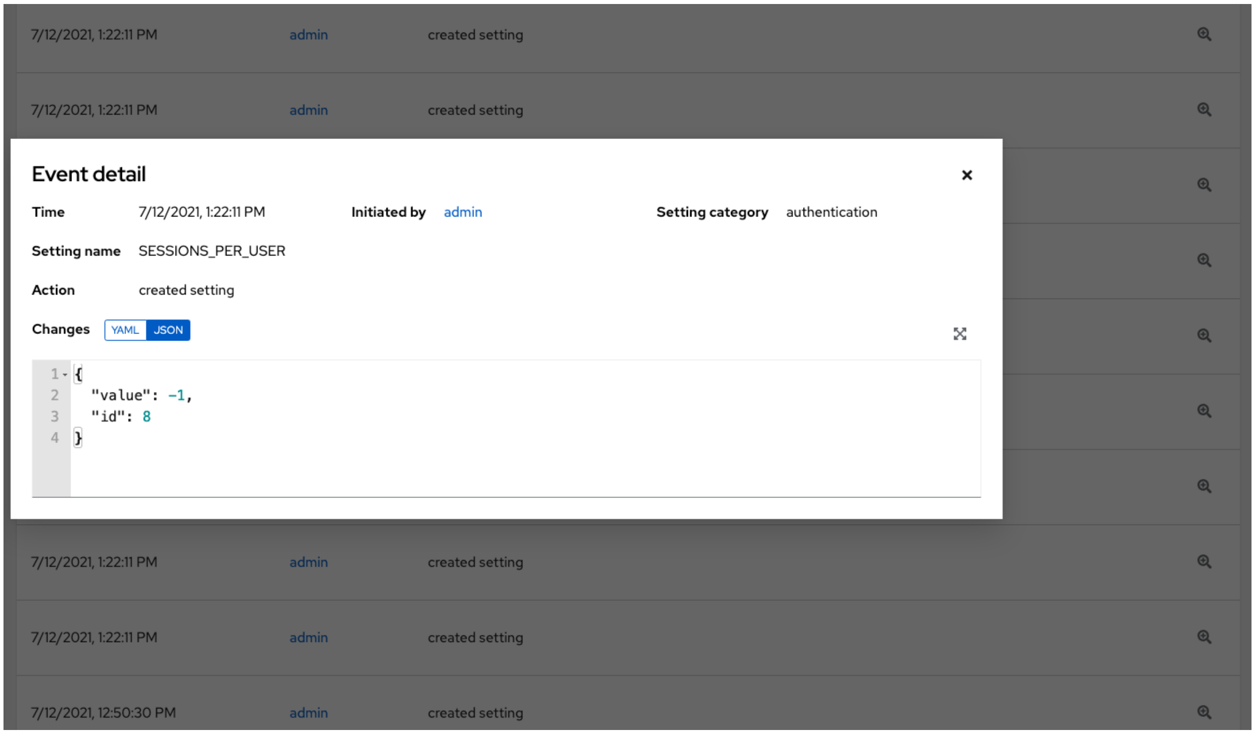
Most screens have an Activity Stream (⟲) button. Clicking this brings up the **Activity Stream** for this object.



Activity Stream

Time	Initiated by	Event	Actions
7/12/2021, 4:51:43 PM	admin	created inventory New inventory	🔍
7/12/2021, 1:22:11 PM	admin	created setting	🔍
7/12/2021, 1:22:11 PM	admin	created setting	🔍
7/12/2021, 1:22:11 PM	admin	created setting	🔍
7/12/2021, 1:22:11 PM	admin	created setting	🔍
7/12/2021, 1:22:11 PM	admin	created setting	🔍
7/12/2021, 1:22:11 PM	admin	created setting	🔍

An Activity Stream shows all changes for a particular object. For each change, the Activity Stream shows the time of the event, the user that initiated the event, and the action. The information displayed varies depending on the type of event. Clicking on the Examine (🔍) button shows the event log for the change.



7/12/2021, 1:22:11 PM admin created setting

7/12/2021, 1:22:11 PM admin created setting

Event detail

Time 7/12/2021, 1:22:11 PM

Initiated by admin

Setting category authentication

Setting name SESSIONS_PER_USER

Action created setting

Changes

```
{
  "value": -1,
  "id": 8
}
```

7/12/2021, 1:22:11 PM admin created setting

7/12/2021, 1:22:11 PM admin created setting

7/12/2021, 12:50:30 PM admin created setting

The Activity Stream can be filtered by the initiating user (or the system, if it was system initiated), and by any related object, such as a particular credential, job template, or schedule.

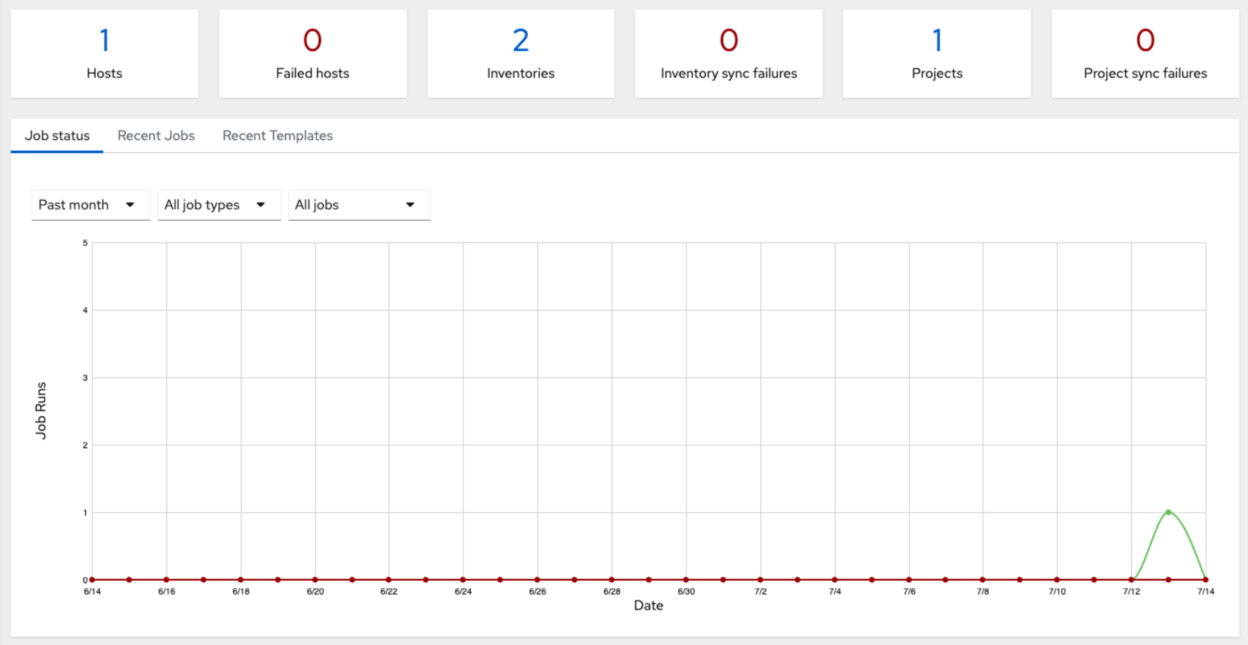
The Activity Stream on the main Dashboard shows the Activity Stream for the entire instance. Most pages allow viewing an activity stream filtered for that specific object.

5.2 Views

The User Interface provides several options for viewing information.

5.2.1 Dashboard view

The **Dashboard** view begins with a summary of your hosts, inventories, and projects. Each of these is linked to the corresponding objects for easy access.

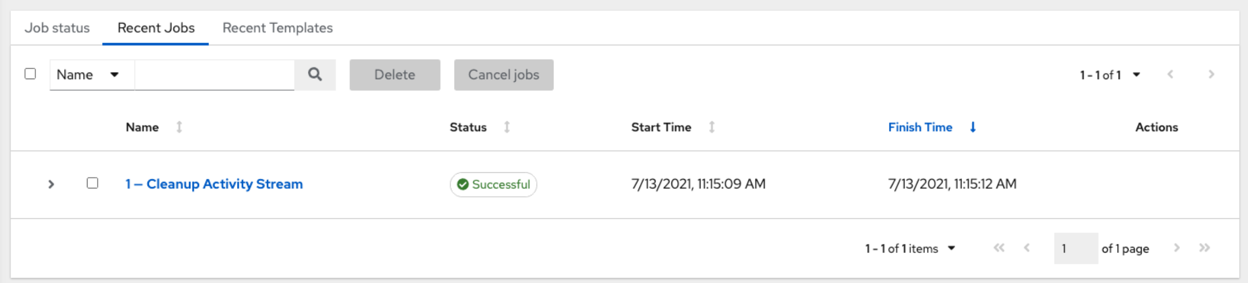


1	0	2	0	1	0
Hosts	Failed hosts	Inventories	Inventory sync failures	Projects	Project sync failures

On the main Dashboard screen, a summary appears listing your current **Job Status**. The **Job Status** graph displays the number of successful and failed jobs over a specified time period. You can choose to limit the job types that are viewed, and to change the time horizon of the graph.

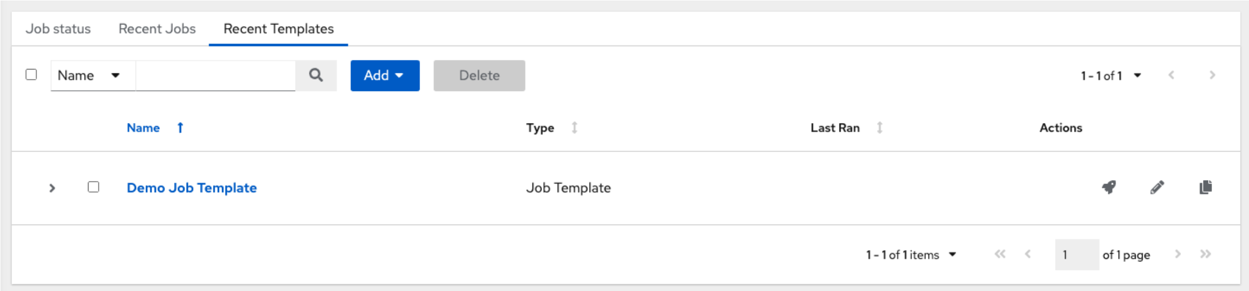
Also available for view are summaries of **Recent Jobs** and **Recent Templates** on their respective tabs.

The **Recent Jobs** section displays which jobs were most recently run, their status, and time when they were run as well.



Name	Status	Start Time	Finish Time	Actions
1 – Cleanup Activity Stream	Successful	7/13/2021, 11:15:09 AM	7/13/2021, 11:15:12 AM	

The **Recent Templates** section of this display shows a summary of the most recently used templates. You can also access this summary by clicking **Templates** from the left navigation bar.

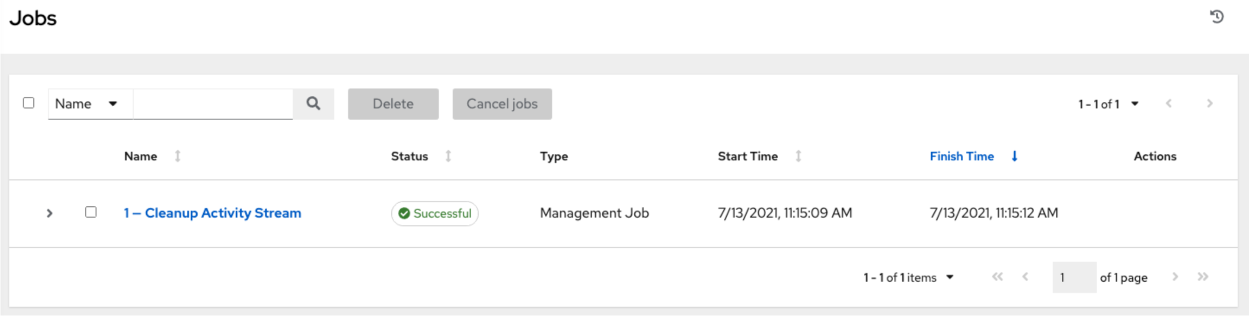


	Name	Type	Last Ran	Actions
› □	Demo Job Template	Job Template		

Note: Clicking on **Dashboard** from the left navigation bar or the Ansible Automation Platform logo at any time returns you to the Dashboard.

5.2.2 Jobs view

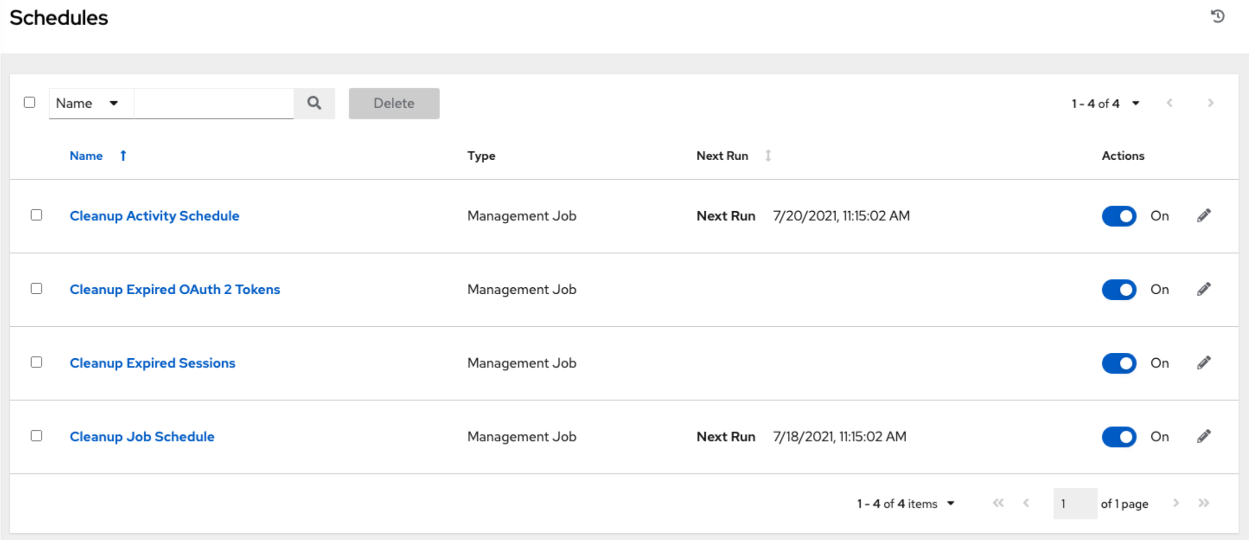
Access the **Jobs** view by clicking **Jobs** from the left navigation bar. This view shows all the jobs that have ran, including projects, templates, management jobs, SCM updates, playbook runs, etc.



	Name	Status	Type	Start Time	Finish Time	Actions
› □	1 – Cleanup Activity Stream	✓ Successful	Management Job	7/13/2021, 11:15:09 AM	7/13/2021, 11:15:12 AM	

5.2.3 Schedules view

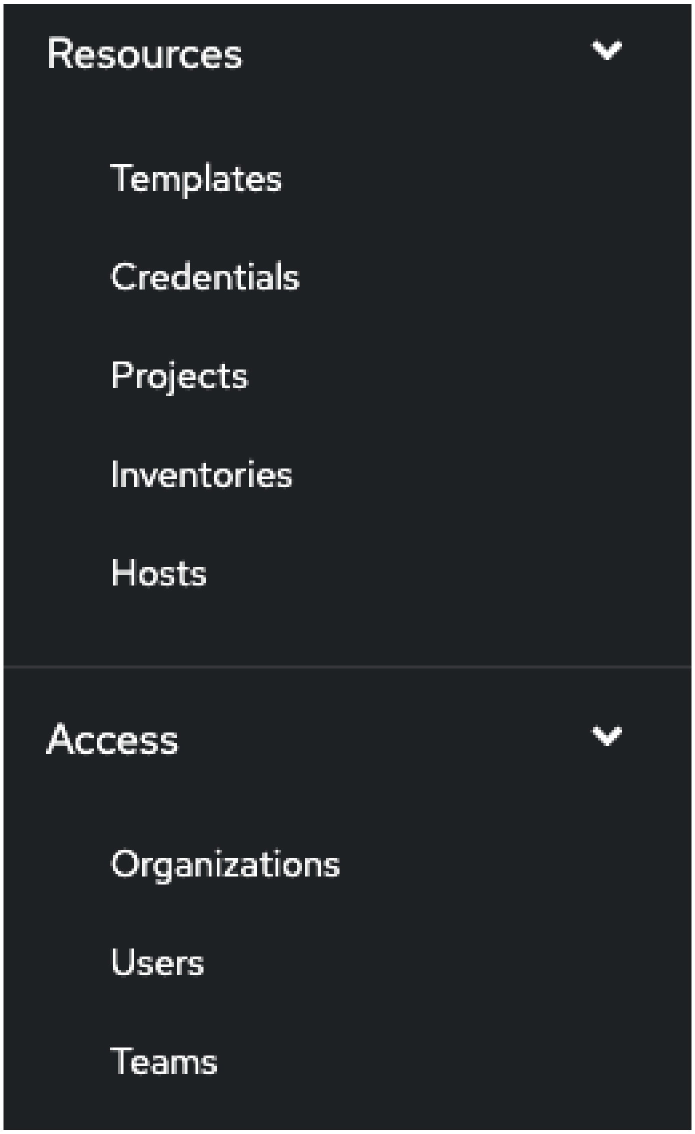
Access the Schedules view by clicking **Schedules** from the left navigation bar. This view shows all the scheduled jobs that are configured.



	Name	Type	Next Run	Actions
□	Cleanup Activity Schedule	Management Job	Next Run 7/20/2021, 11:15:02 AM	On
□	Cleanup Expired OAuth 2 Tokens	Management Job		On
□	Cleanup Expired Sessions	Management Job		On
□	Cleanup Job Schedule	Management Job	Next Run 7/18/2021, 11:15:02 AM	On

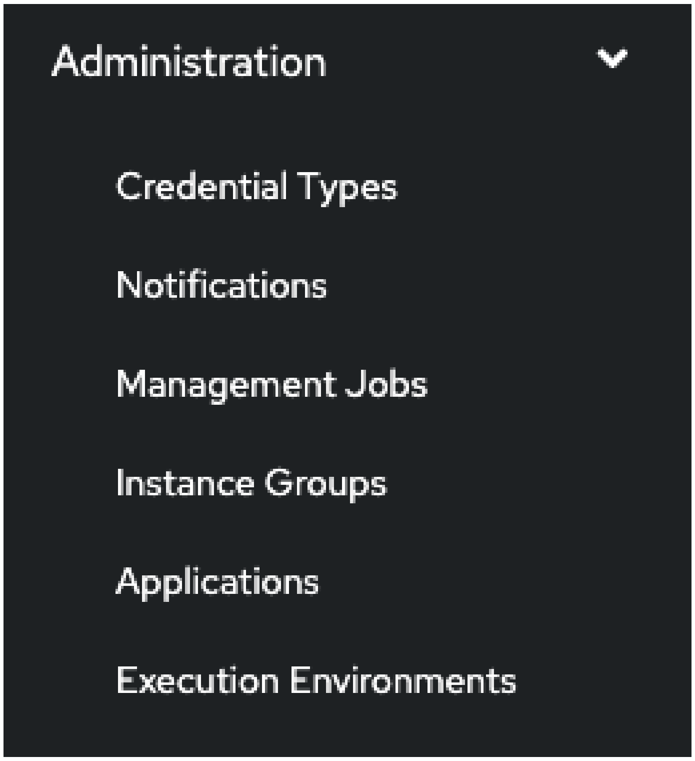
5.3 Resources and Access

The **Resources** and **Access** menus provide you access to the various components of automation controller and allow you to configure who has permissions for which of those resources.



5.4 Administration Menu

The **Administration** menu provides access to the various administrative options:

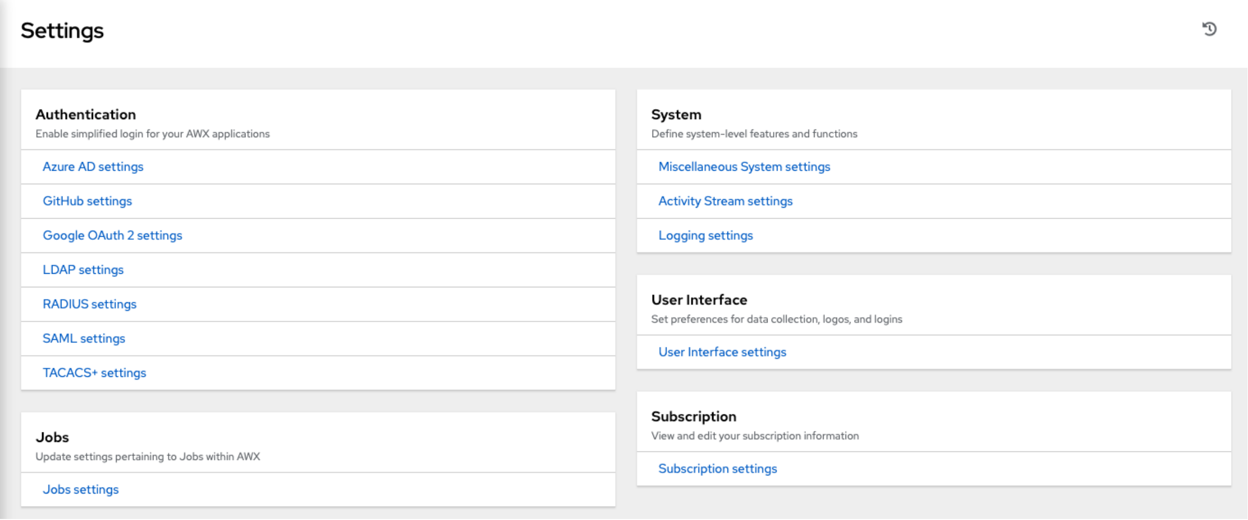


From here, you can create, view, and edit *custom credential types*, *notifications*, management jobs, *tokens and applications*, and configure *Execution Environments*.

5.5 The Settings Menu

Configuring global and system-level settings is accomplished through the **Settings** menu, which is described in further detail in the proceeding section. The **Settings** menu offers access to administrative configuration options.

To enter the Settings window for automation controller, click **Settings** from the left navigation bar. This page allows you to modify your controller's configuration, such as settings associated with authentication, jobs, system, user interface, and view or retrieve your subscription information.



Settings

Authentication
Enable simplified login for your AWX applications

- Azure AD settings
- GitHub settings
- Google OAuth 2 settings
- LDAP settings
- RADIUS settings
- SAML settings
- TACACS+ settings

Jobs
Update settings pertaining to Jobs within AWX

- Jobs settings

System
Define system-level features and functions

- Miscellaneous System settings
- Activity Stream settings
- Logging settings

User Interface
Set preferences for data collection, logos, and logins

- User Interface settings

Subscription
View and edit your subscription information

- Subscription settings

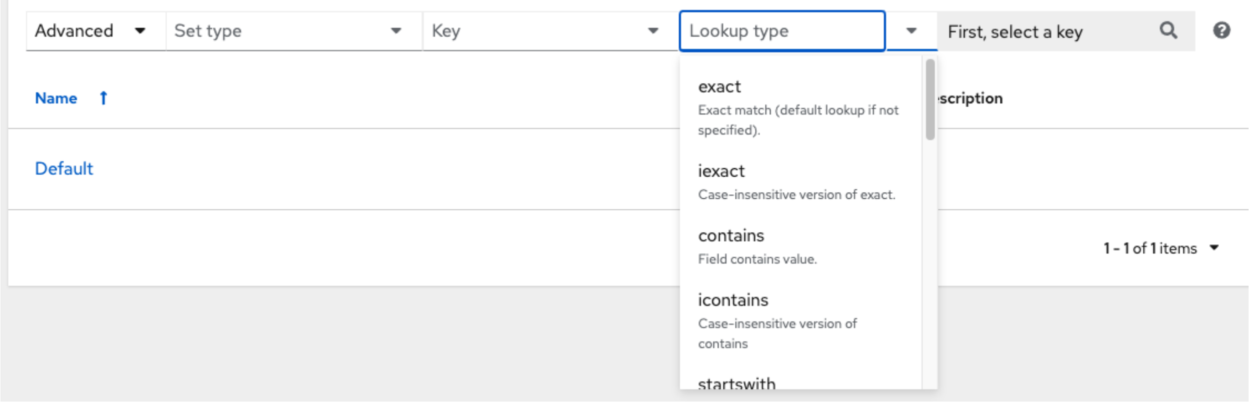
For more information on configuring these settings, refer to Controller Configuration section of the *Automation Controller Administration Guide*.

CHAPTER

SIX

SEARCH

The automation controller has a powerful search tool that provides both search and filter capabilities that span across multiple functions. Acceptable search criteria are provided in an expandable “cheat-sheet” accessible from the **Advanced** option from the **Name** drop-down menu in the search field. From there, use the combination of **Set Type**, **Key**, **Lookup type** to filter.



6.1 Searching Tips

These searching tips assume that you are not searching hosts. Most of this section still applies to hosts but with some subtle differences. A typical syntax of a search consists a field (left-hand side) and a value (right-hand side). A colon is used to separate the field that you want to search from the value. If a search doesn’t have a colon (see example 3) it is treated as a simple string search where `?search=foobar` is sent. Here are the examples of syntax used for searching:

1. `name:localhost` In this example, the string before the colon represents the field that you want to search on. If that string does not match something from **Fields** or **Related Fields** then it’s treated the same way Example 3 is (string search). The string after the colon is the string that you want to search for within the name attribute.

2. `organization.name:Default` This example shows a Related Field Search. The period in the left-hand portion separates the model from the field in this case. Depending on how deep/complex the search is, you could have multiple periods in that left-hand portion.

3. `foobar` Simple string (key term) search that will find all instances of that term using an `icontains` search against the name and description fields. If a space is used between terms (e.g. foo bar), then any results that contain both terms will be returned. If the terms are wrapped in quotes (e.g. “foo bar”), the controller will search for the entire string with the terms appearing together. Specific name searches will search against the API name. For example, `Management job` in the user interface is `system_job` in the API.

4. `organization:Default` This example shows a Related Field search but without specifying a field to go along with the organization. This is supported by the API and is analogous to a simple string search but done against the organization (will do an `icontains` search against both the name and description).

6.1.1 Values for search fields

To find values for certain fields, refer to the API endpoint for extensive options and their valid values. For example, if you want to search against `/api/v2/jobs` -> `type` field, you can find the values by performing an **OPTIONS** request to `/api/v2/jobs` and look for entries in the API for `"type"`. Additionally, you can view the related searches by scrolling to the bottom of each screen. In the example for `/api/v2/jobs`, the related search shows:

```
"related_search_fields": [
   "modified_by__search",
   "project__search",
   "project_update__search",
   "credentials__search",
   "unified_job_template__search",
   "created_by__search",
   "inventory__search",
   "labels__search",
   "schedule__search",
   "webhook_credential__search",
   "job_template__search",
   "job_events__search",
   "dependent_jobs__search",
   "launch_config__search",
   "unifiedjob_ptr__search",
   "notifications__search",
   "unified_job_node__search",
   "instance_group__search",
   "hosts__search",
   "job_host_summaries__search"
```

The values for Fields come from the keys in a **GET** request. `url`, `related`, and `summary_fields` are not used. The values for Related Fields also come from the **OPTIONS** response, but from a different attribute. Related Fields is populated by taking all the values from `related_search_fields` and stripping off the `__search` from the end.

Any search that does not start with a value from Fields or a value from the Related Fields, will be treated as a generic string search. Searching for something like `localhost` will result in the UI sending `?search=localhost` as a query parameter to the API endpoint. This is a shortcut for an `icontains` search on the name and description fields.

6.1.2 Searching using values from Related Fields

Searching a Related Field requires you to start the search string with the Related Field. This example describes how to search using values from the Related Field, *organization*.

The left-hand side of the search string must start with *organization* (ex: `organization:Default`). Depending on the related field, you might want to provide more specific direction for the search by providing secondary/tertiary fields. An example of this would be to specify that you want to search for all job templates that use a project matching a certain name. The syntax on this would look like: `job_template.project.name:"A Project"`.

Note: This query would execute against the `unified_job_templates` endpoint which is why it starts with `job_template`. If we were searching against the `job_templates` endpoint, then you wouldn't need the `job_template` portion of that query.

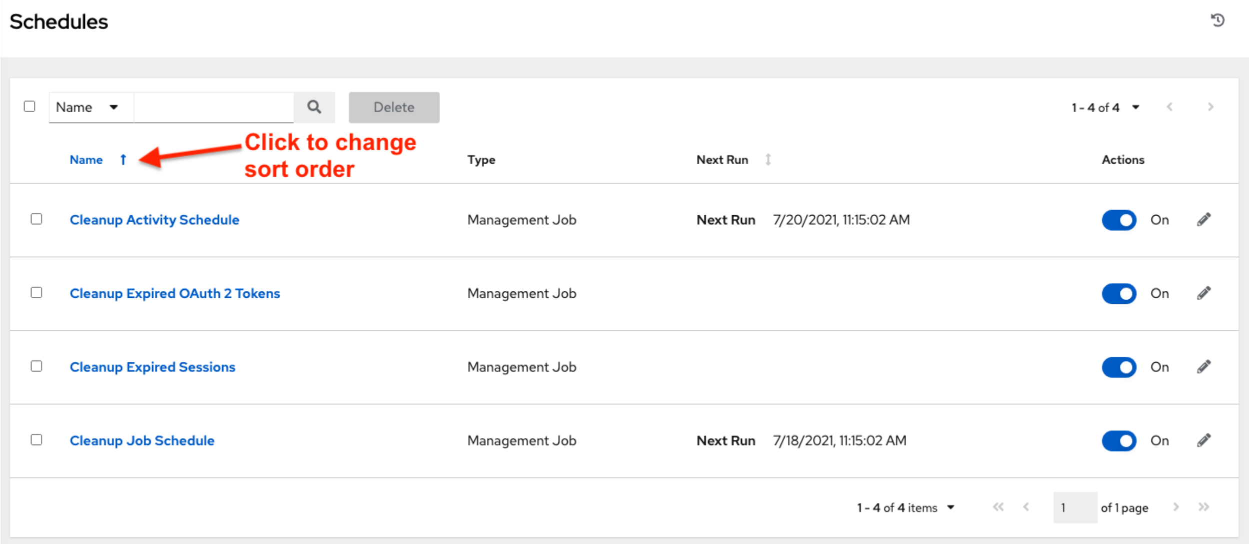
6.1.3 Other search considerations

The following are a few things about searching in the controller that you should be aware of:

- There's currently no supported syntax for **OR** queries. All search terms get **AND**'d in the query parameters.
- The left-hand portion of a search parameter can be wrapped in quotes to support searching for strings with spaces.
- Currently, the values in the Fields are direct attributes expected to be returned in a **GET** request. Whenever you search against one of the values, the controller essentially does an `__icontains` search. So, for example, `name:localhost` would send back `?name__icontains=localhost`. The controller currently performs this search for every Field value, even `id`, which is not ideal.

6.2 Sort

Where applicable, use the arrows in each column to sort by ascending or descending order (following is an example from the schedules list).



Click to change sort order

□	Name ↑	Type	Next Run ↕	Actions
□	Cleanup Activity Schedule	Management Job	Next Run 7/20/2021, 11:15:02 AM	On
□	Cleanup Expired OAuth 2 Tokens	Management Job		On
□	Cleanup Expired Sessions	Management Job		On
□	Cleanup Job Schedule	Management Job	Next Run 7/18/2021, 11:15:02 AM	On

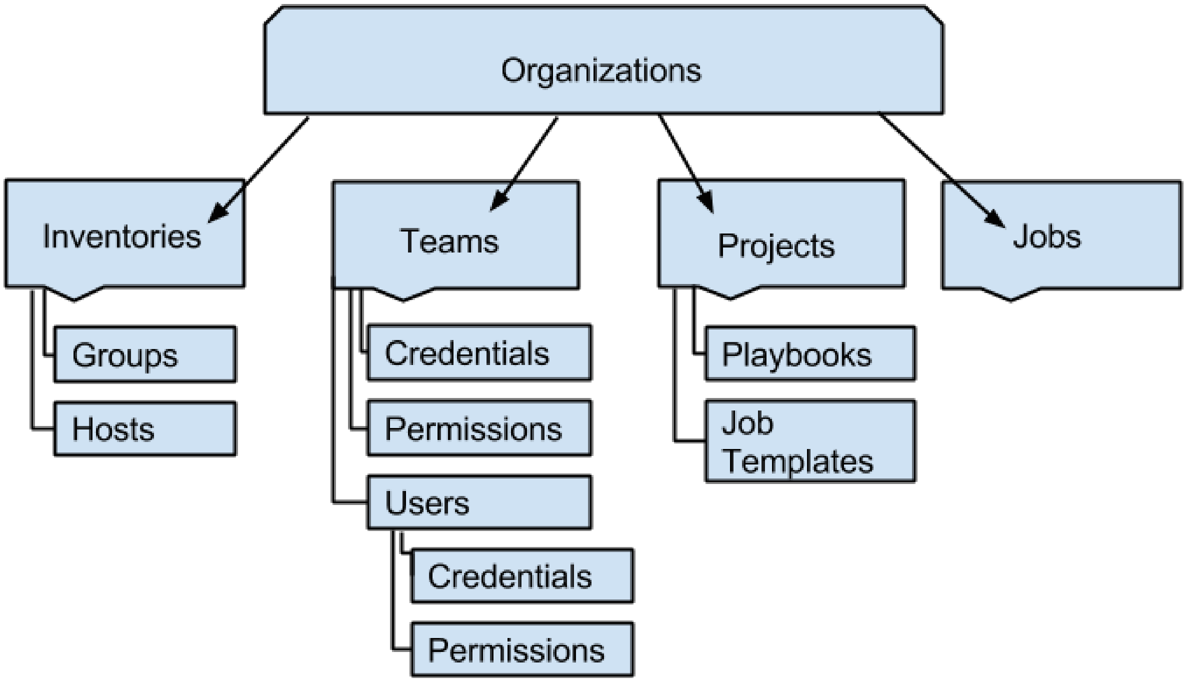
The direction of the arrow indicates the sort order of the column.

CHAPTER

SEVEN

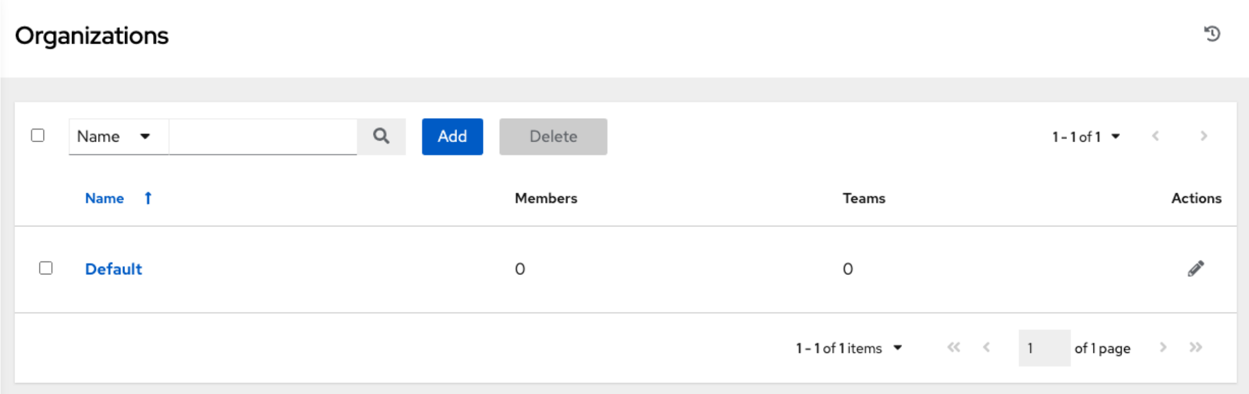
ORGANIZATIONS

An *Organization* is a logical collection of **Users**, **Teams**, **Projects**, and **Inventories**, and is the highest level in the automation controller object hierarchy.



Access the Organizations page by clicking **Organizations** from the left navigation bar. The Organizations page displays all of the existing organizations for your installation. Organizations can be searched by **Name** or **Description**. Modify and remove organizations using the **Edit** and **Delete** buttons.

Note: A default organization is automatically created.



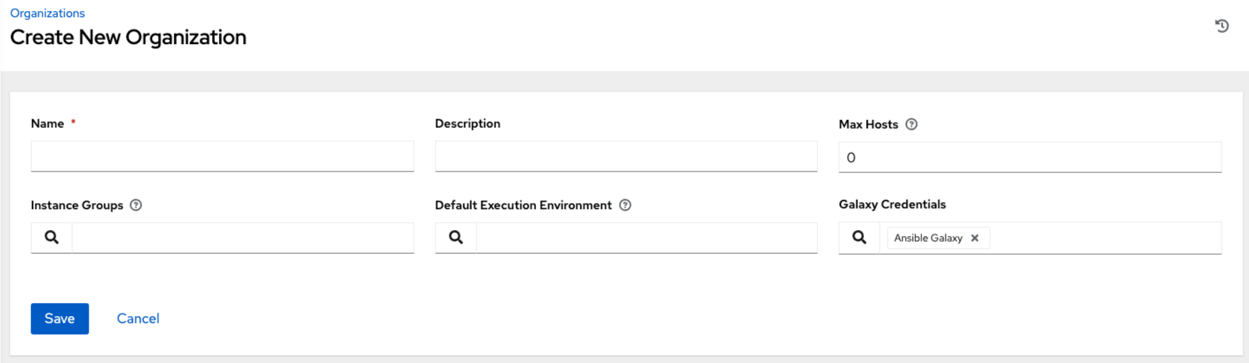
Organizations

□	Name ↑	Members	Teams	Actions
□	Default	0	0	✎

From this list view, you can edit the details of an organization (✎) from the **Actions** menu.

7.1 Creating a New Organization

1. You can create a new organization by clicking the **Add** button.



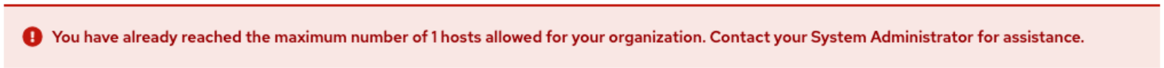
Organizations

Create New Organization

Name *	Description	Max Hosts ⓘ
		0
Instance Groups ⓘ	Default Execution Environment ⓘ	Galaxy Credentials
		Ansible Galaxy ×

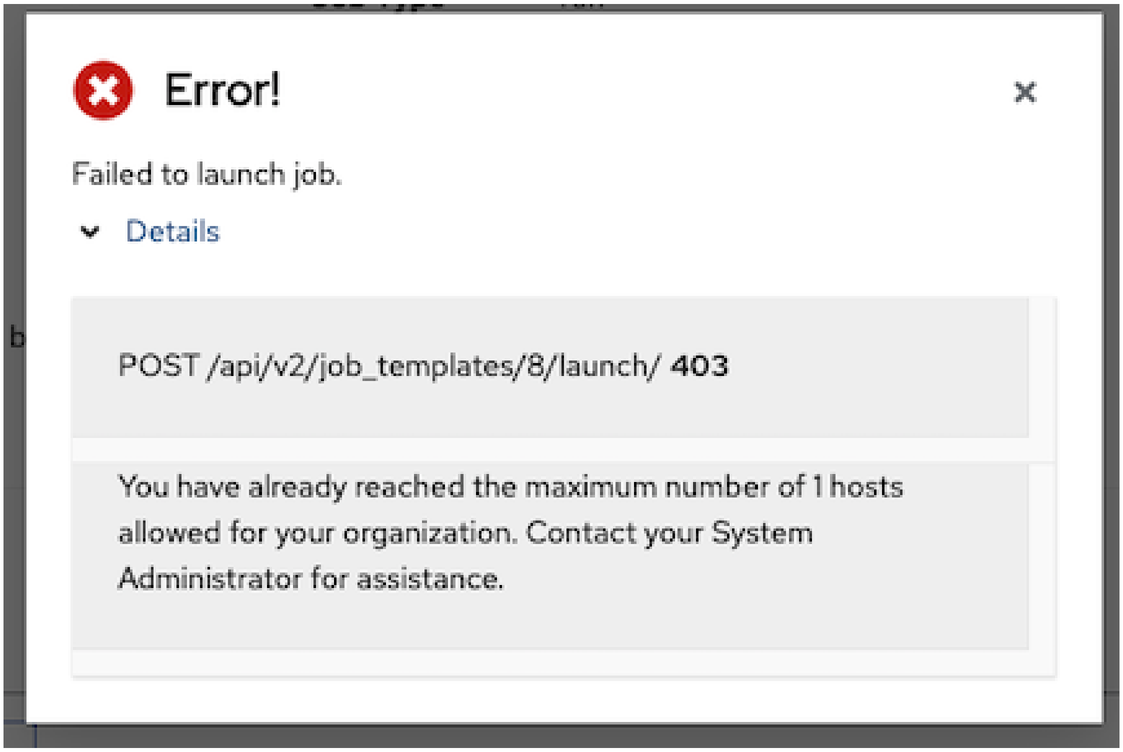
2. An organization has several attributes that may be configured:

- Enter the **Name** for your organization (required).
- Enter a **Description** for the organization.
- The **Max Hosts** is only editable by a superuser to set an upper limit on the number of license hosts that an organization can have. Setting this value to **0** signifies no limit. If you try to add a host to an organization that has reached or exceeded its cap on hosts, an error message displays:



You have already reached the maximum number of 1 hosts allowed for your organization. Contact your System Administrator for assistance.

The inventory sync output view also shows the host limit error. Click **Details** for additional detail about the error.



Error!

Failed to launch job.

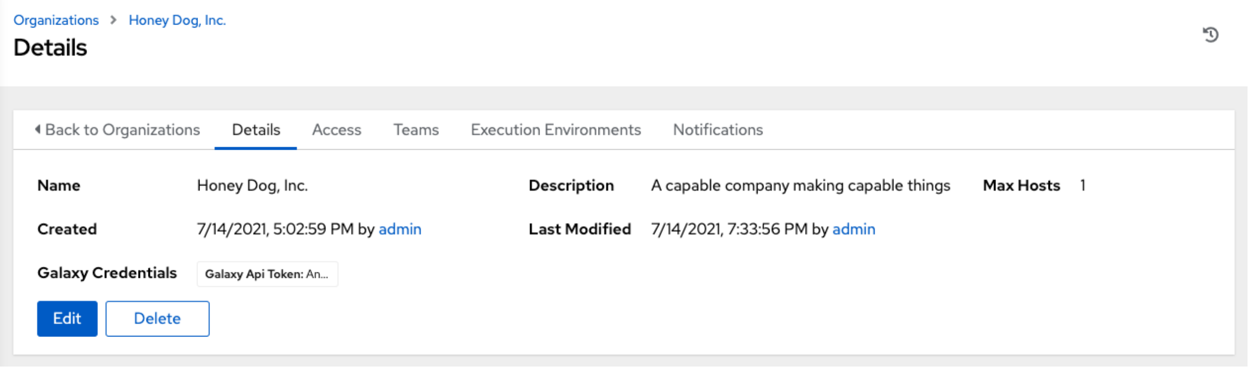
POST /api/v2/job_templates/8/launch/ 403

You have already reached the maximum number of 1 hosts allowed for your organization. Contact your System Administrator for assistance.

- Enter **Instance Groups** on which to run this organization.
- Enter the name of the execution environment or search for an existing **Default Execution Environment** on which to run this organization. See Upgrading to Execution Environments in the *Ansible Automation Platform Upgrade and Migration Guide* for more information.
- If used, enter the **Galaxy Credentials** or search from a list of existing ones.

3. Click **Save** to finish creating the organization.

Once created, automation controller displays the Organization details, and allows for the managing access and execution environments for the organization.

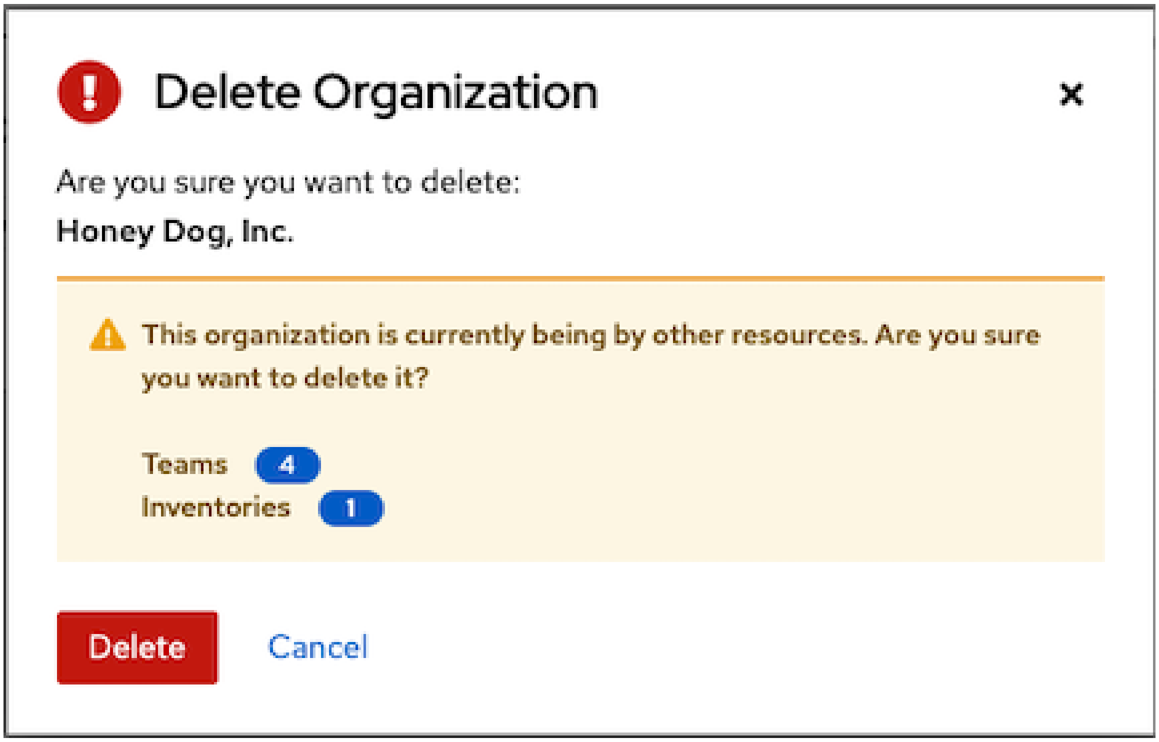


Details

Name	Honey Dog, Inc.	Description	A capable company making capable things	Max Hosts	1
Created	7/14/2021, 5:02:59 PM by admin	Last Modified	7/14/2021, 7:33:56 PM by admin		
Galaxy Credentials	Galaxy Api Token: An...				

From the **Details** tab, you can edit or delete the organization.

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:



Delete Organization

Are you sure you want to delete:

Honey Dog, Inc.

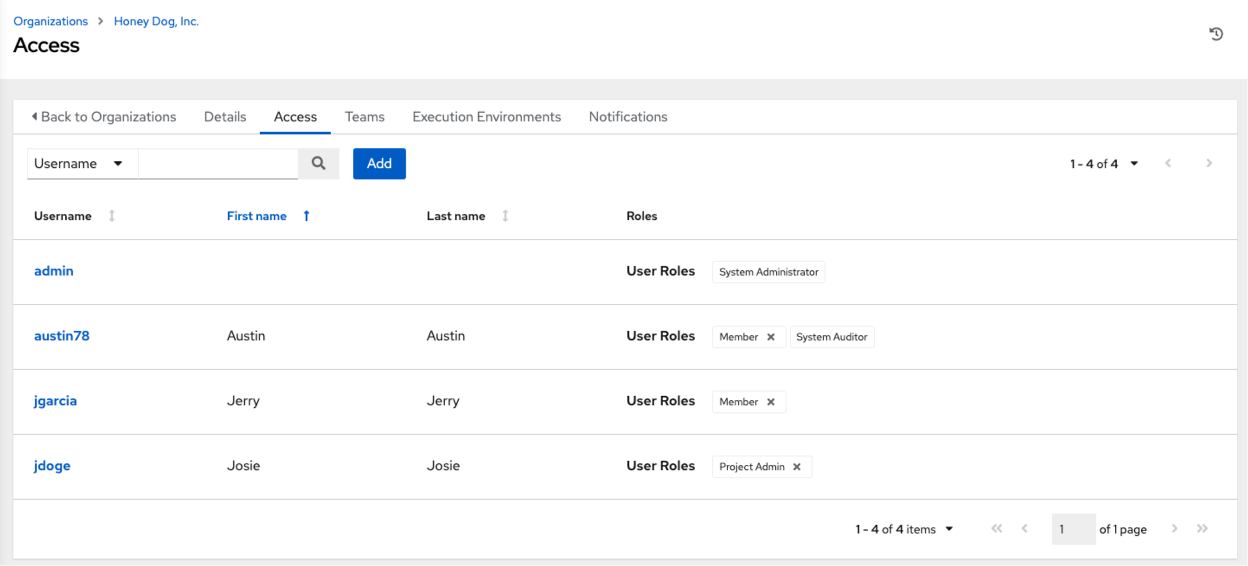
This organization is currently being by other resources. Are you sure you want to delete it?

Teams 4

Inventories 1

7.2 Work with Access

Clicking on **Access** (beside **Details** when viewing your organization), displays all the Users associated with this Organization and their roles.



Username	First name	Last name	Roles	
admin			User Roles	System Administrator
austin78	Austin	Austin	User Roles	Member ✕ System Auditor
jgarcia	Jerry	Jerry	User Roles	Member ✕
jdoge	Josie	Josie	User Roles	Project Admin ✕

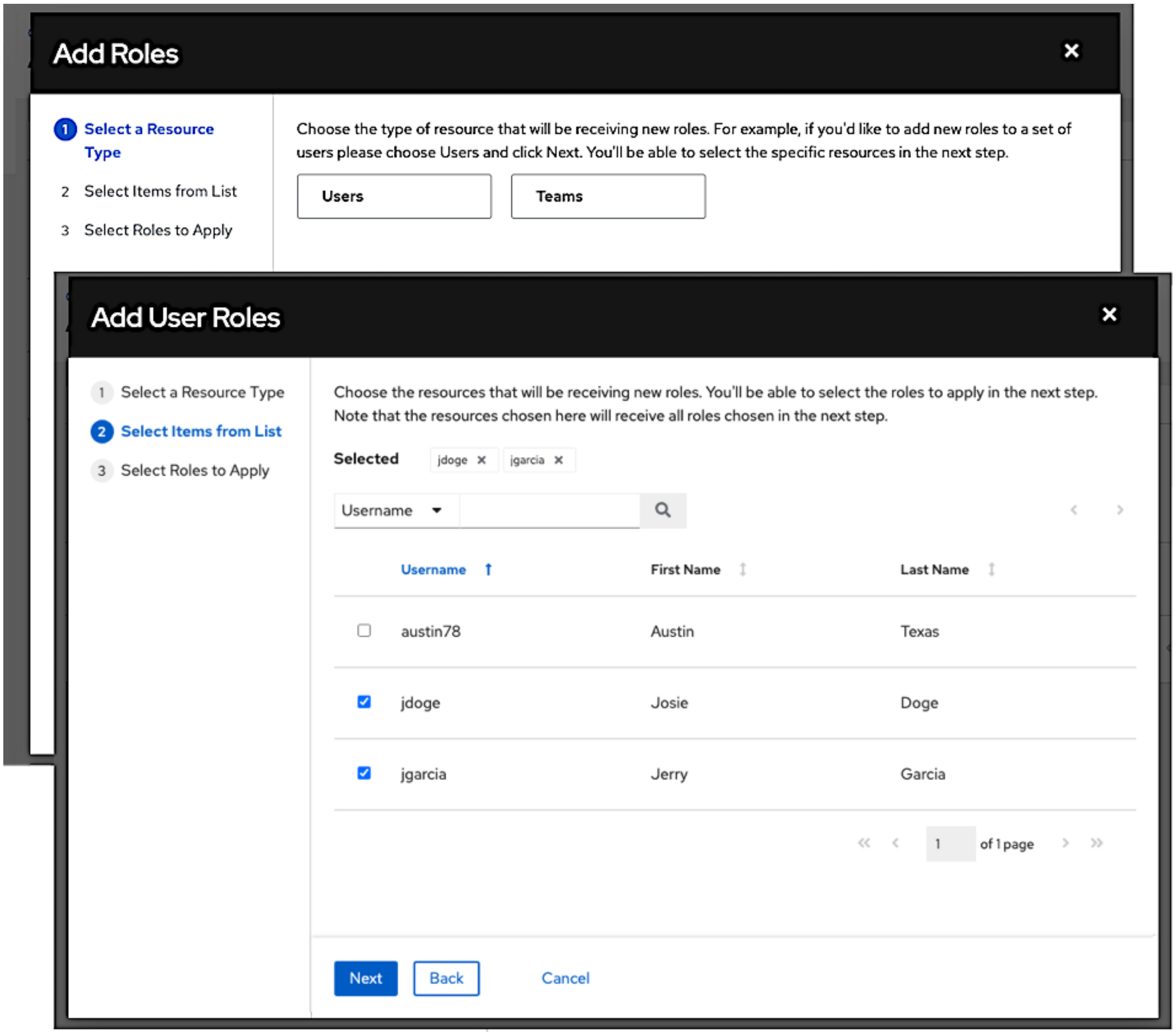
As you can manage the user membership for this Organization here, you can manage user membership on a per-user basis from the Users page by clicking **Users** from the left navigation bar. Organizations have a unique set of roles not described here. You can assign specific users certain levels of permissions within your organization, or allow them to act as an admin for a particular resource. Refer to *Role-Based Access Controls* for more information.

Clicking on a user brings up that user's details, allowing you to review, grant, edit, and remove associated permissions for that user. For more information, refer to *Users*.

7.2.1 Add a User or Team

In order to add a user or team to an organization, the user or team must already be created. See *Create a User* and *Create a Team* for additional detail. To add existing users or team to the Organization:

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**.



Choose the type of resource that will be receiving new roles. For example, if you'd like to add new roles to a set of users please choose Users and click Next. You'll be able to select the specific resources in the next step.

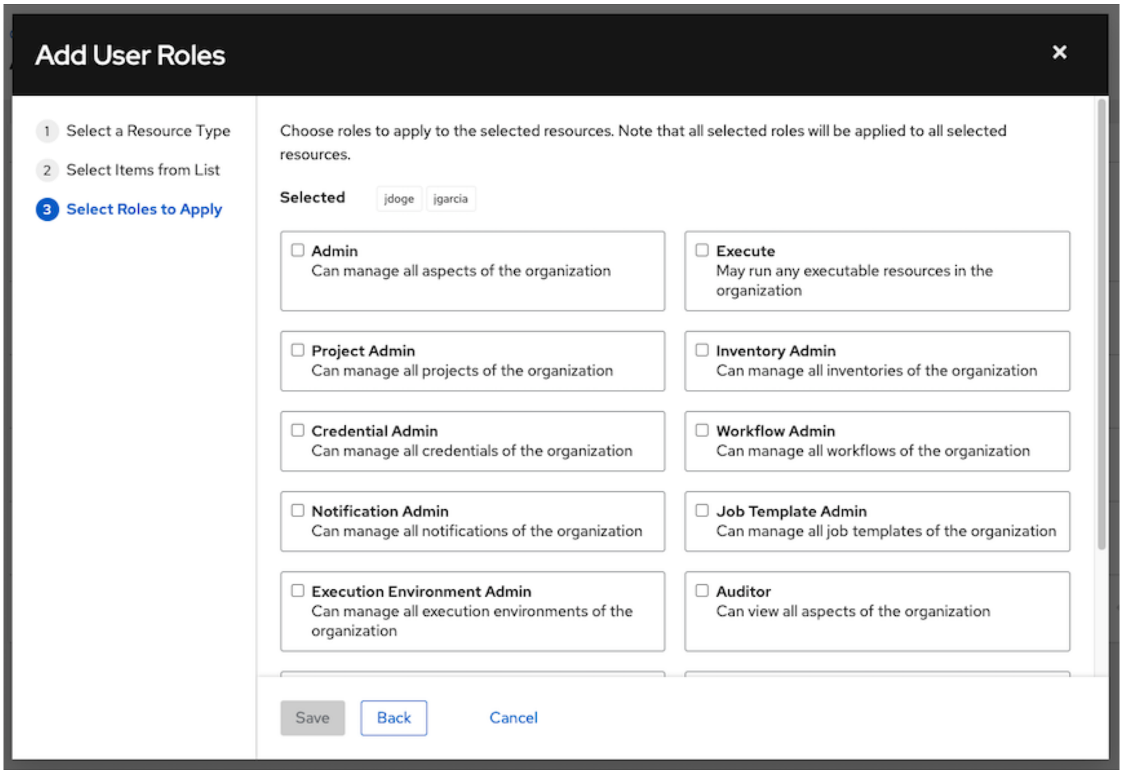
Choose the resources that will be receiving new roles. You'll be able to select the roles to apply in the next step. Note that the resources chosen here will receive all roles chosen in the next step.

Selected jdoge × jgarcia ×

	Username	First Name	Last Name
□	austin78	Austin	Texas
☑	jdoge	Josie	Doge
☑	jgarcia	Jerry	Garcia

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

4. 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.



Add User Roles ×

1 Select a Resource Type
2 Select Items from List
3 Select Roles to Apply

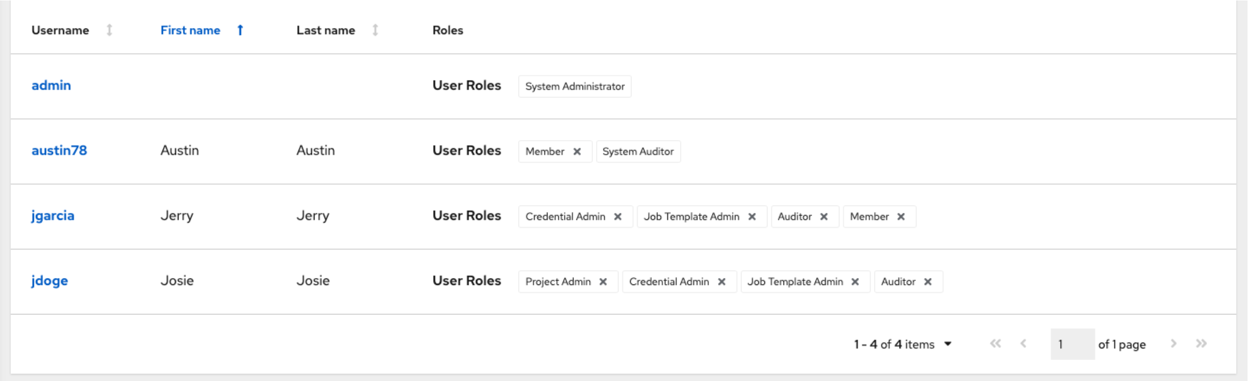
Choose roles to apply to the selected resources. Note that all selected roles will be applied to all selected resources.

Selected jdoge jgarcia

- □ **Admin** Can manage all aspects of the organization
- □ **Execute** May run any executable resources in the organization
- □ **Project Admin** Can manage all projects of the organization
- □ **Inventory Admin** Can manage all inventories of the organization
- □ **Credential Admin** Can manage all credentials of the organization
- □ **Workflow Admin** Can manage all workflows of the organization
- □ **Notification Admin** Can manage all notifications of the organization
- □ **Job Template Admin** Can manage all job templates of the organization
- □ **Execution Environment Admin** Can manage all execution environments of the organization
- □ **Auditor** Can view all aspects of the organization

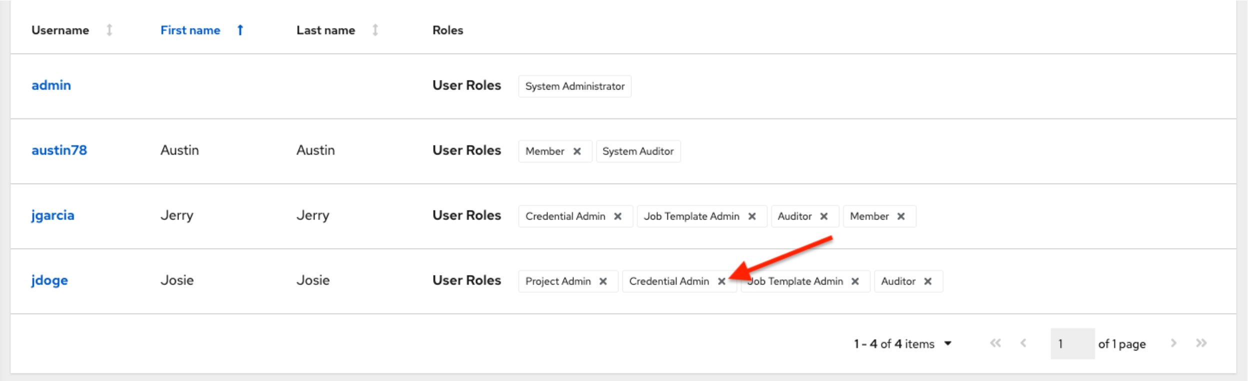
5. Click the **Save** button to apply the roles to the selected user(s) or team(s) and to add them as members.

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



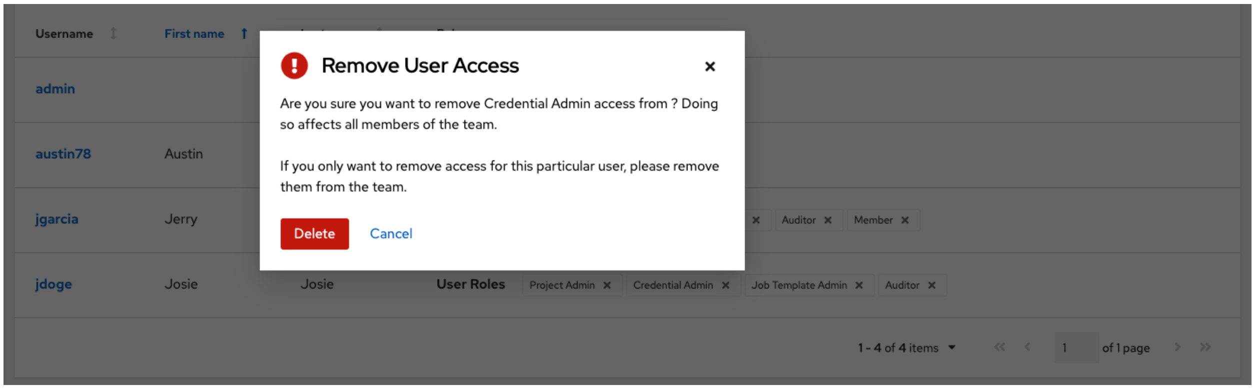
Username	First name	Last name	Roles
admin			**User Roles** System Administrator
austin78	Austin	Austin	**User Roles** Member × System Auditor
jgarcia	Jerry	Jerry	**User Roles** Credential Admin × Job Template Admin × Auditor × Member ×
jdoge	Josie	Josie	**User Roles** Project Admin × Credential Admin × Job Template Admin × Auditor ×

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



Username	First name	Last name	Roles
admin			**User Roles** System Administrator
austin78	Austin	Austin	**User Roles** Member × System Auditor
jgarcia	Jerry	Jerry	**User Roles** Credential Admin × Job Template Admin × Auditor × Member ×
jdoge	Josie	Josie	**User Roles** Project Admin × Credential Admin × Job Template Admin × Auditor ×

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



Remove User Access

Are you sure you want to remove Credential Admin access from ? Doing so affects all members of the team.

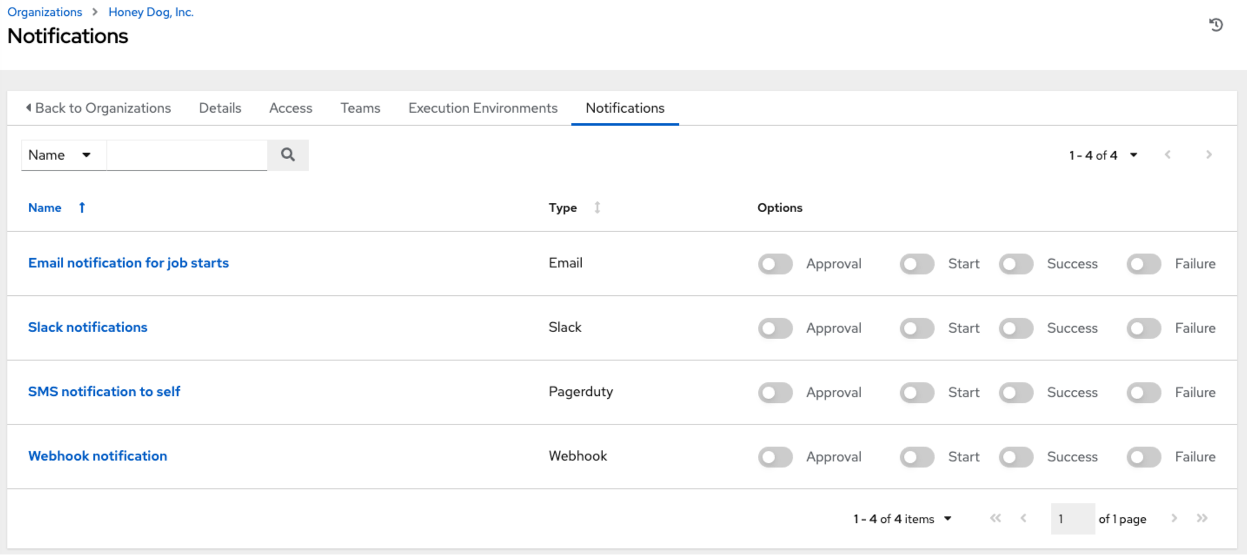
If you only want to remove access for this particular user, please remove them from the team.

Delete Cancel

Note: A user or team with roles associated will retain them even after they have been reassigned to another organization.

7.3 Work with Notifications

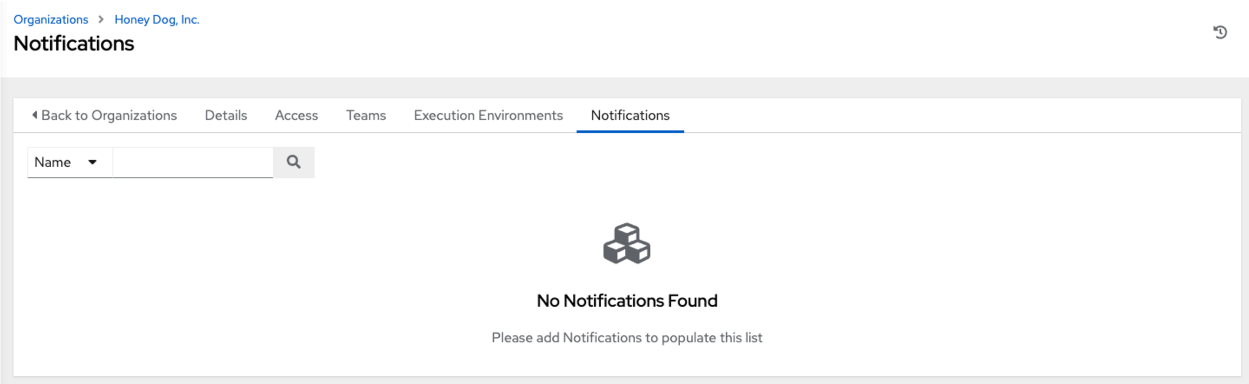
Clicking the **Notifications** tab allows you to review any notification integrations you have setup.



Name	Type	Options			
Email notification for job starts	Email	Approval	Start	Success	Failure
Slack notifications	Slack	Approval	Start	Success	Failure
SMS notification to self	Pagerduty	Approval	Start	Success	Failure
Webhook notification	Webhook	Approval	Start	Success	Failure

Use the toggles to enable or disable the notifications to use with your particular organization. For more detail, see *Enable and Disable Notifications*.

If no notifications have been set up, you must create them from the **Notifications** option on the left navigation bar.



Organizations > Honey Dog, Inc.

Notifications

No Notifications Found

Please add Notifications to populate this list

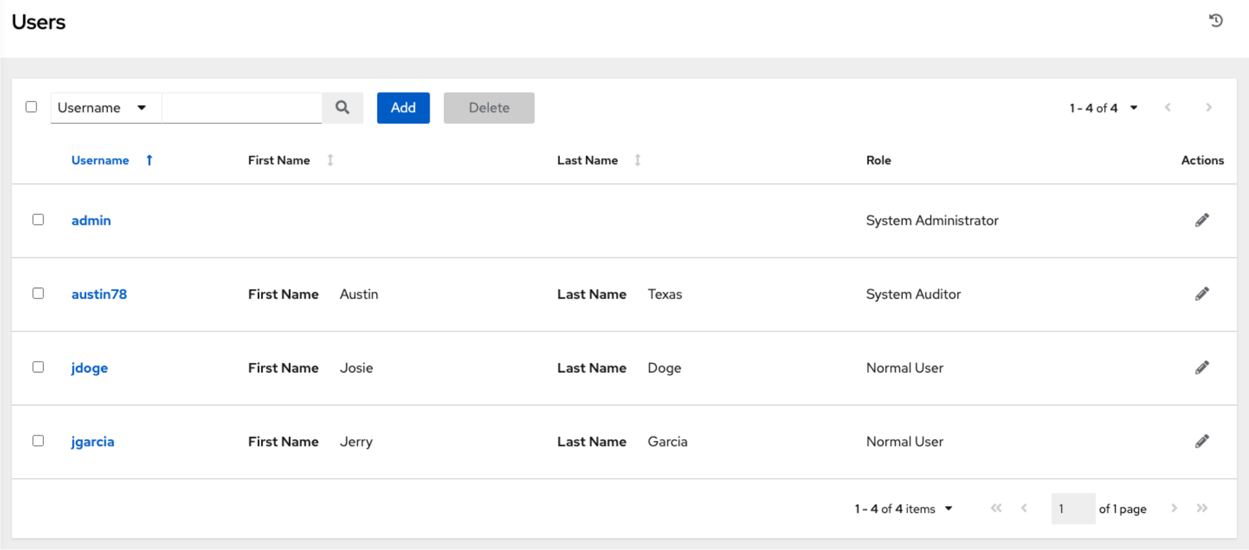
Refer to *Notification Types* for additional details on configuring various notification types.

CHAPTER

EIGHT

USERS

A *User* is someone who has access to automation controller with associated permissions and credentials. Access the Users page by clicking **Users** from the left navigation bar. The User list may be sorted and searched by **Username**, **First Name**, or **Last Name** and click the headers to toggle your sorting preference.



	Username	First Name	Last Name	Role	Actions
□	admin			System Administrator	✎
□	austin78	First Name Austin	Last Name Texas	System Auditor	✎
□	jdoge	First Name Josie	Last Name Doge	Normal User	✎
□	jgarcia	First Name Jerry	Last Name Garcia	Normal User	✎

You can easily view permissions and user type information by looking beside their user name in the User overview screen.

8.1 Create a User

To create a new user:

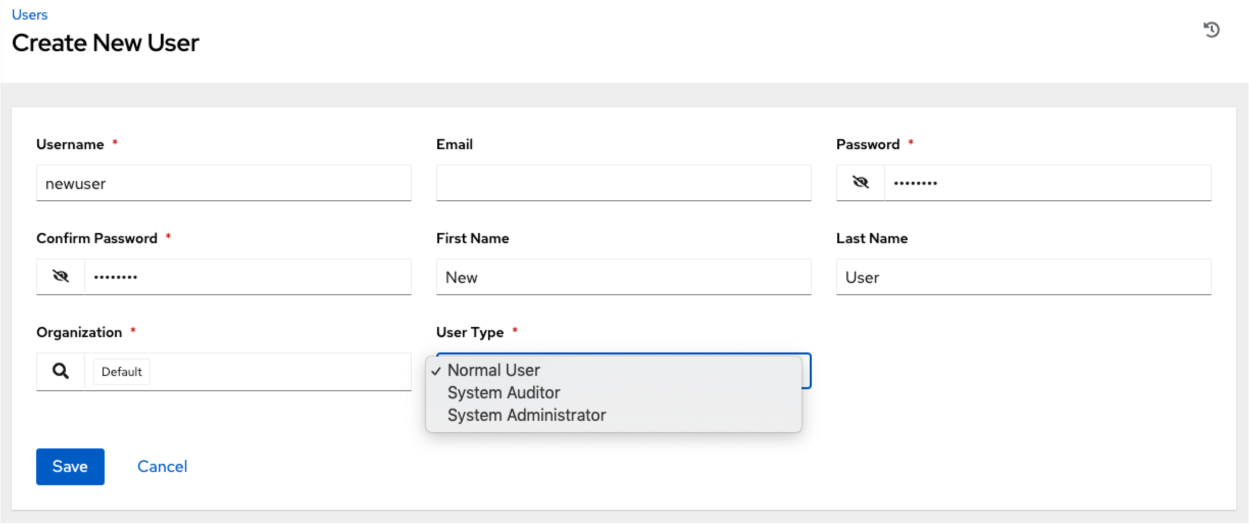
1. Click the **Add** button, which opens the Create User dialog.
2. Enter the appropriate details about your new user. Fields marked with an asterisk (*) are required.

Note: When modifying your own password, log out and log back in again in order for it to take effect.

Three types of Users can be assigned:

- **Normal User**: Normal Users have read and write access limited to the resources (such as inventory, projects, and job templates) for which that user has been granted the appropriate roles and privileges.
- **System Auditor**: Auditors implicitly inherit the read-only capability for all objects within the environment.

- **System Administrator**: A System Administrator (also known as Superuser) has full system administration privileges – with full read and write privileges over the entire installation. A System Administrator is typically responsible for managing all aspects of automation controller and delegating responsibilities for day-to-day work to various Users. Assign with caution!



Users

Create New User

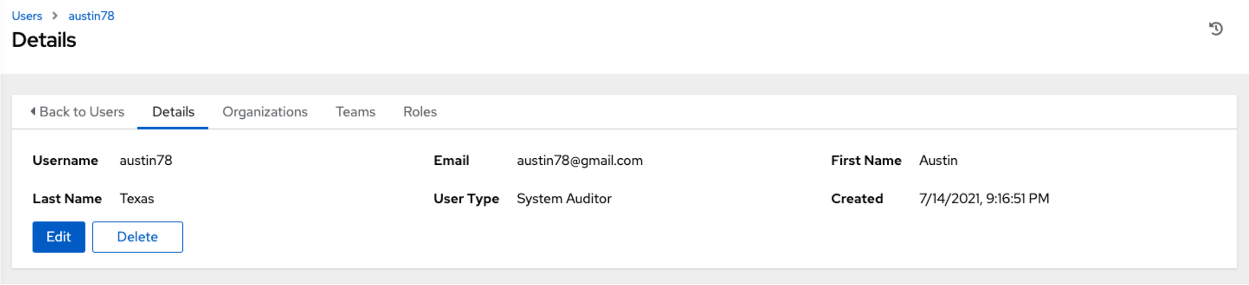
Username *	Email	Password *
newuser		••••••••
Confirm Password *	First Name	Last Name
••••••••	New	User
Organization *	User Type *	
Default	✓ Normal User System Auditor System Administrator	

Note: The initial user (usually "admin") created by the installation process is a Superuser. One Superuser must always exist. To delete the "admin" user account, you must first create another Superuser account.

---

3. Select **Save** when finished.

Once the user is successfully created, the **User** dialog opens for that newly created User.



Users > austin78

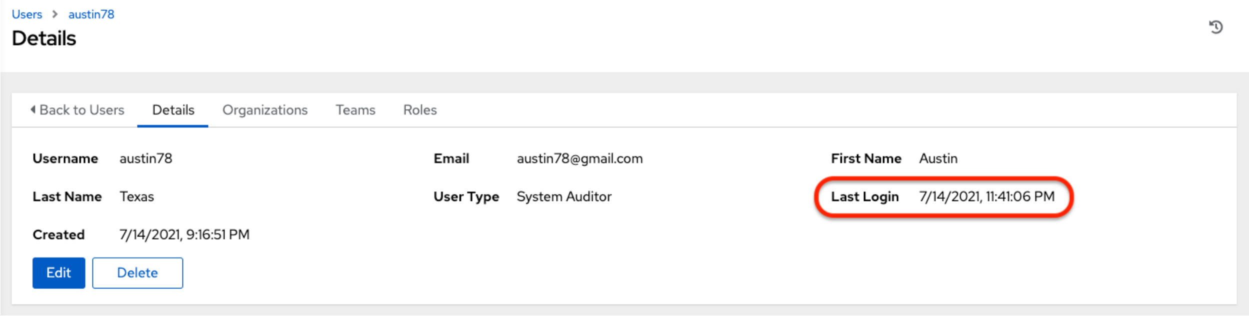
Details

Username	austin78	Email	austin78@gmail.com	First Name	Austin
Last Name	Texas	User Type	System Auditor	Created	7/14/2021, 9:16:51 PM

You may delete the user from its Details screen by clicking **Delete**, or once you exit the details screen, you can delete users from a list of current users. See *Delete a User* for more detail.

The count for the number of users has also been updated, and a new entry for the new user is added to the list of users below the edit form. The same window opens whether you click on the user's name, or the Edit (✎) button beside the user. Here, the User's **Organizations**, **Teams**, and **Permissions**, as well as other user membership details, may be reviewed and modified.

Note: If the user is not a newly-created user, the user's details screen displays the last login activity of that user.

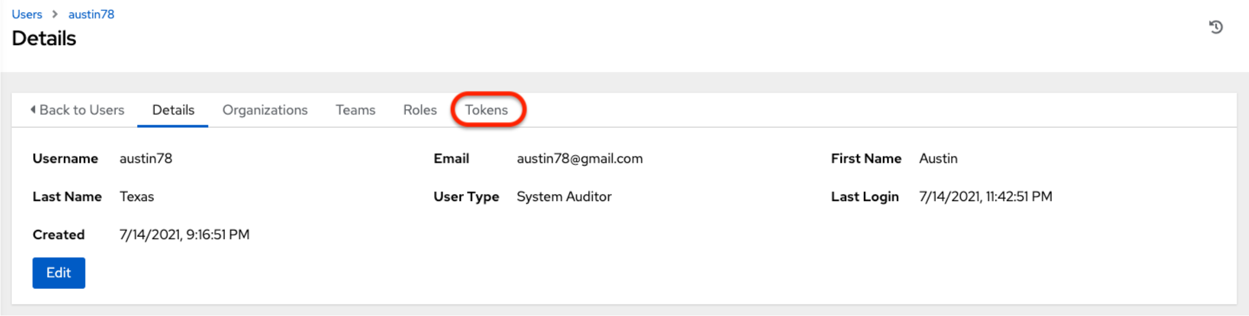


Username austin78 | **Email** austin78@gmail.com | **First Name** Austin

Last Name Texas | **User Type** System Auditor | **Last Login** 7/14/2021, 11:41:06 PM

Created 7/14/2021, 9:16:51 PM

When you log in as yourself, and view the details of your own user profile, you can manage tokens from your user profile. See *Users - Tokens* for more detail.



Username austin78 | **Email** austin78@gmail.com | **First Name** Austin

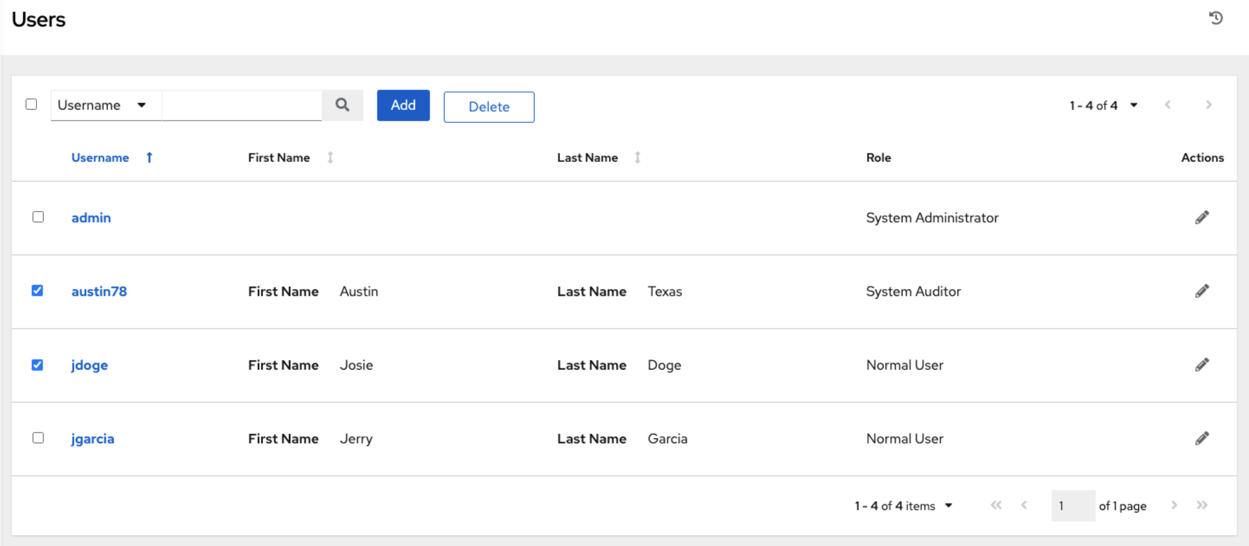
Last Name Texas | **User Type** System Auditor | **Last Login** 7/14/2021, 11:42:51 PM

Created 7/14/2021, 9:16:51 PM

8.2 Delete a User

Before you can delete a user, you must have user permissions. When you delete a user account, the name and email of the user are permanently removed from automation controller.

1. Expand the **Access** menu from the left navigation bar, and click **Users** to display a list of the current users.
2. Select the check box(es) for the user(s) that you want to remove and click **Delete**.

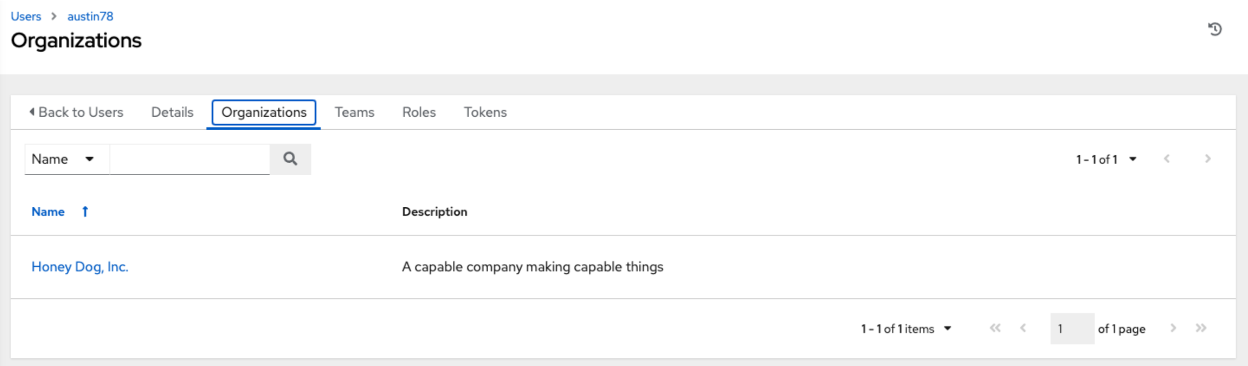


	Username	First Name	Last Name	Role	Actions
□	admin			System Administrator	✎
☑	austin78	First Name Austin	Last Name Texas	System Auditor	✎
☑	jdoge	First Name Josie	Last Name Doge	Normal User	✎
□	jgarcia	First Name Jerry	Last Name Garcia	Normal User	✎

3. Click **Delete** in the confirmation warning message to permanently delete the user.

8.3 Users - Organizations

This displays the list of organizations of which that user is a member. This list may be searched by Organization Name or Description. Organization membership cannot be modified from this display panel.

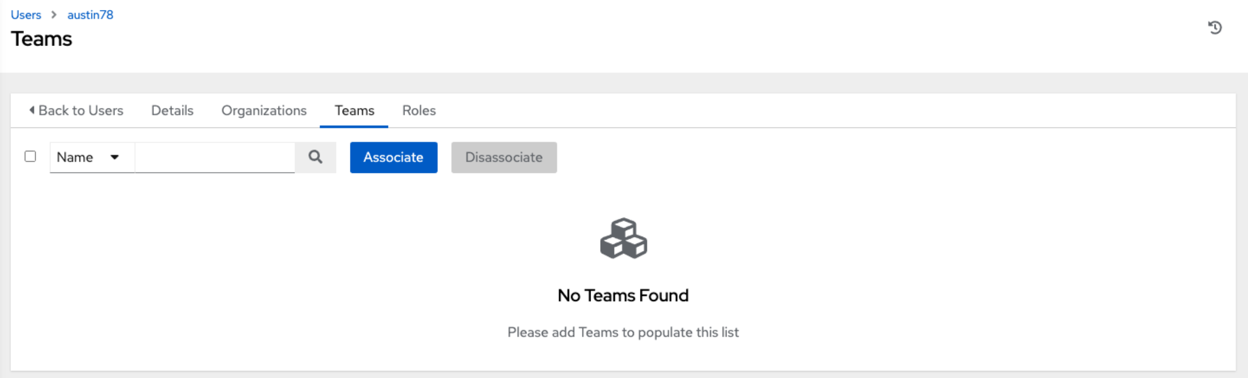


Name	Description
Honey Dog, Inc.	A capable company making capable things

8.4 Users - Teams

This displays the list of teams of which that user is a member. This list may be searched by **Team Name** or **Description**. Team membership cannot be modified from this display panel. For more information, refer to *Teams*.

Until a Team has been created and the user has been assigned to that team, the assigned Teams Details for the User appears blank.



No Teams Found

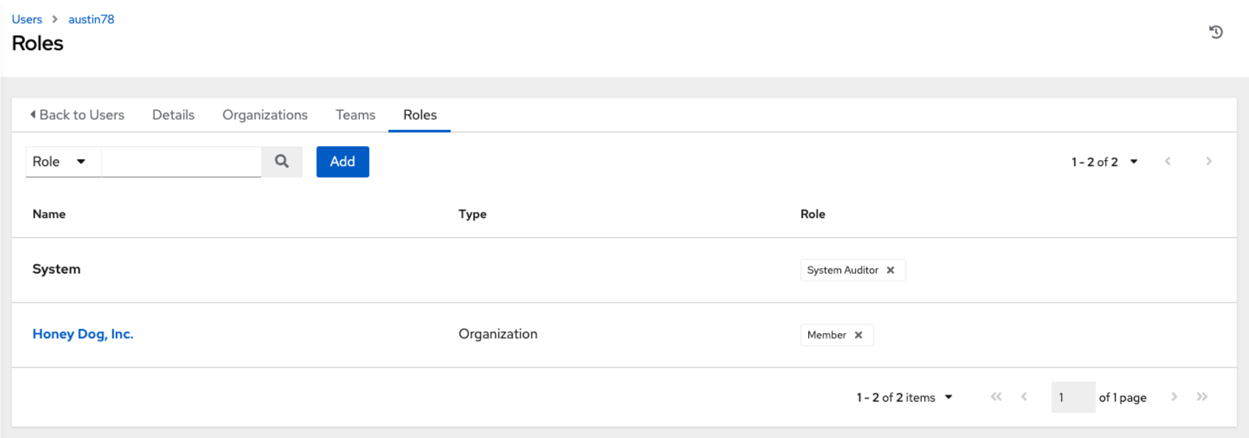
Please add Teams to populate this list

8.5 Users - Permissions

The set of Permissions assigned to this user (role-based access controls) that provide the ability to read, modify, and administer projects, inventories, job templates, and other automation controller elements are Privileges.

Note: It is important to note that the job template administrator may not have access to any inventory, project, or credentials associated with the template. Without access to these, certain fields in the job template aren't editable.

This screen displays a list of the roles that are currently assigned to the selected User and can be sorted and searched by **Name**, **Type**, or **Role**.



Users > austin78

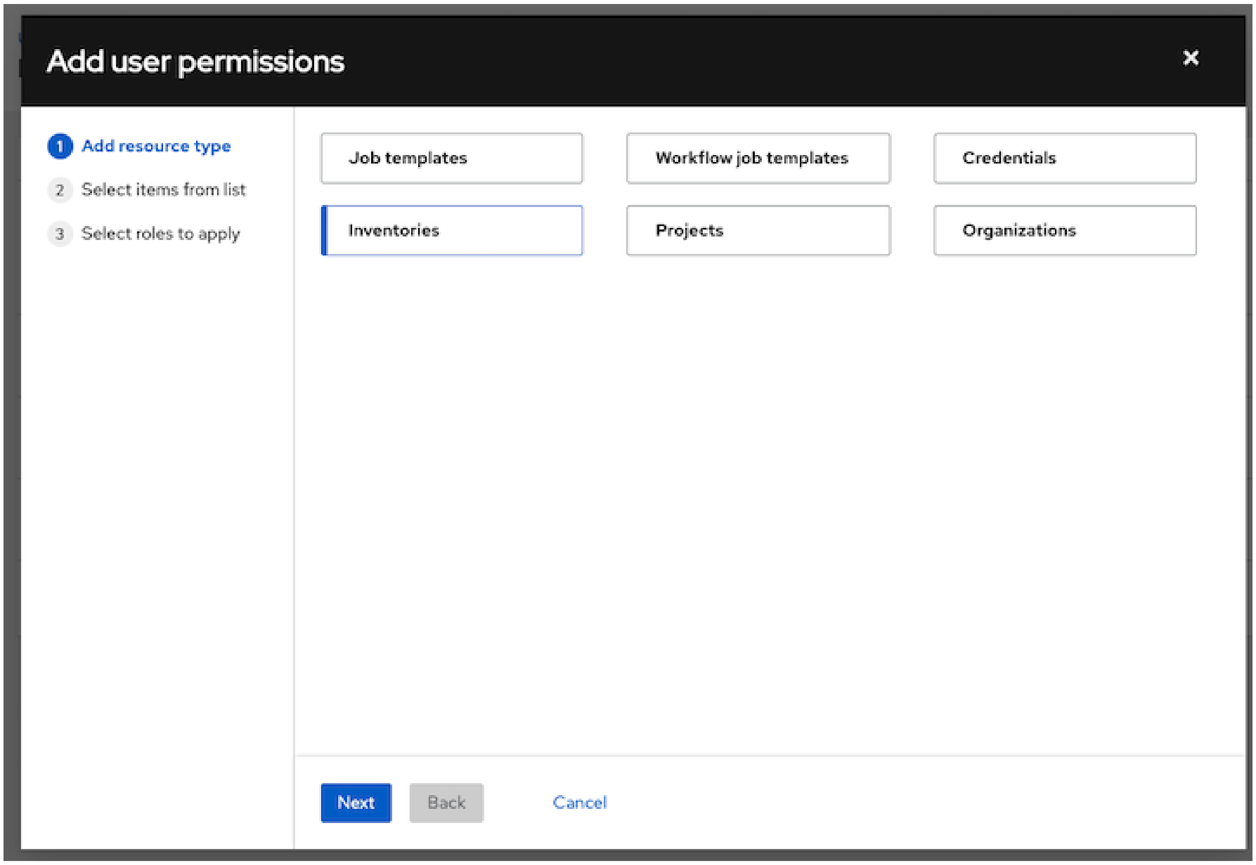
Roles

Name	Type	Role
System		System Auditor ✕
Honey Dog, Inc.	Organization	Member ✕

8.5.1 Add Permissions

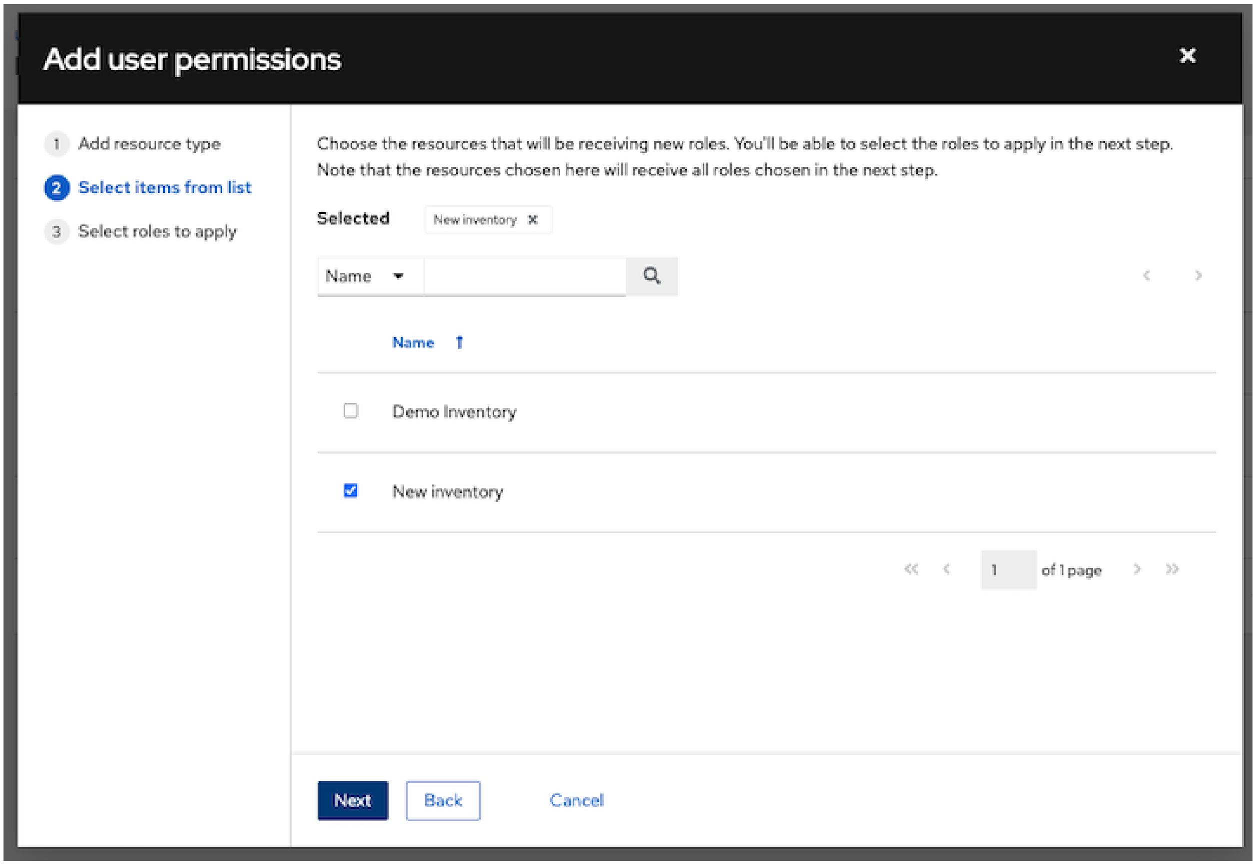
To add permissions to a particular user:

1. Click the **Add** button, which opens the Add Permissions Wizard.



Add user permissions

2. Click to select the object for which the user will have access and click **Next**.
3. Click to select the resource to assign team roles and click **Next**.



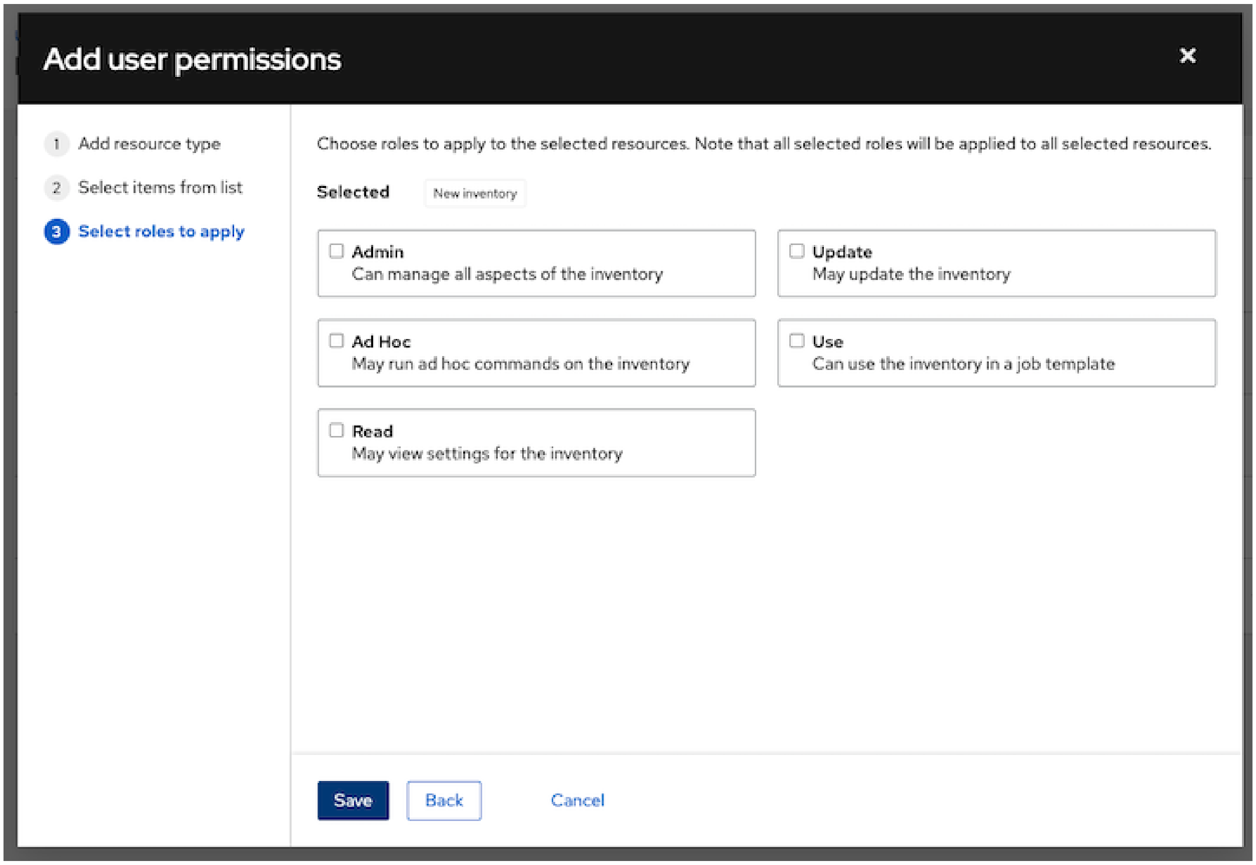
Add user permissions

Choose the resources that will be receiving new roles. You'll be able to select the roles to apply in the next step. Note that the resources chosen here will receive all roles chosen in the next step.

Selected: New inventory

	Name
□	Demo Inventory
☑	New inventory

4. Click the checkbox beside the role to assign that role to your chosen type of resource. Different resources have different options available.



Add user permissions

1 Add resource type

2 Select items from list

3 Select roles to apply

Choose roles to apply to the selected resources. Note that all selected roles will be applied to all selected resources.

Selected New inventory

□ Admin
Can manage all aspects of the inventory

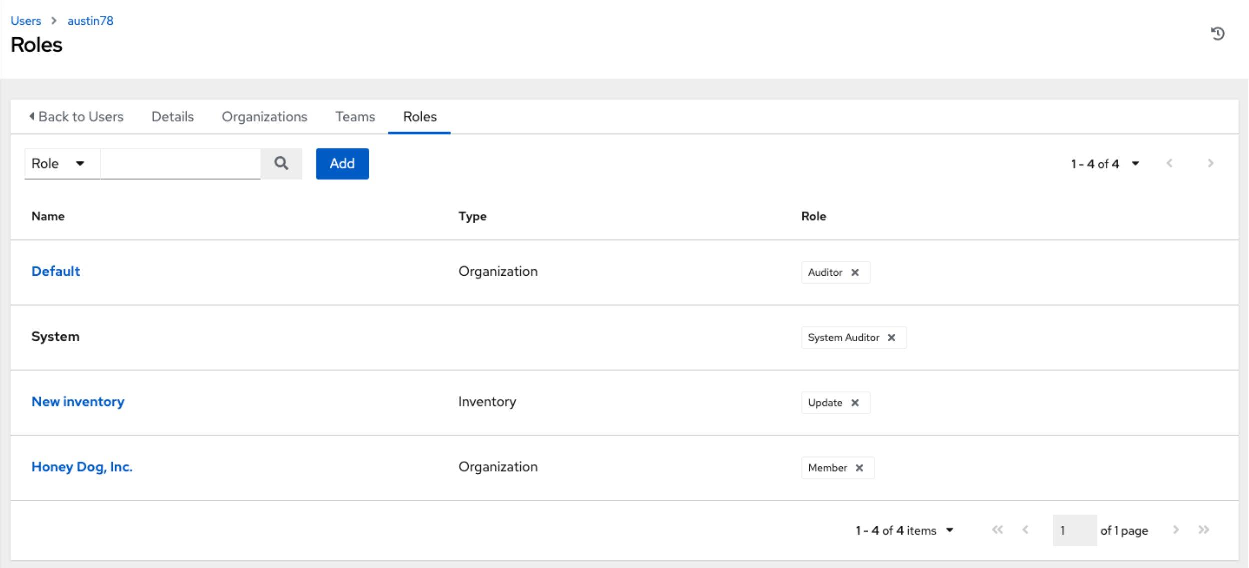
□ Update
May update the inventory

□ Ad Hoc
May run ad hoc commands on the inventory

□ Use
Can use the inventory in a job template

□ Read
May view settings for the inventory

5. Click **Save** when done, and the Add Permissions Wizard closes to display the updated profile for the user with the roles assigned for each selected resource.



Users > austin78

Roles

Name	Type	Role
Default	Organization	Auditor ✕
System		System Auditor ✕
New inventory	Inventory	Update ✕
Honey Dog, Inc.	Organization	Member ✕

To remove Permissions for a particular resource, click the disassociate (x) button next to its resource. This launches a confirmation dialog, asking you to confirm the disassociation.

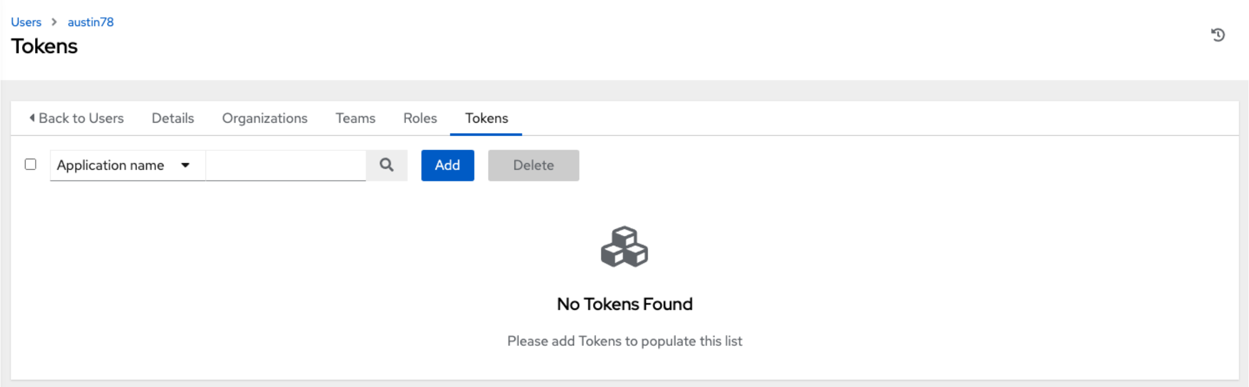
Note: You can also add teams, individual, or multiple users and assign them permissions at the object level (projects, inventories, job templates, and workflow templates) as well. This feature reduces the time for an organization to onboard many users at one time.

8.6 Users - Tokens

The **Tokens** tab will only be present for your user (yourself). Before you add a token for your user, you may want to *create an application* if you want to associate your token to it. You may also create a personal access token (PAT) without associating it with any application. To create a token for your user:

1. If not already selected, click on your user from the Users list view to configure your OAuth 2 tokens.
2. Click the **Tokens** tab from your user's profile.

When no tokens are present, the Tokens screen prompts you to add them:



No Tokens Found

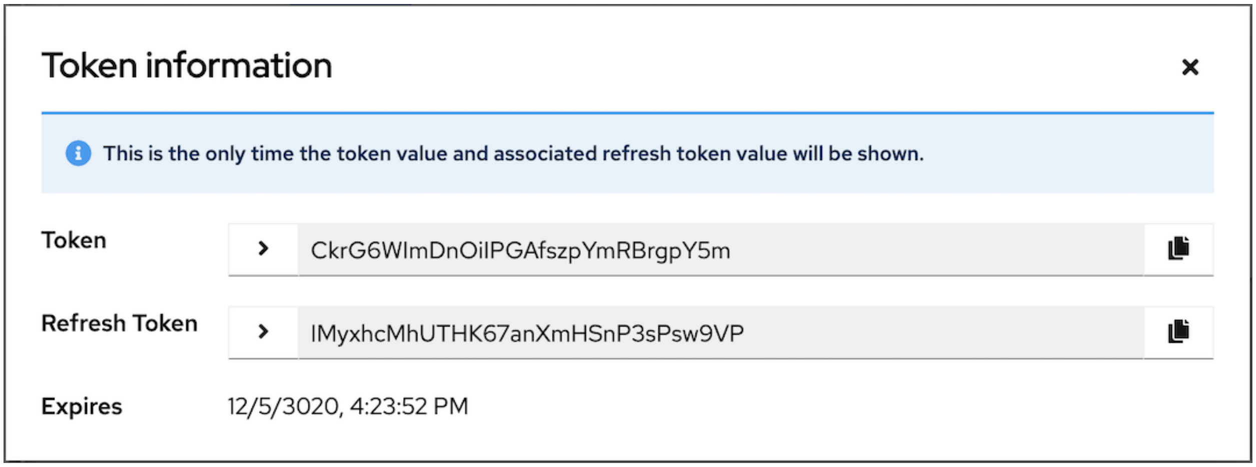
Please add Tokens to populate this list

3. Click the **Add** button, which opens the Create Token window.
4. Enter the following details in Create Token window:

- **Application**: enter the name of the application with which you want to associate your token. Alternatively, you can search for it by clicking the 🔍 button. This opens a separate window that allows you to choose from the available options. Use the Search bar to filter by name if the list is extensive. Leave this field blank if you want to create a Personal Access Token (PAT) that is not linked to any application.
- **Description**: optionally provide a short description for your token.
- **Scope** (required): specify the level of access you want this token to have.

5. When done, click **Save** or **Cancel** to abandon your changes.

After the token is saved, the newly created token for the user displays with the token information and when it expires.



Token information

This is the only time the token value and associated refresh token value will be shown.

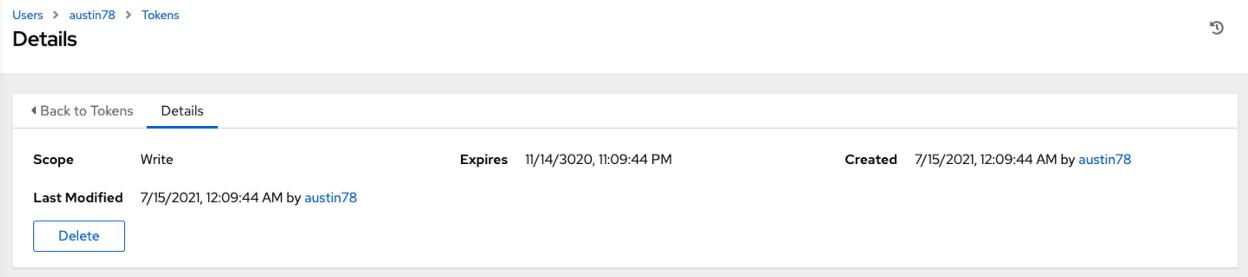
Token: CkrG6WImDnOiIPGAfszpYmRBrgpY5m

Refresh Token: IMyxhcMhUTHK67anXmHSnP3sPsw9VP

Expires: 12/5/3020, 4:23:52 PM

Note: This is the only time the token value and associated refresh token value will ever be shown.

In the user's profile, the application for which it is assigned to and its expiration displays in the token list view.



Users > austin78 > Tokens

Details

Scope: Write

Expires: 11/14/3020, 11:09:44 PM

Created: 7/15/2021, 12:09:44 AM by austin78

Last Modified: 7/15/2021, 12:09:44 AM by austin78

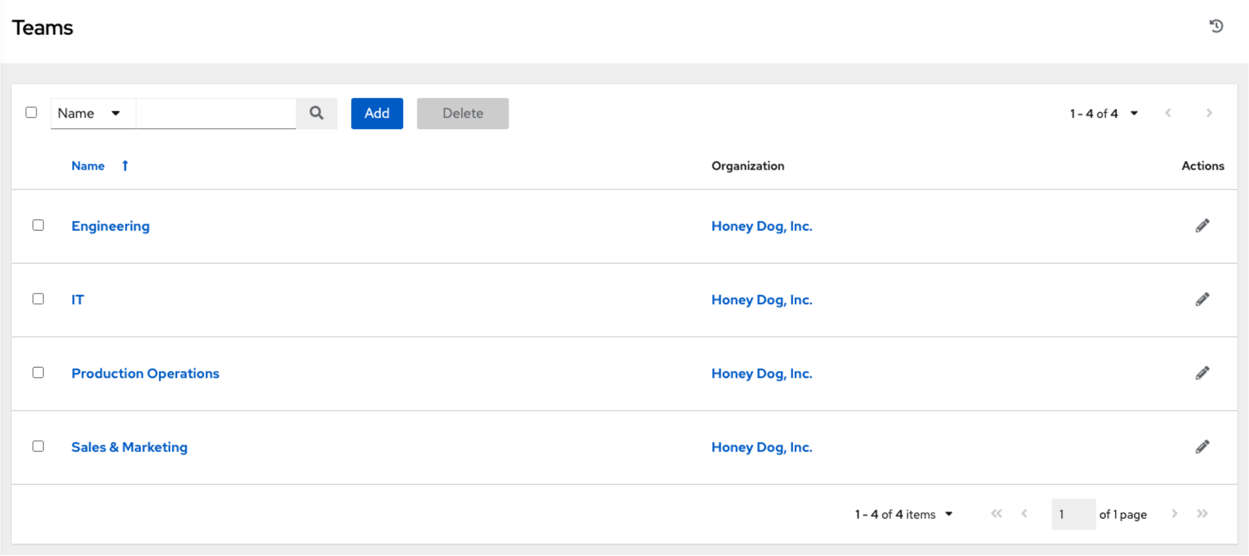
CHAPTER NINE

TEAMS

A *Team* is a subdivision of an organization with associated users, projects, credentials, and permissions. Teams provide a means to implement role-based access control schemes and delegate responsibilities across organizations. For instance, permissions may be granted to a whole Team rather than each user on the Team.

You can create as many Teams of users as make sense for your Organization. Each Team can be assigned permissions, just as with Users. Teams can also scalably assign ownership for Credentials, preventing multiple interface click-throughs to assign the same Credentials to the same user.

Access the Teams page by clicking **Teams** from the left navigation bar. The team list may be sorted and searched by **Name** or **Organization**.



Teams

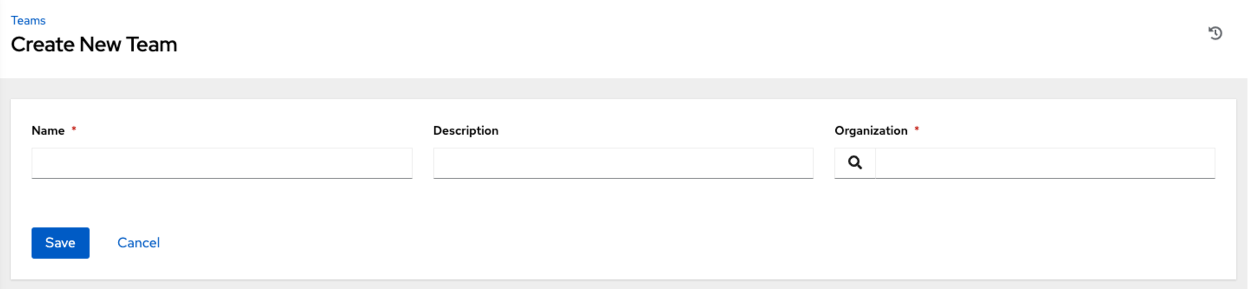
	Name ↑	Organization	Actions
□	Engineering	Honey Dog, Inc.	✎
□	IT	Honey Dog, Inc.	✎
□	Production Operations	Honey Dog, Inc.	✎
□	Sales & Marketing	Honey Dog, Inc.	✎

Clicking the Edit (✎) button next to the list of **Teams** allows you to edit details about the team. You can also review **Users** and **Permissions** associated with this Team.

9.1 Create a Team

To create a new Team:

1. Click the **Add** button.



Teams

Create New Team

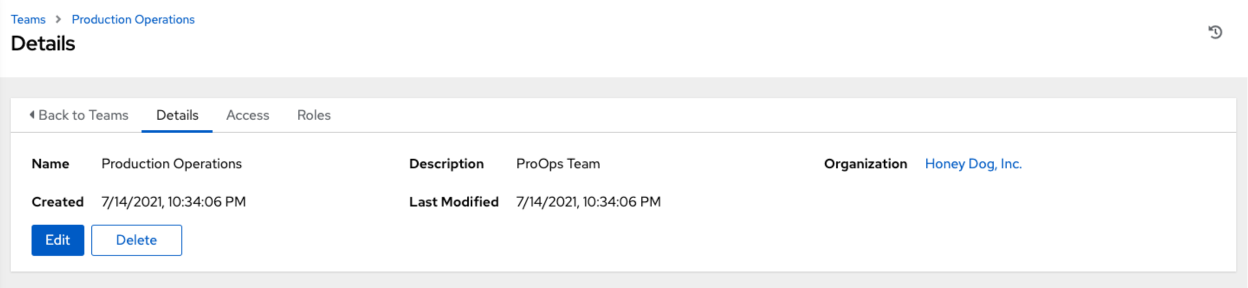
Name *	Description	Organization *

2. Enter the appropriate details into the following fields:

- Name
- Description (optional)
- Organization (Choose from an existing organization)

3. Click **Save**.

Once the Team is successfully created, automation controller opens the **Details** dialog, which also allows you to review and edit your Team information.



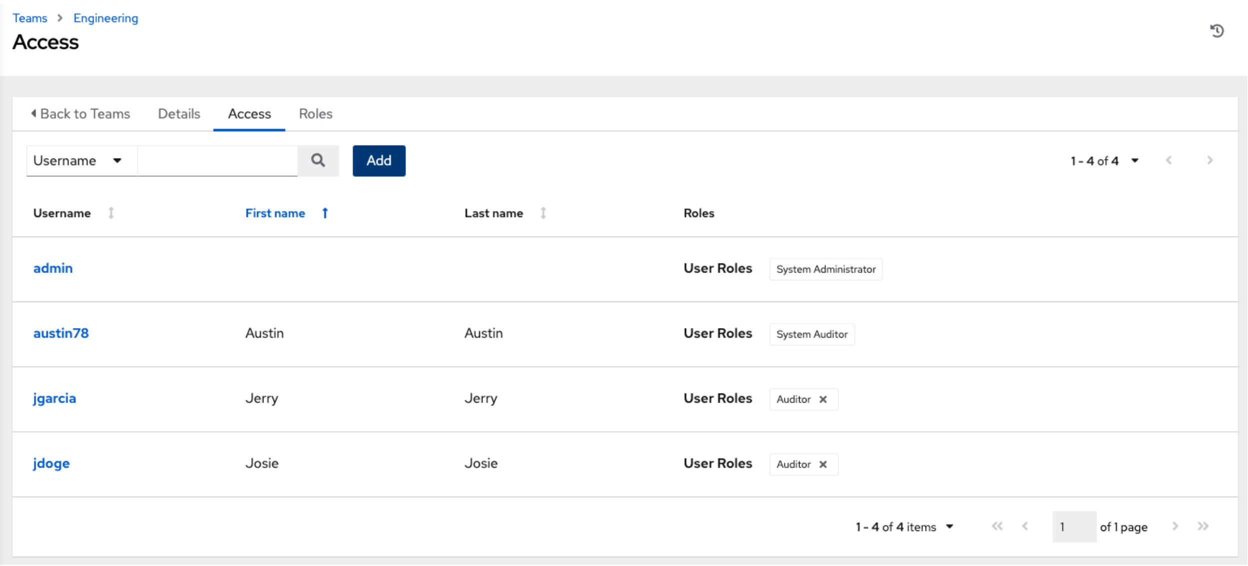
Teams > Production Operations

Details

Name	Production Operations	Description	ProOps Team	Organization	Honey Dog, Inc.
Created	7/14/2021, 10:34:06 PM	Last Modified	7/14/2021, 10:34:06 PM		

9.1.1 Team Access

This tab displays the list of Users that are members of this Team. This list may be searched by **Username**, **First Name**, or **Last Name**. For more information, refer to *Users*.



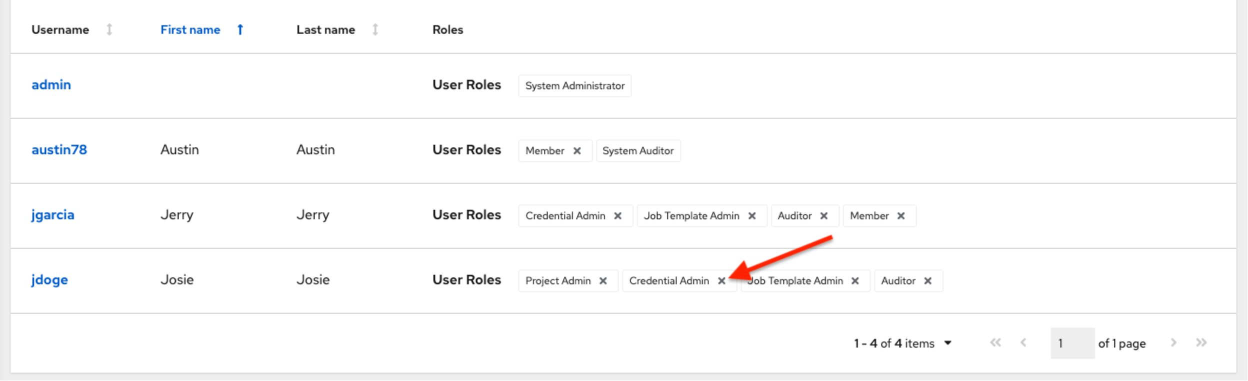
Username	First name	Last name	Roles
admin			User Roles: System Administrator
austin78	Austin	Austin	User Roles: System Auditor
jgarcia	Jerry	Jerry	User Roles: Auditor ✕
jdoge	Josie	Josie	User Roles: Auditor ✕

Add a User

In order to add a user to a team, the user must already be created. Refer to *Create a User* to create a user. Adding a user to a team adds them as a member only, specifying a role for the user on different resources can be done in the **Access** tab . To add existing users to the Team:

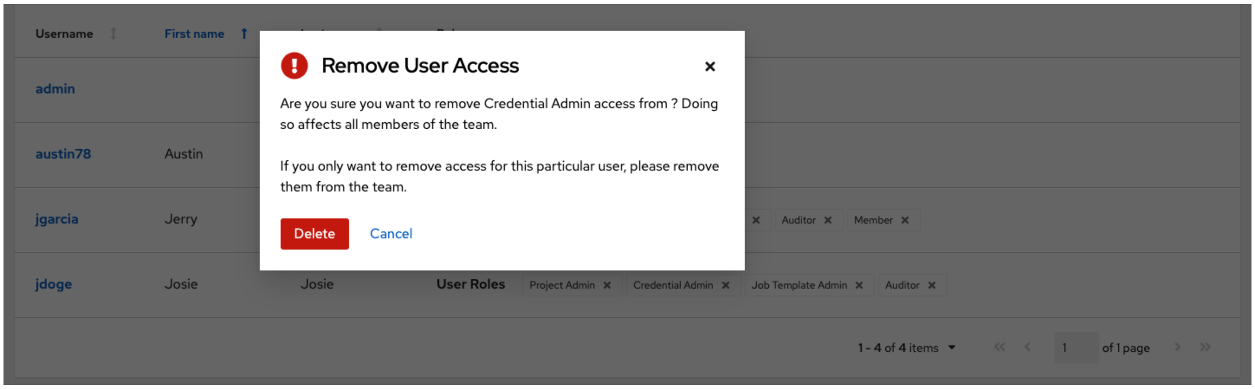
1. In the **Access** tab, click the **Add** button.
2. Follow the prompts to add user(s) and assign them to roles.
3. Click **Save** when done.

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



Username	First name	Last name	Roles
admin			User Roles: System Administrator
austin78	Austin	Austin	User Roles: Member ✕, System Auditor
jgarcia	Jerry	Jerry	User Roles: Credential Admin ✕, Job Template Admin ✕, Auditor ✕, Member ✕
jdoge	Josie	Josie	User Roles: Project Admin ✕, Credential Admin ✕, Job Template Admin ✕, Auditor ✕

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



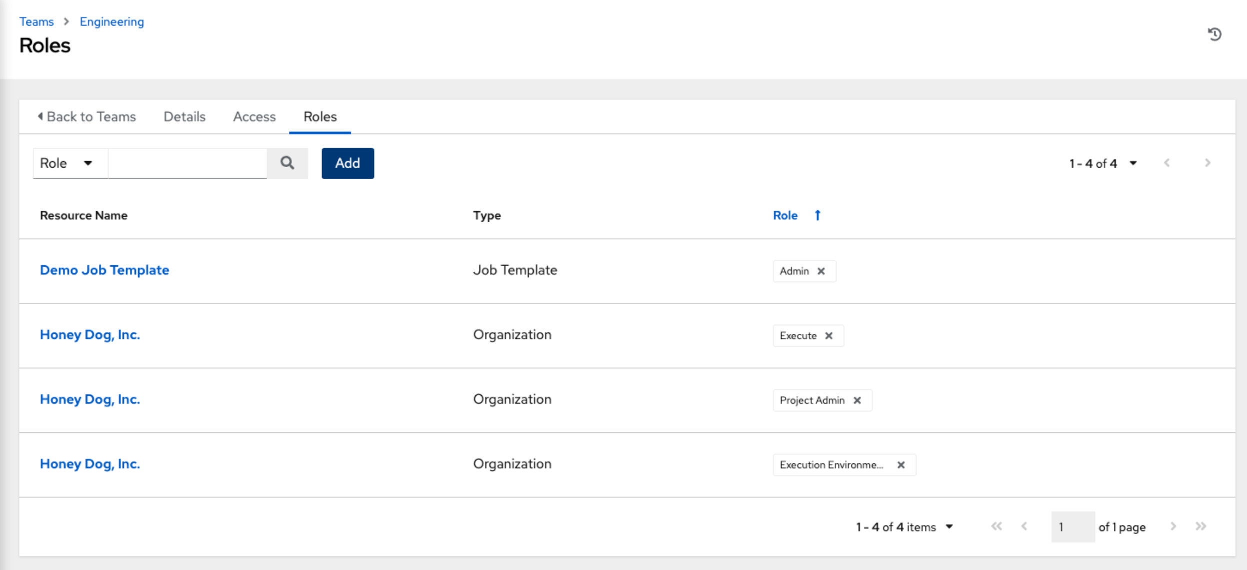
Remove User Access

Are you sure you want to remove Credential Admin access from ? Doing so affects all members of the team.

If you only want to remove access for this particular user, please remove them from the team.

9.1.2 Team Roles

Selecting the **Roles** view displays a list of the permissions that are currently available for this Team. The permissions list may be sorted and searched by **Resource Name**, **Type**, or **Role**.



Resource Name	Type	Role
Demo Job Template	Job Template	Admin
Honey Dog, Inc.	Organization	Execute
Honey Dog, Inc.	Organization	Project Admin
Honey Dog, Inc.	Organization	Execution Environme...

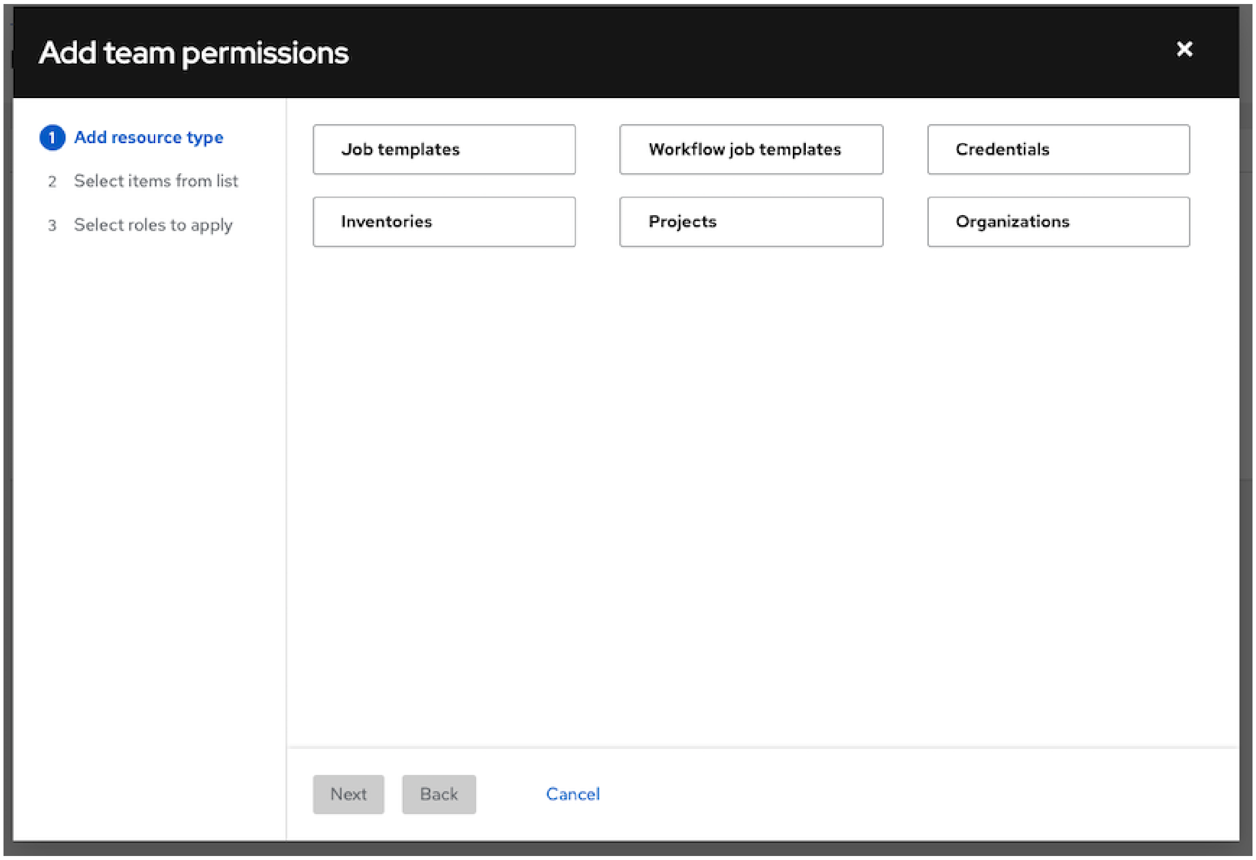
The set of privileges assigned to Teams that provide the ability to read, modify, and administer projects, inventories, and other automation controller elements are permissions. By default, the Team is given the "read" permission (also called a role).

Permissions must be set explicitly via an Inventory, Project, Job Template, or within the Organization view.

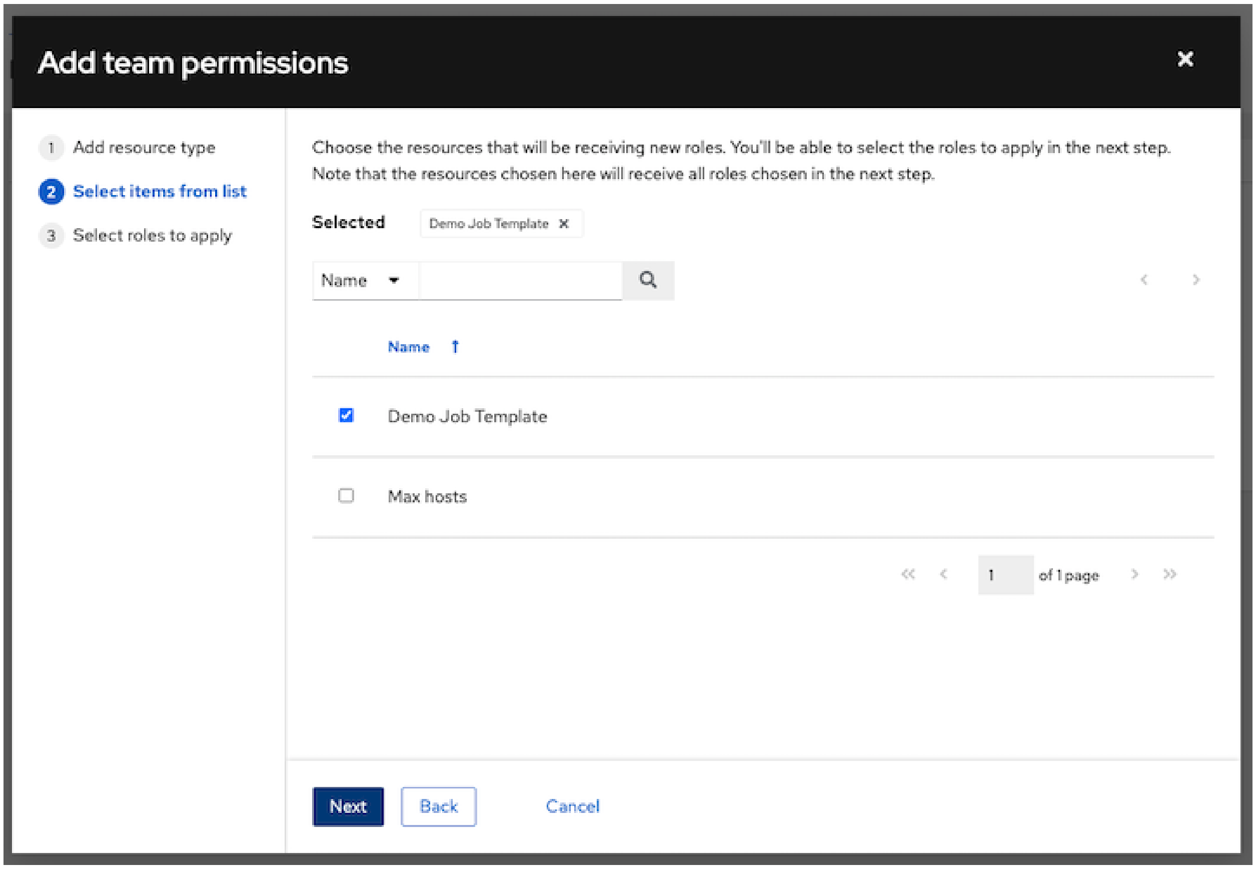
Add Team Permissions

To add permissions to a Team:

1. Click the **Add** button, which opens the Add Permissions Wizard.



2. Click to select the object for which the team will have access and click **Next**.
3. Click to select the resource to assign team roles and click **Next**.



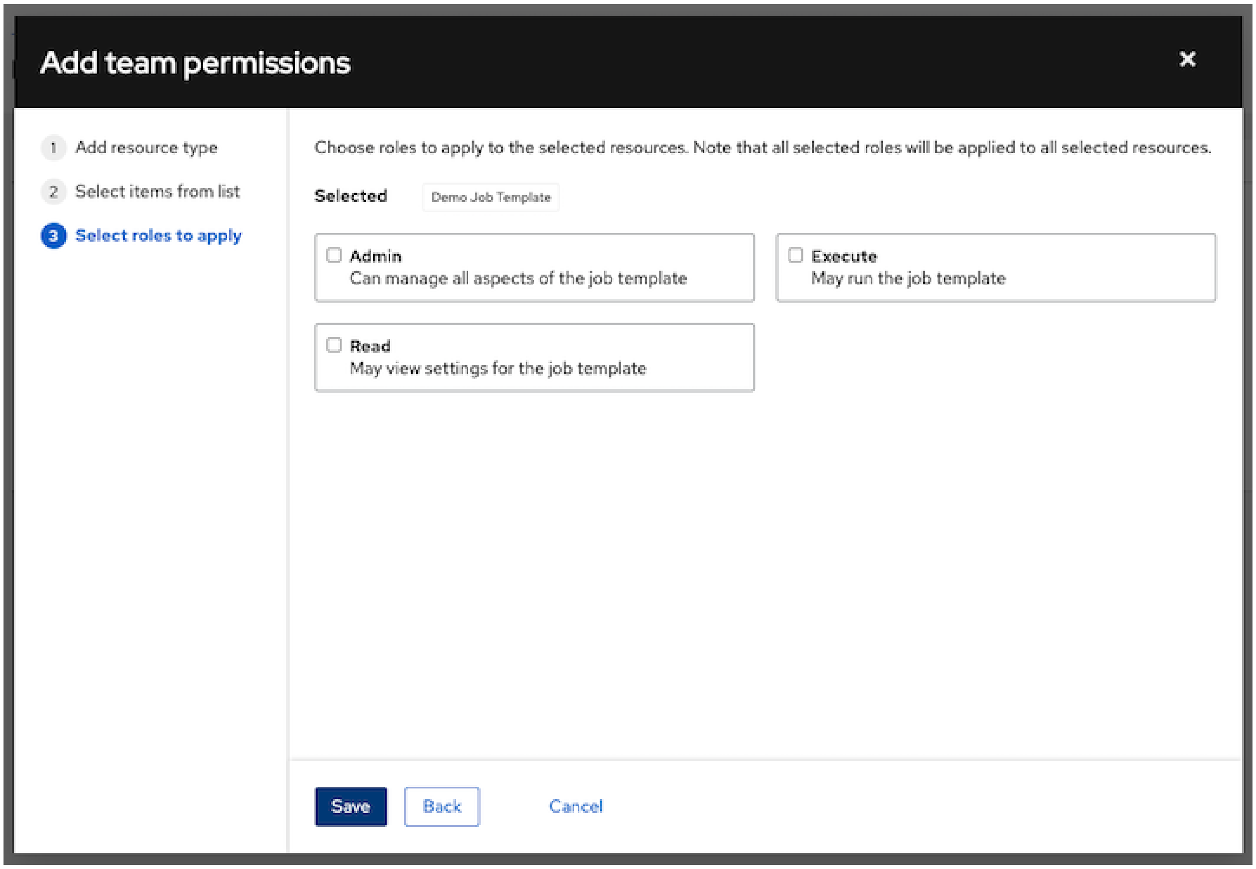
Choose the resources that will be receiving new roles. You'll be able to select the roles to apply in the next step. Note that the resources chosen here will receive all roles chosen in the next step.

Selected: Demo Job Template

	Name
☑	Demo Job Template
□	Max hosts

1 of 1 page

4. Click the checkbox beside the role to assign that role to your chosen type of resource. Different resources have different options available.



Add team permissions ×

1 Add resource type

2 Select items from list

3 Select roles to apply

Choose roles to apply to the selected resources. Note that all selected roles will be applied to all selected resources.

Selected Demo Job Template

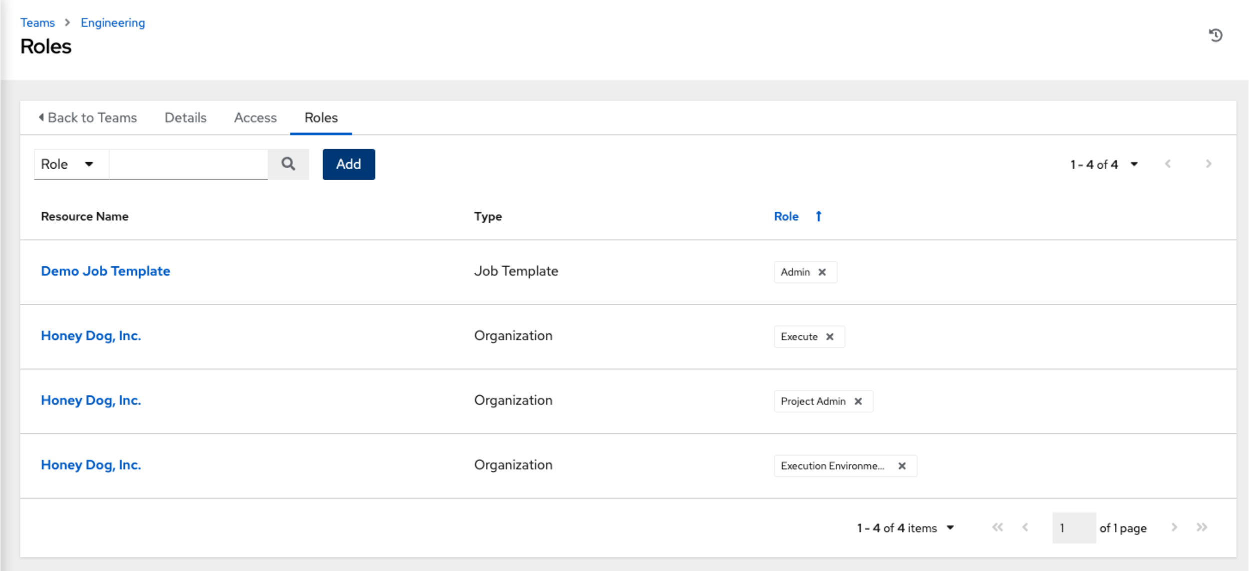
☐ **Admin** Can manage all aspects of the job template

☐ **Execute** May run the job template

☐ **Read** May view settings for the job template

Save Back Cancel

5. Click **Save** when done, and the Add Permissions Wizard closes to display the updated profile for the user with the roles assigned for each selected resource.



Teams > Engineering

Roles

◂ Back to Teams Details Access Roles

Role ▾ Add 1 - 4 of 4 ▾

Resource Name	Type	Role ↑
Demo Job Template	Job Template	Admin ×
Honey Dog, Inc.	Organization	Execute ×
Honey Dog, Inc.	Organization	Project Admin ×
Honey Dog, Inc.	Organization	Execution Environme... ×

1 - 4 of 4 items ▾ 1 of 1 page

To remove Permissions for a particular resource, click the disassociate (x) button next to its resource. This launches a confirmation dialog, asking you to confirm the disassociation.

Note: You can also add teams, individual, or multiple users and assign them permissions at the object level (projects, inventories, job templates, and workflow templates) as well. This feature reduces the time for an organization to onboard many users at one time.

CHAPTER

TEN

CREDENTIALS

Credentials are utilized for authentication when launching Jobs against machines, synchronizing with inventory sources, and importing project content from a version control system.

You can grant users and teams the ability to use these credentials, without actually exposing the credential to the user. If you have a user move to a different team or leave the organization, you don't have to re-key all of your systems just because that credential was available in the automation controller.

Note: The automation controller encrypts passwords and key information in the database and never makes secret information visible via the API. See Automation Controller Administration Guide for details.

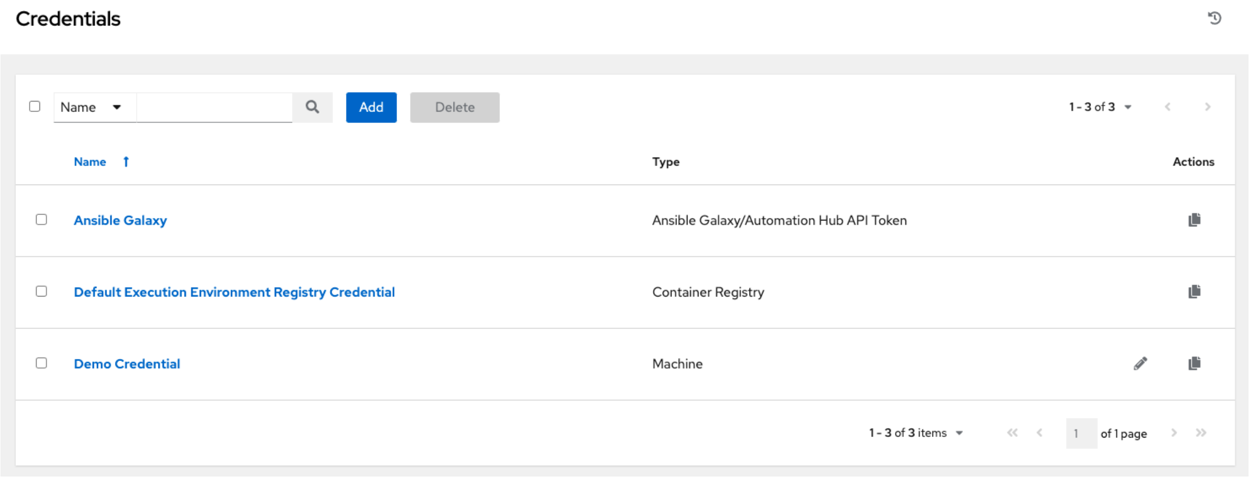
10.1 Understanding How Credentials Work

The automation controller uses SSH to connect to remote hosts (or the Windows equivalent). In order to pass the key from the automation controller to SSH, the key must be decrypted before it can be written a named pipe. The automation controller then uses that pipe to send the key to SSH (so that it is never written to disk).

If passwords are used, the automation controller handles those by responding directly to the password prompt and decrypting the password before writing it to the prompt.

10.2 Getting Started with Credentials

Click **Credentials** from the left navigation bar to access the Credentials page. The Credentials page displays a searchable list of all available Credentials and can be sorted by **Name**.

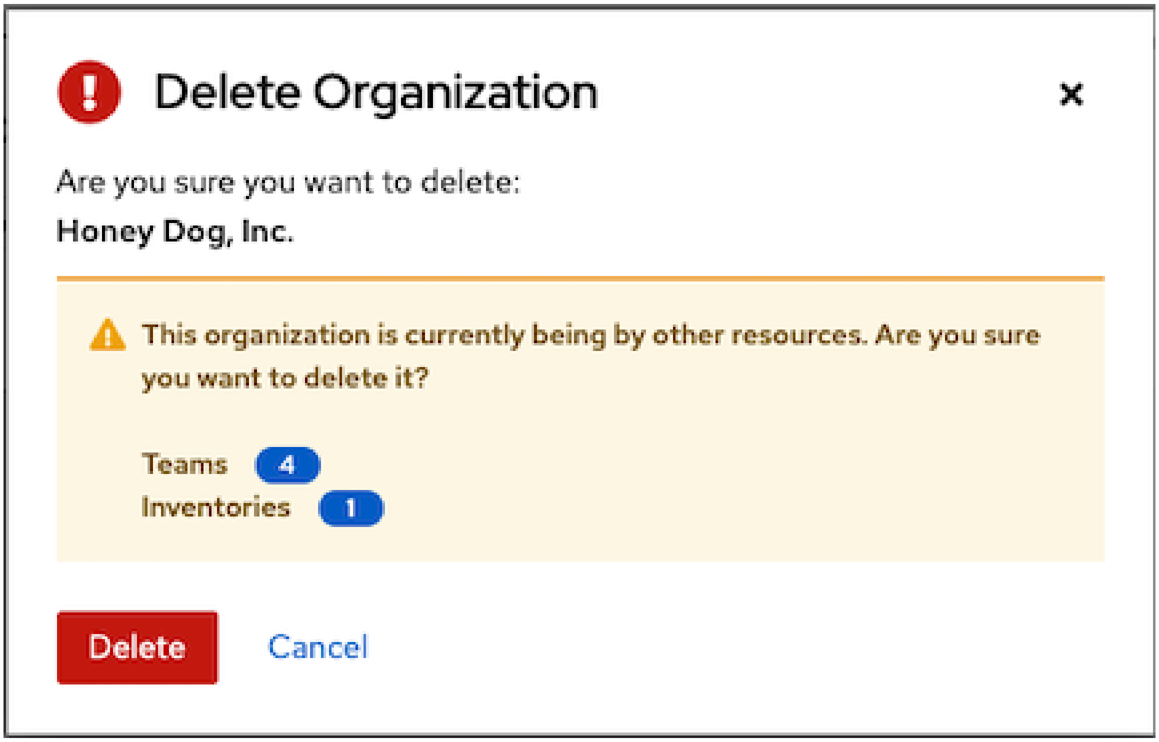


Credentials

	Name ↑	Type	Actions
□	Ansible Galaxy	Ansible Galaxy/Automation Hub API Token	
□	Default Execution Environment Registry Credential	Container Registry	
□	Demo Credential	Machine	

Credentials added to a Team are made available to all members of the Team, whereas credentials added to a User are only available to that specific User by default.

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:



Delete Organization

Are you sure you want to delete:

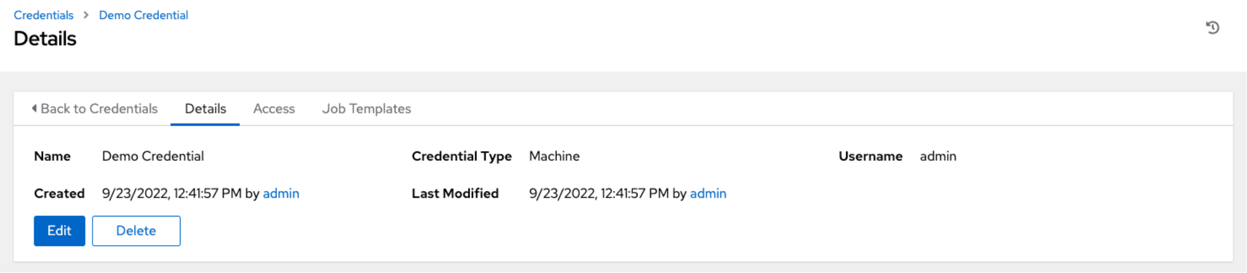
Honey Dog, Inc.

This organization is currently being by other resources. Are you sure you want to delete it?

Teams	4
Inventories	1

To help you get started, a Demo Credential has been created for your use.

Clicking on the link for the **Demo Credential** takes you to the **Details** view of this Credential.



Credentials > Demo Credential

Details

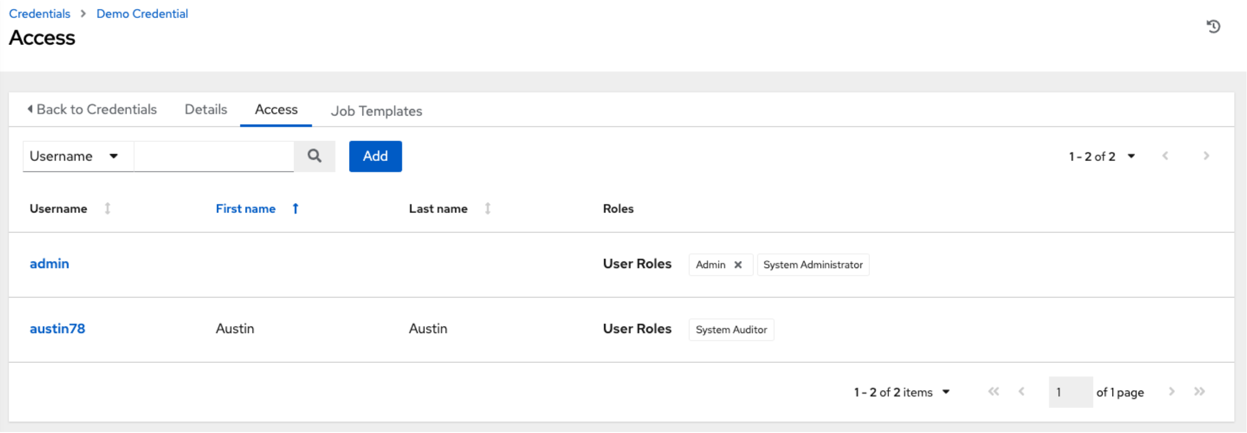
◂ Back to Credentials | Details | Access | Job Templates

Name Demo Credential
Credential Type Machine
Username admin

Created 9/23/2022, 12:41:57 PM by admin
Last Modified 9/23/2022, 12:41:57 PM by admin

Edit | Delete

Clicking the **Access** tab shows you users and teams associated with this Credential and their granted roles (owner, admin, auditor, etc.)



Credentials > Demo Credential

Access

◂ Back to Credentials | Details | Access | Job Templates

Username ▾ | Add | 1 - 2 of 2 ▾

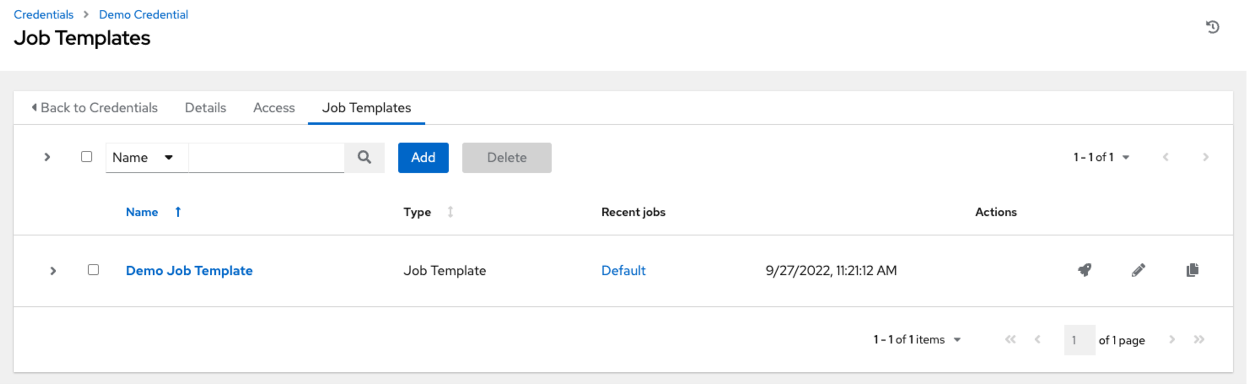
Username	First name	Last name	Roles
admin			User Roles: Admin ✕, System Administrator
austin78	Austin	Austin	User Roles: System Auditor

1 - 2 of 2 items ▾ | 1 of 1 page

Note: A credential with roles associated will retain them even after the credential has been reassigned to another organization.

You can click the **Add** button to assign this **Demo Credential** to additional users. If no users exist, add them from the **Users** menu and refer to the *Users* section for further detail.

Clicking the **Job Templates** tab shows you the job templates associated with this Credential and which jobs recently ran using this particular credential.



Credentials > Demo Credential

Job Templates

◂ Back to Credentials | Details | Access | Job Templates

□ Name ▾ | Add | Delete | 1 - 1 of 1 ▾

Name	Type	Recent jobs		Actions
Demo Job Template	Job Template	Default	9/27/2022, 11:21:12 AM	

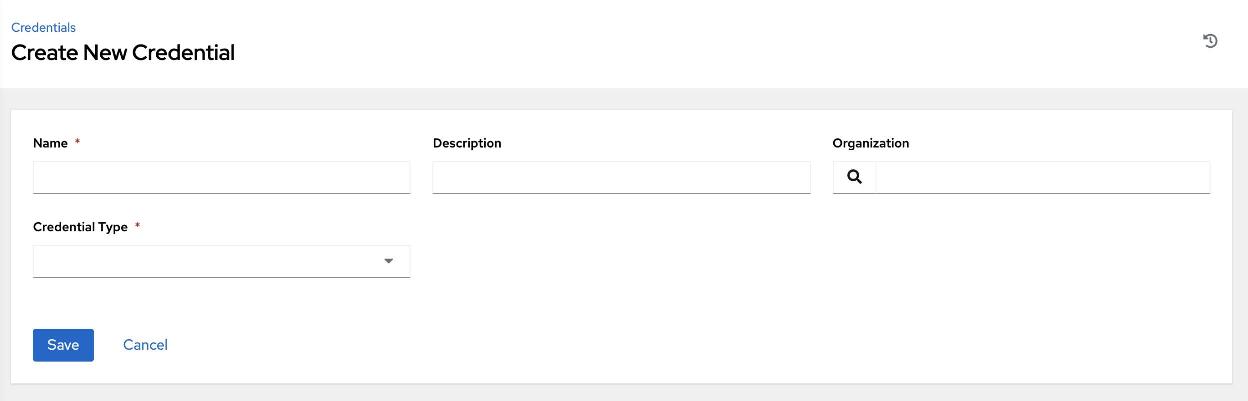
1 - 1 of 1 items ▾ | 1 of 1 page

You can click the **Add** button to assign this **Demo Credential** to additional job templates. Refer to the *Job Templates* section for further detail on creating a new job template.

10.3 Add a New Credential

To create a new credential:

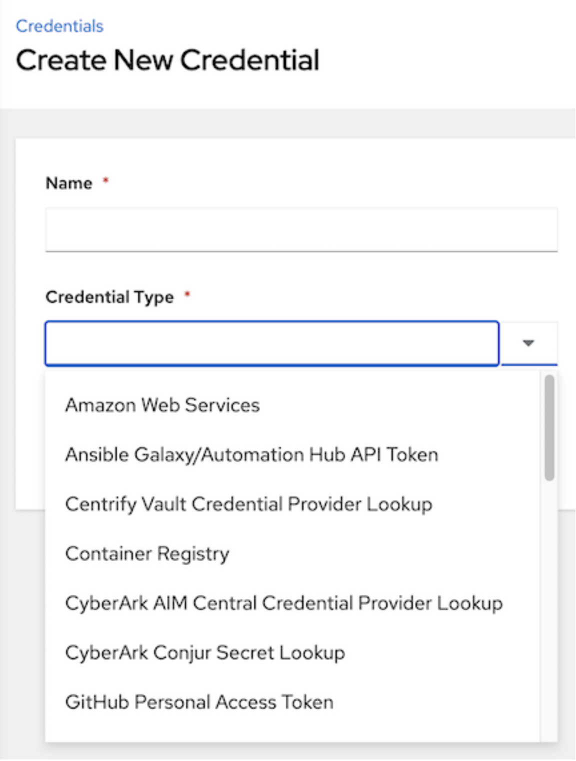
1. Click the **Add** button from the **Credentials** screen.



2. Enter the name for your new credential in the **Name** field.
3. Optionally enter a description and enter or select the name of the organization with which the credential is associated.

Note: A credential with a set of permissions associated with one organization will remain even after the credential is reassigned to another organization.

4. Enter or select the credential type you want to create.


Credentials
Create New Credential
Name *
Credential Type *
Amazon Web Services
Ansible Galaxy/Automation Hub API Token
Centrify Vault Credential Provider Lookup
Container Registry
CyberArk AIM Central Credential Provider Lookup
CyberArk Conjur Secret Lookup
GitHub Personal Access Token

5. Enter the appropriate details depending on the type of credential selected, as described in the next section, *Credential Types*.
6. Click **Save** when done.

10.4 Credential Types

The following credential types are supported with the automation controller:

- *Amazon Web Services*
- *Ansible Galaxy/Automation Hub API Token*
- *Centrify Vault Credential Provider Lookup*
- *Container Registry*
- *CyberArk AIM Credential Provider Lookup*
- *CyberArk Conjur Secret Lookup*
- *GitHub Personal Access Token*
- *GitLab Personal Access Token*
- *Google Compute Engine*
- *GPG Public Key*
- *HashiCorp Vault Secret Lookup*
- *HashiCorp Vault Signed SSH*
- *Insights*
- *Machine*
- *Microsoft Azure Key Vault*
- *Microsoft Azure Resource Manager*
- *Network*
- *OpenShift or Kubernetes API Bearer Token*
- *OpenStack*
- *Red Hat Ansible Automation Platform*
- *Red Hat Satellite 6*
- *Red Hat Virtualization*
- *Source Control*
- *Thycotic DevOps Secrets Vault*
- *Thycotic Secret Server*
- *Vault*
- *VMware vCenter*

The credential types associated with Centrify, CyberArk, HashiCorp Vault, Microsoft Azure Key Management System

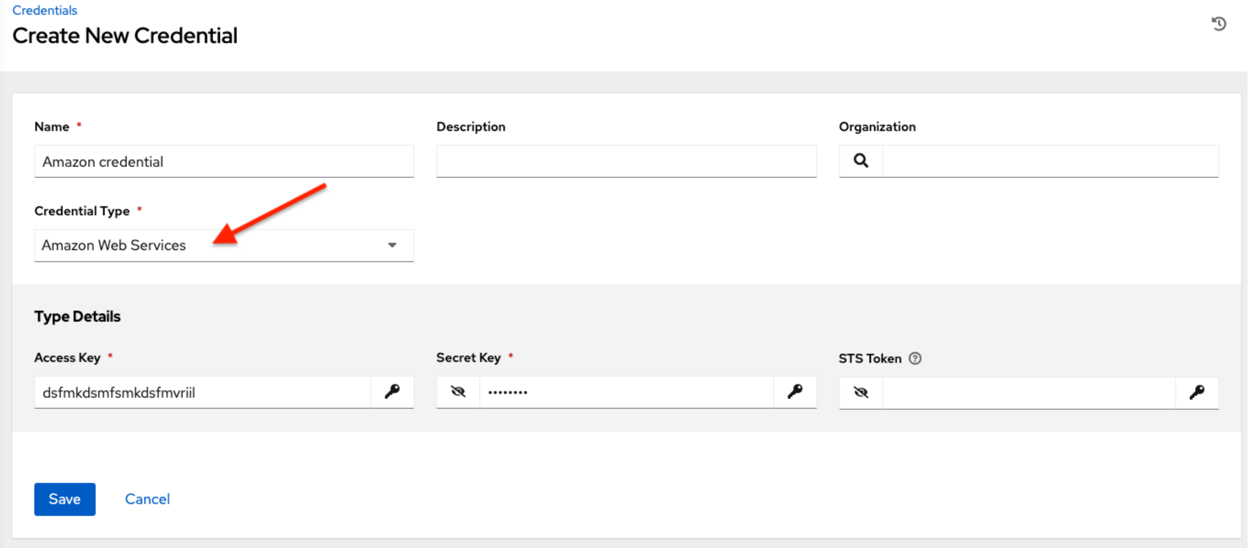
(KMS), and Thycotic are part of the credential plugins capability that allows an external system to lookup your secrets information. See the *Secret Management System* section for further detail.

10.4.1 Amazon Web Services

Selecting this credential type enables synchronization of cloud inventory with Amazon Web Services.

The automation controller uses the following environment variables for AWS credentials and are fields prompted in the user interface:

```
AWS_ACCESS_KEY_ID
AWS_SECRET_ACCESS_KEY
AWS_SECURITY_TOKEN
```



Credentials

Create New Credential

Name * Amazon credential

Description

Organization

Credential Type * Amazon Web Services

Type Details

Access Key * dsfmkdsmfsmkdsfmvriil

Secret Key * ••••••••

STS Token

Traditional Amazon Web Services credentials consist of the AWS **Access Key** and **Secret Key**.

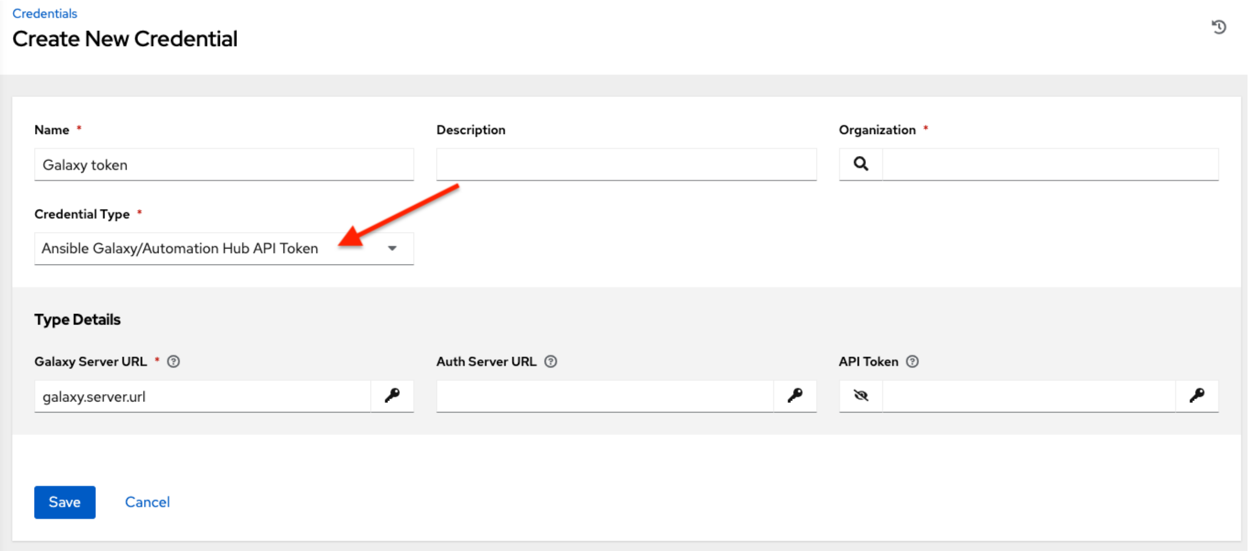
The automation controller provides support for EC2 STS tokens (sometimes referred to as IAM STS credentials). Security Token Service (STS) is a web service that enables you to request temporary, limited-privilege credentials for AWS Identity and Access Management (IAM) users. To learn more about the IAM/EC2 STS Token, refer to: http://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_temp.html

Note: If the value of your tags in EC2 contain booleans (yes/no/true/false), you must remember to quote them.

Warning: To use implicit IAM role credentials, do not attach AWS cloud credentials in the automation controller when relying on IAM roles to access the AWS API. While it may seem to make sense to attach your AWS cloud credential to your job template, doing so will force the use of your AWS credentials and will not “fall through” to use your IAM role credentials (this is due to the use of the boto library.)

10.4.2 Ansible Galaxy/Automation Hub API Token

Selecting this credential allows the automation controller to access Galaxy or use a collection published on a local Automation Hub. See *Using Collections via Hub* for detail. Entering the Galaxy server URL is the only required value on this screen.



Credentials

Create New Credential

Name * Galaxy token

Description

Organization *

Credential Type * Ansible Galaxy/Automation Hub API Token

Type Details

Galaxy Server URL * galaxy.server.url

Auth Server URL

API Token

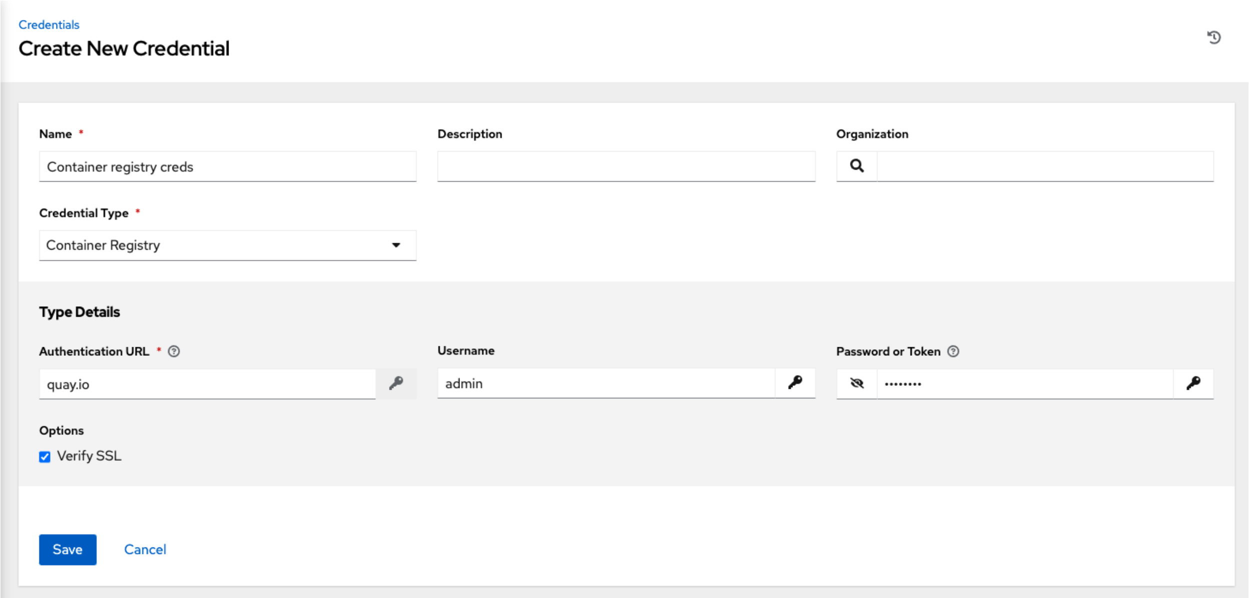
10.4.3 Centrify Vault Credential Provider Lookup

This is considered part of the secret management capability. See *Centrify Vault Credential Provider Lookup* for more detail.

10.4.4 Container Registry

Selecting this credential allows the automation controller to access a collection of container images. See What is a container registry? for more information.

Aside from specifying a name, the **Authentication URL** is the only required field on this screen, and it is already pre-populated with a default value. You may change this default by specifying the authentication endpoint for a different container registry.



Credentials

Create New Credential

Name * Container registry creds

Description

Organization

Credential Type * Container Registry

Type Details

Authentication URL * quay.io

Username admin

Password or Token ••••••••

Options

☑ Verify SSL

10.4.5 CyberArk AIM Credential Provider Lookup

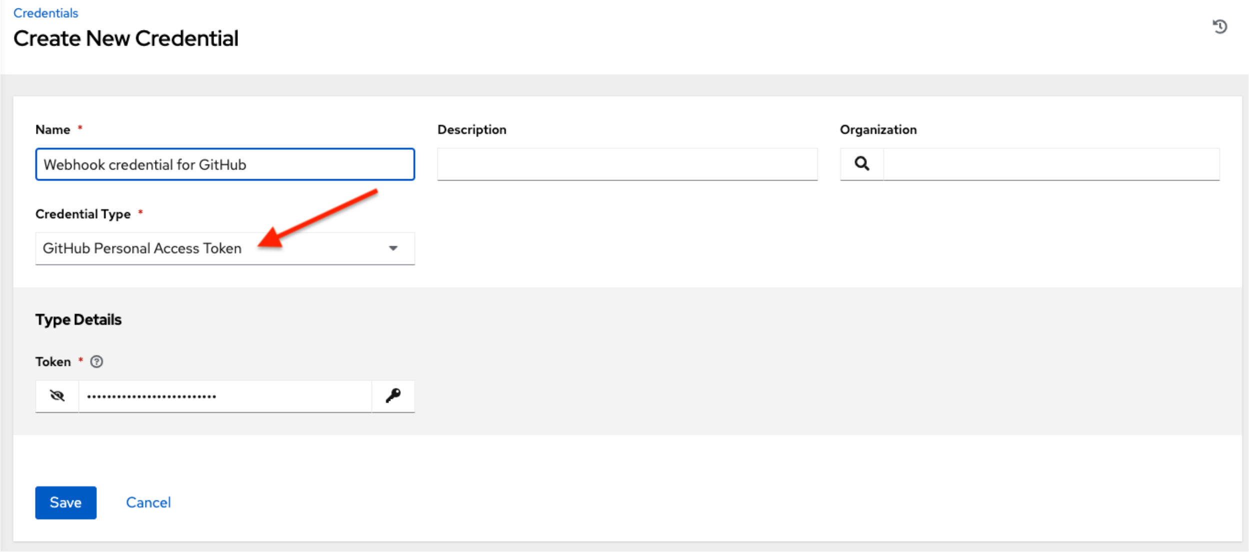
This is considered part of the secret management capability. See *CyberArk AIM Credential Provider Lookup* for more detail.

10.4.6 CyberArk Conjur Secret Lookup

This is considered part of the secret management capability. See *CyberArk Conjur Secret Lookup* for more detail.

10.4.7 GitHub Personal Access Token

Selecting this credential allows you to access GitHub using a Personal Access Token (PAT), which is obtained through GitHub. See *Working with Webhooks* for detail. Entering the provided token is the only required value in this screen.



Credentials

Create New Credential

Name * Webhook credential for GitHub

Description

Organization

Credential Type * GitHub Personal Access Token

Type Details

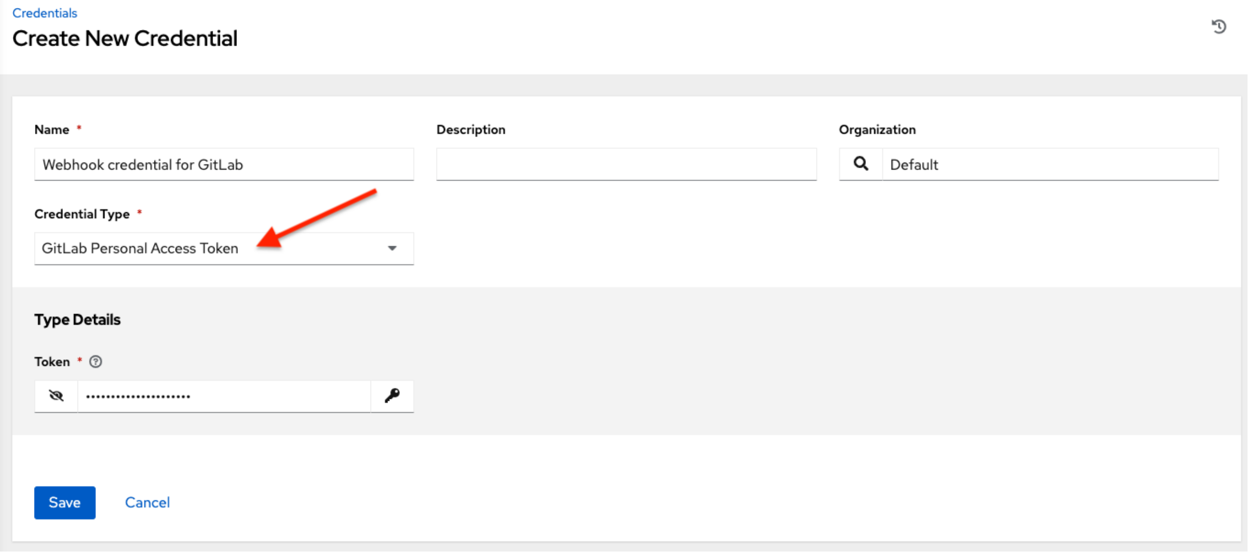
Token * ••••••••••••••••••••••••••••

GitHub PAT credentials require a value in the **Token** field, which is provided in your GitHub profile settings.

This credential can be used for establishing an API connection to GitHub for use in webhook listener jobs, to post status updates.

10.4.8 GitLab Personal Access Token

Selecting this credential allows you to access GitLab using a Personal Access Token (PAT), which is obtained through GitLab. See *Working with Webhooks* for detail. Entering the provided token is the only required value in this screen.



Name * Webhook credential for GitLab

Description

Organization Default

Credential Type * GitLab Personal Access Token

Type Details

Token * ••••••••••••••••••••••

GitLab PAT credentials require a value in the **Token** field, which is provided in your GitLab profile settings.

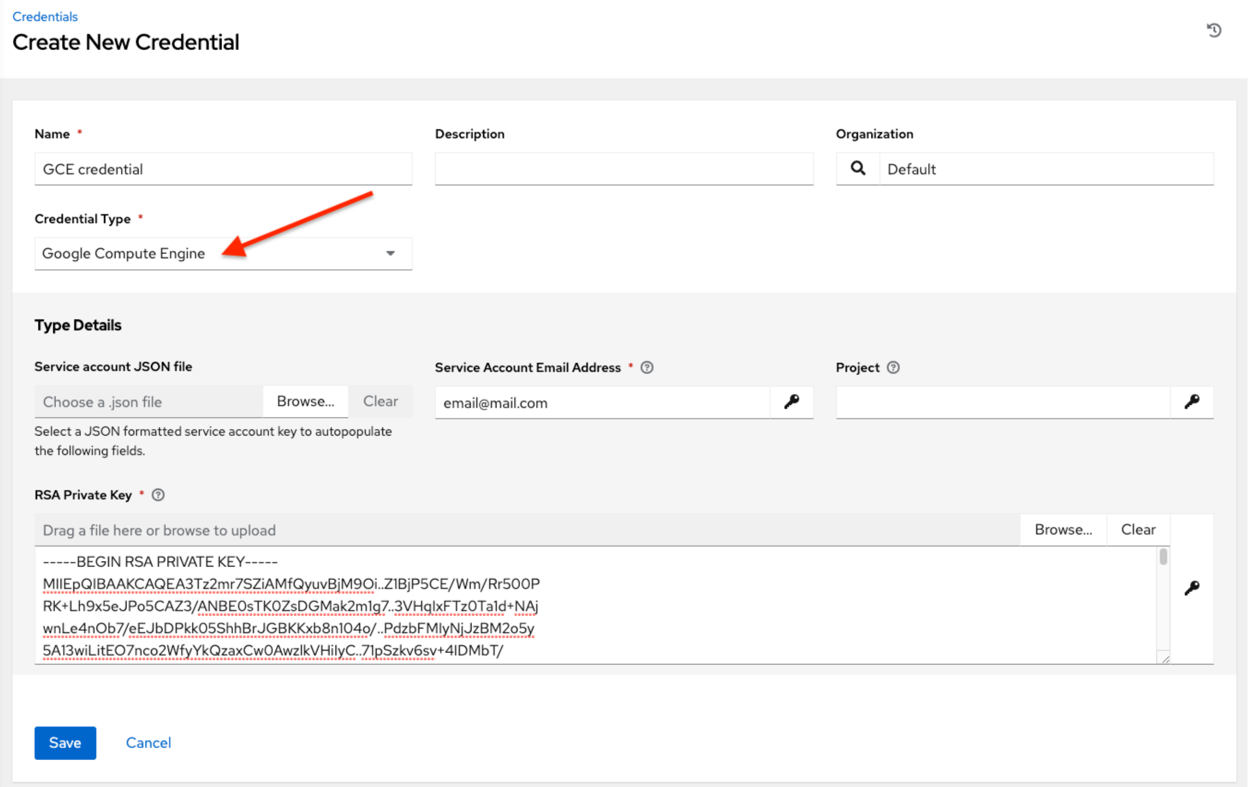
This credential can be used for establishing an API connection to GitLab for use in webhook listener jobs, to post status updates.

10.4.9 Google Compute Engine

Selecting this credential type enables synchronization of cloud inventory with Google Compute Engine (GCE).

The automation controller uses the following environment variables for GCE credentials and are fields prompted in the user interface:

```
GCE_EMAIL
GCE_PROJECT
GCE_CREDENTIALS_FILE_PATH
```



Create New Credential

Name * GCE credential

Description

Organization Default

Credential Type * Google Compute Engine

Type Details

Service account JSON file: Choose a .json file

Select a JSON formatted service account key to autopopulate the following fields.

Service Account Email Address * email@mail.com

Project

RSA Private Key *

Drag a file here or browse to upload

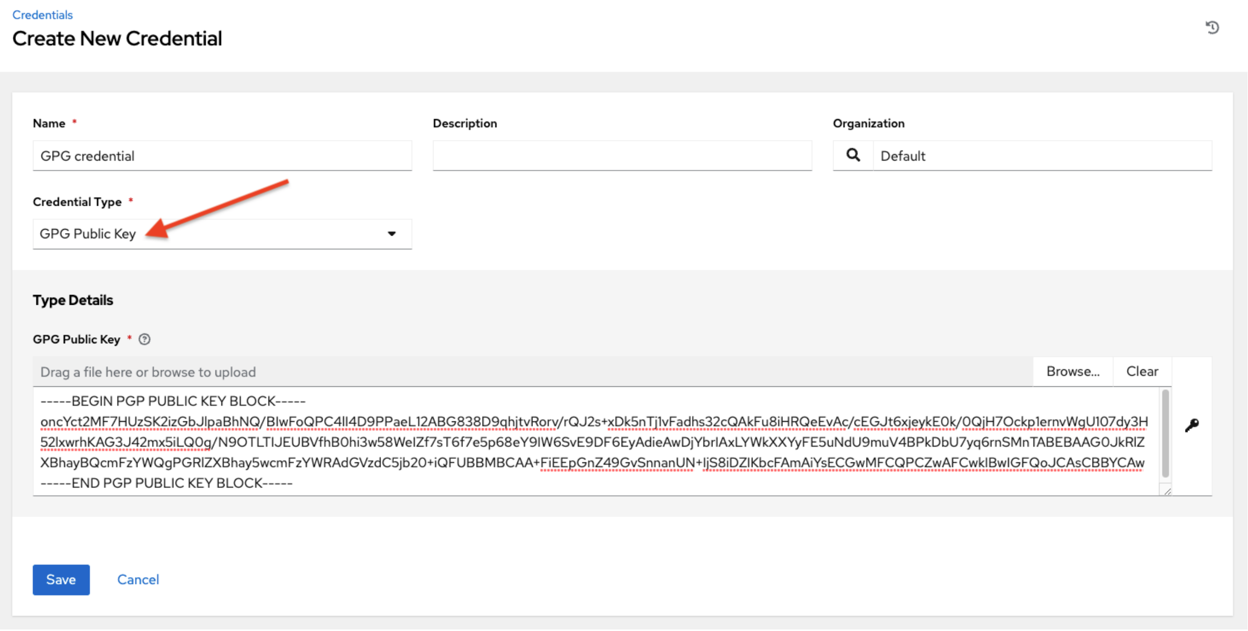
```
-----BEGIN RSA PRIVATE KEY-----
MIIEpQIBAAKCAQEA3Tz2mr7SZiAMfQyuvBjM9Oi..Z1BjP5CE/Wm/Rr500P
RK+Lh9x5eJPo5CAZ3/ANBE0sTK0ZsDGMak2m1g7..3VHqIxFTz0Ta1d+NAj
wnLe4nOb7/eEJbDPkk05ShhBrJGBKKxb8n104o/..PdzbFMIyNjJzBM2o5y
5A13wiLitEO7nco2WfyYkQzaxCw0AwzlkVHilyC..71pSzkv6sv+4IDMbT/
```

GCE credentials have the following inputs that are required:

- **Service Account Email Address**: The email address assigned to the Google Compute Engine **service account**.
- **Project**: Optionally provide the GCE assigned identification or the unique project ID you provided at project creation time.
- **Service Account JSON File**: Optionally upload a GCE service account file. Use the folder (📂) icon to browse for the file that contains the special account information that can be used by services and applications running on your GCE instance to interact with other Google Cloud Platform APIs. This grants permissions to the service account and virtual machine instances.
- **RSA Private Key**: The PEM file associated with the service account email.

10.4.10 GPG Public Key

Selecting this credential type allows you to create a credential that gives the controller the ability to verify the integrity of the project when syncing from source control.



Credentials

Create New Credential

Name * GPG credential

Description

Organization Default

Credential Type * GPG Public Key

Type Details

GPG Public Key *

Drag a file here or browse to upload

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
oncYct2MF7HUzSK2izGbJlpaBhNQ/BlwFoQPC4lI4D9PPaeL12ABG838D9qhjtvRorv/rQJ2s+xDk5nTj1vFadhs32cQAkFu8iHRQeEvAc/cEGJt6xjeykE0k/0QjH7Ockp1ernvWgU107dy3H
52lxwrhKAG3J42mx5iLQ0g/N9OTLTIJEUBVfhB0hi3w58WeIZf7sT6f7e5p68eY9lW6SvE9DF6EyAdieAwDjYbrlAxLYWkXXYyFE5uNdU9muV4BPkDbU7yq6rnSMnTABEBAAG0JkRlZ
XBhayBQcmFzYWQgPGRlZXBhay5wcmFzYWRAdGVzdC5jb20+iQFUBBMBCAA+FiEEpGnZ49GvSnnanUN+ljS8iDZlKbcFAmAiYsECGwMFCQPCZwAFCwkIBwIGFQoJCAsCBBYCAw
-----END PGP PUBLIC KEY BLOCK-----
```

See *Project Signing and Verification* for detailed information on how to generate a valid keypair, use the CLI tool to sign content, and how to add the public key to the controller.

10.4.11 HashiCorp Vault Secret Lookup

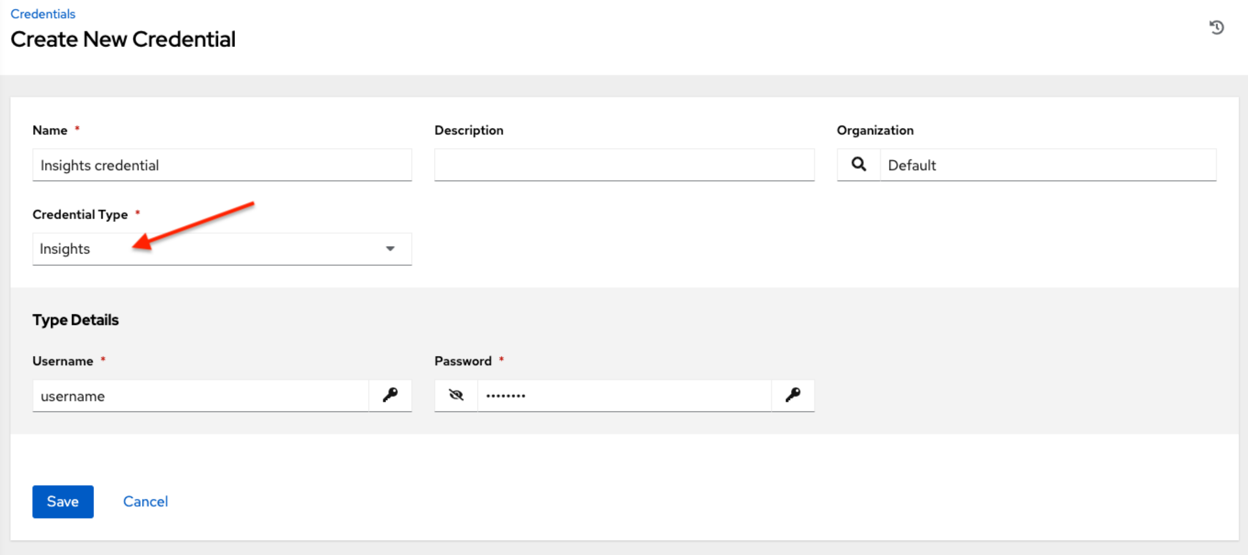
This is considered part of the secret management capability. See *HashiCorp Vault Secret Lookup* for more detail.

10.4.12 HashiCorp Vault Signed SSH

This is considered part of the secret management capability. See *HashiCorp Vault Signed SSH* for more detail.

10.4.13 Insights

Selecting this credential type enables synchronization of cloud inventory with Red Hat Insights.



Credentials

Create New Credential

Name * Insights credential

Description

Organization Default

Credential Type * Insights

Type Details

Username * username

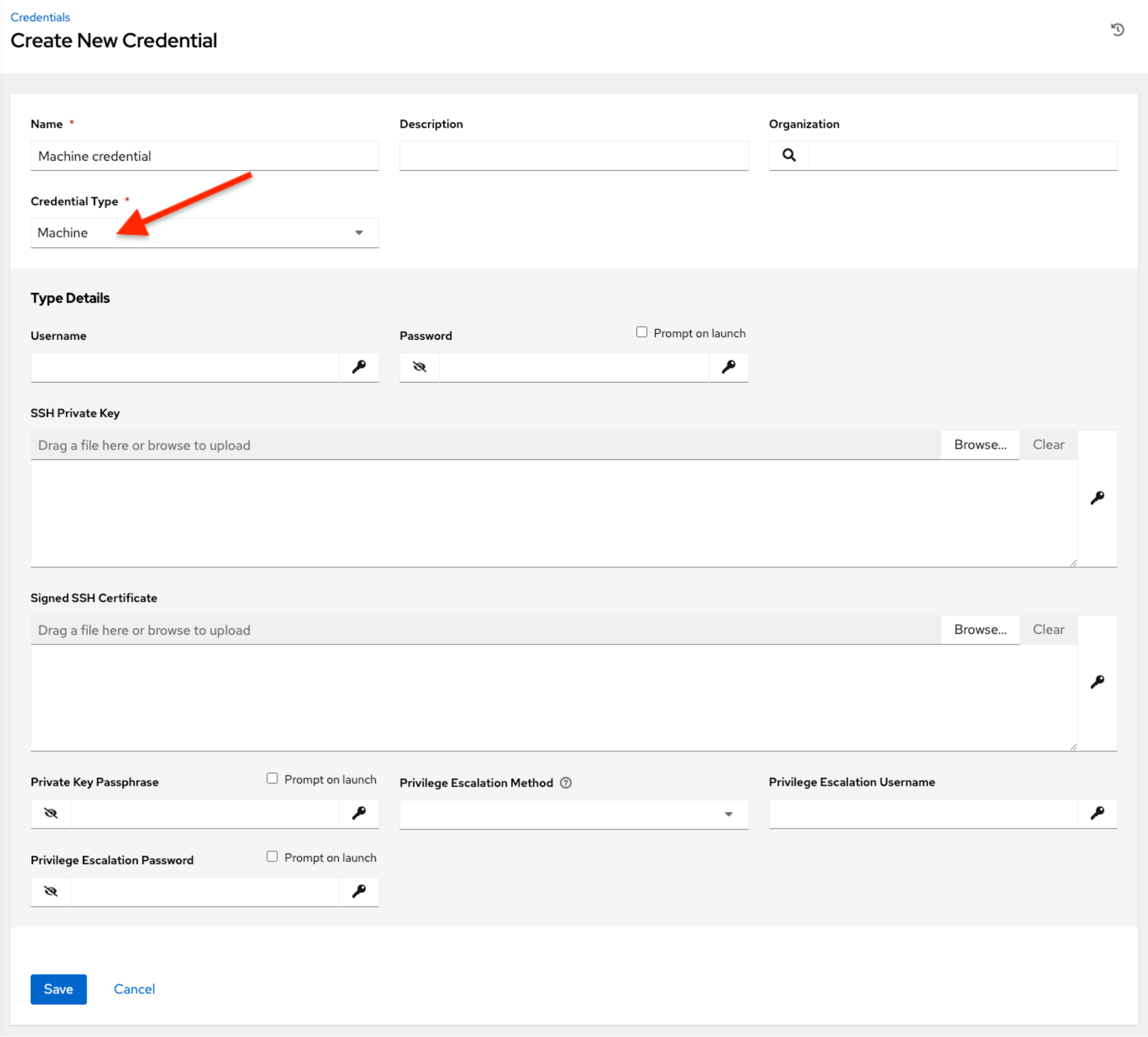
Password * ••••••••

Insights credentials consist of the Insights **Username** and **Password**, which is the user's Red Hat Customer Portal Account username and password.

10.4.14 Machine

Machine credentials enable the automation controller to invoke Ansible on hosts under your management. Just like using Ansible on the command line, you can specify the SSH username, optionally provide a password, an SSH key, a key password, or even have the automation controller prompt the user for their password at deployment time. They define ssh and user-level privilege escalation access for playbooks, and are used when submitting jobs to run playbooks on a remote host. Network connections (`httpapi`, `netconf`, and `network_cli`) use **Machine** for the credential type.

Machine/SSH credentials do not use environment variables. Instead, they pass the username via the `ansible -u` flag, and interactively write the SSH password when the underlying SSH client prompts for it.



Credentials

Create New Credential

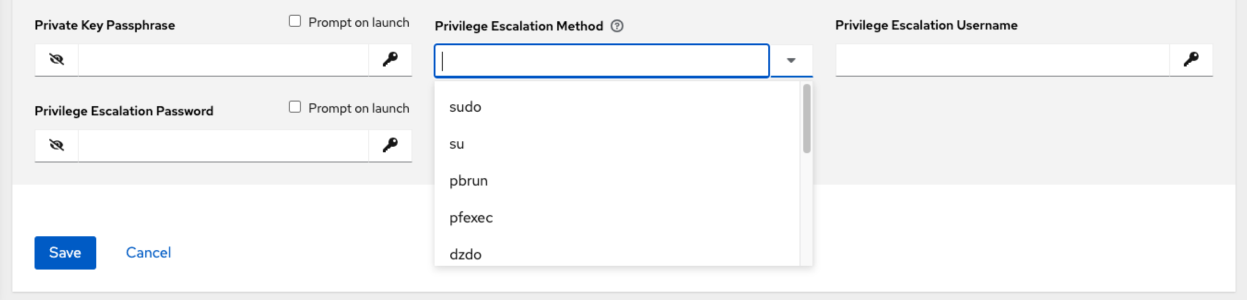
Name * Machine credential

Credential Type * Machine

Machine credentials have several attributes that may be configured:

- **Username**: The username to be used for SSH authentication.
- **Password**: The actual password to be used for SSH authentication. This password will be stored encrypted in the database, if entered. Alternatively, you can configure the automation controller to ask the user for the password at launch time by selecting **Prompt on launch**. In these cases, a dialog opens when the job is launched, promoting the user to enter the password and password confirmation.
- **SSH Private Key**: Copy or drag-and-drop the SSH private key for the machine credential.
- **Private Key Passphrase**: If the SSH Private Key used is protected by a password, you can configure a Key Password for the private key. This password will be stored encrypted in the database, if entered. Alternatively, you can configure the automation controller to ask the user for the password at launch time by selecting **Prompt on launch**. In these cases, a dialog opens when the job is launched, prompting the user to enter the password and password confirmation.
- **Privilege Escalation Method**: Specifies the type of escalation privilege to assign to specific users. This is equivalent to specifying the `--become-method=BECOME_METHOD` parameter, where `BECOME_METHOD` could be any of the typical methods described below, or a custom method you've written. Begin entering the

name of the method, and the appropriate name auto-populates.



- empty selection: If a task/play has `become` set to `yes` and is used with an empty selection, then it will default to `sudo`
- **sudo**: Performs single commands with super user (root user) privileges
- **su**: Switches to the super user (root user) account (or to other user accounts)
- **pbrun**: Requests that an application or command be run in a controlled account and provides for advanced root privilege delegation and keylogging
- **pfexec**: Executes commands with predefined process attributes, such as specific user or group IDs
- **dzdo**: An enhanced version of sudo that uses RBAC information in an Centrify's Active Directory service (see Centrify's site on DZDO)
- **pmrun**: Requests that an application is run in a controlled account (refer to Privilege Manager for Unix 6.0)
- **runas**: Allows you to run as the current user
- **enable**: Switches to elevated permissions on a network device
- **doas**: Allows your remote/login user to execute commands as another user via the doas ("Do as user") utility
- **ksu**: Allows your remote/login user to execute commands as another user via Kerberos access
- **machinectl**: Allows you to manage containers via the systemd machine manager
- **sesu**: Allows your remote/login user to execute commands as another user via the CA Privileged Access Manager

Note: Custom `become` plugins are available only starting with Ansible 2.8. For more detail on this concept, refer to *Understanding Privilege Escalation https://docs.ansible.com/ansible/latest/user_guide/become.html* and the *list of become plugins https://docs.ansible.com/ansible/latest/plugins/become.html#plugin-list*.

- **Privilege Escalation Username** field is only seen if an option for privilege escalation is selected. Enter the username to use with escalation privileges on the remote system.
- **Privilege Escalation Password**: field is only seen if an option for privilege escalation is selected. Enter the actual password to be used to authenticate the user via the selected privilege escalation type on the remote system. This password will be stored encrypted in the database, if entered. Alternatively, you may configure the automation controller to ask the user for the password at launch time by selecting **Prompt on launch**. In these cases, a dialog opens when the job is launched, promoting the user to enter the password and password confirmation.

Note: Sudo Password must be used in combination with SSH passwords or SSH Private Keys, since the automation controller must first establish an authenticated SSH connection with the host prior to invoking sudo to change to the sudo user.

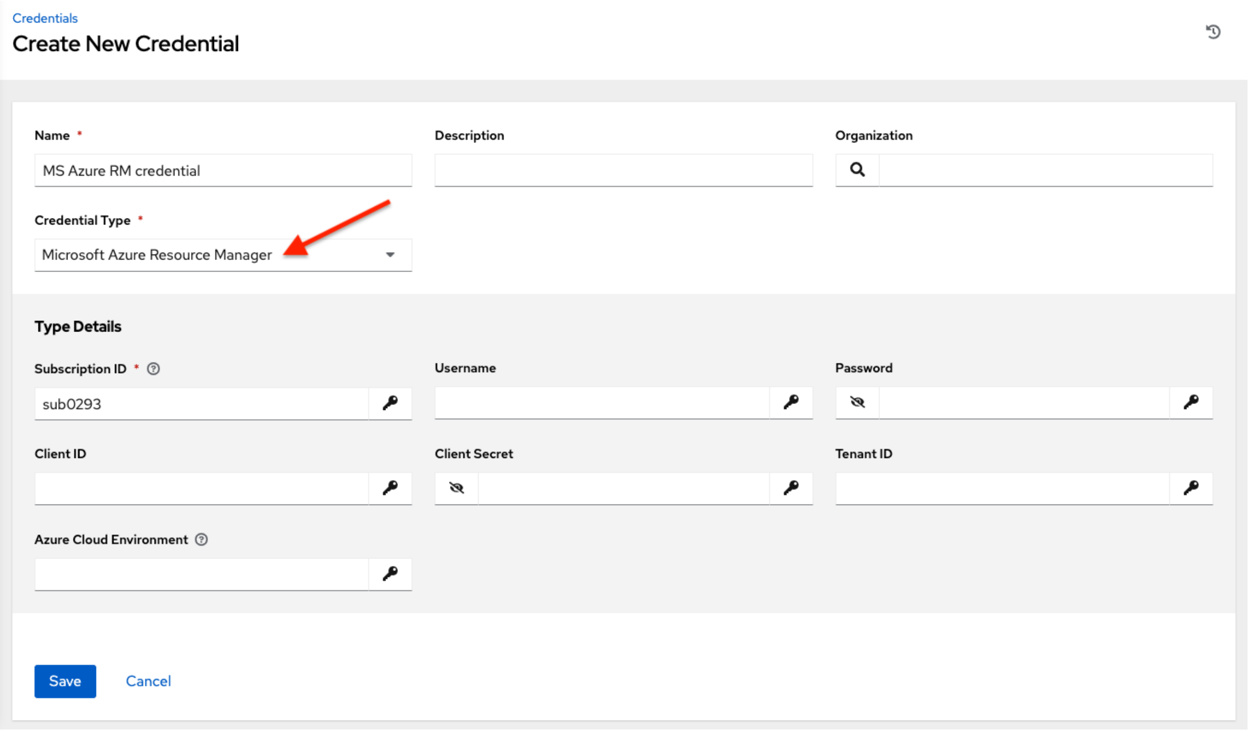
Warning: Credentials which are used in *Scheduled Jobs* must not be configured as "**Prompt on launch**".

10.4.15 Microsoft Azure Key Vault

This is considered part of the secret management capability. See *Microsoft Azure Key Vault* for more detail.

10.4.16 Microsoft Azure Resource Manager

Selecting this credential type enables synchronization of cloud inventory with Microsoft Azure Resource Manager.



Credentials

Create New Credential

Name *: MS Azure RM credential

Description:

Organization:

Credential Type *: Microsoft Azure Resource Manager

Type Details

Subscription ID *: sub0293

Username:

Password:

Client ID:

Client Secret:

Tenant ID:

Azure Cloud Environment:

Microsoft Azure Resource Manager credentials have several attributes that may be configured:

- **Subscription ID**: The Subscription UUID for the Microsoft Azure account (required).
- **Username**: The username to use to connect to the Microsoft Azure account.
- **Password**: The password to use to connect to the Microsoft Azure account.
- **Client ID**: The Client ID for the Microsoft Azure account.
- **Client Secret**: The Client Secret for the Microsoft Azure account.
- **Tenant ID**: The Tenant ID for the Microsoft Azure account.
- **Azure Cloud Environment**: The variable associated with Azure cloud or Azure stack environments.

These fields are equivalent to the variables in the API. To pass service principal credentials, define the following variables:

```
AZURE_CLIENT_ID
AZURE_SECRET
AZURE_SUBSCRIPTION_ID
AZURE_TENANT
AZURE_CLOUD_ENVIRONMENT
```

To pass an Active Directory username/password pair, define the following variables:

```
AZURE_AD_USER
AZURE_PASSWORD
AZURE_SUBSCRIPTION_ID
```

You can also pass credentials as parameters to a task within a playbook. The order of precedence is parameters, then environment variables, and finally a file found in your home directory.

To pass credentials as parameters to a task, use the following parameters for service principal credentials:

```
client_id
secret
subscription_id
tenant
azure_cloud_environment
```

Or, pass the following parameters for Active Directory username/password:

```
ad_user
password
subscription_id
```

10.4.17 Network

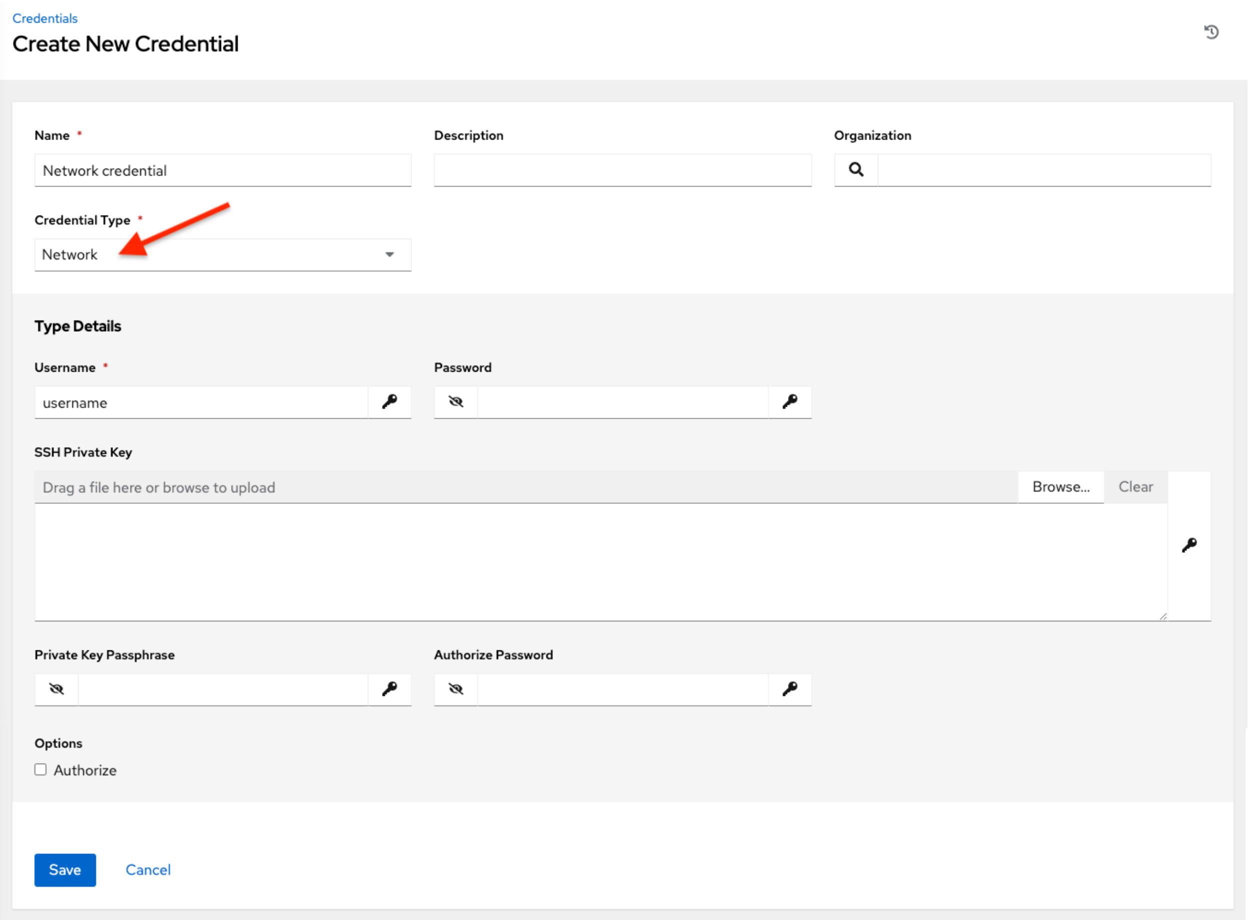
Select the Network credential type **only** if you are using a *local* connection with *provider* to use Ansible networking modules to connect to and manage networking devices. When connecting to network devices, the credential type must match the connection type:

- For `local` connections using `provider`, credential type should be **Network**
- For all other network connections (`httpapi`, `netconf`, and `network_cli`), credential type should be **Machine**

For an overview of connection types available for network devices, refer to Multiple Communication Protocols.

The automation controller uses the following environment variables for Network credentials and are fields prompted in the user interface:

```
ANSIBLE_NET_USERNAME
ANSIBLE_NET_PASSWORD
```



Name * Network credential

Credential Type * Network

Username * username

□ Authorize

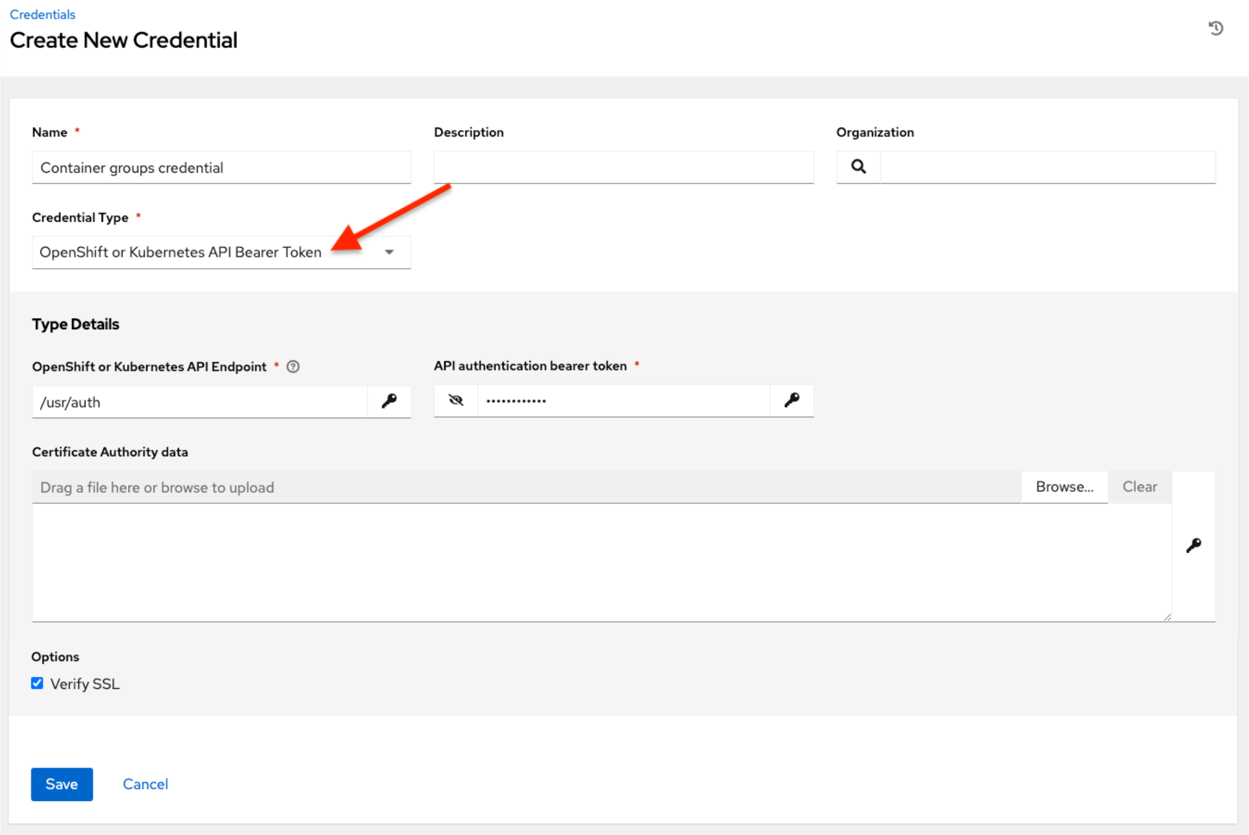
Network credentials have several attributes that may be configured:

- **Username**: The username to use in conjunction with the network device (required).
- **Password**: The password to use in conjunction with the network device.
- **SSH Private Key**: Copy or drag-and-drop the actual SSH Private Key to be used to authenticate the user to the network via SSH.
- **Private Key Passphrase**: The actual passphrase for the private key to be used to authenticate the user to the network via SSH.
- **Authorize**: Select this from the Options field to control whether or not to enter privileged mode.
- If **Authorize** is checked, enter a password in the **Authorize Password** field to access privileged mode.

For more information, refer to the *Inside Playbook* blog, Porting Ansible Network Playbooks with New Connection Plugins.

10.4.18 OpenShift or Kubernetes API Bearer Token

Selecting this credential type allows you to create instance groups that point to a Kubernetes or OpenShift container. For more information about this concept, refer to Container and Instance Groups.



Credentials

Create New Credential

Name * Container groups credential

Description

Organization

Credential Type * OpenShift or Kubernetes API Bearer Token

Type Details

OpenShift or Kubernetes API Endpoint * /usr/auth

API authentication bearer token * ••••••••••••

Certificate Authority data

Drag a file here or browse to upload

Options

☑ Verify SSL

Container credentials have the following inputs:

- **OpenShift or Kubernetes API Endpoint** (required): the endpoint to be used to connect to an OpenShift or Kubernetes container
- **API Authentication Bearer Token** (required): The token to use to authenticate the connection
- **Verify SSL**: Optionally you can check this option to verify the server's SSL certificate is valid and trusted. Environments that use internal or private CA's should leave this option unchecked to disable verification.
- **Certificate Authority Data**: include the `BEGIN CERTIFICATE` and `END CERTIFICATE` lines when pasting the certificate, if provided

A `ContainerGroup` is a type of `InstanceGroup` that has an associated Credential that allows for connecting to an OpenShift cluster. To set up a container group, you must first have the following:

- A namespace you can launch into (every cluster has a "default" namespace, but you may want to use a specific namespace)
- A service account that has the roles that allow it to launch and manage Pods in this namespace
- If you will be using execution environments in a private registry, and have a Container Registry credential associated to them in the automation controller, the service account also needs the roles to get, create, and delete secrets in the namespace. If you do not want to give these roles to the service account, you can pre-create the `ImagePullSecrets` and specify them on the pod spec for the ContainerGroup. In this case, the execution environment should NOT have a Container Registry credential associated, or the controller will attempt to create the secret for you in the namespace.

- A token associated with that service account (OpenShift or Kubernetes Bearer Token)
- A CA certificate associated with the cluster

This section describes creating a Service Account in an Openshift cluster (or K8s) in order to be used to run jobs in a container group via automation controller. After the Service Account is created, its credentials are provided to the controller in the form of an Openshift or Kubernetes API bearer token credential. Below describes how to create a service account and collect the needed information for configuring automation controller.

To configure the controller:

1. To create a service account, you may download and use this sample service account, `containergroup sa` and modify it as needed to obtain the above credentials.
2. Apply the configuration from `containergroup-sa.yml`:

   ```
   oc apply -f containergroup-sa.yml
   ```

3. Get the secret name associated with the service account:

   ```
   export SA_SECRET=$(oc get sa containergroup-service-account -o json | jq '.
   ↪secrets[0].name' | tr -d '"')
   ```

4. Get the token from the secret:

   ```
   oc get secret $(echo ${SA_SECRET}) -o json | jq '.data.token' | xargs | base64 --
   ↪decode > containergroup-sa.token
   ```

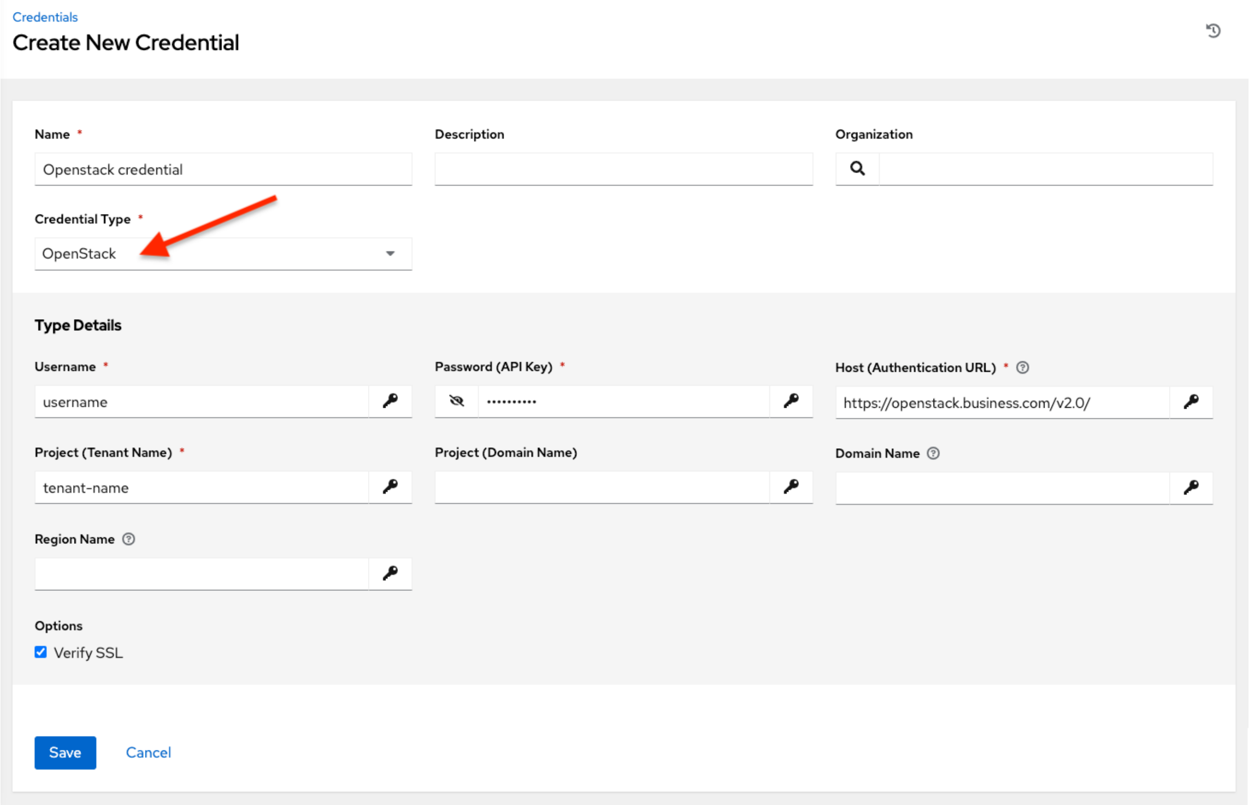
5. Get the CA cert:

   ```
   oc get secret $SA_SECRET -o json | jq '.data["ca.crt"]' | xargs | base64 --decode␣
   ↪> containergroup-ca.crt
   ```

6. Use the contents of `containergroup-sa.token` and `containergroup-ca.crt` to provide the information for the *OpenShift or Kubernetes API Bearer Token* required for the container group.

10.4.19 OpenStack

Selecting this credential type enables synchronization of cloud inventory with OpenStack.



Credentials

Create New Credential

Name * Openstack credential

Description

Organization

Credential Type * OpenStack

Type Details

Username * username

Password (API Key) * ••••••••••

Host (Authentication URL) * https://openstack.business.com/v2.0/

Project (Tenant Name) * tenant-name

Project (Domain Name)

Domain Name

Region Name

Options

☑ Verify SSL

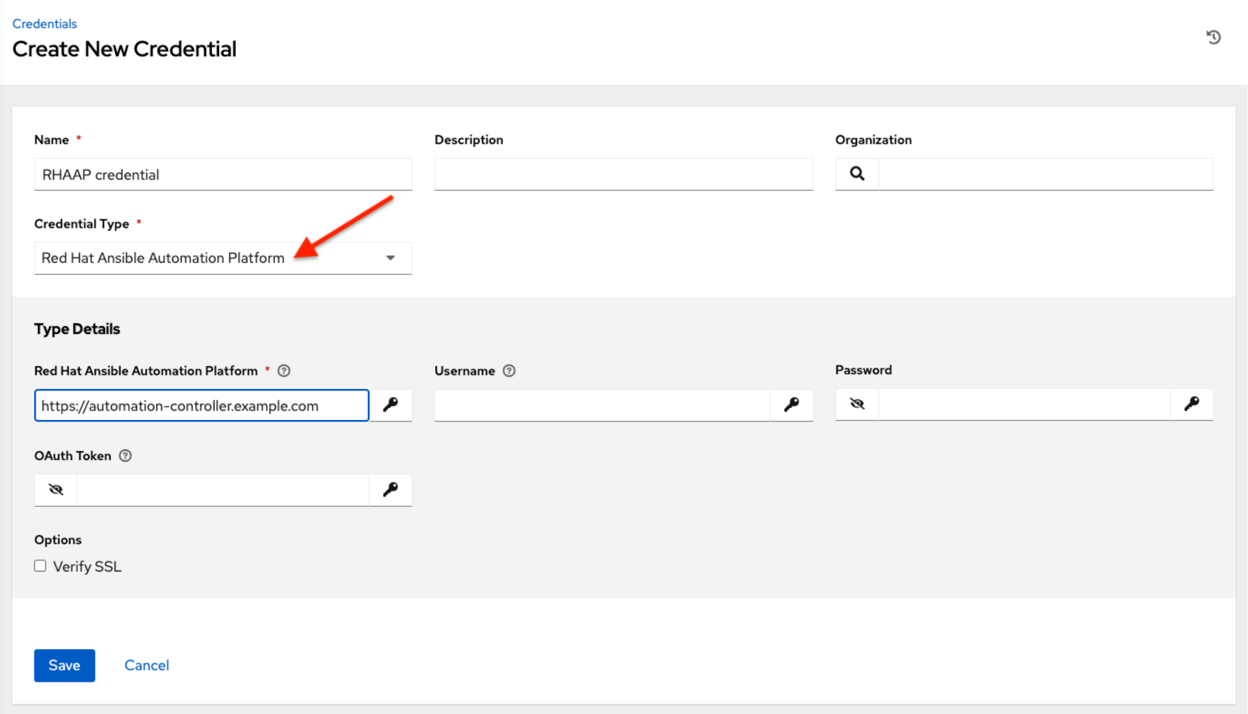
OpenStack credentials have the following inputs that are required:

- **Username**: The username to use to connect to OpenStack.
- **Password (API Key)**: The password or API key to use to connect to OpenStack.
- **Host (Authentication URL)**: The host to be used for authentication.
- **Project (Tenant Name)**: The Tenant name or Tenant ID used for OpenStack. This value is usually the same as the username.
- **Project (Domain Name)**: Optionally provide the project name associated with your domain.
- **Domain name**: Optionally provide the FQDN to be used to connect to OpenStack.

If you are interested in using OpenStack Cloud Credentials, refer to *Utilizing Cloud Credentials* in this guide for more information, including a sample playbook.

10.4.20 Red Hat Ansible Automation Platform

Selecting this credential allows you to access another automation controller instance.



Name * RHAAP credential

Description

Organization

Credential Type * Red Hat Ansible Automation Platform

Type Details

Red Hat Ansible Automation Platform * https://automation-controller.example.com

Username

Password

OAuth Token

Options

□ Verify SSL

Automation controller credentials have the following inputs that are required:

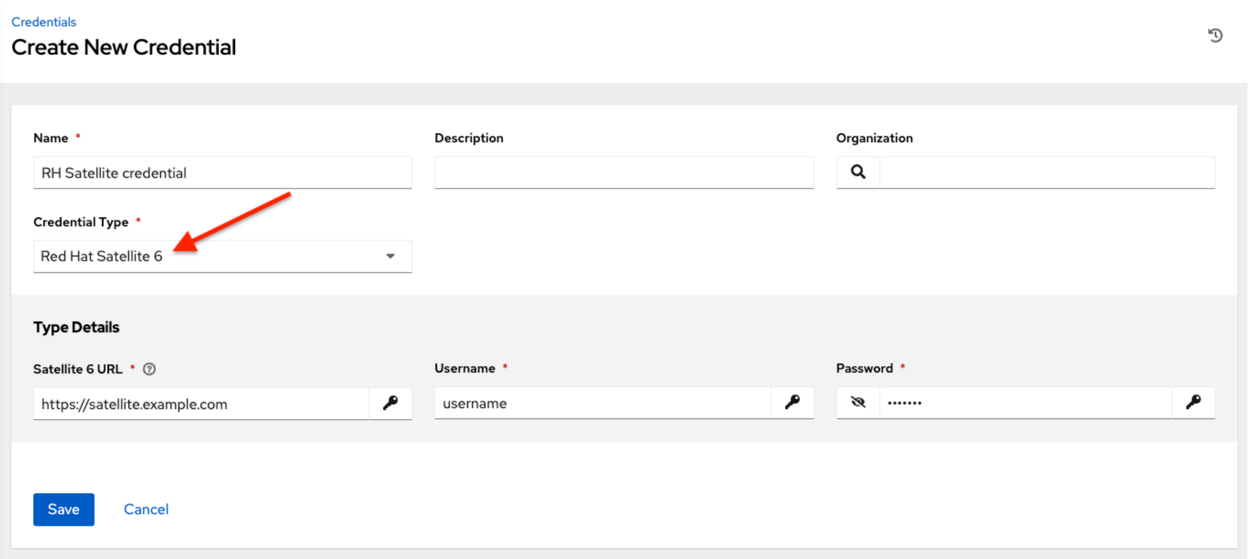
- **Controller Hostname**: The base URL or IP address of the other instance to connect to.
- **Username**: The username to use to connect to it.
- **Password**: The password to use to connect to it.
- **Oauth Token**: If username and password is not used, provide an OAuth token to use to authenticate.

10.4.21 Red Hat Satellite 6

Selecting this credential type enables synchronization of cloud inventory with Red Hat Satellite 6.

The automation controller writes a Satellite configuration file based on fields prompted in the user interface. The absolute path to the file is set in the following environment variable:

```
FOREMAN_INI_PATH
```



Create New Credential

Name * RH Satellite credential

Description

Organization

Credential Type * Red Hat Satellite 6

Type Details

Satellite 6 URL * https://satellite.example.com

Username * username

Password * •••••••

Satellite credentials have the following inputs that are required:

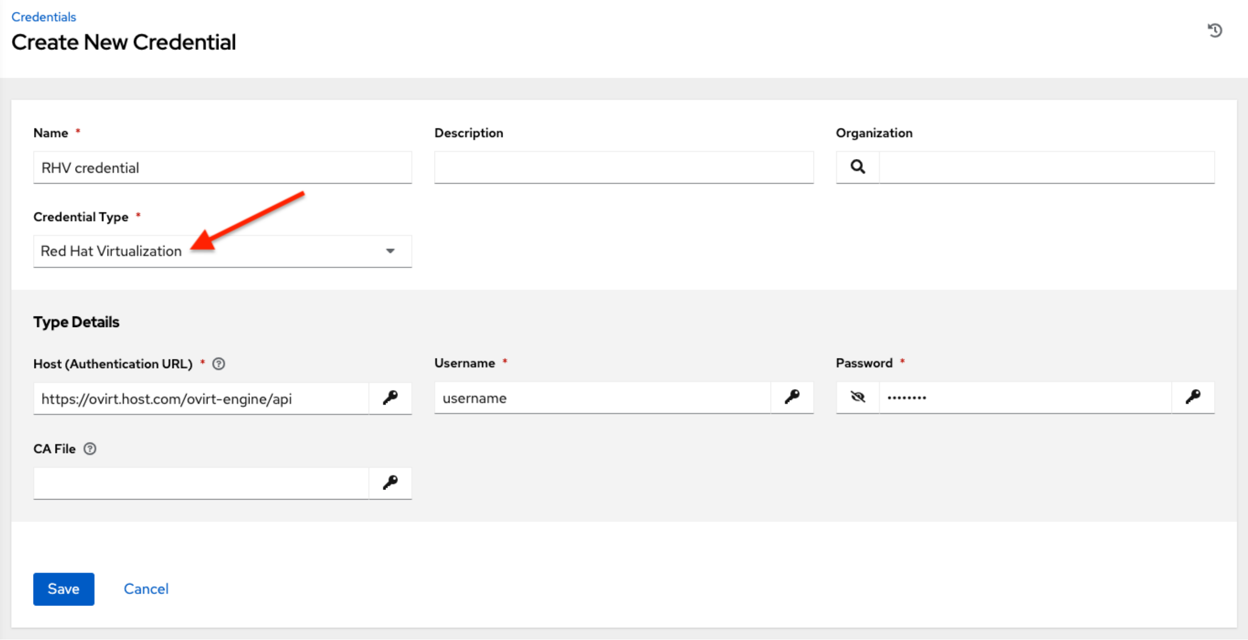
- **Satellite 6 URL**: The Satellite 6 URL or IP address to connect to.
- **Username**: The username to use to connect to Satellite 6.
- **Password**: The password to use to connect to Satellite 6.

10.4.22 Red Hat Virtualization

This credential allows the automation controller to access Ansible's `oVirt4.py` dynamic inventory plugin, which is managed by Red Hat Virtualization (RHV).

The automation controller uses the following environment variables for Red Hat Virtualization credentials and are fields in the user interface:

```
OVIRT_URL
OVIRT_USERNAME
OVIRT_PASSWORD
```



Credentials

Create New Credential

Name *: RHV credential

Description:

Organization:

Credential Type *: Red Hat Virtualization

Type Details

Host (Authentication URL) *: https://ovirt.host.com/ovirt-engine/api

Username *: username

Password *: ••••••••

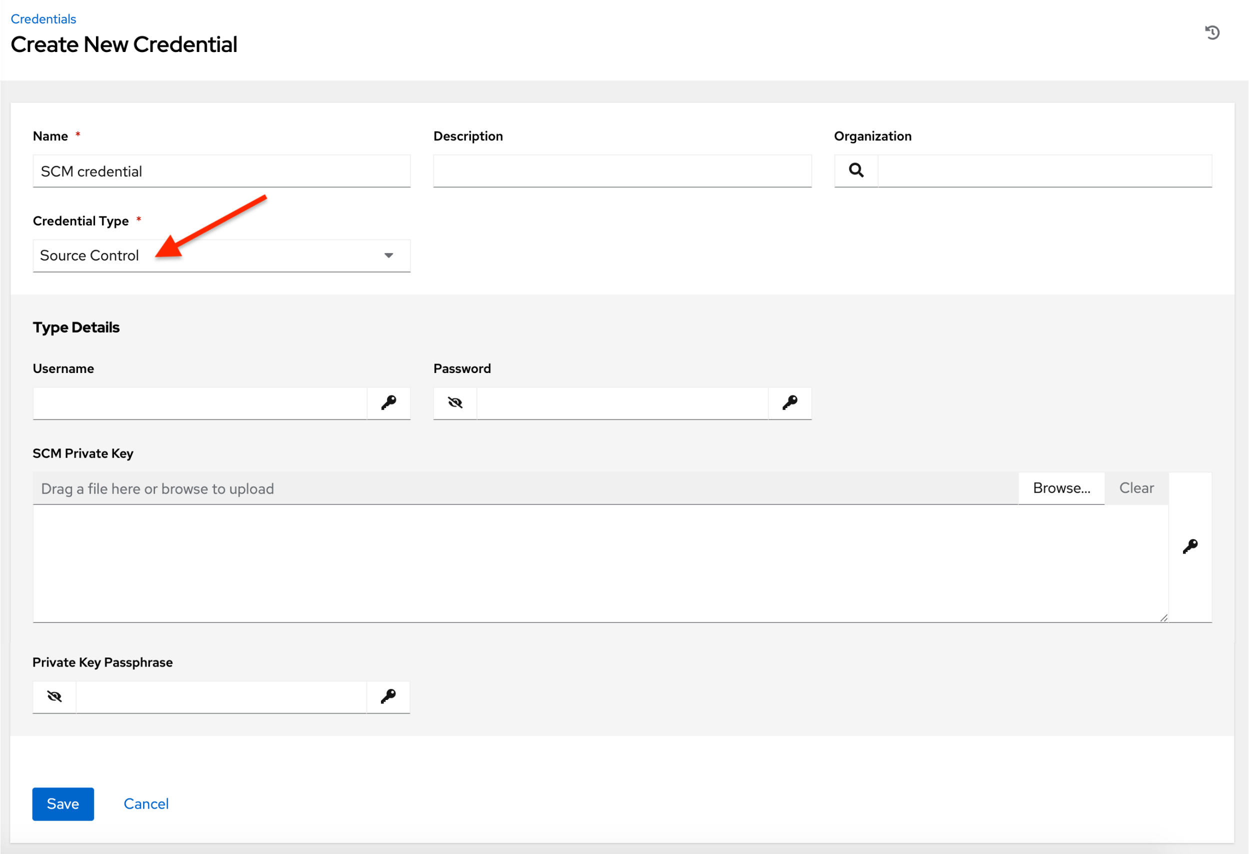
CA File:

RHV credentials have the following inputs that are required:

- **Host (Authentication URL)**: The host URL or IP address to connect to. In order to sync with the inventory, the credential URL needs to include the `ovirt-engine/api` path.
- **Username**: The username to use to connect to oVirt4. This needs to include the domain profile to succeed, for example `username@ovirt.host.com`.
- **Password**: The password to use to connect to it.
- **CA File**: Optionally provide an absolute path to the oVirt certificate file (it may end in `.pem`, `.cer` and `.crt` extensions, but preferably `.pem` for consistency)

10.4.23 Source Control

SCM (source control) credentials are used with Projects to clone and update local source code repositories from a remote revision control system such as Git or Subversion.



Credentials

Create New Credential

Name * : SCM credential

Description :

Organization :

Credential Type * : Source Control

Type Details

Username :

Password :

SCM Private Key :

Private Key Passphrase :

Source Control credentials have several attributes that may be configured:

- **Username**: The username to use in conjunction with the source control system.
- **Password**: The password to use in conjunction with the source control system.
- **SCM Private Key**: Copy or drag-and-drop the actual SSH Private Key to be used to authenticate the user to the source control system via SSH.
- **Private Key Passphrase**: If the SSH Private Key used is protected by a passphrase, you may configure a Key Passphrase for the private key.

Note: Source Control credentials cannot be configured as "**Prompt on launch**". If you are using a GitHub account for a Source Control credential and you have 2FA (Two Factor Authenication) enabled on your account, you will need to use your Personal Access Token in the password field rather than your account password.

10.4.24 Thycotic DevOps Secrets Vault

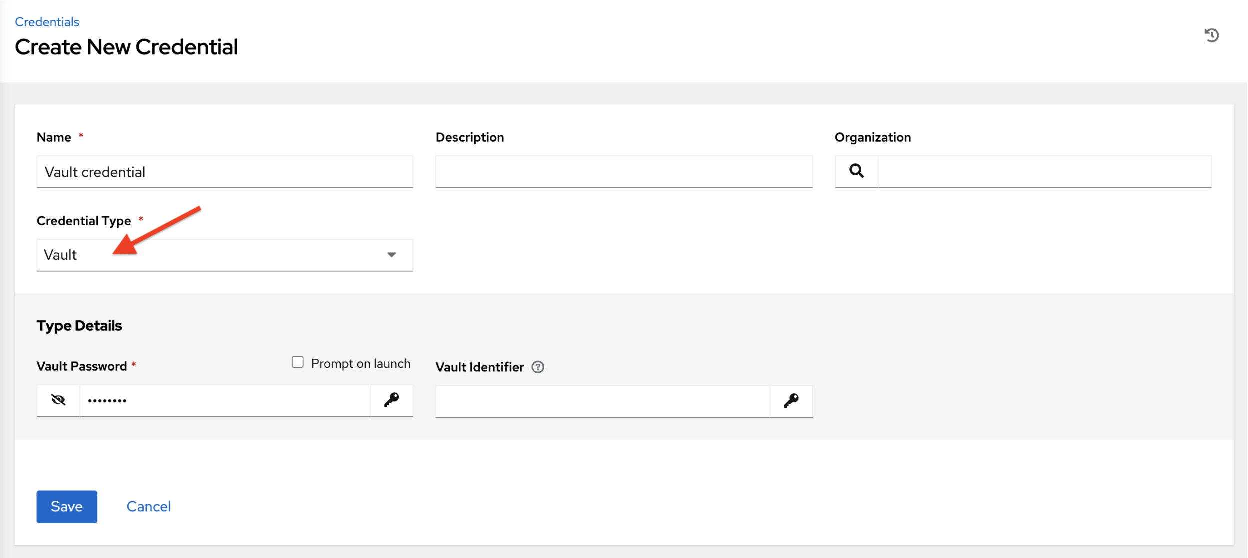
This is considered part of the secret management capability. See *Thycotic DevOps Secrets Vault* for more detail.

10.4.25 Thycotic Secret Server

This is considered part of the secret management capability. See *Thycotic Secret Server* for more detail.

10.4.26 Vault

Selecting this credential type enables synchronization of inventory with Ansible Vault.



Credentials

Create New Credential

Name * Vault credential

Description

Organization

Credential Type * Vault

Type Details

Vault Password * ☐ Prompt on launch ········

Vault Identifier

Vault credentials require the **Vault Password** and an optional **Vault Identifier** if applying multi-Vault credentialing. For more information on the automation controller Multi-Vault support, refer to the Multi-Vault Credentials section of the *Automation Controller Administration Guide*.

You may configure the automation controller to ask the user for the password at launch time by selecting **Prompt on launch**. In these cases, a dialog opens when the job is launched, promoting the user to enter the password and password confirmation.

> **Warning:** Credentials which are used in *Scheduled Jobs* must not be configured as "**Prompt on launch**".

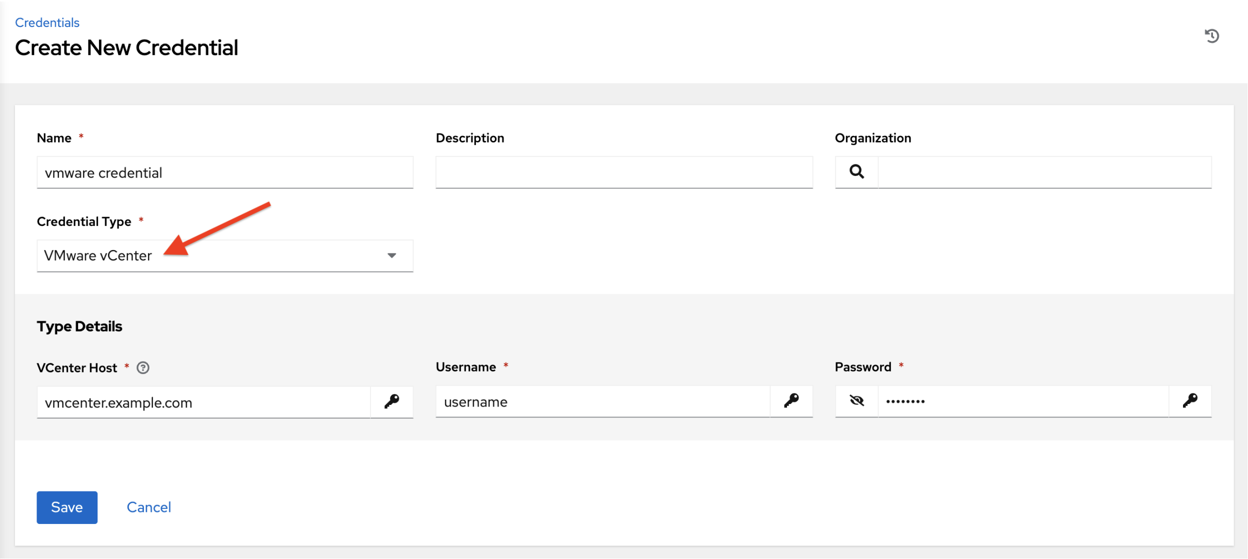
For more information about Ansible Vault, refer to: http://docs.ansible.com/ansible/playbooks_vault.html

10.4.27 VMware vCenter

Selecting this credential type enables synchronization of inventory with VMware vCenter.

The automation controller uses the following environment variables for VMware vCenter credentials and are fields prompted in the user interface:

```
VMWARE_HOST
VMWARE_USER
VMWARE_PASSWORD
VMWARE_VALIDATE_CERTS
```



Credentials

Create New Credential

Name * vmware credential

Description

Organization

Credential Type * VMware vCenter

Type Details

VCenter Host * vmcenter.example.com

Username * username

Password * ••••••••

VMware credentials have the following inputs that are required:

- **vCenter Host**: The vCenter hostname or IP address to connect to.
- **Username**: The username to use to connect to vCenter.
- **Password**: The password to use to connect to vCenter.

Note: If the VMware guest tools are not running on the instance, VMware inventory sync may not return an IP address for that instance.

CHAPTER

ELEVEN

CUSTOM CREDENTIAL TYPES

As an administrator with superuser access, you can define a custom credential type in a standard format using a YAML/JSON-like definition, allowing the assignment of new credential types to jobs and inventory updates. This allows you to define a custom credential type that works in ways similar to existing credential types. For example, you could create a custom credential type that injects an API token for a third-party web service into an environment variable, which your playbook or custom inventory script could consume.

Custom credentials support the following ways of injecting their authentication information:

- Environment variables
- Ansible extra variables
- File-based templating (i.e., generating `.ini` or `.conf` files that contain credential values)

You can attach one SSH and multiple cloud credentials to a Job Template. Each cloud credential must be of a different type. In other words, only one AWS credential, one GCE credential, etc., are allowed. Vault credentials and machine credentials are separate entities.

Note: When creating a new credential type, you are responsible for avoiding collisions in the `extra_vars`, `env`, and file namespaces. Also, avoid environment variable or extra variable names that start with `ANSIBLE_` because they are reserved. You must have Superuser permissions to be able to create and edit a credential type (`CredentialType`) and to be able to view the `CredentialType.injection` field.

11.1 Content sourcing from collections

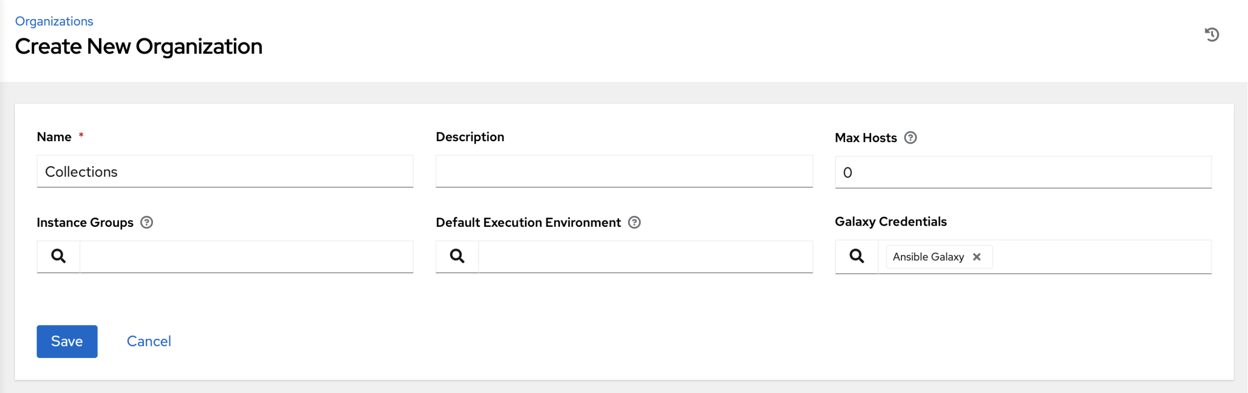
A "managed" credential type of `kind=galaxy` represents a content source for fetching collections defined in `requirements.yml` when project updates are run (e.g., galaxy.ansible.com, cloud.redhat.com, on-premise Automation Hub). This new type will represent a URL and (optional) authentication details necessary to construct the environment variables when a project update runs `ansible-galaxy collection install` as described in the Ansible documentation, Configuring the ansible-galaxy client. It has fields which map directly to the configuration options exposed to the Ansible Galaxy CLI, e.g., per-server. An endpoint in the API reflects an ordered list of these credentials at the Organization level:

```
/api/v2/organizations/N/galaxy_credentials/
```

Installations of the automation controller migrates existing Galaxy-oriented setting values in such a way that post-upgrade, proper credentials are created and attached to every Organization. After upgrading to the latest version , every organization that existed prior to upgrade now has a list of (one or more) "Galaxy" credentials associated with it.

Additionally, post-upgrade, these settings are not be visible (or editable) from the `/api/v2/settings/jobs/` endpoint.

The automation controller should still continue to fetch roles directly from public Galaxy even if galaxy.ansible.com is not the first credential in the list for the Organization. The global "Galaxy" settings are no longer configured at the jobs level, but at the Organization level in the User Interface. The Organization's Add and Edit windows have an optional **Credential** lookup field for credentials of `kind=galaxy`.



Organizations

Create New Organization

Name * : Collections

Description :

Max Hosts : 0

Instance Groups :

Default Execution Environment :

Galaxy Credentials : Ansible Galaxy ×

It is very important to specify the order of these credentials as order sets precedence for the sync and lookup of the content. For more information, see *Creating a New Organization*. For detail on how to set up a project using collections, see *Using Collections via Hub*.

11.2 Backwards-Compatible API Considerations

Support for version 2 of the API (`api/v2/`) means a one-to-many relationship for Job Templates to credentials (including multi-cloud support). Credentials can be filtered using the v2 API:

```
$ curl "https://controller.example.org/api/v2/credentials/?credential_type__
↪namespace=aws"
```

In the V2 CredentialType model, the relationships are defined as follows:

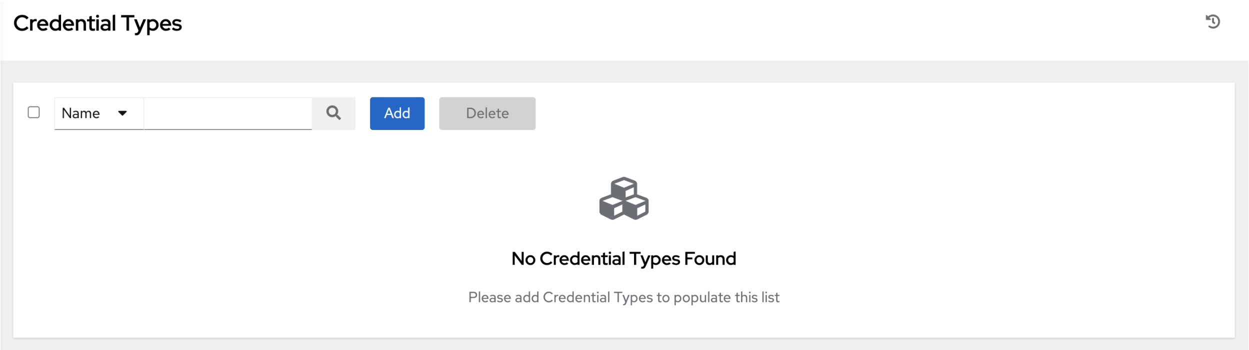
Machine	SSH
Vault	Vault
Network	Sets environment variables (e.g., `ANSIBLE_NET_AUTHORIZE`)
SCM	Source Control
Cloud	EC2, AWS
	Lots of others
Insights	Insights
Galaxy	galaxy.ansible.com, cloud.redhat.com
	on-premise Automation Hub

11.3 Content verification

Automation controller uses GNU Privacy Guard (GPG) to verify content. For more information, refer to The GNU Privacy Handbook.

11.4 Getting Started with Credential Types

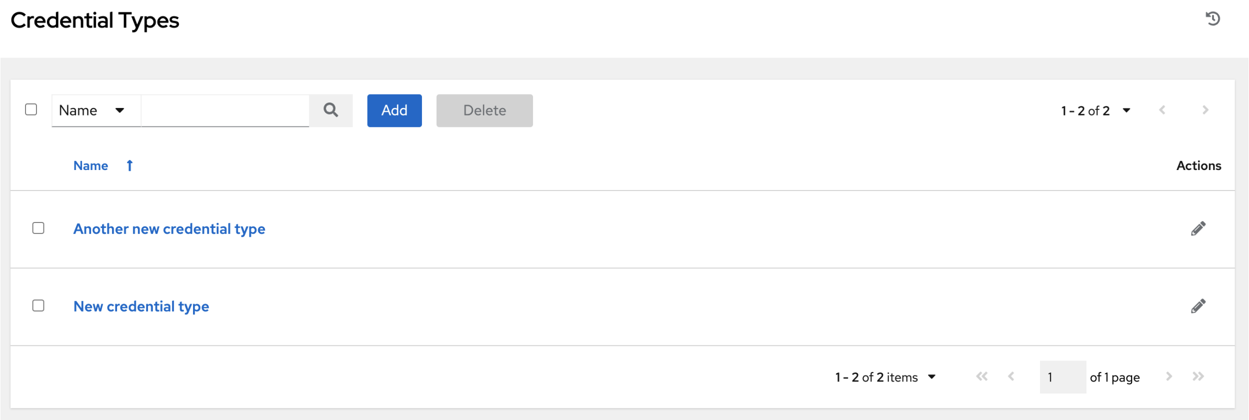
Access the Credentials from clicking **Credential Types** from the left navigation bar. If no custom credential types have been created, the Credential Types view will not have any to display and will prompt you to add one:



No Credential Types Found

Please add Credential Types to populate this list

If credential types have been created, this page displays a list of all existing and available Credential Types.



	Name ↑	Actions
□	Another new credential type	✎
□	New credential type	✎

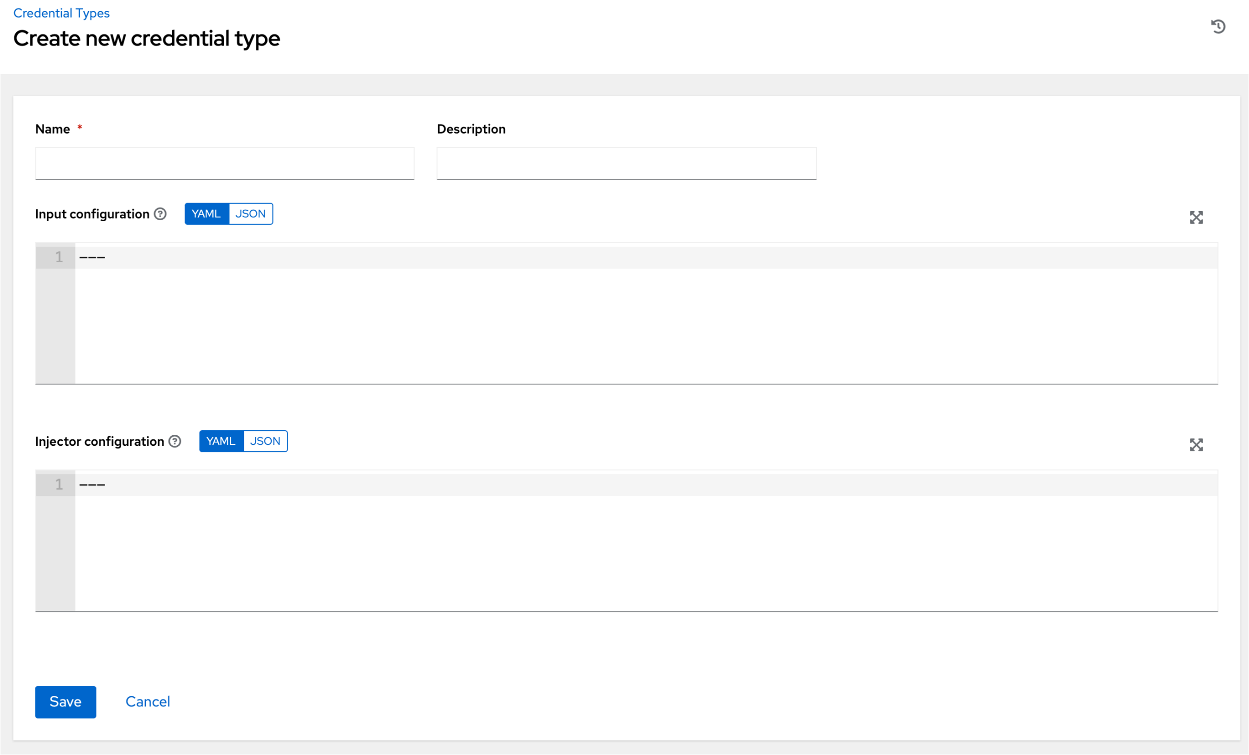
To view more information about a credential type, click on its name or the Edit (✎) button from the **Actions** column.

Each credential type displays its own unique configurations in the **Input Configuration** field and the **Injector Configuration** field, if applicable. Both YAML and JSON formats are supported in the configuration fields.

11.5 Create a New Credential Type

To create a new credential type:

1. Click the **Add** button in the **Credential Types** screen.



2. Enter the appropriate details in the **Name** and **Description** field.

Note: When creating a new credential type, do not use reserved variable names that start with `ANSIBLE_` for the **INPUT** and **INJECTOR** names and IDs, as they are invalid for custom credential types.

3. In the **Input Configuration** field, specify an input schema which defines a set of ordered fields for that type. The format can be in YAML or JSON, as shown:

   YAML

```
fields:
  - type: string
    id: username
    label: Username
  - type: string
    id: password
    label: Password
    secret: true
required:
  - username
  - password
```

   View more YAML examples at http://www.yaml.org/start.html.

   JSON

```
{
"fields": [
  {
  "type": "string",
  "id": "username",
  "label": "Username"
  },
  {
  "secret": true,
  "type": "string",
  "id": "password",
  "label": "Password"
   }
  ],
 "required": ["username", "password"]
}
```

View more JSON examples at www.json.org.

The configuration in JSON format below show each field and how they are used:

```
{
  "fields": [{
    "id": "api_token",                 # required - a unique name used to
                                       # reference the field value

    "label": "API Token",              # required - a unique label for the
                                       # field

    "help_text": "User-facing short text describing the field.",

    "type": ("string" | "boolean")     # defaults to 'string'

    "choices": ["A", "B", "C"]         # (only applicable to `type=string`)

    "format": "ssh_private_key"        # optional, can be used to enforce data
                                       # format validity for SSH private key
                                       # data (only applicable to␣
↪`type=string`)

    "secret": true,                    # if true, the field value will be␣
↪encrypted

    "multiline": false                 # if true, the field should be rendered
                                       # as multi-line for input entry
                                       # (only applicable to `type=string`)
},{
    # field 2...
},{
    # field 3...
}],

"required": ["api_token"]              # optional; one or more fields can be␣
↪marked as required
},
```

When `type=string`, fields can optionally specify multiple choice options:

```
{
  "fields": [{
      "id": "api_token",           # required - a unique name used to␣
↪reference the field value
      "label": "API Token",        # required - a unique label for the field
      "type": "string",
      "choices": ["A", "B", "C"]
  }]
},
```

4. In the **Injector Configuration** field, enter environment variables or extra variables that specify the values a credential type can inject. The format can be in YAML or JSON (see examples in the previous step). The configuration in JSON format below show each field and how they are used:

```
{
  "file": {
      "template": "[mycloud]\ntoken={{ api_token }}"
  },
  "env": {
      "THIRD_PARTY_CLOUD_API_TOKEN": "{{ api_token }}"
  },
  "extra_vars": {
      "some_extra_var": "{{ username }}:{{ password }}"
  }
}
```

Credential Types can also generate temporary files to support .ini files or certificate/key data:

```
{
  "file": {
      "template": "[mycloud]\ntoken={{ api_token }}"
  },
  "env": {
      "MY_CLOUD_INI_FILE": "{{ tower.filename }}"
  }
}
```

In this example, the automation controller will write a temporary file that contains:

```
[mycloud]\ntoken=SOME_TOKEN_VALUE
```

The absolute file path to the generated file will be stored in an environment variable named `MY_CLOUD_INI_FILE`.

An example of referencing multiple files in a custom credential template is as follows:

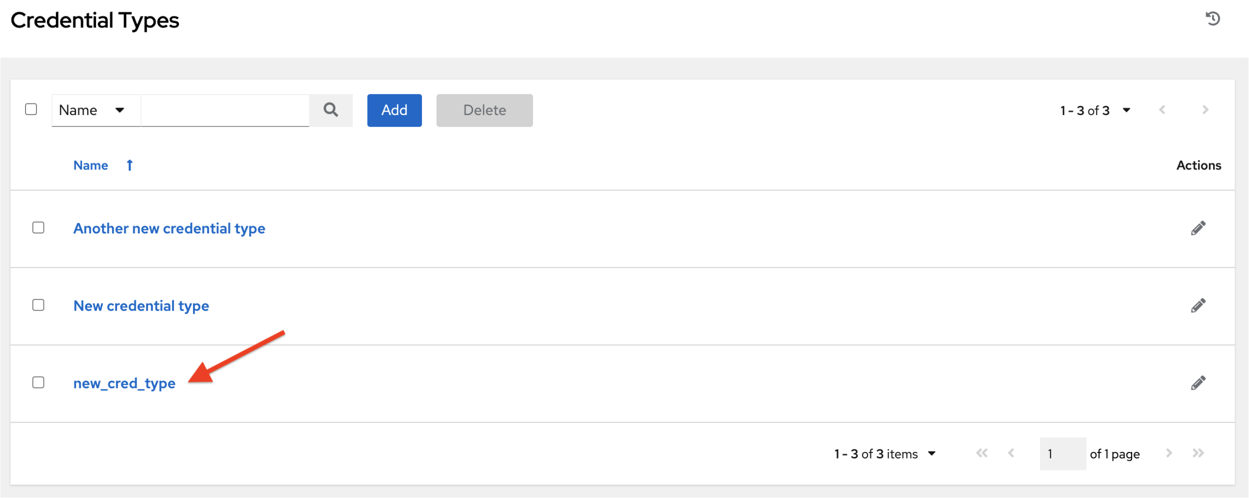
Inputs

```
{
  "fields": [{
    "id": "cert",
    "label": "Certificate",
    "type": "string"
  },{
    "id": "key",
    "label": "Key",
    "type": "string"
  }]
}
```

Injectors

```
 {
  "file": {
   "template.cert_file": "[mycert]\n{{ cert }}",
   "template.key_file": "[mykey]\n{{ key }}"
 },
 "env": {
   "MY_CERT_INI_FILE": "{{ tower.filename.cert_file }}",
   "MY_KEY_INI_FILE": "{{ tower.filename.key_file }}"
 }
}
```

5. Click **Save** when done.
6. Scroll down to the bottom of the screen and your newly created credential type appears on the list of credential types:

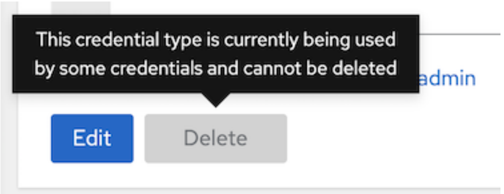


Credential Types

	Name ↑	Actions
□	Another new credential type	✎
□	New credential type	✎
□	new_cred_type	✎

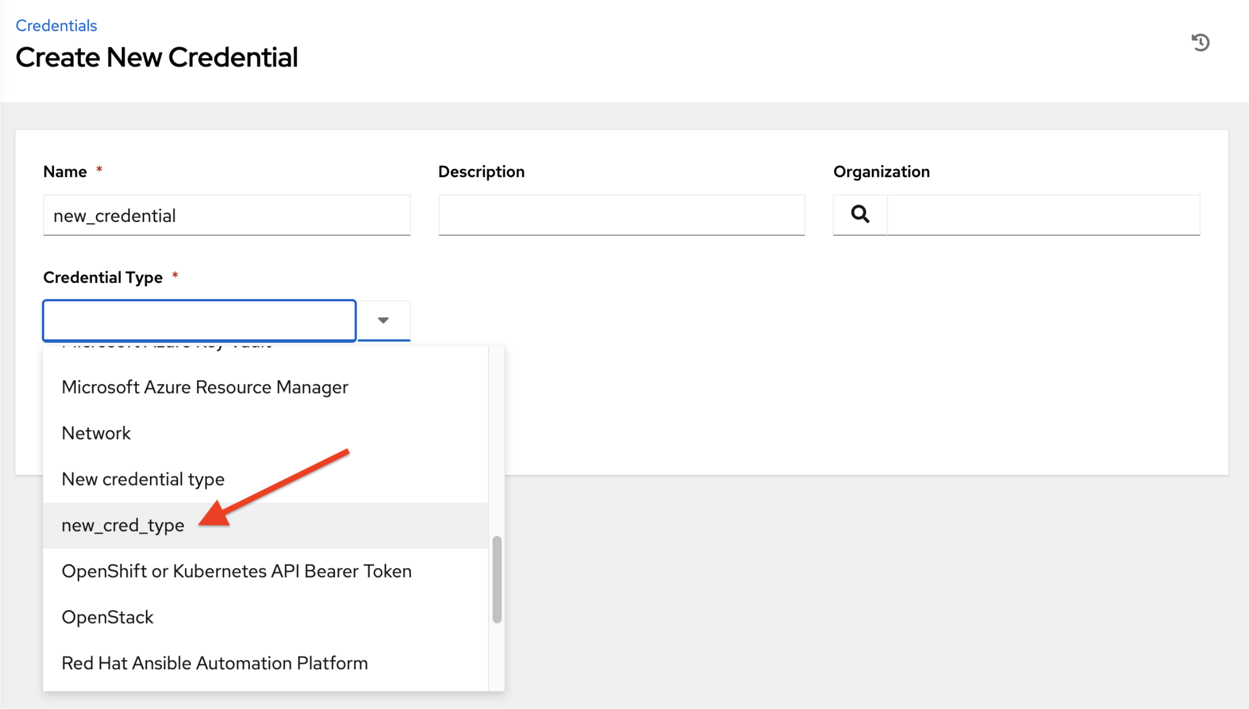
Click ✎ to modify the credential type options under the Actions column.

Note: In the Edit screen, you can modify the details or delete the credential. If the **Delete** button is grayed out, it is indication that the credential type that is being used by a credential, and you must delete the credential type from all the credentials that use it before you can delete it. Below is an example of such a message:



This credential type is currently being used by some credentials and cannot be deleted

7. Verify that the newly created credential type can be selected from the **Credential Type** selection window when creating a new credential:


Credentials
Create New Credential

Name *

new_credential

Description

Organization

Credential Type *

Microsoft Azure Key Vault
Microsoft Azure Resource Manager
Network
New credential type
new_cred_type
OpenShift or Kubernetes API Bearer Token
OpenStack
Red Hat Ansible Automation Platform

For details on how to create a new credential, see *Credentials*.

CHAPTER

TWELVE

SECRET MANAGEMENT SYSTEM

Users and admins upload machine and cloud credentials so that automation can access machines and external services on their behalf. By default, sensitive credential values (such as SSH passwords, SSH private keys, API tokens for cloud services) are stored in the database after being encrypted. With external credentials backed by credential plugins, you can map credential fields (like a password or an SSH Private key) to values stored in a *secret management system* instead of providing them to the controller directly. automation controller provides a secret management system that include integrations for:

- Centrify Vault Credential Provider Lookup
- CyberArk Application Identity Manager (AIM)
- CyberArk Conjur
- HashiCorp Vault Key-Value Store (KV)
- HashiCorp Vault SSH Secrets Engine
- Microsoft Azure Key Management System (KMS)
- Thycotic DevOps Secrets Vault
- Thycotic Secret Server

These external secret values will be fetched prior to running a playbook that needs them. For more information on specifying these credentials in the User Interface, see *Credentials*.

12.1 Configure and link secret lookups

When configuring automation controller to pull a secret from a 3rd-party system, it is in essence linking credential fields to external systems. To link a credential field to a value stored in an external system, select the external credential corresponding to that system and provide *metadata* to look up the desired value. The metadata input fields are part of the *external credential type* definition of the *source credential*.

Automation controller provides a *credential plugin* interface for developers, integrators, admins, and power-users with the ability to add new external credential types to extend it to support other secret management systems. For more detail, see the development docs for credential plugins.

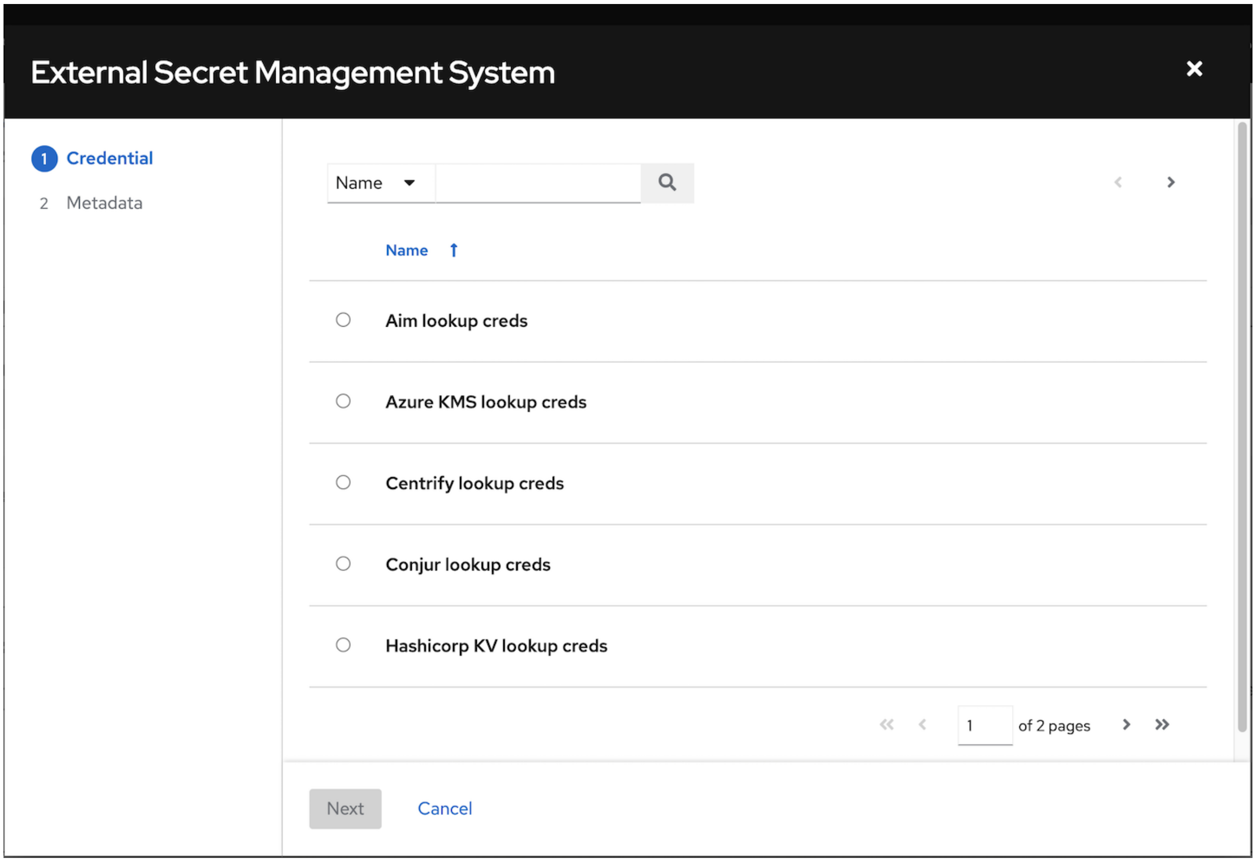
Use the automation controller User Interface to configure and use each of the supported 3-party secret management systems.

1. First, create an external credential for authenticating with the secret management system. At minimum, provide a name for the external credential and select one of the following for the **Credential Type**:

- *Metadata for credential input sources*
- *Centrify Vault Credential Provider Lookup*
- *CyberArk AIM Credential Provider Lookup*
- *CyberArk Conjur Secret Lookup*
- *HashiCorp Vault Secret Lookup*
- *HashiCorp Vault Signed SSH*
- *Microsoft Azure Key Vault*
- *Thycotic DevOps Secrets Vault*
- *Thycotic Secret Server*

2. Navigate to the credential form of the target credential and link one or more input fields to the external credential along with metadata for locating the secret in the external system. In this example, the *Demo Credential* is the target credential.

3. For any of the fields below the **Type Details** area that you want to link to the external credential, click the 🔑 button of the input field. You are prompted to set the input source to use to retrieve your secret information.



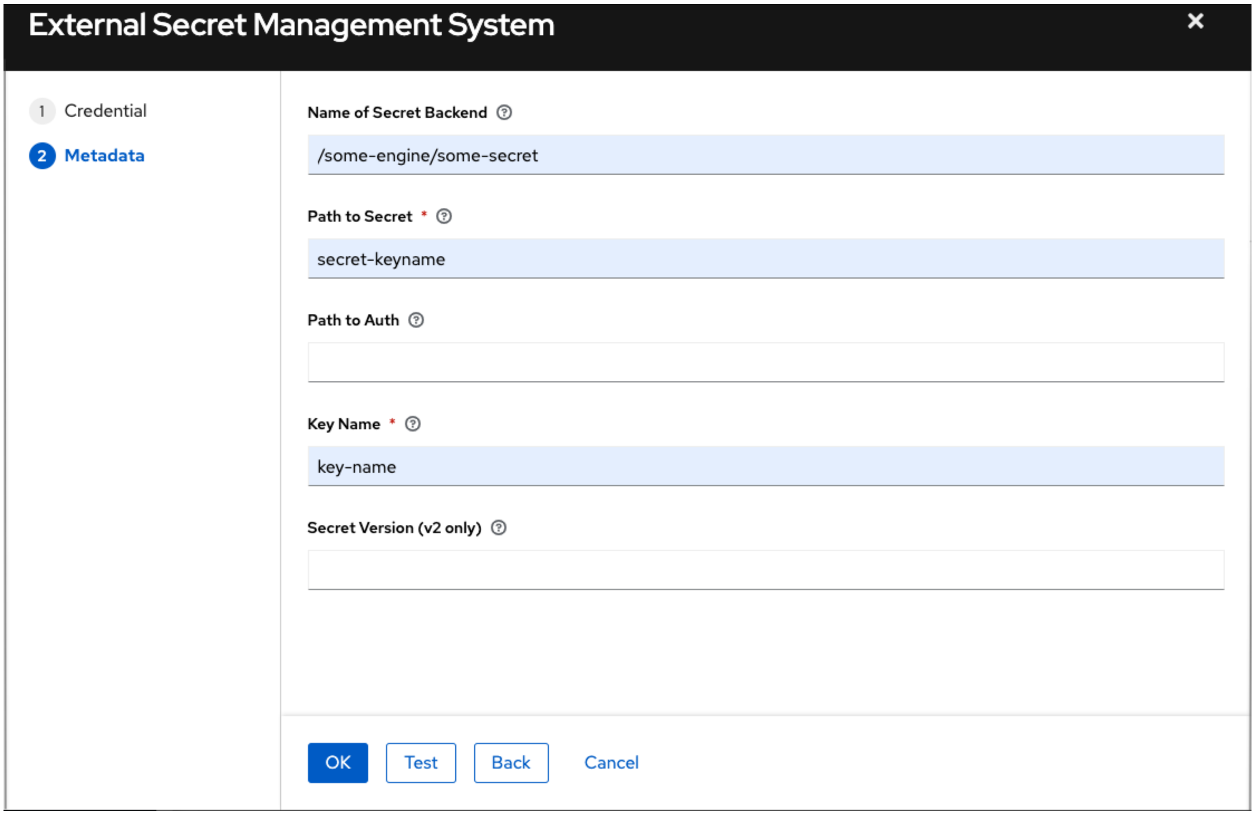
External Secret Management System

1 Credential

2 Metadata

	Name ↑
○	Aim lookup creds
○	Azure KMS lookup creds
○	Centrify lookup creds
○	Conjur lookup creds
○	Hashicorp KV lookup creds

4. Select the credential you want to link to, and click **Next**. This takes you to the **Metadata** tab of the input source. This example shows the Metadata prompt for HashiVault Secret Lookup. Metadata is specific to the input source you select. See the *Metadata for credential input sources* table for details.



External Secret Management System

Name of Secret Backend: /some-engine/some-secret

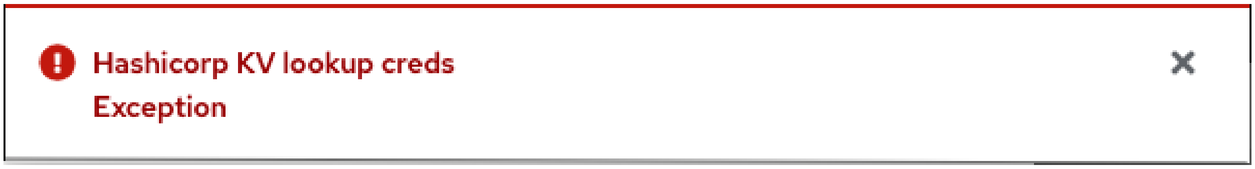
Path to Secret *: secret-keyname

Path to Auth:

Key Name *: key-name

Secret Version (v2 only):

5. Click **Test** to verify connection to the secret management system. If the lookup is unsuccessful, an error message like this one displays:



Hashicorp KV lookup creds Exception

6. When done, click **OK**. This closes the prompt window and returns you to the Details screen of your target credential. **Repeat these steps**, starting with *step 3 above* to complete the remaining input fields for the target credential. By linking the information in this manner, automation controller retrieves sensitive information, such as username, password, keys, certificates, and tokens from the 3rd-party management systems and populates that data into the remaining fields of the target credential form.

7. If necessary, supply any information manually for those fields that do not use linking as a way of retrieving sensitive information. Refer to the appropriate *Credential Types* for more detail about each of the fields.

8. Click **Save** when done.

12.1.1 Metadata for credential input sources

Centrify Vault Credential Provider Lookup

Metadata	Description
Account Name (Required)	Name of the system account or domain associated with Centrify Vault.
System Name	Specify the name used by the Centrify portal.

CyberArk AIM

Metadata	Description
Object Query (Required)	Lookup query for the object.
Object Query Format	Select `Exact` for a specific secret name, or `Regexp` for a secret that has a dynamically generated name.
Reason	If required per the object's policy, supply a reason for checking out the secret, as CyberArk logs those.

CyberArk Conjur

Metadata	Description
Secret Identifier	The identifier for the secret.
Secret Version	Specify a version of the secret, if necessary, otherwise, leave it empty to use the latest version.

HashiVault Secret Lookup

Metadata	Description
Name of Secret Backend	Specify the name of the KV backend to use. Leave it blank to use the first path segment of the **Path to Secret** field instead.
Path to Secret (required)	Specify the path to where the secret information is stored; for example, `/path/username`.
Key Name (required)	Specify the name of the key to look up the secret information.
Secret Version (V2 Only)	Specify a version if necessary, otherwise, leave it empty to use the latest version.

HashiCorp Signed SSH

Metadata	Description
Unsigned Public Key (required)	Specify the public key of the cert you want to get signed. It needs to be present in the authorized keys file of the target host(s).
Path to Secret (required)	Specify the path to where the secret information is stored; for example, `/path/username`.
Role Name (required)	A role is a collection of SSH settings and parameters that are stored in Hashi vault. Typically, you can specify a couple of them with different privileges, timeouts, etc. So you could have a role that is allowed to get a cert signed for root, and other less privileged ones, for example.
Valid Principals	Specify a user (or users) other than the default, that you are requesting vault to authorize the cert for the stored key. Hashi vault has a default user for whom it signs (e.g., ec2-user).

Azure KMS

Metadata	Description
Secret Name (required)	The actual name of the secret as it is referenced in Azure's Key vault app.
Secret Version	Specify a version of the secret, if necessary, otherwise, leave it empty to use the latest version.

Thycotic DevOps Secrets Vault

Metadata	Description
Secret Path (required)	Specify the path to where the secret information is stored (e.g., /path/username).

Thycotic Secret Server

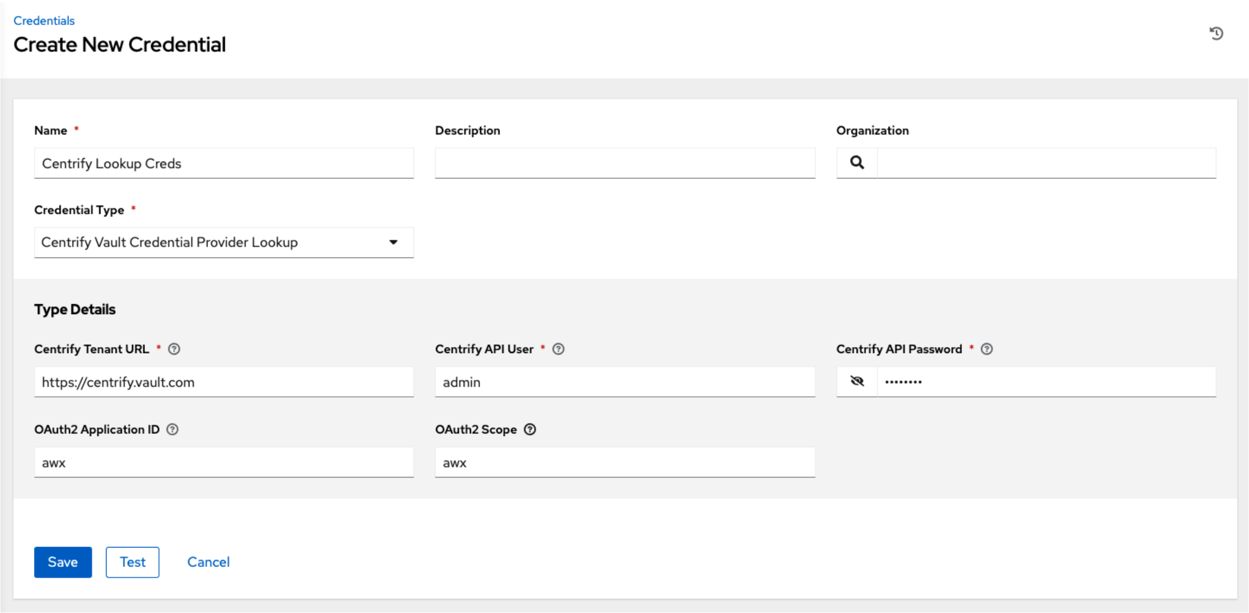
Metadata	Description
Secret ID (required)	The identifier for the secret.
Secret Field	Specify the field to be used from the secret.

12.1.2 Centrify Vault Credential Provider Lookup

You need the Centrify Vault web service running to store secrets in order for this integration to work. When **Centrify Vault Credential Provider Lookup** is selected for **Credential Type**, provide the following metadata to properly configure your lookup:

- **Centrify Tenant URL** (required): provide the URL used for communicating with Centrify's secret management system
- **Centrify API User** (required): provide the username
- **Centrify API Password** (required): provide the password
- **OAuth2 Application ID** : specify the identifier given associated with the OAuth2 client
- **OAuth2 Scope** : specify the scope of the OAuth2 client

Below shows an example of a configured CyberArk AIM credential.



Name * Centrify Lookup Creds

Description

Organization

Credential Type * Centrify Vault Credential Provider Lookup

Centrify Tenant URL * https://centrify.vault.com

Centrify API User * admin

Centrify API Password * ••••••••

OAuth2 Application ID awx

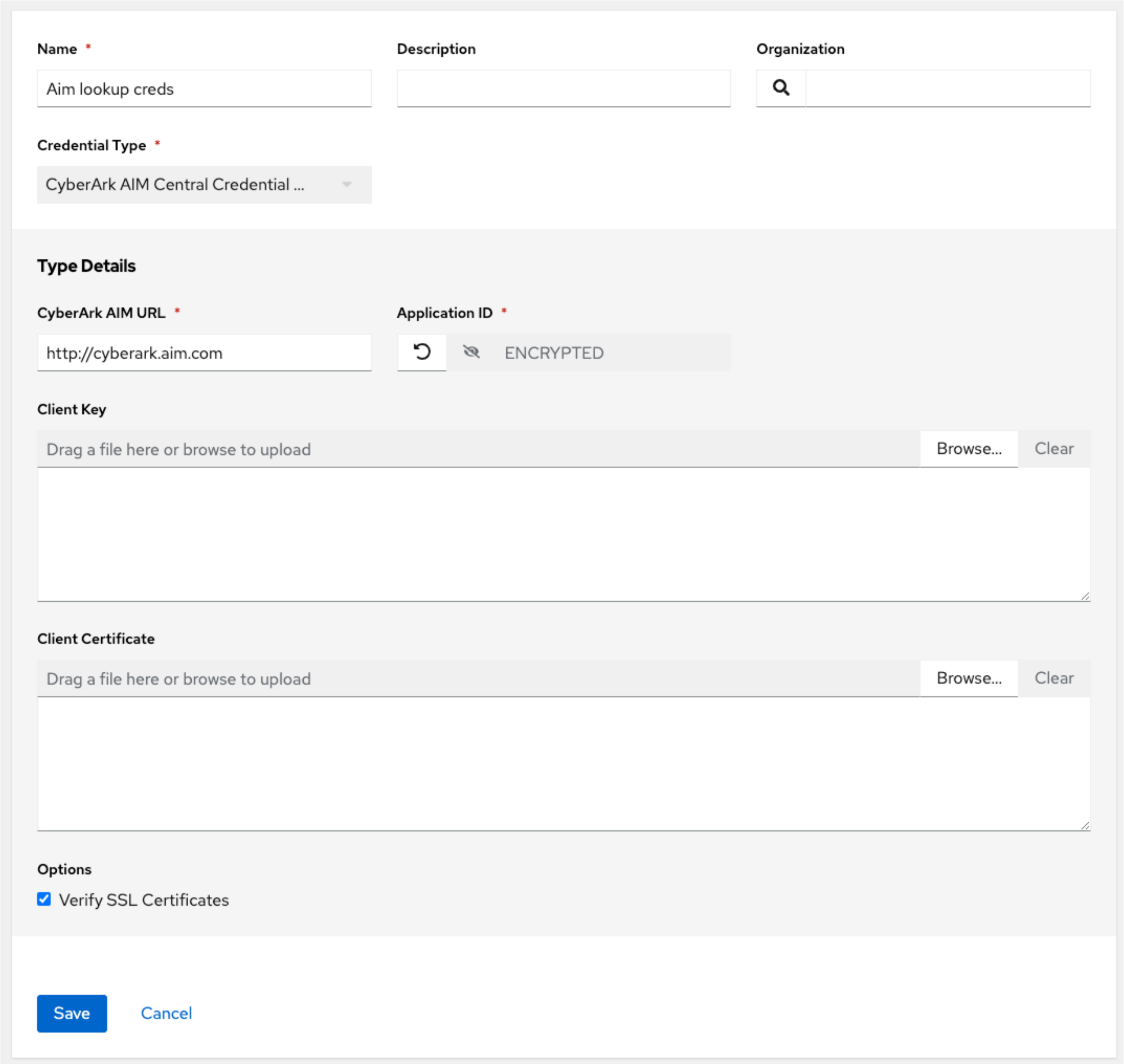
OAuth2 Scope awx

12.1.3 CyberArk AIM Credential Provider Lookup

You need the CyberArk Central Credential Provider web service running to store secrets in order for this integration to work. When **CyberArk AIM Credential Provider Lookup** is selected for **Credential Type**, provide the following metadata to properly configure your lookup:

- **CyberArk AIM URL** (required): provide the URL used for communicating with CyberArk AIM's secret management system
- **Application ID** (required): specify the identifier given by CyberArk AIM services
- **Client Key**: paste the client key if provided by CyberArk
- **Client Certificate**: include the `BEGIN CERTIFICATE` and `END CERTIFICATE` lines when pasting the certificate, if provided by CyberArk
- **Verify SSL Certificates**: this option is only available when the URL uses HTTPS. Check this option to verify the server's SSL certificate is valid and trusted. Environments that use internal or private CA's should leave this option unchecked to disable verification.

Below shows an example of a configured CyberArk AIM credential.



Name * Aim lookup creds

Description

Organization

Credential Type * CyberArk AIM Central Credential ...

Type Details

CyberArk AIM URL * http://cyberark.aim.com

Application ID * ENCRYPTED

Client Key

Client Certificate

Options

☑ Verify SSL Certificates

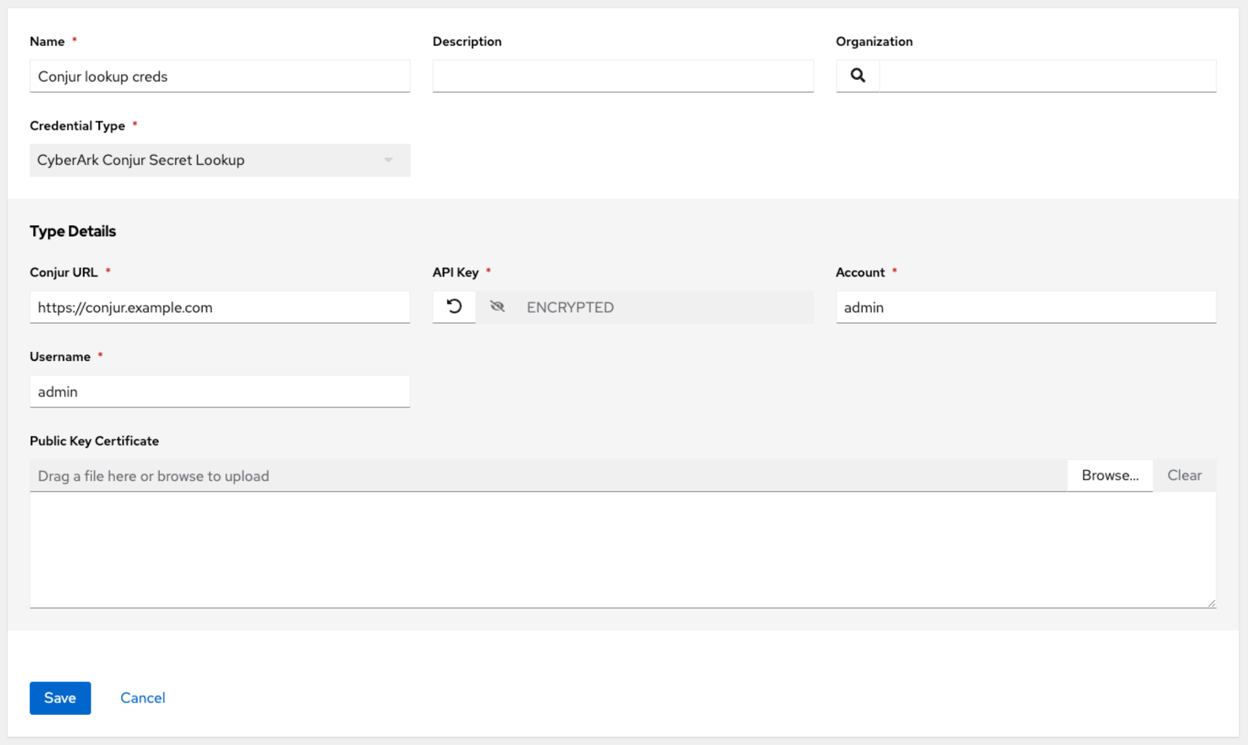
Drag a file here or browse to upload
Browse...
Clear
Drag a file here or browse to upload
Browse...
Clear
Save
Cancel

12.1.4 CyberArk Conjur Secret Lookup

When **CyberArk Conjur Secret Lookup** is selected for **Credential Type**, provide the following metadata to properly configure your lookup:

- **Conjur URL** (required): provide the URL used for communicating with CyberArk Conjur's secret management system
- **API Key** (required): provide the key given by your Conjur admin
- **Account** (required): the organization's account name
- **Username** (required): the specific authenticated user for this service
- **Public Key Certificate**: include the `BEGIN CERTIFICATE` and `END CERTIFICATE` lines when pasting the public key, if provided by CyberArk

Below shows an example of a configured CyberArk Conjur credential.



Name *	Description	Organization
Conjur lookup creds		
Credential Type *		
CyberArk Conjur Secret Lookup		

Type Details

Conjur URL *	API Key *	Account *
https://conjur.example.com	ENCRYPTED	admin
Username *		
admin		

Public Key Certificate

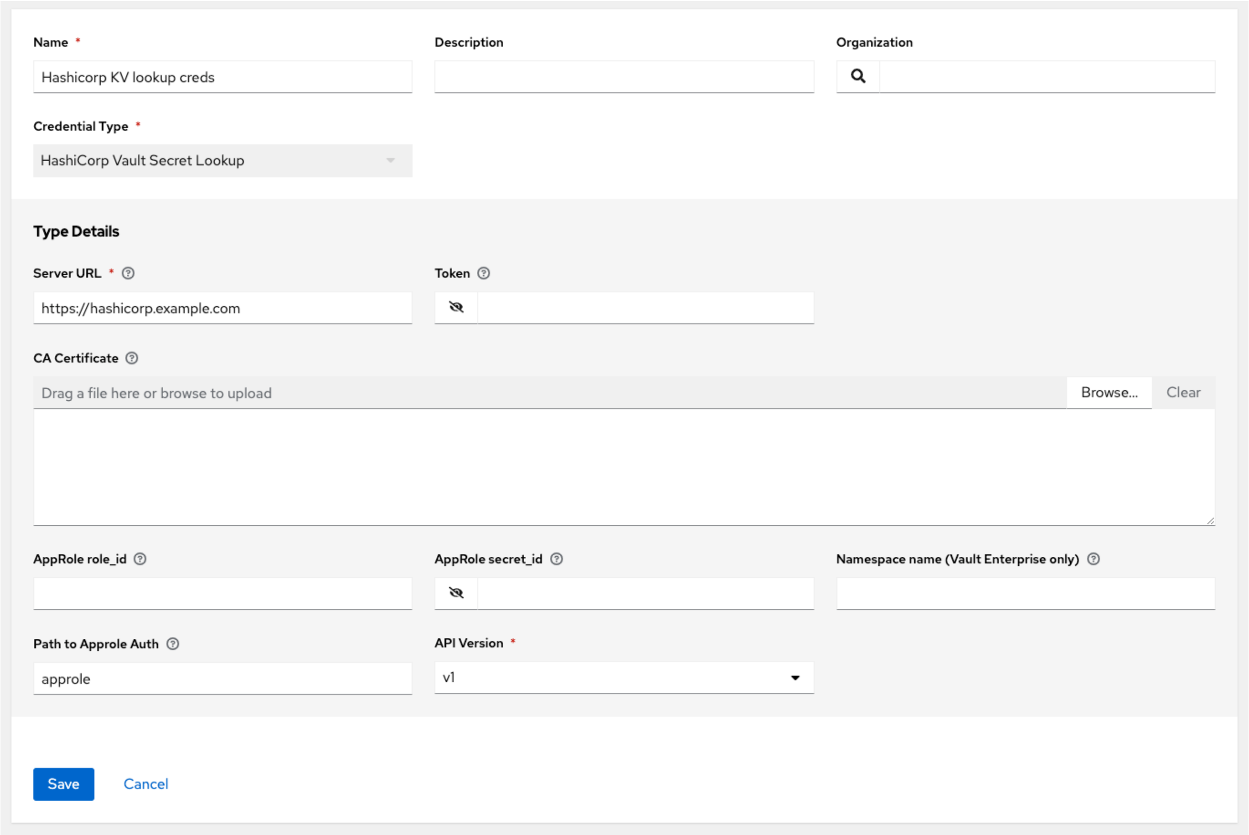
Drag a file here or browse to upload

12.1.5 HashiCorp Vault Secret Lookup

When **HashiCorp Vault Secret Lookup** is selected for **Credential Type**, provide the following metadata to properly configure your lookup:

- **Server URL** (required): provide the URL used for communicating with HashiCorp Vault's secret management system
- **Token**: specify the access token used to authenticate HashiCorp's server
- **CA Certificate**: specify the CA certificate used to verify HashiCorp's server
- **Approle Role_ID**: specify the ID for Approle authentication
- **Approle Secret_ID**: specify the corresponding secret ID for Approle authentication
- **Path to Approle Auth**: specify a path if other than the default path of `/approle`
- **API Version** (required): select v1 for static lookups and v2 for versioned lookups

For more detail about Approle and its fields, refer to the Vault documentation for Approle Auth Method. Below shows an example of a configured HashiCorp Vault Secret Lookup credential.



Name * Hashicorp KV lookup creds

Description

Organization

Credential Type * HashiCorp Vault Secret Lookup

Type Details

Server URL * https://hashicorp.example.com

Token

CA Certificate

Drag a file here or browse to upload

AppRole role_id

AppRole secret_id

Namespace name (Vault Enterprise only)

Path to Approle Auth approle

API Version * v1

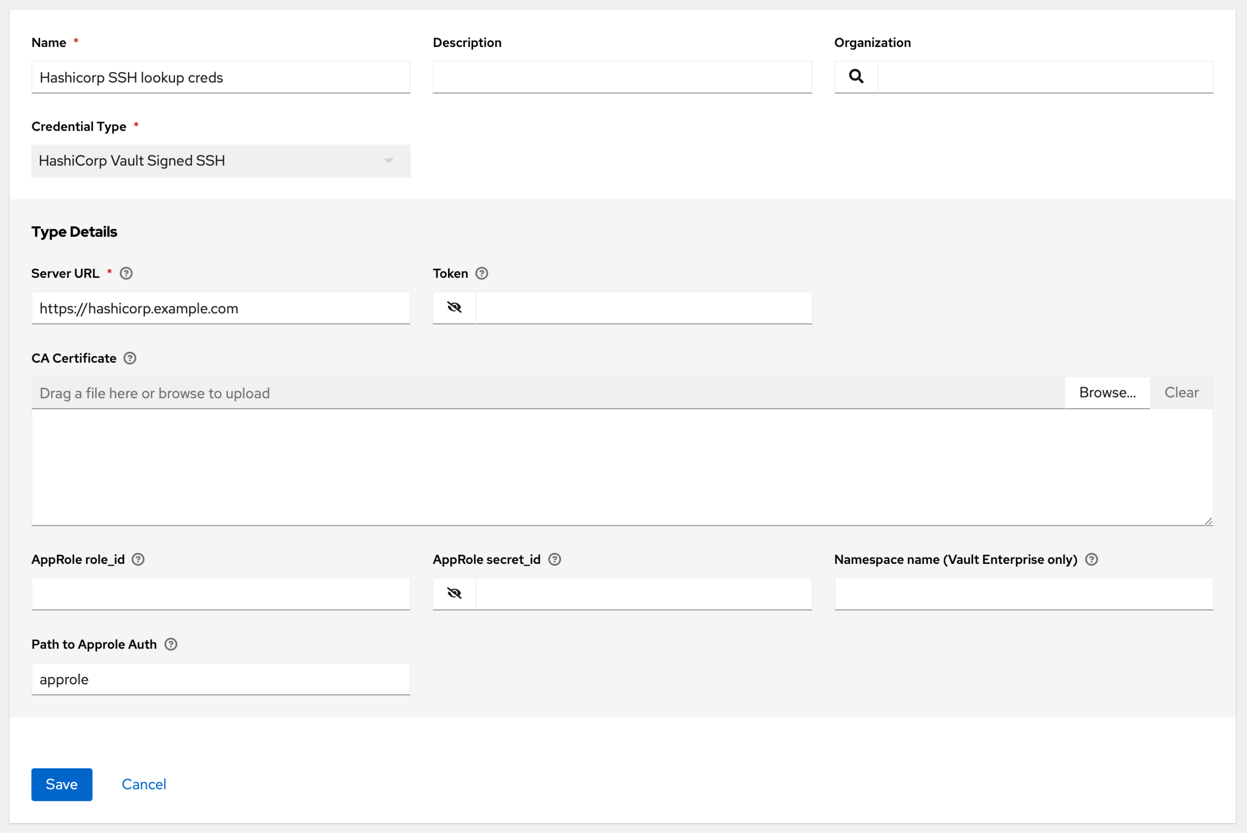
12.1.6 HashiCorp Vault Signed SSH

When **HashiCorp Vault Signed SSH** is selected for **Credential Type**, provide the following metadata to properly configure your lookup:

- **Server URL** (required): provide the URL used for communicating with HashiCorp Signed SSH's secret management system
- **Token**: specify the access token used to authenticate HashiCorp's server
- **CA Certificate**: specify the CA certificate used to verify HashiCorp's server
- **Approle Role_ID**: specify the ID for Approle authentication
- **Approle Secret_ID**: specify the corresponding secret ID for Approle authentication
- **Path to Approle Auth**: specify a path if other than the default path of `/approle`

For more detail about Approle and its fields, refer to the Vault documentation for Approle Auth Method.

Below shows an example of a configured HashiCorp SSH Secrets Engine credential.



Name *

Hashicorp SSH lookup creds

Description

Organization

Credential Type *

HashiCorp Vault Signed SSH

Type Details

Server URL *

https://hashicorp.example.com

Token

CA Certificate

Drag a file here or browse to upload

AppRole role_id

AppRole secret_id

Namespace name (Vault Enterprise only)

Path to Approle Auth

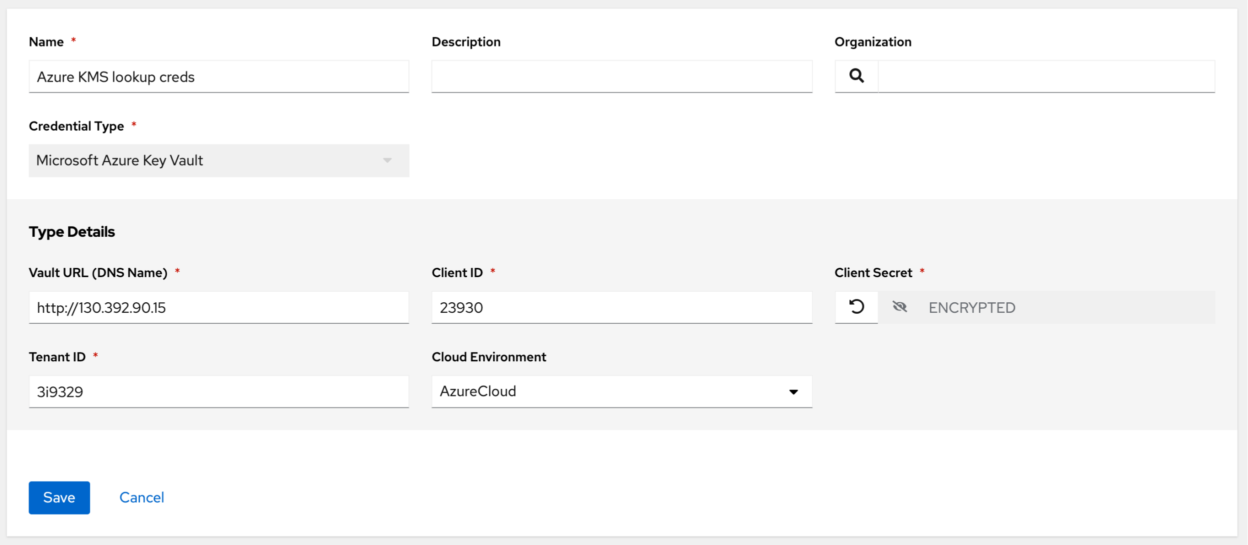
approle

12.1.7 Microsoft Azure Key Vault

When **Microsoft Azure Key Vault** is selected for **Credential Type**, provide the following metadata to properly configure your lookup:

- **Vault URL (DNS Name)** (required): provide the URL used for communicating with MS Azure's key management system
- **Client ID** (required): provide the identifier as obtained by the Azure Active Directory
- **Client Secret** (required): provide the secret as obtained by the Azure Active Directory
- **Tenant ID** (required): provide the unique identifier that is associated with an Azure Active Directory instance within an Azure subscription
- **Cloud Environment**: select the applicable cloud environment to apply

Below shows an example of a configured Microsoft Azure KMS credential.



Name *	Description	Organization
Azure KMS lookup creds		
Credential Type *		
Microsoft Azure Key Vault		

Type Details

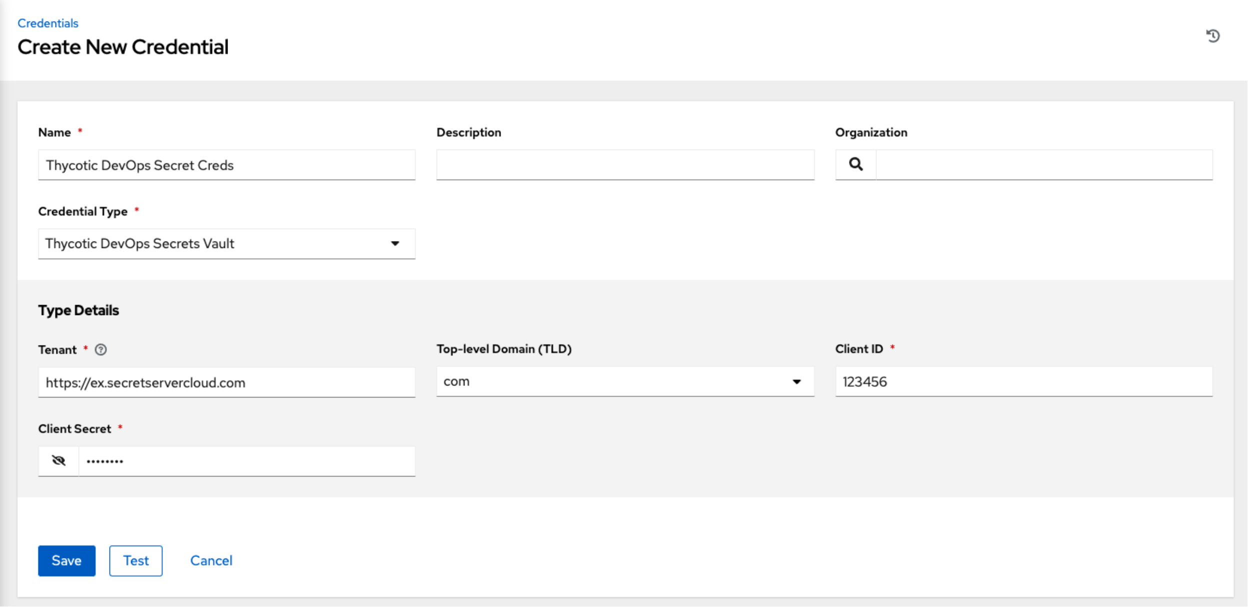
Vault URL (DNS Name) *	Client ID *	Client Secret *
http://130.392.90.15	23930	ENCRYPTED
Tenant ID *	Cloud Environment	
3i9329	AzureCloud	

12.1.8 Thycotic DevOps Secrets Vault

When **Thycotic DevOps Secrets Vault** is selected for **Credential Type**, provide the following metadata to properly configure your lookup:

- **Tenant** (required): provide the URL used for communicating with Thycotic's secret management system
- **Top-level Domain (TLD)** : provide the top-level domain designation (e.g., com, edu, org) associated with the secret vault you want to integrate
- **Client ID** (required): provide the identifier as obtained by the Thycotic secret management system
- **Client Secret** (required): provide the secret as obtained by the Thycotic secret management system

Below shows an example of a configured Thycotic DevOps Secrets Vault credential.



Create New Credential

Name *	Description	Organization
Thycotic DevOps Secret Creds		
Credential Type *		
Thycotic DevOps Secrets Vault		

Type Details

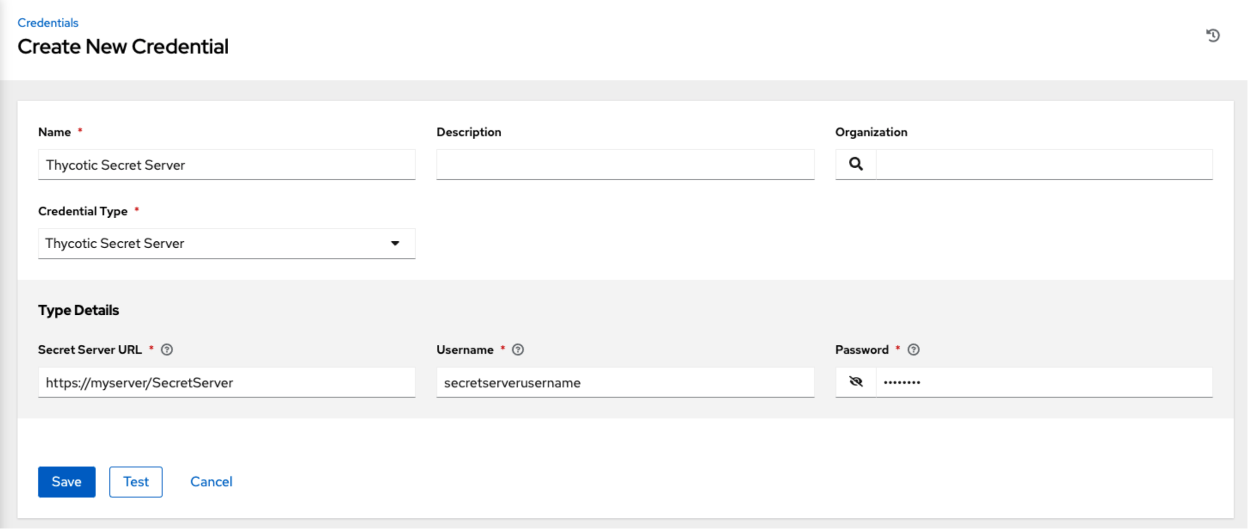
Tenant *	Top-level Domain (TLD)	Client ID *
https://ex.secretservercloud.com	com	123456
Client Secret *		
••••••••		

12.1.9 Thycotic Secret Server

When **Thycotic Secrets Server** is selected for **Credential Type**, provide the following metadata to properly configure your lookup:

- **Secret Server URL** (required): provide the URL used for communicating with the Thycotic Secrets Server management system
- **Username** (required): specify the authenticated user for this service
- **Password** (required): provide the password associated with the user

Below shows an example of a configured Thycotic Secret Server credential.



Credentials

Create New Credential

Name *: Thycotic Secret Server

Description:

Organization:

Credential Type *: Thycotic Secret Server

Type Details

Secret Server URL *: https://myserver/SecretServer

Username *: secretserverusername

Password *: ••••••••

CHAPTER

THIRTEEN

APPLICATIONS

Creating and configuring token-based authentication for external applications is available starting in automation controller 3.3. This makes it easier for external applications such as ServiceNow and Jenkins to integrate with automation controller. OAuth 2 allows you to use tokens to share certain data with an application without disclosing login information, and furthermore, these tokens can be scoped as “read-only”. You create an application that is representative of the external application you are integrating with, then use it to create tokens for that application to use on behalf of the users of the external application.

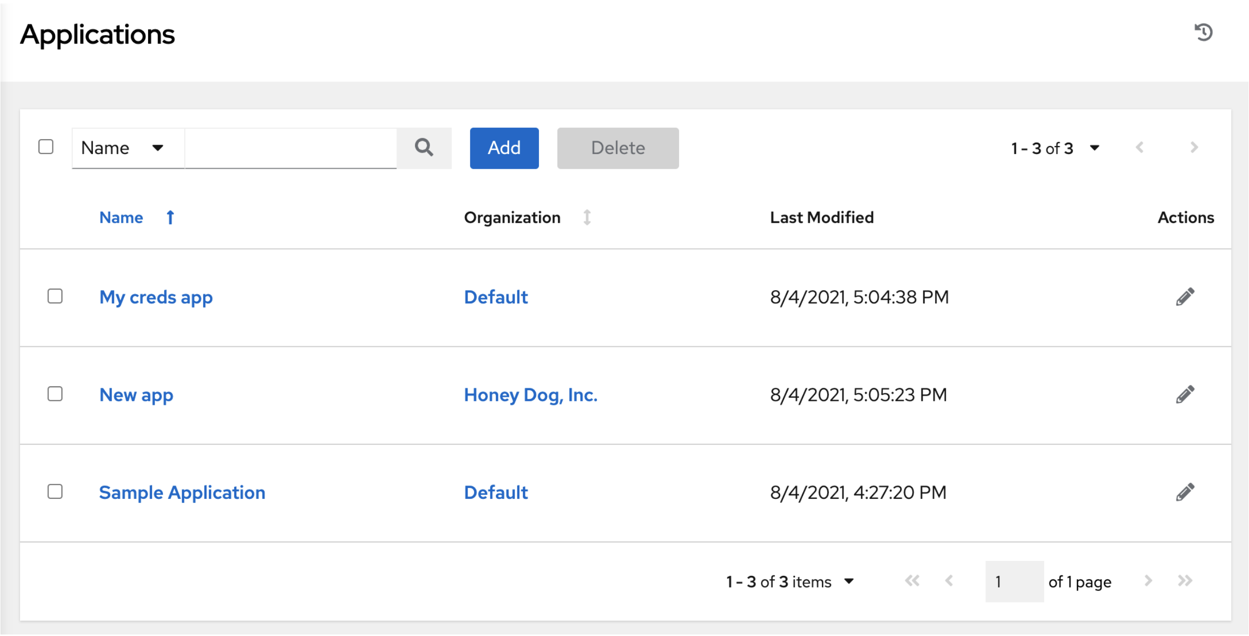
Having these tokens associated to an application resource gives you the ability to manage all tokens issued for a particular application more easily. By separating token issuance under Applications, you can revoke all tokens based on the Application without having to revoke all tokens in the system.

When integrating an external web app with automation controller that web app may need to create OAuth2 Tokens on behalf of users in that other web app. Creating an application with the Authorization Code grant type is the preferred way to do this because:

- external applications can obtain a token for users, using their credentials
- compartmentalized tokens issued for a particular application, allows those tokens to be easily managed (revoke all tokens associated with that application, for example)

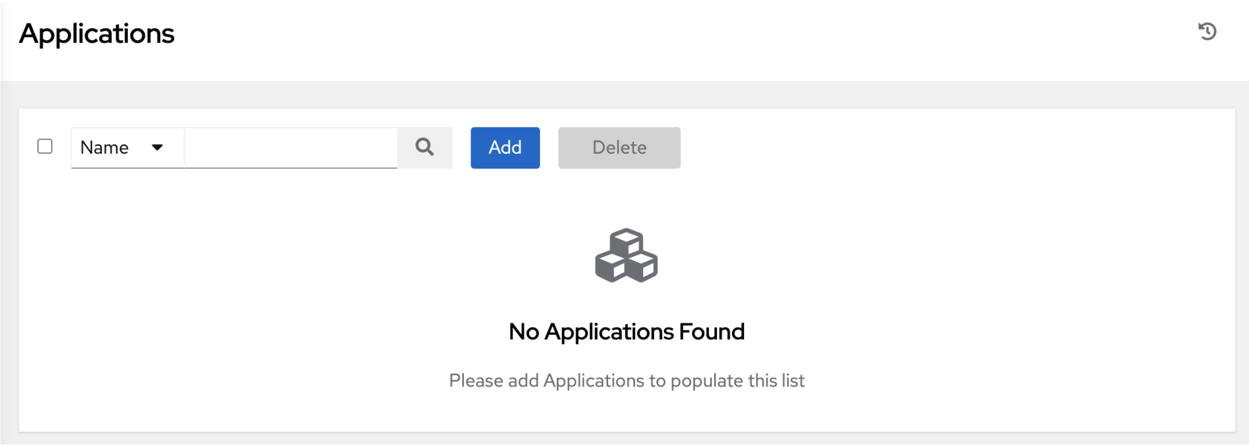
13.1 Getting Started with Applications

Access the Applications page by clicking **Applications** from the left navigation bar. The Applications page displays a search-able list of all available Applications currently managed by the controller and can be sorted by **Name**.



Name	Organization	Last Modified	Actions
My creds app	Default	8/4/2021, 5:04:38 PM	
New app	Honey Dog, Inc.	8/4/2021, 5:05:23 PM	
Sample Application	Default	8/4/2021, 4:27:20 PM	

If no other applications exist, only a gray box with a message to add applications displays.



No Applications Found

Please add Applications to populate this list

13.2 Create a new application

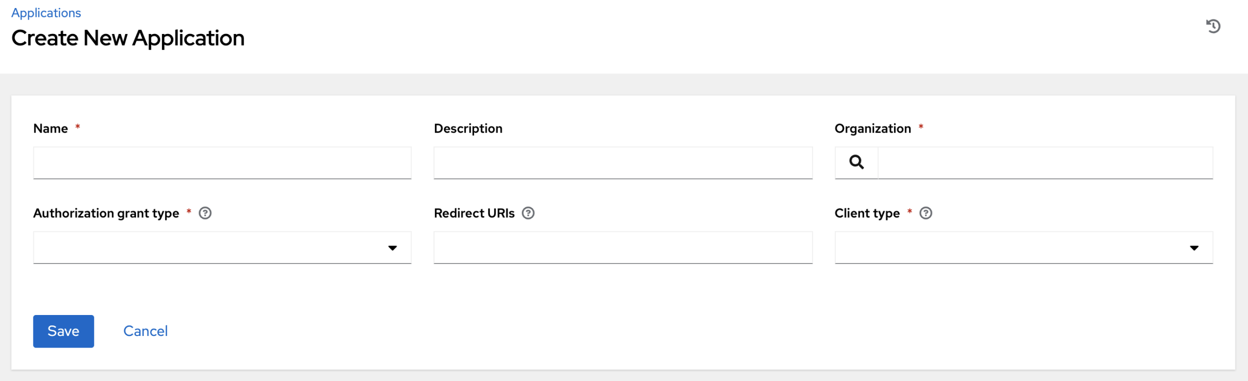
Token-based authentication for users can be configured in the Applications window.

1. In the automation controller User Interface, click **Applications** from the left navigation bar.

The Applications window opens.

2. Click the **Add** button located in the upper right corner of the Applications window.

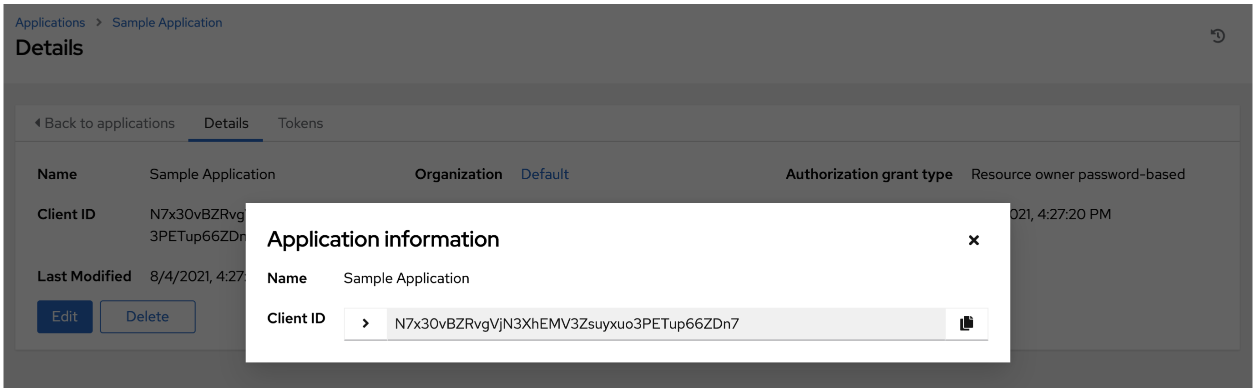
The New Application window opens.



3. Enter the following details in **Create New Application** window:

- **Name** (required): provide a name for the application you want to create
- **Description**: optionally provide a short description for your application
- **Organization** (required): provide an organization for which this application is associated
- **Authorization Grant Type** (required): Select from one of the grant types to use in order for the user to acquire tokens for this application. Refer to grant types in the Applications section of the *Automation Controller Administration Guide*.
- **Redirect URIS**: Provide a list of allowed URIs, separated by spaces. This is required if you specified the grant type to be **Authorization code**.
- **Client Type** (required): Select the level of security of the client device

4. When done, click **Save** or **Cancel** to abandon your changes. Upon saving, the client ID displays in a pop-up window.



Name Sample Application

Organization Default

Authorization grant type Resource owner password-based

Client ID N7x30vBZRvg 3PETup66ZDn

Last Modified 8/4/2021, 4:27

021, 4:27:20 PM

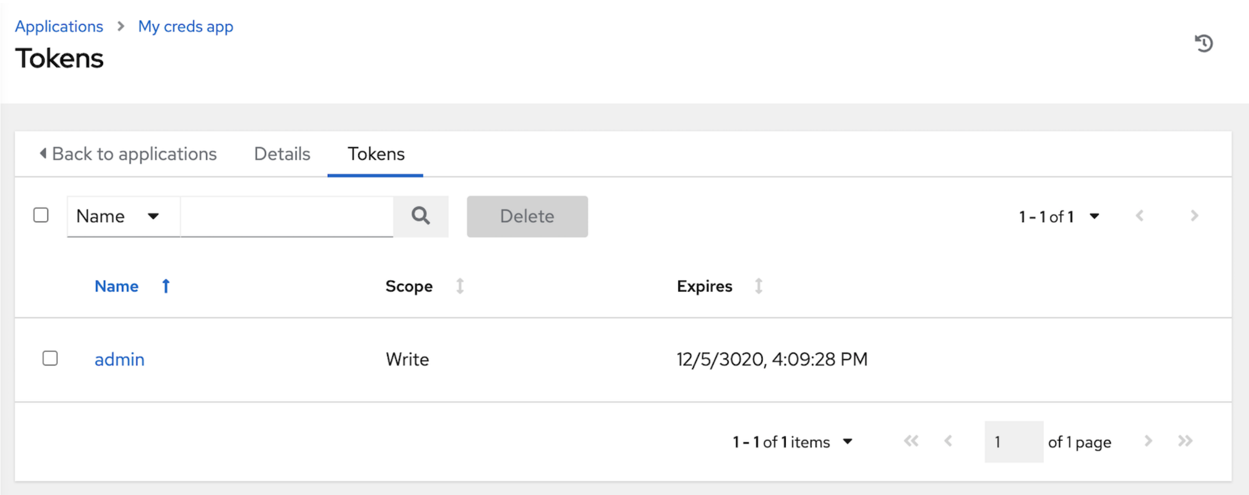
Application information

Name Sample Application

Client ID N7x30vBZRvgVjN3XhEMV3Zsuyxuo3PETup66ZDn7

13.2.1 Applications - Tokens

Selecting the **Tokens** view displays a list of the users that have tokens to access the application.



Name	Scope	Expires
admin	Write	12/5/3020, 4:09:28 PM

Tokens can only access resources that its associated user can access, and can be limited further by specifying the scope of the token.

Add Tokens

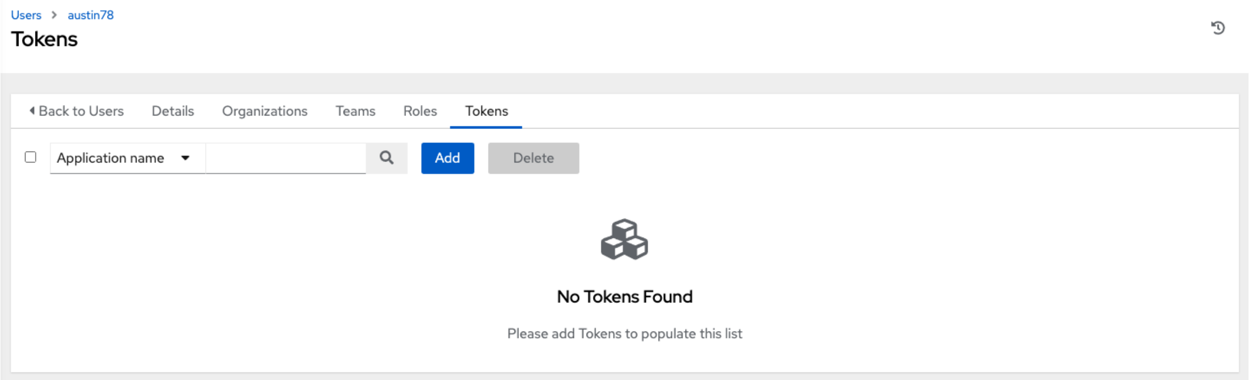
Tokens are added through the Users screen and can be associated with an application at that time. Specifying an application can be performed directly in the User's token settings. You can create a token for *your* user in the Tokens configuration tab, meaning only you can create and see your tokens in your own user screen. To add a token:

1. Access the Users list view by clicking **Users** from the left navigation bar then click on your user to configure your OAuth 2 tokens.

Note: You can only create OAuth 2 Tokens for your user via the API or UI, which means you can only access your own user profile in order to configure or view your tokens. If you are an admin and need to create or remove tokens for other users, see the revoke and create commands in the Token and session management section of the *Automation Controller Administration Guide*.

2. Click the **Tokens** tab from your user's profile.

When no tokens are present, the Tokens screen prompts you to add them:



No Tokens Found

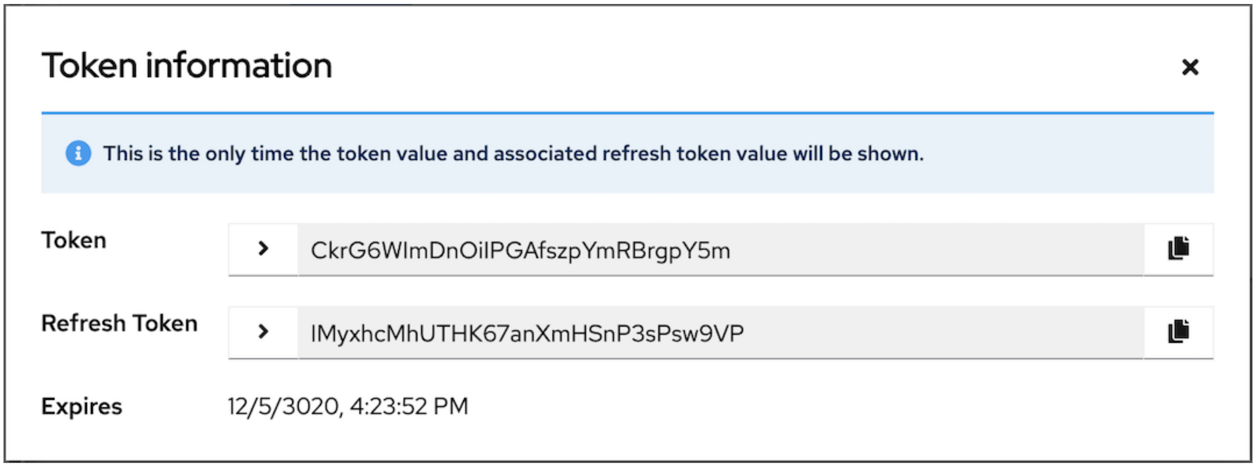
Please add Tokens to populate this list

3. Click the **Add** button, which opens the Create Token window.
4. Enter the following details in Create Token window:

- **Application**: enter the name of the application with which you want to associate your token. Alternatively, you can search for it by clicking the 🔍 button. This opens a separate window that allows you to choose from the available options. Use the Search bar to filter by name if the list is extensive. Leave this field blank if you want to create a Personal Access Token (PAT) that is not linked to any application.
- **Description**: optionally provide a short description for your token.
- **Scope** (required): specify the level of access you want this token to have.

5. When done, click **Save** or **Cancel** to abandon your changes.

After the token is saved, the newly created token for the user displays with the token information and when it expires.



Token information

This is the only time the token value and associated refresh token value will be shown.

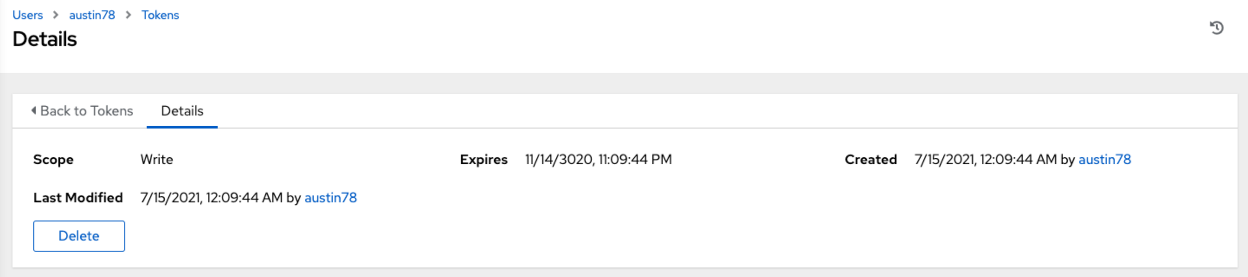
Token: CkrG6WlmDnOiIPGAfszpYmRBrgpY5m

Refresh Token: IMyxhcMhUTHK67anXmHSnP3sPsw9VP

Expires: 12/5/3020, 4:23:52 PM

Note: This is the only time the token value and associated refresh token value will ever be shown.

In the user's profile, the application for which it is assigned to and its expiration displays in the token list view.



Users > austin78 > Tokens

Details

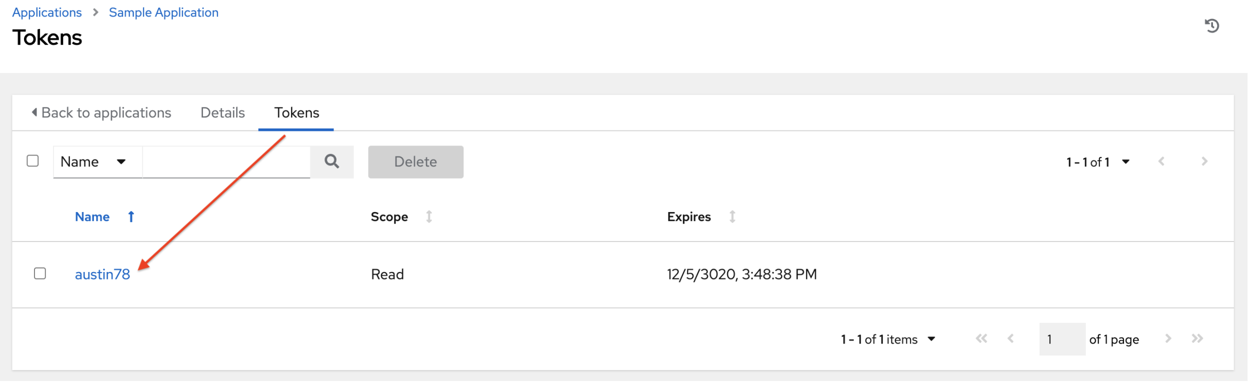
Scope: Write

Expires: 11/14/3020, 11:09:44 PM

Created: 7/15/2021, 12:09:44 AM by austin78

Last Modified: 7/15/2021, 12:09:44 AM by austin78

To verify the application in the example above now shows the user with the appropriate token, go to the **Tokens** tab of the Applications window:



	Name	Scope	Expires
□	austin78	Read	12/5/3020, 3:48:38 PM

CHAPTER

FOURTEEN

EXECUTION ENVIRONMENTS

The ability to build and deploy Python virtual environments for automation has been replaced by Ansible execution environments. Unlike legacy virtual environments, execution environments are container images that make it possible to incorporate system-level dependencies and collection-based content. Each execution environment allows you to have a customized image to run jobs, and each of them contain only what you need when running the job, nothing more.

14.1 Building an Execution Environment

Using Ansible content that depends on non-default dependencies (custom virtual environments) can be tricky. Packages must be installed on each node, play nicely with other software installed on the host system, and be kept in sync. Previously, jobs ran inside of a virtual environment at `/var/lib/awx/venv/ansible` by default, which was pre-loaded with dependencies for ansible-runner and certain types of Ansible content used by the Ansible control machine.

To help simplify this process, container images can be built that serve as Ansible control nodes. These container images are referred to as automation execution environments, which you can create with ansible-builder and then ansible-runner can make use of those images.

14.1.1 Install ansible-builder

In order to build images, either installations of podman or docker is required along with the ansible-builder Python package. The `--container-runtime` option needs to correspond to the Podman/Docker executable you intend to use.

To install from PyPi:

```
$ pip install ansible-builder
```

To install from the mainline development branch:

```
$ pip install https://github.com/ansible/ansible-builder/archive/devel.zip
```

To install from a specific tag or branch, replace `<ref>` in the following example:

```
$ pip install https://github.com/ansible/ansible-builder/archive/<ref>.zip
```

14.1.2 Build an execution environment

Ansible-builder is used to create an execution environment.

An execution environment is expected to contain:

- Ansible
- Ansible Runner
- Ansible Collections
- Python and/or system dependencies of:
  - modules/plugins in collections
  - content in ansible-base
  - custom user needs

Building a new execution environment involves a definition (a `.yml` file) that specifies which content you would like to include in your execution environment, such as collections, Python requirements, and system-level packages. The content from the output generated from migrating to execution environments has some of the required data that can be piped to a file or pasted into this definition file. See Migrate legacy venvs to execution environments for more detail. If you did not migrate from a virtual environment, you can create a definition file with the required data outlined in *Execution environment definition*.

Collection developers can declare requirements for their content by providing the appropriate metadata. For more information, refer to *Collection-level metadata*.

14.1.3 Run the builder

Once you created a definition, use this procedure to build your execution environment.

The `ansible-builder build` command takes an execution environment definition as an input. It outputs the build context necessary for building an execution environment image, and proceeds with building that image. The image can be re-built with the build context elsewhere, and produces the same result. By default, it looks for a file named `execution-environment.yml` in the current directory.

For the illustration purposes, the following example `execution-environment.yml` file is used as a starting point:

```
---
version: 1
dependencies:
  galaxy: requirements.yml
```

The content of `requirements.yml`:

```
---
collections:
  - name: awx.awx
```

To build an execution environment using the files above, run:

```
$ ansible-builder build
...
STEP 7: COMMIT my-awx-ee
--> 09c930f5f6a
```

(continues on next page)

(continued from previous page)

```
09c930f5f6ac329b7ddb321b144a029dbbfcc83bdfc77103968b7f6cdfc7bea2
Complete! The build context can be found at: context
```

In addition to producing a ready-to-use container image, the build context is preserved, which can be rebuilt at a different time and/or location with the tooling of your choice, such as `docker build` or `podman build`.

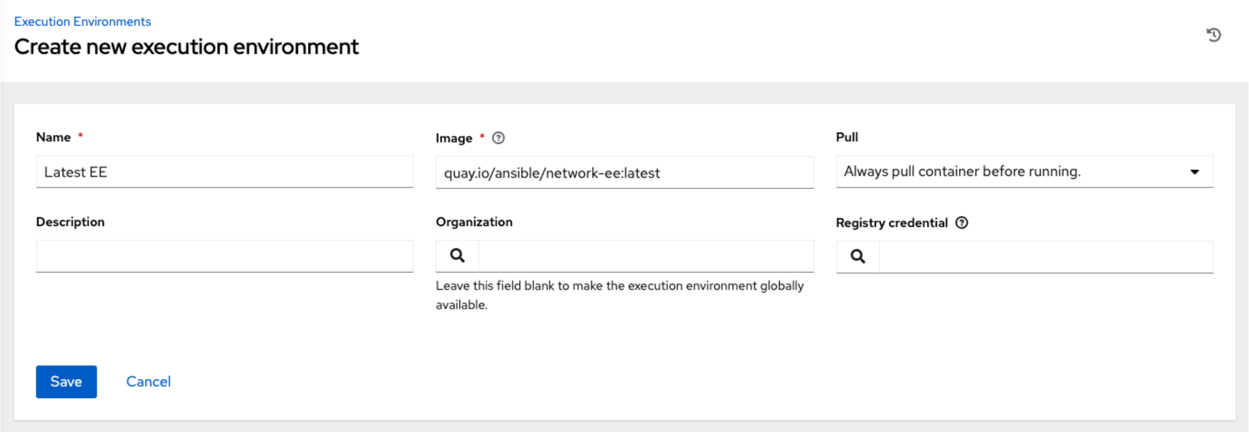
14.2 Use an execution environment in jobs

In order to use an execution environment in a job, a few components are required:

- An execution environment must have been created using ansible-builder. See *Build an execution environment* for detail. Once an execution environment is created, you can use it to run jobs. Use the automation controller user interface to specify the execution environment to use in your job templates.
- Depending on whether an execution environment is made available for global use or tied to an organization, you must have the appropriate level of administrator privileges in order to use an execution environment in a job. Execution environments tied to an organization require Organization administrators to be able to run jobs with those execution environments.
- Before running a job or job template that uses an execution environment that has a credential assigned to it, be sure that the credential contains a username, host, and password.

1. Click **Execution Environments** from the left navigation bar of the controller user interface.
2. Add an execution environment by selecting the **Add** button.
3. Enter the appropriate details into the following fields:

- **Name**: Enter a name for the execution environment (required).
- **Image**: Enter the image name (required). The image name requires its full location (repo), the registry, image name, and version tag in the example format of `quay.io/ansible/awx-ee:latestrepo/project/image-name:tag`.
- **Pull**: optionally choose the type of pull when running jobs:
- **Always pull container before running**: Pulls the latest image file for the container.
- **Only pull the image if not present before running**: Only pulls latest image if none specified.
- **Never pull container before running**: Never pull the latest version of the container image.
- **Description**: optional.
- **Organization**: optionally assign the organization to specifically use this execution environment. To make the execution environment available for use across multiple organizations, leave this field blank.
- **Registry credential**: If the image has a protected container registry, provide the credential to access it.



Execution Environments

Create new execution environment

Name * Latest EE

Image * quay.io/ansible/network-ee:latest

Pull: Always pull container before running.

Description

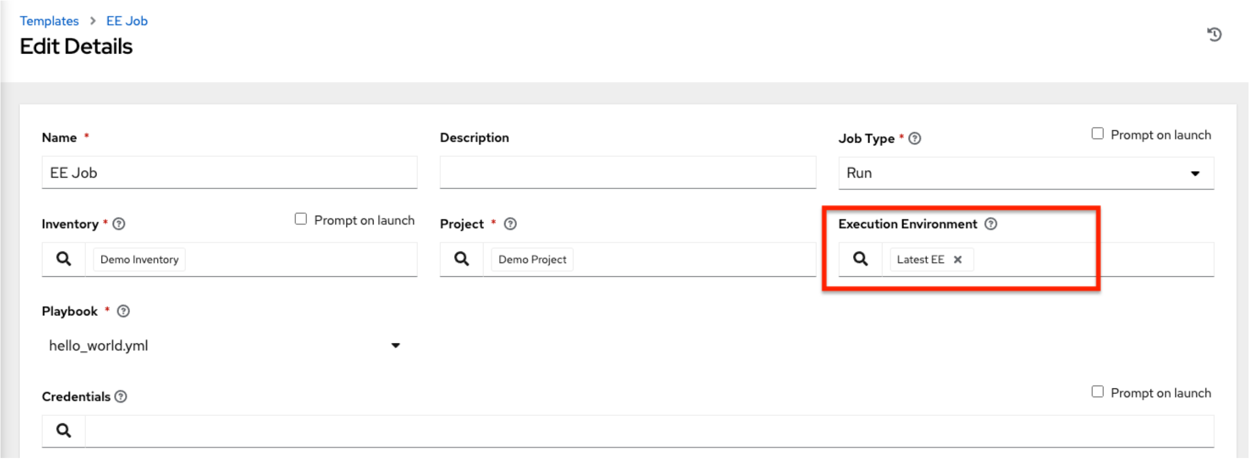
Organization

Leave this field blank to make the execution environment globally available.

Registry credential

4. Click **Save**.

Now your newly added execution environment is ready to be used in a job template. To add an execution environment to a job template, specify it in the **Execution Environment** field of the job template, as shown in the example below. For more information on setting up a job template, see *Job Templates* in the *Automation Controller User Guide*.



Templates > EE Job

Edit Details

Name * EE Job

Description

Job Type * Run □ Prompt on launch

Inventory * Demo Inventory □ Prompt on launch

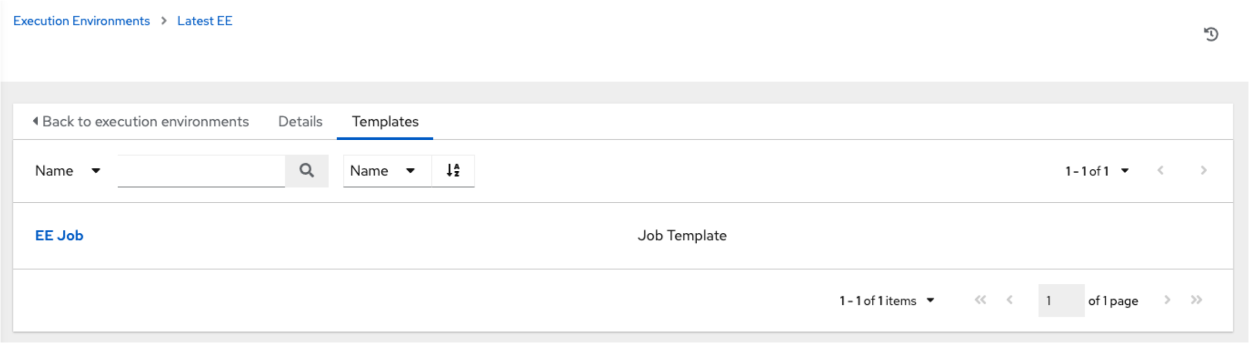
Project * Demo Project

Execution Environment: Latest EE

Playbook * hello_world.yml

Credentials □ Prompt on launch

Once you added an execution environment to a job template, you can see those templates listed in the **Templates** tab of the execution environment:



Execution Environments > Latest EE

EE Job	Job Template

14.3 Execution environment mount options

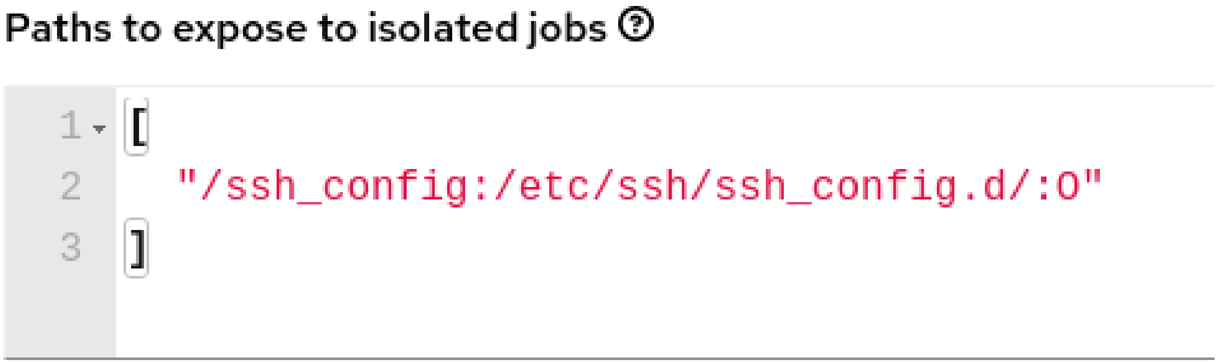
Rebuilding an execution environment is one way to add certs, but inheriting certs from the host provides a more convenient solution. For VM-based installs, the controller automatically mounts the system trust store in the execution environment when jobs run.

Additionally, you may customize execution environment mount options and mount paths in the **Paths to expose to isolated jobs** field of the Job Settings page, where it supports podman-style volume mount syntax. Refer to the Podman documentation for detail.

In some cases where the `/etc/ssh/*` files were added to the execution environment image due to customization of an execution environment, an SSH error may occur. For example, exposing the `/etc/ssh/ssh_config.d:/etc/ssh/ssh_config.d:O` path allows the container to be mounted, but the ownership permissions are not mapped correctly.

If you encounter this error, or have upgraded from an older version of the controller (e.g. 3.8.x), perform the following steps:

1. Change the container ownership on the mounted volume to `root`.
2. In the **Paths to expose to isolated jobs** field of the Job Settings page, using the current example, expose the path as such:



Paths to expose to isolated jobs

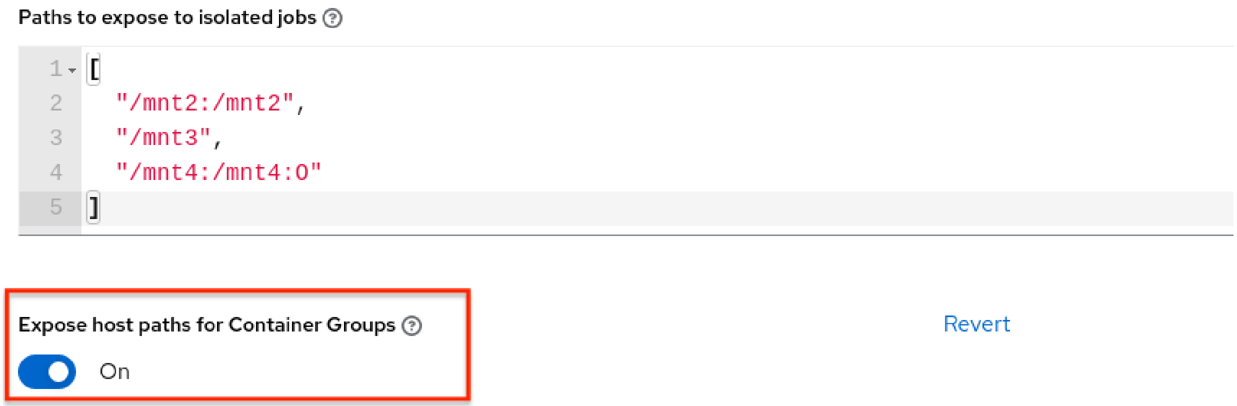
```
[
  "/ssh_config:/etc/ssh/ssh_config.d/:O"
]
```

Note: The `:O` option is only supported for directories. It is highly recommended that you be as specific as possible, especially when specifying system paths. Mounting `/etc` or `/usr` directly have impact that make it difficult to troubleshoot.

This informs podman to run a command similar to the example below, where the configuration is mounted and the `ssh` command works as expected.

```
podman run -v /ssh_config:/etc/ssh/ssh_config.d/:O ...
```

To expose isolated paths in OpenShift or Kubernetes containers as HostPath, assume the following configuration:



Paths to expose to isolated jobs

```
[
  "/mnt2:/mnt2",
  "/mnt3",
  "/mnt4:/mnt4:0"
]
```

Expose host paths for Container Groups

On

Use the **Expose host paths for Container Groups** toggle to enable it.

Once the playbook runs, the resulting Pod spec will display similar to the example below. Note the details of the `volumeMounts` and `volumes` sections.

```
apiVersion: v1
kind: Pod
spec:
  containers:
  - image: registry.redhat.io/ansible-automation-platform-22/ee-minimal-rhel8
    args:
      - ansible-runner
      - worker
      - --private-data-dir=/runner
    volumeMounts:
      - mountPath: /mnt2
        name: volume-0
        readOnly: true
      - mountPath: /mnt3
        name: volume-1
        readOnly: true
      - mountPath: /mnt4
        name: volume-2
        readOnly: true
  volumes:
    - hostPath:
        path: /mnt2
        type: ""
      name: volume-0
    - hostPath:
        path: /mnt3
        type: ""
      name: volume-1
    - hostPath:
        path: /mnt4
        type: ""
      name: volume-2
```

CHAPTER

FIFTEEN

EXECUTION ENVIRONMENT SETUP REFERENCE

This section contains reference information associated with setting up and building execution environments.

15.1 Execution environment definition

A definition file is a `.yml` file that is required to build an image for an execution environment. An example of an execution environment definition schema is as follows:

```
---
version: 1

build_arg_defaults:
  EE_BASE_IMAGE: 'quay.io/ansible/ansible-runner:stable-2.10-devel'

ansible_config: 'ansible.cfg'

dependencies:
  galaxy: requirements.yml
  python: requirements.txt
  system: bindep.txt

additional_build_steps:
  prepend: |
    RUN whoami
    RUN cat /etc/os-release
  append:
    - RUN echo This is a post-install command!
    - RUN ls -la /etc
```

15.1.1 Build arguments and base image

Default values for build arguments can be specified in the definition file in the `default_build_args` section as a dictionary. This is an alternative to using the `--build-arg` CLI flag.

Build arguments used by `ansible-builder` are the following:

- `ANSIBLE_GALAXY_CLI_COLLECTION_OPTS` allows the user to pass the `-pre` flag to enable the installation of pre-releases collections.
- `EE_BASE_IMAGE` specifies the parent image for the execution environment.
- `EE_BUILDER_IMAGE` specifies the image used for compiling type tasks.

Values given inside of `default_build_args` will be hard-coded into the Containerfile, so they will persist if `podman build` is called manually.

If the same variable is specified in the CLI `--build-arg` flag, the CLI value will take higher precedence.

15.1.2 Ansible config file path

When using an `ansible.cfg` file to pass a token and other settings for a private account to an Automation Hub server, listing the config file path here (as a string) will enable it to be included as a build argument in the initial phase of the build.

15.1.3 Ansible Galaxy dependencies

The `galaxy` entry points to a valid requirements file for the `ansible-galaxy collection install -r ...` command.

The entry `requirements.yml` may be a relative path from the directory of the execution environment definition's folder, or an absolute path.

15.1.4 Python dependencies

The python entry points to a valid requirements file for the `pip install -r ...` command.

The entry `requirements.txt` may be a relative path from the directory of the execution environment definition's folder, or an absolute path.

15.1.5 System-level dependencies

The `system` entry points to a `bindep` requirements file. This will be processed by `bindep` and then passed to `dnf`, other platforms are not yet supported. For more information about bindep, refer to the OpenDev documentation.

15.1.6 Additional custom build steps

Additional commands may be specified in the `additional_build_steps` section, either for before the main build steps (`prepend`) or after (`append`). The syntax needs to be one of the following:

- a multi-line string (example shown in the `prepend` section above)
- a dictionary (as shown via `append`)

15.2 ansible-builder build options

The following options can be used with the `ansible-builder build` command:

Flag	Syntax	Description
`--tag`	`$ ansible-builder build --tag=my-custom-ee`	To customize the tagged name applied to the built image.
`--file`	`$ ansible-builder build --file=my-ee.yml`	To use a definition file named something other than `execution-environment.yml`.
`--context`	`$ ansible-builder build --context=/path/to/dir`	To specify a location other than the default directory named `context` created in the current working directory.
`--build-arg`	`$ ansible-builder build --build-arg FOO=bar`	To use Podman or Docker build-time variables, specify them the same way you would with `podman build` or `docker build`. By default, the Containerfile or Dockerfile contains a build argument `EE_BASE_IMAGE`, that lets you rebuild without modifying files.
`--build-arg`	`$ ansible-builder build --build-arg EE_BASE_IMAGE=registry.example.com/another-ee`	To use a custom base image, replaces previously discontinued `--base-image` option.
`--container-runtime`	`$ ansible-builder build --container-runtime=docker`	To use Docker to build images instead of the Podman default.
`--verbosity`	`$ ansible-builder build --verbosity 2`	To customize the level of verbosity.

15.2.1 Examples

The example in `test/data/pytz` requires the `awx.awx` collection in the execution environment definition. The lookup plugin `awx.awx.tower_schedule_rrule` requires the PyPI `pytz` and another library to work. If `test/data/pytz/execution-environment.yml` file is provided to the `ansible-builder build` command, then it will install the collection inside the image, read the `requirements.txt` file inside of the collection, and then install `pytz` into the image.

The image produced can be used inside of an `ansible-runner` project by placing these variables inside the `env/settings` file, inside of the private data directory.

```
---
container_image: image-name
process_isolation_executable: podman # or docker
process_isolation: true
```

The `awx.awx` collection is a subset of content included in the default AWX execution environment. More details can be found in the awx-ee repository.

15.3 Collection-level metadata

Collections inside of the `galaxy` entry of an execution environment will contribute their Python and system requirements to the image.

Requirements from a collection can be recognized in these ways:

- A file `meta/execution-environment.yml` references the Python and/or bindep requirements files
- A file named `requirements.txt` is in the root level of the collection
- A file named `bindep.txt` is in the root level of the collection

If any of these files are in the `build_ignore` of the collection, it will not work correctly.

Collection maintainers can verify that ansible-builder recognizes the requirements they expect by using the `introspect` command, for example:

```
ansible-builder introspect --sanitize ~/.ansible/collections/
```

15.3.1 Python Dependencies

Python requirements files are combined into a single file using the `requirements-parser` library in order to support complex syntax like references to other files.

Entries from separate collections that give the same package name will be combined into the same entry, with the constraints combined.

There are several package names which are specifically *ignored* by ansible-builder, meaning that if a collection lists these, they will not be included in the combined file. These include test packages and packages that provide Ansible itself. The full list can be found in `EXCLUDE_REQUIREMENTS` in the `ansible_builder.requirements` module.

15.3.2 System-level Dependencies

The `bindep` format provides a way of specifying cross-platform requirements. A minimum expectation is that collections specify necessary requirements for `[platform:rpm]`.

Entries from multiple collections will be combined into a single file. Only requirements with no profiles (runtime requirements) will be installed to the image. Entries from multiple collections which are outright duplicates of each other may be consolidated in the combined file.

CHAPTER

SIXTEEN

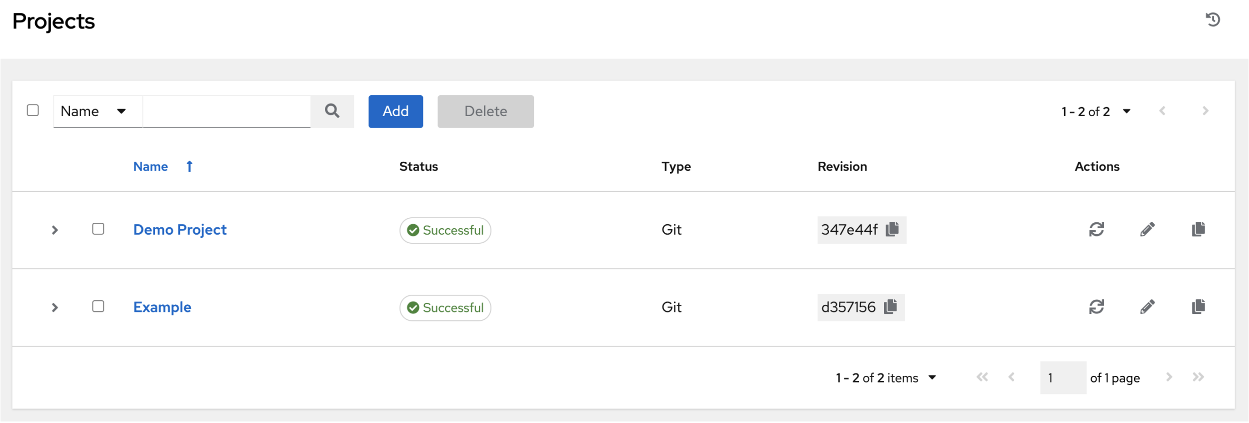
PROJECTS

A *Project* is a logical collection of Ansible playbooks.

You can manage playbooks and playbook directories by either placing them manually under the Project Base Path on your server, or by placing your playbooks into a source code management (SCM) system supported by automation controller, including Git, Subversion, and Red Hat Insights. To create a Red Hat Insights project, refer to *Setting up Insights Remediations*.

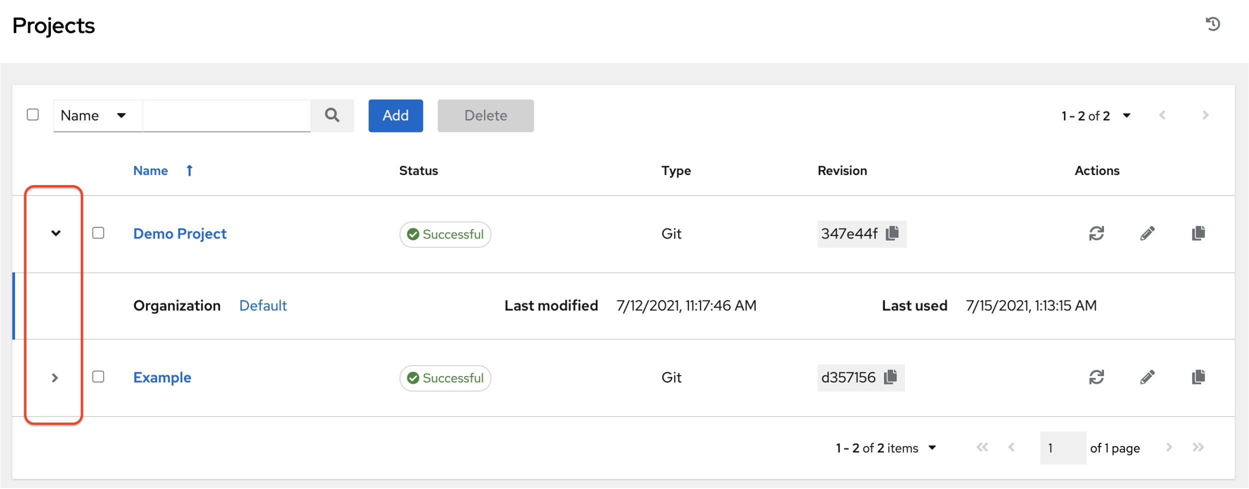
Note: By default, the Project Base Path is `/var/lib/awx/projects`, but this may have been modified by the administrator. It is configured in `/etc/tower/conf.d/custom.py`. Use caution when editing this file, as incorrect settings can disable your installation.

The Projects page displays the list of the projects that are currently available. The default view is collapsed (**Compact**) with project name and its status, but you can use the arrow next to each entry to expand for more information.



Projects

		Name ↑	Status	Type	Revision	Actions
›	□	Demo Project	✓ Successful	Git	347e44f	
›	□	Example	✓ Successful	Git	d357156	



Projects

☐ Name ▾ 🔍 Add Delete 1 - 2 of 2 ▾

		Name ↑	Status	Type	Revision	Actions
⌄	☐	Demo Project	✓ Successful	Git	347e44f	⟳ ✎ ⧉
		Organization Default	Last modified 7/12/2021, 11:17:46 AM		Last used 7/15/2021, 1:13:15 AM	
›	☐	Example	✓ Successful	Git	d357156	⟳ ✎ ⧉

1 - 2 of 2 items ▾ 1 of 1 page

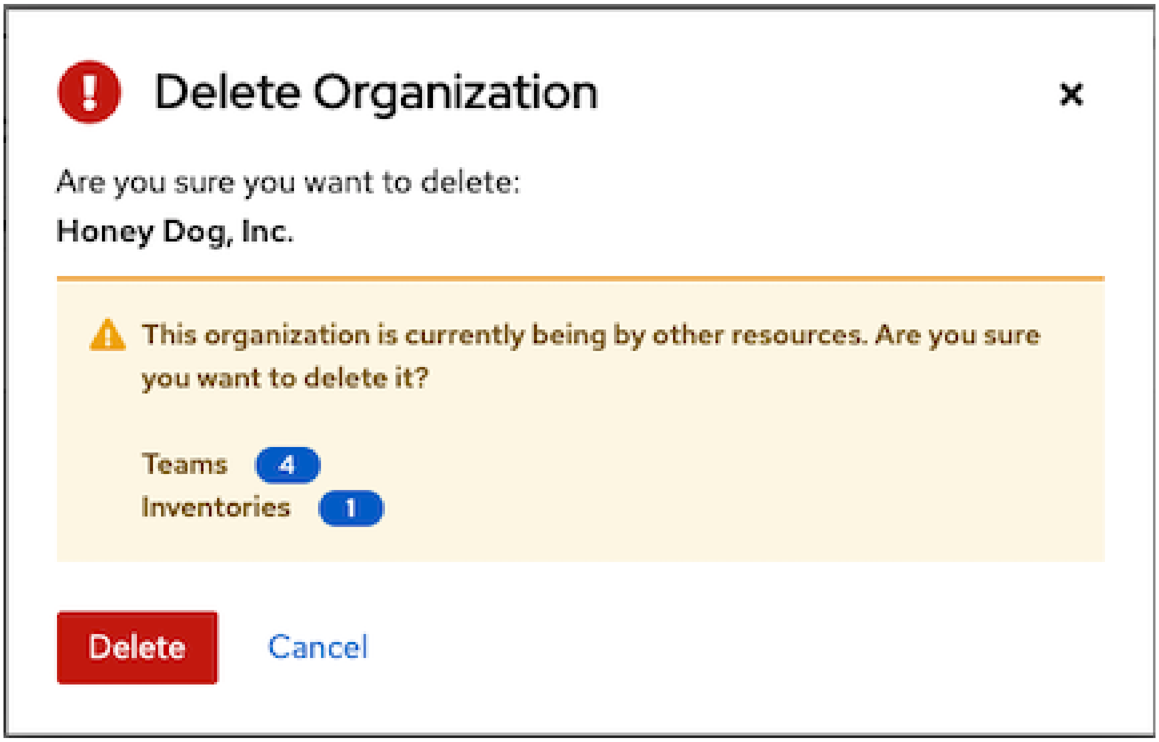
For each project listed, you can get the latest SCM revision (⟳), edit the project (✎), or copy the project attributes (⧉), using the respective icons next to each project. Projects are allowed to be updated while a related job is running. In cases where you have a big project (around 10 GB), disk space on `/tmp` may be an issue.

Status indicates the state of the project and may be one of the following (note that you can also filter your view by specific status types):

- **Pending** - The source control update has been created, but not queued or started yet. Any job (not just source control updates) will stay in pending until it's actually ready to be run by the system. Reasons for it not being ready because it has dependencies that are currently running so it has to wait until they are done, or there is not enough capacity to run in the locations it is configured to.
- **Waiting** - The source control update is in the queue waiting to be executed.
- **Running** - The source control update is currently in progress.
- **Successful** - The last source control update for this project succeeded.
- **Failed** - The last source control update for this project failed.
- **Error** - The last source control update job failed to run at all. (To be deprecated.)
- **Canceled** - The last source control update for the project was canceled.
- **Never updated** - The project is configured for source control, but has never been updated.
- **OK** - The project is not configured for source control, and is correctly in place. (To be deprecated.)
- **Missing** - Projects are absent from the project base path of `/var/lib/awx/projects` (applicable for manual or source control managed projects).

Note: Projects of credential type Manual cannot update or schedule source control-based actions without being reconfigured as an SCM type credential.

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:



Delete Organization

Are you sure you want to delete:

Honey Dog, Inc.

This organization is currently being by other resources. Are you sure you want to delete it?

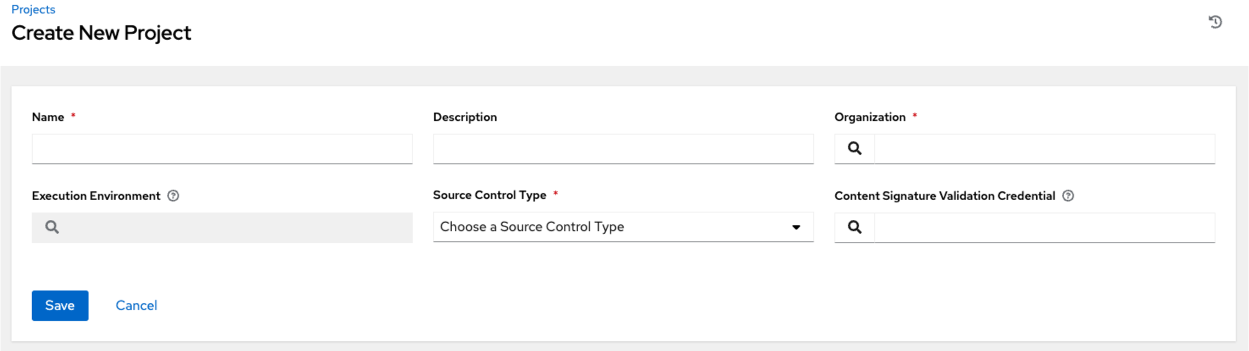
Teams 4

Inventories 1

16.1 Add a new project

To create a new project:

1. Click the **Add** button, which launches the **Create Project** window.



Create New Project

Name *

Description

Organization *

Execution Environment

Source Control Type *

Choose a Source Control Type

Content Signature Validation Credential

2. Enter the appropriate details into the following required fields:

- **Name**
- **Description** (optional)
- **Organization** - A project must have at least one organization. Pick one organization now to create the project, and then after the project is created you can add additional organizations.
- **Execution Environment** (optional) - Enter the name of the execution environment or search from a list of existing ones to run this project. See Upgrading to Execution Environments in the *Ansible Automation Platform Upgrade and Migration Guide* for more information.

- **Source Control Type** - Select from the drop-down menu list an SCM type associated with this project. The options in the subsequent section become available depend on the type you choose. Refer to *Manage playbooks manually* or *Manage playbooks using source control* in the subsequent sections for more detail.
- **Content Signature Validation Credential** - Use this optional field to enable content verification. Specify the GPG key to use for validating content signature during project sync. If the content has been tampered with, the job will not run. See *Project Signing and Verification* for more detail.

3. Click **Save** when done.

16.1.1 Manage playbooks manually

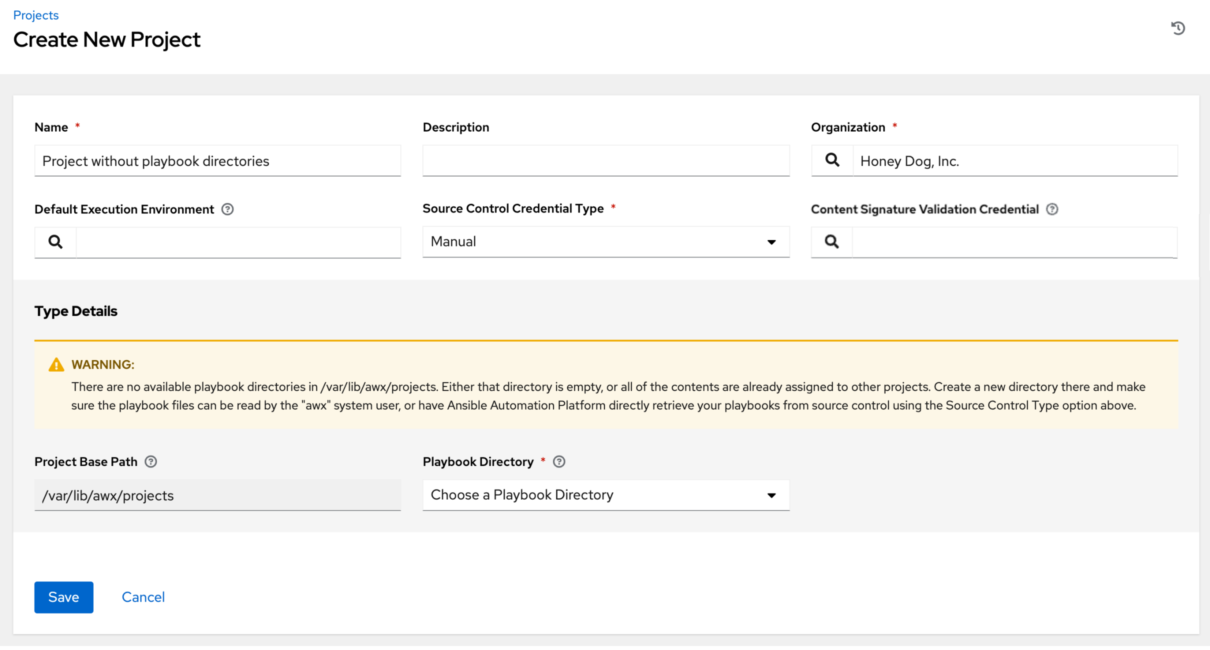
- Create one or more directories to store playbooks under the Project Base Path (for example, /var/lib/awx/projects/).
- Create or copy playbook files into the playbook directory.
- Ensure that the playbook directory and files are owned by the same UNIX user and group that the automation controller service runs as.
- Ensure that the permissions are appropriate for the playbook directories and files.

If adding a manual project, each project path inside of the project root folder can only be assigned to one project. If you receive the following message, ensure that you have not already assigned the project path to an existing project:

```
All of the project paths have been assigned to existing projects, or
there are no directories found in the base path.  You will need to add
a project path before creating a new project.
```

If you have trouble adding a project path, check the permissions and SELinux context settings for the project directory and files.

Warning: If you have not added any Ansible playbook directories to the base project path, you will receive the following message:



Projects

Create New Project

Name *: Project without playbook directories

Description:

Organization *: Honey Dog, Inc.

Default Execution Environment:

Source Control Credential Type *: Manual

Content Signature Validation Credential:

Type Details

WARNING:
There are no available playbook directories in /var/lib/awx/projects. Either that directory is empty, or all of the contents are already assigned to other projects. Create a new directory there and make sure the playbook files can be read by the "awx" system user, or have Ansible Automation Platform directly retrieve your playbooks from source control using the Source Control Type option above.

Project Base Path: /var/lib/awx/projects

Playbook Directory *: Choose a Playbook Directory

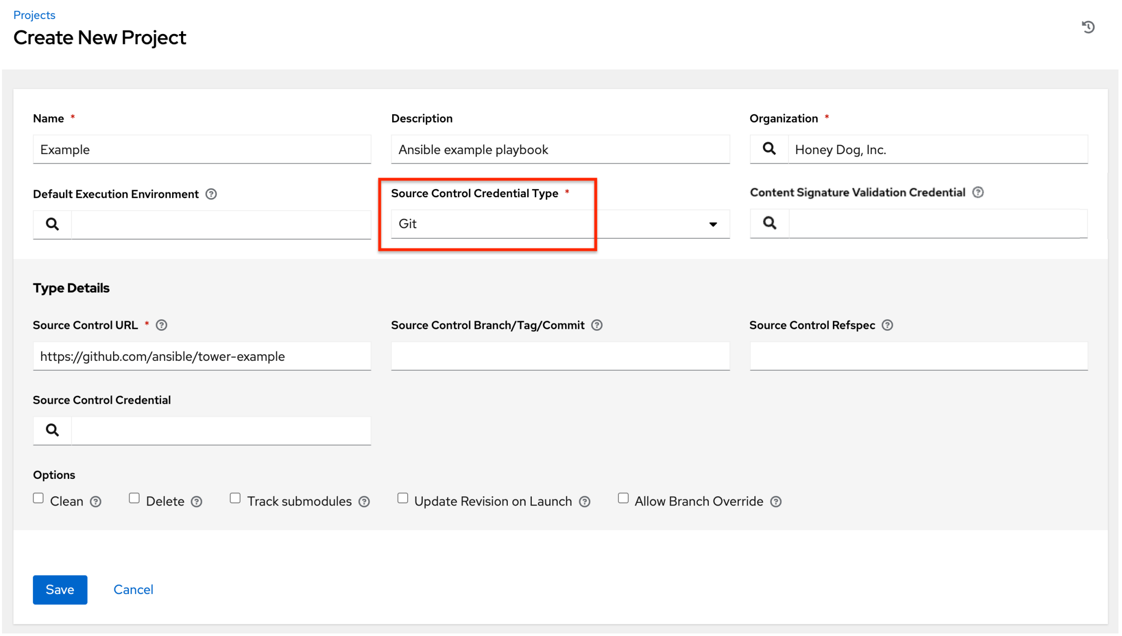
Correct this issue by creating the appropriate playbook directories and checking out playbooks from your SCM or otherwise copying playbooks into the appropriate playbook directories.

16.1.2 Manage playbooks using source control

SCM Types - Git and Subversion

To configure playbooks to use source control, in the Project **Details** tab:

1. Select the appropriate option (Git or Subversion) from the **SCM Type** drop-down menu list.



Projects

Create New Project

Name * : Example

Description: Ansible example playbook

Organization * : Honey Dog, Inc.

Default Execution Environment:

Source Control Credential Type * : Git

Content Signature Validation Credential:

Type Details

Source Control URL * : https://github.com/ansible/tower-example

Source Control Branch/Tag/Commit:

Source Control Refspec:

Source Control Credential:

Options

☐ Clean ☐ Delete ☐ Track submodules ☐ Update Revision on Launch ☐ Allow Branch Override

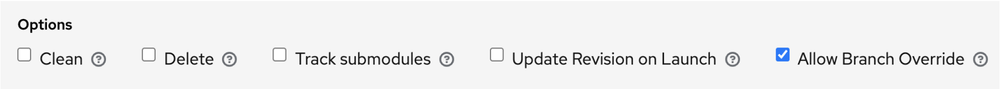
2. Enter the appropriate details into the following fields:

   - **SCM URL** - See an example in the tooltip ?.
   - **SCM Branch/Tag/Commit** - Optionally enter the SCM branch, tags, commit hashes, arbitrary refs, or revision number (if applicable) from the source control (Git or Subversion) to checkout. Some commit hashes and refs may not be available unless you also provide a custom refspec in the next field. If left blank, the default is HEAD which is the last checked out Branch/Tag/Commit for this project.
   - **SCM Refspec** - This field is an option specific to git source control and only advanced users familiar and comfortable with git should specify which references to download from the remote repository. For more detail, see *job branch overriding*.

   - **Source Control Credential** - If authentication is required, select the appropriate source control credential

3. In the **SCM Update Options**, optionally select the launch behavior, if applicable.

   - **Clean** - Removes any local modifications prior to performing an update.
   - **Delete** - Deletes the local repository in its entirety prior to performing an update. Depending on the size of the repository this may significantly increase the amount of time required to complete an update.
   - **Track submodules** - Tracks the latest commit. See more details in the tooltip ?.
   - **Update Revision on Launch** - Updates the revision of the project to the current revision in the remote source control, as well as cache the roles directory from *Galaxy* or *Collections*. Automation controller ensures that the local revision matches and that the roles and collections are up-to-date with the last update. Also, to avoid job overflows if jobs are spawned faster than the project can sync, selecting this allows you to configure a Cache Timeout to cache prior project syncs for a certain number of seconds.
   - **Allow Branch Override** - Allows a job template that uses this project to launch with a specified SCM branch or revision other than that of the project's. For more detail, see *job branch overriding*.



Options

☐ Clean ? ☐ Delete ? ☐ Track submodules ? ☐ Update Revision on Launch ? ☑ Allow Branch Override ?

4. Click **Save** to save your project.

Tip: Using a GitHub link offers an easy way to use a playbook. To help get you started, use the `helloworld.yml` file available at: https://github.com/ansible/tower-example.git

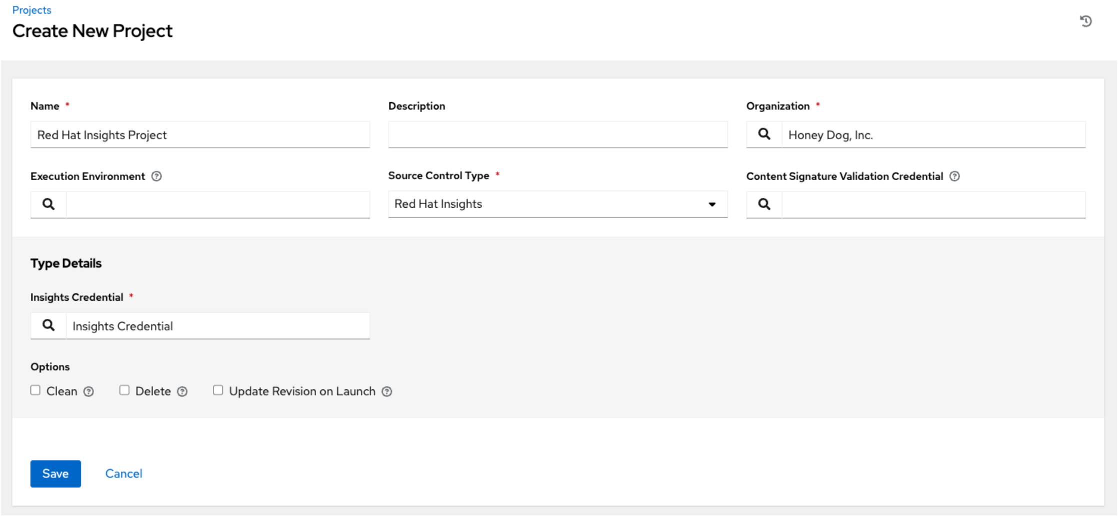
This link offers a very similar playbook to the one created manually in the instructions found in the Automation Controller Quick Setup Guide. Using it will not alter or harm your system in anyway.

SCM Type - Red Hat Insights

To configure playbooks to use Red Hat Insights, in the Project **Details** tab:

1. Select **Red Hat Insights** from the **SCM Type** drop-down menu list.
2. Red Hat Insights requires a credential for authentication. Select from the **Credential** field the appropriate credential for use with Insights.
3. In the **SCM Update Options**, optionally select the launch behavior, if applicable.

   - **Clean** - Removes any local modifications prior to performing an update.
   - **Delete** - Deletes the local repository in its entirety prior to performing an update. Depending on the size of the repository this may significantly increase the amount of time required to complete an update.
   - **Update Revision on Launch** - Updates the revision of the project to the current revision in the remote source control, as well as cache the roles directory from *Galaxy* or *Collections*. Automation controller ensures that the local revision matches and that the roles and collections are up-to-date with the last update. Also, to avoid job overflows if jobs are spawned faster than the project can sync, selecting this allows you to configure a Cache Timeout to cache prior project syncs for a certain number of seconds.



Projects

Create New Project

Name *	Description	Organization *
Red Hat Insights Project		Honey Dog, Inc.
Execution Environment	Source Control Type *	Content Signature Validation Credential
	Red Hat Insights	

Type Details

Insights Credential *

Insights Credential

Options

□ Clean □ Delete □ Update Revision on Launch

3. Click **Save** to save your project.

SCM Type - Remote Archive

Playbooks using a remote archive allow projects to be provided based on a build process that produces a versioned artifact, or release, containing all the requirements for that project in a single archive.

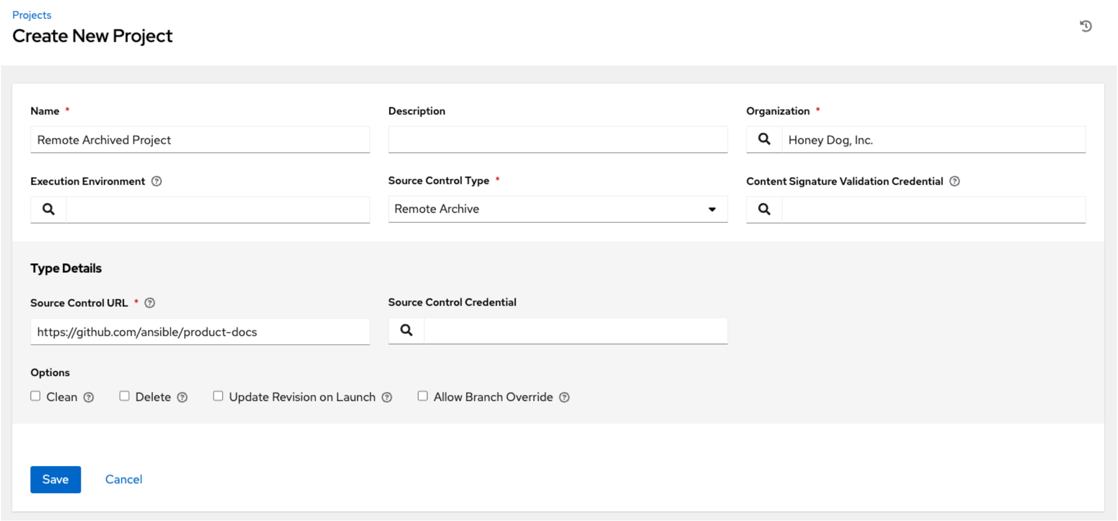
To configure playbooks to use a remote archive, in the Project **Details** tab:

1. Select **Remote Archive** from the **SCM Type** drop-down menu list.
2. Enter the appropriate details into the following fields:

- **SCM URL** - requires a URL to a remote archive, such as a *GitHub Release* or a build artifact stored in *Artifactory* and unpacks it into the project path for use
- **SCM Credential** - If authentication is required, select the appropriate SCM credential

3. In the **SCM Update Options**, optionally select the launch behavior, if applicable.

   - **Clean** - Removes any local modifications prior to performing an update.
   - **Delete** - Deletes the local repository in its entirety prior to performing an update. Depending on the size of the repository this may significantly increase the amount of time required to complete an update.
   - **Update Revision on Launch** - Not recommended, as this option updates the revision of the project to the current revision in the remote source control, as well as cache the roles directory from *Galaxy* or *Collections*.
   - **Allow Branch Override** - Not recommended, as this option allows a job template that uses this project to launch with a specified SCM branch or revision other than that of the project's.



Projects

Create New Project

Name *	Description	Organization *
Remote Archived Project		Honey Dog, Inc.
Execution Environment	Source Control Type *	Content Signature Validation Credential
	Remote Archive	

Type Details

Source Control URL *	Source Control Credential
https://github.com/ansible/product-docs	

Options

☐ Clean ☐ Delete ☐ Update Revision on Launch ☐ Allow Branch Override

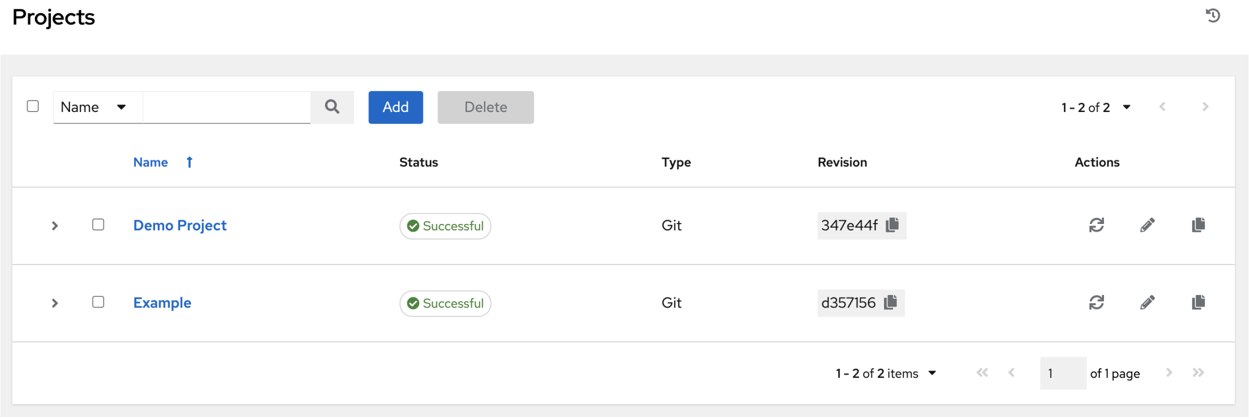
Note: Since this SCM type is intended to support the concept of unchanging artifacts, it is advisable to disable Galaxy integration (for roles, at minimum).

4. Click **Save** to save your project.

16.2 Updating projects from source control

1. Update an existing SCM-based project by selecting the project and clicking the ⟳ button.

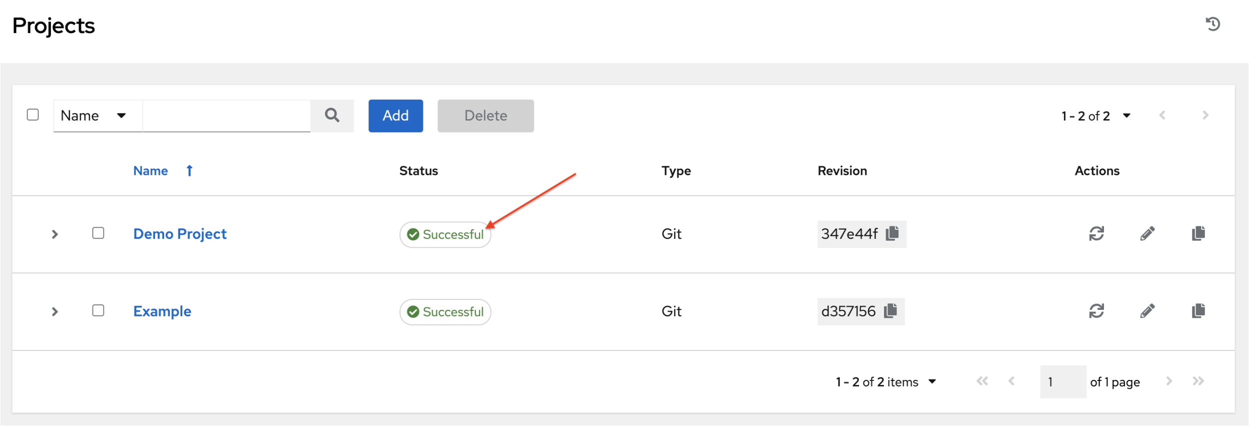
   Note: Please note that immediately after adding a project setup to use source control, a "Sync" starts that fetches the project details from the configured source control.



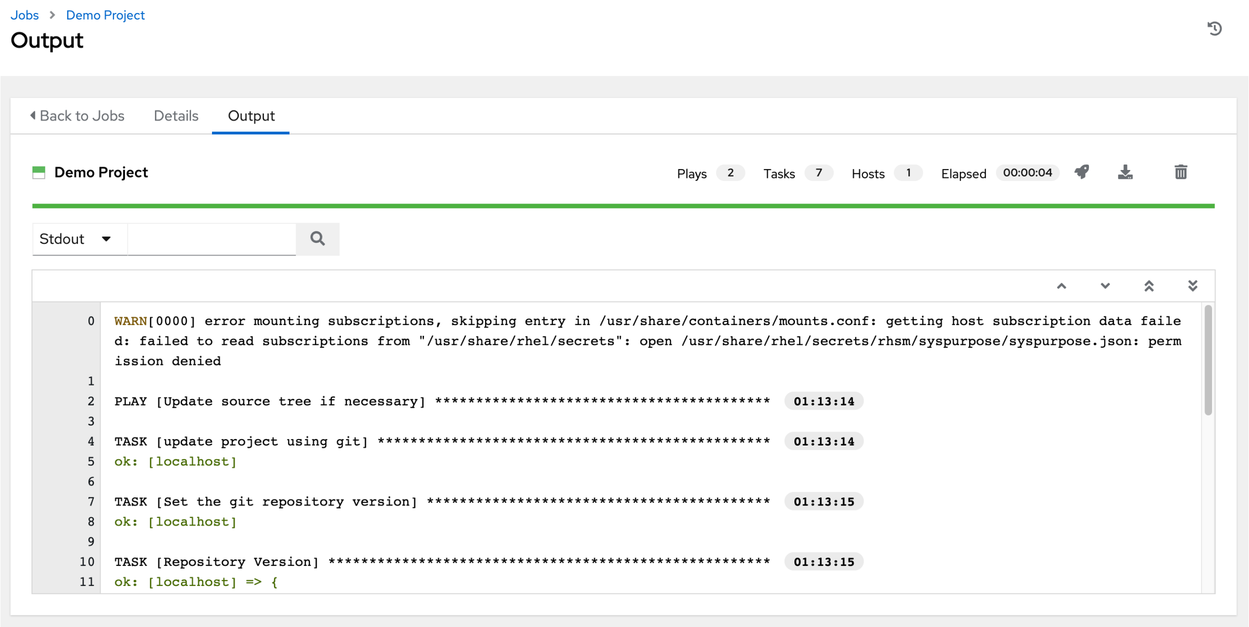
Projects

Name	Status	Type	Revision	Actions
Demo Project	Successful	Git	347e44f	
Example	Successful	Git	d357156	

2. Click on project's status under the **Status** column to get further details about the update process.



		Name ↑	Status	Type	Revision	Actions
›	□	Demo Project	✓ Successful	Git	347e44f	
›	□	Example	✓ Successful	Git	d357156	

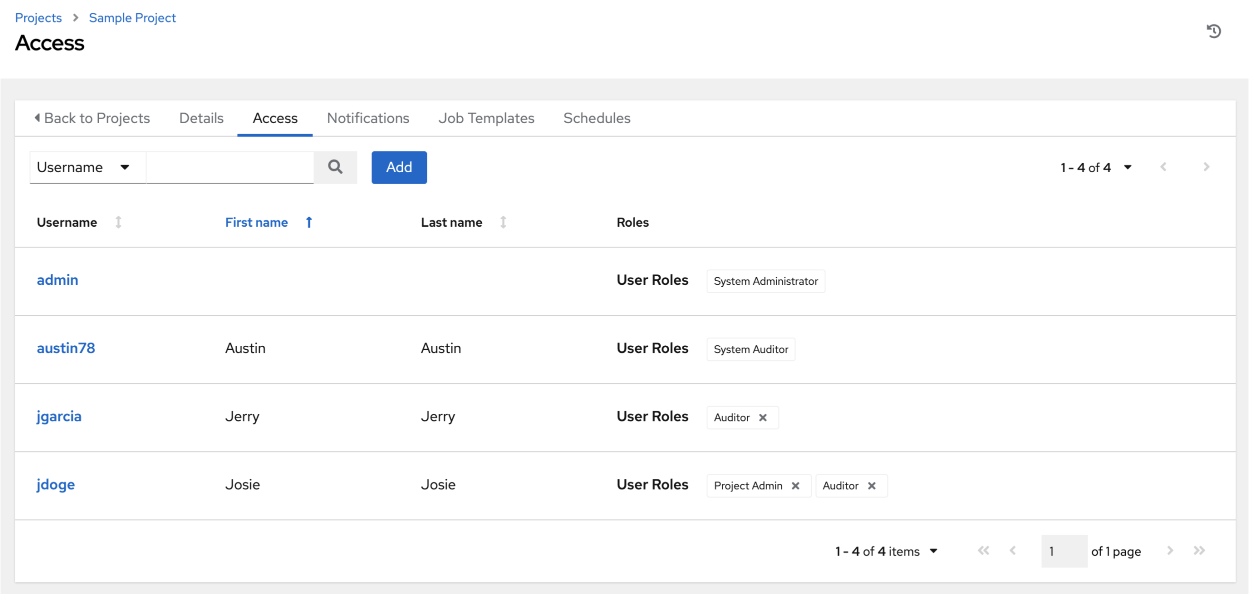


```
WARN[0000] error mounting subscriptions, skipping entry in /usr/share/containers/mounts.conf: getting host subscription data faile
d: failed to read subscriptions from "/usr/share/rhel/secrets": open /usr/share/rhel/secrets/rhsm/syspurpose/syspurpose.json: perm
ission denied

PLAY [Update source tree if necessary] ****************************************  01:13:14

TASK [update project using git] ***********************************************  01:13:14
ok: [localhost]

TASK [Set the git repository version] *****************************************  01:13:15
ok: [localhost]

TASK [Repository Version] *****************************************************  01:13:15
ok: [localhost] => {
```

16.3 Work with Permissions

The set of permissions assigned to this project (role-based access controls) that provide the ability to read, modify, and administer projects, inventories, job templates, and other automation controller elements are Privileges.

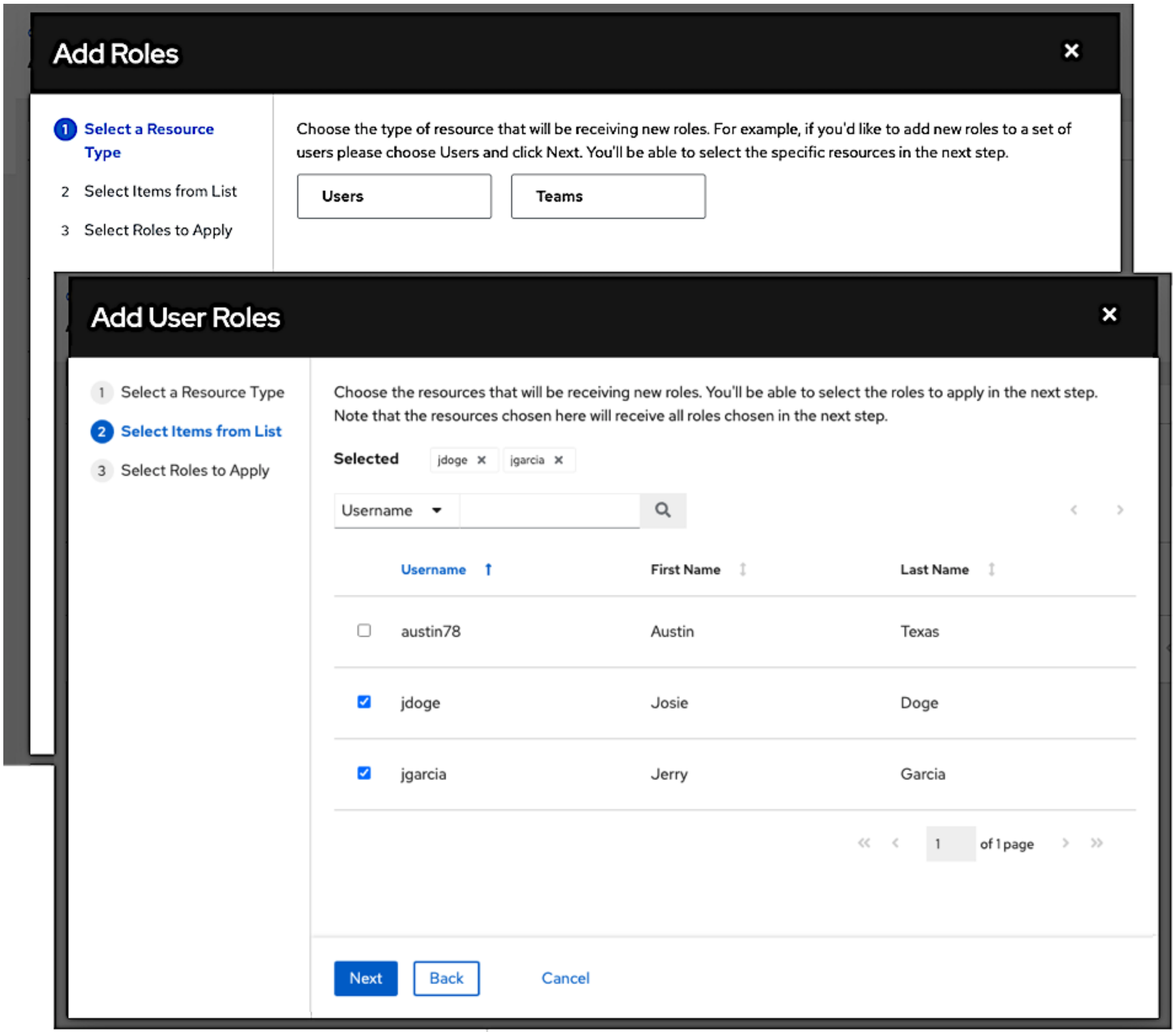
You can access the project permissions via the **Access** tab next to the **Details** tab. This screen displays a list of users that currently have permissions to this project. The list may be sorted and searched by **Username**, **First Name**, or **Last Name**.



Username	First name	Last name	Roles
admin			User Roles System Administrator
austin78	Austin	Austin	User Roles System Auditor
jgarcia	Jerry	Jerry	User Roles Auditor ✕
jdoge	Josie	Josie	User Roles Project Admin ✕ Auditor ✕

16.3.1 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**.



Choose the type of resource that will be receiving new roles. For example, if you'd like to add new roles to a set of users please choose Users and click Next. You'll be able to select the specific resources in the next step.

Choose the resources that will be receiving new roles. You'll be able to select the roles to apply in the next step. Note that the resources chosen here will receive all roles chosen in the next step.

Selected: jdoge × jgarcia ×

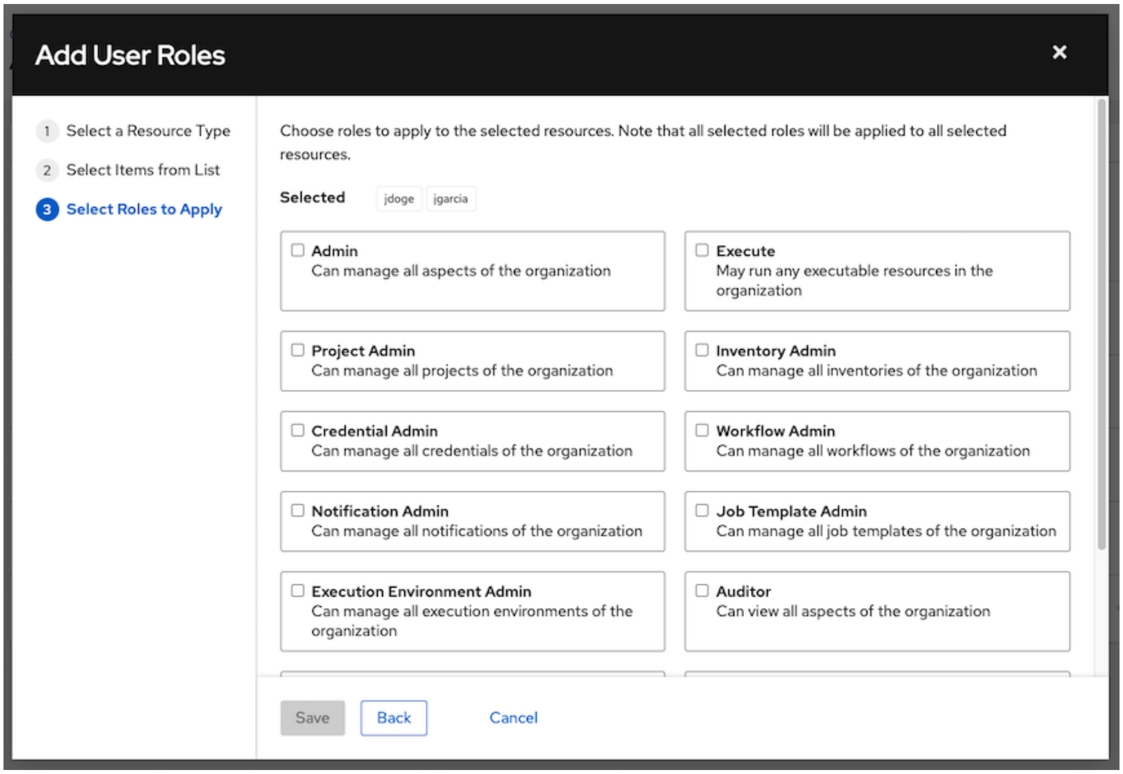
Username ▾

	Username	First Name	Last Name
☐	austin78	Austin	Texas
☑	jdoge	Josie	Doge
☑	jgarcia	Jerry	Garcia

1 of 1 page

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

4. 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.



Add User Roles ×

1 Select a Resource Type
2 Select Items from List
3 Select Roles to Apply

Choose roles to apply to the selected resources. Note that all selected roles will be applied to all selected resources.

Selected jdoge jgarcia

□ **Admin**
Can manage all aspects of the organization

□ **Execute**
May run any executable resources in the organization

□ **Project Admin**
Can manage all projects of the organization

□ **Inventory Admin**
Can manage all inventories of the organization

□ **Credential Admin**
Can manage all credentials of the organization

□ **Workflow Admin**
Can manage all workflows of the organization

□ **Notification Admin**
Can manage all notifications of the organization

□ **Job Template Admin**
Can manage all job templates of the organization

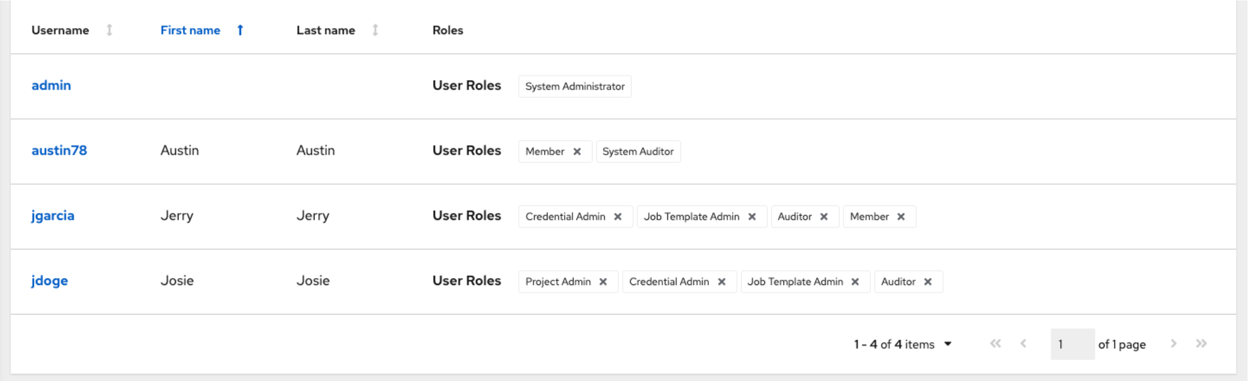
□ **Execution Environment Admin**
Can manage all execution environments of the organization

□ **Auditor**
Can view all aspects of the organization

Save | Back | Cancel

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

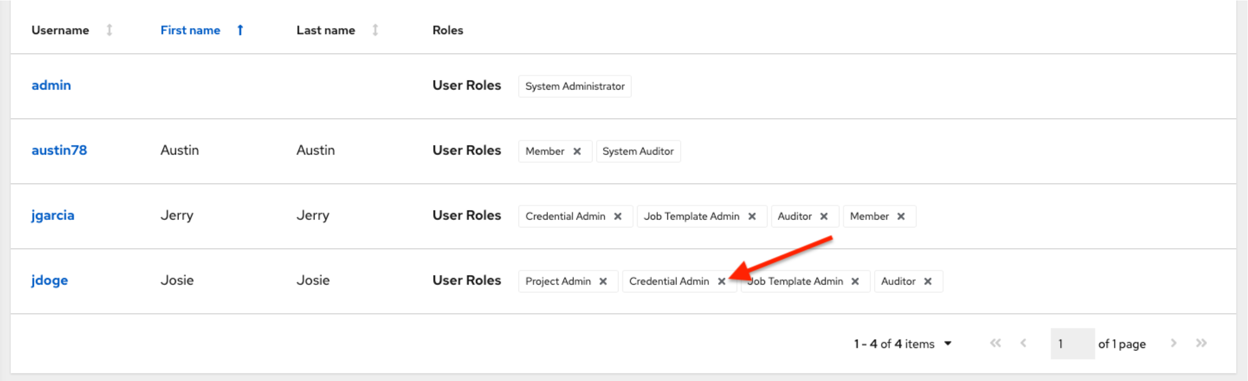
The Add Users/Teams window closes to display the updated roles assigned for each user and team.



Username	First name	Last name	Roles	
admin			User Roles	System Administrator
austin78	Austin	Austin	User Roles	Member × System Auditor
jgarcia	Jerry	Jerry	User Roles	Credential Admin × Job Template Admin × Auditor × Member ×
jdoge	Josie	Josie	User Roles	Project Admin × Credential Admin × Job Template Admin × Auditor ×

1 - 4 of 4 items ▾ « ‹ 1 of 1 page › »

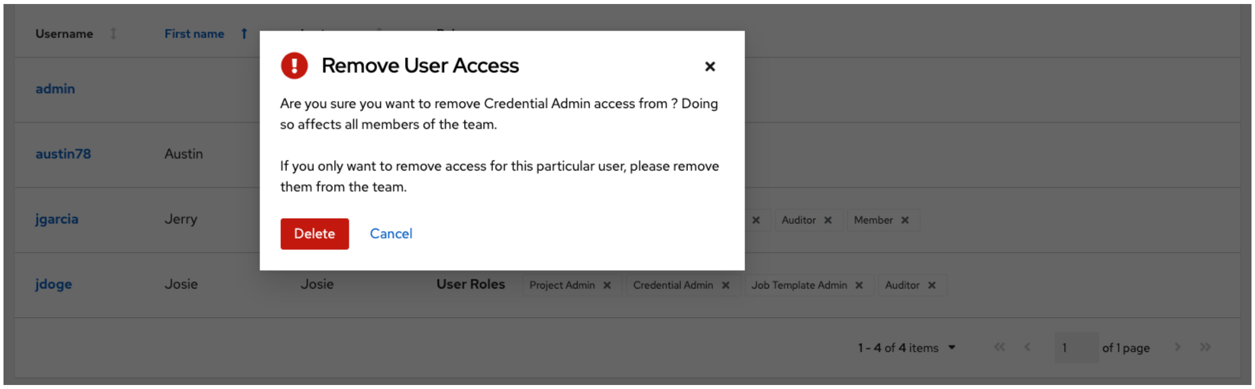
To remove roles for a particular user, click the disassociate (x) button next to its resource.



Username	First name	Last name	Roles	
admin			User Roles	System Administrator
austin78	Austin	Austin	User Roles	Member × System Auditor
jgarcia	Jerry	Jerry	User Roles	Credential Admin × Job Template Admin × Auditor × Member ×
jdoge	Josie	Josie	User Roles	Project Admin × Credential Admin × Job Template Admin × Auditor ×

1 - 4 of 4 items ▾ « ‹ 1 of 1 page › »

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



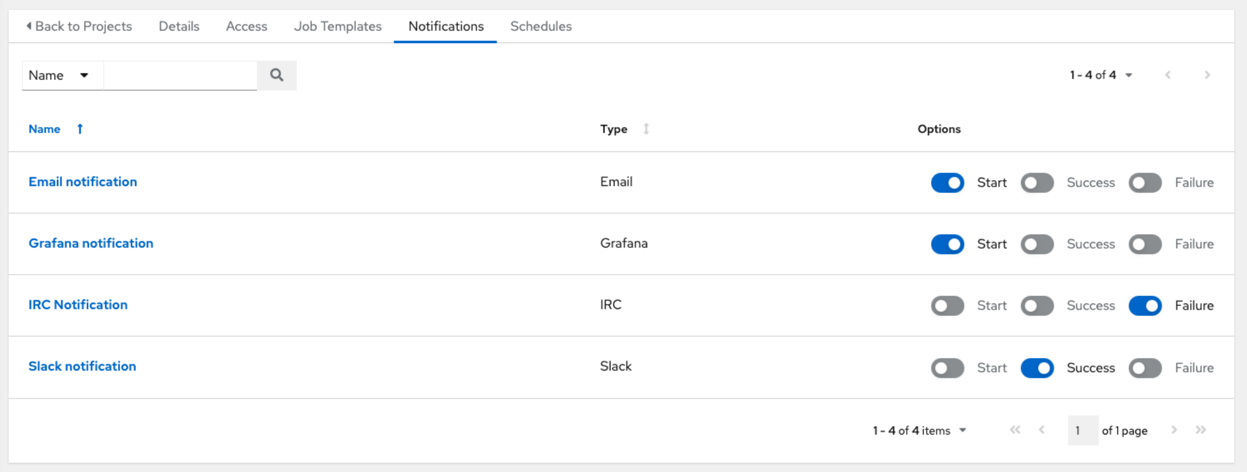
Remove User Access

Are you sure you want to remove Credential Admin access from ? Doing so affects all members of the team.

If you only want to remove access for this particular user, please remove them from the team.

16.4 Work with Notifications

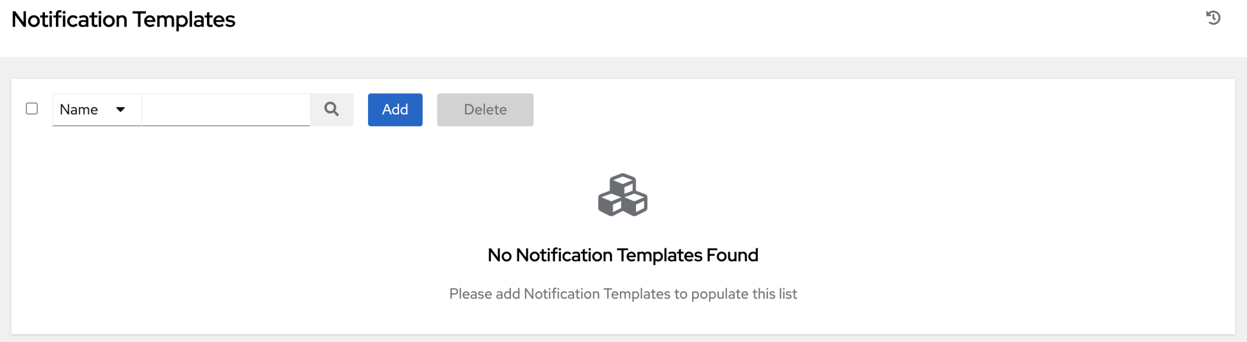
Clicking the **Notifications** tab allows you to review any notification integrations you have setup.



Name	Type	Options
Email notification	Email	● Start ○ Success ○ Failure
Grafana notification	Grafana	● Start ○ Success ○ Failure
IRC Notification	IRC	○ Start ○ Success ● Failure
Slack notification	Slack	○ Start ● Success ○ Failure

Use the toggles to enable or disable the notifications to use with your particular project. For more detail, see *Enable and Disable Notifications*.

If no notifications have been set up, you can configure them from the **Notifications** link from the left navigation bar to create a new notification.



Notification Templates

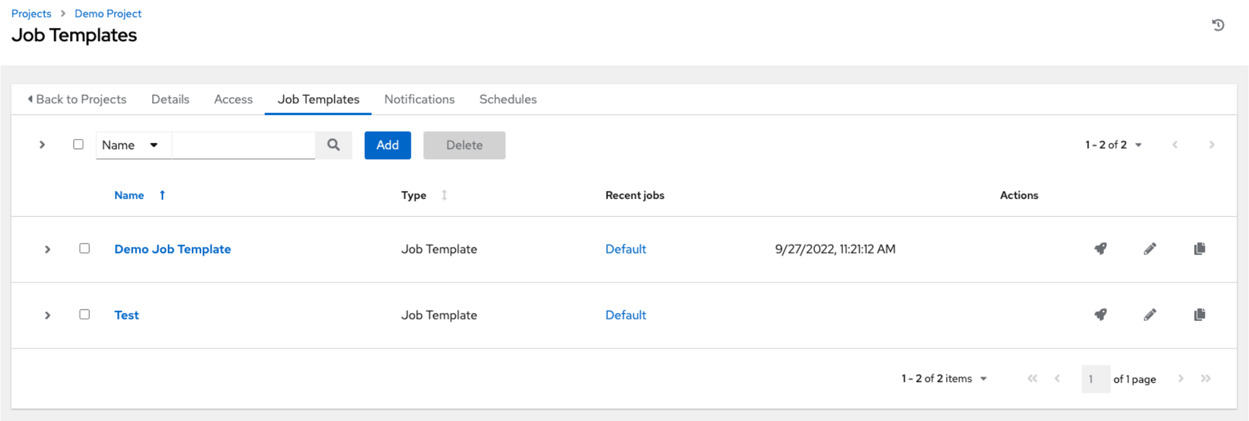
No Notification Templates Found

Please add Notification Templates to populate this list

Refer to *Notification Types* for additional details on configuring various notification types.

16.5 Work with Job Templates

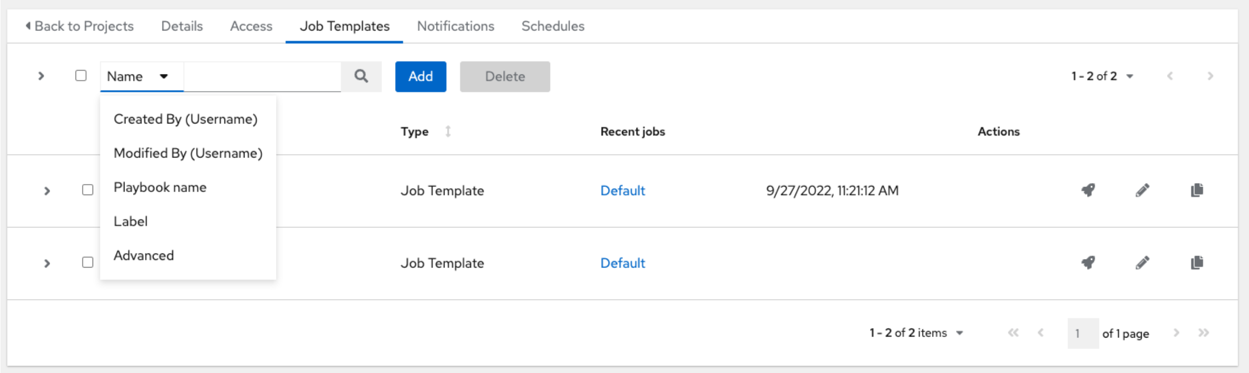
Clicking on **Job Templates** allows you to add and review any job templates or workflow templates associated with this project.



Job Templates

Name ↑	Type	Recent jobs		Actions
Demo Job Template	Job Template	Default	9/27/2022, 11:21:12 AM	🚀 ✏ ⧉
Test	Job Template	Default		🚀 ✏ ⧉

Click on the recent jobs that ran using that template to see its details and other useful information. You can sort this list by various criteria, and perform a search to filter the templates of interest.

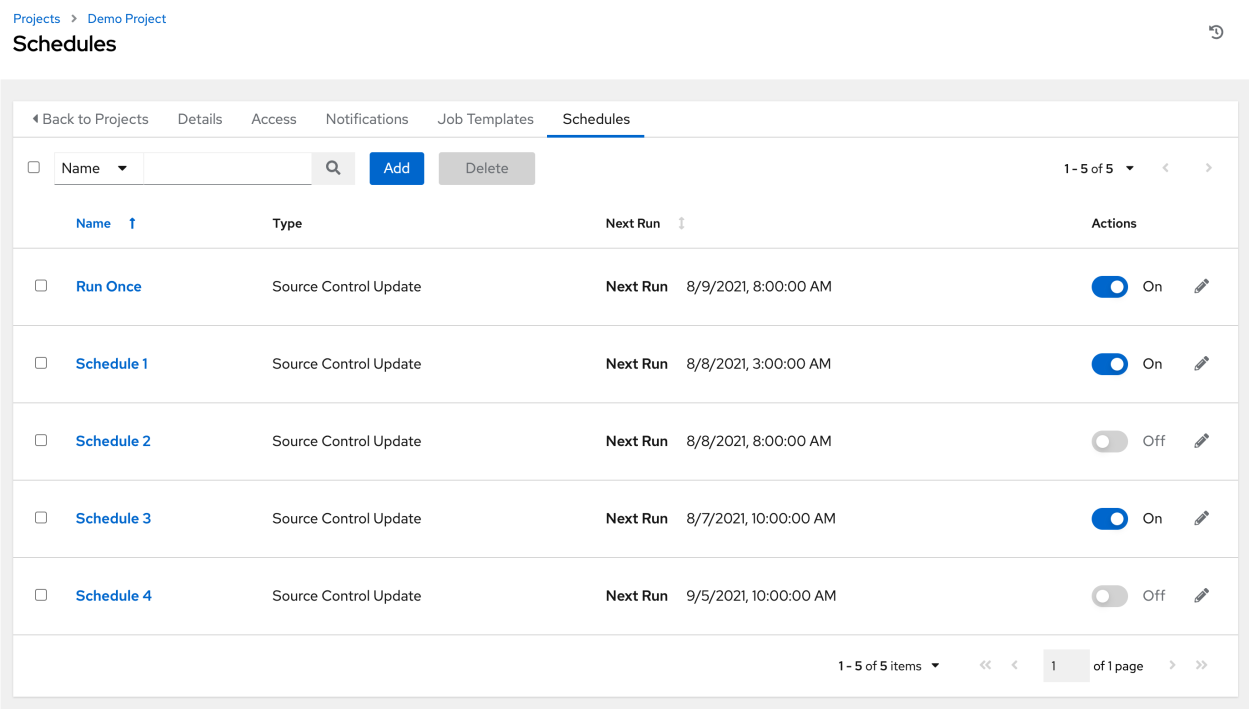


Type	Recent jobs		Actions
Job Template	Default	9/27/2022, 11:21:12 AM	🚀 ✏ ⧉
Job Template	Default		🚀 ✏ ⧉

From this view, you can also launch (🚀), edit (✏), or copy (⧉) the template configuration.

16.6 Work with Schedules

Clicking on **Schedules** allows you to review any schedules set up for this project.



Projects > Demo Project

Schedules

◂ Back to Projects | Details | Access | Notifications | Job Templates | Schedules

□ Name ▾ | Add | Delete | 1 - 5 of 5 ▾

□	Name ↑	Type	Next Run	Actions
□	Run Once	Source Control Update	Next Run 8/9/2021, 8:00:00 AM	On
□	Schedule 1	Source Control Update	Next Run 8/8/2021, 3:00:00 AM	On
□	Schedule 2	Source Control Update	Next Run 8/8/2021, 8:00:00 AM	Off
□	Schedule 3	Source Control Update	Next Run 8/7/2021, 10:00:00 AM	On
□	Schedule 4	Source Control Update	Next Run 9/5/2021, 10:00:00 AM	Off

1 - 5 of 5 items ▾ | 1 | of 1 page

16.6.1 Schedule a Project

To schedule a project run, click the **Schedules** tab.

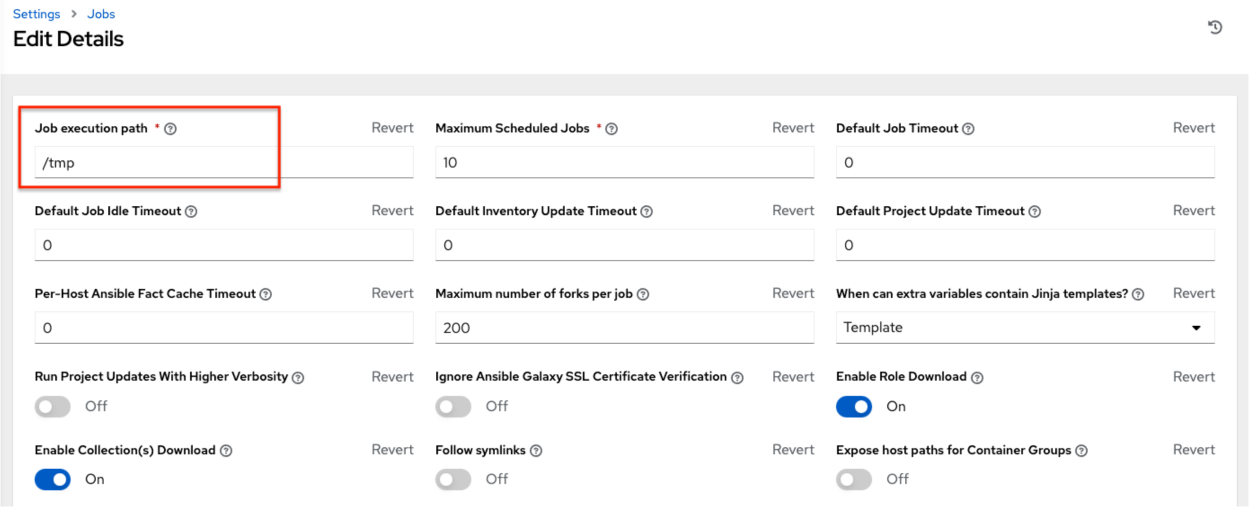
- If schedules are already set up; review, edit, or enable/disable your schedule preferences.
- If schedules have not been set up, refer to *Schedules* for more information.

16.7 Ansible Galaxy Support

At the end of a Project update, automation controller searches for a file called `requirements.yml` in the `roles` directory, located at `<project-top-level-directory>/roles/requirements.yml`. If this file is found, the following command automatically runs:

```
ansible-galaxy role install -r roles/requirements.yml -p <project-specific cache␣
↪location>/requirements_roles -vvv
```

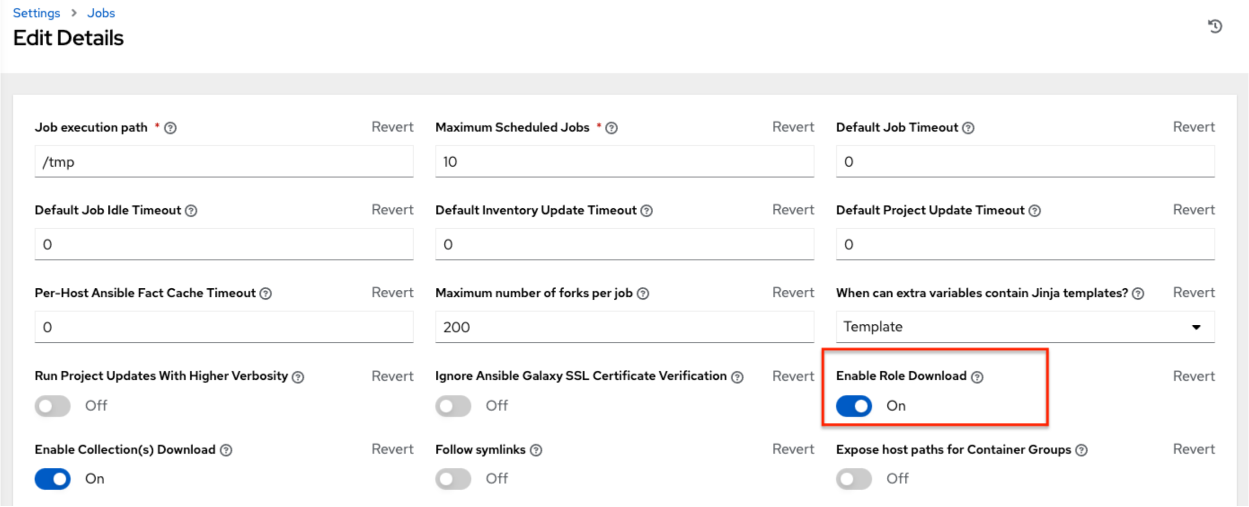
This file allows you to reference Galaxy roles or roles within other repositories which can be checked out in conjunction with your own project. The addition of this Ansible Galaxy support eliminates the need to create git submodules for achieving this result. Given that SCM projects (along with roles/collections) are pulled into and executed from a private job environment, a <private job directory> specific to the project within `/tmp` is created by default. However, you can specify another **Job Execution Path** based on your environment in the Jobs Settings tab of the Settings window:



Job execution path *: /tmp
Maximum Scheduled Jobs *: 10
Default Job Timeout: 0
Default Job Idle Timeout: 0
Default Inventory Update Timeout: 0
Default Project Update Timeout: 0
Per-Host Ansible Fact Cache Timeout: 0
Maximum number of forks per job: 200
When can extra variables contain Jinja templates?: Template
Run Project Updates With Higher Verbosity: Off
Ignore Ansible Galaxy SSL Certificate Verification: Off
Enable Role Download: On
Enable Collection(s) Download: On
Follow symlinks: Off
Expose host paths for Container Groups: Off

The cache directory is a subdirectory inside the global projects folder. The content may be copied from the cache location to `<job private directory>/requirements_roles` location.

By default, automation controller has a system-wide setting that allows roles to be dynamically downloaded from the `roles/requirements.yml` file for SCM projects. You may turn off this setting in the **Jobs settings** screen of the Settings menu by switching the **Enable Role Download** toggle button to **OFF**.

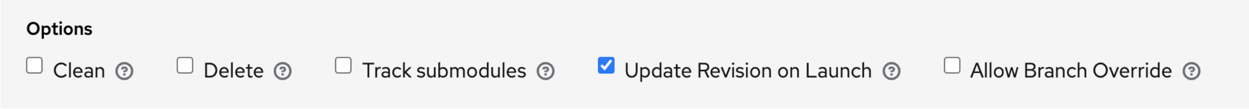


Job execution path *: /tmp
Maximum Scheduled Jobs *: 10
Default Job Timeout: 0
Default Job Idle Timeout: 0
Default Inventory Update Timeout: 0
Default Project Update Timeout: 0
Per-Host Ansible Fact Cache Timeout: 0
Maximum number of forks per job: 200
When can extra variables contain Jinja templates?: Template
Run Project Updates With Higher Verbosity: Off
Ignore Ansible Galaxy SSL Certificate Verification: Off
Enable Role Download: On
Enable Collection(s) Download: On
Follow symlinks: Off
Expose host paths for Container Groups: Off

Whenever a project sync runs, automation controller determines if the project source and any roles from Galaxy and/or Collections are out of date with the project. Project updates will download the roles inside the update.

If jobs need to pick up a change made to an upstream role, updating the project will ensure this happens. A change to the role means that a new commit was pushed to the *provision-role* source control. To make this change take effect in a job, you do not need to push a new commit to the *playbooks* repo, but you **do need** to update the project, which downloads roles to a local cache. For instance, say you have two git repositories in source control. The first one is *playbooks* and the project in automation controller points to this URL. The second one is *provision-role* and it is referenced by the `roles/requirements.yml` file inside of the *playbooks* git repo.

In short, jobs would download the most recent roles before every job run. Roles and collections are locally cached for performance reasons, and you will need to select **Update Revision on Launch** in the project SCM Update Options to ensure that the upstream role is re-downloaded before each job run:



Options

☐ Clean ☐ Delete ☐ Track submodules ☑ Update Revision on Launch ☐ Allow Branch Override

The update happens much earlier in the process than the sync, so this surfaces errors and details faster and in a more logic place.

For more information and examples on the syntax of the `requirements.yml` file, refer to the role requirements section in the Ansible documentation.

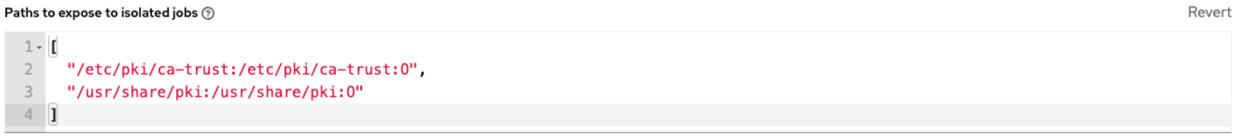
If there are any directories that should specifically be exposed, you can specify those in the Jobs section of the Settings screen in the **Paths to Expose to Isolated Jobs** or by updating the following entry in the settings file:

```
AWX_ISOLATION_SHOW_PATHS = ['/list/of/', '/paths']
```

> **Note:** The primary file you may want to add to `AWX_ISOLATION_SHOW_PATHS` is `/var/lib/awx/.ssh`, if your playbooks need to use keys or settings defined there.

If you made changes in the settings file, be sure to restart services with the `automation-controller-service restart` command after your changes have been saved.

In the User Interface, you can configure these settings in the Jobs settings window.



Paths to expose to isolated jobs

Revert

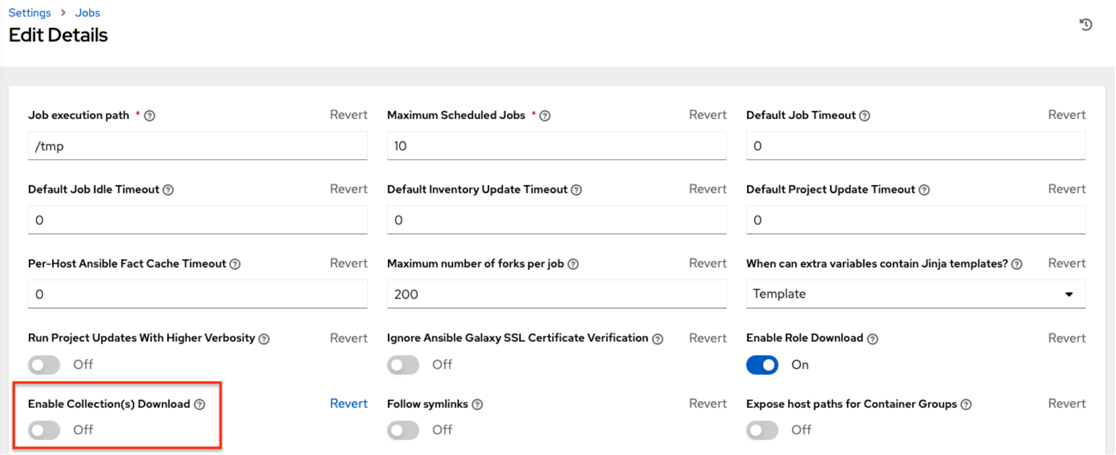
```
[
  "/etc/pki/ca-trust:/etc/pki/ca-trust:O",
  "/usr/share/pki:/usr/share/pki:O"
]
```

Note: The **Primary Galaxy Server Username** and **Primary Galaxy Server Password** fields are no longer configurable in automation controller 3.8. We recommend using tokens to access Galaxy or Automation Hub instead.

16.8 Collections Support

Automation controller supports project-specific Ansible collections in job runs. If you specify a collections requirements file in the SCM at `collections/requirements.yml`, automation controller will install collections in that file in the implicit project sync before a job run.

By default, automation controller has a system-wide setting that allows collections to be dynamically downloaded from the `collections/requirements.yml` file for SCM projects. You may turn off this setting in the **Jobs settings** tab of the Settings menu by switching the **Enable Collections Download** toggle button to **OFF**.



Edit Details

Job execution path * — Revert: /tmp

Maximum Scheduled Jobs * — Revert: 10

Default Job Timeout — Revert: 0

Default Job Idle Timeout — Revert: 0

Default Inventory Update Timeout — Revert: 0

Default Project Update Timeout — Revert: 0

Per-Host Ansible Fact Cache Timeout — Revert: 0

Maximum number of forks per job — Revert: 200

When can extra variables contain Jinja templates? — Revert: Template

Run Project Updates With Higher Verbosity — Revert: Off

Ignore Ansible Galaxy SSL Certificate Verification — Revert: Off

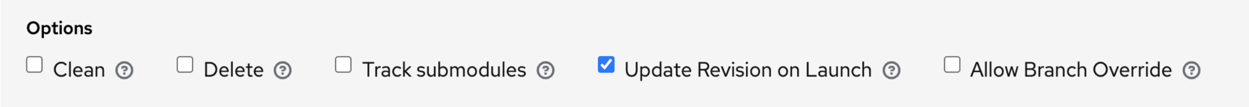
Enable Role Download — Revert: On

Enable Collection(s) Download — Revert: Off

Follow symlinks — Revert: Off

Expose host paths for Container Groups — Revert: Off

Roles and collections are locally cached for performance reasons, and you will need to select **Update Revision on Launch** in the project SCM Update Options to ensure this:



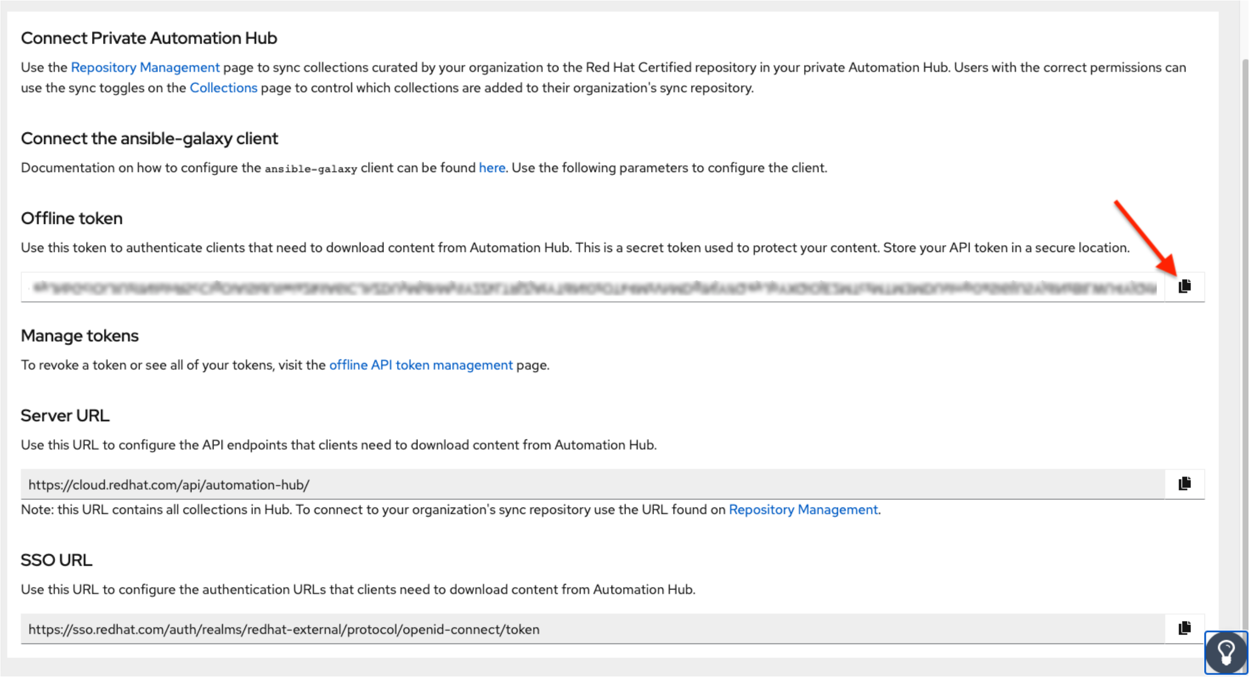
Options

□ Clean □ Delete □ Track submodules ☑ Update Revision on Launch □ Allow Branch Override

16.8.1 Using Collections via Hub

Before automation controller can use Automation Hub as the default source for collections content, you need to create an API token in the Automation Hub UI so that it could be specified in automation controller. You may connect to a private Automation Hub or a public Automation Hub collection, the only difference is which URL you specify.

1. Navigate to https://cloud.redhat.com/ansible/automation-hub/token and click **Load token**.
2. Click the copy icon to copy the API token to the clipboard.



Connect Private Automation Hub

Use the Repository Management page to sync collections curated by your organization to the Red Hat Certified repository in your private Automation Hub. Users with the correct permissions can use the sync toggles on the Collections page to control which collections are added to their organization's sync repository.

Connect the ansible-galaxy client

Documentation on how to configure the `ansible-galaxy` client can be found here. Use the following parameters to configure the client.

Offline token

Use this token to authenticate clients that need to download content from Automation Hub. This is a secret token used to protect your content. Store your API token in a secure location.

[redacted]

Manage tokens

To revoke a token or see all of your tokens, visit the offline API token management page.

Server URL

Use this URL to configure the API endpoints that clients need to download content from Automation Hub.

https://cloud.redhat.com/api/automation-hub/

Note: this URL contains all collections in Hub. To connect to your organization's sync repository use the URL found on Repository Management.

SSO URL

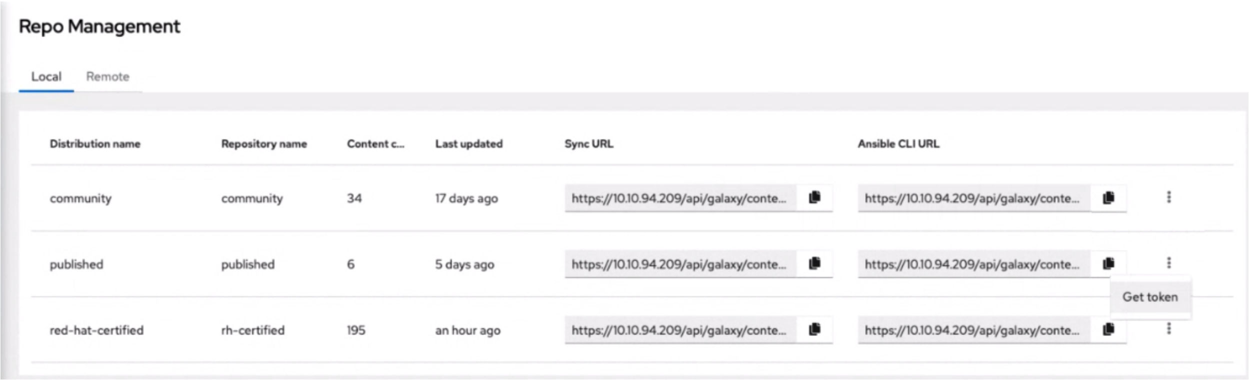
Use this URL to configure the authentication URLs that clients need to download content from Automation Hub.

https://sso.redhat.com/auth/realms/redhat-external/protocol/openid-connect/token

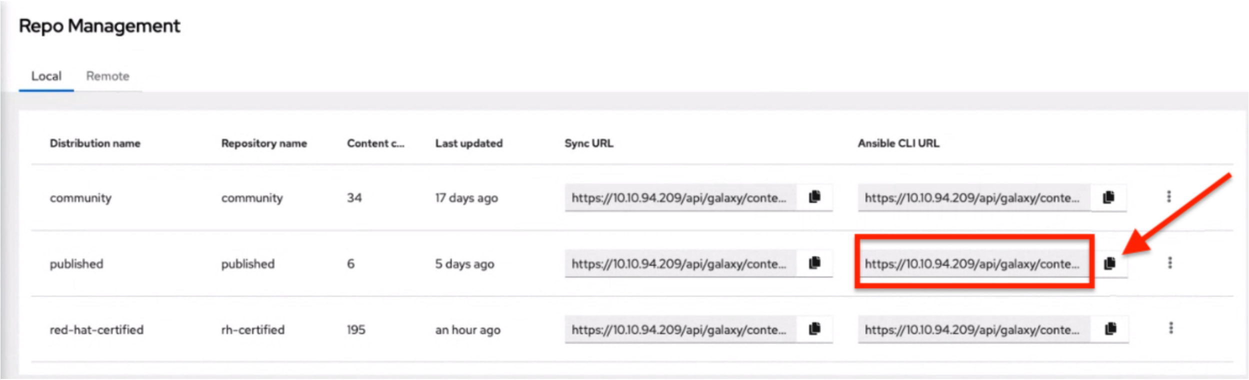
3. To use the public Automation Hub, create an Automation Hub credential using the copied token and pointing to the URLs shown in the **Server URL** and **SSO URL** fields of the token page:

- **Galaxy Server URL** = `https://cloud.redhat.com/api/automation-hub/`
- **AUTH SEVER URL** = `https://sso.redhat.com/auth/realms/redhat-external/protocol/openid-connect/token`

4. To use a private Automation Hub, create an Automation Hub credential using a token retrieved from the Repo Management dashboard of your local Automation Hub and pointing to the published repo URL as shown:

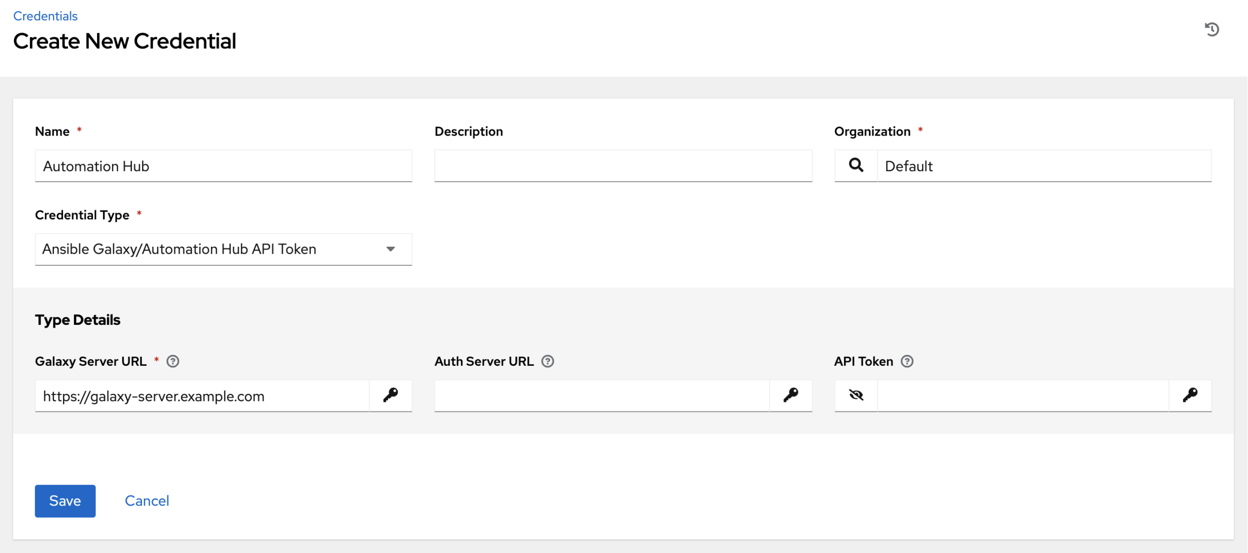

Repo Management

Distribution name	Repository name	Content c...	Last updated	Sync URL	Ansible CLI URL
community	community	34	17 days ago	https://10.10.94.209/api/galaxy/conte...	https://10.10.94.209/api/galaxy/conte...
published	published	6	5 days ago	https://10.10.94.209/api/galaxy/conte...	https://10.10.94.209/api/galaxy/conte...
red-hat-certified	rh-certified	195	an hour ago	https://10.10.94.209/api/galaxy/conte...	https://10.10.94.209/api/galaxy/conte...


Repo Management

Distribution name	Repository name	Content c...	Last updated	Sync URL	Ansible CLI URL
community	community	34	17 days ago	https://10.10.94.209/api/galaxy/conte...	https://10.10.94.209/api/galaxy/conte...
published	published	6	5 days ago	https://10.10.94.209/api/galaxy/conte...	https://10.10.94.209/api/galaxy/conte...
red-hat-certified	rh-certified	195	an hour ago	https://10.10.94.209/api/galaxy/conte...	https://10.10.94.209/api/galaxy/conte...

You can create different repos with different namespaces/collections in them. But for each repo in Automation Hub you need to create a different Automation Hub credential. Copy the **Ansible CLI URL** from the Automation Hub UI in the format of `https://$<hub_url>/api/galaxy/content/<repo you want to pull from>` into the **Galaxy Server URL** field of the *Create Credential* form:



Credentials

Create New Credential

Name * Automation Hub

Description

Organization * Default

Credential Type * Ansible Galaxy/Automation Hub API Token

Type Details

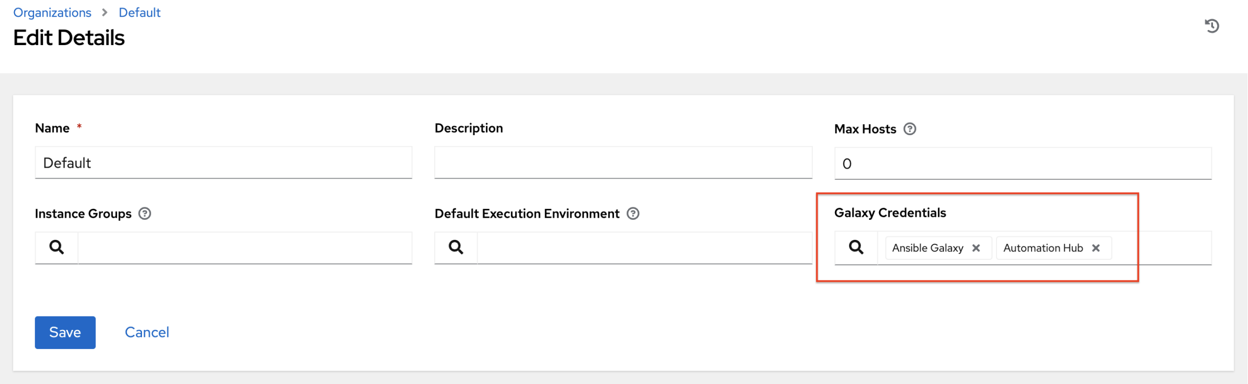
Galaxy Server URL * https://galaxy-server.example.com

Auth Server URL

API Token

Refer to Managing Red Hat Certified and Ansible Galaxy Collections in Ansible Hub for Automation Hub UI-specific instructions.

5. Navigate to the organization for which you want to be able to sync content from Automation Hub and add the new Automation Hub credential to the organization. This step allows you to associate each organization with the Automation Hub credential (i.e. repo) that you want to be able to use content from.



Organizations > Default

Edit Details

Name * Default

Description

Max Hosts 0

Instance Groups

Default Execution Environment

Galaxy Credentials: Ansible Galaxy, Automation Hub

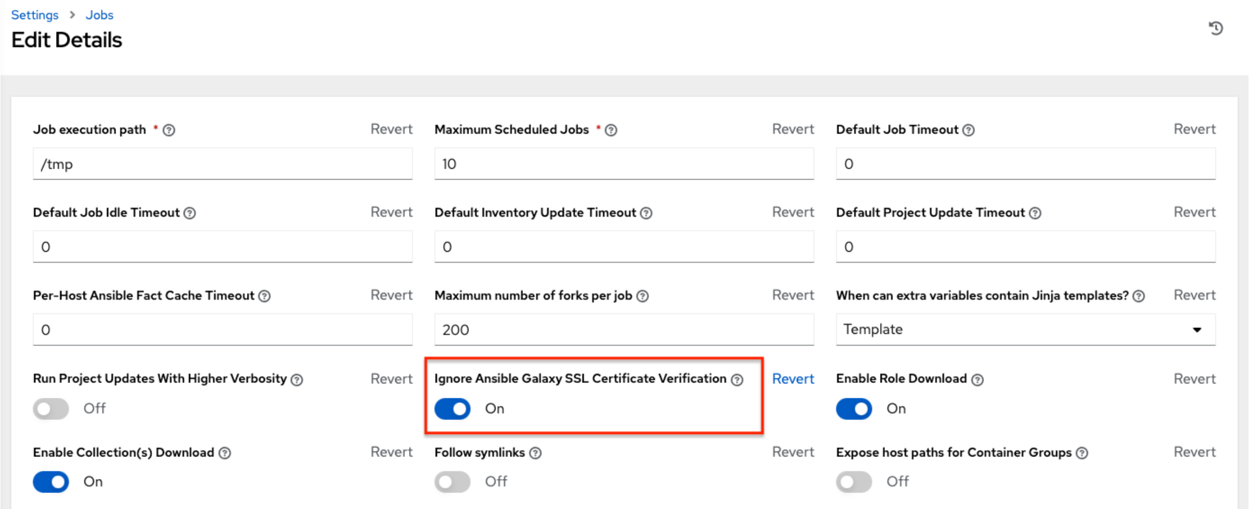
Note: Suppose you have two repos:

- *Prod*: `Namespace 1` and `Namespace 2`, each with collection `A` and `B` so: `namespace1.collectionA:v2.0.0` and `namespace2.collectionB:v2.0.0`
- *Stage*: `Namespace 1` with only collection `A` so: `namespace1.collectionA:v1.5.0` on Automation Hub, you will have a repo URL for *Prod* and *Stage*.

You can create an Automation Hub credential for each one. Then you can assign different levels of access to different organizations. For example, you can create a Developers organization has access to both repos, while an Operations organization just has access to the Automation Hub **Prod** repo only.

Refer to Managing User Access in Ansible Hub for Automation Hub UI-specific instructions.

6. If the Automation Hub has self-signed certificates, click the toggle to enable the setting **Ignore Ansible Galaxy SSL Certificate Verification**. For **public Automation Hub**, which uses a signed certificate, click the toggle to disable it instead. Note this is a global setting:



Settings > Jobs

Edit Details

Job execution path * ⓘ Revert
/tmp

Maximum Scheduled Jobs * ⓘ Revert
10

Default Job Timeout ⓘ Revert
0

Default Job Idle Timeout ⓘ Revert
0

Default Inventory Update Timeout ⓘ Revert
0

Default Project Update Timeout ⓘ Revert
0

Per-Host Ansible Fact Cache Timeout ⓘ Revert
0

Maximum number of forks per job ⓘ Revert
200

When can extra variables contain Jinja templates? ⓘ Revert
Template

Run Project Updates With Higher Verbosity ⓘ Revert
Off

Ignore Ansible Galaxy SSL Certificate Verification ⓘ Revert
On

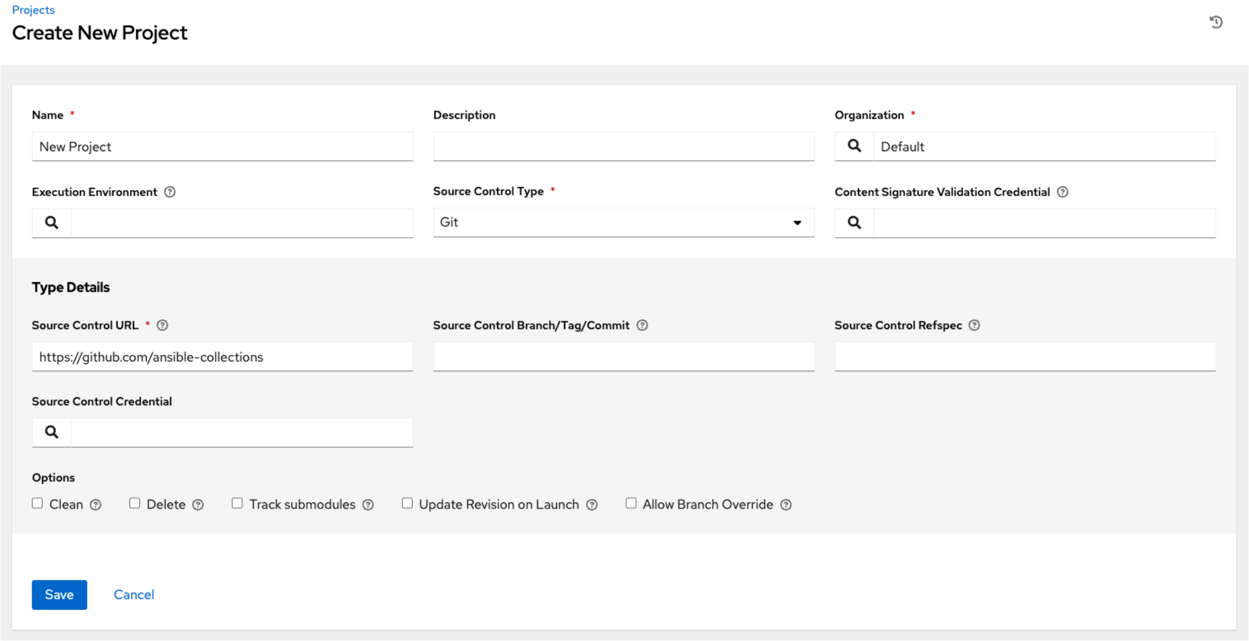
Enable Role Download ⓘ Revert
On

Enable Collection(s) Download ⓘ Revert
On

Follow symlinks ⓘ Revert
Off

Expose host paths for Container Groups ⓘ Revert
Off

7. Create a project, where the source repository specifies the necessary collections in a requirements file located in the `collections/requirements.yml` file. Refer to the syntax described in the Ansible documentation: https://docs.ansible.com/ansible/latest/user_guide/collections_using.html#install-multiple-collections-with-a-requirements-file.



Projects

Create New Project

Name *
New Project

Description

Organization *
Default

Execution Environment ⓘ

Source Control Type *
Git

Content Signature Validation Credential ⓘ

Type Details

Source Control URL * ⓘ
https://github.com/ansible-collections

Source Control Branch/Tag/Commit ⓘ

Source Control Refspec ⓘ

Source Control Credential

Options

□ Clean ⓘ □ Delete ⓘ □ Track submodules ⓘ □ Update Revision on Launch ⓘ □ Allow Branch Override ⓘ

8. In the Projects list view, click 🔄 to run an update against this project. Automation controller fetches the Galaxy collections from the `collections/requirements.yml` file and report it as changed; and the collections will now be installed for any job template using this project.

Note: If updates are needed from Galaxy or Collections, a sync is performed that downloads the required roles, consuming that much more space in your /tmp file. In cases where you have a big project (around 10 GB), disk space on `/tmp` may be an issue.

For more information on collections, refer to Using Collections. For more information on how Red Hat itself publishes one of these official collections, which can be used to automate your automation controller install directly, refer to the

AWX Ansible Collection documentation. This page is accessible with your Red Hat customer credentials as part of your Red Hat Ansible Automation Platform subscription.

CHAPTER

SEVENTEEN

PROJECT SIGNING AND VERIFICATION

Project signing and verification provides the ability to sign files in your project directory and then verify whether or not that content has changed in any way, or files have been added or removed from the project unexpectedly. To accomplish this, a private key for signing and a matching public key for verifying are needed.

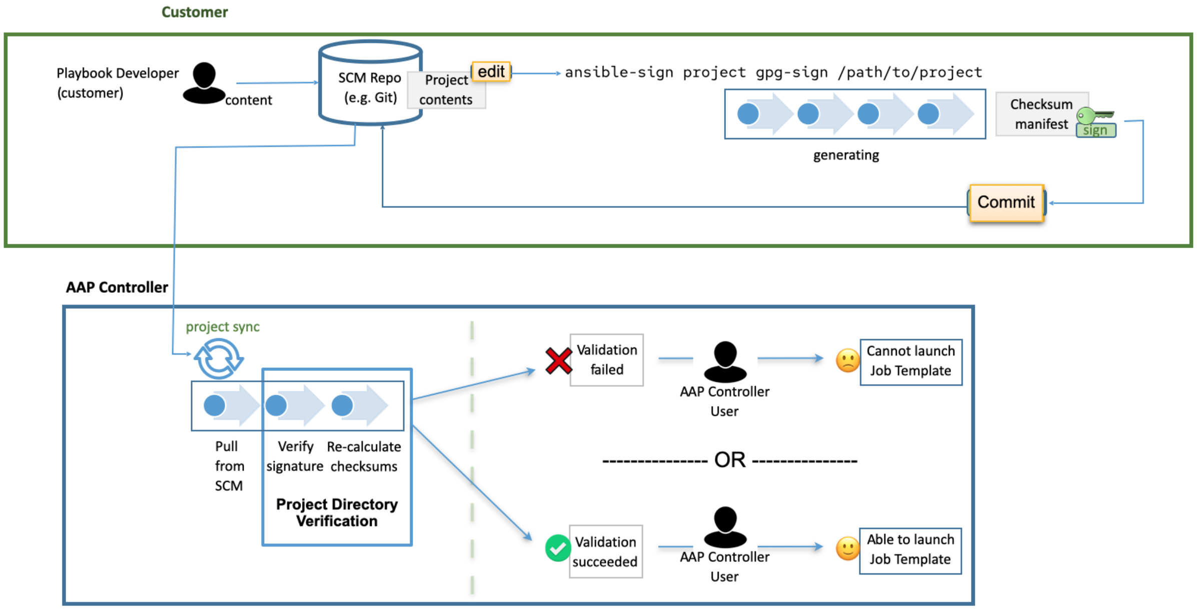
For project maintainers, the supported way to perform content signing is to use a utility called, `ansible-sign`, through the command-line interface (CLI) that comes with it.

The CLI aims to make it easy to use cryptographic technology like GNU Privacy Guard (GPG) to validate that specified files within a project have not been tampered with in any way. Currently, GPG is the only supported means of signing and validation.

The Ansible Automation controller is used to verify the signed content. After a matching public key has been associated with the signed project, the controller will verify that the files included during signing have not changed, and that files have been added or removed unexpectedly. If the signature is not valid or a file has changed, the project will fail to update, and no jobs making use of the project will be able to launch. Verification status of the project ensures that only secure, untampered content is run in jobs.

Assuming that the repository has already been configured for signing and verification (see below), the usual workflow for altering the project becomes the following:

1. User has a project repository set up already and wants to make a change to a file.
2. User makes the change, runs `ansible-sign project gpg-sign /path/to/project`, which updates a checksum manifest and signs it.
3. User commits the change and the updated checksum manifest and the signature to the repository.
4. When the user syncs the project, the controller (already configured, in this scenario) pulls in the new changes, checks that the public key associated with the project in the controller matches the private key that the checksum manifest was signed with (this prevents tampering with the checksum manifest itself), then re-calculates checksums of each file in the manifest to ensure that the checksum matches (and thus that no file has changed). It also looks to ensure that all files are accounted for: They must have been either included in, or excluded from, the `MANIFEST.in` file discussed below; if files have been added or removed unexpectedly, verification will fail.



17.1 Prerequisites

- RHEL nodes must properly be subscribed to:
  - RHEL subscription and with **baseos** and **appstream** repositories enabled
  - Ansible Automation Platform subscription and the proper Ansible Automation Platform channel enabled:

```
ansible-automation-platform-2.3-for-rhel-8-x86_64-rpms for RHEL 8
ansible-automation-platform-2.3-for-rhel-9-x86_64-rpms for RHEL 9
```

- A valid GPG public/private keypair is required for signing content. Refer to How to create GPG keypairs for details.

  Vist the GnuPG documentation for more information regarding GPG keys.

  You can verify that you have a valid GPG keypair and in your default GnuPG keyring, with the following command:

```
$ gpg --list-secret-keys
```

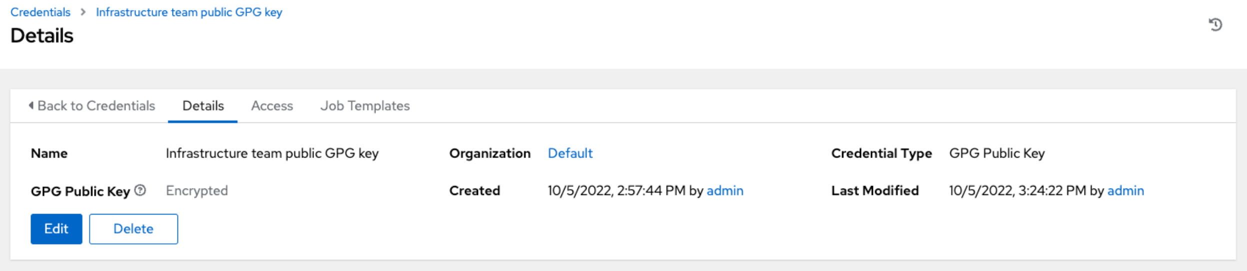
  If the above command produces no output, or one line of output that states, `trustdb was created`, then you do not have a secret key in your default keyring. In this case, refer to How to create GPG keypairs to learn how to create a new keypair before proceeding. If it produces output other than that, you have a valid secret key and are ready to move on to using `ansible-sign`.

17.2 Add a GPG key to Ansible Automation Controller

In order to use the GPG key for content singing and validation in the controller, you must add it running the following command in the CLI:

```
$ gpg --list-keys
$ gpg --export --armour <key fingerprint> > my_public_key.asc
```

1. In the controller user interface, click **Credentials** from the left side navigation menu then click the **Add** button.
2. Provide the new credential a meaningful name (for example, "Infrastructure team public GPG key")
3. In the Credential Type field, select **GPG Public Key**.
4. Click **Browse** to locate and select the public key file (e.g., `my_public_key.asc`)
5. Click **Save** when done.

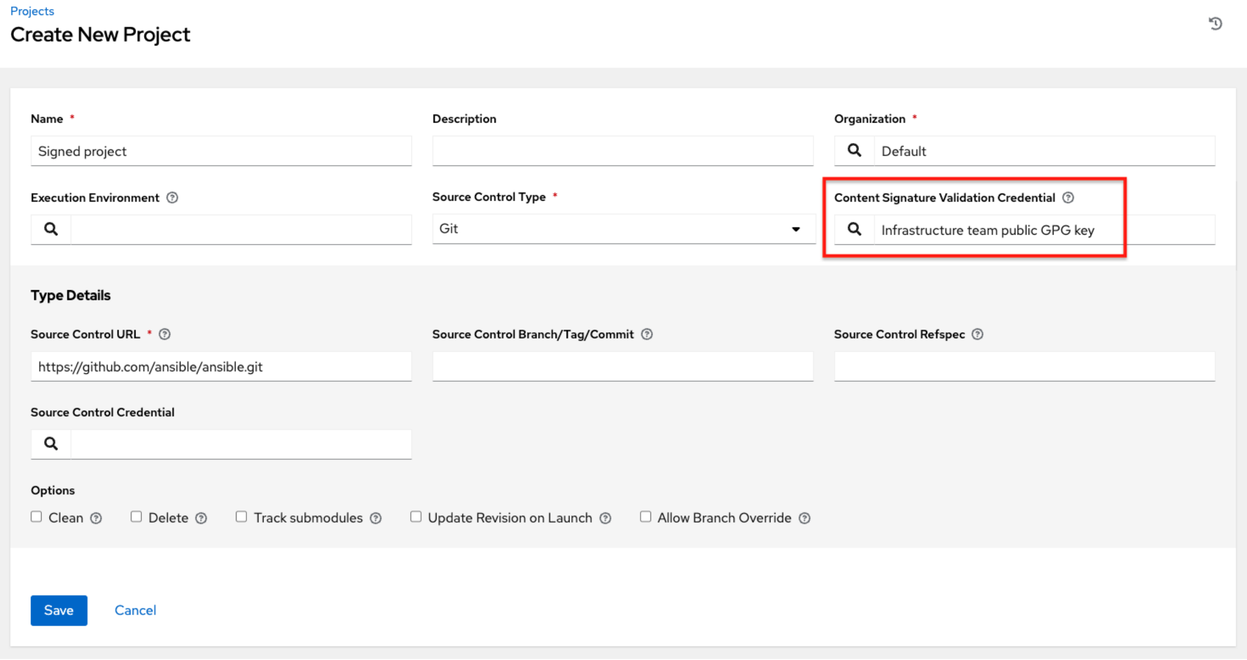


Credentials > Infrastructure team public GPG key

Details

Name	Infrastructure team public GPG key	Organization	Default	Credential Type	GPG Public Key
GPG Public Key	Encrypted	Created	10/5/2022, 2:57:44 PM by admin	Last Modified	10/5/2022, 3:24:22 PM by admin

This credential can now be selected in *projects*, and content verification will automatically take place on future project syncs.



Projects

Create New Project

Name * : Signed project

Description :

Organization * : Default

Execution Environment :

Source Control Type * : Git

Content Signature Validation Credential : Infrastructure team public GPG key

Type Details

Source Control URL * : https://github.com/ansible/ansible.git

Source Control Branch/Tag/Commit :

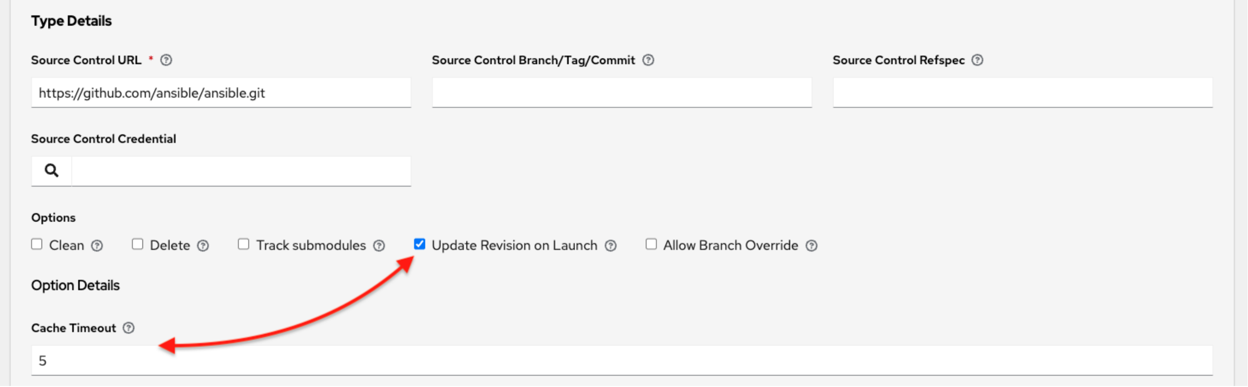
Source Control Refspec :

Source Control Credential :

Options

□ Clean □ Delete □ Track submodules □ Update Revision on Launch □ Allow Branch Override

Note: Use the project cache SCM timeout to control how often you want the controller to re-validate the signed content. When a project is configured to update on launch (of any job template configured to use that project), you can enable the cache timeout setting, which tells it to update after N seconds have passed since the last update. If validation is running too frequently, you can slow down how often project updates occur by specifying the time in the **Cache Timeout** field of the Option Details pane of the project.



Type Details

Source Control URL *

https://github.com/ansible/ansible.git

Source Control Branch/Tag/Commit

Source Control Refspec

Source Control Credential

Options

☐ Clean ☐ Delete ☐ Track submodules ☑ Update Revision on Launch ☐ Allow Branch Override

Option Details

Cache Timeout

5

17.3 Access the `ansible-sign` CLI utility

The `ansible-sign` utility provide options for the user to sign and verify whether the project is signed.

1. Run the following command to install `ansible-sign`:

```
$ dnf install ansible-sign
```

2. Verify that `ansible-sign` was successfully installed:

```
$ ansible-sign --version
```

Output similar to the following displays (possibly with a different version number):

```
ansible-sign 0.1
```

This indicates you have successfully installed `ansible-sign`.

17.4 Signing your project

As the name suggests, signing a project involves an Ansible project directory. Refer to the Ansible documentation for more sophisticated examples of project directory structures.

The following sample project has a very simple structure. An inventory file, and two small playbooks under a playbooks directory:

```
$ cd sample-project/
$ tree -a .
.
├── inventory
└── playbooks
    └── get_uptime.yml
    └── hello.yml

1 directory, 3 files
```

Note: The commands used in this section assume that your working directory is the root of your project. As a rule, `ansible-sign project` commands always take the project root directory as their last argument, and therefore, we use `.` to indicate the current working directory.

The way that `ansible-sign` protects content from tampering is by taking checksums (SHA256) of all of the secured files in the project, compiling those into a checksum manifest file, and then finally signing that manifest file.

The first step toward signing content is to create a file that tells `ansible-sign` which files to protect. This file should be called `MANIFEST.in` and reside in the project root directory.

Internally, `ansible-sign` makes use of the `distlib.manifest` module of Python's distlib library, and thus `MANIFEST.in` must follow the syntax that this library specifies. See the Python Packaging User Guide for an explanation of the `MANIFEST.in` file directives.

In the sample project, included are two directives, resulting in a `MANIFEST.in` file that looks like this:

```
include inventory
recursive-include playbooks *.yml
```

With this file in place, generate your checksum manifest file and sign it. Both of these steps are achieved in a single `ansible-sign` command:

```
$ ansible-sign project gpg-sign .
[OK   ] GPG signing successful!
[NOTE ] Checksum manifest: ./.ansible-sign/sha256sum.txt
[NOTE ] GPG summary: signature created
```

Now the project has been signed.

Notice that the `gpg-sign` subcommand resides under the `project` subcommand. For signing project content, every command will start with `ansible-sign project`. As noted above, as a rule, every `ansible-sign project` command takes the project root directory as its final argument.

As mentioned earlier, `ansible-sign` by default makes use of your default keyring and looks for the first available secret key that it can find, to sign your project. You can specify a specific secret key to use with the `--fingerprint` option, or even a completely independent GPG home directory with the `--gnupg-home` option.

Note: If you are using a desktop environment, GnuPG will automatically prompt you for your secret key's passphrase. If this functionality does not work, or you are working without a desktop environment (e.g., via SSH), you can use the `-p/--prompt-passphrase` flag after `gpg-sign` in the above command, which will cause `ansible-sign` to prompt for the password instead.

Upon viewing the structure of the project directory, notice that a new `.ansible-sign` directory was created. This directory contains the checksum manifest and a detached GPG signature for it.

```
$ tree -a .
.
├── .ansible-sign
│   ├── sha256sum.txt
│   └── sha256sum.txt.sig
├── inventory
├── MANIFEST.in
└── playbooks
    ├── get_uptime.yml
    └── hello.yml
```

17.5 Verifying your project

If you want to verify that a signed Ansible project has not been altered, you can use `ansible-sign` to check whether the signature is valid and that the checksums of the files match what the checksum manifest says they should be. In particular, the `ansible-sign project gpg-verify` command can be used to automatically verify both of these conditions.

```
$ ansible-sign project gpg-verify .
[OK   ] GPG signature verification succeeded.
[OK   ] Checksum validation succeeded.
```

Note: By default, `ansible-sign` makes use of your default GPG keyring to look for a matching public key. You can specify a keyring file with the `--keyring` option, or a different GPG home with the `--gnugpg-home` option.

If verification fails for any reason, information will be displayed to help you debug the cause. More verbosity can be enabled by passing the global `--debug` flag, immediately after `ansible-sign` in your commands.

Note: When a GPG credential is used in a project, content verification will automatically take place on future project syncs.

17.6 Automate signing

In environments with highly-trusted CI environments (e.g., OpenShift, Jenkins, etc.), it is possible to automate the signing process. For example, you could store your GPG private key in a CI platform of choice as a secret, and import that into GnuPG in the CI environment. You could then run through the signing workflow above within the normal CI workflow/container/environment.

When signing a project using GPG, the environment variable `ANSIBLE_SIGN_GPG_PASSPHRASE` can be set to the passphrase of the signing key. This can be injected (and masked/secured) in a CI pipeline.

Depending on the scenario at hand, `ansible-sign` will return with a different exit-code, during both signing and verification. This can also be useful in the context of CI and automation, as a CI environment can act differently based on the failure (for example, sending alerts for some errors but silently failing for others).

These are the exit codes used in `ansible-sign` currently, which can be considered stable:

Exit code	Approximate meaning	Example scenarios
0	Success	• Signing was successful • Verification was successful
1	General failure	• The checksum manifest file contained a syntax error during verification • The signature file did not exist during verification • `MANIFEST.in` did not exist during signing
2	Checksum verification failure	• The checksum hashes calculated during verification differed from what was in the signed checksum manifest (e.g., a project file was changed but the signing process was not re-completed)
3	Signature verification failure	• The signer's public key was not in the user's GPG keyring • The wrong GnuPG home directory or keyring file was specified • The signed checksum manifest file was modified in some way
4	Signing process failure	• The signer's private key was not found in the GPG keyring • The wrong GnuPG home directory or keyring file was specified

CHAPTER

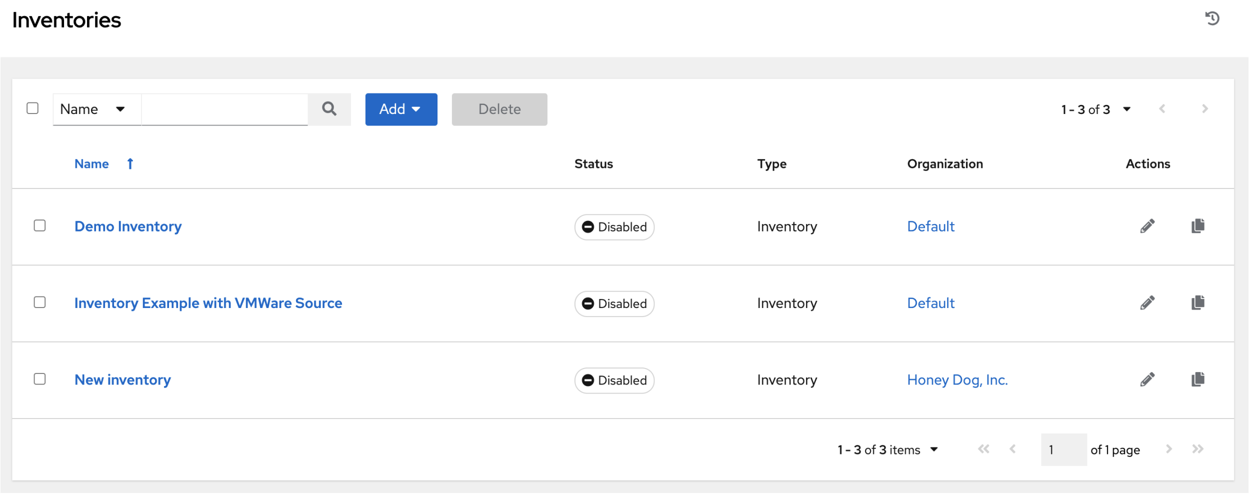
EIGHTEEN

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.



Inventories

	Name ↑	Status	Type	Organization	Actions
□	Demo Inventory	Disabled	Inventory	Default	✎ ⧉
□	Inventory Example with VMWare Source	Disabled	Inventory	Default	✎ ⧉
□	New inventory	Disabled	Inventory	Honey Dog, Inc.	✎ ⧉

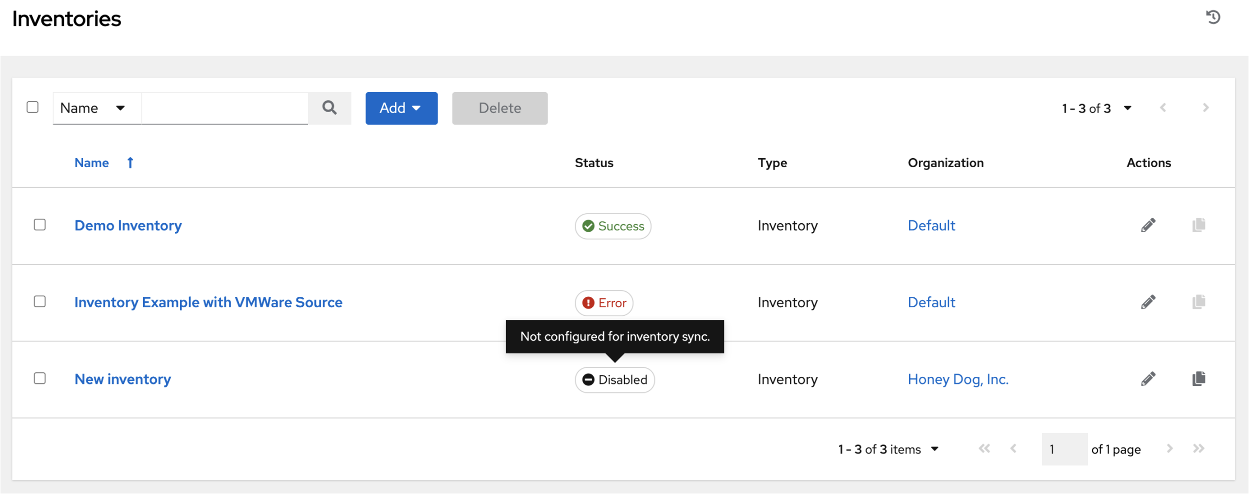
The list of Inventory details includes:

- **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 ✎ 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:



Inventories

□ Name ▾ | Add ▾ | Delete | 1 - 3 of 3 ▾

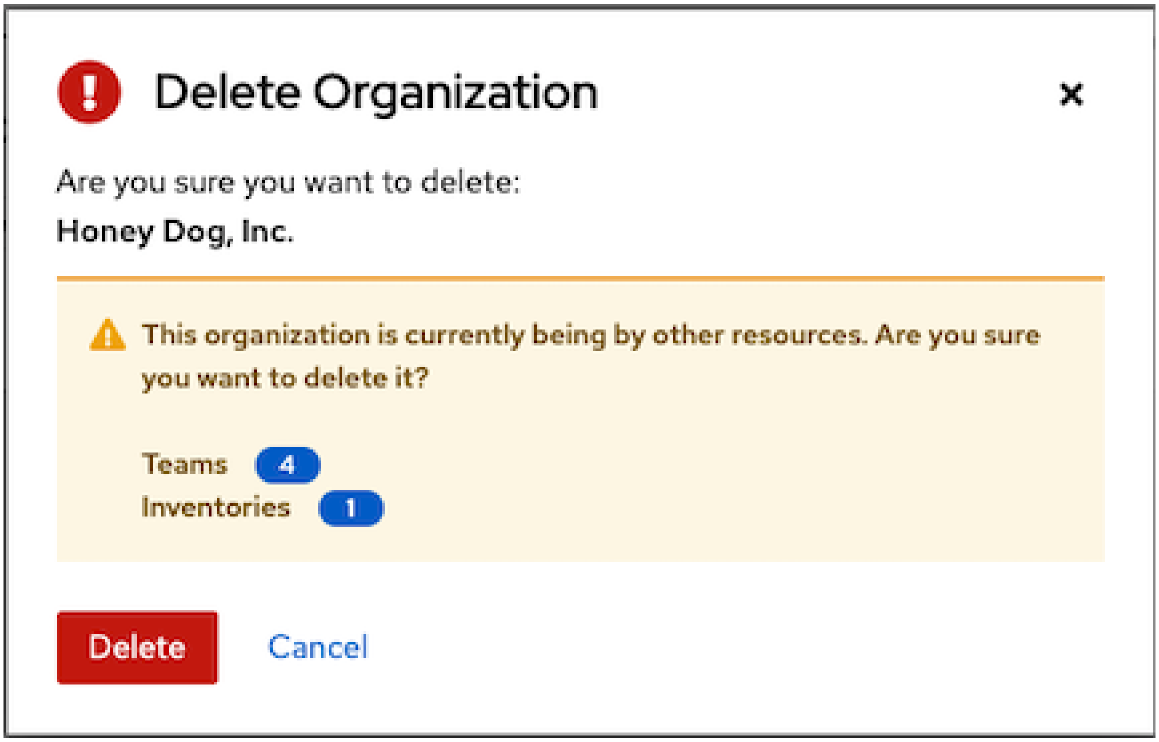
□	Name ↑	Status	Type	Organization	Actions
□	Demo Inventory	Success	Inventory	Default	✎ ⧉
□	Inventory Example with VMWare Source	Error	Inventory	Default	✎ ⧉
□	New inventory	Disabled	Inventory	Honey Dog, Inc.	✎ ⧉

Not configured for inventory sync.

1 - 3 of 3 items ▾ | 1 of 1 page

- **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 the properties for the selected inventory
  - **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:



Delete Organization

Are you sure you want to delete:

Honey Dog, Inc.

This organization is currently being by other resources. Are you sure you want to delete it?

Teams 4

Inventories 1

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
```

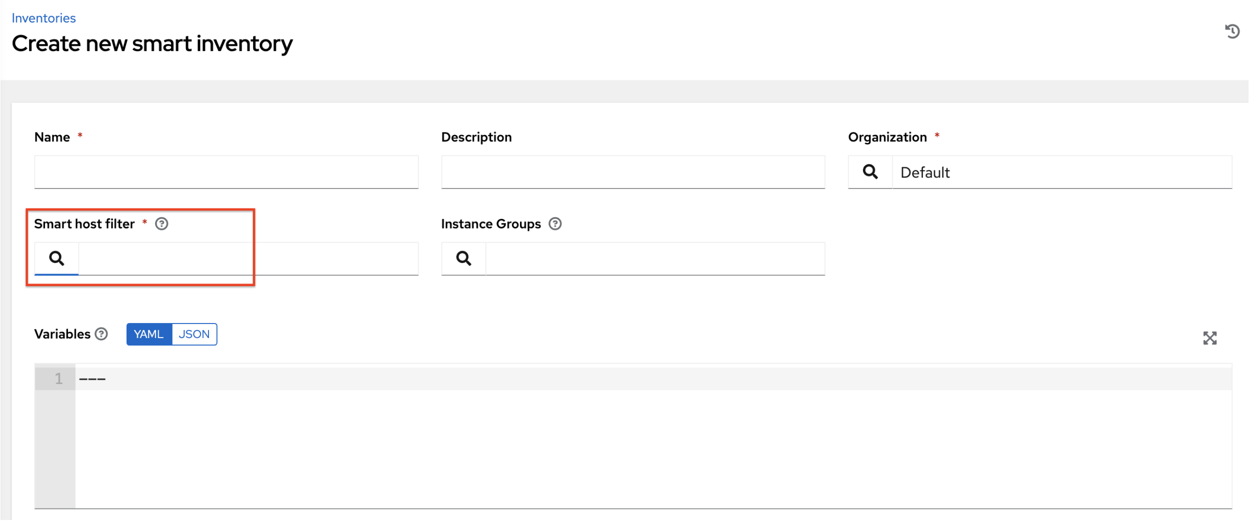
If a search term in `host_filter` is of string type, to make the value a number (e.g. `2.66`), or a JSON keyword (e.g. `null`, `true` or `false`) valid, add double quotations around the value to prevent the controller from mistakenly parsing it as a non-string:

```
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 🔍 button next to the **Smart host filter** field to open a pop-up window to filter hosts for this inventory.



Inventories

Create new smart inventory

Name *

Description

Organization * Default

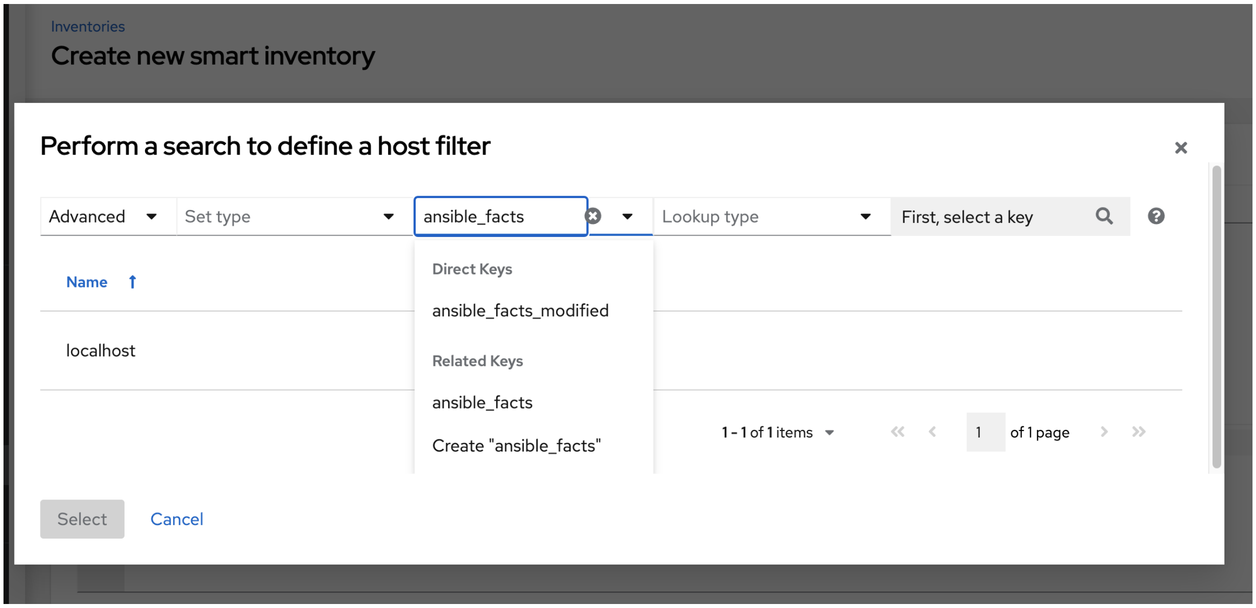
Smart host filter *

Instance Groups

Variables

1 ---

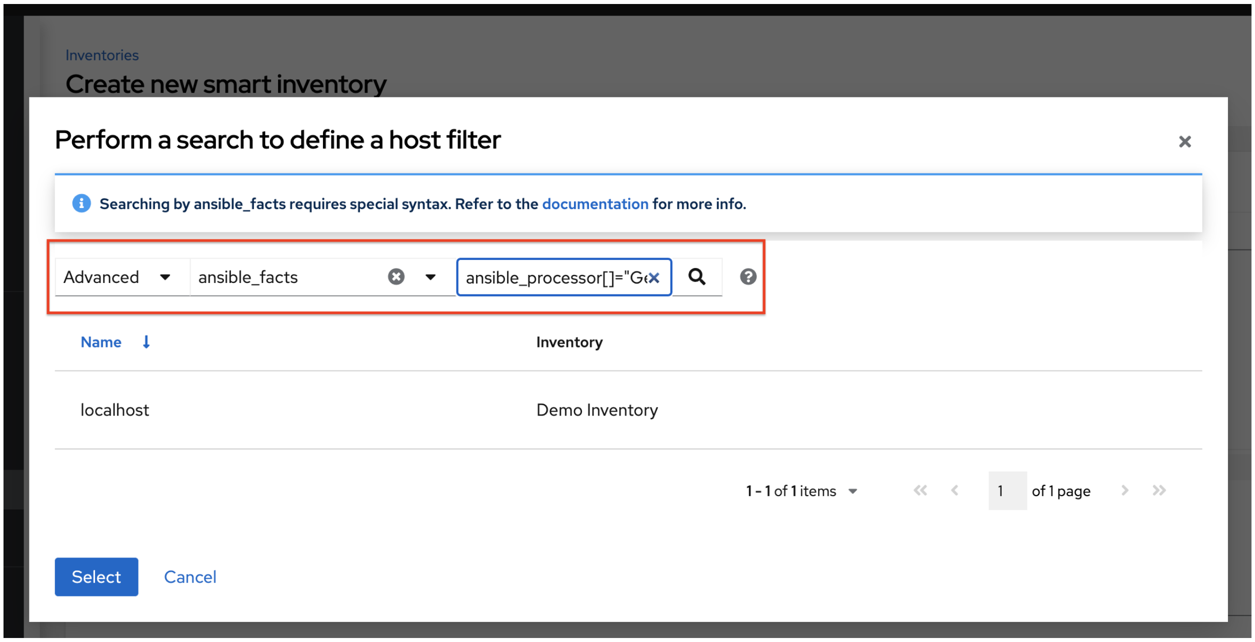
2. In the search pop-up window, change the search criteria from **Name** to **Advanced** and select **ansible_facts** from the **Key** field.



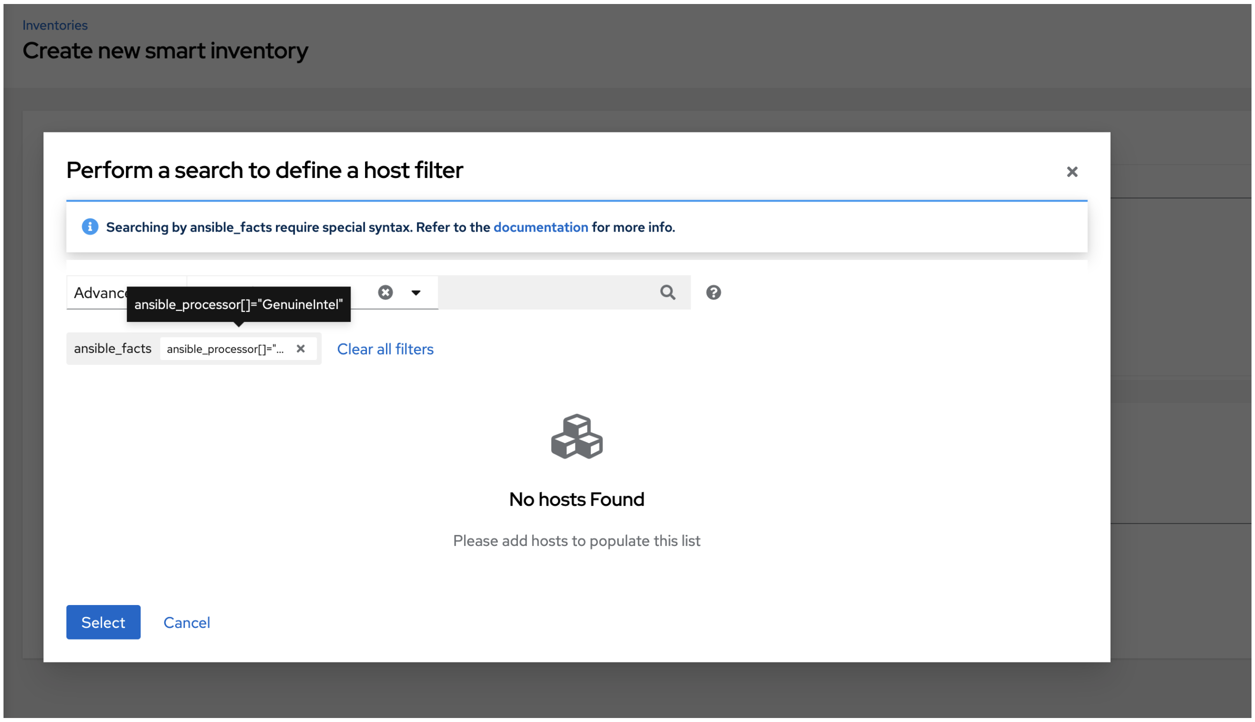
If you wanted to add an ansible fact of

```
/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]**.


Searching by ansible_facts requires special syntax. Refer to the documentation for more info.

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



Inventories

Create new smart inventory

Perform a search to define a host filter

Searching by ansible_facts require special syntax. Refer to the documentation for more info.

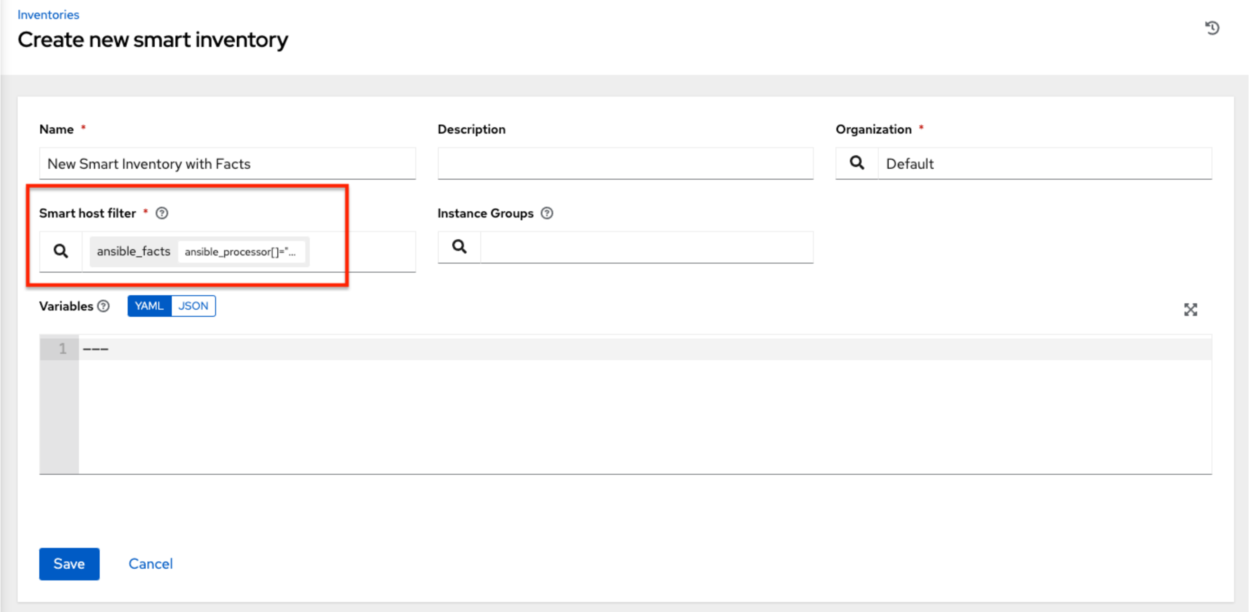
Advanc ansible_processor[]="GenuineIntel"

ansible_facts ansible_processor[]="...

No hosts Found

Please add hosts to populate this list

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



Inventories

Create new smart inventory

Name * New Smart Inventory with Facts

Description

Organization * Default

Smart host filter * ansible_facts ansible_processor[]="...

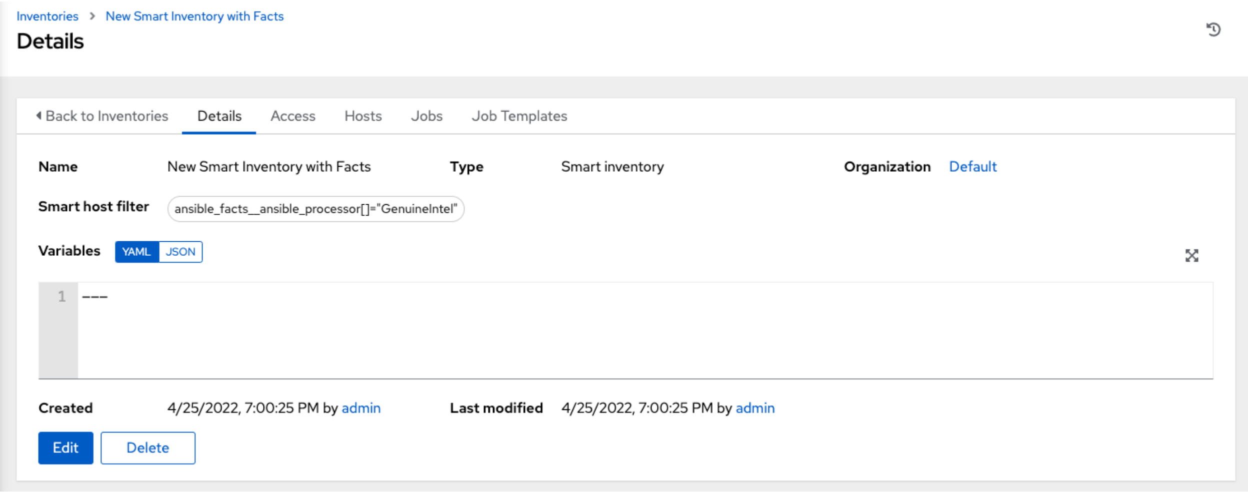
Instance Groups

Variables YAML JSON

```
---
```

4. 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.

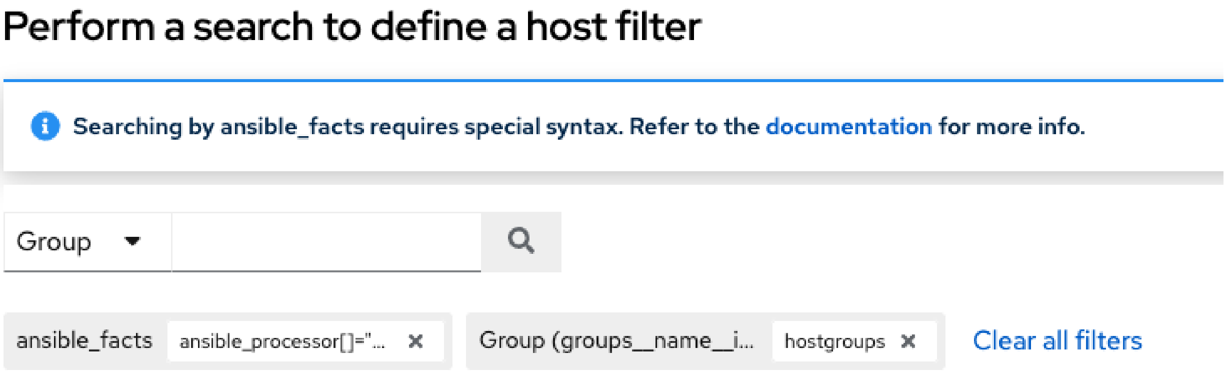


Name: New Smart Inventory with Facts
Type: Smart inventory
Organization: Default
Smart host filter: ansible_facts__ansible_processor[]="GenuineIntel"
Variables:

```
---
```

Created: 4/25/2022, 7:00:25 PM by admin
Last modified: 4/25/2022, 7:00:25 PM by admin

5. 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.



Searching by ansible_facts requires special syntax. Refer to the documentation for more info.

ansible_facts: ansible_processor[]="...
Group (groups__name__i...: hostgroups

18.2 Inventory Plugins

Inventory updates use dynamically-generated YAML files which are parsed by their respective inventory plugin. In Automation Controller Version 4.3.0, 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

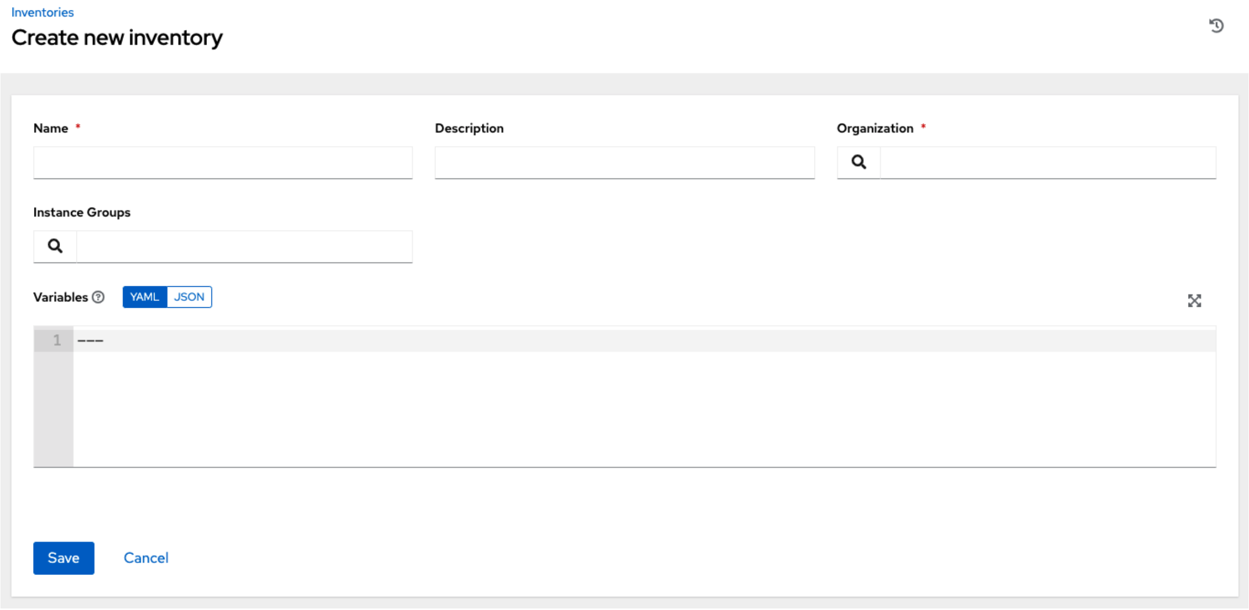
Adding a new inventory involves several components:

- *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.



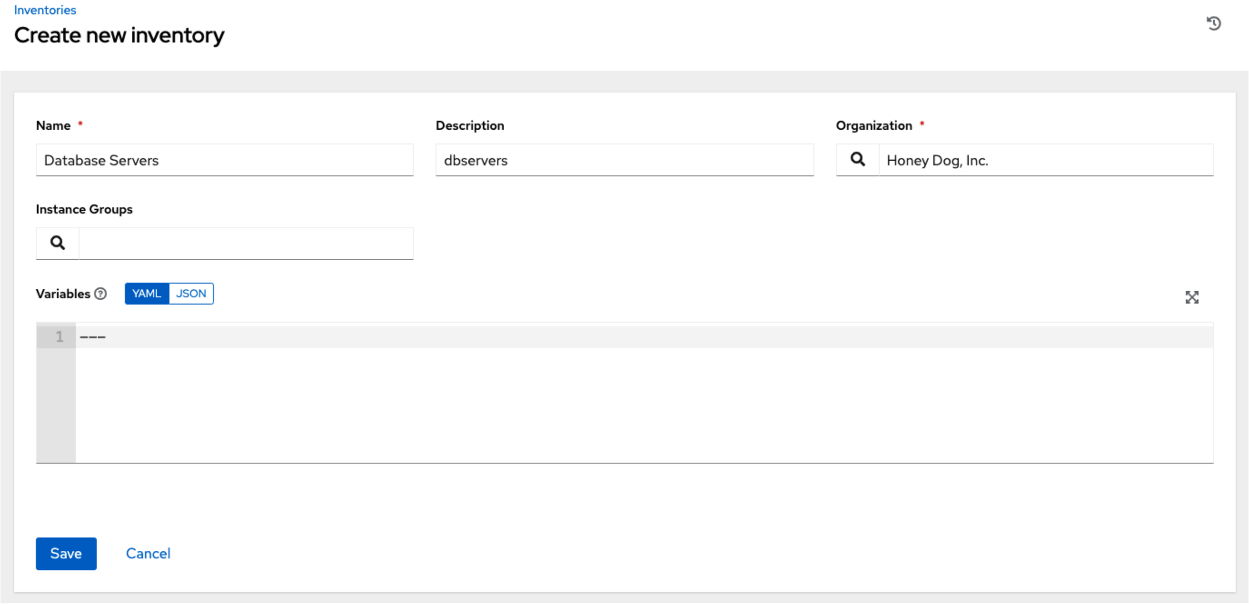
2. Enter the appropriate details into the following fields:

- **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 🔍 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 🔍 button to open a separate window. Choose the instance groups for this inventory to run on. If the list is extensive, use the search to narrow the options.
- **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.



Inventories

Create new inventory

Name * Database Servers

Description dbservers

Organization * Honey Dog, Inc.

Instance Groups

Variables ● YAML ○ JSON

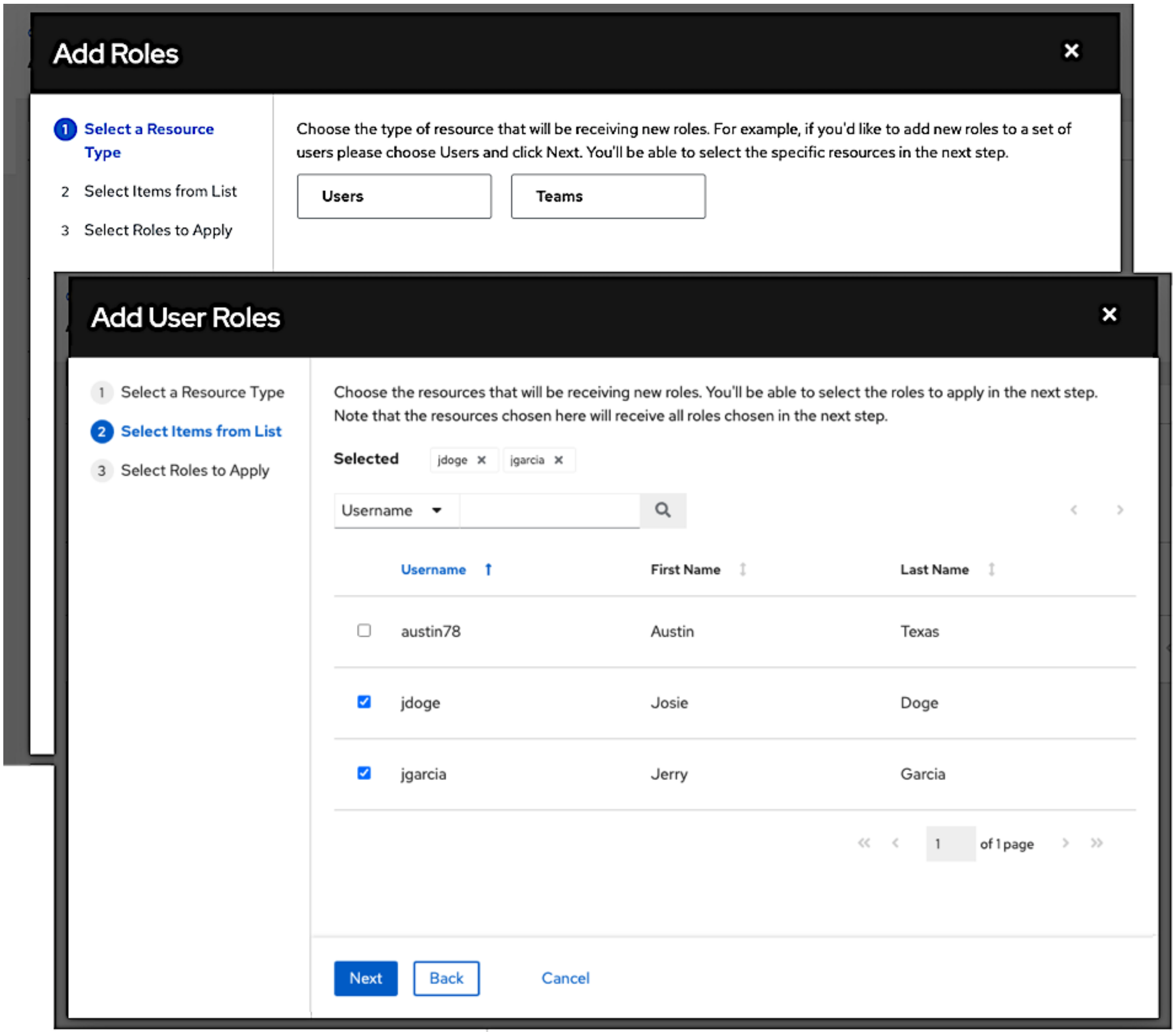
```
---
```

3. 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**.



Add Roles

1 Select a Resource Type

2 Select Items from List

3 Select Roles to Apply

Choose the type of resource that will be receiving new roles. For example, if you'd like to add new roles to a set of users please choose Users and click Next. You'll be able to select the specific resources in the next step.

Users | Teams

Add User Roles

1 Select a Resource Type

2 Select Items from List

3 Select Roles to Apply

Choose the resources that will be receiving new roles. You'll be able to select the roles to apply in the next step. Note that the resources chosen here will receive all roles chosen in the next step.

Selected: jdoge × jgarcia ×

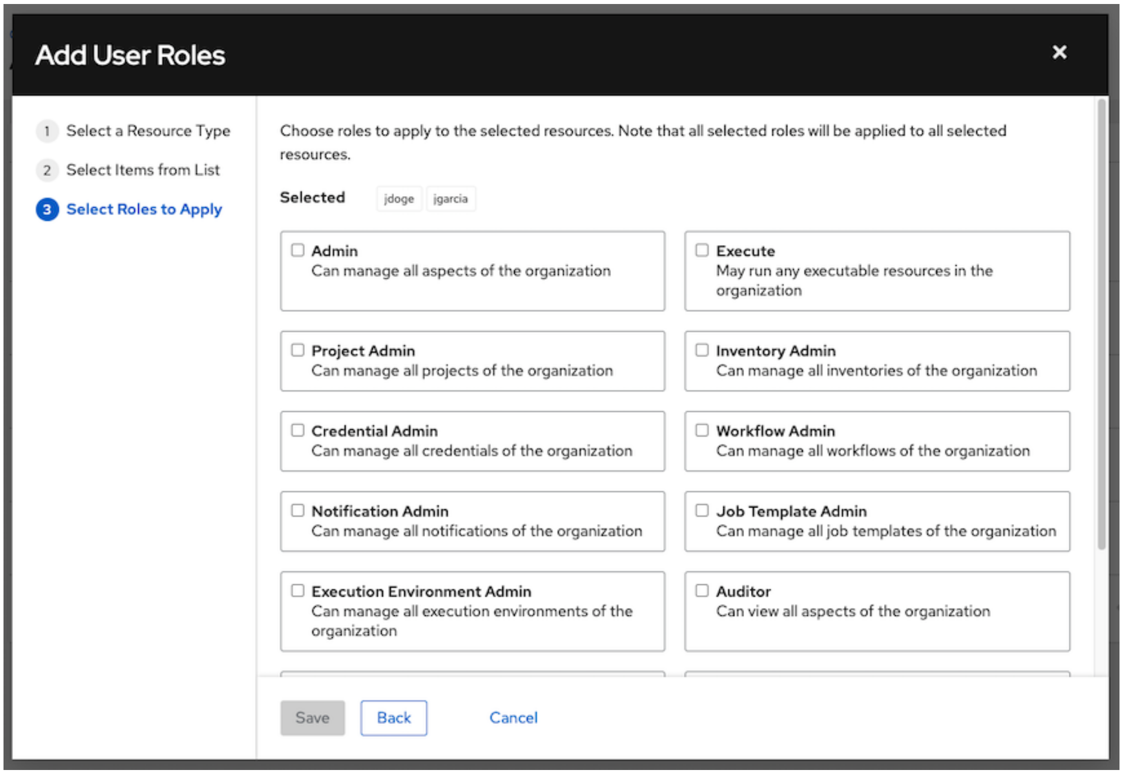
Username ▾

	Username	First Name	Last Name
□	austin78	Austin	Texas
☑	jdoge	Josie	Doge
☑	jgarcia	Jerry	Garcia

1 of 1 page

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

4. 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.



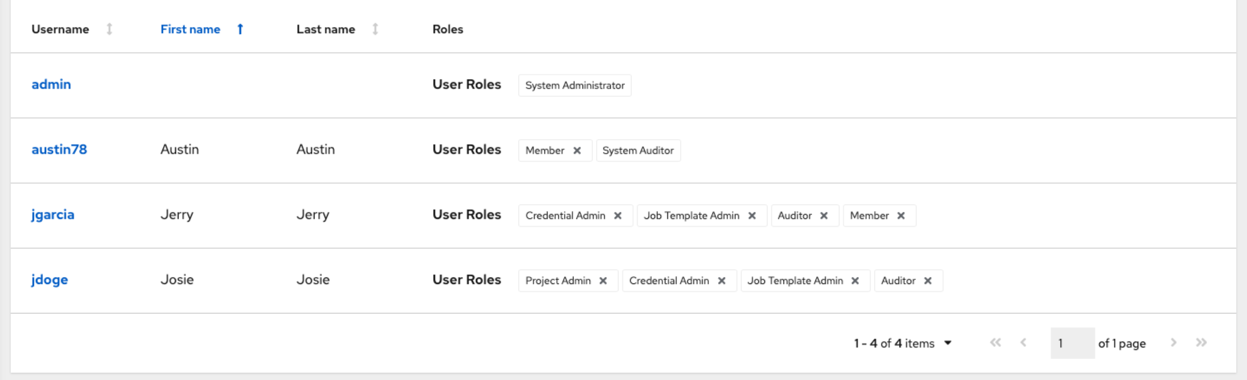
Choose roles to apply to the selected resources. Note that all selected roles will be applied to all selected resources.

Selected: jdoge, jgarcia

- □ **Admin** Can manage all aspects of the organization
- □ **Execute** May run any executable resources in the organization
- □ **Project Admin** Can manage all projects of the organization
- □ **Inventory Admin** Can manage all inventories of the organization
- □ **Credential Admin** Can manage all credentials of the organization
- □ **Workflow Admin** Can manage all workflows of the organization
- □ **Notification Admin** Can manage all notifications of the organization
- □ **Job Template Admin** Can manage all job templates of the organization
- □ **Execution Environment Admin** Can manage all execution environments of the organization
- □ **Auditor** Can view all aspects of the organization

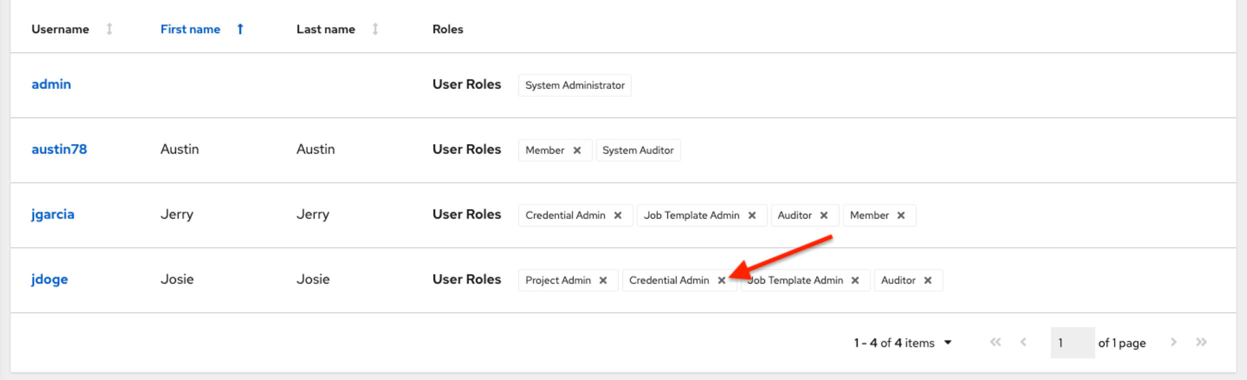
5. Click the **Save** button to apply the roles to the selected user(s) or team(s) and to add them as members.

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



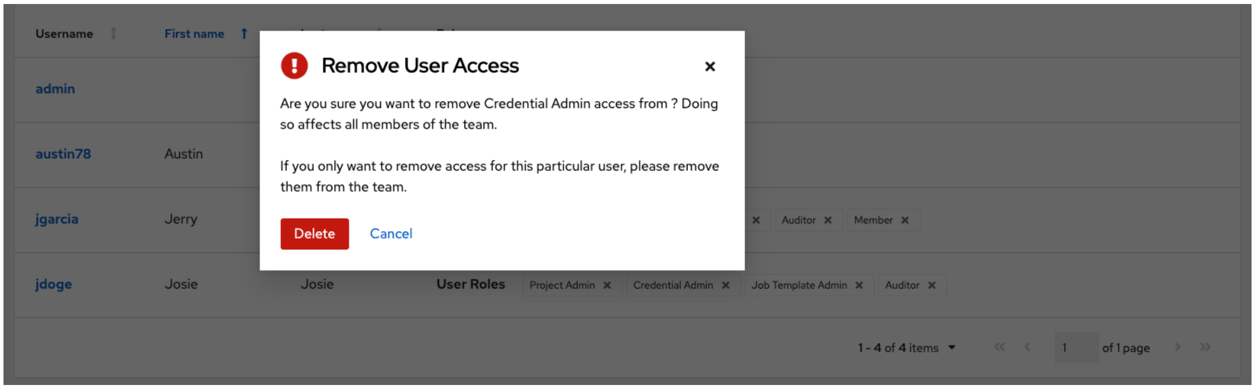
Username	First name	Last name	Roles
admin			User Roles: System Administrator
austin78	Austin	Austin	User Roles: Member ×, System Auditor
jgarcia	Jerry	Jerry	User Roles: Credential Admin ×, Job Template Admin ×, Auditor ×, Member ×
jdoge	Josie	Josie	User Roles: Project Admin ×, Credential Admin ×, Job Template Admin ×, Auditor ×

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



Username	First name	Last name	Roles
admin			User Roles: System Administrator
austin78	Austin	Austin	User Roles: Member ×, System Auditor
jgarcia	Jerry	Jerry	User Roles: Credential Admin ×, Job Template Admin ×, Auditor ×, Member ×
jdoge	Josie	Josie	User Roles: Project Admin ×, Credential Admin ×, Job Template Admin ×, Auditor ×

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



Remove User Access

Are you sure you want to remove Credential Admin access from ? Doing so affects all members of the team.

If you only want to remove access for this particular user, please remove them from the team.

18.3.2 Add groups

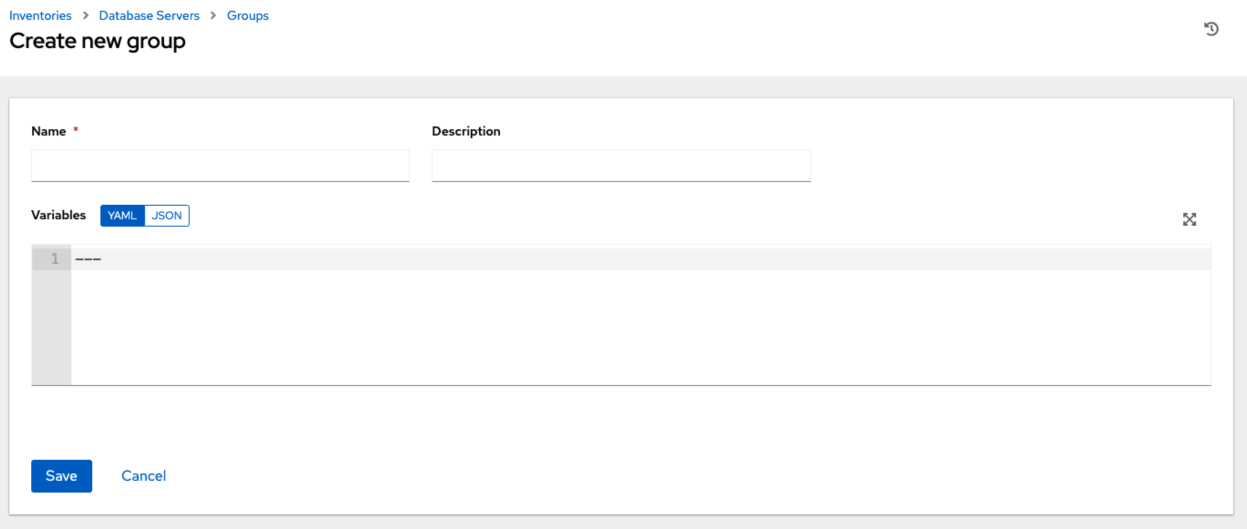
Inventories are divided into groups, which may contain hosts and other groups, and hosts. Groups are only applicable to standard inventories and is not a configurable directly through a Smart Inventory. You can associate an existing group through host(s) that are used with standard inventories. There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

2. 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.

3. When done, click **Save**.

Add groups within groups

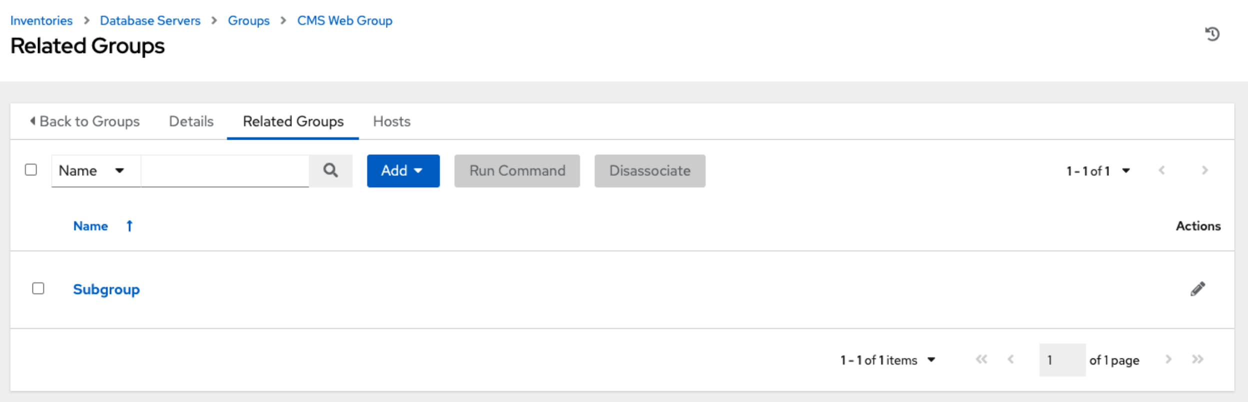
To add groups within groups:

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:

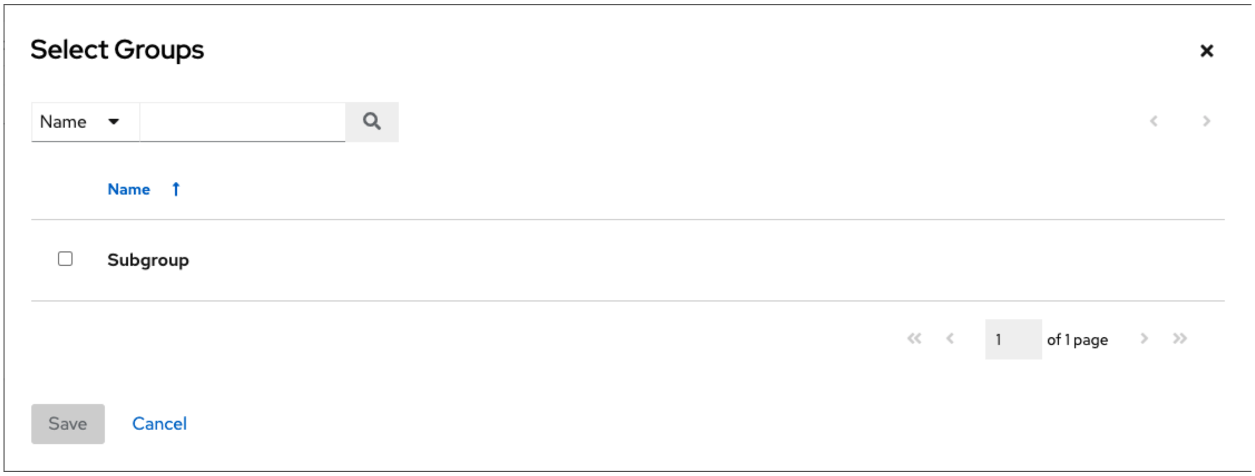
- **Name**: Required
- **Description**: Enter an arbitrary description as appropriate (optional)
- **Variables**: Enter definitions and values to be applied to all hosts in this group. Enter variables using either JSON or YAML syntax. Use the radio button to toggle between the two.

4. 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.



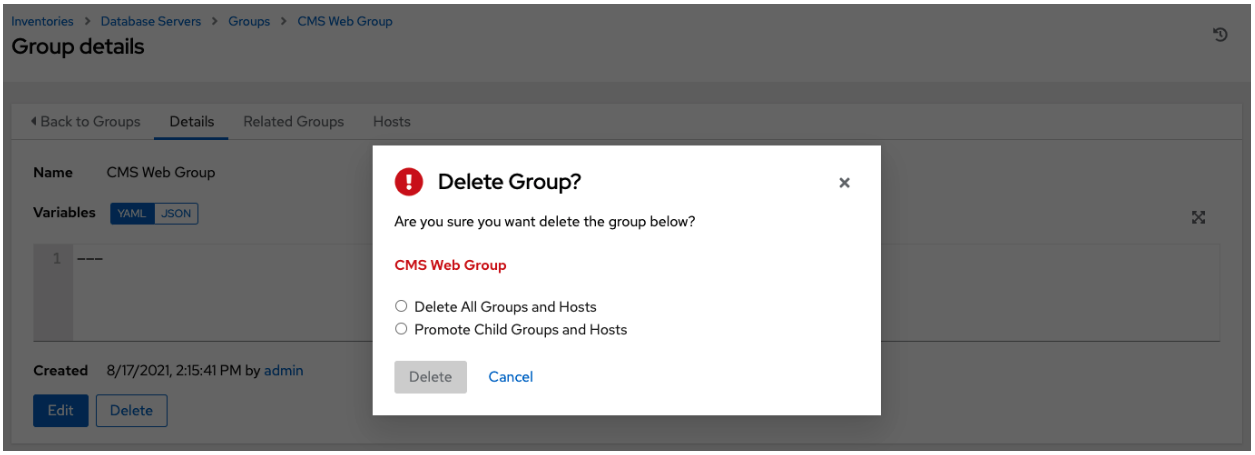
Once a group is selected, it displays as an entry in the list of groups associated with the group.

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.

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).


Inventories > Database Servers > Groups > CMS Web Group

Group details

Name CMS Web Group

Variables YAML JSON

```
1 ---
```

Created 8/17/2021, 2:15:41 PM by admin

Delete Group?

Are you sure you want delete the group below?

CMS Web Group

○ Delete All Groups and Hosts
○ Promote Child Groups and Hosts

18.3.3 Add hosts

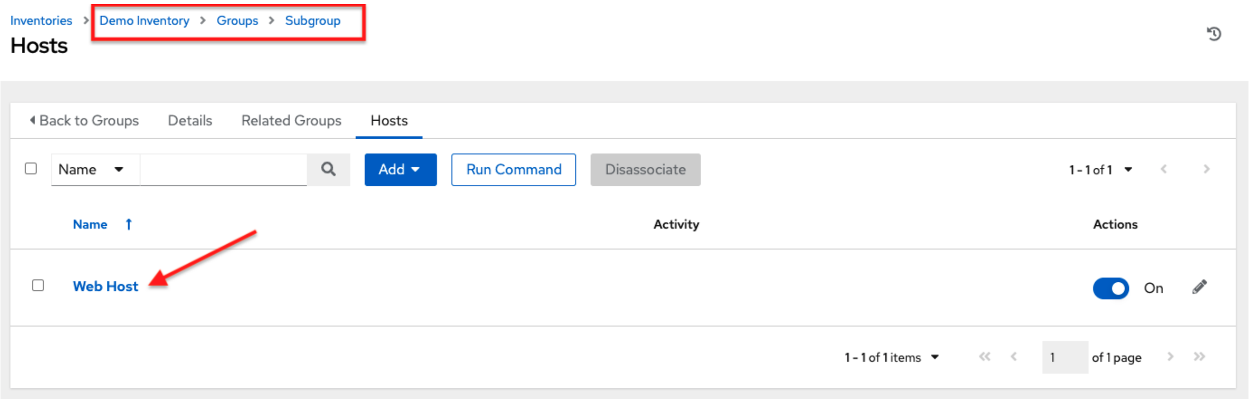
You can configure hosts for the inventory as well as for groups and groups within groups. To configure hosts:

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 ● 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.

5. When done, click **Save**.

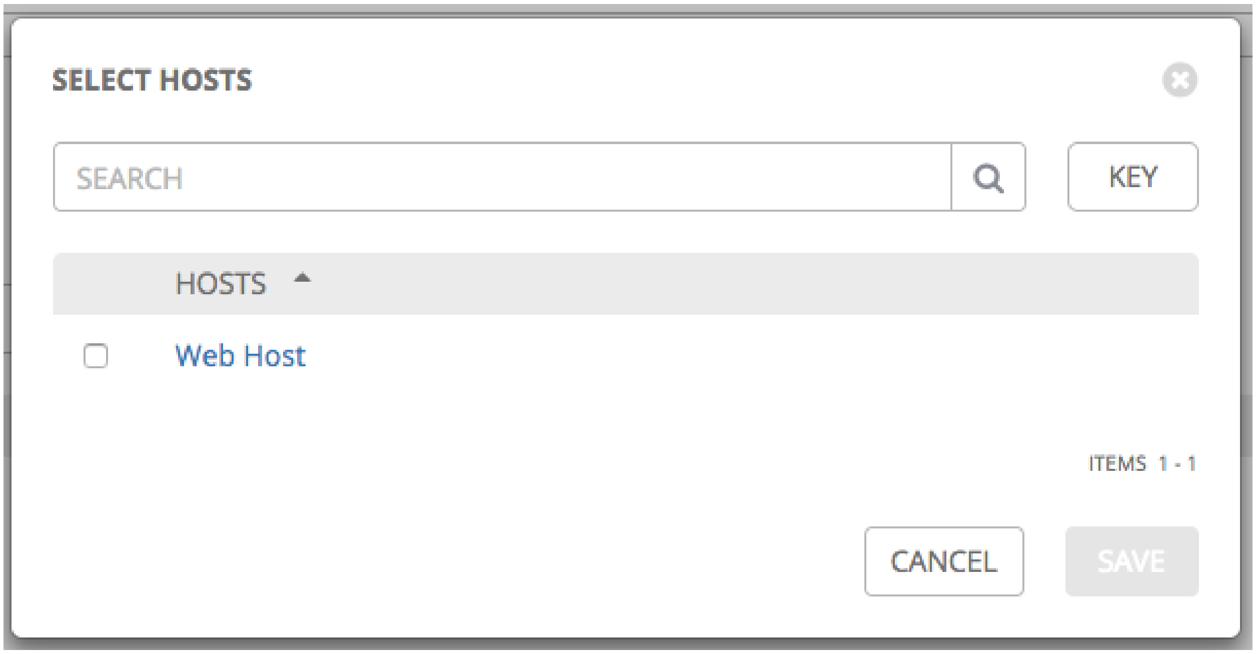
The **Create Host** window closes and the newly created host displays as an entry in the list of hosts associated with the group that it was created for.


Inventories > Demo Inventory > Groups > Subgroup

Hosts

	Name ↑	Activity	Actions
□	Web Host		On

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



SELECT HOSTS

SEARCH

KEY

HOSTS

□ Web Host

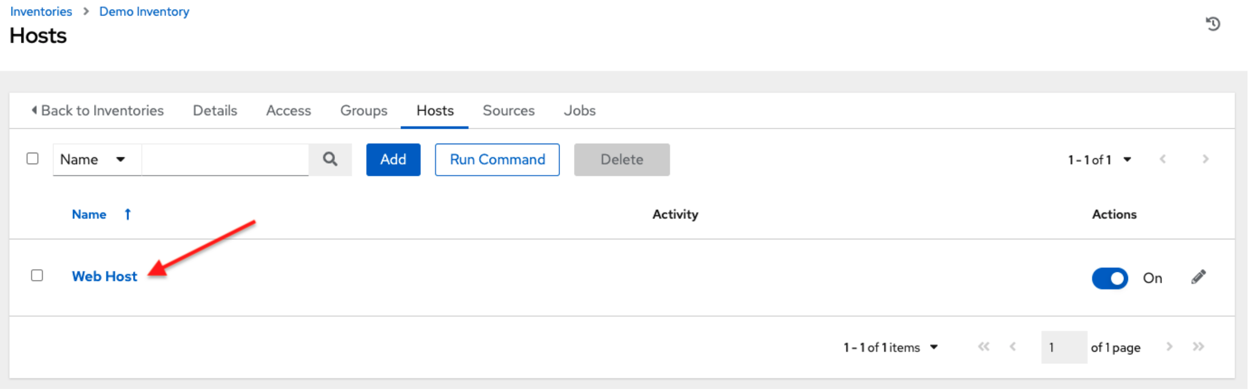
ITEMS 1 - 1

CANCEL SAVE

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.



Inventories > Demo Inventory

Hosts

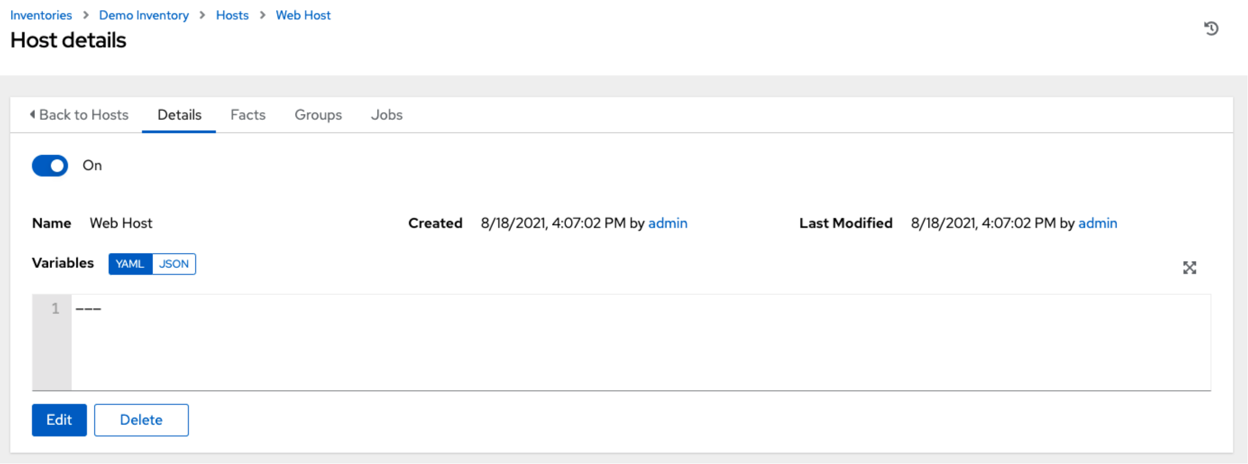
Back to Inventories | Details | Access | Groups | Hosts | Sources | Jobs

□ Name | Add | Run Command | Delete | 1 - 1 of 1

Name	Activity	Actions
□ Web Host		On

1 - 1 of 1 items | 1 | of 1 page

This opens the Details tab of the selected host.



Inventories > Demo Inventory > Hosts > Web Host

Host details

On

Name Web Host

Created 8/18/2021, 4:07:02 PM by admin

Last Modified 8/18/2021, 4:07:02 PM by admin

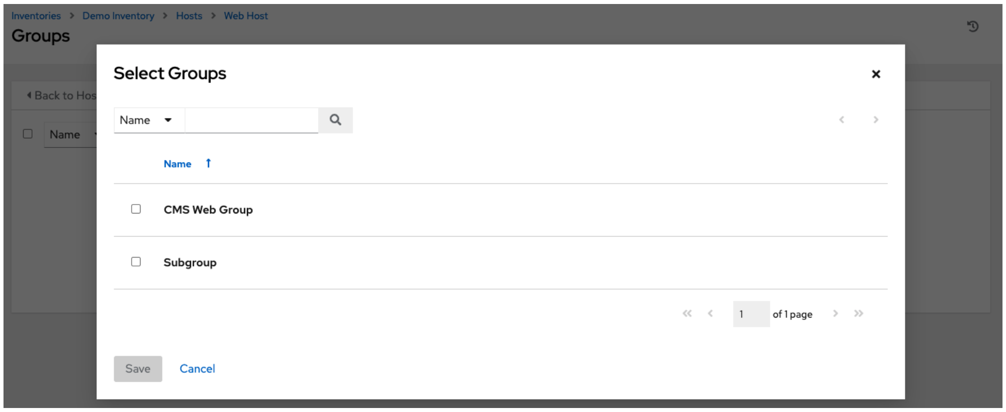
Variables YAML JSON

```
---
```

7. Click the **Groups** tab to configure groups for the host.

   a. Click the **Add** button to associate the host with an existing group.

      Available groups appear in a separate selection window.



Inventories > Demo Inventory > Hosts > Web Host

Groups

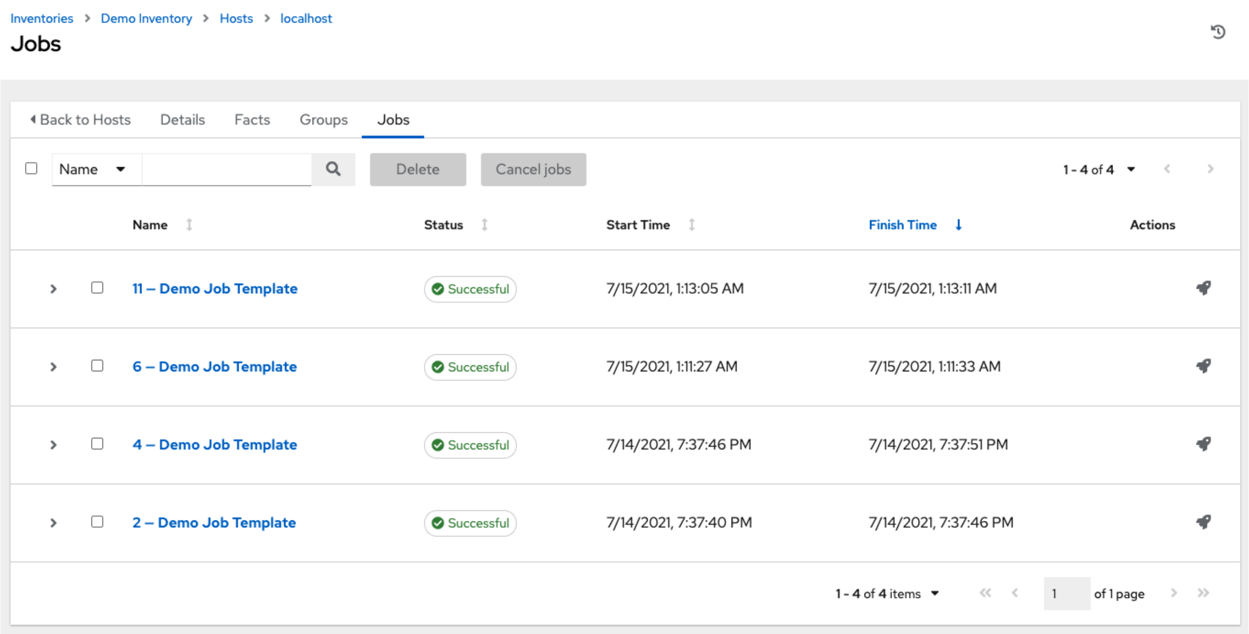
Select Groups

	Name ↑
□	CMS Web Group
□	Subgroup

   b. 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.

8. 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.



Inventories > Demo Inventory > Hosts > localhost

Jobs

	Name	Status	Start Time	Finish Time	Actions
□	11 – Demo Job Template	Successful	7/15/2021, 1:13:05 AM	7/15/2021, 1:13:11 AM	
□	6 – Demo Job Template	Successful	7/15/2021, 1:11:27 AM	7/15/2021, 1:11:33 AM	
□	4 – Demo Job Template	Successful	7/14/2021, 7:37:46 PM	7/14/2021, 7:37:51 PM	
□	2 – Demo Job Template	Successful	7/14/2021, 7:37:40 PM	7/14/2021, 7:37:46 PM	

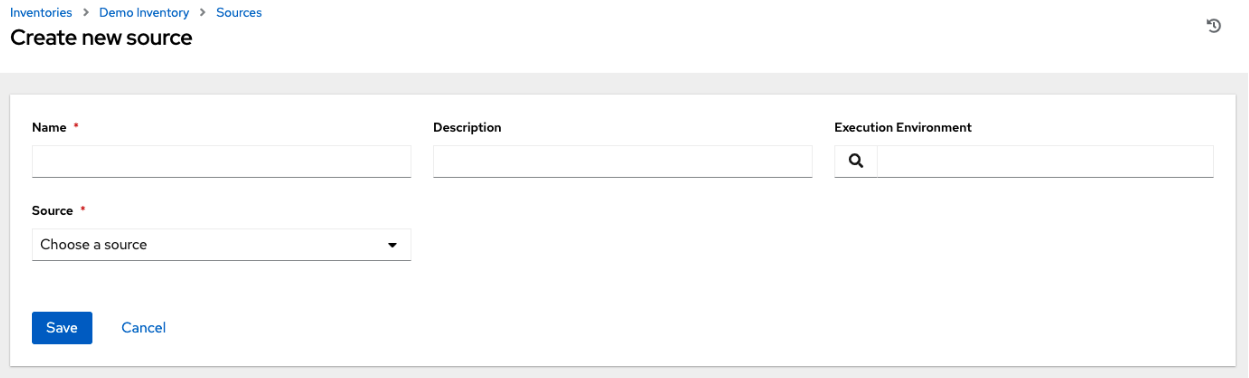
1 - 4 of 4 items 1 of 1 page

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.



Inventories > Demo Inventory > Sources

Create new source

Name *

Description

Execution Environment

Source *

Choose a source

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

- **Name**: Required
- **Description**: Enter an arbitrary description as appropriate (optional)
- **Execution Environment**: Optionally 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.

4. 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.

5. Select the appropriate level of output on any inventory source's update jobs from the **Verbosity** drop-down menu.
6. Use the **Host Filter** field to specify only matching host names to be imported into the controller.
7. 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)`.
8. 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.

9. 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).

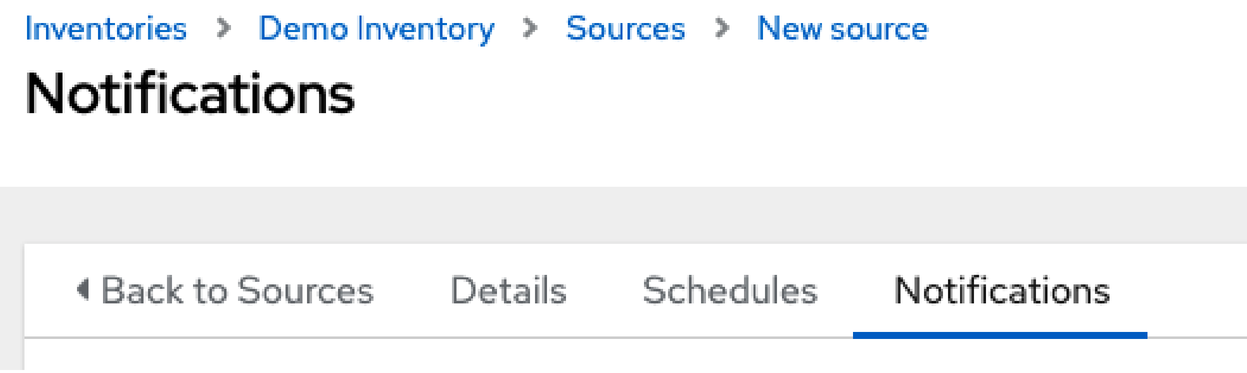
10. Review your entries and selections and click **Save** when done. This allows you to configure additional details, such as schedules and notifications.

11. To configure schedules associated with this inventory source, click the **Schedules** tab.

 a. If schedules are already set up; review, edit, or enable/disable your schedule preferences.

 b. 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.



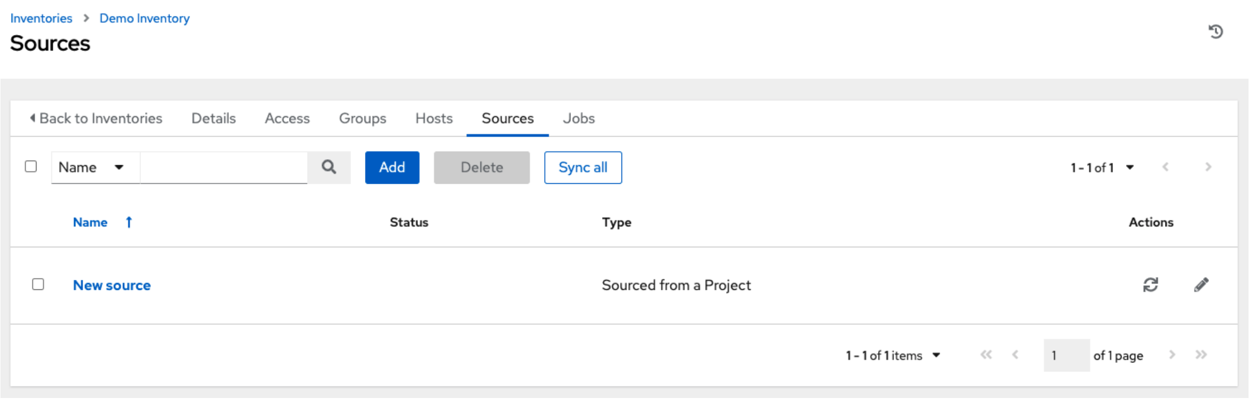
12. To configure notifications for the source, click the **Notifications** tab.

 a. 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*.

 b. if notifications have not been set up, refer to *Notifications* for more information.

13. 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.



Name	Status	Type	Actions
New source		Sourced from a Project	

Inventory Sources

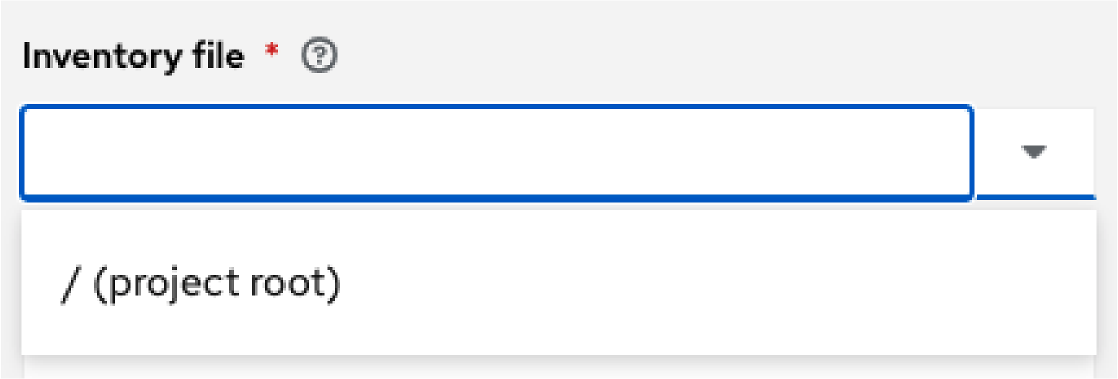
Choose a source which matches the inventory type against which a host can be entered:

- *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*

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 🔍 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.

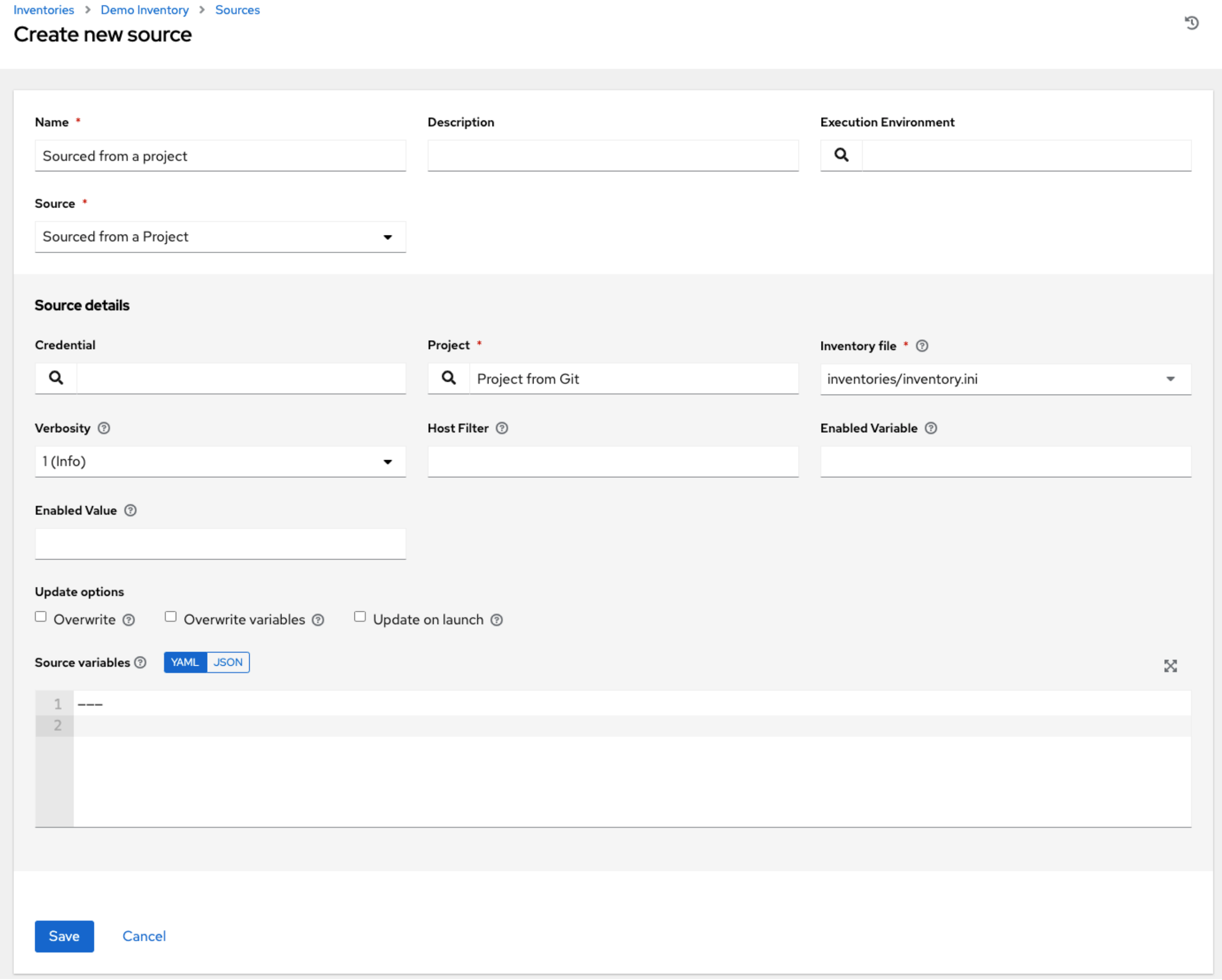


Inventory file *

/ (project root)

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. 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.



Inventories > Demo Inventory > Sources

Create new source

Name *: Sourced from a project

Description:

Execution Environment:

Source *: Sourced from a Project

Source details

Credential:

Project *: Project from Git

Inventory file *: inventories/inventory.ini

Verbosity: 1 (Info)

Host Filter:

Enabled Variable:

Enabled Value:

Update options

□ Overwrite □ Overwrite variables □ Update on launch

Source variables: YAML JSON

```
---
```

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.

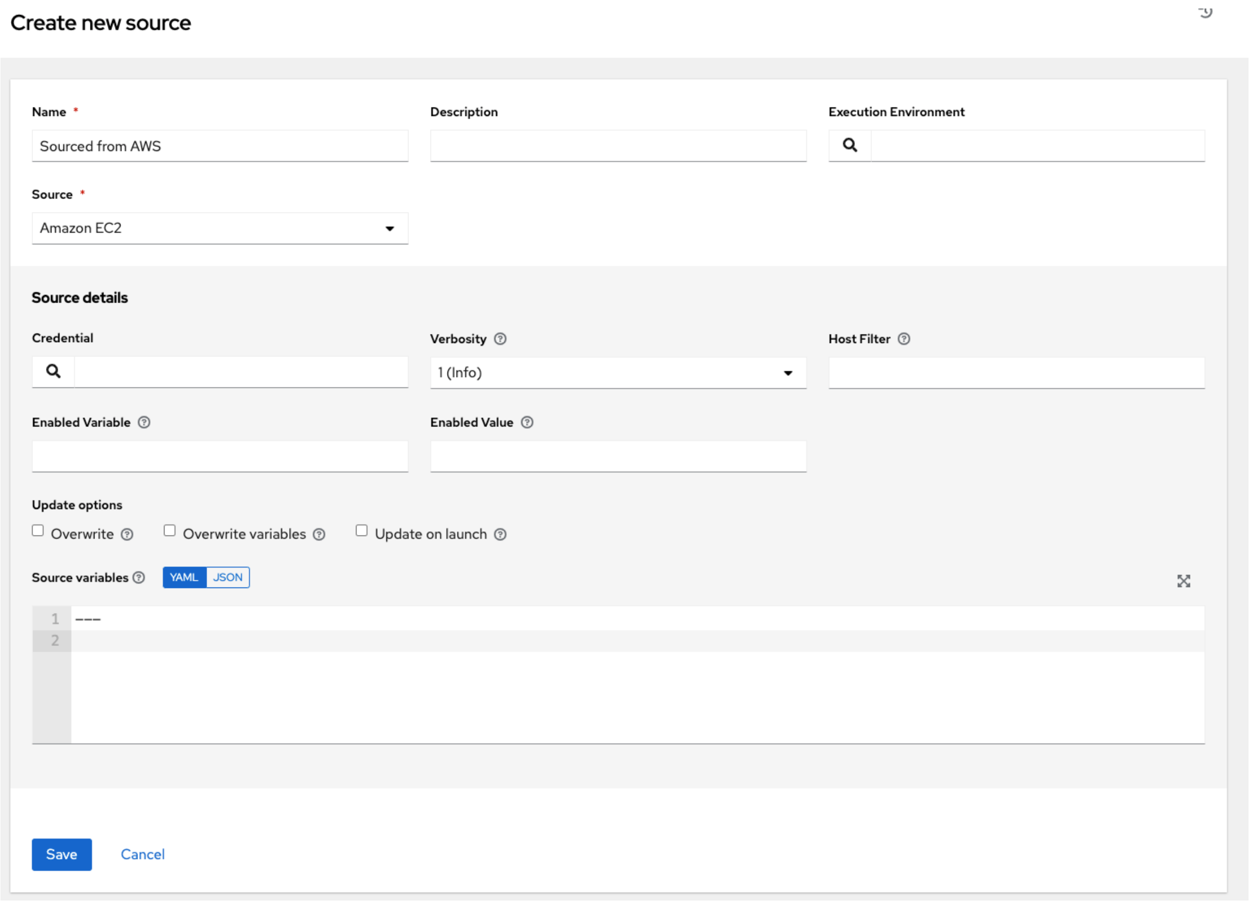
Amazon Web Services EC2

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.

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 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.



Create new source

Name * Sourced from AWS

Description

Execution Environment

Source * Amazon EC2

Source details

Credential

Verbosity 1 (Info)

Host Filter

Enabled Variable

Enabled Value

Update options

□ Overwrite □ Overwrite variables □ Update on launch

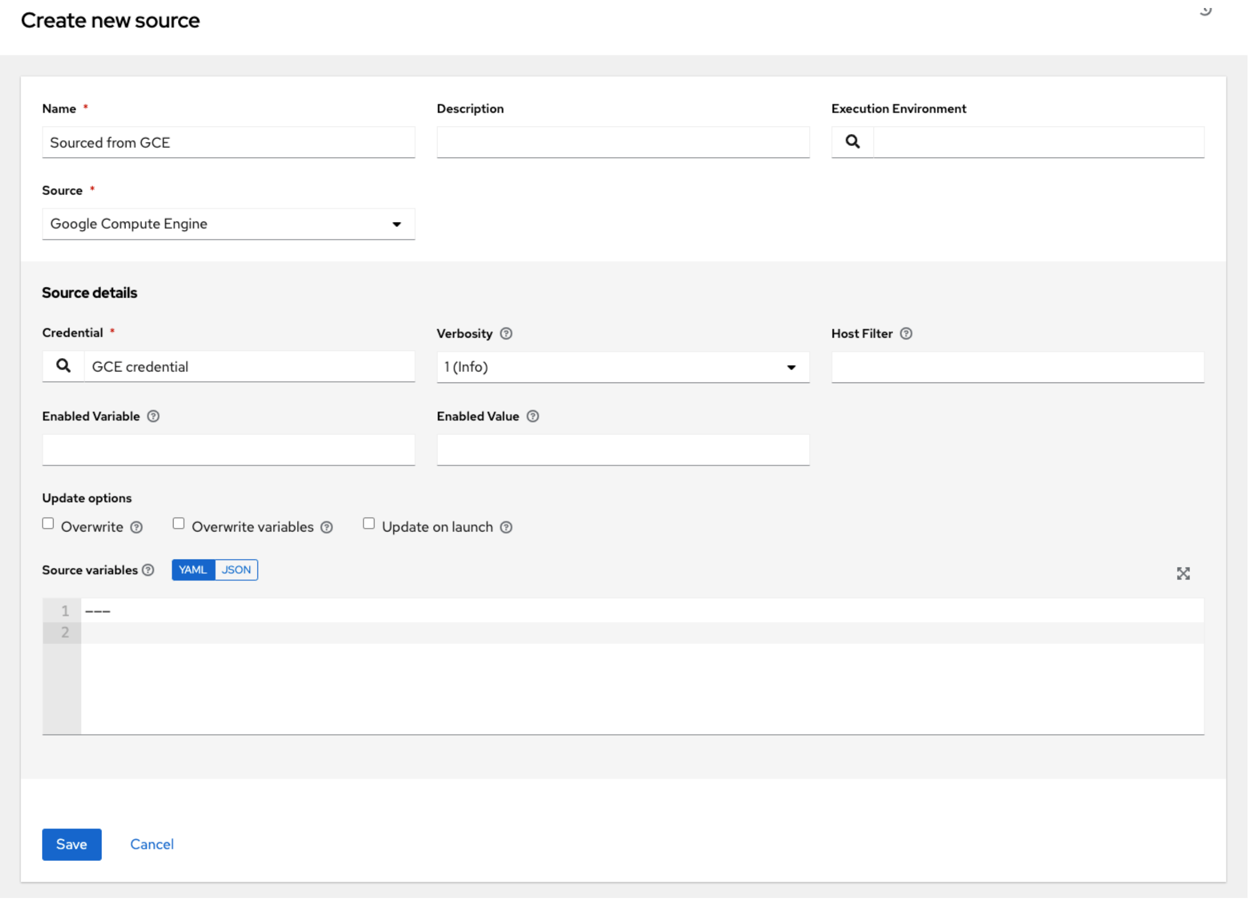
Source variables YAML JSON

```
---
```

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.

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*.



Create new source

Name *: Sourced from GCE

Description:

Execution Environment:

Source *: Google Compute Engine

Source details

Credential *: GCE credential

Verbosity: 1 (Info)

Host Filter:

Enabled Variable:

Enabled Value:

Update options

□ Overwrite □ Overwrite variables □ Update on launch

Source variables: YAML | JSON

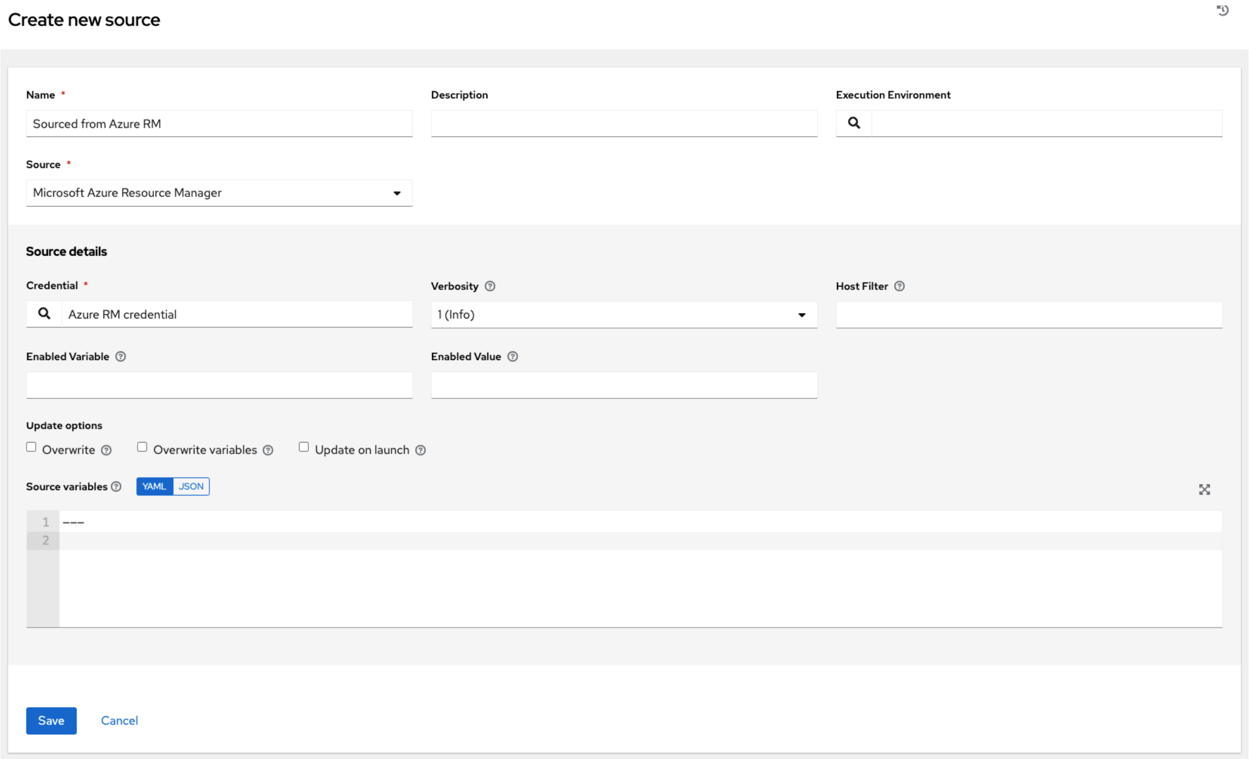
```
---
```

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 `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.

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.


Create new source

Name * Sourced from Azure RM

Description

Execution Environment

Source * Microsoft Azure Resource Manager

Source details

Credential * Azure RM credential

Verbosity 1 (Info)

Host Filter

Enabled Variable

Enabled Value

Update options

□ Overwrite □ Overwrite variables □ Update on launch

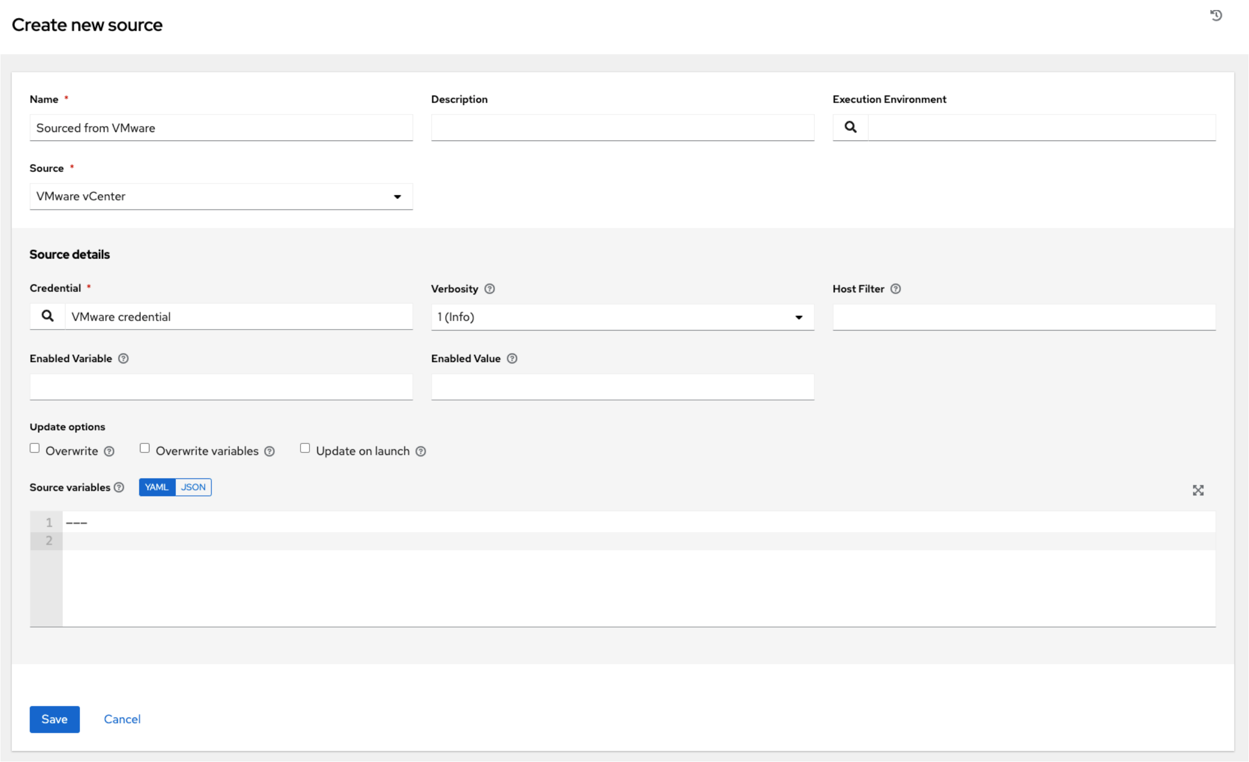
Source variables YAML JSON

```
---
```

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.



Create new source

Name * Sourced from VMware

Description

Execution Environment

Source * VMware vCenter

Source details

Credential * VMware credential

Verbosity 1 (Info)

Host Filter

Enabled Variable

Enabled Value

Update options

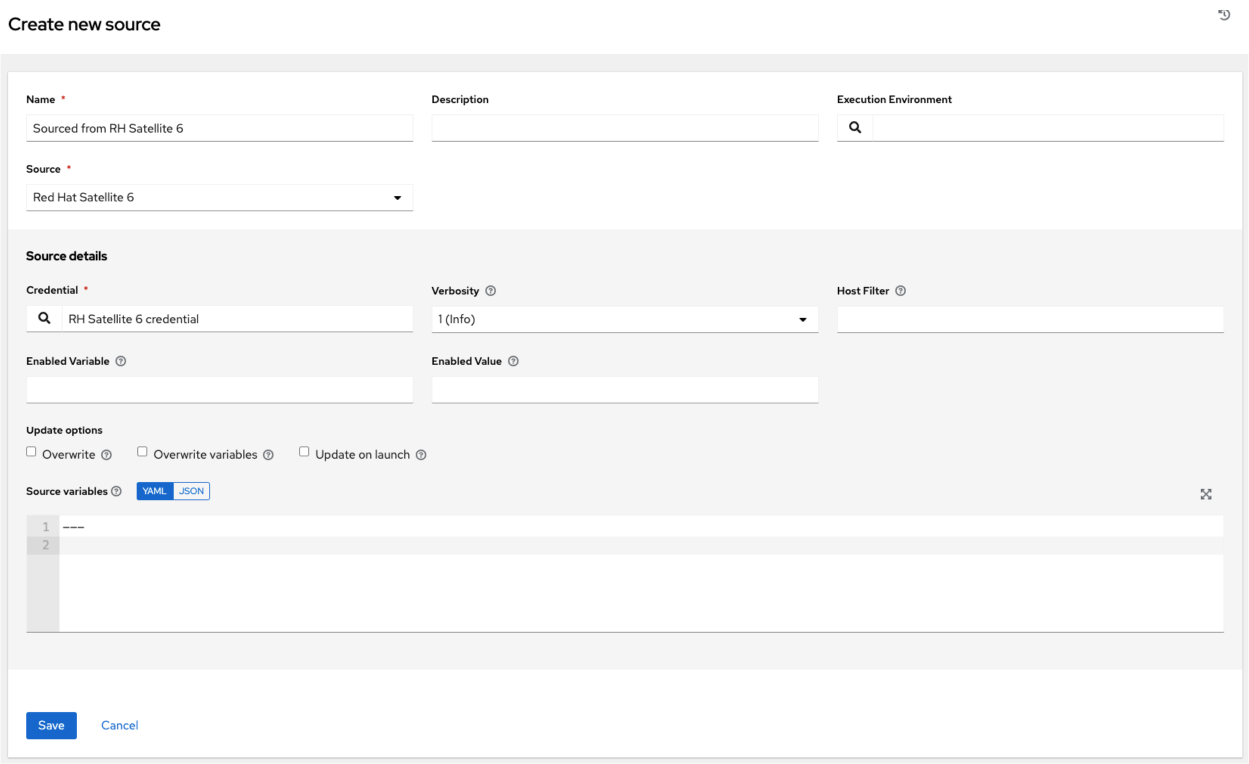
□ Overwrite □ Overwrite variables □ Update on launch

Source variables YAML JSON

```
---
```

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.



Create new source

Name * Sourced from RH Satellite 6

Description

Execution Environment

Source * Red Hat Satellite 6

Source details

Credential * RH Satellite 6 credential

Verbosity 1 (Info)

Host Filter

Enabled Variable

Enabled Value

Update options

□ Overwrite □ Overwrite variables □ Update on launch

Source variables

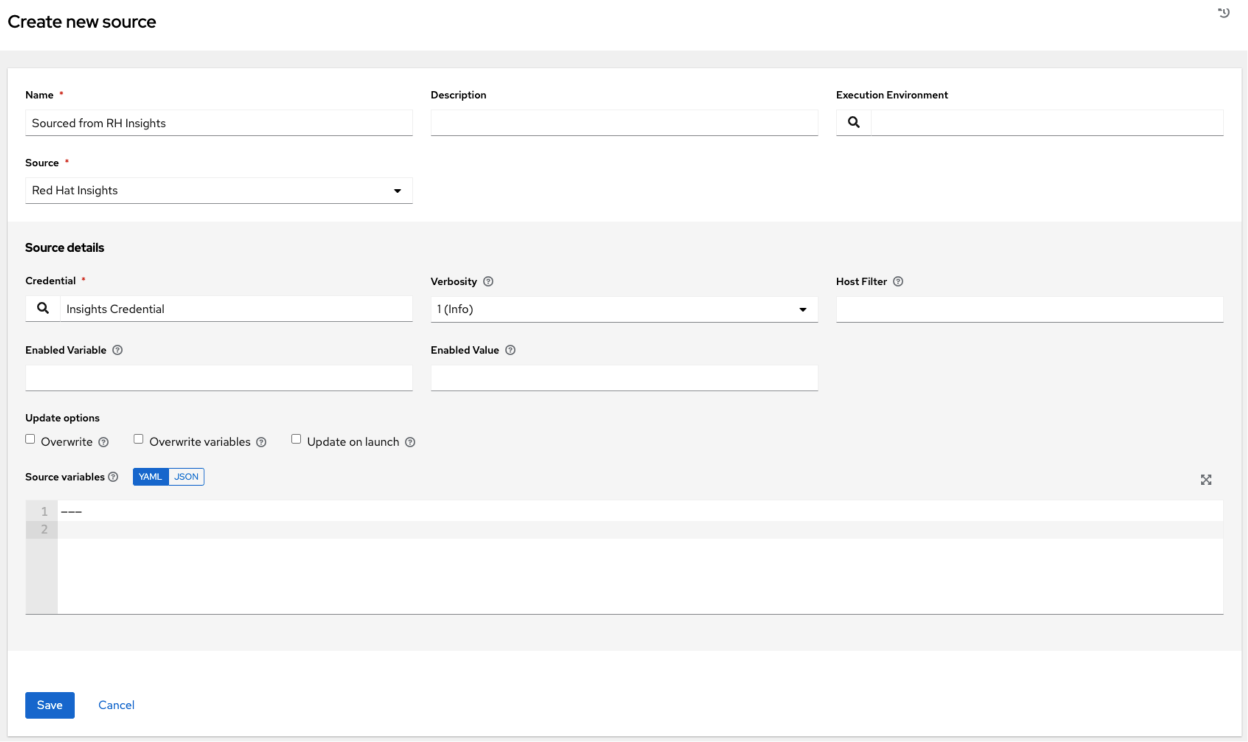
```
---
```

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.

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.



Create new source

Name * Sourced from RH Insights

Description

Execution Environment

Source * Red Hat Insights

Source details

Credential * Insights Credential

Verbosity 1 (Info)

Host Filter

Enabled Variable

Enabled Value

Update options

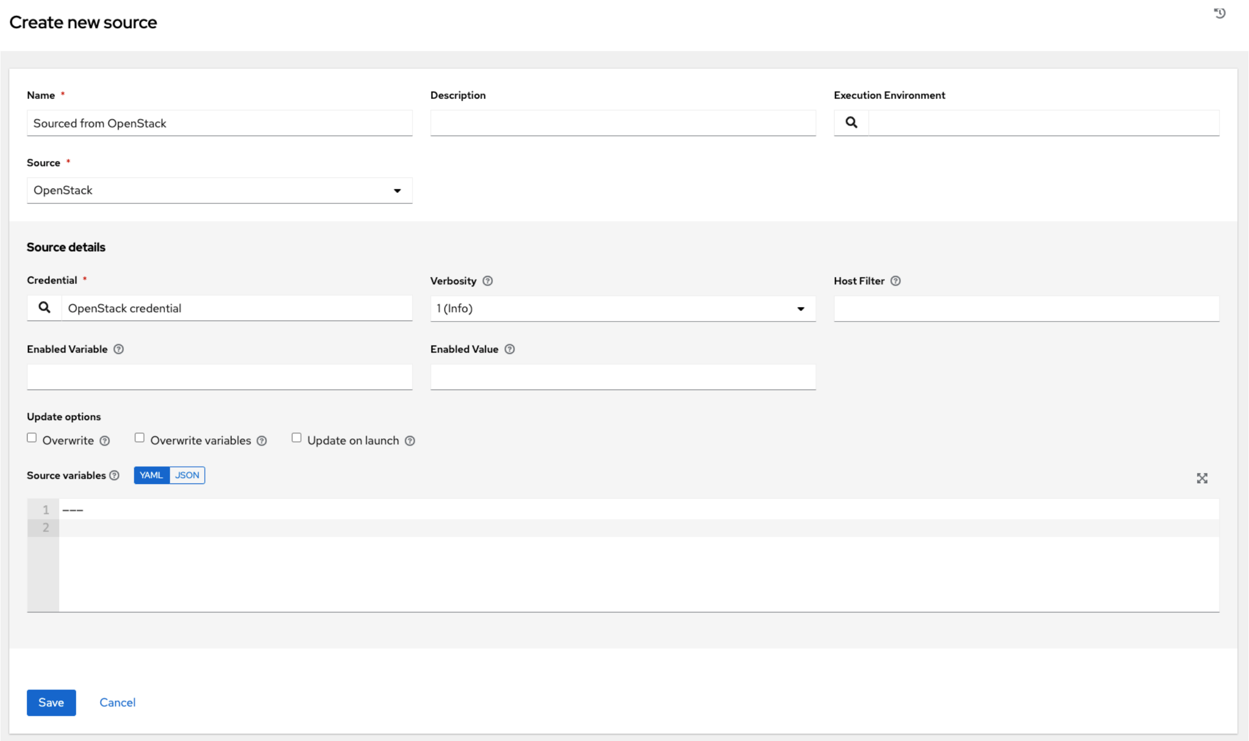
☐ Overwrite ☐ Overwrite variables ☐ Update on launch

Source variables YAML JSON

```
---
```

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.



Create new source

Name * Sourced from OpenStack

Description

Execution Environment

Source * OpenStack

Source details

Credential * OpenStack credential

Verbosity 1 (Info)

Host Filter

Enabled Variable

Enabled Value

Update options

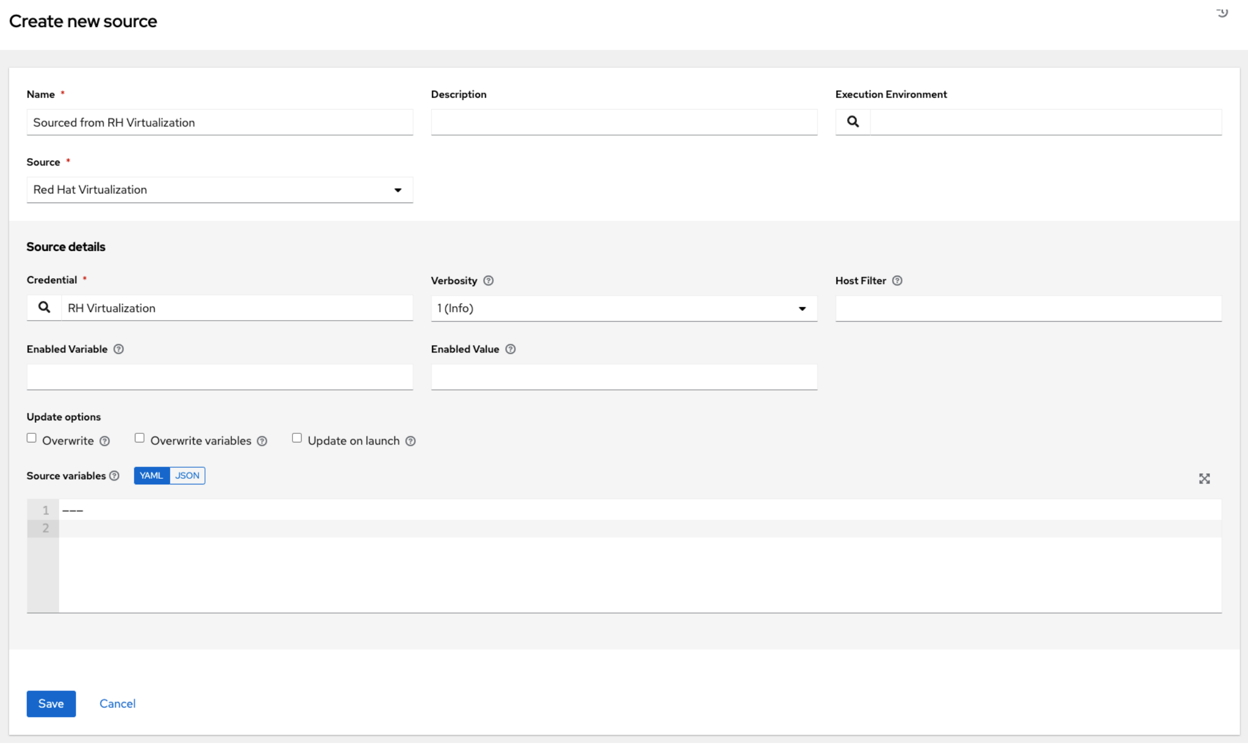
□ Overwrite □ Overwrite variables □ Update on launch

Source variables YAML JSON

```
---
```

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.



Create new source

Name * Sourced from RH Virtualization

Description

Execution Environment

Source * Red Hat Virtualization

Source details

Credential * RH Virtualization

Verbosity 1 (Info)

Host Filter

Enabled Variable

Enabled Value

Update options

□ Overwrite □ Overwrite variables □ Update on launch

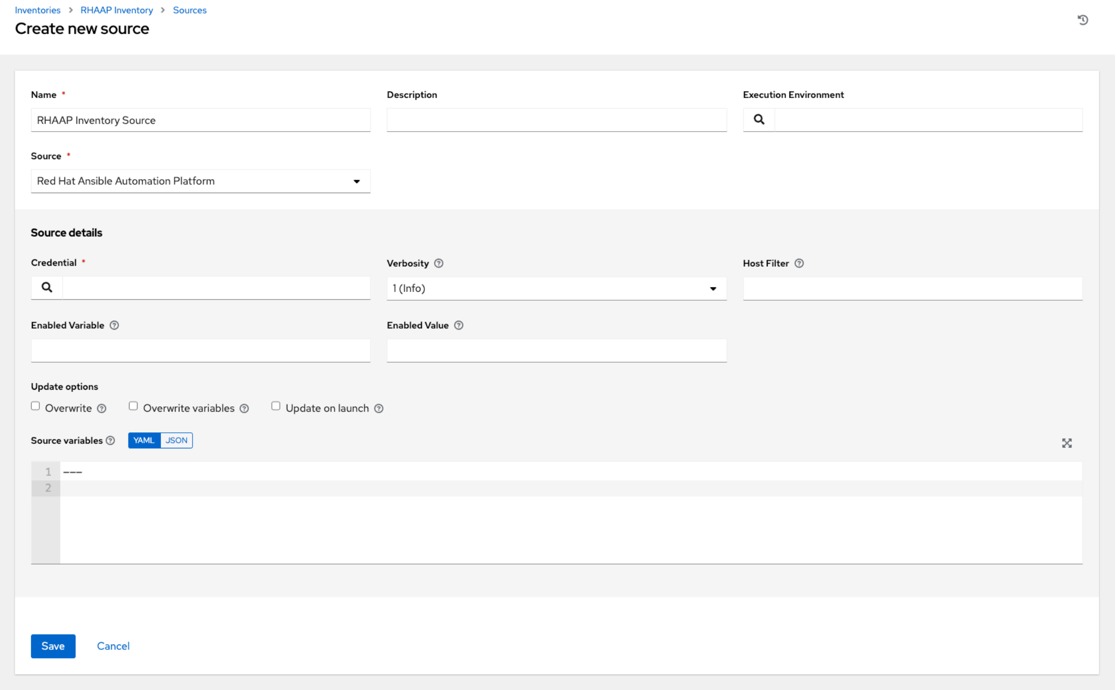
Source variables YAML JSON

```
---
```

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.

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*.



Inventories > RHAAP Inventory > Sources

Create new source

Name * RHAAP Inventory Source

Description

Execution Environment

Source * Red Hat Ansible Automation Platform

Source details

Credential *

Verbosity 1 (Info)

Host Filter

Enabled Variable

Enabled Value

Update options

□ Overwrite □ Overwrite variables □ Update on launch

Source variables YAML JSON

```
---
```

4. 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).

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_
```

(continues on next page)

(continued from previous page)

```
_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.

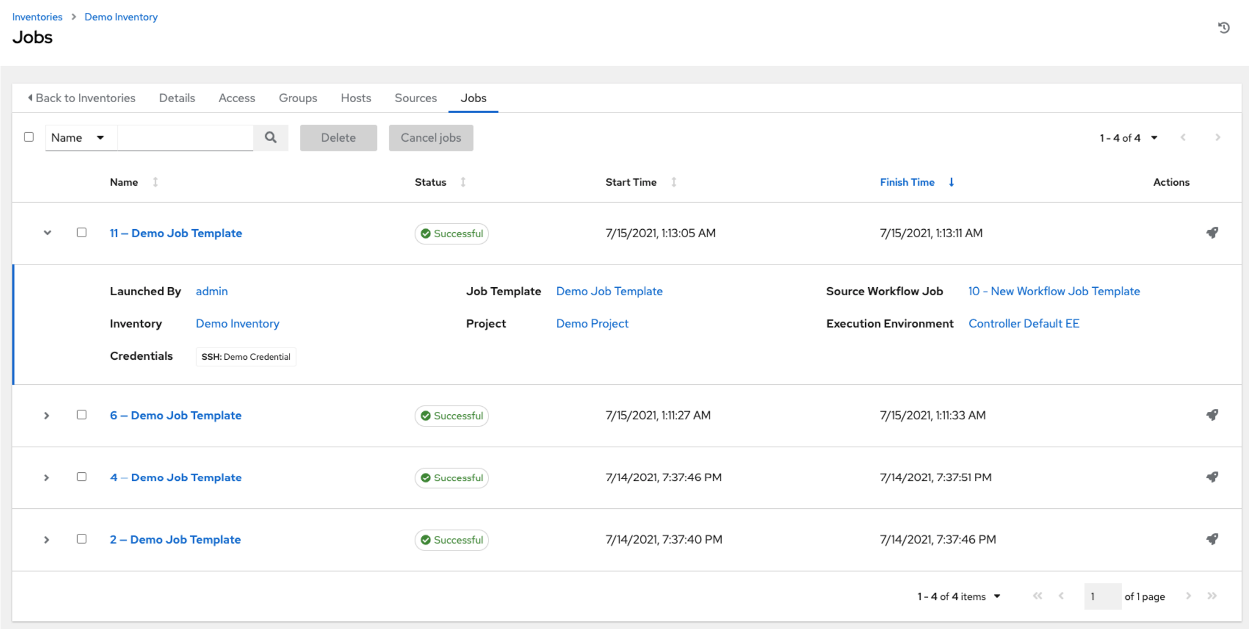
```
$ ansible-inventory -i ./my_scripts/_11__inventory_script_upperorder --list --export
```

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.



Inventories > Demo Inventory

Jobs

◂ Back to Inventories | Details | Access | Groups | Hosts | Sources | Jobs

□ Name ▾ | Delete | Cancel jobs | 1 - 4 of 4 ▾

		Name	Status	Start Time	Finish Time	Actions
˅	□	11 – Demo Job Template	Successful	7/15/2021, 1:13:05 AM	7/15/2021, 1:13:11 AM	🚀

Launched By	admin	Job Template	Demo Job Template	Source Workflow Job	10 - New Workflow Job Template
Inventory	Demo Inventory	Project	Demo Project	Execution Environment	Controller Default EE
Credentials	SSH: Demo Credential				

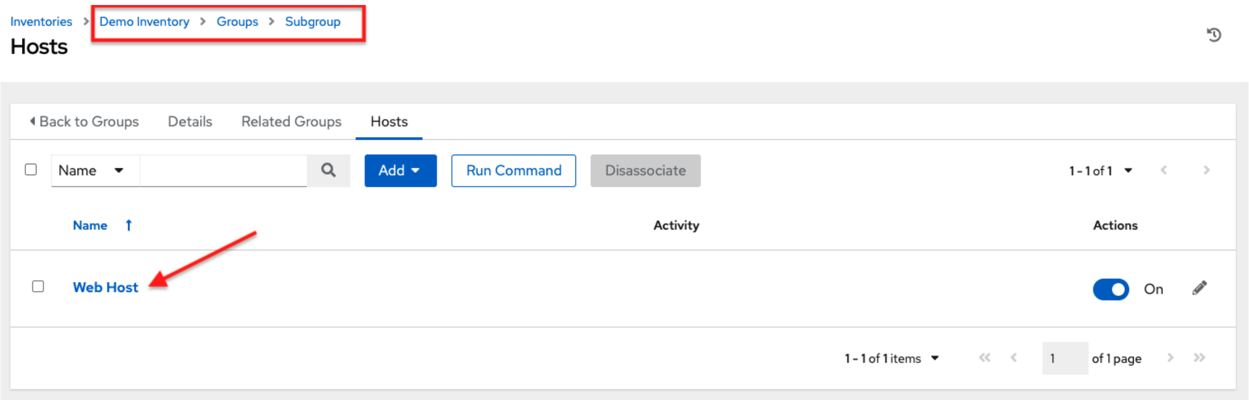
		Name	Status	Start Time	Finish Time	Actions
›	□	6 – Demo Job Template	Successful	7/15/2021, 1:11:27 AM	7/15/2021, 1:11:33 AM	🚀
›	□	4 – Demo Job Template	Successful	7/14/2021, 7:37:46 PM	7/14/2021, 7:37:51 PM	🚀
›	□	2 – Demo Job Template	Successful	7/14/2021, 7:37:40 PM	7/14/2021, 7:37:46 PM	🚀

1 - 4 of 4 items ▾ | 1 of 1 page

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.



Inventories > Demo Inventory > Groups > Subgroup

Hosts

◂ Back to Groups | Details | Related Groups | Hosts

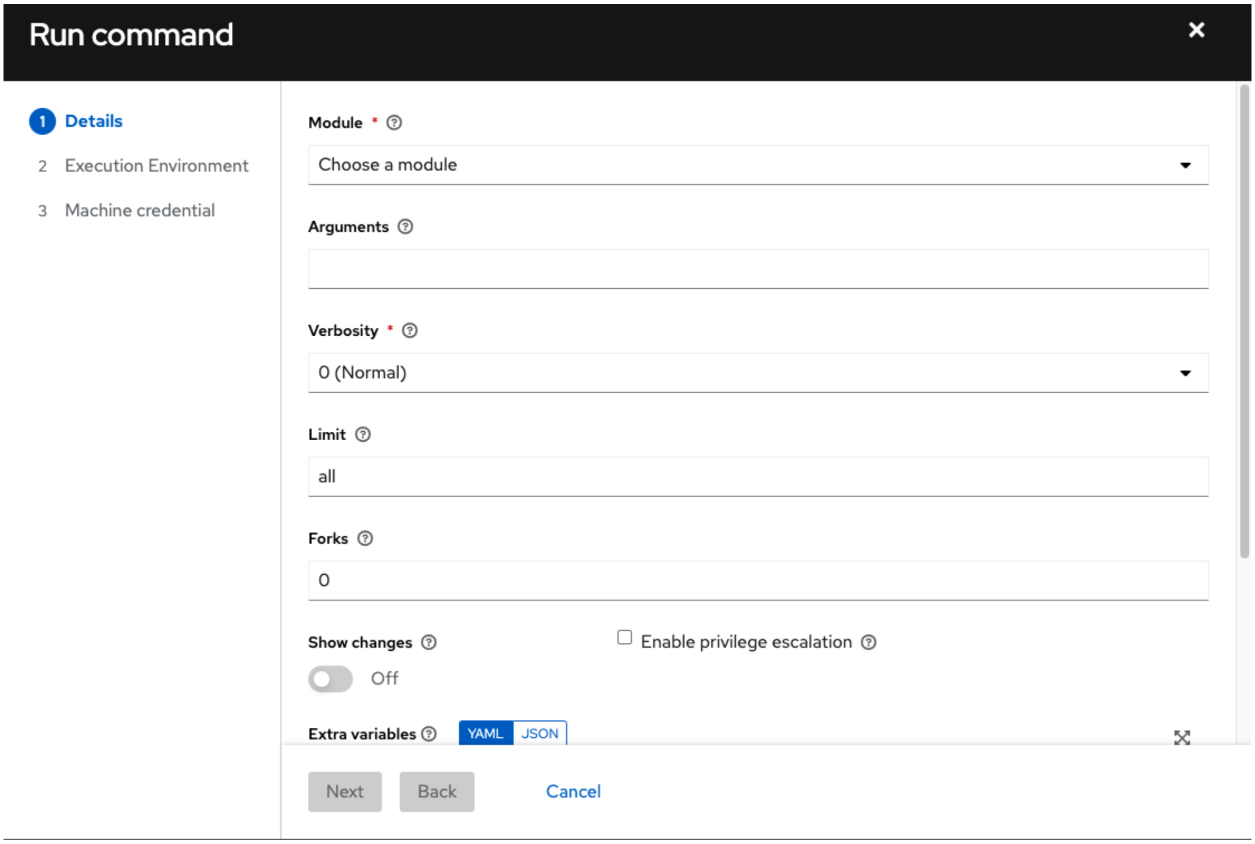
□ Name ▾ | Add ▾ | Run Command | Disassociate | 1 - 1 of 1 ▾

	Name	Activity	Actions
□	Web Host		On ✎

1 - 1 of 1 items ▾ | 1 of 1 page

2. Click the **Run Command** button.

The Run command window opens.



Run command

1 Details
2 Execution Environment
3 Machine credential

Module *: Choose a module

Arguments:

Verbosity *: 0 (Normal)

Limit: all

Forks: 0

Show changes: Off

□ Enable privilege escalation

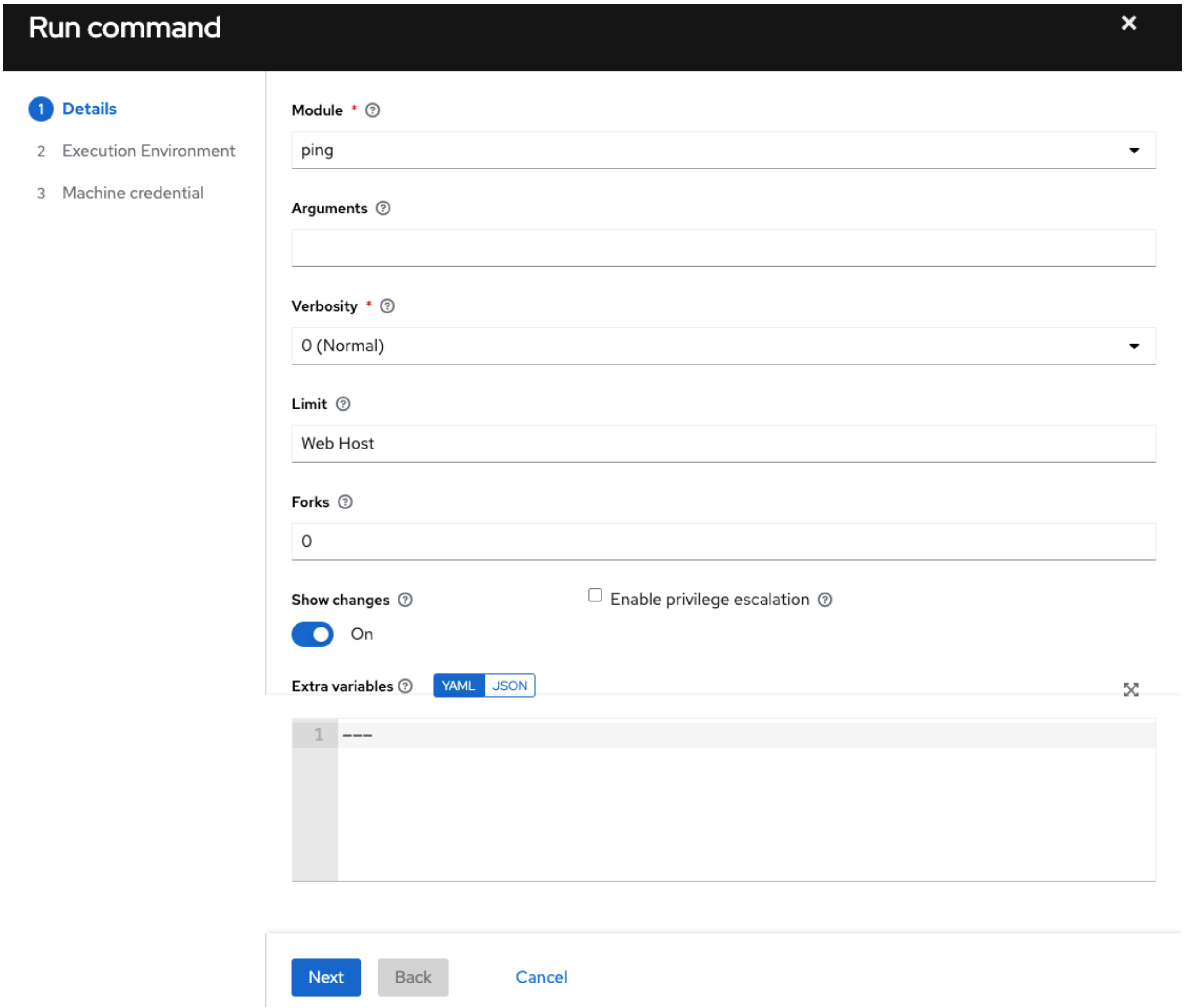
Extra variables: YAML JSON

3. 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.



Module * ping

Arguments

Verbosity * 0 (Normal)

Limit Web Host

Forks 0

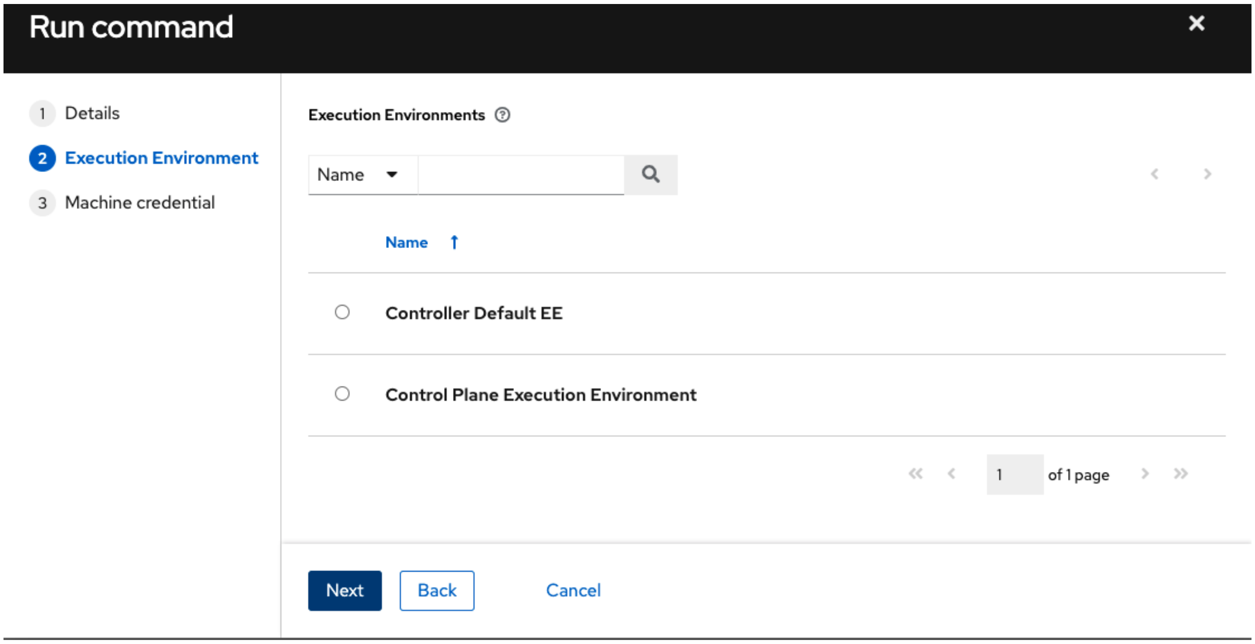
Show changes On

□ Enable privilege escalation

Extra variables

```
---
```

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



Run command

1 Details
2 Execution Environment
3 Machine credential

Execution Environments

Name

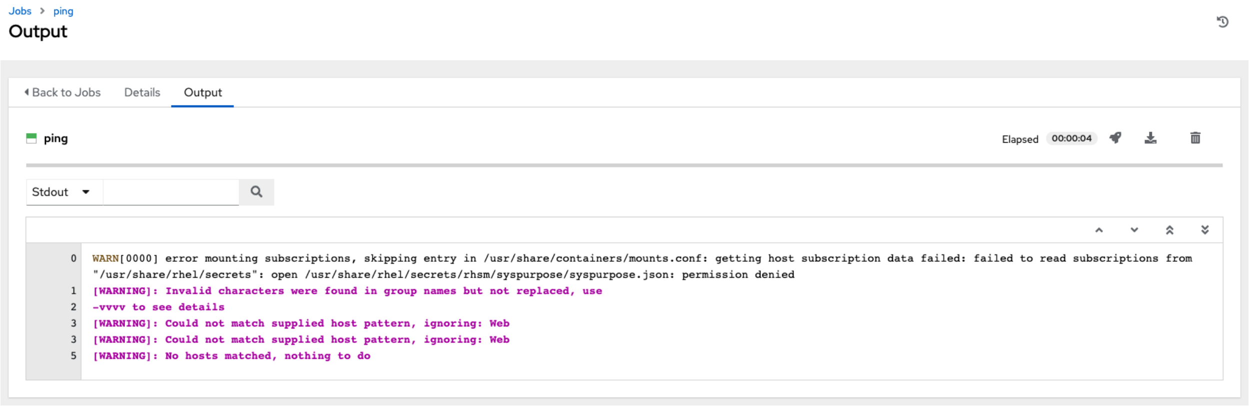
○ Controller Default EE

○ Control Plane Execution Environment

1 of 1 page

5. 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.



Jobs > ping

Output

ping

Elapsed 00:00:04

Stdout

```
WARN[0000] error mounting subscriptions, skipping entry in /usr/share/containers/mounts.conf: getting host subscription data failed: failed to read subscriptions from
"/usr/share/rhel/secrets": open /usr/share/rhel/secrets/rhsm/syspurpose/syspurpose.json: permission denied
[WARNING]: Invalid characters were found in group names but not replaced, use
-vvvv to see details
[WARNING]: Could not match supplied host pattern, ignoring: Web
[WARNING]: Could not match supplied host pattern, ignoring: Web
[WARNING]: No hosts matched, nothing to do
```

CHAPTER

NINETEEN

SUPPORTED INVENTORY PLUGIN TEMPLATES

Upon upgrade to 4.x, existing configurations will be migrated to the new format that will produce a backwards compatible inventory output. Use the templates below to help aid in migrating your inventories to the new style inventory plugin output.

19.1 Amazon Web Services EC2

```
compose:
  ansible_host: public_ip_address
  ec2_account_id: owner_id
  ec2_ami_launch_index: ami_launch_index | string
  ec2_architecture: architecture
  ec2_block_devices: dict(block_device_mappings | map(attribute='device_name') | list␣
↪| zip(block_device_mappings | map(attribute='ebs.volume_id') | list))
  ec2_client_token: client_token
  ec2_dns_name: public_dns_name
  ec2_ebs_optimized: ebs_optimized
  ec2_eventsSet: events | default("")
  ec2_group_name: placement.group_name
  ec2_hypervisor: hypervisor
  ec2_id: instance_id
  ec2_image_id: image_id
  ec2_instance_profile: iam_instance_profile | default("")
  ec2_instance_type: instance_type
  ec2_ip_address: public_ip_address
  ec2_kernel: kernel_id | default("")
```

(continues on next page)

(continued from previous page)

```
  ec2_key_name: key_name
  ec2_launch_time: launch_time | regex_replace(" ", "T") | regex_replace("(\+)(\d\
↪d):(\d)(\d)$", ".\g<2>\g<3>Z")
  ec2_monitored: monitoring.state in ['enabled', 'pending']
  ec2_monitoring_state: monitoring.state
  ec2_persistent: persistent | default(false)
  ec2_placement: placement.availability_zone
  ec2_platform: platform | default("")
  ec2_private_dns_name: private_dns_name
  ec2_private_ip_address: private_ip_address
  ec2_public_dns_name: public_dns_name
  ec2_ramdisk: ramdisk_id | default("")
  ec2_reason: state_transition_reason
  ec2_region: placement.region
  ec2_requester_id: requester_id | default("")
  ec2_root_device_name: root_device_name
  ec2_root_device_type: root_device_type
  ec2_security_group_ids: security_groups | map(attribute='group_id') | list |  join(
↪',')
  ec2_security_group_names: security_groups | map(attribute='group_name') | list | ␣
↪join(',')
  ec2_sourceDestCheck: source_dest_check | default(false) | lower | string
  ec2_spot_instance_request_id: spot_instance_request_id | default("")
  ec2_state: state.name
  ec2_state_code: state.code
  ec2_state_reason: state_reason.message if state_reason is defined else ""
  ec2_subnet_id: subnet_id | default("")
  ec2_tag_Name: tags.Name
  ec2_virtualization_type: virtualization_type
  ec2_vpc_id: vpc_id | default("")
filters:
  instance-state-name:
  - running
groups:
  ec2: true
hostnames:
  - network-interface.addresses.association.public-ip
  - dns-name
  - private-dns-name
keyed_groups:
  - key: image_id | regex_replace("[^A-Za-z0-9\_]", "_")
    parent_group: images
    prefix: ''
    separator: ''
  - key: placement.availability_zone
    parent_group: zones
    prefix: ''
    separator: ''
  - key: ec2_account_id | regex_replace("[^A-Za-z0-9\_]", "_")
    parent_group: accounts
    prefix: ''
    separator: ''
  - key: ec2_state | regex_replace("[^A-Za-z0-9\_]", "_")
    parent_group: instance_states
    prefix: instance_state
  - key: platform | default("undefined") | regex_replace("[^A-Za-z0-9\_]", "_")
    parent_group: platforms
```

(continues on next page)

(continued from previous page)

```
    prefix: platform
  - key: instance_type | regex_replace("[^A-Za-z0-9\_]", "_")
    parent_group: types
    prefix: type
  - key: key_name | regex_replace("[^A-Za-z0-9\_]", "_")
    parent_group: keys
    prefix: key
  - key: placement.region
    parent_group: regions
    prefix: ''
    separator: ''
  - key: security_groups | map(attribute="group_name") | map("regex_replace", "[^A-Za-
↪z0-9\_]", "_") | list
    parent_group: security_groups
    prefix: security_group
  - key: dict(tags.keys() | map("regex_replace", "[^A-Za-z0-9\_]", "_") | list |␣
↪zip(tags.values()
      | map("regex_replace", "[^A-Za-z0-9\_]", "_") | list))
    parent_group: tags
    prefix: tag
  - key: tags.keys() | map("regex_replace", "[^A-Za-z0-9\_]", "_") | list
    parent_group: tags
    prefix: tag
  - key: vpc_id | regex_replace("[^A-Za-z0-9\_]", "_")
    parent_group: vpcs
    prefix: vpc_id
  - key: placement.availability_zone
    parent_group: '{{ placement.region }}'
    prefix: ''
    separator: ''
plugin: amazon.aws.aws_ec2
use_contrib_script_compatible_sanitization: true
```

19.2 Google Compute Engine

```
auth_kind: serviceaccount
compose:
  ansible_ssh_host: networkInterfaces[0].accessConfigs[0].natIP |␣
↪default(networkInterfaces[0].networkIP)
  gce_description: description if description else None
  gce_id: id
  gce_image: image
  gce_machine_type: machineType
  gce_metadata: metadata.get("items", []) | items2dict(key_name="key", value_name=
↪"value")
  gce_name: name
  gce_network: networkInterfaces[0].network.name
  gce_private_ip: networkInterfaces[0].networkIP
  gce_public_ip: networkInterfaces[0].accessConfigs[0].natIP | default(None)
  gce_status: status
  gce_subnetwork: networkInterfaces[0].subnetwork.name
  gce_tags: tags.get("items", [])
  gce_zone: zone
hostnames:
```

(continues on next page)

(continued from previous page)

```
- name
- public_ip
- private_ip
keyed_groups:
- key: gce_subnetwork
  prefix: network
- key: gce_private_ip
  prefix: ''
  separator: ''
- key: gce_public_ip
  prefix: ''
  separator: ''
- key: machineType
  prefix: ''
  separator: ''
- key: zone
  prefix: ''
  separator: ''
- key: gce_tags
  prefix: tag
- key: status | lower
  prefix: status
- key: image
  prefix: ''
  separator: ''
plugin: google.cloud.gcp_compute
retrieve_image_info: true
use_contrib_script_compatible_sanitization: true
```

19.3 Microsoft Azure Resource Manager

```
conditional_groups:
  azure: true
default_host_filters: []
fail_on_template_errors: false
hostvar_expressions:
  computer_name: name
  private_ip: private_ipv4_addresses[0] if private_ipv4_addresses else None
  provisioning_state: provisioning_state | title
  public_ip: public_ipv4_addresses[0] if public_ipv4_addresses else None
  public_ip_id: public_ip_id if public_ip_id is defined else None
  public_ip_name: public_ip_name if public_ip_name is defined else None
  tags: tags if tags else None
  type: resource_type
keyed_groups:
- key: location
  prefix: ''
  separator: ''
- key: tags.keys() | list if tags else []
  prefix: ''
  separator: ''
- key: security_group
  prefix: ''
  separator: ''
```

(continues on next page)

(continued from previous page)

```
- key: resource_group
  prefix: ''
  separator: ''
- key: os_disk.operating_system_type
  prefix: ''
  separator: ''
- key: dict(tags.keys() | map("regex_replace", "^(.*)$", "\1_") | list | zip(tags.
↪values() | list)) if tags else []
  prefix: ''
  separator: ''
plain_host_names: true
plugin: azure.azcollection.azure_rm
use_contrib_script_compatible_sanitization: true
```

19.4 VMware vCenter

```
compose:
  ansible_host: guest.ipAddress
  ansible_ssh_host: guest.ipAddress
  ansible_uuid: 99999999 | random | to_uuid
  availablefield: availableField
  configissue: configIssue
  configstatus: configStatus
  customvalue: customValue
  effectiverole: effectiveRole
  guestheartbeatstatus: guestHeartbeatStatus
  layoutex: layoutEx
  overallstatus: overallStatus
  parentvapp: parentVApp
  recenttask: recentTask
  resourcepool: resourcePool
  rootsnapshot: rootSnapshot
  triggeredalarmstate: triggeredAlarmState
filters:
- runtime.powerState == "poweredOn"
keyed_groups:
- key: config.guestId
  prefix: ''
  separator: ''
- key: '"templates" if config.template else "guests"'
  prefix: ''
  separator: ''
plugin: community.vmware.vmware_vm_inventory
properties:
- availableField
- configIssue
- configStatus
- customValue
- datastore
- effectiveRole
- guestHeartbeatStatus
- layout
- layoutEx
- name
```

(continues on next page)

(continued from previous page)

```
- network
- overallStatus
- parentVApp
- permission
- recentTask
- resourcePool
- rootSnapshot
- snapshot
- triggeredAlarmState
- value
- capability
- config
- guest
- runtime
- storage
- summary
strict: false
with_nested_properties: true
```

19.5 Red Hat Satellite 6

```
group_prefix: foreman_
keyed_groups:
- key: foreman['environment_name'] | lower | regex_replace(' ', '') | regex_replace(
↪'[^A-Za-z0-9_]', '_') | regex_replace('none', '')
  prefix: foreman_environment_
  separator: ''
- key: foreman['location_name'] | lower | regex_replace(' ', '') | regex_replace('[^A-
↪Za-z0-9_]', '_')
  prefix: foreman_location_
  separator: ''
- key: foreman['organization_name'] | lower | regex_replace(' ', '') | regex_replace(
↪'[^A-Za-z0-9_]', '_')
  prefix: foreman_organization_
  separator: ''
- key: foreman['content_facet_attributes']['lifecycle_environment_name'] | lower |␣
↪regex_replace(' ', '') | regex_replace('[^A-Za-z0-9_]', '_')
  prefix: foreman_lifecycle_environment_
  separator: ''
- key: foreman['content_facet_attributes']['content_view_name'] | lower | regex_
↪replace(' ', '') | regex_replace('[^A-Za-z0-9_]', '_')
  prefix: foreman_content_view_
  separator: ''
legacy_hostvars: true
plugin: theforeman.foreman.foreman
validate_certs: false
want_facts: true
want_hostcollections: false
want_params: true
```

19.6 OpenStack

```
expand_hostvars: true
fail_on_errors: true
inventory_hostname: uuid
plugin: openstack.cloud.openstack
```

19.7 Red Hat Virtualization

```
compose:
  ansible_host: (devices.values() | list)[0][0] if devices else None
keyed_groups:
- key: cluster
  prefix: cluster
  separator: _
- key: status
  prefix: status
  separator: _
- key: tags
  prefix: tag
  separator: _
ovirt_hostname_preference:
- name
- fqdn
ovirt_insecure: false
plugin: ovirt.ovirt.ovirt
```

19.8 Red Hat Ansible Automation Platform

```
include_metadata: true
inventory_id: <inventory_id or url_quoted_named_url>
plugin: awx.awx.tower
validate_certs: <true or false>
```

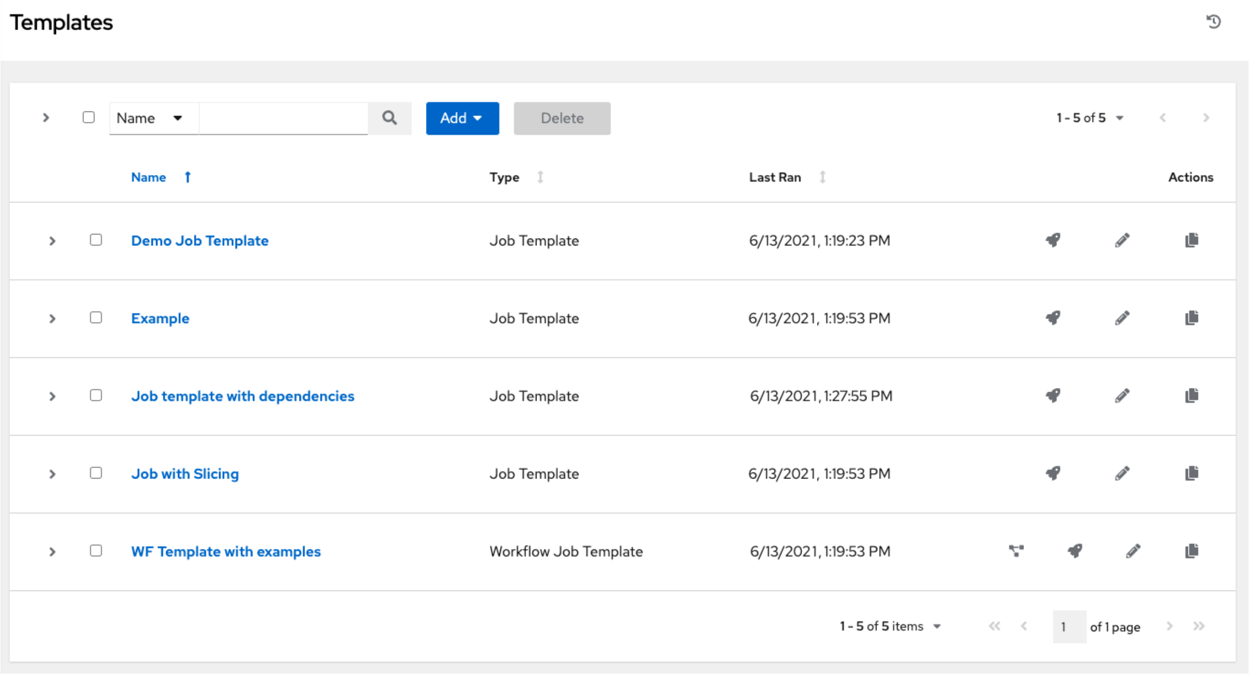
CHAPTER

TWENTY

JOB TEMPLATES

A *job template* is a definition and set of parameters for running an Ansible job. Job templates are useful to execute the same job many times. Job templates also encourage the reuse of Ansible playbook content and collaboration between teams.

The **Templates** menu opens a list of the job templates that are currently available. The default view is collapsed (Compact), showing the template name, template type, and the timestamp of last job that ran using that template. You can click **Expanded** (arrow next to each entry) to expand to view more information. This list is sorted alphabetically by name, but you can sort by other criteria, or search by various fields and attributes of a template.



Templates

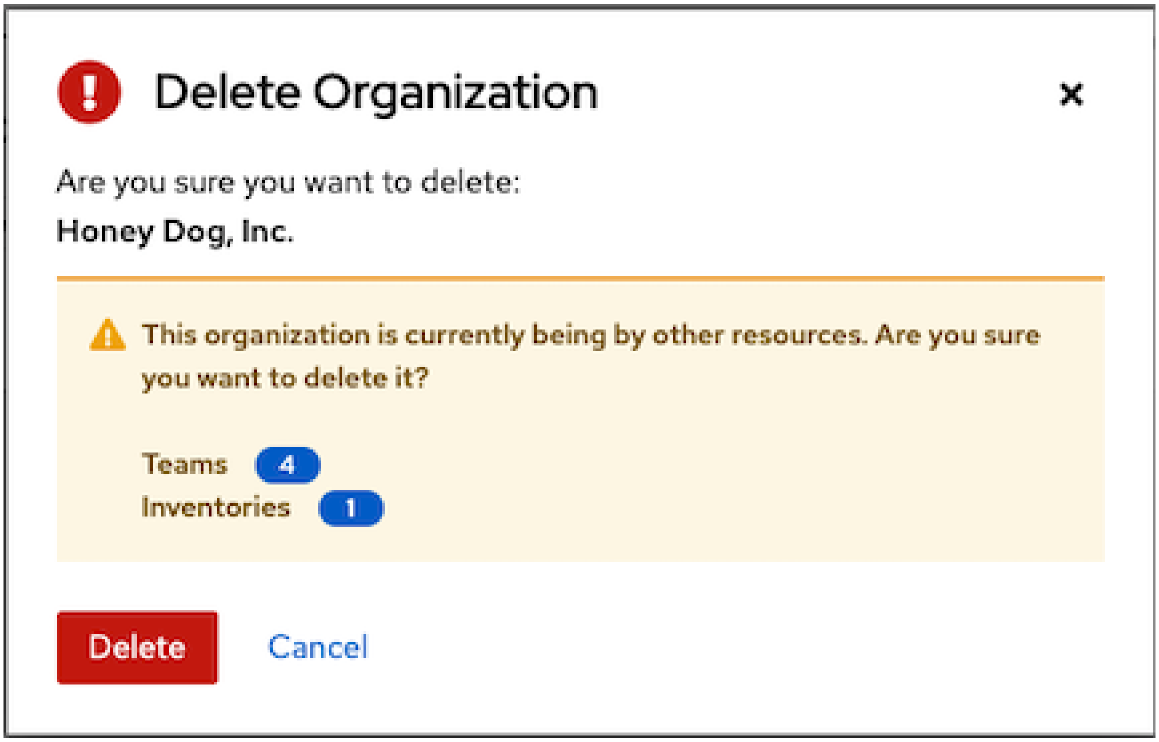
› □ Name ▾ Add ▾ Delete 1 - 5 of 5 ▾ ‹ ›

		Name ↑	Type	Last Ran	Actions
›	□	Demo Job Template	Job Template	6/13/2021, 1:19:23 PM	🚀 ✏ ⧉
›	□	Example	Job Template	6/13/2021, 1:19:53 PM	🚀 ✏ ⧉
›	□	Job template with dependencies	Job Template	6/13/2021, 1:27:55 PM	🚀 ✏ ⧉
›	□	Job with Slicing	Job Template	6/13/2021, 1:19:53 PM	🚀 ✏ ⧉
›	□	WF Template with examples	Workflow Job Template	6/13/2021, 1:19:53 PM	⋱ 🚀 ✏ ⧉

1 - 5 of 5 items ▾ « ‹ 1 of 1 page › »

From this screen, you can launch (🚀), edit (✏), and copy (⧉) a job template. To delete a job template, you must select one or more templates and click the **Delete** button. Before deleting a job template, be sure it is not used in a workflow job template.

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:



Delete Organization

Are you sure you want to delete:

Honey Dog, Inc.

This organization is currently being by other resources. Are you sure you want to delete it?

Teams 4

Inventories 1

Note: Job templates can be used to build a workflow template. For templates that show the Workflow Visualizer ( ) icon next to them are workflow templates. Clicking it allows you to graphically build a workflow. Many parameters in a job template allow you to enable **Prompt on Launch** that can be modified at the workflow level, and do not affect the values assigned at the job template level. For instructions, see the *Workflow Visualizer* section.

20.1 Create a Job Template

To create a new job template:

1. Click the **Add** button then select **Job Template** from the menu list.
2. Enter the appropriate details into the following fields:

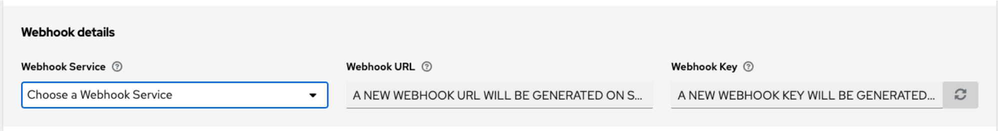
Note: If a field has the **Prompt on launch** checkbox selected, launching the job will prompt you for the value for that field upon launch. Most prompted values will override any values set in the job template; exceptions are noted below.

Field	Options	Prompt on Launch
Name	Enter a name for the job.	N/A
Description	Enter an arbitrary description as appropriate (optional).	N/A
Job Type	**Choose a job type:** - **Run**: Execute the playbook when launched, running Ansible tasks on the selected hosts. - **Check**: Perform a "dry run" of the playbook and report changes that would be made without actually making them. Tasks that do not support check mode will be skipped and will not report potential changes. More information on job types can be found in the Playbooks: Special Topics section of the Ansible documentation.	Yes
Inventory	Choose the inventory to be used with this job template from the inventories available to the currently logged in user.	Yes. Inventory prompts will show up as its own step in a subsequent prompt window.
Project	Choose the project to be used with this job template from the projects available to the currently logged in user.	N/A
SCM Branch	This field is only present if you chose a project that allows branch override. Specify the overriding branch to use in your job run. If left blank, the specified SCM branch (or commit hash or tag) from the project is used. For more detail, see *job branch overriding*.	Yes
Execution Environment	Select the container image to be used to run this job. A project must be selected before you can select an execution environment.	Yes. Execution environment prompts will show up as its own step in a subsequent prompt window.
Playbook	Choose the playbook to be launched with this job template from the available playbooks. This field automatically populates with the names of the playbooks found in the project base path for the selected project. Alternatively, you can enter the name of the playbook if it is not listed, such as the name of a file (like `foo.yml`) you want to use to run with that playbook. If you enter a filename that is not valid, the template will display an error, or cause the job to fail.	N/A
Credentials	Click the 🔍 button to open a separate window. Choose the credential from the available options to be used with this job template. Use the drop-down menu list to filter by credential type if the list is extensive. Some credential types are not listed because they do not apply to certain job templates.	- If selected, upon launching a job template that has a default credential and supplying another credential will replace the default credential if it is the same type. Example of such a message: `Job Template default credentials must be replaced with one of the same type. Please select a credential for the following types in order to proceed: Machine.` - Alternatively, you can add more credentials as you see fit. - Credential prompts will show up as its own step in a subse-

3. **Options**: Specify options for launching this template, if necessary.

- **Privilege Escalation**: If checked, you enable this playbook to run as an administrator. This is the equivalent of passing the `--become` option to the `ansible-playbook` command.
- **Provisioning Callbacks**: If checked, you enable a host to call back to automation controller via the REST API and invoke the launch of a job from this job template. Refer to *Provisioning Callbacks* for additional information.
- **Enable Webhook**: Turns on the ability to interface with a predefined SCM system web service that is used to launch a job template. Currently supported SCM systems are GitHub and GitLab.

  If you enable webhooks, other fields display, prompting for additional information:

  Webhook details

  Webhook Service: Choose a Webhook Service

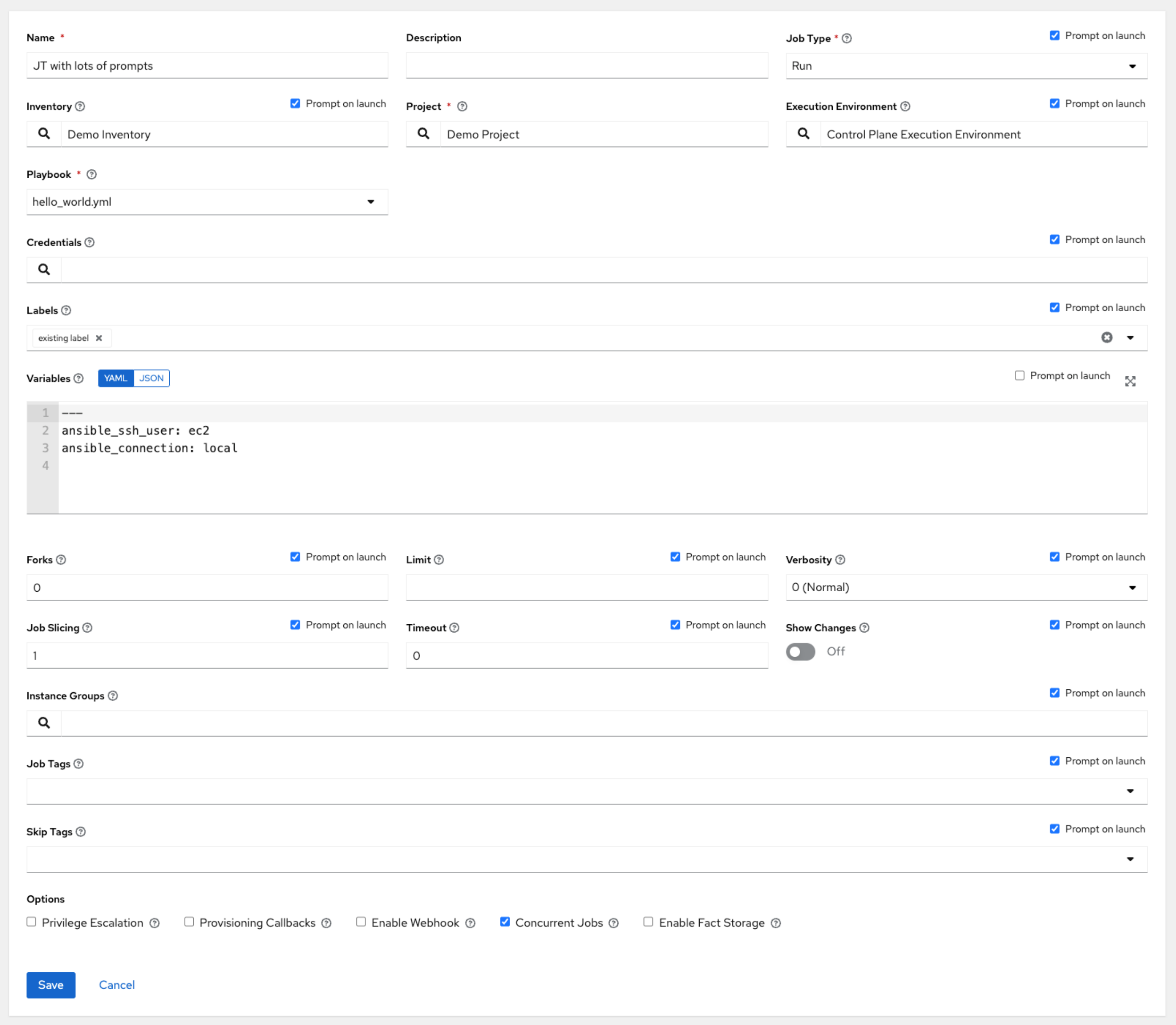
  Webhook URL: A NEW WEBHOOK URL WILL BE GENERATED ON S...

  Webhook Key: A NEW WEBHOOK KEY WILL BE GENERATED...

  - **Webhook Service**: Select which service to listen for webhooks from
  - **Webhook URL**: Automatically populated with the URL for the webhook service to POST requests to.
  - **Webhook Key**: Generated shared secret to be used by the webhook service to sign payloads sent to automation controller. This must be configured in the settings on the webhook service in order for automation controller to accept webhooks from this service.
  - **Webhook Credential**: Optionally, provide a GitHub or GitLab personal access token (PAT) as a credential to use to send status updates back to the webhook service. Before you can select it, the credential must exist. See *Credential Types* to create one.

  For additional information on setting up webhooks, see *Working with Webhooks*.

  - **Concurrent Jobs**: If checked, you are allowing jobs in the queue to run simultaneously if not dependent on one another. Check this box if you want to run job slices simultaneously. Refer to *Automation Controller Capacity Determination and Job Impact* for additional information.
  - **Enable Fact Storage**: When checked, automation controller will store gathered facts for all hosts in an inventory related to the job running.



Name * : JT with lots of prompts

Description:

Job Type * : Run — ☑ Prompt on launch

Inventory: Demo Inventory — ☑ Prompt on launch

Project * : Demo Project

Execution Environment: Control Plane Execution Environment — ☑ Prompt on launch

Playbook * : hello_world.yml

Credentials: — ☑ Prompt on launch

Labels: existing label ✖ — ☑ Prompt on launch

Variables (YAML | JSON) — ☐ Prompt on launch

```
---
ansible_ssh_user: ec2
ansible_connection: local
```

Forks: 0 — ☑ Prompt on launch

Limit: — ☑ Prompt on launch

Verbosity: 0 (Normal) — ☑ Prompt on launch

Job Slicing: 1 — ☑ Prompt on launch

Timeout: 0 — ☑ Prompt on launch

Show Changes: Off — ☑ Prompt on launch

Instance Groups: — ☑ Prompt on launch

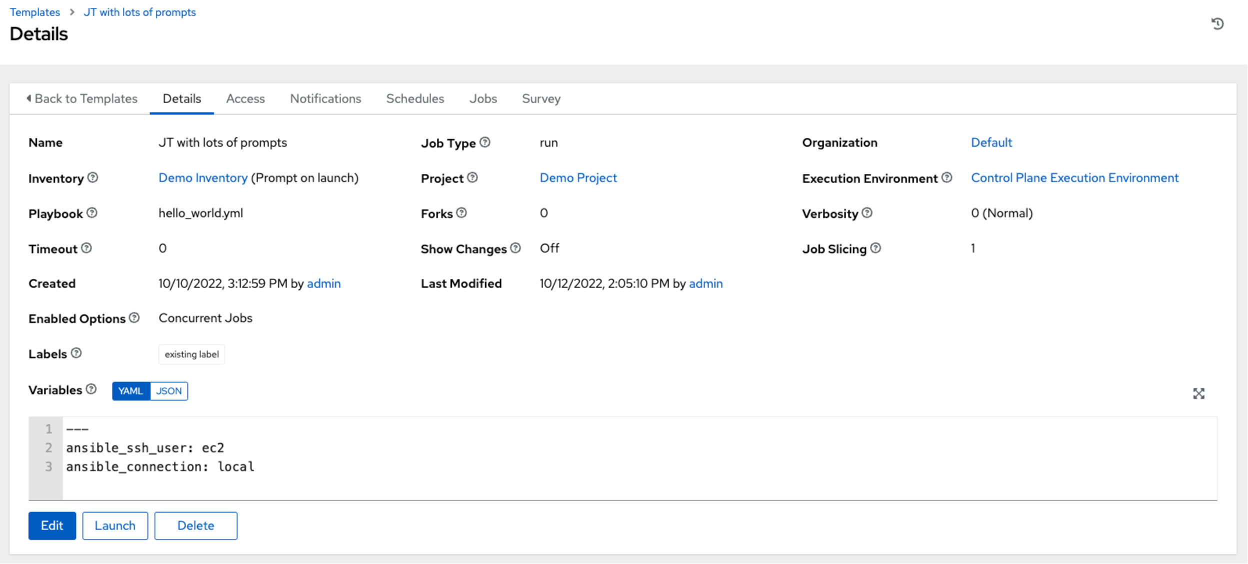
Job Tags: — ☑ Prompt on launch

Skip Tags: — ☑ Prompt on launch

Options: ☐ Privilege Escalation ☐ Provisioning Callbacks ☐ Enable Webhook ☑ Concurrent Jobs ☐ Enable Fact Storage

4. When you have completed configuring the details of the job template, click **Save**.

Saving the template does not exit the job template page but advances to the Job Template Details tab for viewing. After saving the template, you can click **Launch** to launch the job, or click **Edit** to add or change the attributes of the template, such as permissions, notifications, view completed jobs, and add a survey (if the job type is not a scan). You must first save the template prior to launching, otherwise, the **Launch** button remains grayed-out.



Name	JT with lots of prompts	Job Type	run	Organization	Default
Inventory	Demo Inventory (Prompt on launch)	Project	Demo Project	Execution Environment	Control Plane Execution Environment
Playbook	hello_world.yml	Forks	0	Verbosity	0 (Normal)
Timeout	0	Show Changes	Off	Job Slicing	1
Created	10/10/2022, 3:12:59 PM by admin	Last Modified	10/12/2022, 2:05:10 PM by admin		
Enabled Options	Concurrent Jobs				
Labels	existing label				

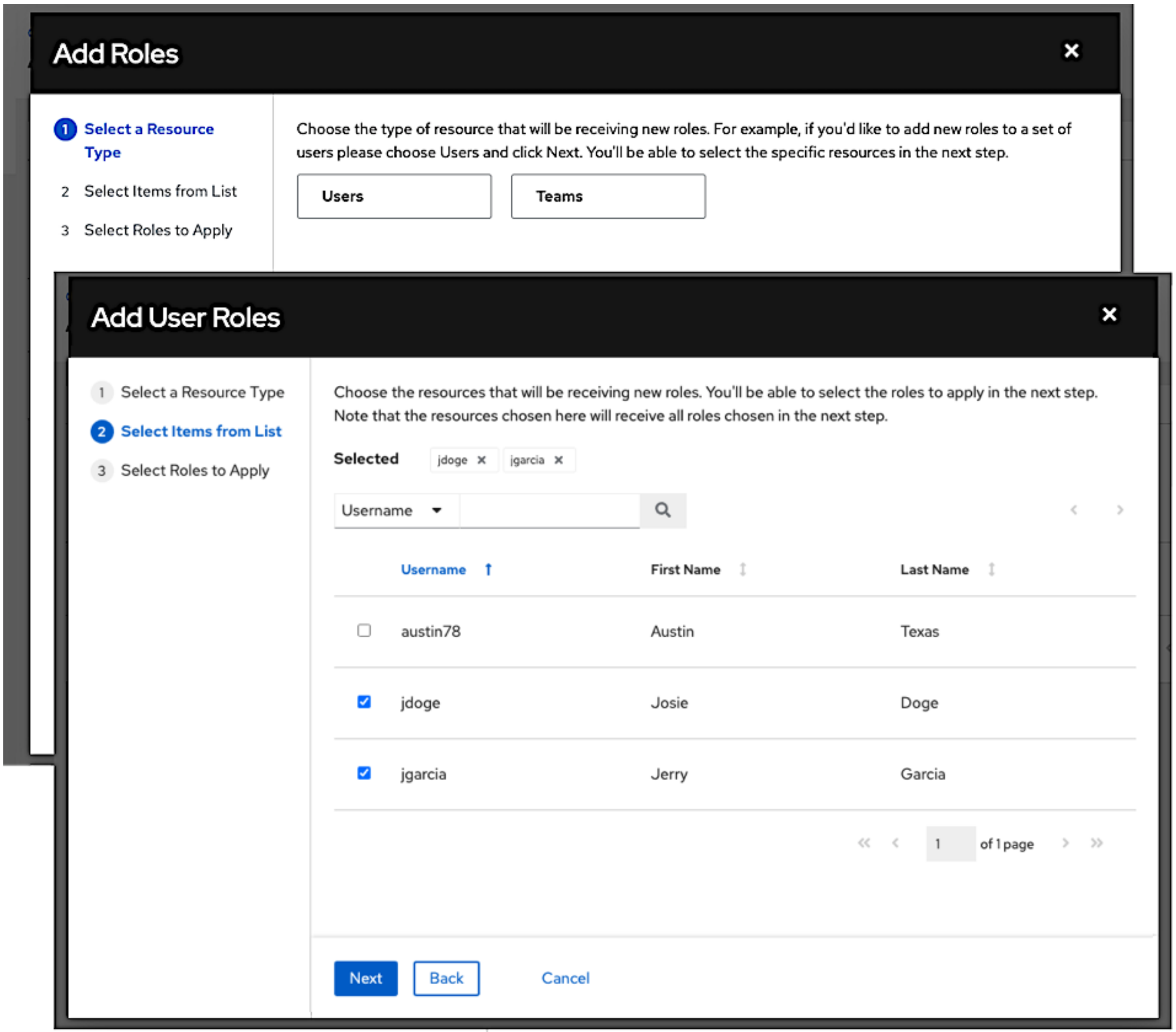
Variables

```
---
ansible_ssh_user: ec2
ansible_connection: local
```

You can verify the template is saved when the newly created template appears on the Templates list view.

20.2 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**.



Choose the type of resource that will be receiving new roles. For example, if you'd like to add new roles to a set of users please choose Users and click Next. You'll be able to select the specific resources in the next step.

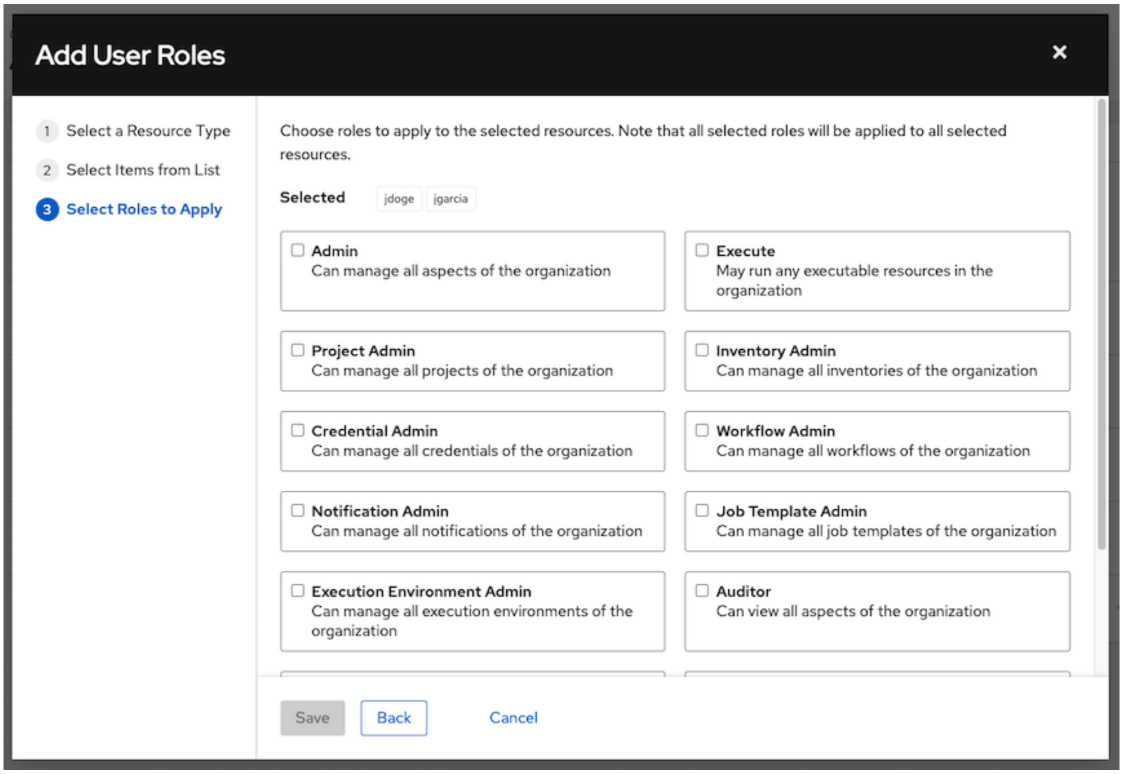
Choose the resources that will be receiving new roles. You'll be able to select the roles to apply in the next step. Note that the resources chosen here will receive all roles chosen in the next step.

Selected jdoge ✕ jgarcia ✕

	Username	First Name	Last Name
□	austin78	Austin	Texas
☑	jdoge	Josie	Doge
☑	jgarcia	Jerry	Garcia

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

4. 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.



Add User Roles ×

1 Select a Resource Type
2 Select Items from List
3 Select Roles to Apply

Choose roles to apply to the selected resources. Note that all selected roles will be applied to all selected resources.

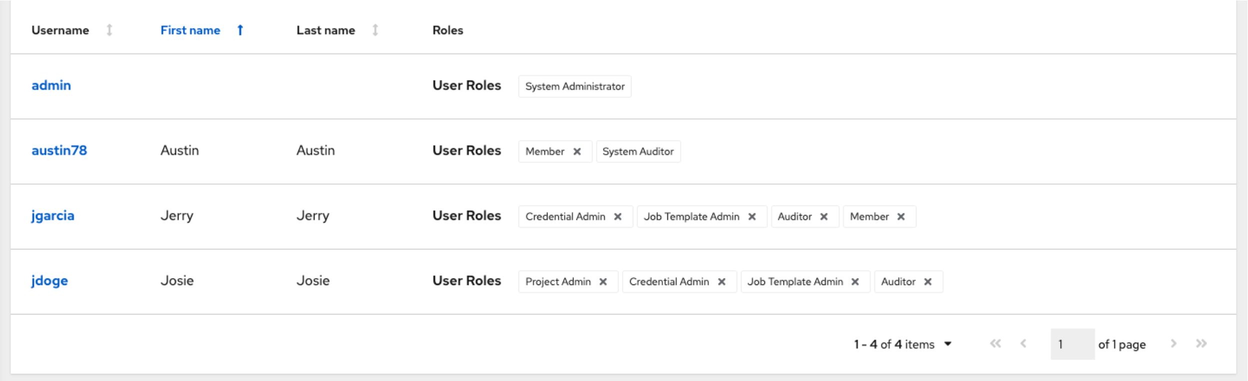
Selected jdoge jgarcia

- □ **Admin** Can manage all aspects of the organization
- □ **Execute** May run any executable resources in the organization
- □ **Project Admin** Can manage all projects of the organization
- □ **Inventory Admin** Can manage all inventories of the organization
- □ **Credential Admin** Can manage all credentials of the organization
- □ **Workflow Admin** Can manage all workflows of the organization
- □ **Notification Admin** Can manage all notifications of the organization
- □ **Job Template Admin** Can manage all job templates of the organization
- □ **Execution Environment Admin** Can manage all execution environments of the organization
- □ **Auditor** Can view all aspects of the organization

Save Back Cancel

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

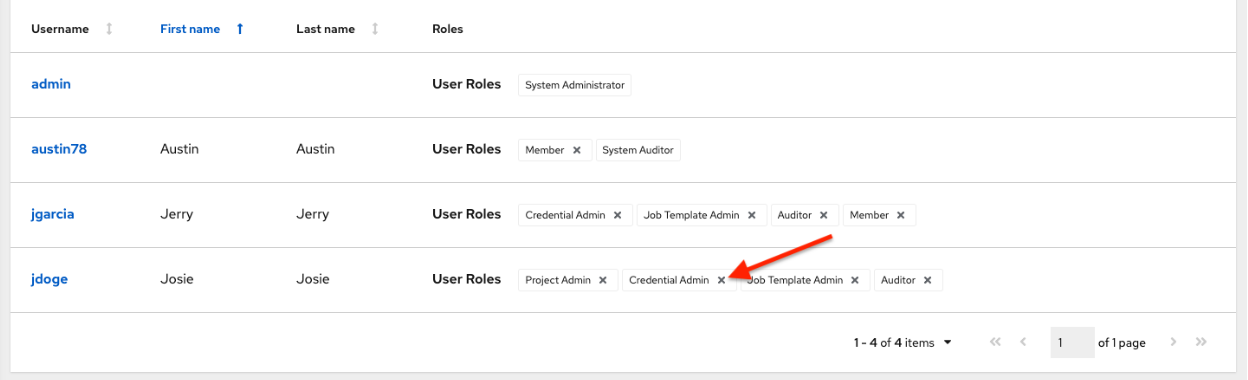
The Add Users/Teams window closes to display the updated roles assigned for each user and team.



Username	First name	Last name	Roles	
admin			User Roles	System Administrator
austin78	Austin	Austin	User Roles	Member × System Auditor
jgarcia	Jerry	Jerry	User Roles	Credential Admin × Job Template Admin × Auditor × Member ×
jdoge	Josie	Josie	User Roles	Project Admin × Credential Admin × Job Template Admin × Auditor ×

1 - 4 of 4 items 1 of 1 page

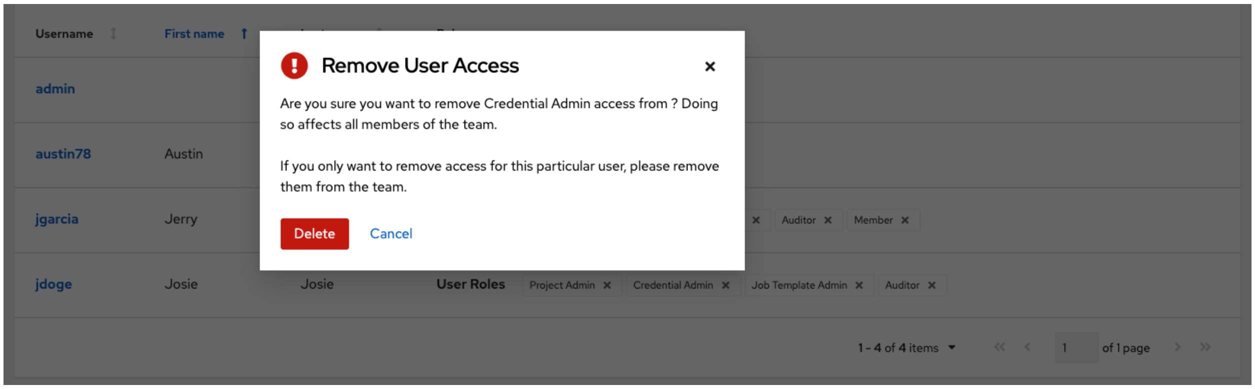
To remove roles for a particular user, click the disassociate (x) button next to its resource.



Username	First name	Last name	Roles	
admin			User Roles	System Administrator
austin78	Austin	Austin	User Roles	Member × System Auditor
jgarcia	Jerry	Jerry	User Roles	Credential Admin × Job Template Admin × Auditor × Member ×
jdoge	Josie	Josie	User Roles	Project Admin × Credential Admin × Job Template Admin × Auditor ×

1 - 4 of 4 items 1 of 1 page

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



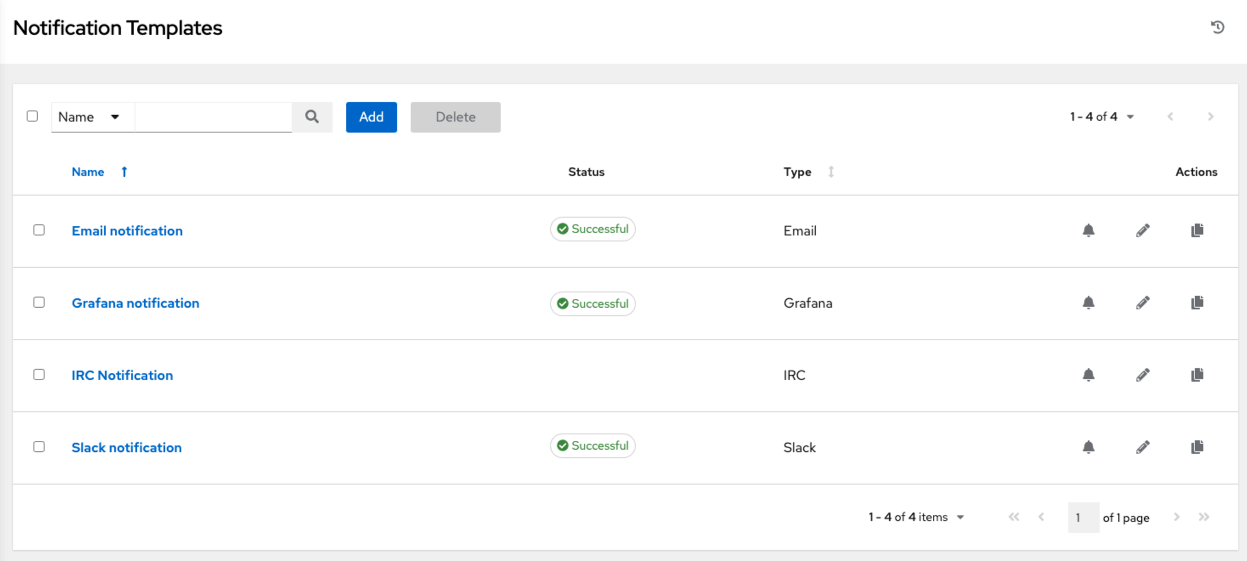
Remove User Access

Are you sure you want to remove Credential Admin access from ? Doing so affects all members of the team.

If you only want to remove access for this particular user, please remove them from the team.

20.3 Work with Notifications

Clicking the **Notifications** tab allows you to review any notification integrations you have setup and their statuses, if they have ran.



Notification Templates

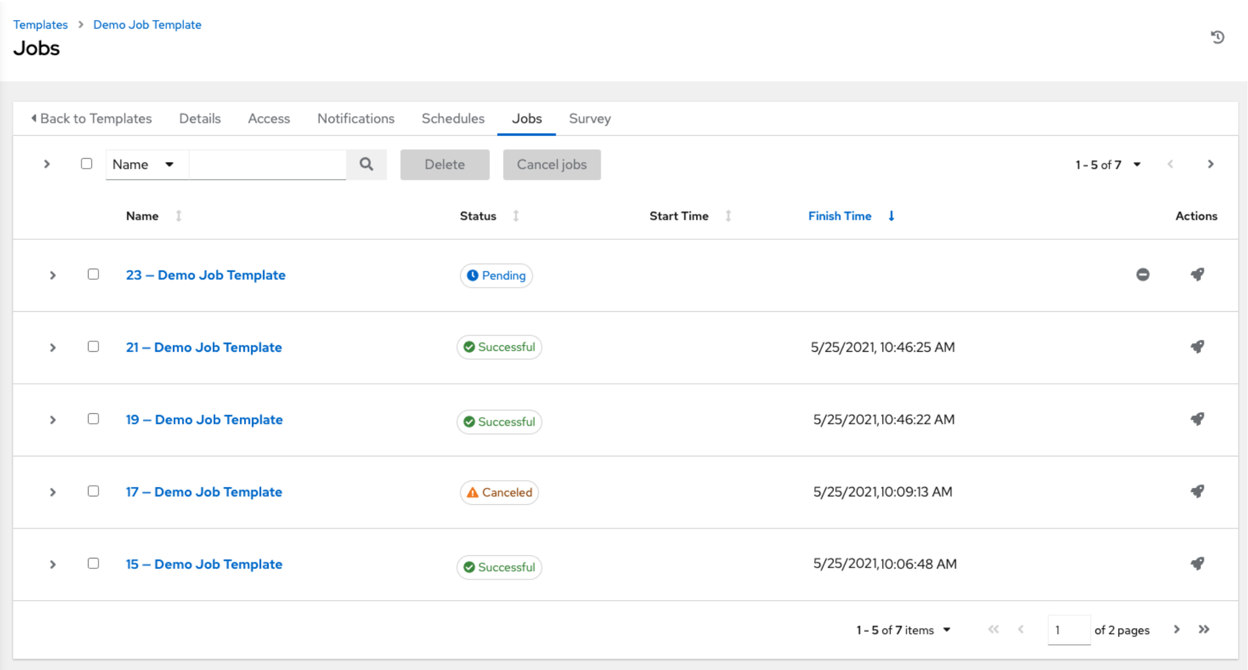
	Name	Status	Type	Actions
□	Email notification	Successful	Email	
□	Grafana notification	Successful	Grafana	
□	IRC Notification		IRC	
□	Slack notification	Successful	Slack	

Use the toggles to enable or disable the notifications to use with your particular template. For more detail, see *Enable and Disable Notifications*.

If no notifications have been set up, click the **Add** button to create a new notification. Refer to *Notification Types* for additional details on configuring various notification types and extended messaging.

20.4 View Completed Jobs

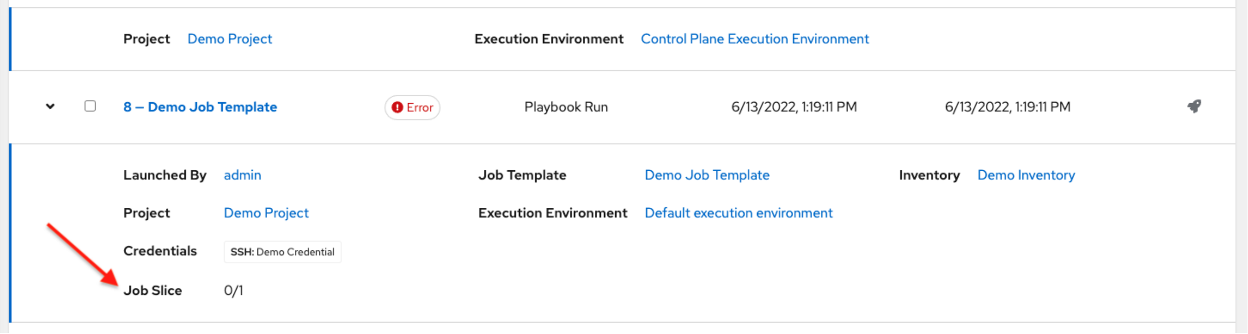
The **Completed Jobs** tab provides the list of job templates that have ran. Click **Expanded** to view details of each job, including its status, ID, and name; type of job, time started and completed, who started the job; and which template, inventory, project, and credential were used. You can filter the list of completed jobs using any of these criteria.



Name	Status	Start Time	Finish Time	Actions
23 – Demo Job Template	Pending			
21 – Demo Job Template	Successful		5/25/2021, 10:46:25 AM	
19 – Demo Job Template	Successful		5/25/2021,10:46:22 AM	
17 – Demo Job Template	Canceled		5/25/2021,10:09:13 AM	
15 – Demo Job Template	Successful		5/25/2021,10:06:48 AM	

1 - 5 of 7 items | 1 of 2 pages

Sliced jobs that display on this list are labeled accordingly, with the number of sliced jobs that have run:



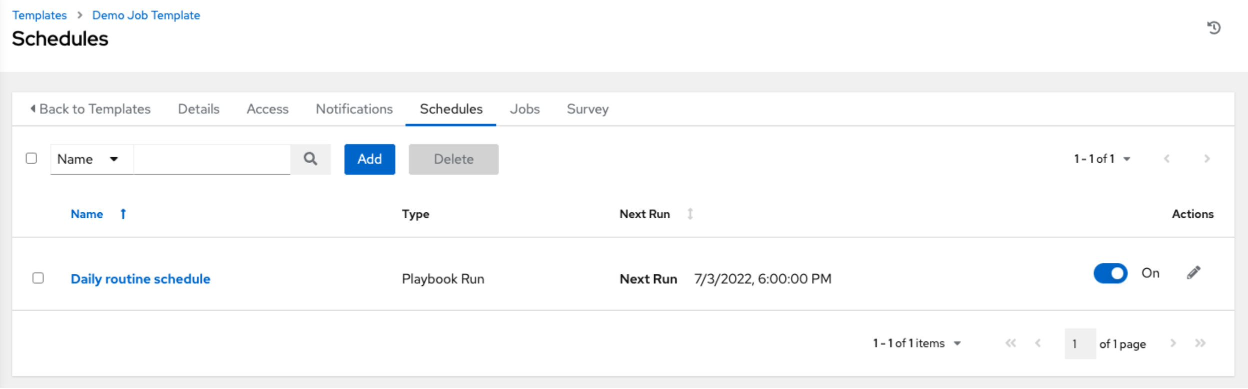
Project: Demo Project | Execution Environment: Control Plane Execution Environment

8 – Demo Job Template	Error	Playbook Run	6/13/2022, 1:19:11 PM	6/13/2022, 1:19:11 PM

Launched By	admin	Job Template	Demo Job Template	Inventory	Demo Inventory
Project	Demo Project	Execution Environment	Default execution environment		
Credentials	SSH: Demo Credential				
Job Slice	0/1				

20.5 Scheduling

Access the schedules for a particular job template from the **Schedules** tab.



Name	Type	Next Run	Actions
Daily routine schedule	Playbook Run	Next Run 7/3/2022, 6:00:00 PM	On

20.5.1 Schedule a Job Template

To schedule a job template run, click the **Schedules** tab.

- If schedules are already set up; review, edit, or enable/disable your schedule preferences.
- If schedules have not been set up, refer to *Schedules* for more information.

If **Prompt on Launch** was selected for the **Credentials** field, and you create or edit scheduling information for your job template, a **Prompt** button displays at the bottom of the Schedules form. You will not be able to remove the default machine credential in the Prompt dialog without replacing it with another machine credential before you can save it. Below is an example of such a message:

Note: To able to set `extra_vars` on schedules, you must select **Prompt on Launch** for **Variables** on the job template, or a configure and enable a survey on the job template, then those answered survey questions become `extra_vars`.

20.6 Surveys

Job types of Run or Check will provide a way to set up surveys in the Job Template creation or editing screens. Surveys set extra variables for the playbook similar to 'Prompt for Extra Variables' does, but in a user-friendly question and answer way. Surveys also allow for validation of user input. Click the **Survey** tab to create a survey.

Use cases for surveys are numerous. An example might be if operations wanted to give developers a "push to stage" button they could run without advanced Ansible knowledge. When launched, this task could prompt for answers to questions such as, "What tag should we release?"

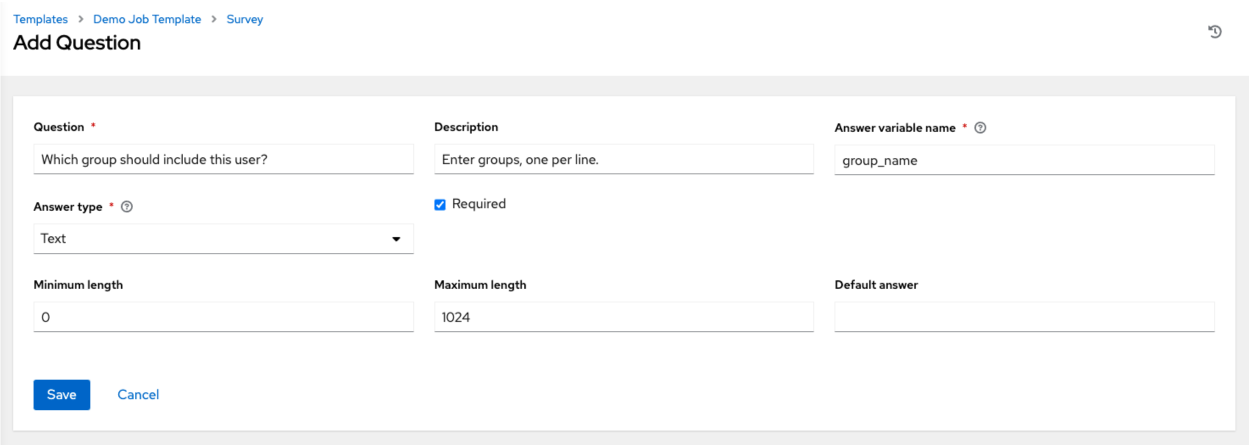
Many types of questions can be asked, including multiple-choice questions.

20.6.1 Create a Survey

To create a survey:

1. Click the **Survey** tab and click the **Add** button.
2. A survey can consist of any number of questions. For each question, enter the following information:

- **Question**: The question to ask the user
- **Description**: (optional) A description of what's being asked of the user.
- **Answer Variable Name**: The Ansible variable name to store the user's response in. This is the variable to be used by the playbook. Variable names cannot contain spaces.
- **Answer Type**: Choose from the following question types.
  - *Text*: A single line of text. You can set the minimum and maximum length (in characters) for this answer.
  - *Textarea*: A multi-line text field. You can set the minimum and maximum length (in characters) for this answer.
  - *Password*: Responses are treated as sensitive information, much like an actual password is treated. You can set the minimum and maximum length (in characters) for this answer.
  - *Multiple Choice (single select)*: A list of options, of which only one can be selected at a time. Enter the options, one per line, in the **Multiple Choice Options** box.
  - *Multiple Choice (multiple select)*: A list of options, any number of which can be selected at a time. Enter the options, one per line, in the **Multiple Choice Options** box.
  - *Integer*: An integer number. You can set the minimum and maximum length (in characters) for this answer.
  - *Float*: A decimal number. You can set the minimum and maximum length (in characters) for this answer.
- **Required**: Whether or not an answer to this question is required from the user.
- **Minimum length** and **Maximum length**: Specify if a certain length in the answer is required.
- **Default answer**: The default answer to the question. This value is pre-filled in the interface and is used if the answer is not provided by the user.



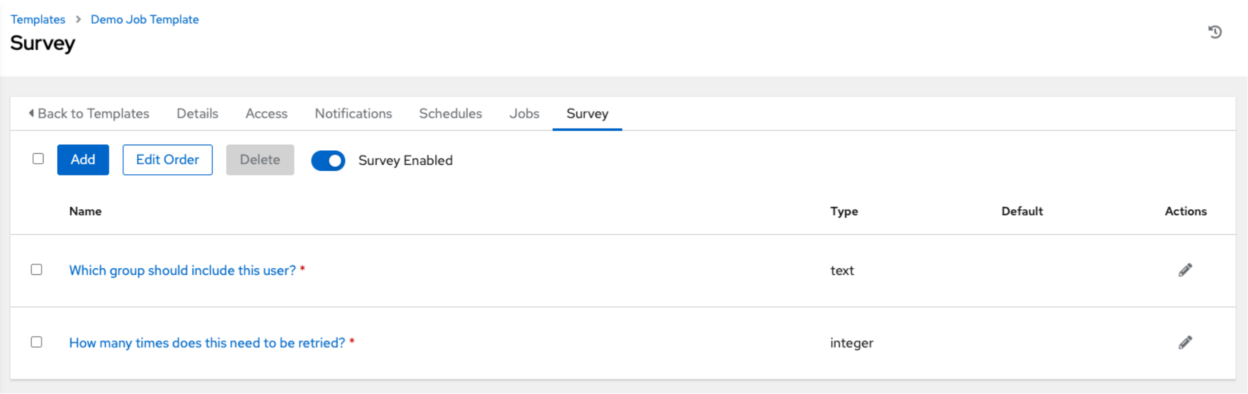
Templates > Demo Job Template > Survey

Add Question

Question *	Description	Answer variable name *
Which group should include this user?	Enter groups, one per line.	group_name
Answer type *	☑ Required	
Text		
Minimum length	Maximum length	Default answer
0	1024	

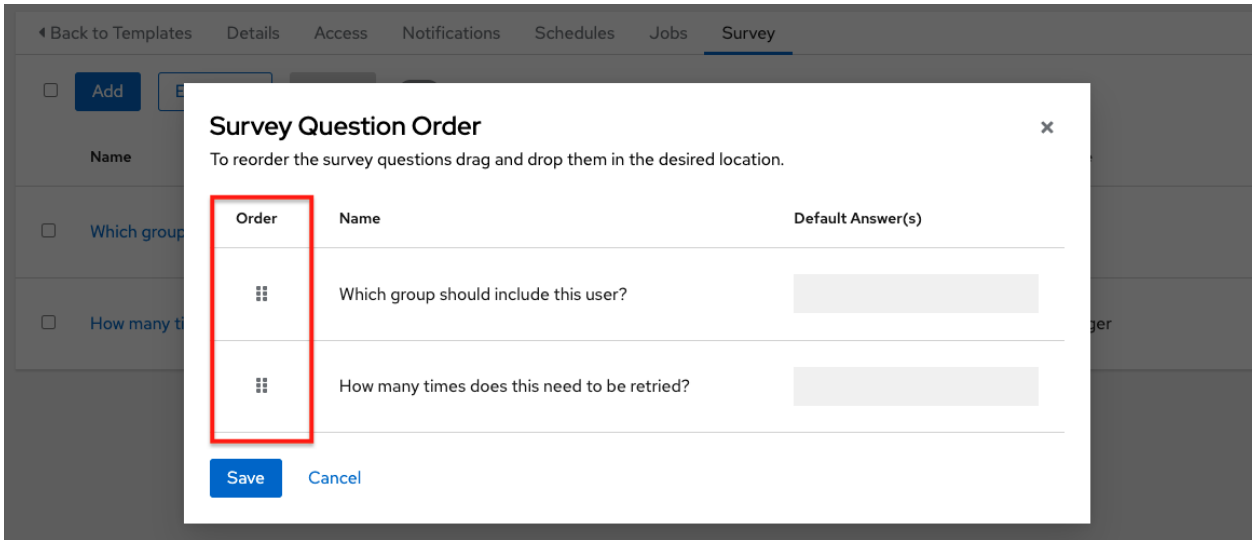
3. Once you have entered the question information, click **Save** to add the question.

The survey question displays in the Survey list. For any question, you can click ✏ to edit the question, or check the box next to each question and click **Delete** to delete the question, or use the toggle button at the top of the screen to enable or disable the survey prompt(s).



	Name	Type	Default	Actions
□	Which group should include this user? *	text		✎
□	How many times does this need to be retried? *	integer		✎

If you have more than one survey question, use the **Edit Order** button to rearrange the order of the questions by clicking and dragging on the grid icon.



Survey Question Order

To reorder the survey questions drag and drop them in the desired location.

Order	Name	Default Answer(s)
⠿	Which group should include this user?	
⠿	How many times does this need to be retried?	

4. To add more questions, click the **Add** button to add additional questions.

20.6.2 Optional Survey Questions

The **Required** setting on a survey question determines whether the answer is optional or not for the user interacting with it.

Behind the scenes, optional survey variables can be passed to the playbook in `extra_vars`, even when they aren't filled in.

- If a non-text variable (input type) is marked as optional, and is not filled in, no survey `extra_var` is passed to the playbook.
- If a text input or text area input is marked as optional, is not filled in, and has a minimum `length > 0`, no survey `extra_var` is passed to the playbook.
- If a text input or text area input is marked as optional, is not filled in, and has a minimum `length === 0`, that survey `extra_var` is passed to the playbook, with the value set to an empty string (“”).

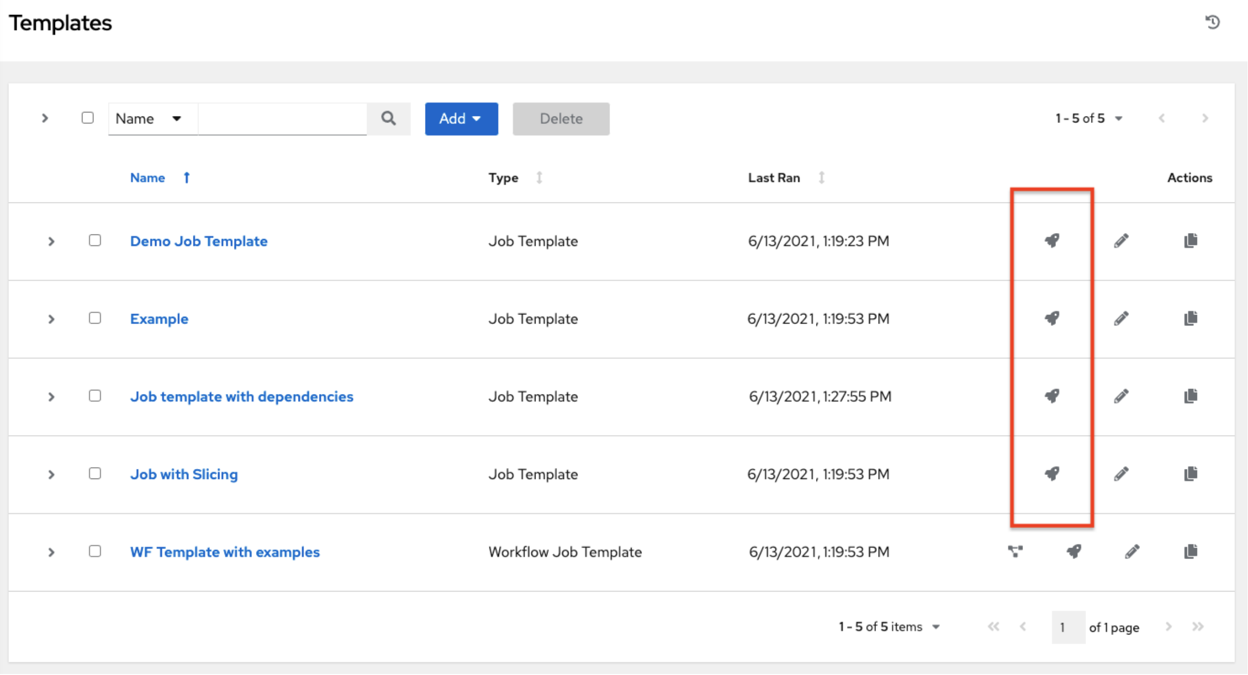
20.7 Launch a Job Template

A major benefit of automation controller is the push-button deployment of Ansible playbooks. You can easily configure a template to store all parameters you would normally pass to the ansible-playbook on the command line–not just the playbooks, but the inventory, credentials, extra variables, and all options and settings you can specify on the command line.

Easier deployments drive consistency, by running your playbooks the same way each time, and allow you to delegate responsibilities–even users who aren't Ansible experts can run playbooks written by others.

Launch a job template by any of the following ways:

- Access the job template list from the **Templates** menu on the left navigation bar or while in the Job Template Details view, scroll to the bottom to access the 🚀 button from the list of templates.



Name ↑	Type	Last Ran	Actions
Demo Job Template	Job Template	6/13/2021, 1:19:23 PM	🚀 ✎ ⧉
Example	Job Template	6/13/2021, 1:19:53 PM	🚀 ✎ ⧉
Job template with dependencies	Job Template	6/13/2021, 1:27:55 PM	🚀 ✎ ⧉
Job with Slicing	Job Template	6/13/2021, 1:19:53 PM	🚀 ✎ ⧉
WF Template with examples	Workflow Job Template	6/13/2021, 1:19:53 PM	🚀 ✎ ⧉

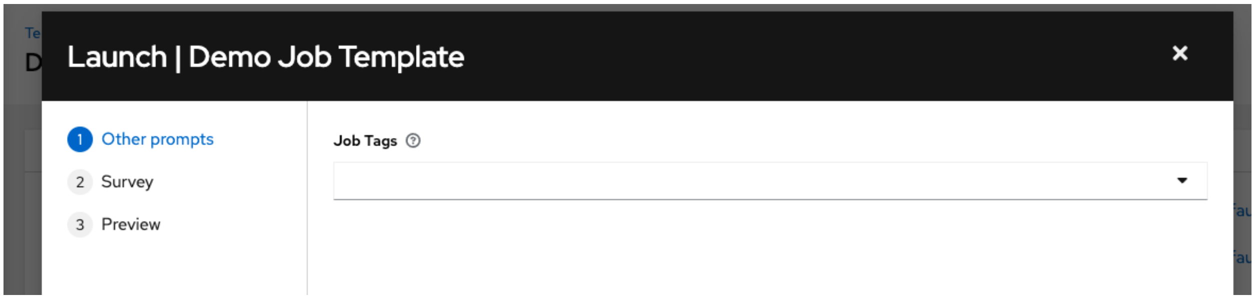
- While in the Job Template Details view of the job template you want to launch, click **Launch**.

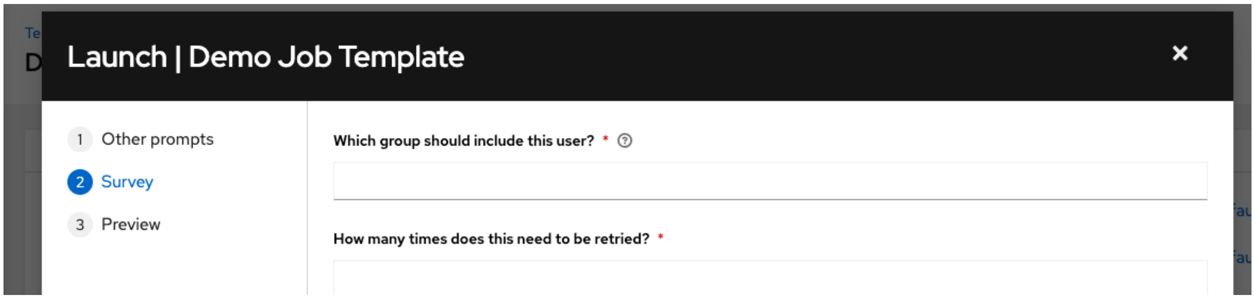
A job may require additional information to run. The following data may be requested at launch:

- Credentials that were setup
- The option `Prompt on Launch` is selected for any parameter
- Passwords or passphrases that have been set to **Ask**
- A survey, if one has been configured for the job templates
- Extra variables, if requested by the job template

Note: If a job has user-provided values, then those are respected upon relaunch. If the user did not specify a value, then the job uses the default value from the job template. Jobs are not relaunched as-is. They are relaunched with the user prompts re-applied to the job template.

Below is an example job launch that prompts for Job Tags, and runs the example survey created in *Surveys*.




Launch | Demo Job Template
×
1 Other prompts
2 Survey
3 Preview
Which group should include this user? *
How many times does this need to be retried? *

Note: Providing values on one tab, and going back to a previous tab, and then continuing on to the next tab will result in having to re-provide values on the rest of the tabs. Make sure you fill in the tabs in the order the prompts appear to avoid this.

Along with any extra variables set in the job template and survey, automation controller automatically adds the following variables to the job environment. Also note, awx_``* variables are defined by the system and cannot be overridden. Variables about the job context, like ``awx_job_template_name are not affected if they are set in extra_vars.

- awx_job_id: The Job ID for this job run
- awx_job_launch_type: The description to indicate how the job was started:
  - **manual**: Job was started manually by a user.
  - **relaunch**: Job was started via relaunch.
  - **callback**: Job was started via host callback.
  - **scheduled**: Job was started from a schedule.
  - **dependency**: Job was started as a dependency of another job.
  - **workflow**: Job was started from a workflow job.
  - **sync**: Job was started from a project sync.
  - **scm**: Job was created as an Inventory SCM sync.
- awx_job_template_id: The Job Template ID that this job run uses
- awx_job_template_name: The Job Template name that this job uses
- awx_project_revision: The revision identifier for the source tree that this particular job uses (it is also the same as the job's field scm_revision)
- awx_project_scm_branch: The configured default project SCM branch for the project the job template uses
- awx_job_scm_branch If the SCM Branch is overwritten by the job, the value is shown here

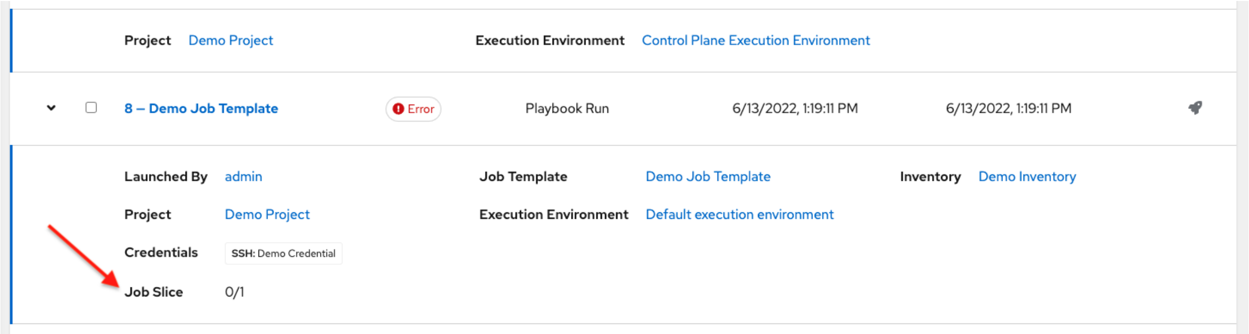
- awx_user_email: The user email of the controller user that started this job. This is not available for callback or scheduled jobs.
- awx_user_first_name: The user's first name of the controller user that started this job. This is not available for callback or scheduled jobs.
- awx_user_id: The user ID of the controller user that started this job. This is not available for callback or scheduled jobs.
- awx_user_last_name: The user's last name of the controller user that started this job. This is not available for callback or scheduled jobs.
- awx_user_name: The user name of the controller user that started this job. This is not available for callback or scheduled jobs.
- awx_schedule_id: If applicable, the ID of the schedule that launched this job
- awx_schedule_name: If applicable, the name of the schedule that launched this job
- awx_workflow_job_id: If applicable, the ID of the workflow job that launched this job
- awx_workflow_job_name: If applicable, the name of the workflow job that launched this job. Note this is also the same as the workflow job template.
- awx_inventory_id: If applicable, the ID of the inventory this job uses
- awx_inventory_name: If applicable, the name of the inventory this job uses

For compatibility, all variables are also given an "awx" prefix, for example, awx_job_id.

Upon launch, automation controller automatically redirects the web browser to the Job Status page for this job under the **Jobs** tab.

Note: You can re-launch the most recent job from the list view to re-run on all hosts or just failed hosts in the specified inventory. Refer to *Jobs* in the *Automation Controller User Guide* for more detail.

When slice jobs are running, job lists display the workflow and job slices, as well as a link to view their details individually.



Project Demo Project
Execution Environment Control Plane Execution Environment

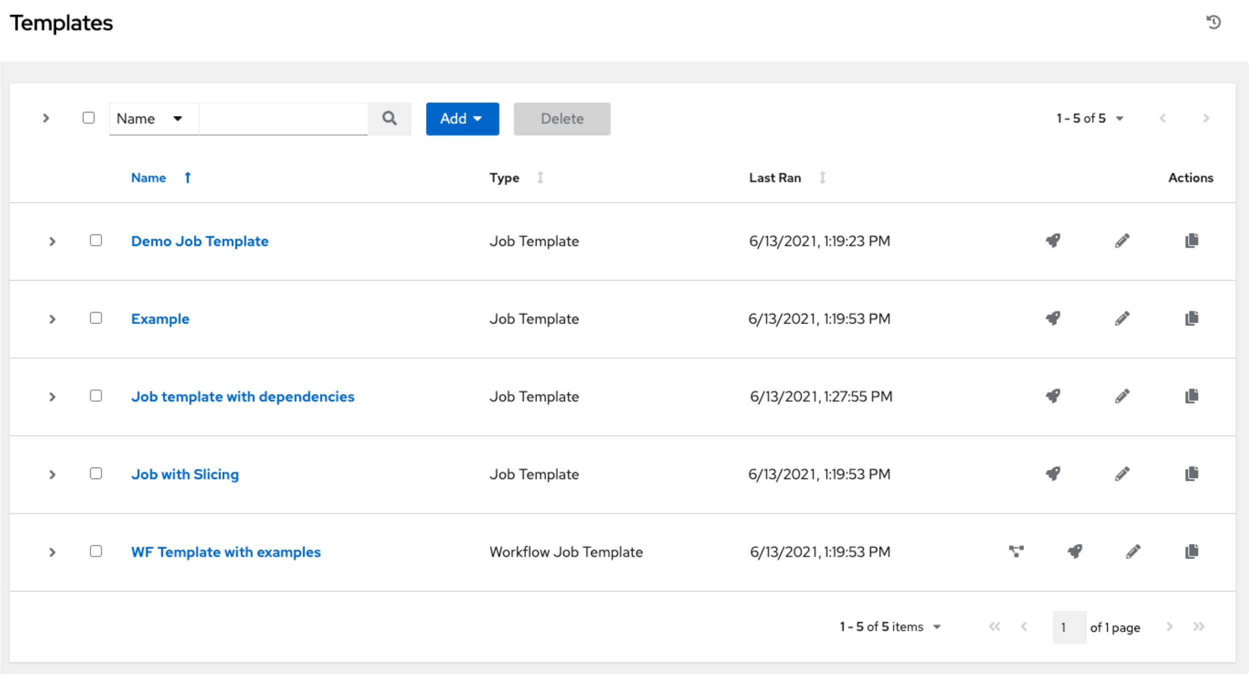
8 – Demo Job Template | Error | Playbook Run | 6/13/2022, 1:19:11 PM | 6/13/2022, 1:19:11 PM

Launched By admin
Job Template Demo Job Template
Inventory Demo Inventory
Project Demo Project
Execution Environment Default execution environment
Credentials SSH: Demo Credential
Job Slice 0/1

20.8 Copy a Job Template

If you choose to copy Job Template, it **does not** copy any associated schedule, notifications, or permissions. Schedules and notifications must be recreated by the user or admin creating the copy of the Job Template. The user copying the Job Template will be granted the admin permission, but no permissions are assigned (copied) to the Job Template.

1. Access the job template list from the **Templates** menu on the left navigation bar or while in the Job Template Details view, scroll to the bottom to access it from the list of templates.



Templates

Name	Type	Last Ran	Actions
Demo Job Template	Job Template	6/13/2021, 1:19:23 PM	
Example	Job Template	6/13/2021, 1:19:53 PM	
Job template with dependencies	Job Template	6/13/2021, 1:27:55 PM	
Job with Slicing	Job Template	6/13/2021, 1:19:53 PM	
WF Template with examples	Workflow Job Template	6/13/2021, 1:19:53 PM	

2. Click the ⧉ button associated with the template you want to copy.

The new template with the name of the template from which you copied and a timestamp displays in the list of templates.

3. Click to open the new template and click **Edit**.
4. Replace the contents of the **Name** field with a new name, and provide or modify the entries in the other fields to complete this page.
5. Click **Save** when done.

20.9 Scan Job Templates

Scan jobs are no longer supported starting with automation controller 3.2. This system tracking feature was used as a way to capture and store facts as historical data. Facts are now stored in the controller via fact caching. For more information, see *Fact Caching*.

If you have Job Template Scan Jobs in your system prior to automation controller 3.2, they have been converted to type run (like normal job templates) and retained their associated resources (i.e. inventory, credential). Job Template Scan Jobs that do not have a related project are assigned a special playbook by default, or you can specify a project with your own scan playbook. A project was created for each organization that points to https://github.com/ansible/

tower-fact-modules and the Job Template was set to the playbook, https://github.com/ansible/tower-fact-modules/blob/master/scan_facts.yml.

20.9.1 Fact Scan Playbooks

The scan job playbook, `scan_facts.yml`, contains invocations of three `fact scan modules` - packages, services, and files, along with Ansible's standard fact gathering. The `scan_facts.yml` playbook file looks like the following:

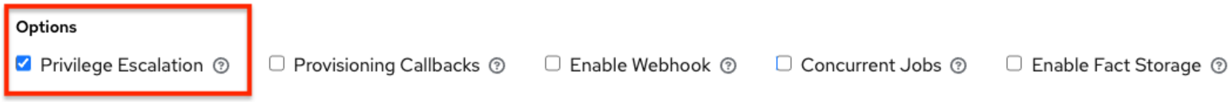
```
- hosts: all
  vars:
    scan_use_checksum: false
    scan_use_recursive: false
  tasks:
    - scan_packages:
    - scan_services:
    - scan_files:
        paths: '{{ scan_file_paths }}'
        get_checksum: '{{ scan_use_checksum }}'
        recursive: '{{ scan_use_recursive }}'
      when: scan_file_paths is defined
```

The `scan_files` fact module is the only module that accepts parameters, passed via `extra_vars` on the scan job template.

```
scan_file_paths: '/tmp/'
scan_use_checksum: true
scan_use_recursive: true
```

- The `scan_file_paths` parameter may have multiple settings (such as `/tmp/` or `/var/log`).
- The `scan_use_checksum` and `scan_use_recursive` parameters may also be set to false or omitted. An omission is the same as a false setting.

Scan job templates should enable `become` and use credentials for which `become` is a possibility. You can enable become by checking the **Enable Privilege Escalation** from the Options menu:



Options

☑ Privilege Escalation

□ Provisioning Callbacks

□ Enable Webhook

□ Concurrent Jobs

□ Enable Fact Storage

20.9.2 Supported OSes for `scan_facts.yml`

If you use the `scan_facts.yml` playbook with use fact cache, ensure that your OS is supported:

- Red Hat Enterprise Linux 5, 6, & 7
- Ubuntu 16.04 (Support for Unbuntu is deprecated and will be removed in a future release)
- OEL 6 & 7
- SLES 11 & 12
- Debian 6, 7, 8
- Fedora 22, 23, 24
- Amazon Linux 2016.03

- Windows Server 2008 and later

Note that some of these operating systems may require initial configuration in order to be able to run python and/or have access to the python packages (such as `python-apt`) that the scan modules depend on.

20.9.3 Pre-scan Setup

The following are examples of playbooks that configure certain distributions so that scan jobs can be run against them.

Bootstrap Ubuntu (16.04)

```
---

- name: Get Ubuntu 16, and on ready
 hosts: all
 sudo: yes
 gather_facts: no

 tasks:

 - name: install python-simplejson
   raw: sudo apt-get -y update
   raw: sudo apt-get -y install python-simplejson
   raw: sudo apt-get install python-apt
```

Bootstrap Fedora (23, 24)

```
---

- name: Get Fedora ready
 hosts: all
 sudo: yes
 gather_facts: no

 tasks:

 - name: install python-simplejson
   raw: sudo dnf -y update
   raw: sudo dnf -y install python-simplejson
   raw: sudo dnf -y install rpm-python
```

20.9.4 Custom Fact Scans

A playbook for a custom fact scan is similar to the example of the Fact Scan Playbook above. As an example, a playbook that only uses a custom `scan_foo` Ansible fact module would look like this:

scan_custom.yml:

```
- hosts: all
  gather_facts: false
  tasks:
    - scan_foo:
```

scan_foo.py:

```
def main():
    module = AnsibleModule(
        argument_spec = dict())

    foo = [
      {
        "hello": "world"
      },
      {
        "foo": "bar"
      }
    ]
    results = dict(ansible_facts=dict(foo=foo))
    module.exit_json(**results)

main()
```

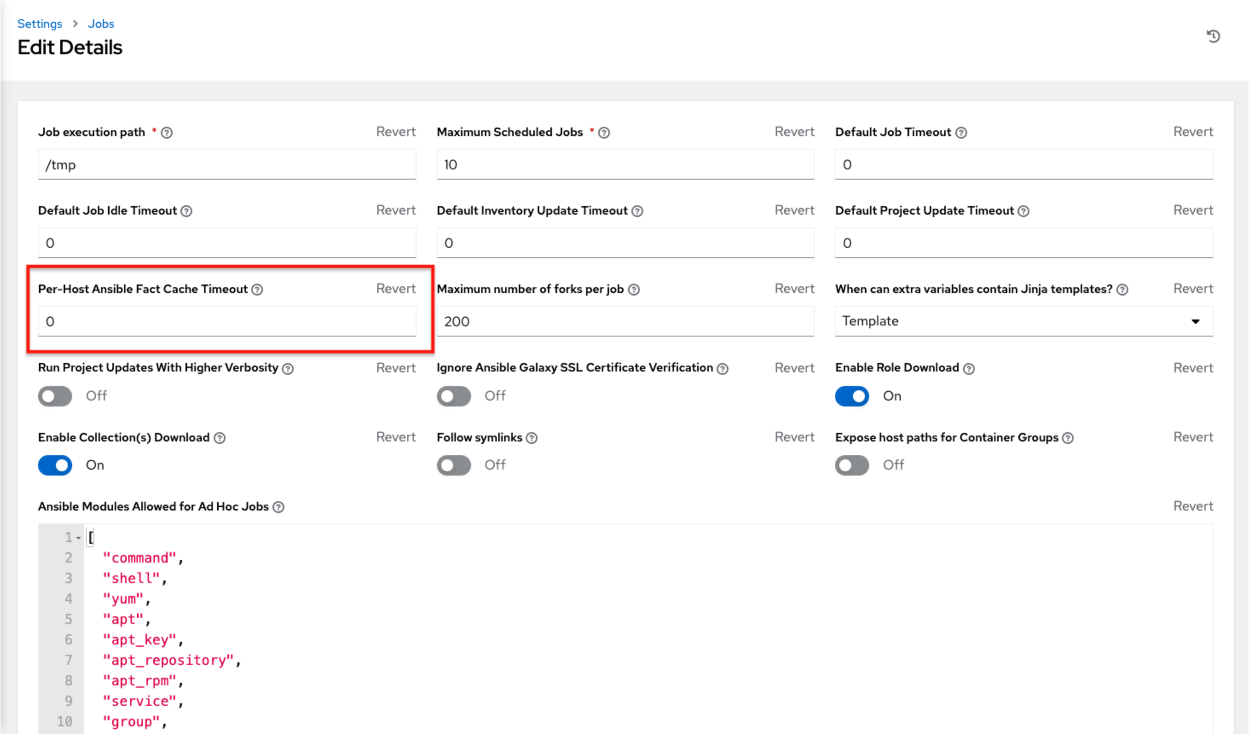
To use a custom fact module, ensure that it lives in the `/library/` subdirectory of the Ansible project used in the scan job template. This fact scan module is very simple, returning a hard-coded set of facts:

```
[
   {
     "hello": "world"
   },
   {
     "foo": "bar"
   }
 ]
```

Refer to the Module Provided 'Facts' section of the Ansible documentation for more information.

20.10 Fact Caching

Automation controller can store and retrieve facts on a per-host basis through an Ansible Fact Cache plugin. This behavior is configurable on a per-job template basis. Fact caching is turned off by default but can be enabled to serve fact requests for all hosts in an inventory related to the job running. This allows you to use job templates with `--limit` while still having access to the entire inventory of host facts. A global timeout setting that the plugin enforces per-host, can be specified (in seconds) through the Jobs settings menu:



Settings > Jobs

Edit Details

Job execution path * — Revert /tmp	Maximum Scheduled Jobs * — Revert 10	Default Job Timeout — Revert 0
Default Job Idle Timeout — Revert 0	Default Inventory Update Timeout — Revert 0	Default Project Update Timeout — Revert 0
Per-Host Ansible Fact Cache Timeout — Revert 0	Maximum number of forks per job — Revert 200	When can extra variables contain Jinja templates? — Revert Template
Run Project Updates With Higher Verbosity — Revert Off	Ignore Ansible Galaxy SSL Certificate Verification — Revert Off	Enable Role Download — Revert On
Enable Collection(s) Download — Revert On	Follow symlinks — Revert Off	Expose host paths for Container Groups — Revert Off

Ansible Modules Allowed for Ad Hoc Jobs — Revert

```
[
  "command",
  "shell",
  "yum",
  "apt",
  "apt_key",
  "apt_repository",
  "apt_rpm",
  "service",
  "group",
```

Upon launching a job that uses fact cache (use_fact_cache=True), the controller will store all ansible_facts associated with each host in the inventory associated with the job. The Ansible Fact Cache plugin that ships with automation controller will only be enabled on jobs with fact cache enabled (use_fact_cache=True).

When a job that has fact cache enabled (use_fact_cache=True) finishes running, the controller will restore all records for the hosts in the inventory. Any records with update times *newer* than the currently stored facts per-host will be updated in the database.

New and changed facts will be logged via the controller's logging facility. Specifically, to the system_tracking namespace or logger. The logging payload will include the fields:

- host_name
- inventory_id
- ansible_facts

where ansible_facts is a dictionary of all Ansible facts for host_name in the controller inventory, inventory_id.

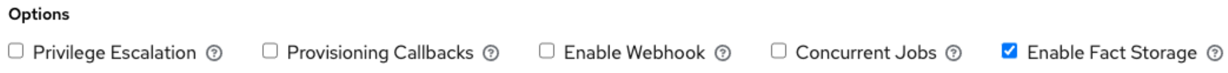
Note: If a hostname includes a forward slash (/), fact cache will not work for that host. If you have an inventory with 100 hosts and one host has a / in the name, 99 of those hosts will still collect facts.

20.10.1 Benefits of Fact Caching

Fact caching saves a significant amount of time over running fact gathering. If you have a playbook in a job that runs against a thousand hosts and forks, you could easily spend 10 minutes gathering facts across all of those hosts. But if you run a job on a regular basis, the first run of it caches these facts and the next run will just pull them from the database. This cuts the runtime of jobs against large inventories, including Smart Inventories, by an enormous magnitude.

Note: Do not modify the `ansible.cfg` file to apply fact caching. Custom fact caching could conflict with the controller's fact caching feature. It is recommended to use the fact caching module that comes with automation controller. Fact caching is not supported for isolated nodes.

You can choose to use cached facts in your job by enabling it in the **Options** field of the Job Templates window.



Options

□ Privilege Escalation ⓘ □ Provisioning Callbacks ⓘ □ Enable Webhook ⓘ □ Concurrent Jobs ⓘ ☑ Enable Fact Storage ⓘ

To clear facts, you need to run the Ansible `clear_facts` meta task. Below is an example playbook that uses the Ansible `clear_facts` meta task.

```
- hosts: all
  gather_facts: false
  tasks:
    - name: Clear gathered facts from all currently targeted hosts
      meta: clear_facts
```

You can find the API endpoint for fact caching at:

```
http://<controller server name>/api/v2/hosts/x/ansible_facts
```

20.11 Utilizing Cloud Credentials

Cloud Credentials can be used when syncing a respective cloud inventory. Cloud Credentials may also be associated with a Job Template and included in the runtime environment for use by a playbook. Cloud Credentials currently supported:

- *OpenStack*
- *Amazon Web Services*
- *Google*
- *Azure*
- *VMware*

20.11.1 OpenStack

The sample playbook below invokes the nova_compute Ansible OpenStack cloud module and requires credentials to do anything meaningful, and specifically requires the following information: auth_url, username, password, and project_name. These fields are made available to the playbook via the environmental variable OS_CLIENT_CONFIG_FILE, which points to a YAML file written by the controller based on the contents of the cloud credential. This sample playbook loads the YAML file into the Ansible variable space.

OS_CLIENT_CONFIG_FILE example:

```
clouds:
  devstack:
    auth:
      auth_url: http://devstack.yoursite.com:5000/v2.0/
      username: admin
      password: your_password_here
      project_name: demo
```

Playbook example:

```
- hosts: all
  gather_facts: false
  vars:
    config_file: "{{ lookup('env', 'OS_CLIENT_CONFIG_FILE') }}"
    nova_tenant_name: demo
    nova_image_name: "cirros-0.3.2-x86_64-uec"
    nova_instance_name: autobot
    nova_instance_state: 'present'
    nova_flavor_name: m1.nano

    nova_group:
      group_name: antarctica
      instance_name: deceptacon
      instance_count: 3
  tasks:
    - debug: msg="{{ config_file }}"
    - stat: path="{{ config_file }}"
      register: st
    - include_vars: "{{ config_file }}"
      when: st.stat.exists and st.stat.isreg

    - name: "Print out clouds variable"
      debug: msg="{{ clouds|default('No clouds found') }}"

    - name: "Setting nova instance state to: {{ nova_instance_state }}"
      local_action:
        module: nova_compute
        login_username: "{{ clouds.devstack.auth.username }}"
        login_password: "{{ clouds.devstack.auth.password }}"
```

20.11.2 Amazon Web Services

Amazon Web Services cloud credentials are exposed as the following environment variables during playbook execution (in the job template, choose the cloud credential needed for your setup):

- AWS_ACCESS_KEY_ID
- AWS_SECRET_ACCESS_KEY

All of the AWS modules will implicitly use these credentials when run via the controller without having to set the aws_access_key_id or aws_secret_access_key module options.

20.11.3 Google

Google cloud credentials are exposed as the following environment variables during playbook execution (in the job template, choose the cloud credential needed for your setup):

- GCE_EMAIL
- GCE_PROJECT
- GCE_CREDENTIALS_FILE_PATH

All of the Google modules will implicitly use these credentials when run via the controller without having to set the service_account_email, project_id, or pem_file module options.

20.11.4 Azure

Azure cloud credentials are exposed as the following environment variables during playbook execution (in the job template, choose the cloud credential needed for your setup):

- AZURE_SUBSCRIPTION_ID
- AZURE_CERT_PATH

All of the Azure modules implicitly use these credentials when run via the controller without having to set the subscription_id or management_cert_path module options.

20.11.5 VMware

VMware cloud credentials are exposed as the following environment variables during playbook execution (in the job template, choose the cloud credential needed for your setup):

- VMWARE_USER
- VMWARE_PASSWORD
- VMWARE_HOST

The sample playbook below demonstrates usage of these credentials:

```
- vsphere_guest:
    vcenter_hostname: "{{ lookup('env', 'VMWARE_HOST') }}"
    username: "{{ lookup('env', 'VMWARE_USER') }}"
    password: "{{ lookup('env', 'VMWARE_PASSWORD') }}"
    guest: newvm001
    from_template: yes
    template_src: linuxTemplate
```

(continues on next page)

(continued from previous page)

```
    cluster: MainCluster
    resource_pool: "/Resources"
    vm_extra_config:
      folder: MyFolder
```

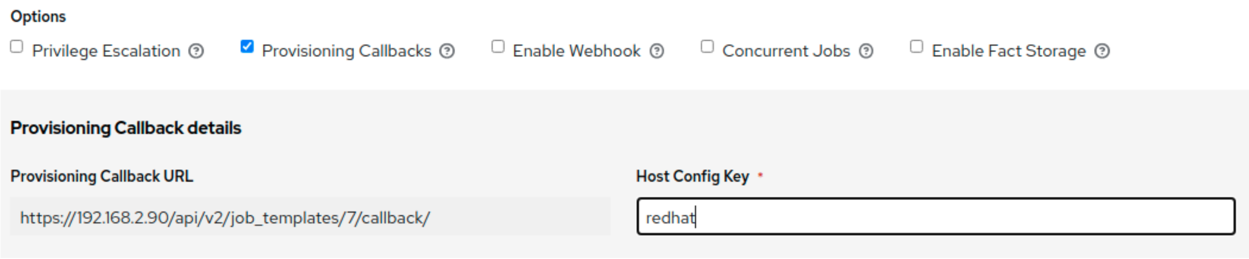
20.12 Provisioning Callbacks

Provisioning callbacks are a feature of Automation Controller that allow a host to initiate a playbook run against itself, rather than waiting for a user to launch a job to manage the host from the Automation Controller console. Please note that provisioning callbacks are *only* used to run playbooks on the calling host. Provisioning callbacks are meant for cloud bursting (i.e. new instances with a need for client to server communication for configuration (such as transmitting an authorization key)), not to run a job against another host. This provides for automatically configuring a system after it has been provisioned by another system (such as AWS auto-scaling, or a OS provisioning system like kickstart or preseed) or for launching a job programmatically without invoking the Automation Controller API directly. The Job Template launched only runs against the host requesting the provisioning.

Frequently this would be accessed via a firstboot type script, or from cron.

To enable callbacks, check the *Provisioning Callbacks* checkbox in the Job Template. This displays the **Provisioning Callback URL** for this job template.

Note: If you intend to use Automation Controller's provisioning callback feature with a dynamic inventory, Update on Launch should be set for the inventory group used in the Job Template.



Options

☐ Privilege Escalation ☑ Provisioning Callbacks ☐ Enable Webhook ☐ Concurrent Jobs ☐ Enable Fact Storage

Provisioning Callback details

Provisioning Callback URL: https://192.168.2.90/api/v2/job_templates/7/callback/

Host Config Key *: redhat

Callbacks also require a Host Config Key, to ensure that foreign hosts with the URL cannot request configuration. Please provide a custom value for Host Config Key. The host key may be reused across multiple hosts to apply this job template against multiple hosts. Should you wish to control what hosts are able to request configuration, the key may be changed at any time.

To callback manually via REST, look at the callback URL in the UI, which is of the form:

```
https://<CONTROLLER_SERVER_NAME>/api/v2/job_templates/7/callback/
```

The '7' in this sample URL is the job template ID in Automation Controller.

The request from the host must be a POST. Here is an example using curl (all on a single line):

```
curl -k -f -i -H 'Content-Type:application/json' -XPOST -d '{"host_config_key":
↪"redhat"}' \
                  https://<CONTROLLER_SERVER_NAME>/api/v2/job_templates/7/callback/
```

The requesting host must be defined in your inventory for the callback to succeed. If Automation Controller fails to locate the host either by name or IP address in one of your defined inventories, the request is denied. When running a Job Template in this way, the host initiating the playbook run against itself must be in the inventory. If the host is missing from the inventory, the Job Template will fail with a "No Hosts Matched" type error message.

Note: If your host is not in inventory and `Update on Launch` is set for the inventory group, Automation Controller attempts to update cloud based inventory source before running the callback.

Successful requests result in an entry on the Jobs tab, where the results and history can be viewed.

While the callback can be accessed via REST, the suggested method of using the callback is to use one of the example scripts that ships with Automation Controller - `/usr/share/awx/request_tower_configuration.sh` (Linux/UNIX) or `/usr/share/awx/request_tower_configuration.ps1` (Windows). Usage is described in the source code of the file by passing the `-h` flag, as shown below:

```
./request_tower_configuration.sh -h
Usage: ./request_tower_configuration.sh <options>

Request server configuration from Ansible Tower.

OPTIONS:
 -h      Show this message
 -s      Controller server (e.g. https://ac.example.com) (required)
 -k      Allow insecure SSL connections and transfers
 -c      Host config key (required)
 -t      Job template ID (required)
 -e      Extra variables
```

This script has some intelligence, it knows how to retry commands and is therefore a more robust way to use callbacks than a simple curl request. As written, the script retries once per minute for up to ten minutes.

Note: Please note that this is an example script. You should edit this script if you need more dynamic behavior when detecting failure scenarios, as any non-200 error code may not be a transient error requiring retry.

Most likely you will use callbacks with dynamic inventory in Automation Controller, such as pulling cloud inventory from one of the supported cloud providers. In these cases, along with setting *Update On Launch*, be sure to configure an inventory cache timeout for the inventory source, to avoid hammering of your Cloud's API endpoints. Since the `request_tower_configuration.sh` script polls once per minute for up to ten minutes, a suggested cache invalidation time for inventory (configured on the inventory source itself) would be one or two minutes.

While we recommend against running the `request_tower_configuration.sh` script from a cron job, a suggested cron interval would be perhaps every 30 minutes. Repeated configuration can be easily handled by scheduling in Automation Controller, so the primary use of callbacks by most users is to enable a base image that is bootstrapped into the latest configuration upon coming online. To do so, running at first boot is a better practice. First boot scripts are just simple init scripts that typically self-delete, so you would set up an init script that called a copy of the `request_tower_configuration.sh` script and make that into an autoscaling image.

20.12.1 Passing Extra Variables to Provisioning Callbacks

Just as you can pass `extra_vars` in a regular Job Template, you can also pass them to provisioning callbacks. To pass `extra_vars`, the data sent must be part of the body of the POST request as application/json (as the content type). Use the following JSON format as an example when adding your own `extra_vars` to be passed:

```
'{"extra_vars": {"variable1":"value1","variable2":"value2",...}}'
```

You can also pass extra variables to the Job Template call using `curl`, such as is shown in the following example:

```
root@localhost:~$ curl -f -H 'Content-Type: application/json' -XPOST \
                 -d '{"host_config_key": "redhat", "extra_vars": "{\"foo\": \"bar\"}"}
↪' \
                 https://<CONTROLLER_SERVER_NAME>/api/v2/job_templates/7/callback
```

For more information, refer to Launching Jobs with Curl.

20.13 Extra Variables

Note: `extra_vars` passed to the job launch API are only honored if one of the following is true:

- They correspond to variables in an enabled survey
- `ask_variables_on_launch` is set to True

When you pass survey variables, they are passed as extra variables (`extra_vars`) within the controller. This can be tricky, as passing extra variables to a job template (as you would do with a survey) can override other variables being passed from the inventory and project.

For example, say that you have a defined variable for an inventory for `debug = true`. It is entirely possible that this variable, `debug = true`, can be overridden in a job template survey.

To ensure that the variables you need to pass are not overridden, ensure they are included by redefining them in the survey. Keep in mind that extra variables can be defined at the inventory, group, and host levels.

If specifying the `ALLOW_JINJA_IN_EXTRA_VARS` parameter, refer to the Controller Tips and Tricks section of the *Automation Controller Administration Guide* to configure it in the Jobs Settings screen of the controller UI.

Note: The Job Template extra variables dictionary is merged with the Survey variables.

Here are some simplified examples of extra_vars in YAML and JSON formats:

The configuration in YAML format:

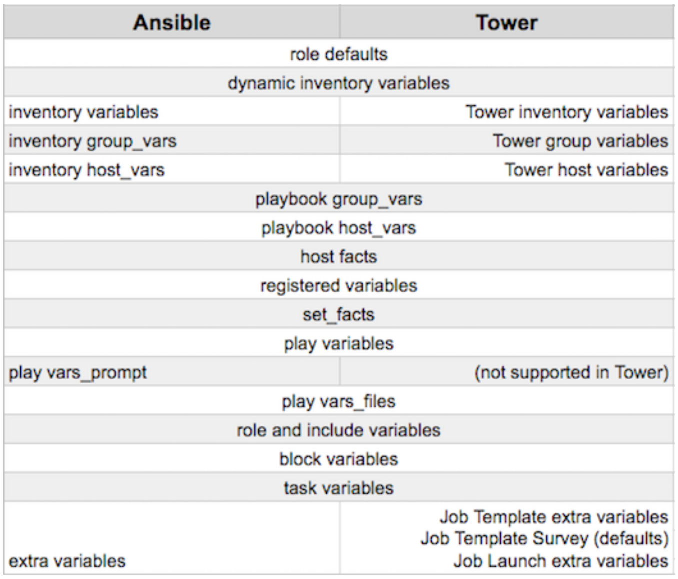
```
launch_to_orbit: true
satellites:
  - sputnik
  - explorer
  - satcom
```

The configuration in JSON format:

```
{
  "launch_to_orbit": true,
  "satellites": ["sputnik", "explorer", "satcom"]
}
```

The following table notes the behavior (hierarchy) of variable precedence in automation controller as it compares to variable precedence in Ansible.

Automation Controller Variable Precedence Hierarchy (last listed wins)



Ansible	Tower
role defaults	
dynamic inventory variables	
inventory variables	Tower inventory variables
inventory group_vars	Tower group variables
inventory host_vars	Tower host variables
playbook group_vars	
playbook host_vars	
host facts	
registered variables	
set_facts	
play variables	
play vars_prompt	(not supported in Tower)
play vars_files	
role and include variables	
block variables	
task variables	
extra variables	Job Template extra variables Job Template Survey (defaults) Job Launch extra variables

20.13.1 Relaunching Job Templates

Instead of manually relaunching a job, a relaunch is denoted by setting `launch_type` to `relaunch`. The relaunch behavior deviates from the launch behavior in that it **does not** inherit `extra_vars`.

Job relaunching does not go through the inherit logic. It uses the same `extra_vars` that were calculated for the job being relaunched.

For example, say that you launch a Job Template with no `extra_vars` which results in the creation of a Job called **j1**. Next, say that you edit the Job Template and add in some `extra_vars` (such as adding `"{ "hello": "world" }"`).

Relaunching **j1** results in the creation of **j2**, but because there is no inherit logic and **j1** had no `extra_vars`, **j2** will not have any `extra_vars`.

To continue upon this example, if you launched the Job Template with the `extra_vars` you added after the creation of **j1**, the relaunch job created (**j3**) will include the `extra_vars`. And relaunching **j3** results in the creation of **j4**, which would also include `extra_vars`.

CHAPTER

TWENTYONE

JOB SLICING

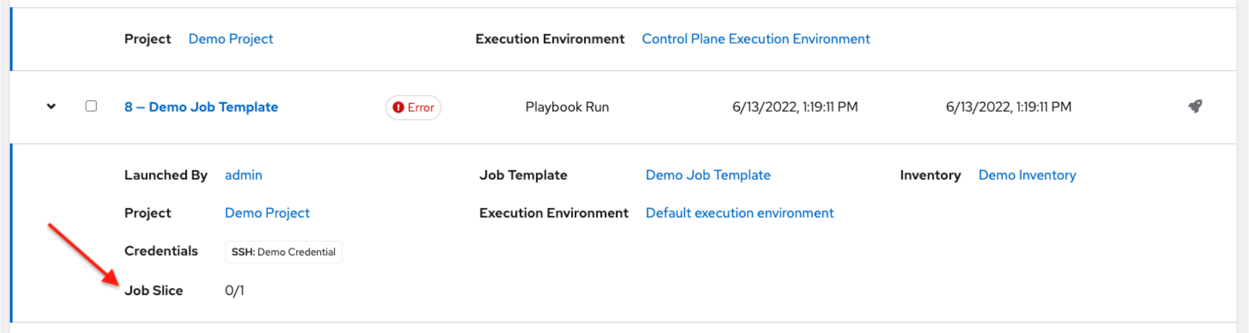
A *sliced job* refers to the concept of a distributed job. Distributed jobs are used for running a job across a very large number of hosts, allowing you to run multiple ansible-playbooks, each on a subset of an inventory, that can be scheduled in parallel across a cluster.

By default, Ansible runs jobs from a single control instance. For jobs that do not require cross-host orchestration, job slicing takes advantage of automation controller's ability to distribute work to multiple nodes in a cluster. Job slicing works by adding a Job Template field `job_slice_count`, which specifies the number of jobs into which to slice the Ansible run. When this number is greater than 1, automation controller will generate a workflow from a job template instead of a job. The inventory will be distributed evenly amongst the slice jobs. The workflow job is then started, and proceeds as though it were a normal workflow. When launching a job, the API will return either a job resource (if `job_slice_count = 1`) or a workflow job resource. The corresponding User Interface will redirect to the appropriate screen to display the status of the run.

21.1 Job slice considerations

Consider the following when setting up job slices:

- A sliced job creates a workflow job, and then that creates jobs.
- A job slice consists of a job template, an inventory, and a slice count.
- When executed, a sliced job splits each inventory into a number of "slice size" chunks. It then queues jobs of ansible-playbook runs on each chunk of the appropriate inventory. The inventory fed into ansible-playbook is a pared-down version of the original inventory that only contains the hosts in that particular slice. The completed sliced job that displays on the Jobs list are labeled accordingly, with the number of sliced jobs that have run:



Project Demo Project
Execution Environment Control Plane Execution Environment

8 – Demo Job Template | Error | Playbook Run | 6/13/2022, 1:19:11 PM | 6/13/2022, 1:19:11 PM

Launched By admin
Job Template Demo Job Template
Inventory Demo Inventory
Project Demo Project
Execution Environment Default execution environment
Credentials SSH: Demo Credential
Job Slice 0/1

- These sliced jobs follow normal scheduling behavior (number of forks, queuing due to capacity, assignation to instance groups based on inventory mapping).

Note: Job slicing is intended to scale job executions horizontally. Enabling job slicing on a job template divides an inventory to be acted upon in the number of slices configured at launch time and then starts a job for each slice.

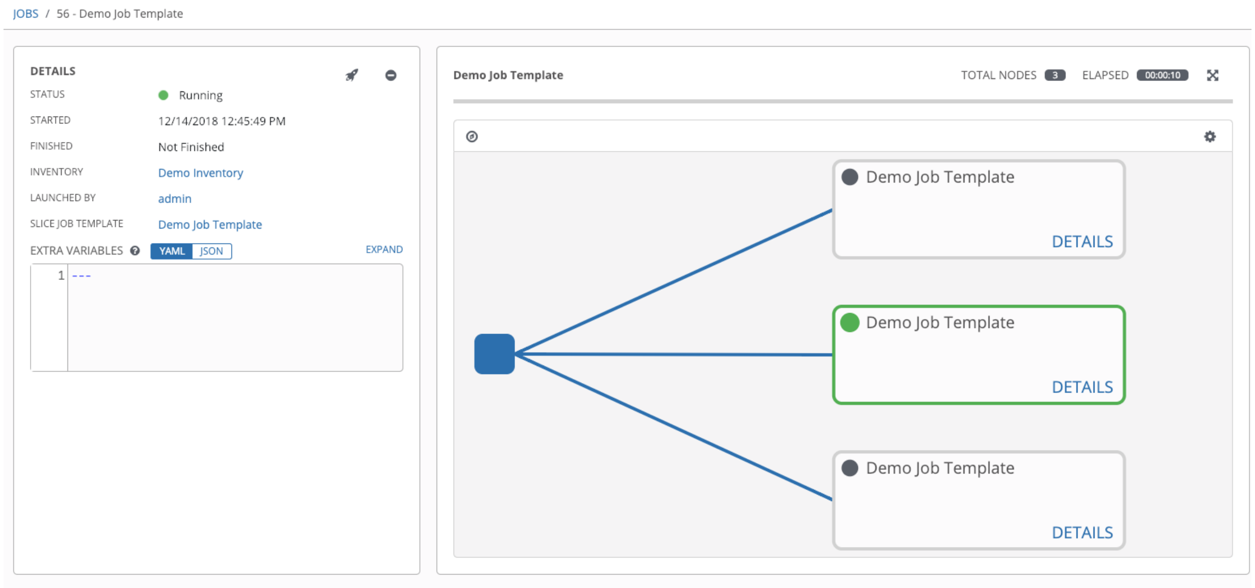
It is expected that the number of slices will be equal to or less than the number of controller nodes. Setting an extremely high number of job slices (e.g., thousands), while allowed, can cause performance degradation as the job scheduler is not designed to schedule simultaneously thousands of workflow nodes, which are what the sliced jobs become.

- Sliced job templates with prompts and/or extra variables behave the same as standard job templates, applying all variables and limits to the entire set of slice jobs in the resulting workflow job. However, when passing a limit to a Sliced Job, if the limit causes slices to have no hosts assigned, those slices will fail, causing the overall job to fail.
- A job slice job status of a distributed job is calculated in the same manner as workflow jobs; failure if there are any unhandled failures in its sub-jobs.

Warning: Any job that intends to orchestrate across hosts (rather than just applying changes to individual hosts) should not be configured as a slice job. Any job that does, may fail, and automation controller will not attempt to discover or account for playbooks that fail when run as slice jobs.

21.2 Job slice execution behavior

When jobs are sliced, they can run on any node and some may not run at the same time (insufficient capacity in the system, for example). When slice jobs are running, job details display the workflow and job slice(s) currently running, as well as a link to view their details individually.



JOBS / 56 - Demo Job Template

DETAILS

STATUS ● Running

STARTED 12/14/2018 12:45:49 PM

FINISHED Not Finished

INVENTORY Demo Inventory

LAUNCHED BY admin

SLICE JOB TEMPLATE Demo Job Template

EXTRA VARIABLES YAML JSON EXPAND

```
---
```

Demo Job Template TOTAL NODES 3 ELAPSED 00:00:10

By default, job templates are not normally configured to execute simultaneously (`allow_simultaneous` must be checked in the API or **Enable Concurrent Jobs** in the UI). Slicing overrides this behavior and implies `allow_simultaneous` even if that setting is unchecked. See *Job Templates* for information on how to specify this, as well as the number of job slices on your job template configuration.

The *Job Templates* section provides additional detail on performing the following operations in the User Interface:

- Launch workflow jobs with a job template that has a slice number greater than one

- Cancel the whole workflow or individual jobs after launching a slice job template
- Relaunch the whole workflow or individual jobs after slice jobs finish running
- View the details about the workflow and slice jobs after a launching a job template
- Search slice jobs specifically after you create them (see subsequent section, *Search job slices*)

21.3 Search job slices

To make it easier to find slice jobs, use the Search functionality to apply a search filter to:

- job lists to show only slice jobs
- job lists to show only parent workflow jobs of job slices
- job templates lists to only show job templates that produce slice jobs

To show only slice jobs in job lists, as with most cases, you can filter either on the type (jobs here) or `unified_jobs`:

```
/api/v2/jobs/?job_slice_count__gt=1
```

To show only parent workflow jobs of job slices:

```
/api/v2/workflow_jobs/?job_template__isnull=false
```

To show only job templates that produce slice jobs:

```
/api/v2/job_templates/?job_slice_count__gt=1
```

CHAPTER

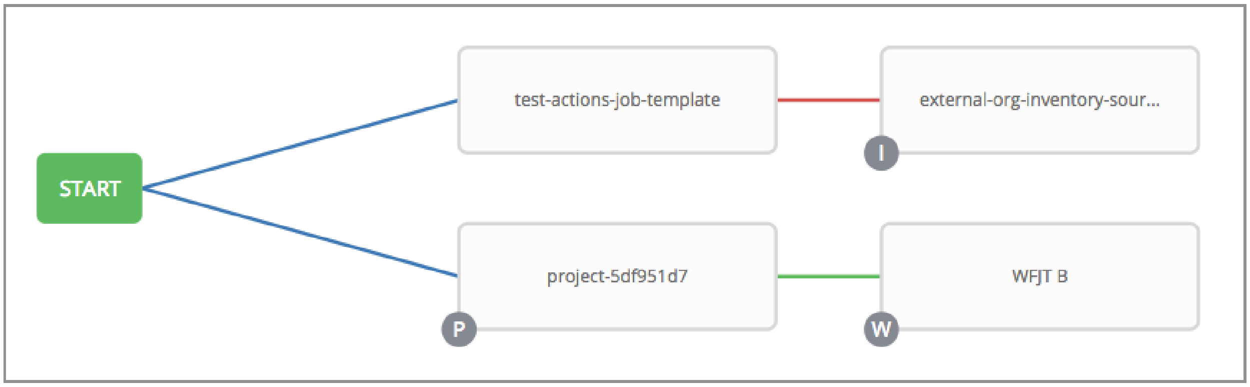
TWENTYTWO

WORKFLOWS

Workflows allow you to configure a sequence of disparate job templates (or workflow templates) that may or may not share inventory, playbooks, or permissions. However, workflows have 'admin' and 'execute' permissions, similar to job templates. A workflow accomplishes the task of tracking the full set of jobs that were part of the release process as a single unit.

Job or workflow templates are linked together using a graph-like structure called nodes. These nodes can be jobs, project syncs, or inventory syncs. A template can be part of different workflows or used multiple times in the same workflow. A copy of the graph structure is saved to a workflow job when you launch the workflow.

The example below shows a workflow that contains all three, as well as a workflow job template:

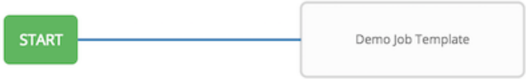


As the workflow runs, jobs are spawned from the node's linked template. Nodes linking to a job template which has prompt-driven fields (`job_type`, `job_tags`, `skip_tags`, `limit`) can contain those fields, and will not be prompted on launch. Job templates with promptable credential and/or inventory, WITHOUT defaults, will not be available for inclusion in a workflow.

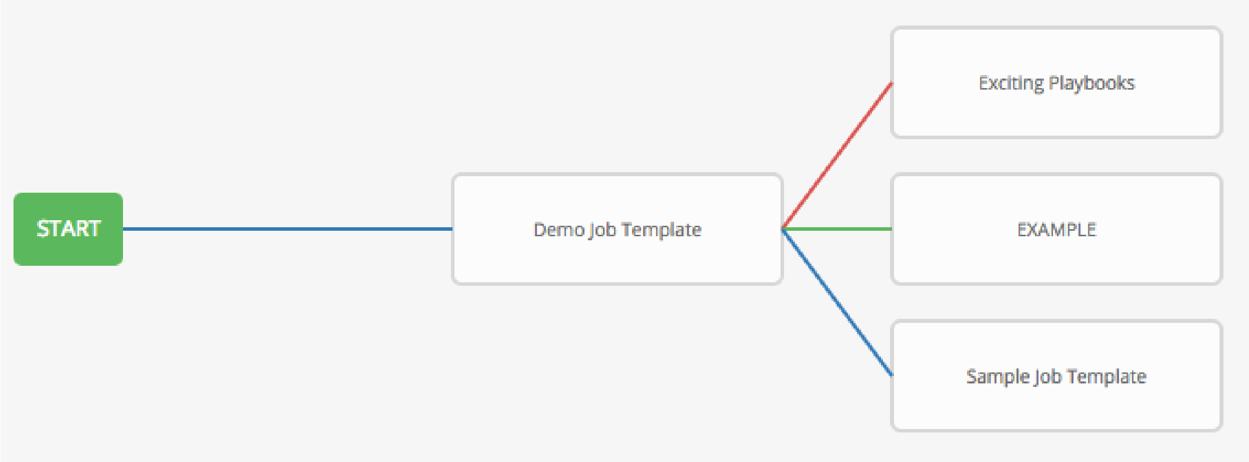
22.1 Workflow scenarios and considerations

Consider the following scenarios for building workflows:

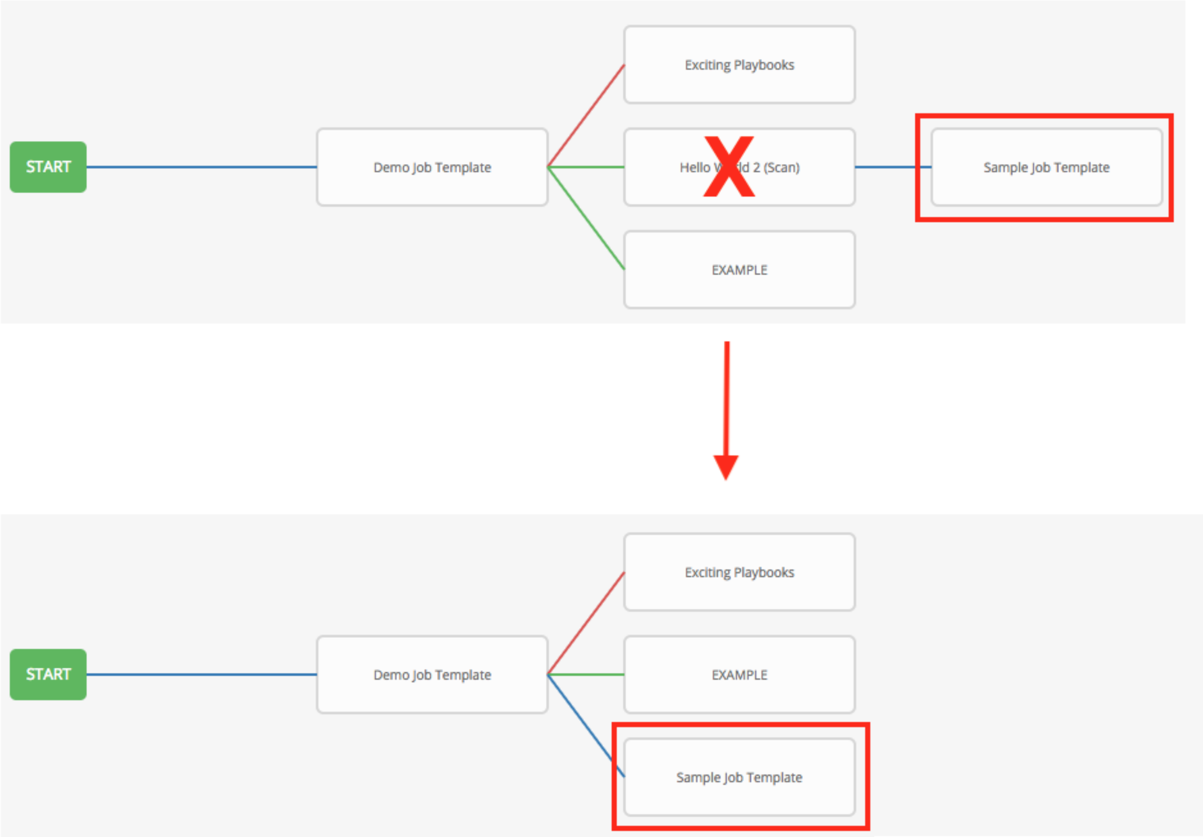
- A root node is set to ALWAYS by default and it not editable.



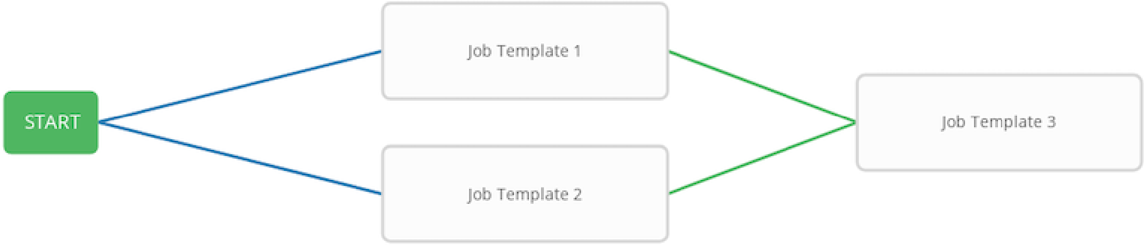
- A node can have multiple parents and children may be linked to any of the states of success, failure, or always. If always, then the state is neither success or failure. States apply at the node level, not at the workflow job template level. A workflow job will be marked as successful unless it is canceled or encounters an error.



- If you remove a job or workflow template within the workflow, the node(s) previously connected to those deleted, automatically get connected upstream and retains its edge type as in the example below:

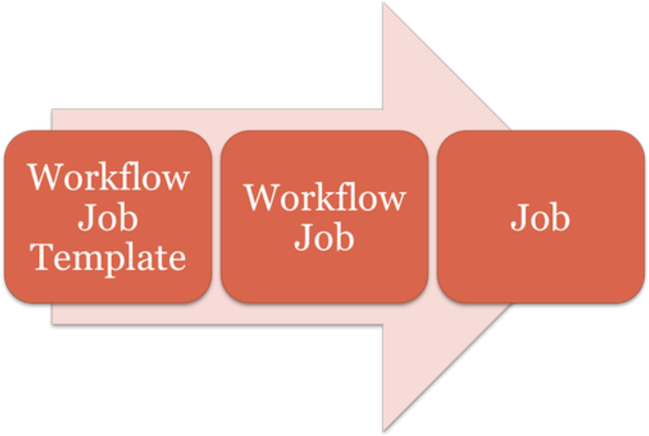


- You could have a convergent workflow, where multiple jobs converge into one. In this scenario, any of the jobs or all of them must complete before the next one runs, as shown in the example below:



In the example provided, automation controller runs the first two job templates in parallel. When they both finish and succeed as specified, the 3rd downstream (*convergence node*), will trigger.

- Prompts for inventory and surveys will apply to workflow nodes in workflow job templates.
- If you launch from the API, running a `get` command displays a list of warnings and highlights missing components. The basic workflow for a workflow job template is illustrated below.



- It is possible to launch several workflows simultaneously, and set a schedule for when to launch them. You can set notifications on workflows, such as when a job completes, similar to that of job templates.

Note: Job slicing is intended to scale job executions horizontally. Enabling job slicing on a job template divides an inventory to be acted upon in the number of slices configured at launch time and then starts a job for each slice.

It is expected that the number of slices will be equal to or less than the number of controller nodes. Setting an extremely high number of job slices (e.g., thousands), while allowed, can cause performance degradation as the job scheduler is not designed to schedule simultaneously thousands of workflow nodes, which are what the sliced jobs become.

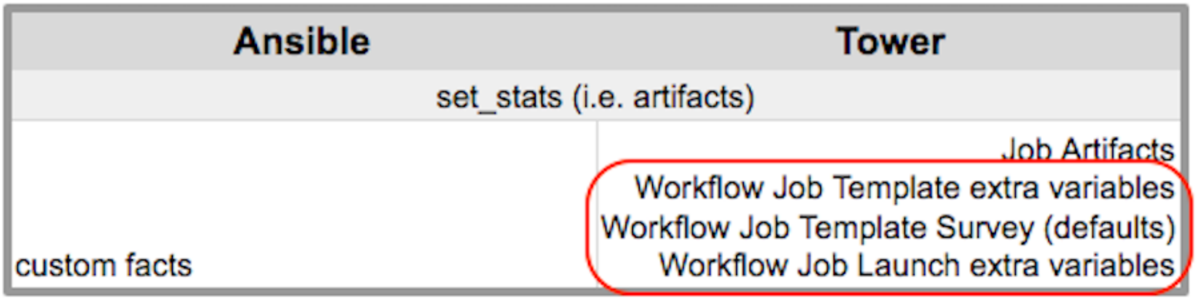
- You can build a recursive workflow, but if automation controller detects an error, it will stop at the time the nested workflow attempts to run.
- Artifacts gathered in jobs in the sub-workflow will be passed to downstream nodes.
- An inventory can be set at the workflow level, or prompt for inventory on launch.
- When launched, all job templates in the workflow that have `ask_inventory_on_launch=true` will use the workflow level inventory.
- Job templates that do not prompt for inventory will ignore the workflow inventory and run against their own inventory.

- If a workflow prompts for inventory, schedules and other workflow nodes may provide the inventory.
- In a workflow convergence scenario, `set_stats` data will be merged in an undefined way, so it is recommended that you set unique keys.

22.2 Extra Variables

Also similar to job templates, workflows use surveys to specify variables to be used in the playbooks in the workflow, called extra_vars. Survey variables are combined with extra_vars defined on the workflow job template, and saved to the workflow job extra_vars. extra_vars in the workflow job are combined with job template variables when spawning jobs within the workflow.

Workflows utilize the same behavior (hierarchy) of variable precedence as Job Templates with the exception of three additional variables. Refer to the Variable Precedence Hierarchy in the *Extra Variables* section of the Job Templates chapter of this guide. The three additional variables include:



Ansible	Tower
set_stats (i.e. artifacts)	
custom facts	Job Artifacts Workflow Job Template extra variables Workflow Job Template Survey (defaults) Workflow Job Launch extra variables

Workflows included in a workflow will follow the same variable precedence - they will only inherit variables if they are specifically prompted for, or defined as part of a survey.

In addition to the workflow `extra_vars`, jobs and workflows ran as part of a workflow can inherit variables in the artifacts dictionary of a parent job in the workflow (also combining with ancestors further upstream in its branch). These can be defined by the `set_stats` Ansible module.

If you use the `set_stats` module in your playbook, you can produce results that can be consumed downstream by another job, for example, notify users as to the success or failure of an integration run. In this example, there are two playbooks that can be combined in a workflow to exercise artifact passing:

- **invoke_set_stats.yml**: first playbook in the workflow:

```
---
- hosts: localhost
  tasks:
    - name: "Artifact integration test results to the web"
      local_action: 'shell curl -F "file=@integration_results.txt" https://file.io'
      register: result

    - name: "Artifact URL of test results to Workflows"
      set_stats:
        data:
          integration_results_url:  "{{ (result.stdout|from_json).link }}"
```

- **use_set_stats.yml**: second playbook in the workflow

```
---
- hosts: localhost
```

(continues on next page)

(continued from previous page)

```
  tasks:
    - name: "Get test results from the web"
      uri:
        url: "{{ integration_results_url }}"
        return_content: true
      register: results

    - name: "Output test results"
      debug:
        msg: "{{ results.content }}"
```

The set_stats module processes this workflow as follows:

1. The contents of an integration results (example: integration_results.txt below) is first uploaded to the web.

```
the tests are passing!
```

2. Through the **invoke_set_stats** playbook, set_stats is then invoked to artifact the URL of the uploaded integration_results.txt into the Ansible variable “integration_results_url”.
3. The second playbook in the workflow consumes the Ansible extra variable “integration_results_url”. It calls out to the web using the uri module to get the contents of the file uploaded by the previous Job Template Job. Then, it simply prints out the contents of the gotten file.

Note: For artifacts to work, keep the default setting, per_host = False in the set_stats module.

22.3 Workflow States

The workflow job can have the following states (no Failed state):

- Waiting
- Running
- Success (finished)
- Cancel
- Error
- Failed

In the workflow scheme, canceling a job cancels the branch, while canceling the workflow job cancels the entire workflow.

22.4 Role-Based Access Controls

To edit and delete a workflow job template, you must have the admin role. To create a workflow job template, you must be an organization admin or a system admin. However, you can run a workflow job template that contains job templates you don't have permissions for. Similar to projects, organization admins can create a blank workflow and then grant an 'admin_role' to a low-level user, after which they can go about delegating more access and building the graph. You must have execute access to a job template to add it to a workflow job template.

Other tasks such as the ability to make a duplicate copy and re-launch a workflow can also be performed, depending on what kinds of permissions are granted to a particular user. Generally, you should have permissions to all the resources used in a workflow (like job templates) before relaunching or making a copy.

For more information on performing the tasks described in this section, refer to the Administration Guide.

CHAPTER TWENTYTHREE

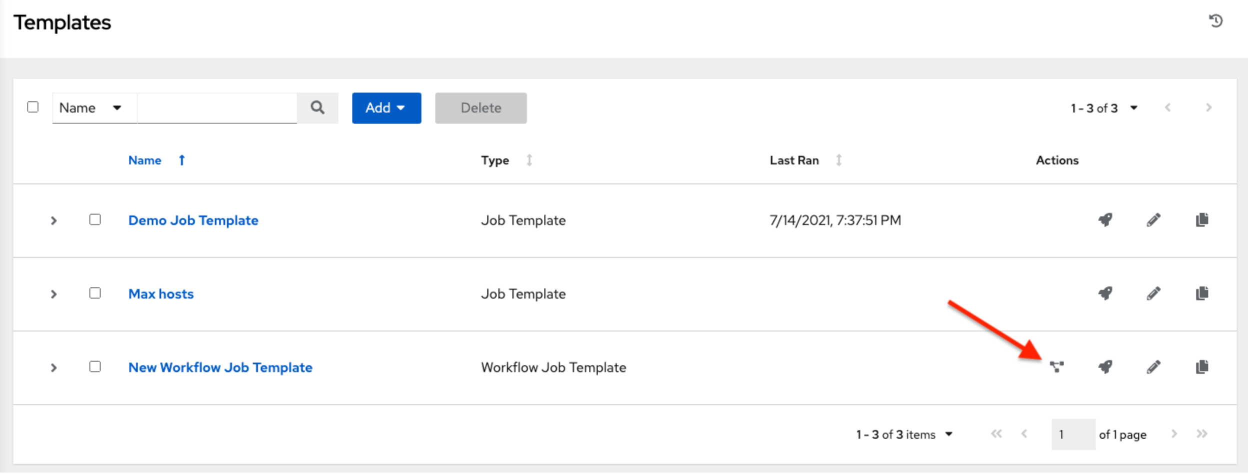
WORKFLOW JOB TEMPLATES

A *workflow job template* links together a sequence of disparate resources that accomplishes the task of tracking the full set of jobs that were part of the release process as a single unit. These resources may include:

- job templates
- workflow templates
- project syncs
- inventory source syncs

The **Templates** menu opens a list of the workflow and job templates that are currently available. The default view is collapsed (Compact), showing the template name, template type, and the statuses of the jobs that ran using that template, but you can click **Expanded** to view more information. This list is sorted alphabetically by name, but you can sort by other criteria, or search by various fields and attributes of a template. From this screen, you can launch (🚀), edit (✎), and copy (⧉) a workflow job template.

Only workflow templates have the Workflow Visualizer icon (⛙) as a shortcut for accessing the workflow editor.

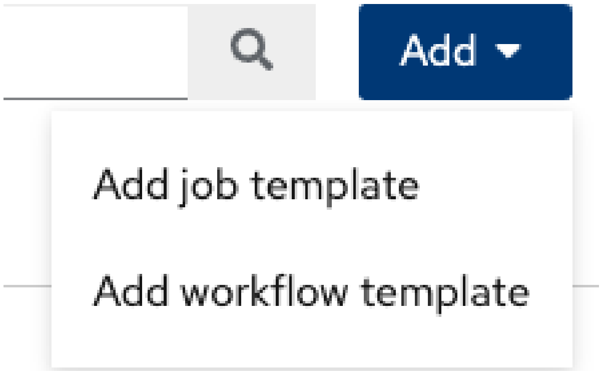


		Name ↑	Type	Last Ran	Actions
›	□	Demo Job Template	Job Template	7/14/2021, 7:37:51 PM	🚀 ✎ ⧉
›	□	Max hosts	Job Template		🚀 ✎ ⧉
›	□	New Workflow Job Template	Workflow Job Template		⛙ 🚀 ✎ ⧉

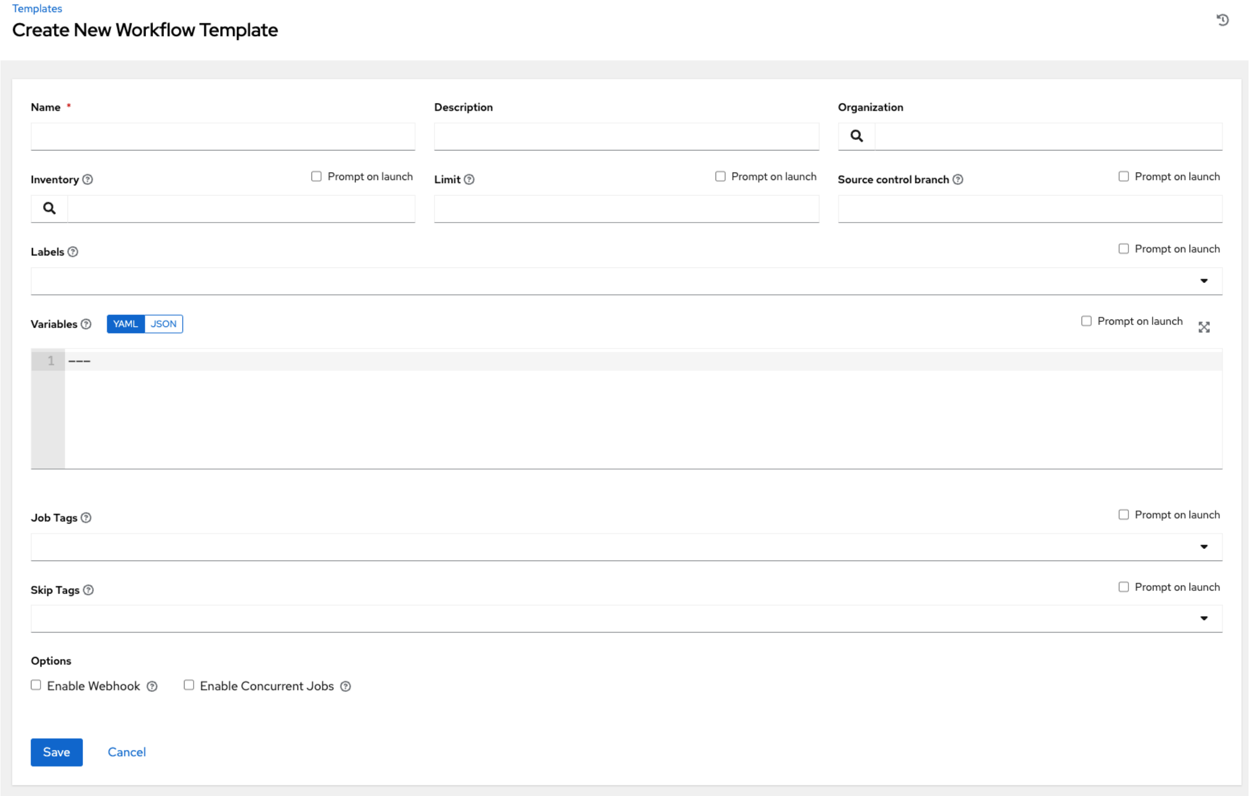
Note: Workflow templates can be used as building blocks for another workflow template. Many parameters in a workflow template allow you to enable **Prompt on Launch** that can be modified at the workflow job template level, and do not affect the values assigned at the individual workflow template level. For instructions, see the *Workflow Visualizer* section.

23.1 Create a Workflow Template

To create a new workflow job template:



1. Click the button then select **Workflow Template** from the menu list.



2. Enter the appropriate details into the following fields:

Note: If a field has the **Prompt on launch** checkbox selected, launching the workflow template, or when the workflow template is used within another workflow template, it will prompt for the value for that field upon launch. Most prompted values will override any values set in the workflow job template; exceptions are noted below.

Field	Options	Prompt on Launch
Name	Enter a name for the job.	N/A
Description	Enter an arbitrary description as appropriate (optional).	N/A
Organization	Choose the organization to be used with this template from the organizations available to the currently logged in user.	N/A
Inventory	Optionally choose the inventory to be used with this template from the inventories available to the currently logged in user.	Yes
Limit	A host pattern to further constrain the list of hosts managed or affected by the playbook. Multiple patterns can be separated by colons (`:`). As with core Ansible, `a:b` means “in group a or b”, `a:b:&c` means “in a or b but must be in c”, and `a:!b` means “in a, and definitely not in b”. For more information and examples refer to Patterns in the Ansible documentation.	Yes
Source control branch	Select a branch for the workflow. This branch is applied to all workflow job template nodes that prompt for a branch.	Yes
Labels	• Optionally supply labels that describe this workflow job template, such as “dev” or “test”. Labels can be used to group and filter workflow job templates and completed jobs in the display. • Labels are created when they are added to the workflow template. Labels are associated to a single Organization using the Project that is provided in the workflow template. Members of the Organization can create labels on a workflow template if they have edit permissions (such as an admin role). • Once the workflow template is saved, the labels appear in the WFJT’s Details view. • Click the (✖) beside a label to remove it. When a label is removed, it is no longer associated with that particular workflow template, but it will remain associated with any other jobs or job templates that reference it. • Labels are only applied to the workflow templates not the job template nodes that are used in the workflow.	• Yes. If selected, even if a default value is supplied, you will be prompted upon launch to supply additional labels if needed. • You will not be able to delete existing labels - clicking (⊗) only removes the newly added labels, not existing default labels.

Variables	• Pass extra command line variables to the playbook. This is the “-e” or “–extra-vars” command line parameter for ansible-playbook that is documented in the Ansible documentation at Passing Variables on the Command Line.
	• Provide key/value pairs using either YAML or JSON. These variables have a maximum value of precedence and overrides other variables specified elsewhere. An example value might be:
	`git_branch: production`
	`release_version: 1.5`
		• Yes. If you want to be able to specify `extra_vars` on a schedule, you must select **Prompt on Launch** for **Variables** on the workflow job template, or a enable a survey on the workflow job template, then those answered survey questions become `extra_vars`.
		• For more information about extra variables, refer to *Extra Variables*.

3. **Options**: Specify options for launching this workflow job template, if necessary.

- Check **Enable Webhooks** to turn on the ability to interface with a predefined SCM system web service that is used to launch a workflow job template. Currently supported SCM systems are GitHub and GitLab.

  If you enable webhooks, other fields display, prompting for additional information:

  - **Webhook Service**: Select which service to listen for webhooks from
  - **Webhook Credential**: Optionally, provide a GitHub or GitLab personal access token (PAT) as a credential to use to send status updates back to the webhook service. Before you can select it, the credential must exist. See *Credential Types* to create one.

  Upon **Save**, additional fields populate and the Workflow Visualizer automatically opens.

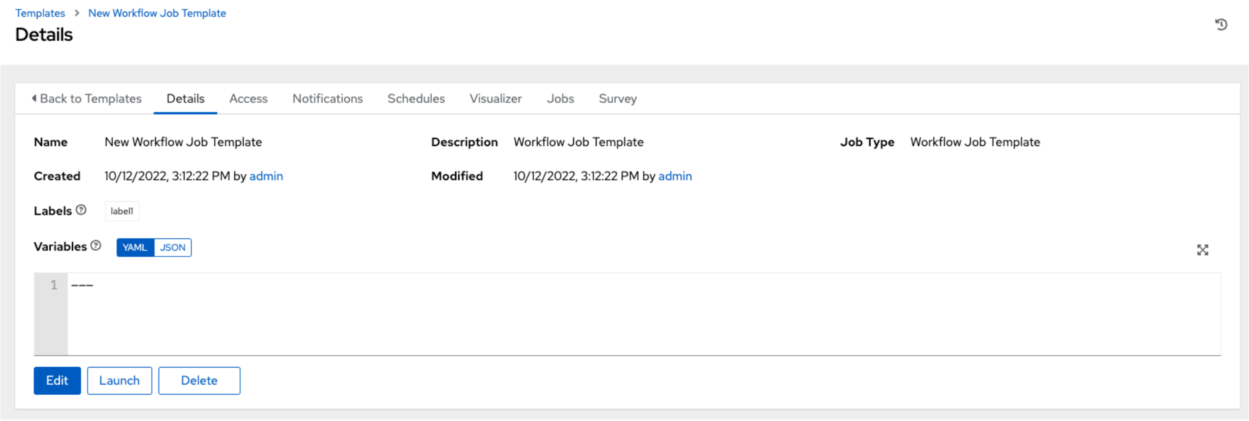
  - **Webhook URL**: Automatically populated with the URL for the webhook service to POST requests to.
  - **Webhook Key**: Generated shared secret to be used by the webhook service to sign payloads sent to automation controller. This must be configured in the settings on the webhook service in order for automation controller to accept webhooks from this service.

  For additional information on setting up webhooks, see *Working with Webhooks*.

- Check **Enable Concurrent Jobs** to allow simultaneous runs of this workflow. Refer to *Automation Controller Capacity Determination and Job Impact* for additional information.

4. When you have completed configuring the workflow template, click **Save**.

Saving the template exits the workflow template page and the Workflow Visualizer opens to allow you to build a workflow. See the *Workflow Visualizer* section for further instructions. Otherwise, you may close the Workflow Visualizer to return to the Details tab of the newly saved template in order to review, edit, add permissions, notifications, schedules, and surveys, or view completed jobs and build a workflow template at a later time. Alternatively, you can click **Launch** to launch the workflow, but you must first save the template prior to launching, otherwise, the **Launch** button remains grayed-out. Also, note the **Notifications** tab is present only after the template has been saved.



Name New Workflow Job Template

Description Workflow Job Template

Job Type Workflow Job Template

Created 10/12/2022, 3:12:22 PM by admin

Modified 10/12/2022, 3:12:22 PM by admin

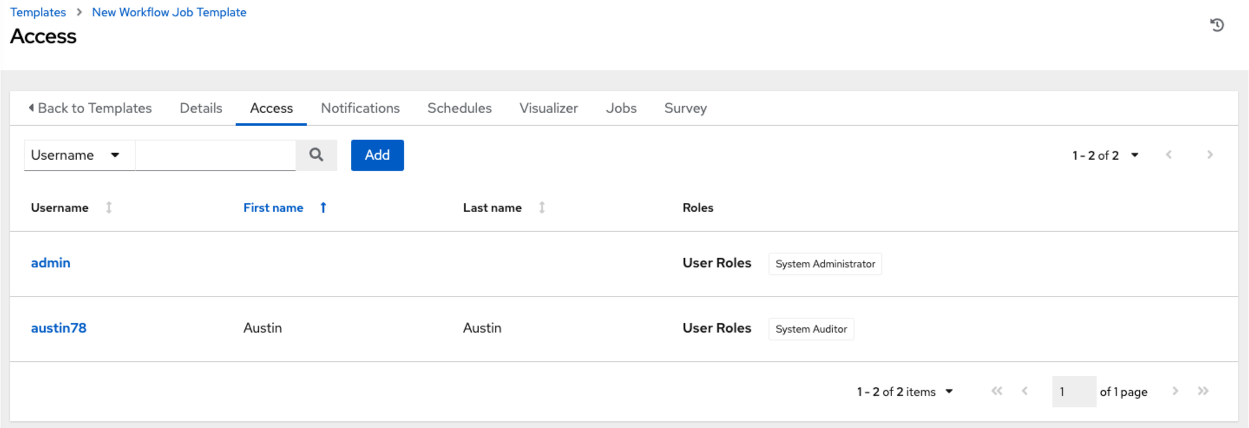
Labels label1

Variables

```
---
```

23.2 Work with Permissions

Clicking on **Access** allows you to review, grant, edit, and remove associated permissions for users as well as team members.



Username	First name	Last name	Roles
admin			User Roles System Administrator
austin78	Austin	Austin	User Roles System Auditor

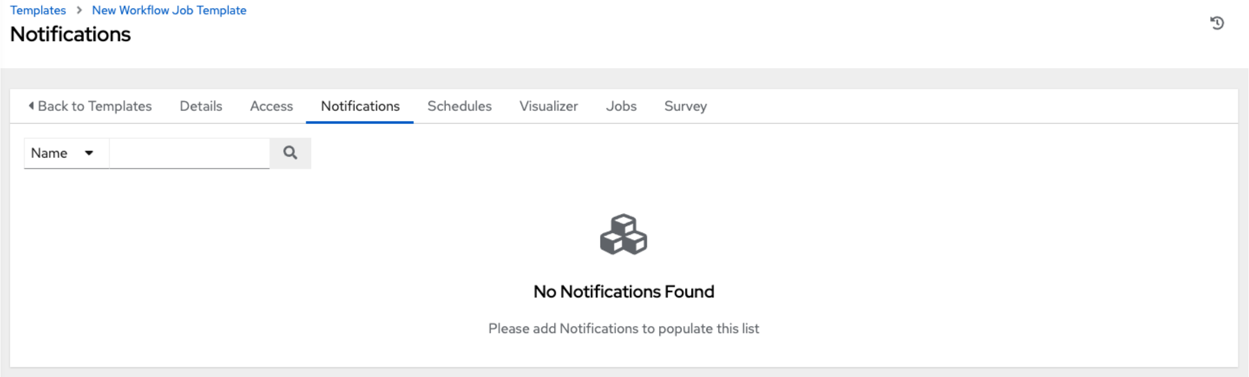
Click the **Add** button to create new permissions for this workflow template by following the prompts to assign them accordingly.

23.3 Work with Notifications

Clicking on **Notifications** allows you to review any notification integrations you have setup. The **Notifications** tab is present only after the template has been saved.

Use the toggles to enable or disable the notifications to use with your particular template. For more detail, see *Enable and Disable Notifications*.

If no notifications have been set up, see *Create a Notification Template* for detail.



No Notifications Found

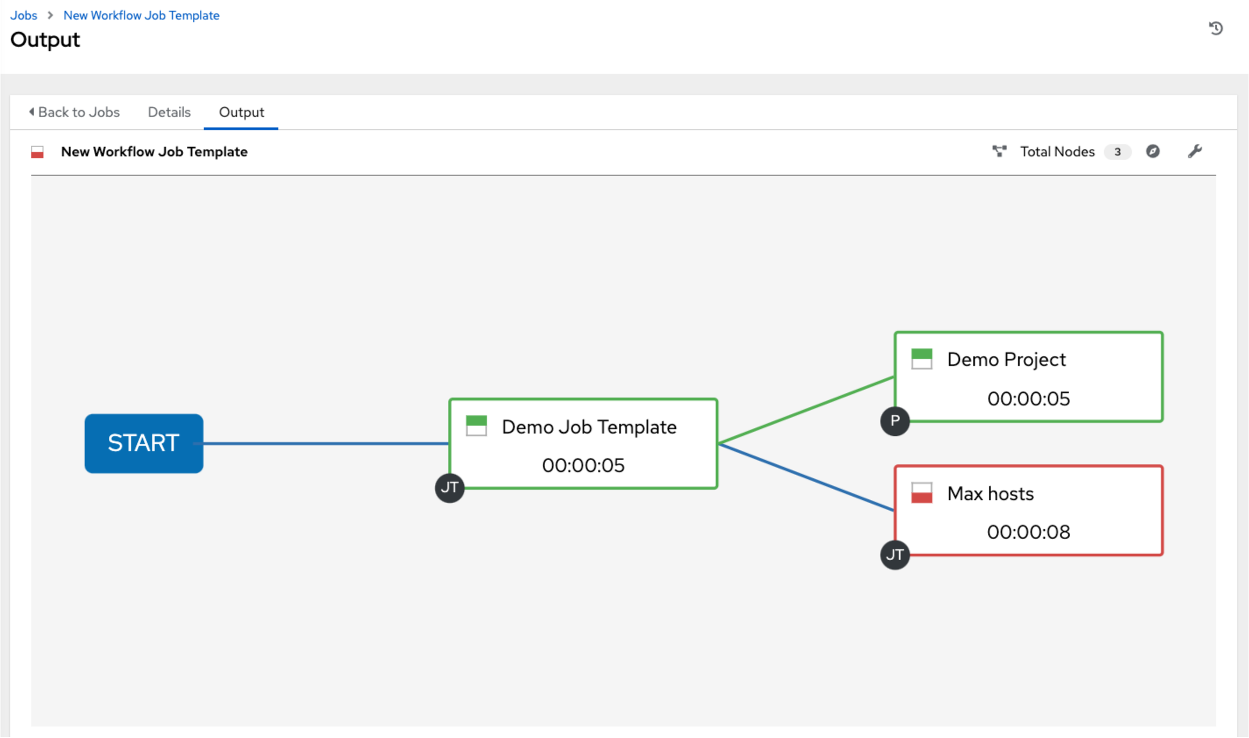
Please add Notifications to populate this list

Refer to *Notification Types* for additional details on configuring various notification types.

23.4 View Completed Jobs

The **Completed Jobs** tab provides the list of workflow templates that have ran. Click **Expanded** to view the various details of each job.

From this view, you can click the job ID - name of the workflow job and see its graphical representation. The example below shows the job details of a workflow job.



The nodes are marked with labels that help you identify them at a glance. See the *legend* in the *Workflow Visualizer* section for more information.

23.5 Work with Schedules

Clicking on **Schedules** allows you to review any schedules set up for this template.

23.5.1 Schedule a Workflow Template

To schedule a wokflow job template run, click the **Schedules** tab.

- If schedules are already set up; review, edit, or enable/disable your schedule preferences.
- If schedules have not been set up, refer to *Schedules* for more information.

If a workflow template used in a nested workflow has a survey, or the **Prompt on Launch** selected for the inventory option, the **PROMPT** button displays next to the **SAVE** and **CANCEL** buttons on the schedule form. Clicking the **PROMPT** button shows an optional INVENTORY step where you can provide or remove an inventory or skip this step without any changes.

23.6 Surveys

Workflows containing job types of Run or Check provide a way to set up surveys in the Workflow Job Template creation or editing screens. Surveys set extra variables for the playbook similar to 'Prompt for Extra Variables' does, but in a user-friendly question and answer way. Surveys also allow for validation of user input. Click the **Survey** tab to create a survey.

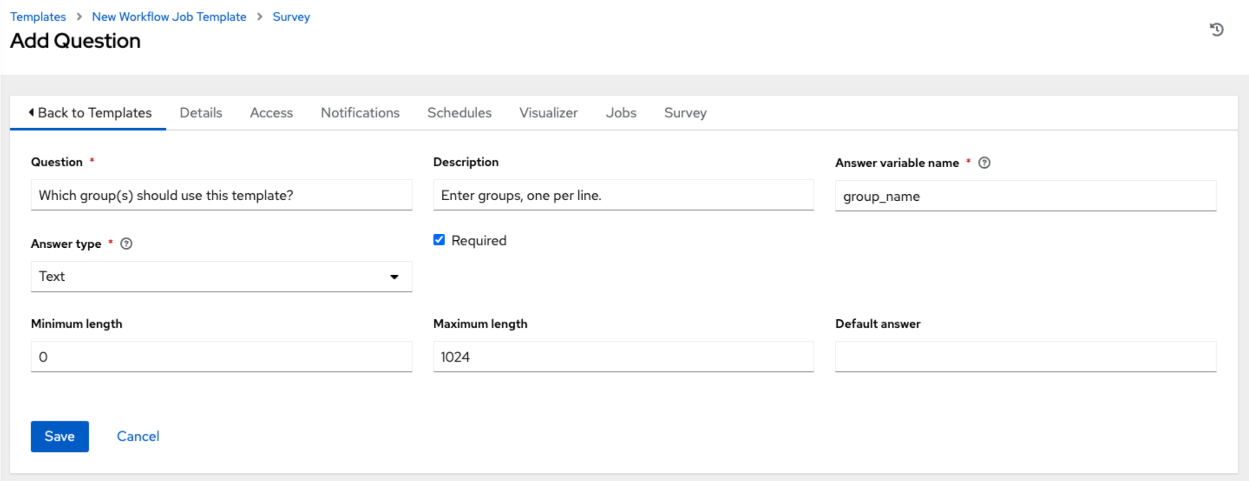
Use cases for surveys are numerous. An example might be if operations wanted to give developers a "push to stage" button they could run without advanced Ansible knowledge. When launched, this task could prompt for answers to questions such as, "What tag should we release?"

Many types of questions can be asked, including multiple-choice questions.

23.6.1 Create a Survey

To create a survey:

1. Click the **Survey** tab to bring up the **Add Survey** window.



Question *: Which group(s) should use this template?

Description: Enter groups, one per line.

Answer variable name *: group_name

Answer type *: Text

☑ Required

Minimum length: 0

Maximum length: 1024

Default answer:

Use the **ON/OFF** toggle button at the top of the screen to quickly activate or deactivate this survey prompt.

2. A survey can consist of any number of questions. For each question, enter the following information:

- **Name**: The question to ask the user.
- **Description**: (optional) A description of what's being asked of the user.
- **Answer Variable Name**: The Ansible variable name to store the user's response in. This is the variable to be used by the playbook. Variable names cannot contain spaces.
- **Answer Type**: Choose from the following question types.
  - *Text*: A single line of text. You can set the minimum and maximum length (in characters) for this answer.
  - *Textarea*: A multi-line text field. You can set the minimum and maximum length (in characters) for this answer.
  - *Password*: Responses are treated as sensitive information, much like an actual password is treated. You can set the minimum and maximum length (in characters) for this answer.
  - *Multiple Choice (single select)*: A list of options, of which only one can be selected at a time. Enter the options, one per line, in the **Multiple Choice Options** box.

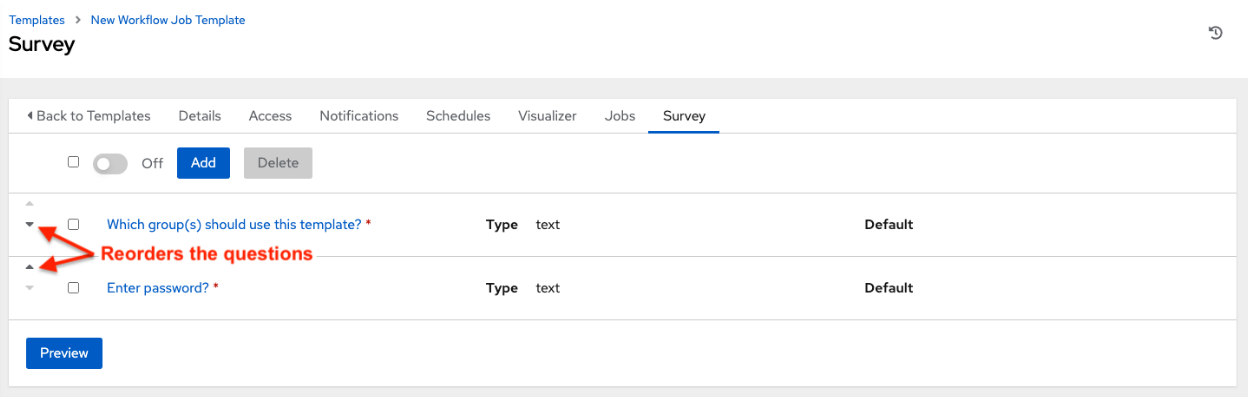
   - *Multiple Choice (multiple select)*: A list of options, any number of which can be selected at a time. Enter the options, one per line, in the **Multiple Choice Options** box.
   - *Integer*: An integer number. You can set the minimum and maximum length (in characters) for this answer.
   - *Float*: A decimal number. You can set the minimum and maximum length (in characters) for this answer.

- **Default Answer**: Depending on which type chosen, you can supply the default answer to the question. This value is pre-filled in the interface and is used if the answer is not provided by the user.
- **Required**: Whether or not an answer to this question is required from the user.

3. Once you have entered the question information, click the **Add** button to add the question.

A stylized version of the survey is presented in the Preview pane. For any question, you can click on the **Edit** button to edit the question, the **Delete** button to delete the question, and click and drag on the grid icon to rearrange the order of the questions.

4. Return to the left pane to add additional questions.
5. When done, click **Save** to save the survey.



Which group(s) should use this template? *	Type text	Default
Enter password? *	Type text	Default

23.6.2 Optional Survey Questions

The **Required** setting on a survey question determines whether the answer is optional or not for the user interacting with it.

Behind the scenes, optional survey variables can be passed to the playbook in `extra_vars`, even when they aren't filled in.

- If a non-text variable (input type) is marked as optional, and is not filled in, no survey `extra_var` is passed to the playbook.
- If a text input or text area input is marked as optional, is not filled in, and has a minimum `length > 0`, no survey `extra_var` is passed to the playbook.
- If a text input or text area input is marked as optional, is not filled in, and has a minimum `length === 0`, that survey `extra_var` is passed to the playbook, with the value set to an empty string ("").

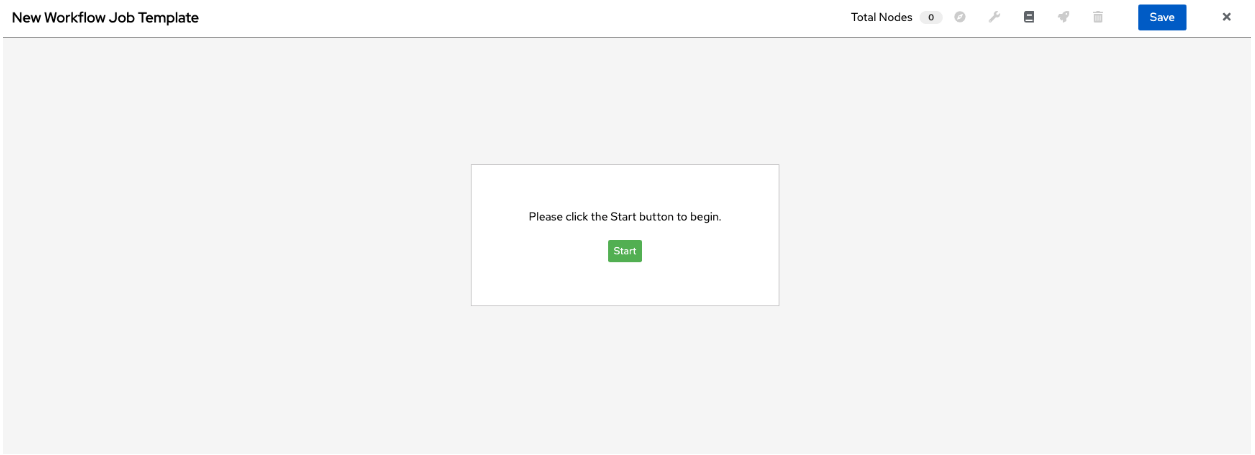
23.7 Workflow Visualizer

The Workflow Visualizer provides a graphical way of linking together job templates, workflow templates, project syncs, and inventory syncs to build a workflow template. Before building a workflow template, refer to the *Workflows* section for considerations associated with various scenarios on parent, child, and sibling nodes.

23.7.1 Build a Workflow

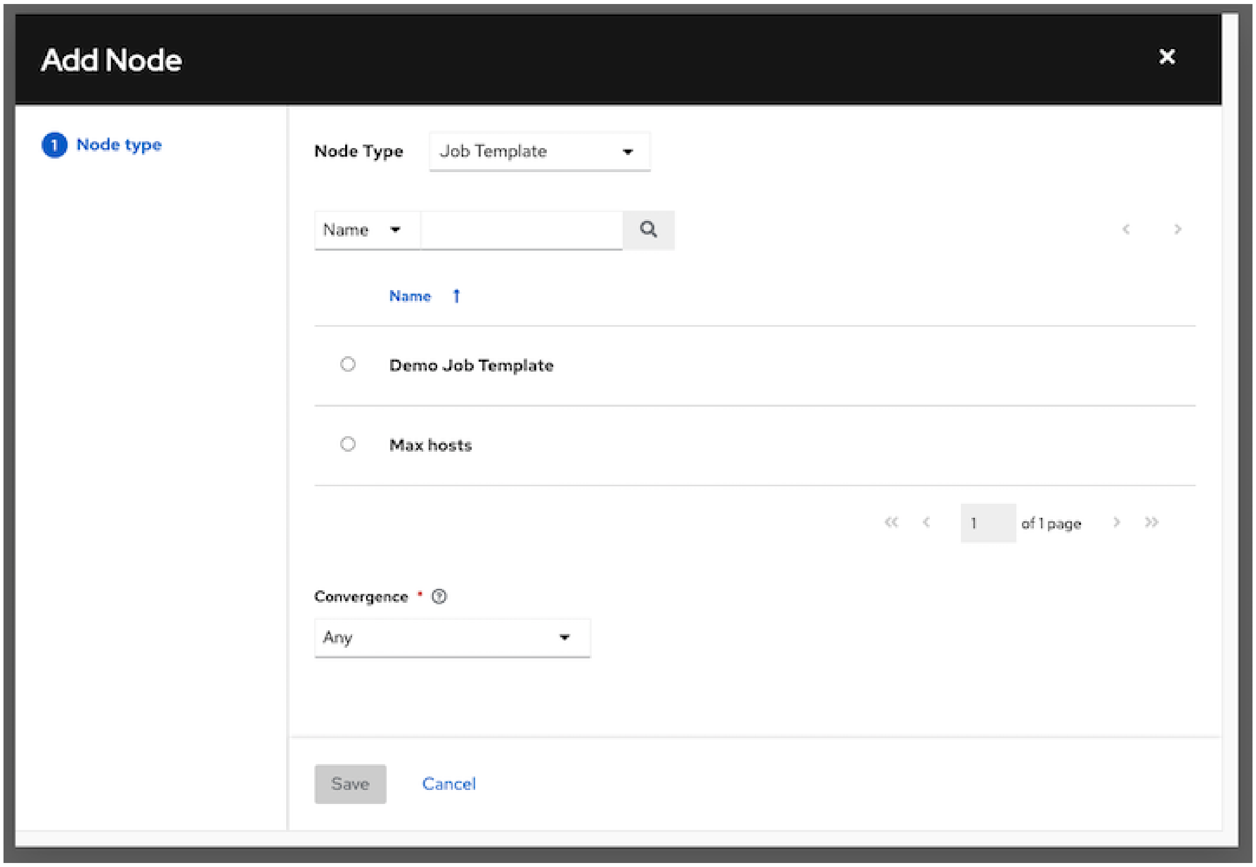
You can set up any combination of two or more of the following node types to build a workflow: Template (Job Template or Workflow Job Template), Project Sync, Inventory Sync, or Approval. Each node is represented by a rectangle while the relationships and their associated edge types are represented by a line (or link) that connects them.

1. In the details/edit view of a workflow template, click the **Visualizer** tab or from the Templates list view, click the ( ) icon to launch the Workflow Visualizer.



Please click the Start button to begin.

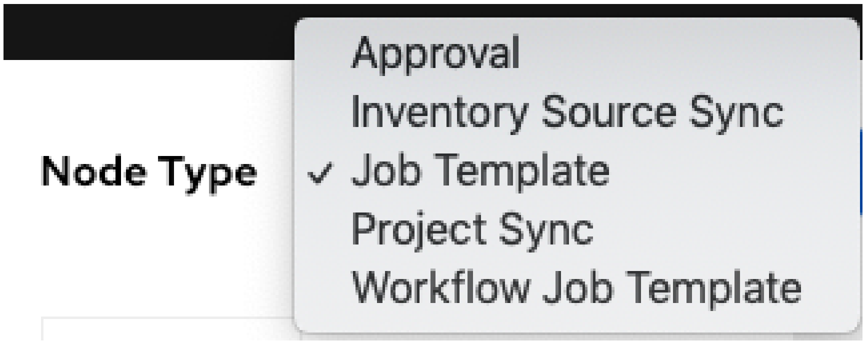
2. Click the START button to display a list of nodes to add to your workflow.



Node Type: Job Template

Convergence *: Any

3. On the right pane, select the type of node you want to add from the drop-down menu:



If selecting an **Approval** node, see *Approval nodes* for further detail.

Selecting a node provides the available valid options associated with it.

Note: If you select a job template that does not have a default inventory when populating a workflow graph, the inventory of the parent workflow will be used. Though a credential is not required in a job template, you will not be able to choose a job template for your workflow if it has a credential that requires a password, unless the credential is replaced by a prompted credential.

4. Once a node is selected, the workflow begins to build, and you must specify the type of action to be taken for the selected node. This action is also referred to as *edge type*.
5. If the node is a root node, the edge type defaults to **Always** and is non-editable.

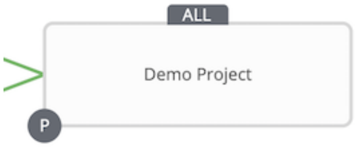
For subsequent nodes, you can select one of the following scenarios (edge type) to apply to each:

- **Always**: Continue to execute regardless of success or failure.
- **On Success**: Upon successful completion, execute the next template.
- **On Failure**: Upon failure, execute a different template.

6. Select the behavior of the node if it is a convergent node from the **Convergence** field:

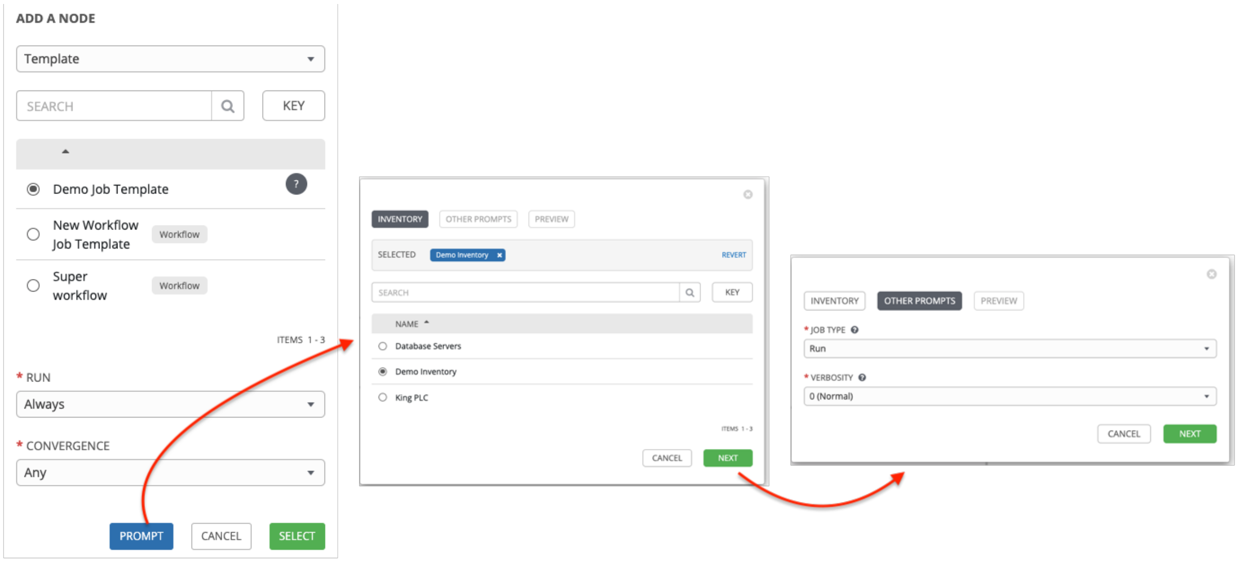
   - **Any** is the default behavior, allowing *any* of the nodes to complete as specified, before triggering the next converging node. As long as the status of one parent meets one of those run conditions, an ANY child node will run. In other words, an ANY node requires **all** nodes to complete, but only one node must complete with the expected outcome.
   - Choose **All** to ensure that *all* nodes complete as specified, before converging and triggering the next node. The purpose of ALL nodes is to make sure that every parent met it's expected outcome in order to run the child node. The workflow checks to make sure every parent behaved as expected in order to run the child node. Otherwise, it will not run the child node.

   If selected, the graphical view will label the node as **ALL**.



Note: If a node is a root node, or a node that does not have any nodes converging into it, setting the **Convergence** rule does not apply, as its behavior is dictated by the action that triggers it.

7. If a job template used in the workflow has **Prompt on Launch** selected for any of its parameters, a **Prompt** button appears, allowing you to change those values at the node level. Use the wizard to change the value(s) in each of the tabs and click **Confirm** in the Preview tab.


ADD A NODE

Template

SEARCH KEY

● Demo Job Template

○ New Workflow Job Template Workflow

○ Super workflow Workflow

ITEMS 1 - 3

* RUN: Always

* CONVERGENCE: Any

PROMPT CANCEL SELECT

INVENTORY OTHER PROMPTS PREVIEW

SELECTED Demo Inventory REVERT

SEARCH KEY

NAME

○ Database Servers

● Demo Inventory

○ King PLC

ITEMS 1 - 3

CANCEL NEXT

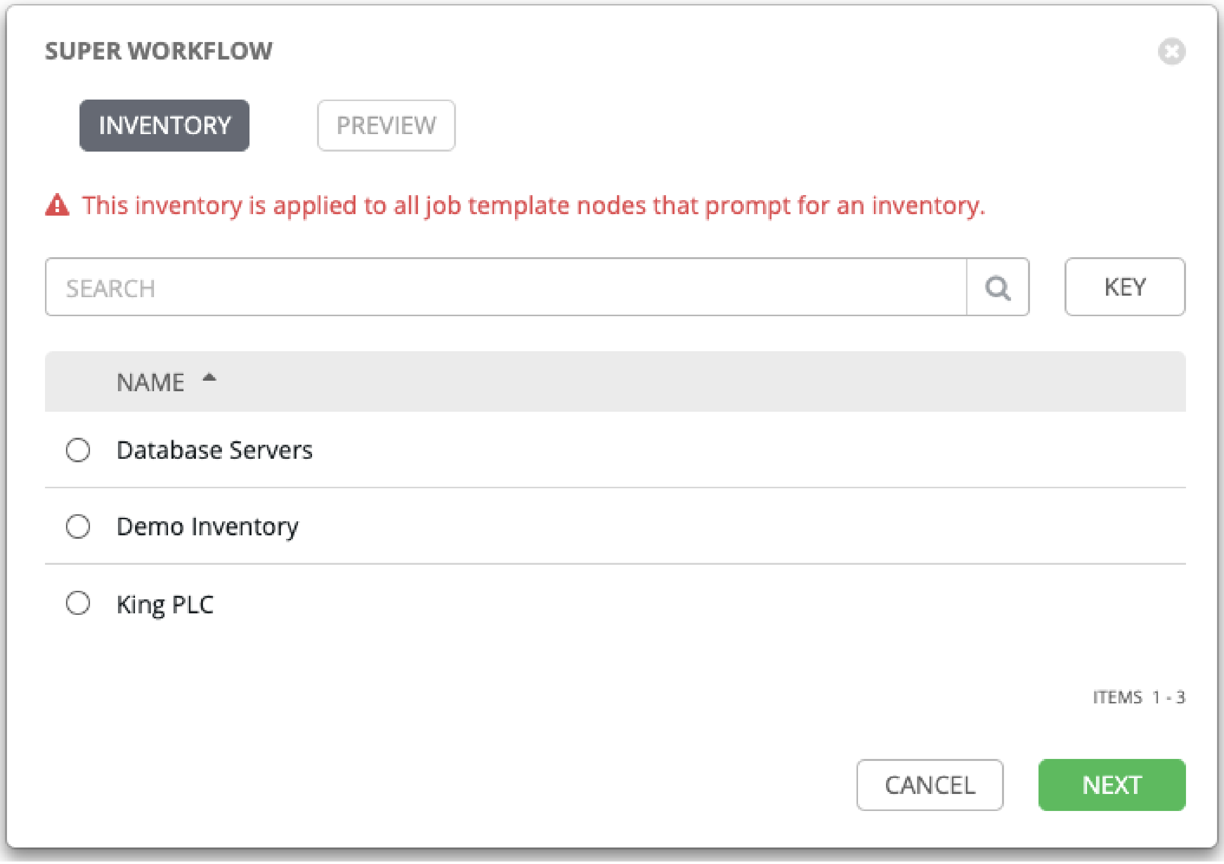
INVENTORY OTHER PROMPTS PREVIEW

* JOB TYPE: Run

* VERBOSITY: 0 (Normal)

CANCEL NEXT

Likewise, if a workflow template used in the workflow has **Prompt on Launch** selected for the inventory option, use the wizard to supply the inventory at the prompt. If the parent workflow has its own inventory, it will override any inventory that is supplied here.



SUPER WORKFLOW

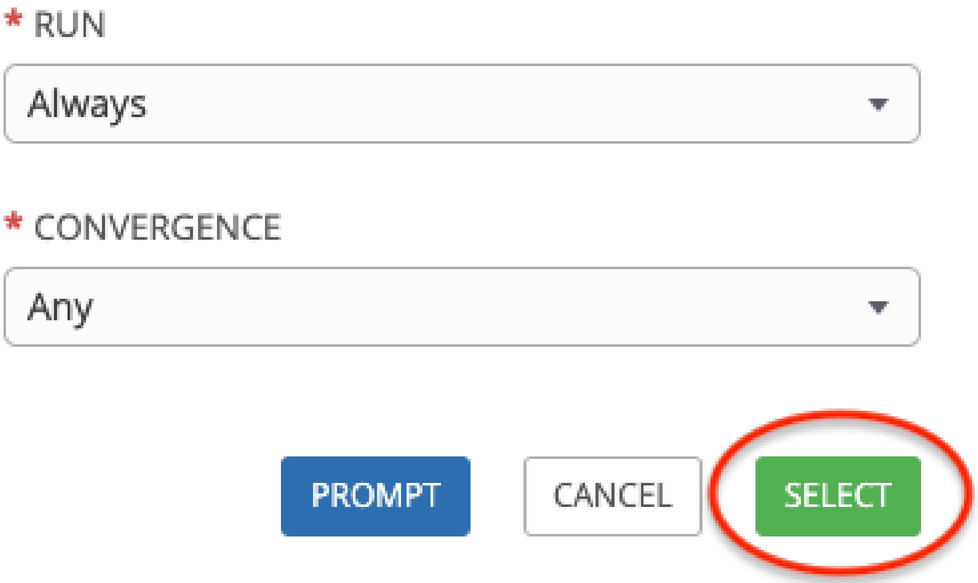
⚠ This inventory is applied to all job template nodes that prompt for an inventory.

	NAME ▲
○	Database Servers
○	Demo Inventory
○	King PLC

ITEMS 1 - 3

Note: For workflow job templates with promptable fields that are required, but do not have a default, you must provide those values when creating a node before the **Select** button becomes enabled. The two cases that disable the **Select** button until a value is provided via the **Prompt** button: 1) when you select the **Prompt on Launch** checkbox in a workflow job template, but do not provide a default, or 2) when you create a survey question that is required but do not provide a default answer. However, this is **NOT** the case with credentials. Credentials that require a password on launch are **not permitted** when creating a workflow node, since everything needed to launch the node must be provided when the node is created. So, if a workflow job template prompts for credentials, automation controller prevents you from being able to select a credential that requires a password.

You must also click **Select** when the prompt wizard closes in order to apply the changes at that node. Otherwise, any changes you make will revert back to the values set in the actual job template.



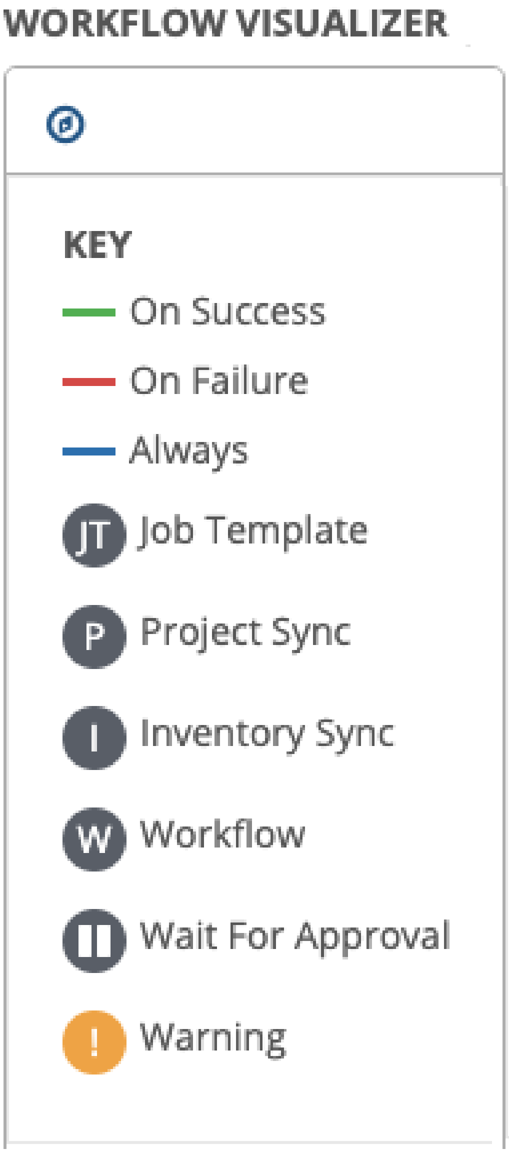
* RUN

Always

* CONVERGENCE

Any

Once the node is created, it is labeled with its job type. A template that is associated with each workflow node will run based on the selected run scenario as it proceeds. Click the compass ( ) icon to display the legend for each run scenario and their job types.

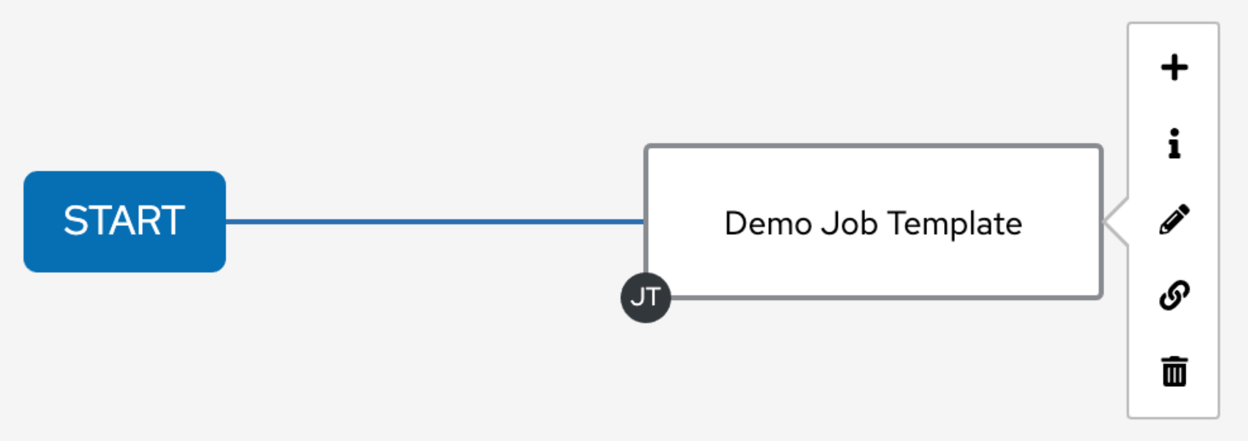


WORKFLOW VISUALIZER

KEY

- 🟩 On Success
- 🟥 On Failure
- 🟦 Always
- JT Job Template
- P Project Sync
- I Inventory Sync
- W Workflow
- ⏸ Wait For Approval
- ! Warning

8. Hovering over a node allows you to add ✚ another node, view info ℹ about the node, edit ✎ the node details, edit an existing link 🔗, or delete 🗑 the selected node.



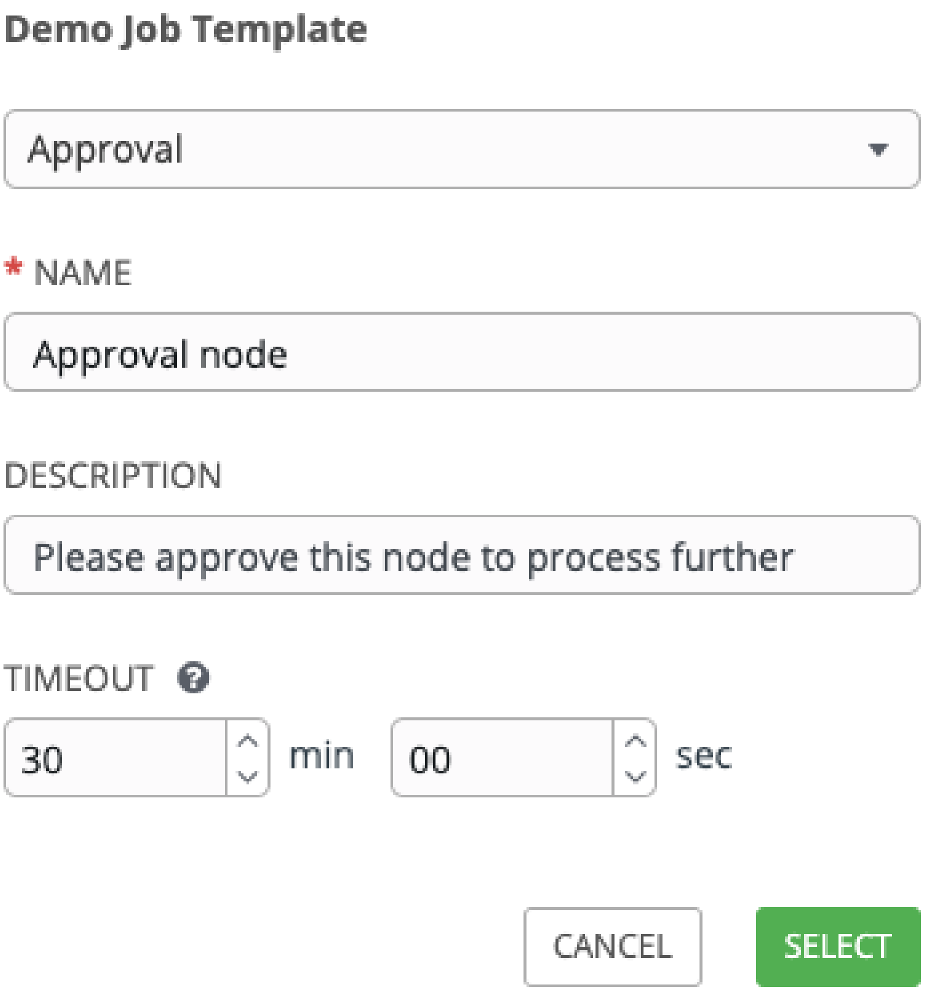
9. When done adding/editing a node, click **Select** to save any modifications and render it on the graphical view. For possible ways to build your workflow, see *Node building scenarios*.

10. When done with building your workflow template, click **Save** to save your entire workflow template and return to the new workflow template details page.

Important: Clicking **Close** on this pane will not save your work, but instead, closes the entire Workflow Visualizer and you will have to start over.

Approval nodes

Choosing an **Approval** node requires user intervention in order to advance the workflow. This functions as a means to pause the workflow in between playbooks so that a user can give approval to continue on to the next playbook in the workflow, giving the user a specified amount of time to intervene, but also allows the user to continue as quickly as possible without having to wait on some other trigger.



Demo Job Template

Approval

* NAME

Approval node

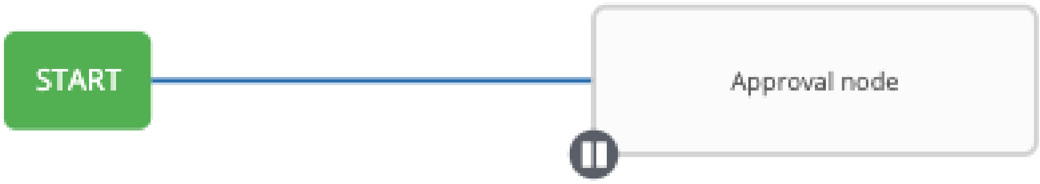
DESCRIPTION

Please approve this node to process further

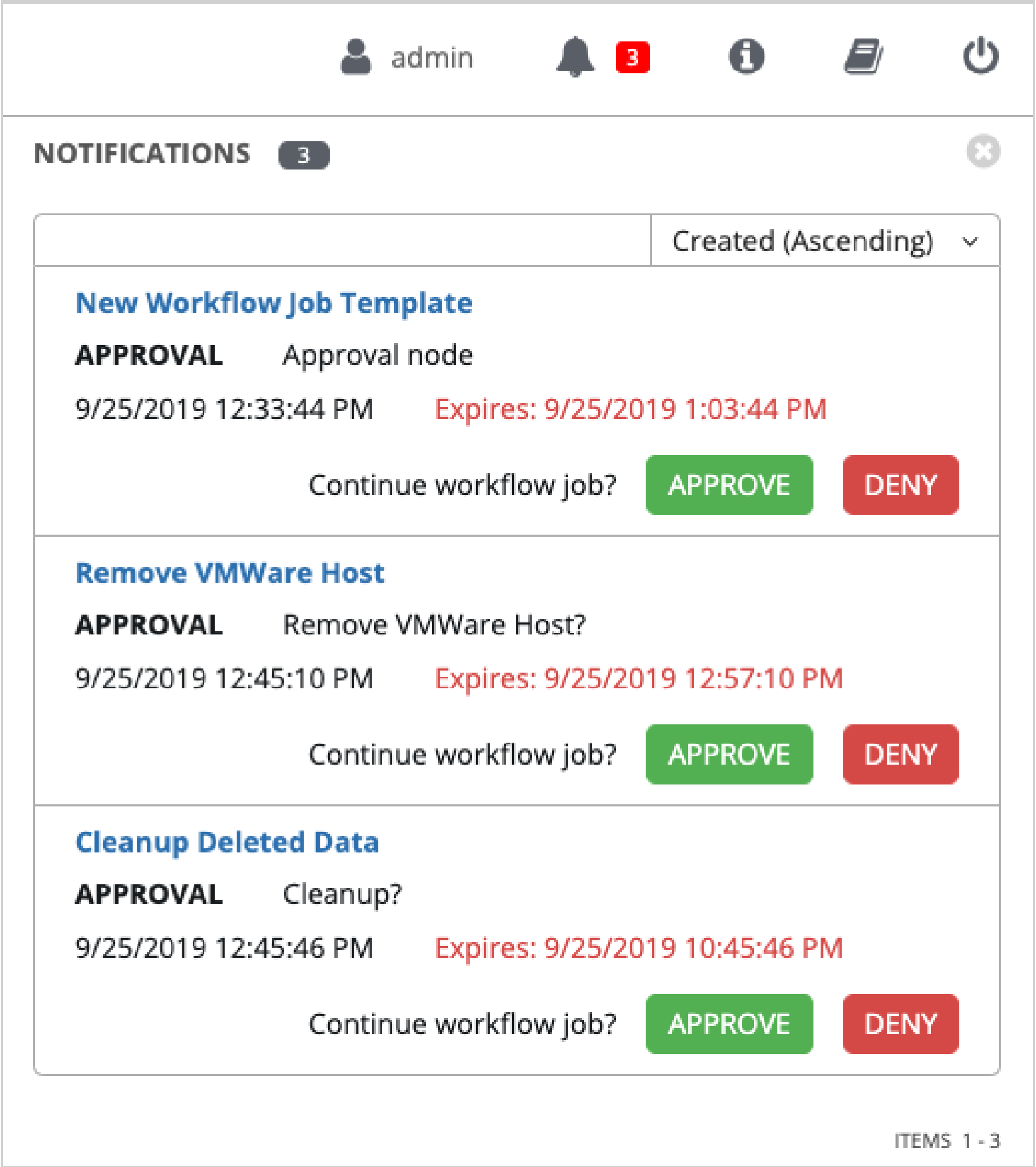
TIMEOUT

30 min 00 sec

The default for the timeout is none, but you can specify the length of time before the request expires and automatically gets denied. After selecting and supplying the information for the approval node, it displays on the graphical view with a pause (⏸) icon next to it.



The approver is anyone who can execute the workflow job template containing the approval nodes, has org admin or above privileges (for the org associated with that workflow job template), or any user who has the *Approve* permission explicitly assigned to them within that specific workflow job template.



New Workflow Job Template

APPROVAL Approval node

9/25/2019 12:33:44 PM Expires: 9/25/2019 1:03:44 PM

Continue workflow job? APPROVE DENY

Remove VMWare Host

APPROVAL Remove VMWare Host?

9/25/2019 12:45:10 PM Expires: 9/25/2019 12:57:10 PM

Continue workflow job? APPROVE DENY

Cleanup Deleted Data

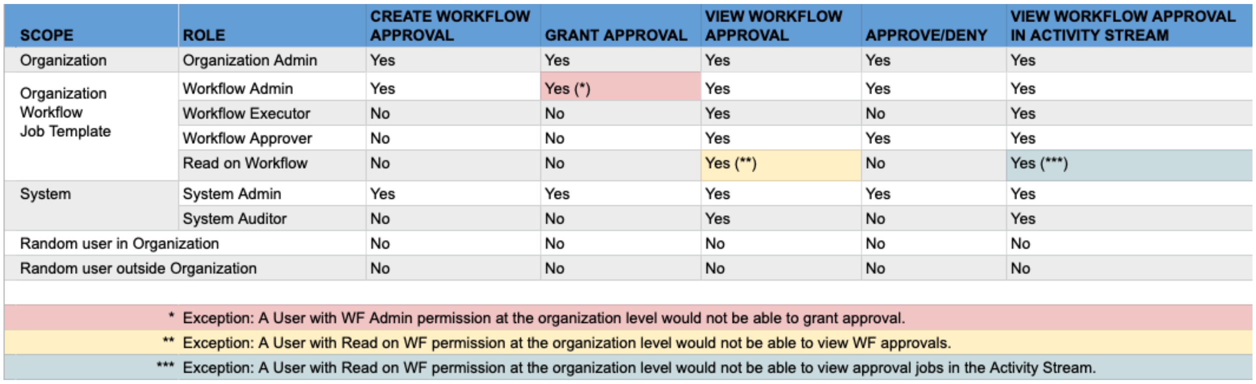
APPROVAL Cleanup?

9/25/2019 12:45:46 PM Expires: 9/25/2019 10:45:46 PM

Continue workflow job? APPROVE DENY

If pending approval nodes are not approved within the specified time limit (if an expiration was assigned) or they are denied, then they are marked as “timed out” or “failed”, respectively, and move on to the next “on fail node” or “always node”. If approved, the “on success” path is taken. If you try to POST in the API to a node that has already been approved, denied or timed out, an error message notifies you that this action is redundant, and no further steps will be taken.

Below shows the various levels of permissions allowed on approval workflows:



SCOPE	ROLE	CREATE WORKFLOW APPROVAL	GRANT APPROVAL	VIEW WORKFLOW APPROVAL	APPROVE/DENY	VIEW WORKFLOW APPROVAL IN ACTIVITY STREAM
Organization	Organization Admin	Yes	Yes	Yes	Yes	Yes
Organization Workflow Job Template	Workflow Admin	Yes	Yes (*)	Yes	Yes	Yes
	Workflow Executor	No	No	Yes	No	Yes
	Workflow Approver	No	No	Yes	Yes	Yes
	Read on Workflow	No	No	Yes (**)	No	Yes (***)
System	System Admin	Yes	Yes	Yes	Yes	Yes
	System Auditor	No	No	Yes	No	Yes
Random user in Organization		No	No	No	No	No
Random user outside Organization		No	No	No	No	No

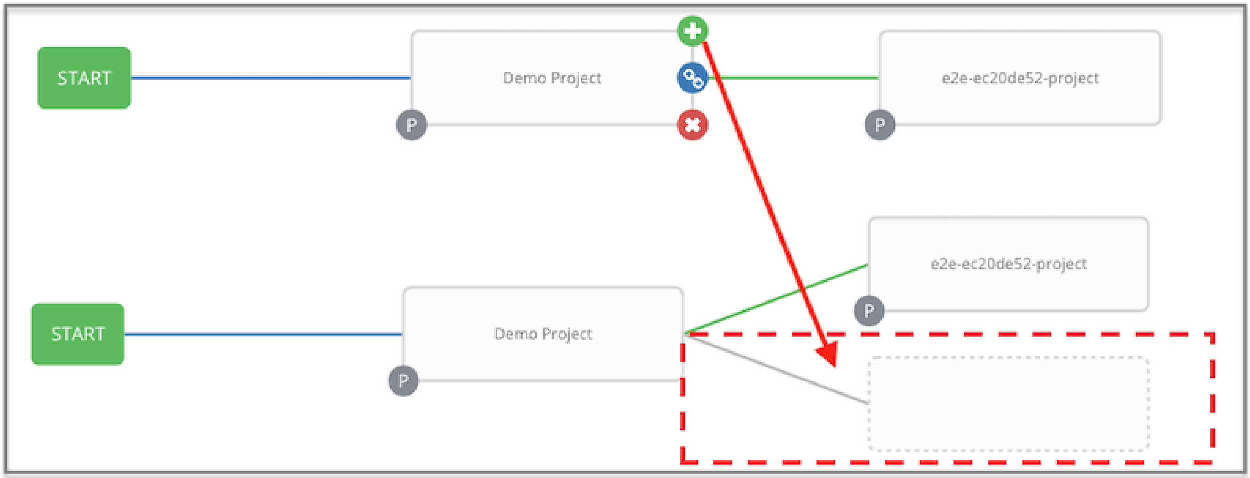
* Exception: A User with WF Admin permission at the organization level would not be able to grant approval.

** Exception: A User with Read on WF permission at the organization level would not be able to view WF approvals.

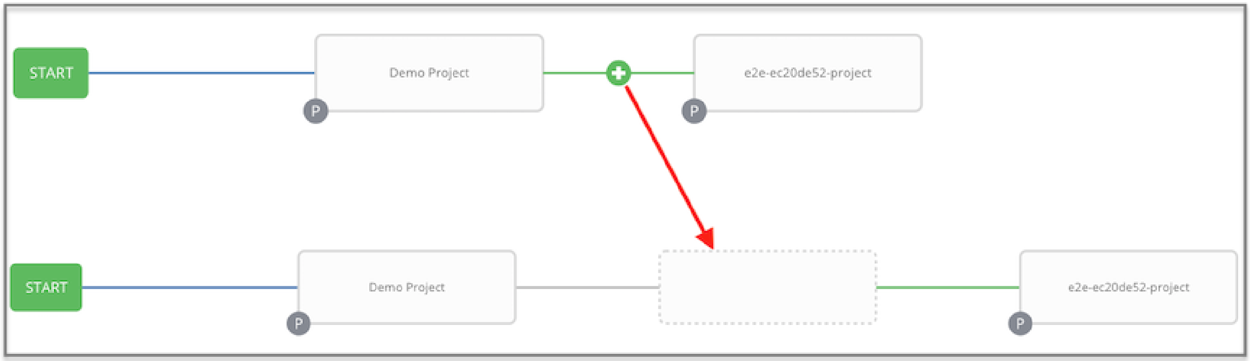
*** Exception: A User with Read on WF permission at the organization level would not be able to view approval jobs in the Activity Stream.

Node building scenarios

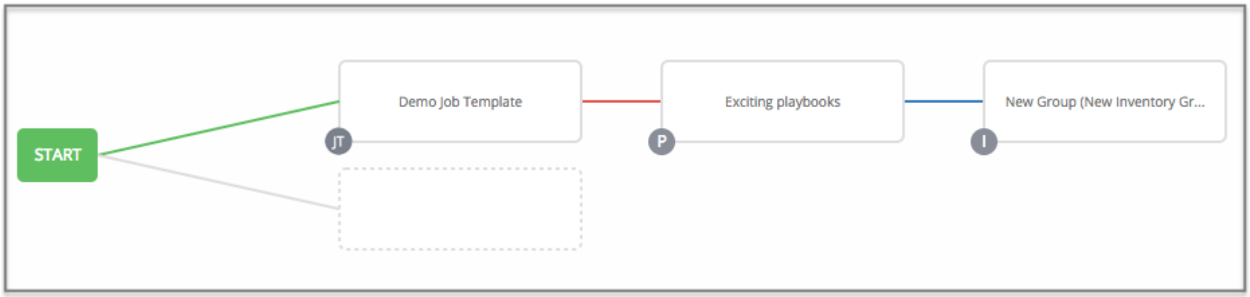
You can add a sibling node by clicking the ✚ on the parent node:



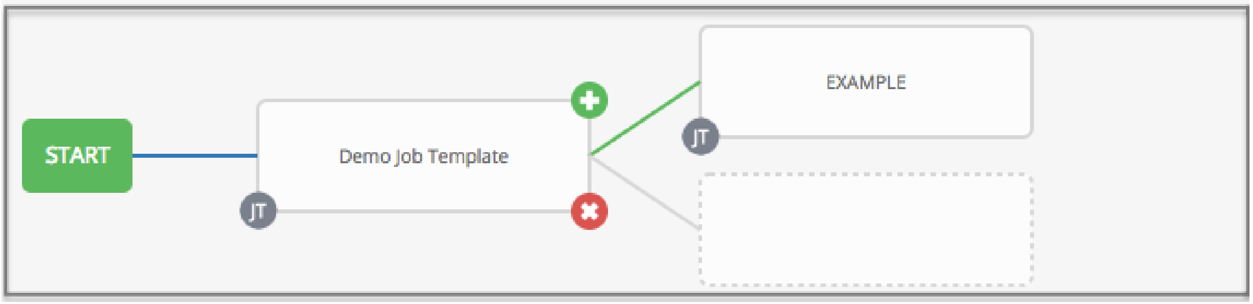
You can insert another node in between nodes by hovering over the line that connects the two until the ✚ appears. Clicking on the ✚ automatically inserts the node between the two nodes.



To add a root node to depict a split scenario, click the START button again:



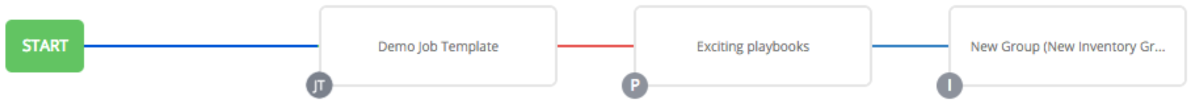
At any node where you want to create a split scenario, hover over the node from which the split scenario begins and click the ✚. This essentially adds multiple nodes from the same parent node, creating sibling nodes:



Note: When adding a new node, the **PROMPT** button applies to workflow templates as well. Workflow templates will prompt for inventory and surveys.

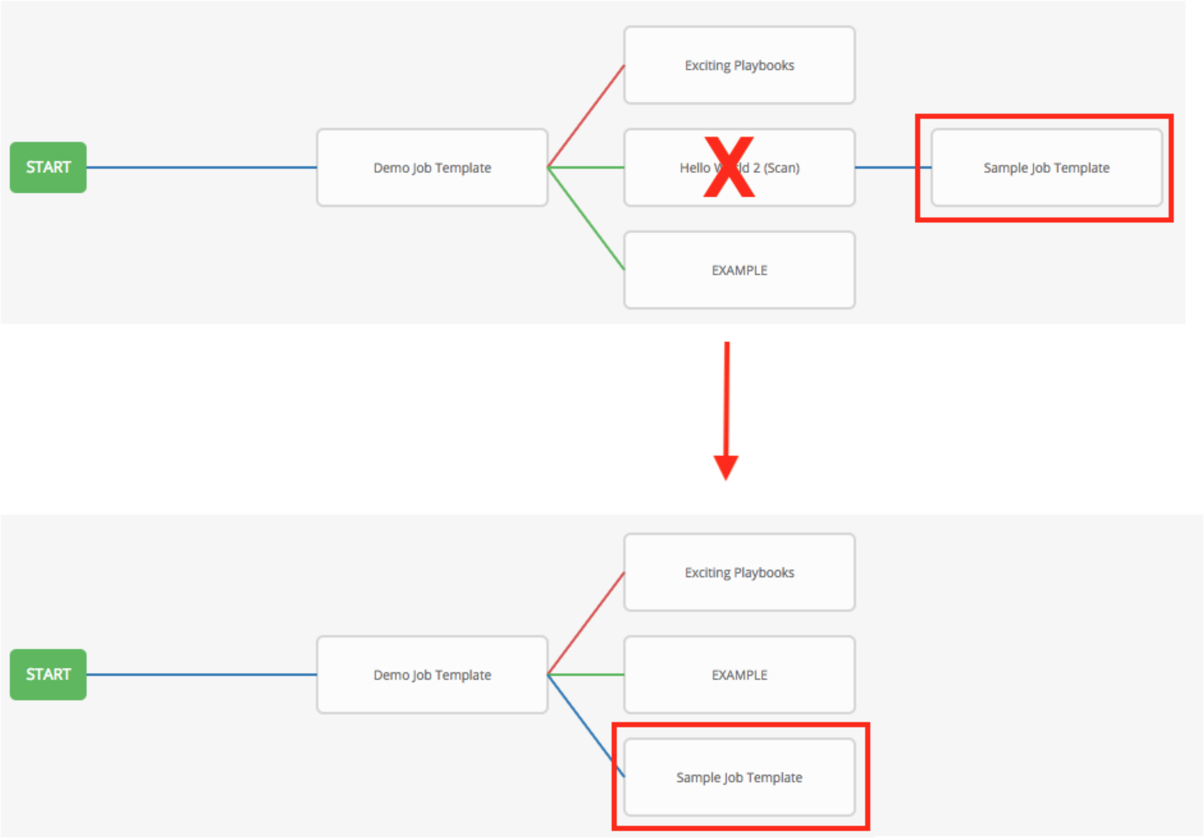
If you want to undo the last inserted node, click on another node without making a selection from the right pane. Or, click **Cancel** from the right pane.

Below is an example of a workflow that contains all three types of jobs that is initiated by a job template that if it fails to run, proceed to the project sync job, and regardless of whether that fails or succeeds, proceed to the inventory sync job.



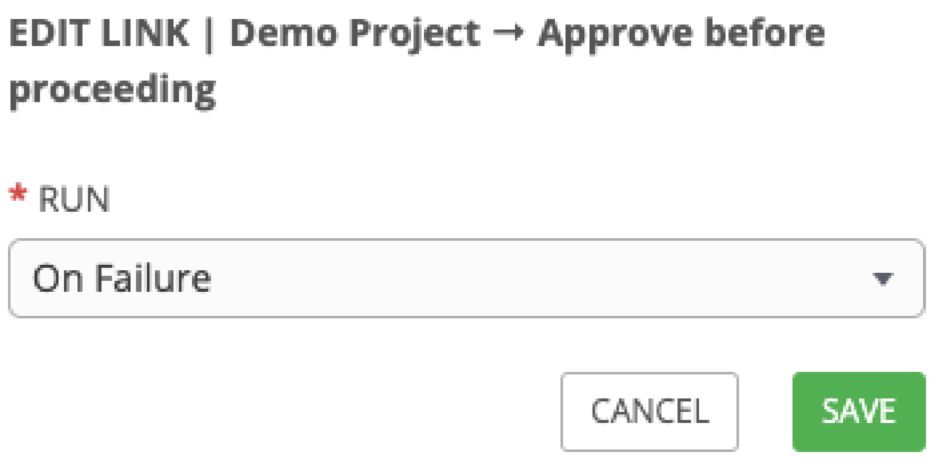
Remember to refer to the Key at the top of the window to identify the meaning of the symbols and colors associated with the graphical depiction.

Note: In a workflow with a set of sibling nodes having varying edge types, and you remove a node that has a follow-on node attached to it, the attached node automatically joins the set of sibling nodes and retains its edge type:



The following ways you can modify your nodes:

- If you want to edit a node, click on the node you want to edit. The right pane displays the current selections. Make your changes and click **Select** to apply them to the graphical view.
- To edit the edge type for an existing link (success/failure/always), click on the link. The right pane displays the current selection. Make your changes and click **Save** to apply them to the graphical view.

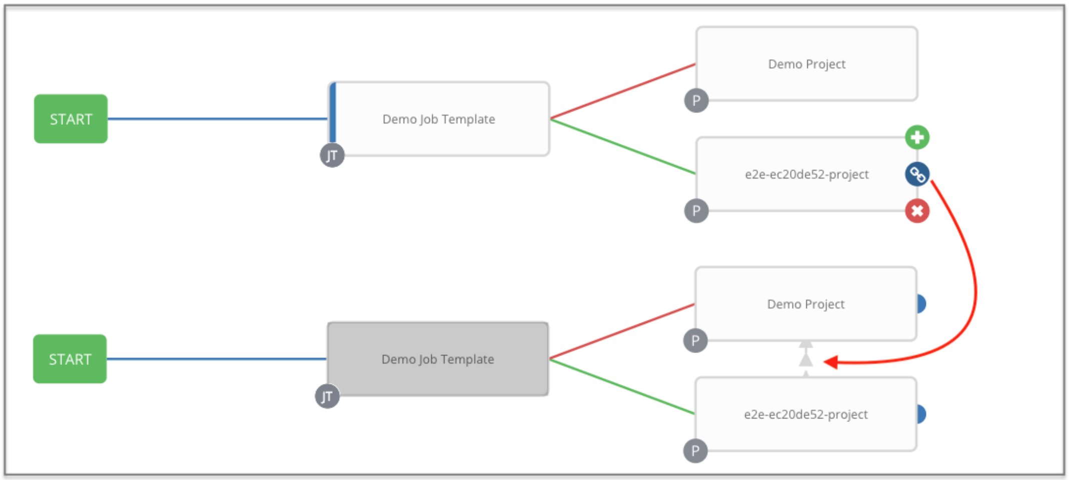


EDIT LINK | Demo Project → Approve before proceeding

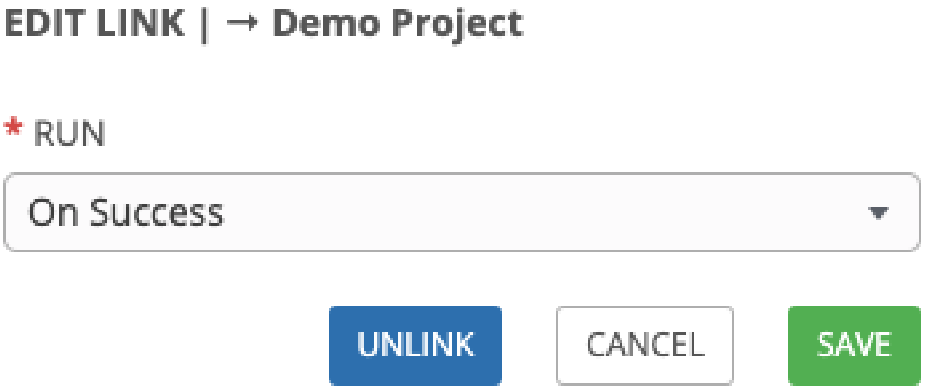
* RUN

On Failure

- To add a new link from one node to another, click the link 🔗 icon that appears on each node. Doing this highlights the nodes that are possible to link to. These feasible options are indicated by the dotted lines. Invalid options are indicated by grayed out boxes (nodes) that would otherwise produce an invalid link. The example below shows the **Demo Project** as a possible option for the **e2e-ec20de52-project** to link to, as indicated by the arrows:



- To remove a link, click the link and click the **Unlink** button.


EDIT LINK | → Demo Project

* RUN

On Success

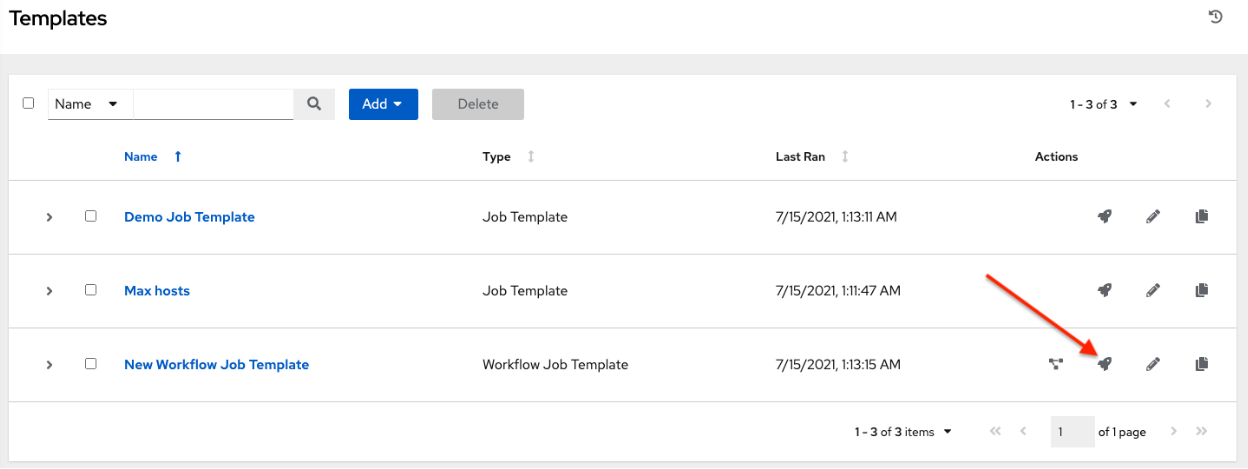
This button only appears in the right hand panel if the target or child node has more than one parent. All nodes must be linked to at least one other node at all times so you must create a new link before removing an old one.

Click the settings icon (⚙) to zoom, pan, or reposition the view. Alternatively, you can drag the workflow diagram to reposition it on the screen or use the scroll on your mouse to zoom.

23.8 Launch a Workflow Template

Launch a workflow template by any of the following ways:

- Access the workflow templates list from the **Templates** menu on the left navigation bar or while in the workflow template Details view, scroll to the bottom to access the 🚀 button from the list of templates.



Templates

		Name ↑	Type	Last Ran	Actions
›	□	Demo Job Template	Job Template	7/15/2021, 1:13:11 AM	🚀 ✎ ⧉
›	□	Max hosts	Job Template	7/15/2021, 1:11:47 AM	🚀 ✎ ⧉
›	□	New Workflow Job Template	Workflow Job Template	7/15/2021, 1:13:15 AM	🚀 ✎ ⧉

- While in the Workflow Job Template Details view of the job you want to launch, click **Launch**.

Along with any extra variables set in the workflow job template and survey, automation controller automatically adds the same variables as those added for a workflow job template upon launch. Additionally, automation controller automatically redirects the web browser to the Jobs Details page for this job, displaying the progress and the results.

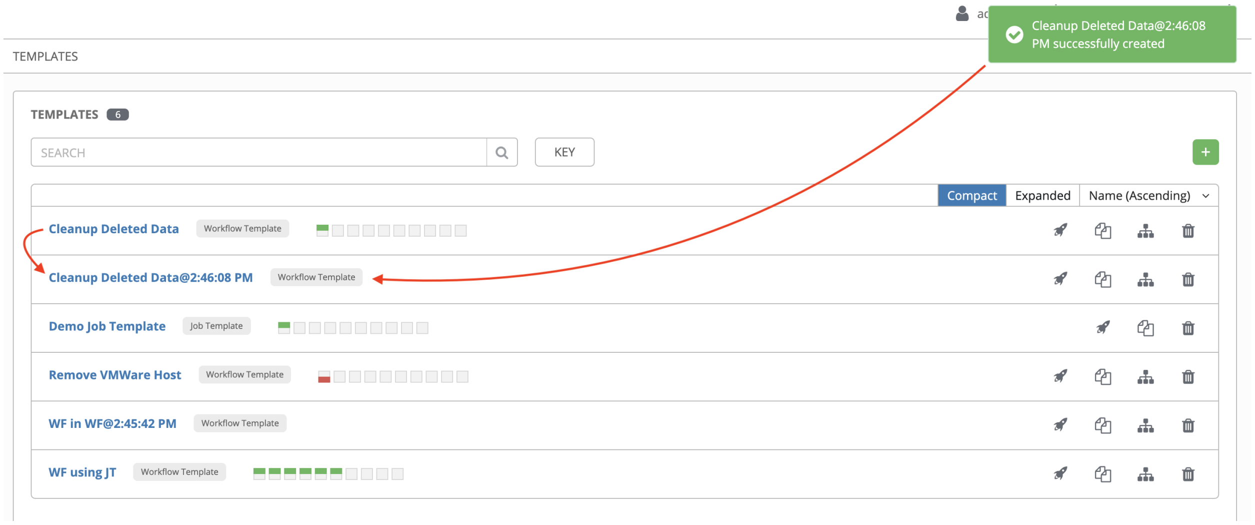
Events related to approvals on workflows display in the Activity Stream (↺) with detailed information about the approval requests, if any.

23.9 Copy a Workflow Template

automation controller allows you the ability to copy a workflow template. If you choose to copy a workflow template, it **does not** copy any associated schedule, notifications, or permissions. Schedules and notifications must be recreated by the user or admin creating the copy of the workflow template. The user copying the workflow template will be granted the admin permission, but no permissions are assigned (copied) to the workflow template.

1. Access the workflow template that you want to copy from the **Templates** menu on the left navigation bar or while in the Workflow Job Template Details view, scroll to the bottom to access it from a list of templates.
2. Click the ⧉ button.

A new template opens with the name of the template from which you copied and a timestamp.



Cleanup Deleted Data@2:46:08 PM successfully created

Cleanup Deleted Data	Workflow Template
Cleanup Deleted Data@2:46:08 PM	Workflow Template
Demo Job Template	Job Template
Remove VMWare Host	Workflow Template
WF in WF@2:45:42 PM	Workflow Template
WF using JT	Workflow Template

Select the copied template and replace the contents of the **Name** field with a new name, and provide or modify the entries in the other fields to complete this template.

3. Click **Save** when done.

Note: If a resource has a related resource that you don't have the right level of permission to, you cannot copy the resource, such as in the case where a project uses a credential that a current user only has *Read* access. However, for a workflow template, if any of its nodes uses an unauthorized job template, inventory, or credential, the workflow template can still be copied. But in the copied workflow template, the corresponding fields in the workflow template node will be absent.

23.10 Extra Variables

Note: `extra_vars` passed to the job launch API are only honored if one of the following is true:

- They correspond to variables in an enabled survey
- `ask_variables_on_launch` is set to True

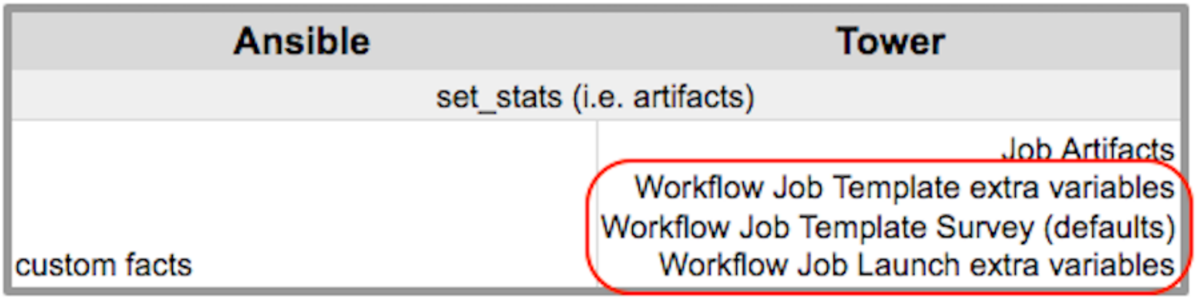
When you pass survey variables, they are passed as extra variables (`extra_vars`). This can be tricky, as passing extra variables to a workflow template (as you would do with a survey) can override other variables being passed from the inventory and project.

For example, say that you have a defined variable for an inventory for `debug = true`. It is entirely possible that this variable, `debug = true`, can be overridden in a workflow template survey.

To ensure that the variables you need to pass are not overridden, ensure they are included by redefining them in the survey. Keep in mind that extra variables can be defined at the inventory, group, and host levels.

The following table notes the behavior (hierarchy) of variable precedence in automation controller as it compares to variable precedence in Ansible.

Variable Precedence Hierarchy (last listed wins)



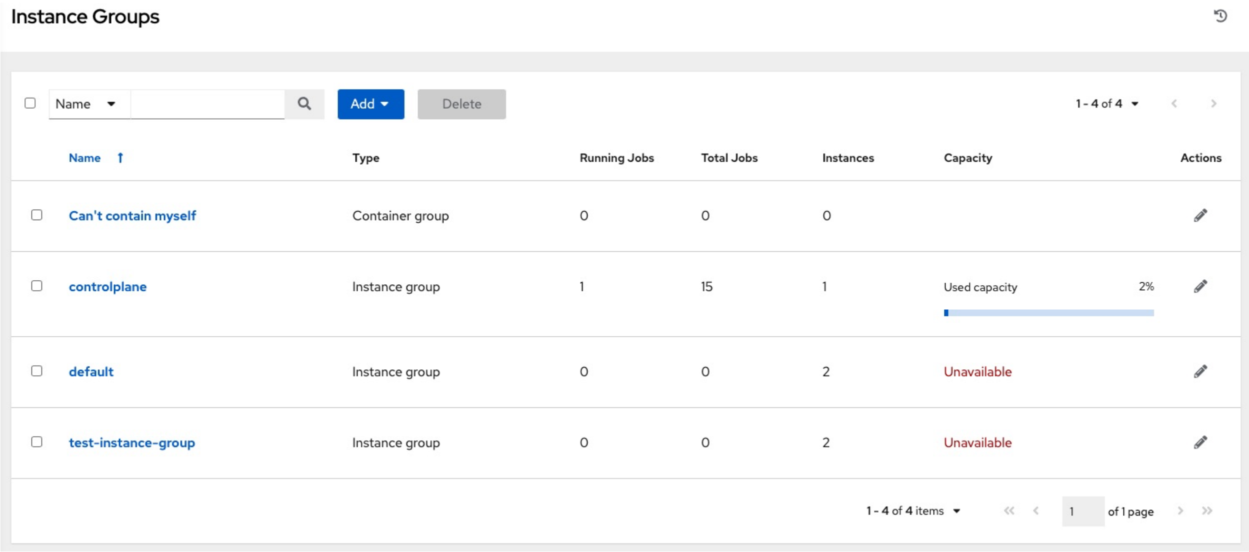
Ansible	Tower
set_stats (i.e. artifacts)	
	Job Artifacts
	Workflow Job Template extra variables
	Workflow Job Template Survey (defaults)
custom facts	Workflow Job Launch extra variables

CHAPTER

TWENTYFOUR

INSTANCE GROUPS

An *Instance Group* provides the ability to group instances in a clustered environment. Additionally, policies dictate how instance groups behave and how jobs are executed. The following view displays the capacity levels based on policy algorithms:



Instance Groups

□	Name ↑	Type	Running Jobs	Total Jobs	Instances	Capacity	Actions
□	Can't contain myself	Container group	0	0	0		✎
□	controlplane	Instance group	1	15	1	Used capacity 2%	✎
□	default	Instance group	0	0	2	Unavailable	✎
□	test-instance-group	Instance group	0	0	2	Unavailable	✎

For more information about the policy or rules associated with instance groups, see the Instance Groups section of the *Automation Controller Administration Guide*.

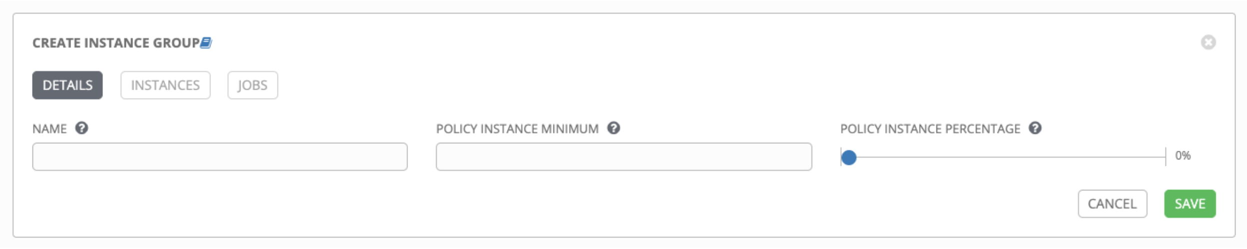
If you want to connect your instance group to a container, refer to Container Groups for further detail.

For an in-depth discussion on these concepts, refer to the *Feature Spotlight: Instance Groups and Isolated Nodes* blog.

24.1 Create an instance group

To create a new instance group:

1. Click **Instance Groups** from the left navigation menu to open the Instance Groups configuration window.
2. Click the **Add** button and select **Create Instance Group**.



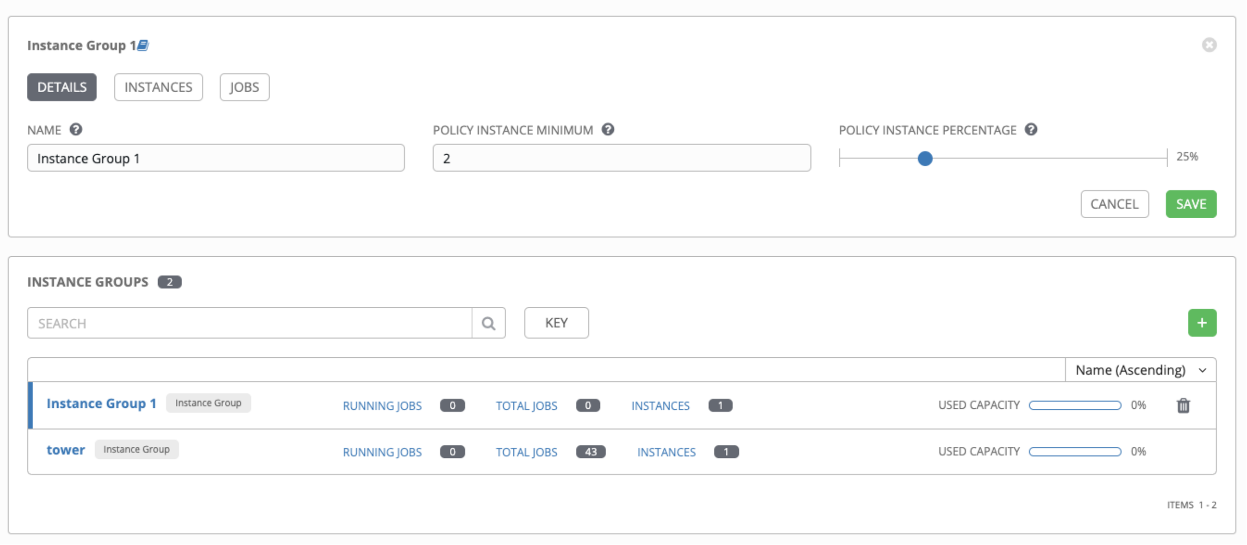
3. Enter the appropriate details into the following fields:

- **Name**. Names must be unique and must not be named *controller*.
- **Policy Instance Minimum**. Enter the minimum number of instances to automatically assign to this group when new instances come online.
- **Policy Instance Percentage**. Use the slider to select a minimum percentage of instances to automatically assign to this group when new instances come online.

Note: Policy Instance fields are not required to create a new instance group. If you do not specify values, then the Policy Instance Minimum and Policy Instance Percentage default to 0.

4. Click **Save**.

Once the instance group is successfully created, the **Details** tab of the newly created instance group remains, which allows you to review and edit your instance group information. This is the same menu that is opened if the Edit (✎) button is clicked from the **Instance Group** link. You can also edit **Instances** and review **Jobs** associated with this instance group.



NAME: Instance Group 1

POLICY INSTANCE MINIMUM: 2

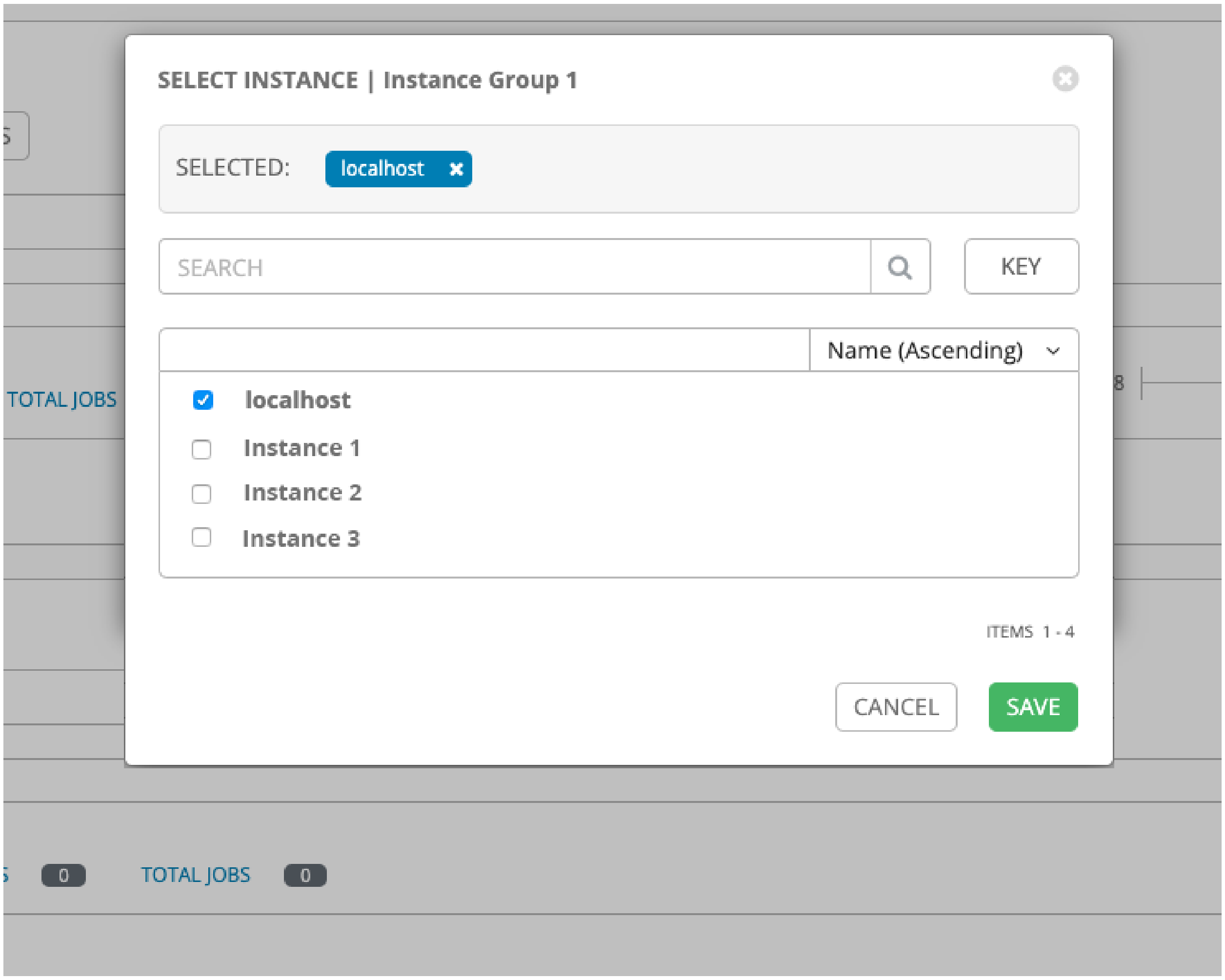
POLICY INSTANCE PERCENTAGE: 25%

Instance Group 1	Instance Group	RUNNING JOBS	0	TOTAL JOBS	0	INSTANCES	1	USED CAPACITY	0%
tower	Instance Group	RUNNING JOBS	0	TOTAL JOBS	43	INSTANCES	1	USED CAPACITY	0%

24.1.1 Associate instances to an instance group

To associate instances to an instance group:

1. Click the **Instances** tab of the Instance Group window and click the **Add** button.
2. Click the checkbox next to one or more available instances from the list to select the instance(s) you want to add to the instance group.

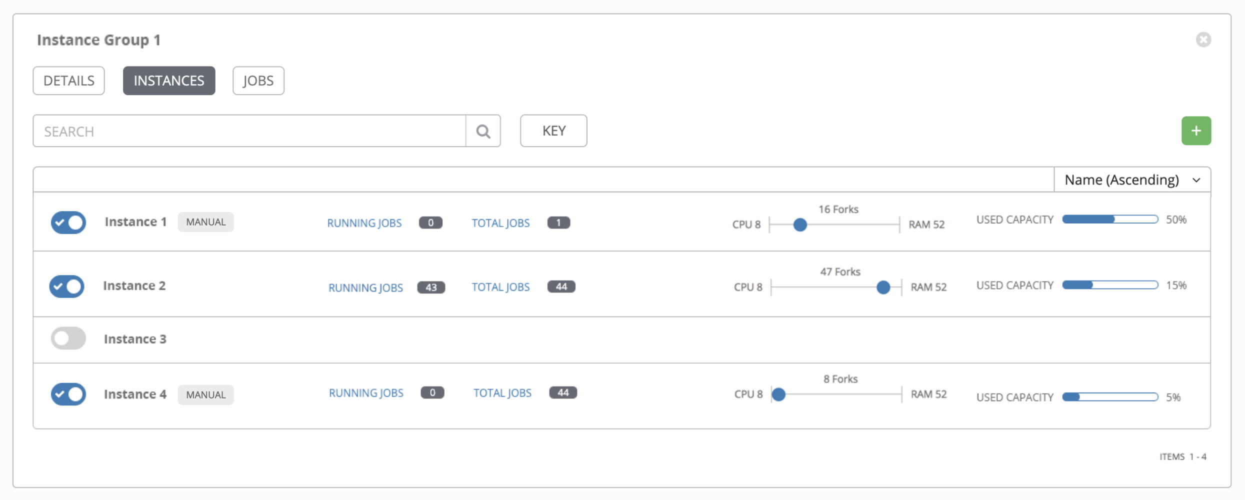


SELECT INSTANCE | Instance Group 1

SELECTED: localhost

☑ localhost
☐ Instance 1
☐ Instance 2
☐ Instance 3

3. In the following example, the instances added to the instance group displays along with information about their capacity.



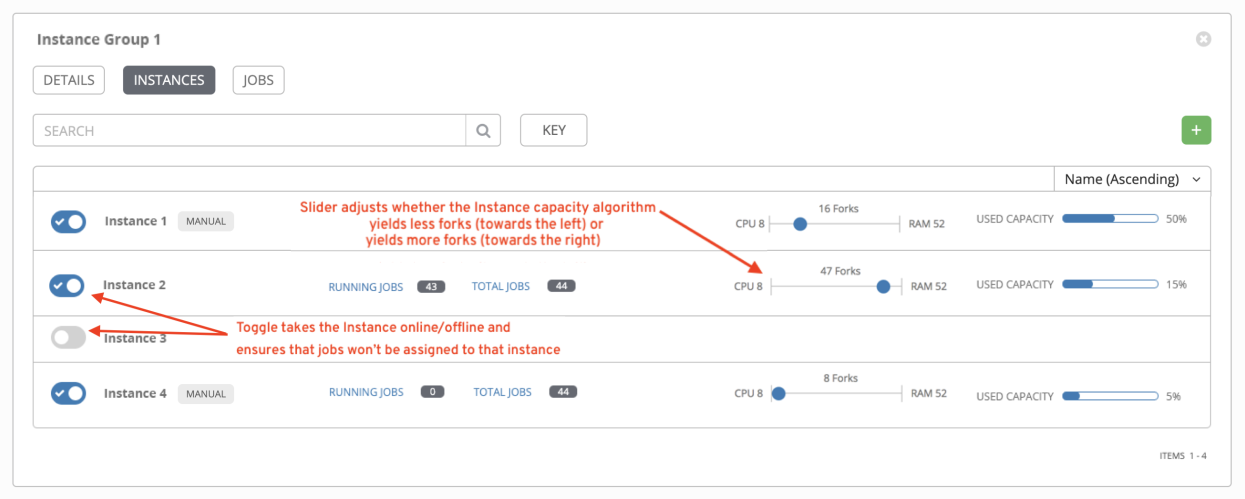
Instance 1 MANUAL RUNNING JOBS 0 TOTAL JOBS 1 CPU 8 16 Forks RAM 52 USED CAPACITY 50%

Instance 2 RUNNING JOBS 43 TOTAL JOBS 44 CPU 8 47 Forks RAM 52 USED CAPACITY 15%

Instance 3

Instance 4 MANUAL RUNNING JOBS 0 TOTAL JOBS 44 CPU 8 8 Forks RAM 52 USED CAPACITY 5%

This view also allows you to edit some key attributes associated with the instances in your instance group:



Instance 1 MANUAL RUNNING JOBS 0 TOTAL JOBS 1 CPU 8 16 Forks RAM 52 USED CAPACITY 50%

Instance 2 RUNNING JOBS 43 TOTAL JOBS 44 CPU 8 47 Forks RAM 52 USED CAPACITY 15%

Instance 3

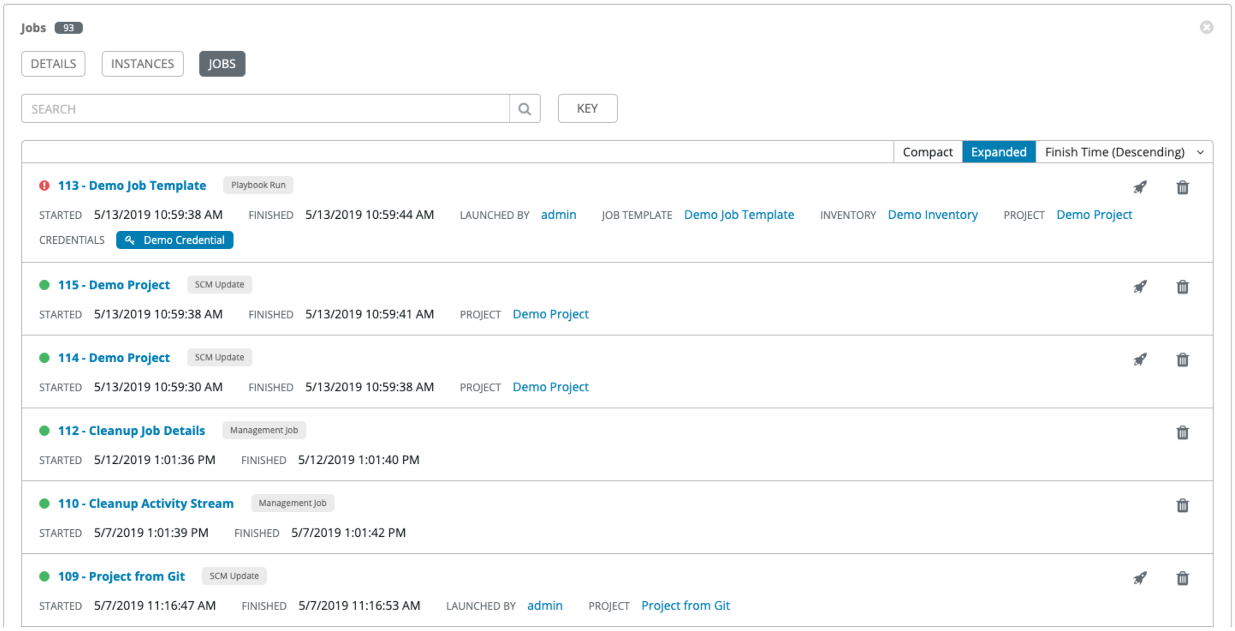
Instance 4 MANUAL RUNNING JOBS 0 TOTAL JOBS 44 CPU 8 8 Forks RAM 52 USED CAPACITY 5%

Slider adjusts whether the Instance capacity algorithm yields less forks (towards the left) or yields more forks (towards the right)

Toggle takes the Instance online/offline and ensures that jobs won't be assigned to that instance

24.1.2 View jobs associated with an instance group

To view the jobs associated with the instance group, click the **Jobs** tab of the Instance Group window and then click **Expanded** to expand the view to show details about each job.



113 - Demo Job Template Playbook Run
STARTED 5/13/2019 10:59:38 AM FINISHED 5/13/2019 10:59:44 AM LAUNCHED BY admin JOB TEMPLATE Demo Job Template INVENTORY Demo Inventory PROJECT Demo Project
CREDENTIALS Demo Credential

115 - Demo Project SCM Update
STARTED 5/13/2019 10:59:38 AM FINISHED 5/13/2019 10:59:41 AM PROJECT Demo Project

114 - Demo Project SCM Update
STARTED 5/13/2019 10:59:30 AM FINISHED 5/13/2019 10:59:38 AM PROJECT Demo Project

112 - Cleanup Job Details Management Job
STARTED 5/12/2019 1:01:36 PM FINISHED 5/12/2019 1:01:40 PM

110 - Cleanup Activity Stream Management Job
STARTED 5/7/2019 1:01:39 PM FINISHED 5/7/2019 1:01:42 PM

109 - Project from Git SCM Update
STARTED 5/7/2019 11:16:47 AM FINISHED 5/7/2019 11:16:53 AM LAUNCHED BY admin PROJECT Project from Git

Each job displays the job status, ID, and name; type of job, time started and completed, who started the job; and which template, inventory, project, and credential were used.

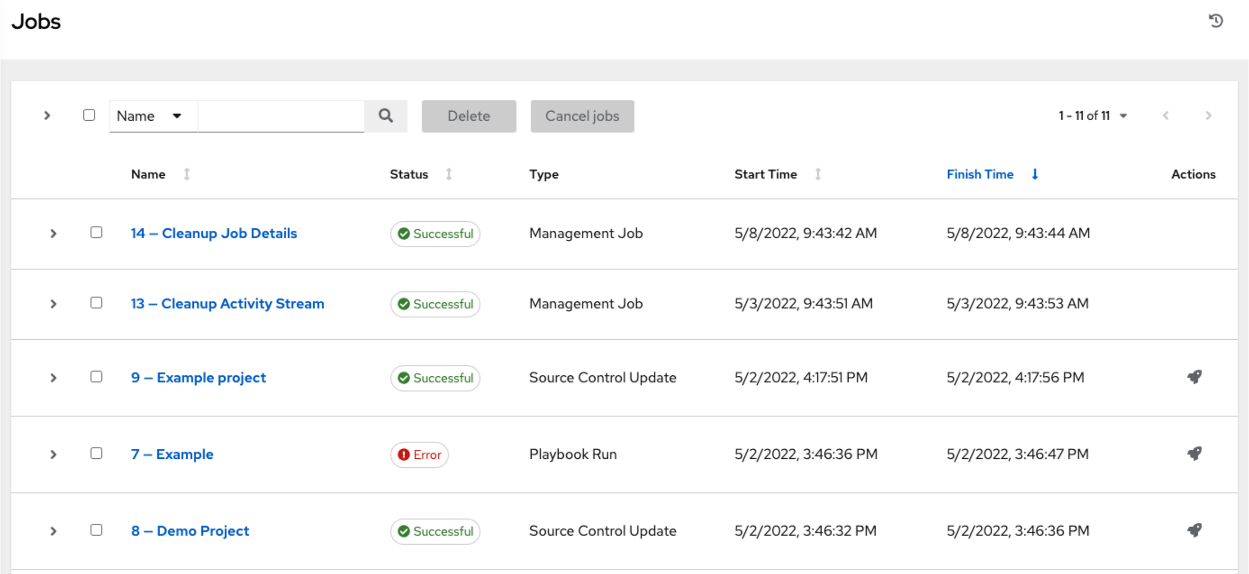
The instances are run in accordance with instance group policies. Refer to Instance Group Policies in the *Automation Controller Administration Guide*.

CHAPTER TWENTYFIVE

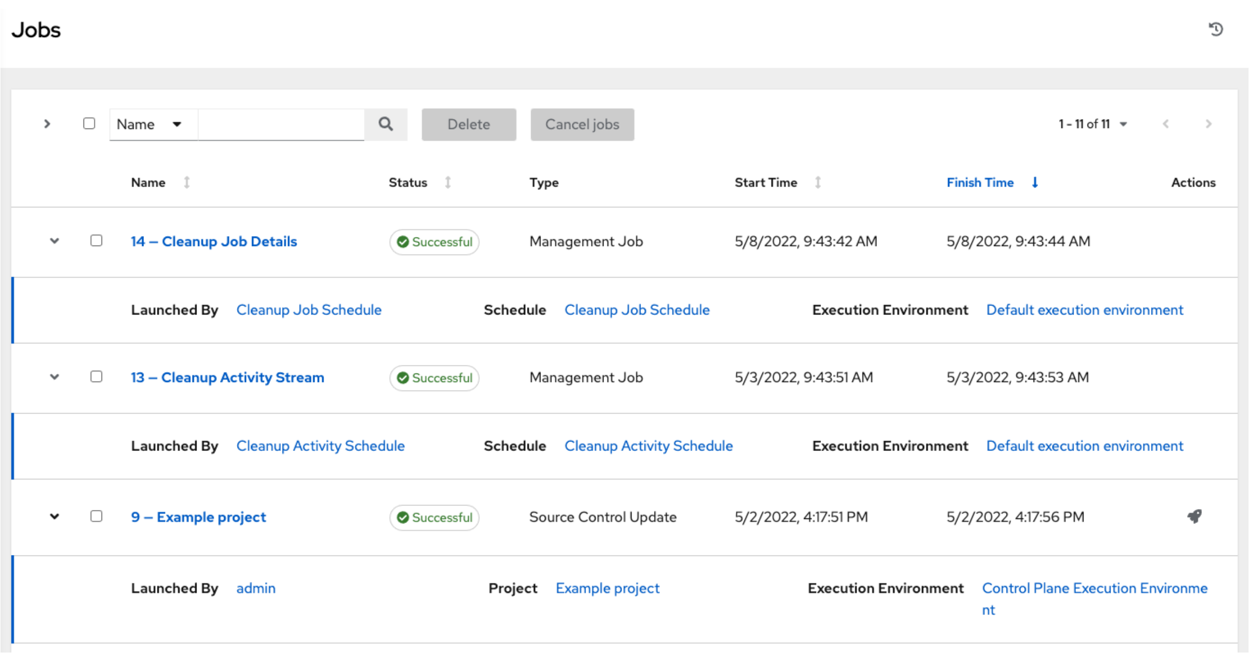
JOBS

A *job* is an instance of automation controller launching an Ansible playbook against an inventory of hosts.

The **Jobs** link displays a list of jobs and their statuses–shown as completed successfully or failed, or as an active (running) job. The default view is collapsed (**Compact**) with the job name, status, job type, and start/finish times, but you can expand to see more information. You can sort this list by various criteria, and perform a search to filter the jobs of interest.



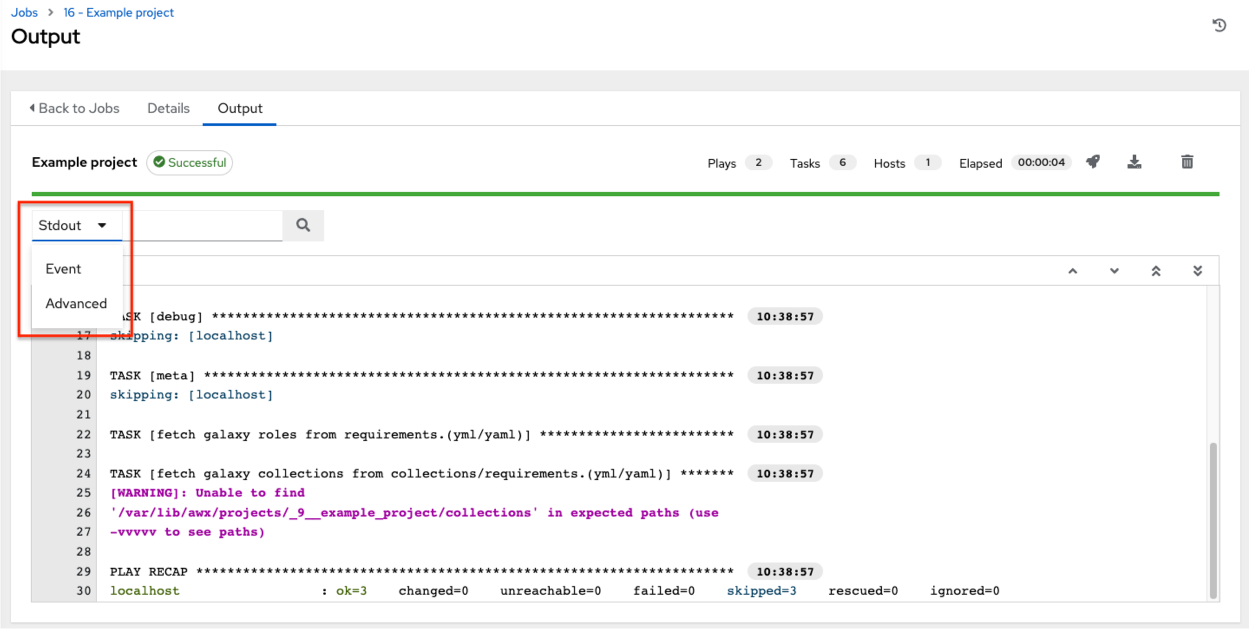
Name	Status	Type	Start Time	Finish Time	Actions
14 – Cleanup Job Details	Successful	Management Job	5/8/2022, 9:43:42 AM	5/8/2022, 9:43:44 AM	
13 – Cleanup Activity Stream	Successful	Management Job	5/3/2022, 9:43:51 AM	5/3/2022, 9:43:53 AM	
9 – Example project	Successful	Source Control Update	5/2/2022, 4:17:51 PM	5/2/2022, 4:17:56 PM	🚀
7 – Example	Error	Playbook Run	5/2/2022, 3:46:36 PM	5/2/2022, 3:46:47 PM	🚀
8 – Demo Project	Successful	Source Control Update	5/2/2022, 3:46:32 PM	5/2/2022, 3:46:36 PM	🚀



Name	Status	Type	Start Time	Finish Time	Actions
14 – Cleanup Job Details	✓ Successful	Management Job	5/8/2022, 9:43:42 AM	5/8/2022, 9:43:44 AM	
Launched By Cleanup Job Schedule		Schedule Cleanup Job Schedule		Execution Environment Default execution environment	
13 – Cleanup Activity Stream	✓ Successful	Management Job	5/3/2022, 9:43:51 AM	5/3/2022, 9:43:53 AM	
Launched By Cleanup Activity Schedule		Schedule Cleanup Activity Schedule		Execution Environment Default execution environment	
9 – Example project	✓ Successful	Source Control Update	5/2/2022, 4:17:51 PM	5/2/2022, 4:17:56 PM	🚀
Launched By admin		Project Example project		Execution Environment Control Plane Execution Environment	

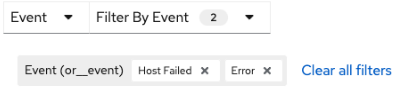
Actions you can take from this screen include viewing the details and standard output of a particular job, relaunching (🚀) jobs, or removing selected jobs. The relaunch operation only applies to relaunches of playbook runs and does not apply to project/inventory updates, system jobs, workflow jobs, etc.

When a job relaunches, you are directed the Jobs Output screen as the job runs. Clicking on any type of job also takes you to the Job Output View for that job, where you can filter jobs by various criteria:



```
ASK [debug] ********************************************************************  10:38:57
skipping: [localhost]

TASK [meta] *********************************************************************  10:38:57
skipping: [localhost]

TASK [fetch galaxy roles from requirements.(yml/yaml)] **************************  10:38:57

TASK [fetch galaxy collections from collections/requirements.(yml/yaml)] *******  10:38:57
[WARNING]: Unable to find
'/var/lib/awx/projects/_9__example_project/collections' in expected paths (use
-vvvvv to see paths)

PLAY RECAP **********************************************************************  10:38:57
localhost                  : ok=3    changed=0    unreachable=0    failed=0    skipped=3    rescued=0    ignored=0
```

- The **Stdout** option is the default display that shows the job processes and output
- The **Event** option allows you to filter by the event(s) of interest, such as errors, host failures, host retries, items skipped, etc. You can include as many events in the filter as necessary.



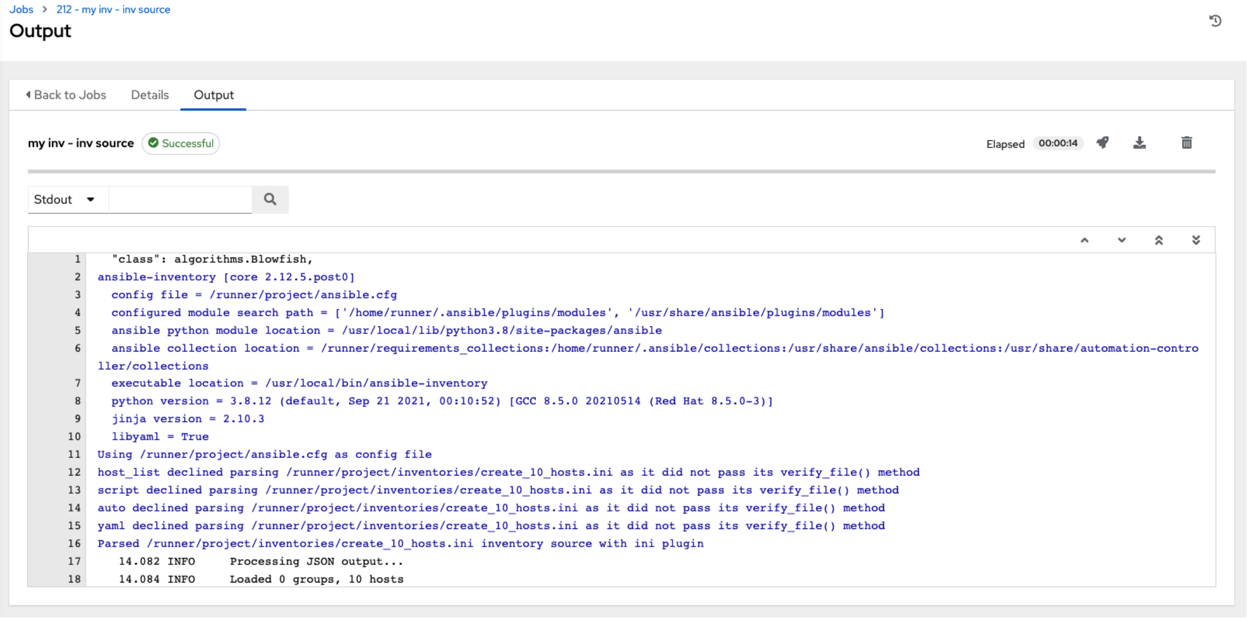
Event (or__event) Host Failed ✕ Error ✕

- The **Advanced** option is a refined search that allows you a combination of including or excluding criteria, searching by key, or by lookup type. For details about using Search, refer to the *Search* chapter.

25.1 Inventory Sync Jobs

When an inventory sync is executed, the full results automatically display in the Output tab. This shows the same information you would see if you ran it through the Ansible command line, and can be useful for debugging. The `ANSIBLE_DISPLAY_ARGS_TO_STDOUT` is set to `False` by default for all playbook runs. This matches Ansible's default behavior. This does not display task arguments in task headers in the Job Detail interface to avoid leaking certain sensitive module parameters to stdout. If you wish to restore the prior behavior (despite the security implications), you can set `ANSIBLE_DISPLAY_ARGS_TO_STDOUT` to `True` via the `AWX_TASK_ENV` configuration setting. For more details, refer to the ANSIBLE_DISPLAY_ARGS_TO_STDOUT.

The icons at the top right corner of the Output tab allow you to relaunch (🚀), download (📥) the job output, or delete (🗑) the job.



my inv - inv source ✓ Successful

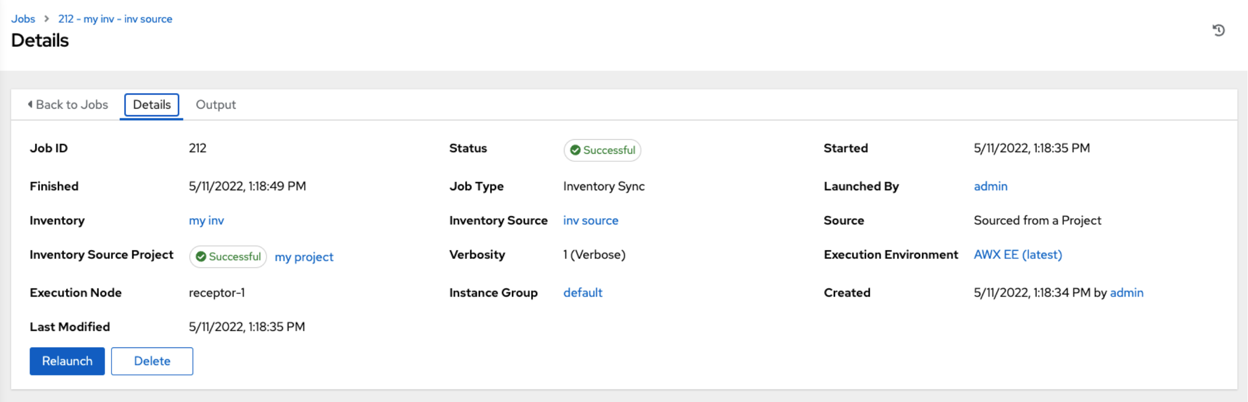
Elapsed 00:00:14

```
  "class": algorithms.Blowfish,
ansible-inventory [core 2.12.5.post0]
  config file = /runner/project/ansible.cfg
  configured module search path = ['/home/runner/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/local/lib/python3.8/site-packages/ansible
  ansible collection location = /runner/requirements_collections:/home/runner/.ansible/collections:/usr/share/ansible/collections:/usr/share/automation-contro
ller/collections
  executable location = /usr/local/bin/ansible-inventory
  python version = 3.8.12 (default, Sep 21 2021, 00:10:52) [GCC 8.5.0 20210514 (Red Hat 8.5.0-3)]
  jinja version = 2.10.3
  libyaml = True
Using /runner/project/ansible.cfg as config file
host_list declined parsing /runner/project/inventories/create_10_hosts.ini as it did not pass its verify_file() method
script declined parsing /runner/project/inventories/create_10_hosts.ini as it did not pass its verify_file() method
auto declined parsing /runner/project/inventories/create_10_hosts.ini as it did not pass its verify_file() method
yaml declined parsing /runner/project/inventories/create_10_hosts.ini as it did not pass its verify_file() method
Parsed /runner/project/inventories/create_10_hosts.ini inventory source with ini plugin
   14.082 INFO     Processing JSON output...
   14.084 INFO     Loaded 0 groups, 10 hosts
```

Note: An inventory update can be performed while a related job is running. In cases where you have a big project (around 10 GB), disk space on `/tmp` may be an issue.

25.1.1 Inventory sync details

Access the **Details** tab to provide details about the job execution.



Job ID	212	Status	Successful	Started	5/11/2022, 1:18:35 PM
Finished	5/11/2022, 1:18:49 PM	Job Type	Inventory Sync	Launched By	admin
Inventory	my inv	Inventory Source	inv source	Source	Sourced from a Project
Inventory Source Project	Successful my project	Verbosity	1 (Verbose)	Execution Environment	AWX EE (latest)
Execution Node	receptor-1	Instance Group	default	Created	5/11/2022, 1:18:34 PM by admin
Last Modified	5/11/2022, 1:18:35 PM				

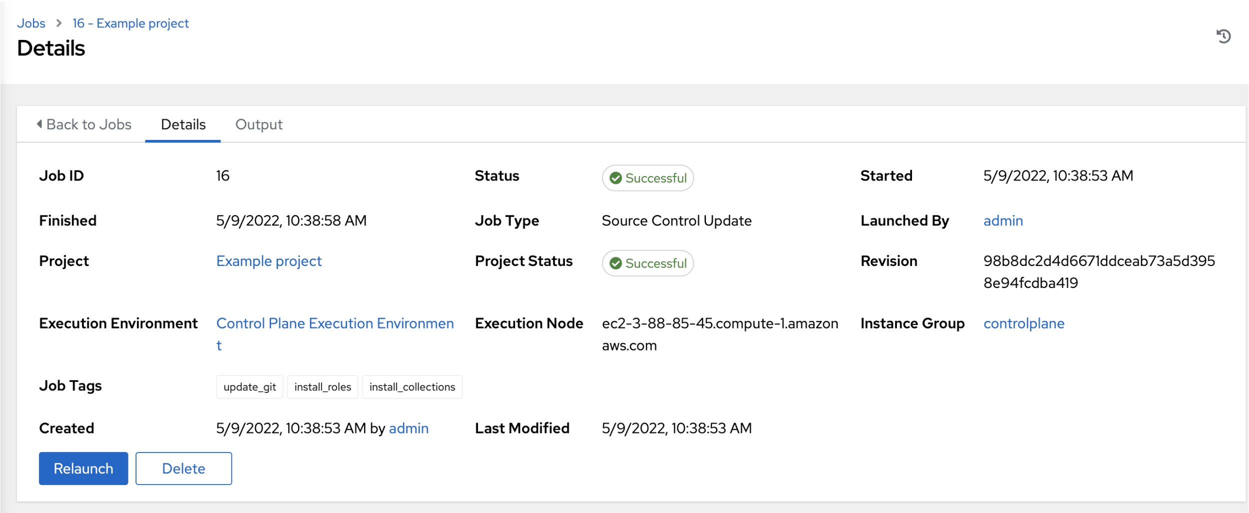
Notable details of the job executed are:

- **Status**: Can be any of the following:
  - **Pending** - The inventory sync has been created, but not queued or started yet. Any job, not just inventory source syncs, will stay in pending until it's actually ready to be run by the system. Reasons for inventory source syncs not being ready include dependencies that are currently running (all dependencies must be completed before the next step can execute), or there is not enough capacity to run in the locations it is configured to.
  - **Waiting** - The inventory sync is in the queue waiting to be executed.
  - **Running** - The inventory sync is currently in progress.
  - **Successful** - The inventory sync job succeeded.
  - **Failed** - The inventory sync job failed.
- **Inventory**: The name of the associated inventory group.
- **Source**: The type of cloud inventory.
- **Inventory Source Project**: The project used as the source of this inventory sync job.
- **Execution Environment**: The execution environment used.
- **Execution node**: The node used to execute the job.
- **Instance Group**: The name of the instance group used with this job (controller is the default instance group).

By clicking on these items, where appropriate, you can view the corresponding job templates, projects, and other objects.

25.2 SCM Inventory Jobs

When an inventory sourced from an SCM is executed, the full results automatically display in the Output tab. This shows the same information you would see if you ran it through the Ansible command line, and can be useful for debugging. The icons at the top right corner of the Output tab allow you to relaunch (🚀), download (📥) the job output, or delete (🗑) the job.



Jobs > 16 - Example project

Details

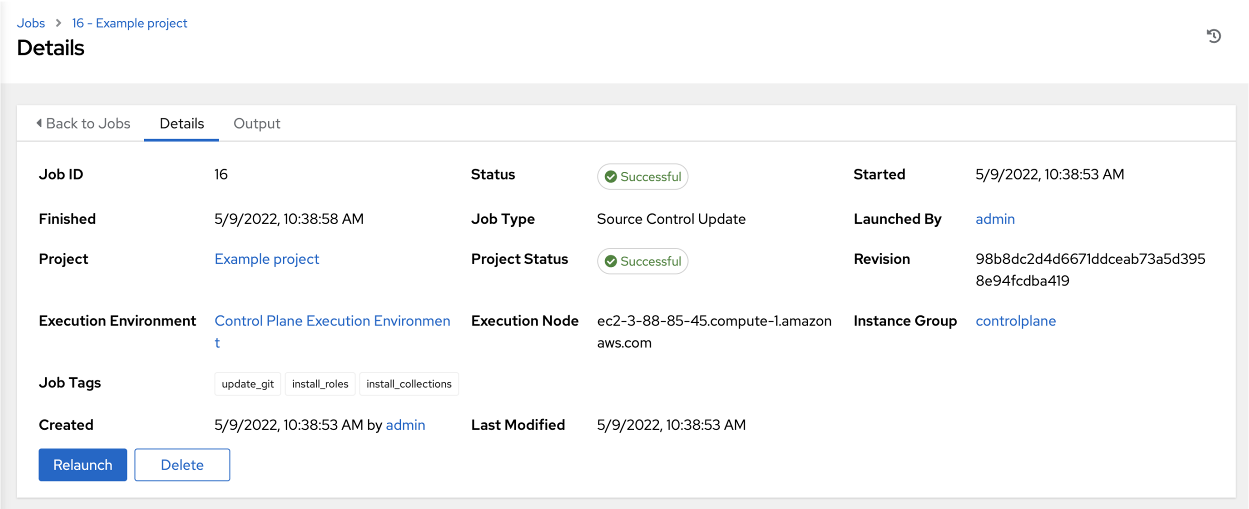
◂ Back to Jobs | Details | Output

Job ID: 16
Status: ✓ Successful
Started: 5/9/2022, 10:38:53 AM
Finished: 5/9/2022, 10:38:58 AM
Job Type: Source Control Update
Launched By: admin
Project: Example project
Project Status: ✓ Successful
Revision: 98b8dc2d4d6671ddceab73a5d395 8e94fcdba419
Execution Environment: Control Plane Execution Environment
Execution Node: ec2-3-88-85-45.compute-1.amazonaws.com
Instance Group: controlplane
Job Tags: update_git, install_roles, install_collections
Created: 5/9/2022, 10:38:53 AM by admin
Last Modified: 5/9/2022, 10:38:53 AM

Relaunch | Delete

25.2.1 SCM inventory details

Access the **Details** tab to provide details about the job execution and its associated project.



Jobs > 16 - Example project

Details

◂ Back to Jobs | Details | Output

Job ID: 16
Status: ✓ Successful
Started: 5/9/2022, 10:38:53 AM
Finished: 5/9/2022, 10:38:58 AM
Job Type: Source Control Update
Launched By: admin
Project: Example project
Project Status: ✓ Successful
Revision: 98b8dc2d4d6671ddceab73a5d395 8e94fcdba419
Execution Environment: Control Plane Execution Environment
Execution Node: ec2-3-88-85-45.compute-1.amazonaws.com
Instance Group: controlplane
Job Tags: update_git, install_roles, install_collections
Created: 5/9/2022, 10:38:53 AM by admin
Last Modified: 5/9/2022, 10:38:53 AM

Relaunch | Delete

Notable details of the job executed are:

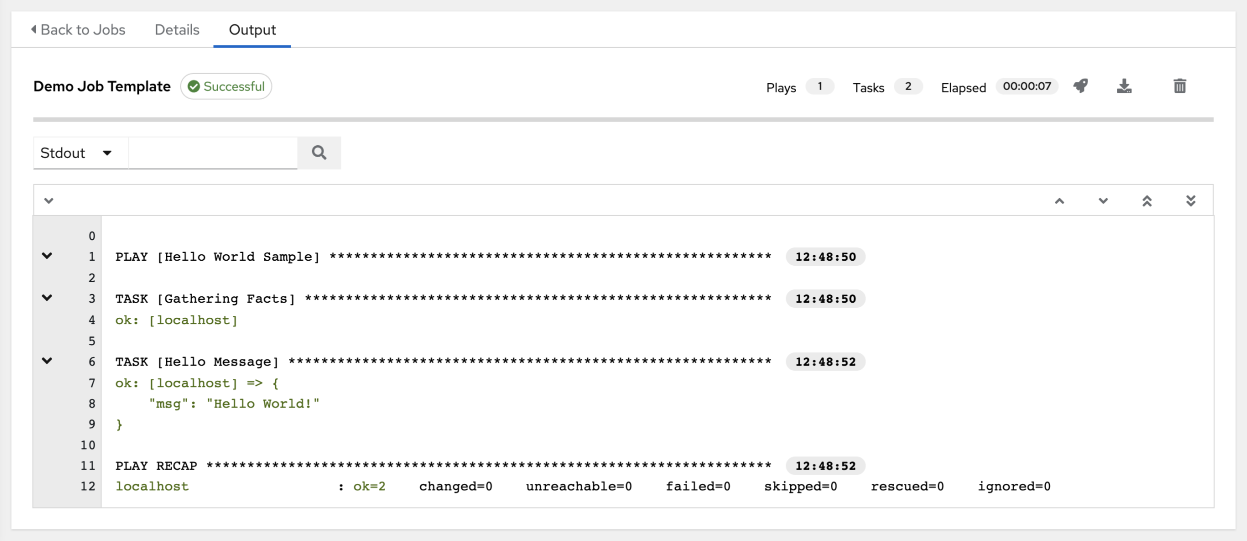
- **Status**: Can be any of the following:
  - **Pending** - The SCM job has been created, but not queued or started yet. Any job, not just SCM jobs, will stay in pending until it's actually ready to be run by the system. Reasons for SCM jobs not being ready include dependencies that are currently running (all dependencies must be completed before the next step can execute), or there is not enough capacity to run in the locations it is configured to.

  - **Waiting** - The SCM job is in the queue waiting to be executed.
  - **Running** - The SCM job is currently in progress.
  - **Successful** - The last SCM job succeeded.
  - **Failed** - The last SCM job failed.
- **Job Type**: SCM jobs display Source Control Update.
- **Project**: The name of the project.
- **Project Status**: Indicates whether the associated project was successfully updated.
- **Revision**: Indicates the revision number of the sourced project that was used in this job.
- **Execution Environment**: Specifies the execution environment used to run this job.
- **Execution Node**: Indicates the node on which the job ran.
- **Instance Group**: Indicates the instance group on which the job ran, if specified.
- **Job Tags**: Tags show the various job operations executed.

By clicking on these items, where appropriate, you can view the corresponding job templates, projects, and other objects.

25.3 Playbook Run Jobs

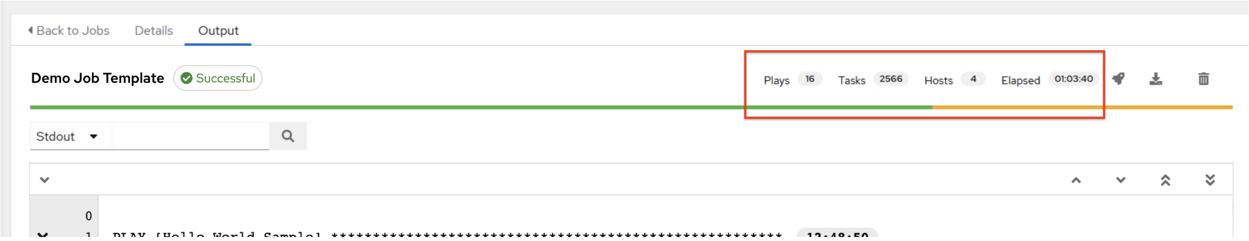
When a playbook is executed, the full results automatically display in the Output tab. This shows the same information you would see if you ran it through the Ansible command line, and can be useful for debugging.



```

PLAY [Hello World Sample] ****************************************************** 12:48:50

TASK [Gathering Facts] ********************************************************* 12:48:50
ok: [localhost]

TASK [Hello Message] *********************************************************** 12:48:52
ok: [localhost] => {
    "msg": "Hello World!"
}

PLAY RECAP ********************************************************************* 12:48:52
localhost                  : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

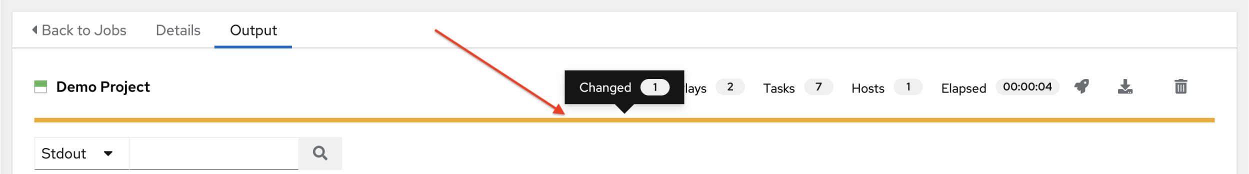
The events summary captures a tally of events that were run as part of this playbook:

- the number of times this playbook has ran in the **Plays** field
- the number of tasks associated with this playbook in the **Tasks** field
- the number of hosts associated with this playbook in the **Hosts** field
- the amount of time it took to complete the playbook run in the **Elapsed** field



The icons next to the events summary allow you to relaunch (🚀), download (📥) the job output, or delete (🗑) the job.

The host status bar runs across the top of the Output view. Hover over a section of the host status bar and the number of hosts associated with that particular status displays.



The output for a Playbook job is also accessible after launching a job from the **Jobs** tab of its Job Templates page.

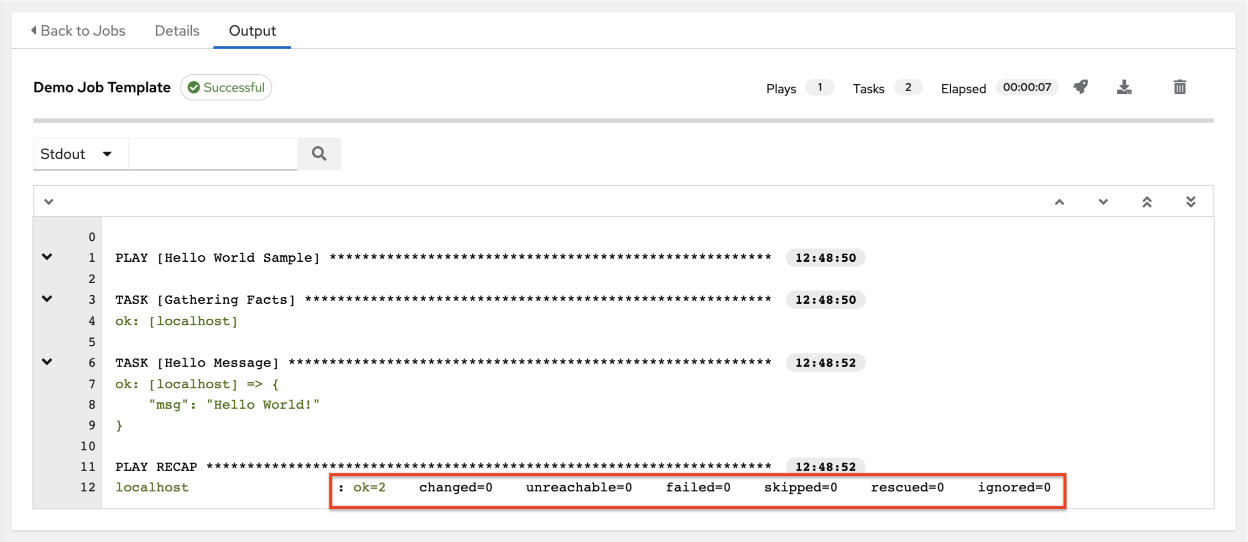
Clicking on the various line item tasks in the output, you can view its host details.

25.3.1 Search

Use Search to look up specific events, hostnames, and their statuses. To filter only certain hosts with a particular status, specify one of the following valid statuses:

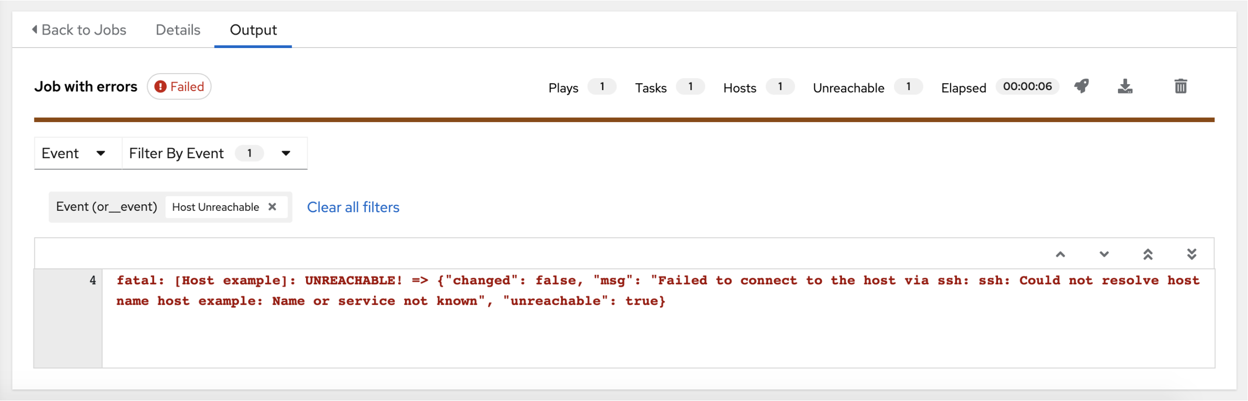
- **OK**: the playbook task returned "Ok".
- **Changed**: the playbook task actually executed. Since Ansible tasks should be written to be idempotent, tasks may exit successfully without executing anything on the host. In these cases, the task would return Ok, but not Changed.
- **Failed**: the task failed. Further playbook execution was stopped for this host.
- **Unreachable**: the host was unreachable from the network or had another fatal error associated with it.
- **Skipped**: the playbook task was skipped because no change was necessary for the host to reach the target state.
- **Rescued**: introduced in Ansible 2.8, this shows the tasks that failed and then executes a rescue section.
- **Ignored**: introduced in Ansible 2.8, this shows the tasks that failed and have `ignore_errors: yes` configured.

These statuses also display at bottom of each Stdout pane, in a group of "stats" called the Host Summary fields.



```
PLAY [Hello World Sample] *******************************************************   12:48:50

TASK [Gathering Facts] **********************************************************   12:48:50
ok: [localhost]

TASK [Hello Message] ************************************************************   12:48:52
ok: [localhost] => {
    "msg": "Hello World!"
}

PLAY RECAP **********************************************************************   12:48:52
localhost                  : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

The example below shows a search with only unreachable hosts.

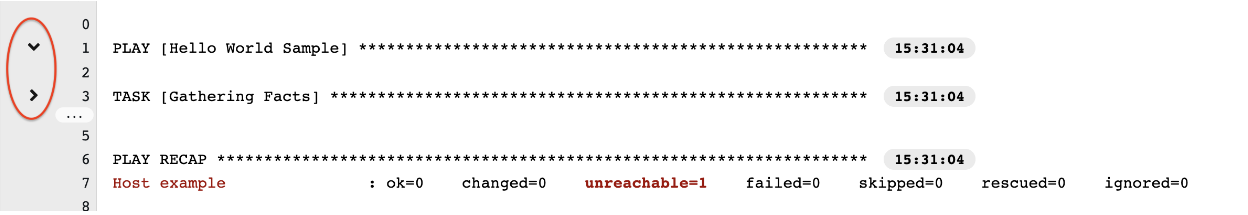


```
fatal: [Host example]: UNREACHABLE! => {"changed": false, "msg": "Failed to connect to the host via ssh: ssh: Could not resolve host
name host example: Name or service not known", "unreachable": true}
```

For more details about using the Search, refer to the *Search* chapter.

The standard output view displays all the events that occur on a particular job. By default, all rows are expanded so that all the details are displayed. Use the collapse-all button (−) to switch to a view that only contains the headers for plays and tasks. Click the (+) button to view all lines of the standard output.

Alternatively, you can display all the details of a specific play or task by clicking on the arrow icons next to them. Click an arrow from sideways to downward to expand the lines associated with that play or task. Click the arrow back to the sideways position to collapse and hide the lines.



```
PLAY [Hello World Sample] *******************************************************   15:31:04

TASK [Gathering Facts] **********************************************************   15:31:04
...

PLAY RECAP **********************************************************************   15:31:04
Host example               : ok=0    changed=0    unreachable=1    failed=0    skipped=0    rescued=0    ignored=0
```

Things to note when viewing details in the expand/collapse mode:

- Each displayed line that is not collapsed has a corresponding line number and start time.
- An expand/collapse icon is at the start of any play or task after the play or task has completed.

- If querying for a particular play or task, it will appear collapsed at the end of its completed process.
- In some cases, an error message will appear, stating that the output may be too large to display. This occurs when there are more than 4000 events. Use the search and filter for specific events to bypass the error.

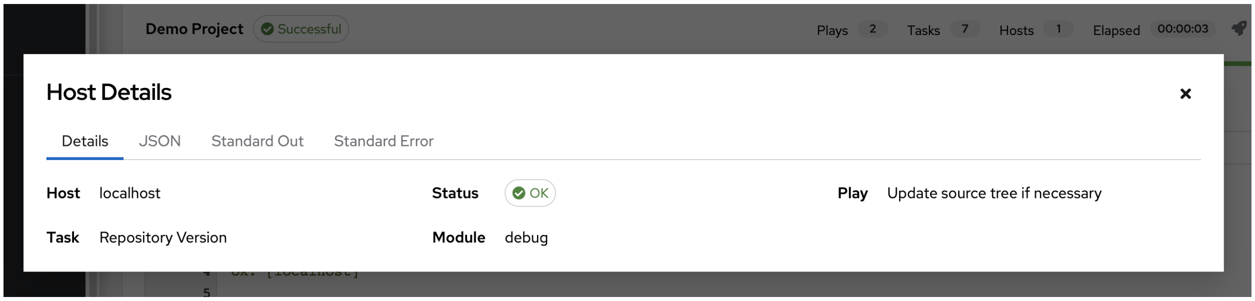
Click on a line of an event from the **Standard Out** pane and a **Host Events** dialog displays in a separate window. This window shows the host that was affected by that particular event.

Note: Upgrading to the latest versions of Ansible Automation Platform involves progressively migrating all historical playbook output and events. This migration process is gradual, and happens automatically in the background after installation is complete. Installations with very large amounts of historical job output (tens, or hundreds of GB of output) may notice missing job output until migration is complete. Most recent data will show up at the top of the output, followed by older events. Migrating jobs with a large amount of events may take longer than jobs with a smaller amount.

25.3.2 Host Details

The **Host Details** dialog shows information about the host affected by the selected event and its associated play and task:

- the **Host**
- the **Status**
- the type of run in the **Play** field
- the type of **Task**
- if applicable, the Ansible **Module** for the task, and any *arguments* for that module

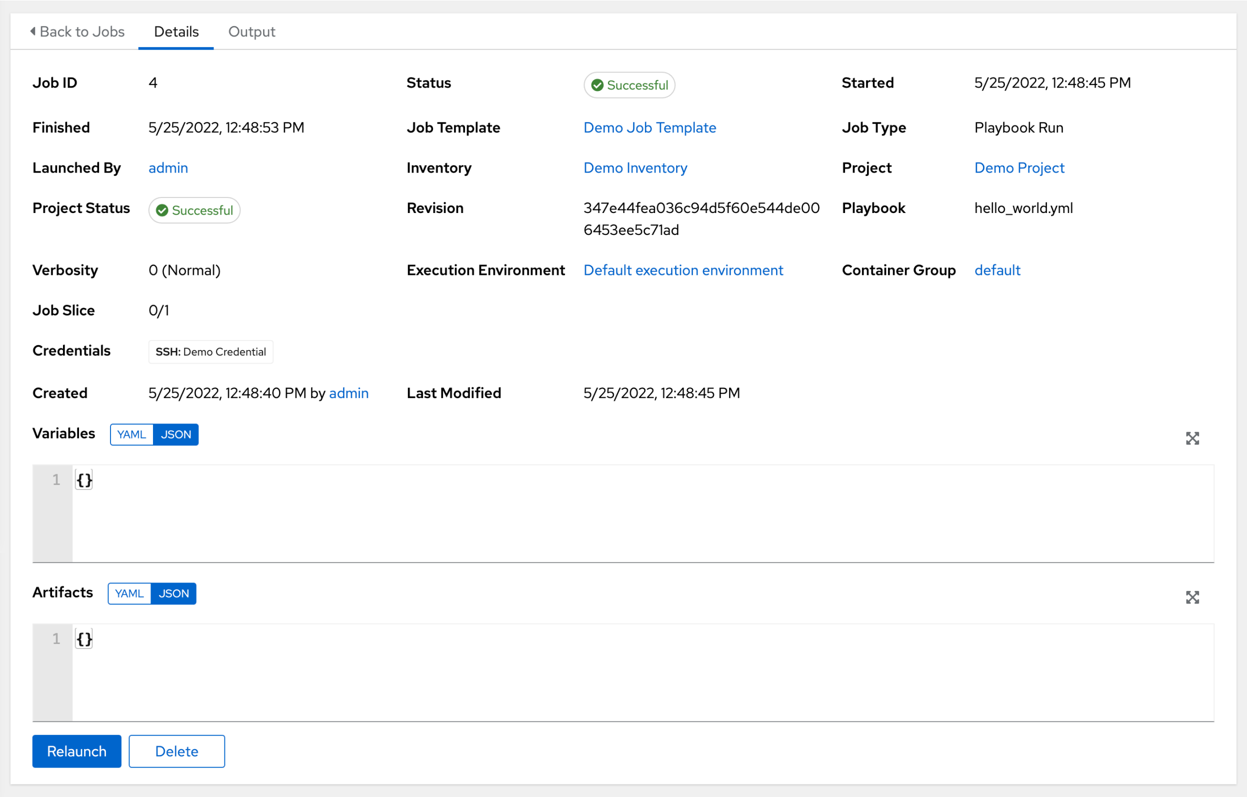


Host localhost
Status OK
Play Update source tree if necessary
Task Repository Version
Module debug

To view the results in JSON format, click on the **JSON** tab. To view the output of the task, click the **Standard Out**. To view errors from the output, click **Standard Error**.

25.3.3 Playbook run details

Access the **Details** tab to provide details about the job execution.



Job ID	4	Status	Successful	Started	5/25/2022, 12:48:45 PM
Finished	5/25/2022, 12:48:53 PM	Job Template	Demo Job Template	Job Type	Playbook Run
Launched By	admin	Inventory	Demo Inventory	Project	Demo Project
Project Status	Successful	Revision	347e44fea036c94d5f60e544de006453ee5c71ad	Playbook	hello_world.yml
Verbosity	0 (Normal)	Execution Environment	Default execution environment	Container Group	default
Job Slice	0/1				
Credentials	SSH: Demo Credential				
Created	5/25/2022, 12:48:40 PM by admin	Last Modified	5/25/2022, 12:48:45 PM		

Variables YAML JSON

```
{}
```

Artifacts YAML JSON

```
{}
```

Notable details of the job executed are:

- **Status**: Can be any of the following:
  - **Pending** - The playbook run has been created, but not queued or started yet. Any job, not just playbook runs, will stay in pending until it is actually ready to be run by the system. Reasons for playbook runs not being ready include dependencies that are currently running (all dependencies must be completed before the next step can execute), or there is not enough capacity to run in the locations it is configured to.
  - **Waiting** - The playbook run is in the queue waiting to be executed.
  - **Running** - The playbook run is currently in progress.
  - **Successful** - The last playbook run succeeded.
  - **Failed** - The last playbook run failed.
- **Job Template**: The name of the job template from which this job was launched.
- **Inventory**: The inventory selected to run this job against.
- **Project**: The name of the project associated with the launched job.
- **Project Status**: The status of the project associated with the launched job.
- **Playbook**: The playbook used to launch this job.
- **Execution Environment**: The name of the execution environment used in this job.
- **Container Group**: The name of the container group used in this job.
- **Credentials**: The credential(s) used in this job.
- **Extra Variables**: Any extra variables passed when creating the job template are displayed here.

By clicking on these items, where appropriate, you can view the corresponding job templates, projects, and other objects.

25.4 Automation Controller Capacity Determination and Job Impact

This section describes how to determine capacity for instance groups and its impact to your jobs. For container groups, see Container capacity limits in the *Automation Controller Administration Guide*.

The automation controller capacity system determines how many jobs can run on an instance given the amount of resources available to the instance and the size of the jobs that are running (referred to as *Impact*). The algorithm used to determine this is based entirely on two things:

- How much memory is available to the system (`mem_capacity`)
- How much CPU is available to the system (`cpu_capacity`)

Capacity also impacts Instance Groups. Since Groups are made up of instances, likewise, instances can be assigned to multiple groups. This means that impact to one instance can potentially affect the overall capacity of other Groups.

Instance Groups (not instances themselves) can be assigned to be used by jobs at various levels (see Clustering). When the Task Manager is preparing its graph to determine which group a job will run on, it will commit the capacity of an Instance Group to a job that hasn't or isn't ready to start yet.

Finally, in smaller configurations, if only one instance is available for a job to run, the Task Manager will allow that job to run on the instance even if it pushes the instance over capacity. This guarantees that jobs themselves won't get stuck as a result of an under-provisioned system.

Therefore, Capacity and Impact is not a zero-sum system relative to jobs and instances/Instance Groups.

For information on sliced jobs and their impact to capacity, see *Job slice execution behavior*.

25.4.1 Resource determination for capacity algorithm

The capacity algorithms are defined in order to determine how many forks a system is capable of running simultaneously. This controls how many systems Ansible itself will communicate with simultaneously. Increasing the number of forks a automation controller system is running will, in general, allow jobs to run faster by performing more work in parallel. The trade-off is that this will increase the load on the system, which could cause work to slow down overall.

Automation controller can operate in two modes when determining capacity. `mem_capacity` (the default) will allow you to over-commit CPU resources while protecting the system from running out of memory. If most of your work is not CPU-bound, then selecting this mode will maximize the number of forks.

Memory relative capacity

`mem_capacity` is calculated relative to the amount of memory needed per fork. Taking into account the overhead for internal components, this comes out to be about 100MB per fork. When considering the amount of memory available to Ansible jobs, the capacity algorithm will reserve 2GB of memory to account for the presence of other services. The algorithm formula for this is:

```
(mem - 2048) / mem_per_fork
```

As an example:

```
(4096 - 2048) / 100 == ~20
```

Therefore, a system with 4GB of memory would be capable of running 20 forks. The value mem_per_fork can be controlled by setting the settings value (or environment variable) SYSTEM_TASK_FORKS_MEM, which defaults to 100.

CPU relative capacity

Often, Ansible workloads can be fairly CPU-bound. In these cases, sometimes reducing the simultaneous workload allows more tasks to run faster and reduces the average time-to-completion of those jobs.

Just as the mem_capacity algorithm uses the amount of memory need per fork, the cpu_capacity algorithm looks at the amount of CPU resources is needed per fork. The baseline value for this is 4 forks per core. The algorithm formula for this is:

```
cpus * fork_per_cpu
```

For example, a 4-core system:

```
4 * 4 == 16
```

The value fork_per_cpu can be controlled by setting the settings value (or environment variable) SYSTEM_TASK_FORKS_CPU which defaults to 4.

25.4.2 Capacity job impacts

When selecting the capacity, it's important to understand how each job type affects capacity.

It's helpful to understand what forks mean to Ansible: https://www.ansible.com/blog/ansible-performance-tuning (see the section on "Know Your Forks").

The default forks value for Ansible is 5. However, if automation controller knows that you're running against fewer systems than that, then the actual concurrency value will be lower.

When a job is run, automation controller will add 1 to the number of forks selected to compensate for the Ansible parent process. So if you are running a playbook against 5 systems with a forks value of 5, then the actual forks value from the perspective of Job Impact will be 6.

Impact of job types in automation controller

Jobs and Ad-hoc jobs follow the above model, forks + 1. If you set a fork value on your job template, your job capacity value will be the minimum of the forks value supplied, and the number of hosts that you have, plus one. The plus one is to account for the parent Ansible process.

Instance capacity determines which jobs get assigned to any specific instance. Jobs and ad hoc commands use more capacity if they have a higher forks value.

Other job types have a fixed impact:

- Inventory Updates: 1
- Project Updates: 1
- System Jobs: 5

If you don't set a forks value on your job template, your job will use Ansible's default forks value of five. Even though Ansible defaults to five forks, it will use fewer if your job has fewer than five hosts. In general, setting a forks value higher than what the system is capable of could cause trouble by running out of memory or over-committing CPU. So, the job template fork values that you use should fit on the system. If you have playbooks using 1000 forks but none of your systems individually has that much capacity, then your systems are undersized and at risk of performance or resource issues.

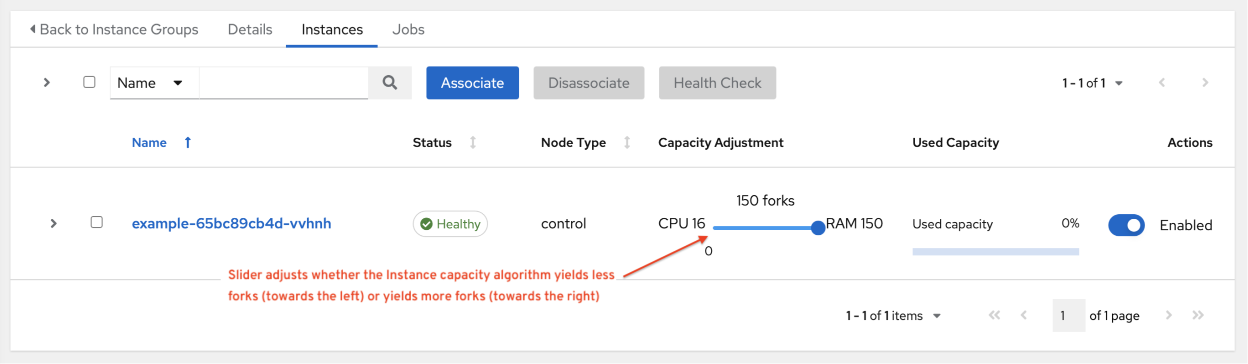
Selecting the right capacity

Selecting a capacity out of the CPU-bound or the memory-bound capacity limits is, in essence, selecting between the minimum or maximum number of forks. In the above examples, the CPU capacity would allow a maximum of 16 forks while the memory capacity would allow 20. For some systems, the disparity between these can be large and often times you may want to have a balance between these two.

The instance field `capacity_adjustment` allows you to select how much of one or the other you want to consider. It is represented as a value between 0.0 and 1.0. If set to a value of 1.0, then the largest value will be used. The above example involves memory capacity, so a value of 20 forks would be selected. If set to a value of 0.0 then the smallest value will be used. A value of 0.5 would be a 50/50 balance between the two algorithms which would be 18:

```
16 + (20 - 16) * 0.5 == 18
```

To view or edit the capacity in the user interface, select the **Instances** tab of the Instance Group.

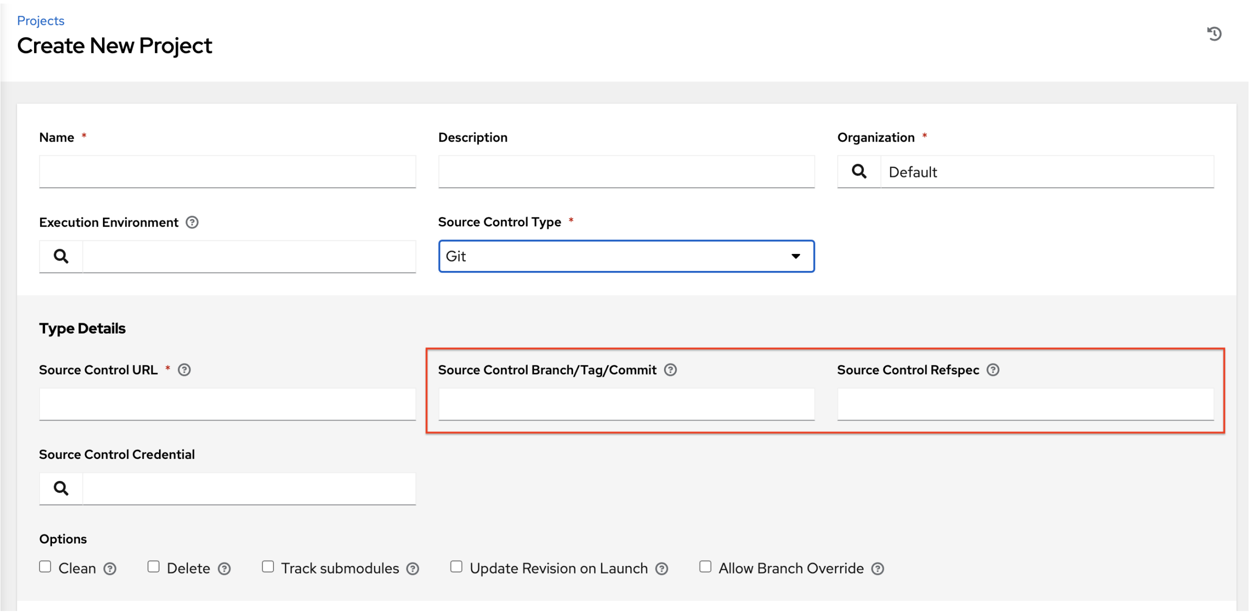


Name	Status	Node Type	Capacity Adjustment	Used Capacity	Actions
example-65bc89cb4d-vvhnh	Healthy	control	CPU 16 — 150 forks — RAM 150 0	Used capacity 0%	Enabled

Slider adjusts whether the Instance capacity algorithm yields less forks (towards the left) or yields more forks (towards the right)

25.5 Job branch overriding

Projects specify the branch, tag, or reference to use from source control in the `scm_branch` field. These are represented by the values specified in the Project Details fields as shown.



Projects

Create New Project

Name *

Description

Organization * Default

Execution Environment

Source Control Type * Git

Type Details

Source Control URL *

Source Control Branch/Tag/Commit

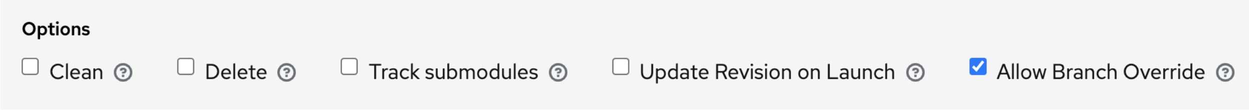
Source Control Refspec

Source Control Credential

Options

□ Clean □ Delete □ Track submodules □ Update Revision on Launch □ Allow Branch Override

Projects have the option to "Allow Branch Override". When checked, project admins can delegate branch selection to the job templates that use that project (requiring only project `use_role`).



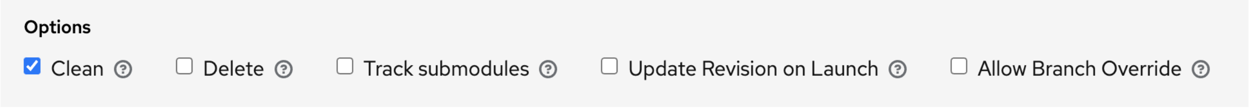
Options

□ Clean □ Delete □ Track submodules □ Update Revision on Launch ☑ Allow Branch Override

25.5.1 Source tree copy behavior

Every job run has its own private data directory. This directory contains a copy of the project source tree for the given `scm_branch` the job is running. Jobs are free to make changes to the project folder and make use of those changes while it is still running. This folder is temporary and is cleaned up at the end of the job run.

If **Clean** is checked, automation controller discards modified files in its local copy of the repository through use of the `force` parameter in its respective Ansible modules pertaining to git or Subversion.

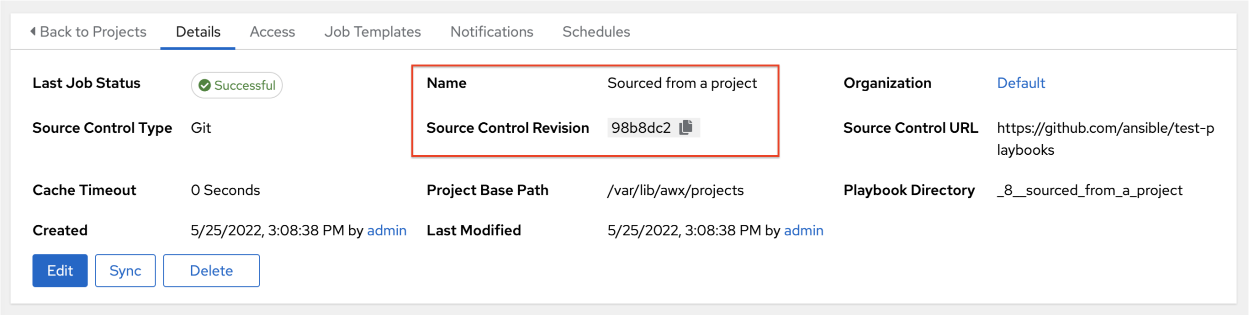


Options

☑ Clean □ Delete □ Track submodules □ Update Revision on Launch □ Allow Branch Override

25.5.2 Project revision behavior

Typically, during a project update, the revision of the default branch (specified in the **SCM Branch** field of the project) is stored when updated, and jobs using that project will employ this revision. Providing a non-default **SCM Branch** (not a commit hash or tag) in a job, the newest revision is pulled from the source control remote immediately before the job starts. This revision is shown in the **Source Control Revision** field of the job and its respective project update.



Last Job Status	Successful	Name	Sourced from a project	Organization	Default
Source Control Type	Git	Source Control Revision	98b8dc2	Source Control URL	https://github.com/ansible/test-playbooks
Cache Timeout	0 Seconds	Project Base Path	/var/lib/awx/projects	Playbook Directory	_8__sourced_from_a_project
Created	5/25/2022, 3:08:38 PM by admin	Last Modified	5/25/2022, 3:08:38 PM by admin		

Consequently, offline job runs are impossible for non-default branches. To be sure that a job is running a static version from source control, use tags or commit hashes. Project updates do not save the revision of all branches, only the project default branch.

The **SCM Branch** field is not validated, so the project must update to assure it is valid. If this field is provided or prompted for, the **Playbook** field of job templates will not be validated, and you will have to launch the job template in order to verify presence of the expected playbook.

25.5.3 Git Refspec

The **SCM Refspec** field specifies which extra references the update should download from the remote. Examples are:

1. `refs/*:refs/remotes/origin/*`: fetches all references, including remotes of the remote
2. `refs/pull/*:refs/remotes/origin/pull/*` (GitHub-specific): fetches all refs for all pull requests
3. `refs/pull/62/head:refs/remotes/origin/pull/62/head`: fetches the ref for that one GitHub pull request

For large projects, you should consider performance impact when using the 1st or 2nd examples here.

The **SCM Refspec** parameter affects the availability of the project branch, and can allow access to references not otherwise available. The examples above allow the user to supply a pull request from the **SCM Branch**, which would not be possible without the **SCM Refspec** field.

The Ansible git module fetches `refs/heads/*` by default. This means that a project's branches and tags (and commit hashes therein) can be used as the SCM Branch if **SCM Refspec** is blank. The value specified in the **SCM Refspec** field affects which **SCM Branch** fields can be used as overrides. Project updates (of any type) will perform an extra `git fetch` command to pull that refspec from the remote.

For example: You could set up a project that allows branch override with the 1st or 2nd refspec example –> Use this in a job template that prompts for the **SCM Branch** –> A client could launch the job template when a new pull request is created, providing the branch `pull/N/head` –> The job template would run against the provided GitGub pull request reference.

For more information on the Ansible git module, see https://docs.ansible.com/ansible/latest/modules/git_module.html.

CHAPTER

TWENTYSIX

WORKING WITH WEBHOOKS

A *Webhook* provides the ability to execute specified commands between apps over the web. automation controller currently provides webhook integration with GitHub and GitLab. This section describes the procedure for setting up a webhook through their respective services.

The webhook post-status-back functionality for GitHub and GitLab is designed for work only under certain CI events. Receiving another kind of event will result in messages like the one below in the service log:

```
awx.main.models.mixins Webhook event did not have a status API endpoint associated,␣
↪skipping.
```

26.1 GitHub webhook setup

Automation controller has the ability to run jobs based on a triggered webhook event coming in. Job status information (pending, error, success) can be sent back only for pull request events. If you determine you do not want automation controller to post job statuses back to the webhook service, skip steps 1-2, and go directly to *step 3*.

1. Optionally generate a personal access token (PAT) for use with automation controller.

    a. In the profile settings of your GitHub account, click **Settings**.

    b. At the very bottom of the settings, click **<> Developer Settings**.

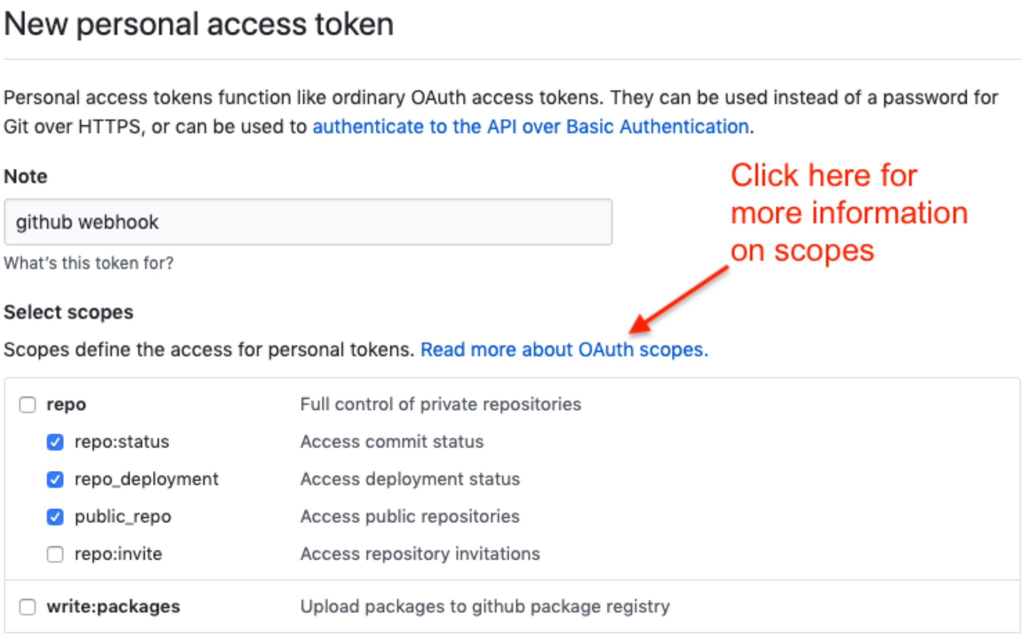
    c. In the Developer settings, click **Personal access tokens**.

    d. From the Personal access tokens screen, click **Generate new token**.

    e. When prompted, enter your GitHub account password to continue.

    f. In the **Note** field, enter a brief description about what this PAT will be used for.

    g. In the Scope fields, the automation webhook only needs repo scope access, with the exception of invites. For information about other scopes, click the link right above the table to access the docs.



New personal access token

Personal access tokens function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to authenticate to the API over Basic Authentication.

Note

github webhook

What's this token for?

Select scopes

Scopes define the access for personal tokens. Read more about OAuth scopes.

Click here for more information on scopes

☐ repo	Full control of private repositories
☑ repo:status	Access commit status
☑ repo_deployment	Access deployment status
☑ public_repo	Access public repositories
☐ repo:invite	Access repository invitations
☐ write:packages	Upload packages to github package registry

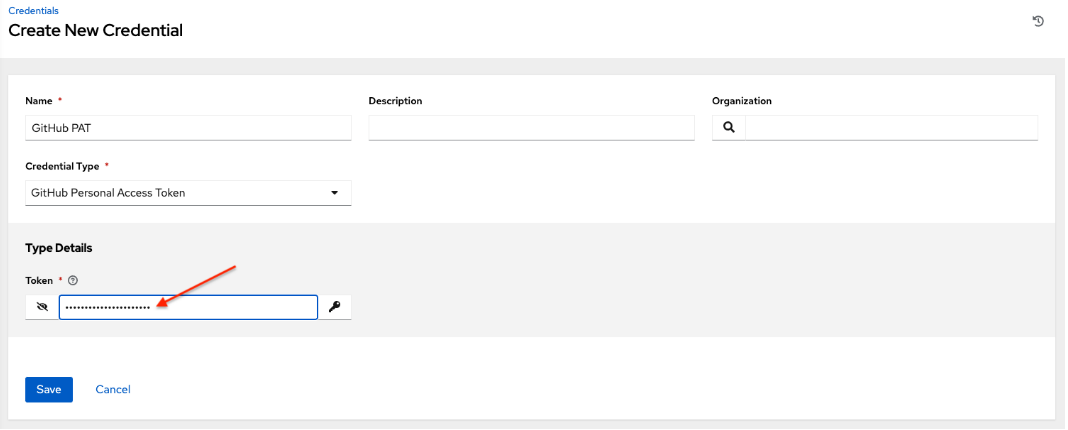
h. Click the **Generate Token** button.

i. Once the token is generated, make sure you copy the PAT, as it will be used in a later step. You will not be able to access this token again in GitHub.

2. Use the PAT to optionally create a GitHub credential:

   a. Go to your instance, and *create a new credential for the GitHub PAT* using the above generated token.

   b. Make note of the name of this credential, as it will be used in the job template that posts back to GitHub.



Credentials

Create New Credential

Name * GitHub PAT

Description

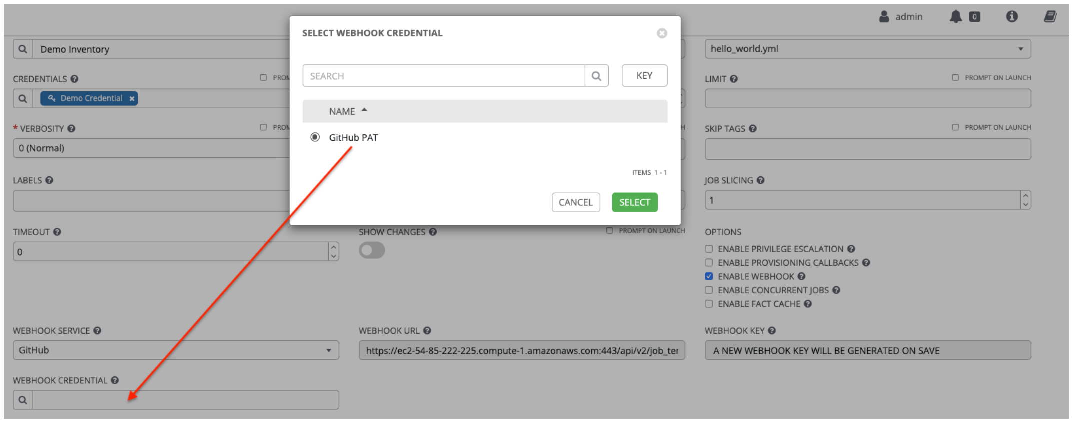
Organization

Credential Type * GitHub Personal Access Token

Type Details

Token * ························

   c. Go to the job template with which you want to enable webhooks, and select the webhook service and credential you created in the previous step.



SELECT WEBHOOK CREDENTIAL

SEARCH | KEY

NAME
● GitHub PAT

ITEMS 1 - 1

CANCEL | SELECT

Demo Inventory

CREDENTIALS: Demo Credential

* VERBOSITY: 0 (Normal)

LABELS

TIMEOUT: 0

hello_world.yml

LIMIT □ PROMPT ON LAUNCH

SKIP TAGS □ PROMPT ON LAUNCH

JOB SLICING: 1

SHOW CHANGES □ PROMPT ON LAUNCH

OPTIONS

- □ ENABLE PRIVILEGE ESCALATION
- □ ENABLE PROVISIONING CALLBACKS
- ☑ ENABLE WEBHOOK
- □ ENABLE CONCURRENT JOBS
- □ ENABLE FACT CACHE

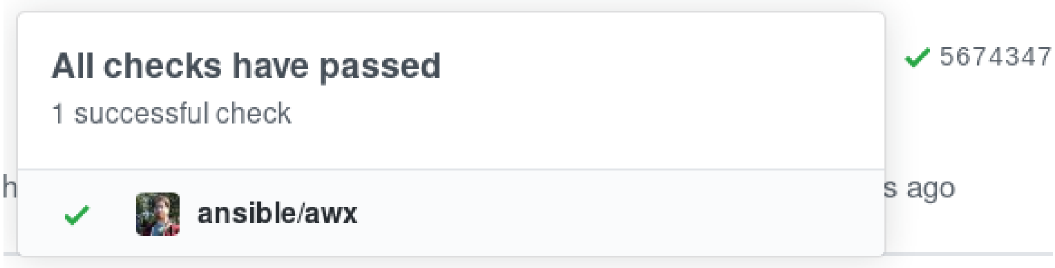
WEBHOOK SERVICE: GitHub

WEBHOOK URL: https://ec2-54-85-222-225.compute-1.amazonaws.com:443/api/v2/job_ter

WEBHOOK KEY: A NEW WEBHOOK KEY WILL BE GENERATED ON SAVE

WEBHOOK CREDENTIAL

d. Click **Save**. Now your job template is set up to be able to post back to GitHub. An example of one may look like this:



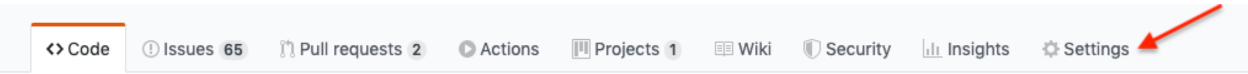
All checks have passed

1 successful check

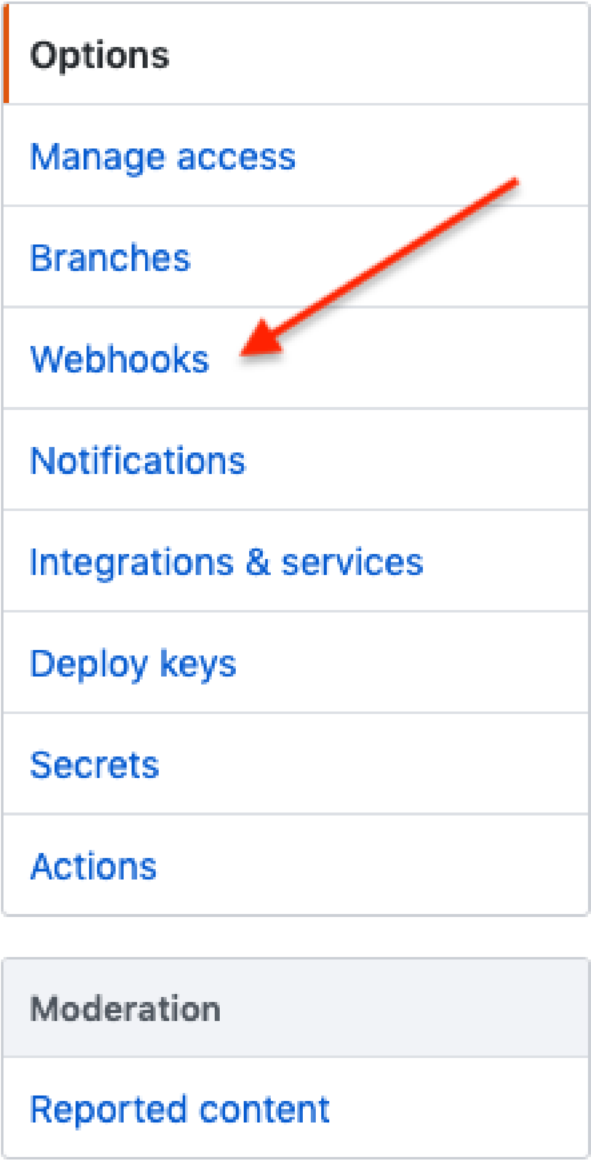
✓ ansible/awx

✓ 5674347

3. Go to a specific GitHub repo you want to configure webhooks and click **Settings**.



4. Under Options, click **Webhooks**.



5. On the Webhooks page, click **Add webhook**.

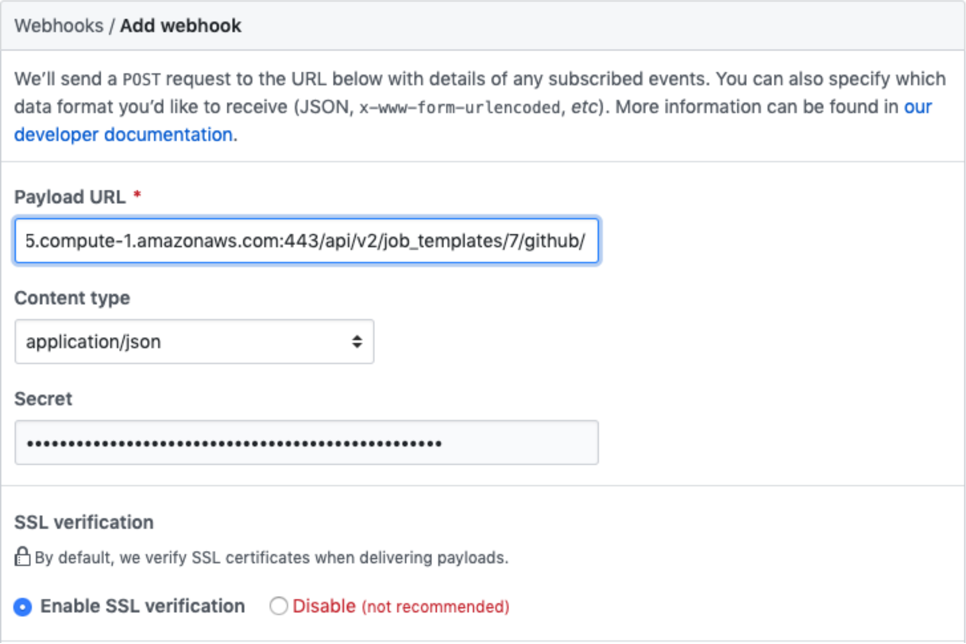
6. To complete the Add Webhook page, you need to *enable webhooks in a job template* (or in a *workflow job template*), which will provide you with the following information:

    a. Copy the contents of the **Webhook URL** from the job template, and paste it in the **Payload URL** field. GitHub uses this address to send results to.

    b. Set the **Content type** to **application/json**.

    c. Copy the contents of the **Webhook Key** from the job template above and paste it in the **Secret** field.

    d. Leave **Enable SSL verification** selected.



Webhooks / **Add webhook**

We'll send a `POST` request to the URL below with details of any subscribed events. You can also specify which data format you'd like to receive (JSON, `x-www-form-urlencoded`, *etc*). More information can be found in our developer documentation.

Payload URL *

5.compute-1.amazonaws.com:443/api/v2/job_templates/7/github/

Content type

application/json

Secret

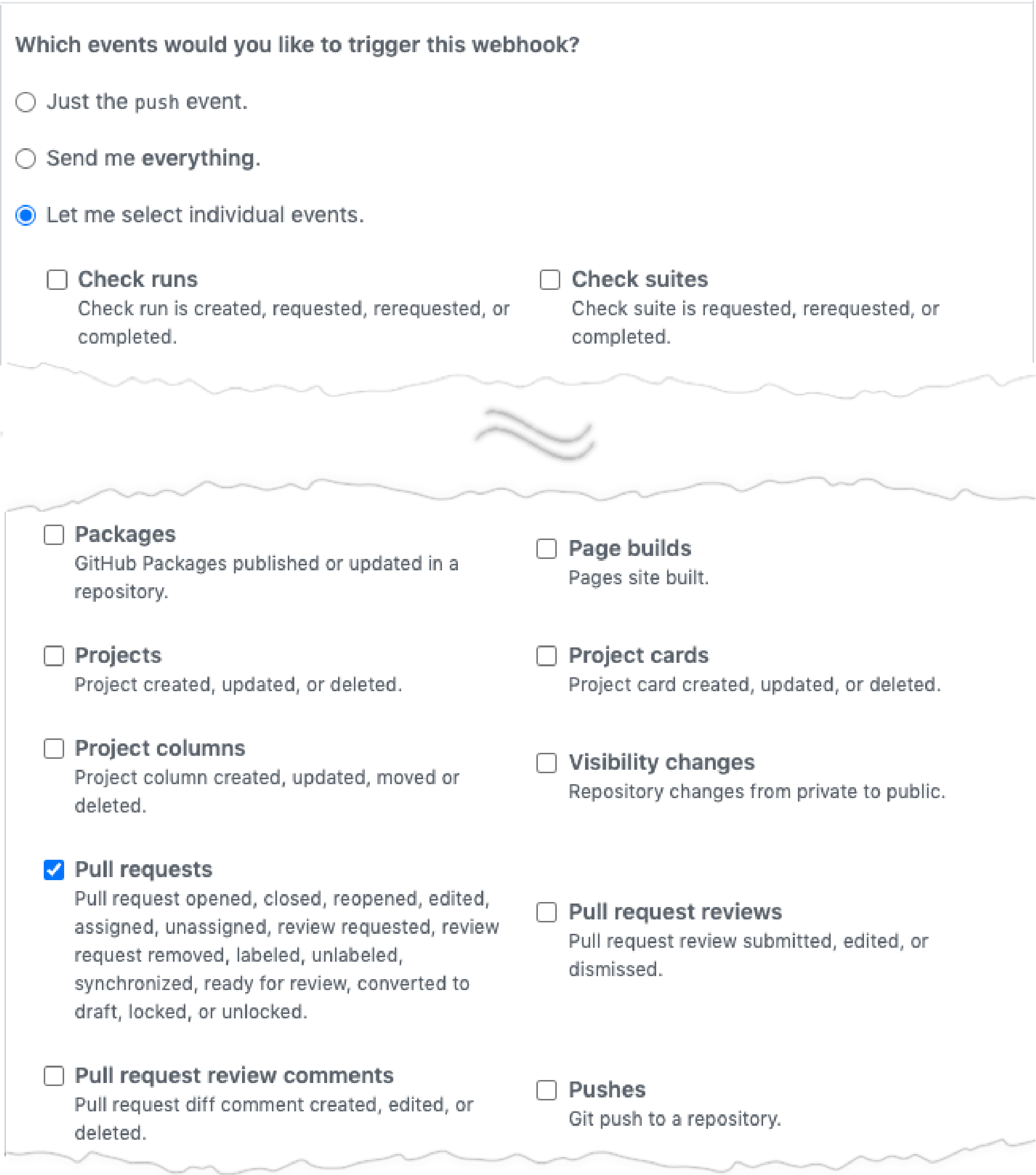
••

SSL verification

By default, we verify SSL certificates when delivering payloads.

● Enable SSL verification ○ Disable (not recommended)

e. Next, you must select the types of events you want to trigger a webhook. Any such event will trigger the Job or Workflow. In order to have job status (pending, error, success) sent back to GitHub, you must select **Pull requests** in the individual events section.



Which events would you like to trigger this webhook?

○ Just the push event.

○ Send me **everything**.

● Let me select individual events.

☐ **Check runs**
Check run is created, requested, rerequested, or completed.

☐ **Check suites**
Check suite is requested, rerequested, or completed.

☐ **Packages**
GitHub Packages published or updated in a repository.

☐ **Page builds**
Pages site built.

☐ **Projects**
Project created, updated, or deleted.

☐ **Project cards**
Project card created, updated, or deleted.

☐ **Project columns**
Project column created, updated, moved or deleted.

☐ **Visibility changes**
Repository changes from private to public.

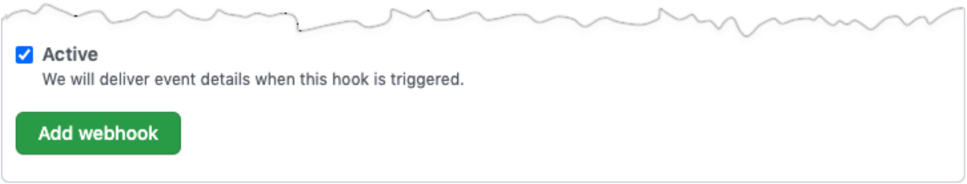
☑ **Pull requests**
Pull request opened, closed, reopened, edited, assigned, unassigned, review requested, review request removed, labeled, unlabeled, synchronized, ready for review, converted to draft, locked, or unlocked.

☐ **Pull request reviews**
Pull request review submitted, edited, or dismissed.

☐ **Pull request review comments**
Pull request diff comment created, edited, or deleted.

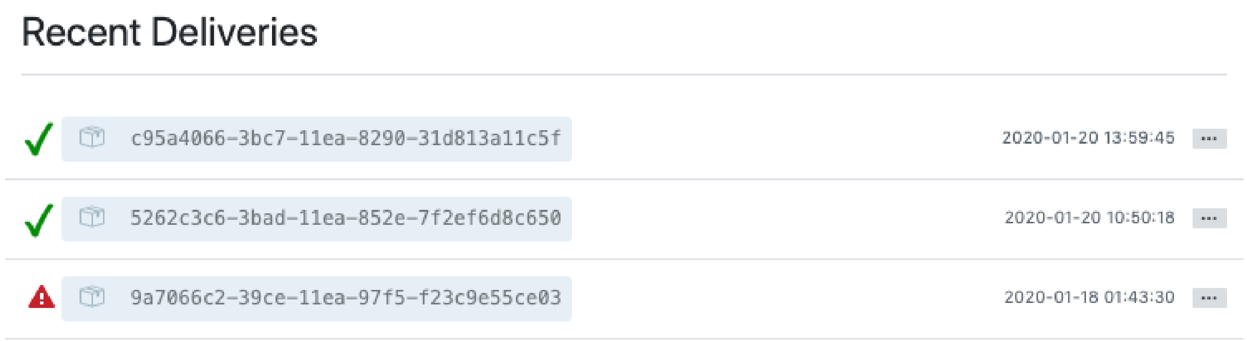
☐ **Pushes**
Git push to a repository.

f. Leave **Active** checked and click **Add Webhook**.



☑ **Active**
We will deliver event details when this hook is triggered.

7. After your webhook is configured, it displays in the list of webhooks active for your repo, along with the ability to edit or delete it. Click on a webhook, and it brings you to the Manage webhook screen. Scroll to the very bottom of the screen to view all the delivery attempts made to your webhook and whether they succeeded or failed.



Recent Deliveries

✓	c95a4066-3bc7-11ea-8290-31d813a11c5f	2020-01-20 13:59:45	...
✓	5262c3c6-3bad-11ea-852e-7f2ef6d8c650	2020-01-20 10:50:18	...
⚠	9a7066c2-39ce-11ea-97f5-f23c9e55ce03	2020-01-18 01:43:30	...

For more information, refer to the GitHub Webhooks developer documentation.

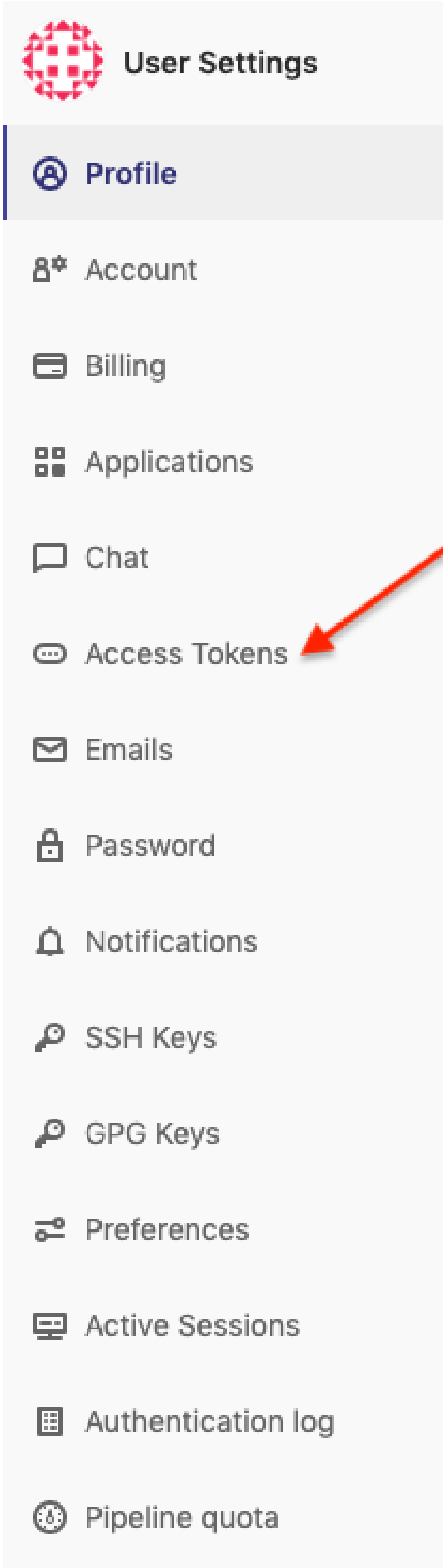
26.2 GitLab webhook setup

Automation controller has the ability to run jobs based on a triggered webhook event coming in. Job status information (pending, error, success) can be sent back only for merge request events. If you determine you do not want automation controller to post job statuses back to the webhook service, skip steps 1-2, and go directly to *step 3*.

1. Optionally, generate a personal access token (PAT). This token gives automation controller the ability to post statuses back when we run jobs based on a webhook coming in.

    a. In the profile settings of your GitLab account, click **Settings**.

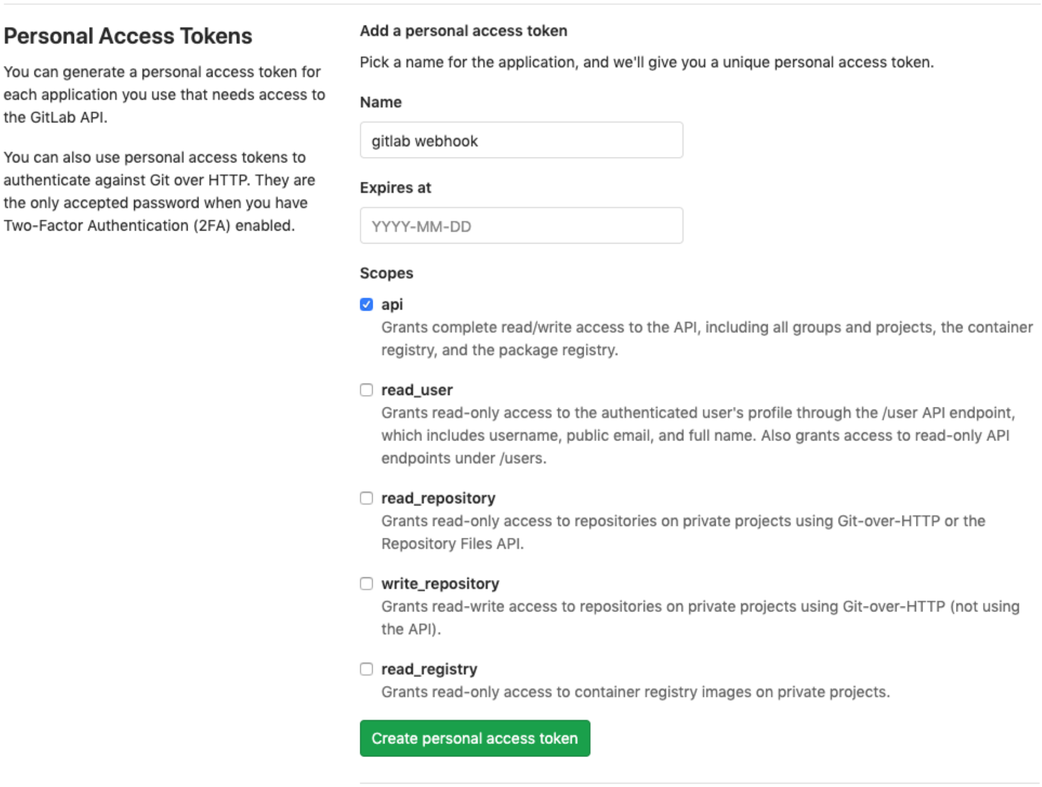
    b. On the sidebar, under User Settings, click **Access Tokens**.


User Settings
Profile
Account
Billing
Applications
Chat
Access Tokens
Emails
Password
Notifications
SSH Keys
GPG Keys
Preferences
Active Sessions
Authentication log
Pipeline quota

c. In the **Name** field, enter a brief description about what this PAT will be used for.

d. Skip the **Expires at** field unless you want to set an expiration date for your webhook.

e. In the Scopes fields, select the ones applicable to your integration. For automation controller, API is the only selection necessary.



Personal Access Tokens

You can generate a personal access token for each application you use that needs access to the GitLab API.

You can also use personal access tokens to authenticate against Git over HTTP. They are the only accepted password when you have Two-Factor Authentication (2FA) enabled.

Add a personal access token

Pick a name for the application, and we'll give you a unique personal access token.

Name

gitlab webhook

Expires at

YYYY-MM-DD

Scopes

☑ **api**
Grants complete read/write access to the API, including all groups and projects, the container registry, and the package registry.

□ **read_user**
Grants read-only access to the authenticated user's profile through the /user API endpoint, which includes username, public email, and full name. Also grants access to read-only API endpoints under /users.

□ **read_repository**
Grants read-only access to repositories on private projects using Git-over-HTTP or the Repository Files API.

□ **write_repository**
Grants read-write access to repositories on private projects using Git-over-HTTP (not using the API).

□ **read_registry**
Grants read-only access to container registry images on private projects.

Create personal access token

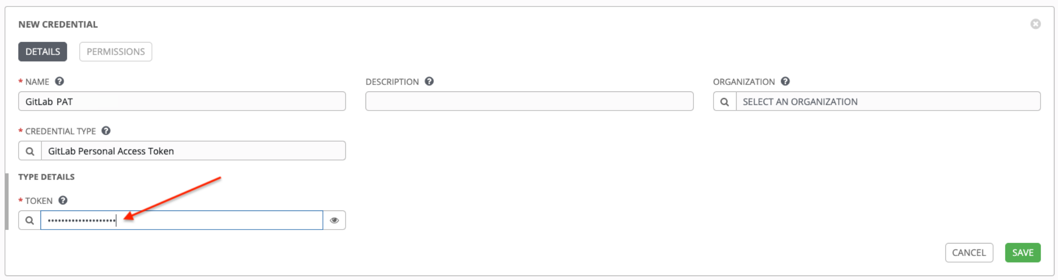
f. Click the **Create personal access token** button.

g. Once the token is generated, make sure you copy the PAT, as it will be used in a later step. You will not be able to access this token again in GitLab.

2. Use the PAT to optionally create a GitLab credential:

a. Go to your instance, and *create a new credential for the GitLab PAT* using the above generated token.

b. Make note of the name of this credential, as it will be used in the job template that posts back to GitHub.



NEW CREDENTIAL

DETAILS | PERMISSIONS

* NAME: GitLab PAT

DESCRIPTION:

ORGANIZATION: SELECT AN ORGANIZATION

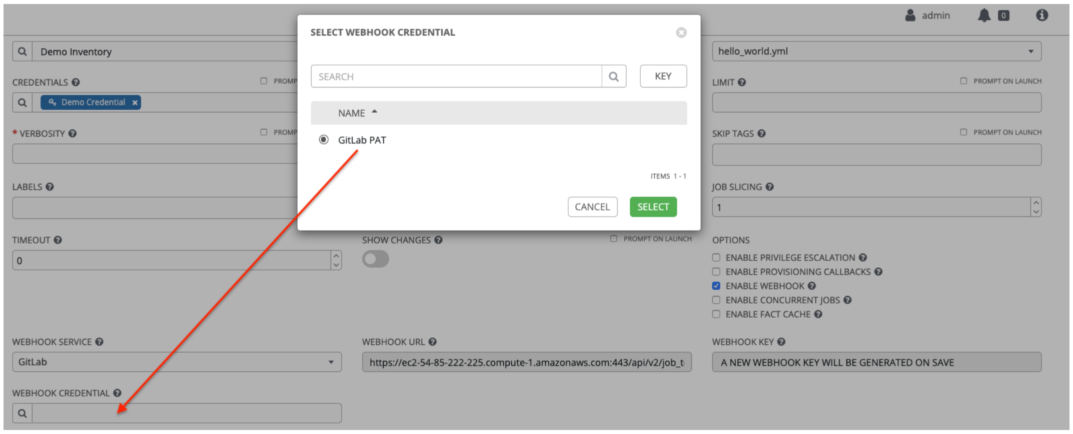
* CREDENTIAL TYPE: GitLab Personal Access Token

TYPE DETAILS

* TOKEN: ••••••••••••••••••••

CANCEL | SAVE

c. Go to the job template with which you want to enable webhooks, and select the webhook service and credential you created in the previous step.



SELECT WEBHOOK CREDENTIAL

NAME
● GitLab PAT

Demo Inventory

CREDENTIALS: Demo Credential

* VERBOSITY

LABELS

TIMEOUT: 0

SHOW CHANGES □ PROMPT ON LAUNCH

WEBHOOK SERVICE: GitLab

WEBHOOK URL: https://ec2-54-85-222-225.compute-1.amazonaws.com:443/api/v2/job_t

WEBHOOK CREDENTIAL

hello_world.yml

LIMIT □ PROMPT ON LAUNCH

SKIP TAGS □ PROMPT ON LAUNCH

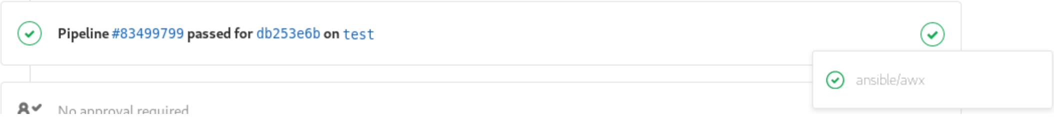
JOB SLICING: 1

OPTIONS

- □ ENABLE PRIVILEGE ESCALATION
- □ ENABLE PROVISIONING CALLBACKS
- ☑ ENABLE WEBHOOK
- □ ENABLE CONCURRENT JOBS
- □ ENABLE FACT CACHE

WEBHOOK KEY: A NEW WEBHOOK KEY WILL BE GENERATED ON SAVE

d. Click **Save**. Now your job template is set up to be able to post back to GitLab. An example of one may look like this:

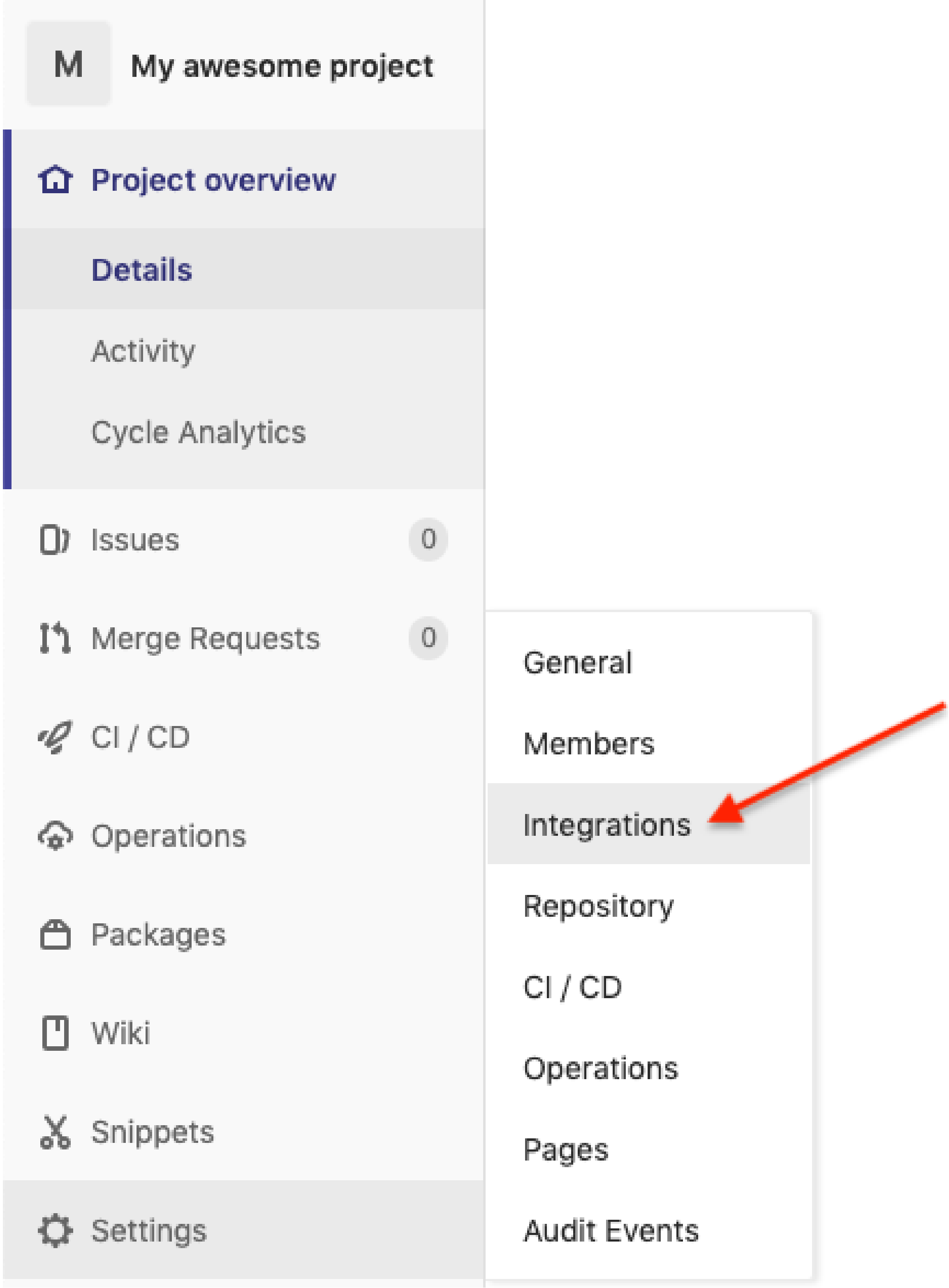


Pipeline #83499799 passed for db253e6b on test

ansible/awx

No approval required

3. Go to a specific GitLab repo you want to configure webhooks and click **Settings > Integrations**.



4. To complete the Integrations page, you need to *enable webhooks in a job template* (or in a *workflow job template*), which will provide you with the following information:

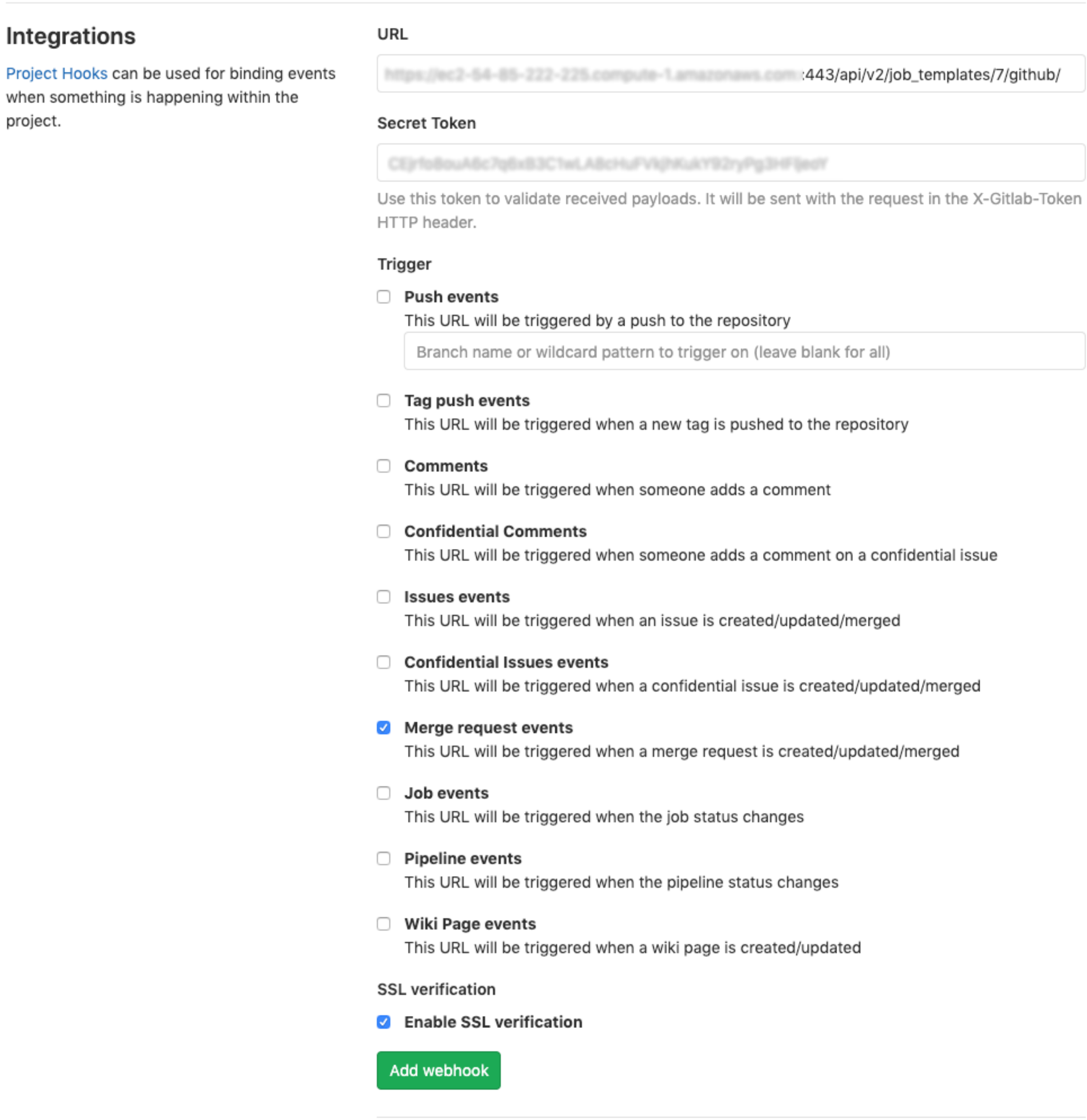
   a. Copy the contents of the **Webhook URL** from the job template above, and paste it in the **URL** field. GitLab uses this address to send results to.

b. Copy the contents of the **Webhook Key** from the job template above and paste it in the **Secret Token** field.

c. Next, you must select the types of events you want to trigger a webhook. Any such event will trigger the Job or Workflow. In order to have job status (pending, error, success) sent back to GitLab, you must select **Merge request events** in the Trigger section.

d. Leave **Enable SSL verification** selected.

e. Click **Add webhook**.



Integrations

Project Hooks can be used for binding events when something is happening within the project.

URL

[redacted]:443/api/v2/job_templates/7/github/

Secret Token

[redacted]

Use this token to validate received payloads. It will be sent with the request in the X-Gitlab-Token HTTP header.

Trigger

☐ **Push events**
This URL will be triggered by a push to the repository
Branch name or wildcard pattern to trigger on (leave blank for all)

☐ **Tag push events**
This URL will be triggered when a new tag is pushed to the repository

☐ **Comments**
This URL will be triggered when someone adds a comment

☐ **Confidential Comments**
This URL will be triggered when someone adds a comment on a confidential issue

☐ **Issues events**
This URL will be triggered when an issue is created/updated/merged

☐ **Confidential Issues events**
This URL will be triggered when a confidential issue is created/updated/merged

☑ **Merge request events**
This URL will be triggered when a merge request is created/updated/merged

☐ **Job events**
This URL will be triggered when the job status changes

☐ **Pipeline events**
This URL will be triggered when the pipeline status changes

☐ **Wiki Page events**
This URL will be triggered when a wiki page is created/updated

SSL verification

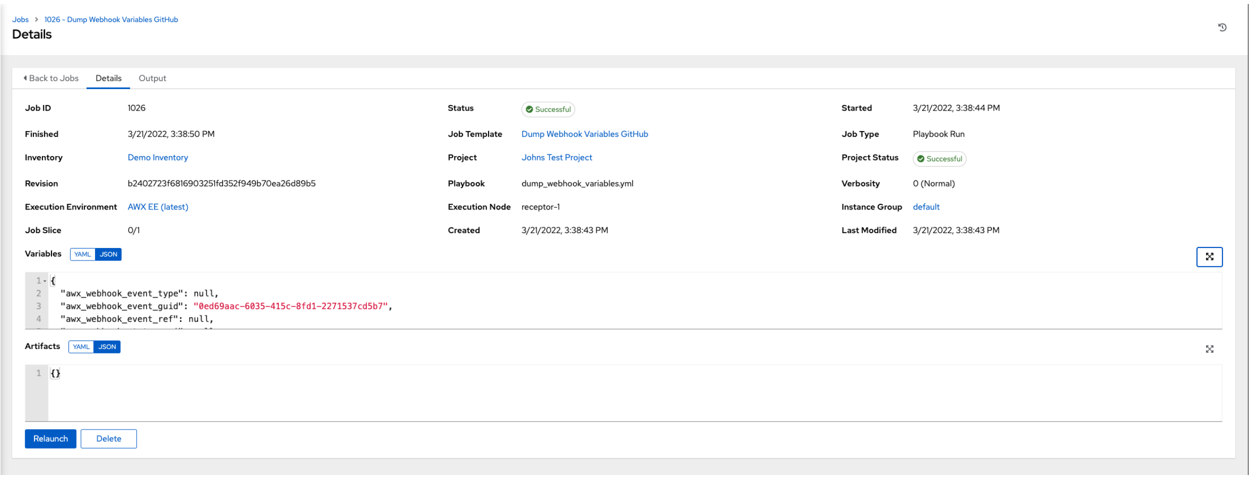
☑ **Enable SSL verification**

5. After your webhook is configured, it displays in the list of Project Webhooks for your repo, along with the ability to test events, edit or delete the webhook. Testing a webhook event displays the results at the top of the page whether it succeeded or failed.

For more information, refer to the GitLab webhooks integrations documentation.

26.3 Payload output

The entire payload is exposed as an extra variable. To view the payload information, go to the Jobs Detail view of the job template that ran with the webhook enabled. In the **Extra Variables** field of the Details pane, view the payload output from the `awx_webhook_payload` variable, as shown in the example below.



Jobs › 1026 - Dump Webhook Variables GitHub

Details

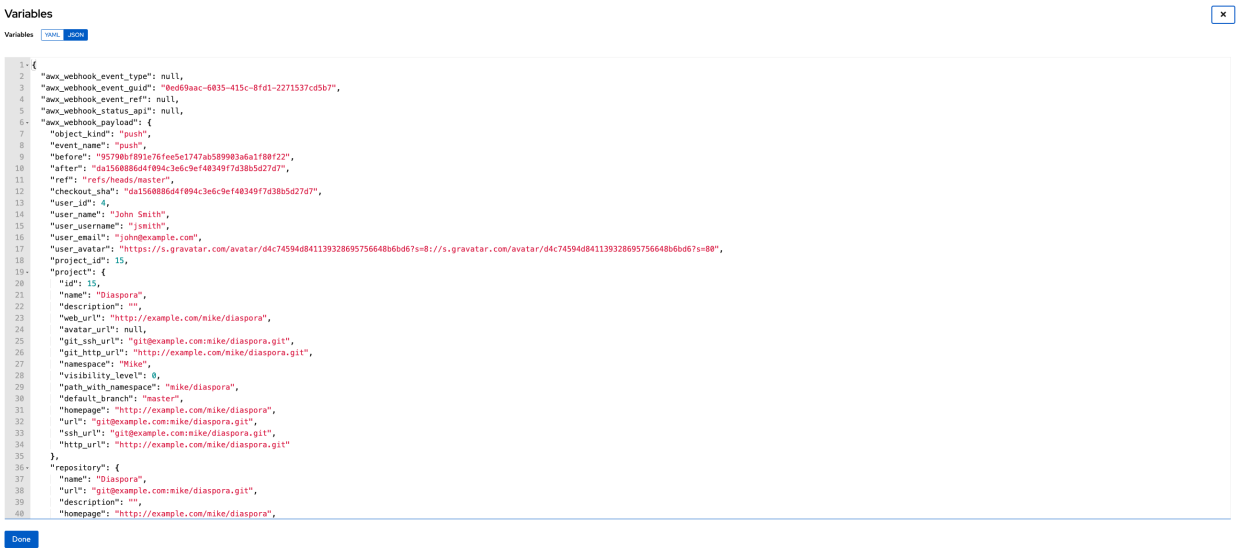
Job ID	1026	Status	Successful	Started	3/21/2022, 3:38:44 PM
Finished	3/21/2022, 3:38:50 PM	Job Template	Dump Webhook Variables GitHub	Job Type	Playbook Run
Inventory	Demo Inventory	Project	Johns Test Project	Project Status	Successful
Revision	b2402723f6816903251fd352f949b70ea26d89b5	Playbook	dump_webhook_variables.yml	Verbosity	0 (Normal)
Execution Environment	AWX EE (latest)	Execution Node	receptor-1	Instance Group	default
Job Slice	0/1	Created	3/21/2022, 3:38:43 PM	Last Modified	3/21/2022, 3:38:43 PM

Variables

```
{
  "awx_webhook_event_type": null,
  "awx_webhook_event_guid": "0ed69aac-6035-415c-8fd1-2271537cd5b7",
  "awx_webhook_event_ref": null,
```

Artifacts

```
{}
```



Variables

Variables

```
{
  "awx_webhook_event_type": null,
  "awx_webhook_event_guid": "0ed69aac-6035-415c-8fd1-2271537cd5b7",
  "awx_webhook_event_ref": null,
  "awx_webhook_status_api": null,
  "awx_webhook_payload": {
    "object_kind": "push",
    "event_name": "push",
    "before": "95790bf891e76fee5e1747ab589903a6a1f80f22",
    "after": "da1560886d4f094c3e6c9ef40349f7d38b5d27d7",
    "ref": "refs/heads/master",
    "checkout_sha": "da1560886d4f094c3e6c9ef40349f7d38b5d27d7",
    "user_id": 4,
    "user_name": "John Smith",
    "user_username": "jsmith",
    "user_email": "john@example.com",
    "user_avatar": "https://s.gravatar.com/avatar/d4c74594d841139328695756648b6bd6?s=8://s.gravatar.com/avatar/d4c74594d841139328695756648b6bd6?s=80",
    "project_id": 15,
    "project": {
      "id": 15,
      "name": "Diaspora",
      "description": "",
      "web_url": "http://example.com/mike/diaspora",
      "avatar_url": null,
      "git_ssh_url": "git@example.com:mike/diaspora.git",
      "git_http_url": "http://example.com/mike/diaspora.git",
      "namespace": "Mike",
      "visibility_level": 0,
      "path_with_namespace": "mike/diaspora",
      "default_branch": "master",
      "homepage": "http://example.com/mike/diaspora",
      "url": "git@example.com:mike/diaspora.git",
      "ssh_url": "git@example.com:mike/diaspora.git",
      "http_url": "http://example.com/mike/diaspora.git"
    },
    "repository": {
      "name": "Diaspora",
      "url": "git@example.com:mike/diaspora.git",
      "description": "",
      "homepage": "http://example.com/mike/diaspora",
```

CHAPTER

TWENTYSEVEN

NOTIFICATIONS

A *Notification Template* is an instance of a *Notification* type (Email, Slack, Webhook, etc.) with a name, description, and a defined configuration.

For example:

- A username, password, server, and recipients are needed for an Email notification template
- The token and a list of channels are needed for a Slack notification template
- The URL and Headers are needed for a Webhook notification template

A Notification is a manifestation of the notification template; for example, when a job fails, a notification is sent using the configuration defined by the notification template.

At a high level, the typical flow for the notification system works as follows:

- A user creates a notification template to the REST API at the `/api/v2/notification_templates` endpoint (either through the API or through the UI).
- A user assigns the notification template to any of the various objects that support it (all variants of job templates as well as organizations and projects) and at the appropriate trigger level for which they want the notification (started, success, or error). For example a user may wish to assign a particular notification template to trigger when Job Template 1 fails. In which case, they will associate the notification template with the job template at `/api/v2/job_templates/n/notification_templates_error` API endpoint.
- You can set notifications on job start, not just job end. Users and teams are also able to define their own notifications that can be attached to arbitrary jobs.

27.1 Notification Hierarchy

Notification templates assigned at certain levels will inherit templates defined on parent objects as such:

- Job Templates will use notification templates defined on it as well as inheriting notification templates from the Project used by the Job Template and from the Organization that it is listed under (via the Project).
- Project Updates will use notification templates defined on the project and will inherit notification templates from the Organization associated with it
- Inventory Updates will use notification templates defined on the Organization that it is listed under
- Ad-hoc commands will use notification templates defined on the Organization that the inventory is associated with

27.2 Workflow

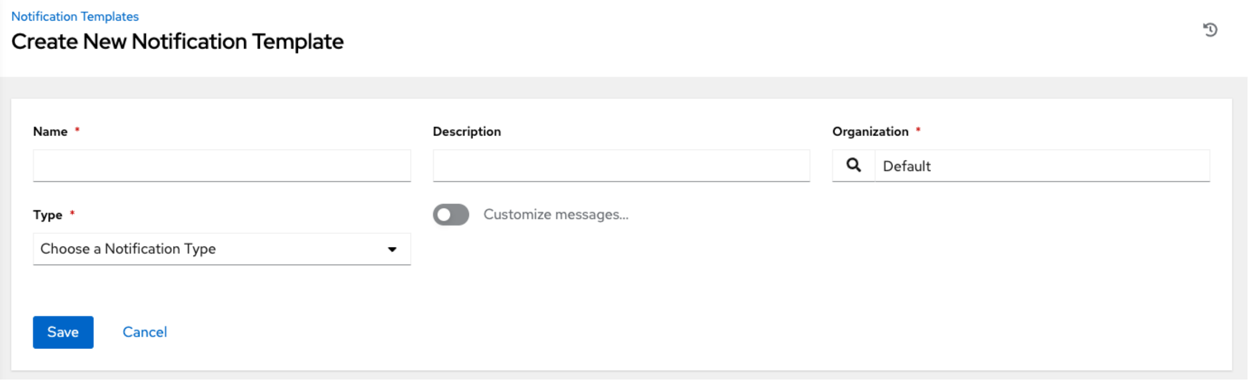
When a job succeeds or fails, the error or success handler will pull a list of relevant notification templates using the procedure defined above. It will then create a Notification object for each one containing relevant details about the job and then sends it to the destination (email addresses, slack channel(s), sms numbers, etc). These Notification objects are available as related resources on job types (jobs, inventory updates, project updates), and also at `/api/v2/notifications`. You may also see what notifications have been sent from a notification templates by examining its related resources.

If a notification fails, it will not impact the job associated to it or cause it to fail. The status of the notification can be viewed at its detail endpoint (`/api/v2/notifications/<n>`).

27.3 Create a Notification Template

To create a Notification Template:

1. Click **Notifications** from the left navigation bar.
2. Click the **Add** button.



3. Enter the name of the notification and a description in their respective fields, and specify the organization (required) it belongs to.
4. Choose a type of notification from the **Type** drop-down menu. Refer to the subsequent sections for additional information.
5. Once all required information is complete, click **Save** to add the notification.

27.4 Notification Types

Notification types supported with automation controller:

- *Email*
- *Grafana*
- *IRC*
- *Mattermost*
- *PagerDuty*

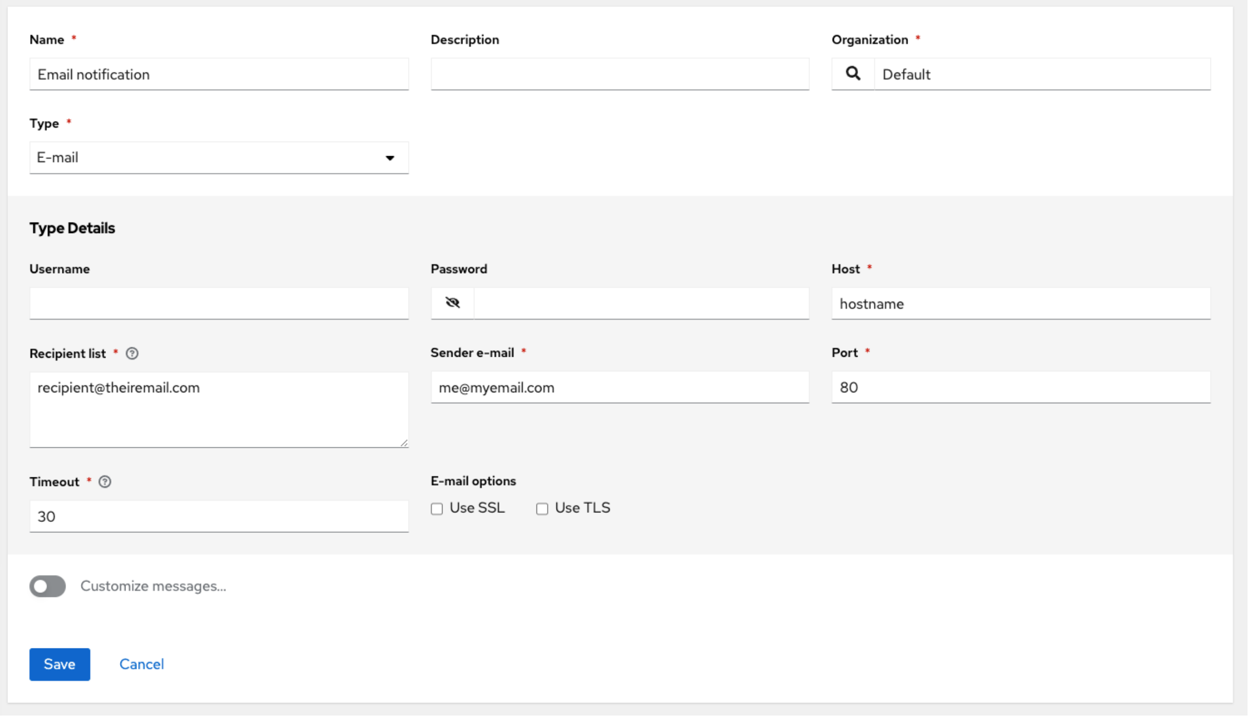
Each of these have their own configuration and behavioral semantics and testing them may need to be approached in different ways. Additionally, you can customize each type of notification down to a specific detail, or a set of criteria to trigger a notification. See *Create custom notifications* for more detail on configuring custom notifications. The following sections will give as much detail as possible on each type of notification.

27.4.1 Email

The email notification type supports a wide variety of SMTP servers and has support for TLS/SSL connections.

You must provide the following details to setup an email notification:

- Host
- Recipient list
- Sender email
- Port
- Timeout (in seconds): allows you to specify up to 120 seconds, the length of time automation controller may attempt connecting to the email server before giving up.



Name * Email notification

Description

Organization * Default

Type * E-mail

Type Details

Username

Password

Host * hostname

Recipient list * recipient@theiremail.com

Sender e-mail * me@myemail.com

Port * 80

Timeout * 30

E-mail options □ Use SSL □ Use TLS

27.4.2 Grafana

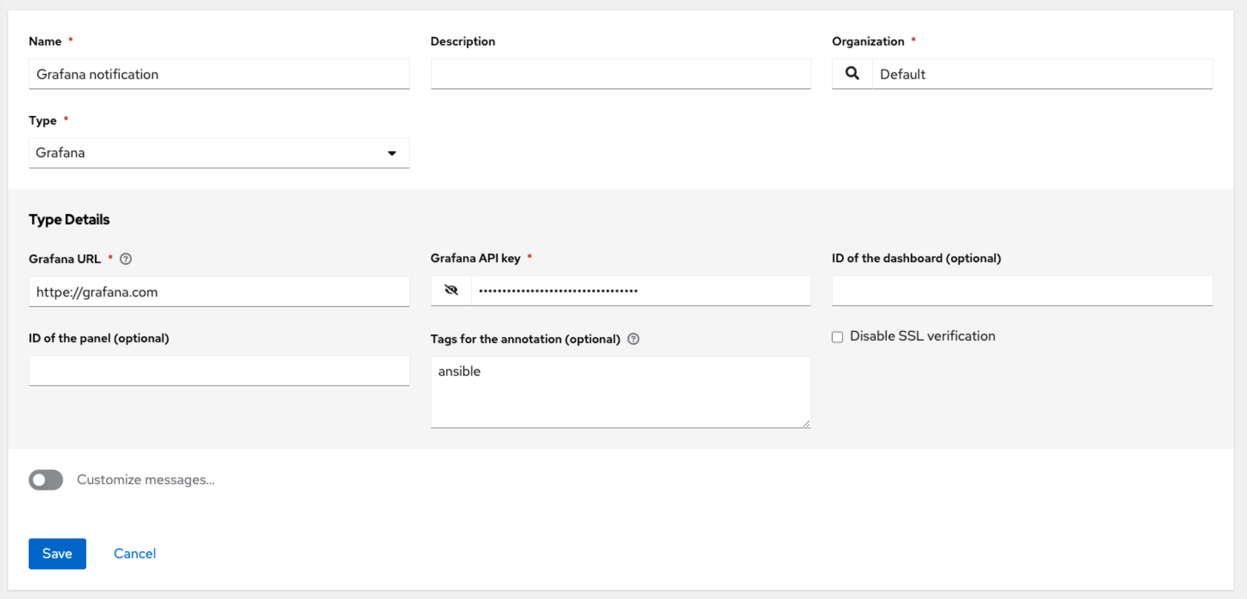
Grafana is a fairly straightforward integration. First, create an API Key in the Grafana system (this is the token that is given to automation controller).

You must provide the following details to setup a Grafana notification:

- Grafana URL: The URL of the Grafana API service, generally `http://yourcompany.grafana.com`.
- Grafana API Key: The user must first create an API Key in the Grafana system (this is the token that is given to automation controller).

The other options of note are:

- ID of the Dashboard: When you created an API Key for the Grafana account, you can set up a dashboard with its own unique ID.
- ID of the Panel: If you added panels and graphs to your Grafana interface, you can specify its ID here.
- Tags for the Annotation: Enter keywords that help identify the type(s) of events(s) of the notification you are configuring.
- Disable SSL Verification: SSL verification is on by default, but you can choose to turn off verification the authenticity of the target's certificate. Environments that use internal or private CA's should select this option to disable verification.



Name *: Grafana notification

Description:

Organization *: Default

Type *: Grafana

Type Details

Grafana URL *: httpe://grafana.com

Grafana API key *: ••••••••••••••••••••••••••••••••

ID of the dashboard (optional):

ID of the panel (optional):

Tags for the annotation (optional): ansible

□ Disable SSL verification

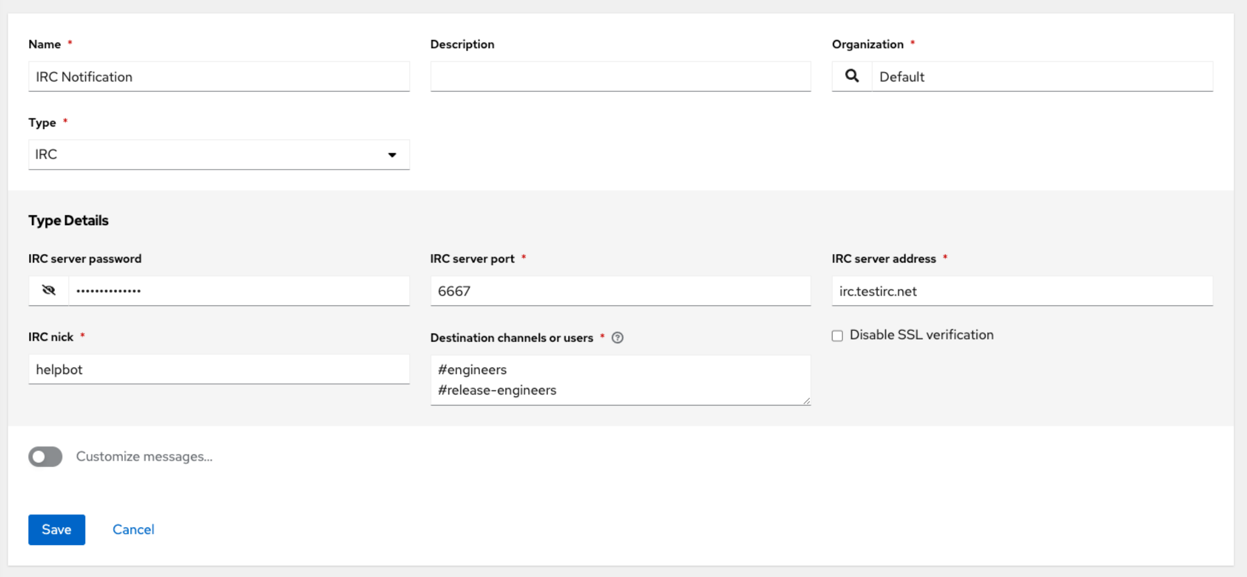
27.4.3 IRC

The IRC notification takes the form of an IRC bot that will connect, deliver its messages to channel(s) or individual user(s), and then disconnect. The notification bot also supports SSL authentication. The bot does not currently support Nickserv identification. If a channel or user does not exist or is not on-line then the Notification will not fail; the failure scenario is reserved specifically for connectivity.

Connectivity information is straightforward:

- IRC Server Password (optional): IRC servers can require a password to connect. If the server does not require one, leave blank
- IRC Server Port: The IRC server Port

- IRC Server Address: The host name or address of the IRC server
- IRC Nick: The bot's nickname once it connects to the server
- Destination Channels or Users: A list of users and/or channels to which to send the notification.
- SSL Connection (optional): Should the bot use SSL when connecting



Name *	Description	Organization *
IRC Notification		Default
Type *		
IRC		

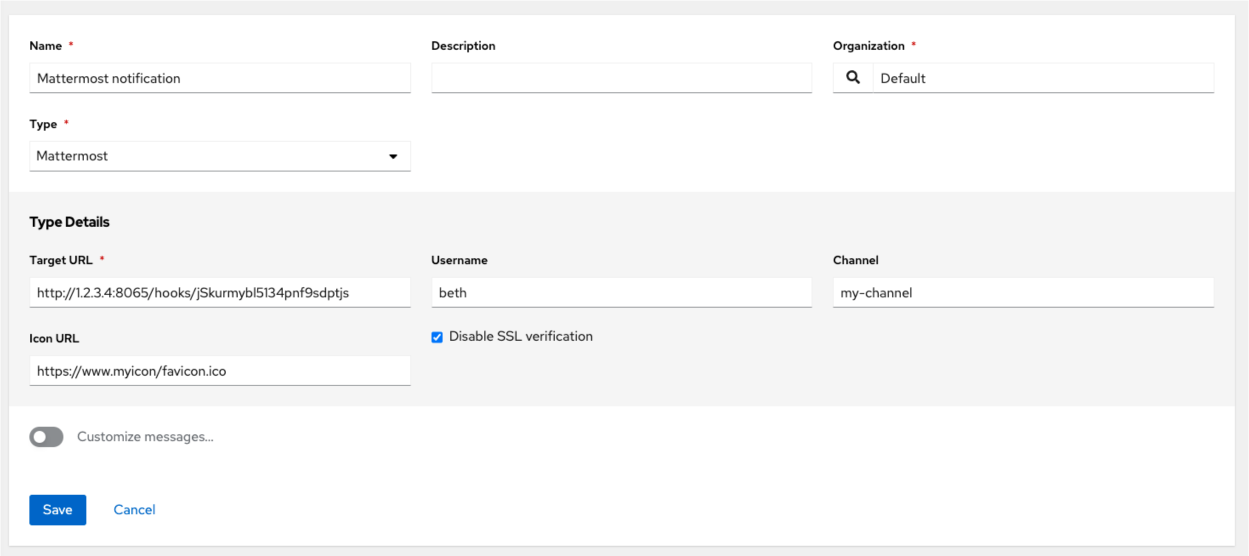
Type Details

IRC server password	IRC server port *	IRC server address *
••••••••••••••	6667	irc.testirc.net
IRC nick *	Destination channels or users *	□ Disable SSL verification
helpbot	#engineers #release-engineers	

27.4.4 Mattermost

The Mattermost notification type provides a simple interface to Mattermost's messaging and collaboration workspace. The parameters that can be specified are:

- Target URL (required): The full URL that will be POSTed to
- Username
- Channel
- Icon URL: specifies the icon to display for this notification
- Disable SSL Verification: Turns off verification of the authenticity of the target's certificate. Environments that use internal or private CA's should select this option to disable verification.



Name * Mattermost notification

Description

Organization * Default

Type * Mattermost

Type Details

Target URL * http://1.2.3.4:8065/hooks/jSkurmybl5134pnf9sdptjs

Username beth

Channel my-channel

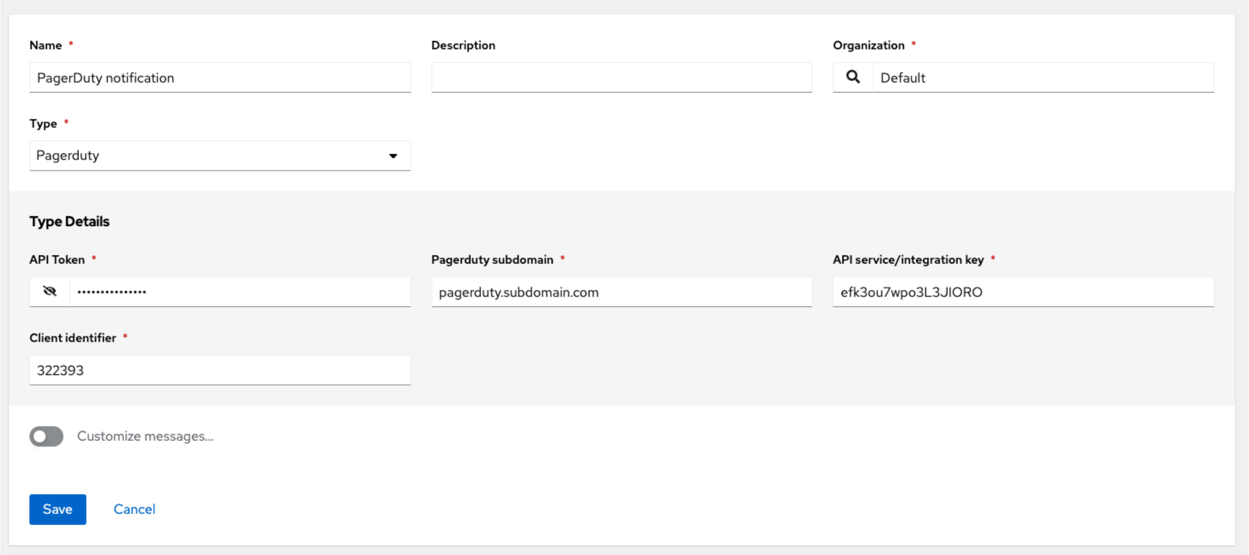
Icon URL https://www.myicon/favicon.ico

☑ Disable SSL verification

27.4.5 PagerDuty

PagerDuty is a fairly straightforward integration. First, create an API Key in the PagerDuty system (this is the token that is given to automation controller) and then create a "Service" which provides an "Integration Key" that will also be given to automation controller. The other required options are:

- API Token: The user must first create an API Key in the PagerDuty system (this is the token that is given to automation controller).
- PagerDuty Subdomain: When you sign up for the PagerDuty account, you receive a unique subdomain to communicate with. For instance, if you signed up as "testuser", the web dashboard will be at `testuser.pagerduty.com` and you will give the API `testuser` as the subdomain (not the full domain).
- API Service/Integration Key
- Client Identifier: This will be sent along with the alert content to the pagerduty service to help identify the service that is using the api key/service. This is helpful if multiple integrations are using the same API key and service.



Name * PagerDuty notification

Description

Organization * Default

Type * Pagerduty

Type Details

API Token * ••••••••••••••

Pagerduty subdomain * pagerduty.subdomain.com

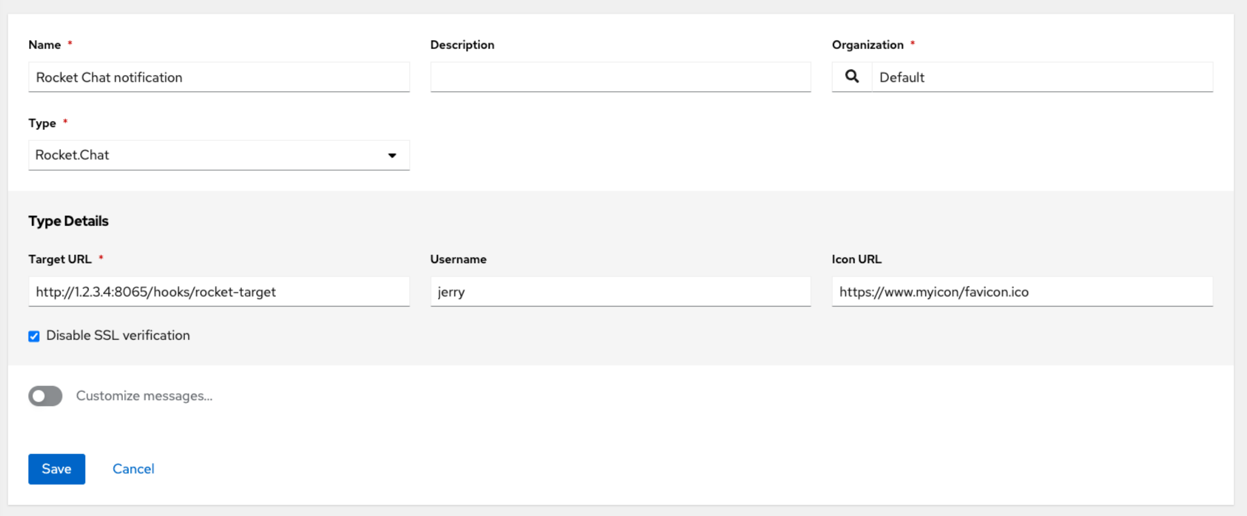
API service/integration key * efk3ou7wpo3L3JIORO

Client identifier * 322393

27.4.6 Rocket.Chat

The Rocket.Chat notification type provides an interface to Rocket.Chat's collaboration and communication platform. The parameters that can be specified are:

- Target URL (required): The full URL that will be POSTed to
- Username
- Icon URL: specifies the icon to display for this notification
- Disable SSL Verification: Turns off verification of the authenticity of the target's certificate. Environments that use internal or private CA's should select this option to disable verification.



Name *
Rocket Chat notification

Description

Organization *
Default

Type *
Rocket.Chat

Type Details

Target URL *
http://1.2.3.4:8065/hooks/rocket-target

Username
jerry

Icon URL
https://www.myicon/favicon.ico

☑ Disable SSL verification

Customize messages...

27.4.7 Slack

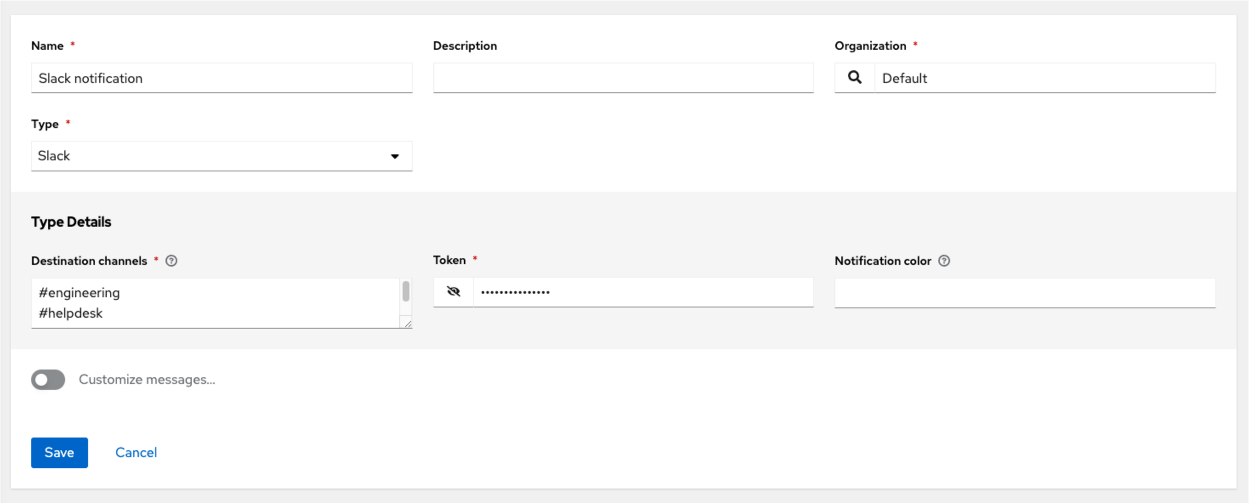
Slack, a collaborative team communication and messaging tool, is pretty easy to configure.

You must supply the following to setup Slack notifications:

- A Slack app (refer to the Basic App Setup page of the Slack documentation for information on how to create one)
- A token (refer to Enabling Interactions with Bots and specific details on bot tokens on the Token Types documentation page)

Once you have a bot/app set up, you must navigate to "Your Apps", click on the newly-created app and then go to **Add features and functionality**, which allows you to configure incoming webhooks, bots, and permissions; as well as **Install your app to your workspace**.

You must also invite the notification bot to join the channel(s) in question in Slack. Note that private messages are not supported.



Name * Slack notification

Description

Organization * Default

Type * Slack

Type Details

Destination channels *
#engineering
#helpdesk

Token * ••••••••••••••••

Notification color

Customize messages...

27.4.8 Twilio

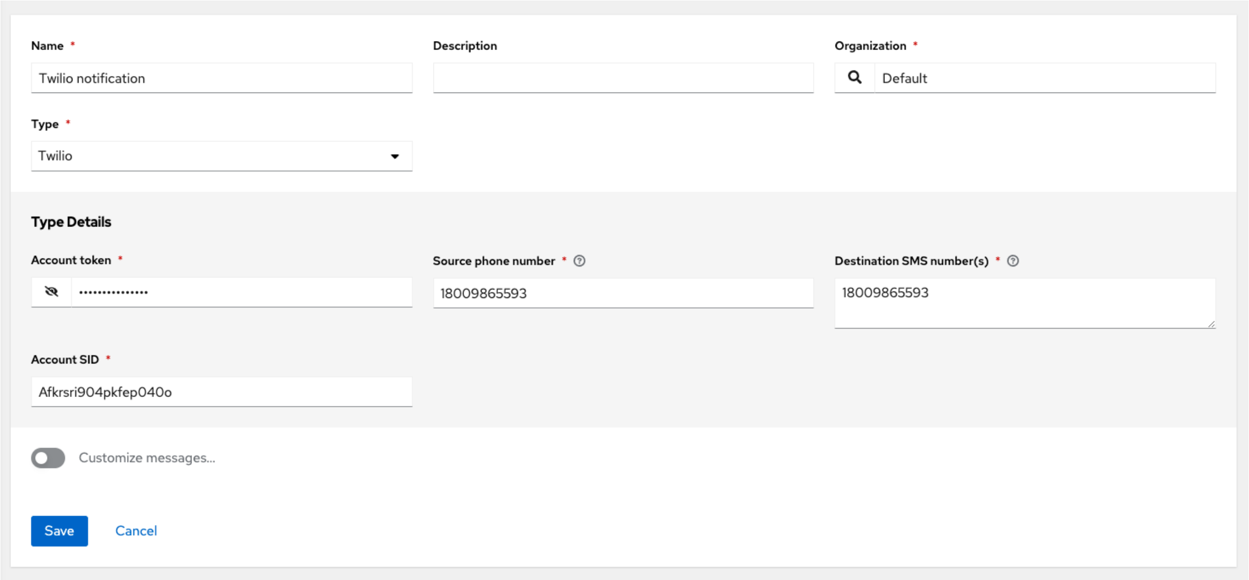
Twilio service is an Voice and SMS automation service. Once you are signed in, you must create a phone number from which the message will be sent. You can then define a "Messaging Service" under Programmable SMS and associate the number you created before with it.

Note that you may need to verify this number or some other information before you are allowed to use it to send to any numbers. The Messaging Service does not need a status callback URL nor does it need the ability to Process inbound messages.

Under your individual (or sub) account settings, you will have API credentials. Twilio uses two credentials to determine which account an API request is coming from. The "Account SID", which acts as a username, and the "Auth Token" which acts as a password.

To setup Twilio, provide the following details:

- Account Token
- Source Phone Number (this is the number associated with the messaging service above and must be given in the form of "+15556667777")
- Destination SMS number (this will be the list of numbers to receive the SMS and should be the 10-digit phone number)
- Account SID



Name * Twilio notification

Description

Organization * Default

Type * Twilio

Type Details

Account token * ···············

Source phone number * 18009865593

Destination SMS number(s) * 18009865593

Account SID * Afkrsri904pkfep040o

Customize messages...

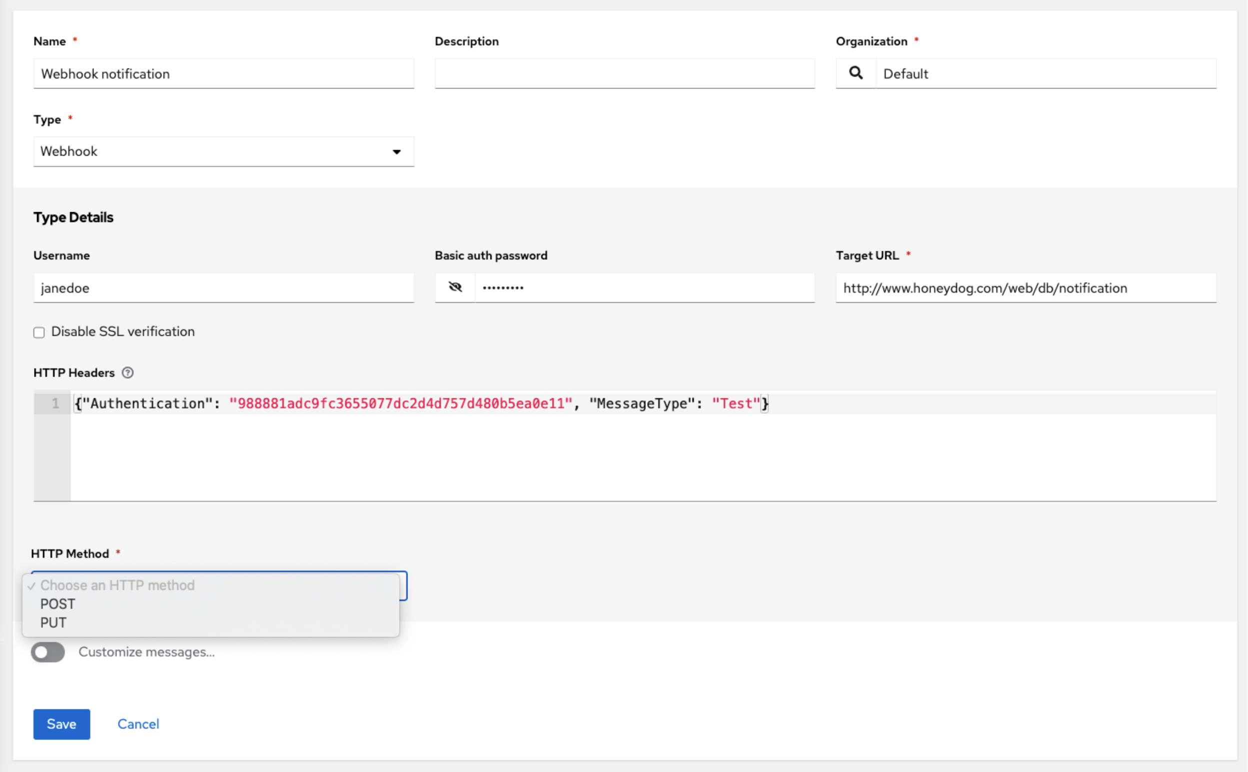
27.4.9 Webhook

The webhook notification type provides a simple interface to sending POSTs to a predefined web service. automation controller will POST to this address using application/json content type with the data payload containing all relevant details in json format. Some web service APIs expect HTTP requests to be in a certain format with certain fields. You can configure more of the webhook notification in the following ways:

- configure the HTTP method (using **POST** or **PUT**)
- body of the outgoing request
- configure authentication (using basic auth)

The parameters for configuring webhooks are:

- Username
- Basic Auth Password
- Target URL (required): The full URL to which the webhook notification will be PUT or POSTed.
- Disable SSL Verification: SSL verification is on by default, but you can choose to turn off verification of the authenticity of the target's certificate. Environments that use internal or private CA's should select this option to disable verification.
- HTTP Headers (required): Headers in JSON form where the keys and values are strings. For example, `{"Authentication": "988881adc9fc3655077dc2d4d757d480b5ea0e11", "MessageType": "Test"}`
- HTTP Method (required). Select the method for your webhook:
  - POST: Creates a new resource. Also acts as a catch-all for operations that do not fit into the other categories. It is likely you need to POST unless you know your webhook service expects a PUT.
  - PUT: Updates a specific resource (by an identifier) or a collection of resources. PUT can also be used to create a specific resource if the resource identifier is known beforehand.



Name *
Webhook notification

Description

Organization *
Default

Type *
Webhook

Type Details

Username
janedoe

Basic auth password
•••••••••

Target URL *
http://www.honeydog.com/web/db/notification

☐ Disable SSL verification

HTTP Headers

```
{"Authentication": "988881adc9fc3655077dc2d4d757d480b5ea0e11", "MessageType": "Test"}
```

HTTP Method *

Webhook payloads

Automation controller sends by default the following data at the webhook endpoint:

```
job id
name
url
created_by
started
finished
status
traceback
inventory
project
playbook
credential
limit
extra_vars
hosts
http method
```

An example of a `started` notifications via webhook message as it is returned by automation controller:

```
{"id": 38, "name": "Demo Job Template", "url": "https://host/#/jobs/playbook/38",
↪"created_by": "bianca", "started":
"2020-07-28T19:57:07.888193+00:00", "finished": null, "status": "running", "traceback
↪": "", "inventory": "Demo Inventory",
"project": "Demo Project", "playbook": "hello_world.yml", "credential": "Demo␣
↪Credential", "limit": "", "extra_vars": "{}",
"hosts": {}}POST / HTTP/1.1
```

Automation controller returns by default the following data at the webhook endpoint for a `success/fail` status:

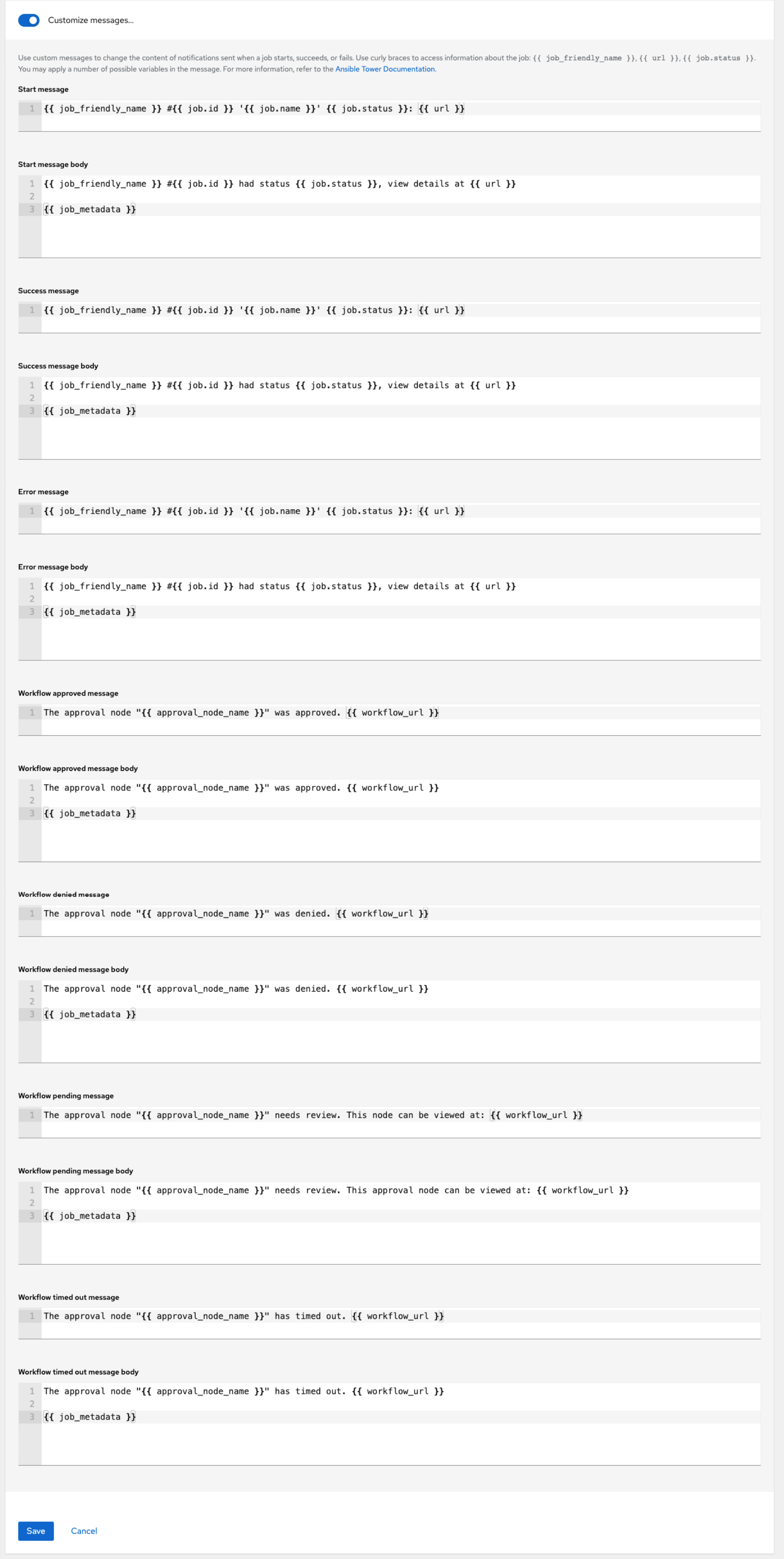
```
job id
name
url
created_by
started
finished
status
traceback
inventory
project
playbook
credential
limit
extra_vars
hosts
```

An example of a `success/fail` notifications via webhook message as it is returned by automation controller:

```
{"id": 46, "name": "AWX-Collection-tests-awx_job_wait-long_running-XVFBGRSAvUUIrYKn",
↪"url": "https://host/#/jobs/playbook/46",
"created_by": "bianca", "started": "2020-07-28T20:43:36.966686+00:00", "finished":
↪"2020-07-28T20:43:44.936072+00:00", "status": "failed",
"traceback": "", "inventory": "Demo Inventory", "project": "AWX-Collection-tests-awx_
↪job_wait-long_running-JJSlglnwtsRJyQmw", "playbook":
"fail.yml", "credential": null, "limit": "", "extra_vars": "{\"sleep_interval\": 300}
↪", "hosts": {"localhost": {"failed": true, "changed": 0,
"dark": 0, "failures": 1, "ok": 1, "processed": 1, "skipped": 0, "rescued": 0,
↪"ignored": 0}}}
```

27.5 Create custom notifications

You can *customize the text content* of each of the *Notification Types* by enabling the **Customize Messages** portion at the bottom of the notifications form using the toggle button.



Customize messages...

Use custom messages to change the content of notifications sent when a job starts, succeeds, or fails. Use curly braces to access information about the job: `{{ job_friendly_name }}`, `{{ url }}`, `{{ job.status }}`. You may apply a number of possible variables in the message. For more information, refer to the Ansible Tower Documentation.

Start message

```
{{ job_friendly_name }} #{{ job.id }} '{{ job.name }}' {{ job.status }}: {{ url }}
```

Start message body

```
{{ job_friendly_name }} #{{ job.id }} had status {{ job.status }}, view details at {{ url }}

{{ job_metadata }}
```

Success message

```
{{ job_friendly_name }} #{{ job.id }} '{{ job.name }}' {{ job.status }}: {{ url }}
```

Success message body

```
{{ job_friendly_name }} #{{ job.id }} had status {{ job.status }}, view details at {{ url }}

{{ job_metadata }}
```

Error message

```
{{ job_friendly_name }} #{{ job.id }} '{{ job.name }}' {{ job.status }}: {{ url }}
```

Error message body

```
{{ job_friendly_name }} #{{ job.id }} had status {{ job.status }}, view details at {{ url }}

{{ job_metadata }}
```

Workflow approved message

```
The approval node "{{ approval_node_name }}" was approved. {{ workflow_url }}
```

Workflow approved message body

```
The approval node "{{ approval_node_name }}" was approved. {{ workflow_url }}

{{ job_metadata }}
```

Workflow denied message

```
The approval node "{{ approval_node_name }}" was denied. {{ workflow_url }}
```

Workflow denied message body

```
The approval node "{{ approval_node_name }}" was denied. {{ workflow_url }}

{{ job_metadata }}
```

Workflow pending message

```
The approval node "{{ approval_node_name }}" needs review. This node can be viewed at: {{ workflow_url }}
```

Workflow pending message body

```
The approval node "{{ approval_node_name }}" needs review. This approval node can be viewed at: {{ workflow_url }}

{{ job_metadata }}
```

Workflow timed out message

```
The approval node "{{ approval_node_name }}" has timed out. {{ workflow_url }}
```

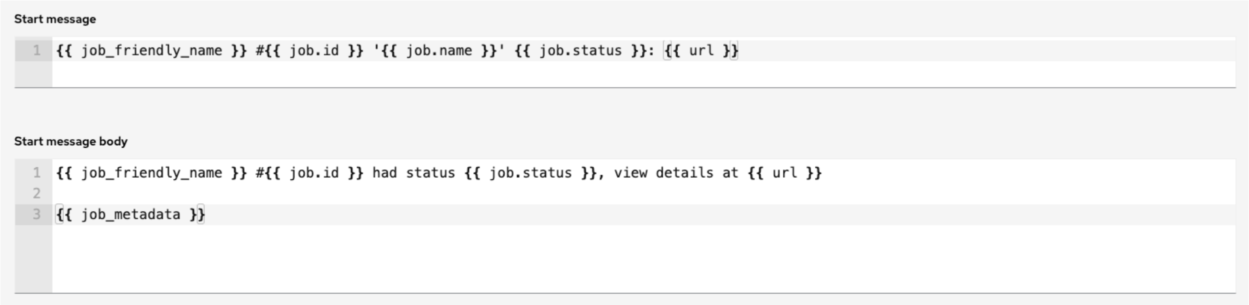
Workflow timed out message body

```
The approval node "{{ approval_node_name }}" has timed out. {{ workflow_url }}

{{ job_metadata }}
```

You can provide a custom message for various job events:

- Start
- Success
- Error
- Workflow approved
- Workflow denied
- Workflow running
- Workflow timed out

The message forms vary depending on the type of notification you are configuring. For example, messages for email and PagerDuty notifications have the appearance of a typical email form with a subject and body, in which case, automation controller displays the fields as **Message** and **Message Body**. Other notification types only expect a **Message** for each type of event:



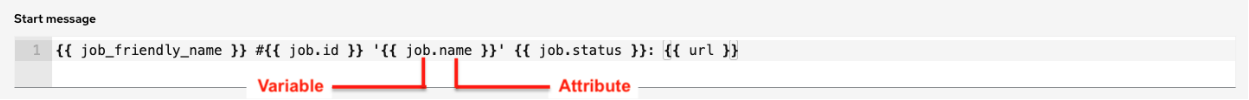
Start message

```
{{ job_friendly_name }} #{{ job.id }} '{{ job.name }}' {{ job.status }}: {{ url }}
```

Start message body

```
{{ job_friendly_name }} #{{ job.id }} had status {{ job.status }}, view details at {{ url }}

{{ job_metadata }}
```

The **Message** fields are pre-populated with a template containing a top-level variable, `job` coupled with an attribute, such as `id` or `name`, for example. Templates are enclosed in curly braces and may draw from a fixed set of fields provided by automation controller, as shown in the pre-populated **Messages** fields.



Start message

```
{{ job_friendly_name }} #{{ job.id }} '{{ job.name }}' {{ job.status }}: {{ url }}
```

This pre-populated field suggests commonly displayed messages to a recipient who is notified of an event. You can, however, customize these messages with different criteria by adding your own attribute(s) for the job as needed. Custom notification messages are rendered using Jinja - the same templating engine used by Ansible playbooks.

Messages and message bodies have different types of content:

- messages will always just be strings (one-liners only; new lines are not allowed)
- message bodies will be either a dictionary or block of text:
  - the message body for *Webhooks* and *PagerDuty* uses dictionary definitions. The default message body for these is `{{ job_metadata }}`, you can either leave that as is or provide your own dictionary
  - the message body for email uses a block of text or a multi-line string. The default message body is:

```
{{ job_friendly_name }} #{{ job.id }} had status {{ job.status }}, view details at {{ url }} {{ job_metadata }}
```

You can tweak this text (leaving `{{ job_metadata }}` in, or drop `{{ job_metadata }}` altogether). Since the body is a block of text, it can really be any string you want.

`{{ job_metadata }}` gets rendered as a dictionary containing fields that describe the job being executed. In all cases, `{{ job_metadata }}` will include the following fields:

- `id`
- `name`
- `url`
- `created_by`
- `started`
- `finished`
- `status`
- `traceback`

Note: At the present time, you cannot query individual fields within `{{ job_metadata }}`. When using `{{ job_metadata }}` in a notification template, all data will be returned.

The resulting dictionary will look something like this:

```
{"id": 18,
 "name": "Project - Space Procedures",
 "url": "https://host/#/jobs/project/18",
 "created_by": "admin",
 "started": "2019-10-26T00:20:45.139356+00:00",
 "finished": "2019-10-26T00:20:55.769713+00:00",
 "status": "successful",
 "traceback": ""
}
```

If `{{ job_metadata }}` is rendered in a job, it will include the following additional fields:

- `inventory`
- `project`
- `playbook`
- `credential`
- `limit`
- `extra_vars`
- `hosts`

The resulting dictionary will look something like:

```
{"id": 12,
 "name": "JobTemplate - Launch Rockets",
 "url": "https://host/#/jobs/playbook/12",
 "created_by": "admin",
 "started": "2019-10-26T00:02:07.943774+00:00",
```

(continues on next page)

(continued from previous page)

```
 "finished": null,
 "status": "running",
 "traceback": "",
 "inventory": "Inventory - Fleet",
 "project": "Project - Space Procedures",
 "playbook": "launch.yml",
 "credential": "Credential - Mission Control",
 "limit": "",
 "extra_vars": "{}",
 "hosts": {}
}
```

If `{{ job_metadata }}` is rendered in a workflow job, it will include the following additional field:

- `body` (this will enumerate all the nodes in the workflow job and includes a description of the job associated with each node)

The resulting dictionary will look something like this:

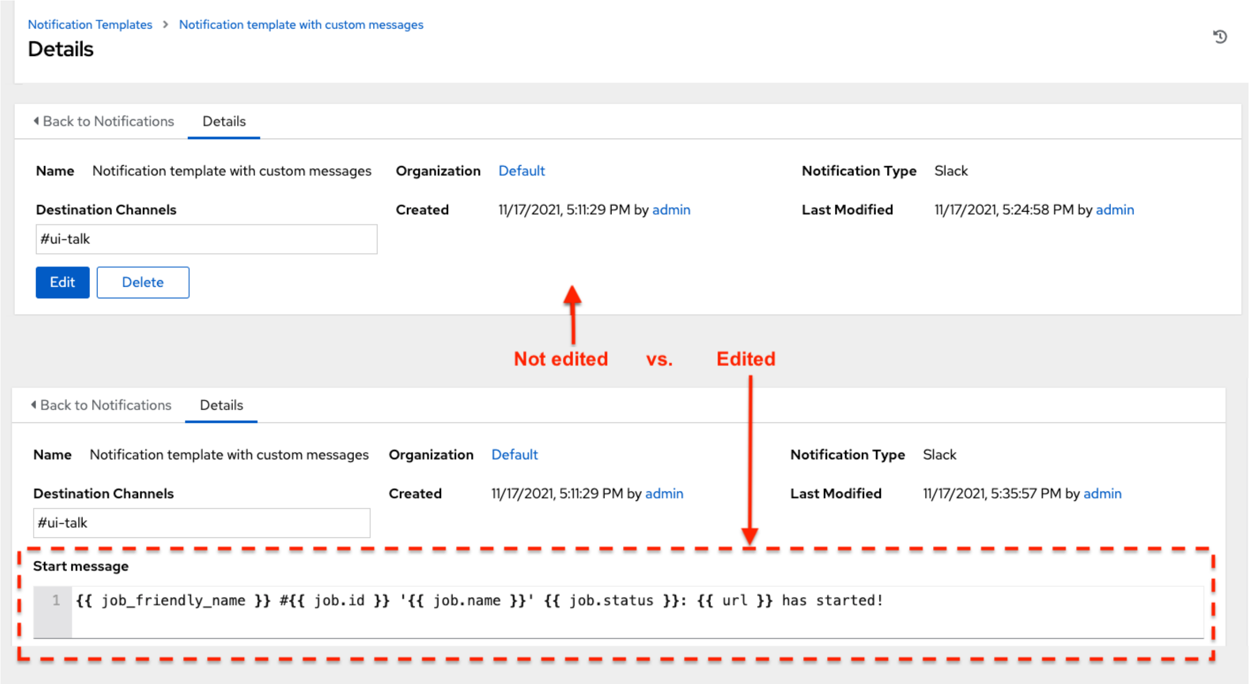
```
{"id": 14,
 "name": "Workflow Job Template - Launch Mars Mission",
 "url": "https://host/#/workflows/14",
 "created_by": "admin",
 "started": "2019-10-26T00:11:04.554468+00:00",
 "finished": "2019-10-26T00:11:24.249899+00:00",
 "status": "successful",
 "traceback": "",
 "body": "Workflow job summary:

         node #1 spawns job #15, \"Assemble Fleet JT\", which finished␣
↪with status successful.
         node #2 spawns job #16, \"Mission Start approval node\", which␣
↪finished with status successful.\n
         node #3 spawns job #17, \"Deploy Fleet\", which finished with␣
↪status successful."
}
```

For more detail, refer to Using variables with Jinja2.

Automation controller requires valid syntax in order to retrieve the correct data to display the messages. For a list of supported attributes and the proper syntax construction, refer to the *Supported Attributes for Custom Notifications* section of this guide.

If you create a notification template that uses invalid syntax or references unusable fields, an error message displays indicating the nature of the error. If you delete a notification's custom message, the default message is shown in its place.

Note: If you save the notifications template without editing the custom message (or edit and revert back to the default values), the **Details** screen assumes the defaults and will not display the custom message tables. If you edit and save any of the values, the entire table displays in the **Details** screen.



Notification Templates > Notification template with custom messages

Details

Back to Notifications | Details

Name: Notification template with custom messages
Organization: Default
Notification Type: Slack
Destination Channels: #ui-talk
Created: 11/17/2021, 5:11:29 PM by admin
Last Modified: 11/17/2021, 5:24:58 PM by admin

Edit | Delete

Not edited vs. Edited

Back to Notifications | Details

Name: Notification template with custom messages
Organization: Default
Notification Type: Slack
Destination Channels: #ui-talk
Created: 11/17/2021, 5:11:29 PM by admin
Last Modified: 11/17/2021, 5:35:57 PM by admin

Start message

```
{{ job_friendly_name }} #{{ job.id }} '{{ job.name }}' {{ job.status }}: {{ url }} has started!
```

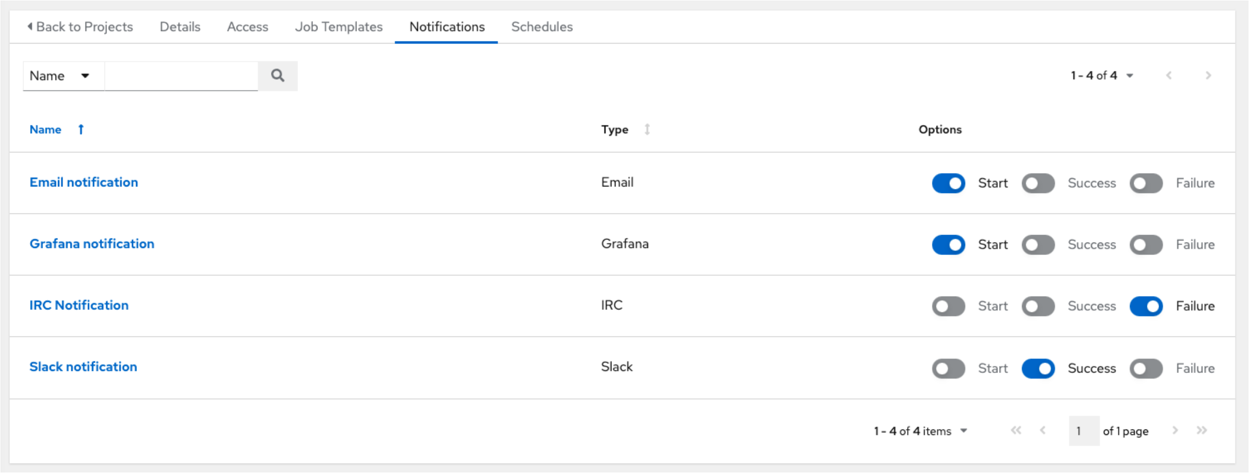
27.6 Enable and Disable Notifications

You can select which notifications to notify you when a specific job starts, in addition to notifying you on success or failure at the end of the job run. Some behaviors to keep in mind:

- if a workflow template (WFJT) has notification on start enabled, and a job template (JT) within that workflow also has notification on start enabled, you will receive notifications for both
- you can enable notifications to run on many JTs within a WFJT
- you can enable notifications to run on a sliced job template (SJT) start and each slice will generate a notification
- when you enable a notification to run on job start, and that notification gets deleted, the JT continues to run, but will result in an error message

You can enable notifications on job start, job success, and job failure, or any combination thereof, from the **Notifications** tab of the following resources:

- Job Template
- Workflow Template
- Projects (shown in the example below)
- Inventory Source
- Organizations



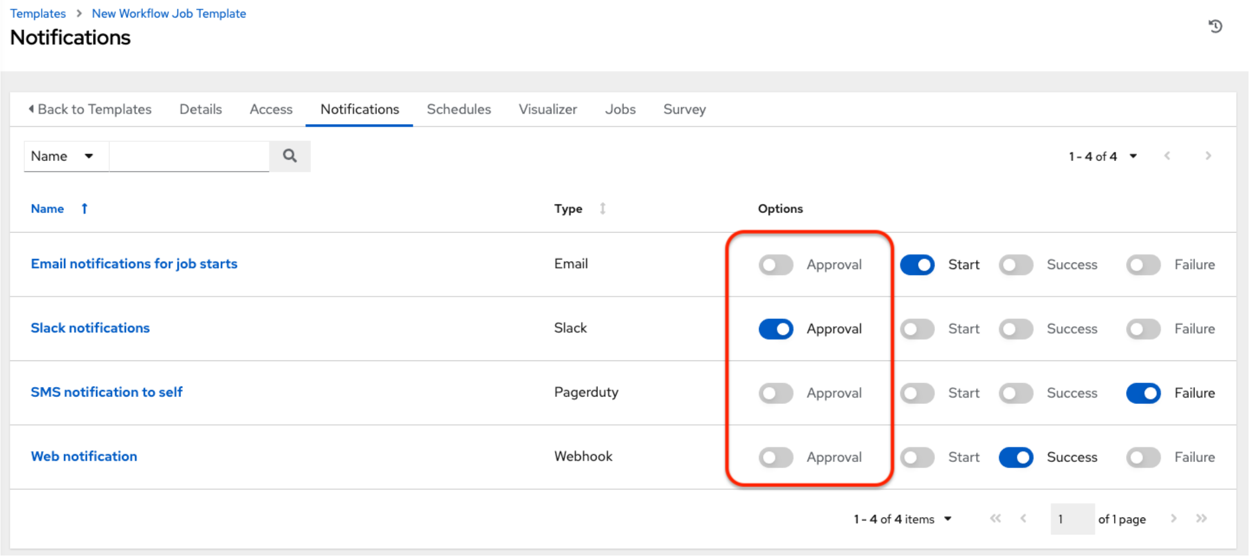
◂ Back to Projects | Details | Access | Job Templates | Notifications | Schedules

Name ▾ 1 - 4 of 4 ▾

Name	Type	Options
Email notification	Email	● Start ○ Success ○ Failure
Grafana notification	Grafana	● Start ○ Success ○ Failure
IRC Notification	IRC	○ Start ○ Success ● Failure
Slack notification	Slack	○ Start ● Success ○ Failure

1 - 4 of 4 items ▾ « ‹ 1 of 1 page › »

For workflow templates that have approval nodes, in addition to *Start*, *Success*, and *Failure*, you can enable or disable certain approval-related events:



Templates > New Workflow Job Template

Notifications

◂ Back to Templates | Details | Access | Notifications | Schedules | Visualizer | Jobs | Survey

Name ▾ 1 - 4 of 4 ▾

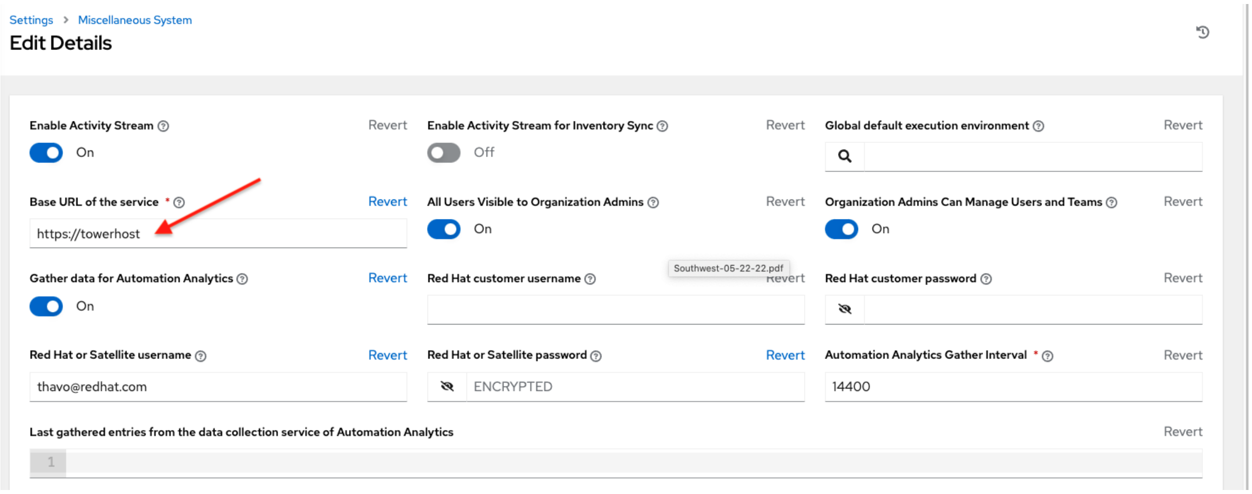
Name	Type	Options
Email notifications for job starts	Email	○ Approval ● Start ○ Success ○ Failure
Slack notifications	Slack	● Approval ○ Start ○ Success ○ Failure
SMS notification to self	Pagerduty	○ Approval ○ Start ○ Success ● Failure
Web notification	Webhook	○ Approval ○ Start ● Success ○ Failure

1 - 4 of 4 items ▾ « ‹ 1 of 1 page › »

Refer to *Approval nodes* for additional detail on working with these types of nodes.

27.7 Configure the `host` hostname for notifications

In the System Settings, you can replace the default value in the **Base URL of the service** field with your preferred hostname to change the notification hostname.



Settings > Miscellaneous System

Edit Details

Enable Activity Stream: On (Revert)

Enable Activity Stream for Inventory Sync: Off (Revert)

Global default execution environment: (Revert)

Base URL of the service *: https://towerhost (Revert)

All Users Visible to Organization Admins: On (Revert)

Organization Admins Can Manage Users and Teams: On (Revert)

Gather data for Automation Analytics: On (Revert)

Red Hat customer username: (Revert)

Southwest-05-22-22.pdf

Red Hat customer password: (Revert)

Red Hat or Satellite username: thavo@redhat.com (Revert)

Red Hat or Satellite password: ENCRYPTED (Revert)

Automation Analytics Gather Interval *: 14400 (Revert)

Last gathered entries from the data collection service of Automation Analytics: 1 (Revert)

Refreshing your license also changes the notification hostname. New installations of automation controller should not have to set the hostname for notifications.

27.7.1 Reset the `TOWER_URL_BASE`

The primary way that automation controller determines how the base URL (`TOWER_URL_BASE`) is defined is by looking at an incoming request and setting the server address based on that incoming request.

Automation controller takes settings values from the database first. If no settings values are found, it falls back to using the values from the settings files. If a user posts a license by navigating to the automation controller host's IP adddress, the posted license is written to the settings entry in the database.

To change the `TOWER_URL_BASE` if the wrong address has been picked up, navigate to **Miscellaneous System settings** from the Settings menu using the DNS entry you wish to appear in notifications, and re-add your license.

27.8 Notifications API

Use the `started`, `success`, or `error` endpoints:

```
/api/v2/organizations/N/notification_templates_started/
/api/v2/organizations/N/notification_templates_success/
/api/v2/organizations/N/notification_templates_error/
```

Additionally, the `../../../N/notification_templates_started` endpoints have **GET** and **POST** actions for:

- Organizations
- Projects
- Inventory Sources
- Job Templates
- System Job Templates
- Workflow Job Templates

CHAPTER

TWENTYEIGHT

SUPPORTED ATTRIBUTES FOR CUSTOM NOTIFICATIONS

This section describes the list of supported job attributes and the proper syntax for constructing the message text for notifications. The supported job attributes are:

- `allow_simultaneous` - (boolean) indicates if multiple jobs can run simultaneously from the JT associated with this job
- `controller_node` - (string) the instance that managed the isolated execution environment
- `created` - (datetime) timestamp when this job was created
- `custom_virtualenv` - (string) custom virtual environment used to execute job
- `description` - (string) optional description of the job
- `diff_mode` - (boolean) if enabled, textual changes made to any templated files on the host are shown in the standard output
- `elapsed` - (decimal) elapsed time in seconds that the job ran
- `execution_node` - (string) node the job executed on
- `failed` - (boolean) true if job failed
- `finished` - (datetime) date and time the job finished execution
- `force_handlers` - (boolean) when handlers are forced, they will run when notified even if a task fails on that host (note that some conditions - e.g. unreachable hosts - can still prevent handlers from running)
- `forks` - (int) number of forks requested for job
- `id` - (int) database id for this job
- `job_explanation` - (string) status field to indicate the state of the job if it wasn't able to run and capture stdout
- `job_slice_count` - (integer) if run as part of a sliced job, the total number of slices (if 1, job is not part of a sliced job)
- `job_slice_number` - (integer) if run as part of a sliced job, the ID of the inventory slice operated on (if not part of a sliced job, attribute is not used)
- `job_tags` - (string) only tasks with specified tags will execute
- `job_type` - (choice) run, check, or scan
- `launch_type` - (choice) manual, relaunch, callback, scheduled, dependency, workflow, sync, or scm
- `limit` - (string) playbook execution limited to this set of hosts, if specified
- `modified` - (datetime) timestamp when this job was last modified
- `name` - (string) name of this job

- `playbook` - (string) playbook executed
- `scm_revision` - (string) scm revision from the project used for this job, if available
- `skip_tags` - (string) playbook execution skips over this set of tag(s), if specified
- `start_at_task` - (string) playbook execution begins at the task matching this name, if specified
- `started` - (datetime) date and time the job was queued for starting
- `status` - (choice) new, pending, waiting, running, successful, failed, error, canceled
- `timeout` - (int) amount of time (in seconds) to run before the task is canceled
- `type` - (choice) data type for this job
- `url` - (string) URL for this job
- `use_fact_cache` - (boolean) if enabled for job, the controller acts as an Ansible Fact Cache Plugin, persisting facts at the end of a playbook run to the database and caching facts for use by Ansible
- `verbosity` - (choice) 0 through 5 (corresponding to Normal through WinRM Debug)
- **`host_status_counts` (count of hosts uniquely assigned to each status)**
    - `skipped` (integer)
    - `ok` (integer)
    - `changed` (integer)
    - `failures` (integer)
    - `dark` (integer)
    - `processed` (integer)
    - `rescued` (integer)
    - `ignored` (integer)
    - `failed` (boolean)
- **`summary_fields`:**
    - **`inventory`**
        - `id` - (integer) database ID for inventory
        - `name` - (string) name of the inventory
        - `description` - (string) optional description of the inventory
        - `has_active_failures` - (boolean) (deprecated) flag indicating whether any hosts in this inventory have failed
        - `total_hosts` - (deprecated) (int) total number of hosts in this inventory.
        - `hosts_with_active_failures` - (deprecated) (int) number of hosts in this inventory with active failures
        - `total_groups` - (deprecated) (int) total number of groups in this inventory
        - `groups_with_active_failures` - (deprecated) (int) number of hosts in this inventory with active failures
        - `has_inventory_sources` - (deprecated) (boolean) flag indicating whether this inventory has external inventory sources

* `total_inventory_sources` - (int) total number of external inventory sources configured within this inventory
* `inventory_sources_with_failures` - (int) number of external inventory sources in this inventory with failures
* `organization_id` - (id) organization containing this inventory
* `kind` - (choice) (empty string) (indicating hosts have direct link with inventory) or 'smart'

- **`project`**
  * `id` - (int) database ID for project
  * `name` - (string) name of the project
  * `description` - (string) optional description of the project
  * `status` - (choices) one of new, pending, waiting, running, successful, failed, error, canceled, never updated, ok, or missing
  * `scm_type (choice)` - one of (empty string), git, hg, svn, insights
- **`job_template`**
  * `id` - (int) database ID for job template
  * `name` - (string) name of job template
  * `description` - (string) optional description for the job template
- **`unified_job_template`**
  * `id` - (int) database ID for unified job template
  * `name` - (string) name of unified job template
  * `description` - (string) optional description for the unified job template
  * `unified_job_type` - (choice) unified job type (job, workflow_job, project_update, etc.)
- **`instance_group`**
  * `id` - (int) database ID for instance group
  * `name` - (string) name of instance group
- **`created_by`**
  * `id` - (int) database ID of user that launched the operation
  * `username` - (string) username that launched the operation
  * `first_name` - (string) first name
  * `last_name` - (string) last name
- **`labels`**
  * `count` - (int) number of labels
  * `results` - list of dictionaries representing labels (e.g. {"id": 5, "name": "database jobs"})

Information about a job can be referenced in a custom notification message using grouped curly braces `{{ }}`. Specific job attributes are accessed using dotted notation, for example `{{ job.summary_fields.inventory.name }}`. Any characters used in front or around the braces, or plain text, can be added for clarification, such as '#' for job ID and single-quotes to denote some descriptor. Custom messages can include a number of variables throughout the message:

```
{{ job_friendly_name }} {{ job.id }} ran on {{ job.execution_node }} in {{ job.
↪elapsed }} seconds.
```

In addition to the job attributes, there are some other variables that can be added to the template:

- `approval_node_name` - (string) the approval node name
- `approval_status` - (choice) one of approved, denied, and timed_out
- `url` - (string) URL of the job for which the notification is emitted (this applies to start, success, fail, and approval notifications)
- `workflow_url` - (string) URL to the relevant approval node. This allows the notification recipient to go to the relevant workflow job page to see what's going on (i.e., `This node can be viewed at: {{ workflow_url }}`). In cases of approval-related notifications, both `url` and `workflow_url` are the same.
- `job_friendly_name` - (string) the friendly name of the job
- `job_metadata` - (string) job metadata as a JSON string, for example:

```
{'url': 'https://towerhost/$/jobs/playbook/13',
 'traceback': '',
 'status': 'running',
 'started': '2019-08-07T21:46:38.362630+00:00',
 'project': 'Stub project',
 'playbook': 'ping.yml',
 'name': 'Stub Job Template',
 'limit': '',
 'inventory': 'Stub Inventory',
 'id': 42,
 'hosts': {},
 'friendly_name': 'Job',
 'finished': False,
 'credential': 'Stub credential',
 'created_by': 'admin'}
```

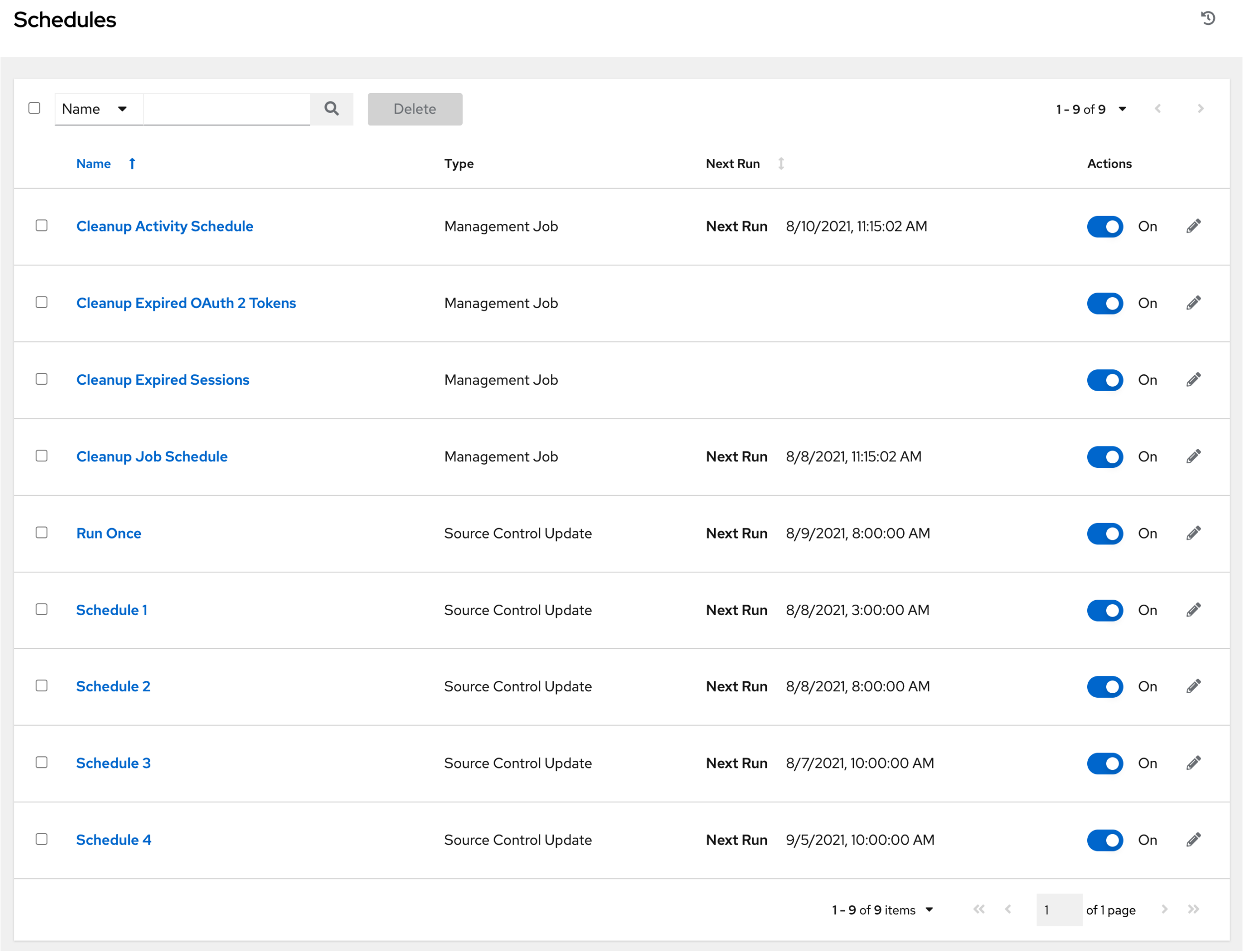
CHAPTER

TWENTYNINE

SCHEDULES

You can access all your configured schedules by clicking **Schedules** from the left navigation bar. The schedules list may be sorted by any of the attributes from each column using the directional arrows. You can also search by name, date, or the name of the month in which a schedule runs.

Each schedule has a corresponding **Actions** column that has options to enable/disable that schedule using the **ON/OFF** toggle next to the schedule name and to allow editing (✎) of that schedule.



Schedules

□ Name ▾ 🔍 Delete 1 - 9 of 9 ▾ ‹ ›

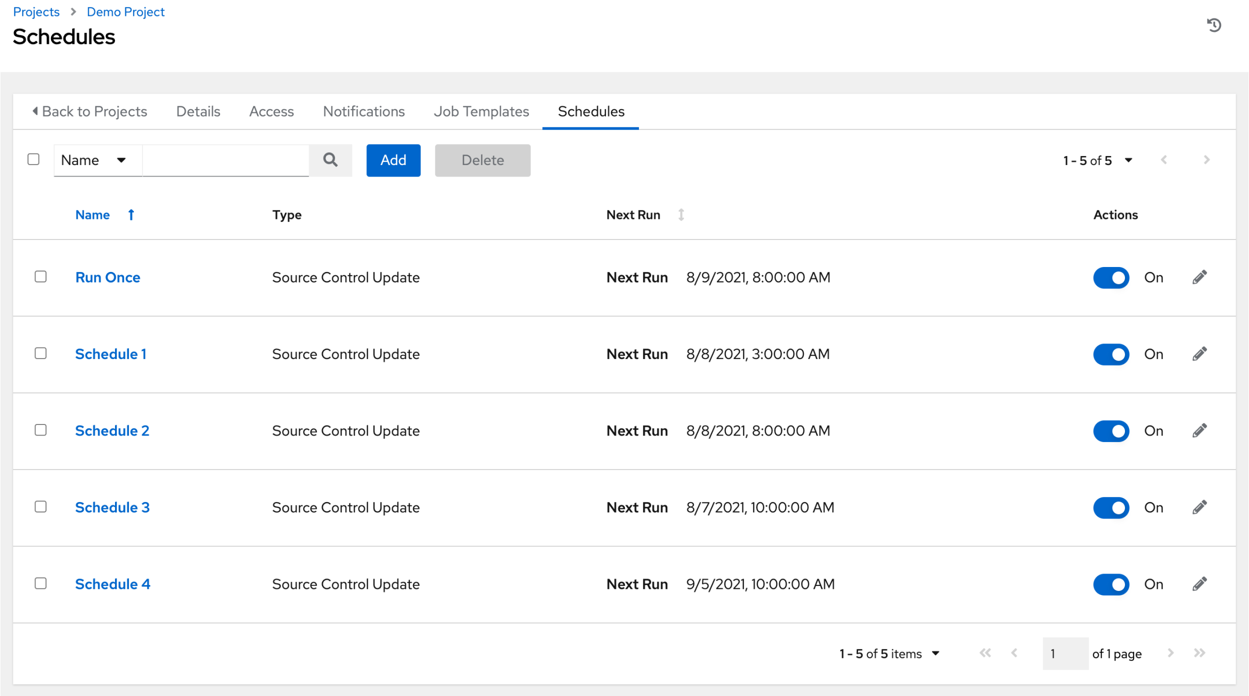
□	Name ↑	Type	Next Run	Actions
□	Cleanup Activity Schedule	Management Job	Next Run 8/10/2021, 11:15:02 AM	On ✎
□	Cleanup Expired OAuth 2 Tokens	Management Job		On ✎
□	Cleanup Expired Sessions	Management Job		On ✎
□	Cleanup Job Schedule	Management Job	Next Run 8/8/2021, 11:15:02 AM	On ✎
□	Run Once	Source Control Update	Next Run 8/9/2021, 8:00:00 AM	On ✎
□	Schedule 1	Source Control Update	Next Run 8/8/2021, 3:00:00 AM	On ✎
□	Schedule 2	Source Control Update	Next Run 8/8/2021, 8:00:00 AM	On ✎
□	Schedule 3	Source Control Update	Next Run 8/7/2021, 10:00:00 AM	On ✎
□	Schedule 4	Source Control Update	Next Run 9/5/2021, 10:00:00 AM	On ✎

1 - 9 of 9 items ▾ « ‹ 1 of 1 page › »

If you are setting up a template, a project, or an inventory source, clicking on the **Schedules** tab allows you to configure schedules for these resources. Once schedules are created, they are listed by:

- **Name**: Clicking the schedule name opens its details

- **Type**: Identifies whether the schedule is associated with a source control update or a system-managed job schedule
- **Next Run**: The next scheduled run of this task



Projects > Demo Project

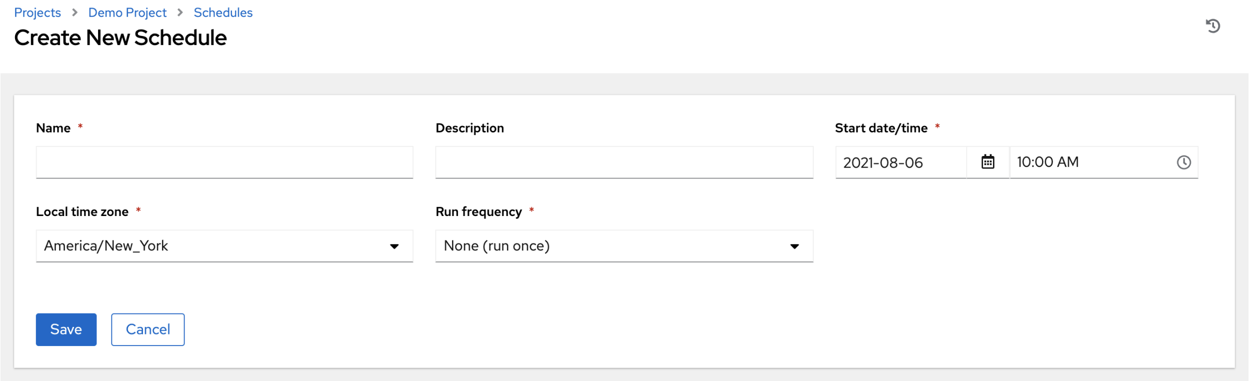
Schedules

	Name	Type	Next Run	Actions
□	Run Once	Source Control Update	Next Run 8/9/2021, 8:00:00 AM	On
□	Schedule 1	Source Control Update	Next Run 8/8/2021, 3:00:00 AM	On
□	Schedule 2	Source Control Update	Next Run 8/8/2021, 8:00:00 AM	On
□	Schedule 3	Source Control Update	Next Run 8/7/2021, 10:00:00 AM	On
□	Schedule 4	Source Control Update	Next Run 9/5/2021, 10:00:00 AM	On

29.1 Add a new schedule

Schedules can only be created from a template, project, or inventory source, and not directly on the main **Schedules** screen itself. To create a new schedule:

1. Click the **Schedules** tab of the resource you are configuring (template, project, or inventory source).
2. Click the **Add** button, which opens the **Create Schedule** window.



Projects > Demo Project > Schedules

Create New Schedule

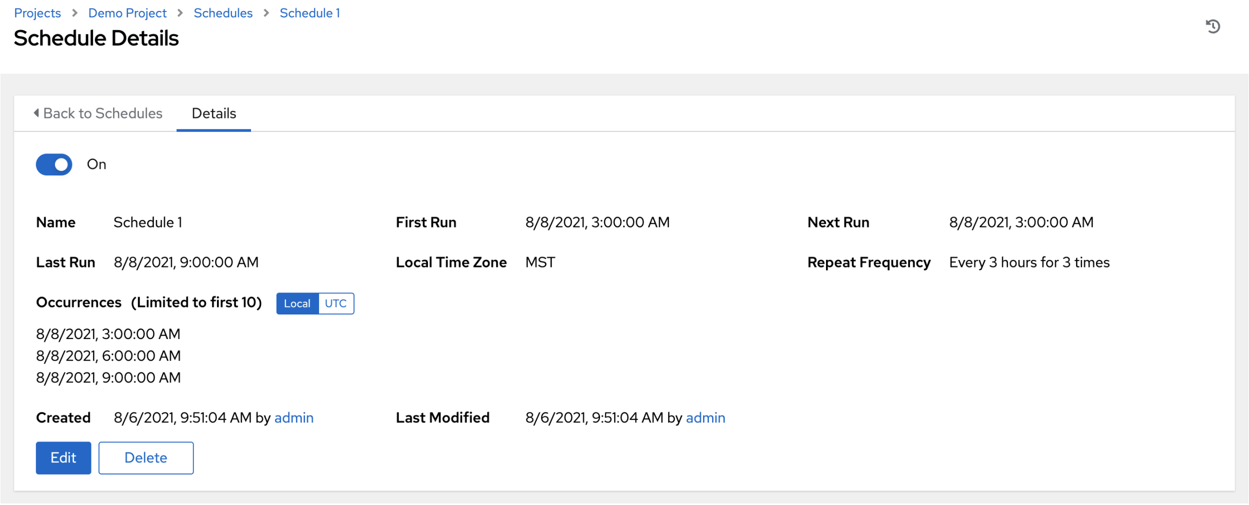
Name *	Description	Start date/time * 2021-08-06 10:00 AM
Local time zone * America/New_York	Run frequency * None (run once)	

3. Enter the appropriate details into the following fields:

- **Name** (required)
- **Start Date** (required)

- **Start Time** (required)
- **Local Time Zone** - The entered Start Time should be in this timezone
- **Repeat Frequency** - Appropriate scheduling options display depending on the frequency you select

The **Schedule Details** displays when you established a schedule, allowing you to review the schedule settings and a list of the scheduled occurrences in the selected Local Time Zone.



On

Name	Schedule 1	First Run	8/8/2021, 3:00:00 AM	Next Run	8/8/2021, 3:00:00 AM
Last Run	8/8/2021, 9:00:00 AM	Local Time Zone	MST	Repeat Frequency	Every 3 hours for 3 times

Occurrences (Limited to first 10) Local UTC

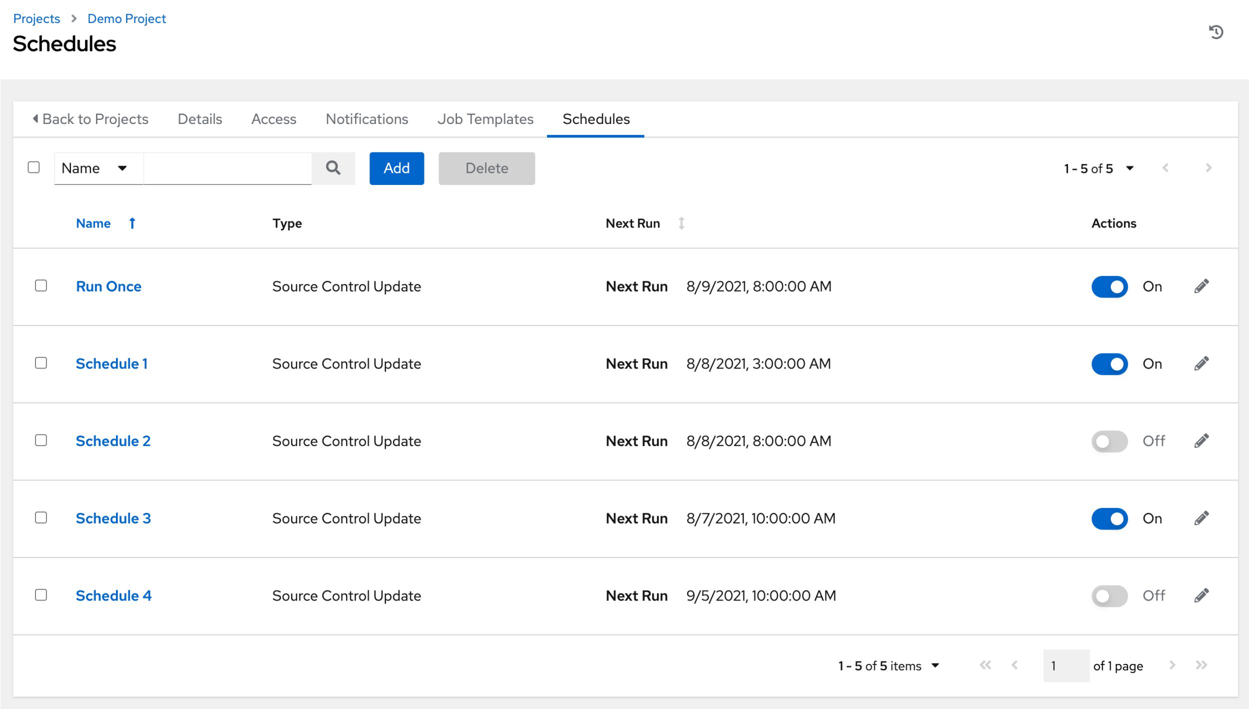
8/8/2021, 3:00:00 AM
8/8/2021, 6:00:00 AM
8/8/2021, 9:00:00 AM

Created	8/6/2021, 9:51:04 AM by admin	Last Modified	8/6/2021, 9:51:04 AM by admin

> **Caution:** Jobs are scheduled in UTC. Repeating jobs that run at a specific time of day may move relative to a local timezone when Daylight Savings Time shifts occur. The system resolves the local time zone based time to UTC when the schedule is saved. To ensure your schedules are correctly set, you should set your schedules in UTC time.

4. Once done, click **Save**.

You can use the **ON/OFF** toggle button to stop an active schedule or activate a stopped schedule.



	Name	Type	Next Run		Actions
□	Run Once	Source Control Update	Next Run	8/9/2021, 8:00:00 AM	On
□	Schedule 1	Source Control Update	Next Run	8/8/2021, 3:00:00 AM	On
□	Schedule 2	Source Control Update	Next Run	8/8/2021, 8:00:00 AM	Off
□	Schedule 3	Source Control Update	Next Run	8/7/2021, 10:00:00 AM	On
□	Schedule 4	Source Control Update	Next Run	9/5/2021, 10:00:00 AM	Off

CHAPTER

THIRTY

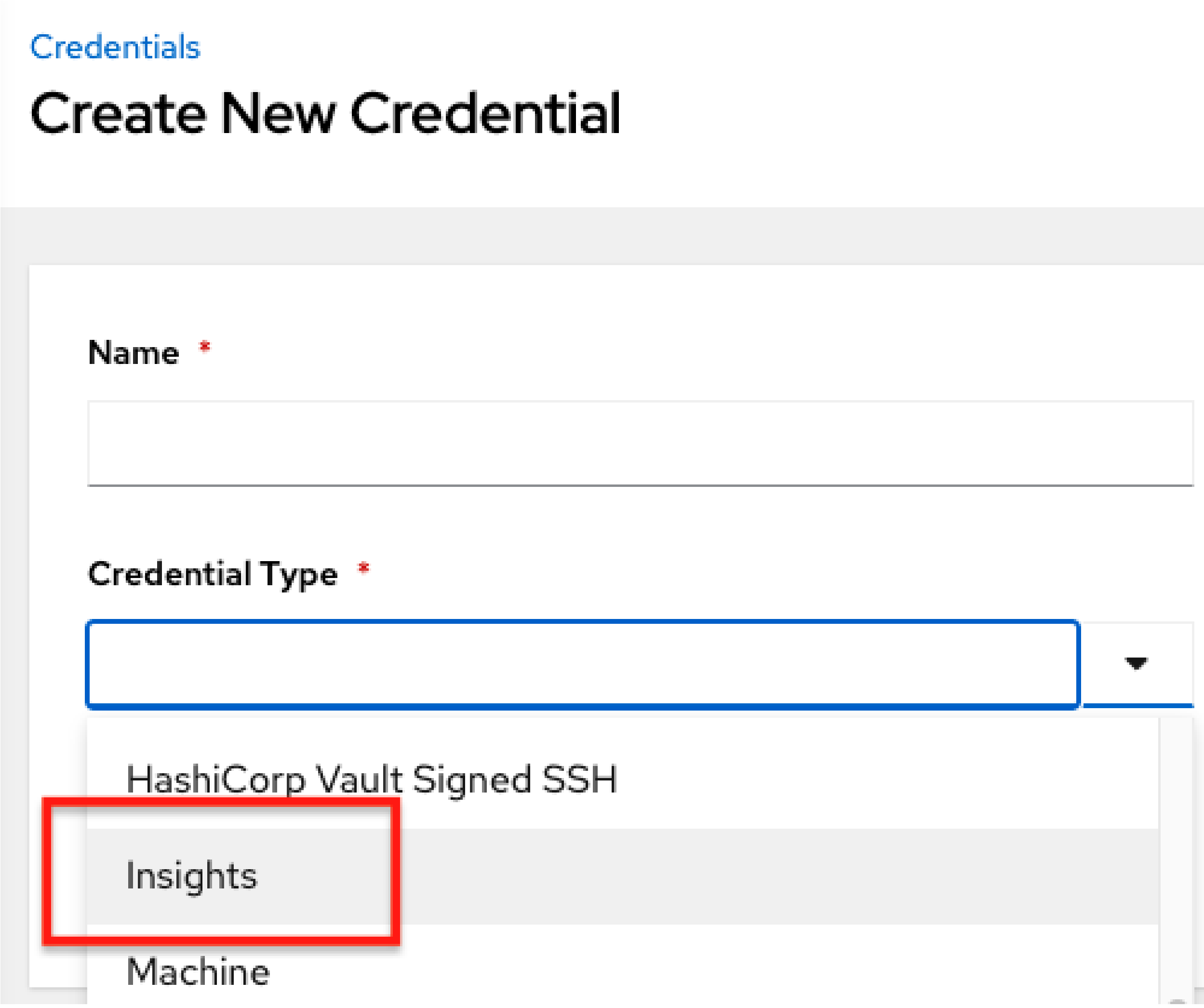
SETTING UP INSIGHTS REMEDIATIONS

Automation controller supports integration with Red Hat Insights. Once a host is registered with Insights, it will be continually scanned for vulnerabilities and known configuration conflicts. Each of the found problems may have an associated fix in the form of an Ansible playbook. Insights users create a maintenance plan to group the fixes and, ultimately, create a playbook to mitigate the problems. Automation controller tracks the maintenance plan playbooks via an Insights project. Authentication to Insights via Basic Auth is backed by a special Insights Credential, which must first be established in automation controller. To ultimately run an Insights Maintenance Plan, you need an Insights project, and an Insights inventory.

30.1 Create Insights Credential

To create a new credential for use with Insights:

1. Click **Credentials** from the left navigation bar to access the Credentials page.
2. Click the **Add** button located in the upper right corner of the Credentials screen.
3. Enter the name of the credential to be used in the **Name** field.
4. Optionally enter a description for this credential in the **Description** field.
5. In the **Organization** field, optionally enter the name of the organization with which the credential is associated, or click the 🔍 button and select it from the pop-up window.
6. In the **Credential Type** field, enter **Insights** or select it from the drop-down list.



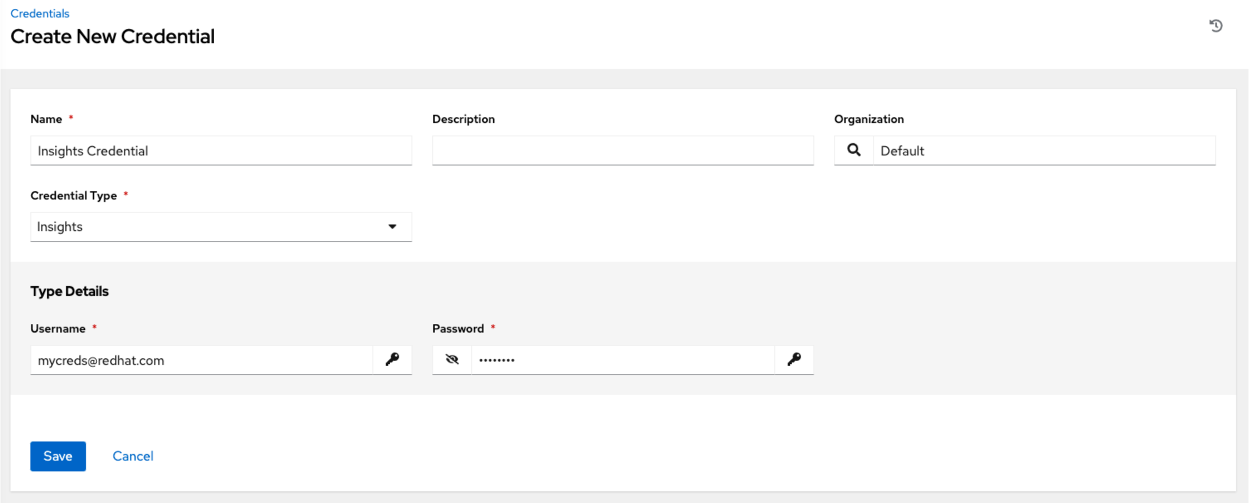
Credentials

Create New Credential

Name *

Credential Type *

7. Enter a valid Insights credential in the **Username** and **Password** fields. The Insights credential is the user's Red Hat Customer Portal account username and password.



Credentials

Create New Credential

Name *: Insights Credential

Description:

Organization: Default

Credential Type *: Insights

Type Details

Username *: mycreds@redhat.com

Password *: ••••••••

8. Click **Save** when done.

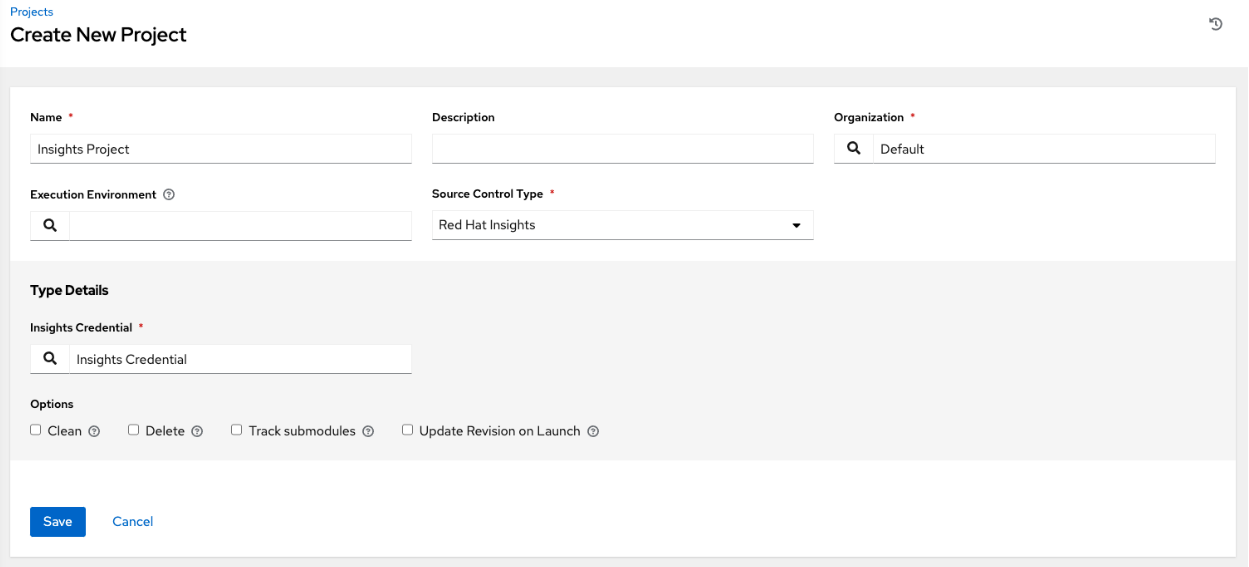
30.2 Create an Insights Project

To create a new Insights project:

1. Click **Projects** from the left navigation bar to access the Projects page.
2. Click the **Add** button located in the upper right corner of the Projects screen.
3. Enter the appropriate details into the required fields, at minimum. Note the following fields requiring specific Insights-related entries:

- **Name**: Enter the name for your Insights project.
- **Organization**: Enter the name of the organization associated with this project, or click the 🔍 button and select it from the pop-up window.
- **SCM Type**: Select **Red Hat Insights**.
- Upon selecting the SCM type, the **Source Details** field expands.

4. The **Credential** field is pre-populated with the Insights credential you previously created. If not, enter the credential, or click the 🔍 button and select it from the pop-up window.
5. Click to select the update option(s) for this project from the **Options** field, and provide any additional values, if applicable. For information about each option, click the tooltip ❔ next to the options.



Projects

Create New Project

Name * : Insights Project

Description :

Organization * : Default

Execution Environment :

Source Control Type * : Red Hat Insights

Type Details

Insights Credential * : Insights Credential

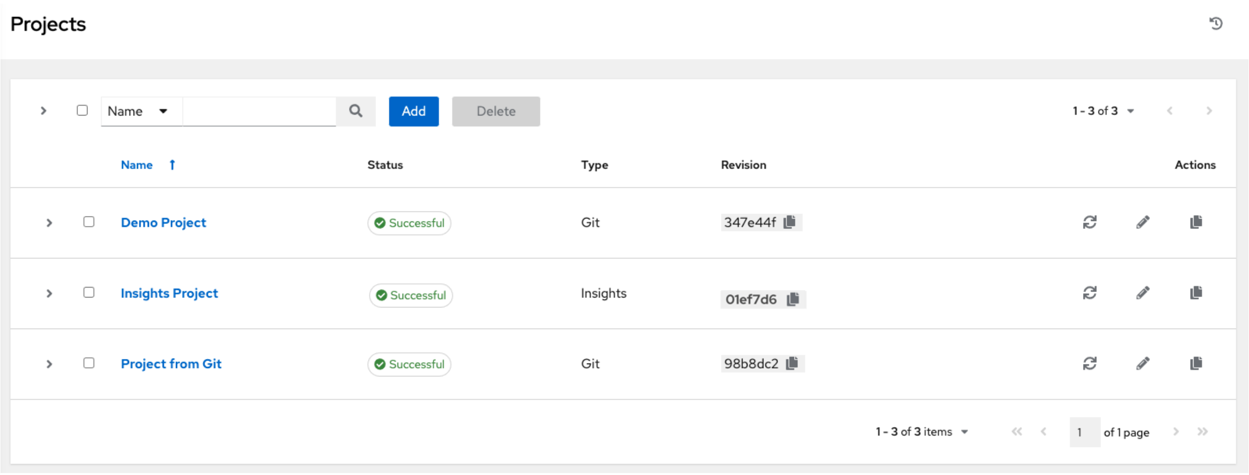
Options

□ Clean □ Delete □ Track submodules □ Update Revision on Launch

6. Click **Save** when done.

All SCM/Project syncs occur automatically the first time you save a new project. However, if you want them to be updated to what is current in Insights, manually update the SCM-based project by clicking the ☁ button under the project's available Actions.

This process syncs your Insights project with your Insights account solution. Notice that the status dot beside the name of the project updates once the sync has run.



Projects

Name	Status	Type	Revision	Actions
Demo Project	Successful	Git	347e44f	
Insights Project	Successful	Insights	01ef7d6	
Project from Git	Successful	Git	98b8dc2	

30.3 Create Insights Inventory

The Insights playbook contains a *hosts:* line where the value is the hostname that Insights itself knows about, which may be different than the hostname that Tower knows about. To use an Insights playbook, you will need an Insights inventory.

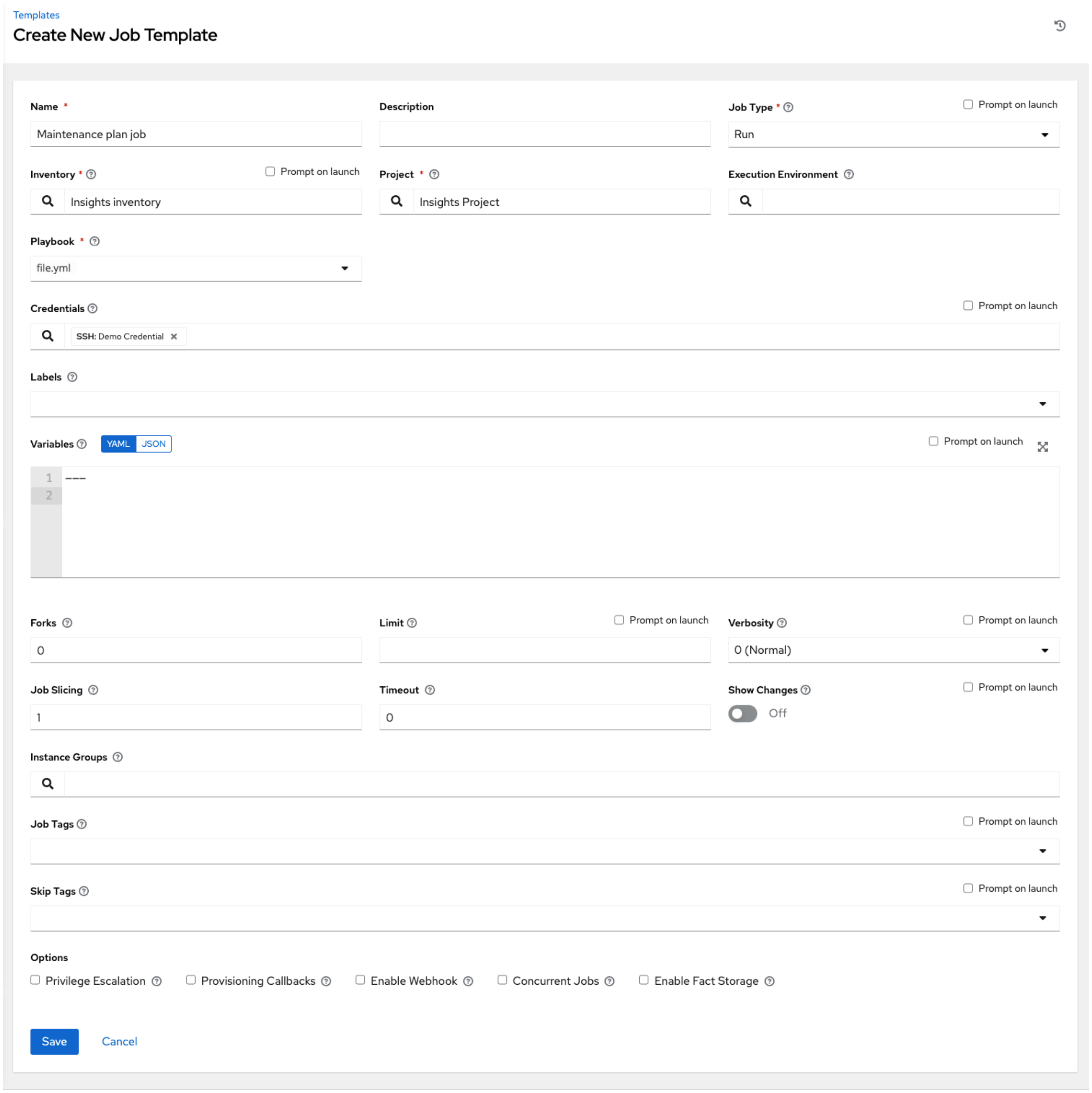
To create a new inventory for use with Insights, see *Red Hat Insights*.

30.4 Remediate Insights Inventory

Remediation of an Insights inventory allows Tower to run Insights playbooks with a single click. This is done by creating a Job Template to run the Insights remediation.

1. Click **Job Templates** from the left navigation bar to access the Job Templates page.
2. Create a new Job Template, with the appropriate details into the required fields, at minimum. Note the following fields requiring specific Insights-related entries:

- **Name**: Enter the name of your Maintenance Plan.
- **Job Type**: If not already populated, select **Run** from the drop-down menu list.
- **Inventory**: Select the Insights Inventory you previously created.
- **Project**: Select the Insights project you previously created.
- **Playbook**: Select a playbook associated with the Maintenance Plan you want to run from the drop-down menu list.
- **Credential**: Enter the credential to use for this project or click the 🔍 button and select it from the pop-up window. The credential does not have to be an Insights credential.
- **Verbosity**: Keep the default setting, or select the desired verbosity from the drop-down menu list.



Templates

Create New Job Template

Name * Maintenance plan job

Description

Job Type * □ Prompt on launch Run

Inventory * □ Prompt on launch Insights inventory

Project * Insights Project

Execution Environment

Playbook * file.yml

Credentials □ Prompt on launch SSH: Demo Credential

Labels

Variables YAML JSON □ Prompt on launch

```
---
```

Forks 0

Limit □ Prompt on launch

Verbosity □ Prompt on launch 0 (Normal)

Job Slicing 1

Timeout 0

Show Changes □ Prompt on launch Off

Instance Groups

Job Tags □ Prompt on launch

Skip Tags □ Prompt on launch

Options

□ Privilege Escalation □ Provisioning Callbacks □ Enable Webhook □ Concurrent Jobs □ Enable Fact Storage

3. Click **Save** when done.

4. Click the 🚀 icon to launch the job template.

Once complete, the job results display in the Job Details page.

CHAPTER

THIRTYONE

BEST PRACTICES

31.1 Use Source Control

While automation controller supports playbooks stored directly on the server, best practice is to store your playbooks, roles, and any associated details in source control. This way you have an audit trail describing when and why you changed the rules that are automating your infrastructure. Plus, it allows for easy sharing of playbooks with other parts of your infrastructure or team.

31.2 Ansible file and directory structure

Please review the Ansible Tips and Tricks from the Ansible documentation. If creating a common set of roles to use across projects, these should be accessed via source control submodules, or a common location such as `/opt`. Projects should not expect to import roles or content from other projects.

Note: Playbooks should not use the `vars_prompt` feature, as automation controller does not interactively allow for `vars_prompt` questions. If you must use `vars_prompt`, refer to and make use of the *Surveys* functionality.

Note: Playbooks should not use the `pause` feature of Ansible without a timeout, as automation controller does not allow for interactively cancelling a pause. If you must use `pause`, ensure that you set a timeout.

Jobs run use the playbook directory as the current working directory, although jobs should be coded to use the `playbook_dir` variable rather than relying on this.

31.3 Use Dynamic Inventory Sources

If you have an external source of truth for your infrastructure, whether it is a cloud provider or a local CMDB, it is best to define an inventory sync process and use the support for dynamic inventory (including cloud inventory sources). This ensures your inventory is always up to date.

Note: Edits and additions to Inventory host variables persist beyond an inventory sync as long as `--overwrite_vars` is **not** set.

31.4 Variable Management for Inventory

Keeping variable data along with the hosts and groups definitions (see the inventory editor) is encouraged, rather than using `group_vars/` and `host_vars/`. If you use dynamic inventory sources, the controller can sync such variables with the database as long as the **Overwrite Variables** option is not set.

31.5 Autoscaling

Using the "callback" feature to allow newly booting instances to request configuration is very useful for auto-scaling scenarios or provisioning integration.

31.6 Larger Host Counts

Consider setting "forks" on a job template to larger values to increase parallelism of execution runs. For more information on tuning Ansible, see the Ansible blog.

31.7 Continuous integration / Continuous Deployment

For a Continuous Integration system, such as Jenkins, to spawn a job, it should make a curl request to a job template. The credentials to the job template should not require prompting for any particular passwords. Refer to the CLI documentation for configuration and usage instructions.

31.8 LDAP authentication performance tips

When an LDAP user authenticates, by default, all user-related attributes will be updated in the database on each log in. In some environments, this operation can be skipped due to performance issues. To avoid it, you can disable the option *AUTH_LDAP_ALWAYS_UPDATE_USER*. Refer to the Knowledge Base Article 5823061 for its configuration and usage instructions. Please note that new users will still be created and get their attributes pushed to the database on their first login.

Warning: With this option set to False, no changes to LDAP user's attributes will be updated. Attributes will only be updated the first time the user is created.

CHAPTER

THIRTYTWO

SECURITY

The following sections will help you gain an understanding of how automation controller handles and lets you control file system security.

All playbooks are executed via the awx file system user. For running jobs, automation controller offers job isolation via the use of Linux containers. This projection ensures jobs can only access playbooks, roles, and data from the Project directory for that job template.

For credential security, users may choose to upload locked SSH keys and set the unlock password to "ask". You can also choose to have the system prompt them for SSH credentials or sudo passwords rather than having the system store them in the database.

32.1 Playbook Access and Information Sharing

Automation controller's use of automation execution environments and Linux containers prevents playbooks from reading files outside of their project directory.

By default, the only data exposed to the ansible-playbook process inside the container is the current project being used.

You can customize this in the Job Settings and expose additional directories from the host into the container. Refer the next section, *Isolation functionality and variables* for more information.

32.1.1 Isolation functionality and variables

Automation controller uses container technology to isolate jobs from each other. By default, only the current project is exposed to the container running a job template.

You may find that you need to customize your playbook runs to expose additional directories. To fine tune your usage of job isolation, there are certain variables that can be set.

By default, automation controller will use the system's tmp directory (/tmp by default) as its staging area. This can be changed in the **Job Execution Path** field of the Jobs settings screen, or in the REST API at /api/v2/settings/jobs:

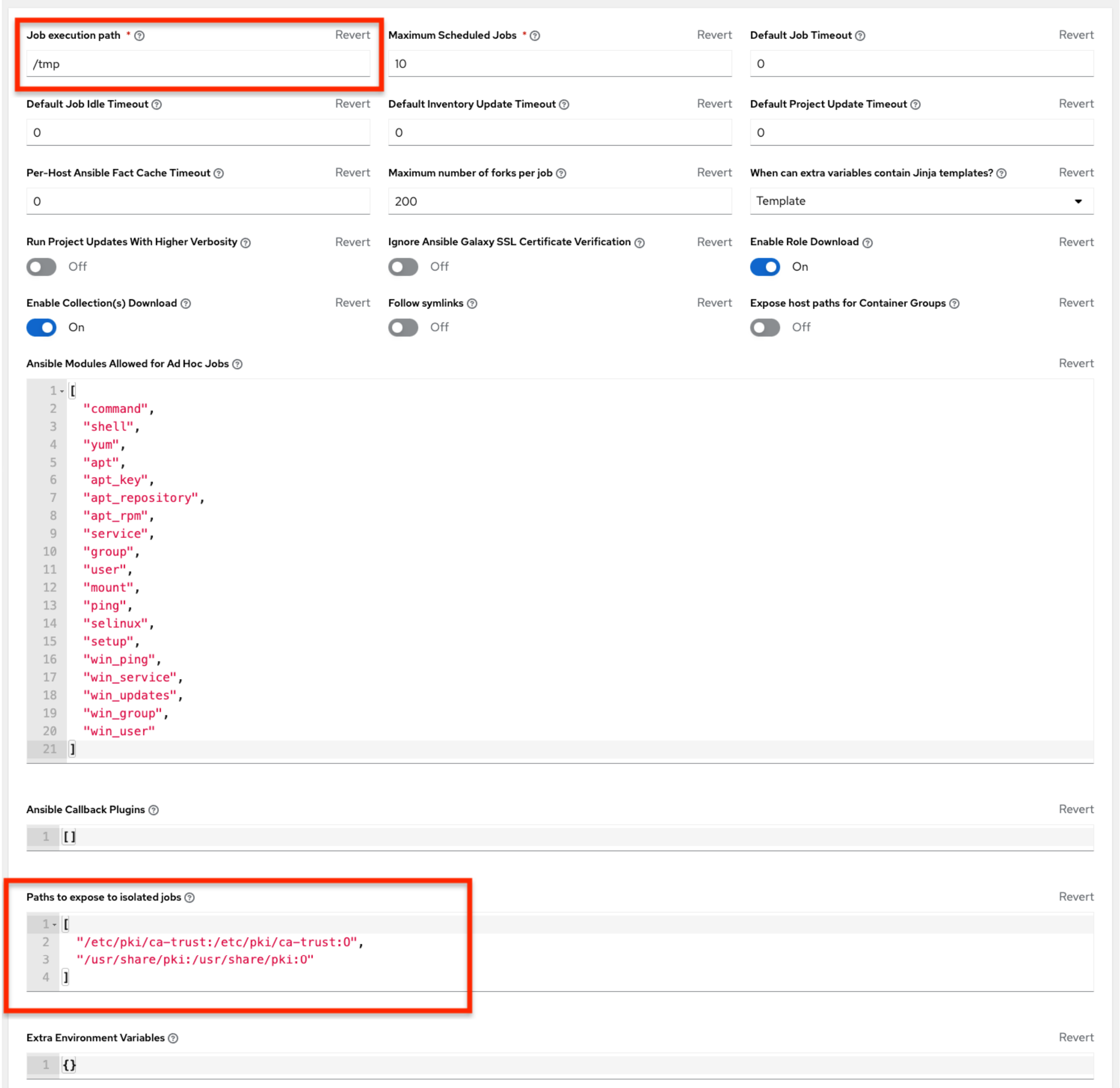
```
AWX_ISOLATION_BASE_PATH = "/opt/tmp"
```

If there are any additional directories that should specifically be exposed from the host to the container that playbooks run in, you can specify those in the **Paths to Expose to Isolated Jobs** field of the Jobs setting scren, or in the REST API at /api/v2/settings/jobs:

```
AWX_ISOLATION_SHOW_PATHS = ['/list/of/', '/paths']
```

Note: The primary file you may want to add to `AWX_ISOLATION_SHOW_PATHS` is `/var/lib/awx/.ssh`, if your playbooks need to use keys or settings defined there.

The above fields can be found in the Jobs Settings window:



Job execution path * ⓘ Revert
/tmp

Maximum Scheduled Jobs * ⓘ Revert
10

Default Job Timeout ⓘ Revert
0

Default Job Idle Timeout ⓘ Revert
0

Default Inventory Update Timeout ⓘ Revert
0

Default Project Update Timeout ⓘ Revert
0

Per-Host Ansible Fact Cache Timeout ⓘ Revert
0

Maximum number of forks per job ⓘ Revert
200

When can extra variables contain Jinja templates? ⓘ Revert
Template ▾

Run Project Updates With Higher Verbosity ⓘ Revert
Off

Ignore Ansible Galaxy SSL Certificate Verification ⓘ Revert
Off

Enable Role Download ⓘ Revert
On

Enable Collection(s) Download ⓘ Revert
On

Follow symlinks ⓘ Revert
Off

Expose host paths for Container Groups ⓘ Revert
Off

Ansible Modules Allowed for Ad Hoc Jobs ⓘ Revert

```
[
  "command",
  "shell",
  "yum",
  "apt",
  "apt_key",
  "apt_repository",
  "apt_rpm",
  "service",
  "group",
  "user",
  "mount",
  "ping",
  "selinux",
  "setup",
  "win_ping",
  "win_service",
  "win_updates",
  "win_group",
  "win_user"
]
```

Ansible Callback Plugins ⓘ Revert

```
[]
```

Paths to expose to isolated jobs ⓘ Revert

```
[
 "/etc/pki/ca-trust:/etc/pki/ca-trust:O",
 "/usr/share/pki:/usr/share/pki:O"
]
```

Extra Environment Variables ⓘ Revert

```
{}
```

32.2 Role-Based Access Controls

Role-Based Access Controls (RBAC) are built into automation controller and allow administrators to delegate access to server inventories, organizations, and more. Administrators can also centralize the management of various credentials, allowing end users to leverage a needed secret without ever exposing that secret to the end user. RBAC controls allow the controller to help you increase security and streamline management.

RBACs are easiest to think of in terms of Roles which define precisely who or what can see, change, or delete an "object" for which a specific capability is being set. RBAC is the practice of granting roles to users or teams.

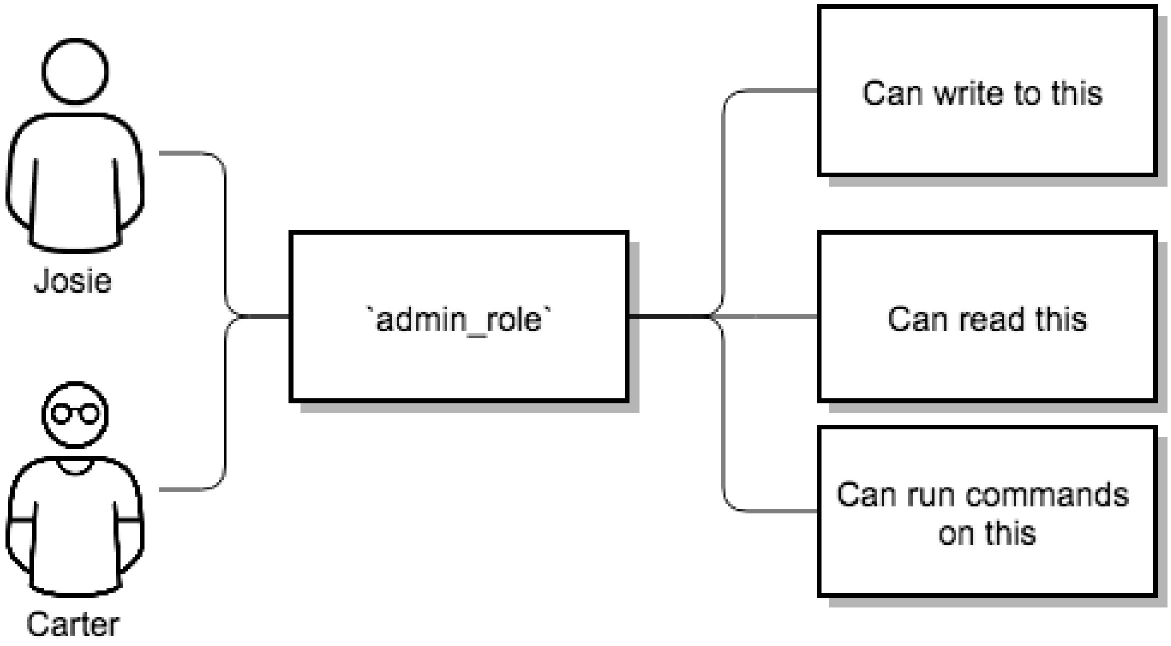
There are a few main concepts that you should become familiar with regarding automation controller's RBAC design–roles, resources, and users. Users can be members of a role, which gives them certain access to any resources associated with that role, or any resources associated with "descendant" roles.

A role is essentially a collection of capabilities. Users are granted access to these capabilities and the controller's resources through the roles to which they are assigned or through roles inherited through the role hierarchy.

Roles associate a group of capabilities with a group of users. All capabilities are derived from membership within a role. Users receive capabilities only through the roles to which they are assigned or through roles they inherit through the role hierarchy. All members of a role have all capabilities granted to that role. Within an organization, roles are relatively stable, while users and capabilities are both numerous and may change rapidly. Users can have many roles.

32.2.1 Role Hierarchy and Access Inheritance

Imagine that you have an organization named "SomeCompany" and want to allow two people, "Josie" and "Carter", access to manage all the settings associated with that organization. You should make both people members of the organization's `admin_role`.

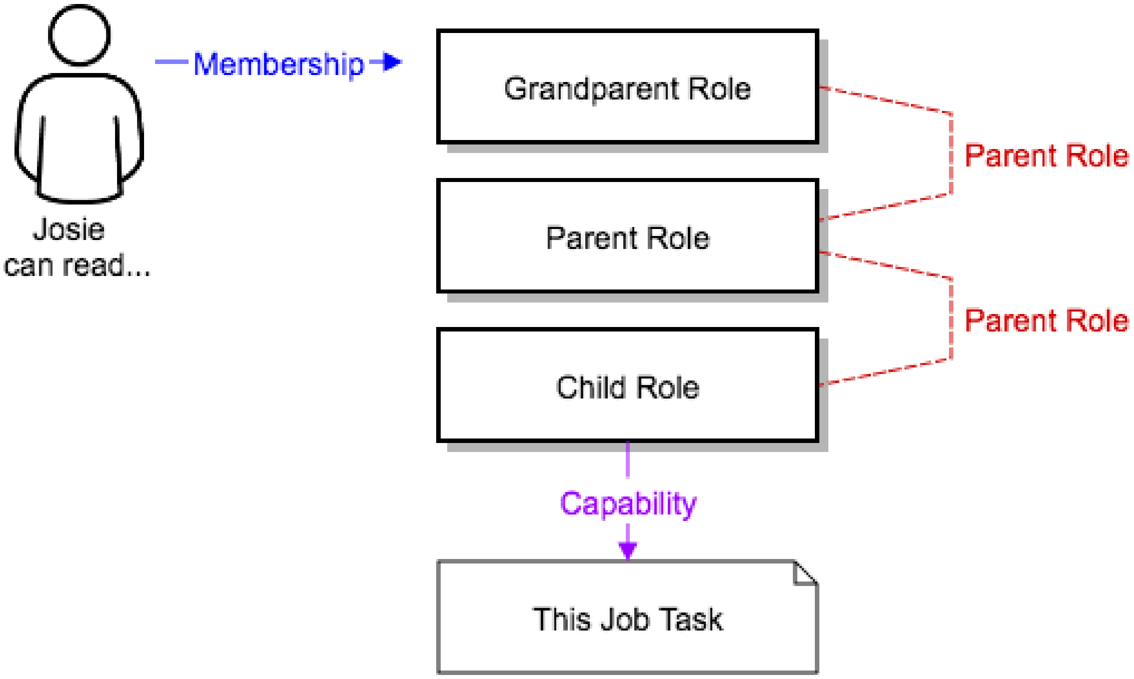


Often, you will have many Roles in a system and you will want some roles to include all of the capabilities of other roles. For example, you may want a System Administrator to have access to everything that an Organization Administrator has access to, who has everything that a Project Administrator has access to, and so on.

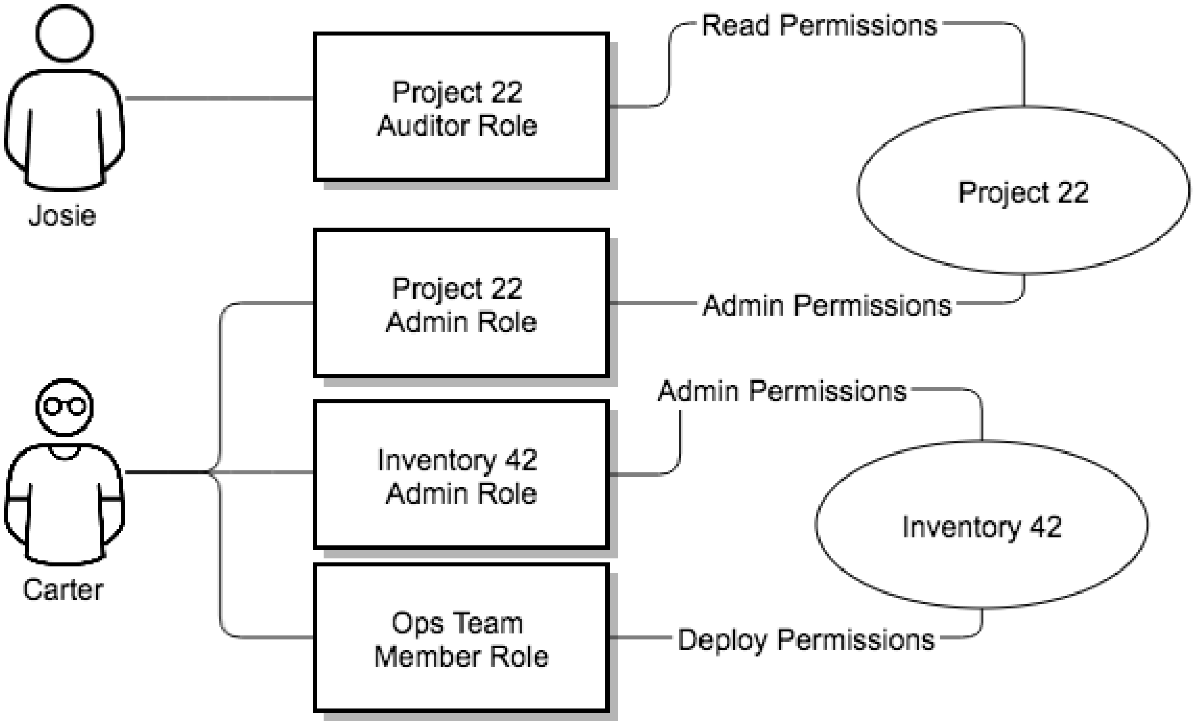
This concept is referred to as the 'Role Hierarchy':

- Parent roles get all capabilities bestowed on any child roles
- Members of roles automatically get all capabilities for the role they are a member of, as well as any child roles.

The Role Hierarchy is represented by allowing Roles to have "Parent Roles". Any capability that a Role has is implicitly granted to any parent roles (or parents of those parents, and so on).



Often, you will have many Roles in a system and you will want some roles to include all of the capabilities of other roles. For example, you may want a System Administrator to have access to everything that an Organization Administrator has access to, who has everything that a Project Administrator has access to, and so on. We refer to this concept as the 'Role Hierarchy' and it is represented by allowing Roles to have "Parent Roles". Any capability that a Role has is implicitly granted to any parent roles (or parents of those parents, and so on). Of course Roles can have more than one parent, and capabilities are implicitly granted to all parents.



RBAC controls also give you the capability to explicitly permit User and Teams of Users to run playbooks against certain sets of hosts. Users and teams are restricted to just the sets of playbooks and hosts to which they are granted capabilities. And, with automation controller, you can create or import as many Users and Teams as you require–create users and teams manually or import them from LDAP or Active Directory.

RBACs are easiest to think of in terms of who or what can see, change, or delete an “object” for which a specific capability is being determined.

32.2.2 Applying RBAC

The following sections cover how to apply automation controller’s RBAC system in your environment.

Editing Users

When editing a user, a automation controller system administrator may specify the user as being either a *System Administrator* (also referred to as the Superuser) or a *System Auditor*.

- System administrators implicitly inherit all capabilities for all objects (read/write/execute) within the automation controller environment.
- System Auditors implicitly inherit the read-only capability for all objects within the automation controller environment.

Editing Organizations

When editing an organization, system administrators may specify the following roles:

- One or more users as organization administrators
- One or more users as organization auditors
- And one or more users (or teams) as organization members

Users/teams that are members of an organization can view their organization administrator.

Users who are organization administrators implicitly inherit all capabilities for all objects within that automation controller organization.

Users who are organization auditors implicitly inherit the read-only capability for all objects within that automation controller organization.

Editing Projects in an Organization

When editing a project in an organization for which they are the administrator, system administrators and organization administrators may specify:

- One or more users/teams that are project administrators
- One or more users/teams that are project members
- And one or more users/teams that may update the project from SCM, from among the users/teams that are members of that organization.

Users who are members of a project can view their project administrators.

Project administrators implicitly inherit the capability to update the project from SCM.

Administrators can also specify one or more users/teams (from those that are members of that project) that can use that project in a job template.

Creating Inventories and Credentials within an Organization

All access that is granted to use, read, or write credentials is now handled through roles. You no longer set the “team” or “user” for a credential. Instead, you use automation controller’s RBAC system to grant ownership, auditor, or usage roles.

System administrators and organization administrators may create inventories and credentials within organizations under their administrative capabilities.

Whether editing an inventory or a credential, System administrators and organization administrators may specify one or more users/teams (from those that are members of that organization) to be granted the usage capability for that inventory or credential.

System administrators and organization administrators may specify one or more users/teams (from those that are members of that organization) that have the capabilities to update (dynamic or manually) an inventory. Administrators can also execute ad hoc commands for an inventory.

Editing Job Templates

System administrators, organization administrators, and project administrators, within a project under their administrative capabilities, may create and modify new job templates for that project.

When editing a job template, administrators (automation controller, organization, and project) can select among the inventory and credentials in the organization for which they have usage capabilities or they may leave those fields blank so that they will be selected at runtime.

Additionally, they may specify one or more users/teams (from those that are members of that project) that have execution capabilities for that job template. The execution capability is valid regardless of any explicit capabilities the user/team may have been granted against the inventory or credential specified in the job template.

User View

A user can:

- See any organization or project for which they are a member
- Create their own credential objects which only belong to them
- See and execute any job template for which they have been granted execution capabilities

If a job template a user has been granted execution capabilities on does not specify an inventory or credential, the user will be prompted at run-time to select among the inventory and credentials in the organization they own or have been granted usage capabilities.

Users that are job template administrators can make changes to job templates; however, to change to the inventory, project, playbook, or credentials used in the job template, the user must also have the “Use” role for the project and inventory currently being used or being set.

32.2.3 Roles

As stated earlier in this documentation, all access that is granted to use, read, or write credentials is now handled through roles, and roles are defined for a resource.

Built-in roles

The following table lists the RBAC system roles and a brief description of the how that role is defined with regard to privileges in automation controller.

System Role	What it can do
System Administrator - System wide singleton	Manages all aspects of the system
System Auditor - System wide singleton	Views all aspects of the system
Ad Hoc Role - Inventory	Runs ad hoc commands on an Inventory
Admin Role - Organizations, Teams, Inventory, Projects, Job Templates	Manages all aspects of a defined Organization, Team, Inventory, Project, or Job Template
Auditor Role - All	Views all aspects of a defined Organization, Project, Inventory, or Job Template
Execute Role - Job Templates	Runs assigned Job Template
Member Role - Organization, Team	Manages all of the settings associated with that Organization or Team
Read Role - Organizations, Teams, Inventory, Projects, Job Templates	Views all aspects of a defined Organization, Team, Inventory, Project, or Job Template
Update Role - Project	Updates the Project from the configured source control management system
Update Role - Inventory	Updates the Inventory using the cloud source update system
Owner Role - Credential	Owns and manages all aspects of this Credential
Use Role - Credential, Inventory, Project	Uses the Credential, Inventory, or Project in a Job Template

A Singleton Role is a special role that grants system-wide permissions. automation controller currently provides two built-in Singleton Roles but the ability to create or customize a Singleton Role is not supported at this time.

Common Team Roles - "Personas"

Automation controller support personnel typically works on ensuring that the controller is available and manages it a way to balance supportability and ease-of-use for users. Often, automation controller support will assign "Organization Owner/Admin" to users in order to allow them to create a new Organization and add members from their team the respective access needed. This minimizes supporting individuals and focuses more on maintaining uptime of the service and assisting users who are using automation controller.

Below are some common roles managed by the automation controller Organization:

System Role (for Organizations)	Common User Roles	Description
Owner	Team Lead - Technical Lead	This user has the ability to control access for other users in their organization. They can add/remove and grant users specific access to projects, inventories, and job templates. This user also has the ability to create/remove/modify any aspect of an organization's projects, templates, inventories, teams, and credentials.
Auditor	Security Engineer - Project Manager	This account can view all aspects of the organization in read-only mode. This may be good for a user who checks in and maintains compliance. This might also be a good role for a service account who manages or ships job data from automation controller to some other data collector.
Member - Team	All other users	These users by default as an organization member do not receive any access to any aspect of the organization. In order to grant them access the respective organization owner needs to add them to their respective team and grant them Admin, Execute, Use, Update, Ad-hoc permissions to each component of the organization's projects, inventories, and job templates.
Member - Team "Owner"	Power users - Lead Developer	Organization Owners can provide "admin" through the team interface, over any component of their organization including projects, inventories, and job templates. These users are able to modify and utilize the respective component given access.

32.3 Function of roles: editing and creating

A new organization "resource roles" functionality was introduced in automation controller 3.3 that are specific to a certain resource type - such as workflows. Being a member of such a role usually provides two types of permissions, in the case of workflows, where a user is given a "workflow admin role" for the organization "Default":

- this user can create new workflows in the organization "Default"
- user can edit all workflows in the "Default" organization

One exception is job templates, where having the role is irrelevant of creation permission (more details on its own section).

32.3.1 Independence of resource roles and organization membership roles

Resource-specific organization roles are independent of the organization roles of admin and member. Having the "workflow admin role" for the "Default" organization will not allow a user to view all users in the organization, but having a "member" role in the "Default" organization will. The two types of roles are delegated independently of each other.

Necessary permissions to edit job templates

Users can edit fields not impacting job runs (non-sensitive fields) with a Job Template admin role alone. However, to edit fields that impact job runs in a job template, a user needs the following:

- **admin** role to the job template
- **use** role to related project
- **use** role to related inventory

An "organization job template admin" role was introduced, but having this role isn't sufficient by itself to edit a job template within the organization if the user does not have use role to the project / inventory a job template uses.

In order to delegate *full* job template control (within an organization) to a user or team, you will need grant the team or user all 3 organization-level roles:

- job template admin
- project admin
- inventory admin

This will ensure that the user (or all users who are members of the team with these roles) have full access to modify job templates in the organization. If a job template uses an inventory or project from another organization, the user with these organization roles may still not have permission to modify that job template. For clarity of managing permissions, it is best-practice to not mix projects / inventories from different organizations.

RBAC permissions

Each role should have a content object, for instance, the org admin role has a content object of the org. To delegate a role, you need admin permission to the content object, with some exceptions that would result in you being able to reset a user's password.

Parent is the organization.

Allow is what this new permission will explicitly allow.

Scope is the parent resource that this new role will be created on. Example: `Organization.project_create_role`.

An assumption is being made that the creator of the resource should be given the admin role for that resource. If there are any instances where resource creation does not also imply resource administration, they will be explicitly called out.

Here are the rules associated with each admin type:

Project Admin

- Allow: Create, read, update, delete any project
- Scope: Organization
- User Interface: *Project Add Screen - Organizations*

Inventory Admin

- Parent: Org admin
- Allow: Create, read, update, delete any inventory
- Scope: Organization
- User Interface: *Inventory Add Screen - Organizations*

Note: As it is with the **Use** role, if you give a user Project Admin and Inventory Admin, it allows them to create Job Templates (not workflows) for your organization.

Credential Admin

- Parent: Org admin
- Allow: Create, read, update, delete shared credentials
- Scope: Organization
- User Interface: *Credential Add Screen - Organizations*

Notification Admin

- Parent: Org admin
- Allow: Assignment of notifications
- Scope: Organization

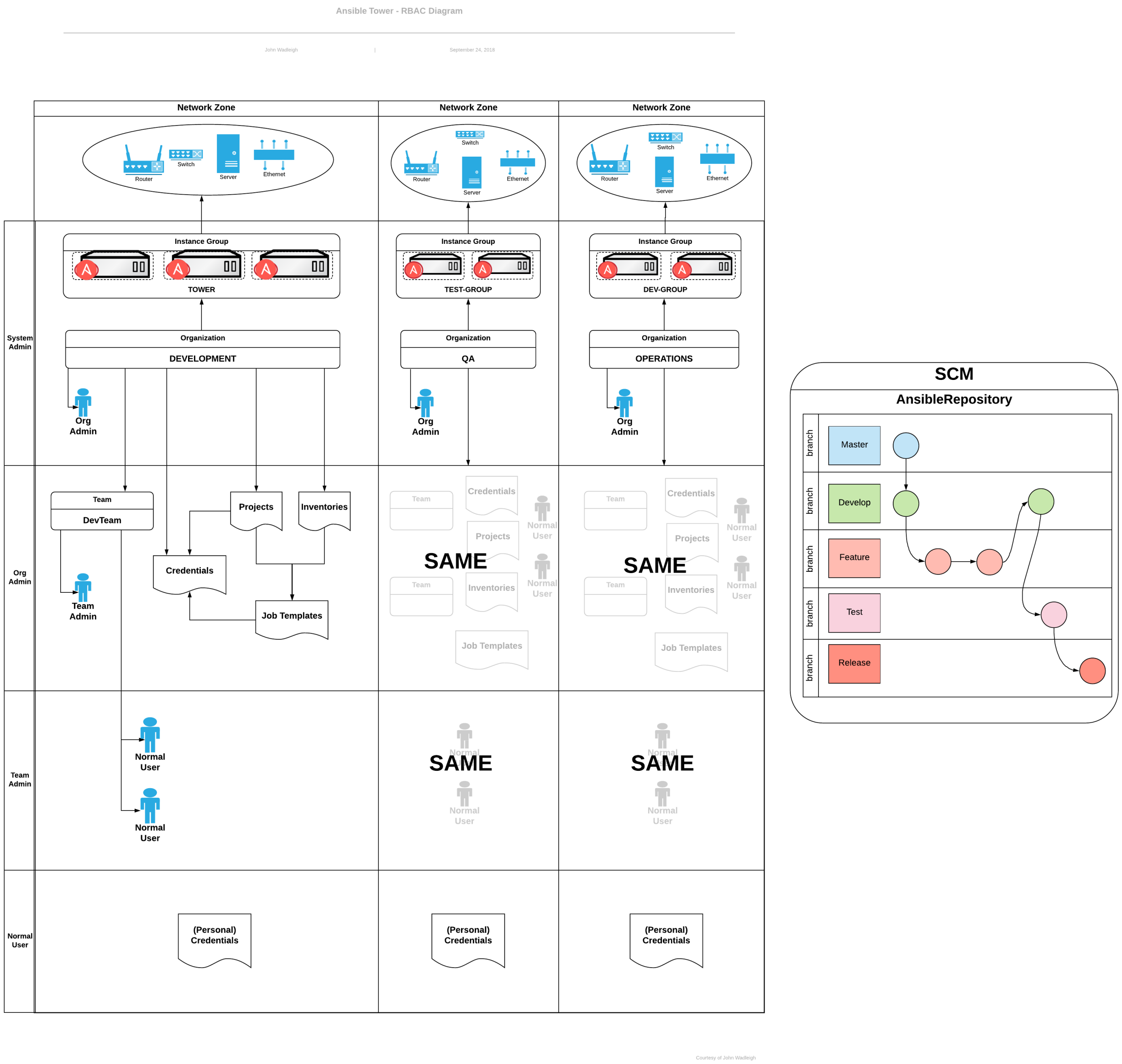
Workflow Admin

- Parent: Org admin
- Allow: Create a workflow
- Scope: Organization

Org Execute

- Parent: Org admin
- Allow: Executing JTs and WFJTs
- Scope: Organization

The following is a sample scenario showing an organization with its roles and which resource(s) each have access to:



CHAPTER THIRTYTHREE

GLOSSARY

Ad Hoc Refers to running Ansible to perform some quick command, using /usr/bin/ansible, rather than the orchestration language, which is /usr/bin/ansible-playbook. An example of an ad hoc command might be rebooting 50 machines in your infrastructure. Anything you can do ad hoc can be accomplished by writing a Playbook, and Playbooks can also glue lots of other operations together.

Callback Plugin Refers to some user-written code that can intercept results from Ansible and do something with them. Some supplied examples in the GitHub project perform custom logging, send email, or even play sound effects.

Control Groups Also known as '*cgroups*', a control group is a feature in the Linux kernel that allows resources to be grouped and allocated to run certain processes. In addition to assigning resources to processes, cgroups can also report actual resource usage by all processes running inside of the cgroup.

Check Mode Refers to running Ansible with the `--check` option, which does not make any changes on the remote systems, but only outputs the changes that might occur if the command ran without this flag. This is analogous to so-called "dry run" modes in other systems, though the user should be warned that this does not take into account unexpected command failures or cascade effects (which is true of similar modes in other systems). Use this to get an idea of what might happen, but it is not a substitute for a good staging environment.

Container Groups Container Groups are a type of Instance Group that specify a configuration for provisioning a pod in a Kubernetes or OpenShift cluster where a job is run. These pods are provisioned on-demand and exist only for the duration of the playbook run.

Credentials Authentication details that may be utilized by the controller to launch jobs against machines, to synchronize with inventory sources, and to import project content from a version control system.

Credential Plugin Python code that contains definitions for an external credential type, its metadata fields, and the code needed for interacting with a secret management system.

Distributed Job A job that consists of a job template, an inventory, and slice size. When executed, a distributed job slices each inventory into a number of "slice size" chunks, which are then used to run smaller job slices.

External Credential Type A managed credential type for automation controller used for authenticating with a secret management system.

Facts Facts are simply things that are discovered about remote nodes. While they can be used in playbooks and templates just like variables, facts are things that are inferred, rather than set. Facts are automatically discovered when running plays by executing the internal setup module on the remote nodes. You never have to call the setup module explicitly, it just runs, but it can be disabled to save time if it is not needed. For the convenience of users who are switching from other configuration management systems, the fact module also pulls in facts from the 'ohai' and 'facter' tools if they are installed, which are fact libraries from Chef and Puppet, respectively.

Forks Ansible and automation controller talk to remote nodes in parallel and the level of parallelism can be set serveral ways–during the creation or editing of a Job Template, by passing `--forks`, or by editing the default in a configuration file. The default is a very conservative 5 forks, though if you have a lot of RAM, you can easily set this to a value like 50 for increased parallelism.

Group A set of hosts in Ansible that can be addressed as a set, of which many may exist within a single Inventory.

Group Vars The `group_vars/` files are files that live in a directory alongside an inventory file, with an optional filename named after each group. This is a convenient place to put variables that will be provided to a given group, especially complex data structures, so that these variables do not have to be embedded in the inventory file or playbook.

Handlers Handlers are just like regular tasks in an Ansible playbook (see Tasks), but are only run if the Task contains a "notify" directive and also indicates that it changed something. For example, if a config file is changed then the task referencing the config file templating operation may notify a service restart handler. This means services can be bounced only if they need to be restarted. Handlers can be used for things other than service restarts, but service restarts are the most common usage.

Host A system managed by automation controller, which may include a physical, virtual, cloud-based server, or other device. Typically an operating system instance. Hosts are contained in Inventory. Sometimes referred to as a "node".

Host Specifier Each Play in Ansible maps a series of tasks (which define the role, purpose, or orders of a system) to a set of systems. This "hosts:" directive in each play is often called the hosts specifier. It may select one system, many systems, one or more groups, or even some hosts that are in one group and explicitly not in another.

Instance Group A group that contains instances for use in a clustered environment. An instance group provides the ability to group instances based on policy.

Inventory A collection of hosts against which Jobs may be launched.

Inventory Script A very simple program (or a complicated one) that looks up hosts, group membership for hosts, and variable information from an external resource–whether that be a SQL database, a CMDB solution, or something like LDAP. This concept was adapted from Puppet (where it is called an "External Nodes Classifier") and works more or less exactly the same way.

Inventory Source Information about a cloud or other script that should be merged into the current inventory group, resulting in the automatic population of Groups, Hosts, and variables about those groups and hosts.

Job One of many background tasks launched by the controller, this is usually the instantiation of a Job Template; the launch of an Ansible playbook. Other types of jobs include inventory imports, project synchronizations from source control, or administrative cleanup actions.

Job Detail The history of running a particular job, including its output and success/failure status.

Job Slice See *Distributed Job*.

Job Template The combination of an Ansible playbook and the set of parameters required to launch it.

JSON Ansible and automation controller use JSON for return data from remote modules. This allows modules to be written in any language, not just Python.

Mesh Describes a network comprising of nodes. Communication between nodes is established at the transport layer by protocols such as TCP, UDP or Unix sockets. See also, *node*.

Metadata Information for locating a secret in the external system once authenticated. The uses provides this information when linking an external credential to a target credential field.

Node A node corresponds to entries in the instance database model, or the `/api/v2/instances/` endpoint, and is a machine participating in the cluster / mesh. The unified jobs API reports `controller_node` and `execution_node` fields. The execution node is where the job runs, and the controller node interfaces between the job and server functions.

Node Type	Description
Control	Nodes that run persistent Ansible Automation Platform services, and delegate jobs to hybrid and execution nodes
Hybrid	Nodes that run persistent Ansible Automation Platform services and execute jobs
Hop	Used for relaying across the mesh only
Execution	Nodes that run jobs delivered from control nodes (jobs submitted from the user's Ansible automation)

Notification Template An instance of a notification type (Email, Slack, Webhook, etc.) with a name, description, and a defined configuration.

Notification A manifestation of the notification template; for example, when a job fails a notification is sent using the configuration defined by the notification template.

Notify The act of a task registering a change event and informing a handler task that another action needs to be run at the end of the play. If a handler is notified by multiple tasks, it will still be run only once. Handlers are run in the order they are listed, not in the order that they are notified.

Organization A logical collection of Users, Teams, Projects, and Inventories. The highest level in the automation controller object hierarchy is the Organization.

Organization Administrator An automation controller user with the rights to modify the Organization's membership and settings, including making new users and projects within that organization. An organization admin can also grant permissions to other users within the organization.

Permissions The set of privileges assigned to Users and Teams that provide the ability to read, modify, and administer Projects, Inventories, and other automation controller objects.

Plays A playbook is a list of plays. A play is minimally a mapping between a set of hosts selected by a host specifier (usually chosen by groups, but sometimes by hostname globs) and the tasks which run on those hosts to define the role that those systems will perform. There can be one or many plays in a playbook.

Playbook An Ansible playbook. Refer to http://docs.ansible.com/ for more information.

Policy Policies dictate how instance groups behave and how jobs are executed.

Project A logical collection of Ansible playbooks, represented in automation controller.

Roles Roles are units of organization in Ansible and automation controller. Assigning a role to a group of hosts (or a set of groups, or host patterns, etc.) implies that they should implement a specific behavior. A role may include applying certain variable values, certain tasks, and certain handlers–or just one or more of these things. Because of the file structure associated with a role, roles become redistributable units that allow you to share behavior among playbooks–or even with other users.

Secret Management System A server or service for securely storing and controlling access to tokens, passwords, certificates, encryption keys, and other sensitive data.

Schedule The calendar of dates and times for which a job should run automatically.

Sliced Job See *Distributed Job*.

Source Credential An external credential that is linked to the field of a target credential.

Sudo Ansible does not require root logins and, since it is daemonless, does not require root level daemons (which can be a security concern in sensitive environments). Ansible can log in and perform many operations wrapped in a `sudo` command, and can work with both password-less and password-based sudo. Some operations that do not normally work with `sudo` (like `scp` file transfer) can be achieved with Ansible's *copy*, *template*, and *fetch* modules while running in `sudo` mode.

Superuser An admin of the automation controller server who has permission to edit any object in the system, whether associated to any organization. Superusers can create organizations and other superusers.

Survey Questions asked by a job template at job launch time, configurable on the job template.

Target Credential A non-external credential with an input field that is linked to an external credential.

Team A sub-division of an Organization with associated Users, Projects, Credentials, and Permissions. Teams provide a means to implement role-based access control schemes and delegate responsibilities across Organizations.

User An automation controller operator with associated permissions and credentials.

Webhook Webhooks allow communication and information sharing between apps. They are used to respond to commits pushed to SCMs and launch job templates or workflow templates.

Workflow Job Template A set consisting of any combination of job templates, project syncs, and inventory syncs, linked together in order to execute them as a single unit.

YAML Ansible and automation controller use YAML to define playbook configuration languages and also variable files. YAML has a minimum of syntax, is very clean, and is easy for people to skim. It is a good data format for configuration files and humans, but is also machine readable. YAML is fairly popular in the dynamic language community and the format has libraries available for serialization in many languages (Python, Perl, Ruby, etc.).

CHAPTER THIRTYFOUR

INDEX

- genindex

CHAPTER

THIRTYFIVE

COPYRIGHT © RED HAT, INC.

Third Party Rights

INDEX