
Ansible Tower User Guide

Release Ansible Tower 3.8.0

Red Hat, Inc.

Feb 11, 2023

CONTENTS

1	Overview	2
1.1	Real-time Playbook Output and Exploration	2
1.2	“Push Button” Automation	2
1.3	Enhanced and Simplified Role-Based Access Control and Auditing	2
1.4	Cloud & Autoscaling Flexibility	3
1.5	The Ideal RESTful API	3
1.6	Backup and Restore	3
1.7	Ansible Galaxy Integration	3
1.8	Inventory Support for OpenStack	3
1.9	Remote Command Execution	4
1.10	System Tracking	4
1.11	Integrated Notifications	4
1.12	Satellite Integration	4
1.13	Run-time Job Customization	4
1.14	Red Hat Insights Integration	5
1.15	Enhanced Tower User Interface	5
1.16	Custom Virtual Environments	5
1.17	Authentication Enhancements	5
1.18	Cluster Management	5
1.19	Container Platform Support	5
1.20	Workflow Enhancements	5
1.21	Job Distribution	6
1.22	Support for deployment in a FIPS-enabled environment	6
1.23	Limit the number of hosts per organization	6
1.24	Inventory Plugins	6
1.25	Secret Management System	7
1.26	Automation Hub Integration	7
2	Tower Licensing, Updates, and Support	8
2.1	Support	8
2.2	Trial / Evaluation	8
2.3	Subscription Types	8
2.4	Node Counting in Licenses	9
2.5	Attaching Subscriptions	9
2.6	Tower Component Licenses	10
3	Logging In	11
4	Import a Subscription	12
4.1	Obtaining a subscriptions manifest	16

4.2	Adding a Tower subscription manually	19
5	The Tower User Interface	20
5.1	Activity Streams	20
5.2	Views	22
5.3	Resources and Access	26
5.4	Tower Administration Menu	26
6	Search	28
6.1	Searching Tips	28
6.2	Sort	30
7	Organizations	32
7.1	Creating a New Organization	33
7.2	Work with Users	35
7.3	Work with Permissions	36
7.4	Work with Notifications	42
7.5	Organization Summary	43
8	Users	44
8.1	Create a User	44
8.2	User Types - Quick View	46
8.3	Users - Organizations	47
8.4	Users - Teams	47
8.5	Users - Permissions	48
8.6	Users - Tokens	51
9	Teams	53
9.1	Create a Team	53
10	Credentials	59
10.1	Understanding How Credentials Work	59
10.2	Getting Started with Credentials	59
10.3	Add a New Credential	61
10.4	Credential Types	62
11	Custom Credential Types	76
11.1	Content sourcing from collections	76
11.2	Backwards-Compatible API Considerations	77
11.3	Getting Started with Credential Types	77
11.4	Create a New Credential Type	78
12	Secret Management System	83
12.1	Configure and link secret lookups	83
13	Applications	90
13.1	Getting Started with Applications	90
13.2	Create a new application	91
14	Projects	95
14.1	Add a new project	97
14.2	Updating projects from source control	101
14.3	Work with Permissions	102
14.4	Work with Notifications	107
14.5	Work with Job Templates	108
14.6	Work with Schedules	108

14.7	Ansible Galaxy Support	109
14.8	Collections Support	112
15	Inventories	117
15.1	Smart Inventories	119
15.2	Inventory Plugins	120
15.3	Add a new inventory	121
15.4	Running Ad Hoc Commands	145
16	Job Templates	148
16.1	Create a Job Template	149
16.2	Add Permissions	155
16.3	Work with Notifications	160
16.4	View Completed Jobs	161
16.5	Scheduling	161
16.6	Surveys	162
16.7	Launch a Job Template	165
16.8	Copy a Job Template	168
16.9	Scan Job Templates	168
16.10	Fact Caching	172
16.11	Utilizing Cloud Credentials	173
16.12	Provisioning Callbacks	176
16.13	Extra Variables	178
17	Job Slicing	180
17.1	Job slice considerations	180
17.2	Job slice execution behavior	181
17.3	Search job slices	182
18	Workflows	183
18.1	Workflow scenarios and considerations	183
18.2	Extra Variables	186
18.3	Workflow States	187
18.4	Role-Based Access Controls	187
19	Workflow Job Templates	188
19.1	Create a Workflow Template	189
19.2	Work with Permissions	192
19.3	Work with Notifications	193
19.4	View Completed Jobs	194
19.5	Work with Schedules	196
19.6	Surveys	196
19.7	Workflow Visualizer	198
19.8	Launch a Workflow Template	212
19.9	Copy a Workflow Template	213
19.10	Extra Variables	214
20	Instance Groups	216
20.1	Create an instance group	216
21	Jobs	221
21.1	Job Details - Inventory Sync	223
21.2	Job Details - SCM	225
21.3	Job Details - Playbook Run	226
21.4	Ansible Tower Capacity Determination and Job Impact	231

21.5	Job branch overriding	234
22	Working with Webhooks	239
22.1	GitHub webhook setup	239
22.2	GitLab webhook setup	246
22.3	Payload output	252
23	Notifications	254
23.1	Notification Hierarchy	254
23.2	Workflow	255
23.3	Create a Notification Template	255
23.4	Notification Types	256
23.5	Create custom notifications	264
23.6	Enable and Disable Notifications	269
23.7	Configure the <code>towerhost</code> hostname for notifications	270
23.8	Notifications API	270
24	Schedules	272
24.1	Add a new schedule	273
25	Setting up an Insights Project	275
25.1	Create Insights Credential	275
25.2	Create an Insights Project	277
25.3	Create Insights Inventory	278
25.4	Create a Scan Project	279
25.5	Create a Scan Job Template	280
25.6	Remediate Insights Inventory	283
26	Best Practices	287
26.1	Use Source Control	287
26.2	Ansible file and directory structure	287
26.3	Use Dynamic Inventory Sources	287
26.4	Variable Management for Inventory	288
26.5	Autoscaling	288
26.6	Larger Host Counts	288
26.7	Continuous integration / Continuous Deployment	288
26.8	LDAP authentication performance tips	288
27	Security	289
27.1	Playbook Access and Information Sharing	289
27.2	Role-Based Access Controls	292
27.3	Function of roles: editing and creating	299
28	Index	302
29	Copyright © Red Hat, Inc.	303
	Index	304

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

The *Ansible Tower User Guide* discusses all of the functionality available in Ansible Tower 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 Ansible Tower v3.8.0.

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!

Ansible Tower Version 3.8.0; November 18, 2020; <https://access.redhat.com/>

OVERVIEW

Thank you for your interest in Ansible Tower. Tower is a graphically-enabled framework accessible via a web interface and a REST API endpoint for Ansible, the open source IT orchestration engine. Whether sharing operations tasks with your team or integrating with Ansible through the Tower REST API, Tower provides many powerful tools to make your automation life easier.

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. Tower 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

Ansible Tower 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, Tower records the history of operations and who made them—including objects edited and jobs launched.

Based on user feedback, Ansible Tower 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 Tower’s RBAC system, and can be assigned to multiple users and/or teams for use.

A new ‘Auditor’ type has been introduced in Tower as well, 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 Tower’s API.) Refer to *Role-Based Access Controls* for more information.

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

1.4 Cloud & Autoscaling Flexibility

Tower 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 Tower 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://<Tower 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 Tower setup playbook, making it easy for you to backup and replicate your Tower 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, Tower 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. Tower 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 Tower's 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

Ansible Tower 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 command line to Tower, you can now prompt for any of the following:

- inventory
- credential
- job tags
- limits

1.14 Red Hat Insights Integration

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

1.15 Enhanced Tower 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

Ansible Tower 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 Tower 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

Tower 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, Ansible Tower 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 Ansible Tower 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. Tower 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 Ansible Tower 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, Ansible Tower deploys and runs in such environments.

1.23 Limit the number of hosts per organization

Lots of large organizations have Tower 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 Tower 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 Ansible Tower 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 Tower so you don't have to provide them to Tower directly.

1.26 Automation Hub Integration

Starting with Ansible Tower 3.8, Automation Hub will act as a content provider for Ansible Tower, which requires both an Ansible Tower deployment and an Automation Hub deployment running alongside each other. Tower and Automation Hub can run on either RHEL 7 or 8, but only Tower (not Automation Hub) is supported on an OpenShift Container Platform (OCP).

TOWER LICENSING, UPDATES, AND SUPPORT

Ansible Tower (“**Ansible Tower**”) 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>

Starting with Ansible Tower 3.8, 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 Ansible Tower 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 Tower 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 Ansible Tower, 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 Tower 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

Starting with Tower 3.8, 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 `[tower]` 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
```

(continues on next page)

(continued from previous page)

```
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>
```

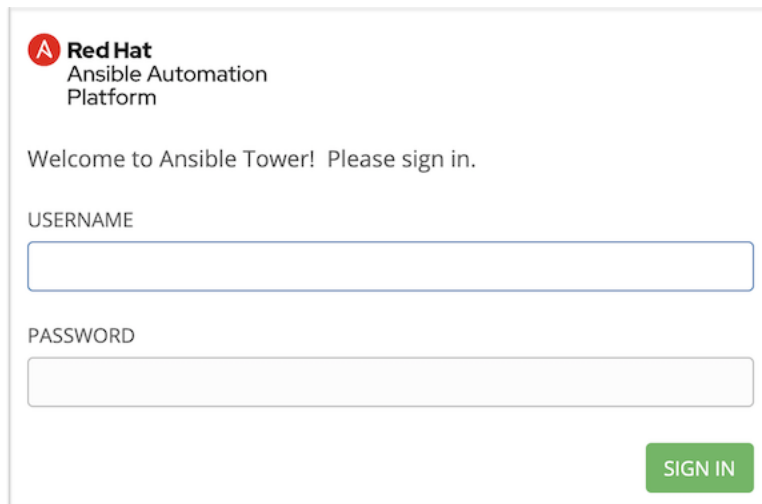
2.6 Tower Component Licenses

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

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

LOGGING IN


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



The screenshot shows the login interface for Red Hat Ansible Automation Platform. At the top left is the Red Hat logo and the text "Red Hat Ansible Automation Platform". Below this is the message "Welcome to Ansible Tower! Please sign in." There are two input fields: "USERNAME" and "PASSWORD". The "PASSWORD" field is currently obscured by a grey rectangle. A green "SIGN IN" button is located at the bottom right of the form.

Log in using a valid Tower username and password.

The default username and password set during installation are *admin* and *password*, but the Tower administrator may have changed these settings during installation. If the default settings have not been changed, you can do so by

accessing the Users () icon from the left navigation bar.

IMPORT A SUBSCRIPTION

Starting with 3.8, Ansible Tower uses available subscriptions or a subscription manifest to authorize the use of Tower. Previously, Tower used a license key and a JSON dictionary of license metadata. Even if you already have valid licenses from previous versions, you must still provide your credentials or a subscriptions manifest again upon upgrading to Ansible Tower 3.8. To obtain your Tower 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* for more detail.

If you **have** a Red Hat Ansible Automation Platform subscription, use your Red Hat customer credentials when you launch Tower 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 Tower 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 Tower 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 Ansible Tower entitlement attached to your Satellite organization's manifest. This can be confirmed by using `hammer subscription list --organization <org_name>`.

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 Tower launches for the first time, the Tower Subscription screen automatically displays.

TOWER SUBSCRIPTION

Welcome to Red Hat Ansible Automation Platform! Please complete the steps below to activate your subscription.

- If you do not have a subscription, you can visit Red Hat to obtain a trial subscription.
- Select your Ansible Automation Platform subscription to use.

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

No file selected.

OR

Provide your Red Hat or Red Hat Satellite credentials below and you can choose from a list of your available subscriptions. The credentials you use will be stored for future use in retrieving renewal or expanded subscriptions.

USERNAME

PASSWORD
- Agree to the End User License Agreement, and click submit.

• END USER LICENSE AGREEMENT

ANSIBLE TOWER BY RED HAT END USER LICENSE AGREEMENT

This end user license agreement ("EULA") governs the use of the Ansible Tower software and any related updates, upgrades, versions, appearance, structure and organization (the "Ansible Tower Software"), regardless of the delivery mechanism.

[License Grant](#) - Subject to the terms of this EULA, Red Hat will not be affiliated with any third party's products.

I agree to the End User License Agreement

TRACKING AND ANALYTICS

By default, Tower collects and transmits analytics data on Tower usage to Red Hat. There are two categories of data collected by Tower. For more information, see [this Tower documentation page](#). Uncheck the following boxes to disable this feature.

- User analytics:** This data is used to enhance future releases of the Tower Software and help streamline customer experience and success.
- Automation analytics:** This data is used to enhance future releases of the Tower Software and to provide Automation Analytics to Tower subscribers.

Use your Red Hat credentials (username and password) to retrieve and import your subscription, or upload a subscription manifest you generate from https://access.redhat.com/management/subscription_allocations.

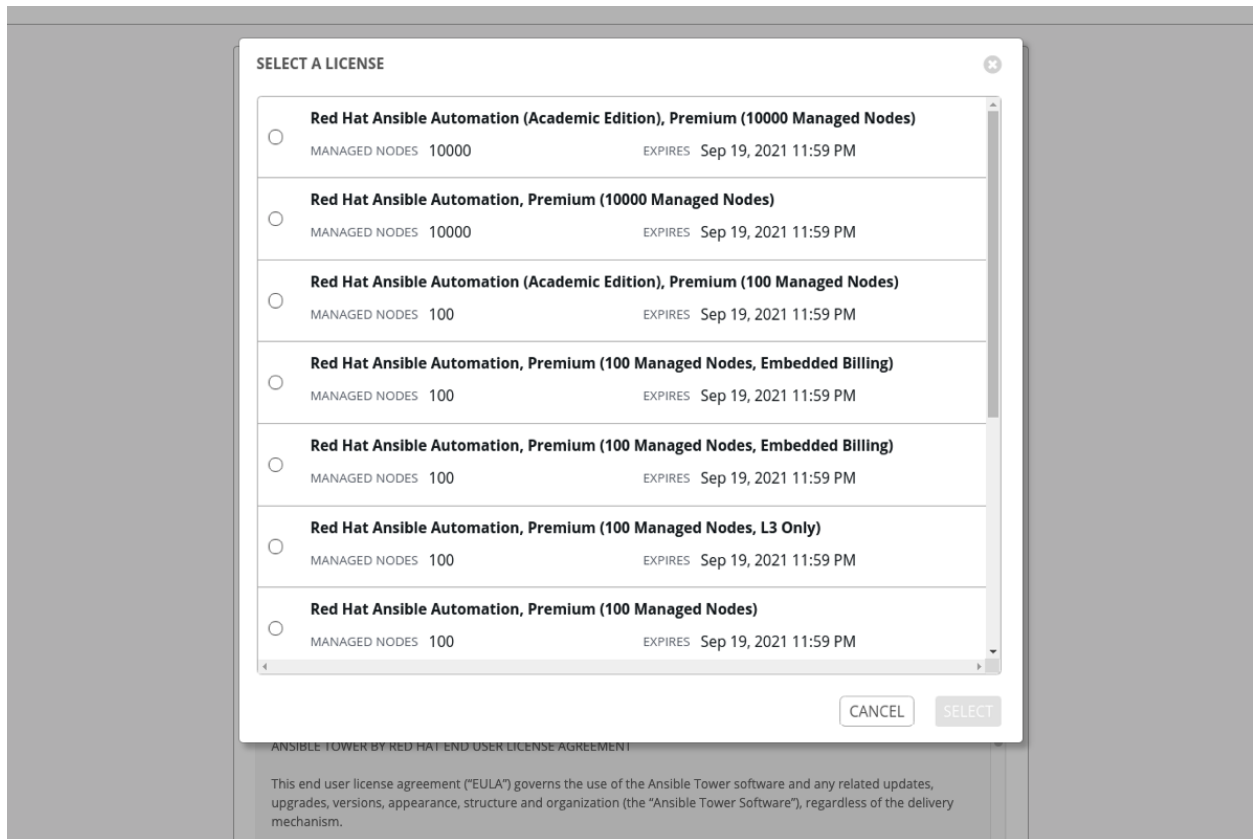
1. Enter your Red Hat customer credentials (username and password) and click **Get Subscriptions**. Use your Satellite username/password if your Tower cluster nodes are registered to Satellite via Subscription Manager. See [Installing Satellite instances on Tower](#) for more information.

Alternatively, if you have a subscriptions manifest, you can upload it by browsing to the location where the file is saved to upload it (the subscription manifest is the complete .zip file, not its component parts). See [Obtaining a subscriptions manifest](#) for more detail.

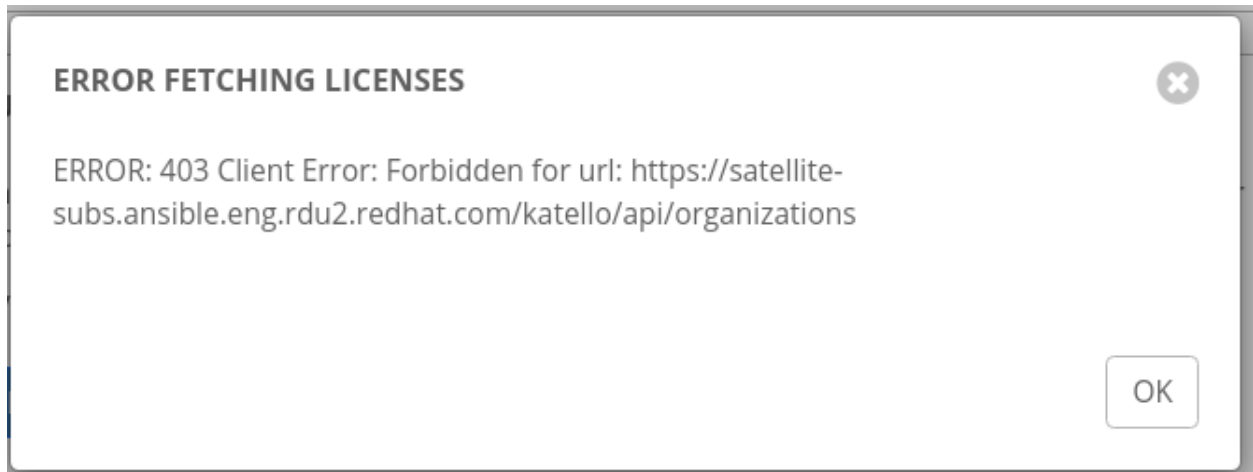
Note: If the **Browse** button is grayed-out, clear the username and password fields to enable the **Browse** button.

2. The subscription metadata is then retrieved from the RHSM/Satellite API, or from the manifest provided.
 - If it is a subscription manifest, Tower will use the first valid subscription included in your manifest file. This is why it is important to only include the subscription you want applied to the Tower installation.
 - If you entered your credential information (username/password), Tower 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 Tower 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 on the License settings in the Configure Tower screen of the UI), you will need to renew it in Tower by one of these two methods.



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



The Satellite username/password is used to query the Satellite API for existing subscriptions. From the Satellite API, Tower 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 Tower 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. Proceed by checking the **End User License Agreement**.
4. The bottom half of the license screen involves analytics data collection. This helps 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 Tower User Interface.
 - **Automation analytics** provides a high level analysis of your automation with Ansible Tower, which is used to help you identify trends and anomalous use of Tower. For opt-in of Automation Analytics to have any effect, your instance of Ansible Tower **must** be running on Red Hat Enterprise Linux. See instructions described in the [Automation Analytics](#) section.

Note: At this time, Automation Insights is not supported when Ansible Tower is running in the OpenShift Container Platform. You may change your analytics data collection preferences at any time, as described in the [Usability Analytics and Data Collection](#) section.

5. After you have specified your tracking and analytics preferences, click **Submit**.

Once your subscription has been accepted, Tower briefly displays the license screen and navigates you to the Dashboard of the Ansible Tower interface. For later reference, you can return to the license screen by clicking the Settings



() icon from the left navigation bar and select the **License** tab from the Settings screen.

LICENSE

DETAILS

SUBSCRIPTION ● Valid

VERSION 3.8.0

SUBSCRIPTION TYPE Enterprise

SUBSCRIPTION Red Hat Ansible Automation Platform For Certified Cloud And Service Providers

EXPIRES ON 09/05/2021

TIME REMAINING 339 Days

HOSTS AVAILABLE 5000

HOSTS USED 1

HOSTS REMAINING 4999

If you are ready to upgrade, please contact us by clicking the button below

[CONTACT US](#)

SUBSCRIPTION MANAGEMENT

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

[BROWSE](#) No file selected.

OR

Provide your Red Hat or Red Hat Satellite credentials below and you can choose from a list of your available subscriptions. The credentials you use will be stored for future use in retrieving renewal or expanded subscriptions.

USERNAME

PASSWORD

[GET SUBSCRIPTIONS](#)

Agree to the End User License Agreement, and click submit.

* END USER LICENSE AGREEMENT

ANSIBLE TOWER BY RED HAT END USER LICENSE AGREEMENT

This end user license agreement ("EULA") governs the use of the Ansible Tower software and any related updates, upgrades, versions, appearance, structure and organization (the "Ansible Tower Software"), regardless of the delivery mechanism.

1. License Grant. Subject to the terms of this EULA, Red Hat, Inc. and its affiliates ("Red Hat") grant to you ("You") a non-transferable, non-exclusive, worldwide, non-sublicensable, limited, revocable license to use the Ansible Tower Software for the term of the associated Red Hat Software Subscription (defined in a monthly email to the number of Red Hat Software Subscriptions).

I agree to the End User License Agreement

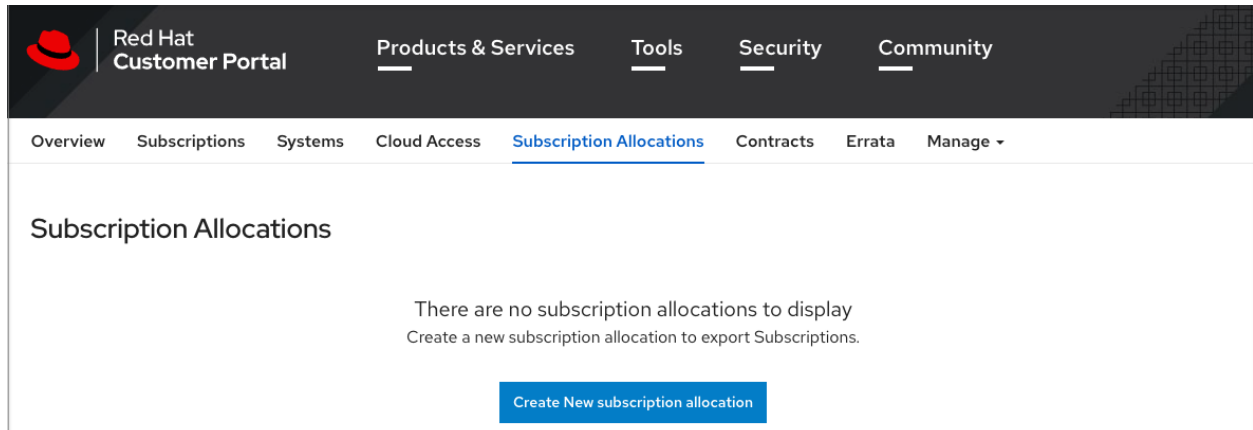
[SUBMIT](#)

4.1 Obtaining a subscriptions manifest

In order to upload a subscriptions manifest into Tower, 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.



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.


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

Details Subscriptions

Basic Information	History
Name my_org_manifest 	Created December 10, 2020
UUID 32868be5-5638-481f-8832-3e06965b06b1	Created by thavo@redhat.com
Type <input type="text" value="Satellite 6.8"/> <input type="button" value="Update"/>	Last Modified Date December 10, 2020

Subscriptions

Simple Content Access ① Disabled

Entitlements 0


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


6. Click the **Subscriptions** tab.

✓ my_org_manifest has been successfully created

[Subscription Allocations](#) » my_org_manifest

my_org_manifest

Details **Subscriptions** 

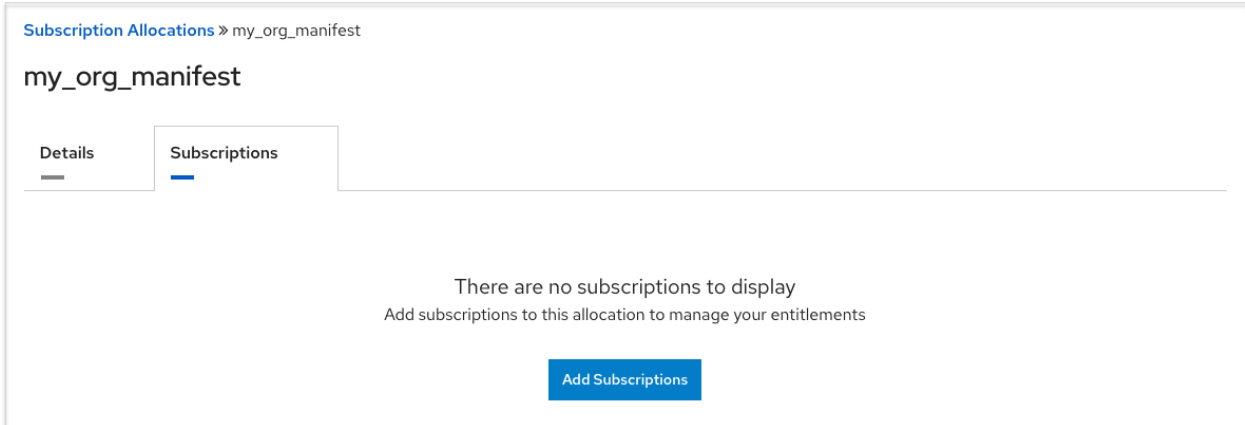
Basic Information	History
Name my_org_manifest 	Created December 10, 2020
UUID 32868be5-5638-481f-8832-3e06965b06b1	Created by thavo@redhat.com
Type <input type="text" value="Satellite 6.8"/> <input type="button" value="Update"/>	Last Modified Date December 10, 2020

Subscriptions

Simple Content Access ① Disabled

Entitlements 0

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



The next screen allows you to select and add an entitlement to put in the manifest file. **Select only one Ansible Automation Platform subscription in your subscription allocation.** Valid Automation Platform subscriptions commonly go by the name “Red Hat Ansible Automation...” or “Red Hat Ansible Automation Platform...”.

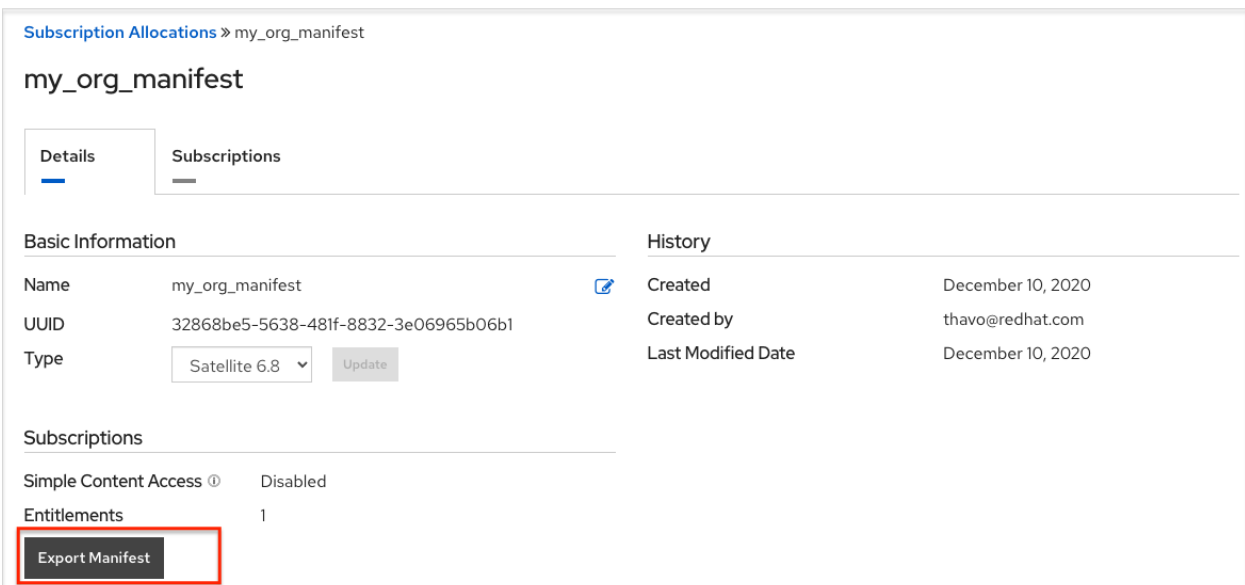
- Specify the number of entitlements/managed nodes to put in the manifest file. This allows you to split up a subscription (for example: 400 nodes on a development cluster and 600 nodes for the production cluster, out of a 1000 node subscription).

Red Hat Ansible Automation Platform for Certified Cloud and Service Providers	12003868	2019-09-05	2021-09-05	4999	<input type="text"/>
Red Hat Ansible Automation, Premium (100 Managed Nodes)	12009552	2019-09-18	2021-09-19	100	<input type="text" value="100"/>
Red Hat Ansible Automation, Premium (100 Managed Nodes, Embedded Billing)	12009552	2019-09-18	2021-09-19	100	<input type="text"/>

- Click **Submit**.

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

- Click the **Details** tab to access the subscription manifest file.
- At the bottom of the details window under *Entitlements*, click the **Export Manifest** button to export the manifest file for this subscription.



A folder pre-pended with `manifest_` in the name is downloaded to your local drive. Only a single subscription from the manifest will be used.

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 Tower subscription manually

If you are unable to apply or update the subscription info using the Tower UI, you can upload the subscriptions manifest manually in an Ansible playbook using the `tower_license` module in the Tower collection:

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

See the [Ansible tower_license module](#) for more information.

THE TOWER USER INTERFACE

The Tower 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 in Tower have an Activity Stream () button. Clicking this brings up the **Activity Stream** for this object.

ACTIVITY STREAM | ALL ACTIVITY

SEARCH KEY All Activity ▼

TIME	INITIATED BY	EVENT	ACTIONS
11/4/2019 4:14:03 PM	system	associated workflow_job New Workflow Job Template to notification	
11/4/2019 4:14:03 PM	system	created notification	
11/4/2019 4:13:53 PM	system	created job Demo Job Template	
11/4/2019 4:13:53 PM	system	associated workflow_job New Workflow Job Template to notification	
11/4/2019 4:13:53 PM	system	created notification	
11/4/2019 4:13:52 PM	admin	created workflow_job New Workflow Job Template	
11/4/2019 4:13:48 PM	admin	associated two nodes on workflow New Workflow Job Template	
11/4/2019 4:13:48 PM	admin	associated two nodes on workflow New Workflow Job Template	
11/4/2019 4:13:48 PM	admin	disassociated two nodes on workflow New Workflow Job Template	
11/4/2019 4:13:48 PM	admin	disassociated two nodes on workflow New Workflow Job Template	
11/4/2019 4:13:48 PM	admin	associated two nodes on workflow New Workflow Job Template	
11/4/2019 4:13:48 PM	admin	associated two nodes on workflow New Workflow Job Template	
11/4/2019 4:13:48 PM	admin	updated workflow_job_template_node New Workflow Job Template	
11/4/2019 4:13:48 PM	admin	created workflow_approval_template New Workflow Job Template Failed approval	
11/4/2019 4:13:48 PM	admin	created workflow_job_template_node New Workflow Job Template	
11/4/2019 4:13:48 PM	admin	created workflow_job_template_node New Workflow Job Template	
11/4/2019 2:06:54 PM	system	associated workflow_job New Workflow Job Template to notification	
11/4/2019 2:06:54 PM	system	created notification	
11/4/2019 2:06:44 PM	system	created job Demo Job Template	
11/4/2019 2:06:44 PM	system	associated workflow_job New Workflow Job Template to notification	

< 1 2 3 4 5 6 7 8 9 10 > PAGE 1 OF 10 ITEMS 1 - 20 OF 186 VIEW PER PAGE 20

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.

ACTIVITY STREAM

ACTIVITY STREAM | ALL ACTIVITY

SEARCH KEY All Activity ▼

TIME

11/4/2019 4:14:03 PM

11/4/2019 4:14:03 PM

11/4/2019 4:13:53 PM

11/4/2019 4:13:53 PM

11/4/2019 4:13:53 PM

11/4/2019 4:13:53 PM

system created notification

EVENT 184

INITIATED BY system on 11/4/2019 4:13:53 PM

ACTION created job Demo Job Template

CHANGES

```

1 {
2   "allow_simultaneous": false,
3   "credentials": [
4     "Demo Credential (1)"
5   ],
6   "description": "",
7   "type": "job_template"

```

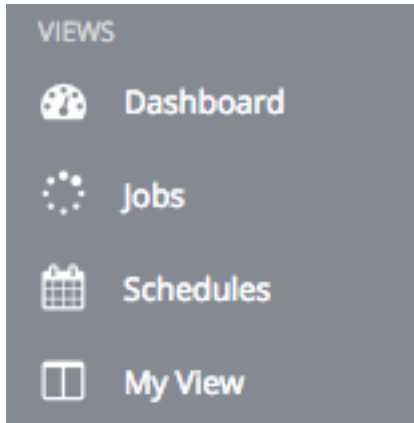
OK

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

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

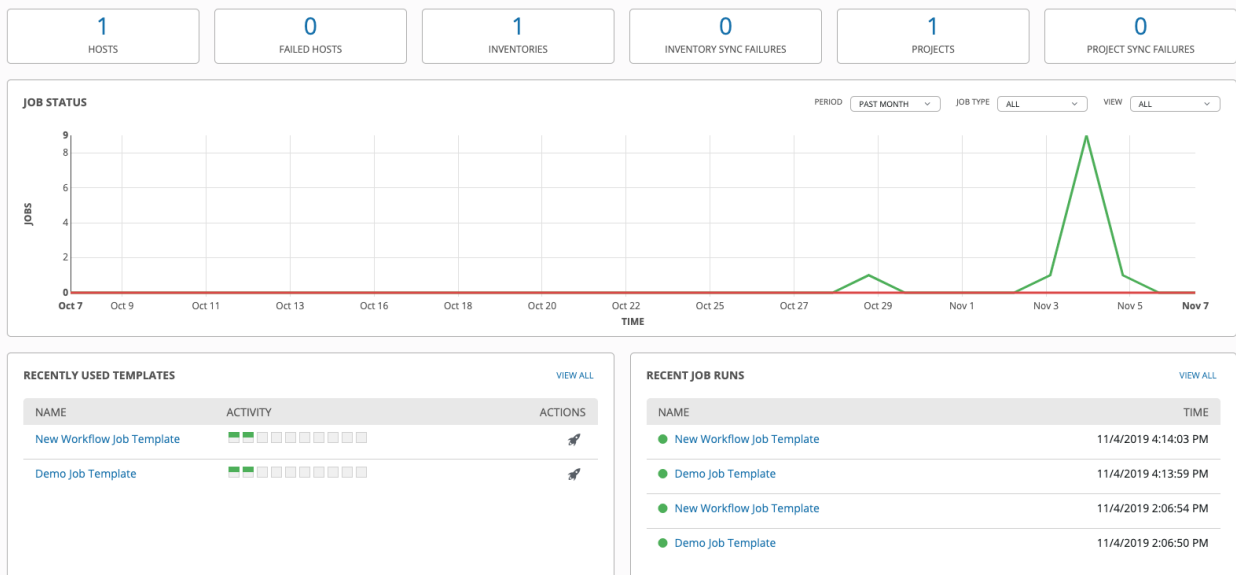
5.2 Views

The Tower 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 in Tower for easy access.



On the main Tower Dashboard screen, a summary appears listing your current **Job Status**. Also available for review are summaries of **Recently Used Templates** and **Recent Job Runs**.



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.


The **Recently Used Templates** section of this display shows a summary of the most recently used templates. You can



also access this summary by clicking the Templates () icon from the left navigation bar.


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

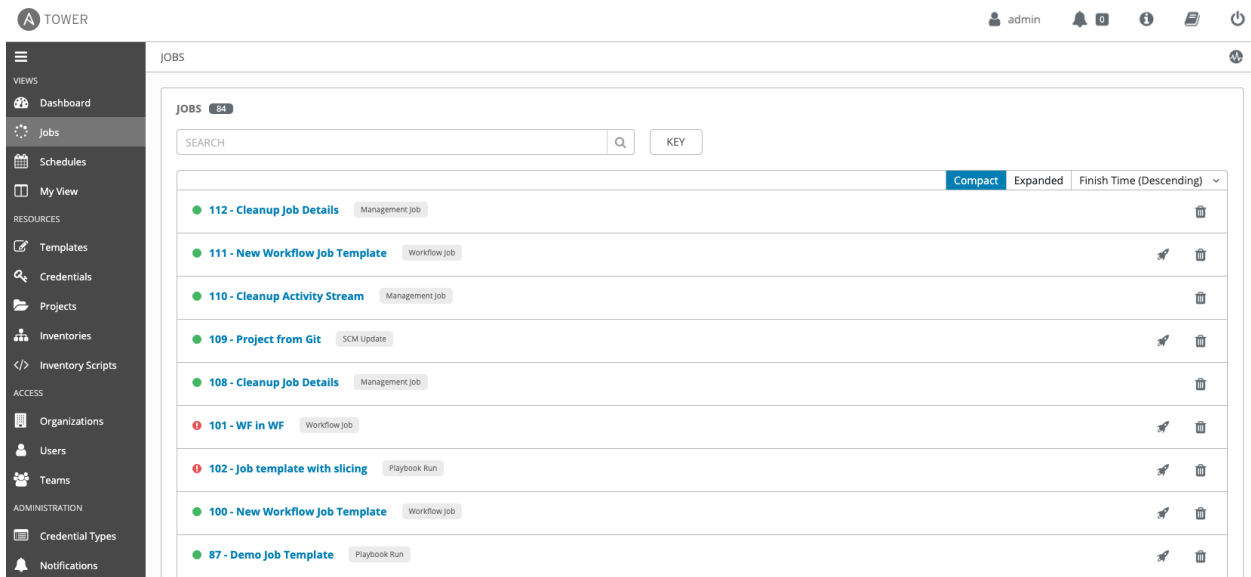


Note: Clicking on the Dashboard () icon from the left navigation bar or the Ansible Tower logo at any time returns you to the Dashboard.

5.2.2 Jobs view



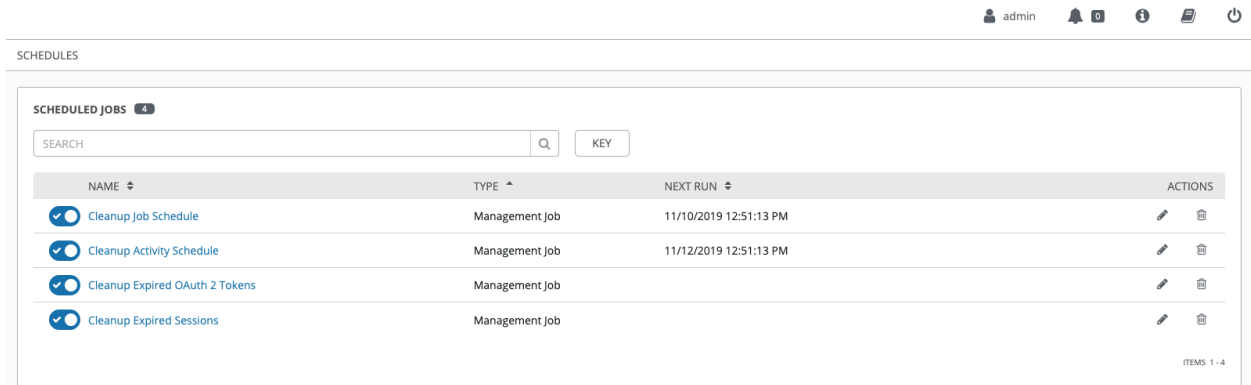
Access the **Jobs** view by clicking the Jobs () icon from the left navigation bar. This view shows all the jobs that have ran in Tower, including projects, templates, management jobs, SCM updates, playbook runs, etc.



5.2.3 Schedules view




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




5.2.4 My View

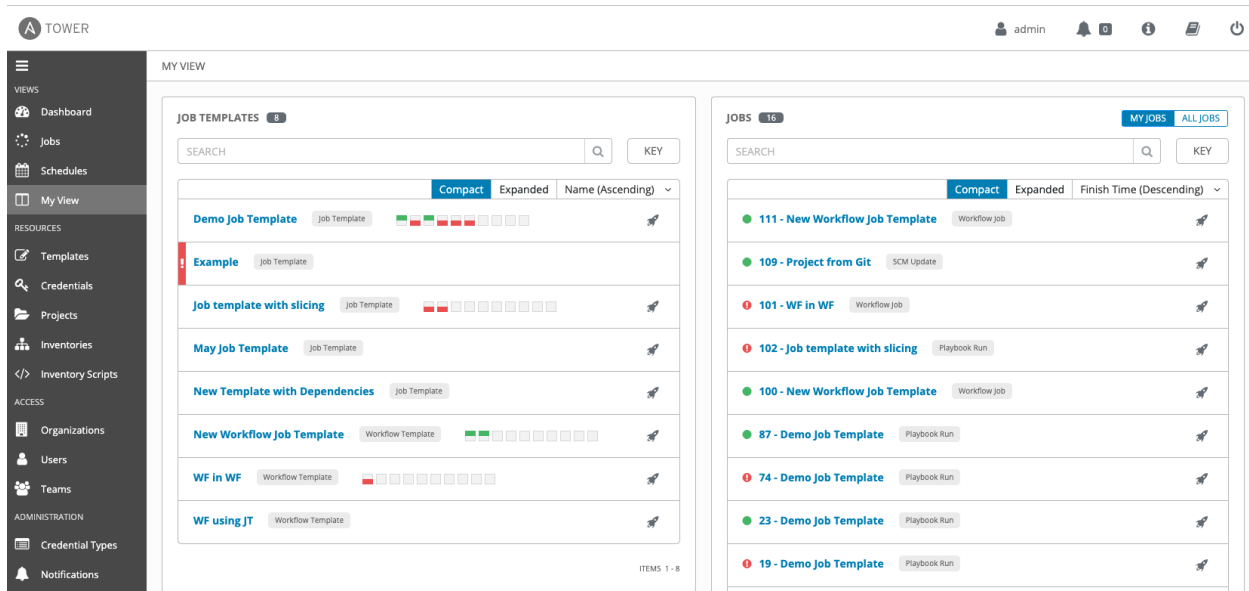



My View, is a user’s single-page view of jobs and job templates. It can be accessed by clicking the My View () icon from the left navigation bar or by navigating to <https://<Tower server name>/portal>.

My View is a simplified interface for users who need to run Ansible jobs, but that do not need an advanced knowledge of Ansible or Tower. My View could be used by, for instance, development teams, or even departmental users in non-technical fields.

My View offers Tower users a simplified, clean interface to the jobs that they are able to run, and the results of jobs that they have run in the past.

Pressing the  button beside a job in My View launches it, potentially asking some survey questions if the job is configured to do so.



My View displays two main sections—**Job Templates** and **Jobs**. The Job Templates panel shows the job templates that are available to be run. To launch a job template, click the  button. This launches the job, which can be viewed in the Jobs panel.

The Jobs pane shows the list of jobs that have run in the past. Sort for jobs specific to you by clicking the **My Jobs** button or review all jobs you have access to view by clicking the **All Jobs** button, above the search bar.

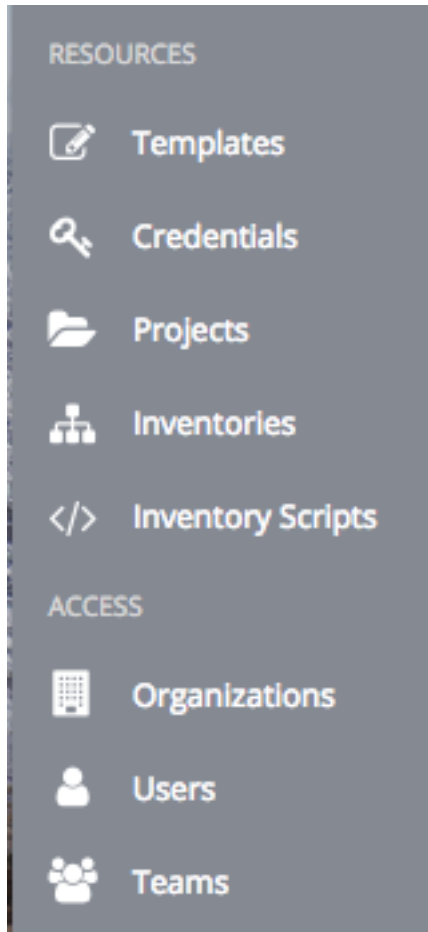
- **My Jobs:** View jobs that you (as the user) ran.
- **All Jobs:** View your team members' completed jobs, viewable based on your RBAC permissions.

For each job, you can view and sort by any number of the job's attributes shown. Clicking on the link for the job opens a new window with the **Job Details** for that job (refer to *Jobs* for more information).

Other portions of the interface are hidden from view until My View is exited.

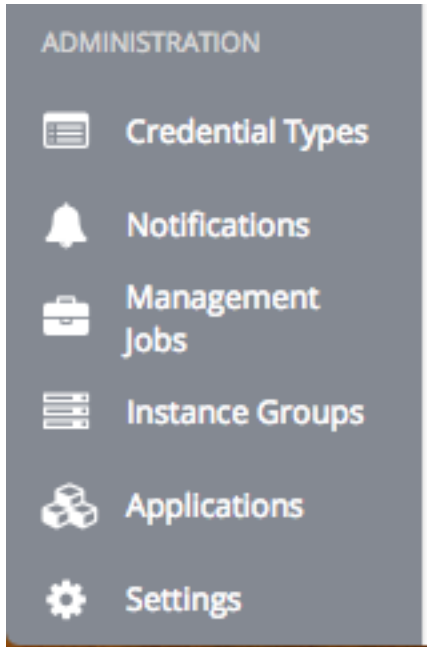
5.3 Resources and Access

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




5.4 Tower Administration Menu


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

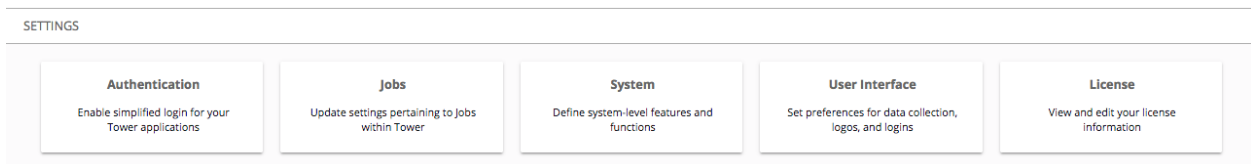


From here, you can create, view, and edit *custom credential types*, *notifications*, management jobs, *tokens and applications*, and configure Tower settings. Configuring Tower settings is accomplished through the **Settings** menu, which is described in further detail in the proceeding section.

5.4.1 Settings Menu

Starting with Ansible Tower 3.0, the Settings () menu offers access to administrative configuration options. Users of older versions of Ansible Tower (2.4.5 or older) can access most of these through the top-level navigational menu or from their “Setup” menu button.

To enter the Settings window for Ansible Tower, click the Settings  icon at the bottom of the left navigation bar. This page allows you to modify your Tower’s configuration, such as settings associated with authentication, jobs, system, user interface, and view or import your license.



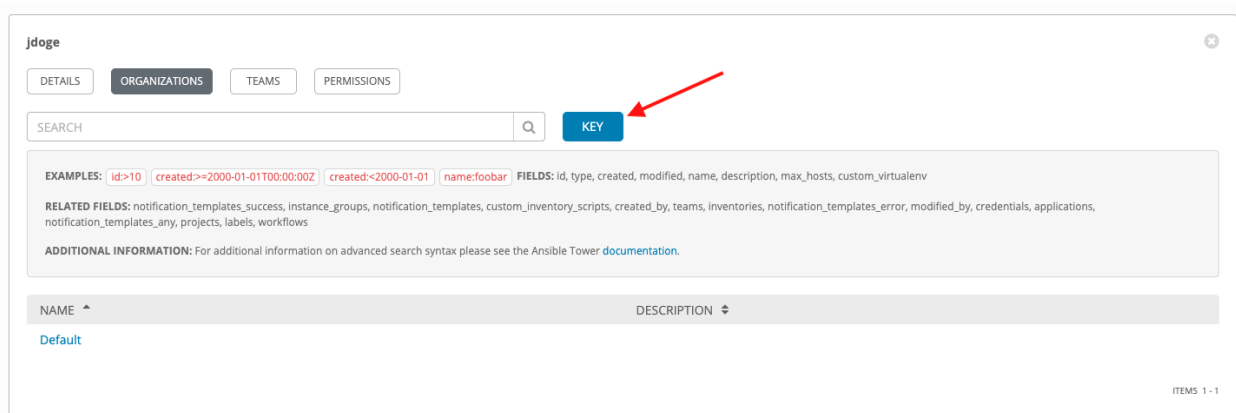
For more information on configuring these settings, refer to [Tower Configuration](#) section of the *Ansible Tower Administration Guide*.

SEARCH

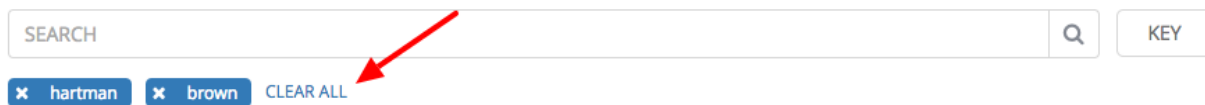
Ansible Tower has a powerful search tool that provides both search and filter capabilities that span across multiple functions.



Acceptable search criteria are provided in an expandable “cheat-sheet” accessible from the **Key** button.



Use the **Clear All** to clear the search criteria.



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 of 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"`), Tower 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 Tower 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, Tower essentially does an `__icontains` search. So, for example, `name:localhost` would send back `?name__icontains=localhost`. Tower 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).

NAME ↕	TYPE ↕	NEXT RUN ↕	ACTIONS
<input checked="" type="checkbox"/> Cleanup Job Schedule	Management Job	11/17/2019 10:51:13 AM	
<input checked="" type="checkbox"/> Cleanup Activity Schedule	Management Job	11/12/2019 10:51:13 AM	
<input checked="" type="checkbox"/> Cleanup Expired OAuth 2 Tokens	Management Job		
<input checked="" type="checkbox"/> Cleanup Expired Sessions	Management Job		

ITEMS 1 - 4

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

SCHEDULED JOBS 4

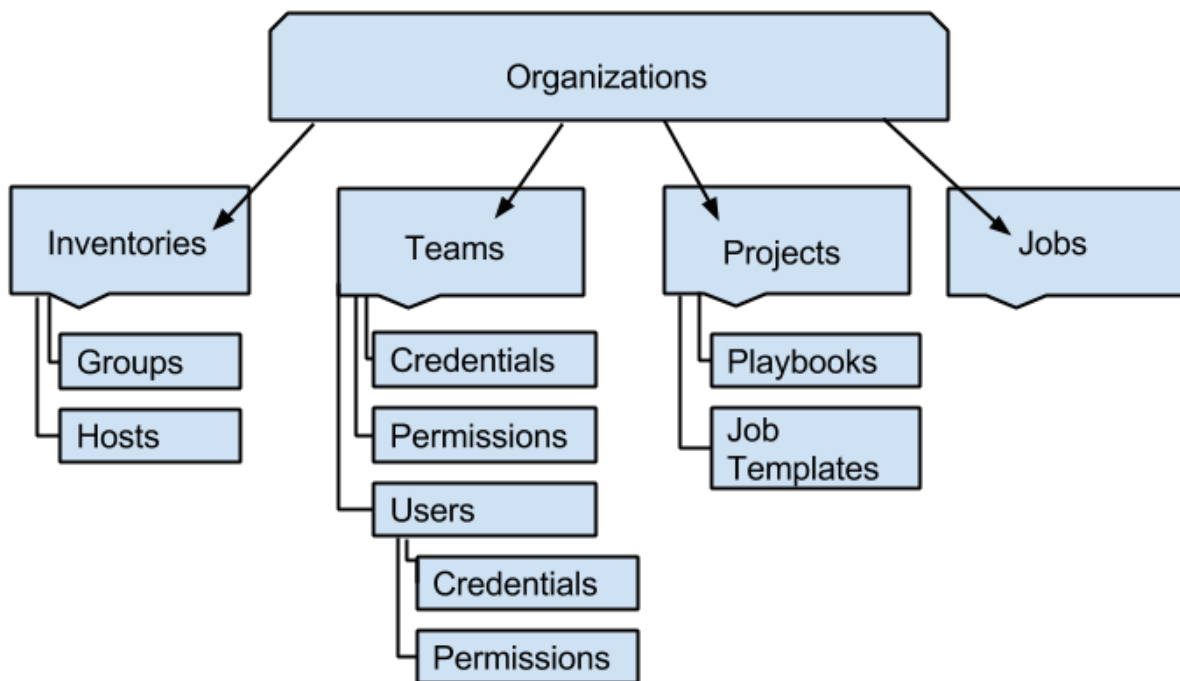
SEARCH


NAME	TYPE	NEXT RUN	ACTIONS
<input checked="" type="checkbox"/> Cleanup Job Schedule	Management Job	11/17/2019 10:51:13 AM	
<input checked="" type="checkbox"/> Cleanup Activity Schedule	Management Job	11/12/2019 10:51:13 AM	
<input checked="" type="checkbox"/> Cleanup Expired OAuth 2 Tokens	Management Job		
<input checked="" type="checkbox"/> Cleanup Expired Sessions	Management Job		

ITEMS 1 - 4

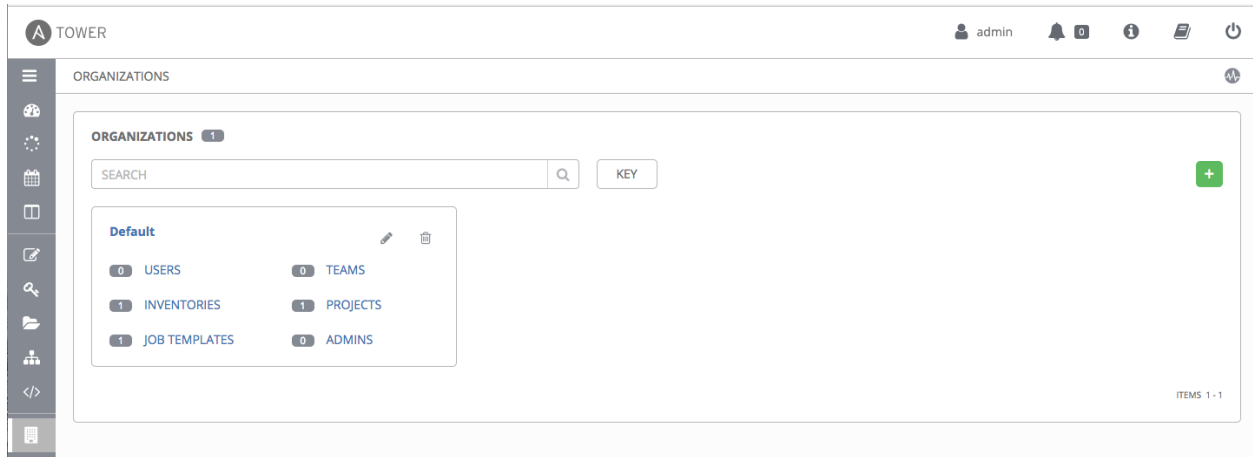
ORGANIZATIONS

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




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

Note: Tower creates a default organization automatically. Users of Tower with a Self-Support level license only have the default organization available and should **not** delete it.



7.1 Creating a New Organization



You can create a new organization by selecting the  button.

ORGANIZATIONS / CREATE ORGANIZATION

NEW ORGANIZATION ✕

DETAILS | USERS | PERMISSIONS

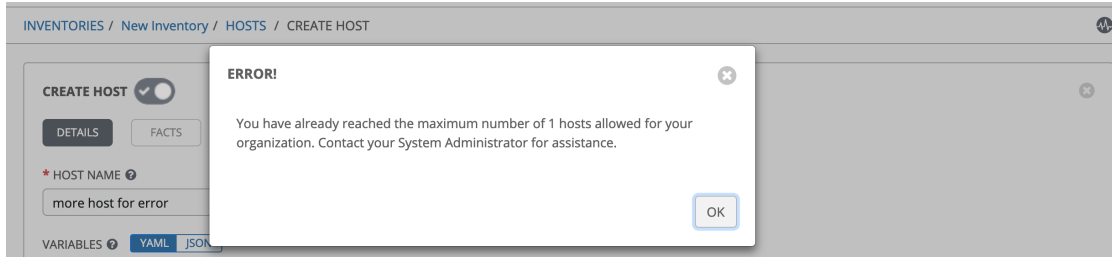
* NAME DESCRIPTION INSTANCE GROUPS


ANSIBLE ENVIRONMENT MAX HOSTS

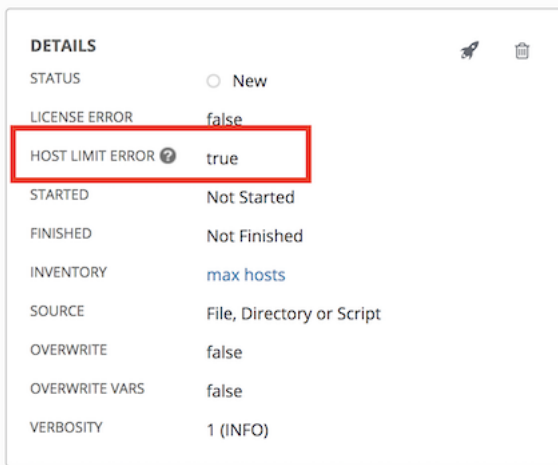
CANCEL SAVE

An organization has several attributes that may be configured:

1. Enter the **Name** for your organization (required).
2. Enter a **Description** for the organization.
3. Enter an **Instance Group** on which to run this organization.
4. Select from the drop-down menu list a custom virtual **Ansible Environment** on which to run this organization. This field is only present if custom environments were previously created. See [Using virtualenv with Ansible Tower](#) in the *Ansible Tower Upgrade and Migration Guide*.
5. 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:

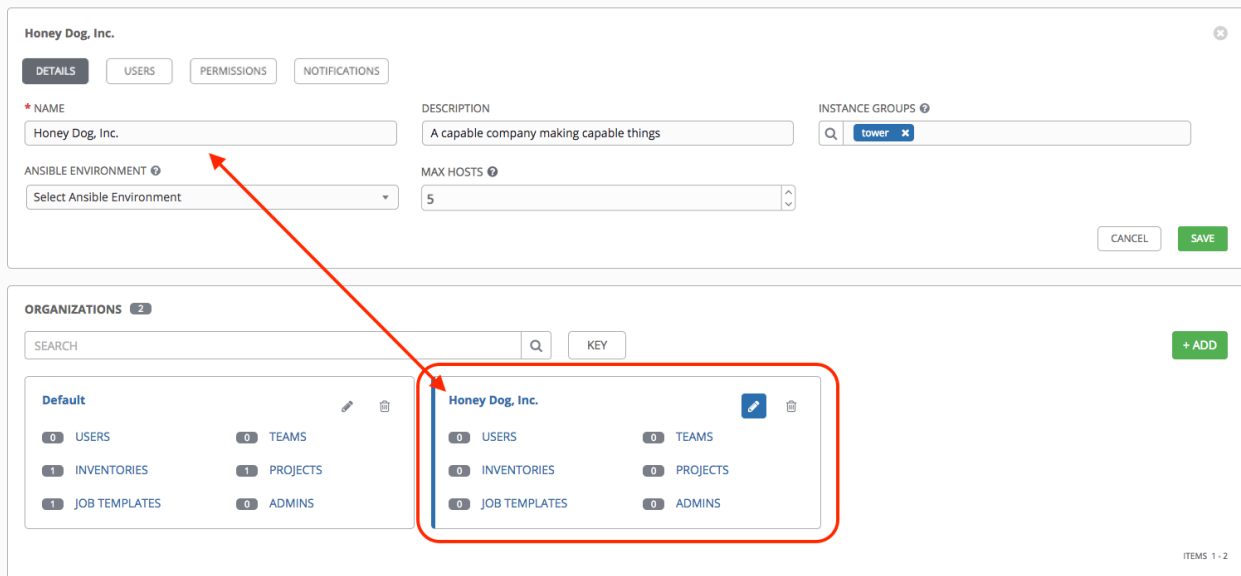


The inventory sync output view also shows the host limit error. Click the  icon for additional detail about the error.



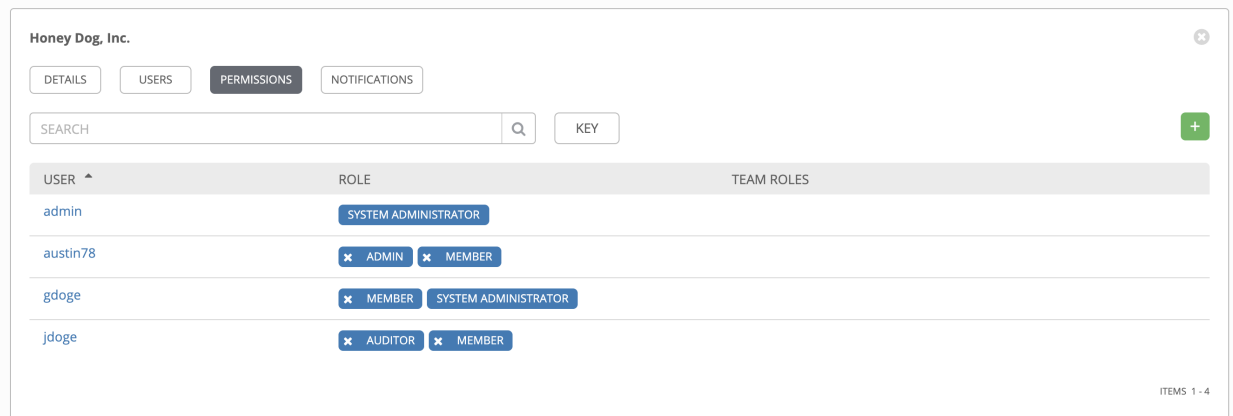
6. Click **Save** to finish creating the organization.

Once created, Tower displays the Organization details, and allows for the managing of users and administrators for the organization.




7.2 Work with Users

Clicking on **Users** (beside **Details** when viewing your organization), displays all the Users associated with this Organization. A User is someone with access to Tower with associated roles and Credentials. Adding a user to an organization adds them as a member only, specifying a role for the user can be done in the the **Permissions** tab, as shown in the example below.



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 the Users () icon from the left navigation bar. The user list from the Organizations view may be sorted by username. Use the Tower Search to search for users by various attributes. Click **Key** for using the search, or refer to the [Search](#) chapter for more information.

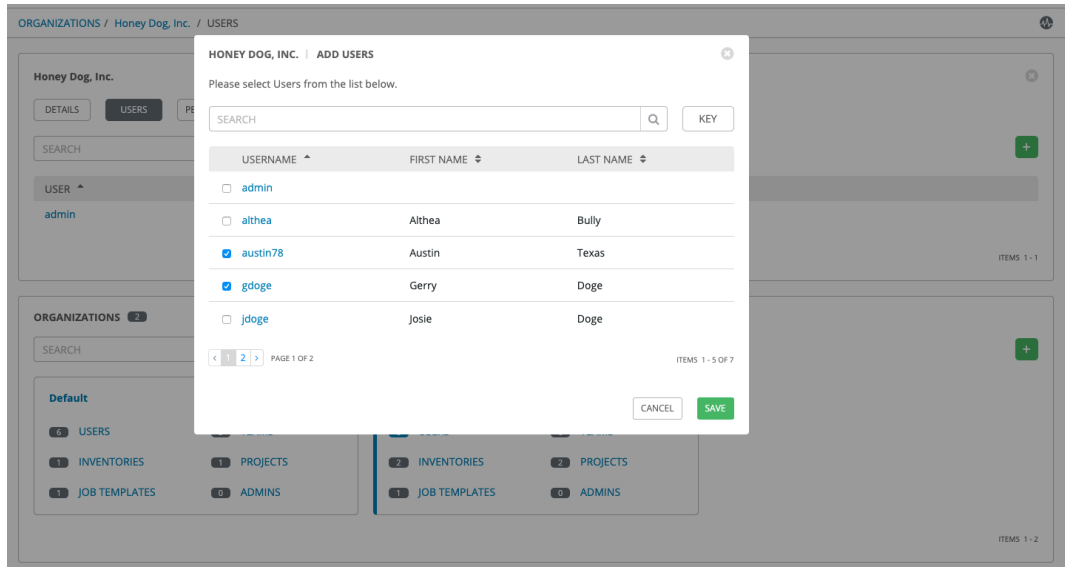
Clicking on a user brings up that user's details, allowing you to review, grant, edit, and remove associated permissions for that user. For more information, refer to [Users](#).

7.2.1 Add a User

In order to add a user to an organization, the user must already be created in Tower. Refer to [Create a User](#) to create a user. To add existing users to the Organization:

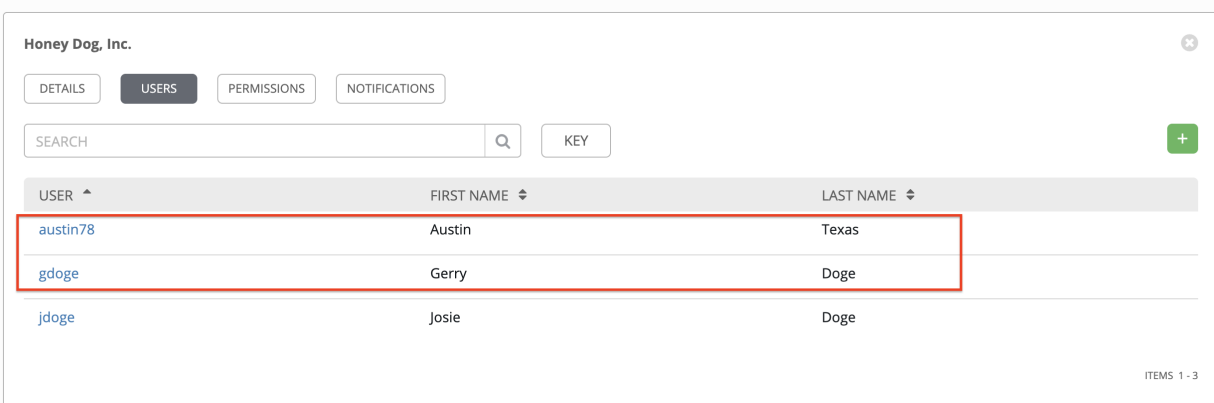


1. Click the  button.
2. Select one or more users from the list of available users by clicking the check box next to the user(s) to add them as members of the organization.



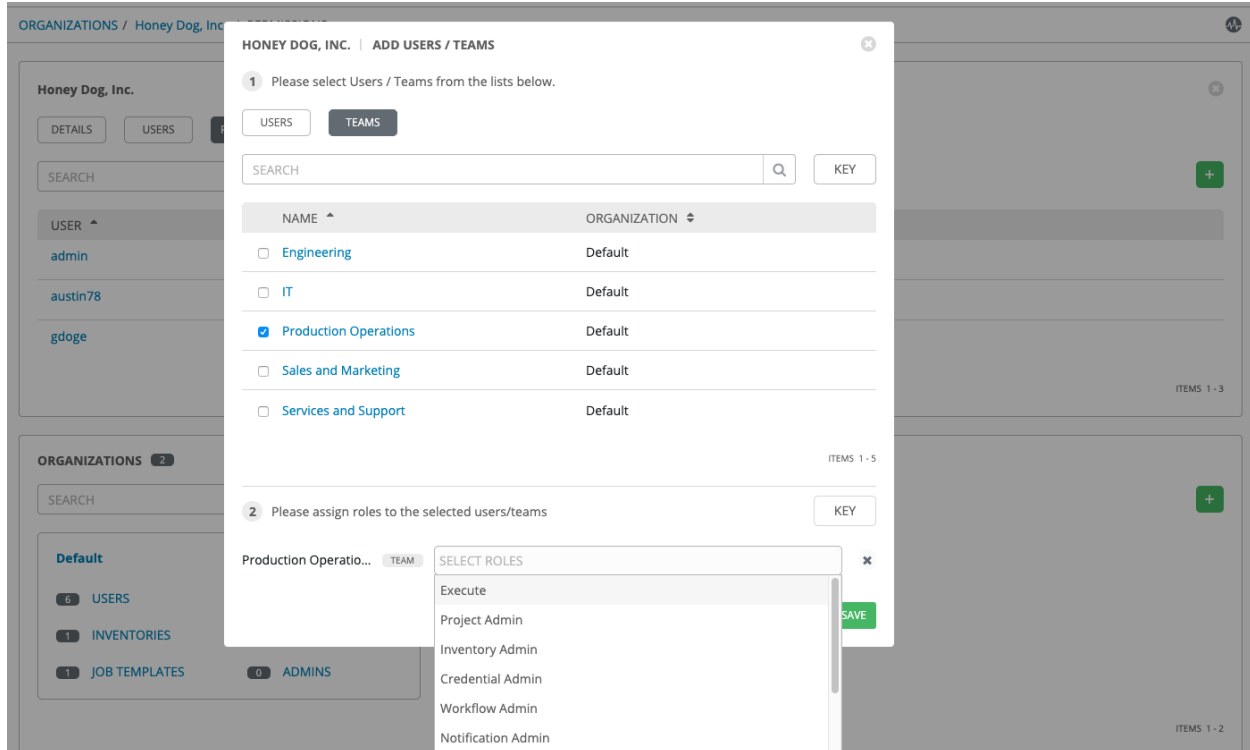
In this example, two users have been selected to be added to this organization.

3. Click the **Save** button when done.



7.3 Work with Permissions

Clicking on **Permissions** (beside **Users** when viewing your organization), allows you to easily manage the permissions for this organization.




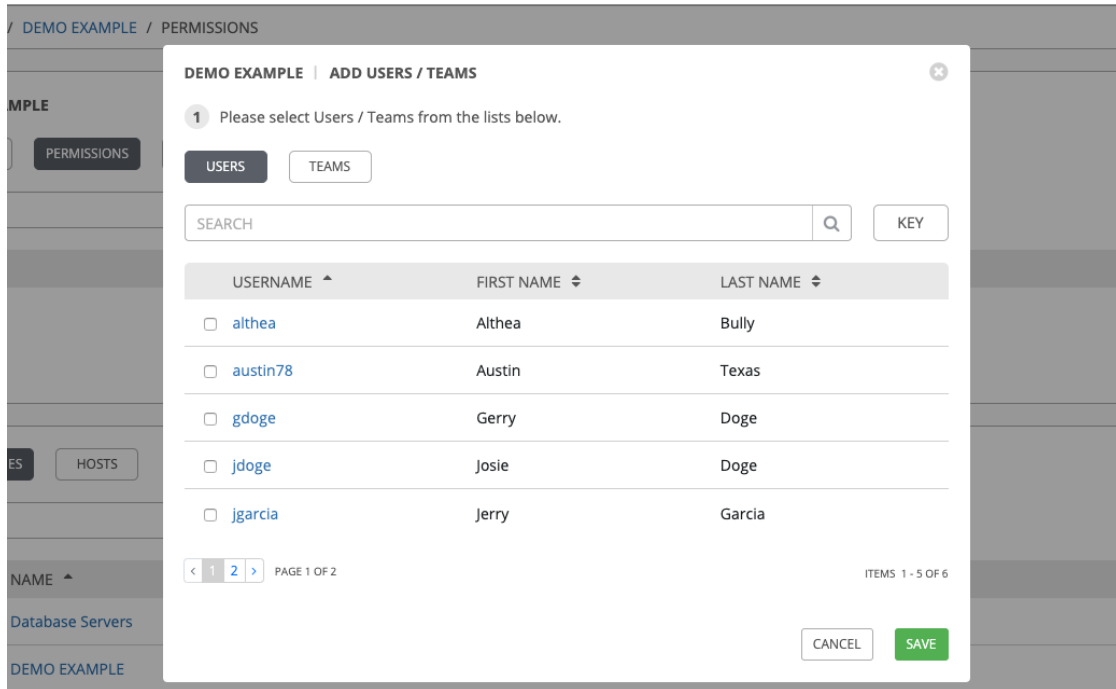
Organizations have a unique set of roles not described here. You can assign specific users certain levels of permissions within your organization, or allow them to act as an admin for a particular resource. Refer to [Role-Based Access Controls](#) for more information.

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

The **Permissions** tab allows you to review, grant, edit, and remove associated permissions for users as well as team members. To assign permissions to a particular user for this resource:

1. Click the **Permissions** tab.

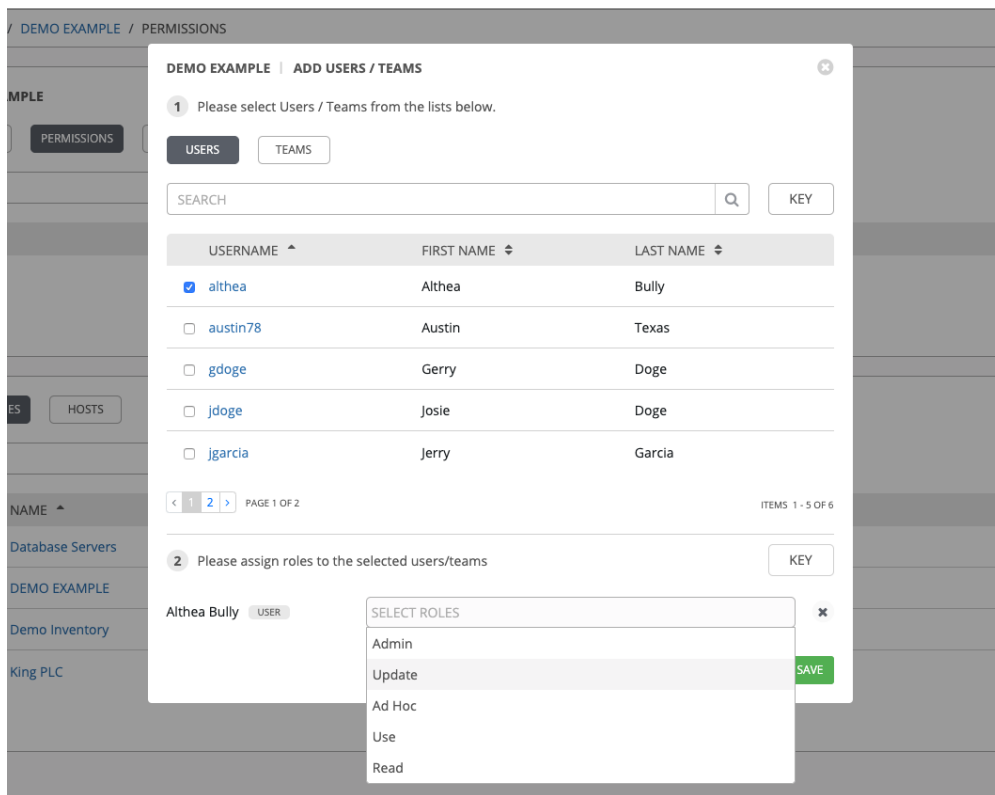
2. Click the  button to open the Add Users/Teams window.



3. Specify the users or teams that will have access then assign them specific roles:
 - a. Click to select one or multiple check boxes beside the name(s) of the user(s) or team(s) to select them.

Note: You can select multiple users and teams at the same time by navigating between the **Users** and **Teams** tabs without saving.

After selections are made, the window expands to allow you to select a role from the drop-down menu list for each user or team you chose.



The example above shows options associated with inventories. Different resources have different options available:

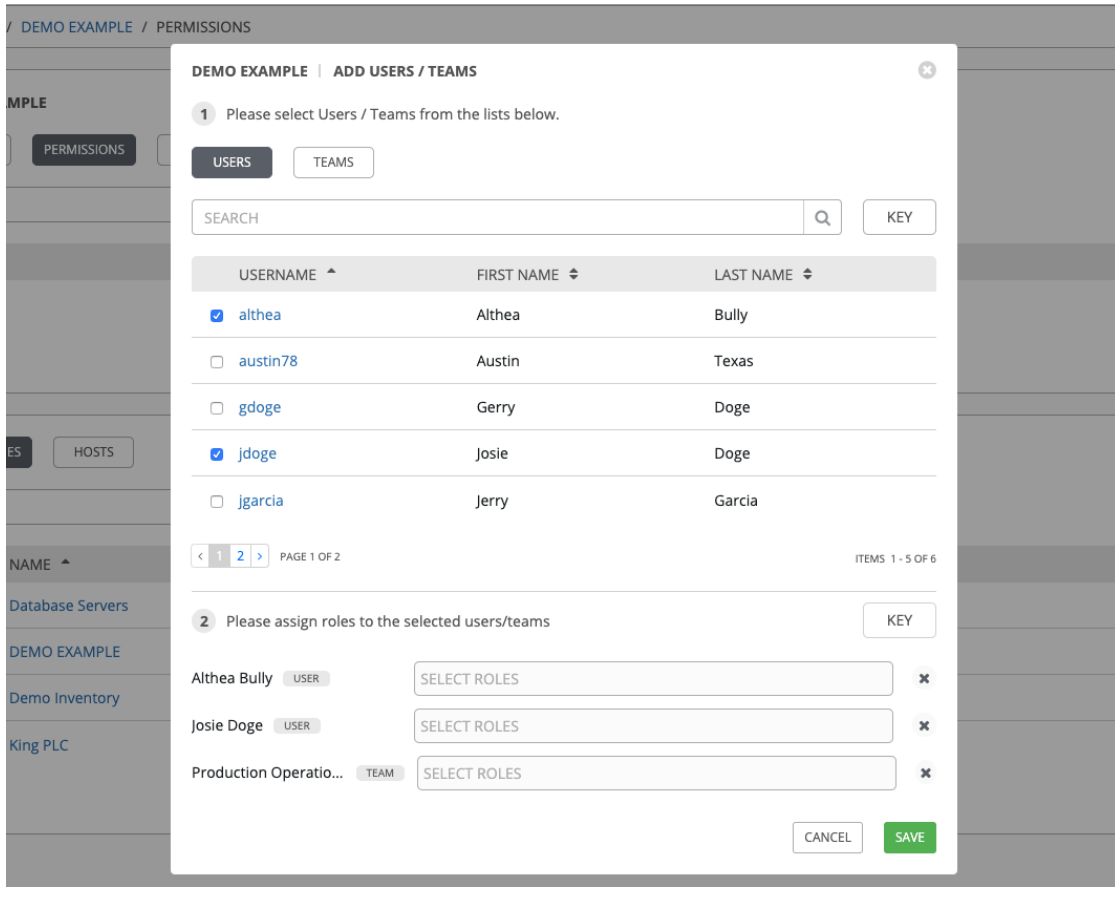
- **Admin** allows read, run, and edit privileges (applies to all resources)
- **Use** allows use of a resource in a job template (applies all resources except job templates)
- **Update** allows updating of project via the SCM Update (applies to projects and inventories)
- **Ad Hoc** allows use of Ad Hoc commands (applies to inventories)
- **Execute** allows launching of a job template (applies to job templates)
- **Read** allows view-only access (applies to all resources)

Tip: Use the **Key** button in the roles selection pane to display a description of each of the roles. For more information, refer to the [Roles](#) section of this guide.

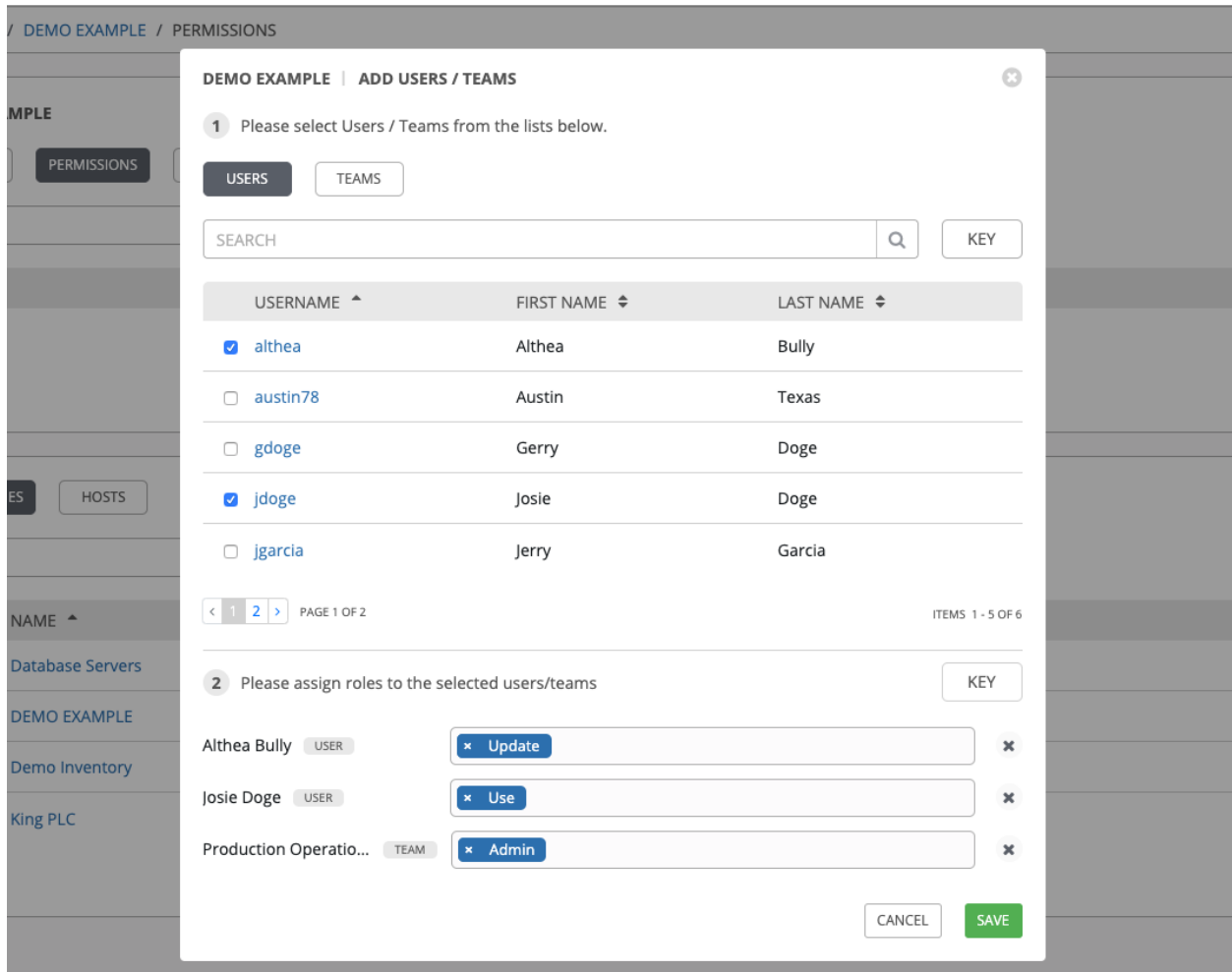
- Select the role to apply to the selected user or team.

Note:

You can assign roles to multiple users and teams by navigating between the **Users** and **Teams** tabs without saving.



4. Review your role assignments for each user and team.



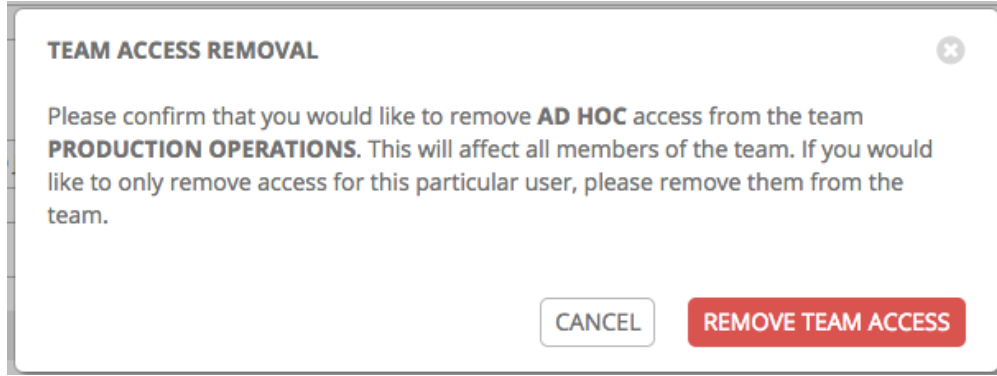
5. Click **Save** when done, and the Add Users/Teams window closes to display the updated roles assigned for each user and team.

USER	ROLE	TEAM ROLES
admin	SYSTEM ADMINISTRATOR	
althea	AD HOC, SYSTEM AUDITOR	USE
jdoge	UPDATE, USE	
mags3707	SYSTEM ADMINISTRATOR	AD HOC, ADMIN, USE
yser	SYSTEM AUDITOR	

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

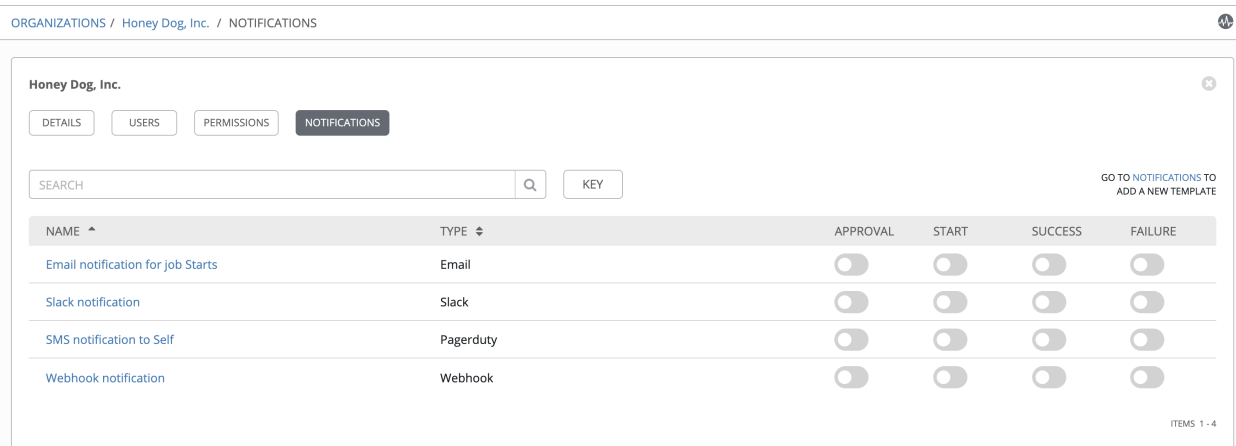
USER	ROLE	TEAM ROLES
admin	SYSTEM ADMINISTRATOR	
althea	AD HOC, SYSTEM AUDITOR	USE
jdoge	UPDATE, USE	
mags3707	SYSTEM ADMINISTRATOR	AD HOC, ADMIN, USE
yser	SYSTEM AUDITOR	

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



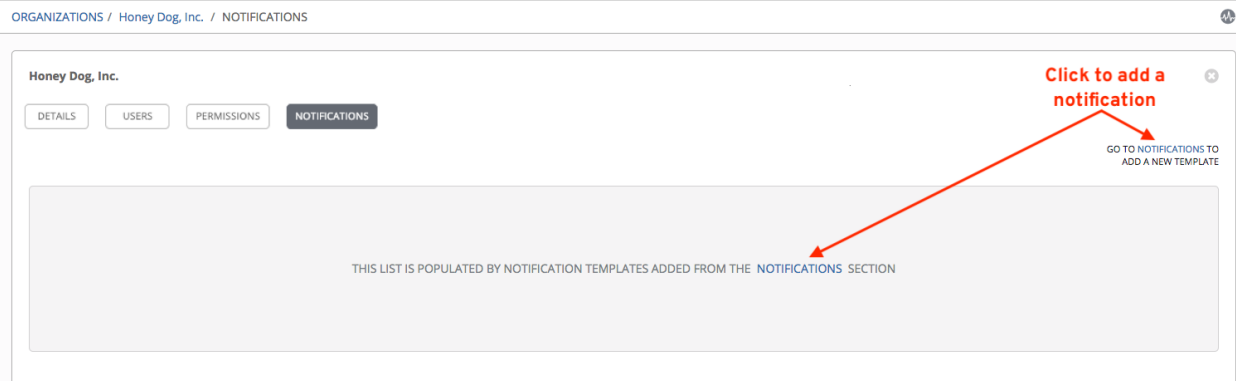
7.4 Work with Notifications

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



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

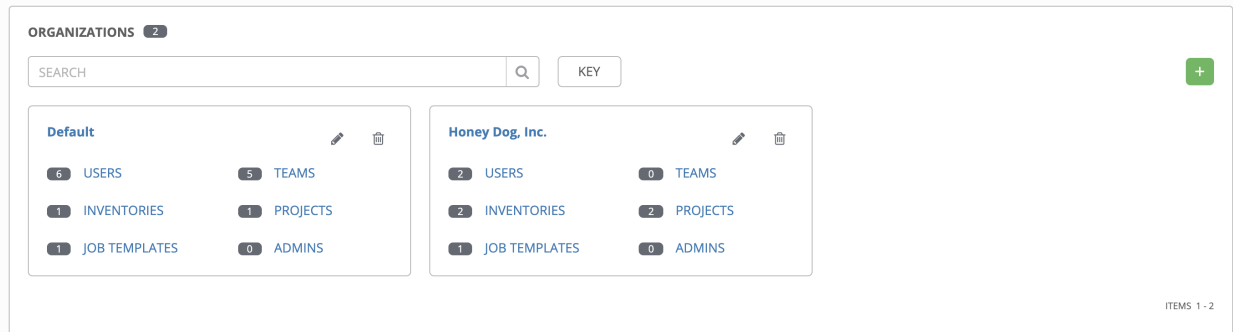
If no notifications have been set up, click the **NOTIFICATIONS** link from above or inside the gray box to add or create a new notification.





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

7.5 Organization Summary

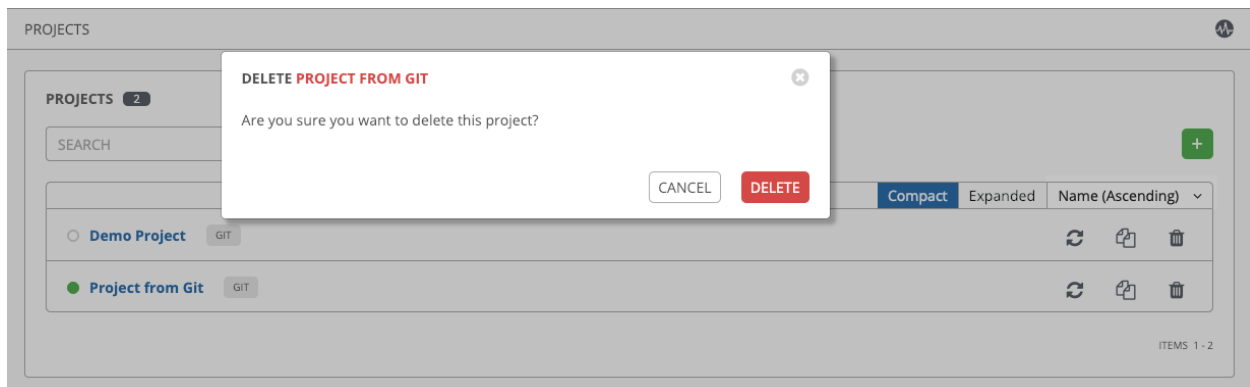
An at-a-glance view of various resources associated with an organization displays at the bottom of each Organization view, called the Organization Summary.



Click on each of the categories to view a list of resources associated with them. Some allow resources to be added, edited, or deleted, such as Users and Admins, while others require editing from another area of the user interface.


From the summary, you can edit the details of an organization () or delete it altogether ().

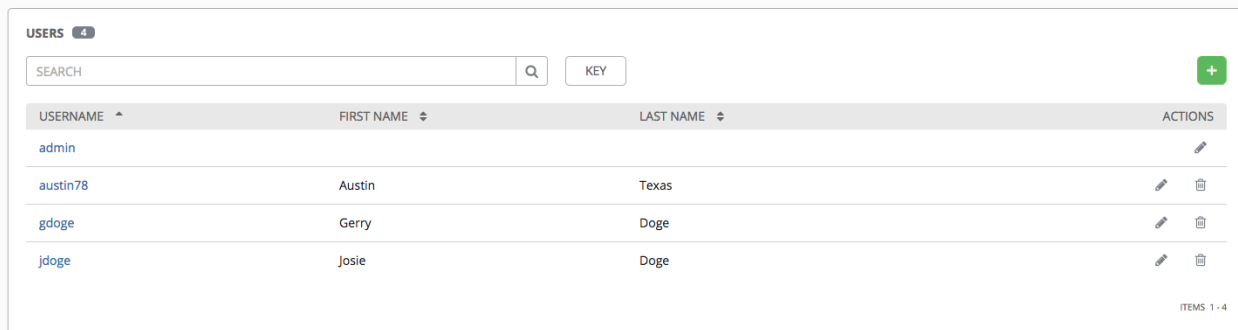
Note: If deleting items that are used by other work items, a message opens listing the items are affected by the deletion and prompts you to confirm the deletion. Some screens will contain items that are invalid or previously deleted, so they will fail to run. Below is an example of such a message:



USERS

A **User** is someone who has access to Tower with associated permissions and credentials. Access the Users page by

clicking the Users () icon from the left navigation bar. The Users page allows you to manage all Tower users. 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	ACTIONS
admin			
austin78	Austin	Texas	
gdoge	Gerry	Doge	
jdoge	Josie	Doge	

8.1 Create a User

To create a new user:



1. Click the  button, which opens the Create User dialog.
2. Enter the appropriate details into the following required fields:
 - First Name
 - Last Name
 - Organization (Choose from an existing organization—this is the default organization if you are using a Self-Supported level license.)
 - Email
 - Username
 - Password
 - Confirmation Password
 - User Type

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

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

USERS / CREATE USER ⌵

NEW USER ✕

DETAILS ORGANIZATIONS TEAMS PERMISSIONS

* FIRST NAME

* LAST NAME

* ORGANIZATION

* EMAIL

* USERNAME

* PASSWORD SHOW

* CONFIRM PASSWORD SHOW

USER TYPE

- Normal User
- Normal User
- System Auditor
- System Administrator

CANCEL SAVE

Note: The initial user (usually “admin”) created by the Tower 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.

austin78 ADMIN ✕

DETAILS ORGANIZATIONS TEAMS PERMISSIONS

* FIRST NAME

* LAST NAME

* EMAIL

* USERNAME

PASSWORD SHOW

CONFIRM PASSWORD SHOW

USER TYPE

System Administrator

CANCEL SAVE

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 edit screen displays the last login activity of that user. This information persists at the top of the screen regardless of which tab you’re viewing.

The screenshot shows the user edit interface for 'austin78'. At the top, the user's name and role 'ADMIN' are displayed. A red box highlights the 'LAST LOGGED IN' field, which contains the text '5/6/2019 2:53:27 PM'. Below this are four tabs: 'DETAILS', 'ORGANIZATIONS', 'TEAMS', and 'PERMISSIONS'. The 'DETAILS' tab is active. The form contains several input fields: 'FIRST NAME' (Austin), 'LAST NAME' (Texas), '* EMAIL' (austin78@mail.com), '* USERNAME' (austin78), 'PASSWORD' (with a 'SHOW' button), and 'CONFIRM PASSWORD' (with a 'SHOW' button). A 'USER TYPE' dropdown menu is set to 'System Administrator'. At the bottom right, there are 'CANCEL' and 'SAVE' buttons.

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.

This screenshot is identical to the previous one, but the 'TOKENS' tab is selected and highlighted with a red box. The 'LAST LOGGED IN' field is no longer visible as it is hidden behind the tabs.

8.2 User Types - Quick View

Once a user has been created, you can easily view permissions and user type information by looking beside their user name in the User overview screen.

The screenshot shows the 'USERS / jdoge' overview page. A user card for 'jdoge' is displayed with the role 'AUDITOR'. A red arrow points to the 'AUDITOR' label with the text 'View user labels here for Auditor, Admin, LDAP, etc.'. The user card has tabs for 'DETAILS', 'ORGANIZATIONS', 'TEAMS', and 'PERMISSIONS'. The 'DETAILS' tab is active, showing fields for '* FIRST NAME' (Josie), '* LAST NAME' (Doge), and '* EMAIL' (jdoge@mail.com).

If the user account is associated with an enterprise-level authentication method (such as SAML, RADIUS, or LDAP), the user type may look like:

USERS / jdoge

jdoge **RADIUS**

DETAILS ORGANIZATIONS TEAMS PERMISSIONS

* FIRST NAME: Josie * LAST NAME: Doge * EMAIL: jdoge@mail.com

If the user account is associated with a social authentication method, the user type will look like:

USERS / jdoge

jdoge **SOCIAL**

DETAILS ORGANIZATIONS TEAMS PERMISSIONS

* FIRST NAME: Josie * LAST NAME: Doge * EMAIL: jdoge@mail.com

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.

jdoge **AUDITOR**

DETAILS **ORGANIZATIONS** TEAMS PERMISSIONS

SEARCH [] Q KEY

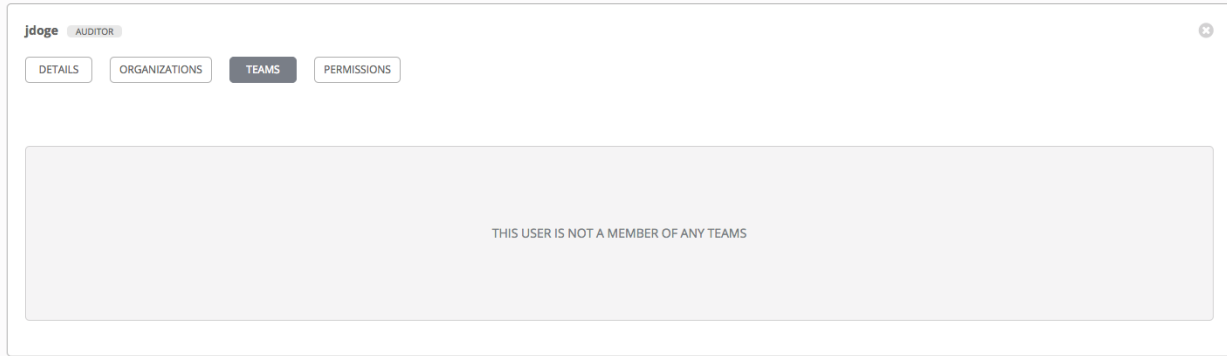
NAME	DESCRIPTION
Honey Dog, Inc.	A capable company making capable things

ITEMS 1 - 1

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.

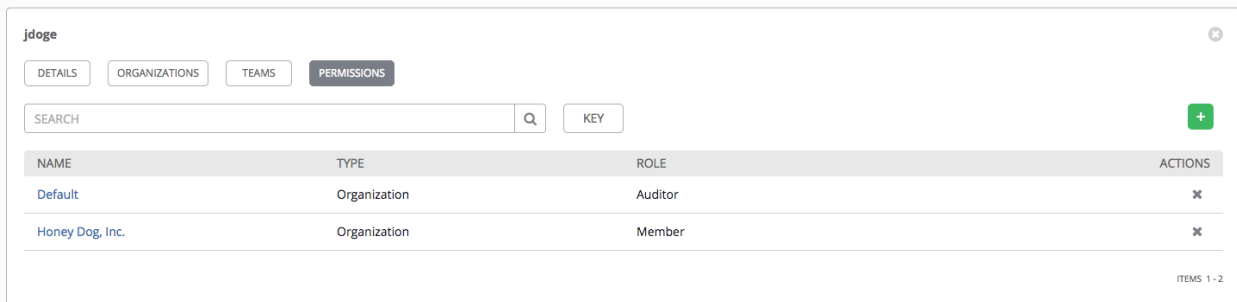


8.5 Users - Permissions

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 Tower 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**.

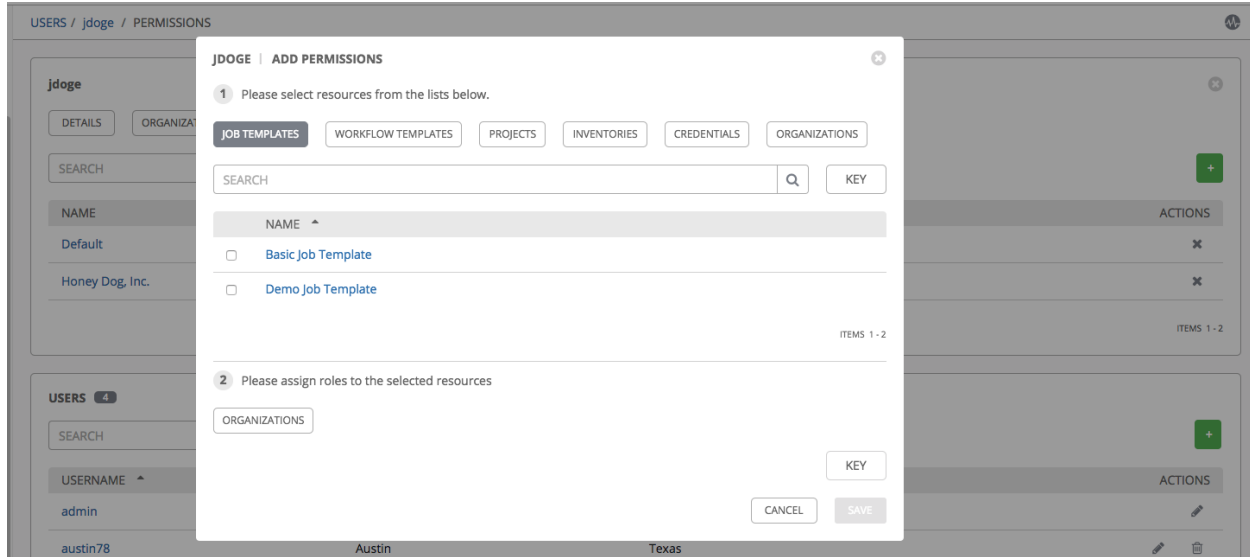


8.5.1 Add Permissions

To add permissions to a particular user:




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

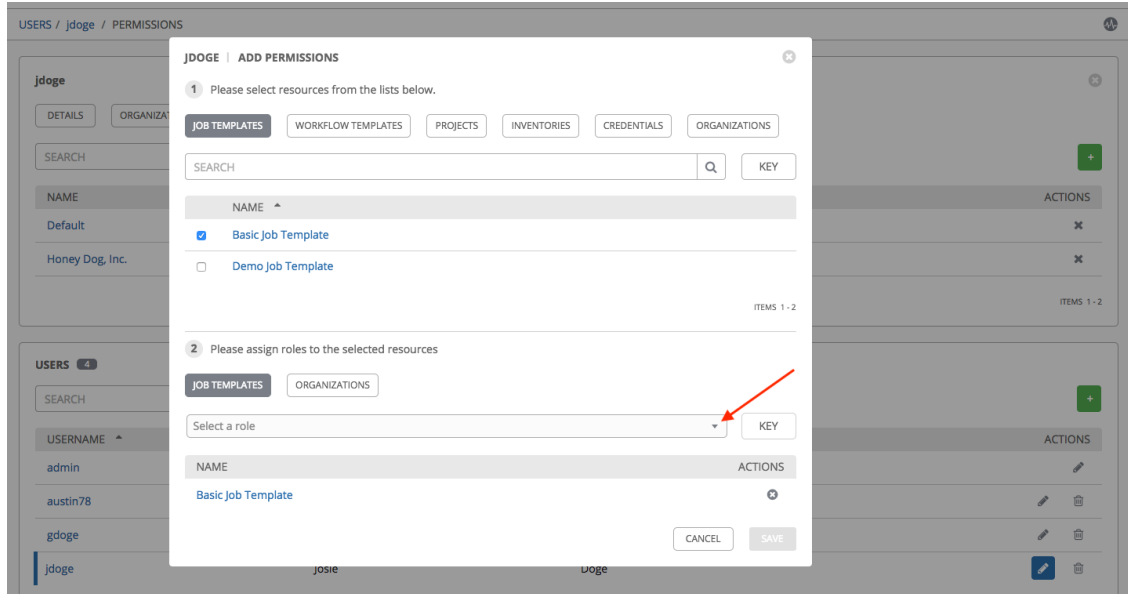


2. Click to select the Tower object for which the user will have access:
 - **Job Templates.** This is the default tab displayed in the Add Permissions Wizard.
 - **Workflow Templates**
 - **Projects**
 - **Inventories**
 - **Credentials**
 - **Organizations**



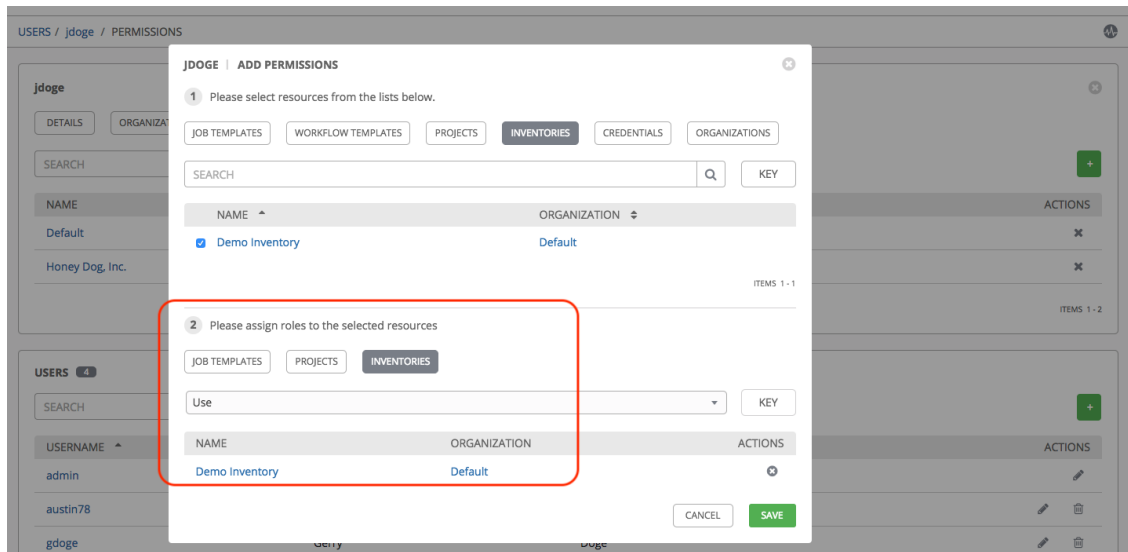
Note: You can assign different roles to different resources all at once to avoid having to click the  button. To do so, simply go from one tab to another after making your selections without saving.

3. Perform the following steps to assign the user specific roles for each type of resource:
 - a. In the desired tab, click the checkbox beside the name of the resource to select it.
The dialog expands to allow you to select the role for the resource you chose.
 - b. Select the role from the drop-down menu list provided. Only some roles are applicable to certain resources.



Tip: Use the **Key** button to display the help text for each of the roles applicable to the resource selected.

- c. Review your role assignments for each of the Tower objects by clicking on their respective buttons in the expanded section 2 of the Add Permissions Wizard.



- d. 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.

NAME	TYPE	ROLE	ACTIONS
Default	Organization	Auditor	✕
Demo Inventory	Inventory	Use	✕
Honey Dog, Inc.	Organization	Member	✕
Sample Project	Project	Admin	✕
Basic Job Template	Job Template	Admin	✕

To remove Permissions for a particular User, click the Disassociate (✕) button under **Actions**. This launches a **Remove Role** dialog, asking you to confirm the disassociation.

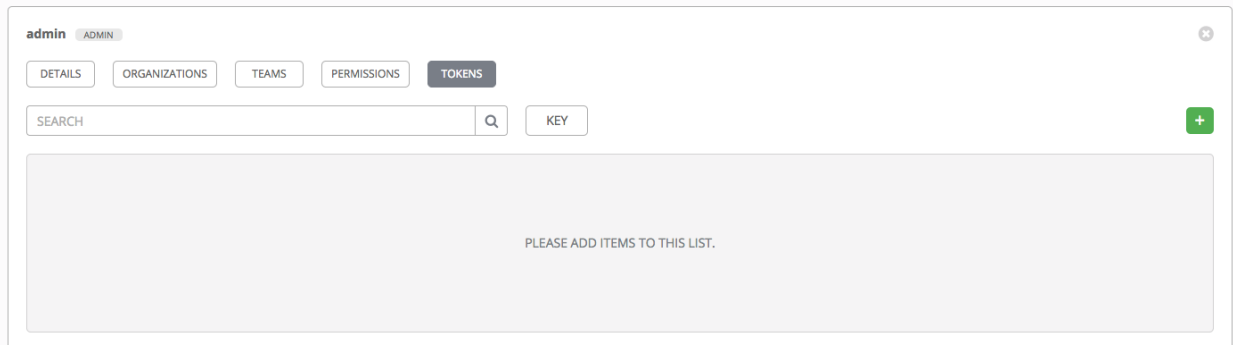
Note: You can also add teams, individual, or multiple users and assign them permissions at the object level (projects, inventories, job templates, and workflow templates) as well. This feature reduces the time for an organization to onboard many users at one time.



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:



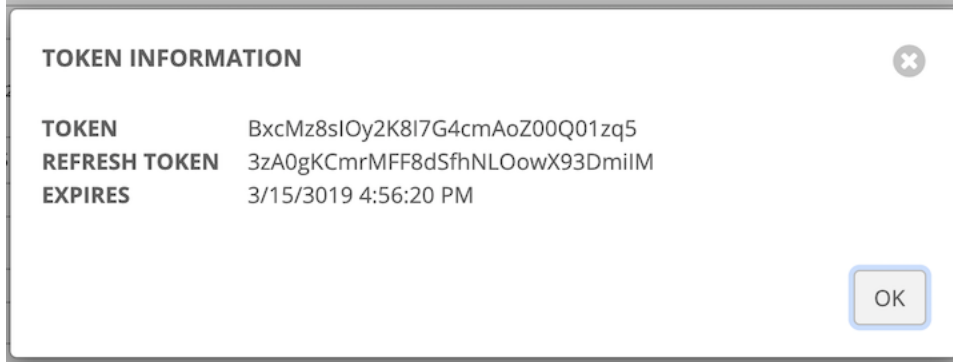
3. Click the  button, which opens the Create Token window.
4. Enter the following details in Create Token window:
 - **Application:** enter the name of the application with which you want to associate your token. Alternatively, you can search for it by clicking the  button. This opens a separate window 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.

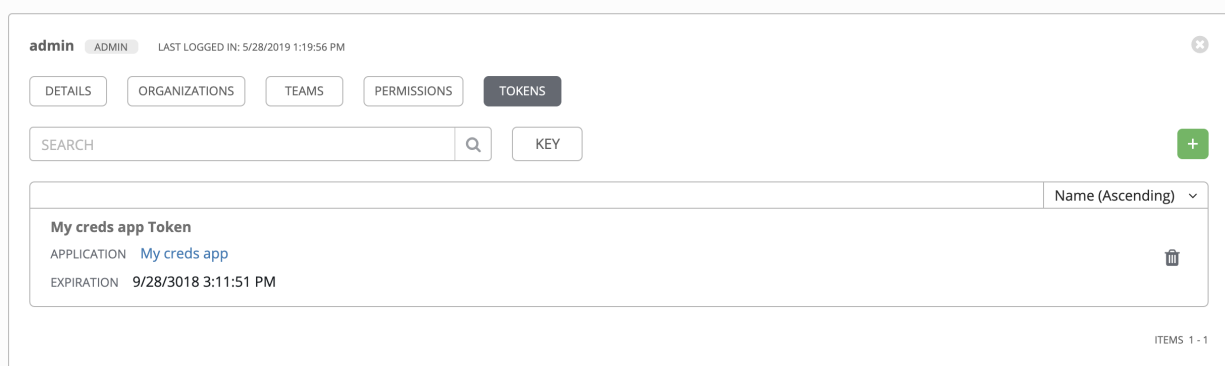
5. When done, click **Save** or **Cancel** to abandon your changes.

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



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.




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 Tower interface click-throughs to assign the same Credentials to the same user.



Access the Teams page by clicking the Teams () icon from the left navigation bar. The Teams page allows you to manage the teams for Tower. The team list may be sorted and searched by **Name** or **Organization**.

NAME	ORGANIZATION	ACTIONS
Engineering	Honey Dog, Inc.	
IT	Honey Dog, Inc.	
Sales and Marketing	Honey Dog, Inc.	
Services and Support	Honey Dog, Inc.	

9.1 Create a Team

To create a new Team:

1. Click the  button.

NEW TEAM

DETAILS | USERS | PERMISSIONS

* NAME:

DESCRIPTION:


* ORGANIZATION:

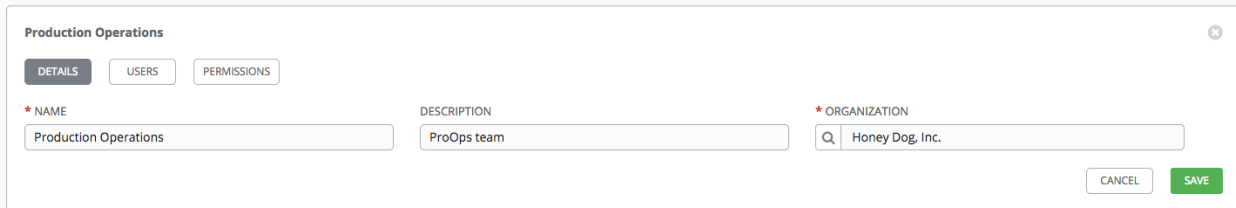
CANCEL SAVE

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, Tower opens the **Details** dialog, which also allows you to review and edit your Team information. This is the same menu that is opened if the Edit () button is clicked from the **Teams** link. You can also review **Users** and **Permissions** associated with this Team.



Production Operations [Close]

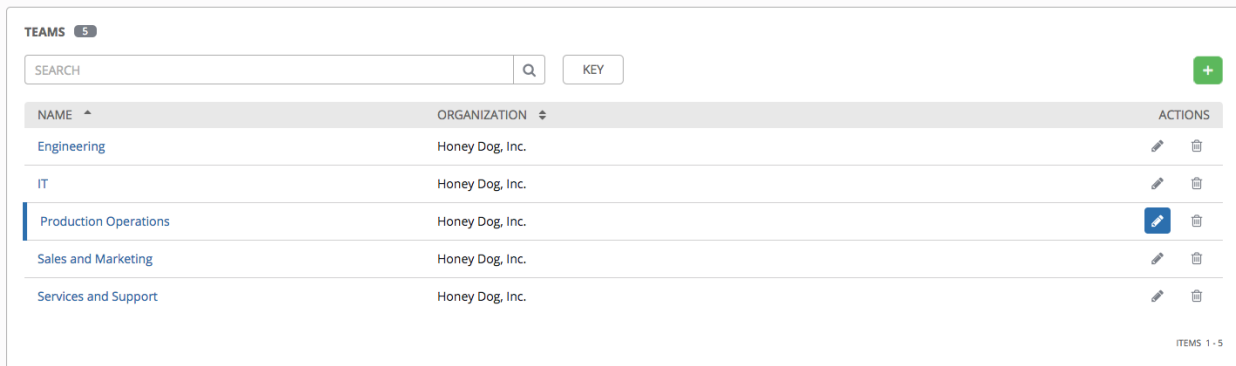
DETAILS | USERS | PERMISSIONS

* NAME: Production Operations

DESCRIPTION: ProOps team

* ORGANIZATION: Honey Dog, Inc.

CANCEL | SAVE



TEAMS 5

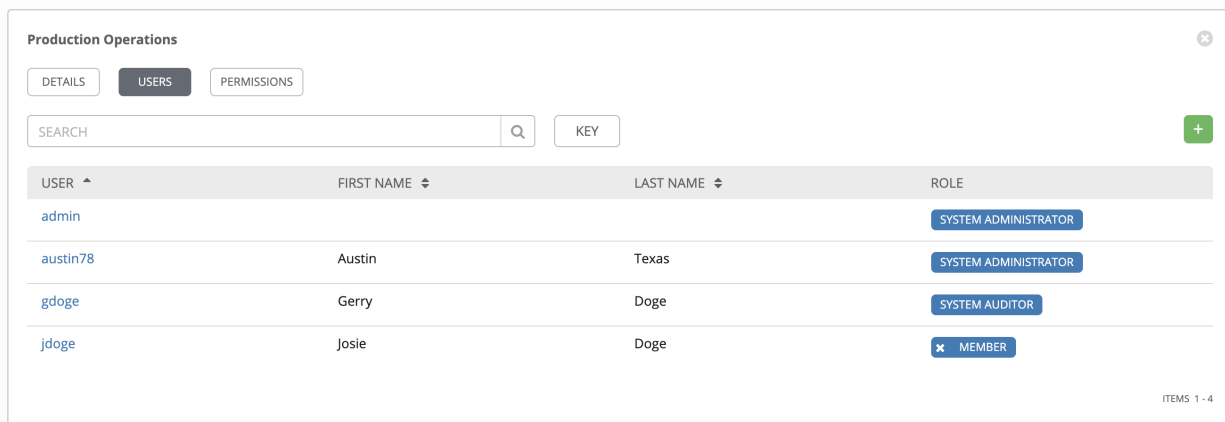
SEARCH [Q] KEY [+] [Add]

NAME	ORGANIZATION	ACTIONS
Engineering	Honey Dog, Inc.	[Edit] [Delete]
IT	Honey Dog, Inc.	[Edit] [Delete]
Production Operations	Honey Dog, Inc.	[Edit] [Delete]
Sales and Marketing	Honey Dog, Inc.	[Edit] [Delete]
Services and Support	Honey Dog, Inc.	[Edit] [Delete]

ITEMS 1 - 5

9.1.1 Teams - Users

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](#).



Production Operations [Close]

DETAILS | **USERS** | PERMISSIONS

SEARCH [Q] KEY [+] [Add]


USER	FIRST NAME	LAST NAME	ROLE
admin			SYSTEM ADMINISTRATOR
austin78	Austin	Texas	SYSTEM ADMINISTRATOR
gdoge	Gerry	Doge	SYSTEM AUDITOR
jdoge	Josie	Doge	MEMBER

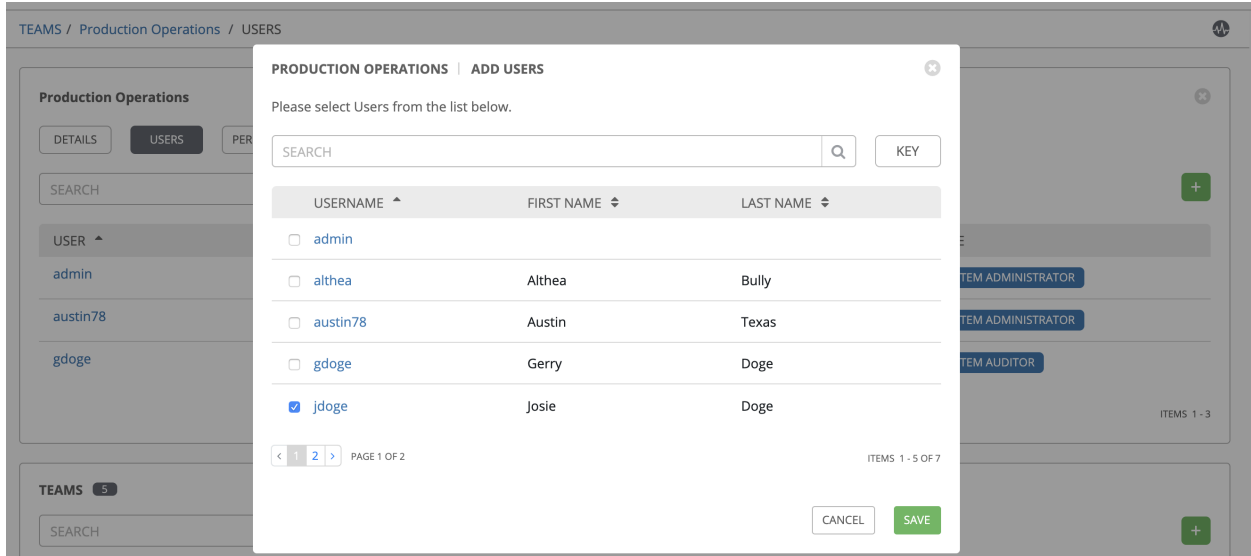
ITEMS 1 - 4

Add a User

In order to add a user to a team, the user must already be created in Tower. 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 **Permissions** tab . To add existing users to the Team:

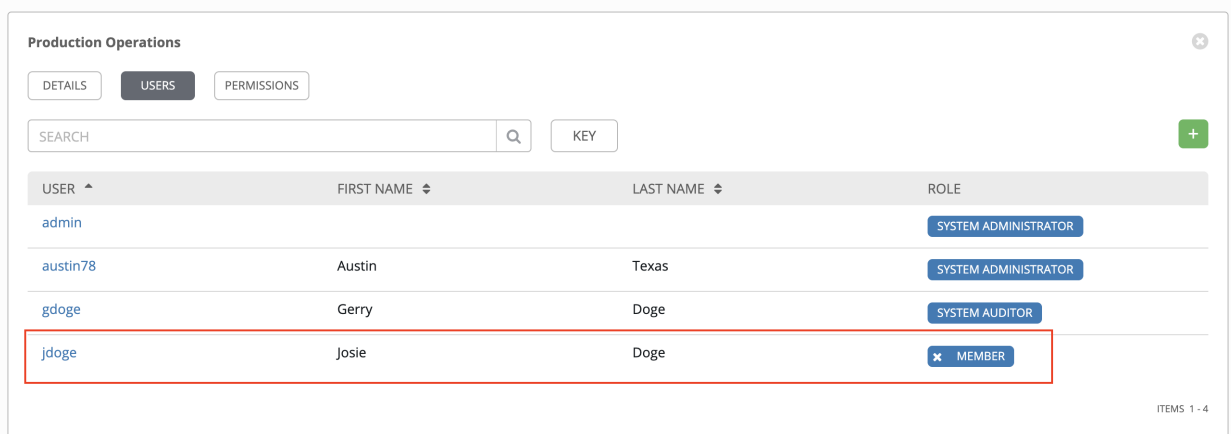


1. Click the  button.
2. Select one or more users from the list of available users by clicking the checkbox next to the user(s) to add them as members of the team.



In this example, one user has been selected to be added to this team.

4. Click the **Save** button when done.



9.1.2 Teams - Permissions

Selecting the **Permissions** view displays a list of the permissions that are currently available for this Team. The permissions list may be sorted and searched by **Name**, **Inventory**, **Project** or **Permission** type.

TEAMS / Production Operations / PERMISSIONS

Production Operations

DETAILS USERS PERMISSIONS

SEARCH Q KEY +

NAME	TYPE	ROLE	ACTIONS
Demo Project	Project	Use	✕
Demo Job Template	Job Template	Execute	✕
King PLC	Inventory	Ad Hoc	✕

ITEMS 1 - 3


The set of privileges assigned to Teams that provide the ability to read, modify, and administer projects, inventories, and other Tower 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  button, which opens the Add Permissions Wizard.

TEAMS / Production Operations / PERMISSIONS

Production Operations

DETAILS USERS PERMISSIONS

PRODUCTION OPERATIONS | ADD PERMISSIONS

1 Please select resources from the lists below.

JOB TEMPLATES WORKFLOW TEMPLATES PROJECTS INVENTORIES CREDENTIALS ORGANIZATIONS

SEARCH Q KEY +

NAME ^

- Basic Job Template
- Demo Job Template


ITEMS 1 - 2

CANCEL SAVE

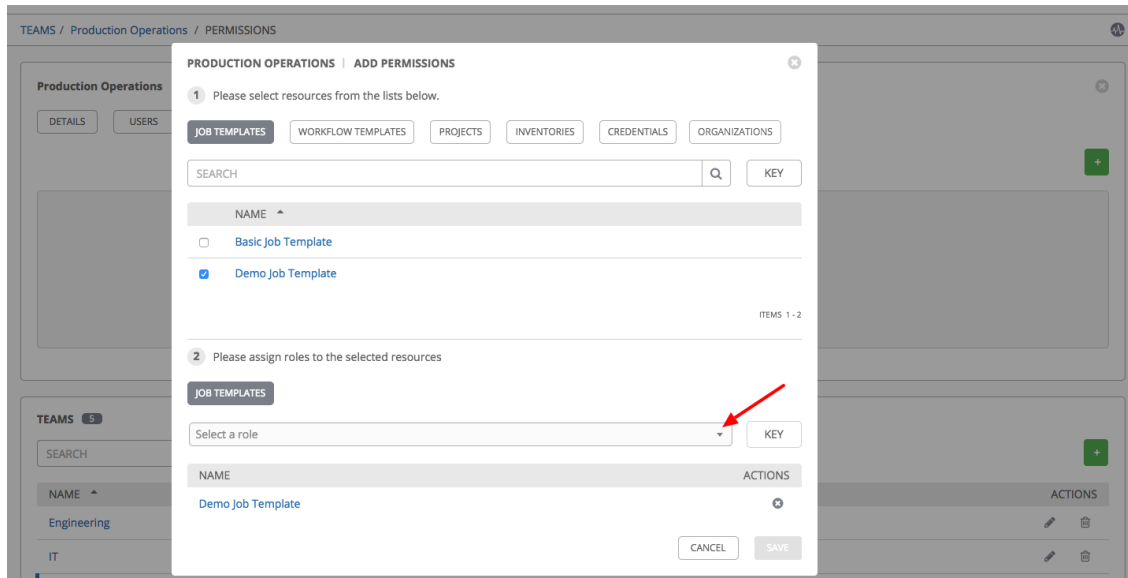
2. Click to select the Tower object for which the user will have access:
 - **Job Templates**. This is the default tab displayed in the Add Permissions Wizard.
 - **Workflow Templates**
 - **Projects**
 - **Inventories**
 - **Credentials**

• Organizations



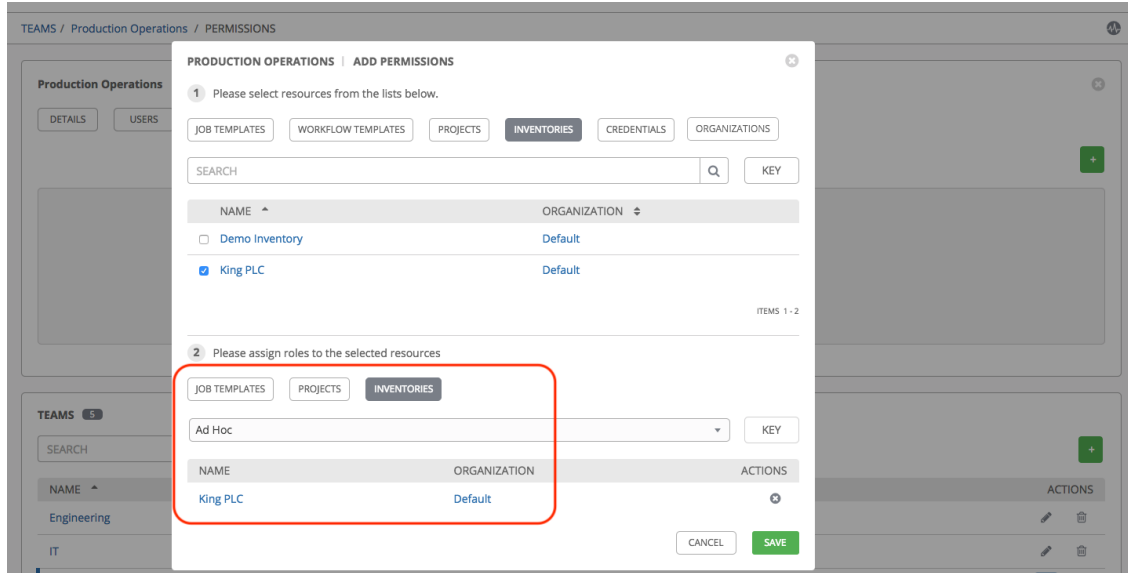
Note: You can assign different roles to different resources all at once to avoid having to click the  button. To do so, simply go from one tab to another after making your selections without saving.

3. Perform the following steps to assign the user specific roles for each type of resource:
 - a. In the desired tab, click the checkbox beside the name of the resource to select it.
The dialog expands to allow you to select the role for the resource you chose.
 - b. Select the role from the drop-down menu list provided. Only some roles are applicable to certain resources.

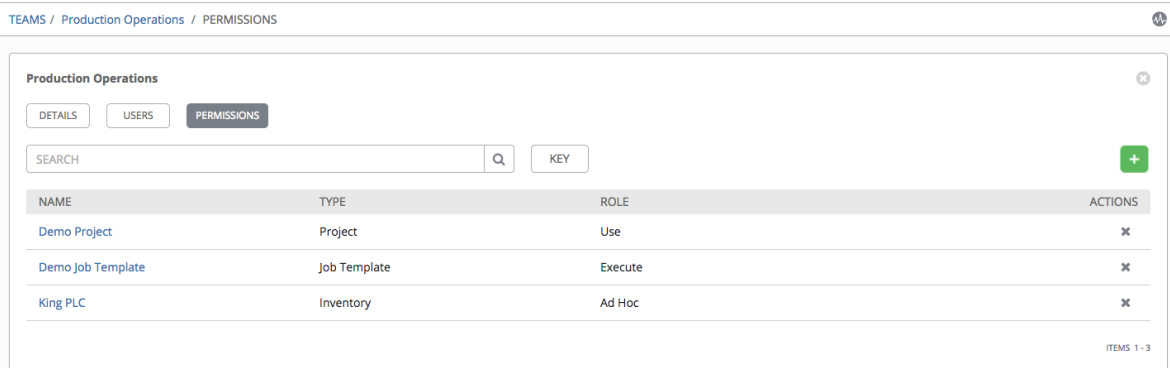


Tip: Use the **Key** button to display the help text for each of the roles applicable to the resource selected.

- c. Review your role assignments for each of the Tower objects by clicking on their respective buttons in the expanded section 2 of the Add Permissions Wizard.



- d. 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.



To remove Permissions for a particular User, click the Disassociate (✕) button under **Actions**. This launches a **Remove Role** 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.

CREDENTIALS

Credentials are utilized by Tower 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 Tower.

Note: Tower encrypts passwords and key information in the Tower database and never makes secret information visible via the API. See [Ansible Tower Administration Guide](#) for details.


10.1 Understanding How Credentials Work

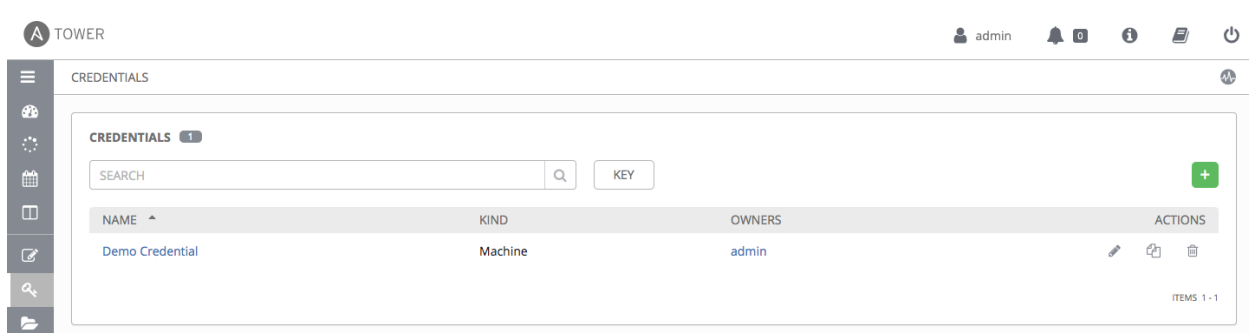
Ansible Tower uses SSH to connect to remote hosts (or the Windows equivalent). In order to pass the key from Tower to SSH, the key must be decrypted before it can be written a named pipe. Tower then uses that pipe to send the key to SSH (so that it is never written to disk).

If passwords are used, Ansible Tower 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



Access the Credentials page by clicking the Credentials () icon from the left navigation bar. The Credentials page displays a search-able list of all available Credentials and can be sorted by **Name**.



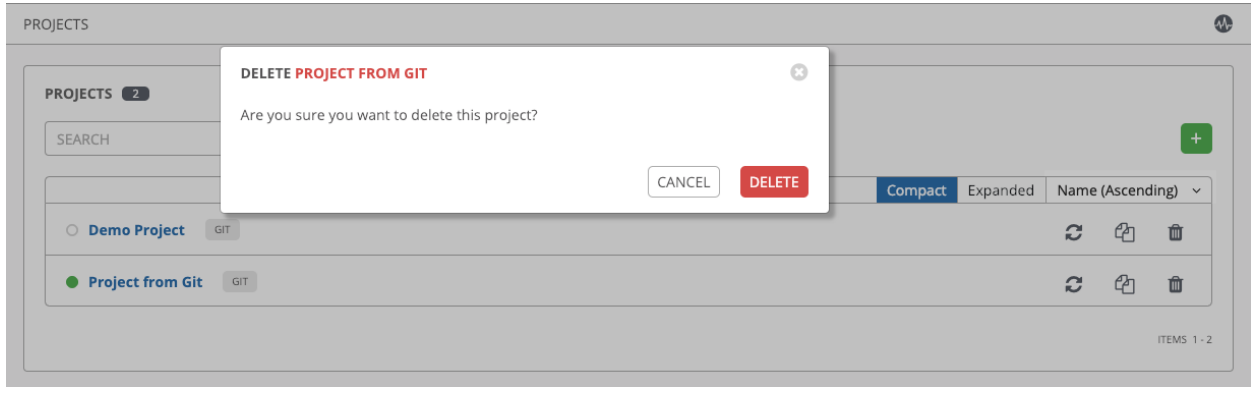
The screenshot shows the Ansible Tower interface for the 'CREDENTIALS' page. At the top, there's a search bar and a 'KEY' button. Below that is a table with the following data:

NAME	KIND	OWNERS	ACTIONS
Demo Credential	Machine	admin	[edit] [copy] [delete]

At the bottom right of the table, it says 'ITEMS 1 - 1'. The left navigation bar is visible on the left side of the screenshot.

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:



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.

Demo Credential
✕

DETAILS

PERMISSIONS

*** NAME** ⓘ

DESCRIPTION ⓘ

ORGANIZATION

*** CREDENTIAL TYPE** ⓘ

TYPE DETAILS

USERNAME

PASSWORD Prompt on launch

SSH PRIVATE KEY HINT: Drag and drop private file on the field below.

SIGNED SSH CERTIFICATE HINT: Drag and drop private file on the field below.

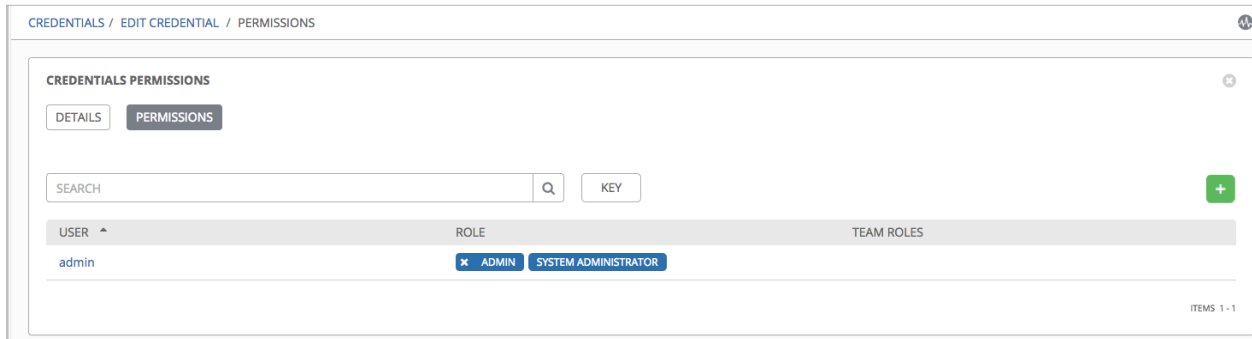
PRIVATE KEY PASSPHRASE Prompt on launch

PRIVILEGE ESCALATION METHOD ⓘ



PRIVILEGE ESCALATION USERNAME

PRIVILEGE ESCALATION PASSWORD Prompt on launch

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




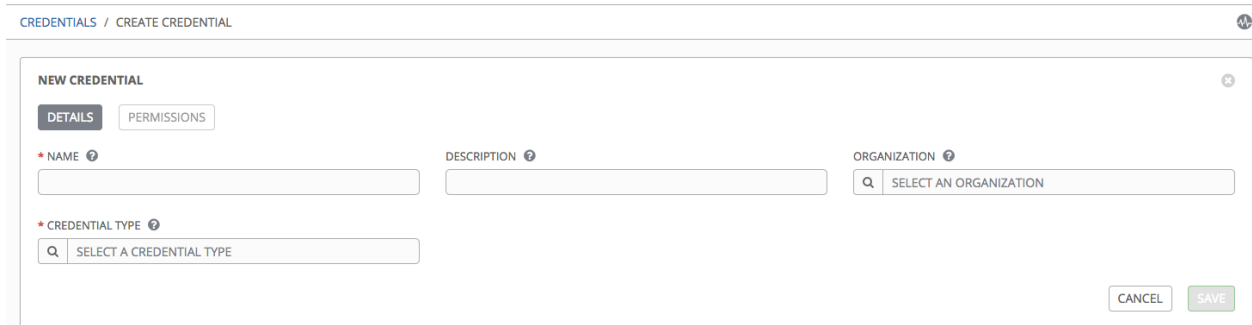
Note: A credential with roles associated will retain them even after the credential has been reassigned to another organization.

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

10.3 Add a New Credential

To create a new credential:

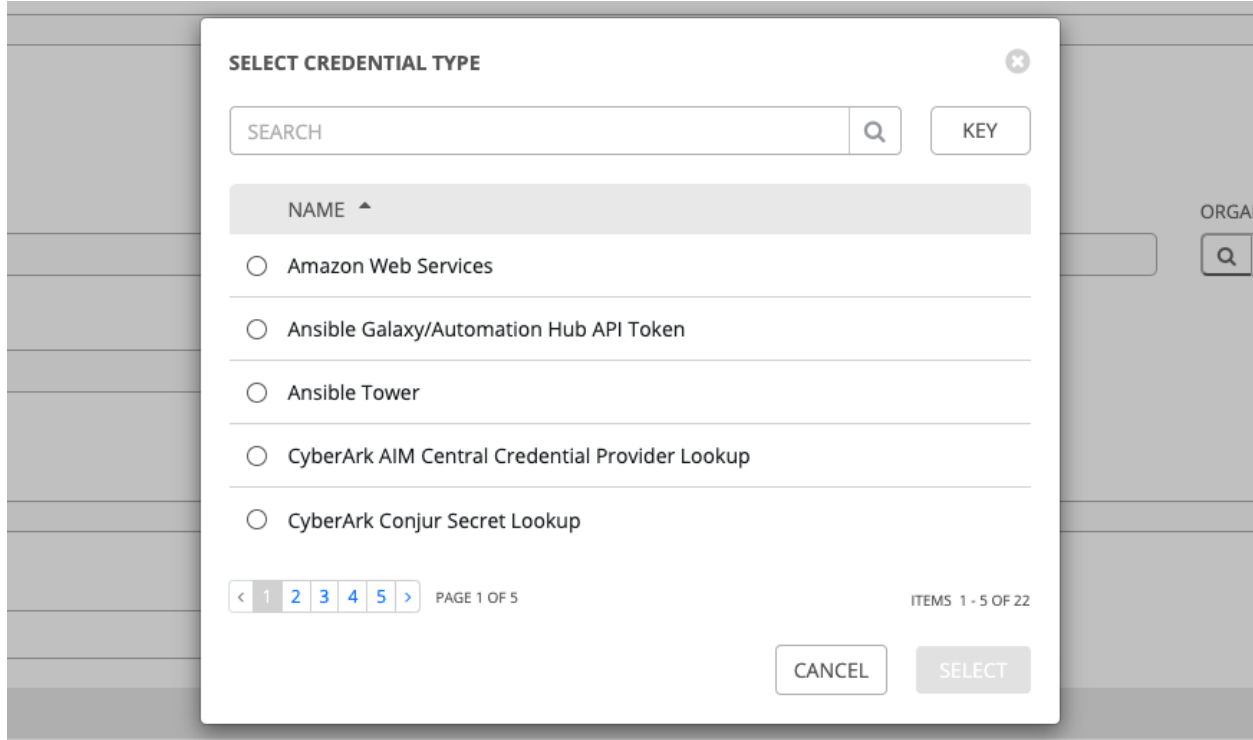
1. Click the  button located in the upper right corner of the **Credentials** screen.



2. Enter the name for your new credential in the **Name** field.
3. Optionally 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.



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 Ansible Tower:

- *Amazon Web Services*
- *Ansible Galaxy/Automation Hub API Token*
- *Ansible Tower*
- *GitHub Personal Access Token*
- *GitLab Personal Access Token*
- *Google Compute Engine*
- *Insights*
- *Machine*
- *Microsoft Azure Resource Manager*
- *Network*
- *OpenShift or Kubernetes API Bearer Token*
- *OpenStack*

- *Red Hat Satellite 6*
- *Red Hat Virtualization*
- *Source Control*
- *Vault*
- *VMware vCenter*

The credential types associated with CyberArk, HashiCorp Vault, and Microsoft Azure Key Management System (KMS) are part of the credential plugins capability. 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.

Tower 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
```

The screenshot shows the 'NEW CREDENTIAL' form with the following fields and values:

- NAME:** (empty)
- DESCRIPTION:** (empty)
- ORGANIZATION:** SELECT AN ORGANIZATION
- CREDENTIAL TYPE:** Amazon Web Services (indicated by a red arrow)
- ACCESS KEY:** (empty)
- SECRET KEY:** (empty)
- STS TOKEN:** (empty)

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

Ansible Tower version 2.4.0 introduced 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 Tower 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 Tower to access Galaxy or use a collection published on a local Automation Hub. See *Using Collections in Tower* for detail. Entering the Galaxy server URL is the only required value on this screen.

The screenshot shows the 'NEW CREDENTIAL' form with the 'DETAILS' tab selected. The 'CREDENTIAL TYPE' dropdown is set to 'Ansible Galaxy/Automation Hub API Token', indicated by a red arrow. The form includes fields for NAME, DESCRIPTION, ORGANIZATION (with a search dropdown), GALAXY SERVER URL, AUTH SERVER URL, and API TOKEN. There are CANCEL and SAVE buttons at the bottom right.

10.4.3 Ansible Tower

Selecting this credential allows you to access another Tower instance.

The screenshot shows the 'NEW CREDENTIAL' form with the 'DETAILS' tab selected. The 'CREDENTIAL TYPE' dropdown is set to 'Ansible Tower', indicated by a red arrow. The form includes fields for NAME, DESCRIPTION, ORGANIZATION (with a search dropdown), ANSIBLE TOWER HOSTNAME, USERNAME, PASSWORD, OAUTH TOKEN, and an OPTIONS section with a 'Verify SSL' checkbox. There are CANCEL and SAVE buttons at the bottom right.

Ansible Tower credentials have the following inputs that are required:

- **Ansible Tower Hostname:** The base URL or IP address of the other Tower 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 Tower.

10.4.4 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.

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.

10.4.5 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.

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.

10.4.6 Google Compute Engine

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

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

```
GCE_EMAIL
GCE_PROJECT
GCE_CREDENTIALS_FILE_PATH
```

The screenshot shows the 'NEW CREDENTIAL' form with the following fields and values:

- NAME:** New Credential
- DESCRIPTION:** (empty)
- ORGANIZATION:** SELECT AN ORGANIZATION
- CREDENTIAL TYPE:** Google Compute Engine (indicated by a red arrow)
- SERVICE ACCOUNT EMAIL ADDRESS:** (empty)
- PROJECT:** (empty)
- SERVICE ACCOUNT JSON FILE:** CHOOSE A FILE
- RSA PRIVATE KEY:** (empty)

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.

10.4.7 Insights

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

The screenshot shows the 'NEW CREDENTIAL' form with the following fields and values:

- NAME:** (empty)
- DESCRIPTION:** (empty)
- ORGANIZATION:** SELECT AN ORGANIZATION
- CREDENTIAL TYPE:** Insights (indicated by a red arrow)
- USERNAME:** (empty)
- PASSWORD:** (empty)

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

10.4.8 Machine

Machine credentials enable Tower 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 Tower 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.

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 Tower database, if entered. Alternatively, you can configure Tower 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 Tower database, if entered. Alternatively, you can configure Tower 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](#))
- **pmsrun:** 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 Tower database, if entered. Alternatively, you may configure Tower 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 Tower 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"**.

10.4.9 Microsoft Azure Resource Manager

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

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
```

10.4.10 Network

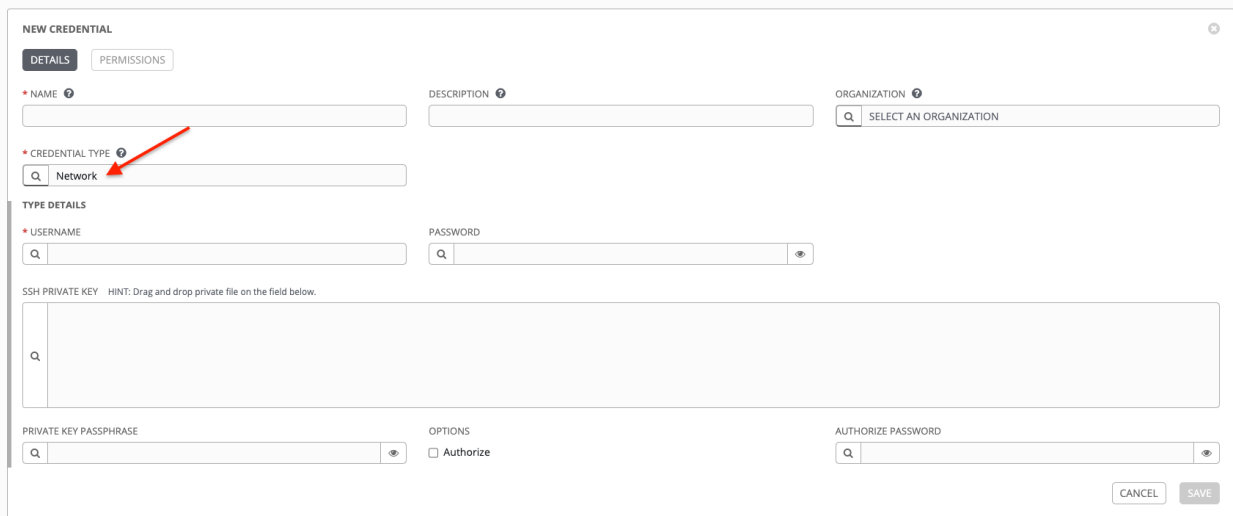
Select the Network credential type **only** if you are using a *local* connection with *provider* to use Ansible networking modules to connect to and manage networking devices. When connecting to network devices, the credential type must match the connection type:

- 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](#).

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

```
ANSIBLE_NET_USERNAME
ANSIBLE_NET_PASSWORD
```



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](#).

10.4.11 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 [Execution Environments](#).

The screenshot shows the 'NEW CREDENTIAL' form with the following details:

- NAME:** Container groups credential
- DESCRIPTION:** (empty)
- ORGANIZATION:** SELECT AN ORGANIZATION
- CREDENTIAL TYPE:** OpenShift or Kubernetes API Bearer Token (indicated by a red arrow)
- TYPE DETAILS:**
 - OPENSIFT OR KUBERNETES API ENDPOINT:** (empty)
 - API AUTHENTICATION BEARER TOKEN:** (empty)
 - OPTIONS:** Verify SSL (checked)
- CERTIFICATE AUTHORITY DATA:** (empty, with a hint: Drag and drop private file on the field below.)

This example `service` account can be used to obtain these credentials:

```
$ NAMESPACE=my-namespace
$ kubectl -n $NAMESPACE create -f https://docs.ansible.com/ansible-tower/latest/html/
  ↳ userguide/_downloads/a49648106e2c857f9b0a41e91d063c47/service-account.yml
$ SECRET_NAME=$(kubectl -n $NAMESPACE get sa awx -o json | jq -r '.secrets[0].name')
$ kubectl -n $NAMESPACE get secret $SECRET_NAME -o json | jq -r '.data["token"] |
  ↳ @base64d' # The token
$ kubectl -n $NAMESPACE get secret $SECRET_NAME -o json | jq -r '.data["ca.crt"] |
  ↳ @base64d' # The CA data
```

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 allow Tower 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

10.4.12 OpenStack

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

The screenshot shows the 'NEW CREDENTIAL' form with the following fields and options:

- DETAILS** (selected) / PERMISSIONS
- * NAME**: Text input field.
- DESCRIPTION**: Text input field.
- ORGANIZATION**: Dropdown menu with 'SELECT AN ORGANIZATION'.
- * CREDENTIAL TYPE**: Dropdown menu with 'OpenStack' selected (indicated by a red arrow).
- TYPE DETAILS**:
 - * USERNAME**: Text input field.
 - * PASSWORD (API KEY)**: Text input field with a visibility toggle.
 - * HOST (AUTHENTICATION URL)**: Text input field.
 - * PROJECT (TENANT NAME)**: Text input field.
 - PROJECT (DOMAIN NAME)**: Text input field.
 - DOMAIN NAME**: Text input field.
- OPTIONS**:
 - Verify SSL
- CANCEL** and **SAVE** buttons at the bottom right.

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.

10.4.13 Red Hat Satellite 6

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

Tower 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
```

The screenshot shows the 'NEW CREDENTIAL' form with the following fields and options:

- DETAILS** (selected) / PERMISSIONS
- * NAME**: Text input field.
- DESCRIPTION**: Text input field.
- ORGANIZATION**: Dropdown menu with 'SELECT AN ORGANIZATION'.
- * CREDENTIAL TYPE**: Dropdown menu with 'Red Hat Satellite 6' selected (indicated by a red arrow).
- TYPE DETAILS**:
 - * SATELLITE 6 URL**: Text input field.
 - * USERNAME**: Text input field.
 - * PASSWORD**: Text input field with a visibility toggle.
- CANCEL** and **SAVE** buttons at the bottom right.

Satellite credentials have the following inputs that are required:

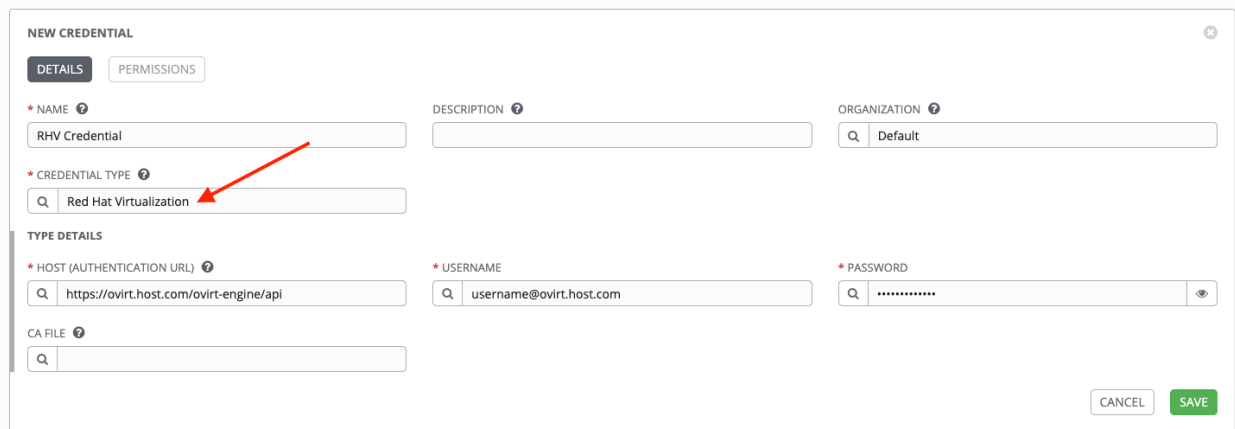
- **Satellite 6 URL:** The Satellite 6 URL or IP address to connect to.
- **Username:** The username to use to connect to Satellite 6.
- **Password:** The password to use to connect to Satellite 6.

10.4.14 Red Hat Virtualization

This credential allows Tower to access Ansible’s `ovirt4.py` dynamic inventory plugin, which is managed by Red Hat Virtualization (RHV).

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

```
OVIRT_URL
OVIRT_USERNAME
OVIRT_PASSWORD
```



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)

10.4.15 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, Subversion, or Mercurial.

The screenshot shows the 'NEW CREDENTIAL' form with the following fields and options:

- NAME:** Text input field.
- DESCRIPTION:** Text input field.
- ORGANIZATION:** Dropdown menu with a search icon and the text 'SELECT AN ORGANIZATION'.
- CREDENTIAL TYPE:** Dropdown menu with 'Source Control' selected, highlighted by a red arrow.
- TYPE DETAILS:**
 - USERNAME:** Text input field.
 - PASSWORD:** Text input field with a visibility toggle (eye icon).
 - SCM PRIVATE KEY:** Large text area with a hint: 'HINT: Drag and drop private file on the field below.' and a search icon.
 - PRIVATE KEY PASSPHRASE:** Text input field with a visibility toggle (eye icon).
- Buttons:** 'CANCEL' and 'SAVE' buttons at the bottom right.

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 Authentication) enabled on your account, you will need to use your Personal Access Token in the password field rather than your account password.

10.4.16 Vault

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

The screenshot shows the 'NEW CREDENTIAL' form with the following fields and options:

- NAME:** Text input field.
- DESCRIPTION:** Text input field.
- ORGANIZATION:** Dropdown menu with a search icon and the text 'SELECT AN ORGANIZATION'.
- CREDENTIAL TYPE:** Dropdown menu with 'Vault' selected, highlighted by a red arrow.
- TYPE DETAILS:**
 - VAULT PASSWORD:** Text input field with a visibility toggle (eye icon).
 - Prompt on launch:** Checkbox.
 - VAULT IDENTIFIER:** Text input field with a search icon.
- Buttons:** 'CANCEL' and 'SAVE' buttons at the bottom right.

Vault credentials require the **Vault Password** and an optional **Vault Identifier** if applying multi-Vault credentialing. For more information on Ansible Tower Multi-Vault support, refer to the [Multi-Vault Credentials](#) section of the *Ansible Tower Administration Guide*.

You may configure Tower 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

10.4.17 VMware vCenter

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

Tower 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
```

The screenshot shows the 'NEW CREDENTIAL' form with the following fields and values:

- NAME:** (empty)
- DESCRIPTION:** (empty)
- ORGANIZATION:** SELECT AN ORGANIZATION
- CREDENTIAL TYPE:** VMware vCenter (indicated by a red arrow)
- TYPE DETAILS:**
 - vCENTER HOST:** (empty)
 - USERNAME:** (empty)
 - PASSWORD:** (empty)

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.

CUSTOM CREDENTIAL TYPES

As a Tower 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. In Ansible Tower 3.2 and later, 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

In 3.8, Tower introduced a new “managed by Tower” credential type of `kind=galaxy`, which 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 Tower API reflects an ordered list of these credentials at the Organization level:

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

Installations of Ansible Tower 3.8 migrates existing Galaxy-oriented setting values in such a way that post-upgrade, proper credentials are created and attached to every Organization in the Tower install. After upgrading to 3.8, 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`.

Tower 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 Tower User Interface. The Organization’s Add and Edit windows have an optional **Credential** lookup field for credentials of kind=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 in Tower*.

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://tower.example.org/api/v2/credentials/?credential_type__namespace=aws"
```

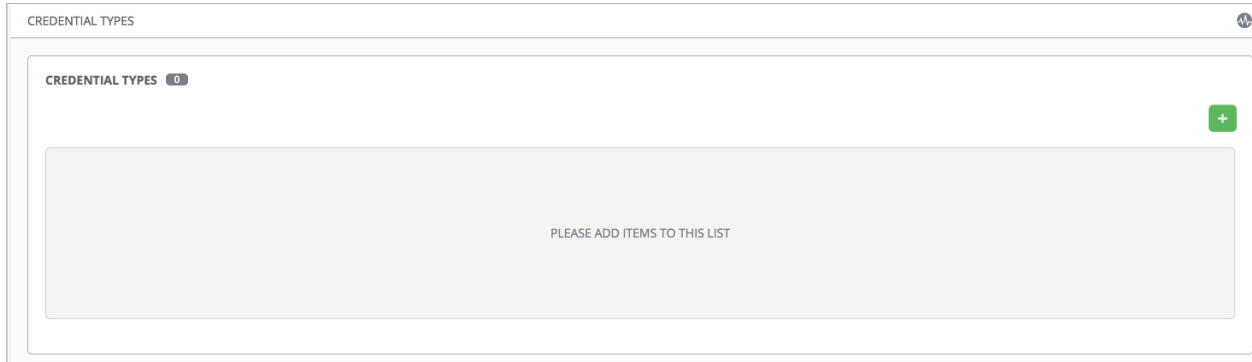
In the V2 CredentialType model, the relationships are defined as follows:

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

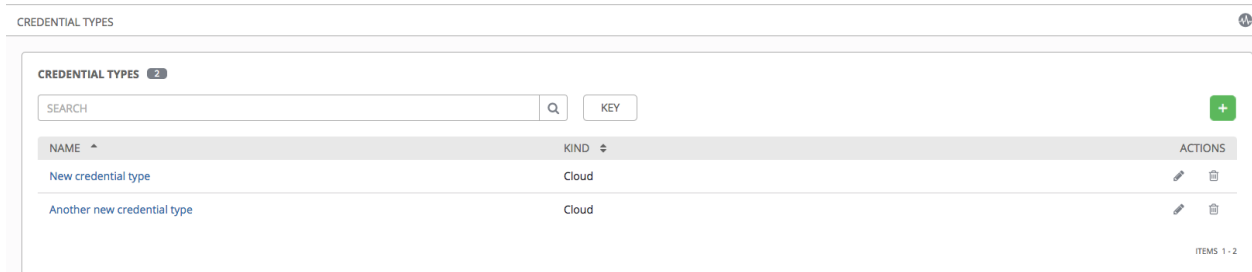
11.3 Getting Started with Credential Types




Access the Credentials from clicking the Credential Types () icon 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:



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




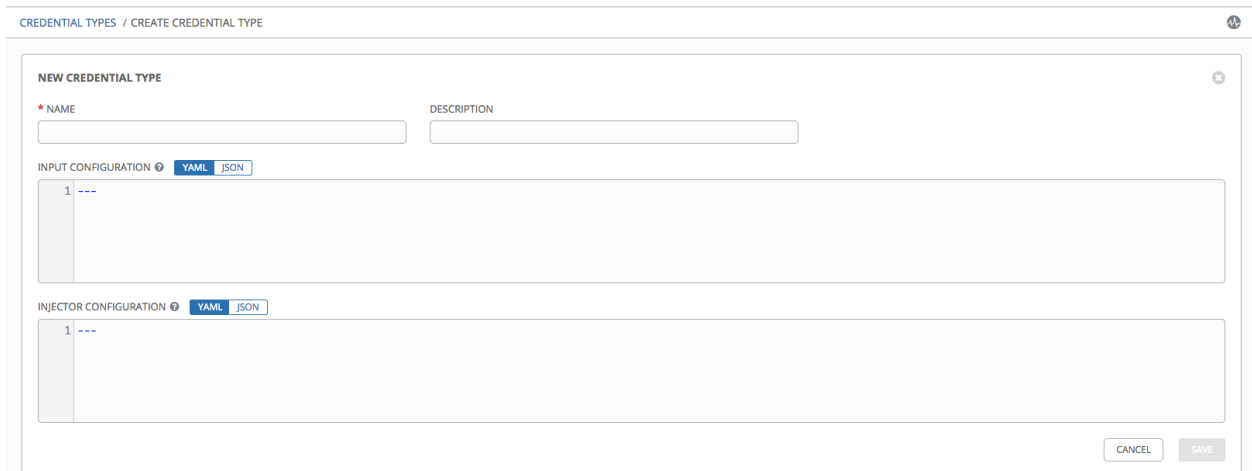
To view more information about a credential type, click on its name or the Edit () button from the **Actions** column.

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.

11.4 Create a New Credential Type

To create a new credential type:

1. Click the  button located in the upper right corner of 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",
    "label": "API Token",
    "help_text": "User-facing short text describing the field.",
    # required - a unique name used to
    # reference the field value
    # required - a unique label for the
    # field
  }]
```

(continues on next page)

(continued from previous page)

```

"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, Tower 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:

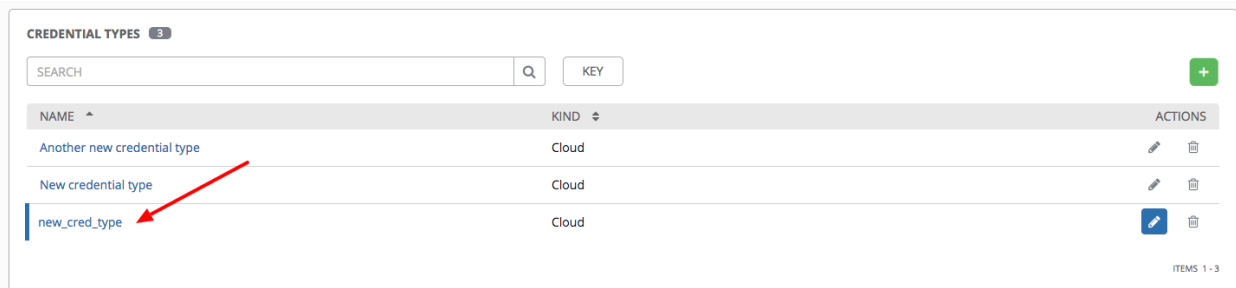
Inputs



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

Injectors

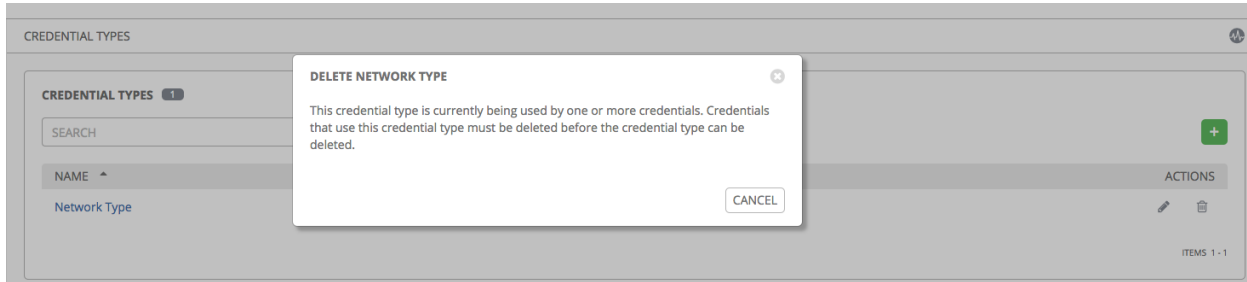
```
{
  "file": {
    "template.cert_file": "[mycert]\n{{ cert }}",
    "template.key_file": "[mykey]\n{{ key }}"
  },
  "env": {
    "MY_CERT_INI_FILE": "{{ tower.filename.cert_file }}",
    "MY_KEY_INI_FILE": "{{ tower.filename.key_file }}"
  }
}
```

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:

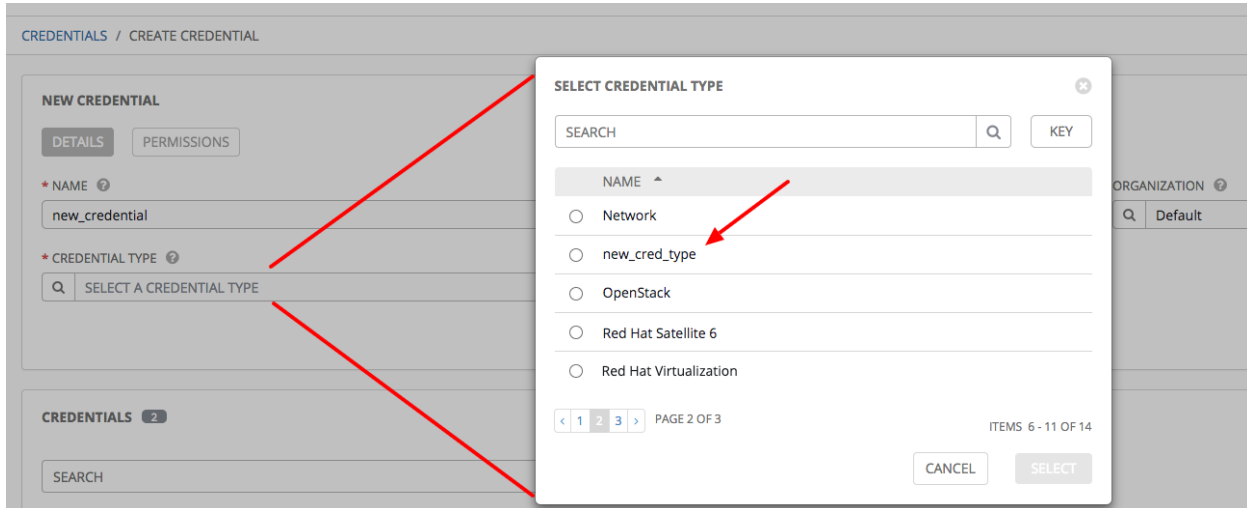


Click  to modify or  to remove the credential type options under the Actions column.

Note: If deleting a credential type that is being used by a credential, 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:



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



For details on how to create a new credential, see [Credentials](#).

SECRET MANAGEMENT SYSTEM

Users and admins upload machine and cloud credentials to Tower so that it 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) in Tower 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 Tower directly. Ansible Tower provides a secret management system that include integrations for:

- CyberArk Application Identity Manager (AIM)
- CyberArk Conjur
- HashiCorp Vault Key-Value Store (KV)
- HashiCorp Vault SSH Secrets Engine
- Microsoft Azure Key Management System (KMS)

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

12.1 Configure and link secret lookups

When configuring Tower 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.


Tower provides a credential plugin interface for developers, integrators, admins, and power-users with the ability to add new external credential types to Tower so it can be extended to support other secret management systems. For more detail, see the [development docs for credential plugins](#).

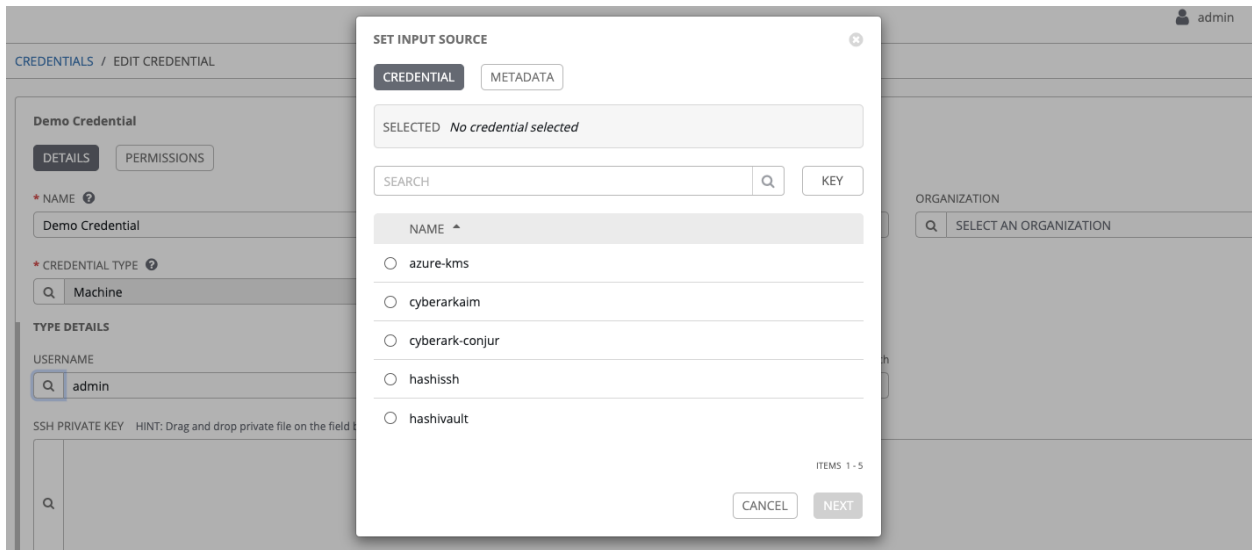
Use the Ansible Tower User Interface to configure and use each of the supported 3-party secret management systems.

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**:

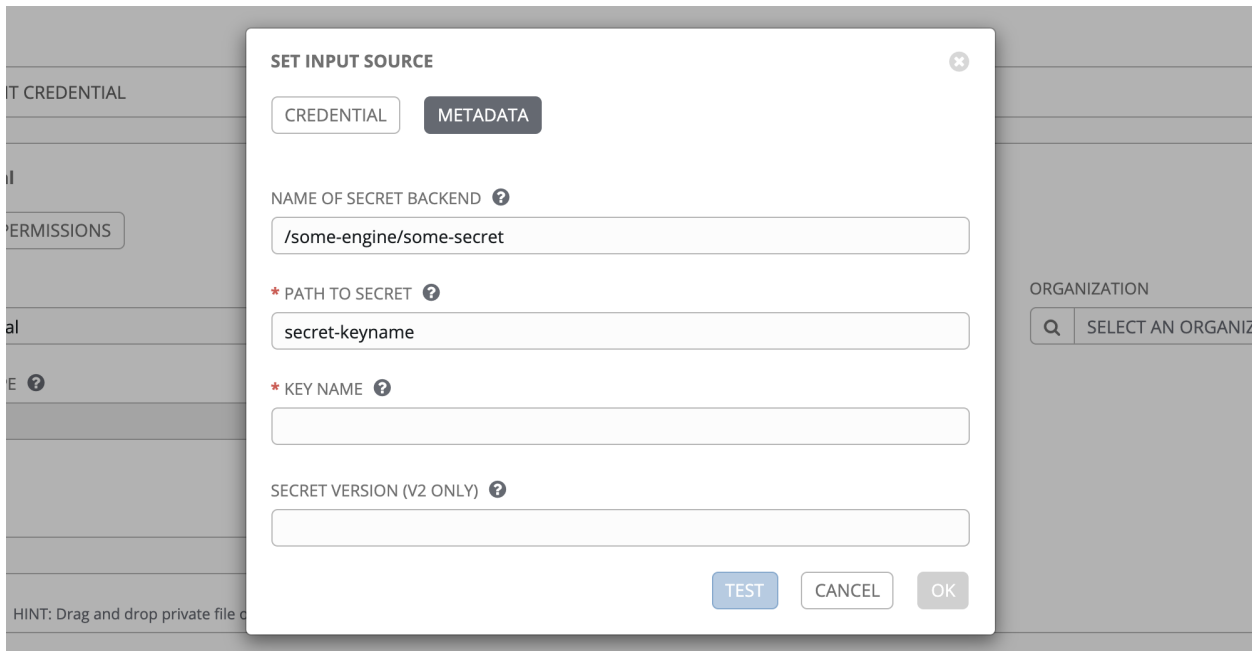
- *CyberArk AIM Credential Provider Lookup*
- *CyberArk Conjur Secret Lookup*
- *HashiCorp Vault Secret Lookup*
- *HashiCorp Vault Signed SSH*

- *Microsoft Azure Key Vault*

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.



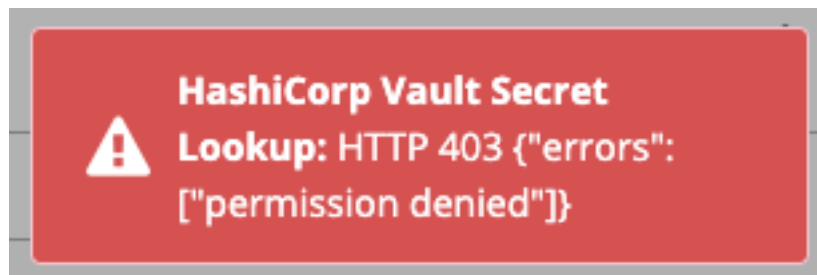
4. Select the credential you want to link to, and click **Next**. This takes you to the Metadata tab of the input source. This example shows the Metadata prompt for HashiVault Secret Lookup.



The metadata required depends on the input source selected:

Input Source	Metadata	Description
CyberArk AIM	Object Query (Required)	Lookup query for the object.
	Object Query Format	Select <code>Exact</code> for a specific secret name, or <code>Regexp`</code> 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	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	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 (e.g., <code>/path/username</code>).
	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	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 (e.g., <code>/path/username</code>).
	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., <code>ec2-user</code>).
Azure KMS	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.

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



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, Tower 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 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 allow Tower 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.

The screenshot shows the configuration page for a credential provider named 'cyberarkaim'. The interface includes the following elements:

- Navigation:** 'DETAILS' and 'PERMISSIONS' tabs.
- Basic Information:**
 - NAME:** 'cyberarkaim'
 - DESCRIPTION:** (empty)
 - ORGANIZATION:** 'SELECT AN ORGANIZATION' (dropdown)
- CREDENTIAL TYPE:** 'CyberArk AIM Central Credential Provider Lookup' (dropdown)
- TYPE DETAILS:**
 - CYBERARK AIM URL:** 'https://mycyberark.com'
 - APPLICATION ID:** 'ENCRYPTED' (with refresh and copy icons)
- CLIENT KEY:** A large text area with a hint: 'HINT: Drag and drop private file on the field below.'
- CLIENT CERTIFICATE:** A large text area with a hint: 'HINT: Drag and drop private file on the field below.'
- OPTIONS:** A checkbox labeled 'Verify SSL Certificates' which is checked.
- Actions:** 'TEST', 'CANCEL', and 'SAVE' buttons at the bottom right.

12.1.2 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.

The screenshot shows the configuration page for a 'cyberark-conjur' credential. The interface includes tabs for 'DETAILS' and 'PERMISSIONS'. The 'DETAILS' tab is active, showing the following fields:

- * NAME**: cyberark-conjur
- DESCRIPTION**: (empty)
- ORGANIZATION**: (dropdown menu with 'SELECT AN ORGANIZATION' text)
- * CREDENTIAL TYPE**: CyberArk Conjur Secret Lookup
- TYPE DETAILS** section:
 - * CONJUR URL**: https://eval.conjur.org
 - * API KEY**: ENCRYPTED (with a copy icon)
 - * ACCOUNT**: email@address.com
 - * USERNAME**: ansibletest
 - PUBLIC KEY CERTIFICATE**: (empty text area)

At the bottom right, there are three buttons: TEST (blue), CANCEL (grey), and SAVE (green).

12.1.3 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.

The screenshot shows the 'hashivault' configuration page in Ansible Tower. The 'DETAILS' tab is active. The form contains the following fields and values:

- NAME:** hashivault
- DESCRIPTION:** (empty)
- ORGANIZATION:** SELECT AN ORGANIZATION
- CREDENTIAL TYPE:** HashiCorp Vault Secret Lookup
- TYPE DETAILS:**
 - SERVER URL:** https://http://vault-port:1234.services.k8s.tower-qa.testing.ansible
 - TOKEN:** ENCRYPTED
 - CA CERTIFICATE:** (empty text area)
 - APPROLE ROLE_ID:** (empty)
 - APPROLE SECRET_ID:** (empty)
 - PATH TO APPROLE AUTH:** approve
 - API VERSION:** v2

Buttons at the bottom right: TEST, CANCEL, SAVE.

12.1.4 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 `/approve`

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.

12.1.5 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.

APPLICATIONS


Creating and configuring token-based authentication for external applications is available starting in Ansible Tower 3.3. This makes it easier for external applications such as ServiceNow and Jenkins to integrate with Ansible Tower. 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”. In Tower, 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.

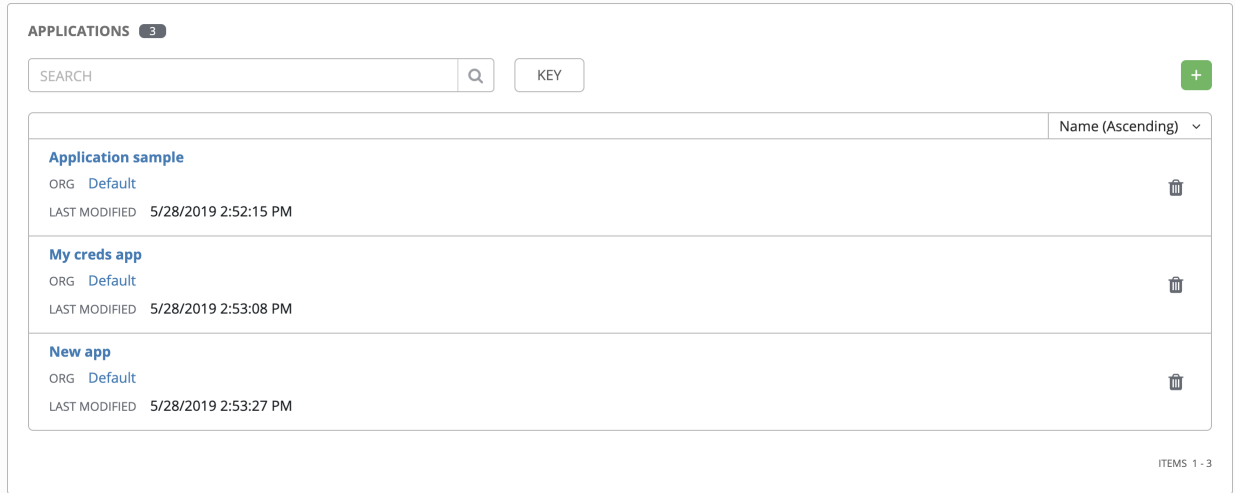
Having these Tower-issued tokens associated to an application resource gives you the ability to manage all tokens issued for a particular application more easily. By separating token issuance under Applications, you can revoke all tokens based on the Application without having to revoke all tokens in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

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

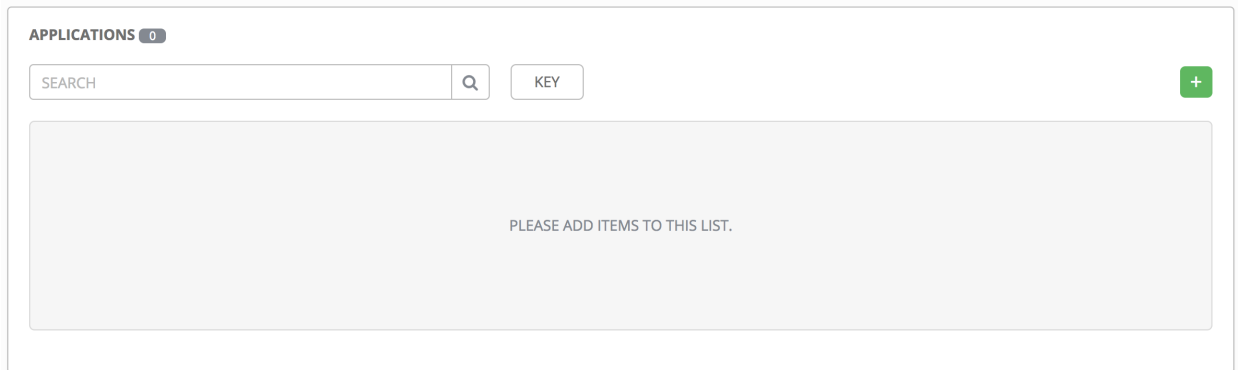
- external applications can obtain a token from Tower for users, using their credentials
- compartmentalized tokens issued for a particular application, allows those tokens to be easily managed (revoke all tokens associated with that application, for example)

13.1 Getting Started with Applications

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



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




13.2 Create a new application

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

1. In the Ansible Tower User Interface, click the Applications () icon from the left navigation bar.

The Applications window opens.

2. Click the  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 *Ansible Tower 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

13.2.1 Applications - Tokens

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

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

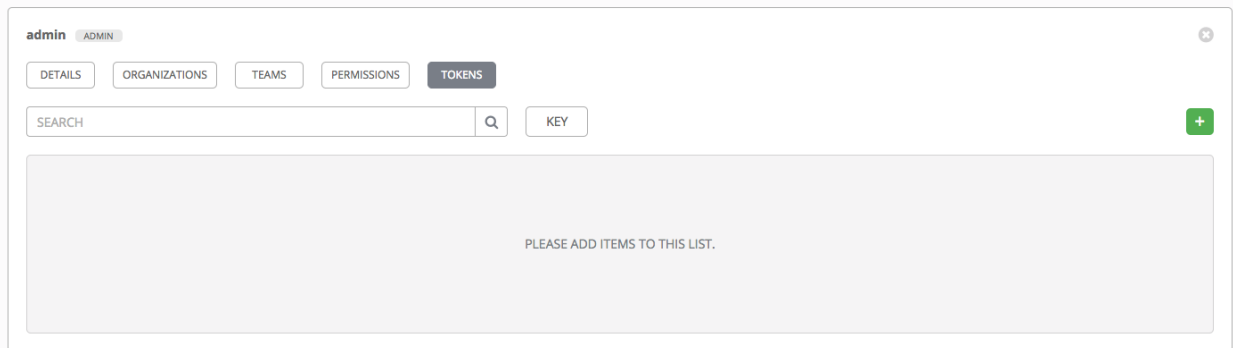
Tokens are added through the Users screen and can be associated with an application at that time. Specifying an application can be performed directly in the User's token settings. You can create a token for *your* user in the Tokens configuration tab, meaning only you can create and see your tokens in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


1. Access the Users list view by clicking the Users () icon 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 *Ansible Tower 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:



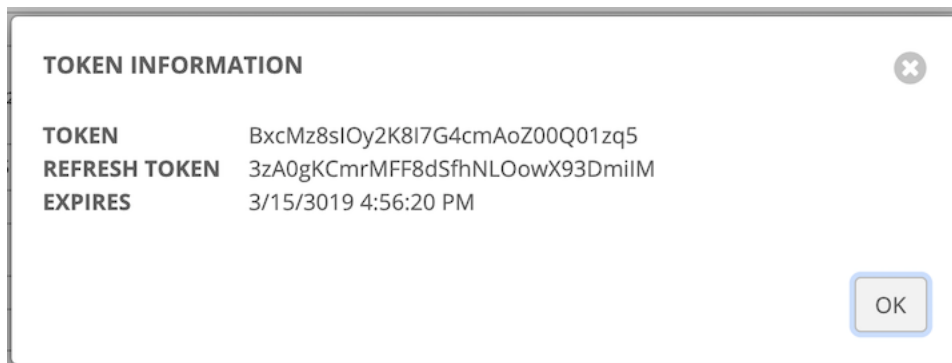
3. Click the  button, which opens the Create Token window.

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.

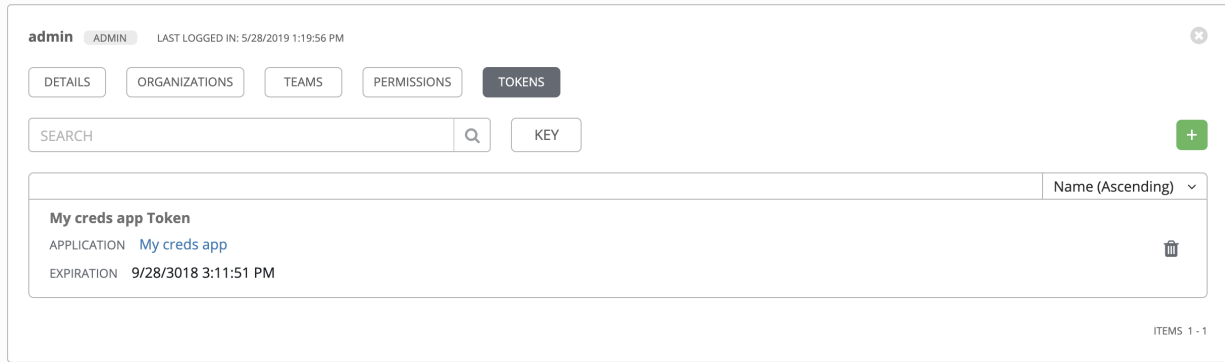
5. When done, click **Save** or **Cancel** to abandon your changes.

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

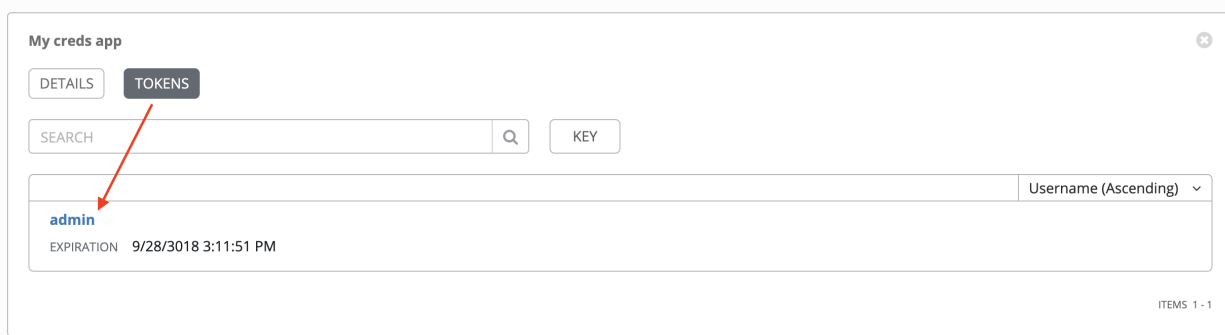


Note: This is the only time the token value and associated refresh token value will ever be shown.

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



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:



PROJECTS

A **Project** is a logical collection of Ansible playbooks, represented in Tower.

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

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




This menu 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 expand to see more information. You can sort this list by various criteria, and perform a search to filter the projects of interest.

The screenshot shows the 'PROJECTS' section with a search bar and a 'KEY' button. The view is set to 'Compact'. Two projects are listed: 'Demo Project' (status: GIT) and 'Project from Git' (status: GIT). Each project has icons for refresh, copy, and delete. A '+ ' button is in the top right. The bottom right corner shows 'ITEMS 1 - 2'.

The screenshot shows the 'PROJECTS' section with a search bar and a 'KEY' button. The view is set to 'Expanded'. Two projects are listed with detailed information:

REVISION	ORGANIZATION	LAST MODIFIED	LAST USED
347e44f	Default	10/28/2019 11:53:19 AM	11/4/2019 2:14:03 PM
818fd82	Default	11/12/2019 9:56:45 PM	11/12/2019 9:53:36

The 'Project from Git' row also includes a 'DESCRIPTION' column with the value 'Shows the description field'. A 'SORT BY' dropdown menu is open, showing options: Name (Ascending), Name (Descending), Modified (Ascending), Modified (Descending), Last Used (Ascending), Last Used (Descending), Organization (Ascending), and Organization (Descending). The bottom right corner shows 'ITEMS 1 - 2'.

For each project listed, you can get the latest SCM revision () , copy the project attributes () , or delete () the project, using the respective icons next to each project. Starting with Ansible Tower 3.7, projects are allowed to

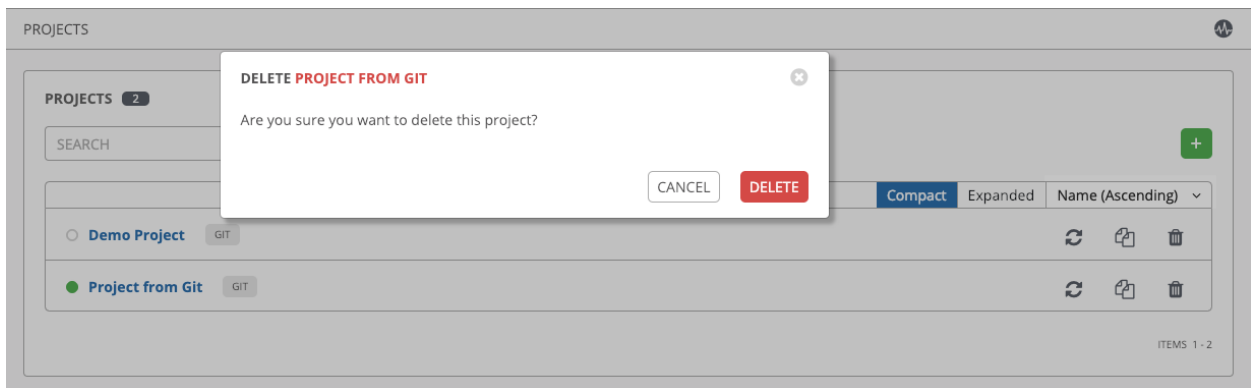
be updated while a related job is running. In cases where you have a big project (around 10 GB), disk space on /tmp may be an issue.

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:



14.1 Add a new project

To create a new project:



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

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.
- **Ansible Environment** (optional) - Select from the drop-down menu list a custom virtual environment on which to run this project. This field is only present if custom environments were previously created. See [Using virtualenv with Ansible Tower](#) in the *Ansible Tower Upgrade and Migration Guide*.
- **SCM 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.

Note: 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.
```

3. Click **Save** when done.

14.1.1 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 Tower service runs as.
- Ensure that the permissions are appropriate for the playbook directories and files.

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 from Tower:

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.

14.1.2 Manage playbooks using source control


- *SCM Types - Git, Mercurial and Subversion*
- *SCM Type - Red Hat Insights*
- *SCM Type - Remote Archive*

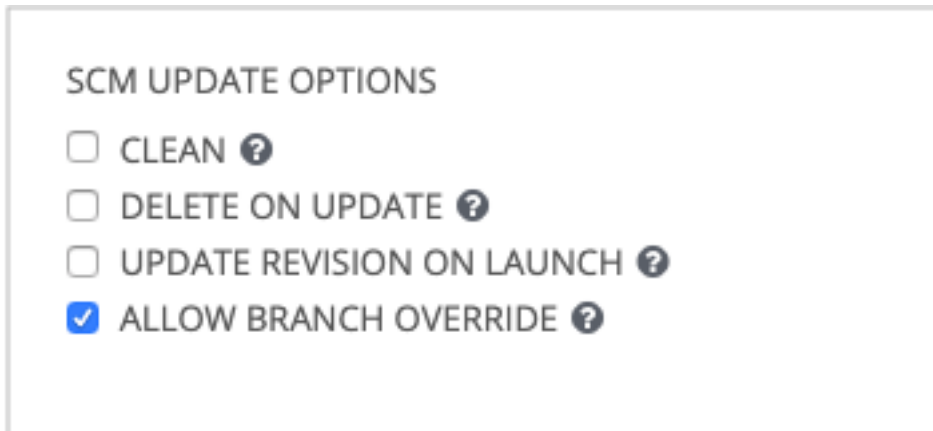
SCM Types - Git, Mercurial and Subversion

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

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

2. Enter the appropriate details into the following fields:

- **SCM URL** - See an example in the help  text.
 - **SCM Branch/Tag/Commit** - Optionally enter the SCM branch, tags, commit hashes, arbitrary refs, or revision number (if applicable) from the source control (Git, Subversion, or Mercurial) to check-out. Some commit hashes and refs may not be available unless you also provide a custom refspec in the next field.
 - **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*.
 - **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 on Update** - 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*. Tower 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*.



3. 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 *Ansible Tower Quick Start 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 on Update** - 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*. Tower 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.

The screenshot shows the 'NEW PROJECT' form with the following fields and options:

- NAME:** Red Hat Insights Project
- DESCRIPTION:** (Empty)
- ORGANIZATION:** Honey Dog, Inc.
- SCM TYPE:** Red Hat Insights
- SOURCE DETAILS:**
 - CREDENTIAL:** Insights Credential
- SCM UPDATE OPTIONS:**
 - CLEAN
 - DELETE ON UPDATE
 - UPDATE REVISION ON LAUNCH

Buttons for CANCEL and SAVE are visible at the bottom right.

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 on Update** - 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.

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).

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

14.2 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.

2. Click on the dot under **Status** (far left, beside the name of the Project) to get further details about the update process.

JOB / DEMO PROJECT

RESULTS

NAME Demo Project

STATUS ● Successful

STARTED 2/27/2017 1:43:26 PM

FINISHED 2/27/2017 1:43:30 PM

ELAPSED 3.775 seconds

LAUNCH TYPE Dependency

PROJECT Demo Project

STANDARD OUT

```

Using /etc/ansible/ansible.cfg as config file

PLAY [all] *****

TASK [delete project directory before update] *****
skipping: [localhost]

TASK [update project using git and accept hostkey] *****
skipping: [localhost]

TASK [Set the git repository version] *****
skipping: [localhost]

TASK [update project using git] *****
ok: [localhost]

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

TASK [update project using hg] *****
skipping: [localhost]

TASK [Set the hg repository version] *****
skipping: [localhost]
                
```

14.3 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 Tower elements are Privileges.

You can access the project permissions via the **Permissions** 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 **User**, **Role**, or **Team Role**.

PROJECTS / New project / PERMISSIONS

New project

DETAILS **PERMISSIONS** NOTIFICATIONS JOB TEMPLATES SCHEDULES

SEARCH Q KEY +

USER	ROLE	TEAM ROLES
admin	SYSTEM ADMINISTRATOR	
jdodge	✕ UPDATE	

ITEMS 1 - 2

14.3.1 Add Permissions

The **Permissions** tab allows you to review, grant, edit, and remove associated permissions for users as well as team members. To assign permissions to a particular user for this resource:

1. Click the **Permissions** tab.



2. Click the  button to open the Add Users/Teams window.

DEMO EXAMPLE | ADD USERS / TEAMS

1 Please select Users / Teams from the lists below.

USERS TEAMS

SEARCH Q KEY

USERNAME ^	FIRST NAME ^	LAST NAME ^
<input type="checkbox"/> althea	Althea	Bully
<input type="checkbox"/> austin78	Austin	Texas
<input type="checkbox"/> gdoge	Gerry	Doge
<input type="checkbox"/> jdoge	Josie	Doge
<input type="checkbox"/> jgarcia	Jerry	Garcia

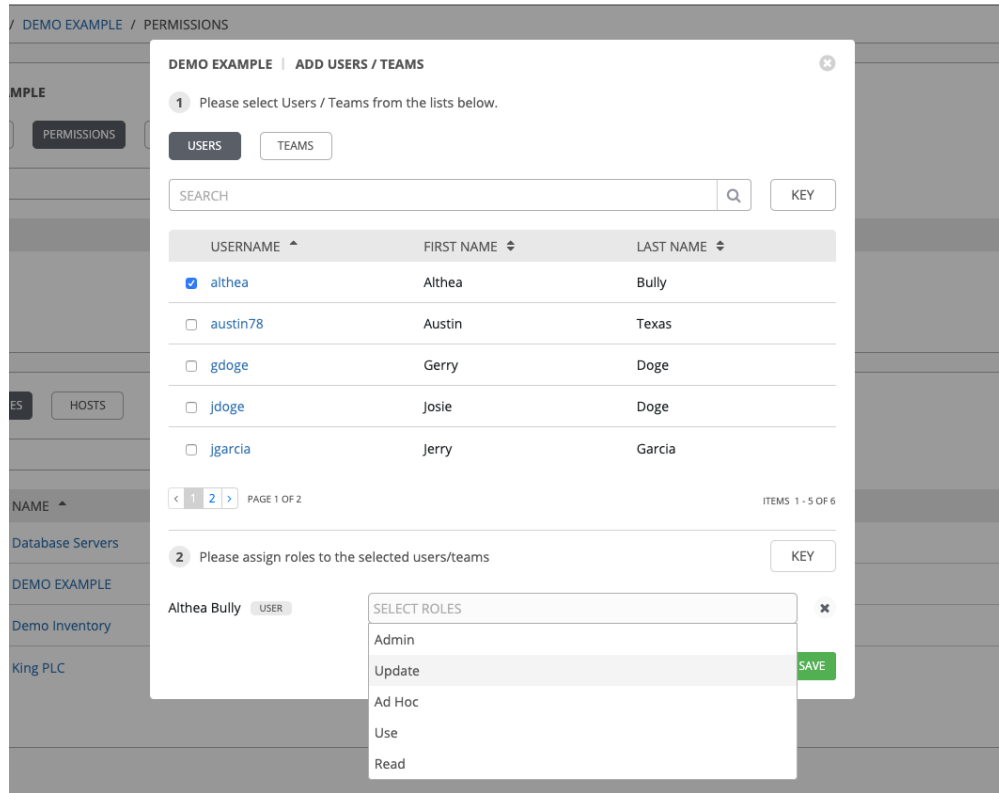
< 1 2 > PAGE 1 OF 2 ITEMS 1 - 5 OF 6

CANCEL SAVE

3. Specify the users or teams that will have access then assign them specific roles:
 - a. Click to select one or multiple check boxes beside the name(s) of the user(s) or team(s) to select them.

Note: You can select multiple users and teams at the same time by navigating between the **Users** and **Teams** tabs without saving.

After selections are made, the window expands to allow you to select a role from the drop-down menu list for each user or team you chose.



The example above shows options associated with inventories. Different resources have different options available:

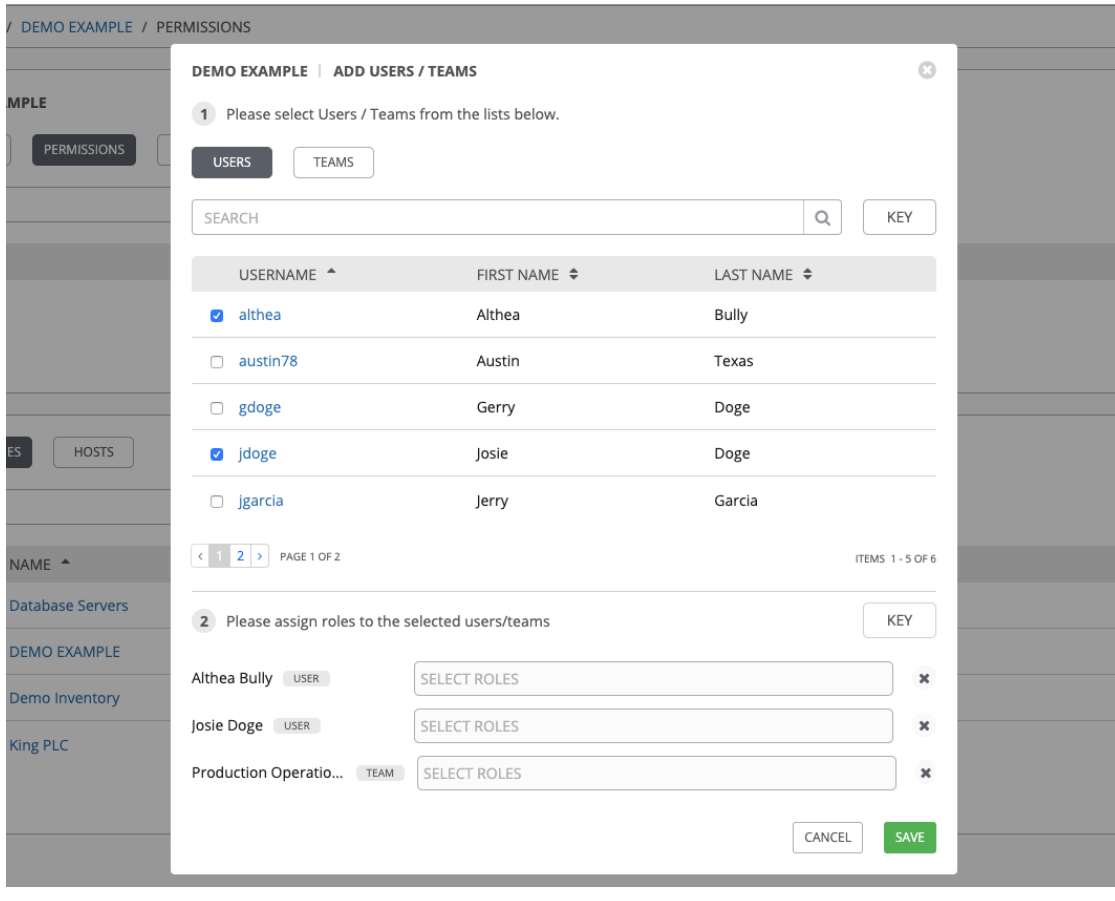
- **Admin** allows read, run, and edit privileges (applies to all resources)
- **Use** allows use of a resource in a job template (applies all resources except job templates)
- **Update** allows updating of project via the SCM Update (applies to projects and inventories)
- **Ad Hoc** allows use of Ad Hoc commands (applies to inventories)
- **Execute** allows launching of a job template (applies to job templates)
- **Read** allows view-only access (applies to all resources)

Tip: Use the **Key** button in the roles selection pane to display a description of each of the roles. For more information, refer to the [Roles](#) section of this guide.

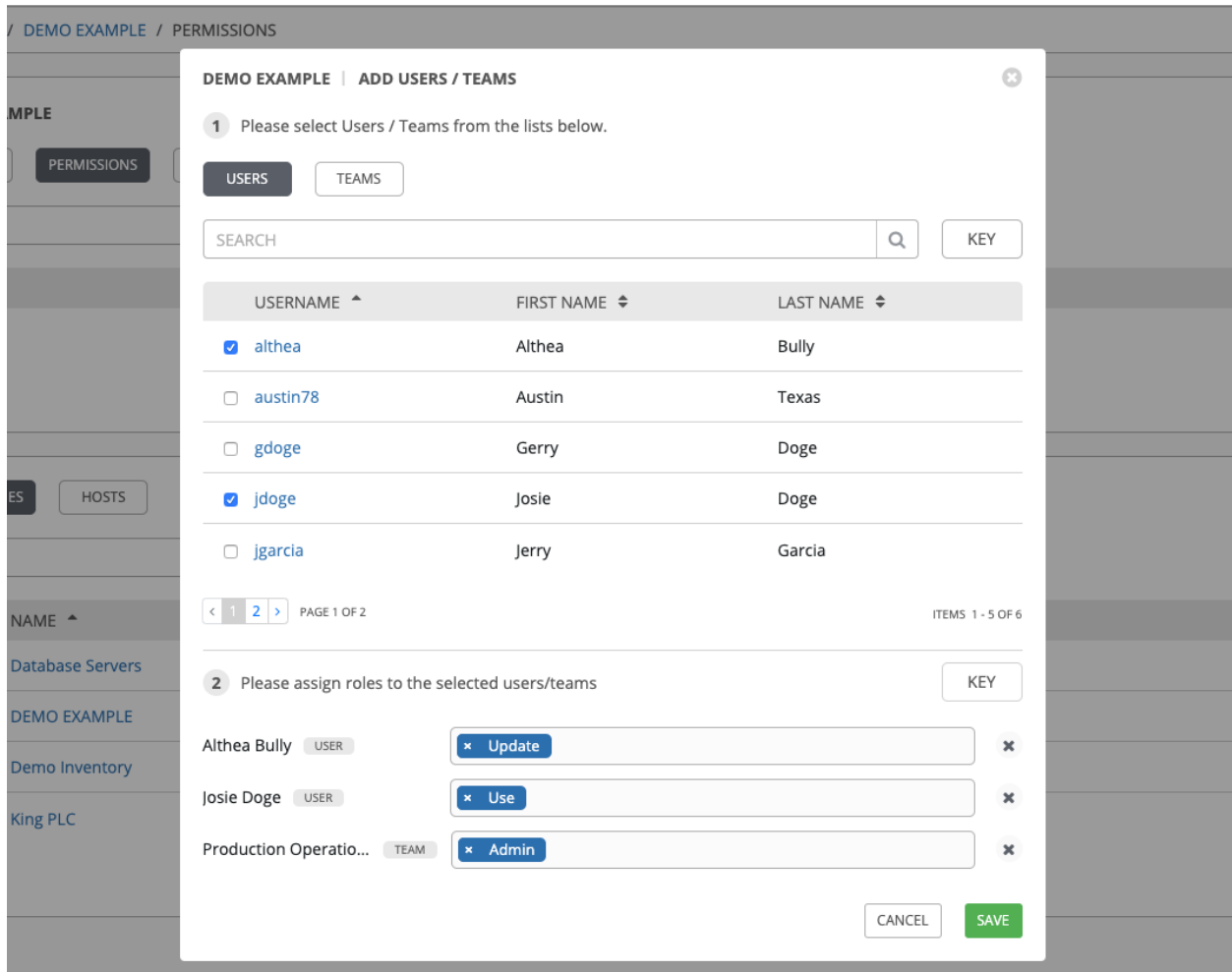
- Select the role to apply to the selected user or team.

Note:

You can assign roles to multiple users and teams by navigating between the **Users** and **Teams** tabs without saving.



4. Review your role assignments for each user and team.



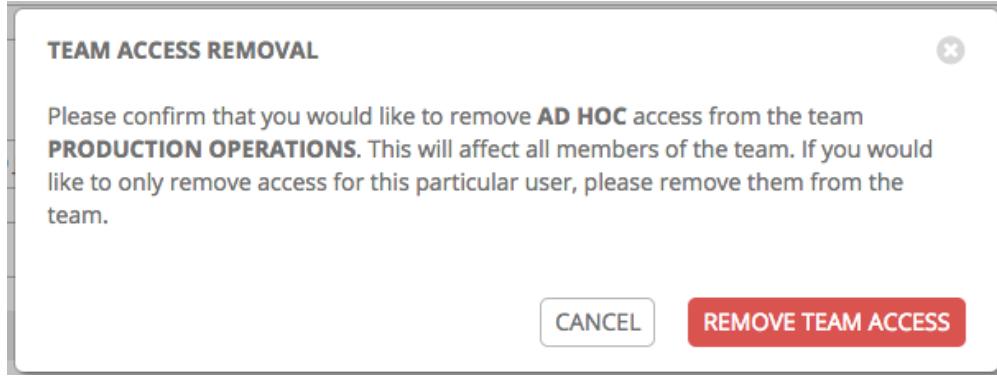
5. Click **Save** when done, and the Add Users/Teams window closes to display the updated roles assigned for each user and team.

USER	ROLE	TEAM ROLES
admin	SYSTEM ADMINISTRATOR	
althea	AD HOC, SYSTEM AUDITOR	USE
jdoge	UPDATE, USE	
mags3707	SYSTEM ADMINISTRATOR	AD HOC, ADMIN, USE
yser	SYSTEM AUDITOR	

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

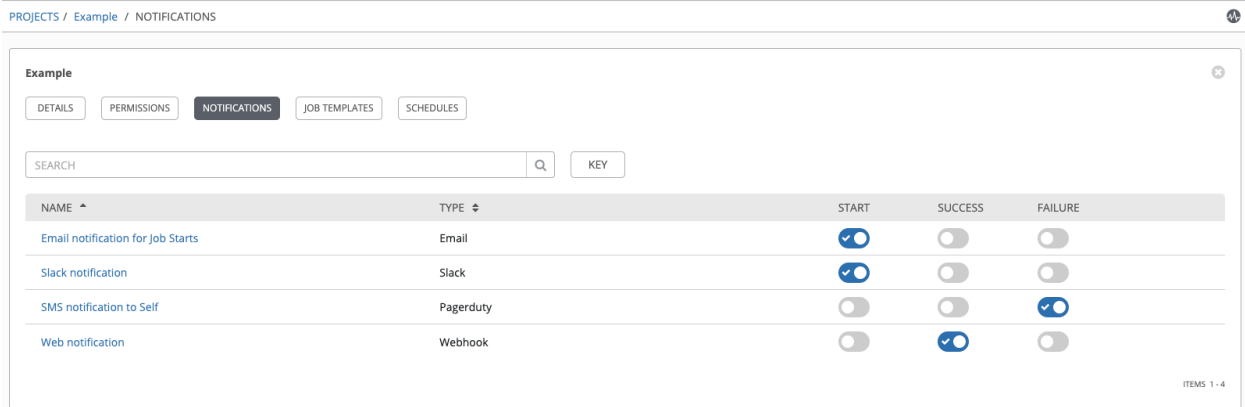
USER	ROLE	TEAM ROLES
admin	SYSTEM ADMINISTRATOR	
althea	AD HOC, SYSTEM AUDITOR	USE
jdoge	UPDATE, USE	
mags3707	SYSTEM ADMINISTRATOR	AD HOC, ADMIN, USE
yser	SYSTEM AUDITOR	

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



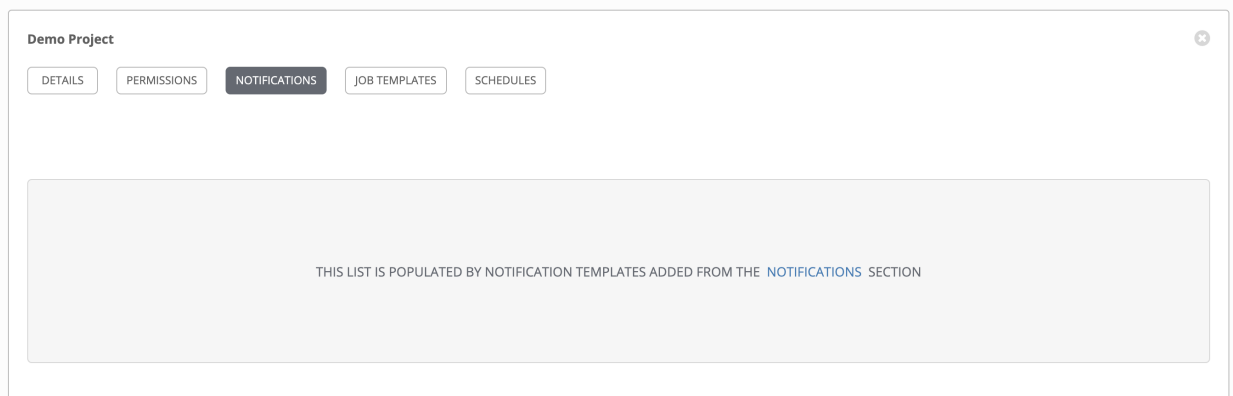
14.4 Work with Notifications

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



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, click the **NOTIFICATIONS** link from inside the gray box to create a new notification.



Refer to [Notification Types](#) for additional details on configuring various notification types.

14.5 Work with Job Templates

Clicking on **Job Templates** allows you to add and review any job templates or workflow templates associated with this project. Click **Expanded** to view details about each template, including the statuses of the jobs that ran using that template, and other useful information. You can sort this list by various criteria, and perform a search to filter the templates of interest.

PROJECTS / Demo Project / JOB TEMPLATES

Demo Project




DETAILS PERMISSIONS NOTIFICATIONS **JOB TEMPLATES** SCHEDULES

SEARCH [] KEY [] +

Compact Expanded Name (Ascending) v

Job Template	Inventory	Project	Last Modified	Last Ran	Actions
Demo Job Template	Demo Inventory	Demo Project	10/28/2019 11:53:20 AM by admin	11/4/2019 2:13:59 PM	[Launch] [Copy] [Delete]
New template with dependencies	King PLC	Demo Project	11/12/2019 10:45:02 PM by admin		[Launch] [Copy] [Delete]
November job template	King PLC	Demo Project	11/12/2019 10:06:50 PM by admin	11/12/2019 10:07:17 PM	[Launch] [Copy] [Delete]

ITEMS 1 - 3

From this view, you can also launch (), copy (), or delete () the template configuration. Note, the example above shows the expanded view.

14.6 Work with Schedules

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

Demo Project

DETAILS PERMISSIONS NOTIFICATIONS JOB TEMPLATES **SCHEDULES**

SEARCH [] KEY [] +

NAME ^	FIRST RUN v	NEXT RUN v	FINAL RUN v	ACTIONS
<input checked="" type="checkbox"/> Run Once	11/13/2019 12:00:00 AM	11/13/2019 12:00:00 AM	11/13/2019 12:00:00 AM	[Edit] [Delete]
<input checked="" type="checkbox"/> Schedule 1	11/15/2019 12:00:00 AM	11/15/2019 12:00:00 AM		[Edit] [Delete]
<input checked="" type="checkbox"/> Schedule 2	12/1/2019 12:02:00 AM	12/1/2019 12:02:00 AM	3/1/2020 12:02:00 AM	[Edit] [Delete]
<input checked="" type="checkbox"/> Schedule 3	12/25/2019 12:03:00 AM	12/25/2019 12:03:00 AM	1/1/2020 12:03:00 AM	[Edit] [Delete]

ITEMS 1 - 4

14.6.1 Schedule a Project

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

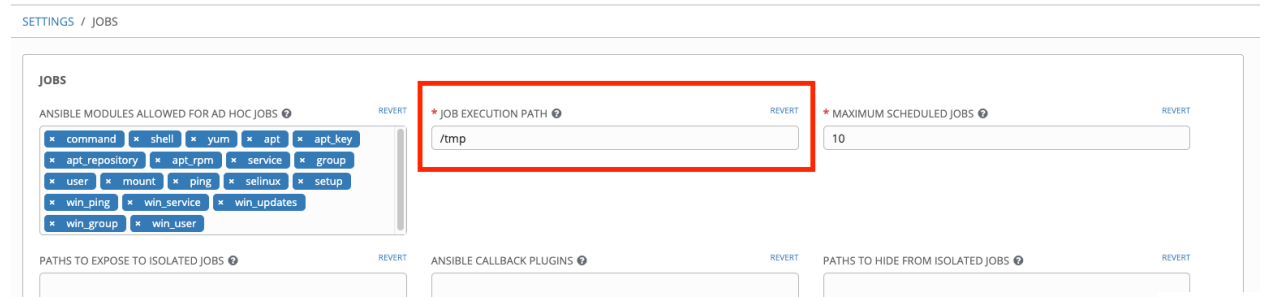
- If schedules are already set up; review, edit, or enable/disable your schedule preferences.
- If schedules have not been set up, refer to *Schedules* for more information.

14.7 Ansible Galaxy Support

At the end of a Project update, Tower 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 Configure Tower window:



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, Ansible Tower 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** tab of the Settings () menu by switching the **Enable Role Download** toggle button to **OFF**.

SETTINGS / JOBS

JOBS

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

* JOB EXECUTION PATH REVERT

/tmp

* MAXIMUM SCHEDULED JOBS REVERT

10

PATHS TO EXPOSE TO ISOLATED JOBS REVERT

ANSIBLE CALLBACK PLUGINS REVERT

PATHS TO HIDE FROM ISOLATED JOBS REVERT

* ENABLE JOB ISOLATION REVERT

DEFAULT PROJECT UPDATE TIMEOUT REVERT

0

DEFAULT INVENTORY UPDATE TIMEOUT REVERT

0

* RUN PROJECT UPDATES WITH HIGHER VERBOSITY REVERT

PER-HOST ANSIBLE FACT CACHE TIMEOUT REVERT

0

MAXIMUM NUMBER OF FORKS PER JOB REVERT

200

ENABLE COLLECTION(S) DOWNLOAD REVERT

IGNORE ANSIBLE GALAXY SSL CERTIFICATE VERIFICATION REVERT

ENABLE ROLE DOWNLOAD REVERT

FOLLOW SYMLINKS REVERT

ISOLATED HOST KEY CHECKING REVERT

* ISOLATED STATUS CHECK INTERVAL REVERT

30

* ISOLATED LAUNCH TIMEOUT REVERT

600

ISOLATED CONNECTION TIMEOUT REVERT

10

ENABLE DETAILED RESOURCE PROFILING ON ALL PLAYBOOK RUNS REVERT

EXTRA ENVIRONMENT VARIABLES REVERT

1 {}

REVERT ALL TO DEFAULT

Whenever a project sync runs, Tower 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.

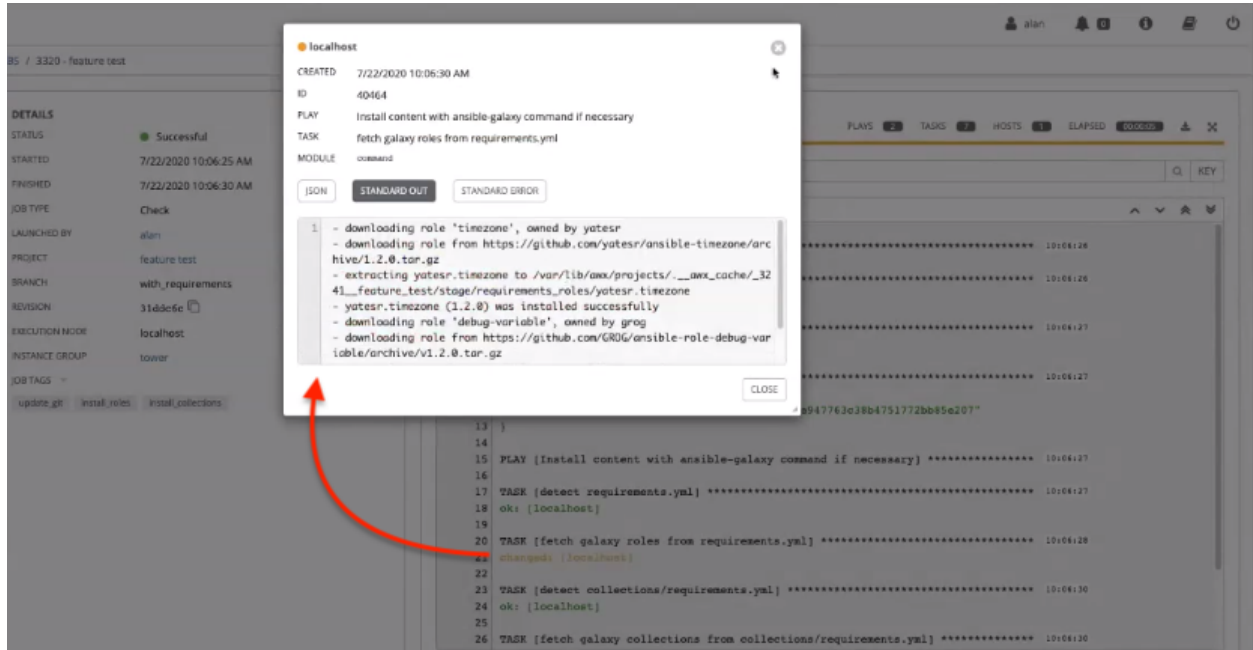
If jobs need to pick up a change made to an upstream role, updating the project will ensure this happens. A change to the role means that a new commit was pushed to the *provision-role* source control. To make this change take effect in a job, you do not need to push a new commit to the *playbooks* repo, 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 Tower 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:

SCM UPDATE OPTIONS

- CLEAN ?
- DELETE ON UPDATE ?
- 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.

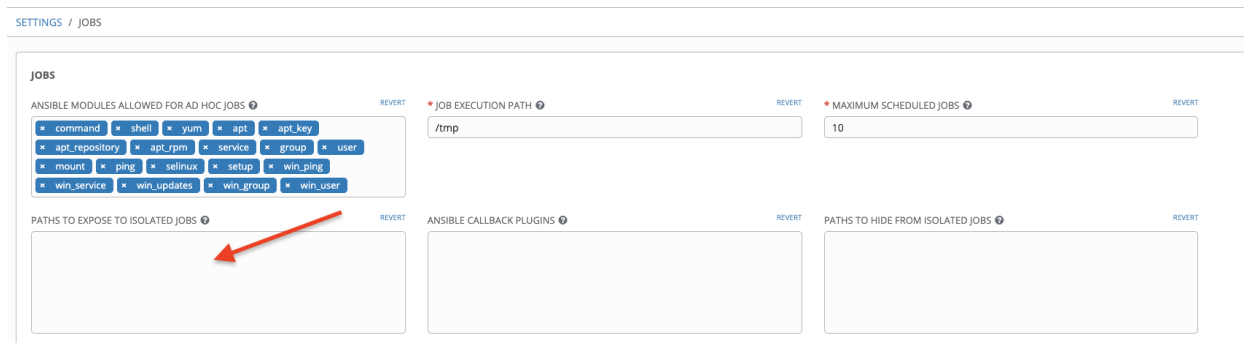
If there are any directories that should specifically be exposed, you can specify those in the Configure Tower screen in the **Paths to Expose to Isolated Jobs** or by updating the following entry in the settings file:

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

Note: The primary file you may want to add to `AWX_PROOT_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 `ansible-tower-service restart` command after your changes have been saved.

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



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

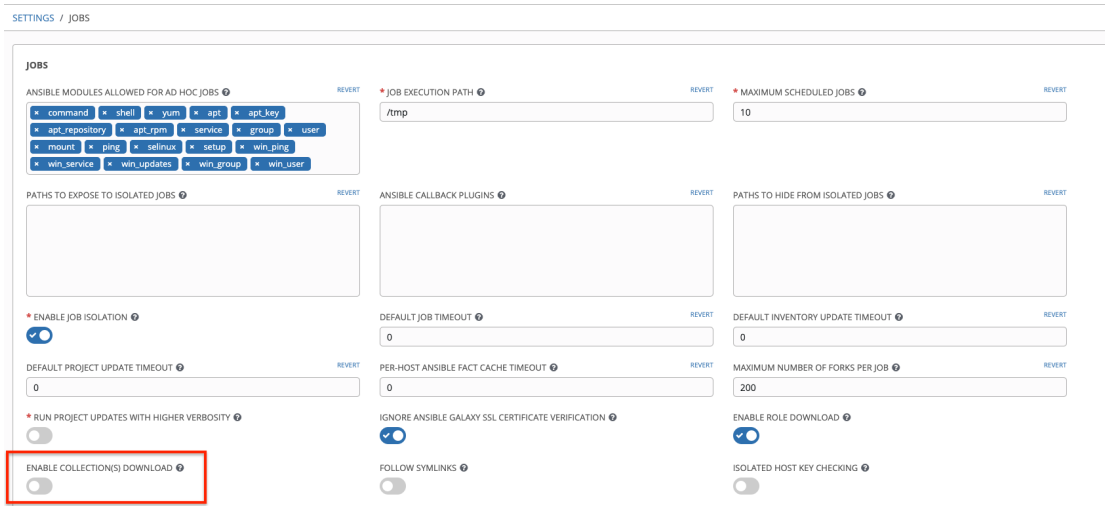
14.8 Collections Support

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

```
ansible-galaxy collection install -r requirements.yml -p <job tmp location>
```

By default, Ansible Tower 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** tab of the

Settings () menu by switching the **Enable Collections Download** toggle button to **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:

SCM UPDATE OPTIONS

- CLEAN ?
- DELETE ON UPDATE ?
- UPDATE REVISION ON LAUNCH ?
- ALLOW BRANCH OVERRIDE ?

Repo Management

Local Remote

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 *Tower Create Credential* form:

NEW CREDENTIAL

DETAILS PERMISSIONS

* NAME Automation Hub DESCRIPTION ORGANIZATION Default

* CREDENTIAL TYPE Ansible Galaxy/Automation Hub API Token

TYPE DETAILS

* GALAXY SERVER URL https://10.10.94.209/api/galaxy/content/published AUTH SERVER URL API TOKEN

CANCEL SAVE

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

Default

DETAILS USERS PERMISSIONS NOTIFICATIONS

* NAME Default DESCRIPTION INSTANCE GROUPS

GALAXY CREDENTIALS Ansible Galaxy Automation Hub MAX HOSTS 0

CANCEL SAVE

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.

- If the Automation Hub has self-signed certificates, click the toggle to enable the Tower 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:


The screenshot shows the 'JOBS' configuration page in Ansible Tower. The 'IGNORE ANSIBLE GALAXY SSL CERTIFICATE VERIFICATION' toggle is highlighted with a red box and is currently turned on. Other visible settings include 'ANSIBLE MODULES ALLOWED FOR AD HOC JOBS', 'JOB EXECUTION PATH' set to '/tmp', 'MAXIMUM SCHEDULED JOBS' set to 10, 'ENABLE JOB ISOLATION' turned on, 'DEFAULT PROJECT UPDATE TIMEOUT' set to 0, 'PER-HOST ANSIBLE FACT CACHE TIMEOUT' set to 0, 'MAXIMUM NUMBER OF FORKS PER JOB' set to 200, 'ENABLE ROLE DOWNLOAD' turned on, 'ISOLATED STATUS CHECK INTERVAL' set to 30, and 'ISOLATED LAUNCH TIMEOUT' set to 600.

- 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.

The screenshot shows the 'NEW PROJECT' form in Ansible Tower. The form is filled out with the following information:

- NAME:** Collections Project
- DESCRIPTION:** (empty)
- ORGANIZATION:** Default
- SCM TYPE:** Git
- SOURCE DETAILS:**
 - SCM URL:** https://github.com/ansible-collections/
 - SCM BRANCH/TAG/COMMIT:** (empty)
 - SCM REFSPEC:** (empty)
- SCM CREDENTIAL:** (empty)
- SCM UPDATE OPTIONS:**
 - CLEAN
 - DELETE ON UPDATE
 - UPDATE REVISION ON LAUNCH
 - ALLOW BRANCH OVERRIDE

 The form has 'CANCEL' and 'SAVE' buttons at the bottom right.

- In the Projects list view, click  to run an update against this project. Tower 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.

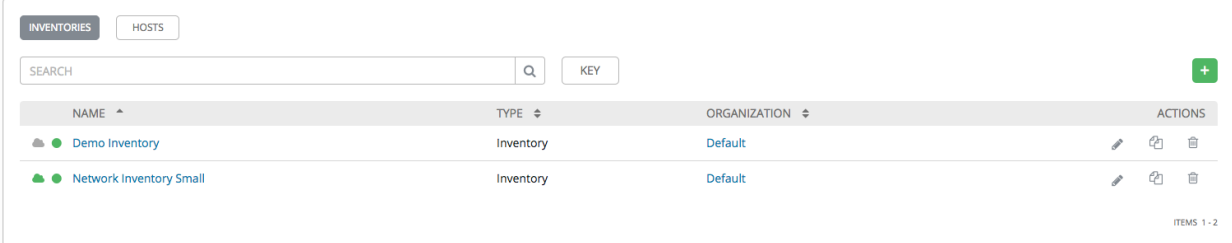
For more information on collections, refer to [Using Collections](#). For more information on how Red Hat itself publishes one of these official collections, which can be used to automate your Tower install directly, refer to the [AWX Ansible Collection](#) documentation. This page is accessible with your Red Hat customer credentials as part of your Red Hat Ansible Automation Platform subscription.









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 Tower, or from one of Ansible Tower's supported cloud providers.

Note: If you have a custom dynamic inventory script, or a cloud provider that is not yet supported natively in Tower, you can also import that into Tower. Refer to [Inventory File Importing](#) in the *Ansible Tower Administration Guide*.


This tab displays a list of the inventories that are currently available. The inventory list may be sorted and searched by **Name**, **Type**, or **Organization**.



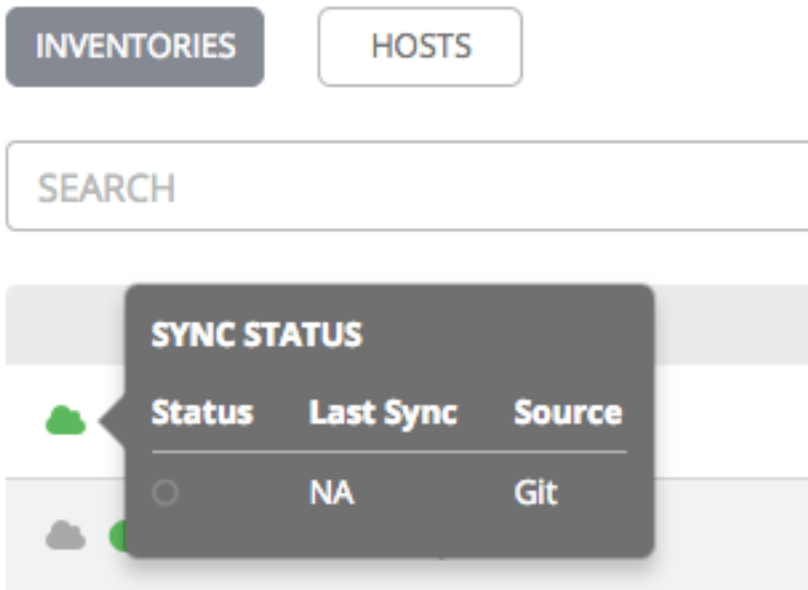
NAME	TYPE	ORGANIZATION	ACTIONS
 Demo Inventory	Inventory	Default	  
 Network Inventory Small	Inventory	Default	  





ITEMS 1 - 2

The list of Inventory details includes:

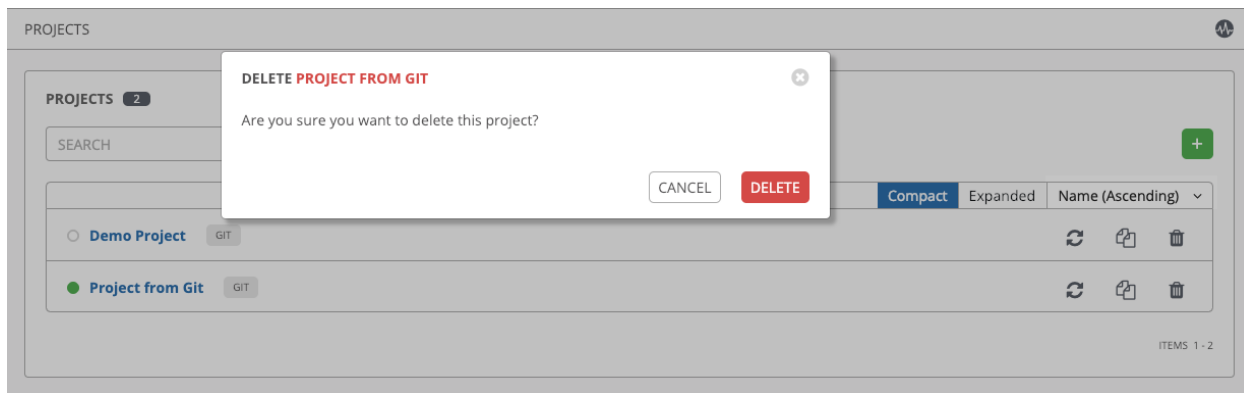
- **Inventory Sync** (

117



- **Status Dot:** This shows the status of recent jobs for this inventory.
- **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.)
- **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
 - **Delete** (): Delete the selected inventory. *This operation cannot be reversed!*

Note: If deleting items that are used by other work items, a message opens listing the items 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:



15.1 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 Smart Inventories. A Smart Inventory is identified by `KIND=smart`. You can define a Smart Inventory using the same method being used with Tower 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" "ad hoc" 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.

15.2 Inventory Plugins

Inventory updates have changed from using deprecated inventory scripts, to using dynamically-generated YAML files which are parsed by their respective inventory plugin. In Ansible Tower 3.8, users can provide the new style inventory plugin config directly to Tower 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*
- *OpenStack*
- *Red Hat Virtualization*
- *Ansible Tower*

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, Tower uses "templates" for each of these sources to force the output of inventory plugins into the legacy format. Refer to [Supported Inventory Plugin Templates](#) in the *Ansible Automation Platform Installation and Reference Guide* for each source and their respective templates to help you migrate to the new style inventory plugin output. .. reused content end.

`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`.

If you already have an inventory source set up, then Tower automatically switches to use the inventory plugins depending on the source and Ansible version, but continue to maintain the same content previously in those scripts. If you need to control the version of Ansible being used, you can use custom virtual environments for the inventory source. Refer to [Using virtualenv with Ansible Tower](#).


15.3 Add a new inventory

Adding a new inventory involves several components. Click below to jump to a specific component:

- [Add permissions](#)
- [Add groups](#)
- [Add hosts](#)
- [Add source](#)
- [View completed jobs](#)


To create a new inventory or Smart Inventory:



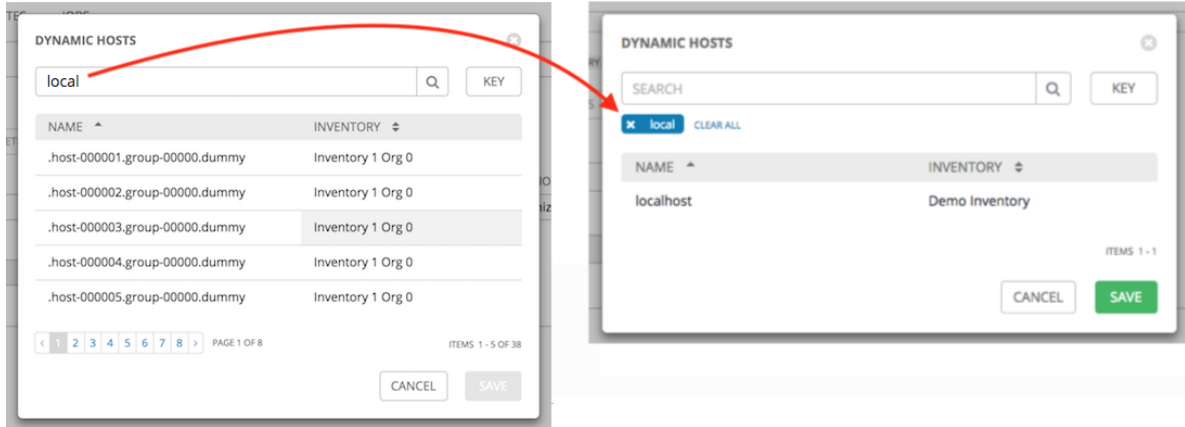
1. Click the  button, and select the type of inventory to create.


The type of inventory is identified by the labels and the row of tabs across the top of the create form.

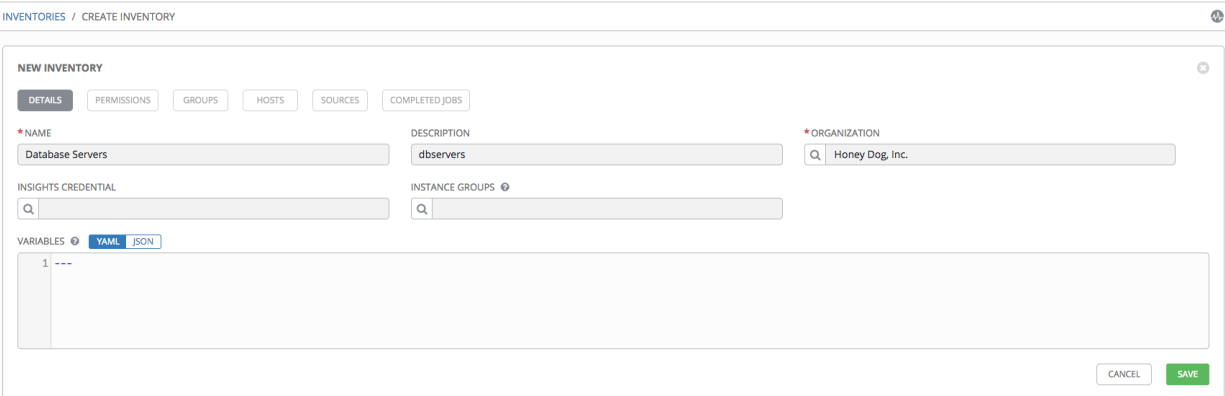
2. Enter the appropriate details into the following fields:

- **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 Dynamic Hosts 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.



- **Insights Credential:** (Only applicable to standard inventories) Enter the appropriate Insights credential if the inventory is used with Insights.
- **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.



3. Click **Save** when done.

After Tower saves 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.

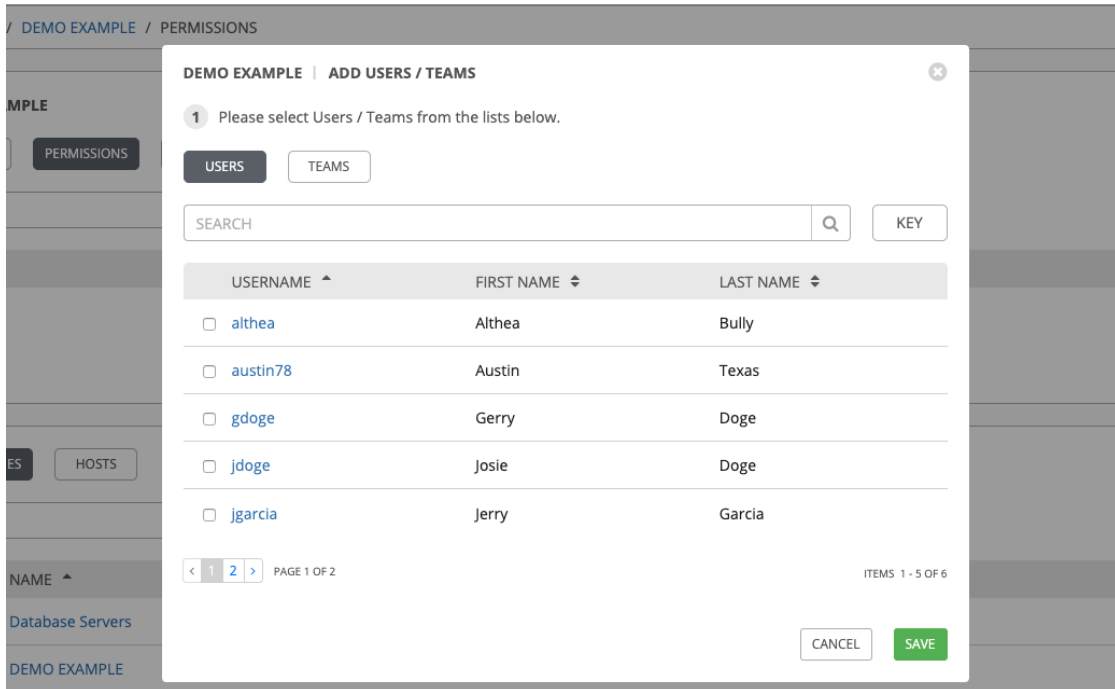
15.3.1 Add permissions

The **Permissions** tab allows you to review, grant, edit, and remove associated permissions for users as well as team members. To assign permissions to a particular user for this resource:

1. Click the **Permissions** tab.



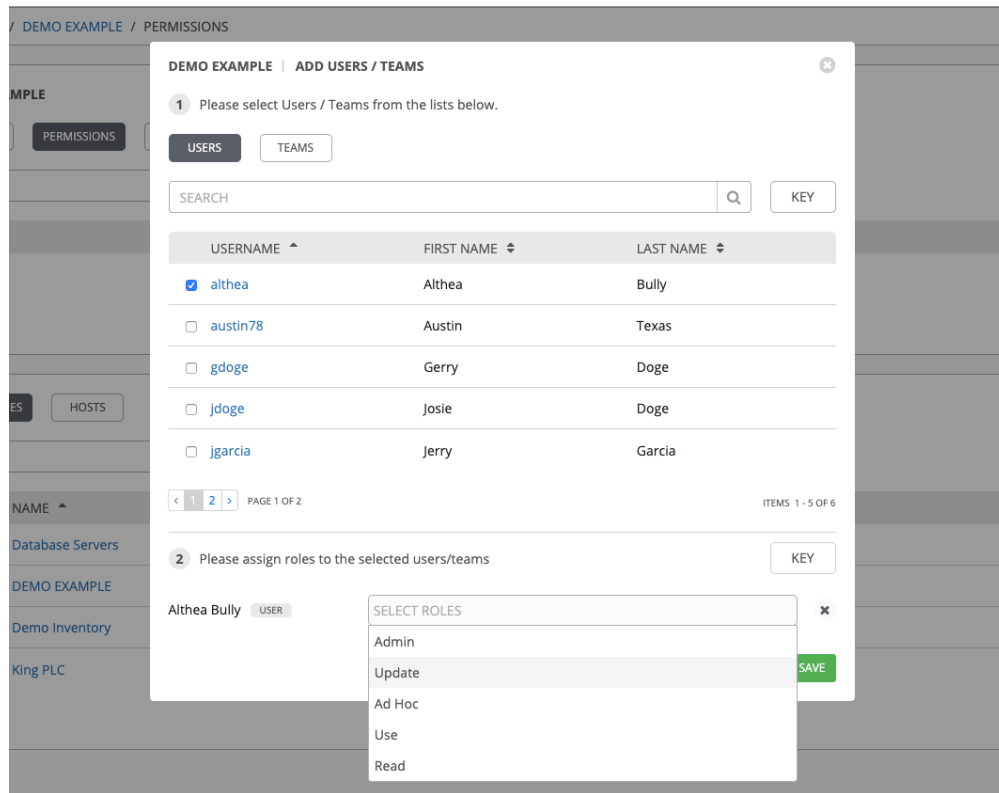
2. Click the  button to open the Add Users/Teams window.



3. Specify the users or teams that will have access then assign them specific roles:
 - a. Click to select one or multiple check boxes beside the name(s) of the user(s) or team(s) to select them.

Note: You can select multiple users and teams at the same time by navigating between the **Users** and **Teams** tabs without saving.

After selections are made, the window expands to allow you to select a role from the drop-down menu list for each user or team you chose.



The example above shows options associated with inventories. Different resources have different options available:

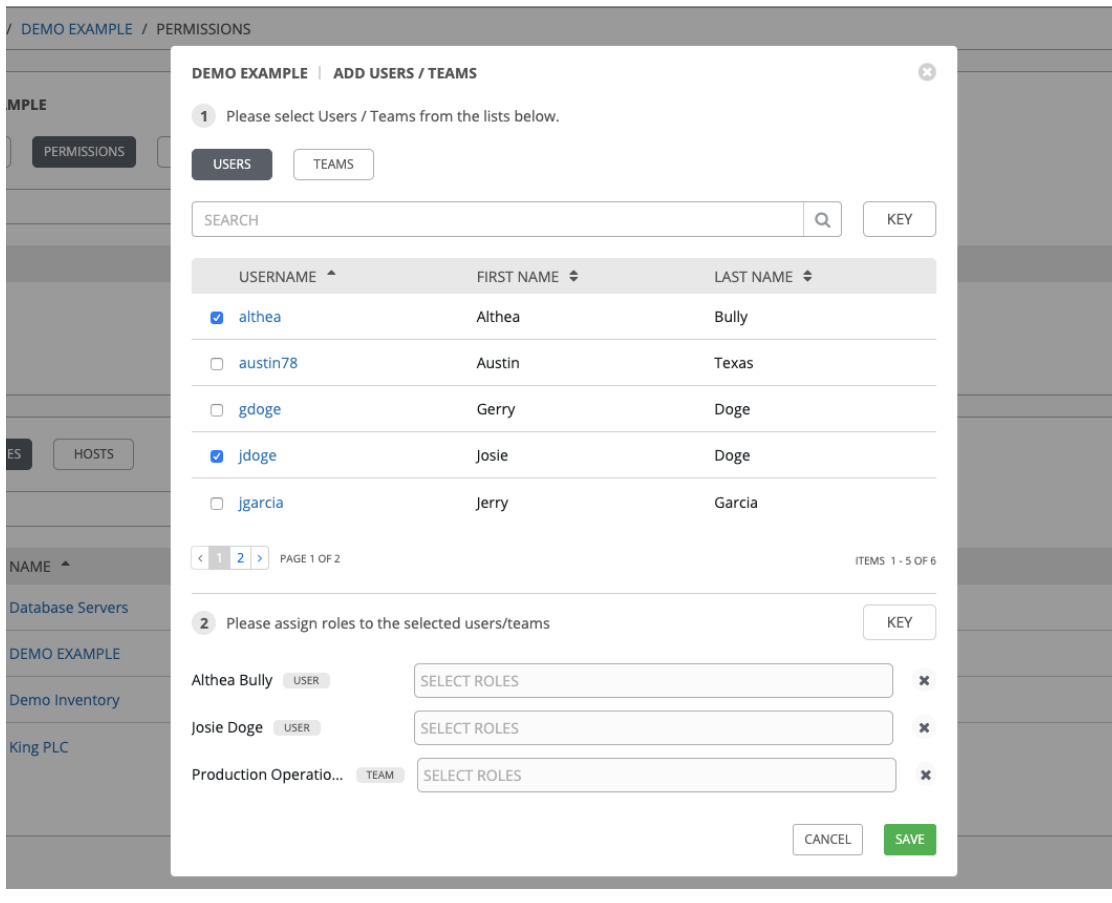
- **Admin** allows read, run, and edit privileges (applies to all resources)
- **Use** allows use of a resource in a job template (applies all resources except job templates)
- **Update** allows updating of project via the SCM Update (applies to projects and inventories)
- **Ad Hoc** allows use of Ad Hoc commands (applies to inventories)
- **Execute** allows launching of a job template (applies to job templates)
- **Read** allows view-only access (applies to all resources)

Tip: Use the **Key** button in the roles selection pane to display a description of each of the roles. For more information, refer to the [Roles](#) section of this guide.

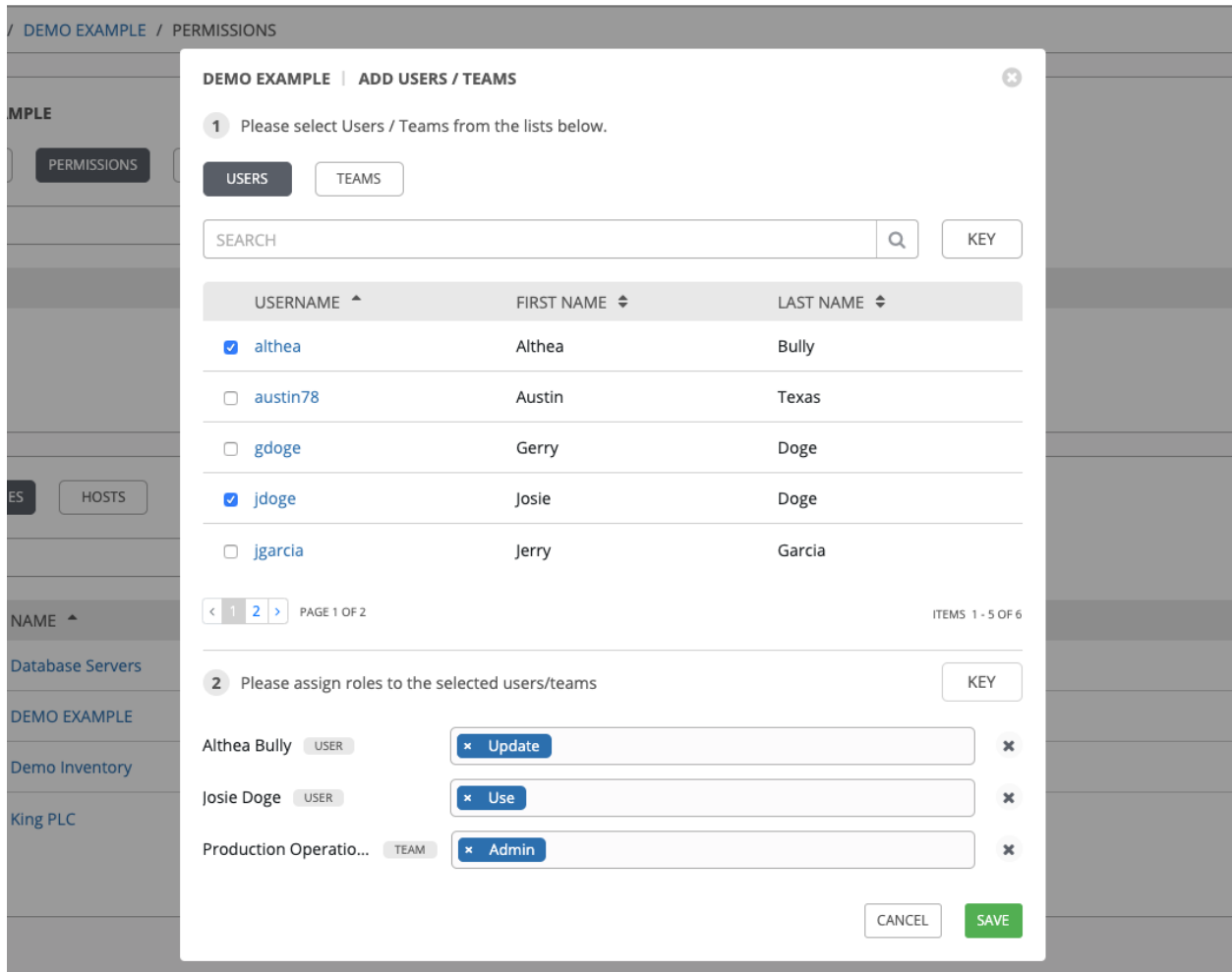
- Select the role to apply to the selected user or team.

Note:

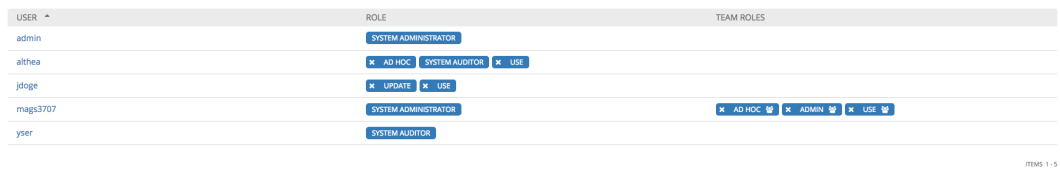
You can assign roles to multiple users and teams by navigating between the **Users** and **Teams** tabs without saving.



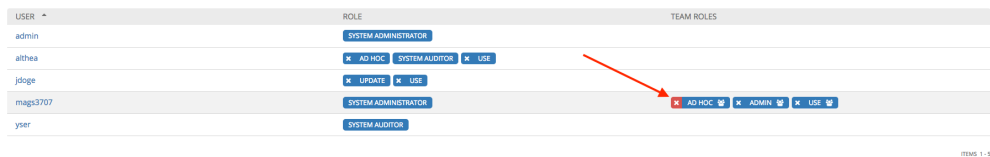
4. Review your role assignments for each user and team.



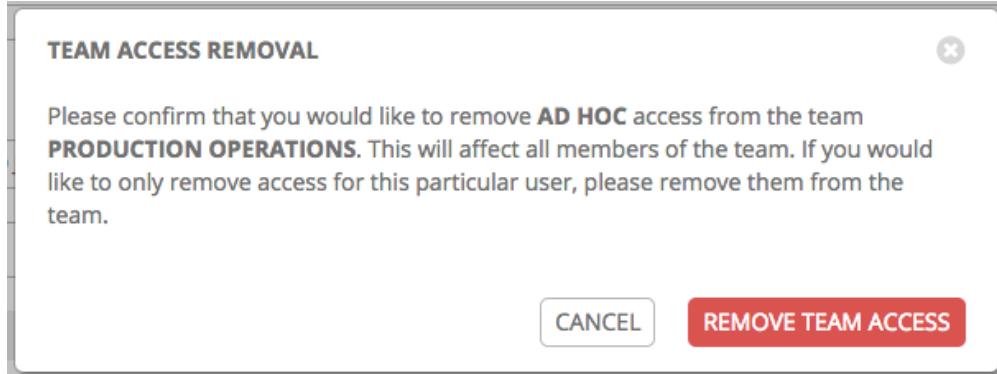
5. Click **Save** when done, and the Add Users/Teams window closes to display the updated roles assigned for each user and team.



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



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



15.3.2 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 no longer associated with groups. Prior versions, spawned groups and hosts would be children of our inventory source group. Now, spawned groups are top-level. These groups 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  button to open the **Create Group** window.

2. Enter the appropriate details into the required and optional fields:
 - **Name:** Required
 - **Description:** Enter an arbitrary description as appropriate (optional)

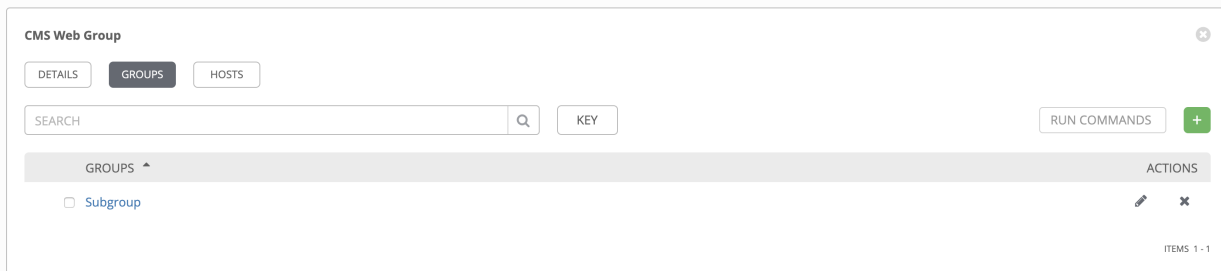
- **Variables:** Enter definitions and values to be applied to all hosts in this group. Enter variables using either JSON or YAML syntax. Use the radio button to toggle between the two.
3. When done, click **Save**.

Add groups within groups

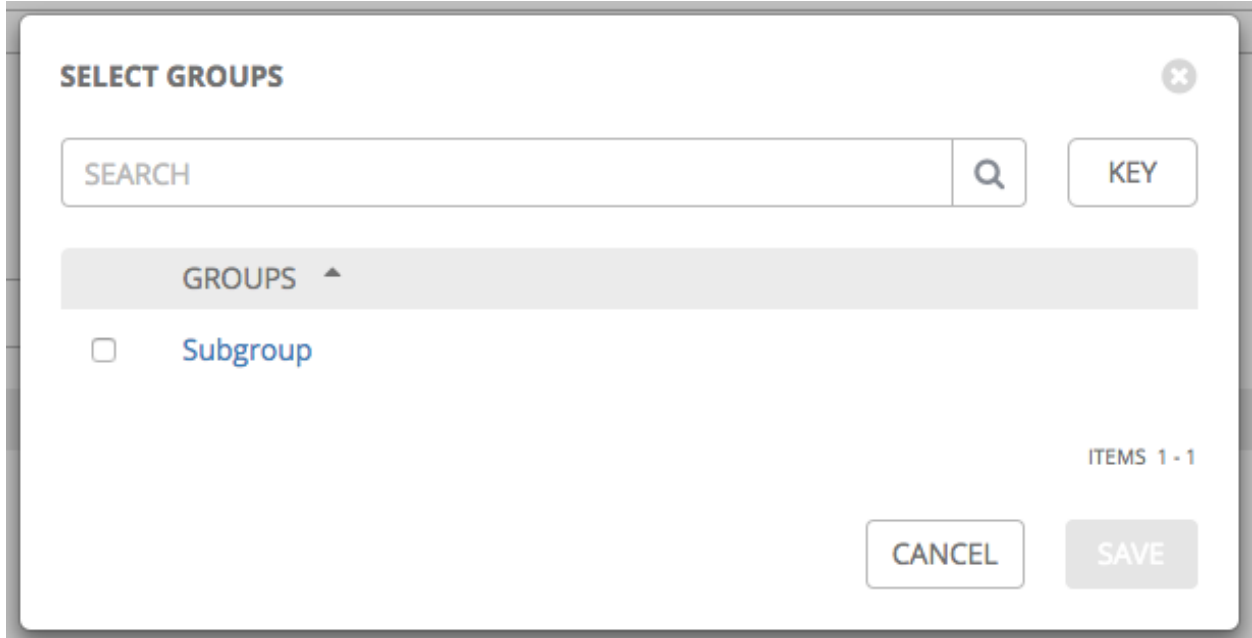
To add groups within groups:

1. Click the **Groups** tab.
2. Click the  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.

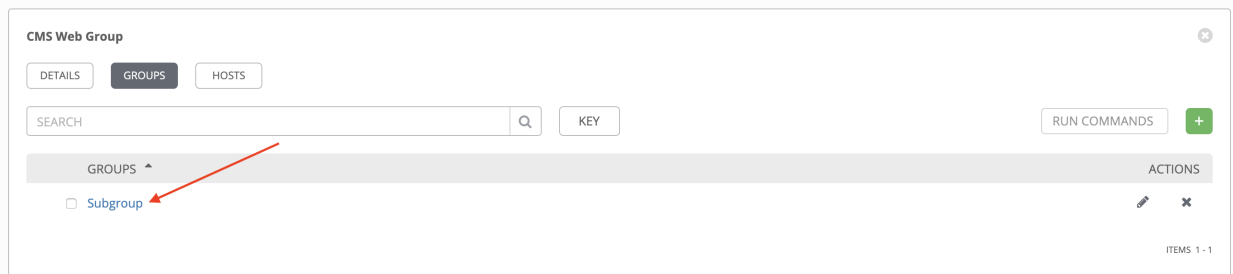


If you chose to add an existing group, available groups will appear in a separate selection window.



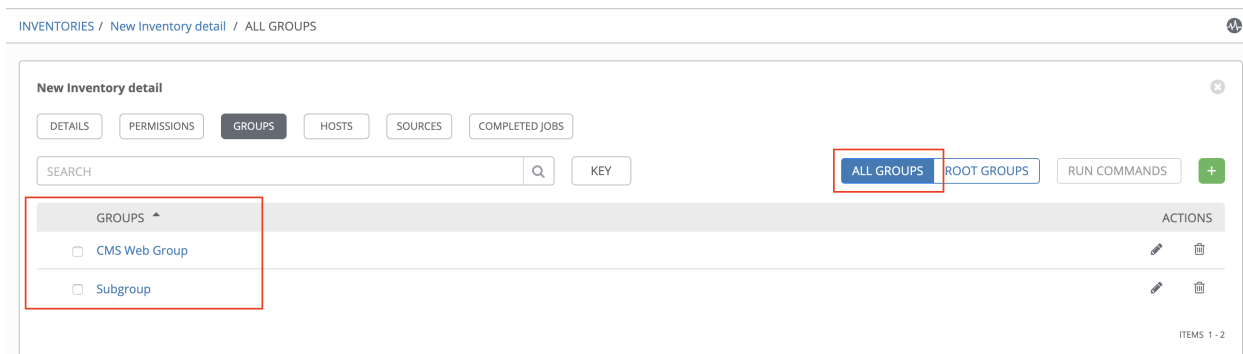
Once a group is selected, it displays as an entry in the list of groups associated with the group.

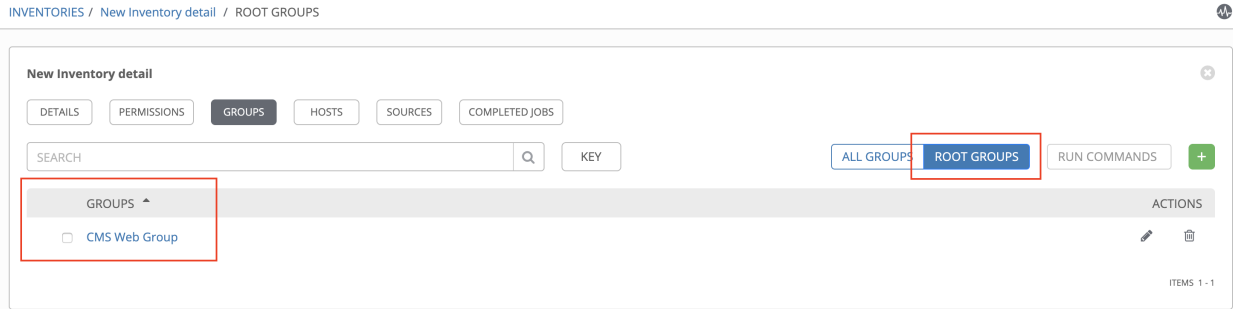
5. To configure additional groups and hosts under the subgroup, click on the name of the subgroup from the list of groups and repeat the same steps described in this section.



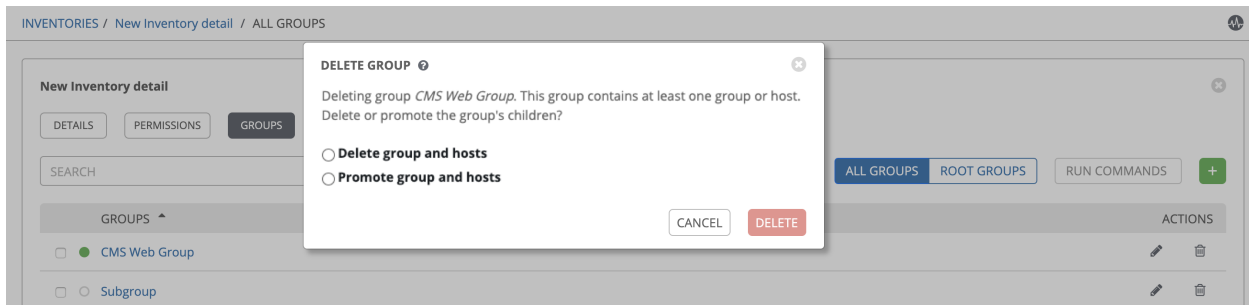
View or edit inventory groups

Starting in Ansible Tower 3.5, you can view 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.

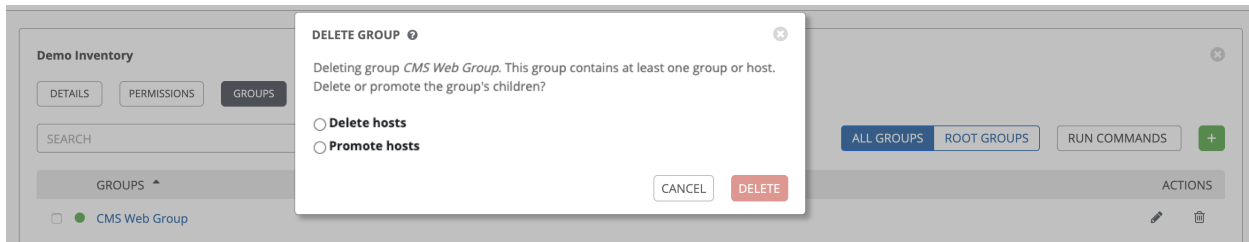




You may be able to delete a subgroup without concern for dependencies, but if you want to delete a root group, Tower will look for dependencies such as any child groups or hosts. If the root group you want to delete has both, a confirmation dialog displays for you to choose whether to delete the root 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).



If the root group has a subgroup that does not have any hosts, the confirmation dialog simply asks if you want to delete everything; or promote your group.





15.3.3 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  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.

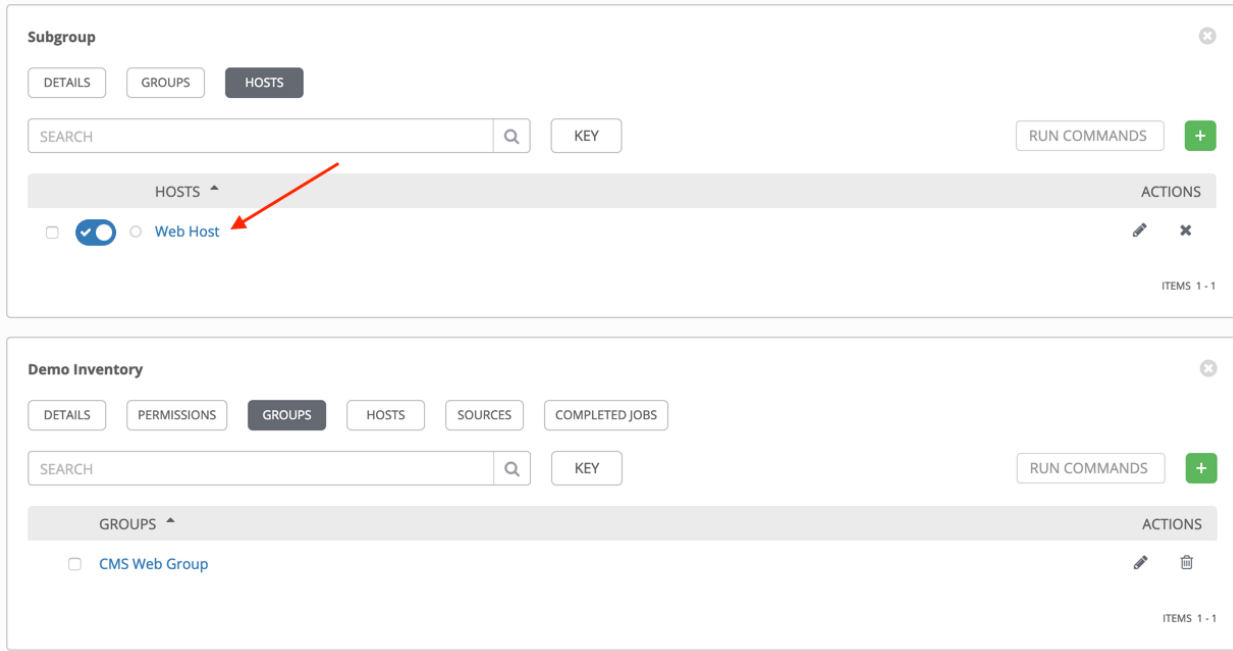
4. Enter the appropriate details into the required and optional fields:

- **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.

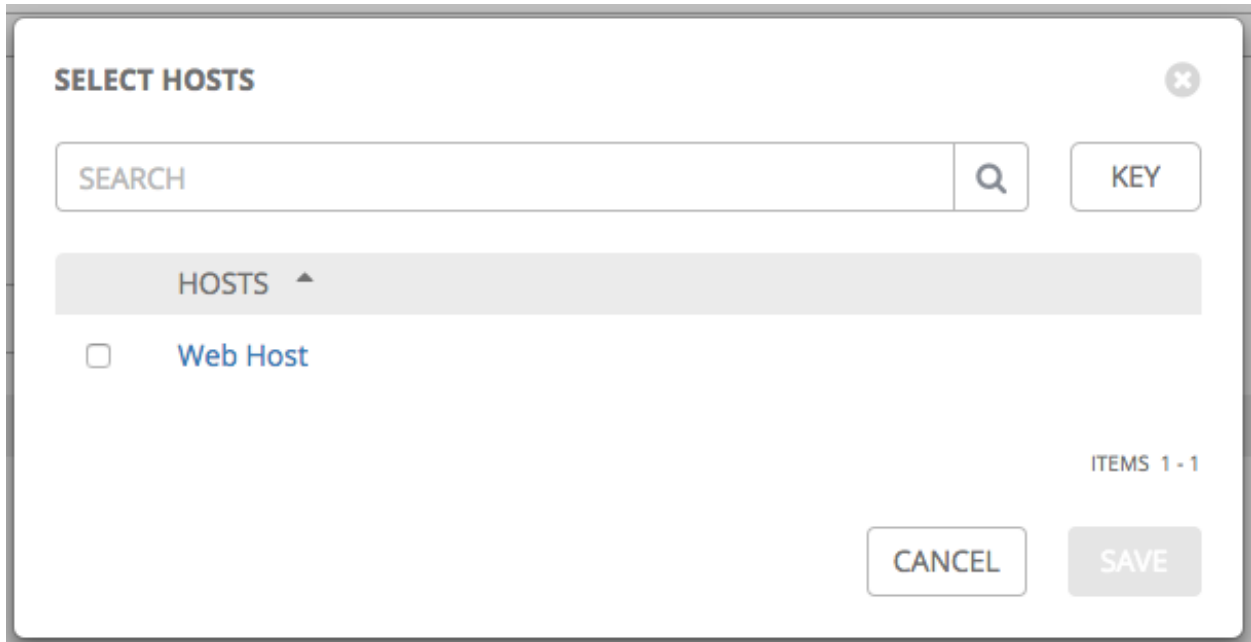
5. When done, click **Save**.

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.



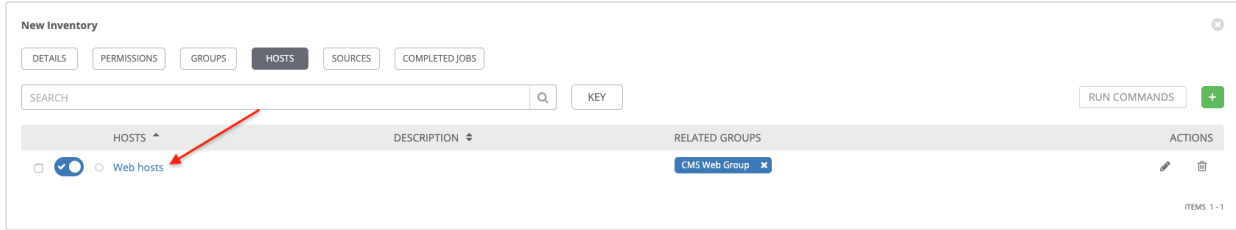
Note: You may also run ad hoc commands from this screen. Refer to *Running Ad Hoc Commands* for more detail.

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

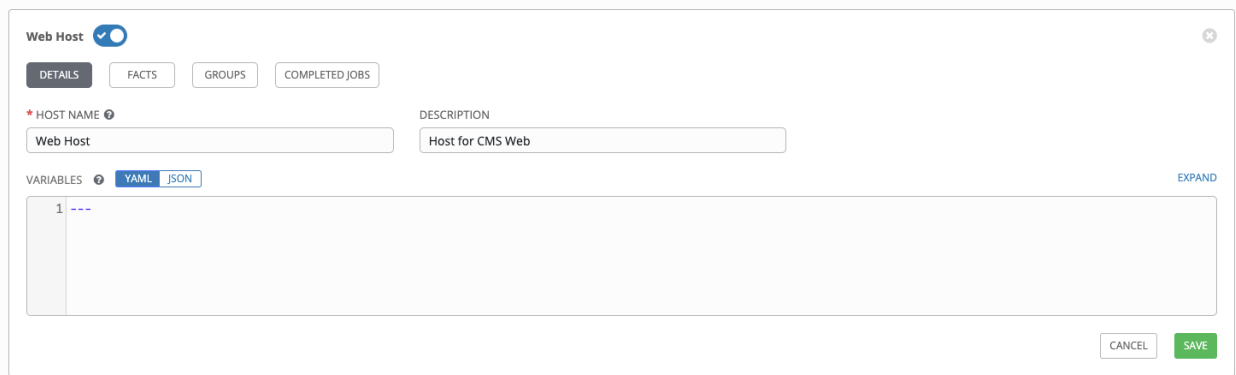


Once a host is selected, it displays as an entry in the list of hosts associated with the group.

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




This opens the Details tab of the selected host.

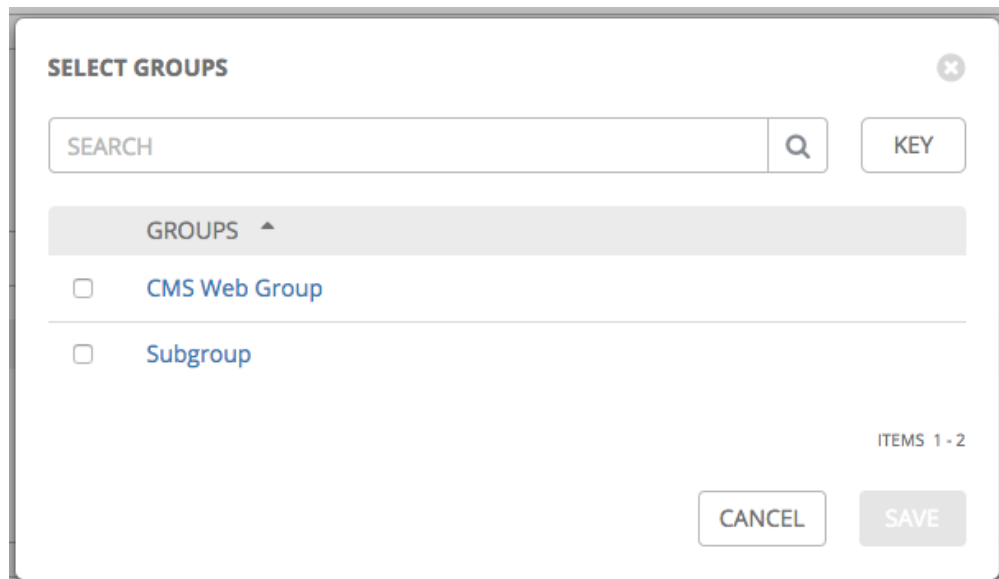


7. Click the **Facts** tab to input facts you want to gather. Refer to the *Fact Caching* section for more information about facts.

8. Click the **Groups** tab to configure groups for the host.

- a. Click the  button to associate the host with an existing group.

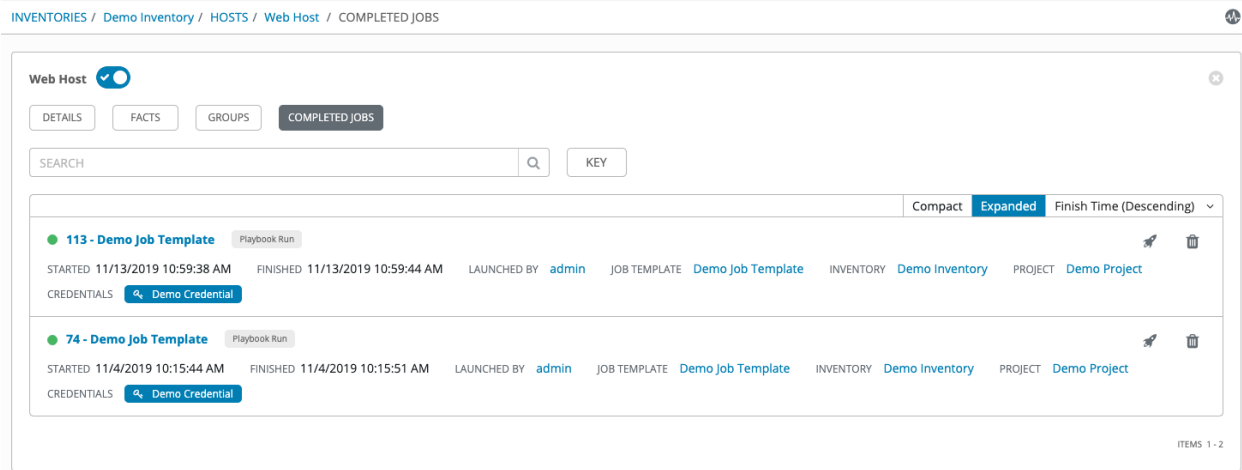
Available groups appear in a separate selection window.



- 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.

9. 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.



15.3.4 Add source

To use an inventory plugin, you must provide an inventory source. Most of the time, this is a file containing host information or a YAML configuration file with options for the plugin. 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  button.

This opens the Create Source window.



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

- **Name:** Required
- **Description:** Enter an arbitrary description as appropriate (optional)
- **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.

- **Ansible Environment:** This field is only present if custom virtual environments (venvs) are setup. Choose the venv with which you want to run your inventory imports. Refer to [Using virtualenv with Ansible Tower](#) for details on setting up a custom venvs.



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 Ansible Tower 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 Tower.
7. In the **Enabled Variable**, specify Tower 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 Tower. 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 Tower 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 Tower’s provisioning callback feature with a dynamic inventory source, “Update on Launch” should be set for the inventory group.

10. Review your entries and selections and click **Save** when done. This allows you to configure additional details, such as notifications and schedules.

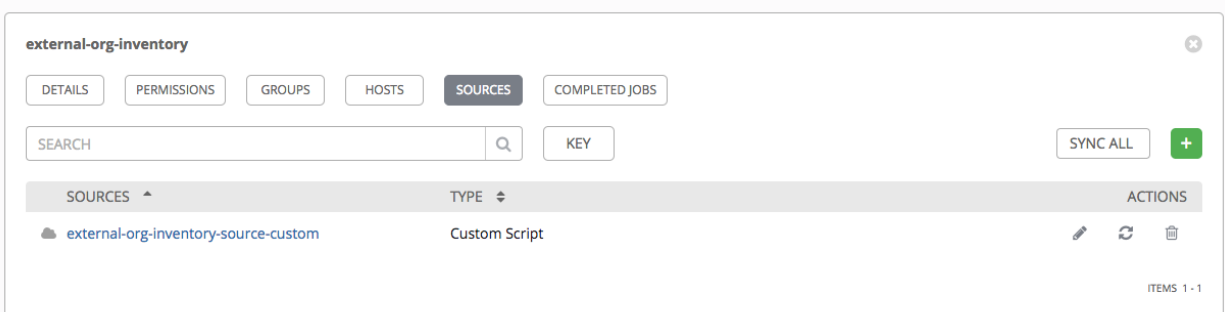
Note: The **Notifications** tab is only present after you save the newly-created source.

Source A



11. 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.
12. 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.
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.




Inventory Sources

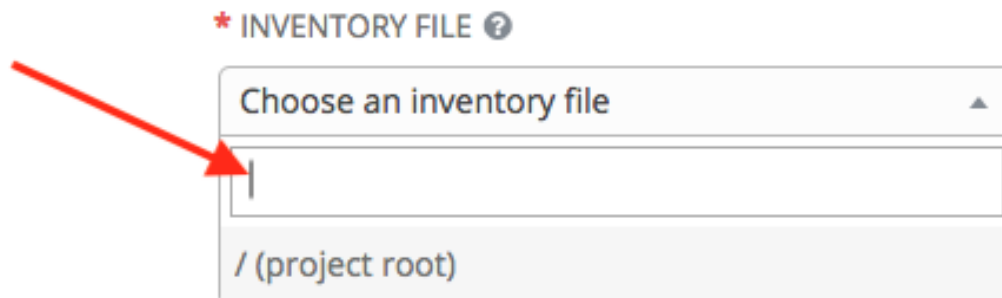
Choose a source which matches the inventory type against which a host can be entered:

- *Sourced from a Project*
- *Amazon Web Services EC2*
- *Google Compute Engine*
- *Microsoft Azure Resource Manager*
- *VMware vCenter*
- *Red Hat Satellite 6*
- *OpenStack*
- *Red Hat Virtualization*
- *Ansible Tower*
- *Custom Script*

Sourced from a Project

An inventory that is sourced from a project means that it uses the SCM type from the project it is tied to. For example, if the project's source is from GitHub, or a Red Hat Insights project, 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.



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 addition to the update options available for cloud inventory sources, you can specify whether or not to update on project changes. Check the **Update on Project Update** option to refresh the inventory from the selected

source after every project update where the SCM revision changes before executing job tasks. For more detail, refer to [Update on Project Update](#) in the *Ansible Tower Administration Guide*.

5. In order to pass to the custom inventory script, you can optionally set environment variables in the **Environment Variables** field.

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, Tower 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 credential (for more information, refer to [Credentials](#)).
If Tower 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 found in `aws_ec2` plugin and used by the inventory update script. 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](#) in the Ansible documentation.

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 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](#).

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 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 found in `azure_rm.ini` and used by the inventory update script. 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.ini inventory script](#) in the Ansible Collections GitHub repo.

The screenshot shows the 'CREATE SOURCE' form for Microsoft Azure Resource Manager. The form is divided into several sections:

- DETAILS** (selected tab):
 - * NAME:** Text input field containing 'Sourced from Azure RM'.
 - DESCRIPTION:** Text input field.
 - * SOURCE:** Dropdown menu with 'Microsoft Azure Resource Manager' selected.
- SOURCE DETAILS:**
 - * CREDENTIAL:** Searchable dropdown menu with 'Inventory Credential' selected.
 - VERBOSITY:** Dropdown menu with '1 (INFO)' selected.
 - HOST FILTER:** Text input field.
 - ENABLED VARIABLE:** Text input field.
 - ENABLED VALUE:** Text input field.
 - UPDATE OPTIONS:** Three checkboxes: 'OVERWRITE', 'OVERWRITE VARIABLES', and 'UPDATE ON LAUNCH', all currently unchecked.
- SOURCE VARIABLES:**
 - Radio buttons for 'YAML' (selected) and 'JSON'.
 - Text area for entering variables, currently containing '1 ---'.

At the bottom right, there are 'CANCEL' and 'SAVE' buttons.

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 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 configure this 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 [VMware Guest inventory source](#) in the Ansible Collections documentation.

Starting with Ansible 2.9, VMWare properties have changed from lower case to camelCase. Tower 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 the primary [documentation for hostvars from VMWare inventory imports](#).

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 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 found in `foreman.ini` and used by the inventory update script. 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 [foreman.ini inventory script](#) in the Ansible Collections GitHub repo.

Note: The variable `want_facts` from `foreman.ini` is hard-coded to `True` and cannot be overridden at this time. If you want to set the `group_patterns`, `group_prefix`, or `want_hostcollections` variables, prefix them with `satellite6`, e.g.: `satellite6_group_prefix: myprefix`

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 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 found in `openstack.yml` and used by the inventory update script. 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 [OpenStack inventory source](#) in the Ansible collections documentation.

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 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](#).

The screenshot shows the 'CREATE SOURCE' form for Red Hat Virtualization. The form is divided into two tabs: 'DETAILS' (selected) and 'SCHEDULES'. The 'DETAILS' tab contains the following fields:

- * NAME:** Text input field containing 'Sourced from RH Virtualization'.
- DESCRIPTION:** Text input field.
- * SOURCE:** Dropdown menu with 'Red Hat Virtualization' selected.
- SOURCE DETAILS:**
 - * CREDENTIAL:** Searchable dropdown menu with 'RHV Credential' selected.
 - VERBOSITY:** Dropdown menu with '1 (INFO)' selected.
 - HOST FILTER:** Text input field.
 - ENABLED VARIABLE:** Text input field.
 - ENABLED VALUE:** Text input field.
 - UPDATE OPTIONS:** Three checkboxes: 'OVERWRITE', 'OVERWRITE VARIABLES', and 'UPDATE ON LAUNCH', all of which are currently unchecked.

At the bottom right of the form, there are 'CANCEL' and 'SAVE' buttons.

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.

Ansible Tower

1. To configure a Ansible Tower-sourced inventory, select **Ansible Tower** from the Source field.
2. The Create Source window expands with the required **Credential** field. Choose from an existing 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](#).

The screenshot shows the 'CREATE SOURCE' form for Ansible Tower. The form is divided into two tabs: 'DETAILS' (selected) and 'SCHEDULES'. The 'DETAILS' tab contains the following fields:

- * NAME:** Text input field containing 'Tower Inventory Source'.
- DESCRIPTION:** Text input field.
- * SOURCE:** Dropdown menu with 'Ansible Tower' selected.
- SOURCE DETAILS:**
 - * CREDENTIAL:** Searchable dropdown menu with 'Inventory Credential' selected.
 - VERBOSITY:** Dropdown menu with '1 (INFO)' selected.
 - HOST FILTER:** Text input field.
 - ENABLED VARIABLE:** Text input field.
 - ENABLED VALUE:** Text input field.
 - UPDATE OPTIONS:** Three checkboxes: 'OVERWRITE', 'OVERWRITE VARIABLES', and 'UPDATE ON LAUNCH', all of which are currently unchecked.

At the bottom right of the form, there are 'CANCEL' and 'SAVE' buttons.

Custom Script

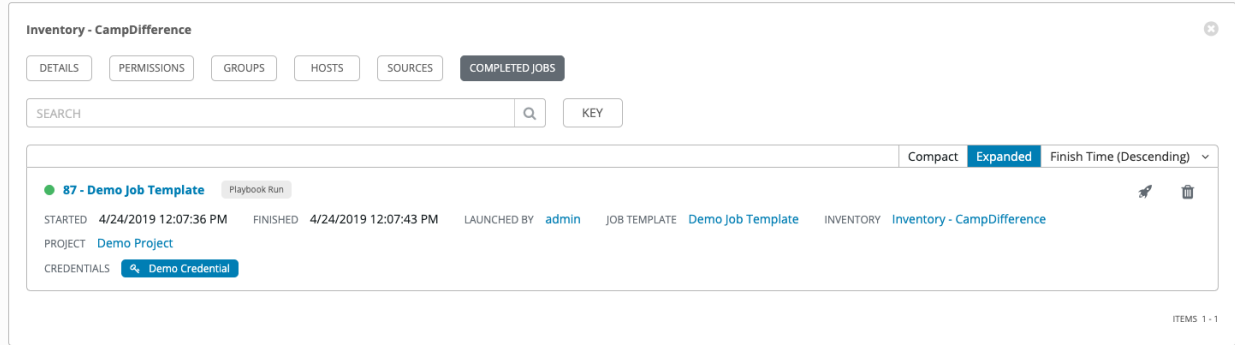
Tower allows you to use a custom dynamic inventory script, if your administrator has added one.

1. To configure a Custom Script-sourced inventory, select **Custom Script** from the Source field.
2. The Create Source window expands with additional fields. Enter the following details:
 - **Credential:** You can optionally provide a credential for custom sources. The kind of credential is limited to cloud and network. Refer to [Custom Credential Types](#) for more information.
 - **Custom Inventory Script:** Required. Choose from an existing Inventory Script (for more information, refer to [Custom Inventory Scripts](#)).
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 **Environment Variables** field to set variables in the environment to be used by the inventory update script. The variables would be specific to the script that you have written. Enter variables using either JSON or YAML syntax. Use the radio button to toggle between the two.

For more information on syncing or using custom inventory scripts, refer to [Custom Inventory Scripts](#) in the *Ansible Tower Administration Guide*.

15.3.5 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.



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 Tower 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
```

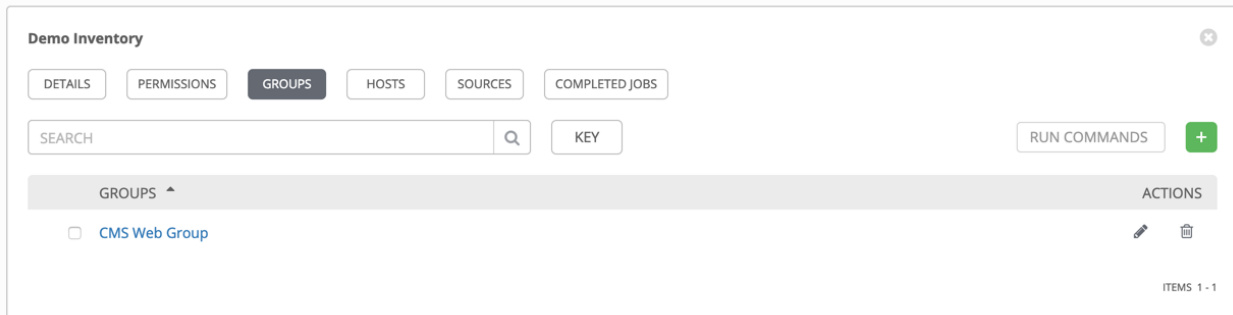
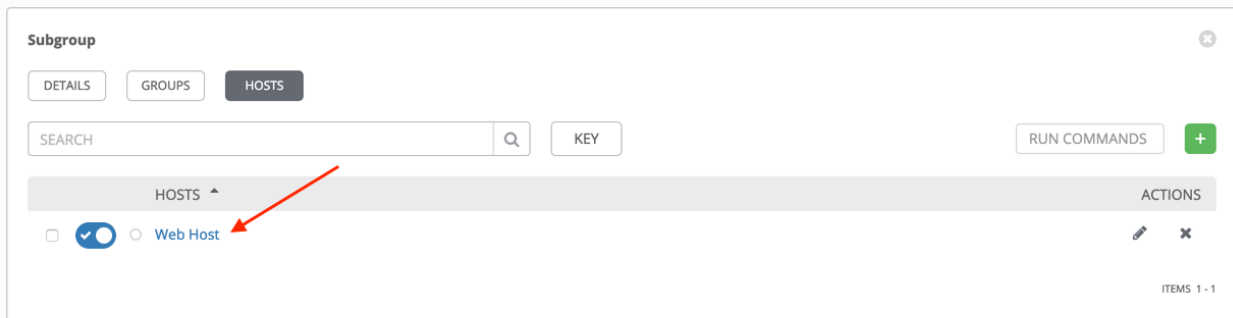
If a search term in `host_filter` is of string type, to make the value a number (e.g. 2.66), or a JSON keyword (e.g. `null`, `true` or `false`) valid, add double quotations around the value to prevent Tower from mistakenly parsing it as a non-string:

```
host_filter=ansible_facts__packages__dnsmasq[]__version="2.66"
```

15.4 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.



2. Click the  button.

The Execute Command window opens.

3. Enter the details for the following fields:

- **Module:** Select one of the modules that Tower 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 Ansible 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.

RUN COMMAND


*MODULE ARGUMENTS LIMIT

*MACHINE CREDENTIAL *VERBOSITY FORKS

SHOW CHANGES ENABLE PRIVILEGE ESCALATION

EXTRA VARIABLES



4. Click the  button to run this ad hoc command.

The results display in the Details pane and Standard Out window.

DETAILS

STATUS ● Successful

STARTED 5/28/2019 5:58:28 PM

FINISHED 5/28/2019 5:58:30 PM

JOB TYPE Run

LAUNCHED BY admin

INVENTORY Database Servers

CREDENTIAL

LIMIT CMS Web Group:Subgroup

EXECUTION NODE localhost

INSTANCE GROUP tower

EXTRA VARIABLES

ping ELAPSED 00:00:01

SEARCH


```

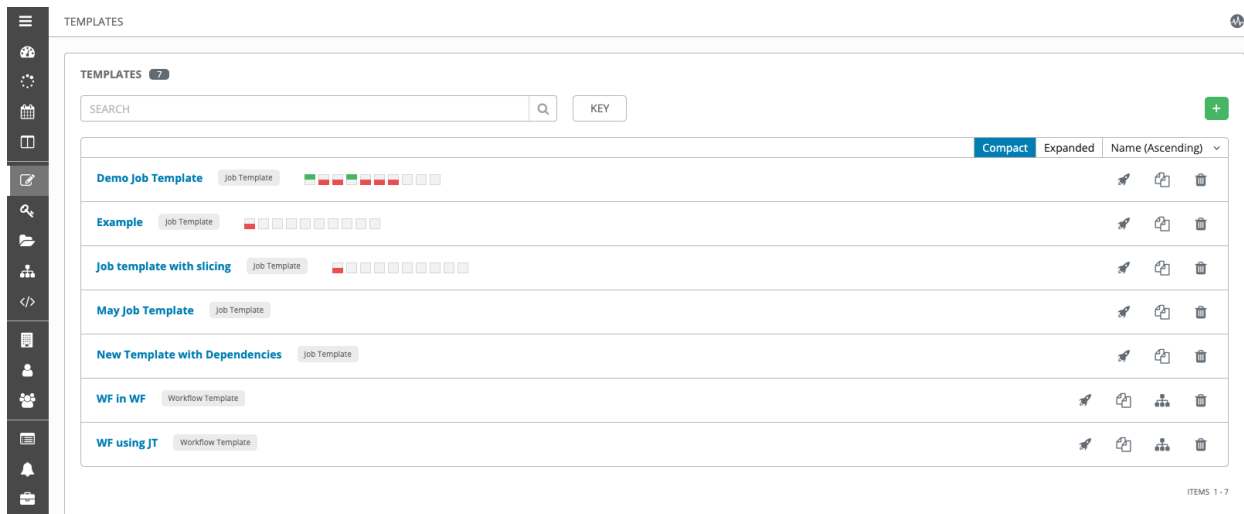
1 [DEPRECATION WARNING]: The TRANSFORM_INVALID_GROUP_CHARS settings is set to
2 allow bad characters in group names by default, this
3 will change, but still be
4 user configurable on deprecation. This feature will
5 be removed in version 2.10.
6 Deprecation warnings can be disabled by setting dep
7 reprecation_warnings=False in
8 ansible.cfg.
9 [WARNING]: Invalid characters were found in group n
10 ames but not replaced, use
11 -vvvv to see details
12
13 [WARNING]: Could not match supplied host pattern,
14 ignoring: CMS
15
16 [WARNING]: Could not match supplied host pattern,
17 ignoring: Web
18
19 [WARNING]: Could not match supplied host pattern,
20 ignoring: Group
                
```





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. While the REST API allows for the execution of jobs directly, Tower requires that you first create a job template.

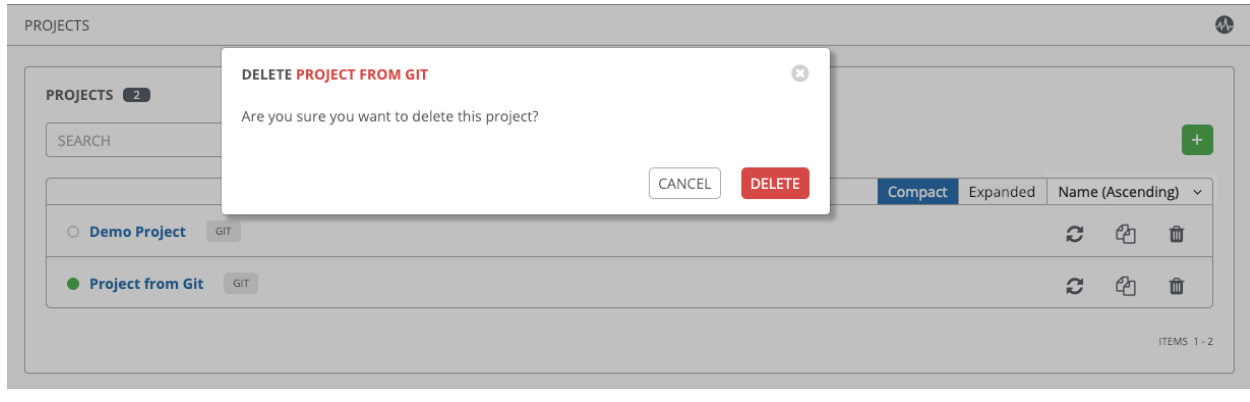



The () menu opens a list of the job templates that are currently available. The default view is collapsed (Compact), showing the template name, template type, and the statuses of the jobs that ran using that template, but you can click **Expanded** to view more information. This list is sorted alphabetically 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 (), copy (), and remove () a job template. 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:



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.

16.1 Create a Job Template

To create a new job template:

1. Click the  button then select **Job Template** from the menu list.

The screenshot shows the configuration interface for a job template named "Job template with slicing". The interface includes several tabs: DETAILS (selected), PERMISSIONS, NOTIFICATIONS, COMPLETED JOBS, SCHEDULES, and ADD SURVEY. The configuration fields are organized into a grid:

- Name:** Job template with slicing
- Description:** (empty)
- Job Type:** Run (dropdown)
- Inventory:** King PLC (dropdown)
- Project:** Project from Git (dropdown)
- Playbook:** gen_host_status.yml (dropdown)
- Credentials:** (empty)
- Forks:** 0 (dropdown)
- Limit:** (empty)
- Verbosity:** 0 (Normal) (dropdown)
- Job Tags:** (empty)
- Skip Tags:** (empty)
- Labels:** (empty)
- Instance Groups:** (empty)
- Job Slicing:** 1 (dropdown)
- Timeout:** 0 (dropdown)
- Show Changes:** (toggle off)
- Options:**
 - ENABLE PRIVILEGE ESCALATION
 - ENABLE PROVISIONING CALLBACKS
 - ENABLE WEBHOOK
 - ENABLE CONCURRENT JOBS
 - ENABLE FACT CACHE
- Extra Variables:** (YAML/JSON tabs)


```

1 ---
2 ansible_ssh_user: ec2
3 ansible_connection: local
      
```


At the bottom right, there are three buttons: LAUNCH (blue), CANCEL (white), and SAVE (green).

2. Enter the appropriate details into the following fields:

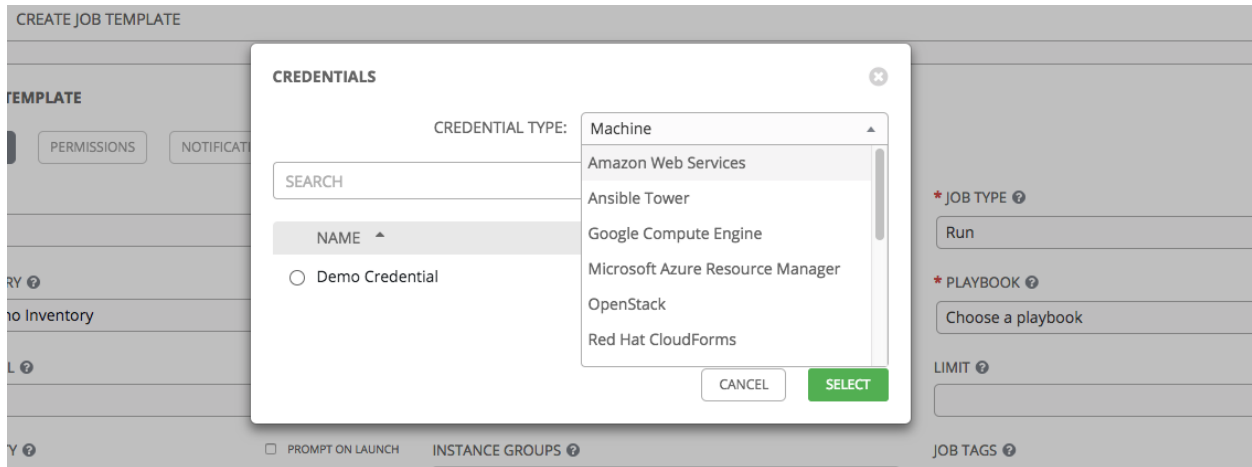
- **Name:** Enter a name for the job.
- **Description:** Enter an arbitrary description as appropriate (optional).
- **Job Type:** Choose a job type:
 - **Run:** Execute the playbook when launched, running Ansible tasks on the selected hosts.
 - **Check:** Perform a “dry run” of the playbook and report 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.
 - **Prompt on Launch:** If selected, even if a default value is supplied, you will be prompted upon launch to choose a job type of run or check.

Note: More information on the *Check* job type, refer to [Using check mode](#) in the Ansible User Guide.

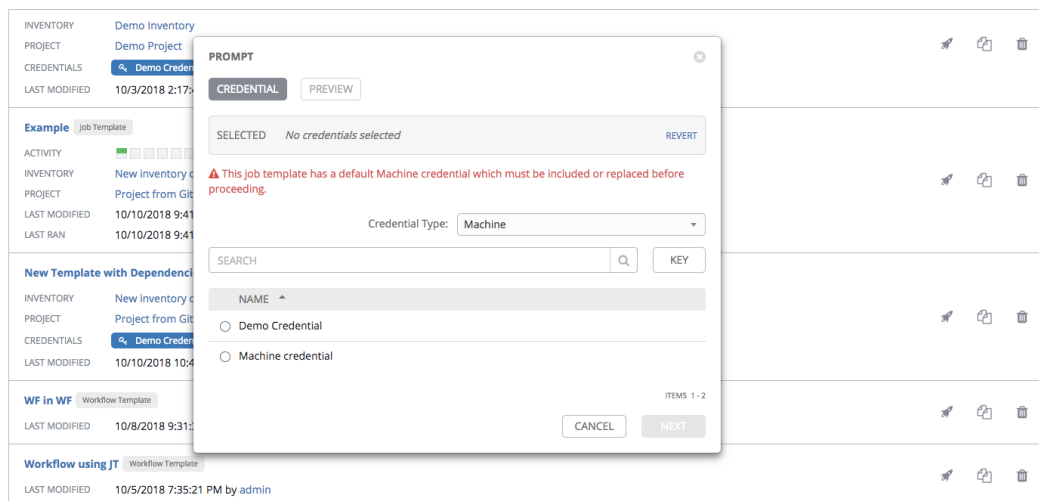
- **Inventory:** Choose the inventory to be used with this job template from the inventories available to the currently logged in Tower user.
- **Prompt on Launch:** If selected, even if a default value is supplied, you will be prompted upon launch to choose an inventory to run this job template against.
- **Project:** Choose the project to be used with this job template from the projects available to the currently logged in Tower user.

- **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](#).
 - **Prompt on Launch:** If selected, even if a default value is supplied, you will be prompted upon launch to choose an SCM branch.
- **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.
- **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.

Note: Some credential types are not listed because they do not apply to certain job templates.



- **Prompt on Launch:** If selected, upon launching a job template that has a default machine credential, you will not be able to remove the default machine credential in the Prompt dialog without replacing it with another machine credential before it can launch. Alternatively, you can add more credentials as you see fit. Below is an example of such a message:



- **Forks:** The number of parallel or simultaneous processes to use while executing the playbook. A value of zero uses the Ansible default setting, which is 5 parallel processes unless overridden in `/etc/ansible/ansible.cfg`.
- **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".
- **Prompt on Launch:** If selected, even if a default value is supplied, you will be prompted upon launch to choose a limit.

Note: For more information and examples refer to [Patterns](#) in the Ansible documentation.

- **Verbosity:** Control the level of output Ansible produces as the playbook executes. Choose the verbosity from Normal to various Verbose or Debug settings. This only appears in the "details" report view. Verbose logging includes the output of all commands. Debug logging is exceedingly verbose and includes information on SSH operations that can be useful in certain support instances. Most users do not need to see debug mode output.

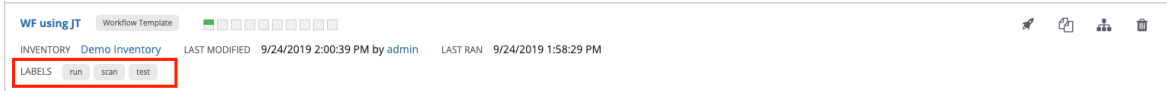
Warning: Verbosity 5 causes Tower to block heavily when jobs are running, which could delay reporting that the job has finished (even though it has) and can cause the browser tab to lock up.


- **Prompt on Launch:** If selected, even if a default value is supplied, you will be prompted upon launch to choose a verbosity.
- **Job Tags:** Provide a comma-separated list of playbook tags to specify what parts of the playbooks should be executed.
- **Prompt on Launch:** If selected, even if a default value is supplied, you will be prompted upon launch to choose a job tag.
- **Skip Tags:** Provide a comma-separated list of playbook tags to skip certain tasks or parts of the playbooks to be executed.
- **Prompt on Launch:** If selected, even if a default value is supplied, you will be prompted upon launch to choose tag(s) to skip.

Note: For more information on tags and examples, refer to [Tags](#) in the Ansible documentation.

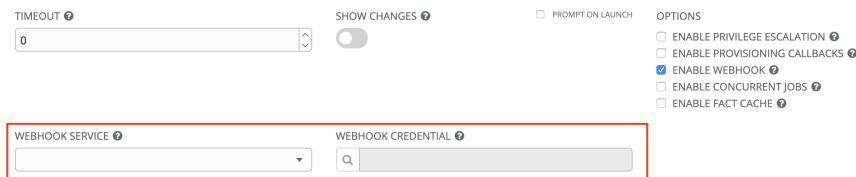
- **Labels:** Supply optional labels that describe this job template, such as "dev" or "test". Labels can be used to group and filter job templates and completed jobs in the Tower display.
- Labels are created when they are added to the Job Template. Labels are associated to a single Organization using the Project that is provided in the Job Template. Members of the Organization can create labels on a Job Template if they have edit permissions (such as admin role).
- Once the Job Template is saved, the labels appear in the Job Templates overview in the *Expanded* view.
- Click on the "x" beside a label to remove it. When a label is removed, and is no longer associated with a Job or Job Template, the label is permanently deleted from the list of Organization labels.
- Jobs inherit labels from the Job Template at the time of launch. If a label is deleted from a Job Template, it is also deleted from the Job.

LABELS ?



- **Instance Groups:** Click the  button to open a separate window. Choose the instance groups on which you want to run this job template. If the list is extensive, use the search to narrow the options.
- **Job Slicing:** Specify the number of slices you want this job template to run. Each slice will run the same tasks against a portion of the inventory. For more information about job slices, see [Job Slicing](#).
- **Timeout:** Allows you to specify the length of time (in seconds) Tower may run the task before it is canceled. Defaults to 0 for no job timeout.
- **Show Changes:** Allows you to see the changes made by Ansible tasks.
- **Prompt on Launch:** If selected, even if a default value is supplied, you will be prompted upon launch to choose whether or not to show changes.
- **Options:** Supply optional labels that describe this job template, such as “dev” or “test”. Labels can be used to group and filter job templates and completed jobs in the Tower display.
- **Enable Privilege Escalation:** If enabled, run this playbook as an administrator. This is the equivalent of passing the `--become` option to the `ansible-playbook` command.
- **Enable Provisioning Callbacks:** Enable a host to call back to Tower via the Tower 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 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 display and remain present while viewing or editing.

- **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 Tower. This must be configured in the settings on the webhook service in order for Tower to accept webhooks from this service.

For additional information on setting up webhooks, see [Working with Webhooks](#).

- **Enable Concurrent Jobs:** Allow 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 [Ansible Tower Capacity Determination and Job Impact](#) for additional information.
- **Enable Fact Cache:** When enabled, Tower will activate an Ansible fact cache plugin for all hosts in an inventory related to the job running.

• **Extra 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, [Defining variables at runtime](#).
- 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
```

For more information about extra variables, refer to [Extra Variables](#).

- **Prompt on Launch:** If selected, even if a default value is supplied, you will be prompted upon launch to choose command line variables.

Note: If you want to be able to specify `extra_vars` on a schedule, you must select **Prompt on Launch** for **EXTRA VARIABLES** on the job template, or enable a survey on the job template, then those answered survey questions become `extra_vars`.

3. When you have completed configuring the details of the job template, click **Save**.

Saving the template does not exit the job template page but remains on the Job Template Details view for further editing, if necessary. After saving the template, you can click **Launch** to launch the job, or proceed with adding more attributes about 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.

Job template with slicing

DETAILS | PERMISSIONS | NOTIFICATIONS | COMPLETED JOBS | SCHEDULES | ADD SURVEY

* NAME: Job template with slicing

DESCRIPTION: [Empty]

* JOB TYPE: Run

* INVENTORY: King PLC

* PROJECT: Project from Git

* PLAYBOOK: gen_host_status.yml

CREDENTIALS: [Empty]

FORKS: 0

LIMIT: [Empty]

* VERBOSITY: 0 (Normal)

JOB TAGS: [Empty]

SKIP TAGS: [Empty]

LABELS: [Empty]

INSTANCE GROUPS: [Empty]

JOB SLICING: 1

TIMEOUT: 0

SHOW CHANGES:

OPTIONS:

- ENABLE PRIVILEGE ESCALATION
- ENABLE PROVISIONING CALLBACKS
- ENABLE WEBHOOK
- ENABLE CONCURRENT JOBS
- ENABLE FACT CACHE

EXTRA VARIABLES (YAML | JSON):

```

1 ---
2 ansible_ssh_user: ec2
3 ansible_connection: local
    
```

LAUNCH | CANCEL | SAVE

You can verify the template is saved when the newly created template appears on the list of templates at the bottom of the screen.

TEMPLATES 3

SEARCH [] KEY [] +

Template Name	Status	Actions
Demo Job Template	Job Template [Progress Bar]	[Refresh] [Copy] [Delete]
Job template with slicing	Job Template [Progress Bar]	[Refresh] [Copy] [Delete]
May Job Template	Job Template [Progress Bar]	[Refresh] [Copy] [Delete]

ITEMS 1 - 3

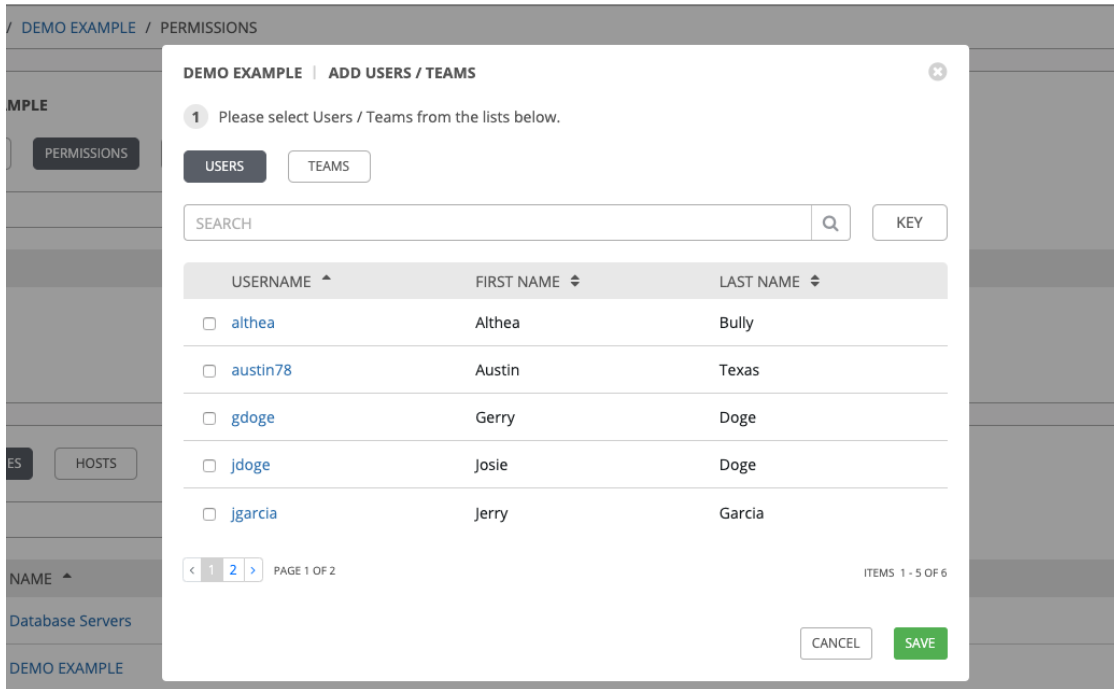
16.2 Add Permissions

The **Permissions** tab allows you to review, grant, edit, and remove associated permissions for users as well as team members. To assign permissions to a particular user for this resource:

1. Click the **Permissions** tab.



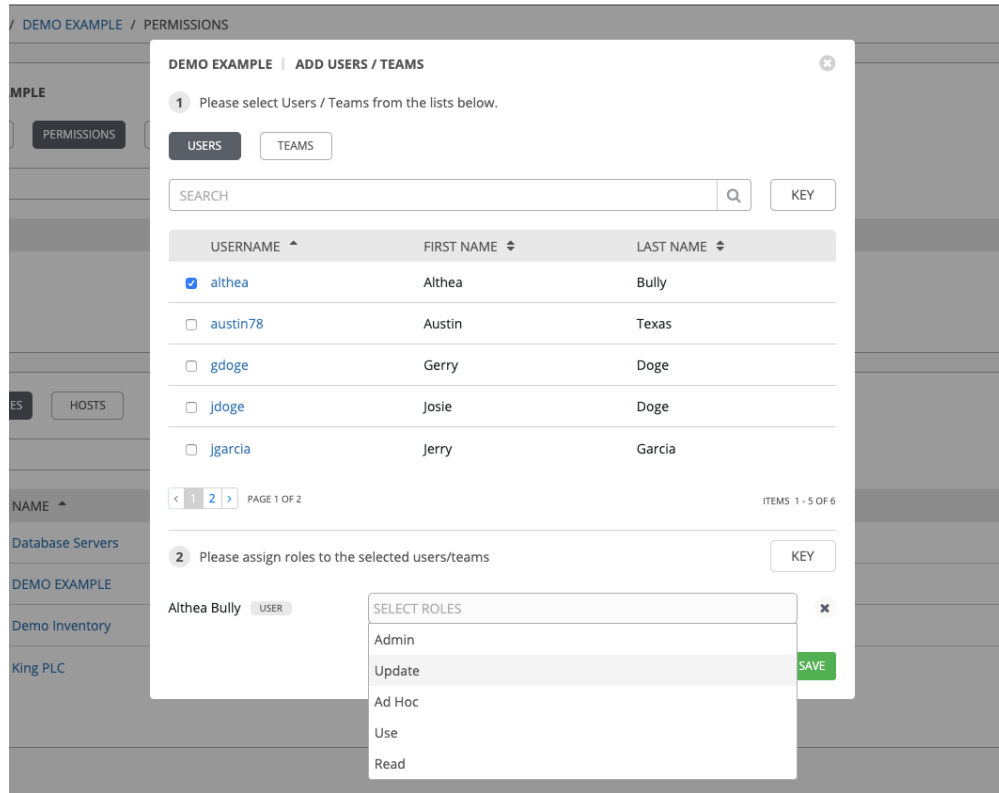
2. Click the  button to open the Add Users/Teams window.



3. Specify the users or teams that will have access then assign them specific roles:
 - a. Click to select one or multiple check boxes beside the name(s) of the user(s) or team(s) to select them.

Note: You can select multiple users and teams at the same time by navigating between the **Users** and **Teams** tabs without saving.

After selections are made, the window expands to allow you to select a role from the drop-down menu list for each user or team you chose.



The example above shows options associated with inventories. Different resources have different options available:

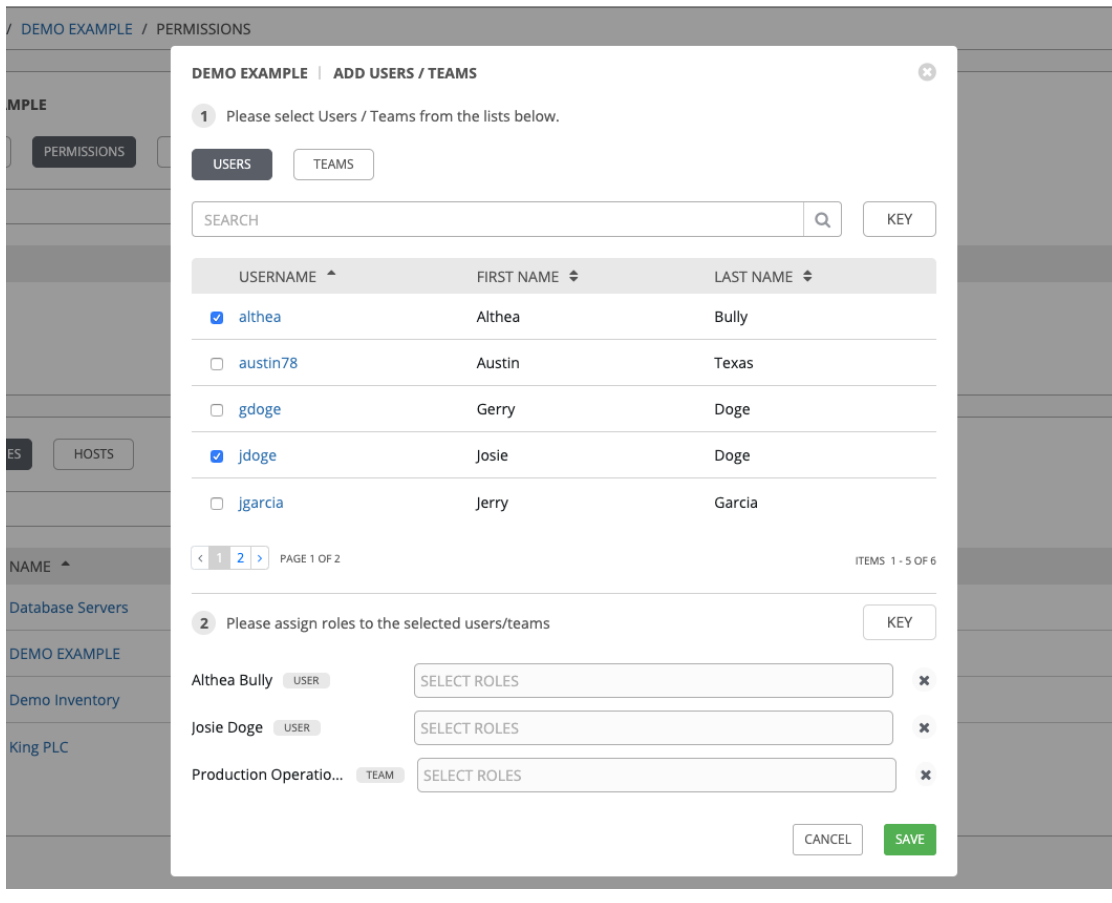
- **Admin** allows read, run, and edit privileges (applies to all resources)
- **Use** allows use of a resource in a job template (applies all resources except job templates)
- **Update** allows updating of project via the SCM Update (applies to projects and inventories)
- **Ad Hoc** allows use of Ad Hoc commands (applies to inventories)
- **Execute** allows launching of a job template (applies to job templates)
- **Read** allows view-only access (applies to all resources)

Tip: Use the **Key** button in the roles selection pane to display a description of each of the roles. For more information, refer to the [Roles](#) section of this guide.

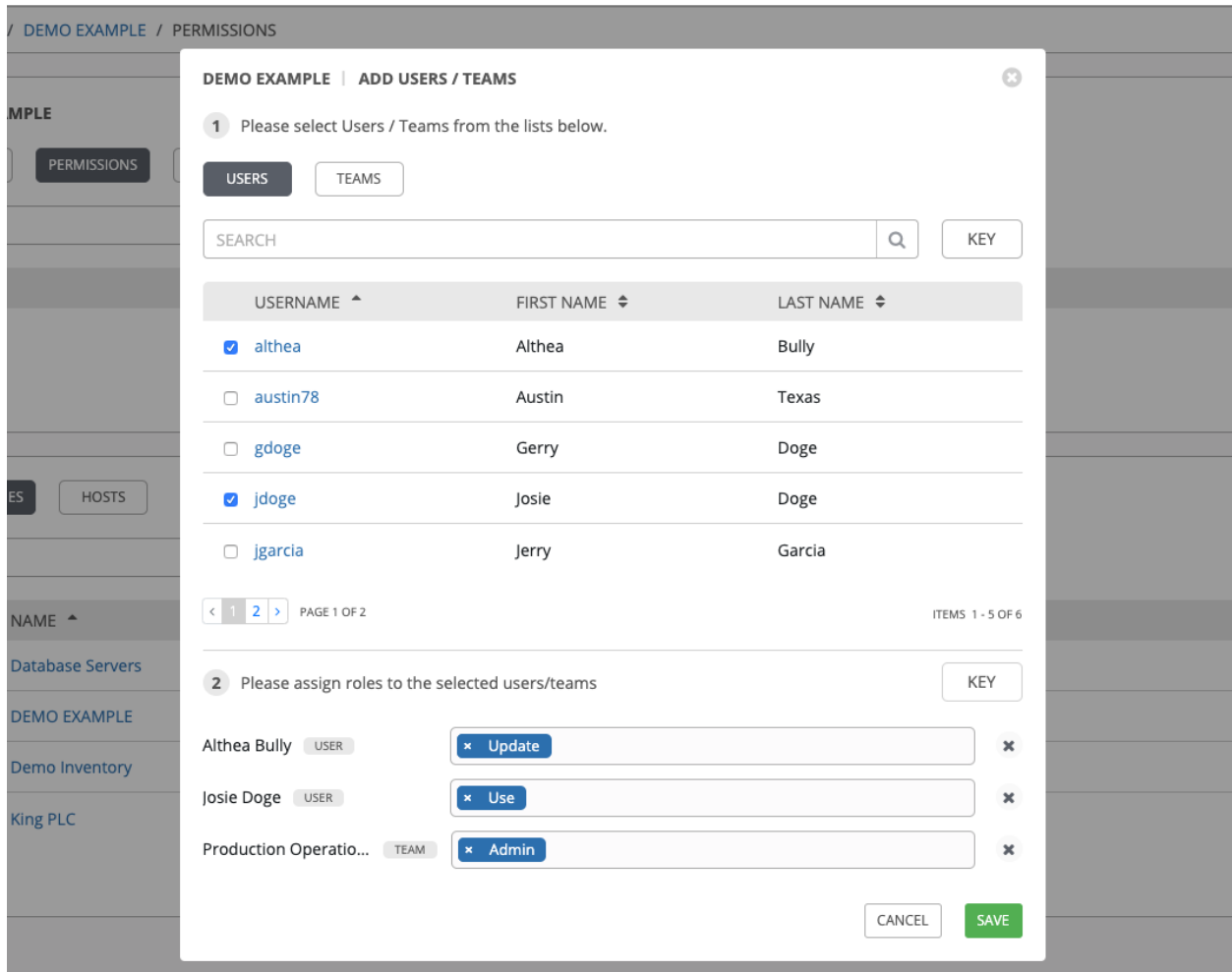
- Select the role to apply to the selected user or team.

Note:

You can assign roles to multiple users and teams by navigating between the **Users** and **Teams** tabs without saving.



4. Review your role assignments for each user and team.



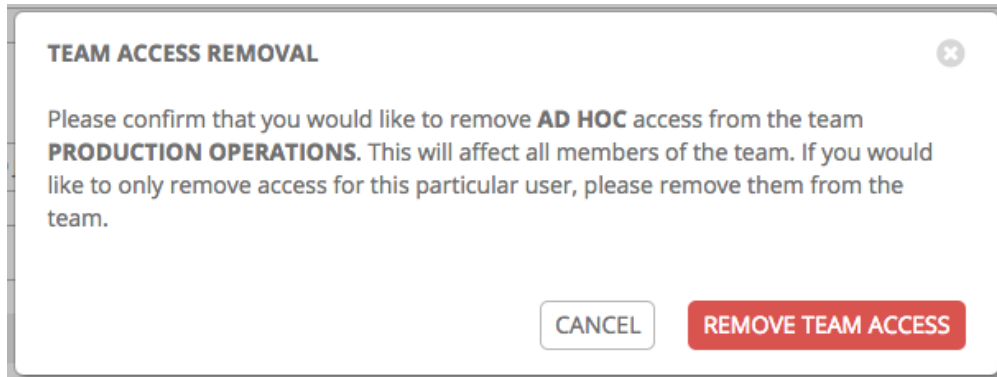
5. Click **Save** when done, and the Add Users/Teams window closes to display the updated roles assigned for each user and team.

USER	ROLE	TEAM ROLES
admin	SYSTEM ADMINISTRATOR	
althea	AD HOC, SYSTEM AUDITOR	USE
jdoge	UPDATE, USE	
mags3707	SYSTEM ADMINISTRATOR	AD HOC, ADMIN, USE
yser	SYSTEM AUDITOR	

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

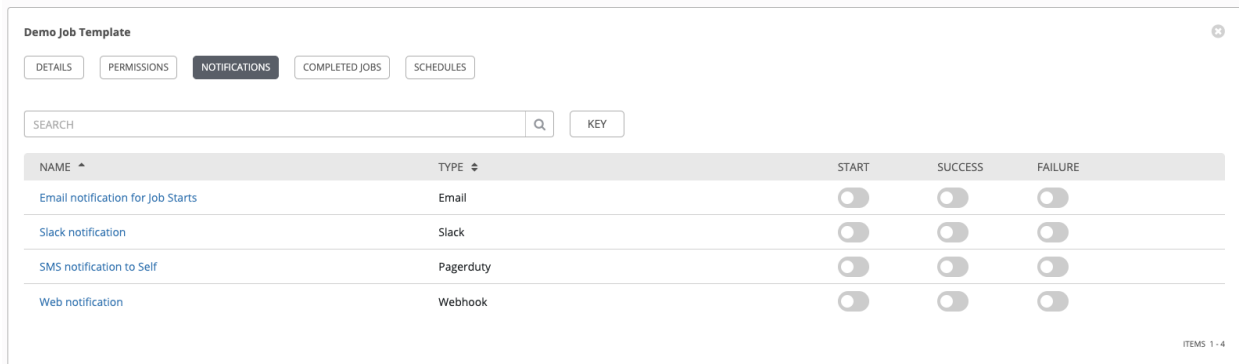
USER	ROLE	TEAM ROLES
admin	SYSTEM ADMINISTRATOR	
althea	AD HOC, SYSTEM AUDITOR	USE
jdoge	UPDATE, USE	
mags3707	SYSTEM ADMINISTRATOR	AD HOC, ADMIN, USE
yser	SYSTEM AUDITOR	

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



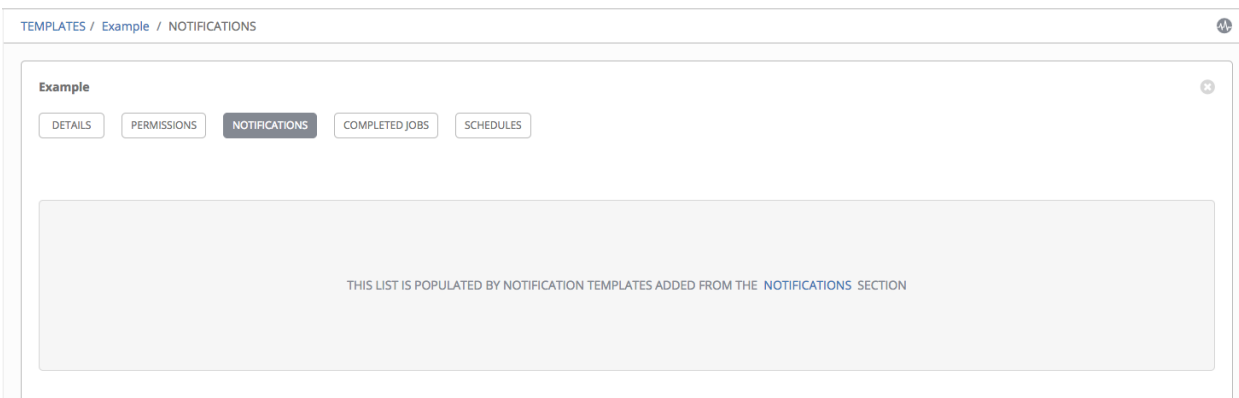
16.3 Work with Notifications

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



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 **NOTIFICATIONS** link from inside the gray box to create a new notification.



Refer to [Notification Types](#) for additional details on configuring various notification types.

16.4 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.

Sliced jobs that display on this list are labeled accordingly, with the number of sliced jobs that have run:

16.5 Scheduling

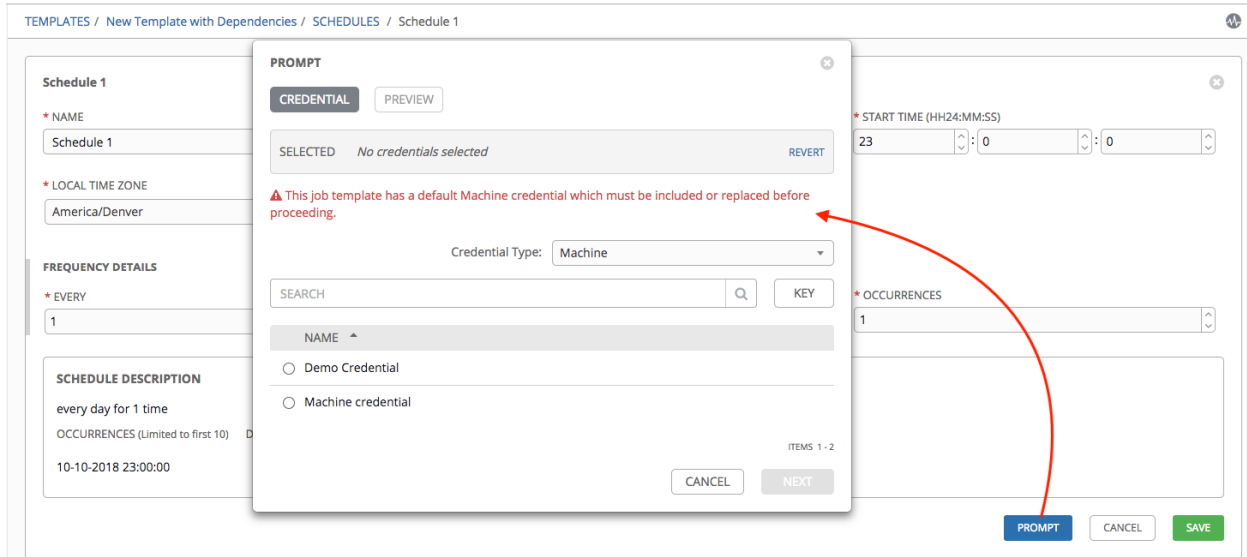
Access the schedules for a particular job template from the **Schedules** tab.

16.5.1 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 **EXTRA VARIABLES** on the job template, or a enable a survey on the job template, then those answered survey questions become `extra_vars`.

16.6 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

ADD SURVEY

answer way. Surveys also allow for validation of user input. Click the **ADD SURVEY** button to create a survey.

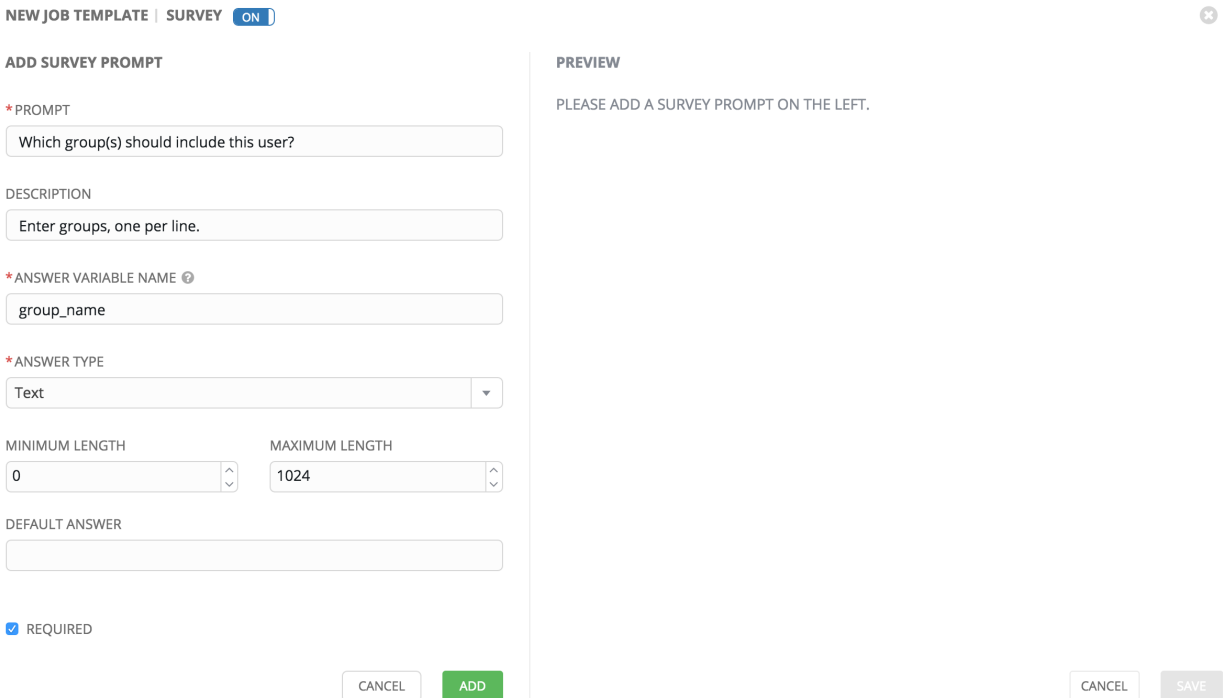
Use cases for surveys are numerous. An example might be if operations wanted to give developers a “push to stage” button they could run without advanced Ansible knowledge. When launched, this task could prompt for answers to questions such as, “What tag should we release?”

Many types of questions can be asked, including multiple-choice questions.

16.6.1 Create a Survey

To create a survey:

1. Click on the  button to bring up the **Add Survey** window.




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**: 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  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.

The screenshot shows the 'NEW JOB TEMPLATE | SURVEY ON' interface. On the left, the 'ADD SURVEY PROMPT' form includes fields for 'PROMPT', 'DESCRIPTION', 'ANSWER VARIABLE NAME', and 'ANSWER TYPE' (a dropdown menu). There is a 'REQUIRED' checkbox checked and 'CANCEL' and 'ADD' buttons at the bottom. On the right, the 'PREVIEW' pane displays the question: '*WHICH GROUP(S) SHOULD INCLUDE THIS USER?' with the instruction 'Enter groups, one per line.' Below the text is a list input field with a grid icon on the left and edit/delete icons on the right. A red arrow points to the grid icon with the text 'Click and hold down to drag the question to reorder it.' At the bottom right of the preview pane are 'CANCEL' and 'SAVE' buttons.

16.6.2 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 ("").

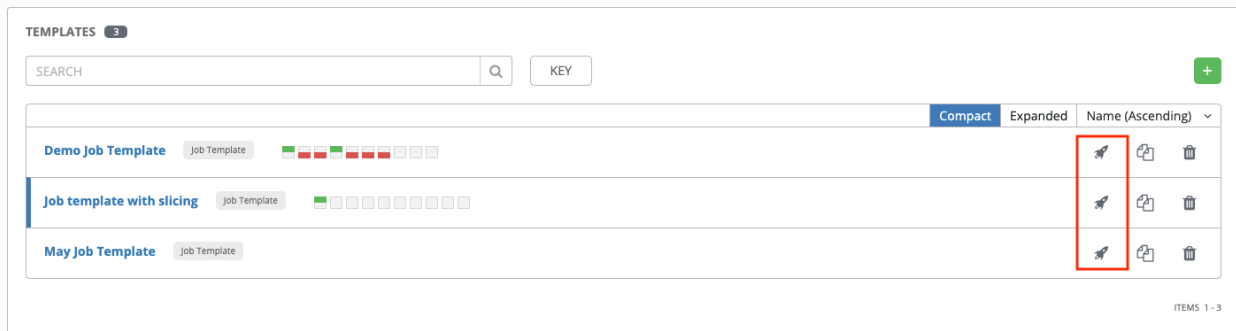
16.7 Launch a Job Template

A major benefit of Ansible Tower is the push-button deployment of Ansible playbooks. You can easily configure a template within Tower 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 Tower playbooks written by others.

Launch a job template by any of the following ways:

- Access the job template list from the  navigational link or while in the Job Template Details view, scroll to the bottom to access the  button from the list of templates.



- 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 JOB | HELLO WORLD ✕

OTHER PROMPTS SURVEY

JOB TAGS

INVENTORY CREDENTIAL

Demo Inventory Demo Credential

CANCEL NEXT

LAUNCH JOB | HELLO WORLD ✕

OTHER PROMPTS SURVEY

*** WHICH GROUP(S) SHOULD INCLUDE THIS USER?**

Enter groups, one per line.

INVENTORY CREDENTIAL

Demo Inventory Demo Credential

CANCEL LAUNCH

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, Tower automatically adds the following variables to the job environment:

- `tower_job_id`: The Job ID for this job run
- `tower_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.
- `tower_job_template_id`: The Job Template ID that this job run uses
 - `tower_job_template_name`: The Job Template name that this job uses
 - `tower_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`)
 - `tower_user_email`: The user email of the Tower user that started this job. This is not available for callback or scheduled jobs.
 - `tower_user_first_name`: The user’s first name of the Tower user that started this job. This is not available for callback or scheduled jobs.
 - `tower_user_id`: The user ID of the Tower user that started this job. This is not available for callback or scheduled jobs.
 - `tower_user_last_name`: The user’s last name of the Tower user that started this job. This is not available for callback or scheduled jobs.
 - `tower_user_name`: The user name of the Tower user that started this job. This is not available for callback or scheduled jobs.
 - `tower_schedule_id`: If applicable, the ID of the schedule that launched this job
 - `tower_schedule_name`: If applicable, the name of the schedule that launched this job
 - `tower_workflow_job_id`: If applicable, the ID of the workflow job that launched this job
 - `tower_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.
 - `tower_inventory_id`: If applicable, the ID of the inventory this job uses
 - `tower_inventory_name`: If applicable, the name of the inventory this job uses

All variables are also given an “awx” prefix, for example, `awx_job_id`.

Upon launch, Tower 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 *Ansible Tower 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.

						Compact	Expanded	Finish Time (Descending) ▾	
118 - Demo Job Template	Playbook Run	Slice Job 2/2							
STARTED	5/13/2019 9:16:11 PM	FINISHED	5/13/2019 9:16:20 PM	LAUNCHED BY	admin	WORKFLOW JOB	Demo Job Template	JOB TEMPLATE	Demo Job Template
INVENTORY	Demo Inventory	PROJECT	Demo Project						
CREDENTIALS	Demo Credential								
117 - Demo Job Template	Playbook Run	Slice Job 1/2							
STARTED	5/13/2019 9:16:11 PM	FINISHED	5/13/2019 9:16:18 PM	LAUNCHED BY	admin	WORKFLOW JOB	Demo Job Template	JOB TEMPLATE	Demo Job Template
INVENTORY	Demo Inventory	PROJECT	Demo Project						
CREDENTIALS	Demo Credential								

16.8 Copy a Job Template

Ansible Tower 3.0 introduced the ability to 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  navigational link or while in the Job Template Details view, scroll to the bottom to access it from the list of templates.

TEMPLATES 3				Compact	Expanded	Name (Ascending) ▾
SEARCH	Q	KEY				
Demo Job Template	Job Template	■ ■ ■ ■ ■ ■ ■ ■ ■ ■				
Job template with slicing	Job Template	■ ■ ■ ■ ■ ■ ■ ■ ■ ■				
May Job Template	Job Template	■ ■ ■ ■ ■ ■ ■ ■ ■ ■				

ITEMS 1 - 3

2. Click the  button.

A new template opens with the name of the template from which you copied and a timestamp.

3. 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.
4. Click **Save** when done.

16.9 Scan Job Templates

Scan jobs are no longer supported starting with Ansible Tower 3.2. This system tracking feature was used as a way to capture and store facts as historical data. Facts are now stored in Tower via fact caching. For more information, see [Fact Caching](#).

If you have Job Template Scan Jobs in your system prior to Ansible Tower 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.

16.9.1 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:

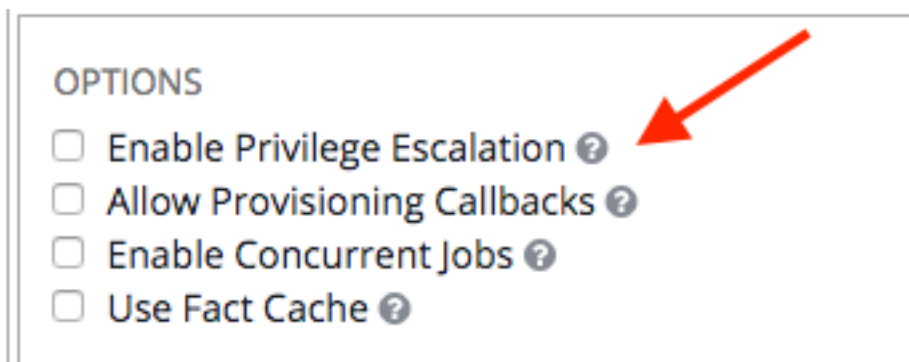
```
- hosts: all
  vars:
    scan_use_checksum: false
    scan_use_recursive: false
  tasks:
    - scan_packages:
    - scan_services:
    - scan_files:
      paths: '{{ scan_file_paths }}'
      get_checksum: '{{ scan_use_checksum }}'
      recursive: '{{ scan_use_recursive }}'
      when: scan_file_paths 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:



Note: If you maintained scan job templates in Ansible Tower 3.1.x and then upgrade to Ansible Tower 3.2, a new “Tower Fact Scan - Default” project is automatically created for you. This project contains the old scan playbook previously used in earlier versions of Ansible Tower.

16.9.2 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
- CentOS 5, 6, & 7
- Ubuntu 16.04 (Support for Ubuntu 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.

16.9.3 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
```

CentOS 5 or Red Hat Enterprise Linux 5 may also need the `simplejson` package installed.

16.9.4 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 [Ansible module development](#) section of the Ansible documentation for more information.

16.10 Fact Caching

Tower 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 Configure Tower interface under the Jobs tab:

The screenshot shows the 'JOBS' configuration page in Ansible Tower. The 'PER-HOST ANSIBLE FACT CACHE TIMEOUT' field is highlighted with a red box. The value is currently set to 0. Other visible settings include 'JOB EXECUTION PATH' set to /tmp, 'MAXIMUM SCHEDULED JOBS' set to 10, 'DEFAULT JOB TIMEOUT' set to 0, 'DEFAULT INVENTORY UPDATE TIMEOUT' set to 0, and 'MAXIMUM NUMBER OF FORKS PER JOB' set to 200. The 'ENABLE JOB ISOLATION' checkbox is checked.

Upon launching a job that uses fact cache (`use_fact_cache=True`), Tower will store all `ansible_facts` associated with each host in the inventory associated with the job. The Ansible Fact Cache plugin that ships with Ansible Tower 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, Tower 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 Tower'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 Tower 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.

16.10.1 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 Tower's fact caching feature. It is recommended to use the fact caching module that comes with Ansible Tower. 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

- ENABLE PRIVILEGE ESCALATION ?
- ENABLE PROVISIONING CALLBACKS ?
- ENABLE WEBHOOK ?
- ENABLE CONCURRENT JOBS ?
- ENABLE FACT CACHE ?

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
```

The API endpoint for fact caching can be found at: `http://<Tower server name>/api/v2/hosts/x/ansible_facts`.

16.11 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*

16.11.1 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 Tower 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 }}"
```

16.11.2 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 Tower without having to set the `aws_access_key_id` or `aws_secret_access_key` module options.

16.11.3 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 Tower without having to set the `service_account_email`, `project_id`, or `pem_file` module options.

16.11.4 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 Tower without having to set the `subscription_id` or `management_cert_path` module options.

16.11.5 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: centosTemplate
```

(continues on next page)

(continued from previous page)

```
cluster: MainCluster
resource_pool: "/Resources"
vm_extra_config:
  folder: MyFolder
```

16.12 Provisioning Callbacks

Provisioning callbacks are a feature of Tower 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 tower console. Please note that provisioning callbacks are *only* used to run playbooks on the calling host. Provisioning callbacks are meant for cloud bursting, ie: 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 Tower 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 *Allow Provisioning Callbacks* checkbox in the Job Template. This displays the **Provisioning Callback URL** for this job template.

Note: If you intend to use Tower’s provisioning callback feature with a dynamic inventory, Update on Launch should be set for the inventory group used in the Job Template.

<p>OPTIONS</p> <p><input type="checkbox"/> Enable Privilege Escalation </p> <p><input checked="" type="checkbox"/> Allow Provisioning Callbacks </p>	<p>PROVISIONING CALLBACK URL </p> <p><input type="text" value="https://10.42.0.42:443/api/v1/job_templates/5/callb"/></p>	<p>HOST CONFIG KEY </p> <p><input type="text" value=""/></p>
--	---	--

Callbacks also require a Host Config Key, to ensure that foreign hosts with the URL cannot request configuration.

Click the  button to create a unique host key for this callback, or enter your own 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:

```
http://<TOWER_SERVER_NAME>/api/v2/job_templates/1/callback/
```

The ‘1’ in this sample URL is the job template ID in Tower.

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":
→"cfbaae23-81c0-47f8-9a40-44493b82f06a"}' \
  https://<TOWER_SERVER_NAME>/api/v2/job_templates/1/callback/
```

The requesting host must be defined in your inventory for the callback to succeed. If Tower 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, Tower 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 Tower - `/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      Tower server (e.g. https://tower.example.com) (required)
  -k      Allow insecure SSL connections and transfers
  -c      Host config key (required)
  -t      Job template ID (required)
  -e      Extra variables
  -s      Number of seconds between retries (default: 60)
```

This script is intelligent in that 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 Tower, 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 Tower, 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.

16.12.1 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": "5a8ec154832b780b9bdef1061764ae5a", "extra_
    ↪vars": "{\"foo\": \"bar\"}"}' \
    http://<TOWER_SERVER_NAME>/api/v2/job_templates/1/callback
```

For more information, refer to [Launching Jobs with Curl](#).

16.13 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 Tower. 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.

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 Ansible Tower as it compares to variable precedence in Ansible.

Ansible Tower 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

16.13.1 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`.

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 Tower'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, Tower 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 Tower User Interface will redirect to the appropriate screen to display the status of the run.

17.1 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:

		Compact	Expanded	Finish Time (Descending) ▾
118 - Demo Job Template	Playbook Run	Slice Job 2/2		
STARTED	5/13/2019 9:16:11 PM	FINISHED	5/13/2019 9:16:20 PM	LAUNCHED BY admin
WORKFLOW JOB	Demo Job Template	JOB TEMPLATE	Demo Job Template	
INVENTORY	Demo Inventory	PROJECT	Demo Project	
CREDENTIALS	Demo Credential			
117 - Demo Job Template	Playbook Run	Slice Job 1/2		
STARTED	5/13/2019 9:16:11 PM	FINISHED	5/13/2019 9:16:18 PM	LAUNCHED BY admin
WORKFLOW JOB	Demo Job Template	JOB TEMPLATE	Demo Job Template	
INVENTORY	Demo Inventory	PROJECT	Demo Project	
CREDENTIALS	Demo Credential			

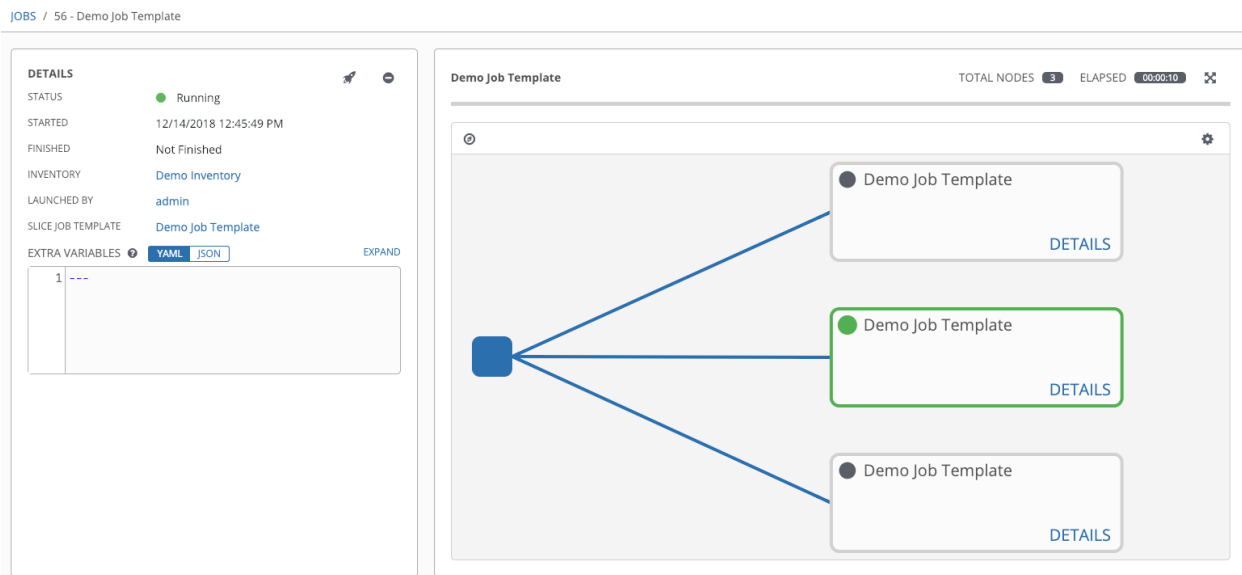
- These sliced jobs follow normal scheduling behavior (number of forks, queuing due to capacity, assignment to instance groups based on inventory mapping).
- Sliced job templates with prompts and/or extra variables behave the same as standard job templates, applying all variables and limits to the entire 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 Tower will not attempt to discover or account for playbooks that fail when run as slice jobs.

17.2 Job slice execution behavior

When jobs are sliced, they can run on any Tower 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.



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 Tower 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*)

17.3 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:

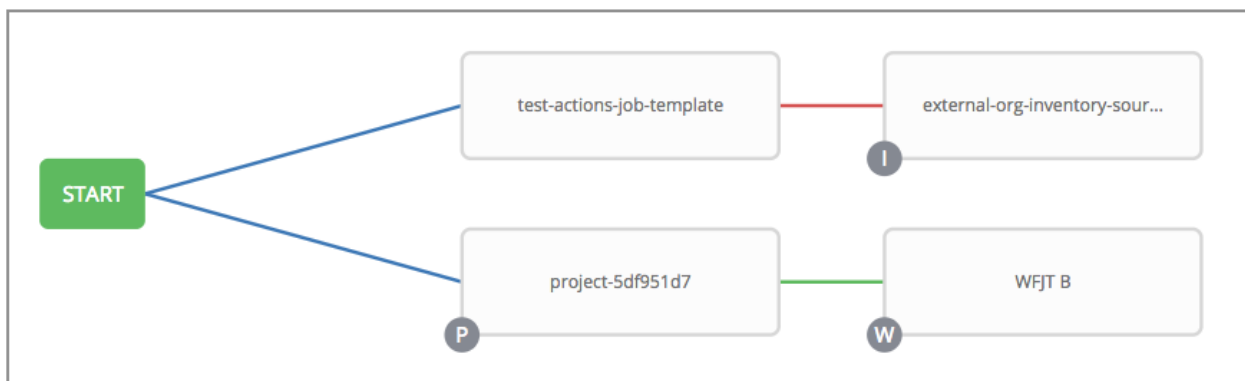
```
/api/v2/job_templates/?job_slice_count__gt=1
```

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.

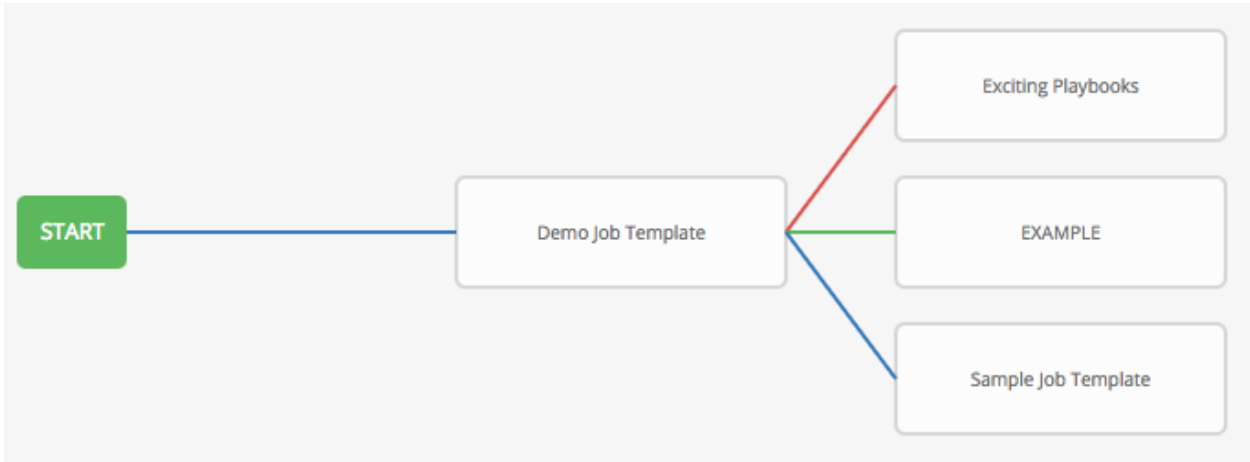
18.1 Workflow scenarios and considerations

Consider the following scenarios for building workflows:

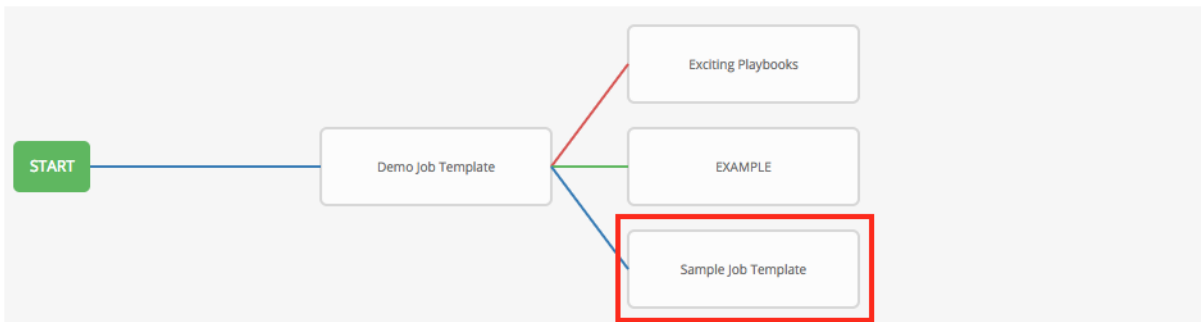
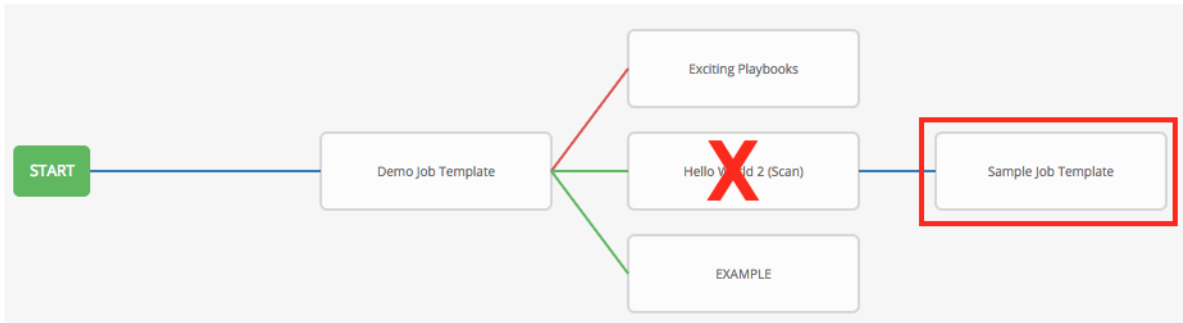
- A root node is set to ALWAYS by default and it not editable.



- 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:

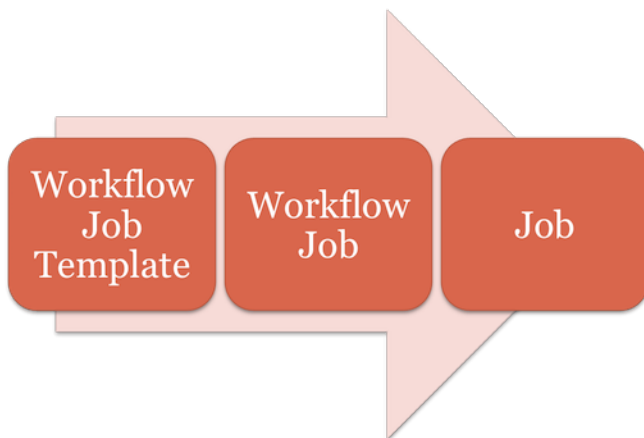


- You could have a convergent workflow, where multiple jobs converge into one. In this scenario, any of the jobs or all of them must complete before the next one runs, as shown in the example below:



In the example provided, Tower 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.

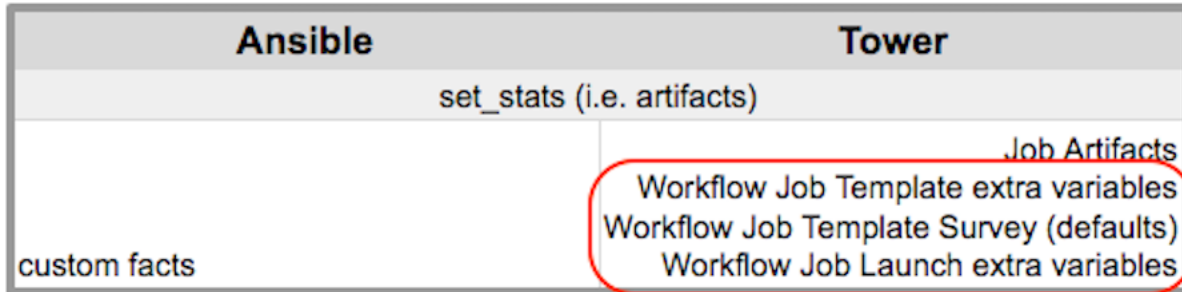


- It is possible to launch several workflows simultaneously, and set a schedule for when to launch them. You can set notifications on workflows, such as when a job completes, similar to that of job templates.
- You can build a recursive workflow, but if Tower detects an error, it will stop at the time the nested workflow attempts to run.
- Artifacts gathered in jobs in the sub-workflow will not 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.

18.2 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 Ansible Tower Variable Precedence Hierarchy in the *Extra Variables* section of the Job Templates chapter of this guide. The three additional variables include:



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 Tower Workflows"
      set_stats:
        data:
          integration_results_url: "{{ (result.stdout|from_json).link }}"
    
```

- **use_set_stats.yml**: second playbook in the workflow

```

---
- hosts: localhost
  tasks:
    - name: "Get test results from the web"
      uri:
        url: "{{ integration_results_url }}"
        return_content: true
      register: results
    
```

(continues on next page)

(continued from previous page)

```
- 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.

18.3 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.

18.4 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 [Ansible Tower Administration Guide](#).




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:

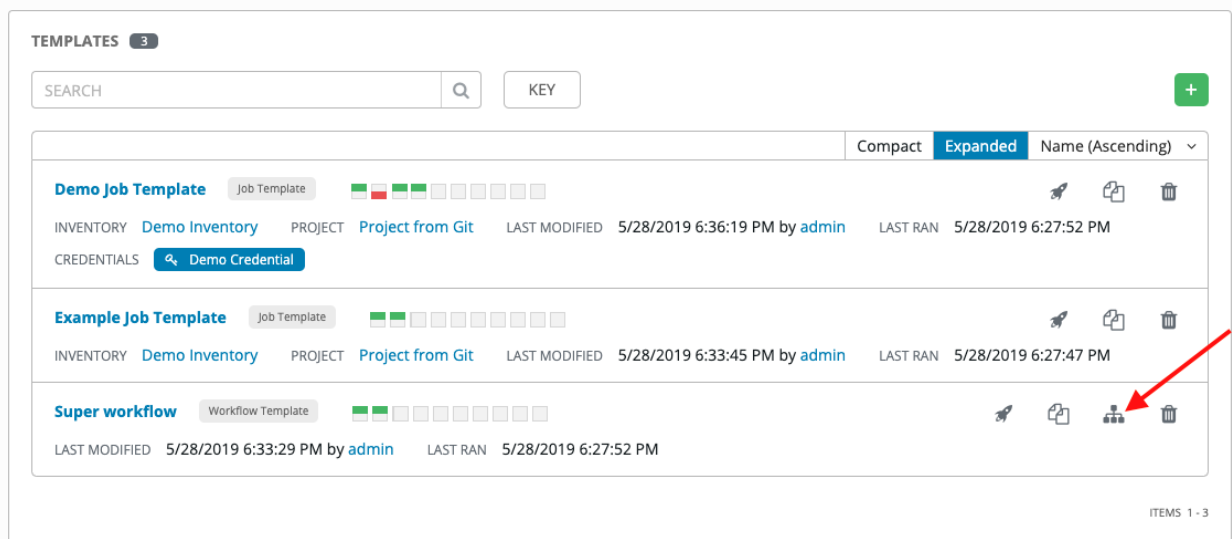
- job templates
- workflow templates
- project syncs
- inventory source syncs



The () 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

(), copy (), and remove () a job template. Before deleting a job template, be sure it is not used in a workflow job template.

Only workflow templates have the Workflow Visualizer icon () as a shortcut for accessing the workflow editor.



The screenshot shows a web interface titled 'TEMPLATES' with a sub-header '3'. It features a search bar and a 'KEY' button. The main content area displays a list of templates in an 'Expanded' view, sorted by 'Name (Ascending)'. The list includes three entries:

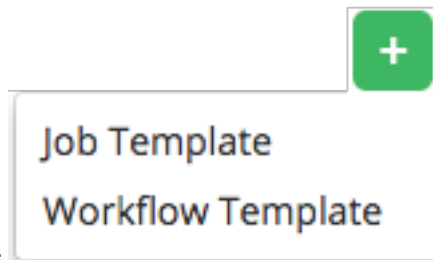
Template Name	Type	Status	Actions
Demo Job Template	Job Template	Progress bars (1 red, 3 green)	Launch, Copy, Remove
Example Job Template	Job Template	Progress bars (2 green)	Launch, Copy, Remove
Super workflow	Workflow Template	Progress bars (2 green)	Launch, Copy, Workflow Visualizer, Remove

Each entry shows details for 'INVENTORY' (Demo Inventory), 'PROJECT' (Project from Git), 'LAST MODIFIED' (5/28/2019 6:36:19 PM by admin), and 'LAST RAN' (5/28/2019 6:27:52 PM). The 'Super workflow' entry also shows 'CREDENTIALS' (Demo Credential). A red arrow points to the Workflow Visualizer icon (hierarchy symbol) for the 'Super workflow' entry. The bottom right corner indicates 'ITEMS 1 - 3'.

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.

19.1 Create a Workflow Template

To create a new workflow job template:



1. Click the **Workflow Template** button then select **Workflow Template** from the menu list.

NEW WORKFLOW JOB TEMPLATE

DETAILS | PERMISSIONS | COMPLETED JOBS | SCHEDULES | ADD SURVEY | WORKFLOW VISUALIZER

* NAME DESCRIPTION ORGANIZATION

INVENTORY PROMPT ON LAUNCH LIMIT PROMPT ON LAUNCH SCM BRANCH PROMPT ON LAUNCH

LABELS OPTIONS
 ENABLE CONCURRENT JOBS
 ENABLE WEBHOOK

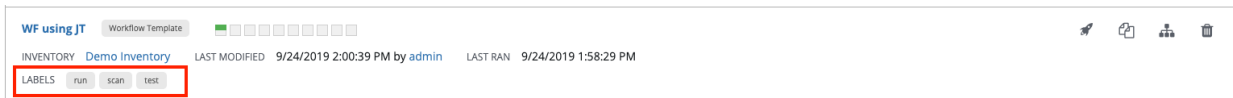
EXTRA VARIABLES PROMPT ON LAUNCH

1 ---

2. Enter the appropriate details into the following fields:
 - **Name:** Enter a name for the workflow template.
 - **Description:** Enter an arbitrary description as appropriate (optional).
 - **Organization:** Optionally enter or search for an organization to associate the workflow.
 - **Inventory:** Optionally enter or search for an inventory to be used with this workflow template from the inventories available to the currently logged in Tower user.
 - **Prompt on Launch:** If selected, you can provide an inventory when this workflow template is launched, or when this workflow template is used within another workflow template.
 - **Limit:** Optionally specify a limit for subset of servers that your workflow is going to run. This value is 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".
 - **Prompt on Launch:** If selected, even if a default value is supplied, you will be prompted upon launch to choose a limit.

- **SCM Branch:** Optionally specify the branch to override all job template nodes that prompt for a branch.
 - **Prompt on Launch:** If selected, even if a default value is supplied, you will be prompted upon launch to choose an SCM branch.
- **Labels:** Supply optional labels that describe this workflow template, such as “dev” or “test”. Labels can be used to group and filter workflow templates and completed jobs in the Tower 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 Templates overview.
 - Click on the “x” beside a label to remove it. When a label is removed, and is no longer associated with a Workflow or Workflow Template, the label is permanently deleted from the list of Organization labels.
 - Jobs inherit labels from the Workflow Template at the time of launch. If a label is deleted from a Workflow Template, it is also deleted from the Job.

LABELS ?



- **Options:**
- Check **Enable Concurrent Jobs** to allow simultaneous runs of this workflow. Refer to *Ansible Tower Capacity Determination and Job Impact* for additional information.
- Check **Enable Webhooks** to turn 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 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 Tower. This must be configured in the settings on the webhook service in order for Tower to accept webhooks from this service.

For additional information on setting up webhooks, see *Working with Webhooks*.

- **Extra 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
```

For more information about extra variables, refer to [Extra Variables](#).

- **Prompt on Launch:** If selected, even if a default value is supplied, you will be prompted upon launch to choose command line variables.

Note: If you want to be able to specify `extra_vars` on a schedule, you must select **Prompt on Launch** for **EXTRA VARIABLES** on the workflow template, or enable a survey on the workflow template, then those answered survey questions become `extra_vars`.

3. 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.

The screenshot shows the configuration page for a workflow template named "WF using JT". The page has several tabs: DETAILS (selected), PERMISSIONS, NOTIFICATIONS, COMPLETED JOBS, SCHEDULES, ADD SURVEY, and WORKFLOW VISUALIZER. The configuration fields include:

- NAME:** WF using JT
- DESCRIPTION:** (empty)
- ORGANIZATION:** (empty)
- INVENTORY:** (empty)
- PROMPT ON LAUNCH:**
- LABELS:** (empty)
- OPTIONS:** ENABLE CONCURRENT JOBS
- EXTRA VARIABLES:** A text area containing "1 ---" with tabs for YAML and JSON. A **PROMPT ON LAUNCH** checkbox is also present.

At the bottom right, there are three buttons: LAUNCH (disabled), CANCEL, and SAVE.

You can verify the template is saved when the newly created workflow template appears on the list of templates at the bottom of the screen.

The screenshot shows the "TEMPLATES" list view in Ansible Tower. It includes a search bar, a "KEY" button, and a table of templates. The table has columns for "Compact", "Expanded", and "Name (Ascending)".

Template Name	Type	Actions
Demo Job Template	Job Template	Refresh, Copy, Delete
WF using JT	Workflow Template	Refresh, Copy, Add, Delete

At the bottom right, it says "ITEMS 1-2".

Note: If an inventory was specified on the workflow template, the inventory displays in the Templates list view.

WF using JT Workflow Template ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■


INVENTORY Demo Inventory LAST MODIFIED 9/24/2019 12:54:52 PM by admin LAST RAN 9/24/2019 12:58:30 PM

19.2 Work with Permissions

Clicking on **Permissions** allows you to review, grant, edit, and remove associated permissions for users as well as team members.

USER	ROLE	TEAM ROLES
admin	SYSTEM ADMINISTRATOR	
gdoge	ADMIN	
jdoge	EXECUTE	



Click the  button to create new permissions for this workflow template.

In this example, two users and one team have been selected and each have been granted permissions for this Workflow Template.

NEW WORKFLOW JOB TEMPLATE | ADD USERS / TEAMS

1 Please select Users / Teams from the lists below.

USERS TEAMS

SEARCH KEY

NAME	ORGANIZATION
<input checked="" type="checkbox"/> Production Operations	Honey Dog, Inc.

ITEMS 1 - 1

2 Please assign roles to the selected users/teams

KEY

Production Operatio... TEAM SELECT ROLES

- Admin
- Execute
- Read

Note that you do not have to choose between teams or users, and that you can assign permissions to both at the same time.

19.3 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.

NAME	TYPE	APPROVAL	START	SUCCESS	FAILURE
Email notification for job starts	Email	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Slack notification	Slack	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
SMS notification to Self	Pagerduty	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Web notification	Webhook	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

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 **NOTIFICATIONS** link from inside the gray box to create a new notification.

THIS LIST IS POPULATED BY NOTIFICATION TEMPLATES ADDED FROM THE **NOTIFICATIONS** SECTION

Refer to [Notification Types](#) for additional details on configuring various notification types.

19.4 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.

TEMPLATES / New Workflow Job Template / COMPLETED JOBS

New Workflow Job Template

DETAILS PERMISSIONS NOTIFICATIONS **COMPLETED JOBS** SCHEDULES

SEARCH [] KEY []

Compact Expanded Finish Time (Descending)

- 111 - New Workflow Job Template Workflow job
 - STARTED 5/10/2019 4:23:24 PM FINISHED 5/10/2019 4:23:24 PM LAUNCHED BY admin
 - LABELS run
- 100 - New Workflow Job Template Workflow job
 - STARTED 5/2/2019 3:47:57 PM FINISHED 5/2/2019 3:47:57 PM LAUNCHED BY admin
 - LABELS run

ITEMS 1 - 2

Note: If a workflow-level inventory was specified at run-time, the inventory name displays in the workflow job in the jobs list:

Compact Expanded Finish Time (Descending)

141 - WF using JT Workflow job

STARTED 5/13/2019 10:50:01 PM FINISHED 5/13/2019 10:50:26 PM LAUNCHED BY admin INVENTORY Demo Inventory

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 the **141 - WF using JT** workflow job.

JOBS / 6 - WF using JT

DETAILS

STATUS Successful

STARTED 9/24/2019 1:58:21 PM

FINISHED 9/24/2019 1:58:29 PM

INVENTORY Demo Inventory

TEMPLATE WF using JT

LAUNCHED BY admin

EXTRA VARIABLES [YAML] [JSON] EXPAND

1 ---

WF using JT TOTAL NODES 3 ELAPSED 00:00:07

Project from Git

Demo Project

Demo Job Template

If a workflow template is used in another workflow, the jobs details indicate a parent workflow.

JOB / 748 - WFJT A

DETAILS

STATUS ● Successful

STARTED 11/5/2018 9:13:07 PM

FINISHED 11/5/2018 9:14:26 PM

TEMPLATE WFJT A

PARENT Overall

WORKFLOW

EXTRA VARIABLES YAML JSON EXPAND

```
1 WFJT: A
2 num_messages: '1'
3
```

LABELS ▼

WFJT

WFJT A TOTAL NODES 3 ELAPSED 00:01:19

```

graph LR
    Start(( )) --> P1[Project from Git]
    P1 --> JT[Demo Job Template]
    P1 --> DP[Demo Project]
    JT --> W1[WF in WF]
    
```

In the above example, click the parent workflow template, **Overall**, to view its Job Details page and the graphical details of the nodes and statuses of each as they were launched.

JOB / 671 - Overall

DETAILS

STATUS ● Successful

STARTED 11/5/2018 9:10:14 PM

FINISHED 11/5/2018 9:14:46 PM

TEMPLATE Overall

LAUNCHED BY admin

EXTRA VARIABLES YAML JSON EXPAND

```
1 WFJT: Overall
2 num_messages: '1'
3
```

LABELS ▼

WFJT

Overall TOTAL NODES 8 ELAPSED 00:04:32

```

graph LR
    Start(( )) --> C1[chaty_tasks JT]
    Start --> C2[chaty_tasks SJT]
    Start --> W1[WFJT A]
    Start --> W2[WFJT B]
    C1 --> W3[WFJT C]
    C2 --> W4[WFJT D]
    W1 --> W5[WFJT A]
    W2 --> W5
    W5 --> W6[WFJT A]
    
```

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.

19.5 Work with Schedules

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

The screenshot shows the 'Schedules' tab for a 'Super workflow'. It features a search bar and a table with columns for NAME, FIRST RUN, NEXT RUN, FINAL RUN, and ACTIONS. Two schedules are listed: 'Monthly monitoring' and 'Repeating Everyday', both with their respective run dates and times.

NAME	FIRST RUN	NEXT RUN	FINAL RUN	ACTIONS
Monthly monitoring	12/1/2020 11:00:00 PM	12/1/2020 11:00:00 PM	2/1/2021 11:00:00 PM	[Edit] [Delete]
Repeating Everyday	11/19/2020 3:00:00 AM	11/19/2020 3:00:00 AM	11/24/2020 3:00:00 AM	[Edit] [Delete]

19.5.1 Schedule a Workflow 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 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.

19.6 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

button 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.

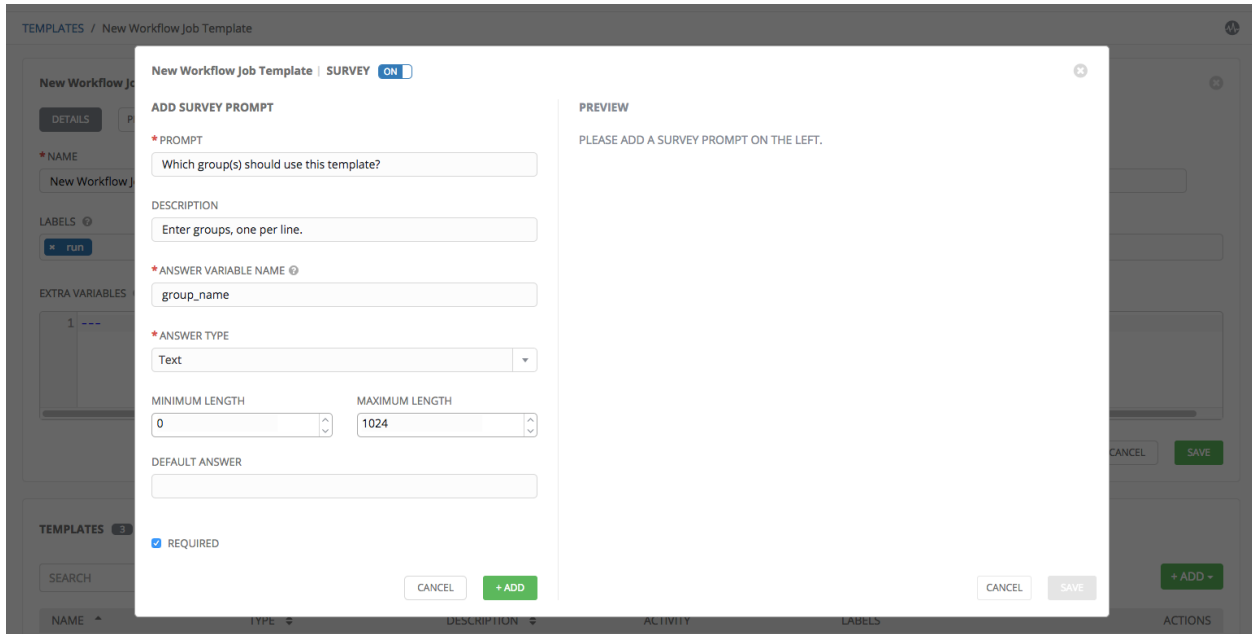
19.6.1 Create a Survey

To create a survey:

1. Click on the button to bring up the **Add Survey** window.


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  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.

19.6.2 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 ("").



19.7 Workflow Visualizer

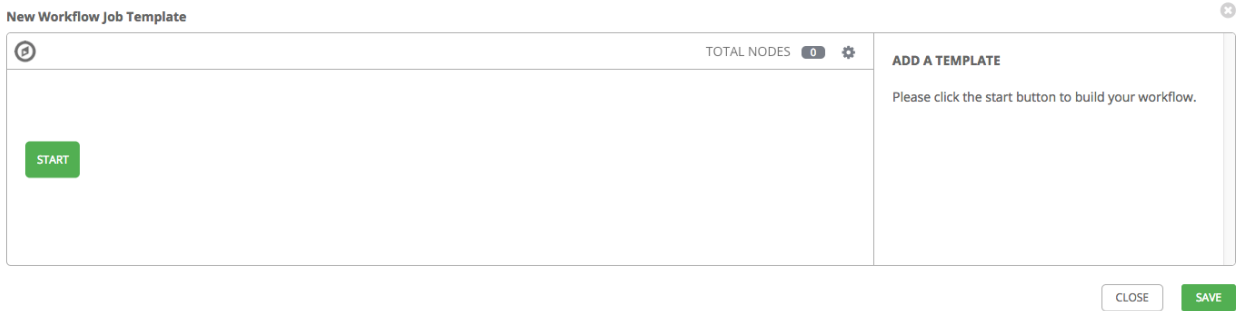
The Workflow Visualizer provides a graphical way of linking together job templates, workflow templates, project syncs, and inventory syncs to build a workflow template. Before building a workflow template, refer to the [Workflows](#) section for considerations associated with various scenarios on parent, child, and sibling nodes.


19.7.1 Build a Workflow

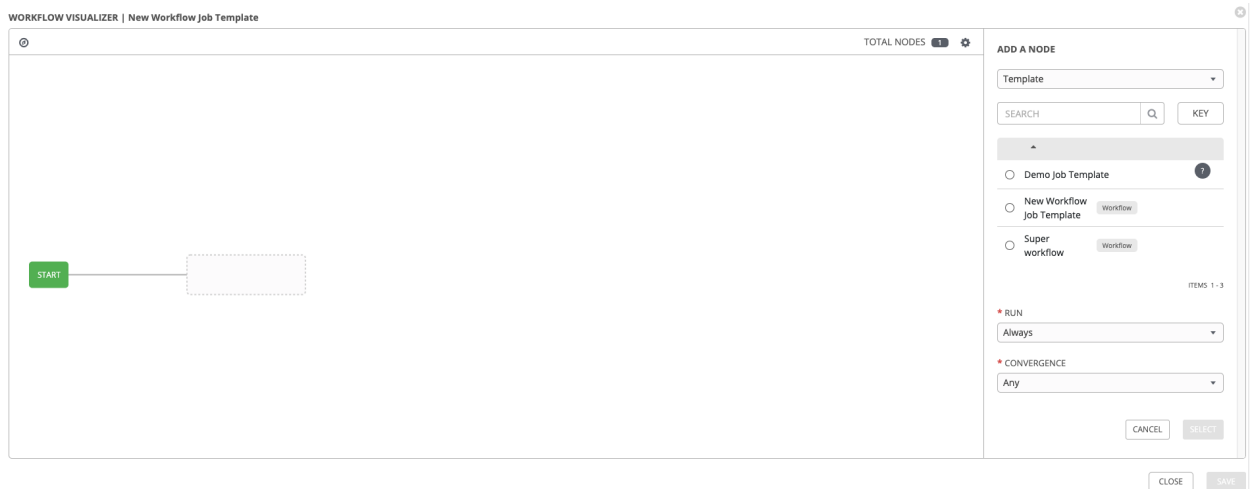
You can set up any combination of two or more of the following node types to build a workflow: Template (Job Template or Workflow Job Template), Project Sync, Inventory Sync, or Approval. Each node is represented by a rectangle while the relationships and their associated edge types are represented by a line (or link) that connects them.



1. In the details/edit view of a workflow template, click the  button or from the Templates list view, click the  icon to launch the Workflow Visualizer.

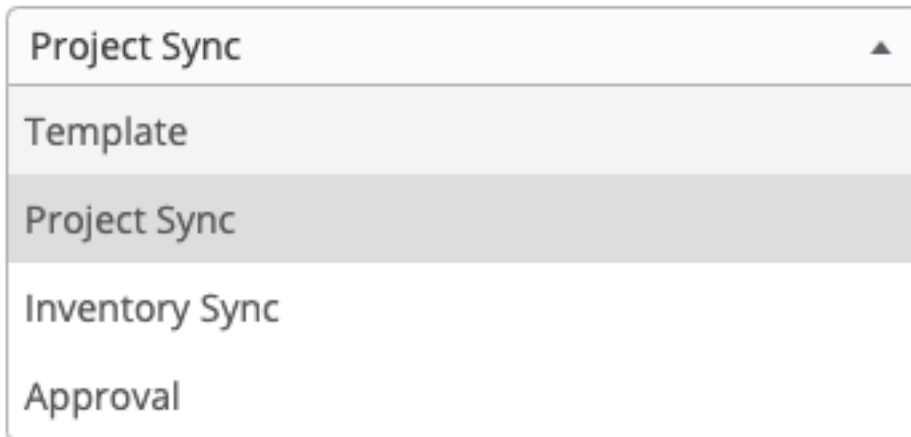


2. Click the  button to display a list of nodes to add to your workflow.



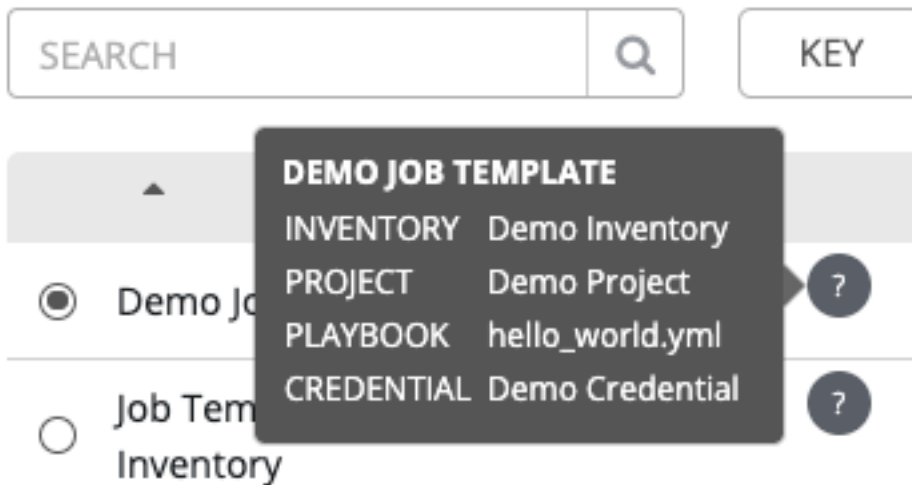
3. On the right pane, select the type of node you want to add from the drop-down menu:

ADD A NODE



If selecting an **Approval** node, see [Approval nodes](#) for further detail.

Selecting a node provides the available valid options associated with it. Access detail about the node by clicking the tooltip next to 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.

* RUN

Always

The inventory of this node will not be overridden by the parent workflow inventory.

CANCEL

SELECT

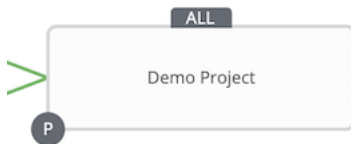
For subsequent nodes, you can select one of the following scenarios (edge type) to apply to each:

- **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:

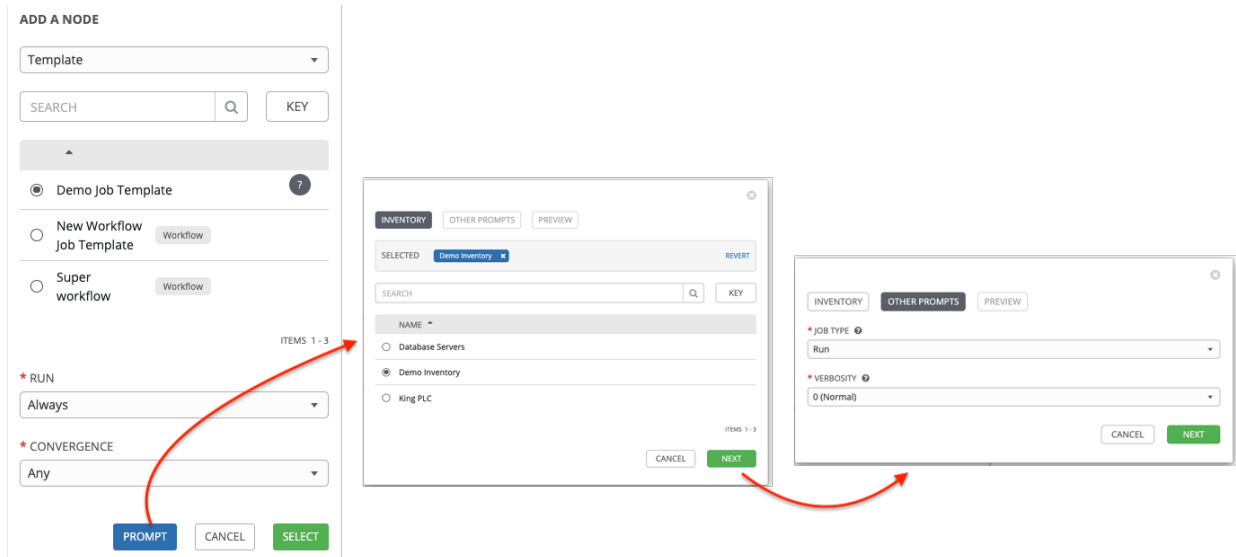
- **Any** is the default behavior, allowing *any* of the nodes to complete as specified, before triggering the next converging node. As long as the status of one parent meets one of those run conditions, an ANY child node will run. In other words, an ANY node requires **all** nodes to complete, but only 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**.

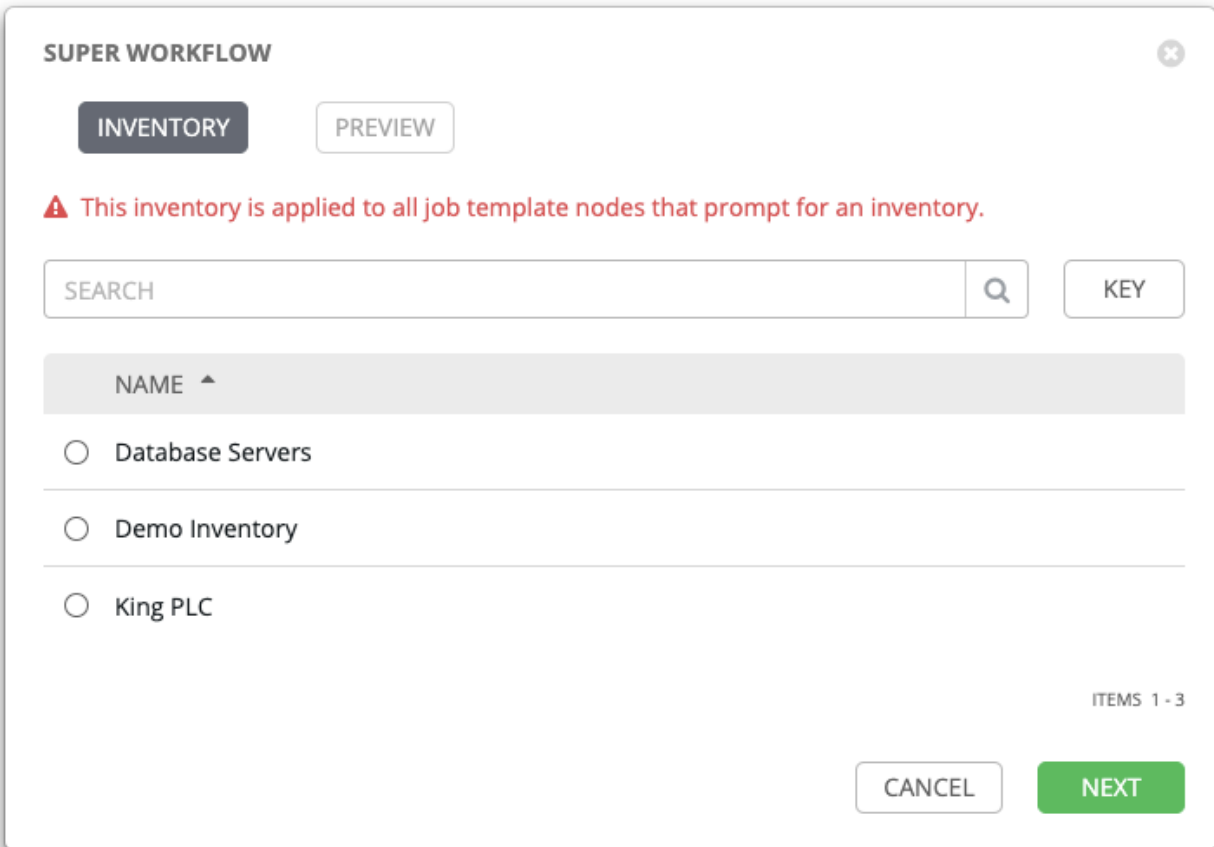


Note: If a node is a root node, or a node that does not have any nodes converging into it, setting the **Convergence** rule does not apply, as its behavior is dictated by the action that triggers it.

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.



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.



Note: For job templates with promptable fields that are required, but don't 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 job

template, but do not provide a default, or 2) when you create a survey question that is required but don't 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 job template prompts for credentials, Tower 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 ▼

PROMPT CANCEL **SELECT**

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, edit an existing link , or delete  the selected node.

WORKFLOW VISUALIZER | New Workflow Job Template


TOTAL NODES 0 



CLOSE
SAVE

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

* NAME

DESCRIPTION

TIMEOUT 

 min sec

CANCEL






SELECT


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.

 admin
 3




NOTIFICATIONS 3


Created (Ascending) 

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

ITEMS 1 - 3

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

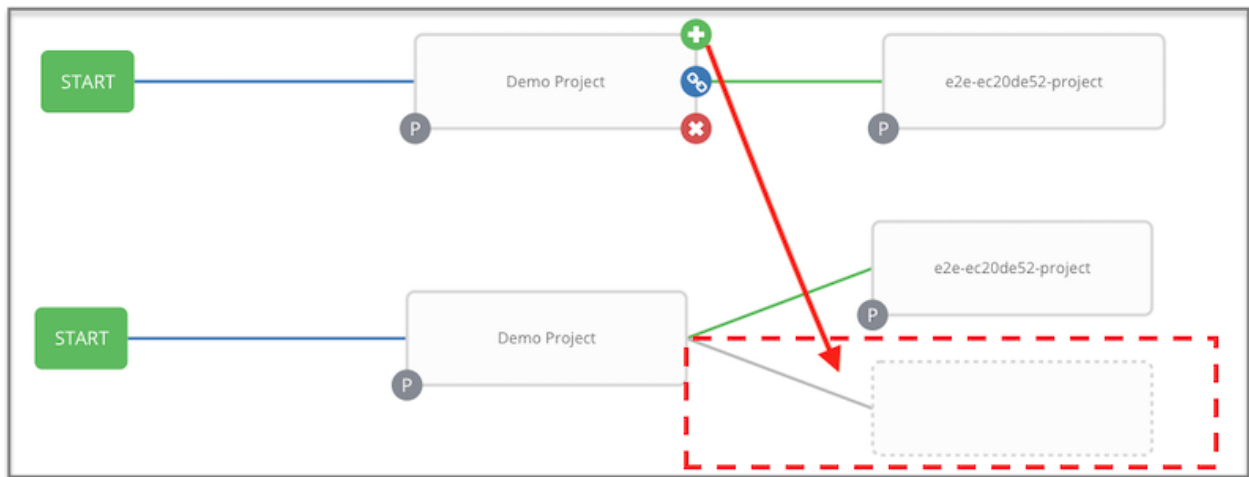
* Exception: A User with WF Admin permission at the organization level would not be able to grant approval.



** Exception: A User with Read on WF permission at the organization level would not be able to view WF approvals.

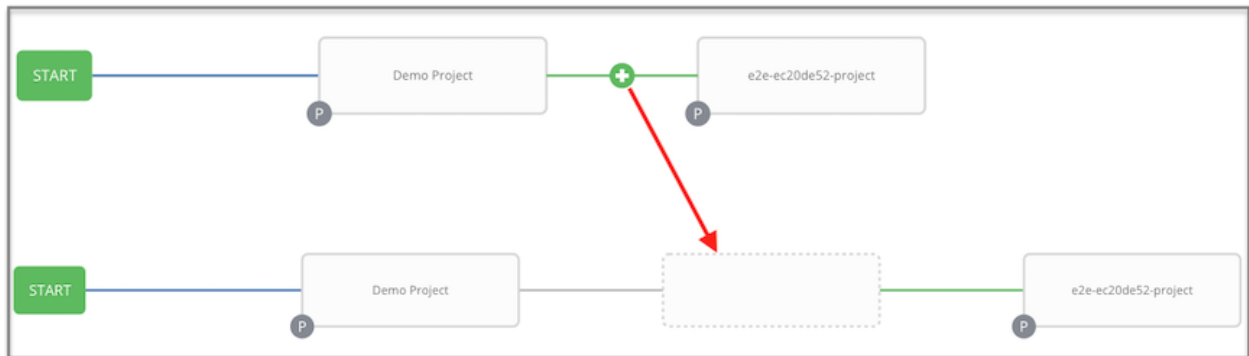
*** Exception: A User with Read on WF permission at the organization level would not be able to view approval jobs in the Activity Stream.


Node building scenarios

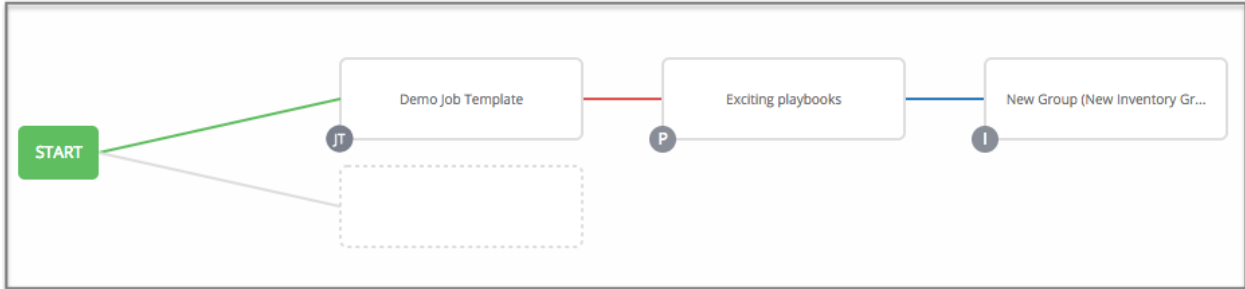
You can add a sibling node by clicking the  on the parent node:




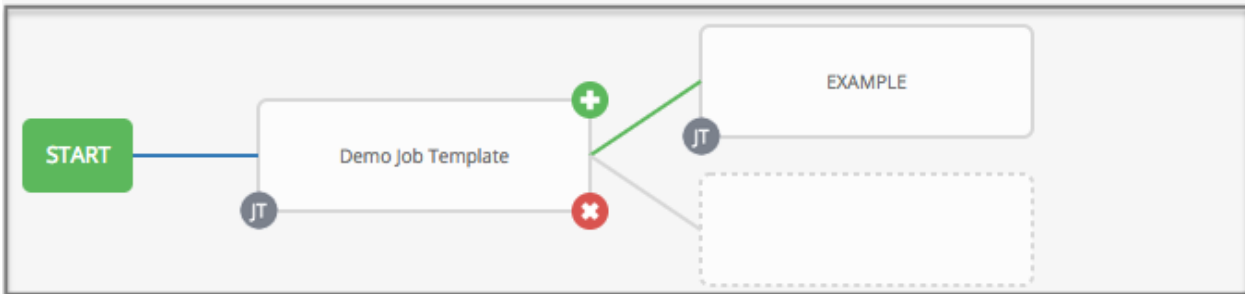
You can insert another node in between nodes by hovering over the line that connects the two until the  appears. Clicking on the  automatically inserts the node between the two nodes.



To add a root node to depict a split scenario, click the  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 ▼

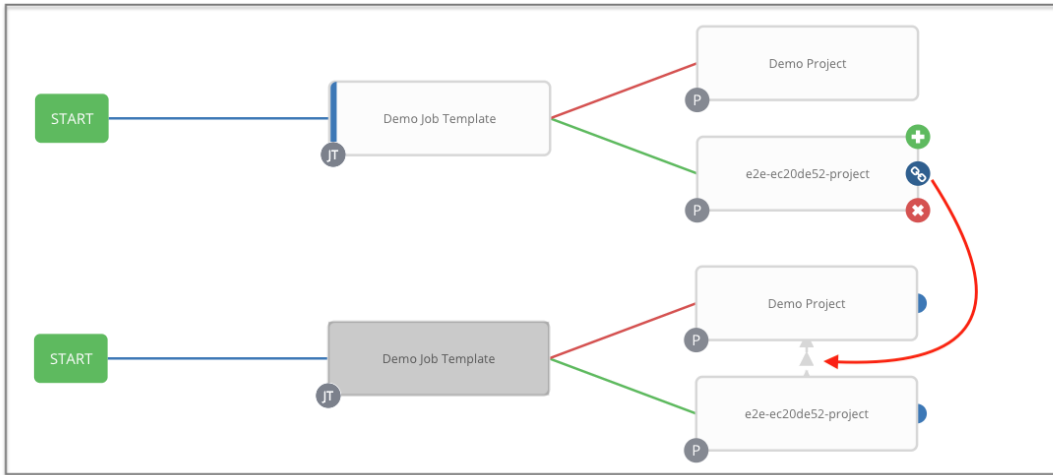
CANCEL

SAVE



- 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 ▾



UNLINK CANCEL SAVE

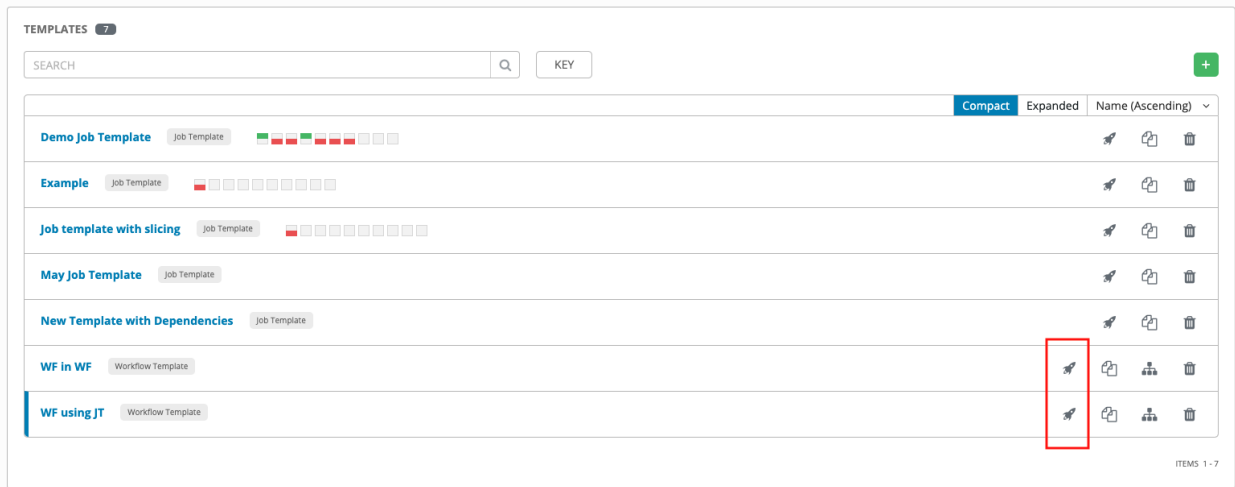
This button only appears in the right hand panel if the target or child node has more than one parent. All nodes must be linked to at least one other node at all times so you must create a new link before removing an old one.

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.

19.8 Launch a Workflow Template

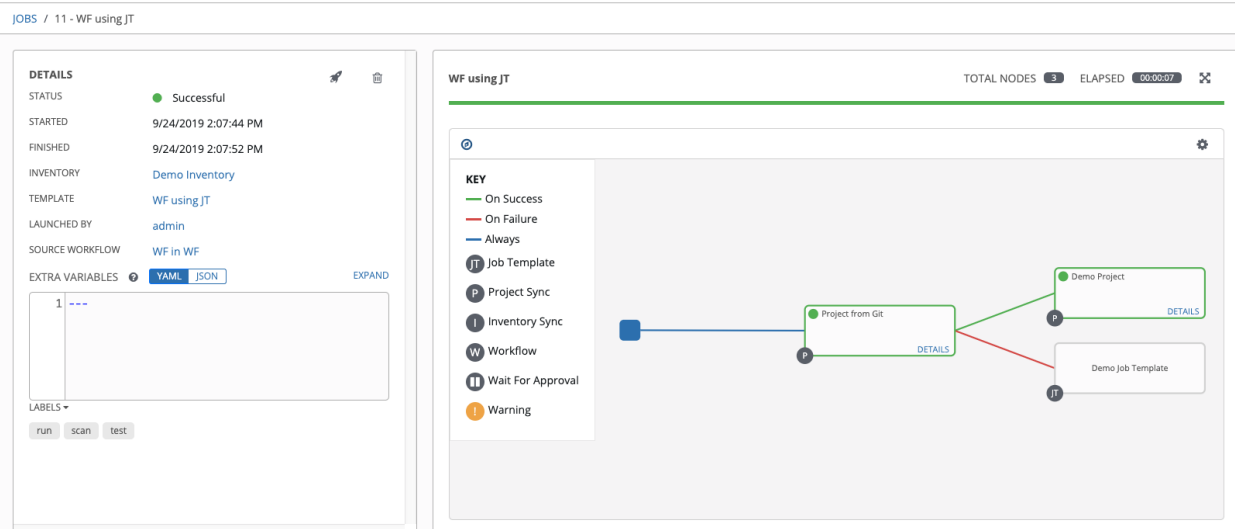
Launch a workflow template by any of the following ways:


- Access the workflow template list from the  navigational link or while in the Workflow Template Details view, scroll to the bottom to access the  button from the list of templates.



- While in the Job Template Details view of the job template you want to launch, click **Launch**.

Along with any extra variables set in the job template and survey, Tower automatically adds the same variables as those added for a job template upon launch. Additionally, Tower 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.

ACTIVITY STREAM

ACTIVITY STREAM | JOBS


SEARCH [] Q KEY [] Jobs []

TIME	INITIATED BY	EVENT	ACTIONS
9/25/2019 12:57:51 PM	admin	approved workflow_approval Cleanup Deleted Data Cleanup?	🔍
9/25/2019 12:57:48 PM	system	created job Demo Job Template	🔍
9/25/2019 12:57:48 PM	admin	approved workflow_approval New Workflow Job Template Approval node	🔍
9/25/2019 12:57:24 PM	system	timed out workflow_approval Remove VMWare Host Remove VMWare Host?	🔍
9/25/2019 12:45:46 PM	system	updated workflow_approval Cleanup Deleted Data Cleanup?	🔍
9/25/2019 12:45:46 PM	system	created workflow_approval Cleanup Deleted Data Cleanup?	🔍
9/25/2019 12:45:10 PM	system	updated workflow_approval Remove VMWare Host Remove VMWare Host?	🔍
9/25/2019 12:45:10 PM	system	created workflow_approval Remove VMWare Host Remove VMWare Host?	🔍
9/25/2019 12:33:44 PM	system	updated workflow_approval New Workflow Job Template Approval node	🔍
9/25/2019 12:33:44 PM	system	created workflow_approval New Workflow Job Template Approval node	🔍

19.9 Copy a Workflow Template

Ansible Tower 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** navigational link () or while in the Workflow Job Template Details view, scroll to the bottom to access it from a list of templates.

TEMPLATES 2

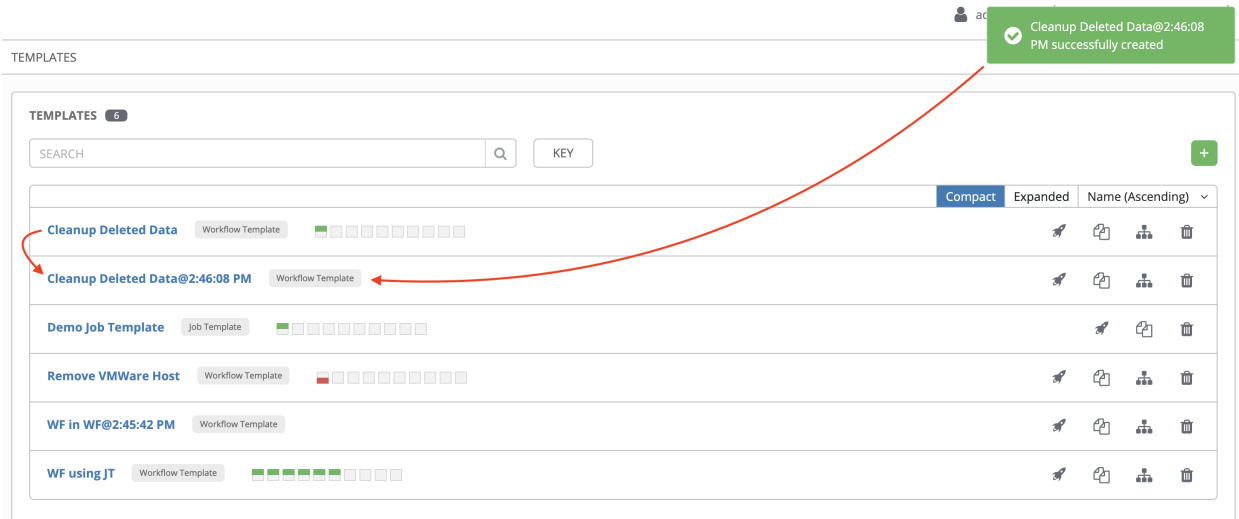
SEARCH [] Q KEY [] [+]

	Compact	Expanded	Name (Ascending)	
Demo Job Template Job Template				🔍 📄 🗑️
WF using JT Workflow Template				🔍 📄 🗑️

ITEMS 1 - 2

2. Click the  button.

A new template opens with the name of the template from which you copied and a timestamp.



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.

19.10 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`) within Tower. 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.

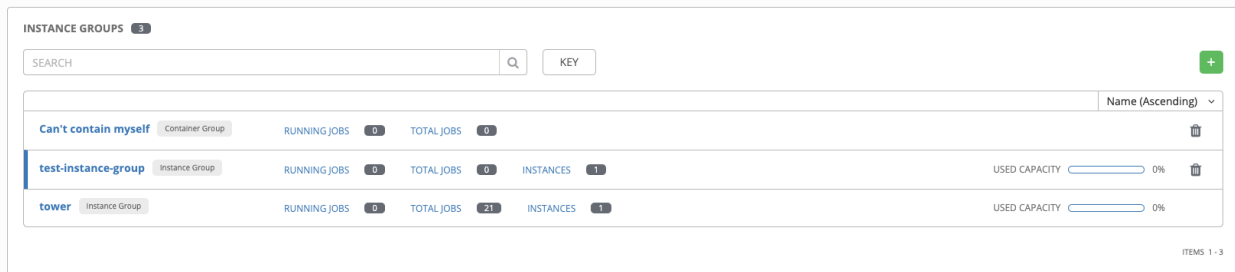
The following table notes the behavior (hierarchy) of variable precedence in Ansible Tower as it compares to variable precedence in Ansible.

Ansible Tower Variable Precedence Hierarchy (last listed wins)

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

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:



Name	Running Jobs	Total Jobs	Instances	Used Capacity
Can't contain myself (Container Group)	0	0		
test-instance-group (Instance Group)	0	0	1	0%
tower (Instance Group)	0	21	1	0%

For more information about the policy or rules associated with instance groups, see the [Instance Groups](#) section of the *Ansible Tower Administration Guide*.



You can set up isolated nodes in an instance group so that you can run your playbooks local to the nodes. For more detail, refer to [Isolated Instance Groups](#).

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 *Ansible Tower Feature Spotlight: Instance Groups and Isolated Nodes* [blog](#).

20.1 Create an instance group

To create a new instance group:

1. Click the  icon from the left navigation menu to open the Instance Groups configuration window.
2. Click the  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 *tower*.
- **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**.

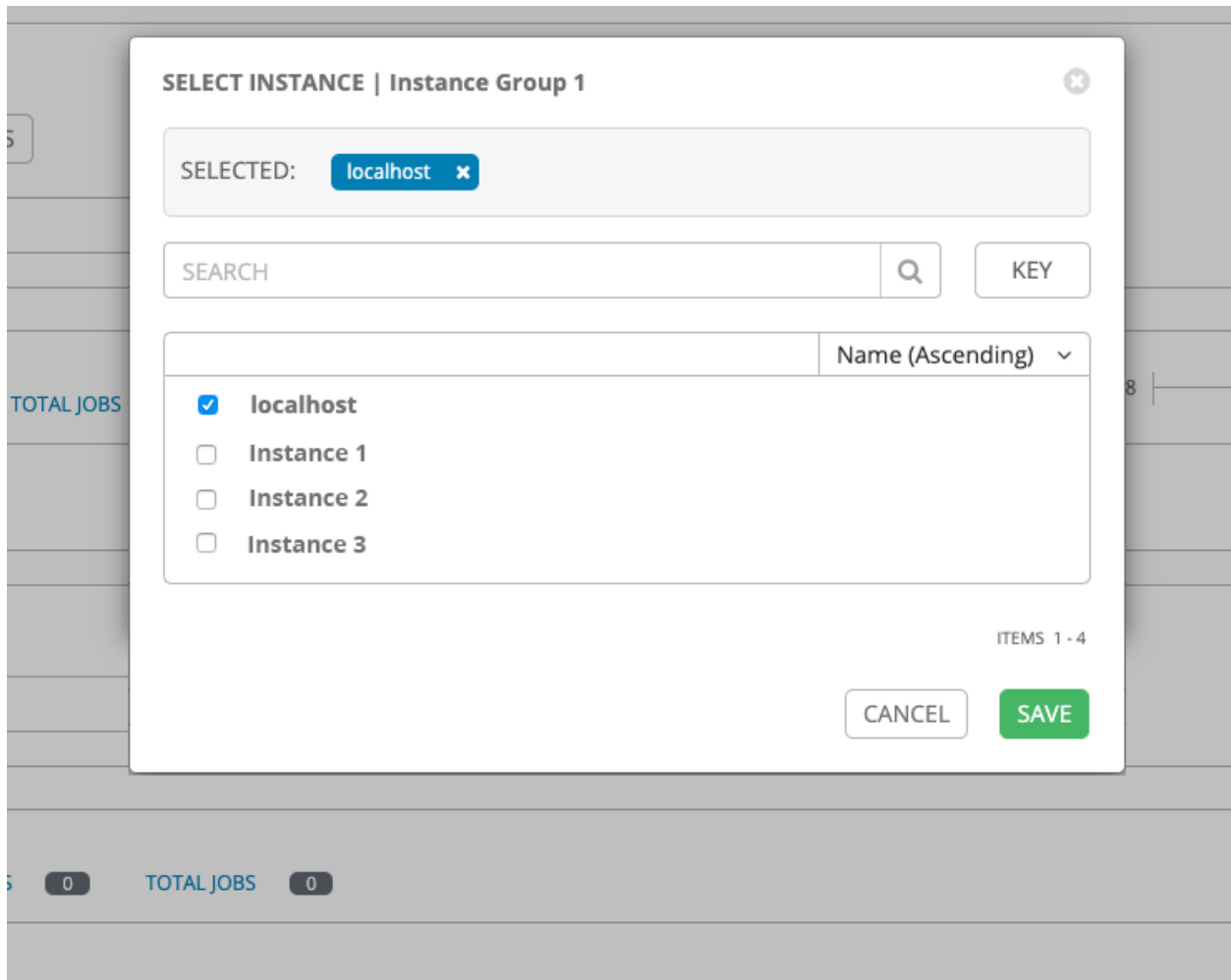
Once the instance group is successfully created, the **Details** tab of the newly created instance group remains, which allows you to review and edit your instance group information. This is the same menu that is opened if the Edit (✎) button is clicked from the **Instance Group** link. You can also edit **Instances** and review **Jobs** associated with this instance group.

Instance Group	RUNNING JOBS	TOTAL JOBS	INSTANCES	USED CAPACITY
Instance Group 1	0	0	1	0%
tower	0	43	1	0%

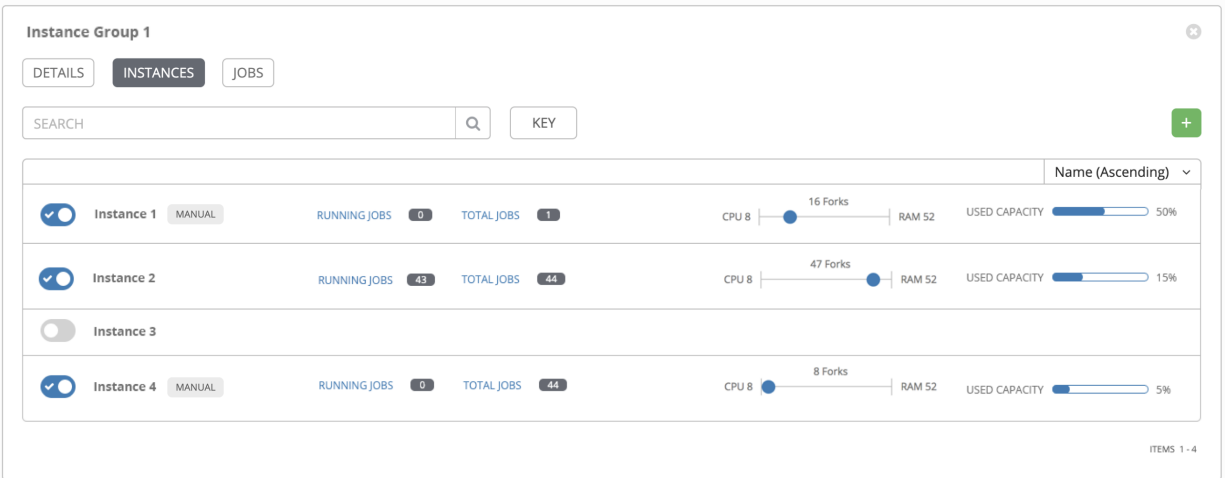
20.1.1 Associate instances to an instance group

To associate instances to an instance group:

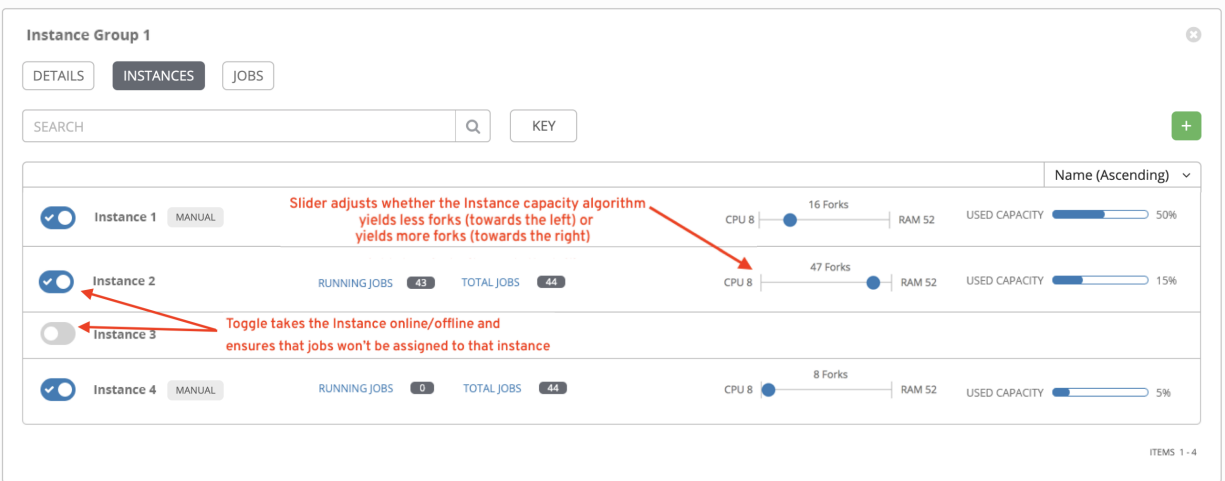
1. Click the **Instances** tab of the Instance Group window and click the  button.
2. Click the checkbox next to one or more available instances from the list to select the instance(s) you want to add to the instance group.



3. In the following example, the instances added to the instance group displays along with information about their capacity.



This view also allows you to edit some key attributes associated with the instances in your instance group:



20.1.2 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.

The screenshot shows the 'Jobs' page in Ansible Tower. At the top, there are tabs for 'DETAILS', 'INSTANCES', and 'JOBS', with 'JOBS' selected. Below the tabs is a search bar with a 'KEY' button. The main content area displays a list of jobs in an 'Expanded' view, sorted by 'Finish Time (Descending)'. Each job entry includes a status indicator (red for failed, green for successful), a job ID, a job name, a job type, and a list of details such as 'STARTED', 'FINISHED', 'LAUNCHED BY', 'JOB TEMPLATE', 'INVENTORY', and 'PROJECT'. Action icons for refresh and delete are present for each job.

Job ID	Job Name	Job Type	Status	Started	Finished	Launched By	Job Template	Inventory	Project
113	Demo Job Template	Playbook Run	Failed	5/13/2019 10:59:38 AM	5/13/2019 10:59:44 AM	admin	Demo Job Template	Demo Inventory	Demo Project
115	Demo Project	SCM Update	Success	5/13/2019 10:59:38 AM	5/13/2019 10:59:41 AM				Demo Project
114	Demo Project	SCM Update	Success	5/13/2019 10:59:30 AM	5/13/2019 10:59:38 AM				Demo Project
112	Cleanup Job Details	Management Job	Success	5/12/2019 1:01:36 PM	5/12/2019 1:01:40 PM				
110	Cleanup Activity Stream	Management Job	Success	5/7/2019 1:01:39 PM	5/7/2019 1:01:42 PM				
109	Project from Git	SCM Update	Success	5/7/2019 11:16:47 AM	5/7/2019 11:16:53 AM	admin			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 *Ansible Tower Administration Guide*.

CHAPTER TWENTYONE

JOBS

A job is an instance of Tower 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 ID, job name, and job type, 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.

The screenshot shows the 'JOBS' page with 82 jobs. The view is set to 'Compact'. The table lists jobs with their IDs, names, and types. Each row has a status indicator (green for success, red for failure) and a trash icon for deletion.



Job ID	Job Name	Job Type	Status	Action
110	Cleanup Activity Stream	Management job	Success	Trash
109	Project from Git	SCM Update	Success	Relaunch, Trash
108	Cleanup Job Details	Management job	Success	Trash
101	WF in WF	Workflow job	Failure	Relaunch, Trash
102	Job template with slicing	Playbook Run	Failure	Relaunch, Trash
100	New Workflow Job Template	Workflow job	Success	Relaunch, Trash
87	Demo Job Template	Playbook Run	Success	Relaunch, Trash
88	Demo Project	SCM Update	Success	Relaunch, Trash

The screenshot shows the 'JOBS' page with 84 jobs. The view is set to 'Expanded'. The table provides more details for each job, including start and finish times, launchers, and project names. A dropdown menu is open, showing sorting options.

Job ID	Job Name	Job Type	Status	Start Time	Finish Time	Launched By	Project	Action
112	Cleanup Job Details	Management job	Success	5/12/2019 1:01:36 PM	5/12/2019 1:01:40 PM			Trash
111	New Workflow Job Template	Workflow job	Success	5/10/2019 4:23:24 PM	5/10/2019 4:23:24 PM	admin		Relaunch, Trash
110	Cleanup Activity Stream	Management job	Success	5/7/2019 1:01:39 PM	5/7/2019 1:01:42 PM			Trash
109	Project from Git	SCM Update	Success	5/7/2019 11:16:47 AM	5/7/2019 11:16:53 AM	admin	Project from Git	Relaunch, Trash
108	Cleanup Job Details	Management job	Success	5/5/2019 1:01:45 PM	5/5/2019 1:01:48 PM			Trash

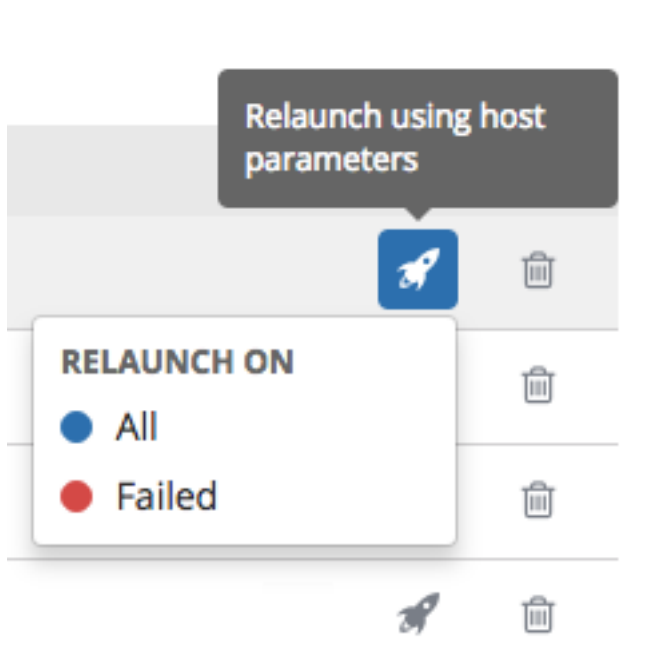
Sort By dropdown menu options:

- Name (Ascending)
- Name (Descending)
- Finish Time (Ascending)
- Start Time (Ascending)
- Start Time (Descending)
- Launched By (Ascending)
- Launched By (Descending)
- Project (Ascending)
- Project (Descending)
- Finish Time (Descending)

Actions you can take from this screen include viewing the details and standard output of a particular job, relaunching () jobs, or removing () jobs.

From the list view, you can re-launch the most recent job. You can re-run on all hosts in the specified inventory, even though some of them already had a successful run. This allows you to re-run the job without running the Playbook on them again. You can also re-run the job on all failed hosts. This will help lower the load on the Ansible Tower nodes as it does not need to process the successful hosts again.

The relaunch operation only applies to relaunched of playbook runs and does not apply to project/inventory updates, system jobs, workflow jobs, etc.



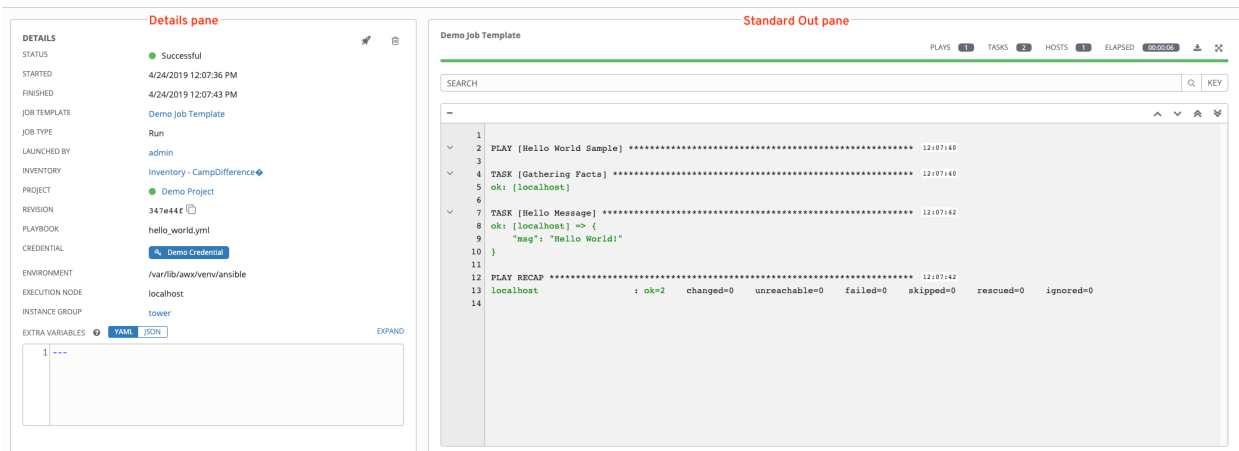
- Selecting **All** relaunches all the hosts.
- Selecting **Failed** relaunches all failed and unreachable hosts.

When it relaunches, you remain on the same page.

Use the Tower Search feature to look up jobs by various criteria. For details about using the Tower Search, refer to the [Search](#) chapter.

Clicking on any type of job takes you to the Job Details View for that job, which consists of two sections:

- The **Details** pane provides information and status about the job
- The **Standard Out** pane displays the job processes and output



21.1 Job Details - Inventory Sync

The screenshot displays the 'Job Details' for an 'Inventory Sync' job. The left pane shows the job status as 'Successful' and lists various configuration options such as 'LICENSE ERROR', 'HOST LIMIT ERROR', 'STARTED', 'FINISHED', 'LAUNCHED BY', 'INVENTORY', 'SOURCE', 'OVERWRITE', 'OVERWRITE VARS', 'VERBOSEITY', 'ENVIRONMENT', 'EXECUTION NODE', and 'INSTANCE GROUP'. The right pane shows a terminal log of the job execution, including the following output:



```

1 1.815 INFO Updating inventory 5: Custom Inventory Source
2 2.114 INFO Reading Ansible inventory source: /tmp/awx_6_583gpczp/tmp0z
25zmgx
3 2.116 INFO Using VIRTUAL_ENV: /var/lib/awx/venv/ansible
4 2.116 INFO Using PATH: /var/lib/awx/venv/ansible/bin:/var/lib/awx/venv/aw
x/bin:/opt/rh/rh-postgresql10/root/usr/bin:/var/lib/awx/venv/awx/bin:/var/lib/awx/
venv/awx/bin:/opt/rh/rh-postgresql10/root/usr/bin:/usr/local/sbin:/usr/local/bin:/
usr/sbin:/usr/bin
5 2.116 INFO Using PYTHONPATH: /var/lib/awx/venv/ansible/lib/python2.7/s
ite-packages:
6 2.621 ERROR ansible-inventory 2.9.7.post0
7 2.621 ERROR config file = /etc/ansible/ansible.cfg
8 2.621 ERROR configured module search path = [u'/var/lib/awx/.ansible/pl
ugins/modules', u'/usr/share/ansible/plugins/modules']
9 2.621 ERROR ansible python module location = /usr/lib/python2.7/site-p
ackages/ansible
10 2.622 ERROR executable location = /usr/bin/ansible-inventory
11 2.622 ERROR python version = 2.7.5 (default, Sep 26 2019, 13:23:47) [GC
C 4.8.5 20150623 (Red Hat 4.8.5-39)]
12 2.622 ERROR Using /etc/ansible/ansible.cfg as config file
13 2.622 ERROR host_list_declined_parsing /tmp/awx_6_583gpczp/tmp0z25zmgx_as

```

Note: Starting in Ansible Tower 3.7, 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.

21.1.1 Details

The **Details** pane shows the basic status of the job and its start time. The icons at the top right corner of the **Details** pane allow you to relaunch () or delete () the job.



The **Details** pane also provides details on the job execution:

- **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.
- **License Error:** Only shown for **Inventory Sync** jobs. If this is *True*, the hosts added by the inventory sync caused Tower to exceed the licensed number of managed hosts.
- **Host Limit Error:** Denotes the job's inventory belongs to an organization that has exceeded its limit of hosts as defined by the system administrator.
- **Started:** The timestamp of when the job was initiated by Tower.

- **Finished:** The timestamp of when the job was completed.
- **Launched By:** The name of the user who launched the job.
- **Inventory:** The name of the associated inventory group.
- **Source:** The type of cloud inventory.
- **Overwrite:** If *True*, any hosts and groups that were previously present on the external source but are now removed, are removed from the Tower inventory. Hosts and groups that were not managed by the inventory source are promoted to the next manually created group or if there is no manually created group to promote them into, they are left in the “all” default group for the inventory. If *False*, local child hosts and groups not found on the external source remain untouched by the inventory update process.
- **Overwrite Vars:** If *True*, all variables for child groups and hosts are removed and replaced by those found on the external source. If *False*, a merge was performed, combining local variables with those found on the external source.
- **Verbosity:** The level of output Ansible will produce for inventory source update jobs.
- **Environment:** The virtual environment used.
- **Execution node:** The node used to execute the job.
- **Instance Group:** The name of the instance group used with this job (tower is the default instance group).

By clicking on these items, where appropriate, you can view the corresponding job templates, projects, and other Tower objects.

21.1.2 Standard Out



The **Standard Out** pane shows the full results of running the Inventory Sync playbook. 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 Standard Out pane allow you to toggle the output as a main view () or to download the output ().

The `ANSIBLE_DISPLAY_ARGS_TO_STDOUT` is set to `False` by default for all playbook runs. This matches Ansible’s default behavior. This causes Tower to no longer 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](#).

21.2 Job Details - SCM

The screenshot displays the 'Job Details' page for an SCM job. On the left, a 'DETAILS' pane shows the job's status as 'Successful' and provides key information: started at 4/24/2019 12:07:36 PM, finished at 4/24/2019 12:07:43 PM, job type is 'Check', launched by 'admin', project is 'e2e-ae53906d-project', and instance group is 'tower'. On the right, the main execution log for 'e2e-ae53906d-project' is visible, showing task output for 'Write Repository Version' and a final 'PLAY RECAP' summary indicating 4 OK, 2 changed, 0 unreachable, and 0 failed tasks.

21.2.1 Details



The **Details** pane shows the basic status of the job and its start time. The icons at the top right corner of the **Details** pane allow you to relaunch () or delete () the job.

The **Details** pane provides details on the job execution:

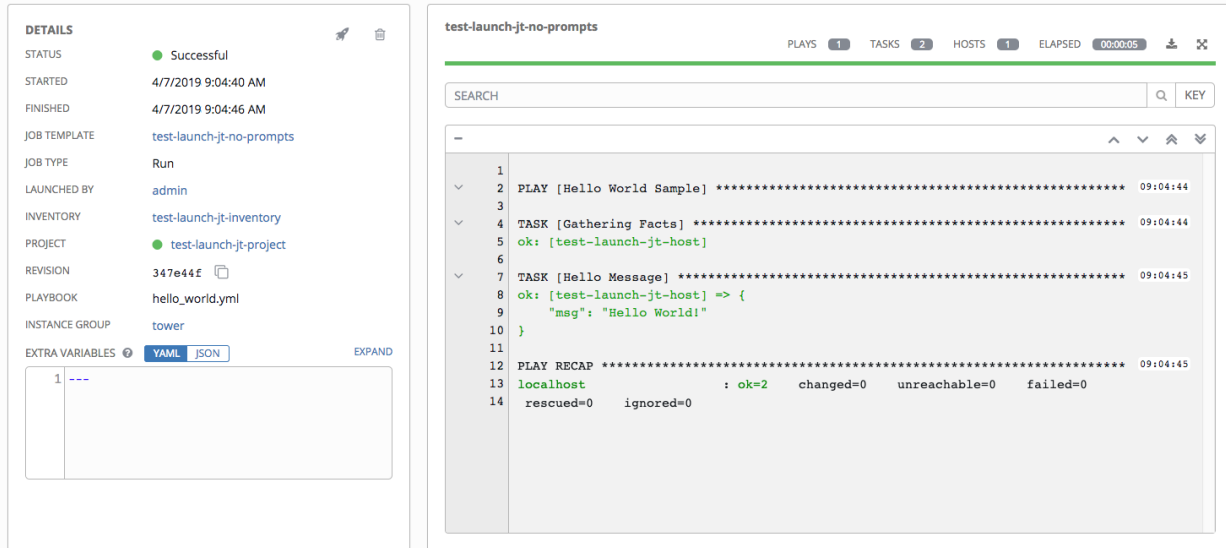
- **Name:** The name of the associated inventory group.
- **Status:** Can be any of the following:
 - **Pending** - The SCM job has been created, but not queued or started yet. Any job, not just SCM jobs, will stay in pending until it's actually ready to be run by the system. Reasons for SCM jobs not 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.
- **Started:** The timestamp of when the job was initiated by Tower.
- **Finished:** The timestamp of when the job was completed.
- **Elapsed:** The total time the job took.
- **Launch Type:** *Manual* or *Scheduled*.
- **Project:** The name of the project.

By clicking on these items, where appropriate, you can view the corresponding job templates, projects, and other Tower objects.

21.2.2 Standard Out

The **Standard Out** pane shows the full results of running the SCM Update. 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 Standard Out pane allow you to toggle the output as a main view () or to download the output ().



21.3 Job Details - Playbook Run



The screenshot displays the 'Job Details' view for a 'Playbook Run' job. On the left, a 'DETAILS' pane shows the job's status as 'Successful', started at 4/7/2019 9:04:40 AM, and finished at 4/7/2019 9:04:46 AM. It also lists the job template, type, launcher, inventory, project, revision, playbook, and instance group. On the right, the 'test-launch-jt-no-prompts' job output is shown, including a search bar and a list of tasks such as 'PLAY [Hello World Sample]', 'TASK [Gathering Facts]', and 'TASK [Hello Message]'. The output also includes a 'PLAY RECAP' section at the bottom.

The Job Details View for a **Playbook Run** job is also accessible after launching a job from the **Job Templates** page.

21.3.1 Details

The **Details** pane shows the basic status of the job and its start time. The icons at the top right corner of the **Details** pane allow you to relaunch () or delete () the job.



The **Details** pane provides details on the job execution:

- **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's 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.
- **Template:** The name of the job template from which this job was launched.
- **Started:** The timestamp of when the job was initiated by Tower.

- **Finished:** The timestamp of when the job was completed.
- **Elapsed:** The total time the job took.
- **Launch By:** The name of the user, job, or scheduled scan job which launched this job.
- **Inventory:** The inventory selected to run this job against.
- **Machine Credential:** The name of the credential used in this job.
- **Verbosity:** The level of verbosity set when creating the job template.
- **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 Tower objects.

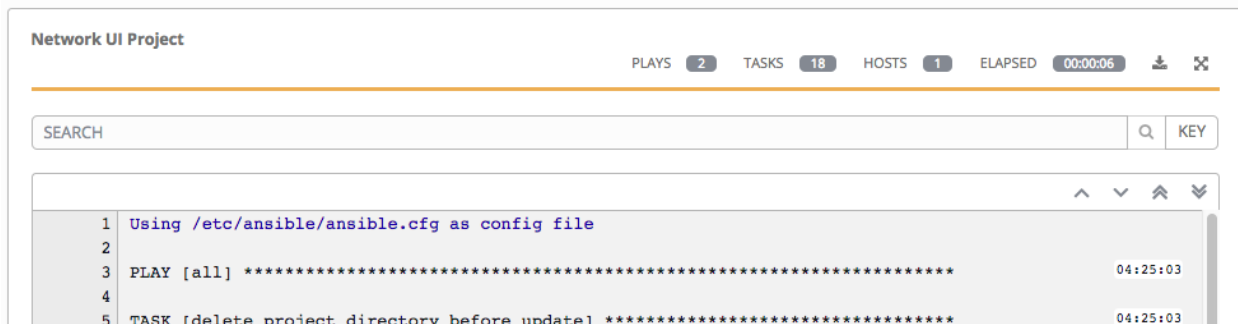
21.3.2 Standard Out Pane

The **Standard Out** pane shows the full results of running the Ansible playbook. This shows the same information you would see if you ran it through the Ansible command line, and can be useful for debugging. You can view the event summary, host status, and the host events. The icons at the top right corner of the Standard Out pane allow you to toggle the output as a main view () or to download the output ().

Events Summary

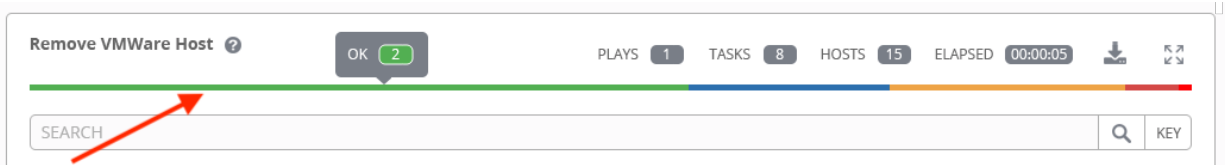
The events summary captures a tally of events that were run as part of this playbook:

- the number of plays
- the number of tasks
- the number of hosts
- the elapsed time to run the job template



Host Status Bar

The host status bar runs across the top of the **Standard Out** pane. Hover over a section of the host status bar and the number of hosts associated with that particular status displays.

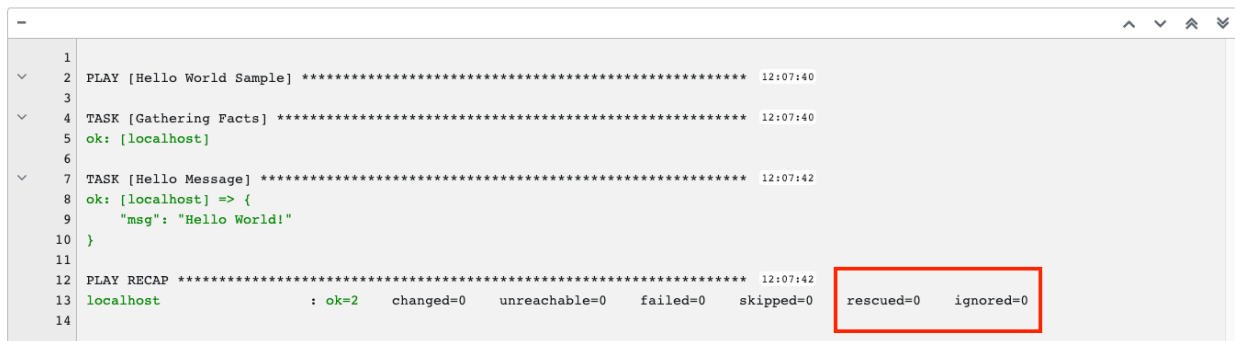


Search

Use the Tower 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:

- **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.
- **OK:** the playbook task returned “Ok”.
- **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 Standard Out pane, in a group of “stats” called the Host Summary fields.



The example below shows a search with only failed hosts.



For more details about using the Tower Search, refer to the [Search](#) chapter.

Standard output view

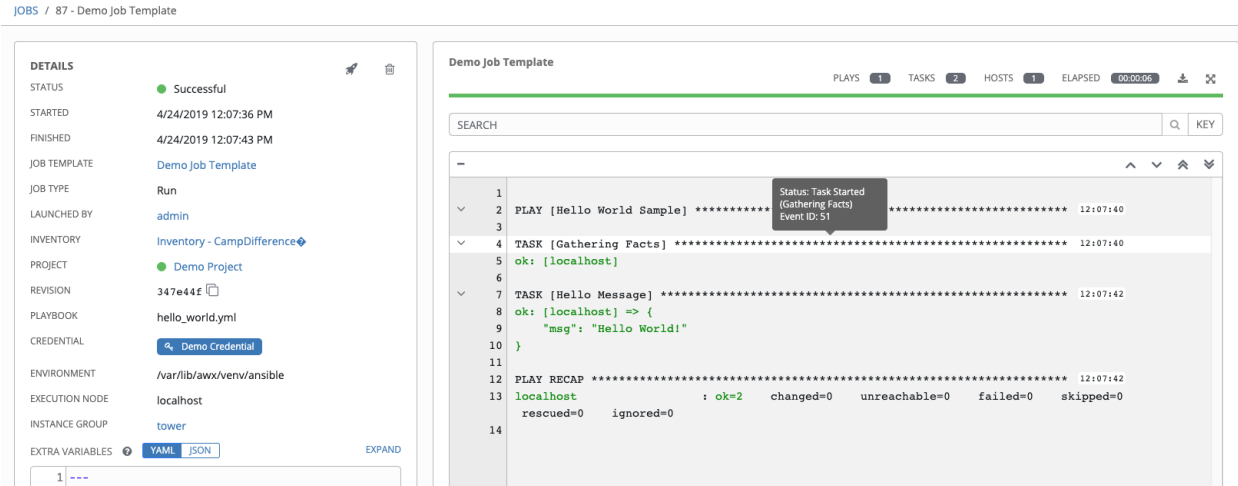
The standard output view displays all the events that occur on a particular job. By default, all rows are expanded so that all the details are displayed. Use the collapse-all button () to switch to a view that only contains the headers for plays and tasks. Click the () button to view all lines of the 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.



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.
- Hover over an event line in the **Standard Out** view, a tooltip displays above that line, giving the total hosts affected by this task and an option to view further details about the breakdown of their statuses.



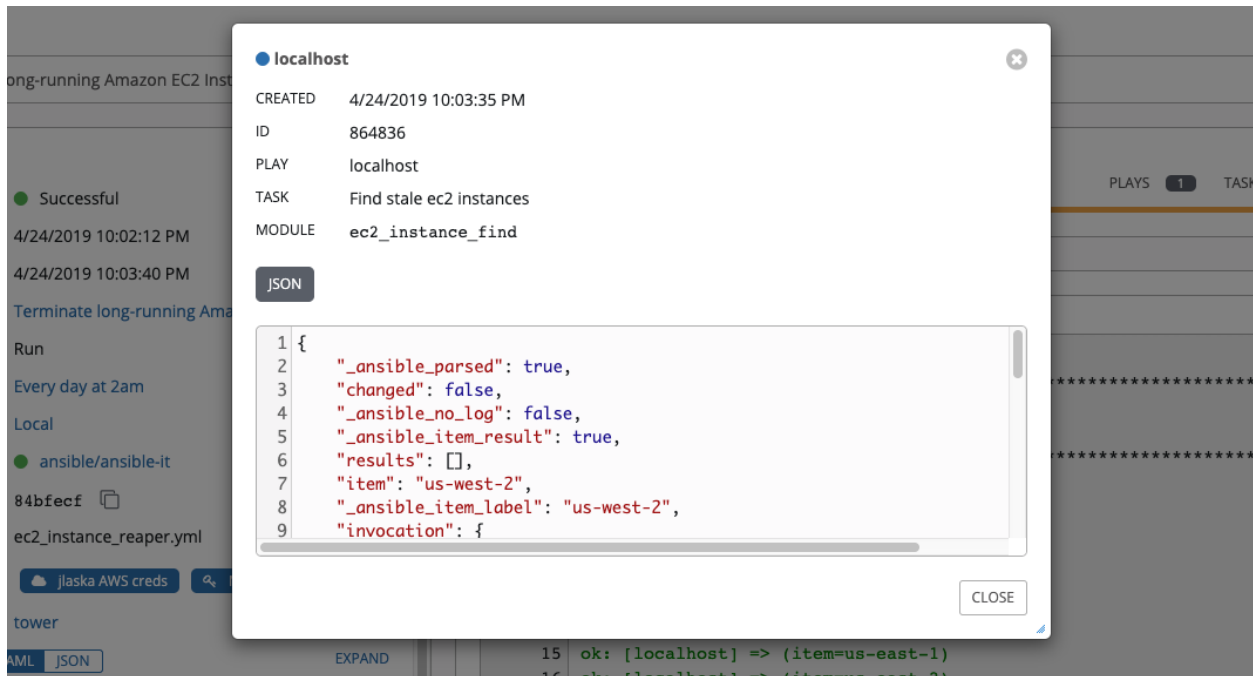
Click on a line of an event from the **Standard Out** pane and a **Host Events** dialog displays in a separate window. This window shows the host that was affected by that particular event.

Note: Upgrading to the latest versions of Tower 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.

Host Events

The **Host Events** dialog shows information about the host affected by the selected event and its associated play and task:

- the **Host**
- the **Status**
- a unique **ID**
- a **Created** time stamp
- the name of the **Play**
- the name of the **Task**
- if applicable, the Ansible **Module** for the task, and any *arguments* for that module
- the **Standard Out** of the task



To view the results in JSON format, click on the **JSON** tab.

21.4 Ansible Tower 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 *Ansible Tower Administration Guide*.

The Ansible Tower 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](#).

21.4.1 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 Tower 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.

Tower 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 Tower's 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 Tower 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 Tower 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 Tower `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 Tower settings value (or environment variable) `SYSTEM_TASK_FORKS_CPU` which defaults to 4.

21.4.2 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 Tower knows that you're running against fewer systems than that, then the actual concurrency value will be lower.

When a job is run, Tower 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 Tower

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 Tower user interface, select the **Instances** tab of the Instance Group.

Instance Group 1

DETAILS INSTANCES RUNNING JOBS

SEARCH [] [] [] []

Instance	Status	Running Jobs	CPU	Forks	RAM	Used Capacity
Instance 1	Enabled	100	8	16	52	50%
Instance 2	Enabled	150	8	47	52	15%
Instance 3	Disabled	-	-	-	-	-
Instance 4	Enabled	150	8	8	52	5%

ITEMS 1 - 4

21.5 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.

NEW PROJECT

DETAILS PERMISSIONS JOB TEMPLATES SCHEDULES

* NAME: Example
 DESCRIPTION: Ansible example playbook
 * ORGANIZATION: Honey Dog, Inc.
 ANSIBLE ENVIRONMENT: Select Ansible Environment
 * SCM TYPE: Git

SOURCE DETAILS

* SCM URL: https://github.com/ansible/tower-example
 SCM BRANCH/TAG/COMMIT: []
 SCM REFSPEC: []

SCM CREDENTIAL: []

SCM UPDATE OPTIONS

- CLEAN
- DELETE ON UPDATE
- UPDATE REVISION ON LAUNCH
- ALLOW BRANCH OVERRIDE

CANCEL SAVE

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`).

SCM UPDATE OPTIONS

- CLEAN
- DELETE ON UPDATE
- UPDATE REVISION ON LAUNCH
- ALLOW BRANCH OVERRIDE

Admins of job templates can further delegate that ability to users executing the job template (requiring only job template `execute_role`) by checking the **Prompt on Launch** box next to the SCM Branch field of the job template.

The screenshot shows the 'Branch Override' configuration page. It includes tabs for 'DETAILS', 'PERMISSIONS', 'NOTIFICATIONS', 'COMPLETED JOBS', 'SCHEDULES', and 'ADD SURVEY'. The 'DETAILS' tab is active. The form contains several fields:

- * NAME:** Branch Override
- DESCRIPTION:** (empty)
- * JOB TYPE:** Run (dropdown menu)
- * INVENTORY:** Demo Inventory (dropdown menu)
- * PROJECT:** Example (dropdown menu)
- SCM BRANCH:** branch (dropdown menu, highlighted with a red box)
- * PLAYBOOK:** free_waiter.yml (dropdown menu)
- CREDENTIAL:** (empty)
- FORKS:** 0 (dropdown menu)

 Checkboxes for 'PROMPT ON LAUNCH' are present next to the 'JOB TYPE', 'SCM BRANCH', and 'CREDENTIAL' fields. The 'PROMPT ON LAUNCH' checkbox for 'SCM BRANCH' is checked.

21.5.1 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, Tower discards modified files in its local copy of the repository through use of the `force` parameter in its respective Ansible modules pertaining to `git`, `Subversion`, and `Mercurial`.

The screenshot shows the 'SOURCE DETAILS' configuration page. It includes fields for:

- * SCM URL:** https://github.com/ansible/tower-example
- SCM BRANCH/TAG/COMMIT:** (empty)
- SCM REFSPEC:** (empty)
- SCM CREDENTIAL:** (empty)



 The 'SCM UPDATE OPTIONS' section is highlighted with a red box and contains the following checkboxes:


- CLEAN
- DELETE ON UPDATE
- UPDATE REVISION ON LAUNCH
- ALLOW BRANCH OVERRIDE

21.5.2 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 **Revision** field of the job and its respective project update.

JOBS / 6 - Branch Override

DETAILS  

STATUS	● Successful
STARTED	8/15/2019 8:48:27 AM
FINISHED	8/15/2019 8:48:58 AM
JOB TEMPLATE	Branch Override
JOB TYPE	Run
LAUNCHED BY	admin
INVENTORY	Demo Inventory
PROJECT	Example
REVISION	806a1d2 
PLAYBOOK	free_waiter.yml
ENVIRONMENT	/var/lib/awx/venv/ansible
EXECUTION NODE	localhost
INSTANCE GROUP	tower
EXTRA VARIABLES	? YAML JSON EXPAND

```
1 ---
```

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.

21.5.3 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 GitHub pull request reference.

For more information on the Ansible git module, see https://docs.ansible.com/ansible/latest/modules/git_module.html.

WORKING WITH WEBHOOKS

A **Webhook** provides the ability to execute specified commands between apps over the web. Ansible Tower currently provides webhook integration with GitHub and GitLab. This section describes the procedure for setting up a webhook in Tower through their respective services.

- *GitHub webhook setup*
- *GitLab webhook setup*
- *Payload output*

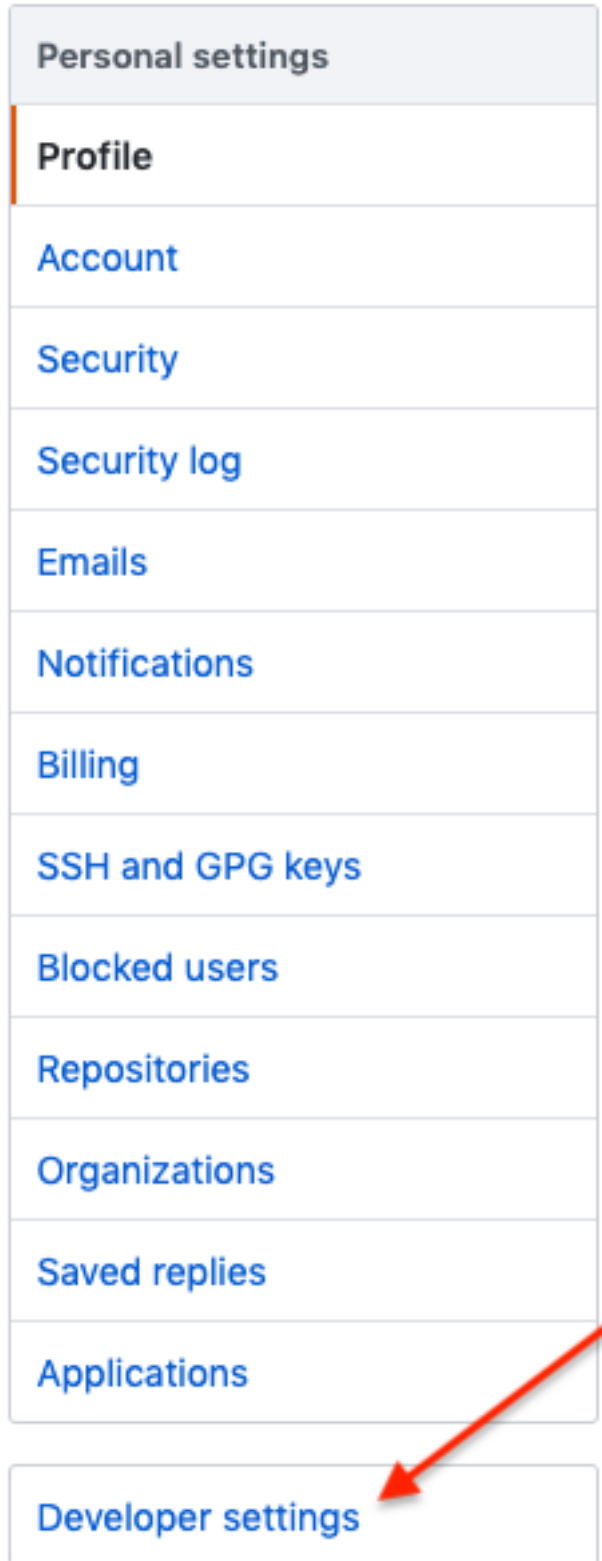
The webhook post-status-back functionality for GitHub and GitLab is designed for work only under certain CI events. Receiving another kind of event will result in messages like the one below in the Tower log:

```
awx.main.models.mixins Webhook event did not have a status API endpoint associated, ↵  
↪ skipping.
```

22.1 GitHub webhook setup

Tower 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 Tower 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 Tower.
 - a. In the profile settings of your GitHub account, click **Settings**.
 - b. Below the Personal 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, Tower 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

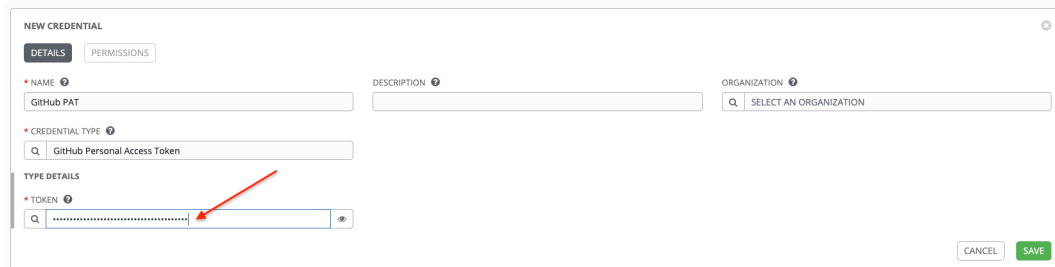
Scopes define the access for personal tokens. [Read more about OAuth scopes](#).

- | | |
|---|--|
| <input type="checkbox"/> repo | Full control of private repositories |
| <input checked="" type="checkbox"/> repo:status | Access commit status |
| <input checked="" type="checkbox"/> repo_deployment | Access deployment status |
| <input checked="" type="checkbox"/> public_repo | Access public repositories |
| <input type="checkbox"/> repo:invite | Access repository invitations |
| <input type="checkbox"/> write:packages | Upload packages to github package registry |

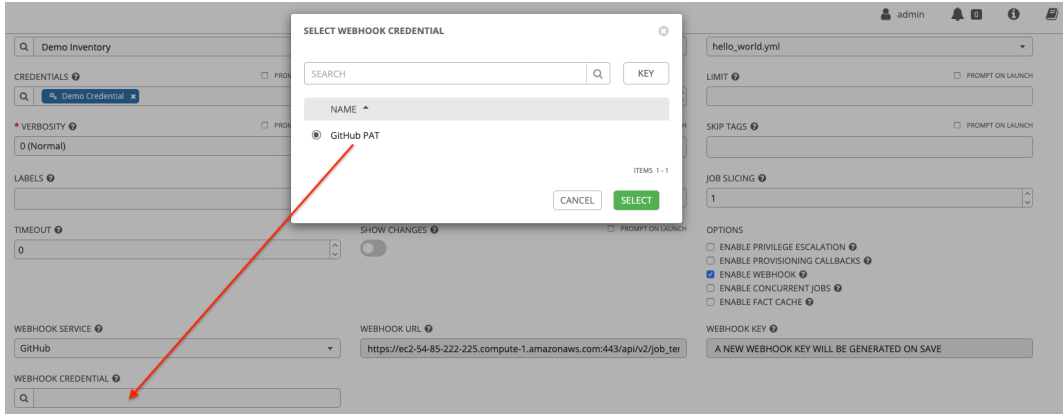
Click here for more information on scopes



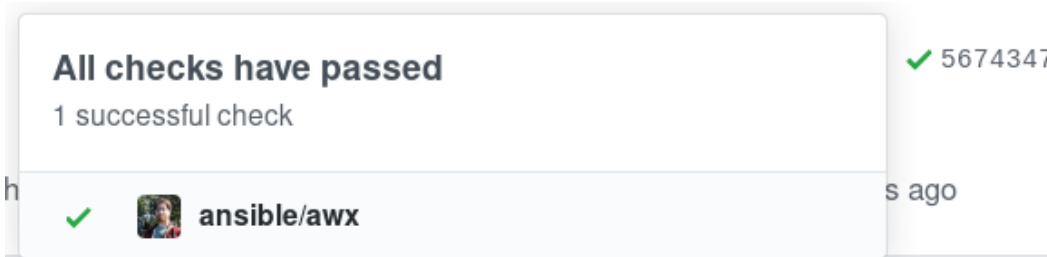
- h. Click the **Generate Token** button.
 - i. Once the token is generated, make sure you copy the PAT, as it will be used by Tower in a later step. You will not be able to access this token again in GitHub.
2. Use the PAT to optionally create a GitHub credential in Tower:
- a. Go to your Tower instance, 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.



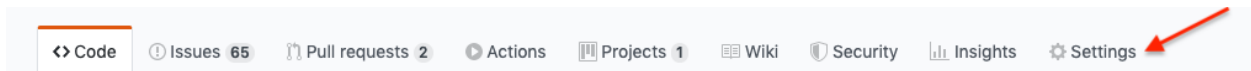
- c. Go to the job template with which you want to enable webhooks, and select the webhook service and credential you created in the previous step.



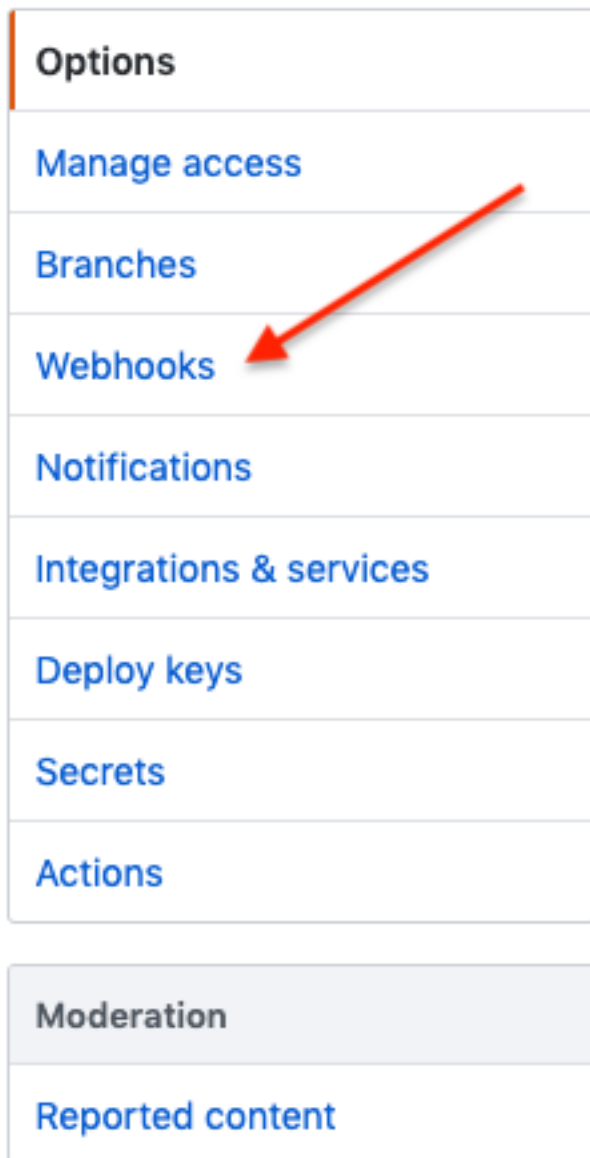
- d. Click **Save**. Now your job template is set up to be able to post back to GitHub. An example of one may look like this:



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 in Tower* (or in a *workflow job template*), which will provide you with the following information:
 - a. Copy the contents of the **Webhook URL** from the Tower 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 Tower 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 *

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.

<input type="checkbox"/> Check runs Check run is created, requested, rerequested, or completed.	<input type="checkbox"/> Check suites Check suite is requested, rerequested, or completed.
---	--

≈

<input type="checkbox"/> Packages GitHub Packages published or updated in a repository.	<input type="checkbox"/> Page builds Pages site built.
<input type="checkbox"/> Projects Project created, updated, or deleted.	<input type="checkbox"/> Project cards Project card created, updated, or deleted.
<input type="checkbox"/> Project columns Project column created, updated, moved or deleted.	<input type="checkbox"/> Visibility changes Repository changes from private to public.
<input checked="" type="checkbox"/> 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.	<input type="checkbox"/> Pull request reviews Pull request review submitted, edited, or dismissed.
<input type="checkbox"/> Pull request review comments Pull request diff comment created, edited, or deleted.	<input type="checkbox"/> 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.

Add webhook

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

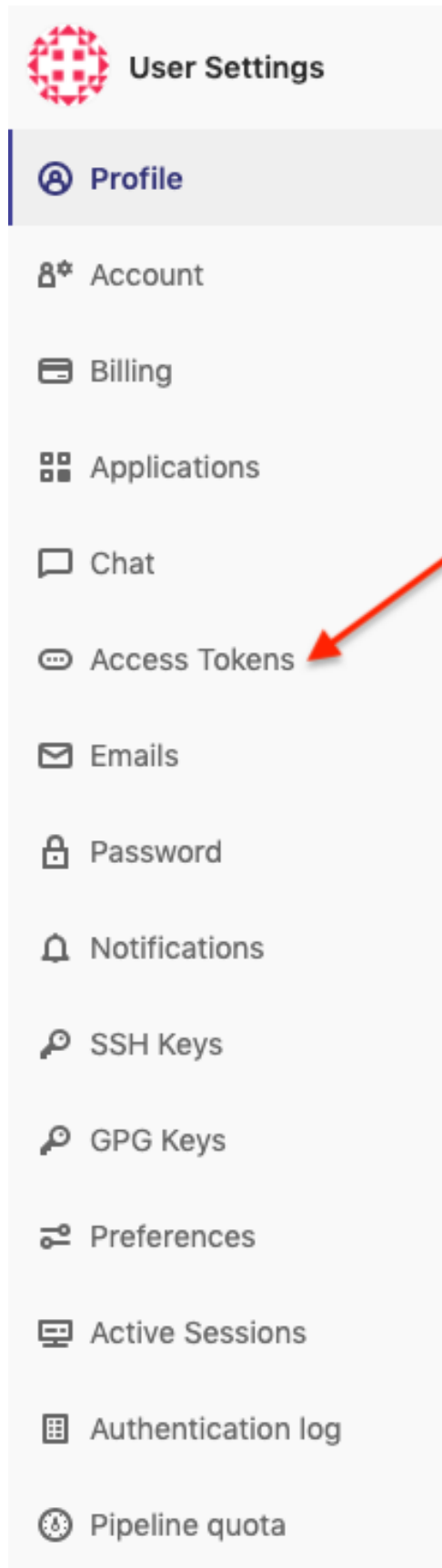
✓	 c95a4066-3bc7-11ea-8290-31d813a11c5f	2020-01-20 13:59:45	...
✓	 5262c3c6-3bad-11ea-852e-7f2ef6d8c650	2020-01-20 10:50:18	...
⚠	 9a7066c2-39ce-11ea-97f5-f23c9e55ce03	2020-01-18 01:43:30	...

For more information, refer to the [GitHub Webhooks developer documentation](#).

22.2 GitLab webhook setup

Tower 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 Tower 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 Tower. This token gives Tower 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**.



- 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 Tower, 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

Expires at

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).

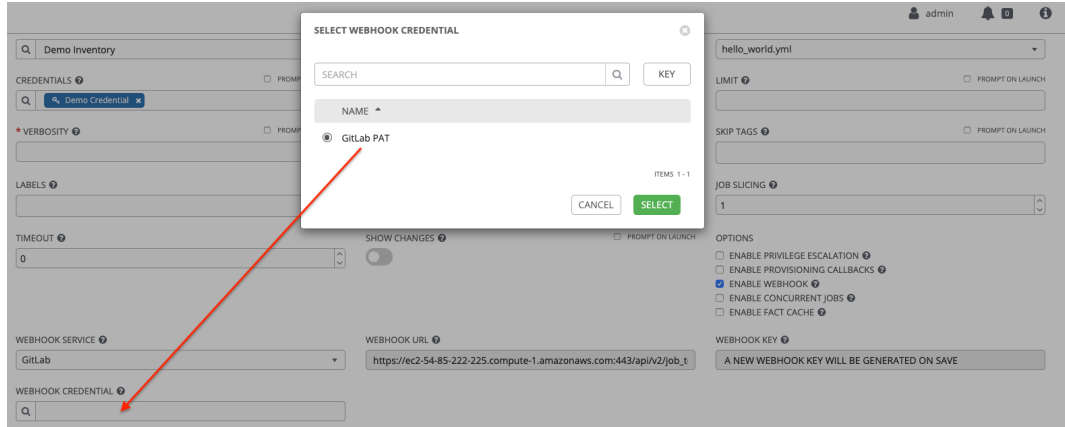
 read_registry

Grants read-only access to container registry images on private projects.

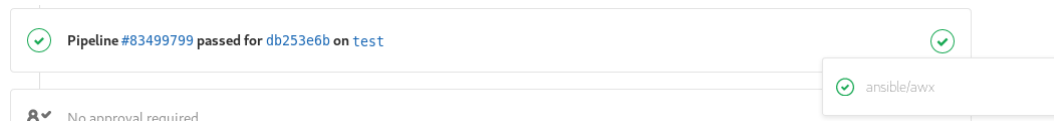
Create personal access token

- 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 by Tower 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 in Tower:
- a. Go to your Tower 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.

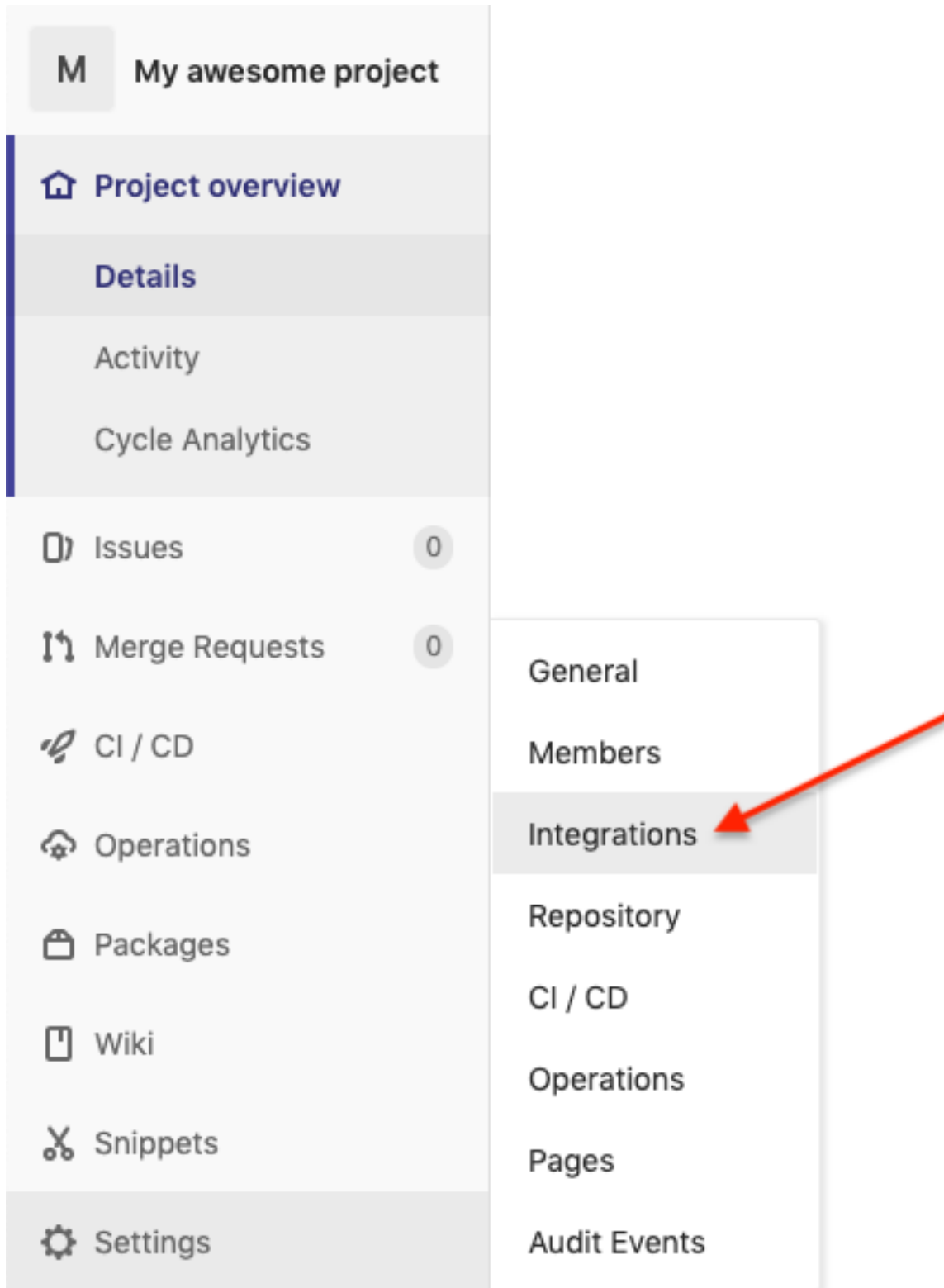
- c. Go to the job template with which you want to enable webhooks, and select the webhook service and credential you created in the previous step.



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:



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 in Tower* (or in a *workflow job template*), which will provide you with the following information:
 - a. Copy the contents of the **Webhook URL** from the Tower 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 Tower 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

Secret Token

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

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](#).



22.3 Payload output

The entire payload is exposed as an extra variable from Tower. 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 `tower_webhook_payload` variable, as shown in the example below.

[JOBS](#) / 3 - Demo Job Template

DETAILS

STATUS	● Successful
STARTED	1/15/2020 3:29:31 PM
FINISHED	1/15/2020 3:29:36 PM
JOB TEMPLATE	Demo Job Template
JOB TYPE	Run
LAUNCHED BY	Webhook
INVENTORY	Demo Inventory
PROJECT	Demo Project
REVISION	347e44f 
PLAYBOOK	hello_world.yml
CREDENTIAL	Demo Credential
ENVIRONMENT	/env/ansible
EXECUTION NODE	awx
INSTANCE GROUP	tower
EXTRA VARIABLES 	<div style="display: flex; align-items: center; gap: 5px;"> YAML JSON </div> <div style="text-align: right; margin-top: 5px;">EXPAND</div>

```

1 tower_webhook_event_guid: b7d56a00-37d5-11ea-8057-ad885c26760a
2 tower_webhook_event_ref: null
3 tower_webhook_event_type: ping
4 tower_webhook_payload:
5   hook:
6     active: true

```

EXTRA VARIABLES ?

YAML JSON

```

1 tower_webhook_event_guid: b7d56a00-37d5-11ea-8057-ad885c26760a
2 tower_webhook_event_ref: null
3 tower_webhook_event_type: ping
4 tower_webhook_payload:
5   hook:
6     active: true
7     config:
8       content_type: json
9       insecure_ssl: '1'
10      secret: '*****'
11      url: 'https://cb37189d.ngrok.io:443/api/v2/job_templates/7/github/'
12      created_at: '2020-01-15T20:29:24Z'
13      events:
14        - push
15      id: 175270916
16      last_response:
17        code: null
18        message: null
19        status: unused
20      name: web
21      ping_url: 'https://api.github.com/repos/jbradberry/awx/hooks/175270916/pings'
22      test_url: 'https://api.github.com/repos/jbradberry/awx/hooks/175270916/test'
23      type: Repository
24      updated_at: '2020-01-15T20:29:24Z'
25      url: 'https://api.github.com/repos/jbradberry/awx/hooks/175270916'
26      hook_id: 175270916
27      repository:
28        archive_url: 'https://api.github.com/repos/jbradberry/awx/{archive_format}/{ref}'
29        archived: false
30        assignees_url: 'https://api.github.com/repos/jbradberry/awx/assignees{/user}'
31        blobs_url: 'https://api.github.com/repos/jbradberry/awx/git/blobs{/sha}'
32        branches_url: 'https://api.github.com/repos/jbradberry/awx/branches{/branch}'
33        clone_url: 'https://github.com/jbradberry/awx.git'
34        collaborators_url: 'https://api.github.com/repos/jbradberry/awx/collaborators{/collaborator}'
35        comments_url: 'https://api.github.com/repos/jbradberry/awx/comments{/number}'
36        commits_url: 'https://api.github.com/repos/jbradberry/awx/commits{/sha}'
37        compare_url: 'https://api.github.com/repos/jbradberry/awx/compare/{base}...{head}'
38        contents_url: 'https://api.github.com/repos/jbradberry/awx/contents/{+path}'
39        contributors_url: 'https://api.github.com/repos/jbradberry/awx/contributors'
40        created_at: '2018-12-20T19:39:26Z'
41        default_branch: devel
42        deployments_url: 'https://api.github.com/repos/jbradberry/awx/deployments'
43        description: AWX Project
44        disabled: false
45        downloads_url: 'https://api.github.com/repos/jbradberry/awx/downloads'
46        events_url: 'https://api.github.com/repos/jbradberry/awx/events'

```

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 Tower REST API at the `/api/v2/notification_templates` endpoint (either through the API or through the Tower 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.

23.1 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

23.2 Workflow

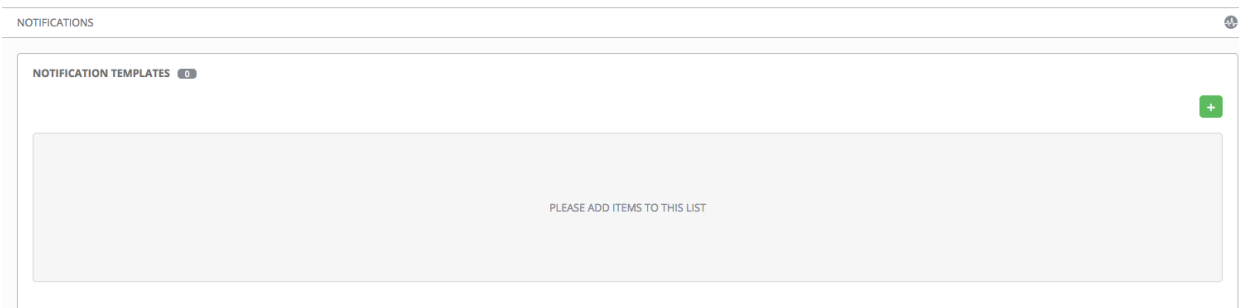
When a job succeeds or fails, the error or success handler will pull a list of relevant notification templates using the procedure defined above. It will then create a Notification object for each one containing relevant details about the job and then sends it to the destination (email addresses, slack 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>`).

23.3 Create a Notification Template

To create a Notification Template:

1. Click the Notifications () icon from the left navigation bar.



2. Click the  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.

23.4 Notification Types

Notification types supported with Ansible Tower:

- *Email*
- *Grafana*
- *HipChat*
- *IRC*
- *Mattermost*
- *PagerDuty*
- *Rocket.Chat*
- *Slack*
- *Twilio*
- *Webhook*
 - *Webhook payloads*

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.

23.4.1 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 Tower may attempt connecting to the email server before giving up.

NEW NOTIFICATION TEMPLATE

* NAME: DESCRIPTION: * ORGANIZATION:

* TYPE:

TYPE DETAILS

USERNAME: PASSWORD: * HOST:

* RECIPIENT LIST : * SENDER EMAIL: * PORT:

* TIMEOUT : EMAIL OPTIONS:

CUSTOMIZE MESSAGES...

23.4.2 Grafana

Grafana is a fairly straightforward integration. First, create an API Key in the [Grafana system](#) (this is the token that is given to Tower).

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 Tower).

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 event(s) of the notification you are configuring.
- **Disable SSL Verification:** SSL verification is on by default, but you can choose to turn off Tower's attempt to verify the authenticity of the target's certificate. Environments that use internal or private CA's should select this option to disable verification.

NEW NOTIFICATION TEMPLATE

<p>* NAME</p> <input type="text" value="Grafana notification"/>	<p>DESCRIPTION</p> <input type="text"/>	<p>* ORGANIZATION</p> <input type="text" value="Default"/>
<p>* TYPE</p> <input type="text" value="Grafana"/>		
TYPE DETAILS		
<p>* GRAFANA URL</p> <input type="text" value="https://grafana.com"/>	<p>* GRAFANA API KEY</p> <input type="text" value="SHOW"/>	<p>ID OF THE DASHBOARD (OPTIONAL)</p> <input type="text"/>
<p>ID OF THE PANEL (OPTIONAL)</p> <input type="text"/>	<p>TAGS FOR THE ANNOTATION (OPTIONAL)</p> <input type="text" value="ansible"/>	<p><input type="checkbox"/> DISABLE SSL VERIFICATION</p>
<p>CUSTOMIZE MESSAGES... <input type="checkbox"/></p>		
		<p><input type="button" value="CANCEL"/> <input type="button" value="SAVE"/></p>

23.4.3 HipChat

HipChat has been discontinued as of Ansible Tower 3.8.

23.4.4 IRC

The Tower 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 Tower notification bot also supports SSL authentication. The Tower 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

NEW NOTIFICATION TEMPLATE

* NAME: IRC Notification

DESCRIPTION: [Empty]

* ORGANIZATION: Honey Dog, Inc.

* TYPE: IRC

TYPE DETAILS

IRC SERVER PASSWORD: SHOW [Masked]

* IRC SERVER PORT: 6667

* IRC SERVER ADDRESS: irc.testirc.net

* IRC NICK: helpbot

* DESTINATION CHANNELS OR USERS: #engineering, #release-engineers

SSL CONNECTION

CUSTOMIZE MESSAGES...

CANCEL SAVE

23.4.5 Mattermost

The Mattermost notification type in Ansible Tower 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 Tower’s attempt to verify the authenticity of the target’s certificate. Environments that use internal or private CA’s should select this option to disable verification.

NEW NOTIFICATION TEMPLATE

* NAME: Mattermost Notification

DESCRIPTION: [Empty]

* ORGANIZATION: Honey Dog, Inc.

* TYPE: Mattermost

TYPE DETAILS

* TARGET URL: http://1.2.3.4:8065/hooks/jSkurmybl5134pnf9sdpjts

USERNAME: beth

CHANNEL: my-channel

ICON URL: https://www.myicon/favicon.ico

DISABLE SSL VERIFICATION

CUSTOMIZE MESSAGES...

CANCEL SAVE

23.4.6 PagerDuty

PagerDuty is a fairly straightforward integration. First, create an API Key in the [PagerDuty system](#) (this is the token that is given to Tower) and then create a “Service” which provides an “Integration Key” that will also be given to Tower. 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 Tower).
- **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 “towertest”, the web dashboard will be at `towertest.pagerduty.com` and you will give the Tower API `towertest` 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.

The screenshot shows a 'NEW NOTIFICATION TEMPLATE' form with the following fields and values:

- * NAME:** PagerDuty Notification
- DESCRIPTION:** (empty)
- * ORGANIZATION:** Honey Dog, Inc.
- * TYPE:** Pagerduty
- TYPE DETAILS:**
 - * API TOKEN:** SHOW [REDACTED]
 - * PAGERDUTY SUBDOMAIN:** pagerduty.subdomain.com
 - * API SERVICE/INTEGRATION KEY:** efk3ou7wpo3L3JJiORO
- * CLIENT IDENTIFIER:** 322393
- CUSTOMIZE MESSAGES...:** (toggle switch is off)

Buttons for CANCEL and SAVE are located at the bottom right of the form.

23.4.7 Rocket.Chat

The Rocket.Chat notification type in Ansible Tower 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 Tower’s attempt to verify the authenticity of the target’s certificate. Environments that use internal or private CA’s should select this option to disable verification.

NEW NOTIFICATION TEMPLATE

* NAME: RocketChat Notification DESCRIPTION: * ORGANIZATION: Honey Dog, Inc.

* 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...

23.4.8 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 Tower. Note that private messages are not supported.

NEW NOTIFICATION TEMPLATE

* NAME: Slack notification DESCRIPTION: * ORGANIZATION: Honey Dog, Inc.

* TYPE: Slack

TYPE DETAILS

* DESTINATION CHANNELS: #engineering, #helpdesk, #support * TOKEN: SHOW NOTIFICATION COLOR:

CUSTOMIZE MESSAGES...

23.4.9 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

The screenshot shows a form titled "NEW NOTIFICATION TEMPLATE" with the following fields and values:

- * NAME:** Twilio Notification
- DESCRIPTION:** (empty)
- * ORGANIZATION:** Honey Dog, Inc.
- * TYPE:** Twilio
- TYPE DETAILS:**
 - * ACCOUNT TOKEN:** SHOW [REDACTED]
 - * SOURCE PHONE NUMBER:** 9109876555
 - * DESTINATION SMS NUMBER:** 9109876555
- * ACCOUNT SID:** Aa54e6dg4345d3t676f60
- CUSTOMIZE MESSAGES...:** (toggle is off)
- Buttons:** CANCEL, SAVE

23.4.10 Webhook

The webhook notification type in Ansible Tower provides a simple interface to sending POSTs to a predefined web service. Tower 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 Tower’s attempt to verify 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.

The screenshot shows the 'NEW NOTIFICATION TEMPLATE' form. Key fields include:

- NAME:** Webhook notification
- DESCRIPTION:** (empty)
- ORGANIZATION:** Honey Dog, Inc.
- TYPE:** Webhook
- TYPE DETAILS:**
 - USERNAME:** janedoe
 - BASIC AUTH PASSWORD:** (masked with dots)
 - TARGET URL:** http://www.honeydog.com/web/db/notification
- DISABLE SSL VERIFICATION:** (unchecked)
- HTTP HEADERS:** [{"Authentication": "988881adc9fc3655077dc2d4d757d480b5ea0e11", "MessageType": "Test"}]
- HTTP METHOD:** (dropdown menu open, showing POST and PUT options)

Webhook payloads

Tower 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
    
```

(continues on next page)

(continued from previous page)

```
hosts
http method
```

An example of a started notifications via webhook message as it is returned by Tower:

```
{"id": 38, "name": "Demo Job Template", "url": "https://towerhost/#/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
```

Tower 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 Tower:

```
{"id": 46, "name": "AWX-Collection-tests-tower_job_wait-long_running-XVFBGRSAvUUIrYKn
↵", "url": "https://towerhost/#/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-
↵tower_job_wait-long_running-JJS1glnwtsRJyQmw", "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}}}
```

23.5 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.

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 }}), or attributes of the job such as ({{ job.status }}). You may apply a number of possible variables in the message. Refer to the Ansible Tower documentation for more details.

START MESSAGE

```
1 {{ job_friendly_name }} #{{ job.id }} '{{ job.name }}' {{ job.status }}: {{ url }}
```

START MESSAGE BODY

```
1 {{ job_friendly_name }} #{{ job.id }} had status {{ job.status }}, view details at {{ url }}
2
3 {{ job_metadata }}
```

SUCCESS MESSAGE

```
1 {{ job_friendly_name }} #{{ job.id }} '{{ job.name }}' {{ job.status }}: {{ url }}
```

SUCCESS MESSAGE BODY

```
1 {{ job_friendly_name }} #{{ job.id }} had status {{ job.status }}, view details at {{ url }}
2
3 {{ job_metadata }}
```

ERROR MESSAGE

```
1 {{ job_friendly_name }} #{{ job.id }} '{{ job.name }}' {{ job.status }}: {{ url }}
```

ERROR MESSAGE BODY

```
1 {{ job_friendly_name }} #{{ job.id }} had status {{ job.status }}, view details at {{ url }}
2
3 {{ job_metadata }}
```

WORKFLOW APPROVED MESSAGE

```
1 The approval node "{{ approval_node_name }}" was approved. {{ workflow_url }}
```

WORKFLOW APPROVED MESSAGE BODY

```
1 The approval node "{{ approval_node_name }}" was approved. {{ workflow_url }}
2
3 {{ job_metadata }}
```

WORKFLOW DENIED MESSAGE

```
1 The approval node "{{ approval_node_name }}" was denied. {{ workflow_url }}
```

WORKFLOW DENIED MESSAGE BODY

```
1 The approval node "{{ approval_node_name }}" was denied. {{ workflow_url }}
2
3 {{ job_metadata }}
```

WORKFLOW PENDING APPROVAL MESSAGE

```
1 The approval node "{{ approval_node_name }}" needs review. This node can be viewed at: {{ workflow_url }}
```

WORKFLOW PENDING APPROVAL MESSAGE BODY

```
1 The approval node "{{ approval_node_name }}" needs review. This approval node can be viewed at: {{ workflow_url }}
2
3 {{ job_metadata }}
```

WORKFLOW TIMED OUT MESSAGE

```
1 The approval node "{{ approval_node_name }}" has timed out. {{ workflow_url }}
```

WORKFLOW TIMED OUT MESSAGE BODY

```
1 The approval node "{{ approval_node_name }}" has timed out. {{ workflow_url }}
2
3 {{ job_metadata }}
```

CANCEL SAVE

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, Tower displays the fields as **Message** and **Message Body**. Other notification types only expect a **Message** for each type of event:

```

MESSAGE
1 {{ job_friendly_name }} #{{ job.id }} '{{ job.name }}' {{ job.status }}: {{ url }}

MESSAGE BODY
1 {{ job_friendly_name }} #{{ job.id }} had status {{ job.status }}, view details at {{ url }}
2
3 {{ 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 Tower, as shown in the pre-populated **Messages** fields.

```

SUCCESS MESSAGE
1 {{ job_friendly_name }} #{{ job.id }} '{{ job.name }}' {{ job.status }}: {{ url }}
    
```

Variable Attribute

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

The resulting dictionary will look something like this:

```
{
  "id": 18,
  "name": "Project - Space Procedures",
  "url": "https://towerhost/#/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://towerhost/#/jobs/playbook/12",
  "created_by": "admin",
  "started": "2019-10-26T00:02:07.943774+00:00",
  "finished": null,
  "status": "running",
  "traceback": ""
}
```

(continues on next page)

(continued from previous page)

```

"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://towerhost/#/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](#).

Tower 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](#) in the *Ansible Automation Platform Installation and Reference 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.

23.6 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

PROJECTS / Example / NOTIFICATIONS

Example

DETAILS PERMISSIONS **NOTIFICATIONS** JOB TEMPLATES SCHEDULES

SEARCH Q KEY

NAME	TYPE	START	SUCCESS	FAILURE
Email notification for job Starts	Email	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Slack notification	Slack	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
SMS notification to Self	Pagerduty	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Web notification	Webhook	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>

ITEMS 1 - 4

For workflow templates that have approval nodes, in addition to *Start*, *Success*, and *Failure*, you can enable or disable certain approval-related events:

New Workflow Job Template

DETAILS PERMISSIONS **NOTIFICATIONS** COMPLETED JOBS SCHEDULES

SEARCH Q KEY

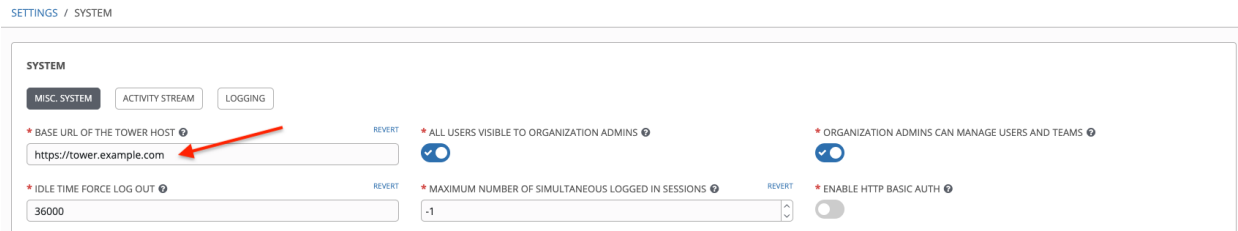
NAME	TYPE	APPROVAL	START	SUCCESS	FAILURE
Email notification for job starts	Email	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Slack notification	Slack	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
SMS notification to Self	Pagerduty	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Web notification	Webhook	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

ITEMS 1 - 4

Refer to *Approval nodes* for additional detail on working with these types of nodes.

23.7 Configure the `towerhost` hostname for notifications

In the *System Settings*, you can replace `https://tower.example.com` in the **Base URL of The Tower Host** field with your preferred hostname to change the notification hostname.



Refreshing your Tower license also changes the notification hostname. New installations of Ansible Tower should not have to set the hostname for notifications.

23.7.1 Reset the `TOWER_URL_BASE`

The primary way that Tower 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.

Tower takes settings values from the database first. If no settings values are found, Tower falls back to using the values from the settings files. If a user posts a license by navigating to the Tower host's IP address, 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 the license from the Tower



Settings () Menu's **License** tab using the DNS entry you wish to appear in notifications, and re-add your license.

23.8 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

SCHEDULES



You can access all your configured schedules by clicking the Schedules icon () 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.

Use the **ON/OFF** toggle next to the schedule name to enable/disable that schedule. Each schedule has a corresponding **Actions** column that has options to allow editing () or deleting () the schedule.

SCHEDULED JOBS 3			
SEARCH <input type="text"/> <input type="button" value="Q"/> <input type="button" value="KEY"/>			
NAME ▾	TYPE ▲	NEXT RUN ▾	ACTIONS
<input checked="" type="checkbox"/> Schedule 1	SCM Update	12/6/2019 12:00:00 AM	
<input checked="" type="checkbox"/> Schedule 2	SCM Update	1/1/2020 12:02:00 AM	
<input checked="" type="checkbox"/> Run Once	SCM Update		
<input checked="" type="checkbox"/> Schedule 3	SCM Update	12/25/2019 12:03:00 AM	
<input checked="" type="checkbox"/> Cleanup Expired Sessions	Management Job		
<input checked="" type="checkbox"/> Cleanup Expired OAuth 2 Tokens	Management Job		
<input checked="" type="checkbox"/> Cleanup Activity Schedule	Management Job	12/10/2019 10:51:13 AM	
<input checked="" type="checkbox"/> Cleanup Job Schedule	Management Job	12/8/2019 10:51:13 AM	

ITEMS 1-8

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 with the following information:

- **Name:** Clicking the schedule name opens the **Edit Schedule** dialog
- **First Run:** The first scheduled run of this task
- **Next Run:** The next scheduled run of this task
- **Final Run:** If the task has an end date, this is the last scheduled run of the task

SEARCH <input type="text"/> <input type="button" value="Q"/> <input type="button" value="KEY"/> <input type="button" value="+"/>				
NAME ▲	FIRST RUN ▾	NEXT RUN ▾	FINAL RUN ▾	ACTIONS
<input checked="" type="checkbox"/> Schedule 1	12/4/2019 12:00:00 AM	12/4/2019 12:00:00 AM		
<input checked="" type="checkbox"/> Schedule 2	12/4/2019 12:00:00 AM	12/4/2019 12:00:00 AM	12/4/2019 12:00:00 AM	
<input checked="" type="checkbox"/> Schedule 3	12/5/2019 11:00:00 PM	12/5/2019 11:00:00 PM	12/26/2019 11:00:00 PM	
<input checked="" type="checkbox"/> Schedule 4	11/3/2019 3:00:00 AM	1/5/2020 3:00:00 AM	2/2/2020 3:00:00 AM	


ITEMS 1-4

24.1 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  button, which opens the **Create Schedule** window.

CREATE SCHEDULE ✕

<p>* NAME</p> <input type="text" value="Schedule name"/>	<p>* START DATE</p> <input type="text" value="1/31/2019"/>	<p>* START TIME (HH24:MM:SS)</p> <input type="text" value="0"/> : <input type="text" value="0"/> : <input type="text" value="0"/>
<p>* LOCAL TIME ZONE</p> <input type="text" value="America/Denver"/>	<p>* REPEAT FREQUENCY</p> <input type="text" value="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 DESCRIPTION** 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.

SCHEDULE DESCRIPTION

every month on the 15th for 15 times

OCCURRENCES (Limited to first 10) DATE FORMAT LOCAL TIME ZONE UTC

```

06-15-2019 02:00:00
07-15-2019 02:00:00
08-15-2019 02:00:00
09-15-2019 02:00:00
10-15-2019 02:00:00
11-15-2019 02:00:00
12-15-2019 02:00:00
01-15-2020 02:00:00
02-15-2020 02:00:00
03-15-2020 02:00:00

```

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. Essentially, Tower 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.

SEARCH

NAME ^	FIRST RUN ↕	NEXT RUN ↕	FINAL RUN ↕	ACTIONS
<input type="checkbox"/> Schedule 1	12/4/2019 12:00:00 AM			
<input checked="" type="checkbox"/> Schedule 2	12/4/2019 12:00:00 AM	12/4/2019 12:00:00 AM	12/4/2019 12:00:00 AM	
<input type="checkbox"/> Schedule 3	12/5/2019 11:00:00 PM		12/26/2019 11:00:00 PM	
<input checked="" type="checkbox"/> Schedule 4	11/3/2019 3:00:00 AM	1/5/2020 3:00:00 AM	2/2/2020 3:00:00 AM	





ITEMS 1 - 4

SETTING UP AN INSIGHTS PROJECT

Tower 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. Tower tracks the maintenance plan playbooks via an Insights project in Tower. Authentication to Insights via Basic Auth, from Tower, is backed by a special Insights Credential, which must first be established in Tower. To ultimately run an Insights Maintenance Plan in Tower, you need an Insights project, an inventory, and a Scan Job template.

25.1 Create Insights Credential

To create a new credential for use with Insights:

1. Click the Credentials () icon from the left navigation bar to access the Credentials page.
2. Click the  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 click the  button and select it from the credential type pop-up window.

SELECT CREDENTIAL TYPE ✕

NAME ▲

GitLab Personal Access Token

Google Compute Engine

HashiCorp Vault Secret Lookup

HashiCorp Vault Signed SSH

Insights

< 1 2 3 4 5 >
PAGE 2 OF 5

ITEMS 6 - 10 OF 22

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 CREDENTIAL 🔒

NEW CREDENTIAL ✕

DETAILS

PERMISSIONS

* NAME ⓘ

DESCRIPTION ⓘ

ORGANIZATION ⓘ

* CREDENTIAL TYPE ⓘ

TYPE DETAILS






* USERNAME

* PASSWORD

8. Click **Save** when done.

25.2 Create an Insights Project



To create a new Insights project:

1. Click the Projects () icon from the left navigation bar to access the Projects page.
2. Click the  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 Help  button next to the options.

PROJECTS / CREATE PROJECT ⌵


NEW PROJECT ✕

DETAILS | PERMISSIONS | JOB TEMPLATES | SCHEDULES




* NAME: Insights Project DESCRIPTION: * ORGANIZATION : 

* SCM TYPE: Red Hat Insights


SOURCE DETAILS

* CREDENTIAL: 

SCM UPDATE OPTIONS

CLEAN  DELETE ON UPDATE  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 Tower 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 6

SEARCH Q KEY +


	Compact	Expanded	Name (Ascending) ▾
<p>● Demo Example GIT</p> <p>REVISION 6d10f00 ORGANIZATION Default LAST MODIFIED 5/2/2019 1:13:23 PM LAST USED 5/2/2019 1:13:23 PM</p>			
<p>● Demo Project GIT</p> <p>REVISION 347e44f ORGANIZATION Default LAST MODIFIED 5/1/2019 12:00:10 AM LAST USED 5/1/2019 12:00:10 AM</p>			
<p>● Insights Project GIT</p> <p>REVISION 6d10f00 ORGANIZATION Default LAST MODIFIED 5/2/2019 5:41:38 PM LAST USED 5/2/2019 5:41:38 PM</p>			
<p>● Multi-Vault Project GIT</p> <p>REVISION 6d10f00 ORGANIZATION Default LAST MODIFIED 5/2/2019 4:56:35 PM LAST USED 5/2/2019 4:56:35 PM</p>			
<p>● Project from Git GIT</p> <p>REVISION dc16012 ORGANIZATION Default LAST MODIFIED 5/2/2019 12:01:15 PM LAST USED 2/4/2019 9:18:37 AM</p>			
<p>● Scan Project GIT</p> <p>REVISION 6d10f00 ORGANIZATION Organization - PinWheel LAST MODIFIED 5/2/2019 5:33:42 PM LAST USED 4/24/2019 12:26:44 PM</p>			

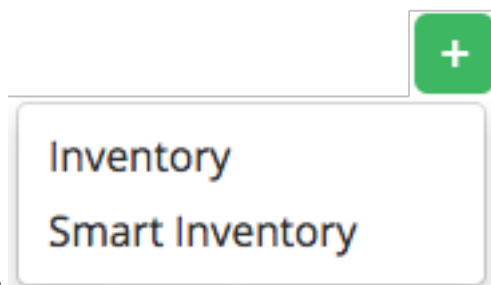
ITEMS 1 - 6


25.3 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. Therefore, make sure that the hostnames in the Tower inventory match up with the system in the Red Hat Insights Portal.

To create a new inventory for use with Insights:

1. Click the Inventories () icon from the left navigation bar to access the Inventories page.



2. Click the **Inventory** button and select **Inventory** from the drop-down menu list to launch a New Inventory window.
3. Enter the name and organization to be used in their respective fields.
4. In the **Insights Credential** field, enter the name of the Insights credential you previously created, or click the  button and select it from the pop-up window.

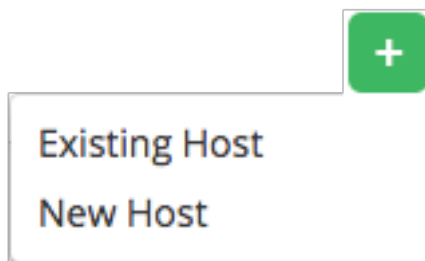
INVENTORIES / Insights Inventory

Insights Inventory

1 ---

5. Click **Save** and proceed to add a host.



Note: Typically, your inventory already contains Insights hosts. Tower just doesn't know about them yet. The Insights credential allows Tower to get information from Insights about an Insights host. Tower identifying a host as an Insights host can occur without an Insights credential with the help of `scan_facts.yml` file. For instructions, refer to the [Create a Scan Job Template](#) section.




6. Click the **Hosts** tab and click the **+** button to open the **Create Host** dialog.
7. Enter the name in the **Host Name** field associated with the Insights host that will be used.
8. Click **Save** when done.


25.4 Create a Scan Project

In order for Tower to utilize Insights Maintenance Plans, it must have visibility to them. Create and run a scan job against the inventory using a stock manual scan playbook.

1. Click the Projects () icon from the left navigation bar to access the Projects page.
2. Click the  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 scan project.

- **Organization:** The name of the organization is pre-populated with the organization you chose from creating the inventory.
 - **SCM Type:** Select **Git**.
 - Upon selecting the SCM type, the **Source Details** field expands.
4. In the **SCM URL** field, enter `https://github.com/ansible/awx-facts-playbooks`. This is the location where the scan job template is stored.
 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 Help  button next to the options.

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.

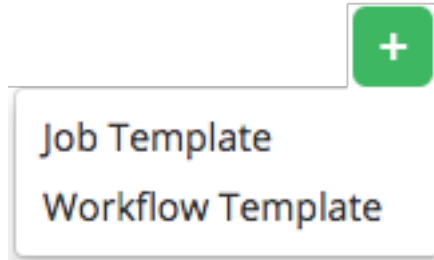
Syncing imports into Tower any Maintenance Plans in your Insights account that has a playbook solution. It will use the default Plan resolution. Notice that the status dot beside the name of the project updates once the sync has run.





25.5 Create a Scan Job Template

Create a scan job template that uses the fact scan playbook:



1. Click the Templates () icon from the left navigation bar to access the Templates page.



2. Click the  button and select **Job Template** from the drop-down menu list to launch a New Job Template window.
3. Enter the appropriate details into the required fields, at minimum. Note the following fields requiring specific Insights-related entries:
 - **Name:** Enter the name of your scan job.
 - **Job Type:** Choose **Run** from the drop-down menu list.
 - **Inventory:** Enter the name of the Insights inventory, or click the  button and select it from the pop-up window.
 - **Project:** Enter the name of the Scan project you previously created, or click the  button and select it from the pop-up window.
 - **Playbook:** Select `scan_facts.yml` from the drop-down menu list. This is the playbook associated with the Scan project you previously set up.
 - **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.
4. Click to select **Enable Privilege Escalation** and **Enable Fact Cache** from the Options field.

A scan job template for Insights should be launched with the Privilege Escalation option enabled to allow the job to access `/etc/redhat-access-insights/machine-id` as a root user in order to obtain the value of `system_id` from the target host. What this does is activate the Insights button from the Host, which is needed to *remediate the Insights inventory*. Otherwise, the `system_id` parameter in the result of your scan job is set to null and the Insights button will not appear.

NEW JOB TEMPLATE

DETAILS | PERMISSIONS | COMPLETED JOBS | SCHEDULES | **ADD SURVEY**

* NAME: Insights Scan | DESCRIPTION: | * JOB TYPE: Run PROMPT ON LAUNCH

* INVENTORY: Insights Inventory PROMPT ON LAUNCH | * PROJECT: Scan Project | * PLAYBOOK: scan_facts.yml

CREDENTIALS: Demo Credential PROMPT ON LAUNCH | FORKS: 0 | LIMIT: PROMPT ON LAUNCH

* VERBOSITY: 0 (Normal) PROMPT ON LAUNCH | JOB TAGS: PROMPT ON LAUNCH | SKIP TAGS: PROMPT ON LAUNCH

LABELS: | INSTANCE GROUPS: | JOB SLICING: 1


TIMEOUT: 0 | SHOW CHANGES: PROMPT ON LAUNCH | OPTIONS:
 ENABLE PRIVILEGE ESCALATION
 ENABLE PROVISIONING CALLBACKS
 ENABLE WEBHOOK
 ENABLE CONCURRENT JOBS
 ENABLE FACT CACHE

EXTRA VARIABLES: **YAML** | JSON PROMPT ON LAUNCH

```
1 ---
```

LAUNCH | CANCEL | **SAVE**

5. Click **Save** when done.

6. Click the  icon to launch the scan job template.

Once complete, the job results display in the Job Details page.

JOBS / 11 - Insights Scan

DETAILS

STATUS: ● Successful

STARTED: 10/6/2017 12:14:22 PM

FINISHED: 10/6/2017 12:14:30 PM

TEMPLATE: Insights Scan

JOB TYPE: Run

LAUNCHED BY: admin

INVENTORY: Insights Inventory

PROJECT: ● Scan Project

REVISION: 77ebb77

PLAYBOOK: scan_facts.yml

MACHINE CREDENTIAL: Demo Credential

FORKS: 0

VERBOSITY: 0 (Normal)

INSTANCE GROUP: tower

Insights Scan | PLAYS: 1 | TASKS: 8 | HOSTS: 1 | ELAPSED: 00:00:07

SEARCH | KEY

```

19 TASK [Scan packages (Windows)] .....
20 skipping: [localhost]
21
22 TASK [Scan services (Windows)] ..... 12:14:29
23 skipping: [localhost]
24
25 TASK [Scan files (Windows)] ..... 12:14:30
26 skipping: [localhost]
27
28 PLAY RECAP ..... 12:14:30
29 localhost : ok=4 changed=0 unreachable=0 failed=0
30


```

Copyright © 2017 Red Hat, Inc.

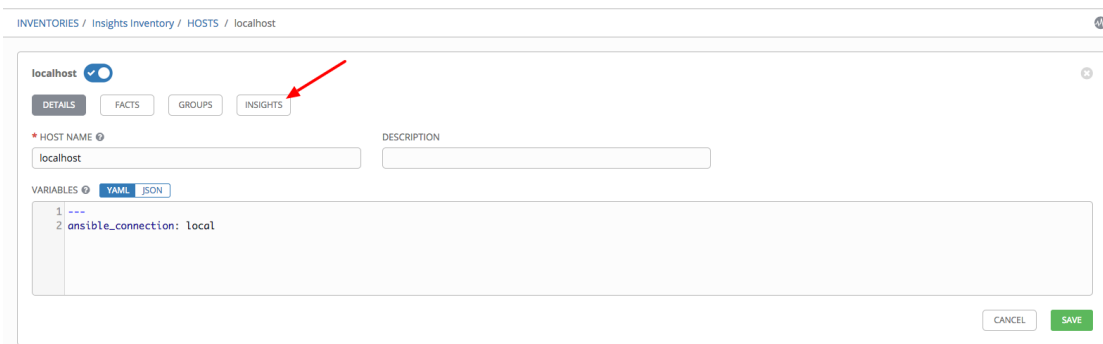
25.6 Remediate Insights Inventory

Remediation of an Insights inventory allows Tower to run Insights playbooks with a single click.



1. Click the Inventories () icon from the left navigation bar to access the Inventories page.
2. In the list of inventories, click to open the details of your Insights inventory.
3. Click the **Hosts** tab to access the Insights hosts that have been loaded from the scan process.
4. Click to open the host that was loaded from Insights.

Notice the Insights tab is now shown on Hosts page. This indicates that Insights and Tower have reconciled the inventories and is now set up for one-click Insights playbook runs.



5. Click **Insights**.

The screen below populates with a list of issues and whether or not the issues can be resolved with a playbook is shown.

INVENTORIES / Insights Inventory / HOSTS / localhost / INSIGHTS

localhost

TOTAL ISSUES 15 HIGH 2 MEDIUM 3 LOW 3 SOLVABLE WITH PLAYBOOK 11 NO REMEDIATION PLAYBOOK AVAILABLE 4

ISSUE: Kernel vulnerable to local privilege escalation via n_hdlc module (CVE-2017-2636) SECURITY

A vulnerability in the Linux kernel allowing local privilege escalation was discovered. The issue was reported as [CVE-2017-2636](https://access.redhat.com/security/cve/CVE-2017-2636).

Demo Insights (17595)
 Unnamed Plan (17705)
 cwang1 (24545)
 CCI demo2 (25155)
 cwang2 (28435)
 cwang3 (28455)
 cwang4 (28475)
 cwang5 (28495)
 cwang6 (28505)
 cwang7 (28515)
 cwang8 (28545)
 cwang9 (28555)
 cwang10 (28575)
 RHEL Demo All Systems (29335)

ISSUE: Kernel vulnerable to privilege escalation via permission bypass (CVE-2016-5195) SECURITY

A flaw was found in the Linux kernel's memory subsystem. An unprivileged local user could use this flaw to write to files they would normally only have read-only access to and thus increase their privileges on the system.

cwang2 (28435)
 cwang3 (28455)
 cwang4 (28475)
 cwang5 (28495)
 cwang6 (28505)
 cwang7 (28515)
 cwang8 (28545)
 cwang9 (28555)
 cwang10 (28575)
 RHEL Demo All Systems (29335)

ISSUE: Kernel vulnerable to man-in-the-middle via payload injection (CVE-2016-5696) SECURITY

A flaw in the Linux kernel's TCP/IP networking subsystem implementation of the [RFC 5961](https://tools.ietf.org/html/rfc5961) challenge ACK rate limiting was found that could allow an attacker located on different subnet to inject or take over a TCP connection between a server and client without needing to use a traditional man-in-the-middle (MITM) attack.

atc2 plan (14115)
 Demo Insights (17595)
 Unnamed Plan (19295)
 cwang1 (24545)
 cwang2 (28435)
 cwang3 (28455)
 cwang4 (28475)
 cwang5 (28495)
 cwang6 (28505)
 cwang7 (28515)
 cwang8 (28545)
 cwang9 (28555)
 cwang10 (28575)
 RHEL Demo All Systems (29335)

ISSUE: Kdump crashkernel reservation failed due to improper configuration of crashkernel parameter STABILITY

The crashkernel configuration has failed to produce a working kdump environment. Configuration changes must be made to enable vmcore capture.

Demo Insights (17595)
 cwang1 (24545)
 cwang2 (28435)
 cwang3 (28455)
 cwang4 (28475)
 cwang5 (28495)
 cwang6 (28505)
 cwang7 (28515)
 cwang8 (28545)
 cwang9 (28555)
 cwang10 (28575)
 RHEL Demo All Systems (29335)

VIEW DATA IN INSIGHTS REMEDIATE INVENTORY CLOSE

6. Scroll down to the bottom of the Insights inventory page, and click the **Remediate Inventory** button to update hosts in the inventory.

Insights Inventory

DETAILS PERMISSIONS GROUPS HOSTS SOURCES COMPLETED JOBS REMEDIATE INVENTORY

SEARCH Q KEY RUN COMMANDS + ADD HOST

HOSTS RELATED GROUPS ACTIONS


localhost

ITEMS 1 - 1

Upon remediation, the New Job Template window opens. Notice the Inventory and Project fields are pre-populated.

Use this new job template to create a job template that pulls Maintenance Plans from Insights.

7. Enter 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:** This field is pre-populated with the Insights inventory you previously created.
- **Project:** This field is pre-populated with 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.

NEW JOB TEMPLATE
✕

DETAILS
PERMISSIONS
COMPLETED JOBS
SCHEDULES
ADD SURVEY

*** NAME**

DESCRIPTION

*** JOB TYPE** ? PROMPT ON LAUNCH

*** INVENTORY** ? PROMPT ON LAUNCH

*** PROJECT** ?

*** PLAYBOOK** ?

CREENTIAL ? PROMPT ON LAUNCH

FORKS ?

LIMIT ? PROMPT ON LAUNCH

*** VERBOSITY** ? PROMPT ON LAUNCH

JOB TAGS ? PROMPT ON LAUNCH

SKIP TAGS ? PROMPT ON LAUNCH

LABELS ?

INSTANCE GROUPS ?

JOB SLICING ?

TIMEOUT ?

SHOW CHANGES ? PROMPT ON LAUNCH

OPTIONS


 ENABLE PRIVILEGE ESCALATION ?
 ENABLE PROVISIONING CALLBACKS ?
 ENABLE WEBHOOK ?
 ENABLE CONCURRENT JOBS ?
 ENABLE FACT CACHE ?

EXTRA VARIABLES ? YAML JSON PROMPT ON LAUNCH

1 ---

LAUNCH
CANCEL
SAVE

8. Click **Save** when done.

9. Click the  icon to launch the job template.

Once complete, the job results display in the Job Details page.

BEST PRACTICES

26.1 Use Source Control

While Tower supports playbooks stored directly on the Tower 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.

26.2 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 Tower does not interactively allow for `vars_prompt` questions. If you must use `vars_prompt`, refer to and make use of the [Surveys](#) functionality of Tower.

Note: Playbooks should not use the `pause` feature of Ansible without a timeout, as Tower does not allow for interactively cancelling a pause. If you must use `pause`, ensure that you set a timeout.

Jobs run in Tower 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.

26.3 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 Tower's support for dynamic inventory (including cloud inventory sources and [custom inventory scripts](#)). 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.

26.4 Variable Management for Inventory

Keeping variable data along with the objects in Tower (see the inventory editor) is encouraged, rather than using `group_vars/` and `host_vars/`. If you use dynamic inventory sources, Tower can sync such variables with the database as long as the **Overwrite Variables** option is not set.

26.5 Autoscaling

Using the “callback” feature to allow newly booting instances to request configuration is very useful for auto-scaling scenarios or provisioning integration.

26.6 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](#).

26.7 Continuous integration / Continuous Deployment

For a Continuous Integration system, such as Jenkins, to spawn an Tower 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 [AWX CLI Ansible Tower documentation](#) for configuration and usage instructions.

26.8 LDAP authentication performance tips

When an LDAP user authenticates in Tower, 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 pushed to Tower. Attributes will only be pushed to Tower the first time the user is created.

SECURITY

The following sections will help you gain an understanding of how Ansible Tower handles and lets you control file system security.

All playbooks are executed via the `awx` file system user. For running jobs, Ansible Tower defaults to offering job isolation via Linux namespacing and `chroots`. This projection ensures jobs can only access playbooks and roles from the Project directory for that job template and common locations such as `/opt`. Playbooks are not able to access roles, playbooks, or data from other Projects by default.

If you need to disable this protection (not recommended), you can edit `/etc/tower/conf.d/custom.py` and set `AWX_PROOT_ENABLED` to `False`.

Note: In this scenario, playbooks have access to the file system and all that implies; therefore, users who have access to edit playbooks **must** be trusted.

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.

27.1 Playbook Access and Information Sharing

By default, Tower’s multi-tenant security prevents playbooks from reading files outside of their project directory. Bubblewrap is used to isolate Tower job processes from the rest of the system, due to its light weight and maintained process isolation system.

By default bubblewrap is enabled, but can be turned off via the Configure Tower - [Jobs Settings](#) screen in the Tower User Interface.

JOBS

ANSIBLE MODULES ALLOWED FOR AD HOC JOBS ? REVERT

x command x shell x yum x apt x apt_key
x apt_repository x apt_rpm x service x group x user
x mount x ping x selinux x setup x win_ping
x win_service x win_updates x win_group x win_user

PATHS TO EXPOSE TO ISOLATED JOBS ? REVERT

ENABLE JOB ISOLATION

Isolates an Ansible job from protected parts of the system to prevent exposing sensitive information.

* ENABLE JOB ISOLATION ?

*JOB EXECUTION PATH ? REVERT

/tmp

ANSIBLE CALLBACK PLUGINS ? REVERT

DEFAULT JOB TIMEOUT ? REVERT

0

Process isolation, when enabled, will be used for the following Job types:

- Job Templates - Launching jobs from regular job templates
- Ad-hoc Commands - Launching ad-hoc commands against one or more hosts in an inventory

By default, process isolation hides the following directories from the above tasks:

- /etc/tower - to prevent exposing Tower configuration
- /etc/ssh - to prevent exposing the system configuration that identifies the private and public key pairs for the host
- /var/lib/awx - with the exception of the current project being used (for regular job templates)
- /var/log
- /tmp (or whatever the system temp directory is) - with the exception of the processes' own temp files

You can customize what to hide or expose when running playbooks, using the Configure Tower screen or the settings file. Refer the next section, *Bubblewrap functionality and variables* for more information.

27.1.1 Bubblewrap functionality and variables

The bubblewrap functionality in Ansible Tower limits which directories on the Tower file system are available for playbooks to see and use during playbook runs. You may find that you need to customize your bubblewrap settings in some cases. To fine tune your usage of bubblewrap, there are certain variables that can be set.

To disable or enable bubblewrap support for running jobs (playbook runs only), ensure you are logged in as the Admin user:



1. Click the Settings () icon from the left navigation bar.
2. Click the **Jobs** tab.
3. In the **Enable Job Isolation**, change the toggle button selection to **OFF** to disable bubblewrap support or select **ON** to enable it.

SETTINGS / JOBS

The screenshot shows the 'JOBS' settings page in Ansible Tower. The 'ENABLE JOB ISOLATION' checkbox is checked and highlighted with a red box. Other settings include:

- ANSIBLE MODULES ALLOWED FOR AD HOC JOBS: 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.
- * JOB EXECUTION PATH: /tmp
- * MAXIMUM SCHEDULED JOBS: 10
- PATHS TO EXPOSE TO ISOLATED JOBS: (empty)
- ANSIBLE CALLBACK PLUGINS: (empty)
- PATHS TO HIDE FROM ISOLATED JOBS: (empty)
- DEFAULT JOB 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

By default, the Tower 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 Configure tower screen, or by updating the following entry in the settings file:

```
AWX_PROOT_BASE_PATH = "/opt/tmp"
```

If there is other information on the system that is sensitive and should be hidden, you can specify those in the Configure Tower screen in the **Paths to Hide From Isolated Jobs** or by updating the following entry in the settings file:

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

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

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

Note: The primary file you may want to add to AWX_PROOT_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:

SETTINGS / JOBS

The screenshot shows the 'JOBS' settings page in Ansible Tower. Three fields are highlighted with red boxes:

- * JOB EXECUTION PATH: /tmp
- PATHS TO EXPOSE TO ISOLATED JOBS: (empty)
- PATHS TO HIDE FROM ISOLATED JOBS: (empty)

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

27.2 Role-Based Access Controls

Role-Based Access Controls (RBAC) are built into Tower and allow Tower 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 Tower 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.

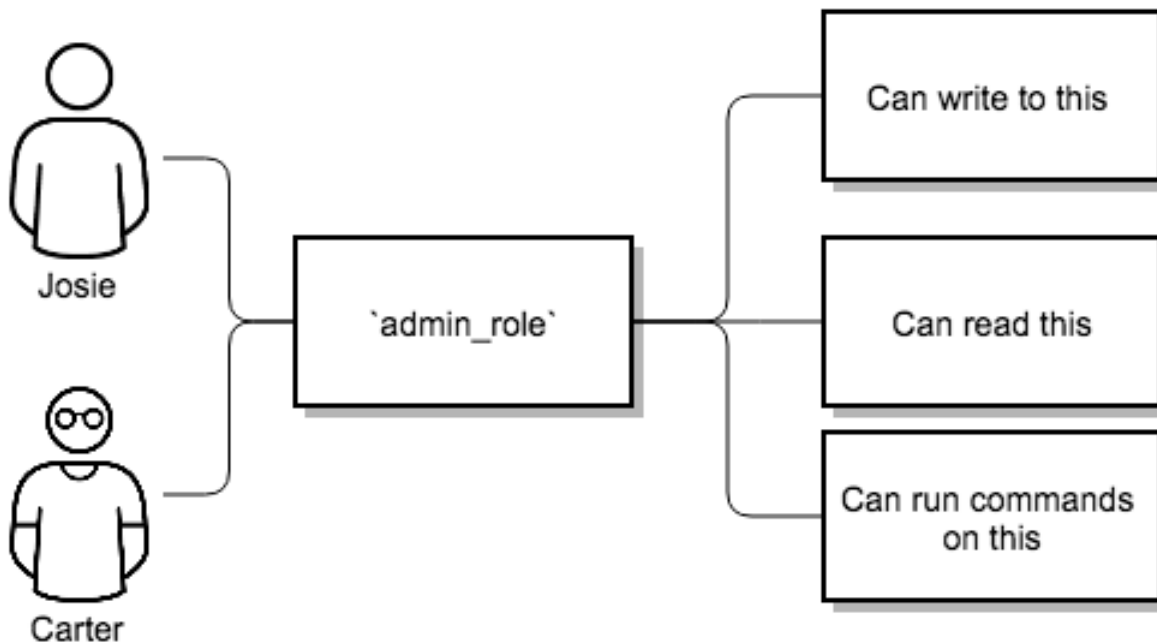
There are a few main concepts that you should become familiar with regarding Tower’s RBAC design—roles, resources, and users. Users can be members of a role, which gives them certain access to any resources associated with that role, or any resources associated with “descendant” roles.

A role is essentially a collection of capabilities. Users are granted access to these capabilities and Tower’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.

27.2.1 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`.

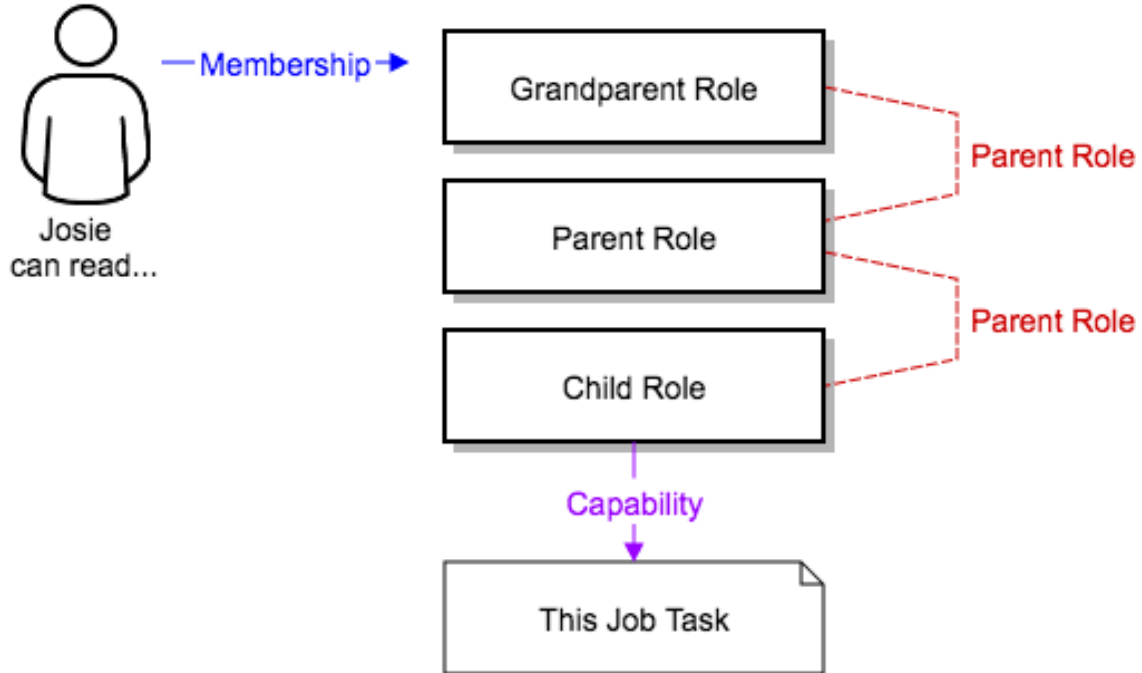


Often, you will have many Roles in a system and you will want some roles to include all of the capabilities of other roles. For example, you may want a System Administrator to have access to everything that an Organization Administrator has access to, who has everything that a Project Administrator has access to, and so on.

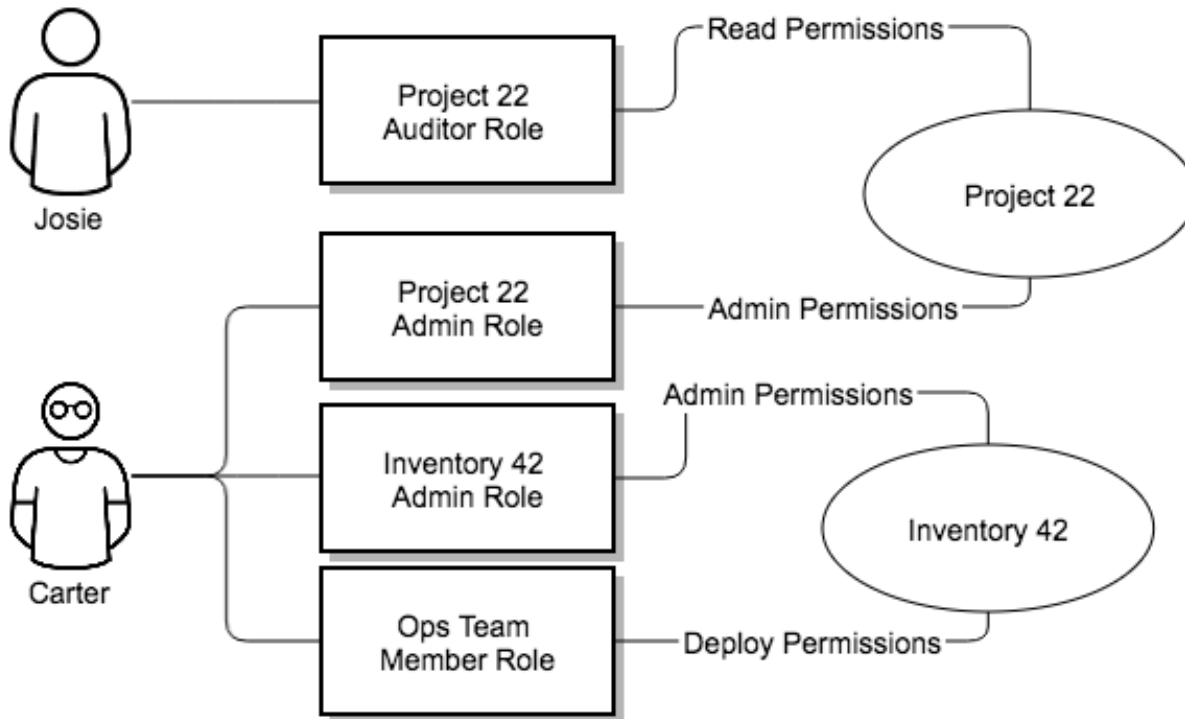
This concept is referred to as the ‘Role Hierarchy’:

- 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 Tower, 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.

27.2.2 Applying RBAC

The following sections cover how to apply Tower’s RBAC system in your environment.

Editing Users

When editing a user, a Tower 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 Tower environment.
- System Auditors implicitly inherit the read-only capability for all objects within the Tower 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 Tower organization.

Users who are organization auditors implicitly inherit the read-only capability for all objects within that Tower 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 Tower’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 (Tower, 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.

27.2.3 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 Tower.

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	User is a member of a defined 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. Ansible Tower 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”

Tower support personnel typically works on ensuring that Tower is available and manages it a way to balance supportability and ease-of-use for users. Often, Ansible Tower 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 Ansible Tower.

Below are some common roles managed by the Tower Organization:

System Role (for Organizations)	Common User Roles	Description
Owner	Team Lead - Technical Lead	<p>This user has the ability to control access for other users in their organization.</p> <p>They can add/remove and grant users specific access to projects, inventories, and job templates.</p> <p>This user also has the ability to create/remove/modify any aspect of an organization's projects, templates, inventories, teams, and credentials.</p>
Auditor	Security Engineer - Project Manager	<p>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.</p> <p>This might also be a good role for a service account who manages or ships job data from Ansible Tower to some other data collector.</p>
Member - Team	All other users	<p>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.</p>
Member - Team "Owner"	Power users - Lead Developer	<p>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.</p>
27.2. Role-Based Access Controls		298
Member -	Developers -	This will be the most common and

27.3 Function of roles: editing and creating

A new organization “resource roles” functionality was introduced in Ansible Tower 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).

27.3.1 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

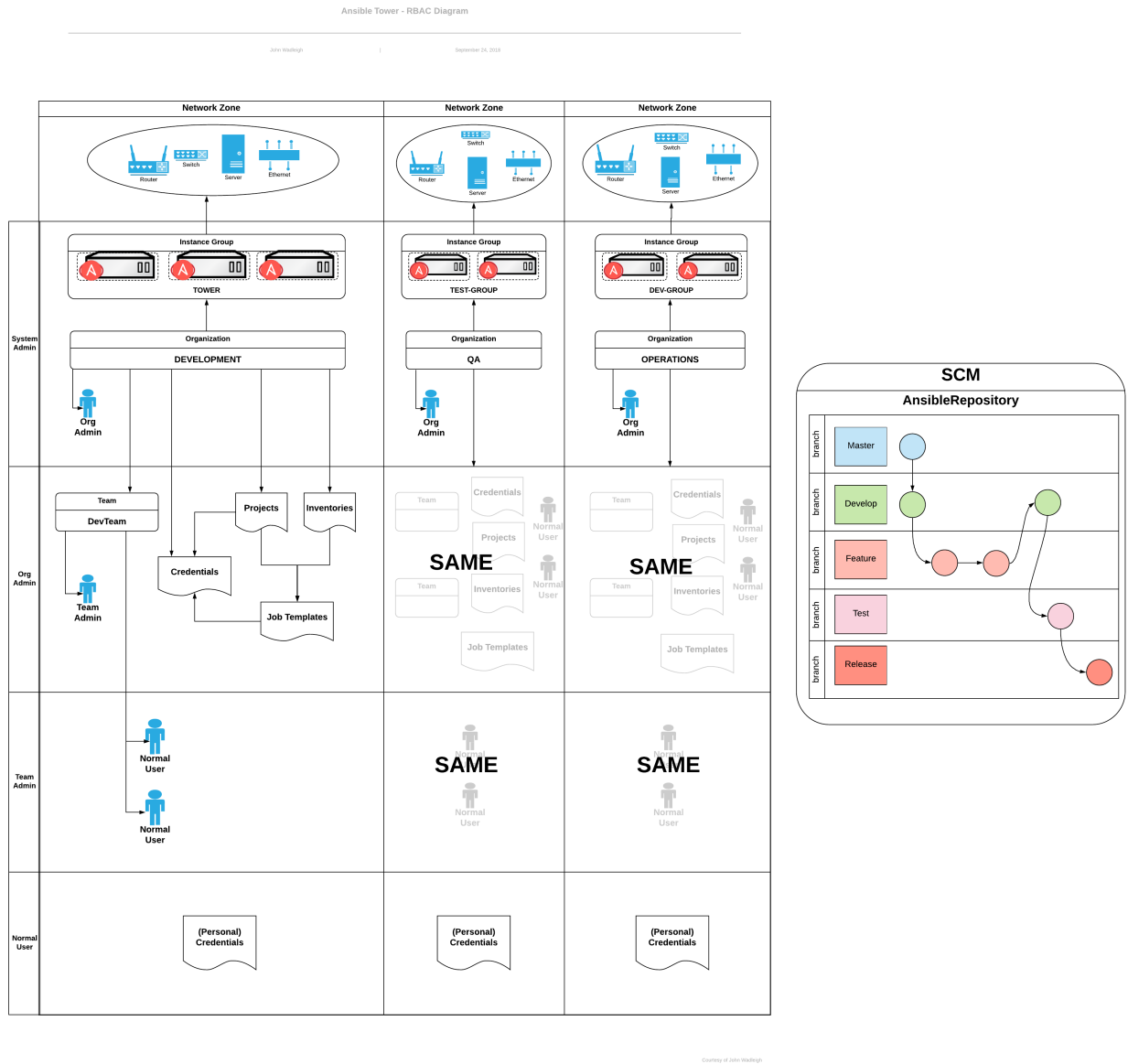
Workflow Admin

- 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
TWENTYEIGHT

INDEX

- genindex

COPYRIGHT © RED HAT, INC.

Ansible, Ansible Tower, Red Hat, and Red Hat Enterprise Linux are trademarks of Red Hat, Inc., registered in the United States and other countries.

If you distribute this document, or a modified version of it, you must provide attribution to Red Hat, Inc. and provide a link to the original version.

Third Party Rights

Ubuntu and Canonical are registered trademarks of Canonical Ltd.

The CentOS Project is copyright protected. The CentOS Marks are trademarks of Red Hat, Inc. (“Red Hat”).

Microsoft, Windows, Windows Azure, and Internet Explore are trademarks of Microsoft, Inc.

VMware is a registered trademark or trademark of VMware, Inc.

Amazon Web Services”, “AWS”, “Amazon EC2”, and “EC2”, are trademarks of Amazon Web Services, Inc. or its affiliates.

OpenStack™ and OpenStack logo are trademarks of OpenStack, LLC.

Chrome™ and Google Compute Engine™ service registered trademarks of Google Inc.

Safari® is a registered trademark of Apple, Inc.

Firefox® is a registered trademark of the Mozilla Foundation.

All other trademarks are the property of their respective owners.

Symbols

|atl|
 credential types, 64

A

activity streams, 20
 ad hoc commands, 145
 inventories, 145
 add new
 inventories, 121
 projects, 97
 smart inventories, 121
 adding new
 applications, 91
 credentials, 61
 adding tokens
 applications, 92
 admin menu, 26
 Amazon Web Services
 credential types, 63
 inventories, 137
 Ansible collections, 112
 Ansible Galaxy, 109
 Ansible Galaxy integration
 features, 3
 Ansible Tower
 inventories, 142
 API bearer token
 credential types, 71
 API considerations
 credential types, 77
 API endpoints
 notifications, 270
 applications
 adding new, 91
 adding tokens, 92
 authentication, 90
 create, 91
 getting started, 90
 tokens, 90, 92
 attaching
 subscription, 9

authentication, 90
 applications, 90
 features, 5
 automation
 features, 2
 Automation Hub, 7
 content provider, 7
 credential types, 64
 autoscaling
 best practices, 288
 autoscaling flexibility
 features, 3
 AWS
 cloud credentials, 175
 Azure
 KMS, credential, 89

B

backup and restore
 features, 3
 best practices, 287
 autoscaling, 288
 deployment, continuous, 288
 dynamic inventory sources, 287
 file and directory structure, 287
 host counts, larger, 288
 integration, continuous, 288
 ldap, 288
 source control, 287
 variable inventory management, 288
 bubblewrap
 functionality, 290
 playbooks, 289
 troubleshooting, 290
 variables, 290

C

callbacks
 extra variables, 177
 capacity
 jobs, 231
 check

- job types, 148
 - cloud credentials
 - AWS, 175
 - Google, 175
 - job templates, 173
 - MS Azure, 175
 - OpenStack, 174
 - VMware, 175
 - cloud flexibility
 - features, 3
 - clustering
 - features, 5
 - collections support, 112
 - components
 - licenses, 10
 - configure Tower
 - settings menu, 27
 - consume
 - subscription, 9
 - container support
 - features, 5
 - content provider
 - Automation Hub, 7
 - create
 - applications, 91
 - create template
 - notifications, 255
 - creating new
 - credential types, 78
 - credential
 - Azure KMS, 89
 - CyberArk AIM, 86
 - CyberArk Conjur, 87
 - HashiCorp KV, 87
 - HashiCorp SSH Secrets Engine, 88
 - MS Azure KMS, 89
 - plugins, 83
 - secret management, 83
 - credential management
 - features, 7
 - credential plugins
 - features, 7
 - credential types, 62, 76
 - |at|, 64
 - Amazon Web Services, 63
 - API bearer token, 71
 - API considerations, 77
 - Automation Hub, 64
 - creating new, 78
 - Galaxy, 64
 - GitHub PAT, 64
 - GitLab PAT, 65
 - Google Compute Engine, 65
 - insights, 66
 - Kubernetes, 71
 - machine, 67
 - Microsoft Azure Resource Manager, 69
 - network, 70
 - OpenShift, 71
 - OpenStack, 72
 - oVirt, 73
 - Red Hat Satellite, 72
 - Red Hat Virtualization, 73
 - rhv, 73
 - source control, 73
 - Vault, 74
 - VMware, 75
 - credentials, 59, 83
 - adding new, 61
 - getting started, 59, 77
 - how they work, 59
 - Insights, 275
 - types, 62
 - custom
 - fact scan job, 171
 - custom environment
 - features, 5
 - custom fact scans
 - playbook, 171
 - system tracking, 171
 - custom script
 - inventories, 143
 - CyberArk AIM, 86
 - credential, 86
 - CyberArk Conjur, 87
 - credential, 87
- ## D
- dashboard, 22
 - host count, 22
 - job status, 22
 - jobs tab, 22
 - main menu, 20
 - schedule status, 22
 - DEB files
 - licenses, 10
 - deployment, continuous
 - best practices, 288
 - distributed
 - job types, 180
 - dynamic inventory sources
 - best practices, 287
- ## E
- Email
 - notifications types, 256
 - environment, FIPS
 - features, 6

- evaluation, 8
 - extra variables
 - callbacks, 177
 - provisioning callbacks, 177
 - surveys, 178, 186, 214
 - extra_vars, 178, 214
- ## F
- fact cache
 - features, 4
 - fact caching
 - playbook, 172
 - fact scan job
 - custom, 171
 - playbook, 169
 - fact scan playbook
 - system tracking, 169
 - facts
 - scan job templates, 172
 - features, 7
 - Ansible Galaxy integration, 3
 - authentication, 5
 - automation, 2
 - autoscaling flexibility, 3
 - backup and restore, 3
 - cloud flexibility, 3
 - clustering, 5
 - container support, 5
 - credential management, 7
 - credential plugins, 7
 - custom environment, 5
 - environment, FIPS, 6
 - fact cache, 4
 - instance groups, 5
 - inventory plugins, 6
 - inventory sources, Red Hat
 - Satellite 6, 4
 - jobs, distribution, 6
 - jobs, slicing, 6
 - limiting, hosts, 6
 - notifications, 4
 - OAuth 2 token, 5
 - OpenStack inventory support, 3
 - overview, 2
 - playbooks, Red Hat Insights, 5
 - real-time playbook, 2
 - remote command execution, 4
 - RESTful API, 3
 - role-based access control, 2
 - run-time job customization, 4
 - secret management system, 7
 - system tracking, 4
 - UI, 5
 - user interface, 5
 - venv, 5
 - workflows, approval, 5
 - workflows, convergence nodes, 5
 - workflows, inventory overrides, 5
 - workflows, nesting, 5
 - workflows, pause, 5
 - file and directory structure
 - best practices, 287
 - forks
 - jobs, 231
 - functionality
 - bubblewrap, 290
- ## G
- Galaxy
 - credential types, 64
 - Galaxy support, 109
 - getting started
 - applications, 90
 - credentials, 59, 77
 - Git
 - source control, 98
 - git refspec
 - templates, 238
 - GitHub
 - webhooks, 239
 - GitHub PAT
 - credential types, 64
 - GitLab
 - webhooks, 239
 - GitLab PAT
 - credential types, 65
 - Google
 - cloud credentials, 175
 - Google Compute Engine
 - credential types, 65
 - inventories, 138
 - Grafana
 - notifications types, 256
 - groups
 - notifications, 254
- ## H
- HashiCorp KV
 - credential, 87
 - HashiCorp Secret Lookup, 87
 - HashiCorp SSH Secrets Engine, 88
 - credential, 88
 - hierarchy
 - notifications, 254
 - Hipchat
 - notifications types, 256
 - host count
 - dashboard, 22

- host counts, larger
 - best practices, 288
- hostname configuration
 - notifications, 270
- how they work
 - credentials, 59
- I
- Insights
 - credentials, 275
 - inventory, 278, 283
 - project, 277
 - projects, 275
 - source control, 100
- insights
 - credential types, 66
- installation bundle
 - licenses, 10
- instance groups, 216
 - features, 5
- integration, continuous
 - best practices, 288
- inventories, 117
 - ad hoc commands, 145
 - add new, 121
 - Amazon Web Services, 137
 - Ansible Tower, 142
 - custom script, 143
 - Google Compute Engine, 138
 - groups, 127
 - groups; add new, 127
 - Microsoft Azure Resource Manager, 139
 - OpenStack, 141
 - plugins, 120
 - project-sourced, 136
 - Red Hat Satellite 6, 140
 - Red Hat Virtualization, 142
 - scan job templates, 168
 - smart, 119
 - VMware vCenter, 139
- inventory
 - Insights, 278, 283
- inventory plugins
 - features, 6
- inventory source
 - scheduling, 272
- inventory sources
 - notifications, 254
- inventory sources, Red Hat Satellite 6
 - features, 4
- inventory sync
 - job results, 223
- IRC
 - notifications types, 256
- J
- job branch
 - overriding, 234
- job results, 222
 - inventory sync, 223
- job slice, 180
- job splitting, 180
- job status
 - dashboard, 22
- job templates, 148
 - cloud credentials, 173
 - job variables, 178
 - jobs, launching, 165
 - provisioning callbacks, 176
 - relaunch, 179
 - scheduling, 161, 162, 272
 - survey creation, 163
 - survey extra variables, 178
 - survey optional questions, 164
 - surveys, 162
- job templates, hierarchy, 178
- job templates, overview, 178
- job types
 - check, 148
 - distributed, 180
 - run, 148
 - scan, 148
 - slice, 180
 - splitting, 180
- job variables
 - job templates, 178
 - workflow templates, 214
- jobs, 221
 - capacity, 231
 - event summary, 228
 - events summary, 227
 - forks, 231
 - host events, 230
 - host status bar, 228
 - host summary, 228
 - job summary, 227
 - notifications, 254
 - results, 222
 - views, 22
- jobs results
 - playbook run, 226
 - SCM, 225
- jobs tab
 - dashboard, 22
- jobs, distribution
 - features, 6
- jobs, launching

- job templates, 165
- workflow templates, 212
- jobs, slicing
 - features, 6

K

- KMS
 - credential Azure, 89
- Kubernetes
 - credential types, 71

L

- ldap
 - best practices, 288
- license, 7, 8
 - nodes, 9
 - trial, 8
 - troubleshooting, 19
 - types, 8
- license features, 7
- license, add manually, 19
- license, viewing, 27
- licenses
 - components, 10
 - DEB files, 10
 - installation bundle, 10
 - RPM files, 10
- limiting, hosts
 - features, 6
- logging in, 11

M

- machine
 - credential types, 67
- main menu
 - dashboard, 20
- manifest
 - subscriptions, 16
- Mattermost
 - notifications types, 256
- Mercurial
 - source control, 98
- Microsoft Azure Resource Manager
 - credential types, 69
 - inventories, 139
- MS Azure
 - cloud credentials, 175
- MS Azure KMS, 89
 - credential, 89
- my view, 22

N

- network
 - credential types, 70

- notifications
 - API endpoints, 270
 - create template, 255
 - features, 4
 - groups, 254
 - hierarchy, 254
 - hostname configuration, 270
 - inventory sources, 254
 - jobs, 254
 - organizations, 42
 - resetting the TOWER_URL_BASE, 270
 - template, 254, 255
 - template workflow, 255
 - troubleshooting TOWER_URL_BASE, 270
 - types, 256
 - types Email, 256
 - types Grafana, 256
 - types Hipchat, 256
 - types IRC, 256
 - types Mattermost, 256
 - types pagerduty, 256
 - types Rocket.Chat, 256
 - types Slack, 256
 - types Twilio, 256
 - types Webhook, 256

O

- OAuth 2 token
 - features, 5
- obtain
 - subscriptions manifest, 16
- OpenShift
 - credential types, 71
- OpenStack
 - cloud credentials, 174
 - credential types, 72
 - inventories, 141
- OpenStack inventory support
 - features, 3
- ordering
 - sorting, 30
- organization
 - summary, 43
- organizations, 32
 - notifications, 42
 - permissions, 36
 - users, 35, 47
- overriding
 - job branch, 234
- overview
 - features, 2
- oVirt
 - credential types, 73

P

- pagerduty
 - notifications types, 256
- payload
 - webhooks, 239
- permissions
 - organizations, 36
 - projects, 102
 - teams, 56
 - users, 48
- playbook
 - custom fact scans, 171
 - fact caching, 172
 - fact scan job, 169
 - scan job, 168
- playbook run
 - jobs results, 226
- playbooks
 - bubblewrap, 289
 - manage manually, 97
 - process isolation, 289
 - projects, 97, 98, 100
 - PRoot settings, 289
 - sharing access, 289
 - sharing content, 289
 - source control, 98, 100
- playbooks, Red Hat Insights
 - features, 5
- plugins
 - credential, 83
 - inventories, 120
- process isolation
 - playbooks, 289
- project
 - Insights, 277
 - Scan, 279
- project-sourced
 - inventories, 136
- projects, 95
 - add new, 97
 - Insights, 275
 - permissions, 102
 - playbooks, 97, 98, 100
 - scheduling, 108, 272
 - source control update, 101
- PRoot settings
 - playbooks, 289
- provisioning callbacks
 - extra variables, 177
 - job templates, 176

R

- RBAC
 - security, 292

- real-time playbook
 - features, 2
- Red Hat Satellite
 - credential types, 72
- Red Hat Satellite 6
 - inventories, 140
- Red Hat Virtualization
 - credential types, 73
 - inventories, 142
- relaunch
 - job templates, 179
- remote archive
 - source control, 100
- remote command execution
 - features, 4
- resetting the TOWER_URL_BASE
 - notifications, 270
- RESTful API
 - features, 3
- rhv
 - credential types, 73
- Rocket.Chat
 - notifications types, 256
- role-based access control
 - features, 2
- role-based access controls, 292
- RPM files
 - licenses, 10
- run
 - job types, 148
- run-time job customization
 - features, 4

S

- Scan
 - project, 279
- scan
 - job types, 148
- scan job
 - playbook, 168
- scan job templates
 - facts, 172
 - inventories, 168
- schedule
 - views, 22
- schedule status
 - dashboard, 22
- scheduling
 - add new, 162, 196
 - inventory source, 272
 - job templates, 161, 162, 272
 - projects, 108, 272
 - workflow template, 196
 - workflow templates, 196, 272

- SCM
 - jobs results, 225
 - types, 98
- SCM types, 98
- searching, 28
- secret management
 - credential, 83
- secret management system
 - features, 7
- security, 289
 - RBAC, 292
- settings menu
 - configure Tower, 27
 - view license, 27
- sharing access
 - playbooks, 289
- sharing content
 - playbooks, 289
- Slack
 - notifications types, 256
- slice
 - job types, 180
- smart
 - inventories, 119
- smart inventories
 - add new, 121
- sorting
 - ordering, 30
- source control
 - best practices, 287
 - credential types, 73
 - Git, 98
 - Insights, 100
 - Mercurial, 98
 - remote archive, 100
 - Subversion, 98
- source control update
 - projects, 101
- splitting
 - job types, 180
- subscription
 - attaching, 9
 - consume, 9
- subscriptions
 - manifest, 16
- subscriptions manifest
 - obtain, 16
- subscriptions, import, 12
- Subversion
 - source control, 98
- summary
 - organization, 43
- support, 7, 8
- survey extra variables

- job templates, 178
 - workflow templates, 214
 - workflows, 186
- surveys
 - creation, 163, 196
 - extra variables, 178, 186, 214
 - job templates, 162
 - optional questions, 164, 198
 - workflow templates, 196
- system tracking
 - custom fact scans, 171
 - fact scan playbook, 169
 - features, 4
 - scan job, 148

T

- teams, 53
 - permissions, 56
 - users, 47, 54
- template
 - notifications, 254, 255
- template workflow
 - notifications, 255
- templates
 - git refs, 238
- token authentication, 90
- tokens
 - applications, 90, 92
- Tower admin menu, 26
- Tower settings menu, 27
- trial, 8
- troubleshooting
 - bubblewrap, 290
 - license, 19
- troubleshooting TOWER_URL_BASE
 - notifications, 270
- Twilio
 - notifications types, 256
- types
 - Email, notifications, 256
 - Grafana, notifications, 256
 - Hipchat, notifications, 256
 - IRC, notifications, 256
 - Mattermost, notifications, 256
 - notifications, 256
 - pagerduty, notifications, 256
 - Rocket.Chat, notifications, 256
 - SCM, 98
 - Slack, notifications, 256
 - Twilio, notifications, 256
 - Webhook, notifications, 256

U

- UI

- features, 5
 - updates, 8
 - user interface
 - features, 5
 - users, 44
 - organizations, 35, 47
 - permissions, 48
 - teams, 47, 54
- V**
- variable inventory management
 - best practices, 288
- variable precedence, 178, 214
- variables
 - bubblewrap, 290
- Vault
 - credential types, 74
- venv
 - features, 5
- view license
 - settings menu, 27
- views
 - jobs, 22
 - schedule, 22
- visualizer
 - workflow, 198
- VMware
 - cloud credentials, 175
 - credential types, 75
- VMware vCenter
 - inventories, 139
- W**
- Webhook
 - notifications types, 256
- webhooks, 239
 - GitHub, 239
 - GitLab, 239
 - payload, 239
- workflow
 - visualizer, 198
- workflow job templates, 188
- workflow template
 - scheduling, 196
- workflow templates
 - job variables, 214
 - jobs, launching, 212
 - scheduling, 196, 272
 - survey creation, 196
 - survey extra variables, 214
 - survey optional questions, 198
 - surveys, 196
 - workflow visualizer, 198
- workflow templates, hierarchy, 214
 - workflow templates, overview, 214
 - workflow visualizer
 - workflow templates, 198
 - workflows, 183
 - survey extra variables, 186
 - workflows, approval
 - features, 5
 - workflows, convergence nodes
 - features, 5
 - workflows, inventory overrides
 - features, 5
 - workflows, nesting
 - features, 5
 - workflows, pause
 - features, 5