
Ansible Tower User Guide

Release Ansible Tower 3.3.6

Red Hat, Inc.

Aug 14, 2019

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	3
1.10	System Tracking	4
1.11	Integrated Notifications	4
1.12	Satellite and CloudForms Integration	4
1.13	Run-time Job Customization	4
1.14	Red Hat Insights Integration	4
1.15	Enhanced Tower User Interface	4
1.16	Custom Virtual Environments	4
1.17	Authentication Enhancements	5
1.18	Cluster Management	5
1.19	Container Platform Support	5
2	Tower Licensing, Updates, and Support	6
2.1	Support	6
2.2	Trial / Evaluation	6
2.3	Subscription Types	6
2.4	Node Counting in Licenses	7
2.5	Tower Component Licenses	7
3	Logging In	8
4	Import a License	9
4.1	Adding a Tower License Manually	10
5	The Tower User Interface	12
5.1	Activity Streams	12
5.2	Views	14
5.3	Resources and Access	17
5.4	Tower Administration Menu	18
6	Search	21
6.1	Searching Tips	21

6.2	Sort	23
7	Organizations	25
7.1	Creating a New Organization	26
8	Users	31
8.1	Create a User	31
8.2	User Types - Quick View	33
8.3	Users - Organizations	33
8.4	Users - Teams	34
8.5	Users - Permissions	34
8.6	Users - Tokens	37
9	Teams	39
9.1	Create a Team	39
10	Credentials	45
10.1	Understanding How Credentials Work	45
10.2	Getting Started with Credentials	46
10.3	Add a New Credential	47
10.4	Credential Types	48
11	Custom Credential Types	61
11.1	Backwards-Compatible API Considerations	61
11.2	Getting Started with Credential Types	62
11.3	Create a New Credential Type	63
12	Applications	67
12.1	Getting Started with Applications	67
12.2	Create a new application	68
13	Projects	71
13.1	Add a new project	72
13.2	Work with Permissions	75
13.3	Work with Notifications	79
13.4	Work with Job Templates	80
13.5	Work with Schedules	80
14	Inventories	84
14.1	Smart Inventories	86
14.2	Add a new inventory	87
14.3	Running Ad Hoc Commands	108
15	Job Templates	111
15.1	Create a Job Template	112
15.2	Add Permissions	117
15.3	Work with Notifications	120
15.4	View Completed Jobs	121
15.5	Scheduling	121
15.6	Surveys	124
15.7	Launch a Job Template	126
15.8	Copy a Job Template	129
15.9	Scan Job Templates	130
15.10	Fact Caching	133
15.11	Utilizing Cloud Credentials	135

15.12 Provisioning Callbacks	137
15.13 Extra Variables	139
16 Workflow Job Templates	142
16.1 Create a Workflow Template	142
16.2 Work with Permissions	144
16.3 Work with Notifications	145
16.4 Work with Schedules	146
16.5 Surveys	148
16.6 Workflow Visualizer	150
16.7 Launch a Workflow Job Template	155
16.8 Copy a Workflow Job Template	156
16.9 Extra Variables	157
17 Instance Groups	159
17.1 Create an instance group	159
18 Jobs	163
18.1 Job Details - Inventory Sync	165
18.2 Job Details - SCM	166
18.3 Job Details - Playbook Run	168
18.4 Ansible Tower Capacity Determination and Job Impact	172
19 Notifications	176
19.1 Notifier Hierarchy	176
19.2 Workflow	176
19.3 Create a Notification Template	177
19.4 Notification Types	177
19.5 Configuring the towerhost hostname	184
20 Workflows	185
20.1 Extra Variables	186
20.2 Workflow States	187
20.3 Role-Based Access Controls	187
21 Setting up an Insights Project	189
21.1 Create Insights Credential	189
21.2 Create an Insights Project	191
21.3 Create Insights Inventory	192
21.4 Create a Scan Project	193
21.5 Create a Scan Job Template	194
21.6 Remediate Insights Inventory	196
22 Best Practices	200
22.1 Use Source Control	200
22.2 Ansible file and directory structure	200
22.3 Use Dynamic Inventory Sources	200
22.4 Variable Management for Inventory	201
22.5 Autoscaling	201
22.6 Larger Host Counts	201
22.7 Continuous integration / Continuous Deployment	201
23 Security	202
23.1 Playbook Access and Information Sharing	202
23.2 Role-Based Access Controls	204

23.3 Function of roles: editing and creating	212
24 Index	215
25 Copyright © 2019 Red Hat, Inc.	216
Index	217

Thank you for your interest in Red Hat Ansible Tower. Ansible Tower is a commercial offering that 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 3.3.6.

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 3.3.6; July 10, 2019; <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.

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. Beginning with version 2.2.0, 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

System tracking (historical facts) feature was deprecated starting with Ansible Tower 3.2. However, you can collect facts by using the fact caching feature. Refer to *Fact Caching* for more detail.

1.11 Integrated Notifications

Starting with version 3.0, 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 success and job failure. The following notification sources are supported: - Slack - E-mail - SMS (via Twilio) - HipChat - Pagerduty - IRC - Webhooks (post to an arbitrary webhook, for integration into other tools)

1.12 Satellite and CloudForms Integration

Ansible Tower 3.0 also adds dynamic inventory sources for Red Hat Satellite 6 and Red Hat CloudForms.

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

1.15 Enhanced Tower User Interface

In Ansible Tower 3.3, the layout of the user interface was reorganized to improve navigational elements. With more information displayed at-a-glance, it is more intuitive to find and use the automation you need.

1.16 Custom Virtual Environments

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

1.17 Authentication Enhancements

Ansible Tower 3.3 enhanced LDAP and SAML support and introduced 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.

TOWER LICENSING, UPDATES, AND SUPPORT

Red Hat Ansible Tower (“**Ansible Tower**”) is a software product provided as part of an annual 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>

2.1 Support

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

If you or your company has purchased a subscription for Ansible Automation, you can contact the support team at <https://access.redhat.com>. To better understand the levels of support which match your Ansible Tower Subscription, refer to *Subscription Types*. For details of what is covered under an Ansible Automation 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 is provided at various levels of support and number of machines as an annual Subscription.

- **Standard (F.K.A. “Enterprise: 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>

- **Enterprise (F.K.A. “Enterprise: 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.

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

2.4 Node Counting in Licenses

The Tower license defines the number of Managed Nodes that can be managed by Ansible Tower. A typical license will say ‘License Count: 500’, which sets the maximum number of Managed Nodes at 500.

Ansible Tower counts Managed Nodes by the number of nodes in inventory. If more Managed Nodes are in the Ansible Tower inventory than are supported by the license, you will be unable to start any Jobs in Ansible Tower. If a dynamic inventory sync causes Ansible Tower to exceed the Managed Node count specified in the license, the dynamic inventory sync will fail.

If you have multiple hosts in inventory that have the same name, such as “webserver1”, they will be counted for licensing purposes as a single node. Note that this differs from the ‘Hosts’ count in Tower’s dashboard, which counts hosts in separate inventories separately. Note that this behavior is case-sensitive; “webserver1” and “WebServer1” will be treated as different nodes.

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

2.5 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 Ansible Tower login page. At the top left is the Ansible Tower logo, which consists of a red circle with a white 'A' inside, followed by the text 'ANSIBLE TOWER by Red Hat'. Below the logo is the text 'Welcome to Ansible Tower! Please sign in.' There are two input fields: one for 'USERNAME' containing the text 'admin', and one for 'PASSWORD' containing a series of dots. A green 'SIGN IN' button is located at the bottom right of the form area.

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 link from the Settings () Menu.

IMPORT A LICENSE

Tower requires a valid license to run. If you did not receive a license from Ansible directly or via email, or have issues with the license you received, refer to <http://www.ansible.com/license> for free and paid license options (including free trial licenses) or contact Ansible via the Red Hat Customer portal at <https://access.redhat.com/>.

Note: To successfully add your license, you must be logged on as the Superuser. Otherwise, the operation will fail.

The screenshot shows the 'CONFIGURE TOWER' interface with the 'LICENSE' tab selected. The 'LICENSE MANAGEMENT' section contains the following elements:


- Instructions: "Choose your license file, agree to the End User License Agreement, and click submit."
- * LICENSE FILE**: A 'BROWSE' button and the text "No file selected."
- * END USER LICENSE AGREEMENT**: A scrollable text area containing the text: "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". Below the text is a checkbox labeled "I agree to the End User License Agreement".
- A 'SUBMIT' button is located at the bottom right of the form.

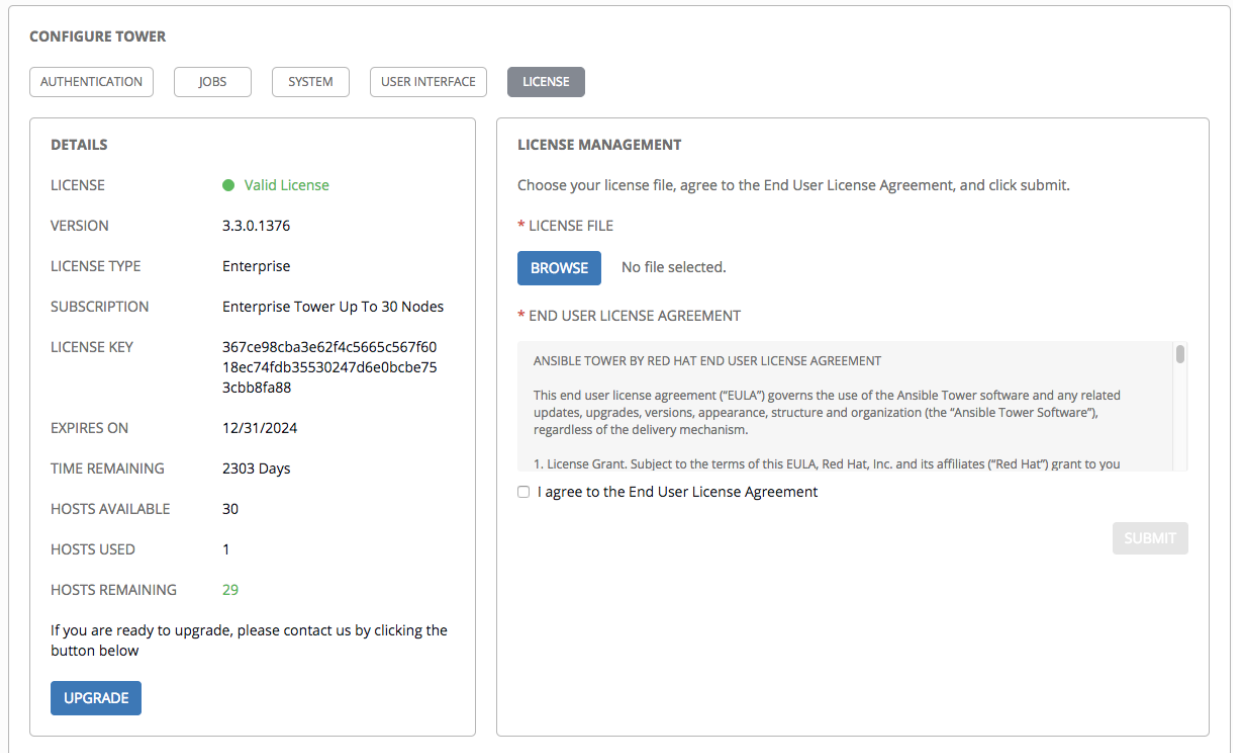
To add your license:

1. Save your license (or save the license contents to a text file locally, if needed).
2. Click the **Browse** button and navigate to the location where the license file is saved to upload it. The uploaded license may be a plain text file or a JSON file, and must include properly formatted JSON code.
3. Once uploaded, check to agree to the End User License Agreement and click **Submit**.

Once your license has been accepted, Tower navigates you to the main Ansible interface for the Dashboard (which you can access by clicking on the Ansible Tower logo at the top left of the screen as well).

For later reference, you can view this license from the **License** tab of the Configure Tower screen, accessible through

the Settings () icon from the left navigation bar.



4.1 Adding a Tower License Manually

If you are in a situation where uploading a file is not allowed due to a locked down environment, you can add the Ansible Tower license by hand using Tower’s API.

Note: To successfully add your license, you must be logged on as the Superuser. Otherwise, the operation will fail. Use only the procedure described here for applying a license via the API. Do not put the license in a file, and manually placing it in the license directory of your Ansible Tower install. The ability to do so has been deprecated in version 3.1.0.

To add the license file manually:

1. In Tower’s REST API, at the `/api/v2/config/` endpoint, scroll down to the POST text entry box.
2. Add your valid license, the one you received directly from Ansible, to the POST box using the following as an example:

```

{"eula_accepted" : "true",
 "subscription_name": "Enterprise Tower up to 100000 Nodes",
 "features": {},
 "instance_count": 100000,
 "trial": false,

```

```
"contact_email": "maddux@hotmail.com",
"company_name": "Dr. Maddux Golden",
"license_type": "enterprise",
"contact_name": "Dr. Maddux Golden",
"license_date": 0000000000,
"license_key":
↪ "xxx111xx111xxx1x1x1x1x11x1xxxx1x1xx1x1xx1x1x1xxx111xx1x1xx1x"
```

3. When finished, click the **POST** button and review your license.

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 Q KEY All Activity ▼

TIME	INITIATED BY	EVENT	ACTIONS
6/29/2016 10:20:41 AM	admin	created job Demo Job Template	🔍
6/29/2016 9:12:08 AM	admin	updated team Production Operations	🔍
6/29/2016 9:11:15 AM	admin	associated Production Operations member_role to jdodge	🔍
6/29/2016 9:11:15 AM	admin	associated Production Operations admin_role to gdodge	🔍
6/29/2016 9:10:54 AM	admin	created team Production Operations	🔍
6/29/2016 9:10:12 AM	admin	associated cdodge member_role to Honey Dog, Inc.	🔍
6/29/2016 9:10:12 AM	admin	updated user cdodge	🔍
6/29/2016 9:10:12 AM	admin	created user cdodge	🔍
6/29/2016 9:09:39 AM	admin	associated jdodge member_role to Honey Dog, Inc.	🔍
6/29/2016 9:09:39 AM	admin	updated user jdodge	🔍
6/29/2016 9:09:39 AM	admin	created user jdodge	🔍
6/29/2016 9:09:12 AM	admin	associated gdodge member_role to Honey Dog, Inc.	🔍
6/29/2016 9:09:12 AM	admin	updated user gdodge	🔍
6/29/2016 9:09:12 AM	admin	created user gdodge	🔍
6/29/2016 9:08:40 AM	admin	associated admin member_role to Honey Dog, Inc.	🔍
6/29/2016 9:08:25 AM	admin	created organization Honey Dog, Inc.	🔍
6/29/2016 9:08:25 AM	admin	associated system_auditor to Honey Dog, Inc.	🔍
6/29/2016 9:08:25 AM	admin	associated system_administrator to Honey Dog, Inc.	🔍
6/29/2016 9:08:02 AM	admin	Event summary not available	🔍
6/29/2016 9:08:02 AM	admin	Event summary not available	🔍

< 1 2 3 > PAGE 1 OF 3 ITEMS 1-20 OF 52

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. Clicking on the Examine (🔍) button shows the event log for the change.

ACTIVITY STREAM

SEARCH All Activity ▼

TIME	INITIATED BY	EVENT	ACTIONS
2/27/2017 12:39:31 PM	system	created host localhost	🔍
2/27/2017 12:39:31 PM	system		🔍
2/27/2017 12:39:31 PM	system		🔍
2/27/2017 12:39:30 PM	system		🔍
2/27/2017 12:39:30 PM	system		🔍
2/27/2017 12:39:30 PM	system		🔍
2/27/2017 12:39:30 PM	system		🔍
2/27/2017 12:39:30 PM	system	Event summary not available	🔍
2/27/2017 12:39:30 PM	system	created project Demo Project	🔍

EVENT 8 ✖

INITIATED BY system on 2/27/2017 12:39:31 PM

ACTION created host localhost

CHANGES

```
{
  "name": "localhost",
  "variables": "ansible_connection: local",
  "enabled": true,
  "instance_id": "",
  "inventory": "Demo Inventory-1",
  "id": 1,
  "description": ""
}
```

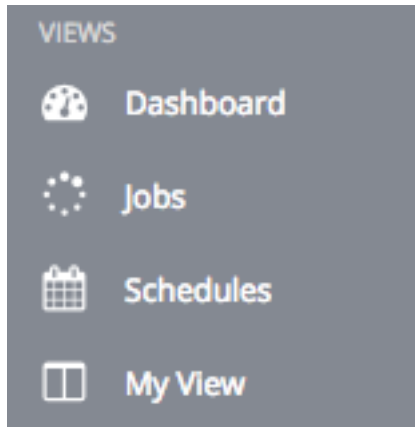
OK

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.

The screenshot shows the 'JOBS' view in Ansible Tower. The left sidebar is expanded to show the 'Jobs' menu item. The main content area displays a list of jobs with the following details:

Job ID	Job Name	Type	Started	Finished	Launched By	Job Template	Inventory	Project	Credentials
17	Demo Job Template	Playbook Run	9/11/2018 11:33:38 PM	9/11/2018 11:33:43 PM	admin	Demo Job Template	Demo Inventory	Demo Project	Demo Credential
18	Demo Project	SCM Update	9/11/2018 11:33:33 PM	9/11/2018 11:33:37 PM				Demo Project	
14	Demo Job Template	Playbook Run	9/11/2018 11:33:10 PM	9/11/2018 11:33:16 PM	admin	Demo Job Template	Demo Inventory	Demo Project	Demo Credential
15	Demo Project	SCM Update	9/11/2018 11:33:05 PM	9/11/2018 11:33:10 PM				Demo Project	
11	Demo Job Template	Playbook Run							

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.

The screenshot shows the 'SCHEDULES' view in Ansible Tower. The left sidebar contains navigation options under 'VIEWS' (Dashboard, Jobs, Schedules, My View) and 'RESOURCES' (Templates, Credentials, Projects). The main content area is titled 'SCHEDULES' and displays a 'SCHEDULED JOBS' section with a search bar and a 'KEY' button. Below this is a table with the following data:

NAME	TYPE	NEXT RUN	ACTIONS
ON Cleanup Job Schedule	Management Job	9/16/2018 12:46:50 PM	
ON Cleanup Activity Schedule	Management Job	9/18/2018 12:46:50 PM	


At the bottom right of the table, it says 'ITEMS 1 - 2'.


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.

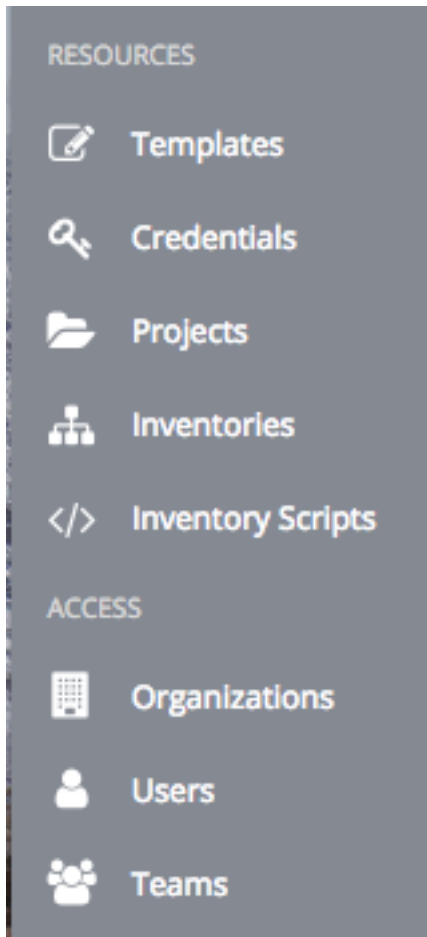
- **My Jobs:** View jobs that you (as the user) ran.
- **All Jobs:** View your team members' completed jobs, viewable based on your RBAC permissions.

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.

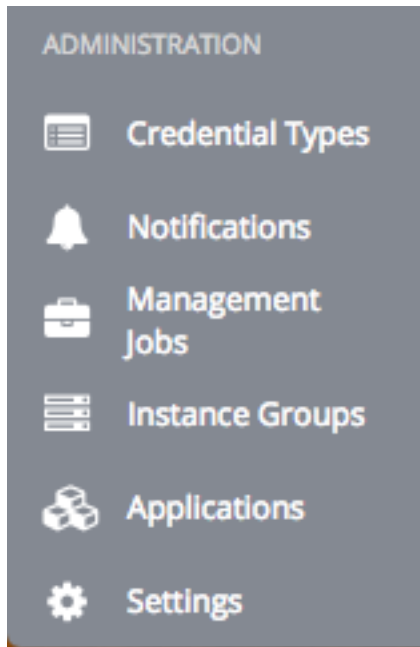
5.3 Resources and Access

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 Configure Tower 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.

The screenshot shows the 'EDIT CONFIGURATION' page in Ansible Tower. The main content area is titled 'CONFIGURE TOWER' and contains several configuration options for Azure AD authentication. At the top, there are tabs for 'AUTHENTICATION', 'JOBS', 'SYSTEM', 'USER INTERFACE', and 'LICENSE'. A 'SUB CATEGORY' dropdown menu is set to 'Azure AD'. Below this, there are three input fields: 'AZURE AD OAUTH2 CALLBACK URL' with the value 'https://35.237.31.162/sso/complete/azuread-oauth2/', 'AZURE AD OAUTH2 KEY' (empty), and 'AZURE AD OAUTH2 SECRET' (empty). Each of these fields has a 'REVERT' link to its right. Below the input fields are two sections for mapping: 'AZURE AD OAUTH2 ORGANIZATION MAP' and 'AZURE AD OAUTH2 TEAM MAP'. Each section contains a table with one row and one column, both containing the number '1'. To the right of each table is a 'REVERT' link. At the bottom left of the configuration area is a 'REVERT ALL TO DEFAULT' link. At the bottom right are 'CANCEL' and 'SAVE' buttons.

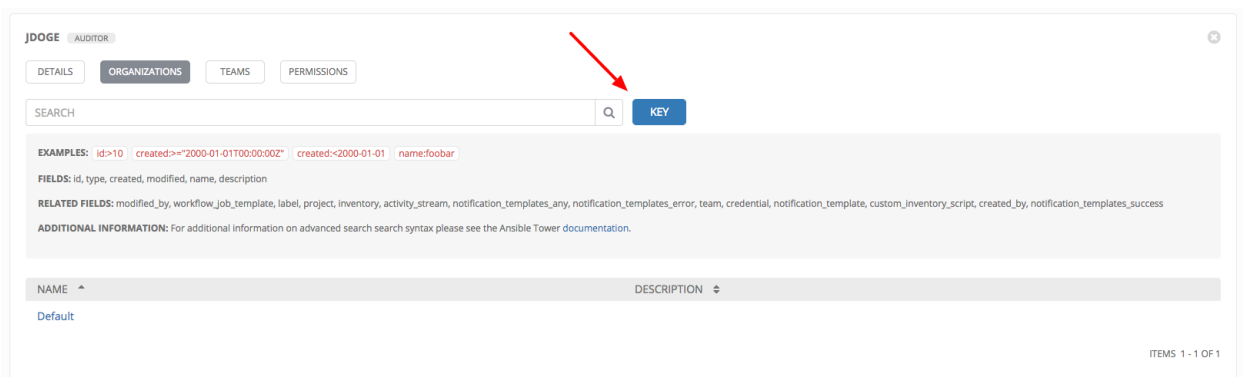
For further detail on configuring these options, refer to the [Tower Configuration](#) section.

SEARCH

Ansible Tower release 3.1 introduced the Tower Search, 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.



JDOGE AUDITOR

DETAILS ORGANIZATIONS TEAMS PERMISSIONS

SEARCH Q KEY

EXAMPLES: `id>10` `created>="2000-01-01T00:00:00Z"` `created<2000-01-01` `name:foobar`

FIELDS: id, type, created, modified, name, description

RELATED FIELDS: modified_by, workflow_job_template, label, project, inventory, activity_stream, notification_templates_any, notification_templates_error, team, credential, notification_template, custom_inventory_script, created_by, notification_templates_success

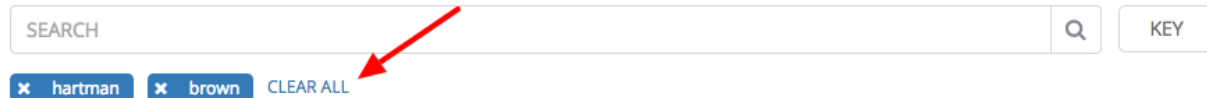
ADDITIONAL INFORMATION: For additional information on advanced search search syntax please see the Ansible Tower documentation.

NAME ^ DESCRIPTION v

Default

ITEMS 1 - 1 OF 1

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 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": [
    "schedule__search",
    "modified_by__search",
    "job_events__search",
    "extra_credentials__search",
    "project__search",
    "inventory__search",
    "unified_job_template__search",
    "unified_job_node__search",
    "unifiedjob_ptr__search",
    "instance_group__search",
    "labels__search",
    "job_host_summaries__search",
    "hosts__search",
    "notifications__search",
    "project_update__search",
    "credential__search",
    "dependent_jobs__search",
    "job_origin__search",
    "created_by__search",
    "job_template__search",
    "vault_credential__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).

SCHEDULED JOBS 13

SEARCH [] [] KEY [] Sort order set at ascending (up arrow)

NAME	TYPE	NEXT RUN	ACTIONS
<input type="checkbox"/> ON sync vmware inventory daily	Inventory Sync	12/18/2018 9:30:00 PM	[]
<input type="checkbox"/> OFF Duplicate and upgrade tower.testing on a schedule	Playbook Run	11/29/2018 4:15:00 AM	[]
<input type="checkbox"/> ON cleanup tower.testing docker stuff	Playbook Run	12/22/2018 10:00:00 PM	[]
<input type="checkbox"/> ON Build Development Container Image	Playbook Run	12/19/2018 12:30:00 AM	[]
<input type="checkbox"/> ON Every day at 2am	Playbook Run	12/18/2018 9:02:00 PM	[]

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

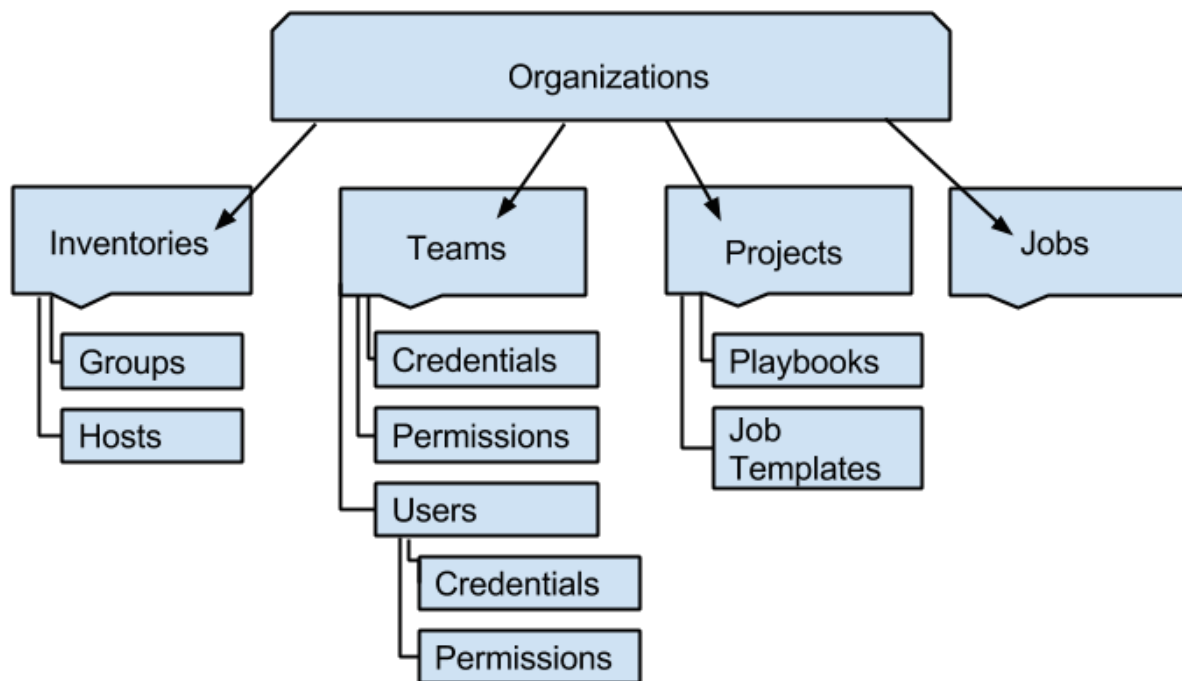
SCHEDULED JOBS 13


SEARCH

NAME [▲]	TYPE [⬇]	NEXT RUN [⬇]	ACTIONS
<input type="checkbox"/> ON Biweekly	Workflow Job	12/21/2018 9:00:00 PM	<input type="button" value="Q"/>
<input type="checkbox"/> ON Build Developer Isolated Container Image	Playbook Run	12/19/2018 12:30:00 AM	<input type="button" value="Q"/>
<input type="checkbox"/> ON Build Development Container Image	Playbook Run	12/19/2018 12:30:00 AM	<input type="button" value="Q"/>
<input type="checkbox"/> ON check users daily	Playbook Run	12/19/2018 5:00:00 AM	<input type="button" value="Q"/>
<input type="checkbox"/> ON Cleanup Activity Stream Schedule	Management Job	12/25/2018 1:00:00 AM	<input type="button" value="Q"/>
<input type="checkbox"/> ON Cleanup Job Details Schedule	Management Job	12/22/2018 10:00:00 PM	<input type="button" value="Q"/>
<input type="checkbox"/> ON cleanup tower.testing docker stuff	Playbook Run	12/22/2018 10:00:00 PM	<input type="button" value="Q"/>
<input checked="" type="checkbox"/> OFF Duplicate and upgrade tower.testing on a schedule	Playbook Run	11/29/2018 4:15:00 AM	<input type="button" value="Q"/>
<input type="checkbox"/> ON Every day at 2am	Playbook Run	12/18/2018 9:02:00 PM	<input type="button" value="Q"/>

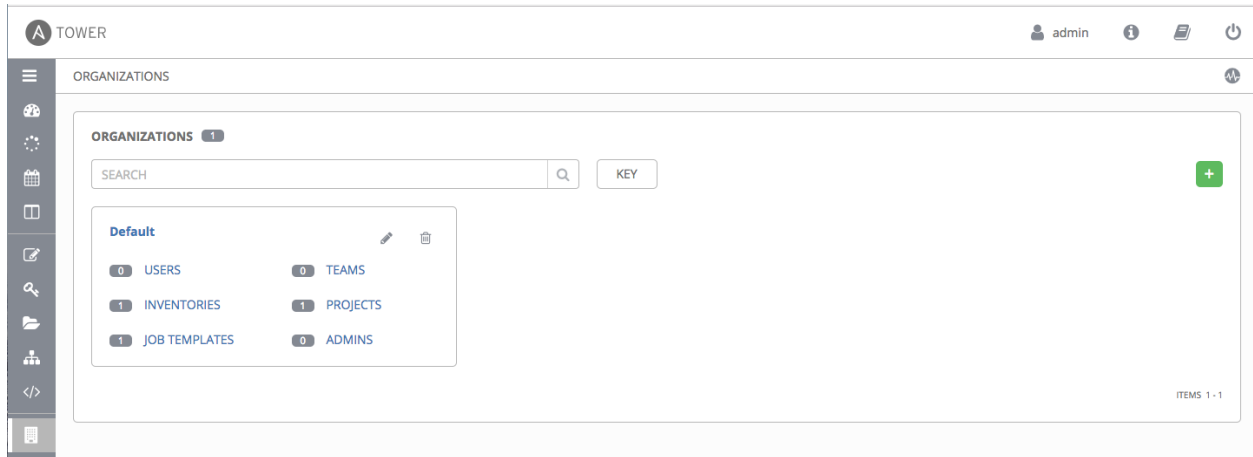
ORGANIZATIONS

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.

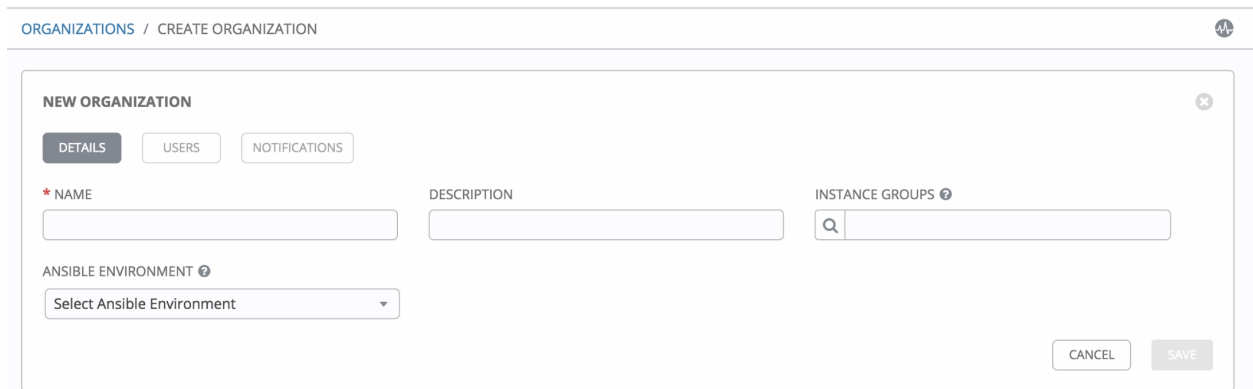
Note: Tower creates a default organization automatically. Users of Tower with a Self-Support level license (formerly called Basic) only have the default organization available and should **not** delete it. Users of older versions of Tower (prior to 2.2) will not see this default organization.



7.1 Creating a New Organization

“Enterprise: Standard” and “Enterprise: Premium” Tower licenses allow you to create a new Organization by selecting the  button.

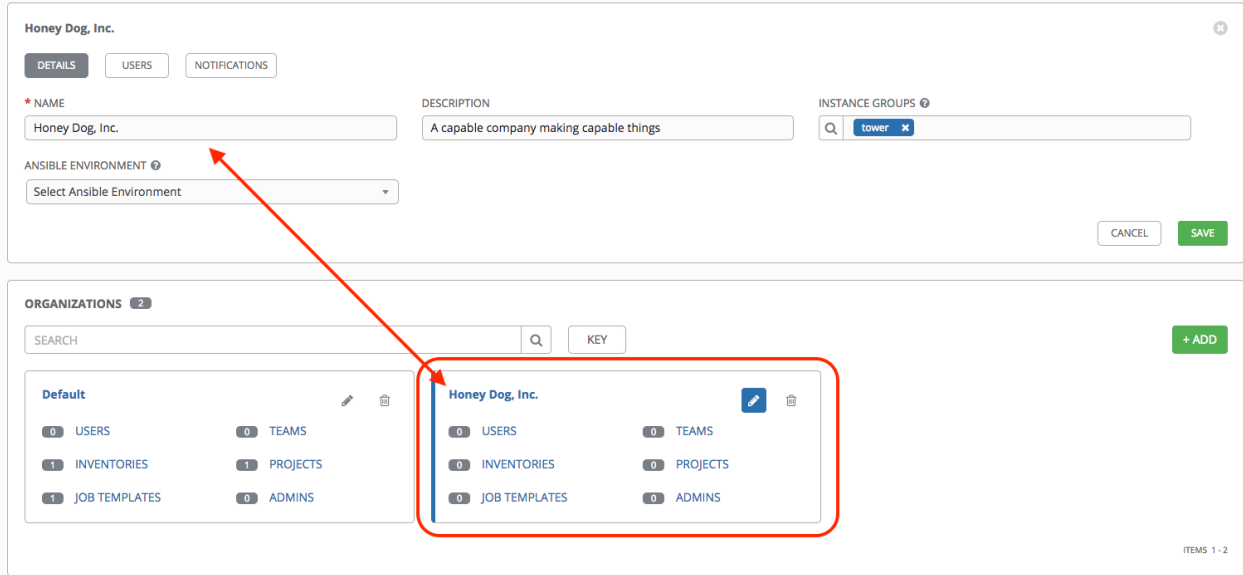
Note: If you are using Ansible Tower with a Self-Support level license (formerly called Basic), you must use the default Organization. Do not delete it and try to add a new Organization, or you will break your Tower setup. Only two Tower license types (Enterprise: Standard or Enterprise: Premium) have the ability to add new Organizations beyond the default.



An organization has several attributes that may be configured:

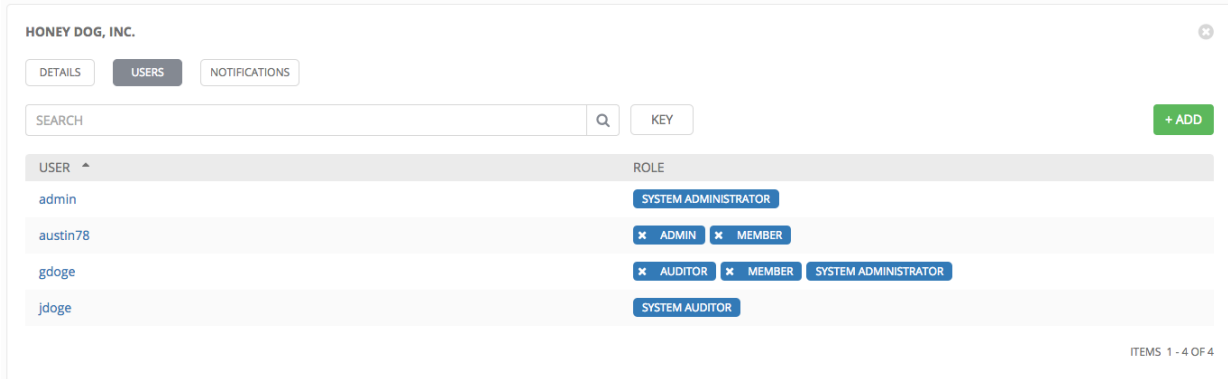
1. Enter the **Name** for your Organization (required).
2. Enter a **Description** for the Organization.
3. Enter an **Instance Group** on which to run this organization.
4. Select from the drop-down menu list a custom virtual **Ansible Environment** on which to run this organization.
5. Click **Save** to finish creating the Organization.


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



7.1.1 Organizations - 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.




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.

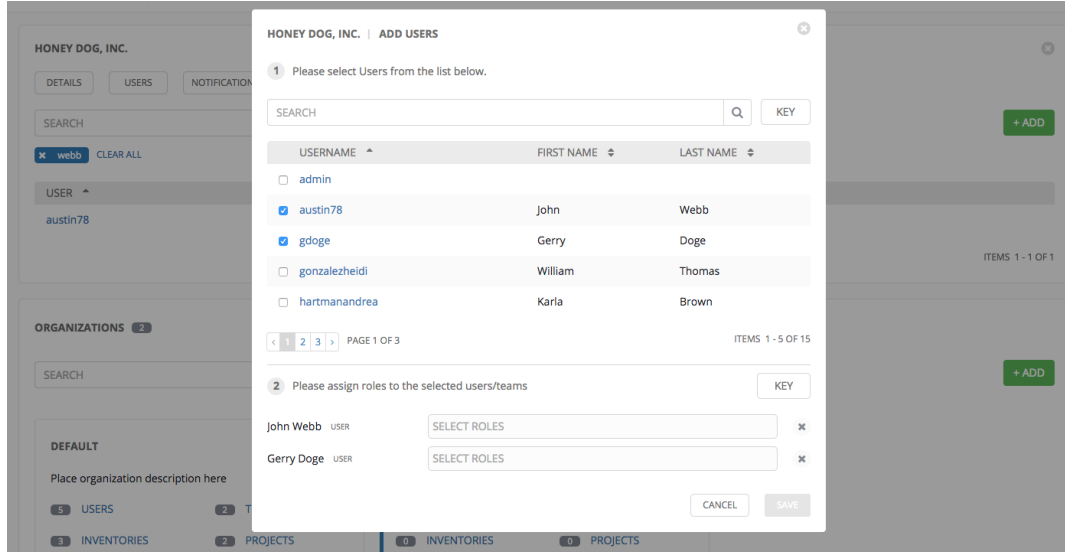
Clicking on a user brings up that user's details, allowing you to review, grant, edit, and remove associated permissions for that user. For more information, refer to [Users](#).

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:

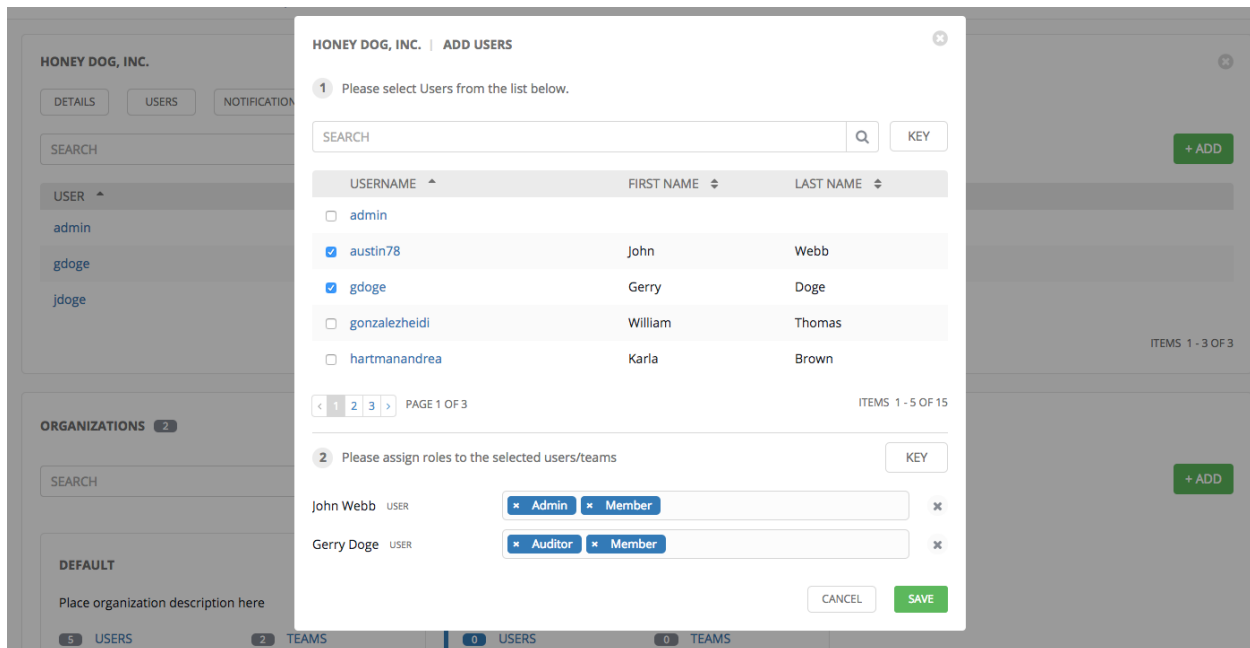


1. Click the  button.
2. Select one or more users from the list of available users by clicking the checkbox next to the user(s). Doing so expands the lower part of the Wizard to assign roles to each user.



3. For each user, click from the drop-down menu to select one or more roles for that user.

Note: For help on what the roles mean, click the **Key** button. For more information, refer to the [Roles](#) section of this guide.

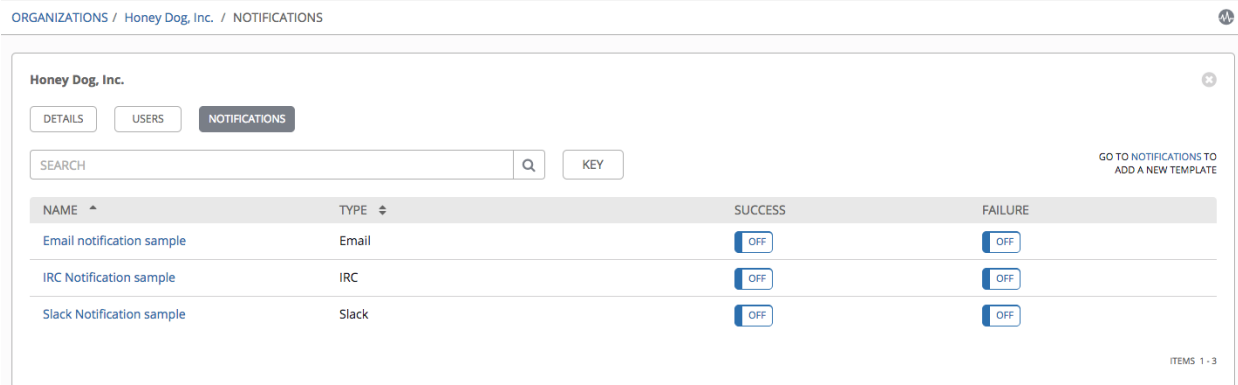


In this example, two users have been selected and each have been granted certain roles within this organization.

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

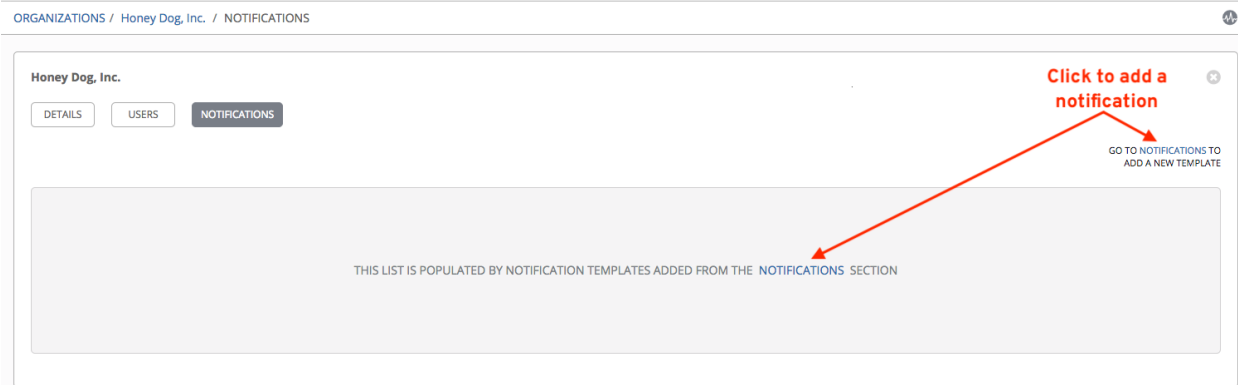
7.1.2 Organizations - Notifications

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



To create a new notification, click the **NOTIFICATIONS** link from the upper-right side of the notifications list view.

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



Supported notification sources include Slack, Email, SMS (via Twilio), HipChat, and more. Refer to *Notifications* for more information.

NEW NOTIFICATION TEMPLATE

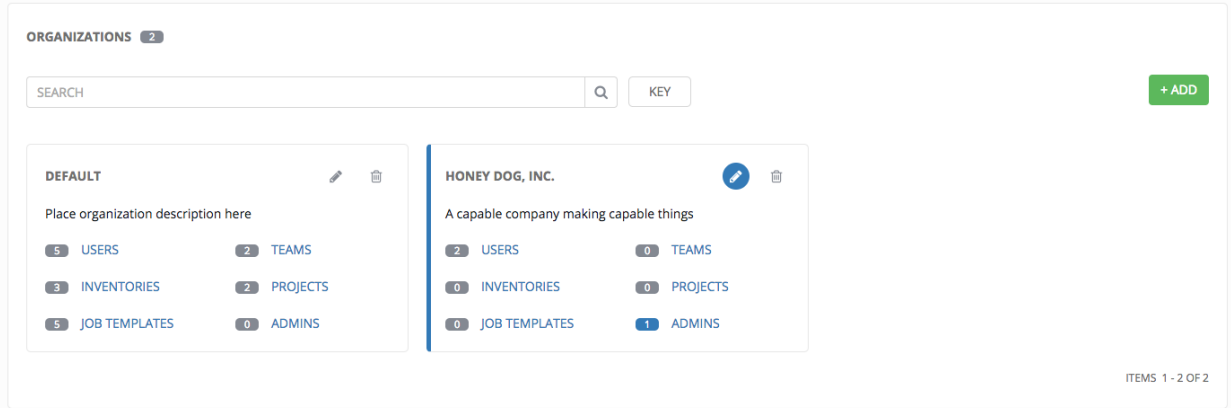
* NAME [] DESCRIPTION [] * ORGANIZATION []

* TYPE [Choose a type]



CANCEL SAVE

7.1.3 Organization - Summary

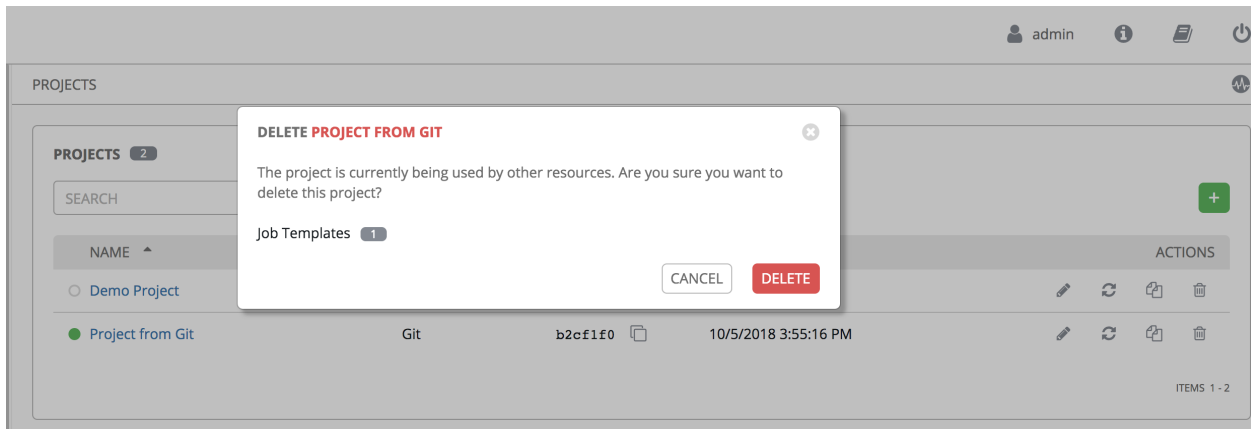
An at-a-glance view of various resources associated with an organization displays at the bottom of each Organization view, called the Organization Summary.



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 ().

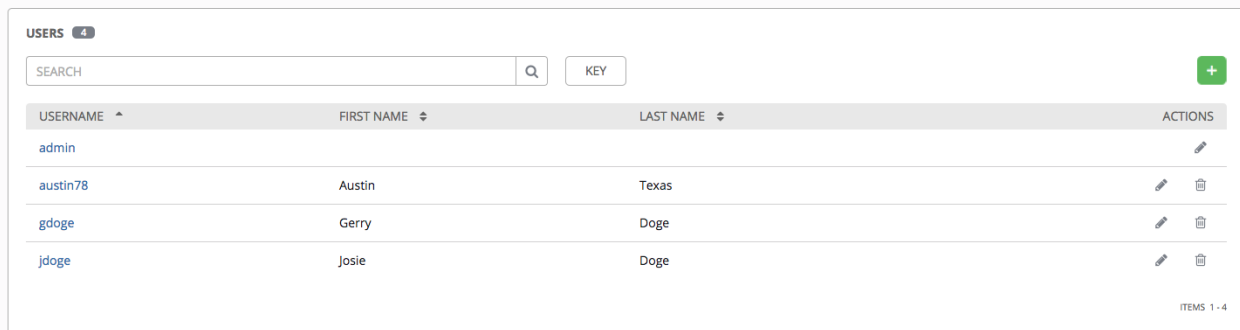
Note: If deleting items that are used by other work items, a message opens listing the items are affected by the deletion and prompts you to confirm the deletion. Some screens will contain items that are invalid or previously deleted, so they will fail to run. Below is an example of such a message:



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

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. Note 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.

austin78 ADMIN ✕

DETAILS ORGANIZATIONS TEAMS PERMISSIONS

* FIRST NAME * LAST NAME * EMAIL

* USERNAME PASSWORD SHOW CONFIRM PASSWORD SHOW

USER TYPE

System Administrator

CANCEL SAVE

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.

The screenshot shows the user profile for 'austin78' with the 'ADMIN' role. The 'TOKENS' tab is highlighted with a red box. The form contains the following fields:

- FIRST NAME: Austin
- LAST NAME: Texas
- * EMAIL: austin78@mail.com
- * USERNAME: austin78
- PASSWORD: [SHOW]
- CONFIRM PASSWORD: [SHOW]
- USER TYPE: System Administrator

Buttons for 'DETAILS', 'ORGANIZATIONS', 'TEAMS', 'PERMISSIONS', and 'TOKENS' are visible at the top. 'CANCEL' and 'SAVE' buttons are at the bottom right.

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 user profile for 'jdoge' with the 'AUDITOR' user type. A red arrow points to the 'AUDITOR' label with the text: "View user labels here for Auditor, Admin, LDAP, etc." The form contains the following fields:

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

Buttons for 'DETAILS', 'ORGANIZATIONS', 'TEAMS', and 'PERMISSIONS' are visible at the top.

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

The screenshot shows the user profile for 'jdoge' with the 'RADIUS' user type. A red arrow points to the 'RADIUS' label. The form contains the following fields:

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

Buttons for 'DETAILS', 'ORGANIZATIONS', 'TEAMS', and 'PERMISSIONS' are visible at the top.

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

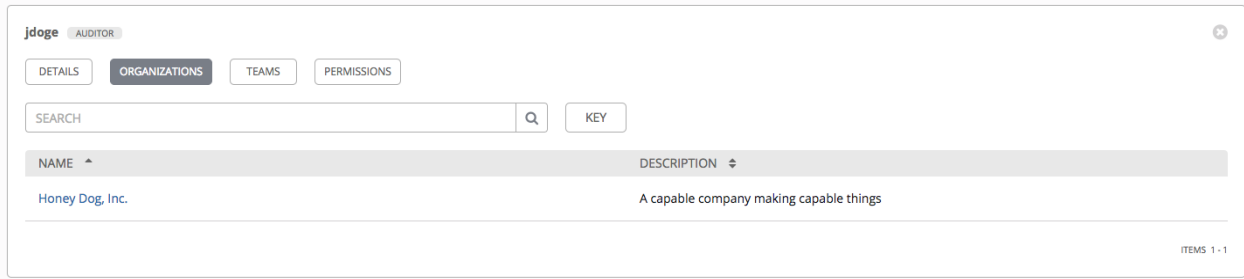
The screenshot shows the user profile for 'jdoge' with the 'SOCIAL' user type. A red arrow points to the 'SOCIAL' label. The form contains the following fields:

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

Buttons for 'DETAILS', 'ORGANIZATIONS', 'TEAMS', and 'PERMISSIONS' are visible at the top.

8.3 Users - Organizations

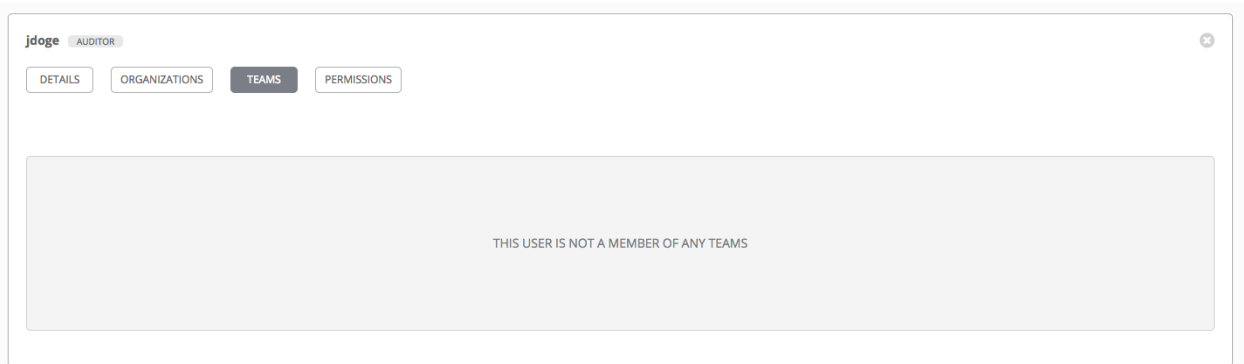
This displays the list of organizations of which that user is a member. This list may be searched by Organization Name or Description. Organization membership cannot be modified from this display panel.



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**.

jdoge

DETAILS ORGANIZATIONS TEAMS PERMISSIONS

SEARCH [] Q KEY [] +


NAME	TYPE	ROLE	ACTIONS
Default	Organization	Auditor	✕
Honey Dog, Inc.	Organization	Member	✕

ITEMS 1 - 2

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**



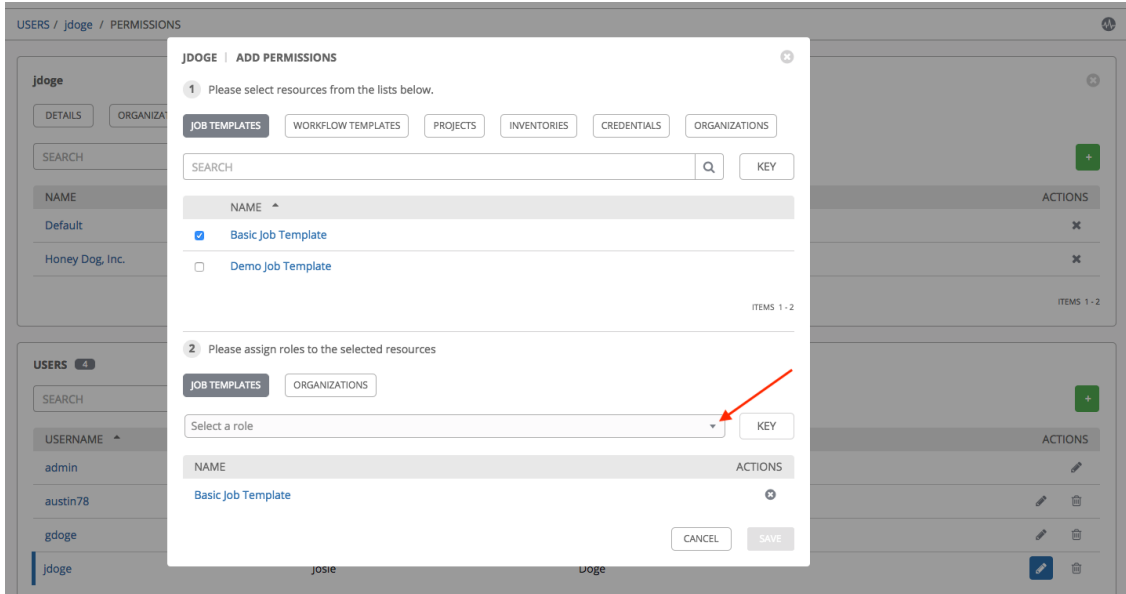
Note: You can assign different roles to different resources all at once to avoid having to click the  button. To do so, simply go from one tab to another after making your selections without saving.

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.

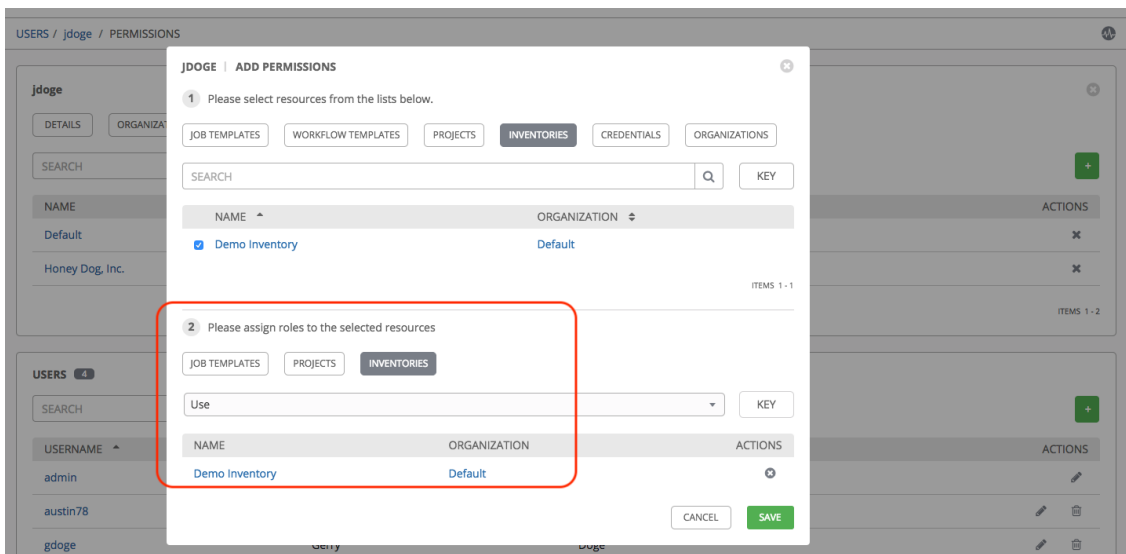
The dialog expands to allow you to select the role for the resource you chose.

(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	x
Demo Inventory	Inventory	Use	x
Honey Dog, Inc.	Organization	Member	x
Sample Project	Project	Admin	x
Basic Job Template	Job Template	Admin	x

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.

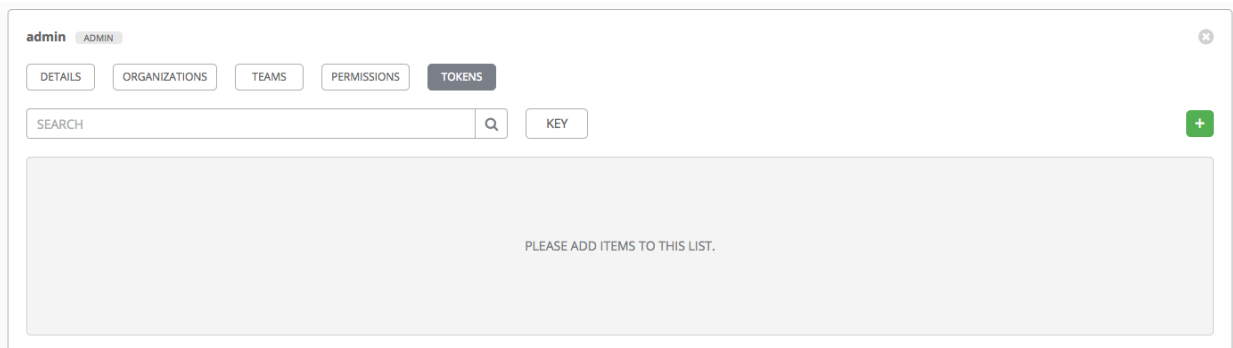
Note: You can also add teams or individual users and assign them permissions at the object level (projects, inventories, job templates, and workflow templates) as well. Ansible Tower release 3.1 introduces the ability to batch assign permissions. This feature reduces the time for an organization to onboard many users at one time. For more details, refer to their respective chapters in the *Ansible Tower User Guide v3.3.6*.



8.6 Users - Tokens

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:



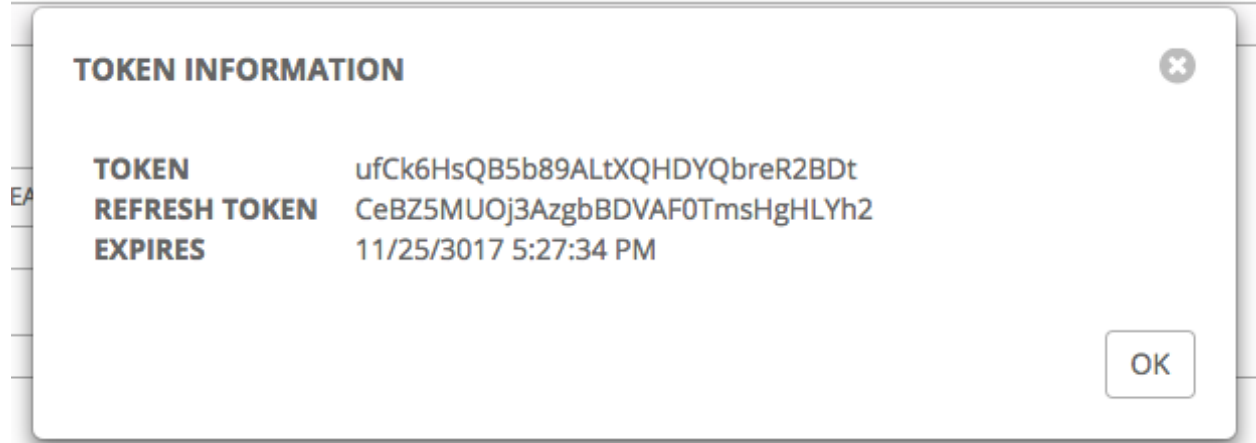
3. Click the  button, which opens the Create Token window.
4. Enter the following details in Create Token window:
 - **Application:** enter the name of the application with which you want to associate your token. Alternatively, you can search for it by clicking the  button. This opens a separate window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

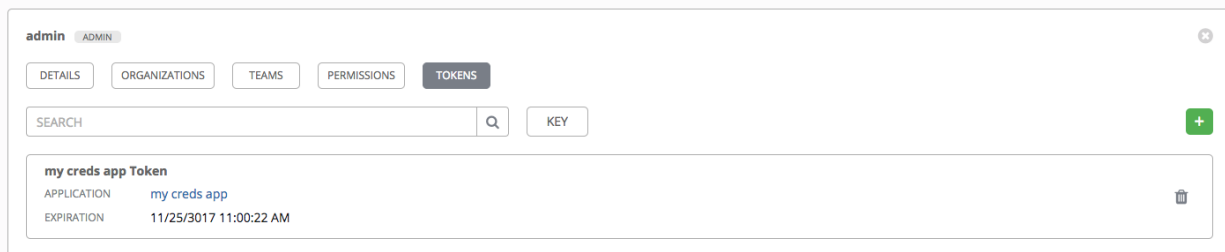
5. When done, click **Save** or **Cancel** to abandon your changes.

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.




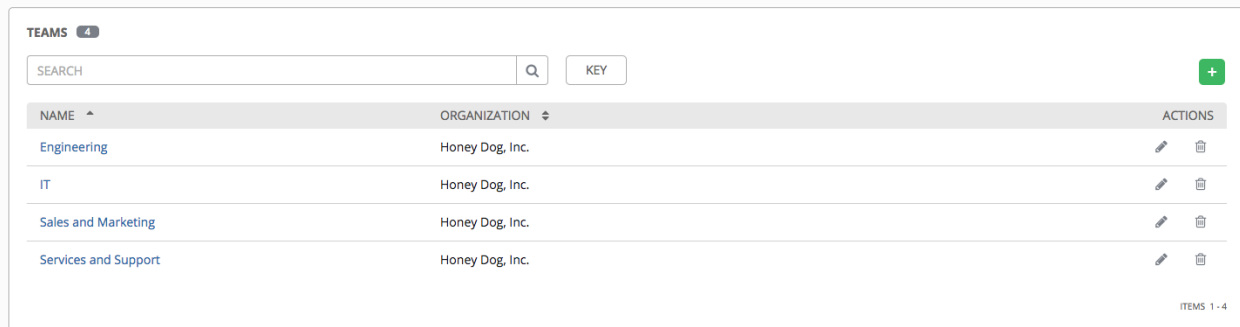
TEAMS

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**.



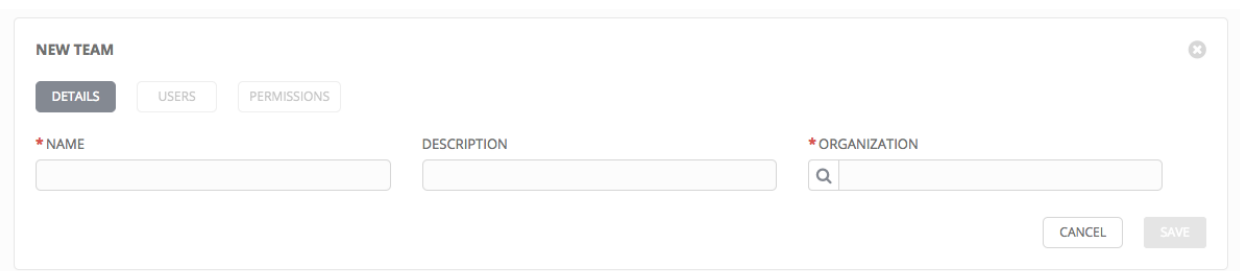
The screenshot shows the 'TEAMS' page with a search bar and a table of teams. The table has columns for NAME, ORGANIZATION, and ACTIONS. The data rows are:

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

9.1 Create a Team

To create a new Team:

1. Click the  button.




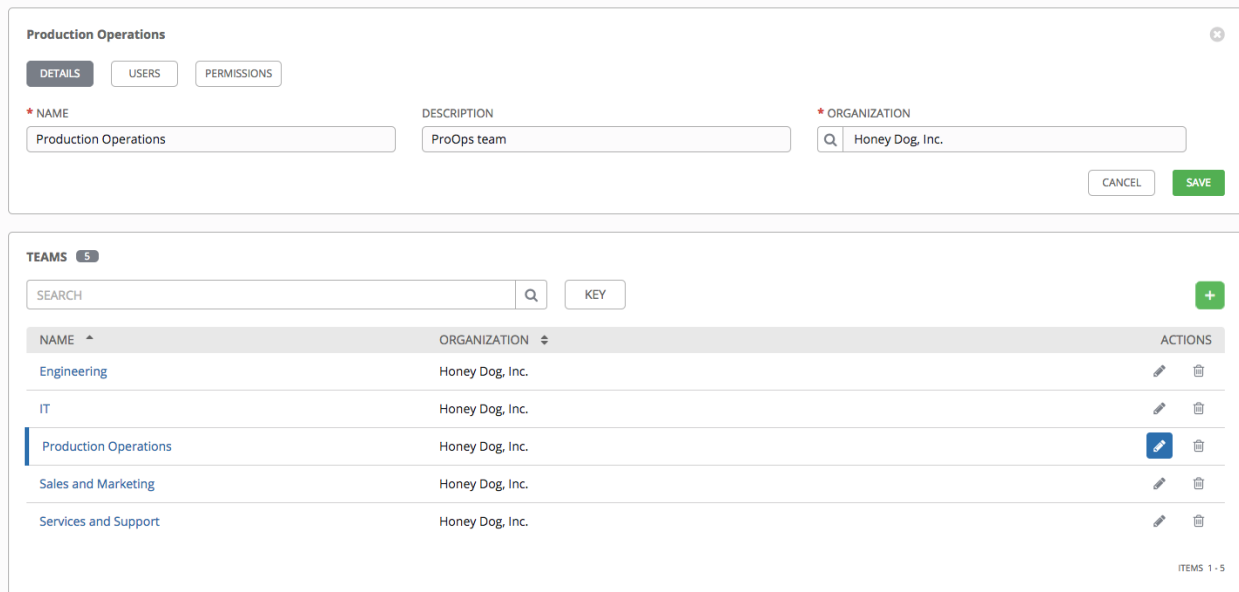
The 'NEW TEAM' form has three tabs: DETAILS (selected), USERS, and PERMISSIONS. It contains three input fields: NAME (required), DESCRIPTION, and ORGANIZATION (required). There are CANCEL and SAVE buttons at the bottom right.

2. Enter the appropriate details into the following fields:

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

3. Click **Save**.

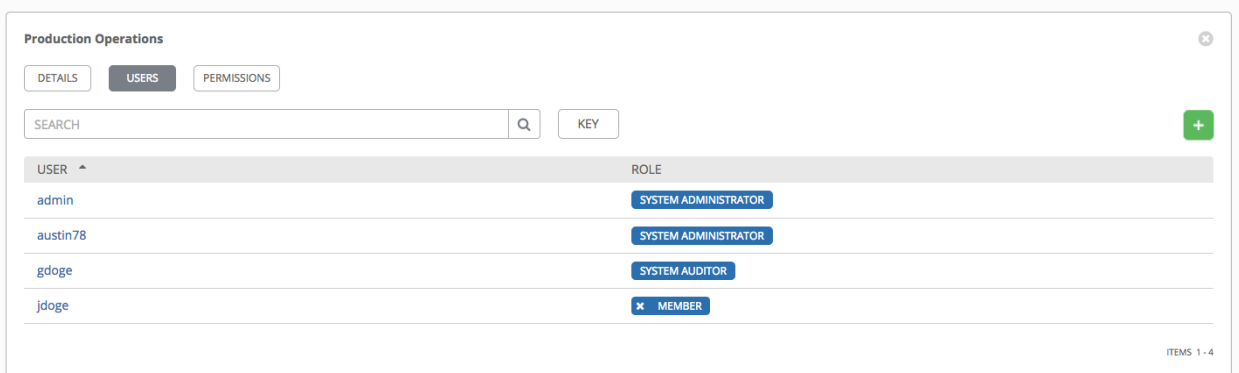
Once the Team is successfully created, Tower opens the **Details** dialog, which also allows you to review and edit your Team information. This is the same menu that is opened if the Edit () button is clicked from the **Teams** link. You can also review **Users** and **Permissions** associated with this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



The image shows two screenshots from the Ansible Tower interface. The top screenshot is the 'Production Operations' team details dialog. It has three tabs: 'DETAILS' (selected), 'USERS', and 'PERMISSIONS'. The 'DETAILS' tab contains three input fields: '* NAME' with the value 'Production Operations', 'DESCRIPTION' with the value 'ProOps team', and '* ORGANIZATION' with the value 'Honey Dog, Inc.'. There are 'CANCEL' and 'SAVE' buttons at the bottom right. The bottom screenshot shows the 'TEAMS' list view. It has a search bar and a 'KEY' button. Below is a table with columns for 'NAME', 'ORGANIZATION', and 'ACTIONS'. The 'Production Operations' team is highlighted. The table lists five teams: Engineering, IT, Production Operations, Sales and Marketing, and Services and Support, all associated with 'Honey Dog, Inc.'. Each team has edit and delete icons. The bottom right corner shows '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*.




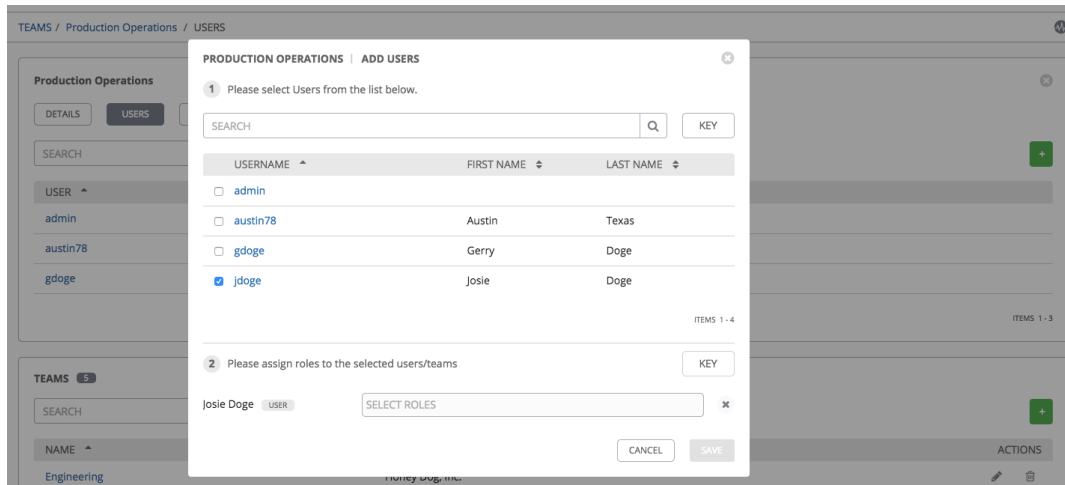
The image shows the 'Production Operations' team users list. It has three tabs: 'DETAILS', 'USERS' (selected), and 'PERMISSIONS'. Below the tabs is a search bar and a 'KEY' button. The main area is a table with columns for 'USER' and 'ROLE'. The table lists four users: 'admin' (SYSTEM ADMINISTRATOR), 'austin78' (SYSTEM ADMINISTRATOR), 'gdoge' (SYSTEM AUDITOR), and 'jdoge' (MEMBER). The 'MEMBER' role has a red 'x' icon next to it. The bottom right corner shows '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. To add existing users to the Team:

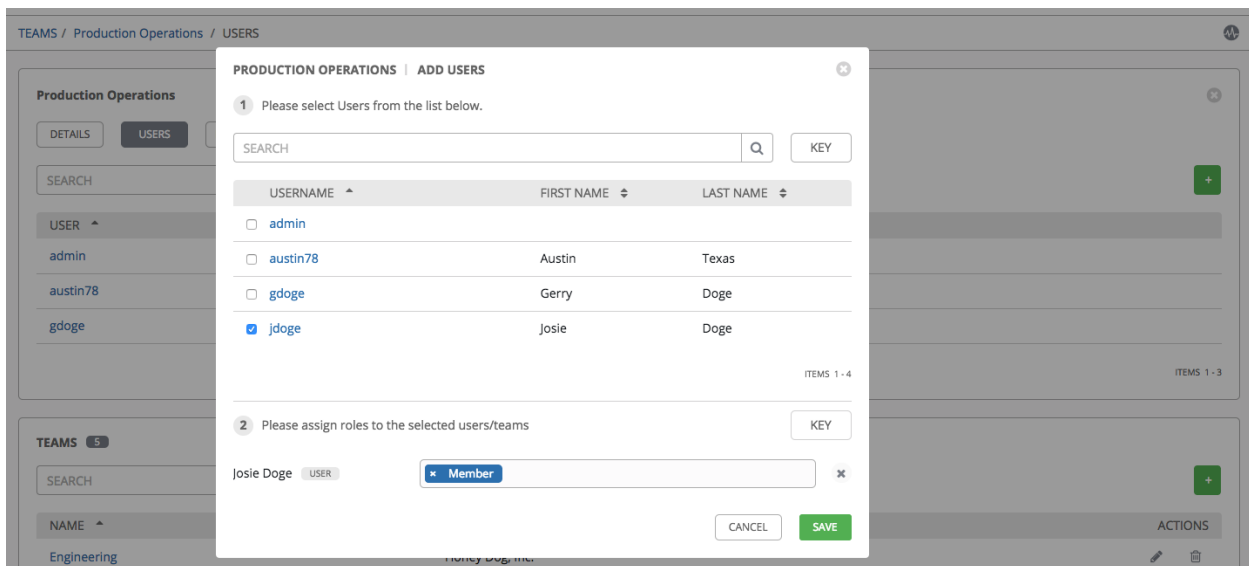


1. Click the  button.
2. Select one or more users from the list of available users by clicking the checkbox next to the user(s). Doing so expands the lower part of the Wizard to assign roles to each user.



3. For each user, click from the drop-down menu to select one or more roles for that user.

Note: For help on what the roles mean, click the **Key** button. For more information, refer to the [Roles](#) section of this guide.



In this example, two users have been selected and each have been granted certain roles within 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

PRODUCTION OPERATIONS | ADD PERMISSIONS

1 Please select resources from the lists below.

JOB TEMPLATES WORKFLOW TEMPLATES PROJECTS INVENTORIES CREDENTIALS

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**



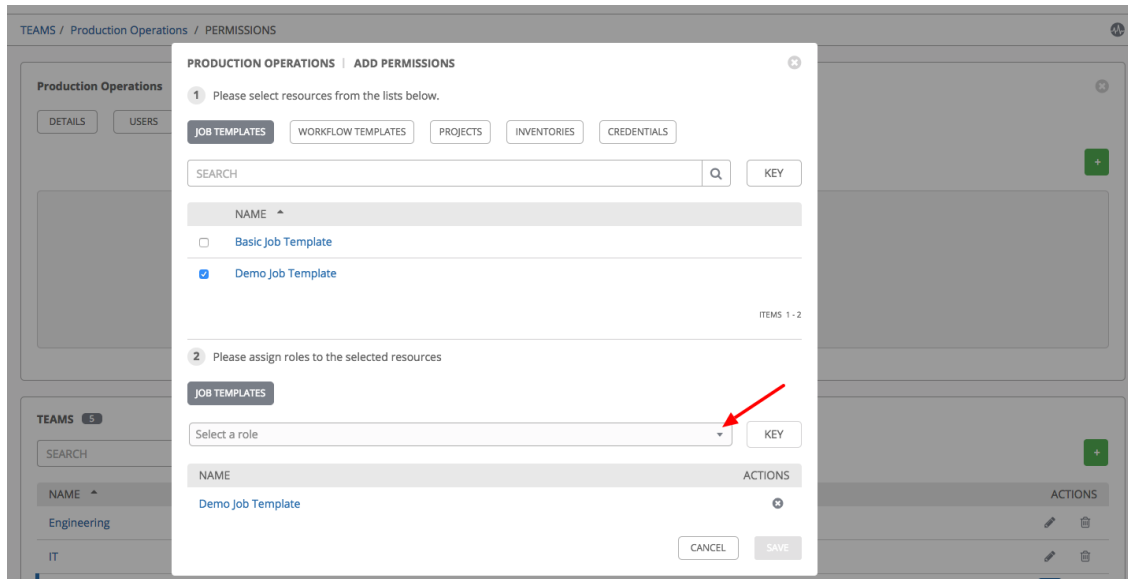
Note: You can assign different roles to different resources all at once to avoid having to click the button. To do so, simply go from one tab to another after making your selections without saving.

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.

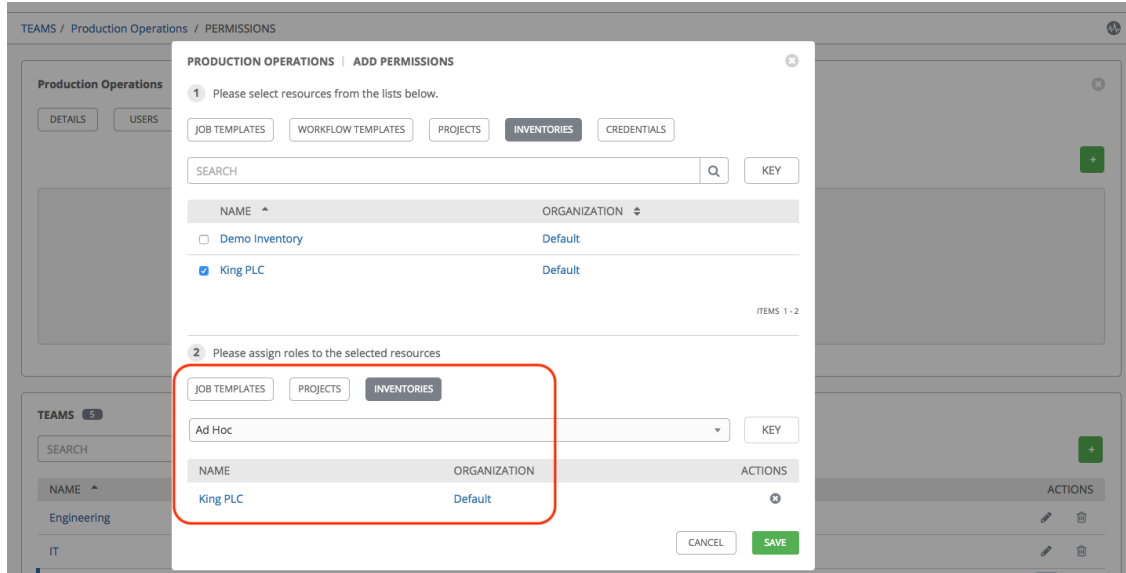
The dialog expands to allow you to select the role for the resource you chose.

(b) Select the role from the drop-down menu list provided. Only some roles are applicable to certain resources.

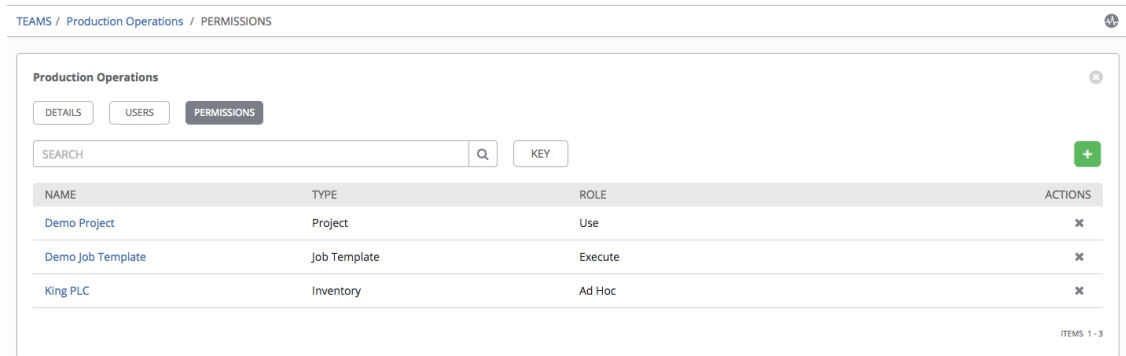



Tip: Use the **Key** button to display the help text for each of the roles applicable to the resource selected.

(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 or individual users and assign them permissions at the object level (projects, inventories, job templates, and workflow templates) as well. Ansible Tower release 3.1 introduces the ability to batch assign permissions. This feature reduces the time for an organization to onboard many users at one time. For more details, refer to their respective chapters in the *Ansible Tower User Guide v3.3.6*.

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.

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.

The encryption/decryption algorithm uses a variation of Fernet: a symmetric encryption cipher utilizing AES-256 in CBC mode alongside a SHA-256 HMAC. The key is derived from the `SECRET_KEY` (found in the `awx` settings). Specific, sensitive, Model fields in Tower are encrypted and include:


```
Credential: password, ssh_key_data, ssh_key_unlock, become_password, vault_password
UnifiedJob: start_args
```

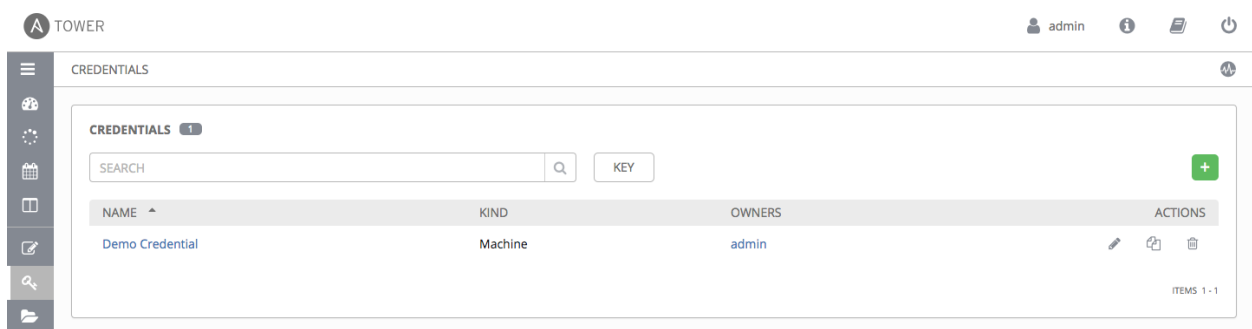
Data is encrypted before it is saved to the database and is decrypted as is needed in Tower. The encryption/decryption process derives the AES-256 bit encryption key from `<SECRET_KEY, field_name, primary_key>` where `field_name` is the name of the Model field and `primary_key` is the database assigned auto-incremented record ID. Thus, if any attribute used in the key generation process changes, Tower fails to correctly decrypt the secret.

Note: The rules of encryption and decryption for Ansible Tower also apply to one field outside of credentials, the Unified Job `start_args` field, which is used through the `job`, `ad_hoc_command`, and `system_job` data types.

10.2 Getting Started with Credentials

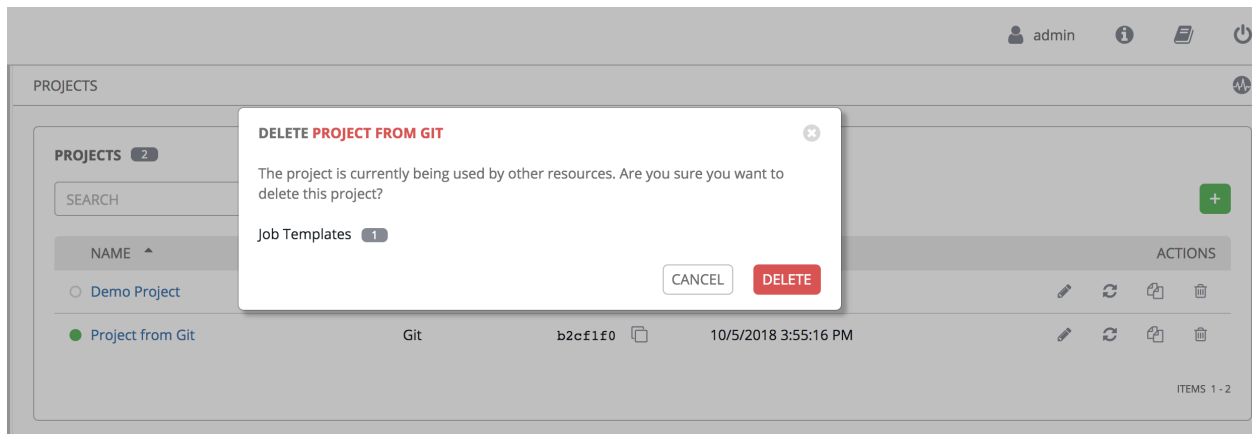


Access the Credentials page by clicking the Credentials () icon from the left navigation bar. The Credentials page displays a search-able list of all available Credentials and can be sorted by **Name**.



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.

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

USER	ROLE	TEAM ROLES
admin	ADMIN, SYSTEM ADMINISTRATOR	




You can click the  button to assign this **Demo Credential** to additional Users or Teams.

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.

CREENTIALS / CREATE CREDENTIAL 🔍

NEW CREDENTIAL ✕

DETAILS | PERMISSIONS

* NAME ? DESCRIPTION ? ORGANIZATION ?

* CREDENTIAL TYPE ?

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

SELECT CREDENTIAL TYPE ✕

NAME ▲

- Amazon Web Services
- Ansible Tower
- Google Compute Engine
- Insights
- Machine

< 1 2 3 > PAGE 1 OF 3
ITEMS 1 - 5 OF 14

5. Enter the appropriate details depending on the type of credential selected, as described in the following sections.
6. Click **Save** when done.

10.4 Credential Types

Topics:

- *Amazon Web Services*
- *Ansible Tower*
- *Google Compute Engine*
- *Insights*
- *Machine*
- *Microsoft Azure Resource Manager*
- *Network*
- *OpenStack*
- *Red Hat CloudForms*
- *Red Hat Satellite 6*
- *Red Hat Virtualization*
- *Source Control*
- *Vault*
- *VMware vCenter*

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 options:

- DETAILS** (selected) / PERMISSIONS
- * NAME: [Empty text box]
- DESCRIPTION: [Empty text box]
- ORGANIZATION: [SELECT AN ORGANIZATION]
- * CREDENTIAL TYPE: Amazon Web Services (indicated by a red arrow)
- TYPE DETAILS**
- * ACCESS KEY: [Empty text box]
- * SECRET KEY: [SHOW] [Empty text box]
- STS TOKEN: [SHOW] [Empty text box]
- CANCEL / SAVE buttons at the bottom right.

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 Tower

Selecting this credential allows you to access another Tower instance.

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.

10.4.3 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, with a hint: "HINT: Drag and drop an SSH private key file on the field below.")

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.

10.4.4 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:** New credential
- DESCRIPTION:** (empty)
- ORGANIZATION:** Default
- CREDENTIAL TYPE:** Insights (indicated by a red arrow)
- USERNAME:** (empty)
- PASSWORD:** (empty, with a 'SHOW' button)

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

10.4.5 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 `sudo` | `su` | `pbrun` | `pfexec` | `dzdo` | `pmrun`.
 - empty selection: Assigns no privilege escalation to this credential.
 - **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](#))
- **pbrun**: Requests that an application is run in a controlled account (refer to [Privilege Manager for Unix 6.0](#))
- **runas**: Allows you to run as current user.

- **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**”.

10.4.6 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
```

10.4.7 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](#).

10.4.8 OpenStack

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

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.
- **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.

10.4.9 Red Hat CloudForms

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

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

```

CLOUDFORMS_INI_PATH
    
```

CloudForms credentials have the following inputs that are required:

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

Additional Resources:

Refer to Red Hat’s blog post series on Ansible Tower Integration in Red Hat CloudForms 4.1 at <http://cloudformsblog.redhat.com/2016/07/22/ansible-tower-in-cloudforms/>.

10.4.10 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
```

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.

10.4.11 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.
- **Username:** The username to use to connect to oVirt4.
- **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)

10.4.12 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.

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**”.

10.4.13 Vault

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

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

10.4.14 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
```


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 Backwards-Compatible API Considerations

With Ansible Tower version 3.2, new support for version 2 of the API (V2) means:

- One-to-many relationship for Job Templates to credentials (including multi-cloud support)
- Custom credentials will not be managed by the V1 API; if a user defines a custom credential type, its credentials will not show up in the V1 API
- POSTs to V1 credential API will transparently work with migrated `CredentialTypes/Credentials`

Credentials have the concept of “Kind” that dictates:

- How or *where* a credential can be used.
- 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.


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

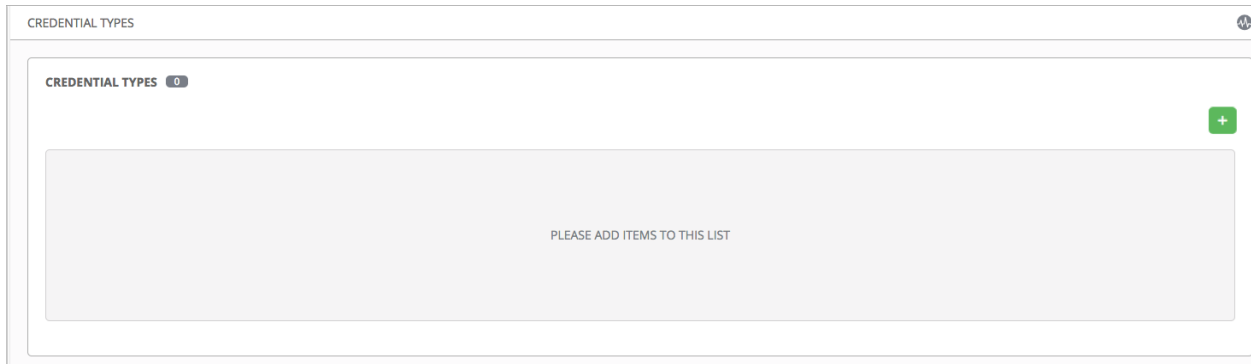
Machine	SSH
Vault	Vault
Network	Sets environment variables (e.g., ANSIBLE_NET_AUTHORIZE)
SCM	Source Control
Cloud	EC2, AWS
	Lots of others
Insights	Insights

Custom type creation and modification are limited to cloud and network kinds.

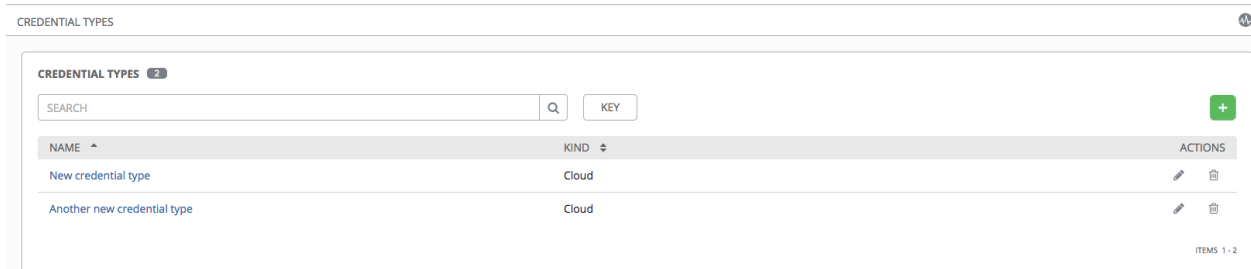
11.2 Getting Started with Credential Types




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. It can be sorted and searched by **Name** and **Kind**.



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.

11.3 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",           # required - a unique name used to
                                # reference the field value

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

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

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

    "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:

```
{
  "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

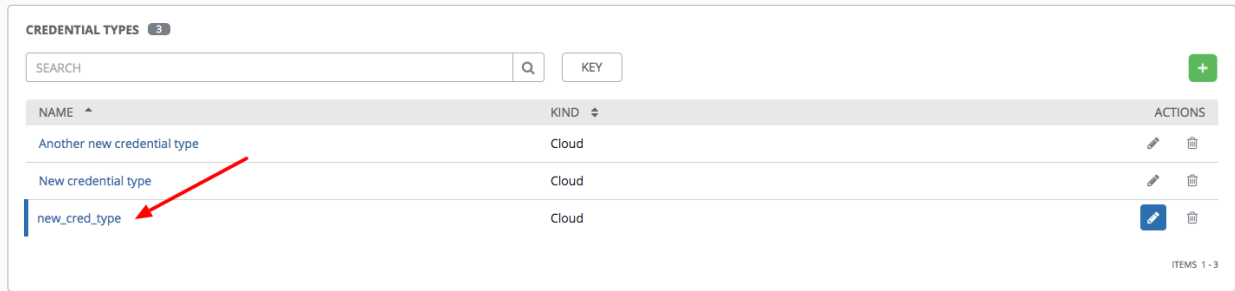
```
{
  "fields": [{
    "id": "cert",
    "label": "Certificate",
    "type": "string"
  }, {
    "id": "key",
    "label": "Key",
    "type": "string"
  }]
}
```



Injectors

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

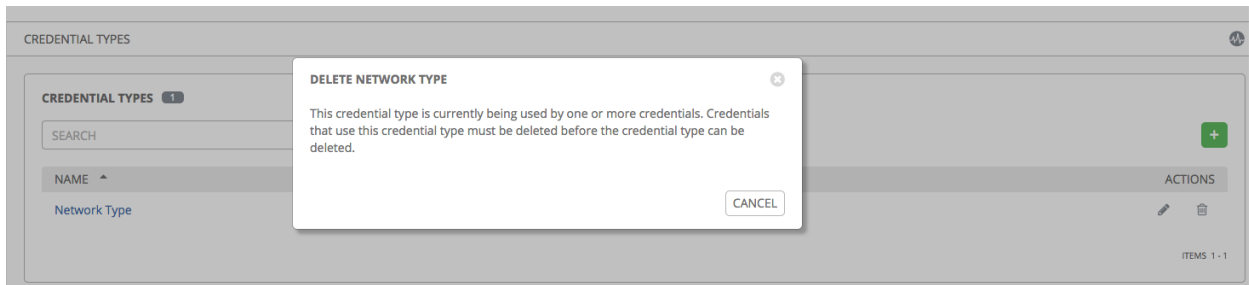
```
}
}
```

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:

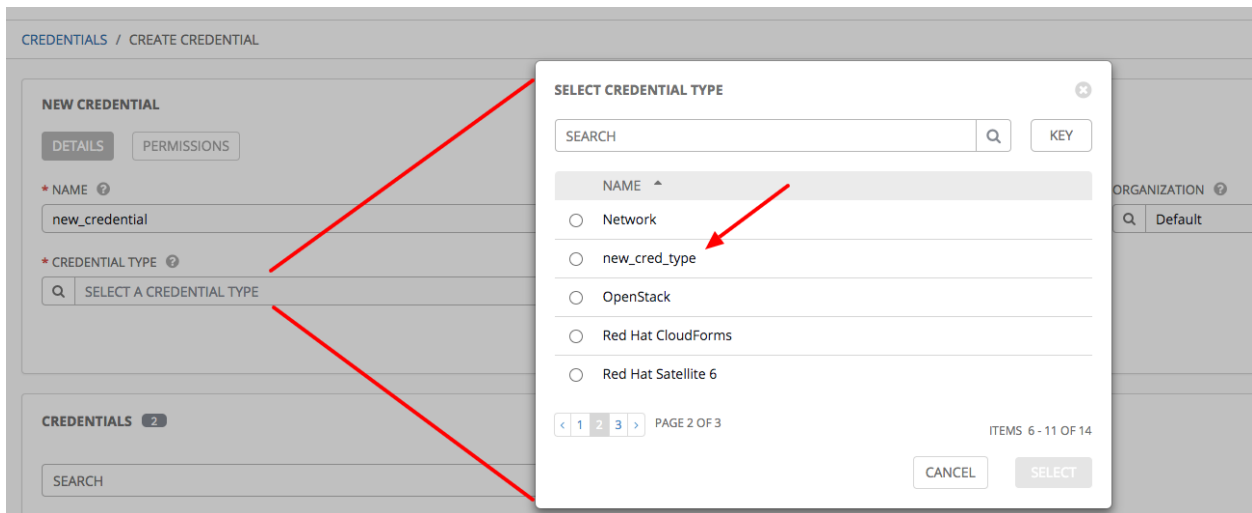


Click  to modify or  to remove the credential type options under the Actions column.

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:




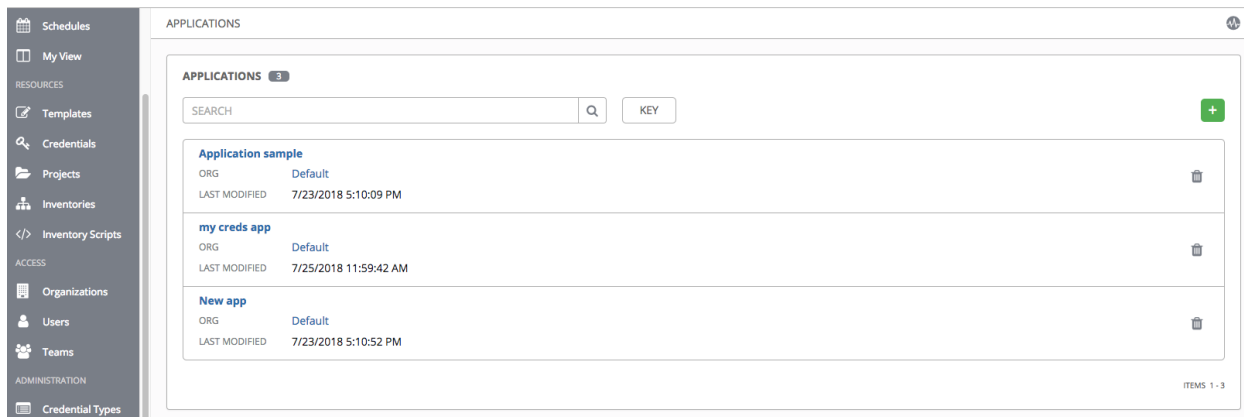
For details on how to create a new credential, see [Credentials](#).

APPLICATIONS

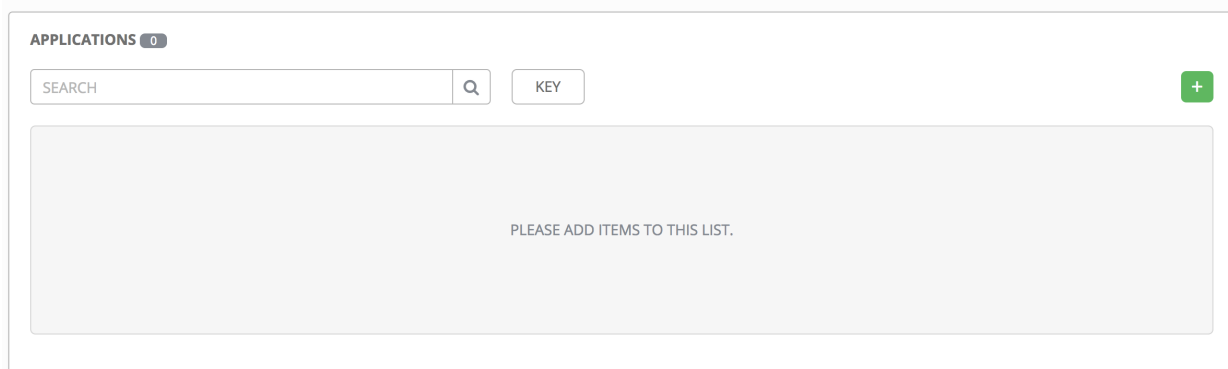
Creating and configuring token-based authentication for external applications is available in Ansible Tower 3.3.

12.1 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.




12.2 Create a new application

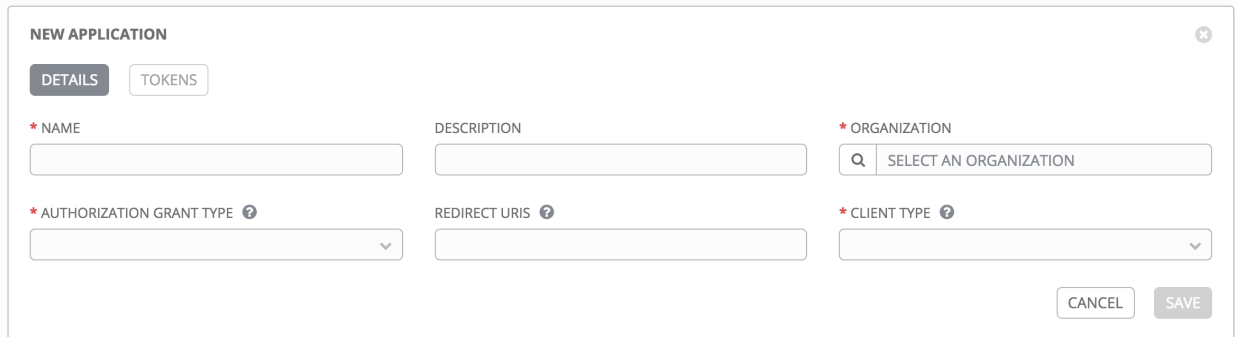
Token-based authentication for users can be configured in the Applications window.

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.



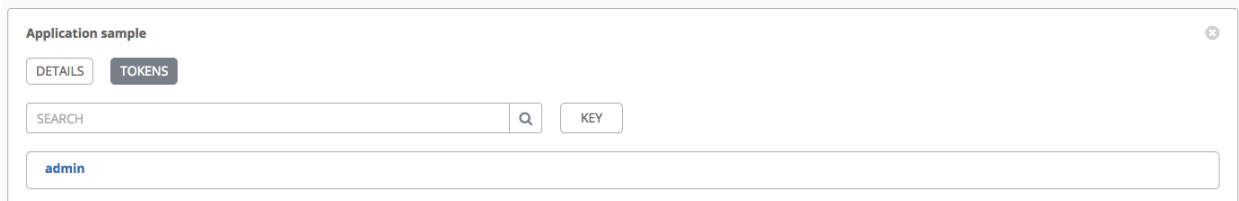
3. Enter the following details in **Create New Application** window:

- **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** or **Implicit**.
- **Client Type** (required): Select the level of security of the client device

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

12.2.1 Applications - Tokens


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



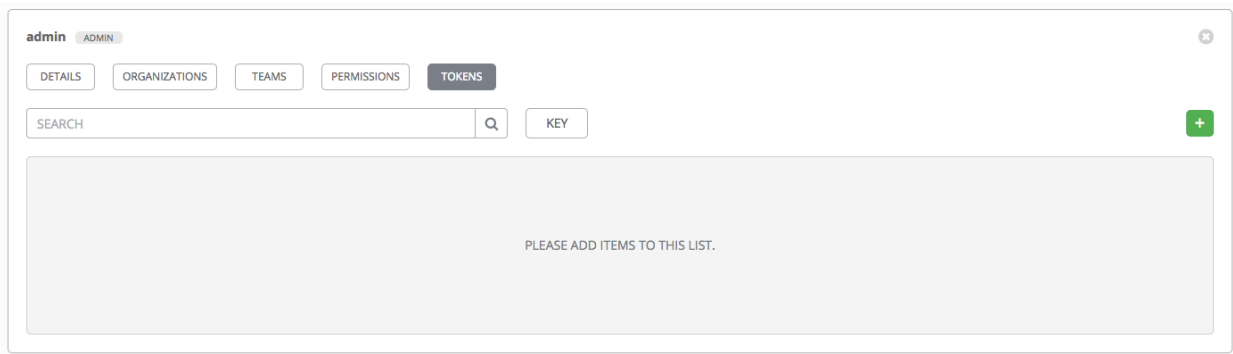
Tokens can only access resources that its associated user can access, and can be limited further by specifying the scope of the token.



Add Tokens

Tokens are added through the Users screen before they can be associated with an application. 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. To add a token:

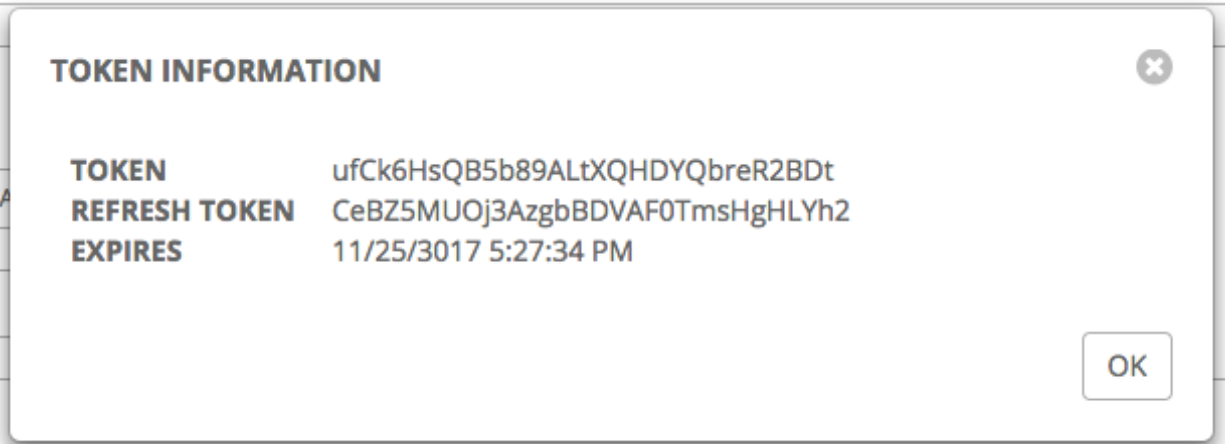
1. Access the Users list view by clicking the Users () icon from the left navigation bar then click on your user to configure your OAuth 2 tokens.
2. Click the **Tokens** tab from your user's profile.

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



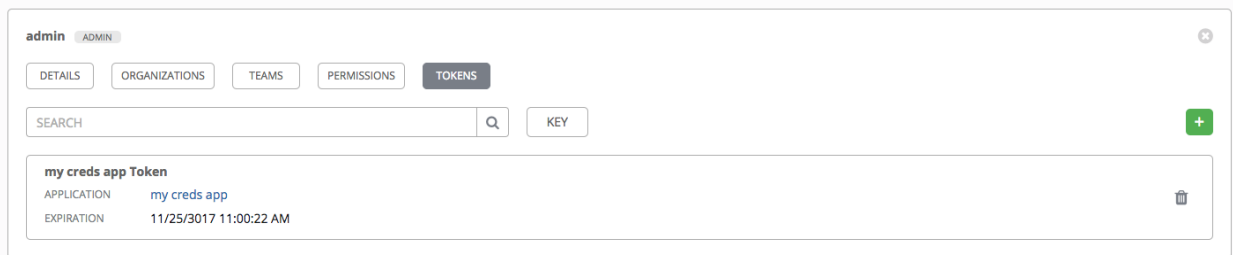
3. Click the  button, which opens the Create Token window.
4. Enter the following details in Create Token window:
 - **Application:** enter the name of the application with which you want to associate your token. Alternatively, you can search for it by clicking the  button. This opens a separate window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

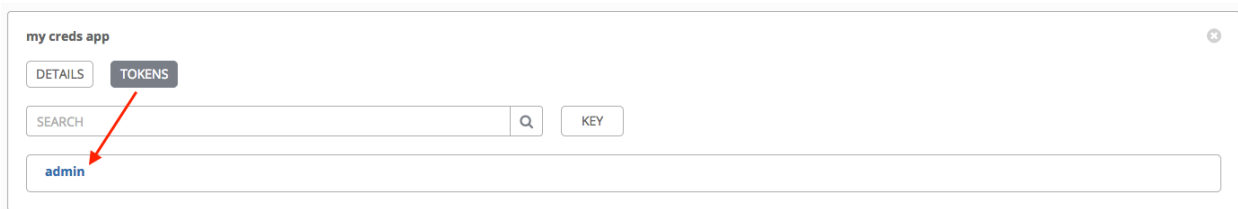


Note: This is the only time the token value and associated refresh token value will ever be shown.

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:



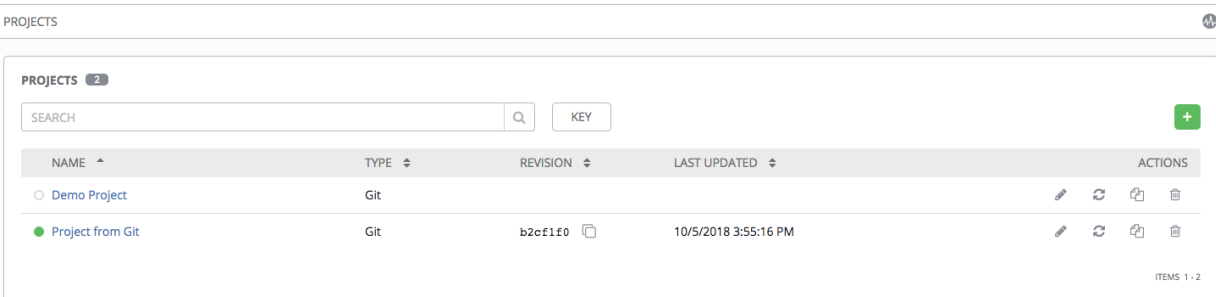
PROJECTS

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/settings.py`. Use caution when editing this file, as incorrect settings can disable your installation.

This menu displays a list of the projects that are currently available. The list of projects may be sorted and searched by any of the table headers displayed.







The screenshot shows the 'PROJECTS' page in Tower. It features a search bar, a 'KEY' button, and a table with columns for NAME, TYPE, REVISION, LAST UPDATED, and ACTIONS. Two projects are listed: 'Demo Project' and 'Project from Git'.

NAME	TYPE	REVISION	LAST UPDATED	ACTIONS
Demo Project	Git			[Edit] [Refresh] [Share] [Delete]
Project from Git	Git	b2cf1f0	10/5/2018 3:55:16 PM	[Edit] [Refresh] [Share] [Delete]

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):

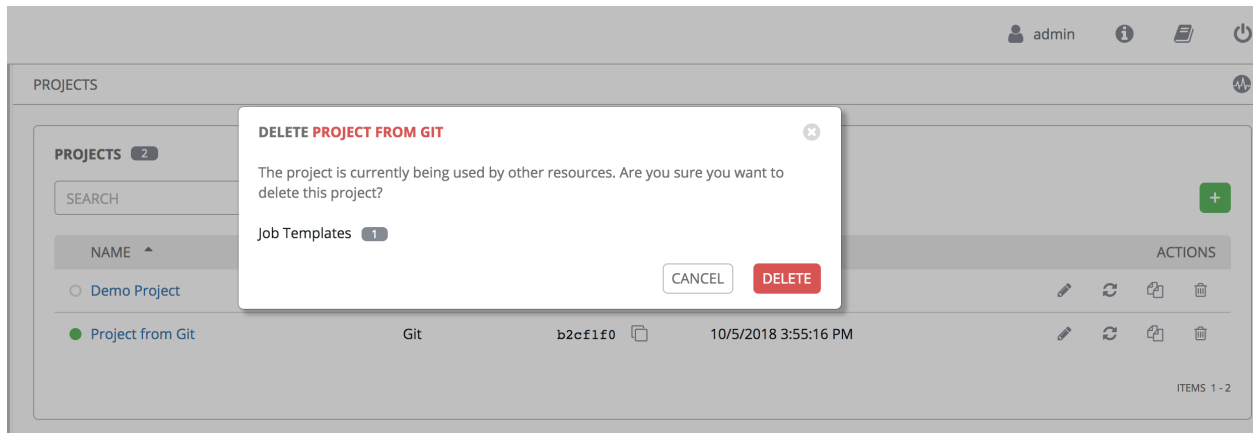
- **Pending** - The source control update has been created, but not queued or started yet. Any job (not just source control updates) will stay in pending until it's actually ready to be run by the system. Reasons for it not being ready because it has dependencies that are currently running so it has 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).

For each project listed, you can edit () project properties, copy the project attributes (), get the latest SCM revision (), or delete () the project, using the respective icons under the **Actions** column.


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:



13.1 Add a new project

To create a new project:

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

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.
- **SCM Type** - Select from the drop-down menu list an SCM type associated with this project. Refer to *Manage playbooks manually* and *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.

13.1.1 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:

The screenshot shows the 'NEW PROJECT' configuration page. At the top, there are tabs for 'DETAILS', 'PERMISSIONS', 'NOTIFICATIONS', 'JOB TEMPLATES', and 'SCHEDULES'. The 'DETAILS' tab is active. The form contains the following fields:

- * NAME:** Example
- DESCRIPTION:** Ansible example playbook
- * ORGANIZATION:** Honey Dog, Inc.
- ANSIBLE ENVIRONMENT:** Select Ansible Environment (dropdown menu)
- * SCM TYPE:** Manual (dropdown menu)
- PROJECT BASE PATH:** /var/lib/awx/projects

A warning message is displayed in a light gray box:


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

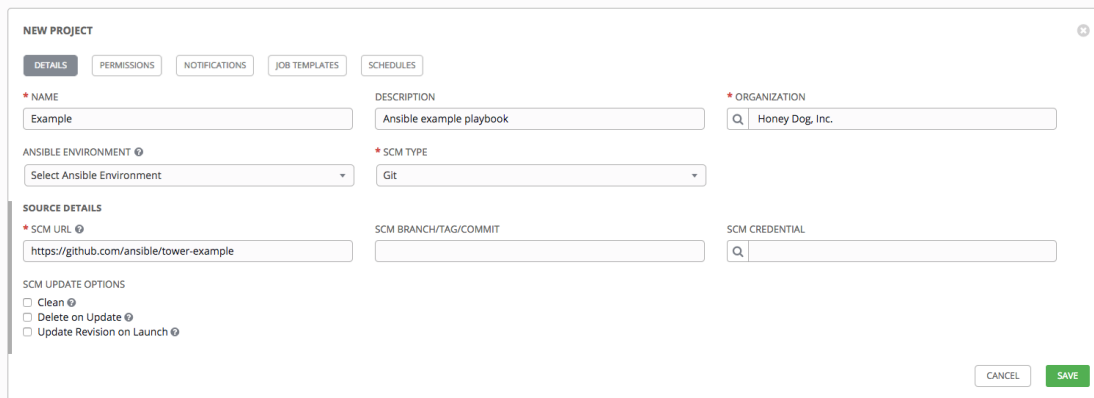
At the bottom right, there are 'CANCEL' and 'SAVE' buttons.

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.

13.1.2 Manage playbooks using Source Control

1. Select the appropriate option 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** - Optionally enter the SCM branch for Mercurial, or the SCM branch, tag, or revision for Git
- **Revision #** - Optionally enter the Revision # for Subversion
- **SCM Credential** - If authentication is required, select the appropriate SCM credential
- **SCM Update Options:**
 - **Clean** - Remove any local modifications prior to performing an update.
 - **Delete on Update** - Delete 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 on Launch** - Each time a job runs using this project, perform an update to the local repository prior to starting the job. 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.




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

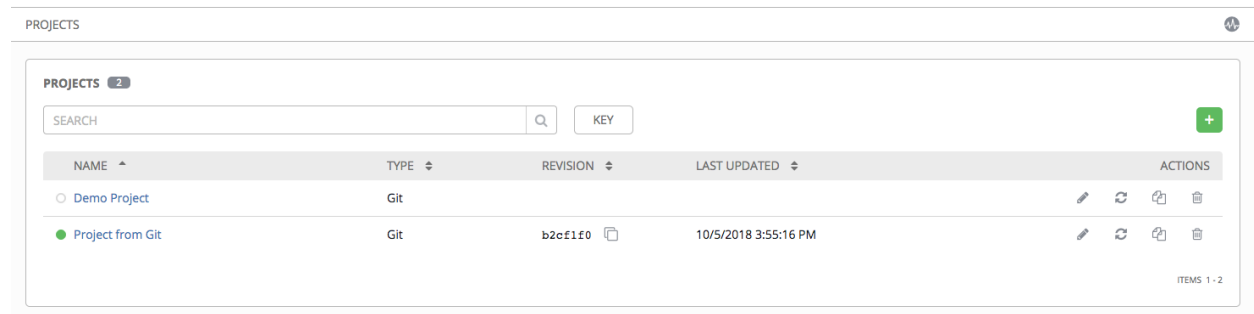
Tip: Using a Github link offers an easy way to use a playbook. To help get you started, use the helloworld.yml file available at: <https://github.com/ansible/tower-example.git>

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.

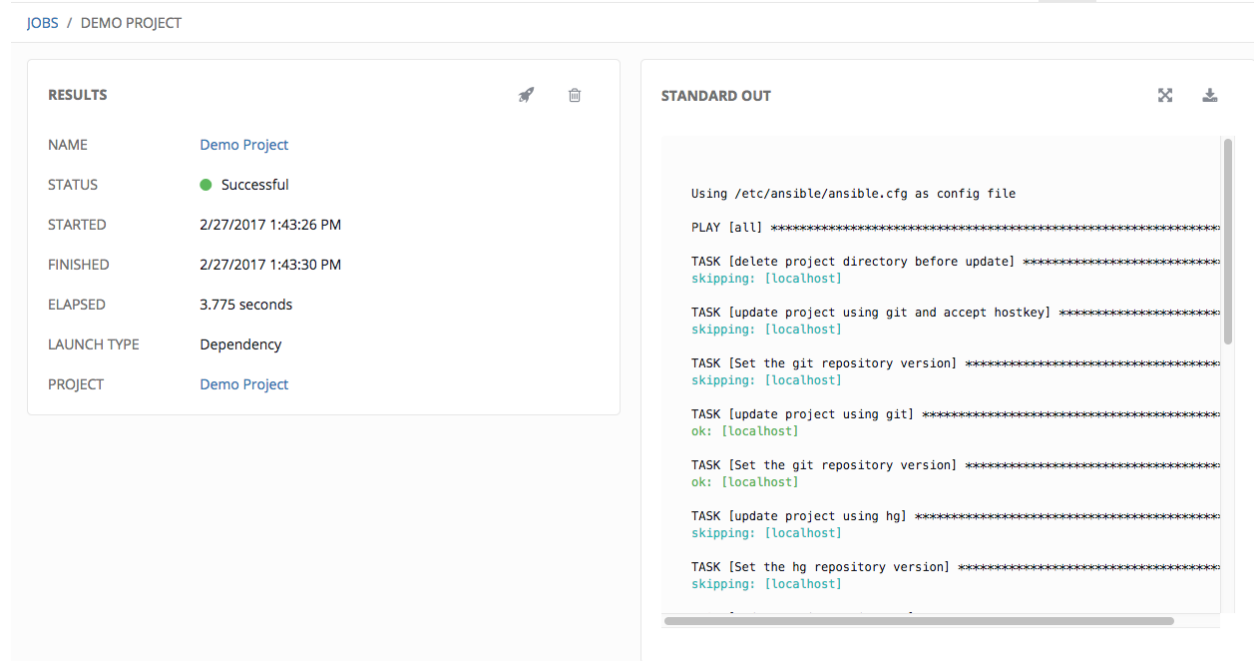
Updating projects from source control

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.



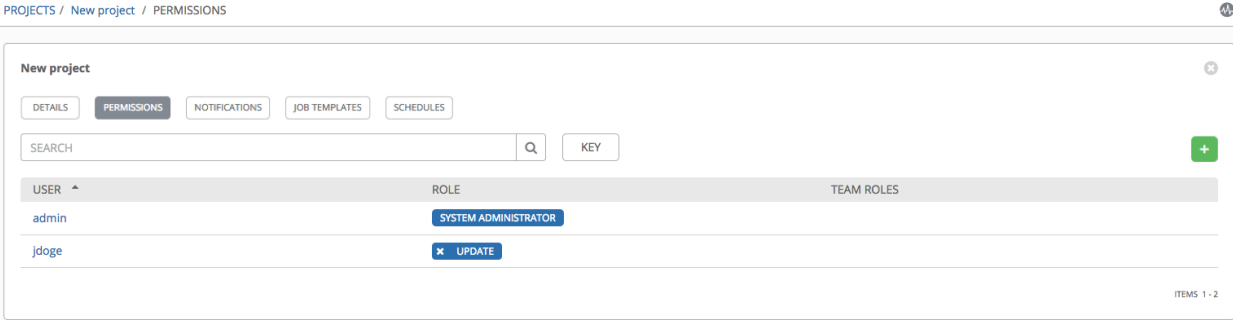
2. Click on the dot under **Status** (far left, beside the name of the Project) to get further details about the update process.



13.2 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**.



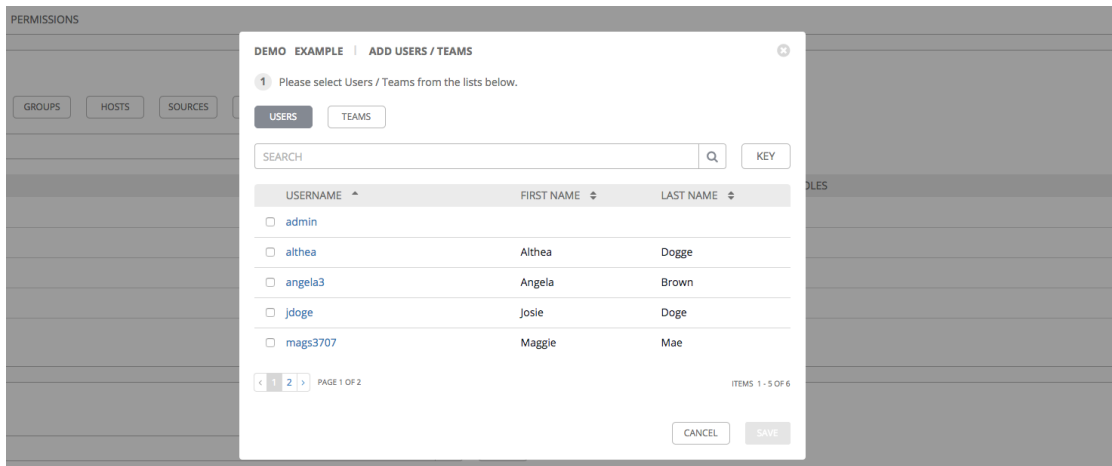
13.2.1 Add Permissions

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.

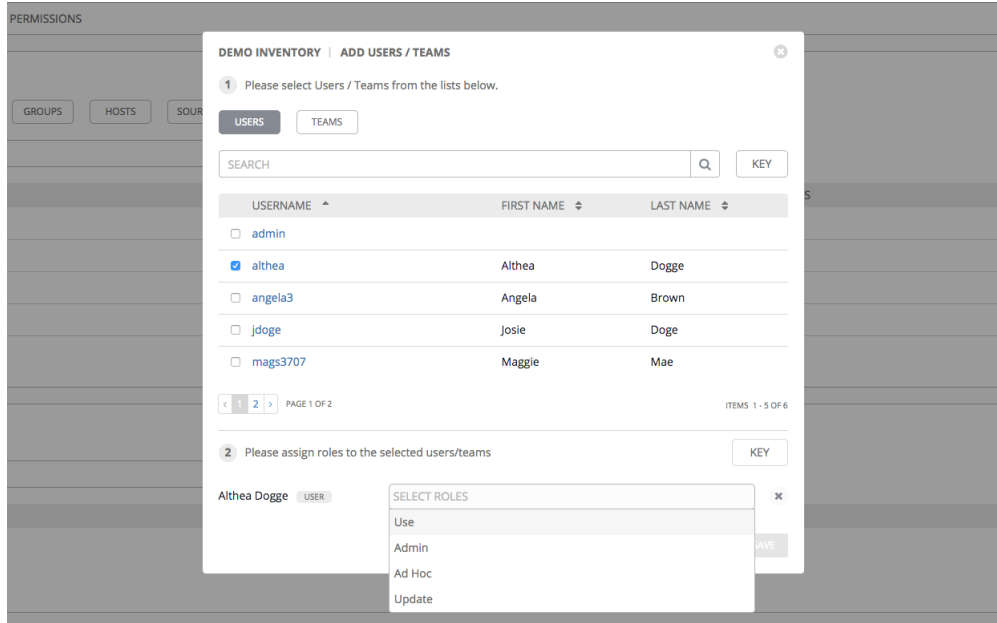


3. Specify the users or teams that will have access then assign them specific roles:

1. Click to select one or multiple checkboxes 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:

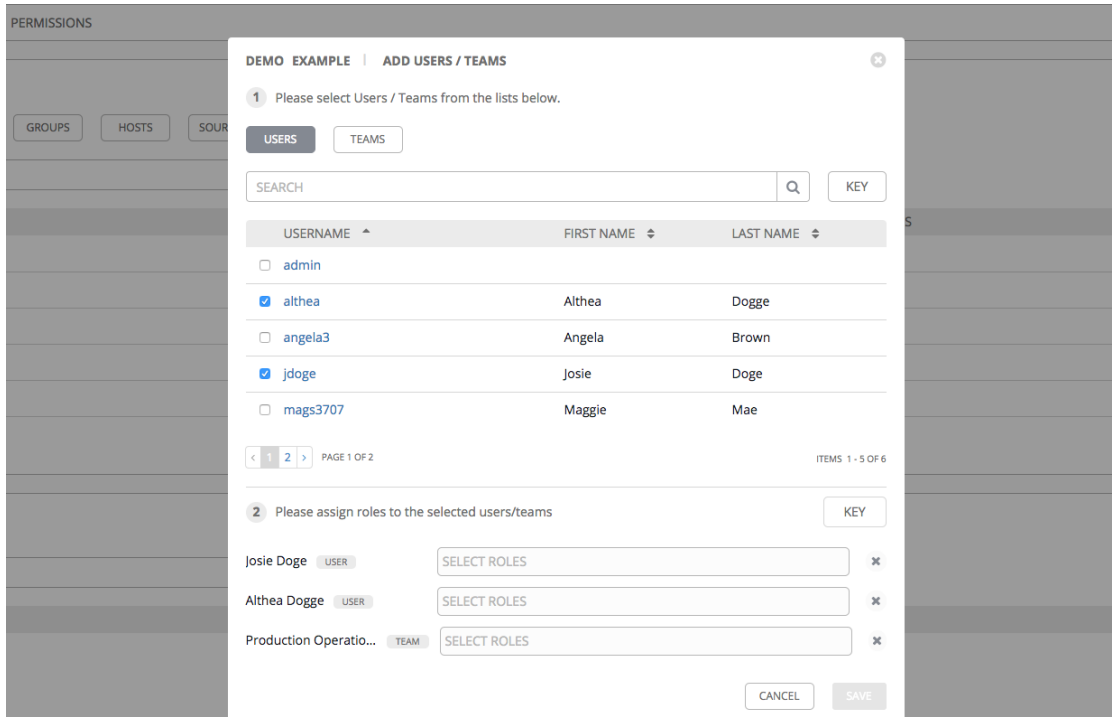
- **Admin** allows read, run, and edit privileges (applies to all resources)
- **Use** allows use of a resource in a job template (applies all resources except job templates)
- **Update** allows updating of project via the SCM Update (applies to projects and inventories)
- **Ad Hoc** allows use of Ad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

Tip: Use the **Key** button in the roles selection pane to display a description of each of the roles.

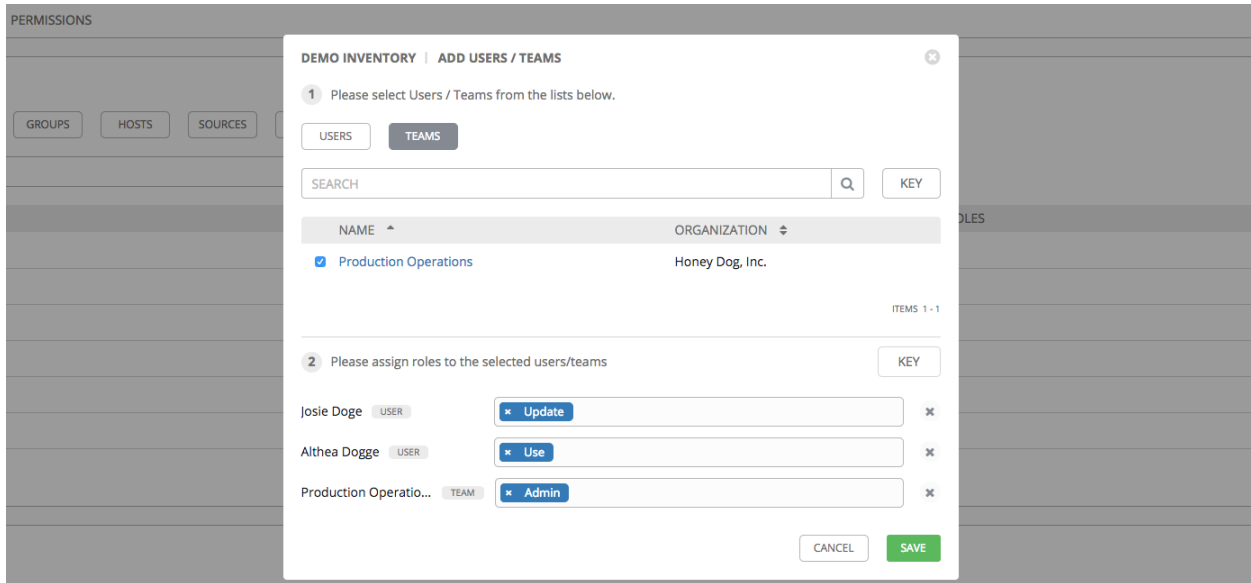
2. Select the role to apply to the selected user or team.

Note:

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



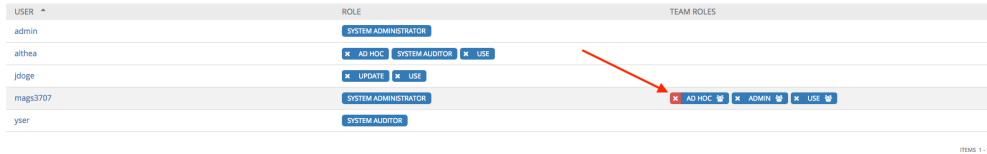
4. Review your role assignments for each user and team.



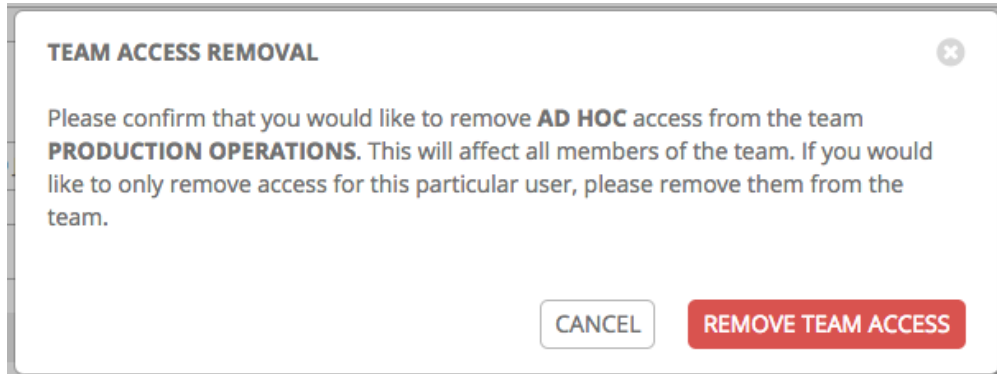
5. Click **Save** when done, and the Add Users/Teams window closes to display the updated roles assigned for each user and team.

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.

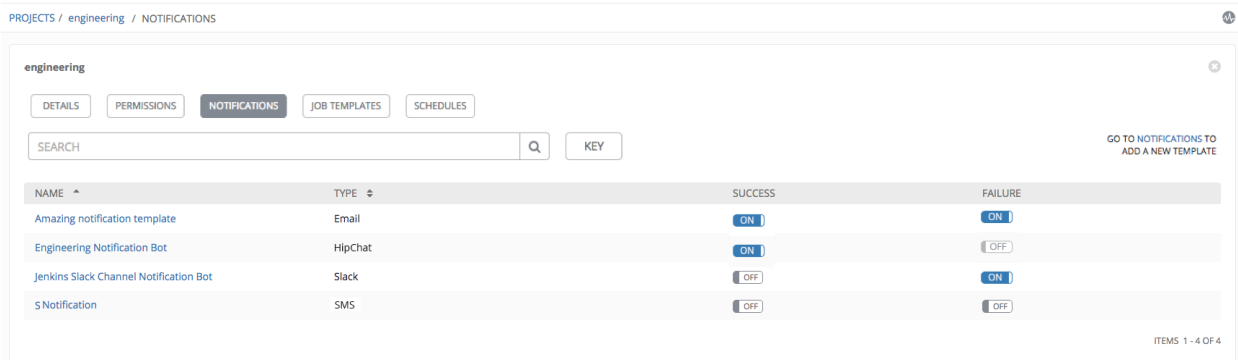


This launches a confirmation dialog, asking you to confirm the disassociation.



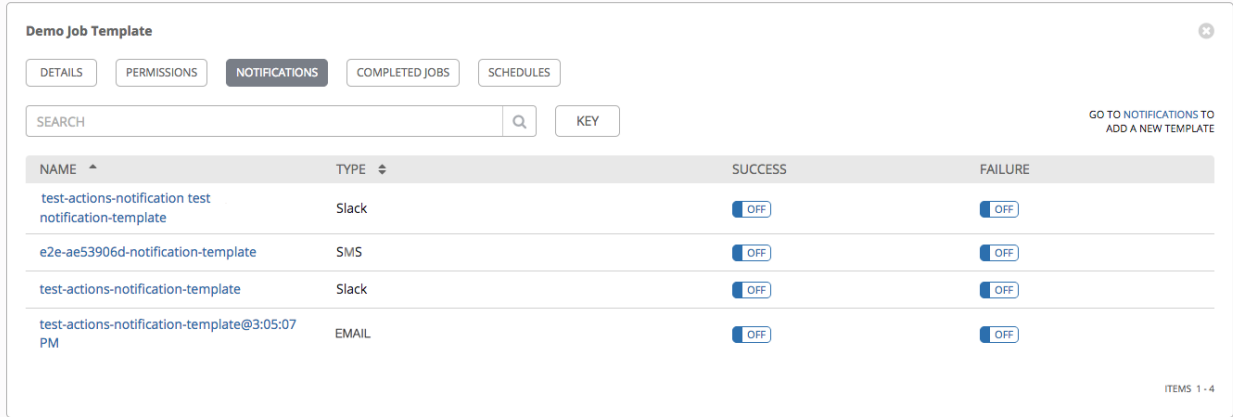
13.3 Work with Notifications

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



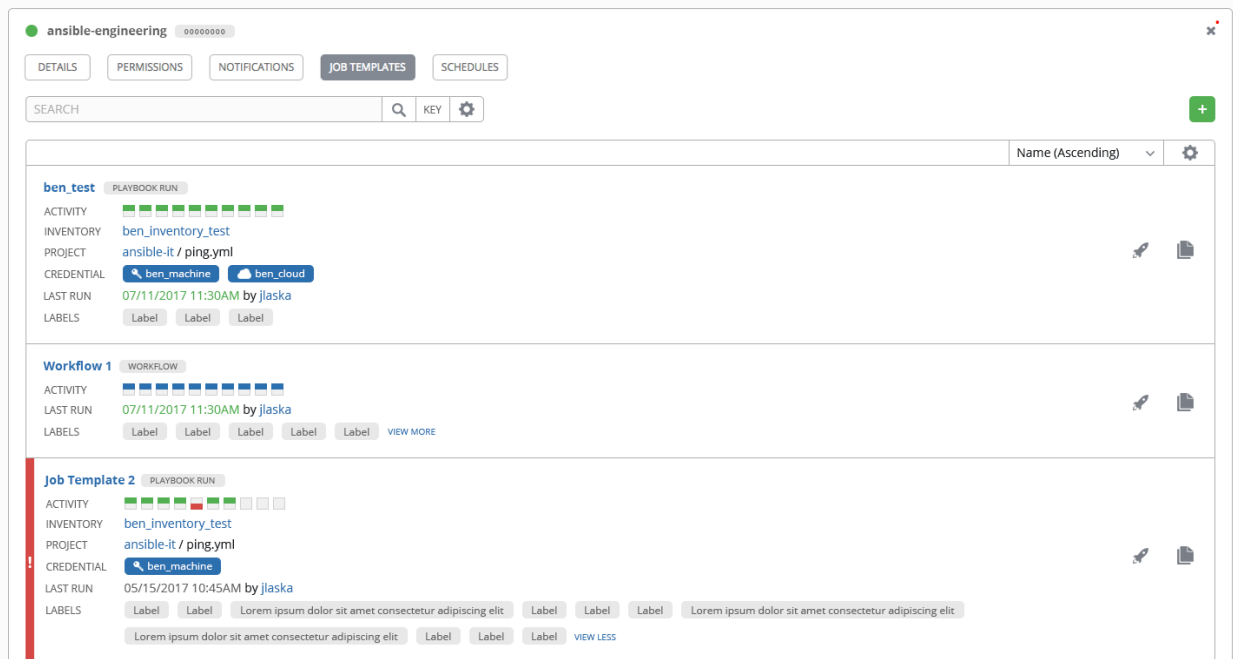
To create a new notification, click the **NOTIFICATIONS** link from the upper-right side of the notifications list view. If no notifications have been set up, click the **NOTIFICATIONS** link from above or inside the gray box to add a new notification to create a notification.

Refer to *Notifications* for more information.



13.4 Work with Job Templates

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



From this view, you can launch or copy the template configuration.

13.5 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 ^	NEXT RUN ^	FINAL RUN ^	ACTIONS
<input type="checkbox"/> Schedule 1	9/13/2018 1:00:00 AM	9/13/2018 1:00:00 AM	9/13/2018 1:00:00 AM	
<input type="checkbox"/> Schedule 2	10/23/2018 9:45:00 AM	10/23/2018 9:45:00 AM		
<input type="checkbox"/> Schedule 3	9/13/2018 11:15:00 PM	9/13/2018 11:15:00 PM	9/27/2018 11:15:00 PM	
<input type="checkbox"/> Schedule 4	10/31/2018 11:30:00 PM	10/31/2018 11:30:00 PM	10/31/2023 11:30:00 PM	

ITEMS 1 - 4

From this view, you can select schedules to edit, turn on or off, or select multiple schedules to delete.


This screen displays a list of the schedules that are currently available for the selected **Project**. The schedule list may be sorted and searched by **Name**.

The list of schedules includes:

- **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
- **Last Modified:** The last time this schedule was modified

13.5.1 Add a new schedule

To create a new schedule:

1. Click the  button, which opens the **Add Schedule** dialog.

NEW SCHEDULE ✕

* NAME

* START DATE (MM/DD/YYYY)

* START TIME (HH:MM:SS) : : UTC START TIME 11/19/2015 03:00:00 UTC

LOCAL TIME ZONE

REPEAT FREQUENCY

VARIABLES YAML JSON

1 ---

SCHEDULE SUMMARY

One time only.

OCCURRENCES (Limited to first 10) DATE FORMAT LOCAL TIME UTC

11/19/2015 00:00:00 EST

2. 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
- **UTC Start Time** - Calculated from Start Time + Local Time Zone
- **Repeat Frequency** - Appropriate scheduling options are displayed depending on the frequency you select

The **SCHEDULE DESCRIPTION** allows you to review the set schedule and a list of the scheduled occurrences in the selected Local Time Zone.

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.

3. Once done, click **Save**.

You can use the **ON/OFF** toggle button to stop an active schedule or activate a stopped schedule.

The schedules overview screen for the project also shows you when the first, next, and final runs are scheduled.

NAME	FIRST RUN	NEXT RUN	FINAL RUN	ACTIONS
<input checked="" type="checkbox"/> New schedule	3/2/2017 5:01:02 AM	3/2/2017 5:01:02 AM	3/2/2017 5:01:02 AM	

ITEMS 1 - 1 OF 1

13.5.2 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 install -r roles/requirements.yml -p ./roles/ --force
```

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.

For more information and examples on the syntax of the `requirements.yml` file, refer to [Advanced Control Over Role Requirements](#) 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.

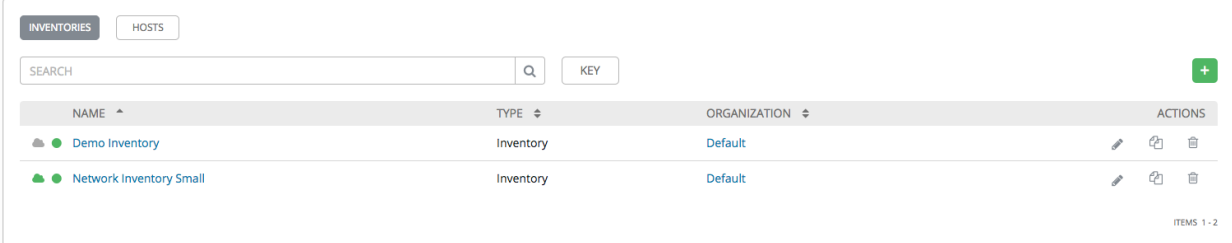
If you made changes in the settings file, be sure to restart services with the `ansible-tower-service restart` command after your changes have been saved.









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**.



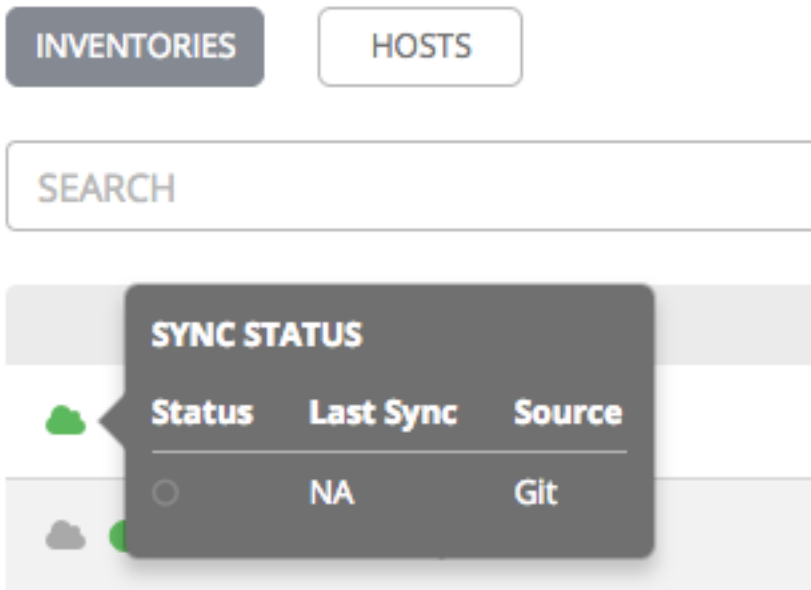
NAME ^	TYPE ↕	ORGANIZATION ↕	ACTIONS
 Demo Inventory	Inventory	Default	  
 Network Inventory Small	Inventory	Default	  


ITEMS 1 - 2

The list of Inventory details includes:

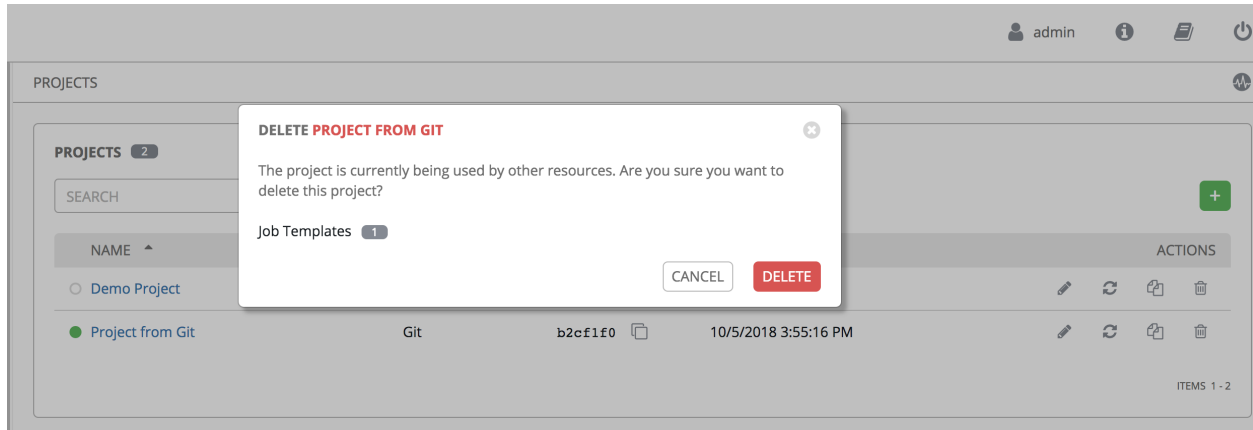
- **Inventory Sync** (

84



- **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 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 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 new field, `smart_inventories` that uses a membership lookup table that identifies a set of all the Smart Inventory a host is associated with. The memberships are generated by a task. The task is launched when:

- 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)

Note: The `update_host_smart_inventory_memberships` task is only run if the `AWX_REBUILD_SMART_MEMBERSHIP` is set to `True` (default is `False`).

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.

14.1.1 `host_filter` Search

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"
```

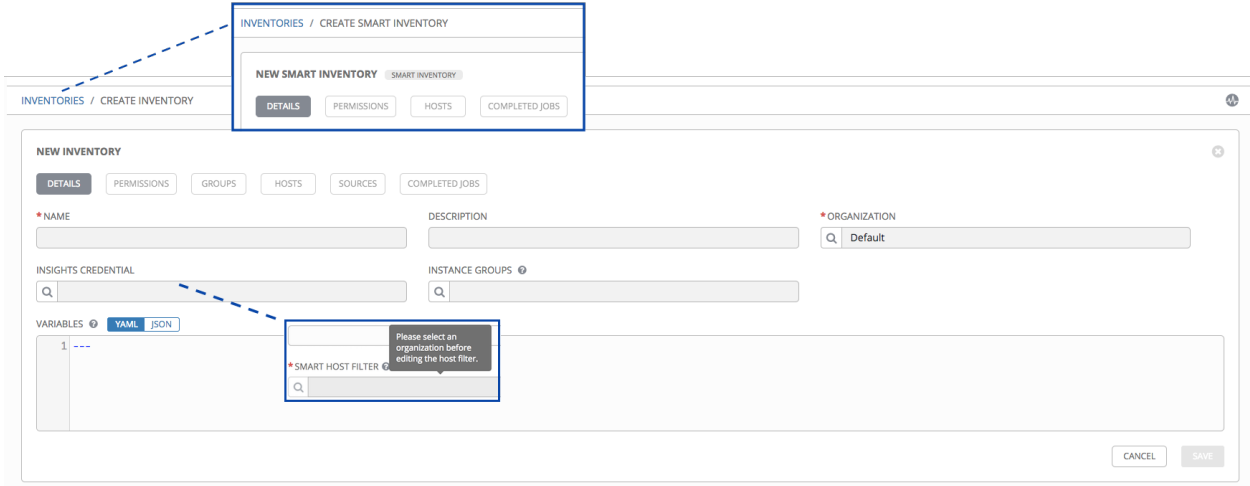
14.2 Add a new inventory

To create a new inventory or Smart Inventory:




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

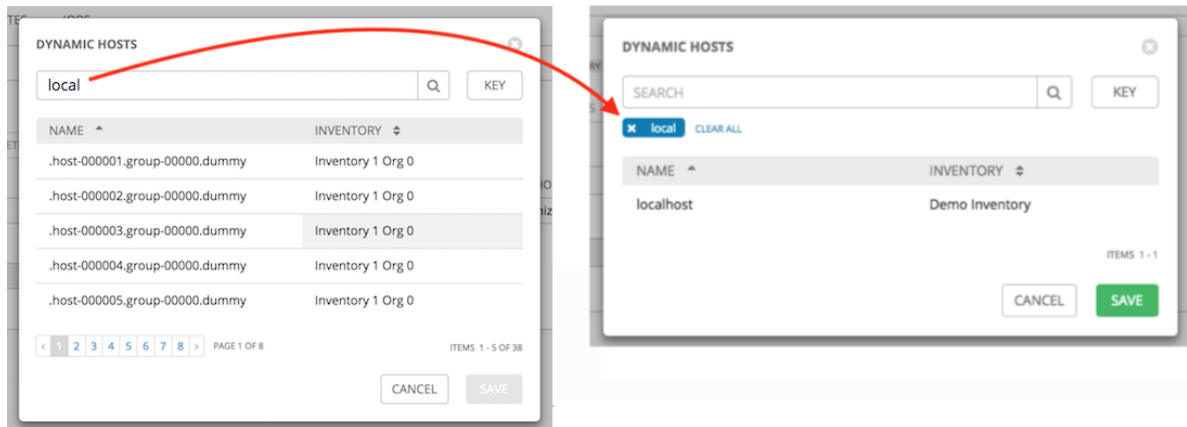
The type of inventory is identified by the labels and the row of tabs across the top of the create form.




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.

14.2.1 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.



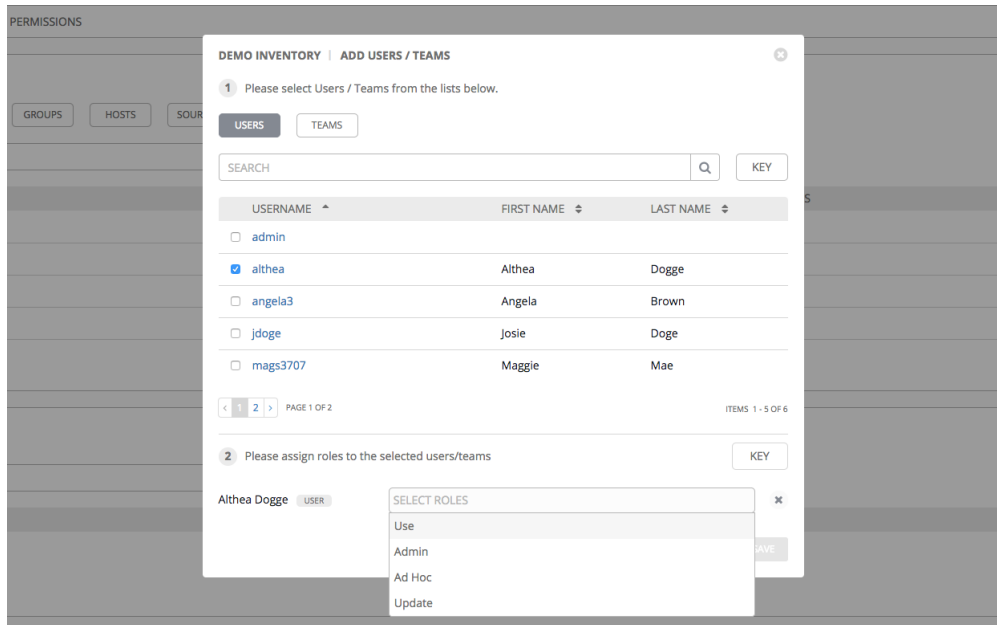
2. Click the  button to open the Add Users/Teams window.

USERNAME	FIRST NAME	LAST NAME
<input type="checkbox"/> admin		
<input type="checkbox"/> althea	Althea	Dogge
<input type="checkbox"/> angela3	Angela	Brown
<input type="checkbox"/> jdodge	Josie	Doge
<input type="checkbox"/> mags3707	Maggie	Mae

3. Specify the users or teams that will have access then assign them specific roles:
 1. Click to select one or multiple checkboxes 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:

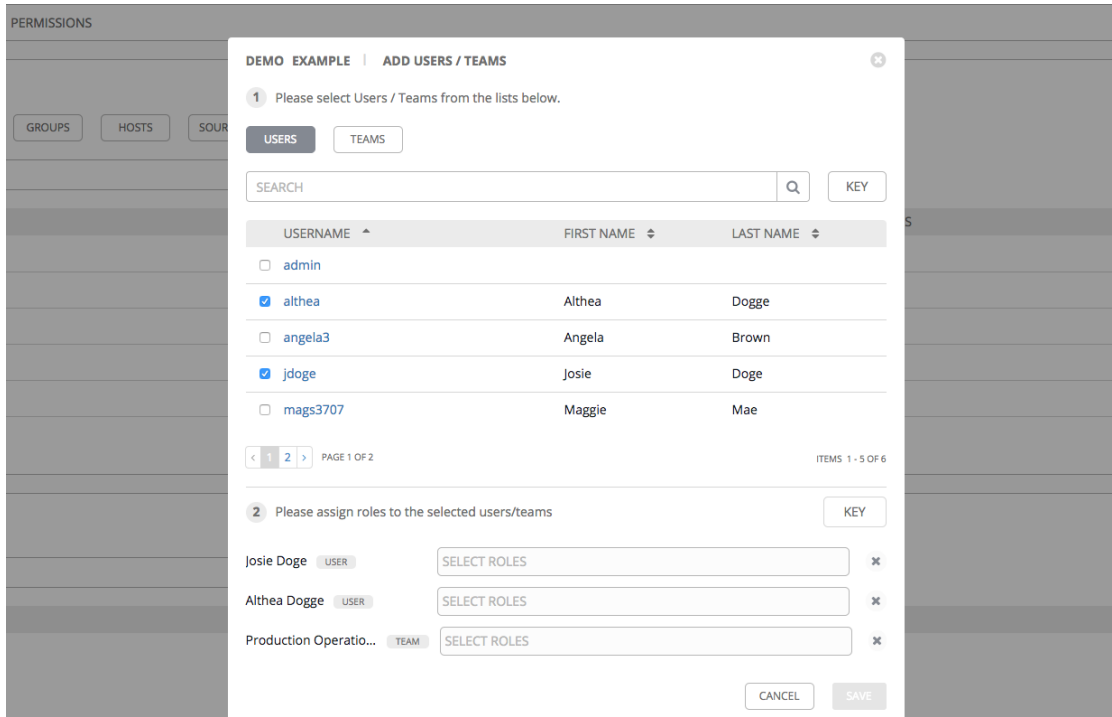
- **Admin** allows read, run, and edit privileges (applies to all resources)
- **Use** allows use of a resource in a job template (applies all resources except job templates)
- **Update** allows updating of project via the SCM Update (applies to projects and inventories)
- **Ad Hoc** allows use of Ad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

Tip: Use the **Key** button in the roles selection pane to display a description of each of the roles.

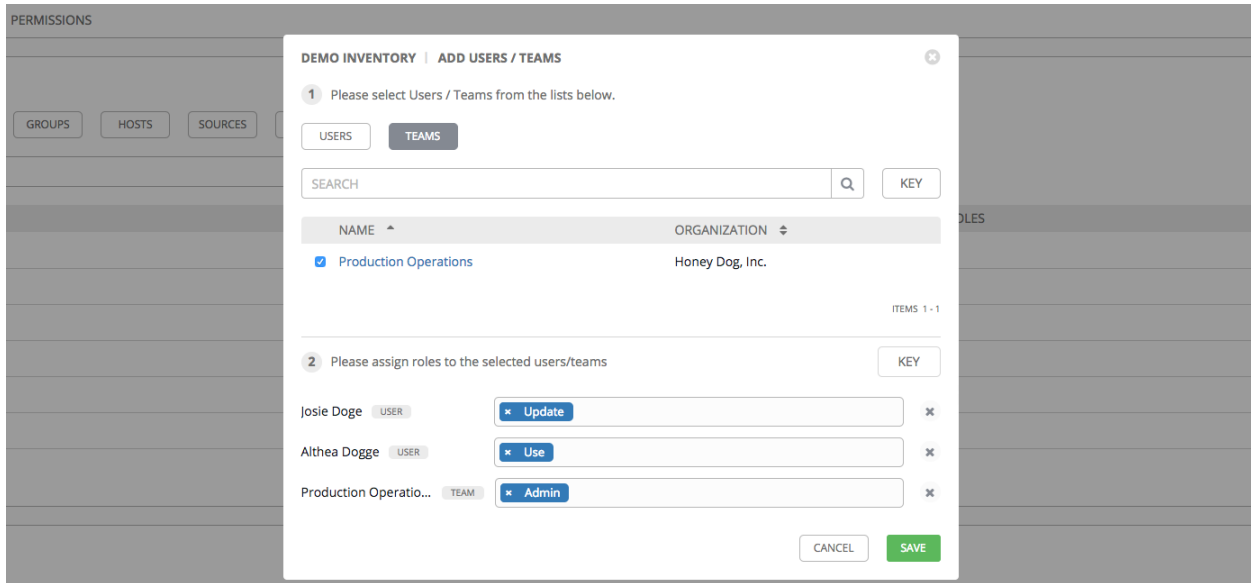
2. Select the role to apply to the selected user or team.
-

Note:

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



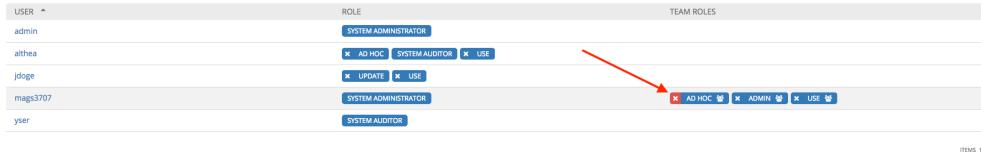
4. Review your role assignments for each user and team.



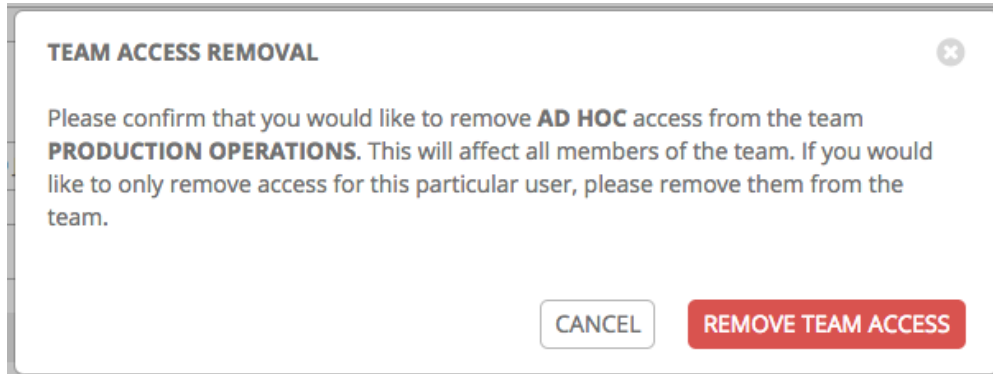
5. Click **Save** when done, and the Add Users/Teams window closes to display the updated roles assigned for each user and team.

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

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.



14.2.2 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: Starting in Ansible Tower 3.2, 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**.

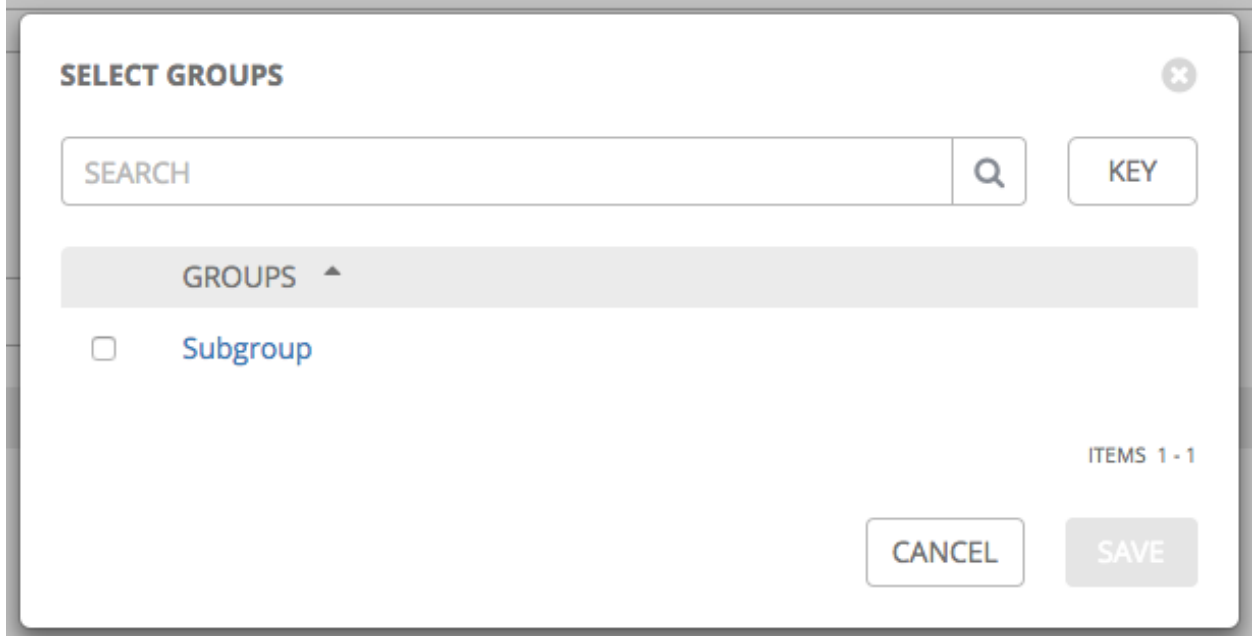
Add groups within groups

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**.

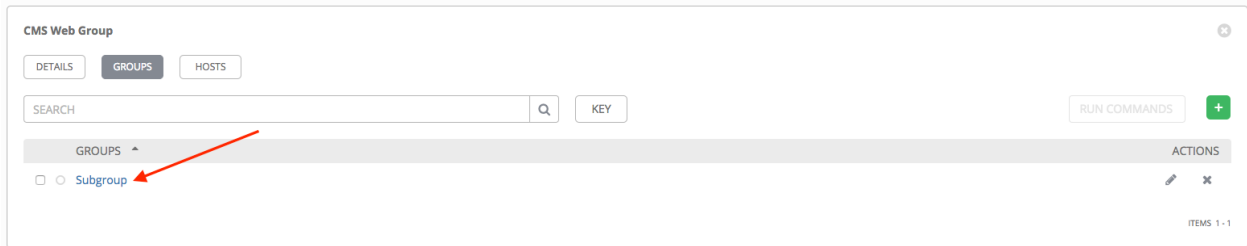
The **Create Group** window closes and the newly created group displays as an entry in the list of groups associated with the group that it was created for.

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.




14.2.3 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.

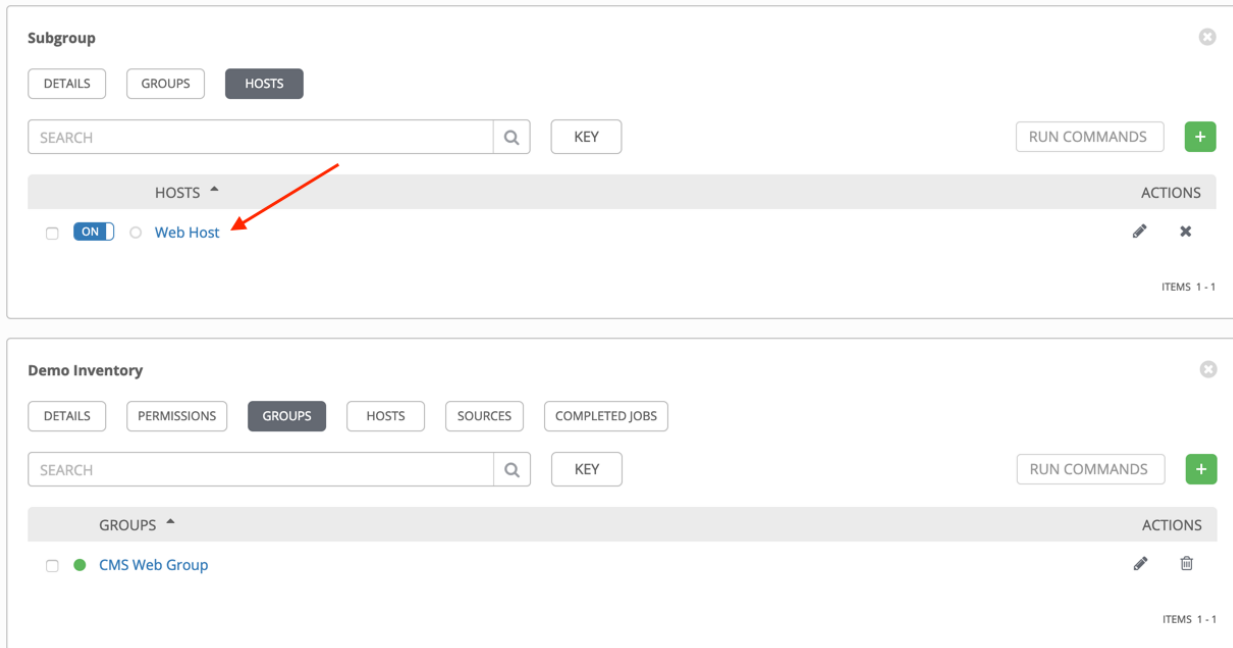
3. If creating a new host, select the  button to specify whether or not to include this host while running jobs.

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

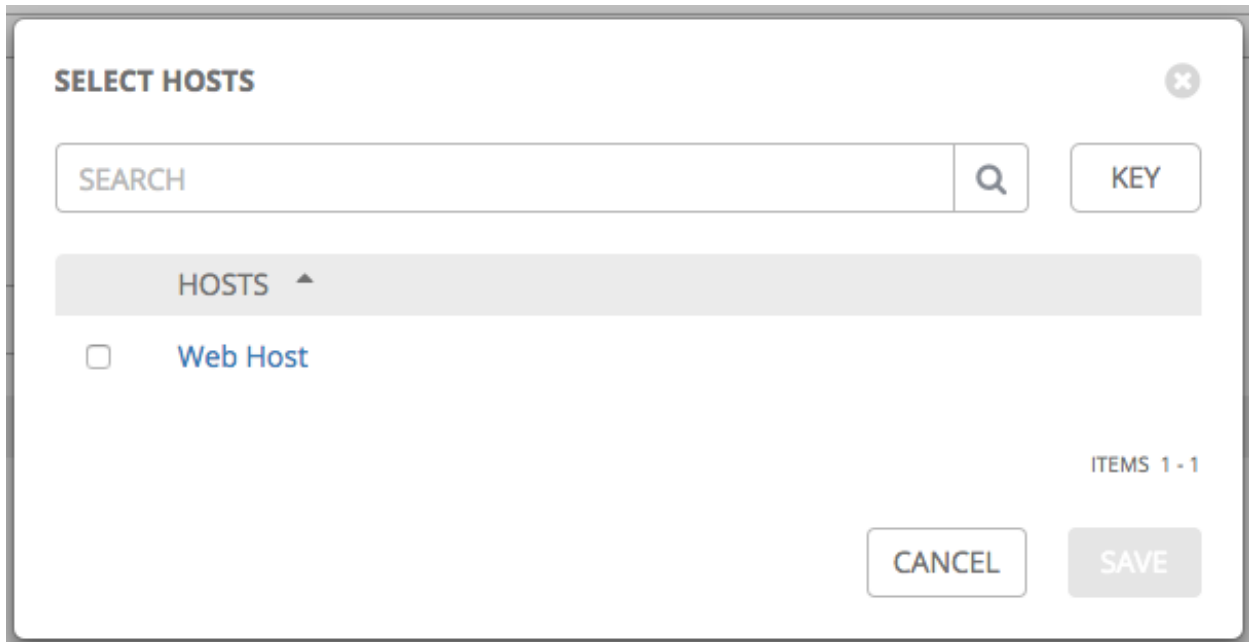
- **Host Name:** Required
- **Description:** Enter an arbitrary description as appropriate (optional)
- **Variables:** Enter definitions and values to be applied to all hosts in this group. Enter variables using either JSON or YAML syntax. Use the radio button to toggle between the two.

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.

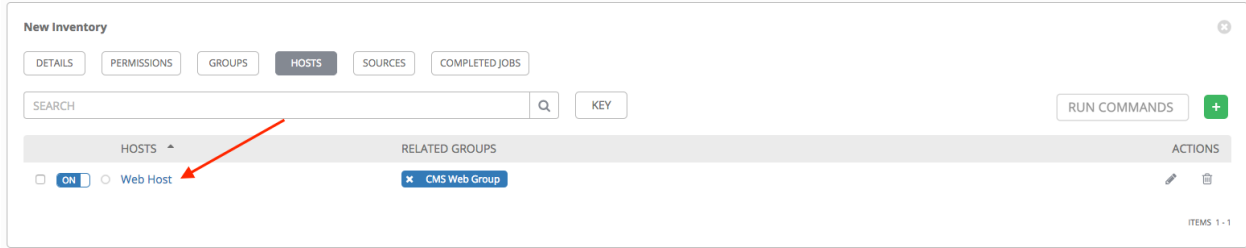


If you chose to add an existing host, available hosts will appear in a separate selection window.

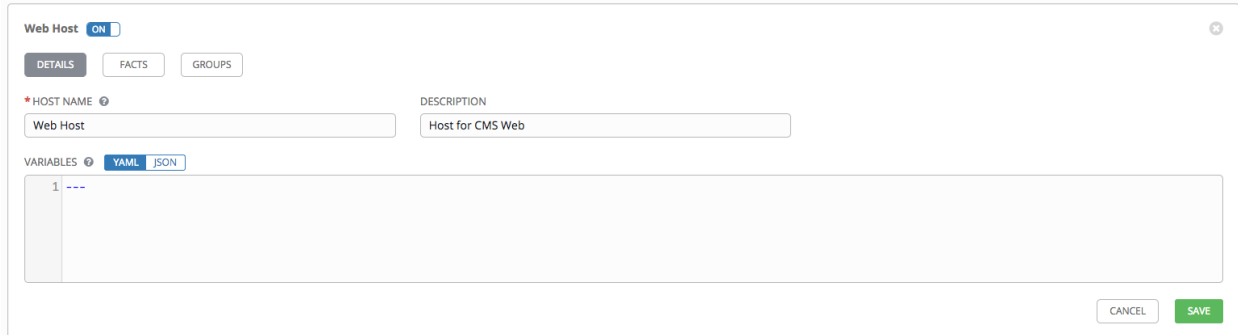


Once a host is selected, it displays as an entry in the list of hosts associated with the group.

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.

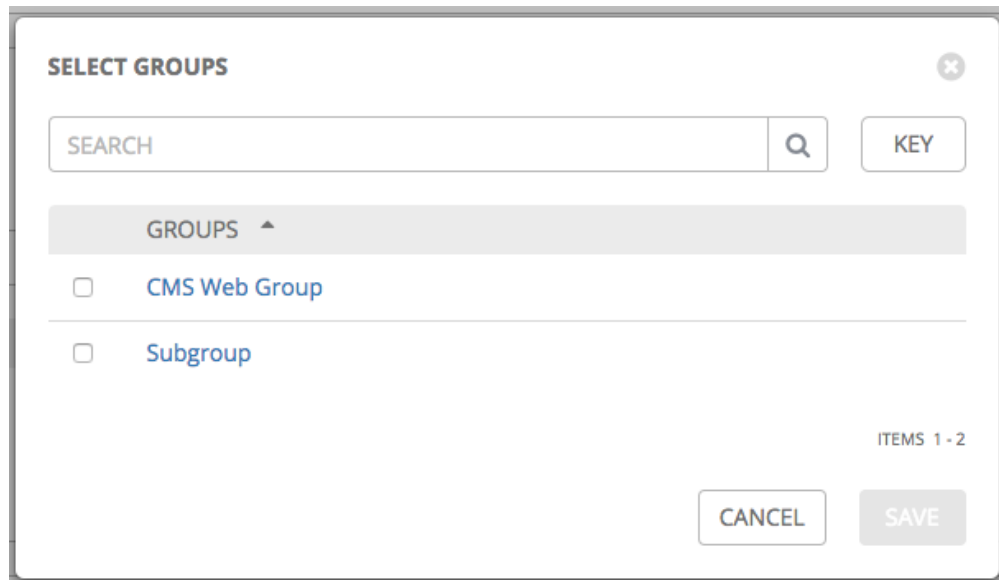


7. Click the **Facts** tab to input facts you want to gather. Refer to the [Fact Caching](#) section for more information about facts.

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

1. Click the  button to associate the host with an existing group.

Available groups appear in a separate selection window.



2. 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.

14.2.4 Add source

Inventory sources are no longer associated with groups. Prior to Ansible Tower 3.2, 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.

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 which matches the credential type against which a host can be entered. Refer to the [Credential Sources](#) section for more information about each source and details for entering the appropriate information.

Note: Starting with Ansible Tower version 3.2, support for Rackspace Cloud Servers was discontinued.

4. You can configure the the level of output on any inventory source’s update jobs by selecting the appropriate option from the **Verbosity** drop-down menu.

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

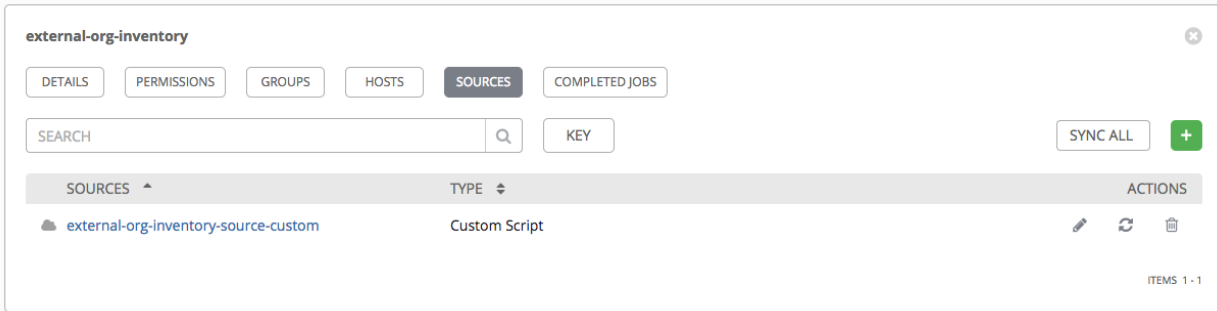
- **Overwrite:** Refer to the on-screen tooltip () for information. In order to guarantee consistent behavior after 3.2 migration, do not set to `True`.
- **Overwrite Variables:** Refer to the on-screen tooltip () for information.
- **Update on Launch:** Each time a job runs using this inventory, refresh 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.

Note: 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.

6. 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.

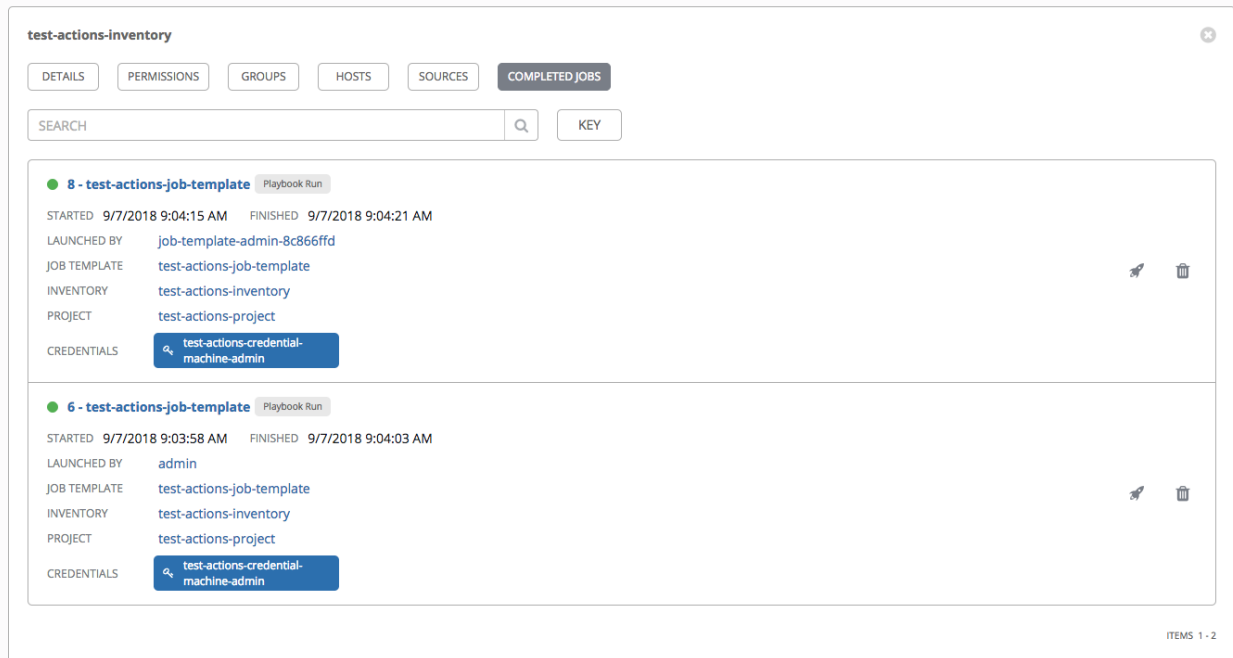


7. To configure notifications for the source, click the **Notifications** tab.

1. If notifications are already set up, select a notification preference.
2. If notifications have not been set up, refer to *Notifications* for more information.

14.2.5 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.



Smart Host Filter

You can use a search filter to populate hosts for an inventory. This feature was introduced in Ansible Tower 3.2 utilizing 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
```

Credential Sources

Topics:


- *Sourced from a Project*
- *Amazon Web Services EC2*
- *Google Compute Engine*
- *Microsoft Azure Classic (deprecated)*
- *Microsoft Azure Resource Manager*
- *VMware vCenter*
- *Red Hat Satellite 6*
- *Red Hat CloudForms*
- *OpenStack*
- *Red Hat Virtualization*

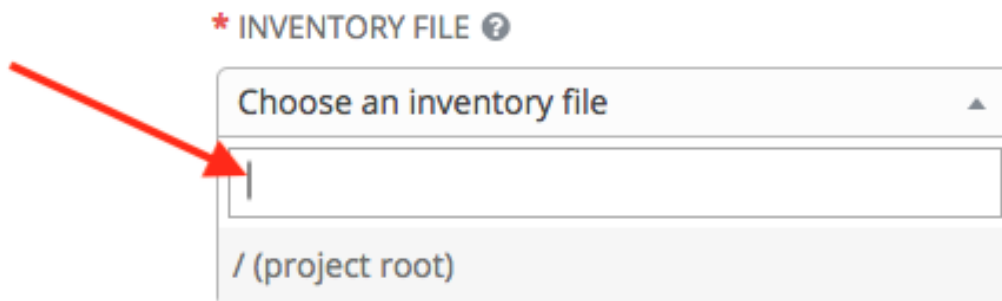
- *Ansible Tower*
- *Custom Script*

Choose a source which matches the credential type against which a host can be entered.

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:** 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. 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 Change** option to refresh the inventory from the selected source after every project update where the SCM revision changes before executing job tasks.
4. In order to pass to the custom inventory script, you can optionally set environment variables in the **Environment Variables** field.

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:** 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](#).

- **Regions:** Click on the regions field to see a list of regions for your cloud provider. You can select multiple regions, or choose “All” to include all regions. Tower will only be updated with Hosts associated with the selected regions.
- **Instance Filters:** Rather than importing your entire Amazon EC2 inventory, filter the instances returned by the inventory script based on a variety of metadata. Hosts are imported if they match any of the filters entered here.

Examples:

- To limit to hosts having the tag TowerManaged: Enter `tag-key=TowerManaged`
- To limit to hosts using either the key-name staging or production: Enter `key-name=staging, key-name=production`
- To limit to hosts where the Name tag begins with test: Enter `tag:Name=test*`

For more information on the filters that can be used here, refer to the [Describe Instances](#) documentation at Amazon.

- **Only Group By:** By default, Tower creates groups based on the following Amazon EC2 parameters:
 - Availability Zones
 - Image ID
 - Instance ID
 - Instance Type

- Key Name
- Region
- Security Group
- Tags (by name)
- VPC ID
- Tag None

If you do not want all these groups created, select from the dropdown the list of groups that you would like created by default. You can also select Instance ID to create groups based on the Instance ID of your instances.

3. Use the **Source Variables** field to override variables found in `ec2.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 ec2.ini in the Ansible GitHub repo](#).

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 additional fields. Enter the following details:
 - **Credential:** Required. Choose from an existing Credential. For more information, refer to [Credentials](#).
 - **Regions:** Click on the regions field to see a list of regions for your cloud provider. You can select multiple regions, or choose “All” to include all regions. Tower will only be updated with Hosts associated with the selected regions.

The screenshot shows the 'CREATE SOURCE' form for Google Compute Engine. It has two tabs: 'DETAILS' (selected) and 'NOTIFICATIONS'. The form contains the following fields and options:

- * NAME:** Text input field containing 'Source from GCE'.
- DESCRIPTION:** Text input field.
- * SOURCE:** Dropdown menu with 'Google Compute Engine' selected.
- SOURCE DETAILS:**
 - * CREDENTIAL:** Text input field with a search icon and 'Inventory Credential'.
 - REGIONS:** Multi-select dropdown menu with 'All' selected.
 - VERBOSITY:** Dropdown menu with '1 (INFO)' selected.
- UPDATE OPTIONS:**
 - Overwrite
 - Overwrite Variables
 - Update on Launch

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

Microsoft Azure Classic (deprecated)

1. To configure a Azure-sourced inventory, select **Microsoft Azure Classic (deprecated)** from the Source field.
2. The Create Source window expands with additional fields. Enter the following details:
 - **Credential:** Required. Choose from an existing Credential. For more information, refer to [Credentials](#).
 - **Regions:** Click on the regions field to see a list of regions for your cloud provider. You can select multiple regions, or choose “All” to include all regions. Tower will only be updated with Hosts associated with the selected regions.

The screenshot shows the 'CREATE SOURCE' form for Microsoft Azure Classic (deprecated). It has two tabs: 'DETAILS' (selected) and 'NOTIFICATIONS'. The form contains the following fields and options:

- * NAME:** Text input field containing 'Source from Azure Classic'.
- DESCRIPTION:** Text input field.
- * SOURCE:** Dropdown menu with 'Microsoft Azure Classic (deprecated)' selected.
- SOURCE DETAILS:**
 - * CREDENTIAL:** Text input field with a search icon and 'Inventory Credential'.
 - REGIONS:** Multi-select dropdown menu with 'US East' selected.
 - VERBOSITY:** Dropdown menu with '1 (INFO)' selected.
- UPDATE OPTIONS:**
 - Overwrite
 - Overwrite Variables
 - Update on Launch

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

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 additional fields. Enter the following details:
 - **Credential:** Required. Choose from an existing Credential. For more information, refer to [Credentials](#).
 - **Regions:** Click on the regions field to see a list of regions for your cloud provider. You can select multiple regions, or choose “All” to include all regions. Tower will only be updated with Hosts associated with the selected regions.

CREATE SOURCE
✕

DETAILS

NOTIFICATIONS

*** NAME**

DESCRIPTION

*** SOURCE**

Microsoft Azure Resource Manager

SOURCE DETAILS

*** CREDENTIAL**

REGIONS ⓘ

VERBOSITY ⓘ

1 (INFO)

UPDATE OPTIONS

Overwrite ⓘ

Overwrite Variables ⓘ

Update on Launch ⓘ

CANCEL

SAVE

VMware vCenter

1. To configure a VMWare-sourced inventory, select **VMware vCenter** from the Source field.
2. The Create Source window expands with additional fields. Enter the following details:
 - **Credential:** Required. Choose from an existing credential (for more information, refer to [Credentials](#)).
 - **Instance Filters:** Rather than importing your entire VMWare inventory, filter the instances returned by the inventory script based on a variety of metadata. Hosts are imported if they match any of the filters entered here.

For more information on the filters that can be used here, refer to the [Quick Filters Available for vSphere Objects](#) documentation at VMware.

- **Only Group By:** By default, Tower creates groups based on user-specified VMWare parameters. For example, enter `Instance ID` to create groups based on the Instance ID of your instances.
3. Use the **Source Variables** field to override variables found in `vmware.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 `vmware_inventory.ini` in the [Ansible GitHub repo](#).

Note: The inventory script for VMware was updated in Ansible Tower 3.1.2 to allow configuration of the `host_filters` or `groupby_patterns` parameter. Specify those values in the **Source Variables** text field of the Create Group screen or Edit Group screen. For example:

SOURCE VARIABLES ⓘ

YAML

JSON

```

1 ---
2 host_filters: "{{ config.guestid == 'rhel7_64Guest' }}"
3 groupby_patterns: "{{ guest.guestid }},{{ 'templates' if config.template else 'guests' }}"
            
```

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 additional fields.
 - **Credential:** Required. Choose from an existing credential (for more information, refer to [Credentials](#)).
 - Use the **Source Variables** field to override variables found in `foreman.ini` and used by the inventory update script.

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`

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 [foreman.ini](#) in the [Ansible GitHub repo](#).

Red Hat CloudForms

1. To configure a Red Hat CloudForms-sourced inventory, select **Red Hat CloudForms** from the Source field.
2. The Create Source window expands with additional fields. Enter the following details:
 - **Credential:** Required. Choose from an existing credential (for more information, refer to [Credentials](#)).
 - Use the **Source Variables** field to override variables found in `cloudforms.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 cloudforms.ini in the Ansible GitHub repo](#).

CREATE SOURCE

DETAILS NOTIFICATIONS

* NAME: Source from RH CloudForms DESCRIPTION: DESCRIPTION * SOURCE: Red Hat CloudForms

SOURCE DETAILS

* CREDENTIAL: Inventory Credential VERBOSITY: 1 (INFO) UPDATE OPTIONS: Overwrite Overwrite Variables Update on Launch

SOURCE VARIABLES: YAML JSON

1 ---

CANCEL SAVE

OpenStack

1. To configure an OpenStack-sourced inventory, select **OpenStack** from the Source field.
2. The Create Source window expands with additional fields. Enter the following details:
 - **Credential:** Required. Choose from an existing credential (for more information, refer to [Credentials](#)).
 - 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.yml in the Ansible GitHub repo](#).

CREATE SOURCE

DETAILS NOTIFICATIONS

* NAME: Source from OpenStack DESCRIPTION: DESCRIPTION * SOURCE: OpenStack

SOURCE DETAILS

* CREDENTIAL: Inventory Credential VERBOSITY: 1 (INFO) UPDATE OPTIONS: Overwrite Overwrite Variables Update on Launch

SOURCE VARIABLES: YAML JSON

1 ---

CANCEL SAVE

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 additional fields. The **Credential** is required. Choose from an existing credential (for more information, refer to *Credentials*).

Ansible Tower

1. To configure a Ansible Tower-sourced inventory, select **Ansible Tower** from the Source field.
2. The Create Source window expands with additional fields. Enter the following details:
 - **Credential:** Required. Choose from an existing credential (for more information, refer to *Credentials*).
 - **Instance Filters:** Rather than importing your entire Tower inventory, filter the instances by an inventory ID/name; then the inventory script would return that inventory from the other Tower instance.

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.

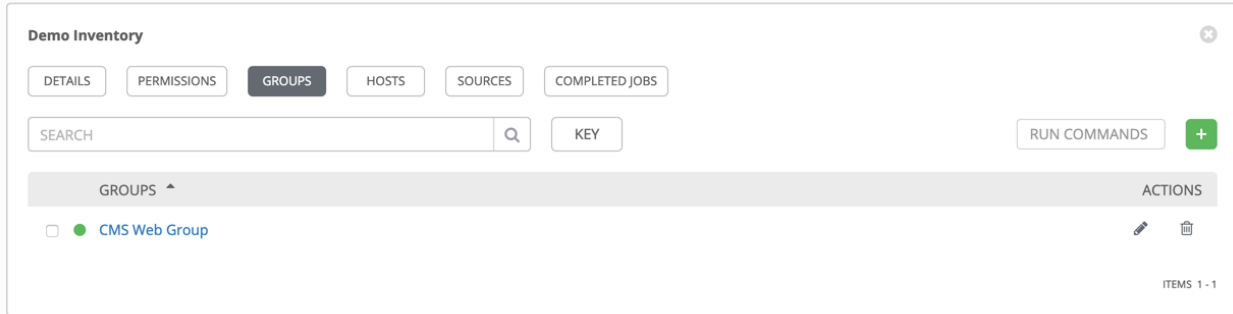
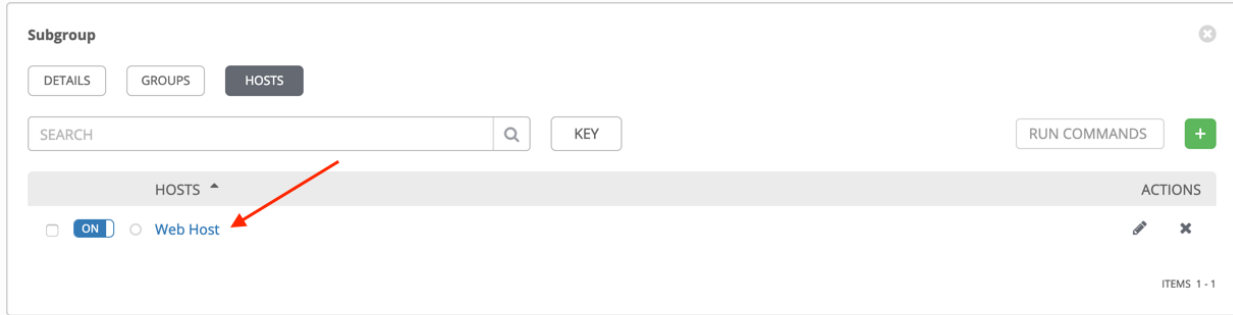
- **Custom Inventory Script:** Required. Choose from an existing Inventory Script (for more information, refer to [Custom Inventory Scripts](#)).
- **Environment Variables:** Set variables in the environment to be used by the inventory update script. The variables would be specific to the script that 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*.

14.3 Running Ad Hoc Commands

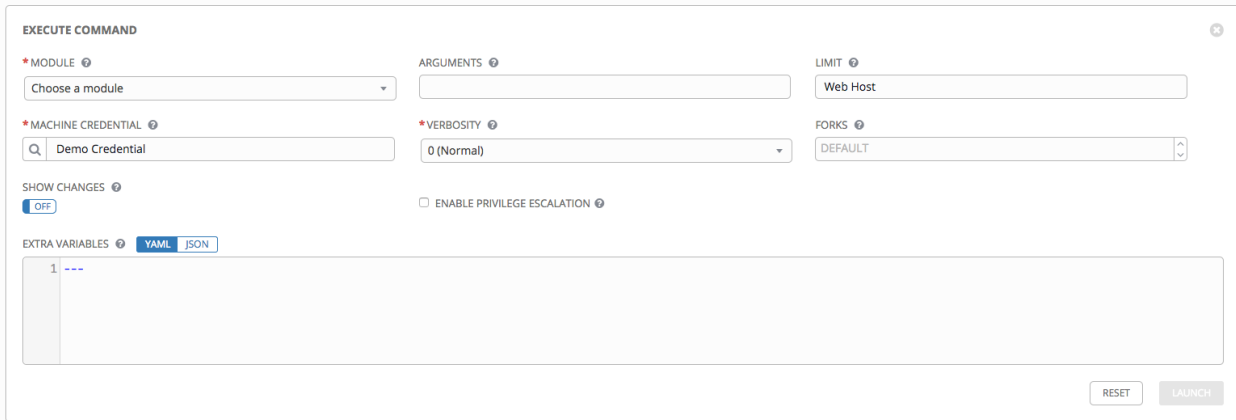
To run an ad hoc command:

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.



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

The results display in the Job Results and Standard Out window.

RESULTS	
NAME	ping
STATUS	Successful
STARTED	11/14/2017 10:00:44 AM
FINISHED	11/14/2017 10:00:47 AM
ELAPSED	2.991 seconds
INVENTORY	Demo Inventory
CREDENTIAL	Demo Credential
LAUNCHED BY	admin
FORKS	0
LIMIT	localhost
VERBOSITY	0
EXTRA VARIABLES	1 ---


```

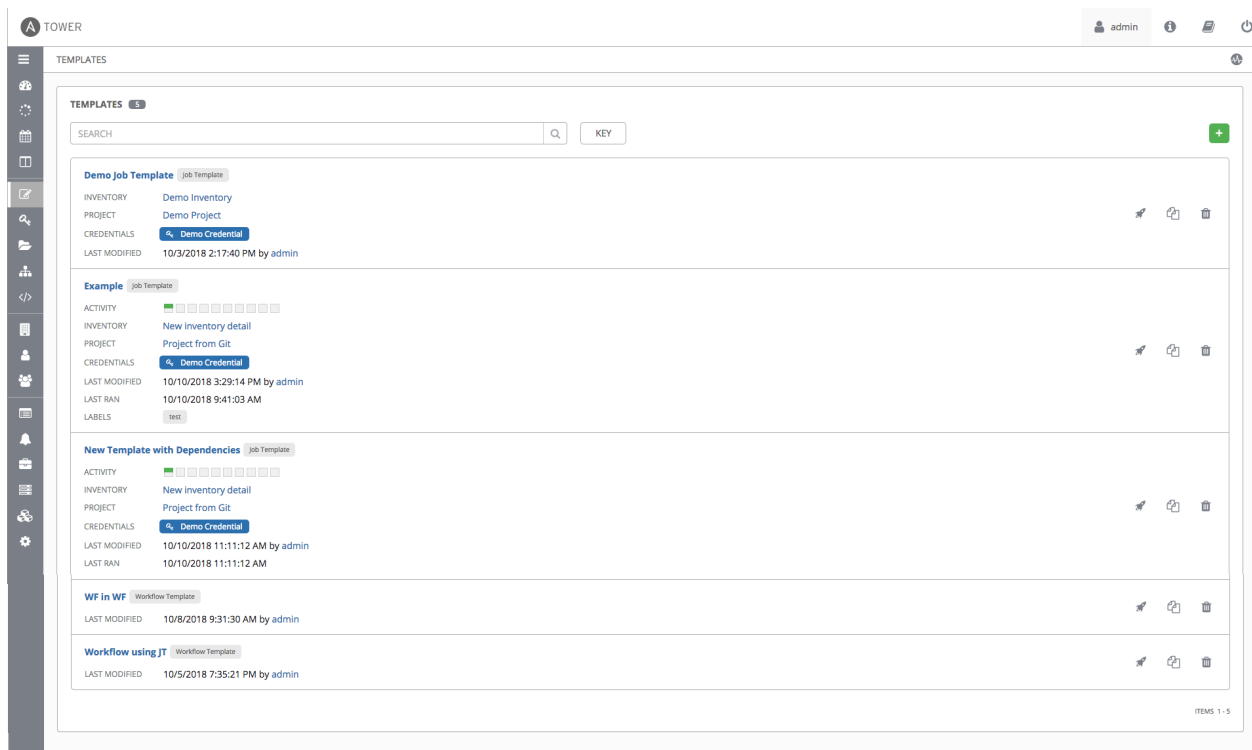
localhost | SUCCESS => {
  "changed": false,
  "failed": false,
  "ping": "pong"
}
    
```

JOB TEMPLATES

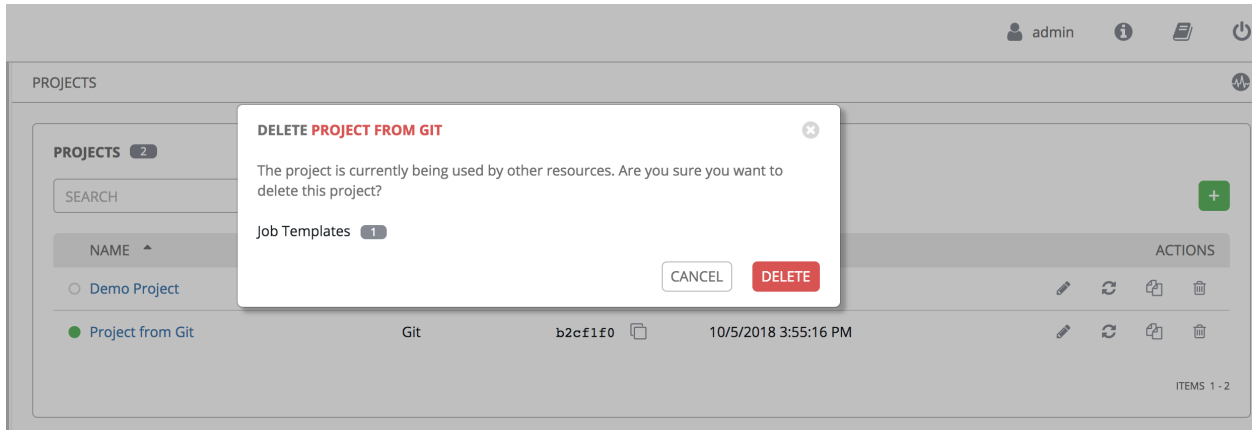
A **job template** is a definition and set of parameters for running an Ansible job. Job templates are useful to execute the same job many times. Job templates also encourage the reuse of Ansible playbook content and collaboration between teams. While the REST API allows for the execution of jobs directly, Tower requires that you first create a job template.



The () menu opens a list of the job templates that are currently available. The job template list is sorted alphabetically by name but you can search by various fields and attributes of the job template. The job template list also enables you to 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:

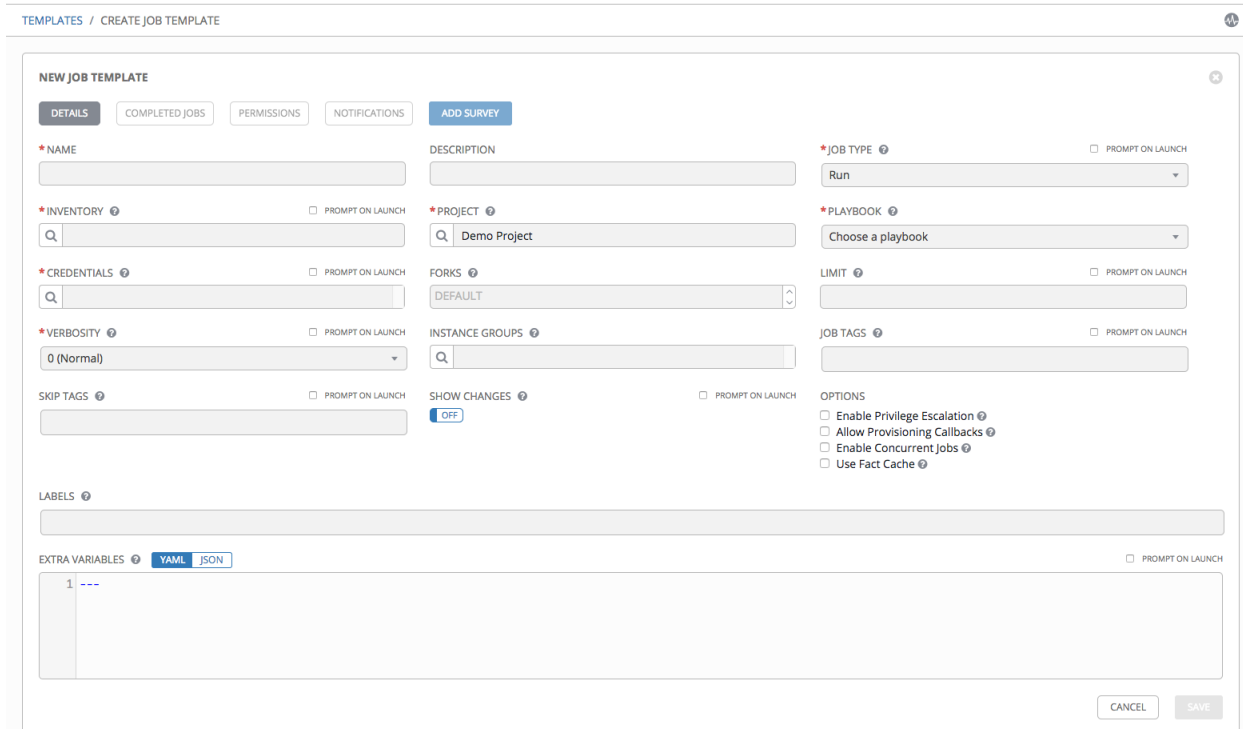


Note: Job templates can be used to build a workflow template. Many parameters in a job template allow you to enable **Prompt on Launch** that can be modified at the workflow level, and do not affect the values assigned at the job template level. For instructions, see the *Workflow Visualizer* section.

15.1 Create a Job Template

To create a new job template:


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

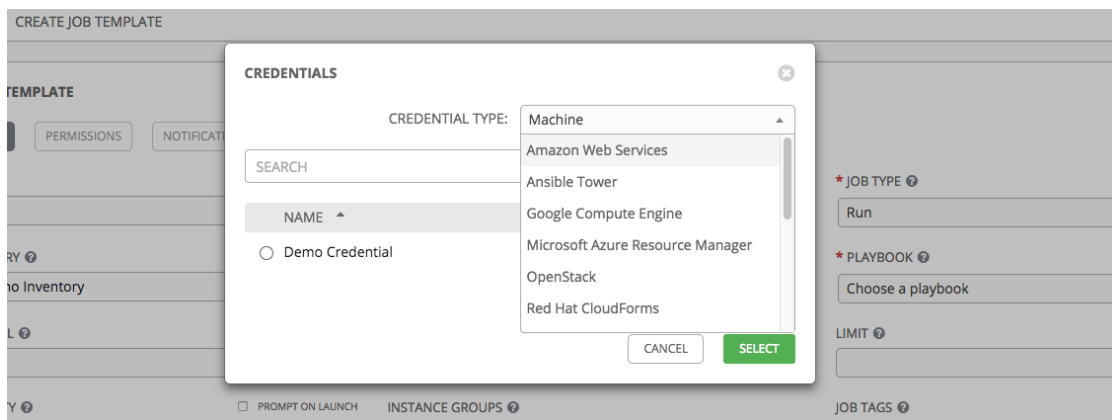


2. Enter the appropriate details into the following fields:

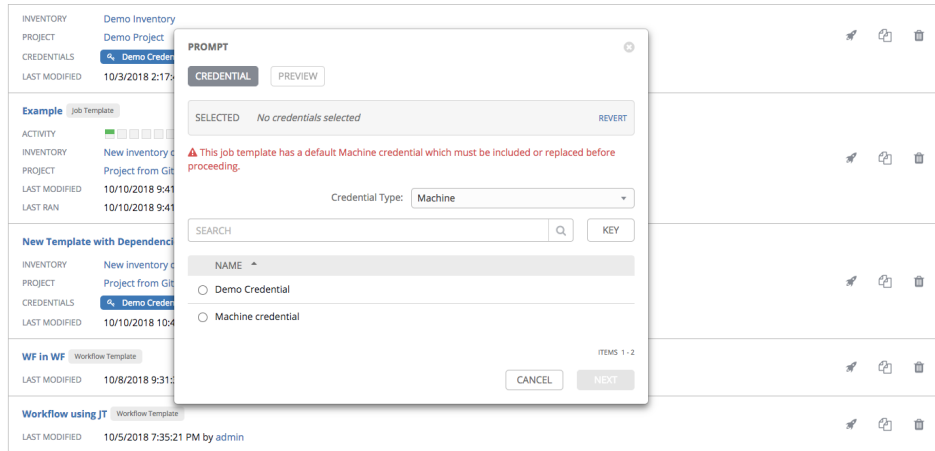
- **Name:** Enter a name for the job.
- **Description:** Enter an arbitrary description as appropriate (optional).
- **Job Type:**
 - **Run:** Execute the playbook when launched, running Ansible tasks on the selected hosts.
 - **Check:** Perform a “dry run” of the playbook and report changes that would 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 job types can be found in the [Playbooks: Special Topics](#) section of the Ansible documentation.

- **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.
- **Playbook:** Choose the playbook to be launched with this job template from the available playbooks. This menu is automatically populated with the names of the playbooks found in the project base path for the selected project. For example, a playbook named “jboss.yml” in the project path appears in the menu as “jboss”.
- **Credential:** 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.



- **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. Set the verbosity to any of Default, Verbose, or Debug. 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. For more information and examples refer to [Tags](#) in the Ansible documentation.
- **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. For more information and examples refer to [Tags](#) in the Ansible documentation.
- **Prompt on Launch** – If selected, even if a default value is supplied, you will be prompted upon launch to choose tag(s) to skip.
- **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.
- 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.



- **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.
- **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.
- **Allow 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 Concurrent Jobs:** Allow jobs in the queue to run simultaneously if not dependent on one another. Refer to *Ansible Tower Capacity Determination and Job Impact* for additional information.
- **Use 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 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 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, select **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 now 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 can verify the template is saved when the newly created template appears on the list of templates at the bottom of the screen.

NAME	INVENTORY	PROJECT	CREDENTIALS	LAST MODIFIED
Demo Job Template	Demo Inventory	Demo Project	DEMO CREDENTIAL	2/9/2018 10:58:18 AM by admin
Example	Demo Inventory	Demo Example	DEMO CREDENTIAL	2/9/2018 4:10:35 PM by admin

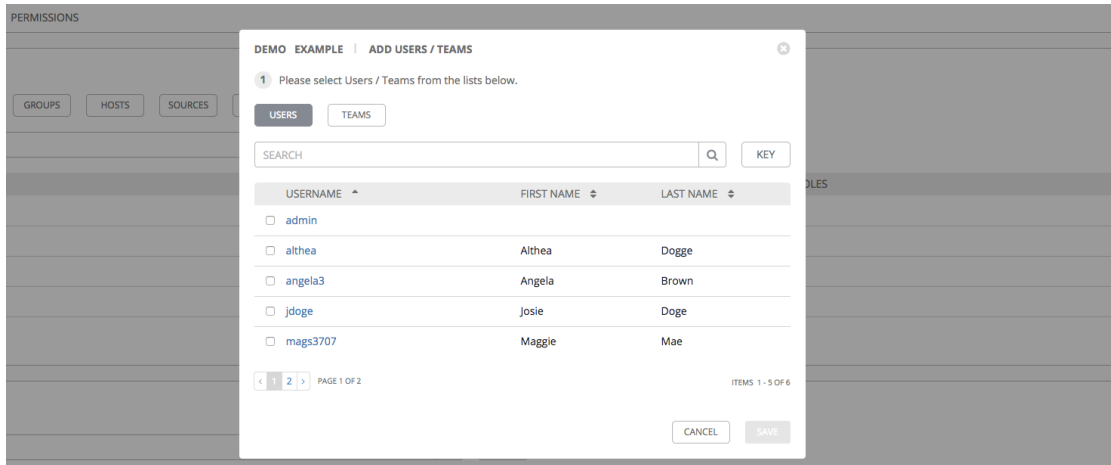
15.2 Add Permissions

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.

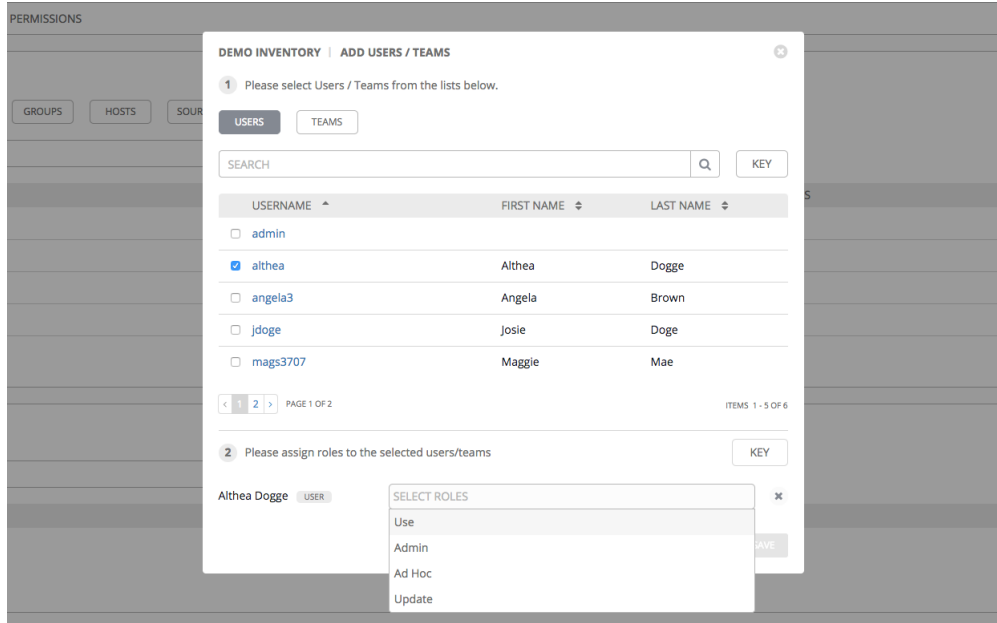


3. Specify the users or teams that will have access then assign them specific roles:

1. Click to select one or multiple checkboxes 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:

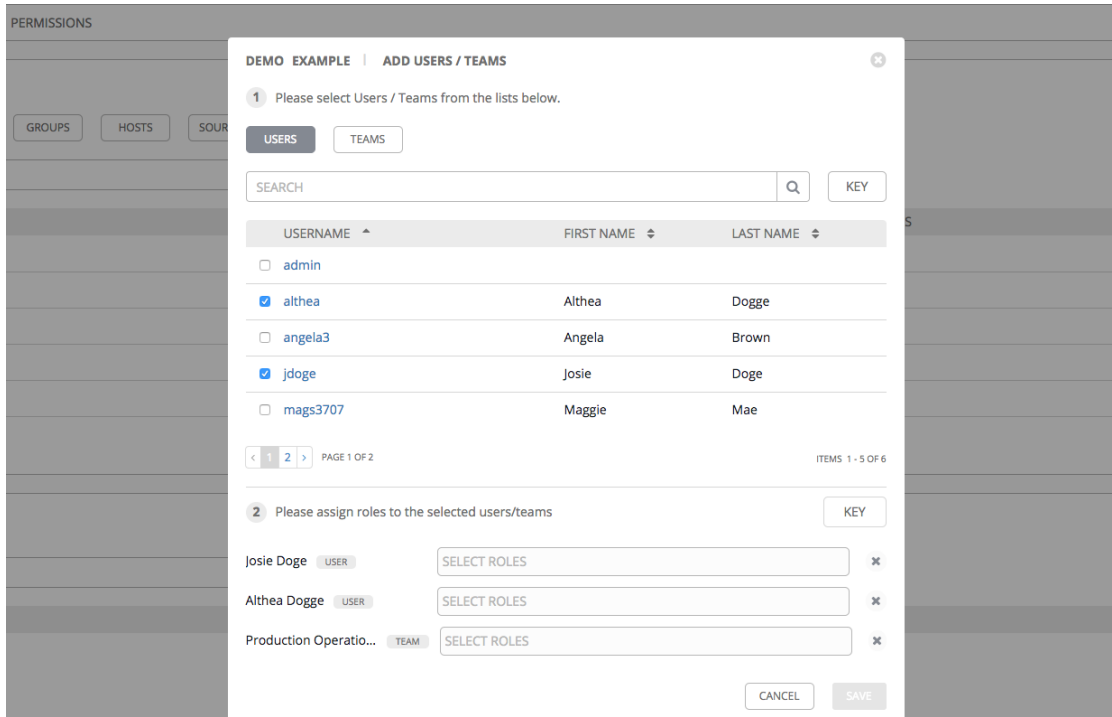
- **Admin** allows read, run, and edit privileges (applies to all resources)
- **Use** allows use of a resource in a job template (applies all resources except job templates)
- **Update** allows updating of project via the SCM Update (applies to projects and inventories)
- **Ad Hoc** allows use of Ad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

Tip: Use the **Key** button in the roles selection pane to display a description of each of the roles.

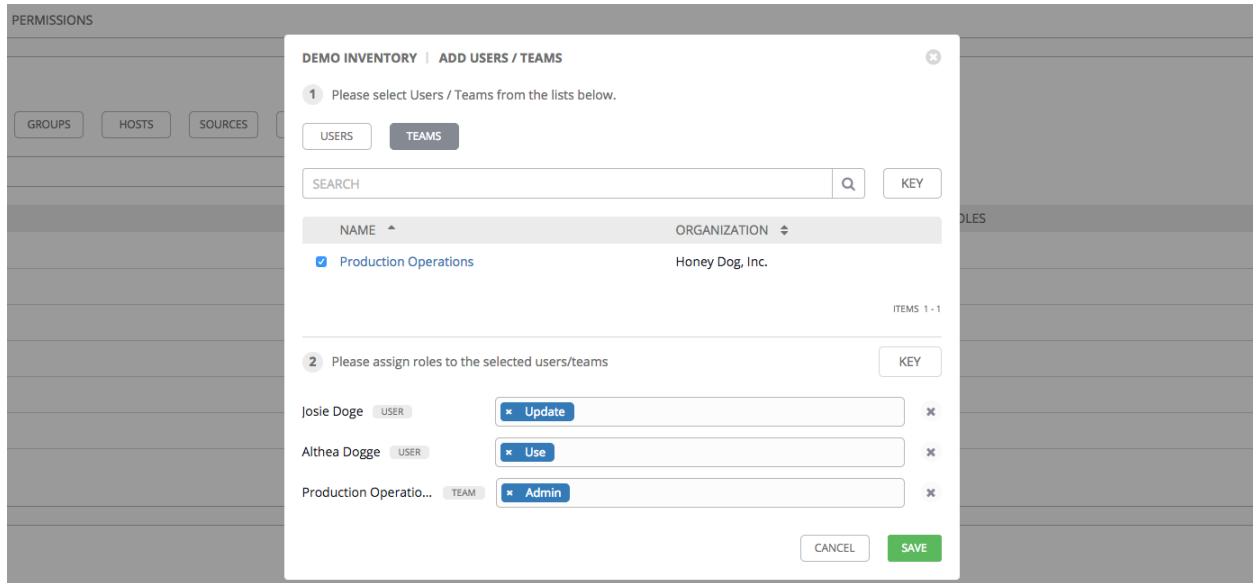
2. Select the role to apply to the selected user or team.

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.



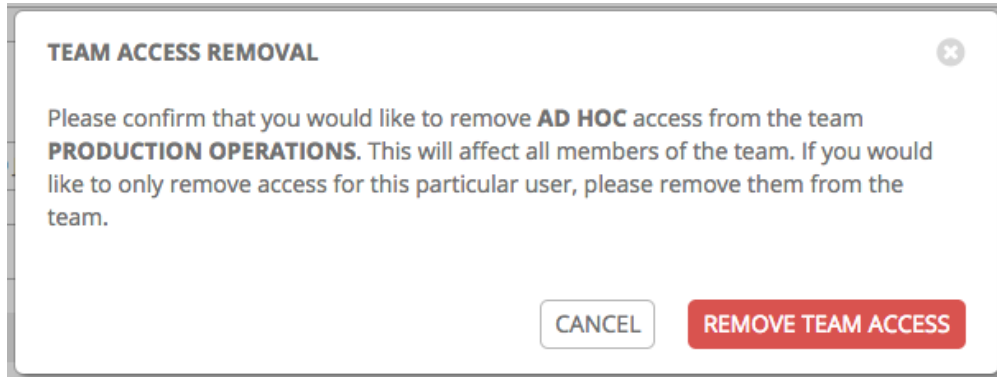
5. Click **Save** when done, and the Add Users/Teams window closes to display the updated roles assigned for each user and team.

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.

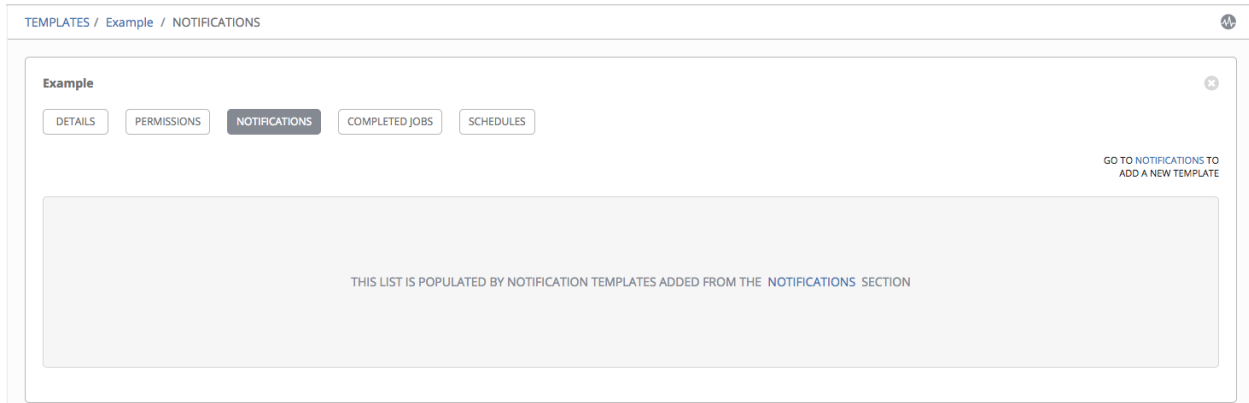
USER	ROLE	TEAM ROLES
admin	SYSTEM ADMINISTRATOR	
athea	AD HOC SYSTEM AUDITOR USE	
jdjge	UPDATE USE	
mags3707	SYSTEM ADMINISTRATOR	AD HOC ADMIN USE
yser	SYSTEM AUDITOR	

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



15.3 Work with Notifications

Clicking the **Notifications** tab allows you to review any notification integrations you have setup. If none are setup, the following screen displays with links to create one:



Follow the on-screen links to create a notification template. Refer to *Notifications* for more information.

Demo Job Template

DETAILS PERMISSIONS **NOTIFICATIONS** COMPLETED JOBS SCHEDULES

SEARCH Q KEY GO TO NOTIFICATIONS TO ADD A NEW TEMPLATE

NAME ^	TYPE ⇅	SUCCESS	FAILURE
test-actions-notification test notification-template	Slack	<input type="checkbox"/>	<input type="checkbox"/>
e2e-ae53906d-notification-template	SMS	<input type="checkbox"/>	<input type="checkbox"/>
test-actions-notification-template	Slack	<input type="checkbox"/>	<input type="checkbox"/>
test-actions-notification-template@3:05:07 PM	EMAIL	<input type="checkbox"/>	<input type="checkbox"/>

ITEMS 1 - 4

15.4 View Completed Jobs

The **Completed Jobs** tab provides details of how this job template has been run. It provides you with the ID, Name, Job Type, when it completed, and allows you to relaunch or delete the job. You can filter the list of completed jobs using the job ID, Name, Type, or if the Job Failed.

Demo Job Template


DETAILS PERMISSIONS NOTIFICATIONS **COMPLETED JOBS** SCHEDULES

SEARCH Q KEY

<p>17 - Demo Job Template Playbook Run</p> <p>STARTED 9/11/2018 11:33:38 PM FINISHED 9/11/2018 11:33:43 PM</p> <p>LAUNCHED BY admin</p> <p>JOB TEMPLATE Demo Job Template 🔍 🗑️</p> <p>INVENTORY Demo Inventory</p> <p>PROJECT Demo Project</p> <p>CREDENTIALS <input type="text" value="Demo Credential"/></p>
<p>14 - Demo Job Template Playbook Run</p> <p>STARTED 9/11/2018 11:33:10 PM FINISHED 9/11/2018 11:33:16 PM</p> <p>LAUNCHED BY admin</p> <p>JOB TEMPLATE Demo Job Template 🔍 🗑️</p> <p>INVENTORY Demo Inventory</p> <p>PROJECT Demo Project</p> <p>CREDENTIALS <input type="text" value="Demo Credential"/></p>
<p>11 - Demo Job Template Playbook Run</p> <p>STARTED 9/11/2018 11:31:51 PM FINISHED 9/11/2018 11:31:56 PM</p> <p>LAUNCHED BY admin</p> <p>JOB TEMPLATE Demo Job Template 🔍 🗑️</p> <p>INVENTORY Demo Inventory</p> <p>PROJECT Demo Project</p> <p>CREDENTIALS <input type="text" value="Demo Credential"/></p>

15.5 Scheduling

Access the schedules for a particular job template from the **Schedules** tab. Otherwise, you can launch the scheduled

jobs list via the  button. Scheduling from the job template page opens the **Schedules** page.

TEMPLATES / New Template with Dependencies / SCHEDULES

New Template with Dependencies

DETAILS PERMISSIONS NOTIFICATIONS COMPLETED JOBS SCHEDULES

SEARCH [] Q KEY [] +

NAME	FIRST RUN	NEXT RUN	FINAL RUN	ACTIONS
ON Schedule 1	10/10/2018 11:00:00 PM	10/10/2018 11:00:00 PM	10/10/2018 11:00:00 PM	[] []

ITEMS 1-1

This page displays a list of the schedules that are currently available for the selected **Job Template**. The schedule list may be sorted and searched by any of the following:


- **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 run of the task

Buttons located in the upper right corner of the **Schedules** screen provide the following actions:

- View Activity Stream
- Add a new schedule

15.5.1 Schedule a Job Template

To create a new schedule:

1. From the Schedules screen, click the  button.
2. Enter the appropriate details into the following fields:
 - **Name**
 - **Start Date**
 - **Start Time**
 - **Local Time Zone:** the entered Start Time should be in this timezone
 - **Repeat Frequency:** the appropriate options display as the update frequency is modified

Note: Jobs are scheduled in UTC. Repeating jobs that runs at a specific time of day may move relative to a local timezone when Daylight Saving Time shifts occur.

The Schedule Description below displays the specifics of the schedule and a list of the scheduled occurrences in the selected Local Time Zone.

TEMPLATES / Demo Job Template / SCHEDULES / CREATE SCHEDULE

Daily Scan

* NAME: Daily Scan

* START DATE: 4/03/2017

* START TIME (HH24:MM:SS): 22:45:00

* LOCAL TIME ZONE: America/New_York

* REPEAT FREQUENCY: Day

FREQUENCY DETAILS

* EVERY: 2 DAYS

* END: On Date

* END DATE: 7/17/2017

* END TIME (HH24:MM:SS): 23:45:00

SCHEDULE DESCRIPTION

every 2 days until July 17, 2017

OCCURRENCES (Limited to first 10) DATE FORMAT LOCAL TIME UTC

4/3/2017 22:45:00 EDT
 4/5/2017 22:45:00 EDT
 4/7/2017 22:45:00 EDT
 4/9/2017 22:45:00 EDT
 4/11/2017 22:45:00 EDT
 4/13/2017 22:45:00 EDT
 4/15/2017 22:45:00 EDT
 4/17/2017 22:45:00 EDT
 4/19/2017 22:45:00 EDT
 4/21/2017 22:45:00 EDT

EXTRA VARIABLES @YAML JSON

1 ---

CANCEL SAVE

Note: 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:

TEMPLATES / New Template with Dependencies / SCHEDULES / Schedule 1

Schedule 1

* NAME: Schedule 1

* LOCAL TIME ZONE: America/Denver

FREQUENCY DETAILS

* EVERY: 1

* START TIME (HH24:MM:SS): 23:00:00

* OCCURRENCES: 1

SCHEDULE DESCRIPTION

every day for 1 time

OCCURRENCES (Limited to first 10) D

10-10-2018 23:00:00

PROMPT

CREENTIAL PREVIEW

SELECTED No credentials selected REVERT

⚠ This job template has a default Machine credential which must be included or replaced before proceeding.

Credential Type: Machine

SEARCH Q KEY

NAME

Demo Credential

Machine credential

ITEMS 1 - 2

CANCEL NEXT

PROMPT CANCEL SAVE

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.`

3. When satisfied with the schedule specifics, click **Save**.

Once the schedule is saved, the list of schedules display for the associated job template.


Use the **ON/OFF** toggle button to quickly activate or deactivate this schedule.

Other actions for schedules are available under the Actions column:

- Edit Schedule
- Delete schedule

15.6 Surveys

Job types of Run or Check will provide a way to set up surveys in the Job Template creation or editing screens. Surveys set extra variables for the playbook similar to ‘Prompt for Extra Variables’ does, but in a user-friendly question and

answer way. Surveys also allow for validation of user input. Click the  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.

Note: Surveys are only available to those with Enterprise-level licenses.

15.6.1 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.

NEW JOB TEMPLATE | SURVEY ON ✕

ADD SURVEY PROMPT

* PROMPT

DESCRIPTION

* ANSWER VARIABLE NAME ?

* ANSWER TYPE

MINIMUM LENGTH MAXIMUM LENGTH


DEFAULT ANSWER

REQUIRED

PREVIEW
 PLEASE ADD A SURVEY PROMPT ON THE LEFT.

- *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.

NEW JOB TEMPLATE | SURVEY ON

ADD SURVEY PROMPT

* PROMPT

DESCRIPTION

* ANSWER VARIABLE NAME

* ANSWER TYPE

REQUIRED

CANCEL ADD

PREVIEW

* WHICH GROUP(S) SHOULD INCLUDE THIS USER?

Enter groups, one per line.

Click and hold down to drag the question to reorder it.

CANCEL SAVE

15.6.2 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 ("").

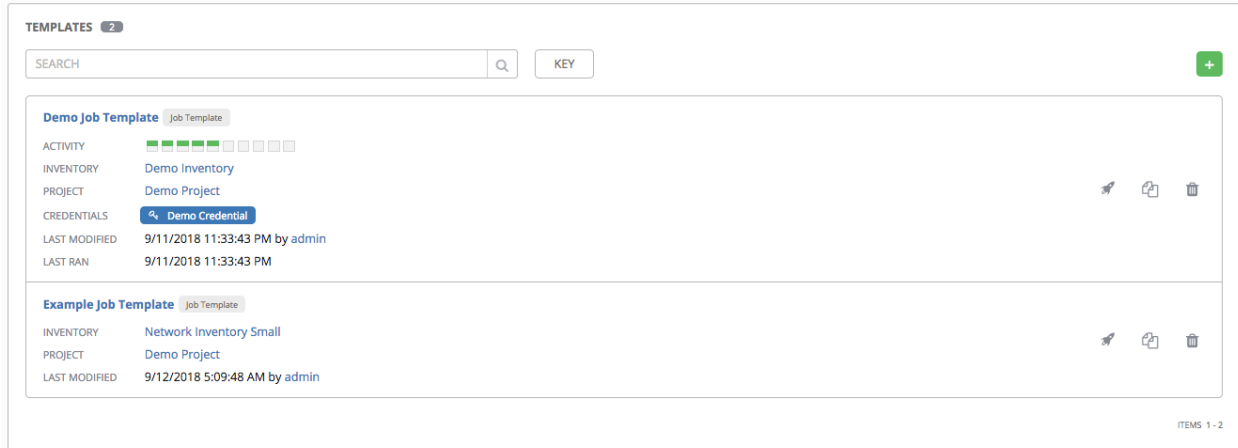
15.7 Launch a Job Template

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.

To launch a job template:

1. Access the job template from the **Templates** navigational link or while in the Job Template Details view, scroll to the bottom to access it from a list of templates.

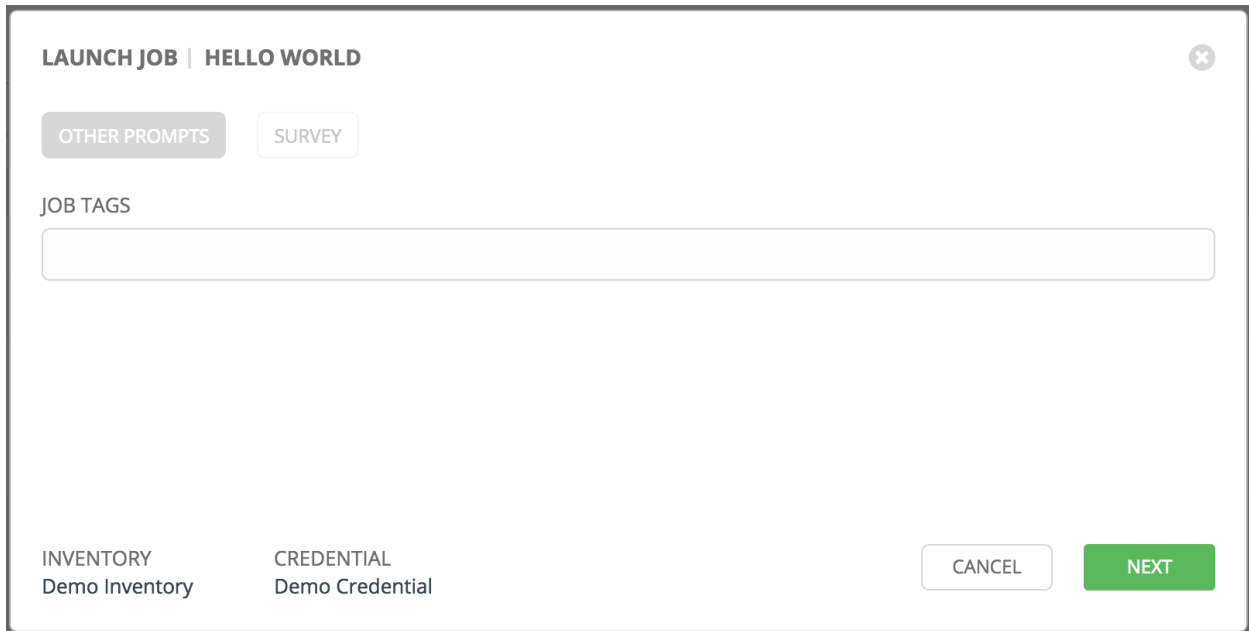


2. Click the  button.

A job may require additional information to run. The following data may be requested at launch:

- Credentials that were setup
- 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

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

***WHICH GROUP(S) SHOULD INCLUDE THIS USER?**
Enter groups, one per line.

INVENTORY CREDENTIAL
 Demo Inventory Demo Credential

CANCEL LAUNCH

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.

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

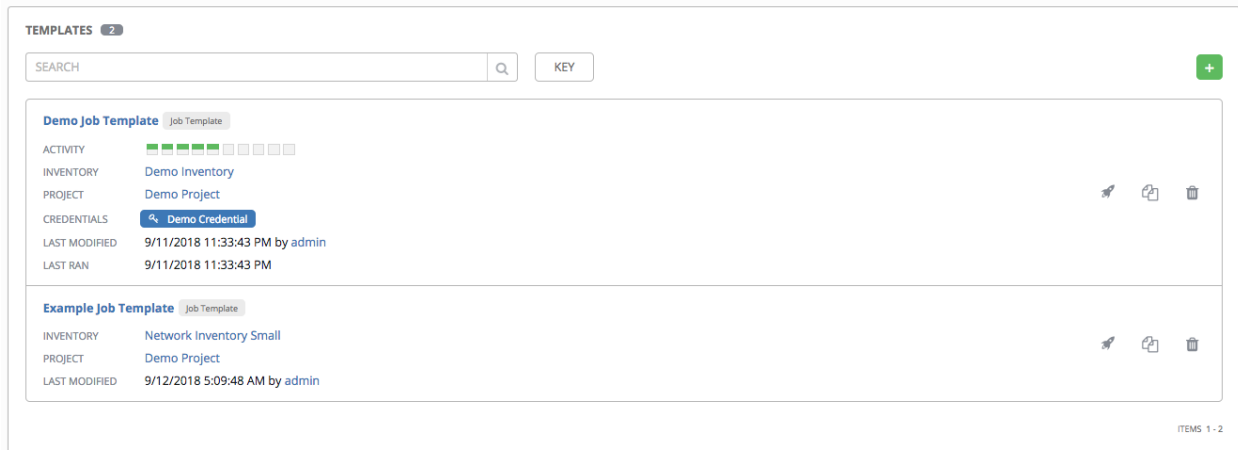
Upon launch, Tower automatically redirects the web browser to the Job Status page for this job under the **Jobs** tab.

Note: Starting with Ansible Tower 3.3, 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.

15.8 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 that you want to copy from the **Templates** navigational link or while in the Job Template Details view, scroll to the bottom to access it from a list of templates.



2. Click the  button.

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

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.

15.9 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.

15.9.1 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:

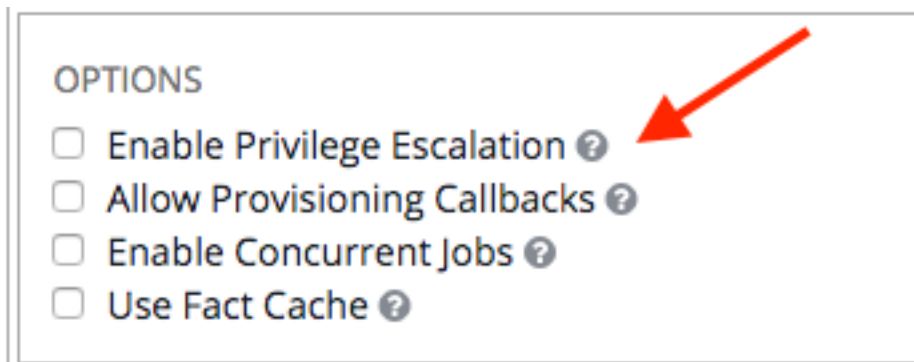
```
- hosts: all
  vars:
    scan_use_checksum: false
    scan_use_recursive: false
  tasks:
    - scan_packages:
    - scan_services:
    - scan_files:
      paths: '{{ scan_file_paths }}'
      get_checksum: '{{ scan_use_checksum }}'
      recursive: '{{ scan_use_recursive }}'
      when: scan_file_paths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

15.9.2 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 12.04, 14.04, 16.04
- 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.

15.9.3 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 15, 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.

15.9.4 Custom Fact Scans

A playbook for a custom fact scan is similar to the example of the Fact Scan Playbook above. As an example, a playbook that only uses a custom `scan_foo` Ansible fact module would look like this:

scan_custom.yml:

```
- hosts: all
  gather_facts: false
  tasks:
  - scan_foo:
```

scan_foo.py:

```
def main():
    module = AnsibleModule(
        argument_spec = dict())

    foo = [
        {
            "hello": "world"
        },
        {
            "foo": "bar"
        }
    ]
    results = dict(ansible_facts=dict(foo=foo))
    module.exit_json(**results)

main()
```

To use a custom fact module, ensure that it lives in the `/library/` subdirectory of the Ansible project used in the scan job template. This fact scan module is very simple, returning a hard-coded set of facts:

```
[
  {
    "hello": "world"
  },
  {
    "foo": "bar"
  }
]
```

Refer to the [Module Provided 'Facts'](#) section of the Ansible documentation for more information.

15.10 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:

SETTINGS / EDIT CONFIGURATION

CONFIGURE Tower

AUTHENTICATION | **JOBS** | SYSTEM | USER INTERFACE

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 ON

DEFAULT JOB TIMEOUT REVERT 0

DEFAULT INVENTORY UPDATE TIMEOUT REVERT 0

DEFAULT PROJECT UPDATE TIMEOUT REVERT 0

PER-HOST ANSIBLE FACT CACHE TIMEOUT REVERT 0

EXTRA ENVIRONMENT VARIABLES

```
1 {
2   "HOME": "/var/lib/awx",
3   "USER": "awx"
4 }
```

REVERT ALL TO DEFAULT

CANCEL SAVE

Copyright © 2017 Red Hat, Inc.

Upon launching a job that uses fact cache (`use_fact_cache=True`), Tower will inject into memcached, all `ansible_facts` associated with each host in the inventory associated with the job. A list of all hosts in the inventory is also injected into memcached with the `inventory_id` key and host name values. The Ansible Fact Cache plugin that ships with Ansible Tower will only be enabled on jobs with fact cache enabled (`use_fact_cache=True`). The fact cache plugin running in Ansible will connect to the same memcached instance.

When a job finishes running that has fact cache enabled (`use_fact_cache=True`), Tower will go through memcached and retrieve all records for the hosts in the inventory. Any records with update times newer than the currently stored facts per-host will be saved to the database.

Tower will always inject the host `ansible_facts` into `memcached`. The cached values may or may not be displayed, based on the per-host newly-stored facts and the timeout value specified for the global fact cache setting.

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`.

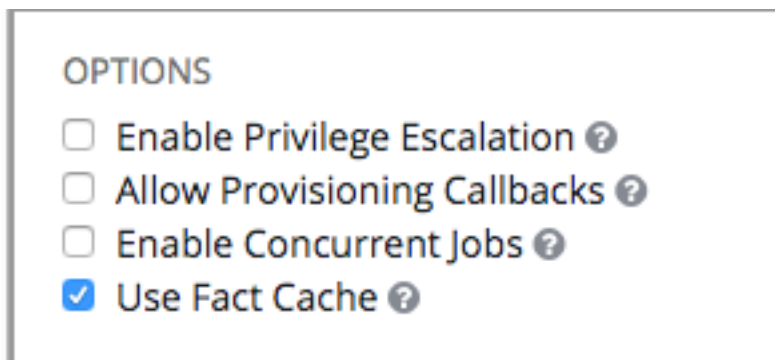
Note: If a hostname includes a forward slash (/), fact cache will not work for that host. If you have an inventory with 100 hosts and one host has a / in the name, 99 of those hosts will still collect facts.

15.10.1 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 `tower.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.



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`.

15.11 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. The use of Cloud Credentials was introduced in Ansible Tower version 2.4.0.

15.11.1 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 }}"
```

15.11.2 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.

15.11.3 Rackspace

Rackspace cloud credentials are exposed as the following environment variables during playbook execution (in the job template, choose the cloud credential needed for your setup):

- `RAX_USERNAME`
- `RAX_API_KEY`

All of the Rackspace modules will implicitly use these credentials when run via Tower without having to set the `username` or `api_key` module options.

15.11.4 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.

15.11.5 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.

15.11.6 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
  cluster: MainCluster
  resource_pool: "/Resources"
  vm_extra_config:
    folder: MyFolder
```

15.12 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/callback"/></p>	<p>HOST CONFIG KEY ?</p> <p><input type="text"/></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):

```
root@localhost:~$ 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.

15.12.1 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", ...}}'
```

(Added in Ansible Tower version 2.2.0.)

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](#).

15.12.2 Provisioning Callback through tower-cli

As an alternative to running the `request_tower_configuration.sh` script or a custom script, you can use `tower-cli` to make a provisioning callback, as in the following example:

```
tower-cli job_template callback --host-config-key="5a8ec154832b780b9bdef1061764ae5a" -
↪-extra-vars="foo: bar"
```

15.13 Extra Variables

Note: Additional strict `extra_vars` validation was added in Ansible Tower 3.0.0. `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: Beginning with Ansible Tower version 2.4, the behavior for Job Template extra variables and Survey variables has changed. Previously, variables set using a Survey overrode any extra variables specified in the Job Template. In 2.4 and later, the Job Template extra variables dictionary is merged with the Survey variables. This may result in a change of behavior upon upgrading to 2.4.

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

15.13.1 Relaunching Job Templates

Another change for Ansible Tower version 2.4 introduced a `launch_type` setting for your jobs. 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**,

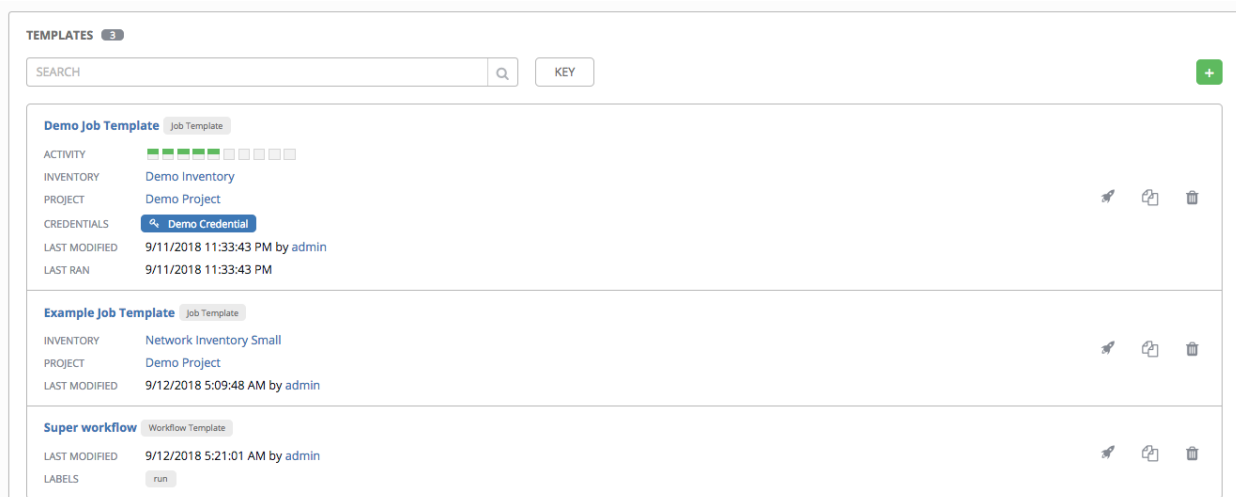
which would also include `extra_vars`.

WORKFLOW JOB TEMPLATES

A **workflow job template** links together a sequence of disparate job templates that accomplishes the task of tracking the full set of jobs that were part of the release process as a single unit.

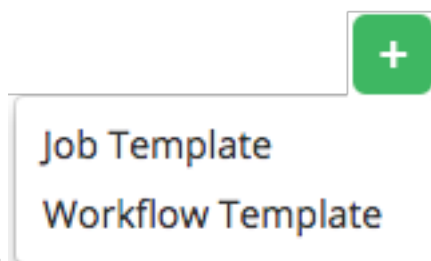


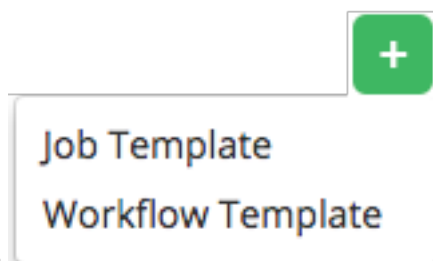
The () menu opens a list of the workflow and job templates that are currently available. The workflow/job template list is sorted alphabetically by name but you can search by various fields and attributes of the workflow/job template. The workflow/job template list also enables you to launch, copy, and remove a job template.



16.1 Create a Workflow Template

To create a new workflow job template:



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

The screenshot shows the 'New Workflow Job Template' interface. At the top, there are five tabs: 'DETAILS' (selected), 'PERMISSIONS', 'NOTIFICATIONS', 'ADD SURVEY', and 'WORKFLOW EDITOR'. Below the tabs are three input fields: '*NAME', 'DESCRIPTION', and 'ORGANIZATION'. Underneath is a 'LABELS' section with a text input field and a small 'x' icon. At the bottom, there is an 'EXTRA VARIABLES' section with radio buttons for 'YAML' (selected) and 'JSON'. A code editor area shows a single line of text: '1 ---'. At the bottom right, there are 'CANCEL' and 'SAVE' buttons.

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:** Enter or search for an organization to associate the workflow.
- **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 ?



The screenshot shows the 'Example' job template overview page. It has a 'JOB TEMPLATE' header. The page lists several fields: 'INVENTORY' (Demo Inventory), 'PROJECT' (Demo Example), 'CREDENTIALS' (DEMO CREDENTIAL), and 'LAST MODIFIED' (2/9/2018 4:13:41 PM by admin). At the bottom, the 'LABELS' field contains three buttons: 'run', 'scan', and 'test'. A red arrow points to the 'test' label.

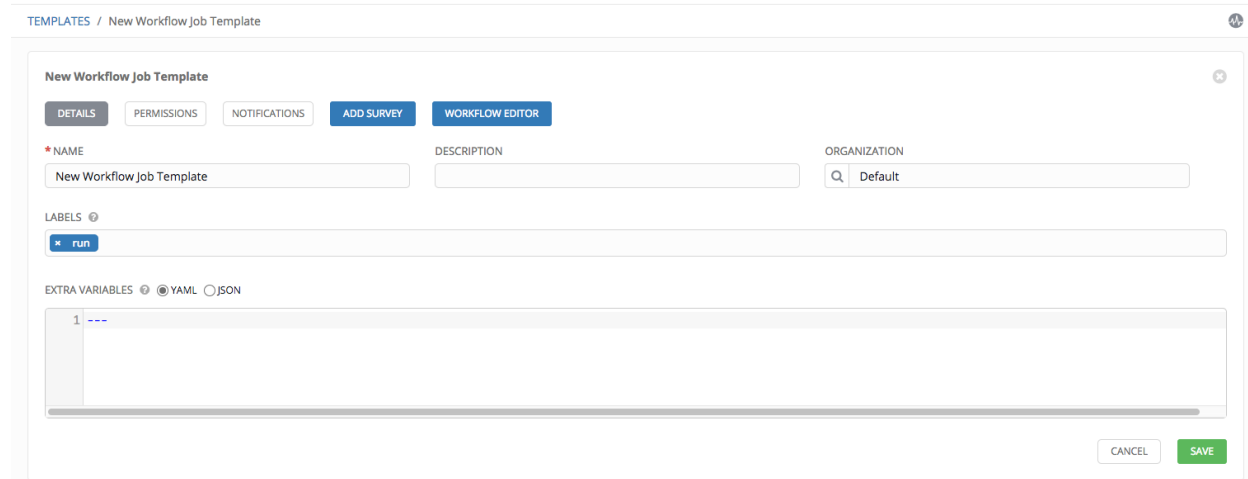
- **Extra Variables:**
 - Pass extra command line variables to the playbook. This is the “-e” or “-extra-vars” command line parameter for ansible-playbook that is documented in the Ansible documentation at [Passing Variables on the Command Line](#).
 - Provide key/value pairs using either YAML or JSON. 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
```

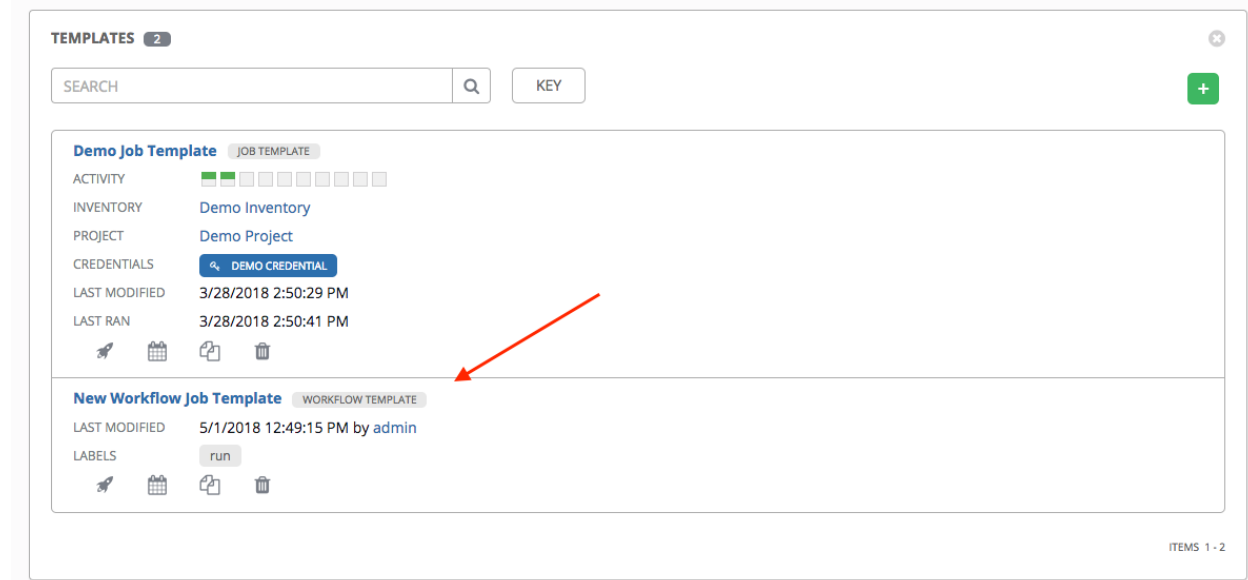
For more information about extra variables, refer to [Extra Variables](#).

3. When you have completed configuring the workflow job template, select **Save**.

Saving the template does not exit the Workflow Job Template page but remains on the Workflow Job Template Details view for further editing, if necessary. The Details tab of a saved template allows you to review, edit, and add permissions, notifications, surveys, or build a workflow job template using an editor.

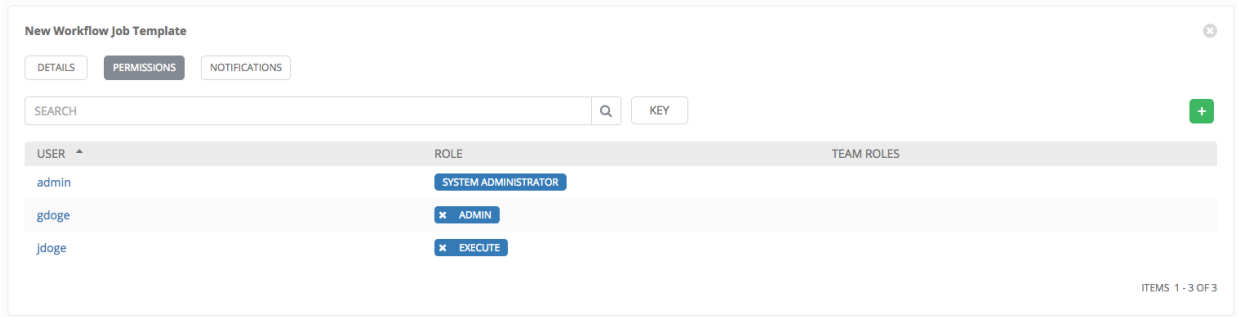



You can verify the template is saved when the newly created workflow job template appears on the list of templates at the bottom of the screen.



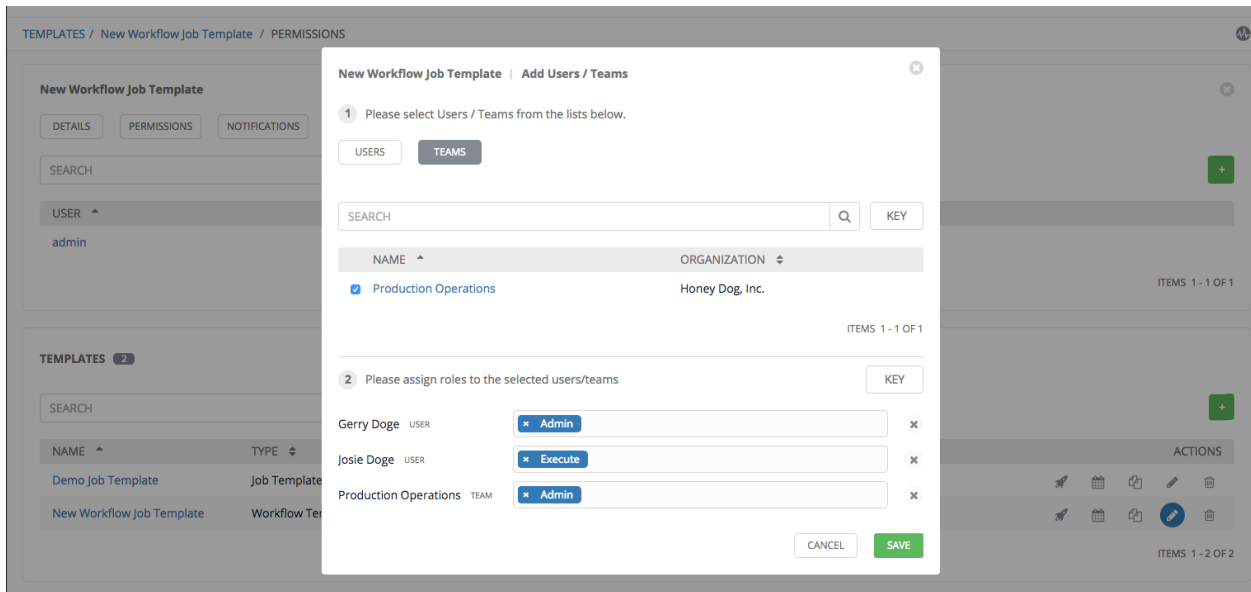
16.2 Work with Permissions

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



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

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



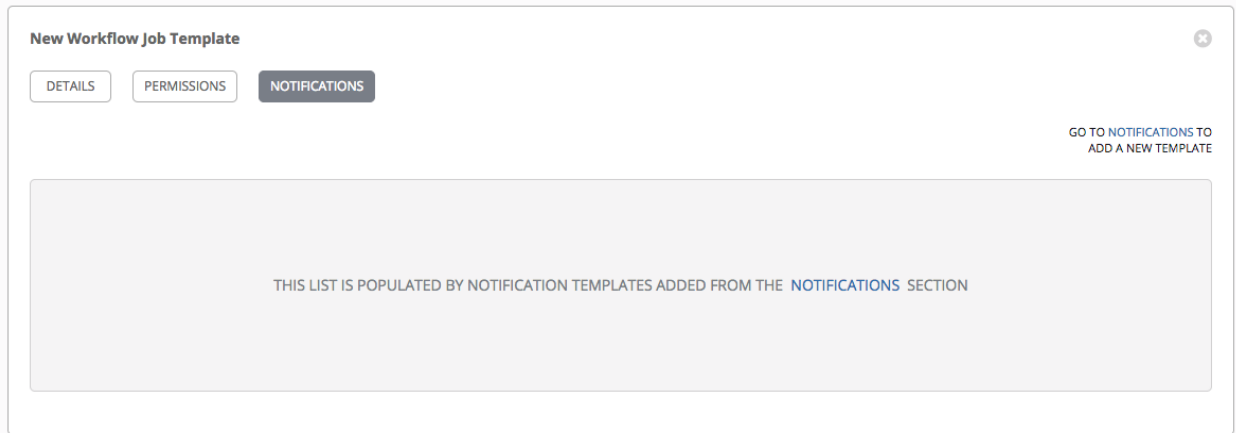
Note that you do not have to choose between teams or users, and that you can assign permissions to both at the same time.

16.3 Work with Notifications

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

Clicking on **Notifications** (beside **Users** when viewing your organization), allows you to easily manage notifications for this organization. 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 *Notifications* for more information.



16.4 Work with Schedules

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

NAME ^	FIRST RUN ↕	NEXT RUN ↕	FINAL RUN ↕	ACTIONS
<input checked="" type="checkbox"/> Monthly monitoring	10/15/2018 11:00:00 PM	10/15/2018 11:00:00 PM	1/15/2019 11:00:00 PM	
<input checked="" type="checkbox"/> Repeating Everyday	9/13/2018 3:00:00 PM	9/13/2018 3:00:00 PM	9/16/2018 3:00:00 PM	

From this view, you can select schedules to edit, turn on or off, or select multiple schedules to delete.

This screen displays a list of the schedules that are currently available for the selected **Project**. The schedule list may be sorted and searched by **Name**.


The list of schedules includes:

- **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
- **Last Modified:** The last time this schedule was modified

16.4.1 Schedule a Workflow Job Template

To create a new schedule:



1. From the Schedules screen, click the  button.
2. Enter the appropriate details into the following fields:

- **Name**
- **Start Date**
- **Start Time**
- **Local Time Zone:** the entered Start Time should be in this timezone
- **Repeat Frequency:** the appropriate options display as the update frequency is modified

Note: Jobs are scheduled in UTC. Repeating jobs that runs at a specific time of day may move relative to a local timezone when Daylight Saving Time shifts occur.

The Schedule Description below displays the specifics of the schedule and a list of the scheduled occurrences in the selected Local Time Zone.

TEMPLATES / New Workflow Job Template / SCHEDULES / Daily workflow run

Daily workflow run

* NAME: Daily workflow run

* START DATE: 03/27/2017

* START TIME (HH24:MM:SS): 20:45:00

* LOCAL TIME ZONE: America/Denver

* REPEAT FREQUENCY: Day

FREQUENCY DETAILS

* EVERY: 2 DAYS

* END: On Date

* END DATE: 07/17/2017

* END TIME (HH24:MM:SS): 23:45:00

SCHEDULE DESCRIPTION

every 2 days until July 17, 2017

OCCURRENCES (limited to first 10) DATE FORMAT LOCAL TIME UTC

3/27/2017 20:45:00 MDT
 3/29/2017 20:45:00 MDT
 3/31/2017 20:45:00 MDT
 4/2/2017 20:45:00 MDT
 4/4/2017 20:45:00 MDT
 4/6/2017 20:45:00 MDT
 4/8/2017 20:45:00 MDT
 4/10/2017 20:45:00 MDT
 4/12/2017 20:45:00 MDT
 4/14/2017 20:45:00 MDT

EXTRA VARIABLES @YAML JSON

1 ---

CANCEL SAVE

3. When satisfied with the schedule specifics, click **Save**.

Once the schedule is saved, the list of schedules display for the associated workflow job template.

Super workflow

DETAILS PERMISSIONS NOTIFICATIONS COMPLETED JOBS **SCHEDULES**

SEARCH [] Q KEY [] +

NAME ^	FIRST RUN ↓	NEXT RUN ↓	FINAL RUN ↓	ACTIONS
<input checked="" type="checkbox"/> Monthly monitoring	10/15/2018 11:00:00 PM	10/15/2018 11:00:00 PM	1/15/2019 11:00:00 PM	
<input checked="" type="checkbox"/> Repeating Everyday	9/13/2018 3:00:00 PM	9/13/2018 3:00:00 PM	9/16/2018 3:00:00 PM	

ITEMS 1 - 2

Use the **ON/OFF** toggle button to quickly activate or deactivate this schedule.

16.5 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

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.

Note: Surveys are only available to those with Enterprise-level licenses.

16.5.1 Create a Survey

To create a survey:

ADD SURVEY

1. Click on the **ADD SURVEY** 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.

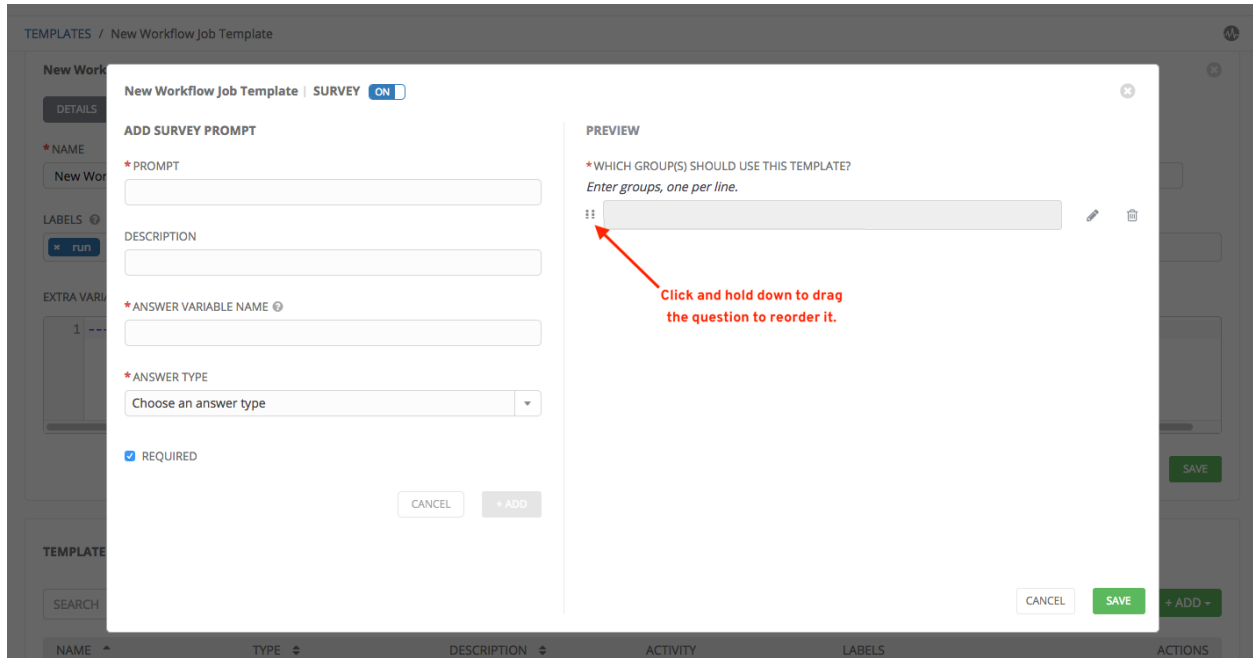
- *Password*: Responses are treated as sensitive information, much like an actual password is treated. You can set the minimum and maximum length (in characters) for this answer.
- *Multiple Choice (single select)*: A list of options, of which only one can be selected at a time. Enter the options, one 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.



16.5.2 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 ("").

16.6 Workflow Visualizer

Starting with Ansible Tower 3.1, the Workflow Visualizer provides a graphical way of linking together job templates, project syncs, and inventory syncs to build a workflow job template.

16.6.1 Build a Workflow

Make sure you have any combination of two of the following templates to build a workflow: jobs, project sync, or inventory sync.

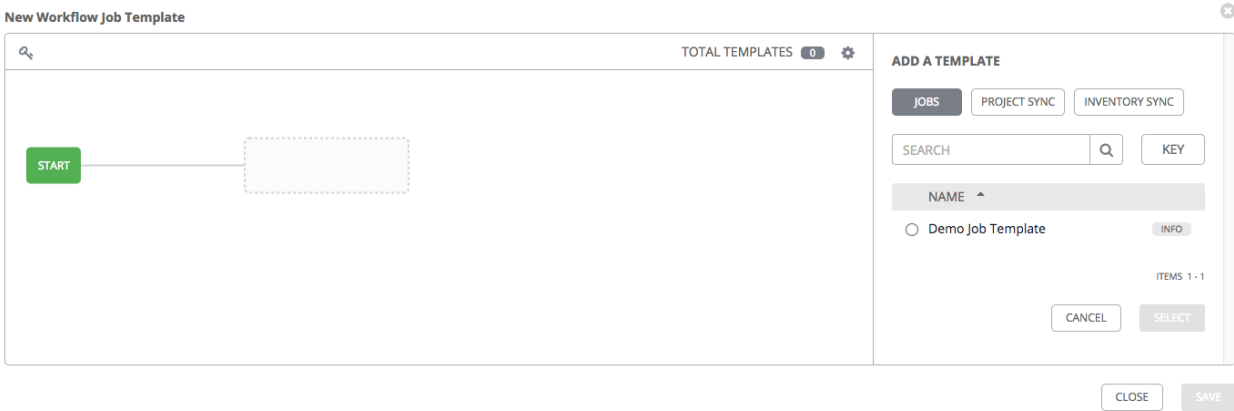
WORKFLOW VISUALIZER

1. In the details/edit view of a workflow job template, click the

button



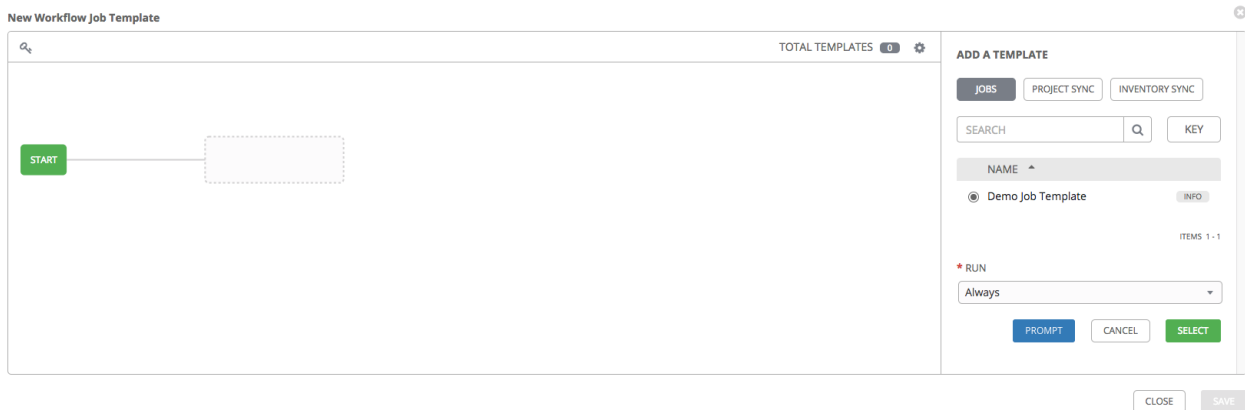
2. Click the **START** button to display a list of templates to add to your workflow.



3. On the right pane, select a template from the list of templates to add. To switch between jobs, project syncs, and inventory syncs, click the appropriate button above. Each template added represents a node.

Note: You will not be able to select job templates that don't have a default inventory when populating a workflow graph. Though 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 template is selected, the workflow begins to build, and a prompt appears, requesting for the type of action to be taken for the selected template:



5. Select one of the following scenarios (edge type) to apply to this template:
 - **On Success:** Upon successful completion, execute the next template.
 - **On Failure:** Upon failure, execute a different template.
 - **Always:** Continue to execute regardless of success or failure.
6. 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) and click **Confirm**.

PROMPT

OTHER PROMPTS PREVIEW

LIMIT

15

* VERBOSITY

3 (Debug)

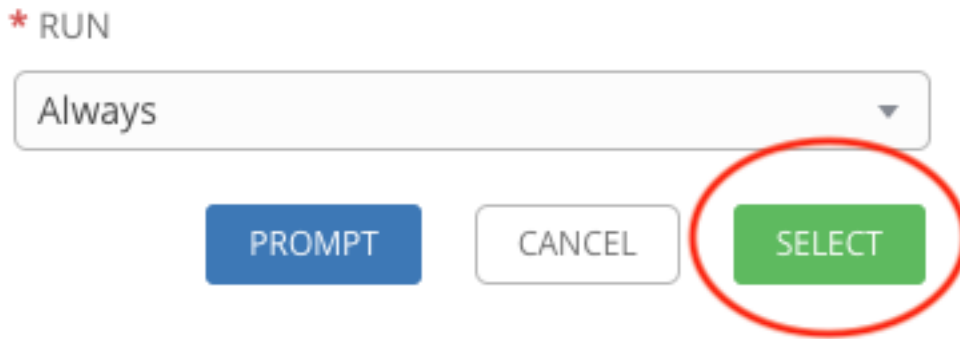
JOB TAGS

3037

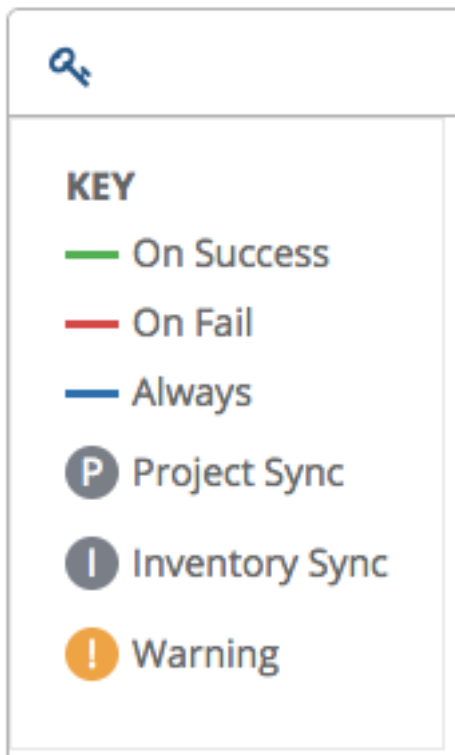
CANCEL NEXT



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.

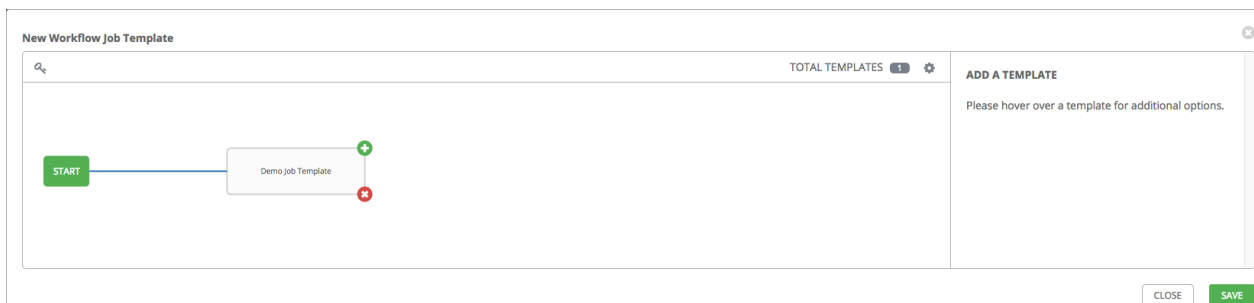
You must also click **Select** when the prompt wizard closes in order to apply the changes at that node. Otherwise, any changes you make will revert back to the values set in the actual job template.



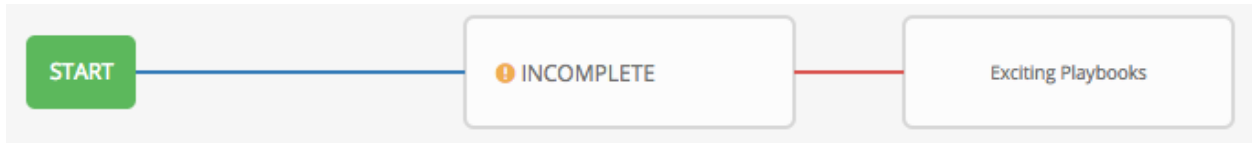
A template that is associated with each workflow node will run based on the selected run scenario as it proceeds. Click the key icon to display the legend for each run scenario and their job types.



Hovering over the node allows you to delete the selected node , or add another one .



Note: If you create a workflow job template with any number of strung-together job templates or job/inventory syncs, and one of those are deleted from your inventory of resources (the resource no longer exists), returning to the Workflow Visualizer for the affected workflow job template will show the missing resource as an incomplete node.



To fix the incomplete node, update it with a new job template or delete it from the workflow.

You can insert another node in between nodes and drag the diagram to depict a split scenario:



If you want to undo the last inserted node, when the right pane opens, click **Cancel** or proceed to click on another node without selecting from the right pane.

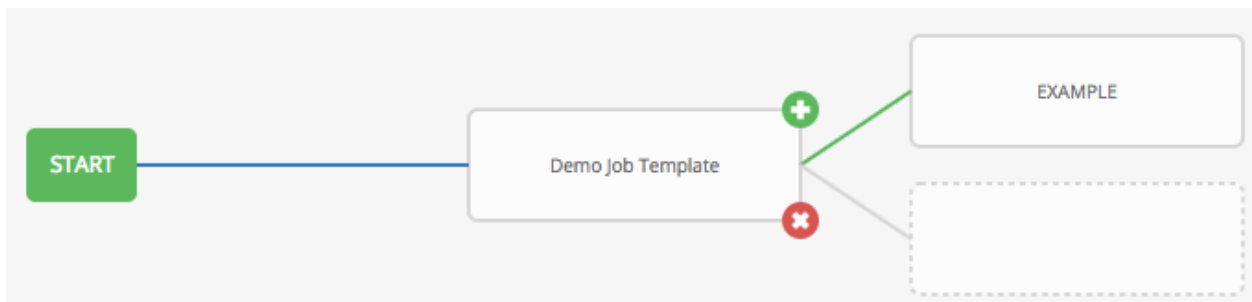
If you want to edit a node, click on the node you want to edit and the right pane displays the current selections. Make your changes and click **Select** to apply them to the graphical view.

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.

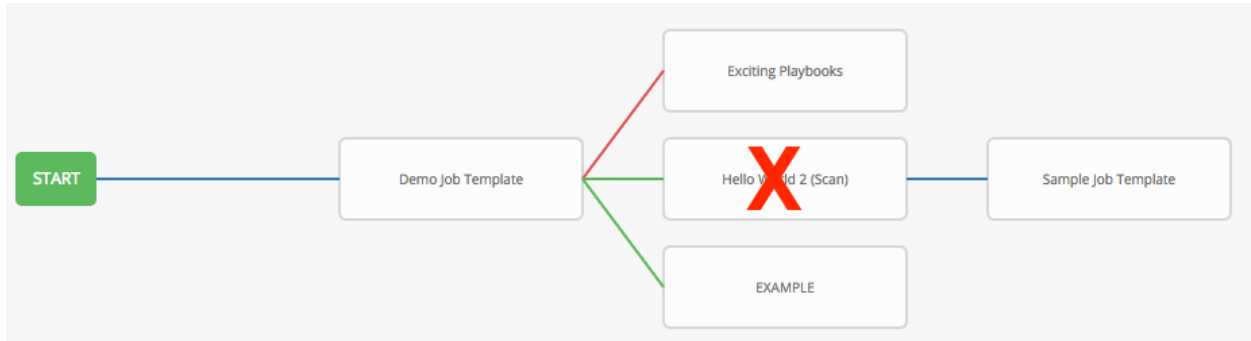


Use the Key at the top of the window to identify the meaning of the symbols and colors associated with the graphical depiction.

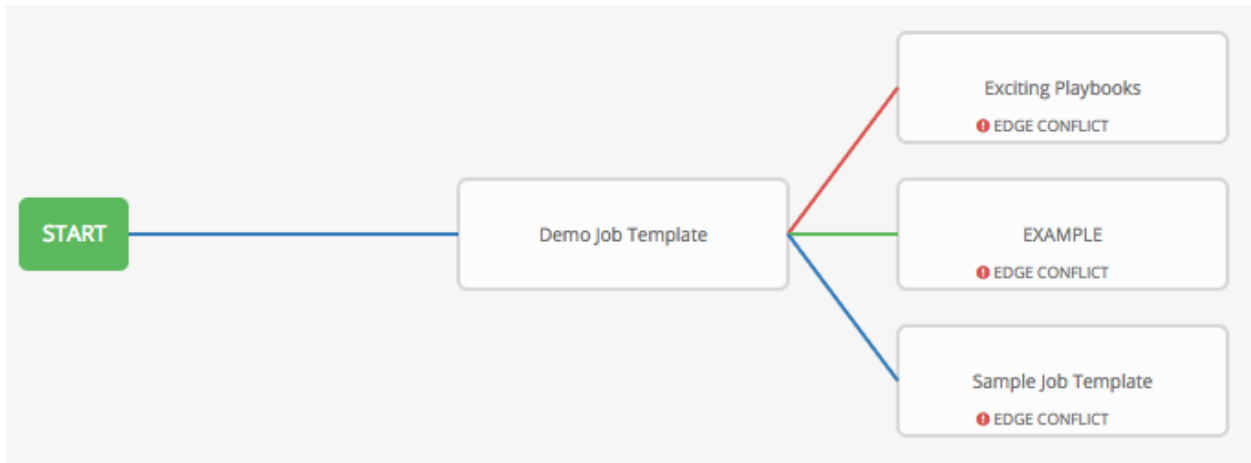
You may add multiple nodes from the same parent node, creating sibling nodes:



Note: In a workflow with a set of sibling steps having varying edge types, and one of those siblings has a follow-on step attached to it, gets removed from the workflow:




The follow-on step will automatically join the set of sibling steps. If there is more than one edge type among the now-siblings, an edge conflict occurs:



You must resolve this conflict before you can save the workflow job template. To resolve the conflict, make all the siblings have the same edge types.



Click the settings icon () to zoom, pan, or reposition the view. Drag the workflow diagram to reposition it on the screen.

7. When done adding/editing a node, click **Select** to save any modifications and render it on the graphical view.

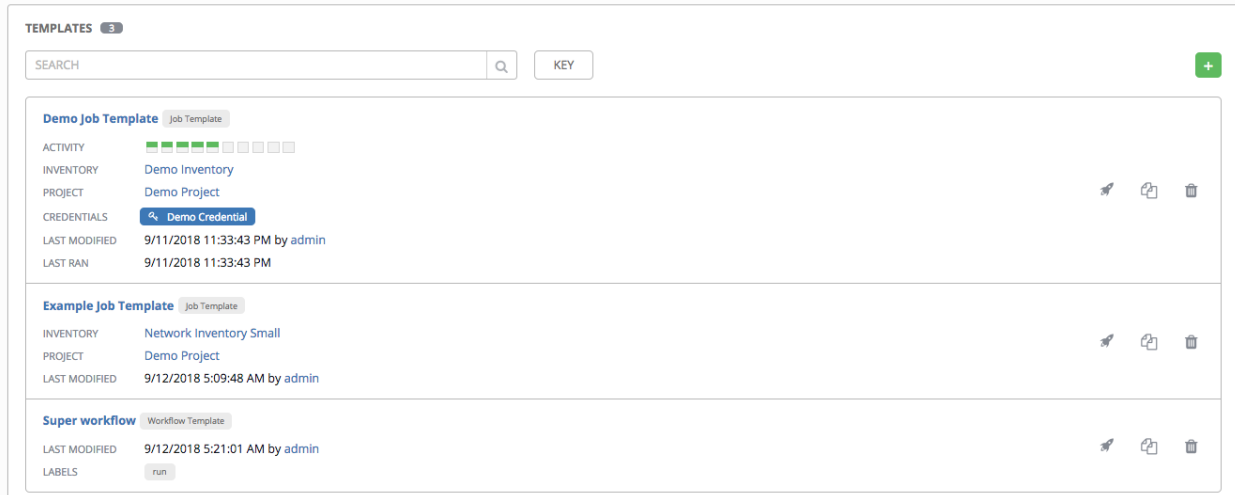
Important: Clicking **Close** on this pane will not save your work, but instead, closes the entire Workflow Visualizer and you will have to start over.


8. When done with building your workflow job template, click **Save** to save your entire workflow job template and return to the new Workflow Template details page.

16.7 Launch a Workflow Job Template

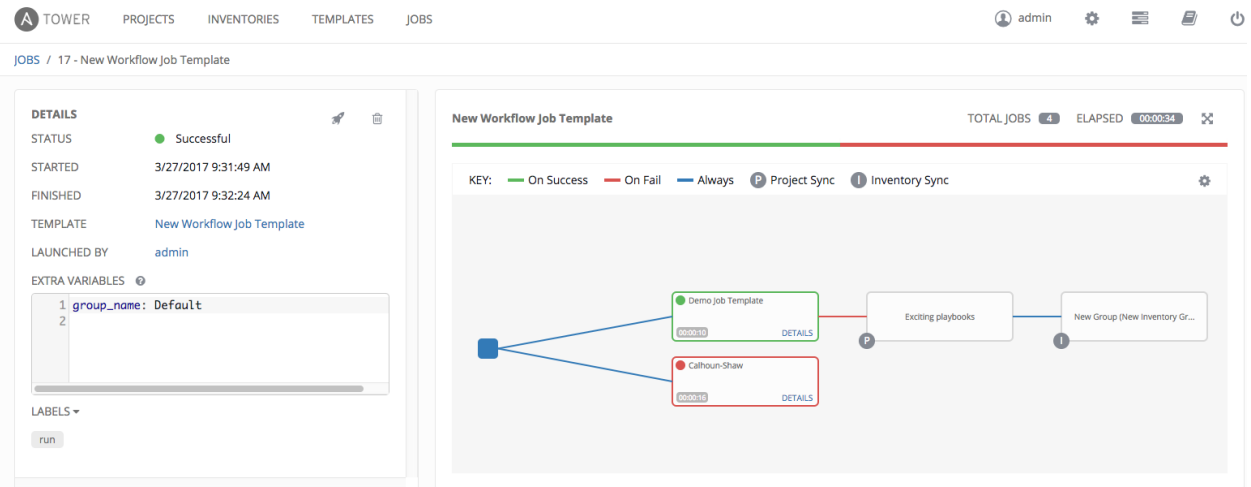
To launch the workflow job template:

1. Access the workflow job template 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.



2. Click the  icon next to the workflow you want to 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.



16.8 Copy a Workflow Job Template

Ansible Tower allows you the ability to copy a workflow job template. If you choose to copy a workflow 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 workflow job template. The user copying the workflow job template will be granted the admin permission, but no permissions are assigned (copied) to the workflow job template.

1. Access the workflow job 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.

The screenshot shows the 'TEMPLATES' section in Ansible Tower. It features a search bar and a '+ KEY' button. Three templates are listed:

- Demo Job Template** (Job Template): Activity is shown with a progress bar. Inventory is 'Demo Inventory', Project is 'Demo Project', and Credentials is 'Demo Credential'. Last modified and last ran are both '9/11/2018 11:33:43 PM by admin'.
- Example Job Template** (Job Template): Inventory is 'Network Inventory Small', Project is 'Demo Project', and Last modified is '9/12/2018 5:09:48 AM by admin'.
- Super workflow** (Workflow Template): Last modified is '9/12/2018 5:21:01 AM by admin' and has a 'run' label.

2. Click the  button.

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

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.

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 job template, if any of its nodes uses an unauthorized job template, inventory, or credential, the workflow job template can still be copied. But in the copied workflow job template, the corresponding fields in the workflow job template node will be absent.

16.9 Extra Variables

Note: Additional strict `extra_vars` validation was added in Ansible Tower 3.0.0. `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.

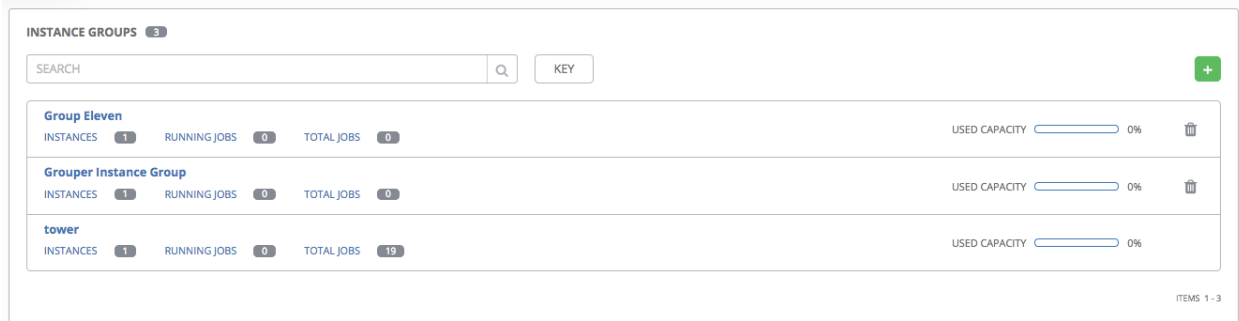
The following table notes the behavior (hierarchy) of variable precedence in Ansible Tower as it compares to variable precedence in Ansible.

Ansible Tower Variable Precedence Hierarchy (last listed wins)

Ansible	Tower
	set_stats (i.e. artifacts)
custom facts	<div style="border: 1px solid red; border-radius: 15px; padding: 5px;"> Job Artifacts Workflow Job Template extra variables Workflow Job Template Survey (defaults) Workflow Job Launch extra variables </div>


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:



17.1 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.

The 'CREATE INSTANCE GROUP' form includes the following fields and controls:


- DETAILS** (selected), INSTANCES, JOBS (tabs)
- NAME** (text input)
- POLICY INSTANCE MINIMUM** (text input)
- POLICY INSTANCE PERCENTAGE** (slider, currently at 0%)
- POLICY INSTANCE LIST** (text input with search icon)
- CANCEL** and **SAVE** buttons

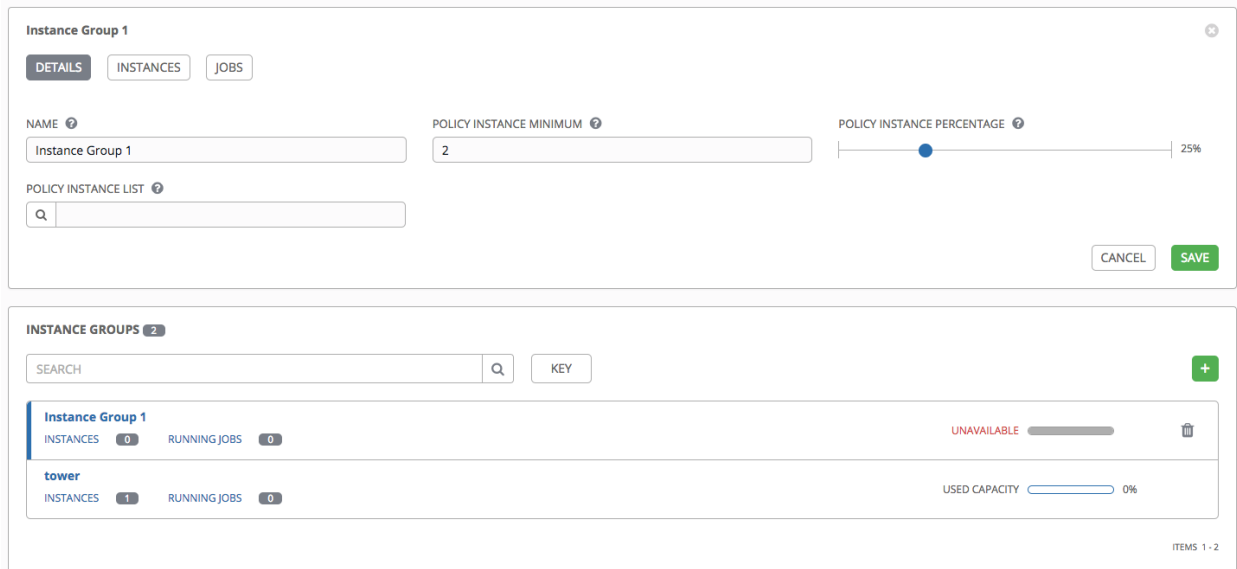
3. Enter the appropriate details into the following fields:

- **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.
- **Policy Instance List.** Specify instances you want to assign to this group.

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.

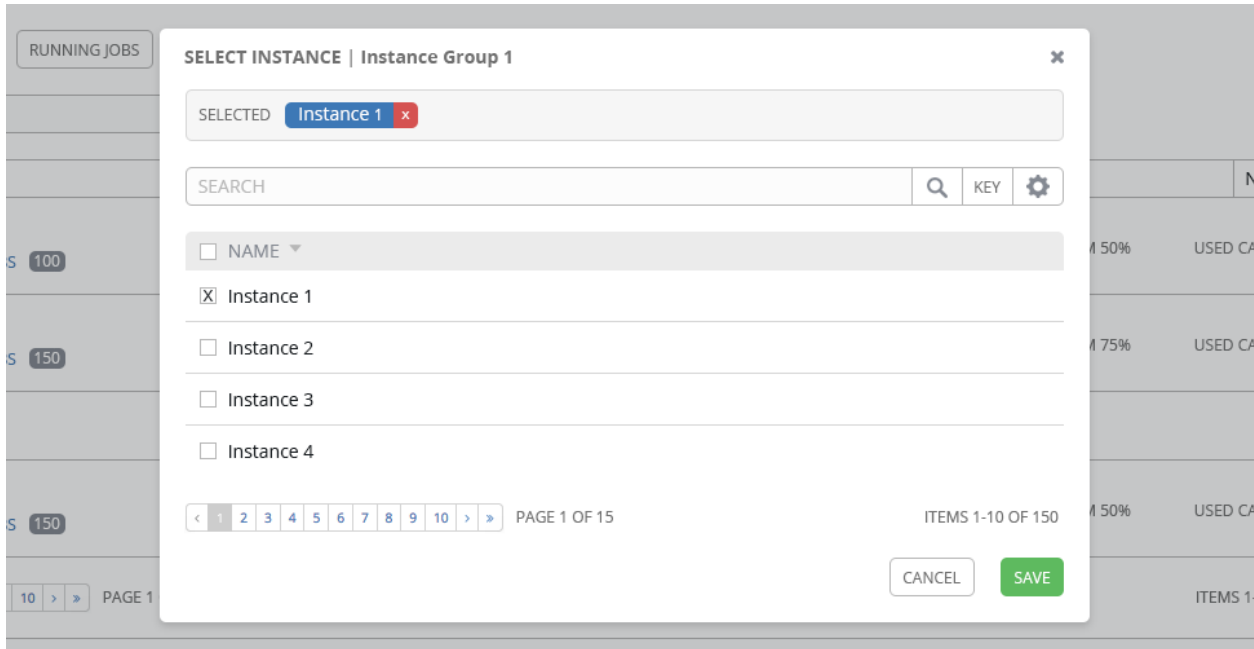


The screenshot displays two parts of the Ansible Tower interface. The top part is the 'Instance Group 1' details form, which includes tabs for 'DETAILS', 'INSTANCES', and 'JOBS'. The 'DETAILS' tab is active, showing fields for 'NAME' (Instance Group 1), 'POLICY INSTANCE MINIMUM' (2), 'POLICY INSTANCE PERCENTAGE' (25%), and 'POLICY INSTANCE LIST'. There are 'CANCEL' and 'SAVE' buttons at the bottom right. The bottom part of the screenshot shows a list of 'INSTANCE GROUPS' with a search bar and a '+ KEY' button. The list contains two entries: 'Instance Group 1' with 0 instances and 0 running jobs, and 'tower' with 1 instance and 0 running jobs. A 'USED CAPACITY' bar is shown at 0%.

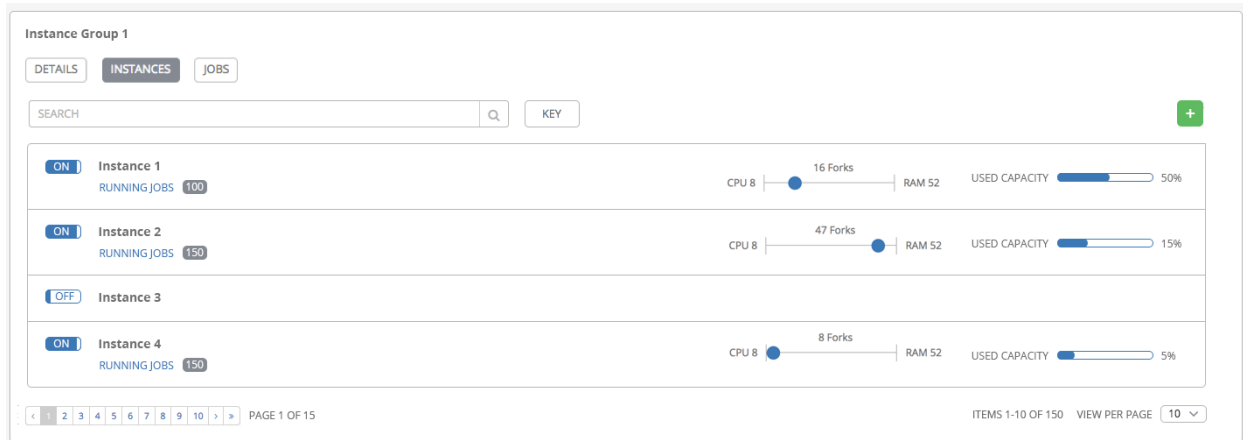
17.1.1 Associate instances to an instance group

To associate instances to an instance group:

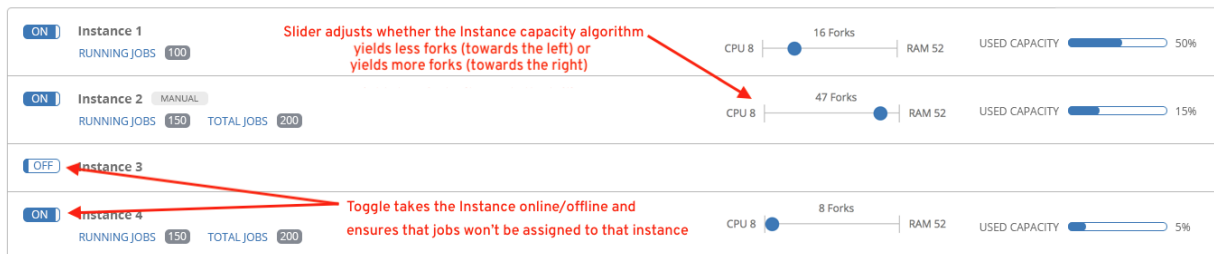
1. Click the **Instances** tab of the Instance Group window and click the  button.
2. Click the checkbox next to one or more available instances from the list to select the instance(s) you want to add to the instance group.



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



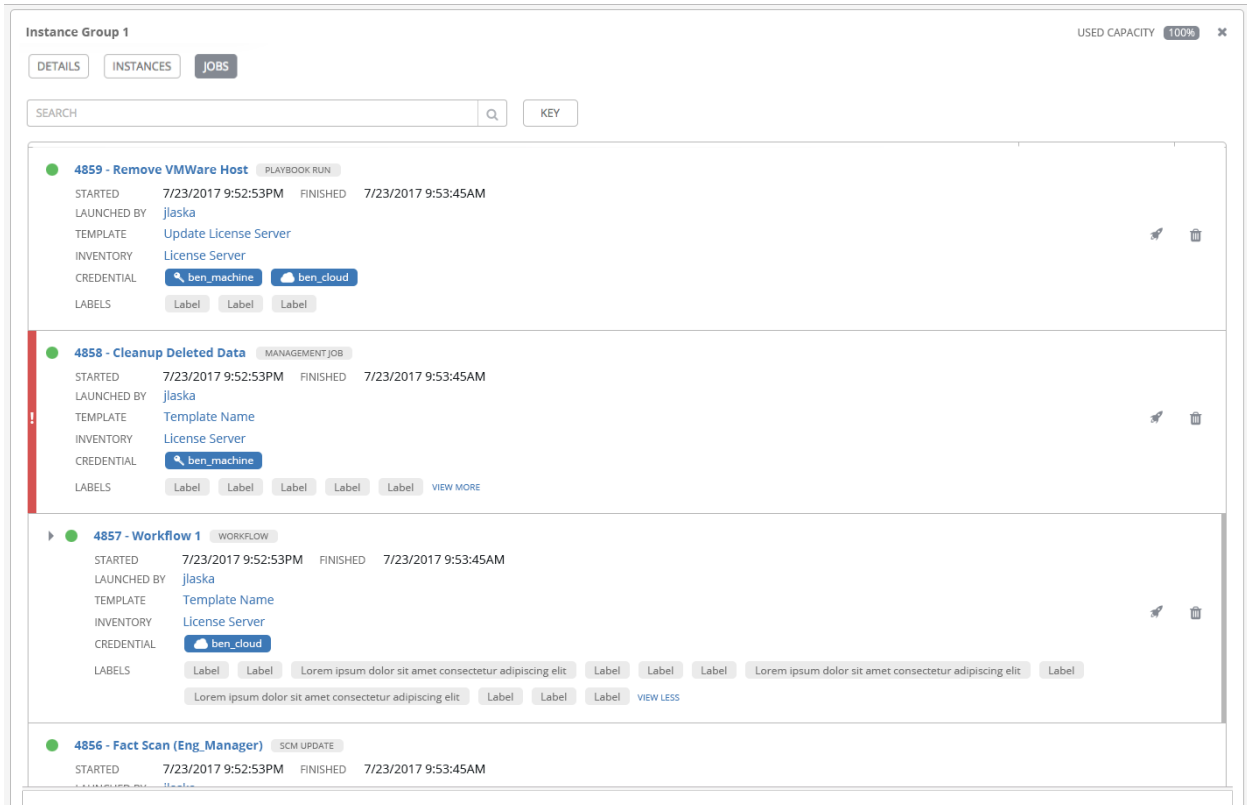
This view also allows you to edit some key attributes associated with the instances in your instance group:



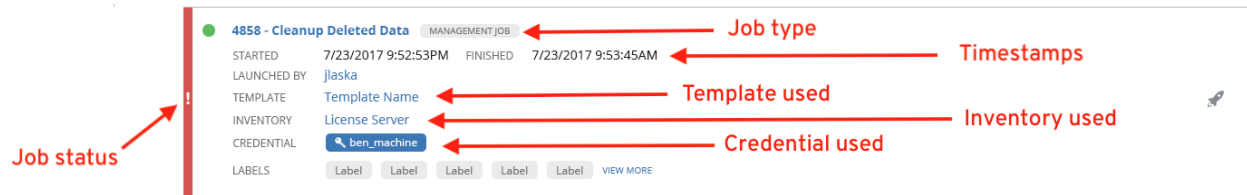
17.1.2 View jobs associated with an instance group

To view the jobs associated with the instance group:

1. Click the **Jobs** tab of the Instance Group window.



2. Each job displays the job status, ID, and name; type of job, time started and completed, who started the job; and which template, inventory, and credential were used.

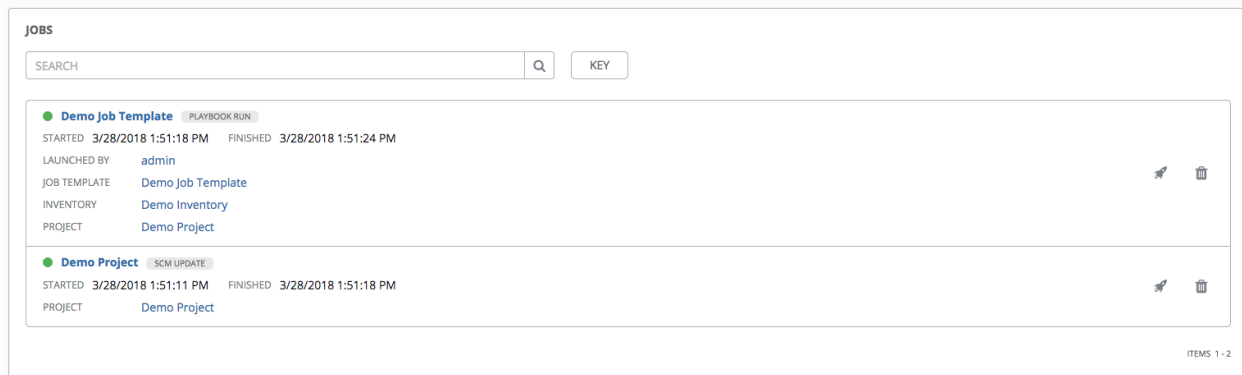


The instances are run in accordance with instance group policies. Refer to [Instance Group Policies](#) in the *Ansible Tower Administration Guide*.

JOBS

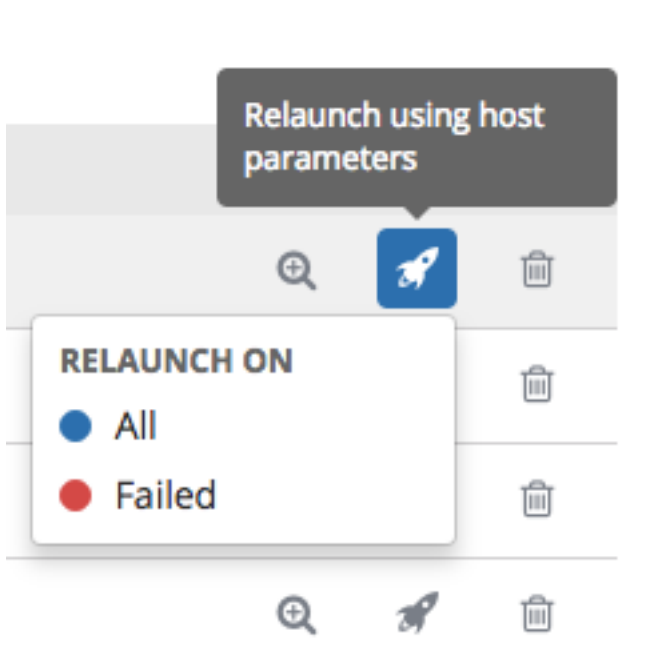
A **job** is an instance of Tower launching an Ansible playbook against an inventory of hosts.

The Jobs link displays a list of jobs and their status—shown as completed successfully or failed, or as an active (running) job. Actions you can take from this screen include viewing the details and standard output of a particular job, relaunch jobs, or remove jobs.



Starting with Ansible Tower 3.3, 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 a system job, project/inventory updates, system jobs, workflow jobs, etc.

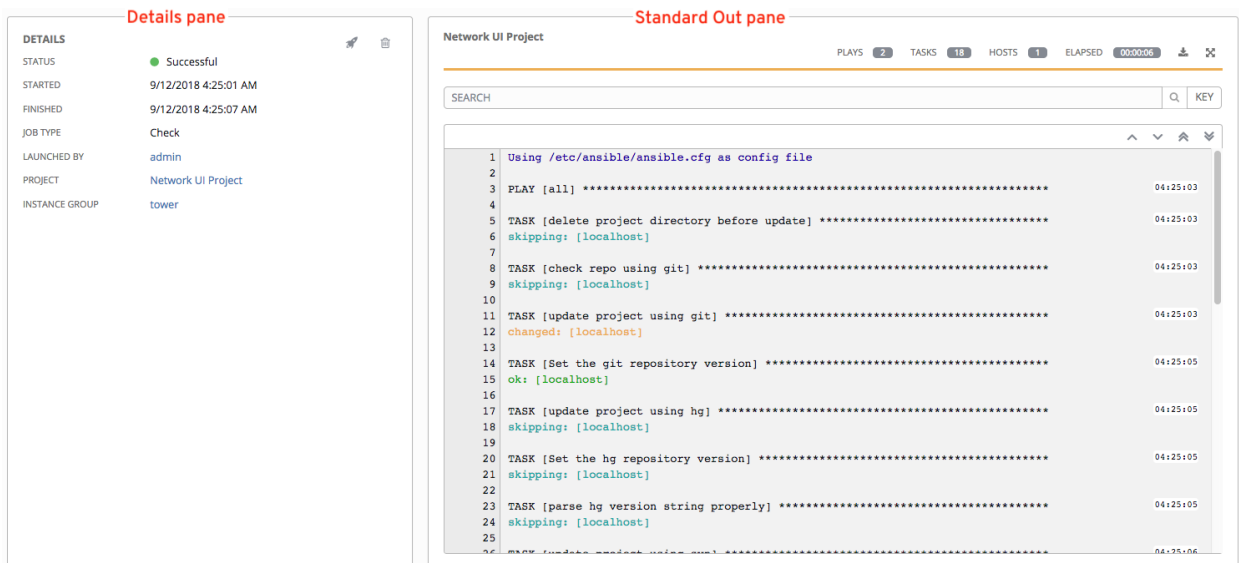


- Selecting **All**, relaunches all the hosts.
- Selecting **Failed**, relaunches all failed and unreachable hosts.

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:

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



18.1 Job Details - Inventory Sync

JOBS / Custom Inventory (inventory-custom - 34) - inventory-custom - 280

RESULTS

NAME Custom Inventory (inventory-custom - 34) - inventory-custom - 280

STATUS ● Successful

EXPLANATION

LICENSE ERROR False

STARTED 9/27/2017 8:28:40 PM

FINISHED 9/27/2017 8:28:59 PM

ELAPSED 18.918 seconds

LAUNCH TYPE Manual

SOURCE Custom Script

OVERWRITE True

OVERWRITE VARS False



STANDARD OUT

```

14.487 INFO Updating inventory 8: inventory-custom
14.569 INFO Reading Ansible inventory source: /tmp/awx_inventory_paBj8p/tmpdn3C9B
15.043 INFO Processing JSON output...
15.044 INFO Loaded 1 groups, 5 hosts
15.161 WARNING Group "Custom Inventory" from v1 API is not deleted by overwrite
15.164 INFO Group "Custom Inventory" from v1 API child group/host connections preserved
15.185 INFO Inventory variables unmodified
15.189 INFO Group "hosts" variables unmodified
15.212 INFO Host "host-00" variables unmodified
15.212 INFO Host "host-01" variables unmodified
15.212 INFO Host "host-02" variables unmodified
15.212 INFO Host "host-03" variables unmodified
15.212 INFO Host "host-04" variables unmodified
15.235 INFO Host "host-00" already in group "hosts"
15.235 INFO Host "host-01" already in group "hosts"
15.235 INFO Host "host-02" already in group "hosts"
15.235 INFO Host "host-03" already in group "hosts"
15.235 INFO Host "host-04" already in group "hosts"
15.529 INFO Inventory import completed for Custom Inventory (inventory-custom - 34) - inventory-custom - 280 in 1.

```

18.1.1 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 provides details on the job execution:

- **Name:** The name of the associated inventory group.
- **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 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 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.
- **Explanation:** Describes reason(s) for failure.
- **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.
- **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*, *Scheduled*, or *Dependency*
- **Credential:** The credential used in this inventory sync.

- **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.

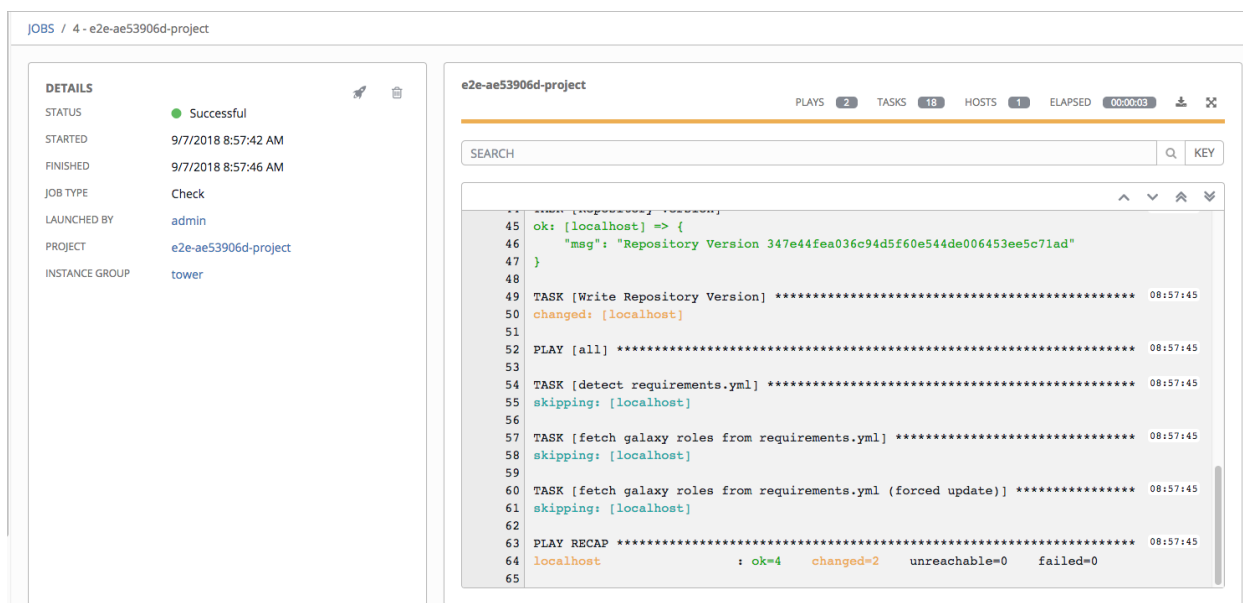
By clicking on these items, where appropriate, you can view the corresponding job templates, projects, and other Tower objects.

18.1.2 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 ().



Starting in Ansible Tower 3.3, 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 detail, refer to the [ANSIBLE_DISPLAY_ARGS_TO_STDOUT](#).

18.2 Job Details - SCM



The screenshot displays the 'Job Details - SCM' interface for a job named 'e2e-ae53906d-project'. On the left, a 'DETAILS' sidebar shows the job's status as 'Successful', started on 9/7/2018 at 8:57:42 AM, and finished at 8:57:46 AM. The job type is 'Check', launched by 'admin', and associated with the 'tower' instance group. The main area shows the 'Standard Out' output, which includes a search bar and a list of tasks. The tasks shown are: 'Write Repository Version' (changed), 'detect requirements.yml' (skipping), and 'fetch galaxy roles from requirements.yml' (skipping). A 'PLAY RECAP' at the bottom indicates 4 OK, 2 changed, 0 unreachable, and 0 failed.

18.2.1 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 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 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.



18.2.2 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 ().

18.3 Job Details - Playbook Run

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

18.3.1 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) 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 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.

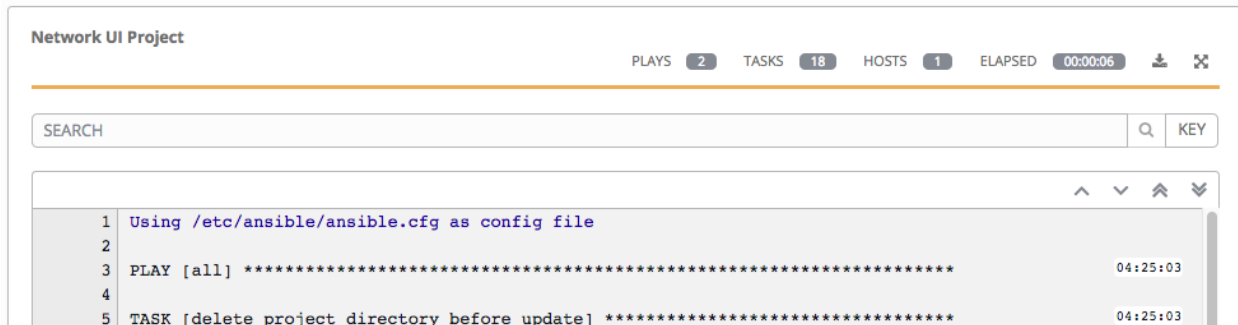
18.3.2 Standard Out Pane

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

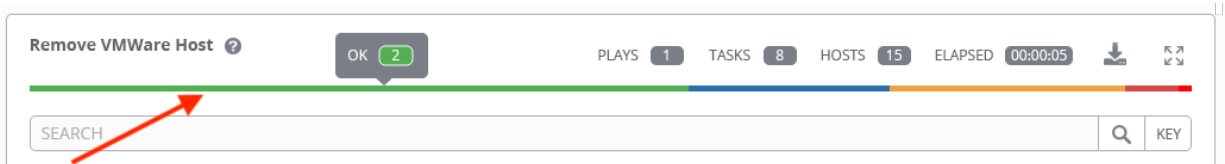
The events summary captures a tally of events that were run as part of this playbook:

- 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.

The example below shows a search with only failed hosts.

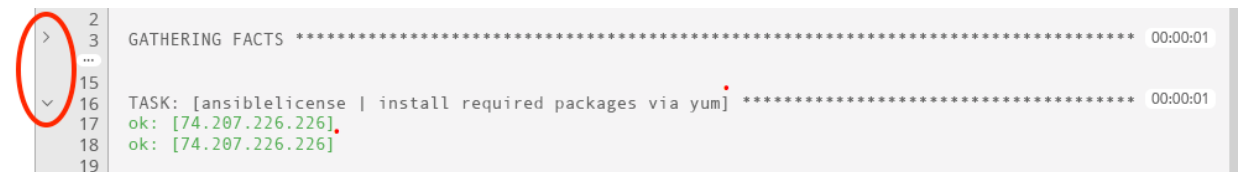


For more details about using the Tower Search, refer to the [Search](#) chapter.

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.

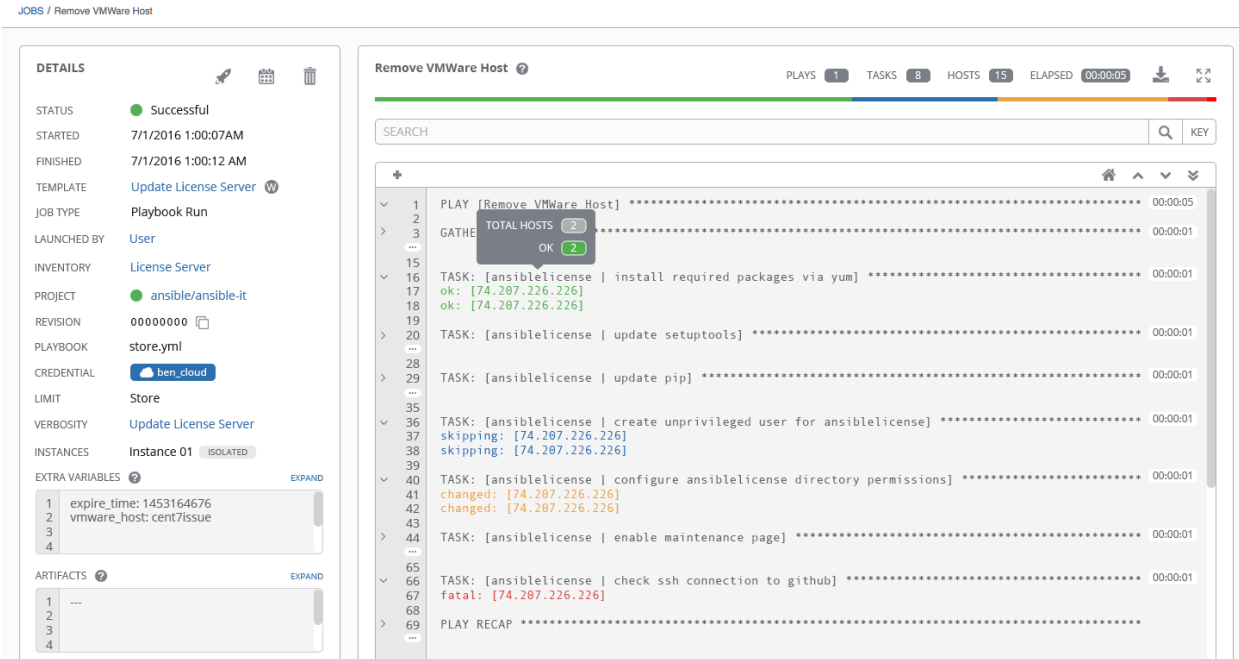
Alternatively, you can display all the details of a specific play or task by clicking on the arrow icons next to them. Click an arrow from sideways to downward to expand the lines associated with that play or task. Click the arrow back to the sideways position to collapse and hide the lines.



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.

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

The screenshot shows a modal window titled 'localhost' with the following details:

- CREATED: 9/14/2018 1:30:30 AM
- ID: 799537
- PLAY: Build and push Tower Development Container Image
- TASK: Cleanup Tower
- MODULE: file

Below the details is a 'JSON' tab showing the following output:

```

1 {
2   "_ansible_parsed": true,
3   "_ansible_no_log": false,
4   "changed": true,
5   "state": "absent",
6   "diff": {
7     "after": {
8       "path": "/tmp/tower",
9       "state": "absent"
10    }
11  }

```

The background interface shows a job list on the left and a terminal output window on the right. The terminal output includes a 'PLAY RECAP' section:

```

36 PLAY RECAP *****
37 *****
38 localhost : ok=10  changed=8  unrea
39 0  failed=0

```

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

18.4 Ansible Tower Capacity Determination and Job Impact

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, and 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 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.

18.4.1 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 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.

18.4.2 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 let use Ansible's default forks value of five. Ansible defaults to five forks, but will use less if your job has less than five hosts. In general, setting a forks value higher than what the system is capable of could cause trouble 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.

An 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. In the above example, that would be 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 [] Q KEY [] [+]

<input checked="" type="checkbox"/> Instance 1 RUNNING JOBS 100	CPU 8 ● 16 Forks RAM 52	USED CAPACITY 50%
<input checked="" type="checkbox"/> Instance 2 RUNNING JOBS 150	CPU 8 ● 47 Forks RAM 52	USED CAPACITY 15%
<input type="checkbox"/> Instance 3		
<input checked="" type="checkbox"/> Instance 4 RUNNING JOBS 150	CPU 8 ● 8 Forks RAM 52	USED CAPACITY 5%

Slider adjusts whether the Instance capacity algorithm yields less forks (towards the left) or yields more forks (towards the right)

< 1 2 3 4 5 6 7 8 9 10 > PAGE 1 OF 15 ITEMS 1-10 OF 150 VIEW PER PAGE 10

NOTIFICATIONS

A **Notifier** 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 notifier
- The token and a list of channels are needed for a Slack notifier
- The URL and Headers are needed for a Webhook notifier

A Notification is a manifestation of the notifier; for example, when a job fails, a notification is sent using the configuration defined by the Notifier.

At a high level, the typical flow for the notification system works as follows:

- A user creates a notifier to the Tower REST API at the `/api/v2/notifiers` endpoint (either through the API or through the Tower UI).
- A user assigns the notifier 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 (error, success, or any). For example a user may wish to assign a particular Notifier to trigger when Job Template 1 fails. In which case, they will associate the notifier with the job template at `/api/v2/job_templates/n/notifiers_error` API endpoint.

19.1 Notifier Hierarchy

Notifiers assigned at certain levels will inherit notifiers defined on parent objects as such:

- Job Templates will use notifiers defined on it as well as inheriting notifiers from the Project used by the Job Template and from the Organization that it is listed under (via the Project).
- Project Updates will use notifiers defined on the project and will inherit notifiers from the Organization associated with it
- Inventory Updates will use notifiers defined on the Organization that it is listed under
- Ad-hoc commands will use notifiers defined on the Organization that the inventory is associated with

19.2 Workflow

When a job succeeds or fails, the error or success handler will pull a list of relevant notifiers 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

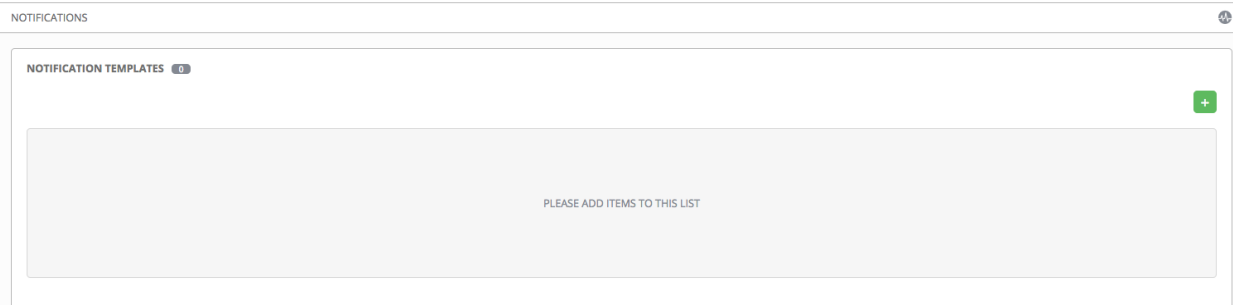
related resources on job types (jobs, inventory updates, project updates), and also at `/api/v2/notifications`. You may also see what notifications have been sent from a notifier by examining its related resources.

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>`).

19.3 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, a description, and the organization it belongs to in their respective fields.
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.

19.4 Notification Types

Topics:

- *Email*
- *Slack*

- *Twilio*
- *PagerDuty*
- *HipChat*
- *Webhook*
- *Mattermost*
- *Rocket.Chat*
- *IRC*

Each of these have their own configuration and behavioral semantics and testing them may need to be approached in different ways. The following sections will give as much detail as possible.

19.4.1 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: - Username - Host - Sender email - Recipient list - Password - Port

NEW NOTIFICATION TEMPLATE

* NAME Tell Me	DESCRIPTION notification test	* ORGANIZATION Honey Dog, Inc.
* TYPE Email		
TYPE DETAILS		
* USERNAME luckydog	* HOST honeydog.com	* SENDER EMAIL info@honeydog.com
* RECIPIENT LIST engineering@honeydog.com release-managers@honeydog.com	* PASSWORD SHOW <small>SHOW</small>	* PORT 1
OPTIONS <input type="checkbox"/> Use TLS <input type="checkbox"/> Use SSL		
<input type="button" value="CANCEL"/>		<input type="button" value="SAVE"/>

Caution: TLS and SSL connections are mutually exclusive and should not be selected at the same time. Be sure to only select one—checking both causes the notification to silently fail.

19.4.2 Slack

Slack, a collaborative team communication and messaging tool, is pretty easy to configure.

You must supply the following to setup Slack notifications:

- A token (which you can obtain from creating a bot in the integrations settings for the Slack team at <https://api.slack.com/bot-users>)

- Destination channel(s)

You must also invite the notification bot to join the channel(s) in question. Note that private messages are not supported.

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

- * NAME:** Tell Me
- DESCRIPTION:** notification test
- * ORGANIZATION:** Honey Dog, Inc.
- * TYPE:** Slack
- TYPE DETAILS:**
 - * DESTINATION CHANNELS:** #engineering, #helpdesk, #support
 - * TOKEN:** [SHOW] [.....]

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

19.4.3 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 phone number (this will be the list of numbers to receive the SMS and should be the 10-digit phone number)
- Account SID

NEW NOTIFICATION TEMPLATE ✕

<p>* NAME</p> <input type="text" value="Tell Me"/>	<p>DESCRIPTION</p> <input type="text" value="notification test"/>	<p>* ORGANIZATION</p> <input type="text" value="Honey Dog, Inc."/>
<p>* TYPE</p> <input type="text" value="Twilio"/>		
<p>TYPE DETAILS</p>		
<p>* ACCOUNT TOKEN</p> <input type="text" value="SHOW"/> <input type="text" value="....."/>	<p>* SOURCE PHONE NUMBER ⓘ</p> <input type="text" value="9109876555"/>	<p>* DESTINATION SMS NUMBER ⓘ</p> <input type="text" value="9109676565"/>
<p>* ACCOUNT SID</p> <input type="text" value="Aa54e6dg4345d3t676f60"/>		
		<input type="button" value="CANCEL"/> <input type="button" value="SAVE"/>

19.4.4 PagerDuty

PagerDuty is a fairly straightforward integration. The user must 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 options of note 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.

NEW NOTIFICATION TEMPLATE ✕

<p>* NAME</p> <input type="text" value="Tell Me"/>	<p>DESCRIPTION</p> <input type="text" value="notification test"/>	<p>* ORGANIZATION</p> <input type="text" value="Honey Dog, Inc."/>
<p>* TYPE</p> <input type="text" value="Pagerduty"/>		
<p>TYPE DETAILS</p>		
<p>* API TOKEN</p> <input type="text" value="SHOW"/> <input type="text"/>	<p>* PAGERDUTY SUBDOMAIN</p> <input type="text"/>	<p>* API SERVICE/INTEGRATION KEY</p> <input type="text"/>
<p>* CLIENT IDENTIFIER</p> <input type="text"/>		
		<input type="button" value="CANCEL"/> <input type="button" value="SAVE"/>

19.4.5 HipChat

There are several ways to integrate with HipChat. The Tower implementation uses HipChat “Integrations”. Currently you can find this at the bottom right of the main HipChat webview. From there, you will select “Build your own Integration”. After creating that, it will list the `auth_token` that needs to be supplied to Tower. Some other relevant details on the fields accepted by Tower for the HipChat notification type:

- **Destination Channels:** Channels which should receive the notification (“engineering” or “#support”, for example).
- **Token:** The token listed after building your own HipChat integration.
- **Label to be shown with notification:** Along with the integration name itself this will put another label on the notification (which could be helpful if multiple services are using the same integration to distinguish them from each other).
- **API URL:** The URL of the Hipchat API service. If you create a team hosted by them it will be something like: `https://team.hipchat.com`. For a self-hosted integration, use a base URL similar to `https://hipchat.yourcompany.com/` and add in appropriate Destination Channels without the # leading them (“engineering” rather than “#engineering”).
- **Notification Color:** This will highlight the message as the given color. If set to something HipChat does not expect, then the notification will generate an error in the given color.
- **Notify Channel:** Selecting this will cause the bot to “notify” channel members. Normally it will just be stuck as a message in the chat channel without triggering anyone’s notifications. This option will notify users of the channel respecting their existing notification settings (browser notification, email fallback, etc.).

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

- * NAME:** Tell Me
- DESCRIPTION:** notification test
- * ORGANIZATION:** Honey Dog, Inc.
- * TYPE:** HipChat
- TYPE DETAILS:**
 - * DESTINATION CHANNELS:** #engineering, #support
 - * TOKEN:** SHOW [REDACTED]
 - * LABEL TO BE SHOWN WITH NOTIFICATION:** hipchat-notifier-tell-me
- * API URL:** https://mycompany.hipchat.com
- * NOTIFICATION COLOR:** red
- NOTIFY CHANNEL:**

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

19.4.6 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.

The parameters are pretty straightforward:

- **Target URL:** The full URL that will be POSTed to
- **HTTP Headers:** Headers in JSON form where the keys and values are strings. For example:

```
{ "Authentication": "988881adc9fc3655077dc2d4d757d480b5ea0e11", "MessageType":
↪ "Test" }
```

NEW NOTIFICATION TEMPLATE

* NAME: Tell Me DESCRIPTION: notification test * ORGANIZATION: Honey Dog, Inc.

* TYPE: Webhook

TYPE DETAILS

* TARGET URL: http://www.honeydog.com/web/db/notification

* HTTP HEADERS

```
1 { "Authentication": "988881adc9fc3655077dc2d4d757d480b5ea0e11", "MessageType": "Test" }
2
```

CANCEL SAVE

19.4.7 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 notifier
- 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: Tell me DESCRIPTION: notification test * ORGANIZATION: Honey Dog, Inc.

* TYPE: Mattermost

TYPE DETAILS

* TARGET URL: http://1.2.3.4:8065/hooks/j5kurmybl513b4pnf9sdpl USERNAME: beth CHANNEL: my-channel

ICON URL: https://www.myicon/favicon.ico DISABLE SSL VERIFICATION

CANCEL SAVE

19.4.8 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 notifier
- 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: Tell me

DESCRIPTION: notification test

* ORGANIZATION: Honey Dog, Inc.

* TYPE: Rocket.Chat

TYPE DETAILS

* TARGET URL: http://1.2.3.4:8065/hooks/dj4ruwjew847w84q9330f

USERNAME: jerry

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

DISABLE SSL VERIFICATION

CANCEL SAVE

19.4.9 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: 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: Should the bot use SSL when connecting

NEW NOTIFICATION TEMPLATE ✕

<p>* NAME</p> <input type="text" value="Tell Me"/>	<p>DESCRIPTION</p> <input type="text" value="notification test"/>	<p>* ORGANIZATION</p> <input type="text" value="Honey Dog, Inc."/>
<p>* TYPE</p> <input type="text" value="IRC"/>		
TYPE DETAILS		
<p>* IRC SERVER PASSWORD</p> <input style="width: 100%;" type="text" value="SHOW"/>	<p>* IRC SERVER PORT</p> <input type="text" value="6667"/>	<p>* IRC SERVER ADDRESS</p> <input type="text" value="irc.testirc.net"/>
<p>* IRC NICK</p> <input type="text" value="helpbot"/>	<p>* DESTINATION CHANNELS OR USERS ⓘ</p> <div style="border: 1px solid #ccc; padding: 5px; min-height: 40px;"> <p>#engineering #release-engineers</p> </div>	<p><input type="checkbox"/> SSL CONNECTION</p>

19.5 Configuring the towerhost hostname

In `/etc/tower/settings.py`, you can modify `TOWER_URL_BASE='https://tower.example.com'` to change the notification hostname, replacing `https://tower.example.com` with your preferred hostname. You must restart Tower services after saving your changes with `ansible-tower-service restart`.

Refreshing your Tower license also changes the notification hostname. New installations of Ansible Tower 3.0 should not have to set the hostname for notifications.

19.5.1 Resetting 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.

WORKFLOWS

Workflows allow you to configure a sequence of disparate job 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.

Note: Workflows are only available to those with Enterprise-level licenses.

Job templates are linked together using a graph-like structure called nodes. Job template nodes are associated with job templates. A job template can be a 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.

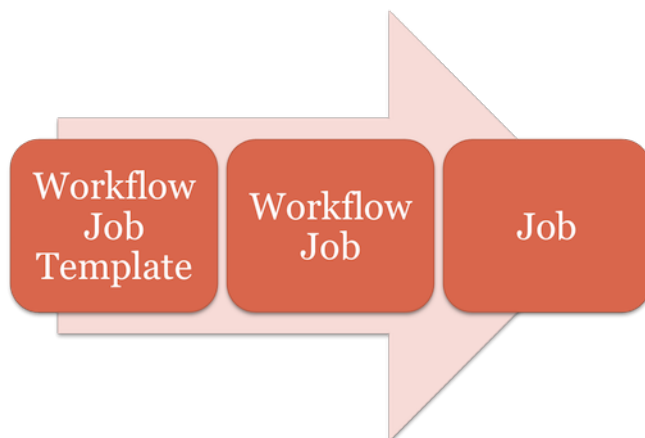
As the workflow runs, jobs are spawned from the node’s linked template. Nodes linking to a job template which has prompt-driven fields (job_type, job_tags, skip_tags, limit) can contain those fields, and will not be prompted on launch. Job templates with promptable credential and/or inventory, WITHOUT defaults, will not be available for inclusion in a workflow.

A node can have only one parent and can only have children that is linked to a state 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 attempt to launch a workflow job template that has the following missing pieces, the user interface will notify you as a warning but will still proceed:

- Job template deleted from the node
- A prompted field is provided, but the job template is not set to prompt on launch for the field

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.

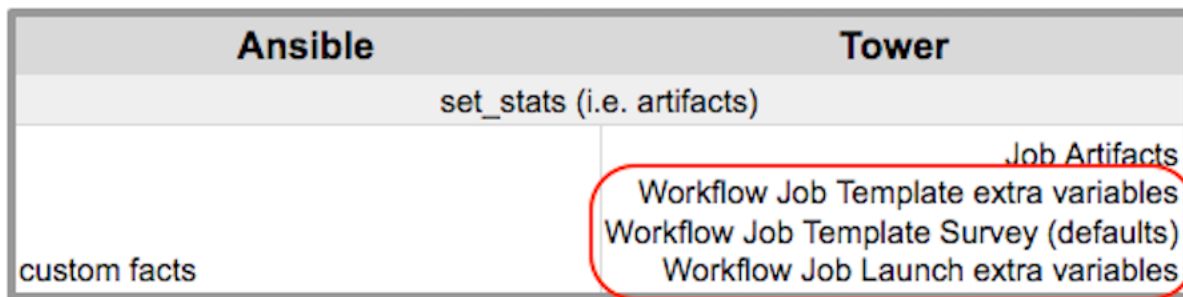


It is possible to launch several workflows simultaneously, and set a schedule for when to launch them. You can set notifications on workflows, such as when a job completes, similar to that of job templates.

20.1 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:



In addition to the workflow `extra_vars`, jobs 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, version 2.2.2 or later.

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
```

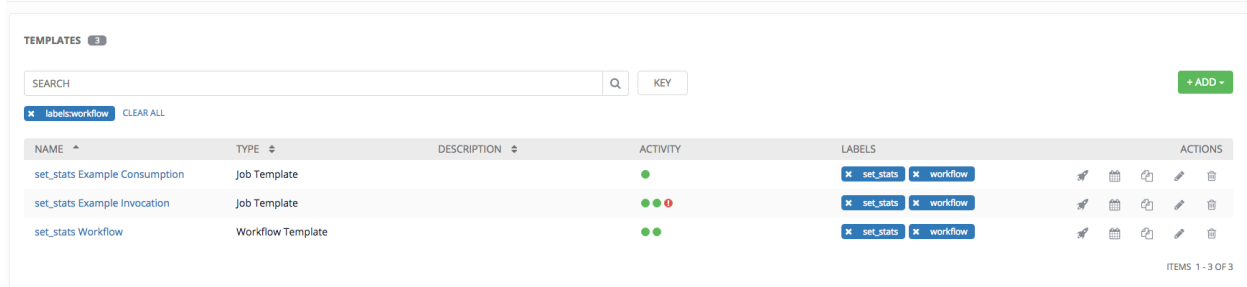
```
- name: "Output test results"
  debug:
    msg: "{{ results.content }}"
```

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.



NAME	TYPE	DESCRIPTION	ACTIVITY	LABELS	ACTIONS
set_stats Example Consumption	Job Template		●	set_stats workflow	[Refresh] [Calendar] [Copy] [Edit] [Delete]
set_stats Example Invocation	Job Template		●●●	set_stats workflow	[Refresh] [Calendar] [Copy] [Edit] [Delete]
set_stats Workflow	Workflow Template		●●	set_stats workflow	[Refresh] [Calendar] [Copy] [Edit] [Delete]

Note: For artifacts to work, keep the default setting, `per_host = False` in the `set_stats` module.

20.2 Workflow States

The workflow job can have the following states (no Failed state):

- Waiting
- Running
- Success (finished)
- Cancel
- Error

In the workflow scheme, canceling a job cancels the branch, while canceling the workflow job cancels the entire workflow. Deleting a job template does not delete the job node, but will indicate that it is invalid by displaying in the user interface, a broken link in the workflow, which prompts for correction without adverse impact to the structure of the workflow.

20.3 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](#).

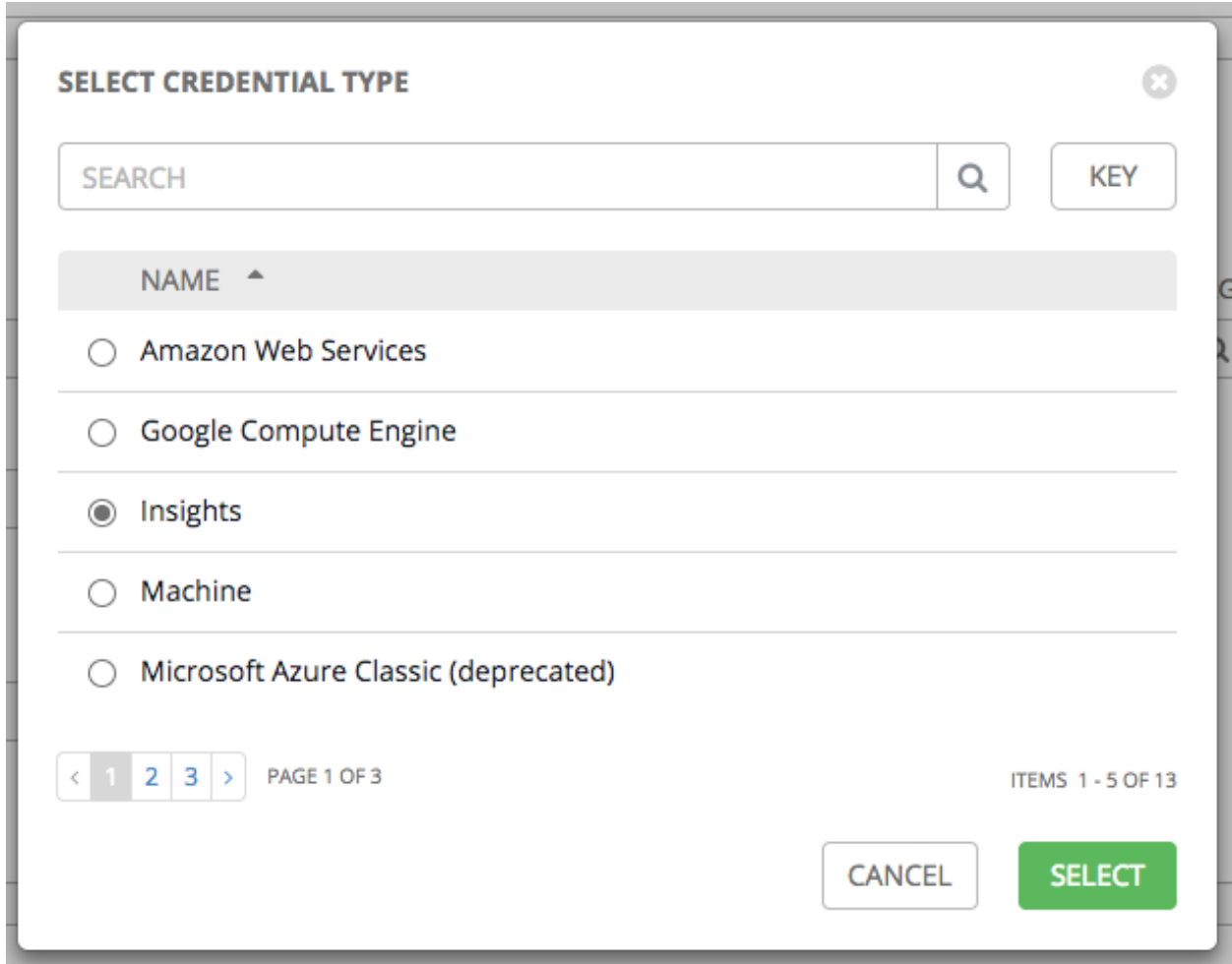
SETTING UP AN INSIGHTS PROJECT

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.

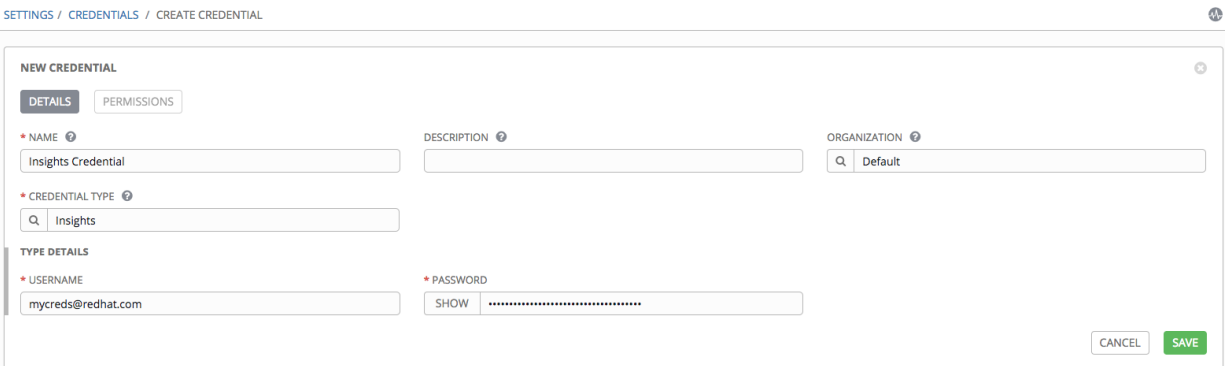
21.1 Create Insights Credential

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.








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.

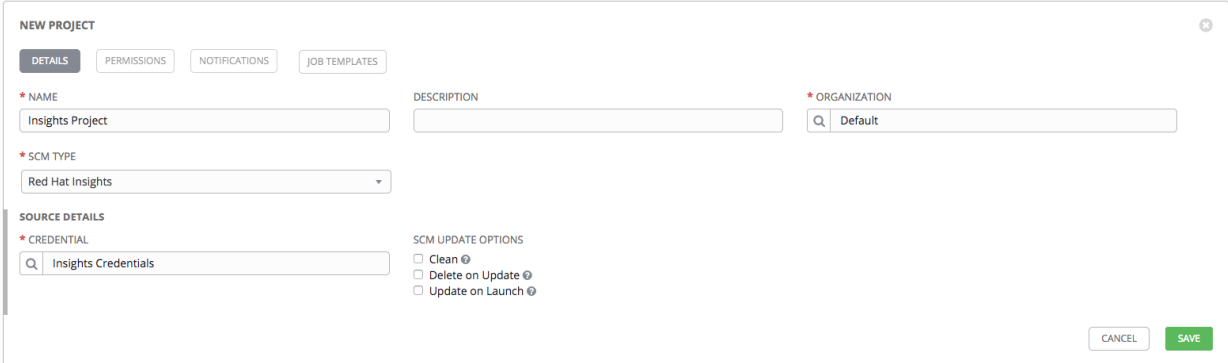


8. Click **Save** when done.

21.2 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 NOTIFICATIONS JOB TEMPLATES


* NAME: Insights Project DESCRIPTION: ORGANIZATION: Default


* SCM TYPE: Red Hat Insights


SOURCE DETAILS

* CREDENTIAL: Insights Credentials

SCM UPDATE OPTIONS


Clean 

Delete on Update 

Update on Launch 

CANCEL SAVE

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 2

SEARCH Q KEY + ADD


NAME ^	TYPE ^	REVISION ^	LAST UPDATED	ACTIONS
<input type="radio"/> Demo Project	Git			
<input checked="" type="radio"/> Insights Project	Red Hat Insights	644f3-8	10/9/2017 11:16:19 PM	

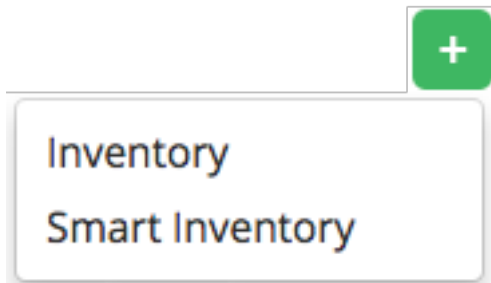
ITEMS 1-2



21.3 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  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.

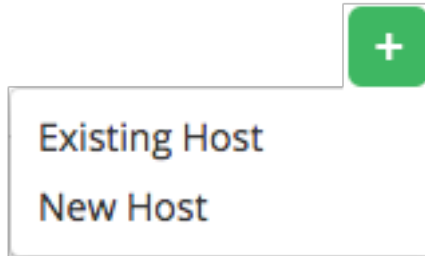
INVENTORIES / Insights Inventory


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.

21.4 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.

PROJECTS / CREATE PROJECT

NEW PROJECT

DETAILS | PERMISSIONS | NOTIFICATIONS | JOB TEMPLATES

* NAME: DESCRIPTION: * ORGANIZATION:

* SCM TYPE:

SOURCE DETAILS


* SCM URL: SCM BRANCH/TAG/COMMIT: SCM CREDENTIAL:

SCM UPDATE OPTIONS

Clean Delete on Update Update on Launch

CANCEL SAVE

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.

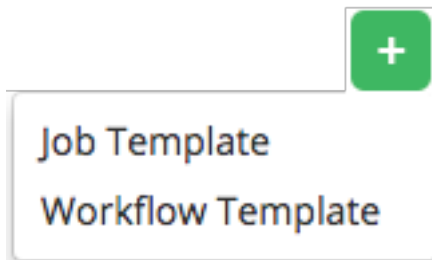
Syncing imports into Tower any Maintenance Plans in your Insights account that has a playbook solution. It will use the default Plan resolution. Notice that the status dot beside the name of the project updates once the sync has run.

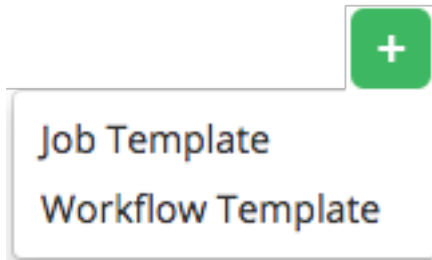
21.5 Create a Scan Job Template

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 **Use Fact Cache** from the Options field.

TEMPLATES / Insights Scan 🔍

Insights Scan ✖

DETAILS
PERMISSIONS
NOTIFICATIONS
COMPLETED JOBS
ADD SURVEY

<p>* NAME 🔗</p> <input type="text" value="Insights Scan"/>	<p>DESCRIPTION</p> <input type="text"/>	<p>*JOB TYPE 🔗 <input type="checkbox"/> PROMPT ON LAUNCH</p> <input type="text" value="Run"/>
<p>* INVENTORY 🔗 <input type="checkbox"/> PROMPT ON LAUNCH</p> <input type="text" value="Insights Inventory"/>	<p>* PROJECT 🔗</p> <input type="text" value="Scan Project"/>	<p>* PLAYBOOK 🔗</p> <input type="text" value="scan_facts.yml"/>
<p>CREDENTIAL 🔗 <input type="checkbox"/> PROMPT ON LAUNCH</p> <input type="text" value="DEMO CREDENTIAL"/>	<p>FORKS 🔗</p> <input type="text" value="DEFAULT"/>	<p>LIMIT 🔗 <input type="checkbox"/> PROMPT ON LAUNCH</p> <input type="text"/>
<p>* VERBOSITY 🔗 <input type="checkbox"/> PROMPT ON LAUNCH</p> <input type="text" value="0 (Normal)"/>	<p>INSTANCE GROUPS 🔗</p> <input type="text"/>	<p>JOB TAGS 🔗 <input type="checkbox"/> PROMPT ON LAUNCH</p> <input type="text"/>
<p>SKIP TAGS 🔗 <input type="checkbox"/> PROMPT ON LAUNCH</p> <input type="text"/>	<p>LABELS 🔗</p> <input type="text"/>	<p>SHOW CHANGES 🔗 <input type="checkbox"/> PROMPT ON LAUNCH</p> <input type="text" value="OFF"/>

OPTIONS

Enable Privilege Escalation 🔗
 Allow Provisioning Callbacks 🔗
 Enable Concurrent Jobs 🔗
 Use Fact Cache 🔗


EXTRA VARIABLES 🔗 PROMPT ON LAUNCH

YAML JSON

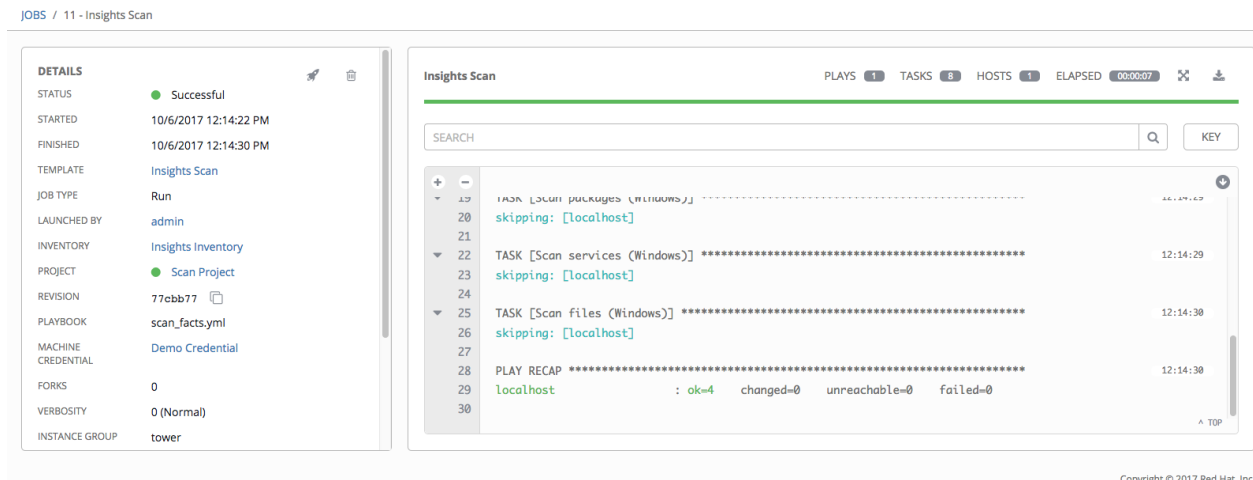
1 ---

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.



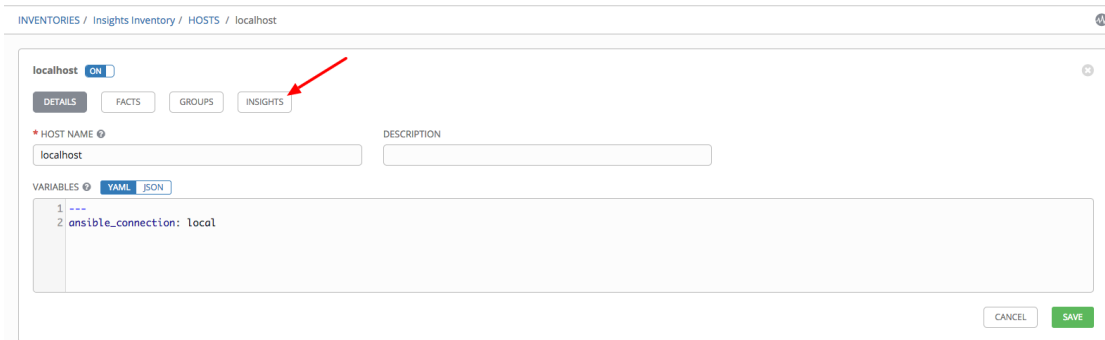
21.6 Remediate Insights Inventory

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 **ON**

DETAILS FACTS GROUPS **INSIGHTS**

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

ON localhost

ITEMS 1 - 1

Upon remediation, the New Job Template window opens. Notice the Inventory and Project fields are pre-populated.

NEW JOB TEMPLATE

DETAILS | PERMISSIONS | NOTIFICATIONS | COMPLETED JOBS | ADD SURVEY

* NAME

DESCRIPTION

* JOB TYPE PROMPT ON LAUNCH

* INVENTORY PROMPT ON LAUNCH

* PROJECT PROMPT ON LAUNCH

* PLAYBOOK

CREDENTIAL PROMPT ON LAUNCH

FORKS

LIMIT PROMPT ON LAUNCH

* VERBOSITY PROMPT ON LAUNCH

INSTANCE GROUPS

JOB TAGS PROMPT ON LAUNCH

SKIP TAGS PROMPT ON LAUNCH

LABELS


SHOW CHANGES OFF PROMPT ON LAUNCH

OPTIONS

Enable Privilege Escalation

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 | NOTIFICATIONS | COMPLETED JOBS | **ADD SURVEY**

* NAME: Maintenance Plan Job

DESCRIPTION: [Empty]

* JOB TYPE: Run PROMPT ON LAUNCH

* INVENTORY: Insights Inventory PROMPT ON LAUNCH

* PROJECT: Insights Project

* PLAYBOOK: ansiblefest_demo-33675.yml

CREDENTIAL: DEMO CREDENTIAL PROMPT ON LAUNCH

FORKS: DEFAULT

LIMIT: [Empty] PROMPT ON LAUNCH

* VERBOSITY: 0 (Normal) PROMPT ON LAUNCH

INSTANCE GROUPS: [Empty]

JOB TAGS: [Empty] PROMPT ON LAUNCH

SKIP TAGS: [Empty] PROMPT ON LAUNCH

LABELS: [Empty]

SHOW CHANGES: OFF PROMPT ON LAUNCH

OPTIONS


- Enable Privilege Escalation
- Allow Provisioning Callbacks
- Enable Concurrent Jobs
- Use Fact Cache

EXTRA VARIABLES: [YAML | JSON] PROMPT ON LAUNCH

```
1 ---
```

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

22.1 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.

22.2 Ansible file and directory structure

Please review the Ansible best practices from the Ansible documentation at http://docs.ansible.com/playbooks_best_practices.html. 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` Ansible 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` Ansible feature without a timeout, as Tower does not allow for interactively cancelling a pause. If you must use `pause`, ensure that you set a timeout.

Note: Playbooks should not use the `meta: refresh_inventory` Ansible feature, as it is incompatible with how Tower processes inventory, if a playbook does refresh inventory, it will refresh back to the start of the inventory the job began with. Instead, use a workflow with separate inventory refresh steps.

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.

22.3 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: With the release of Ansible Tower 2.4.0, edits and additions to Inventory host variables now persist beyond an inventory sync as long as `--overwrite_vars` is **not** set. To have inventory syncs behave as they did before, it is now required that both `--overwrite` and `--overwrite_vars` are set.

22.4 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.

22.5 Autoscaling

Using the “callback” feature to allow newly booting instances to request configuration is very useful for auto-scaling scenarios or provisioning integration.

22.6 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](#).

22.7 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, or use the [Tower CLI](#) tool. The credentials to the job template should not require prompting for any particular passwords. Using the API to spawn jobs is covered in the [Tower API guide](#).

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/settings.py` and set `AWX_PROOT_ENABLED` to `False`.

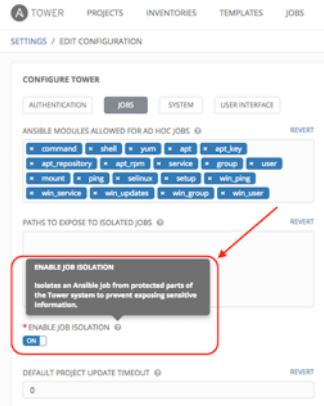
Note: In this scenario, playbooks have access to the file system and all that implies; therefore, users who have access to edit playbooks **must** be trusted.

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.

23.1 Playbook Access and Information Sharing

By default, Tower’s multi-tenant security prevents playbooks from reading files outside of their project directory. In older version of Ansible Tower a system called `proot` was used to isolate tower job processes from the rest of the system. For Tower version 3.1 and later, `bubblewrap` is used instead, due to its light weight and maintained process isolation system.

By default `bubblewrap` is enabled, but can be turned off via the Configure Tower screen in the Tower User Interface or from the tower settings file.



To access the Configure Tower screen, refer to the [Tower Configuration](#) section. To customize your bubblewrap settings through the settings file, navigate to the `/etc/tower/settings.py` file.

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
- `/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.

23.1.1 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. Scroll down until you see “Enable Job Isolation” and change the toggle button selection to **OFF** to disable bubblewrap support or select **ON** to enable it.

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 Isolation 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 to 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.

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.

23.2 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. In releases prior to Ansible Tower version 3.0, RBAC was thought of in terms of granting permissions to users or teams. Starting with Tower 3.0, RBAC is best thought of as granting roles to users or teams, which is a more intuitive approach.

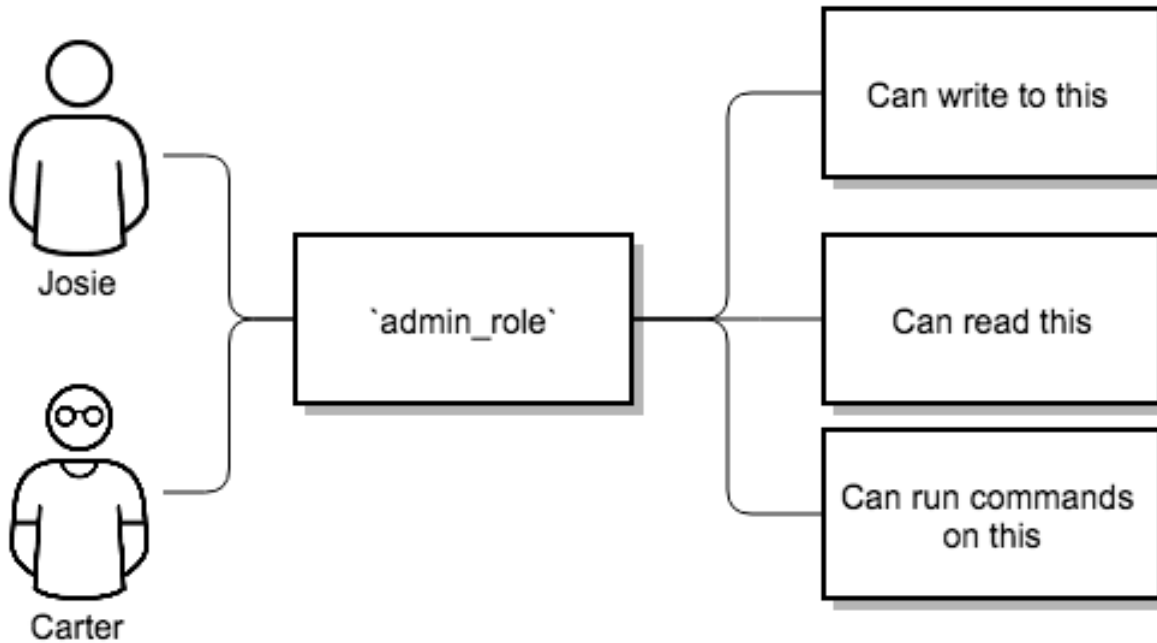
There are a few main concepts that you should become familiar with regarding Tower’s RBAC design—roles, resources, and users. Users can be members of a role, which gives them certain access to any resources associated with that role, or any resources associated with “descendant” roles.

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.

23.2.1 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`.

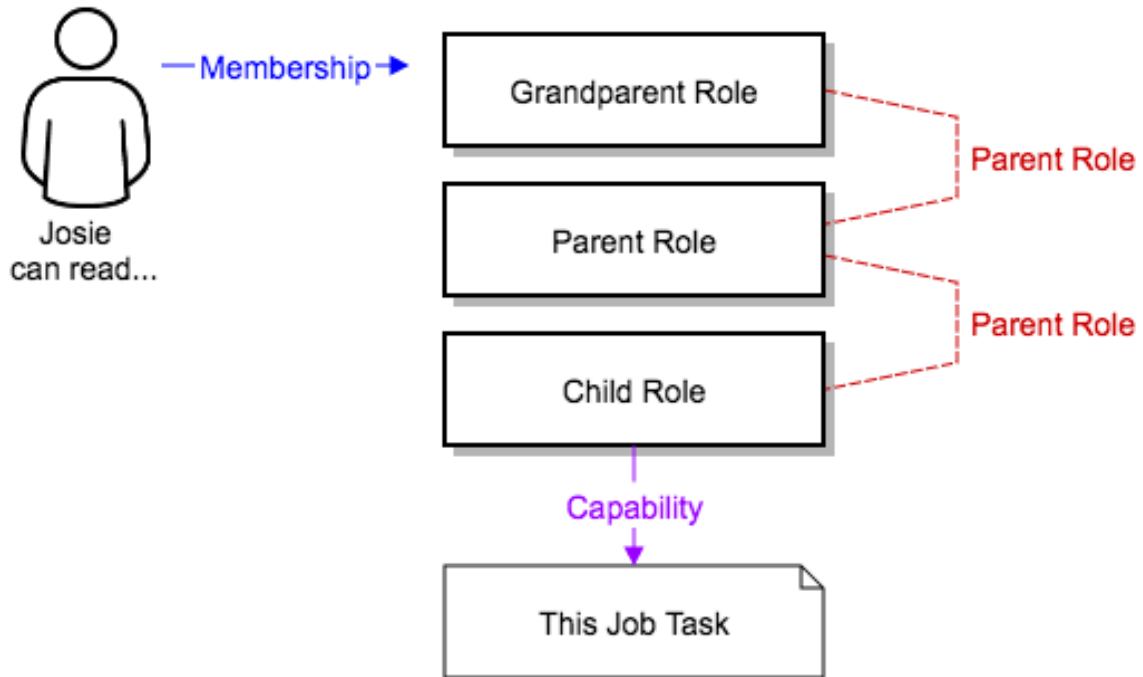


Often, you will have many Roles in a system and you will want some roles to include all of the capabilities of other roles. For example, you may want a System Administrator to have access to everything that an Organization Administrator has access to, who has everything that a Project Administrator has access to, and so on.

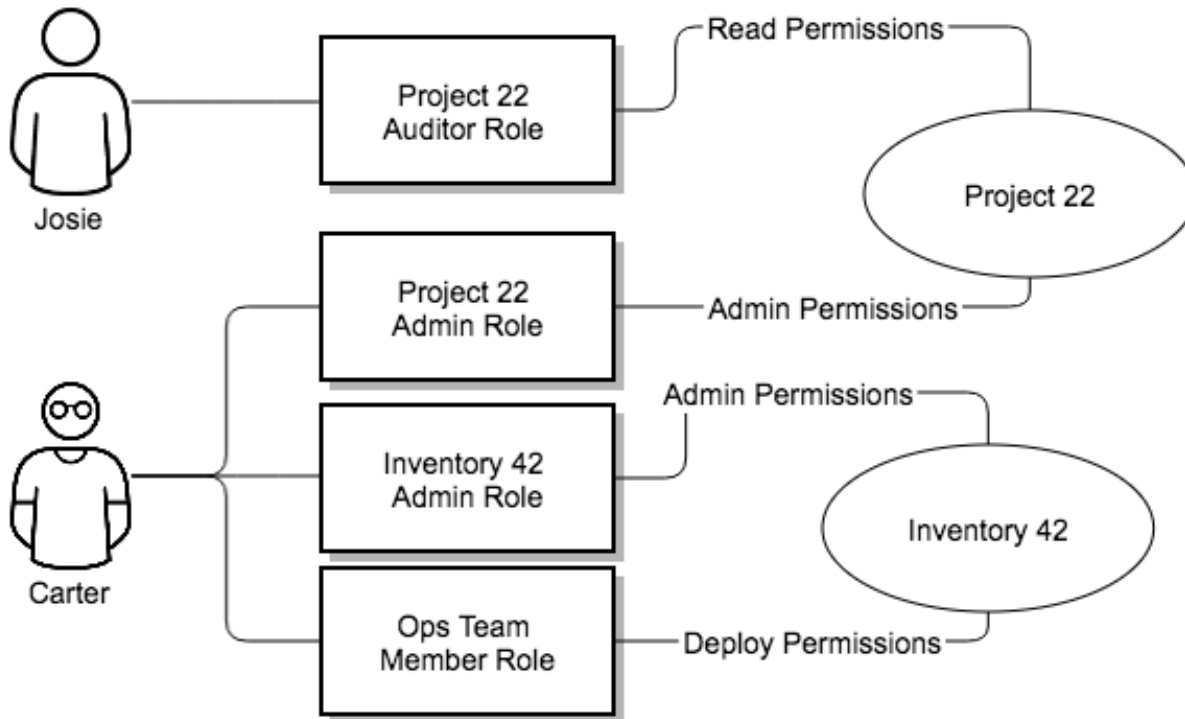
This concept is referred to as the ‘Role Hierarchy’:

- 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.

23.2.2 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.

23.2.3 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 support-ability 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.</p> <p>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>

23.2. Role-Based Access Controls

Member - Team "Execute"	Developers - Engineers	<p>This will be the most common and allows the organization member the</p>
----------------------------	---------------------------	----------------------------------------------------------------------------

23.3 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).

23.3.1 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

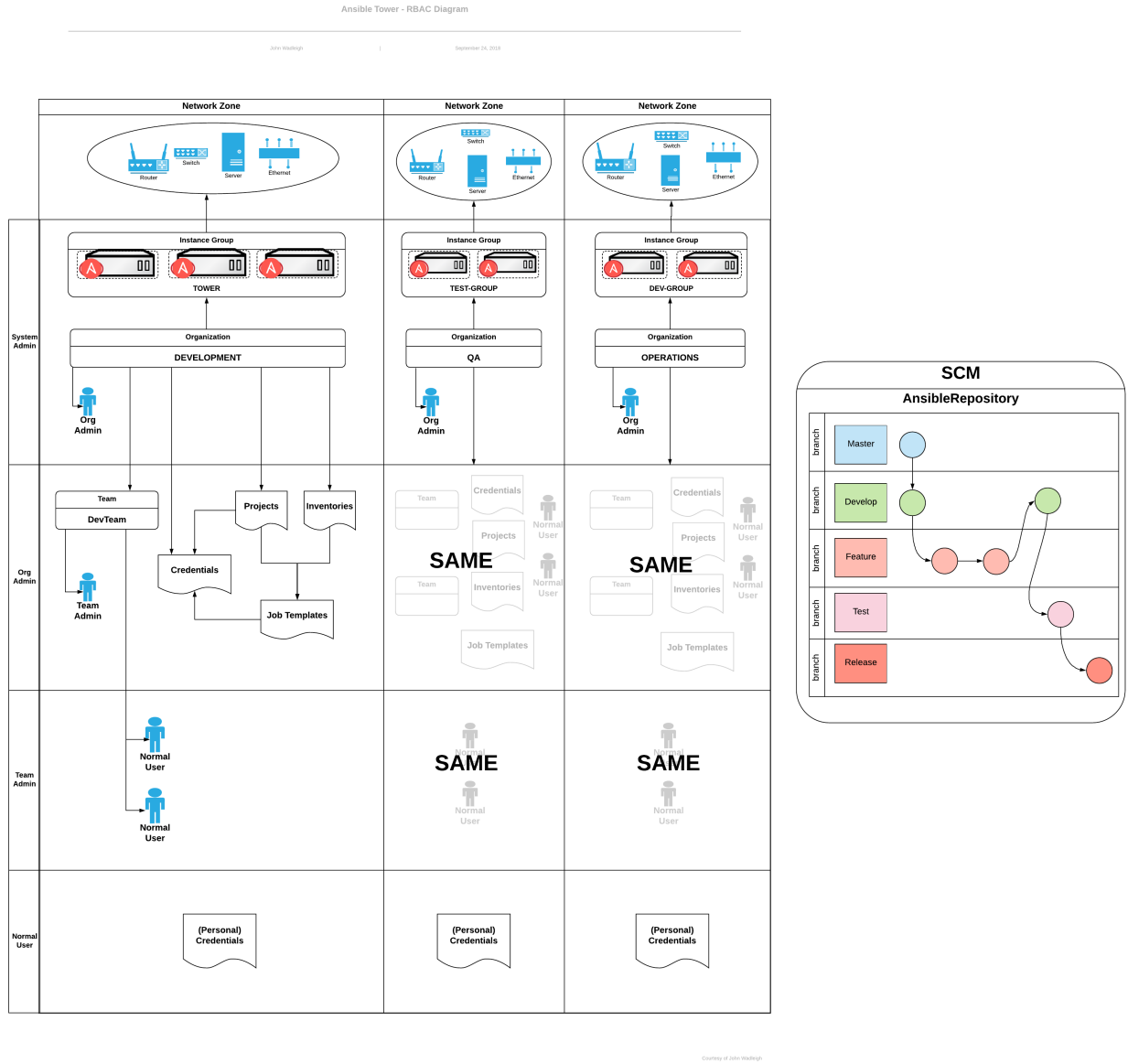
Workflow Admin

- 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:



INDEX

- genindex

COPYRIGHT © 2019 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.

Rackspace trademarks, service marks, logos and domain names are either common-law trademarks/service marks or registered trademarks/service marks of Rackspace US, Inc., or its subsidiaries, and are protected by trademark and other laws in the United States and other countries.

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

latl

credential types, 50

A

activity streams, 12

ad hoc commands, 108

inventories, 108

add new

inventories, 87

smart inventories, 87

adding new

applications, 68

credentials, 47

adding tokens

applications, 68

admin menu, 18

Amazon Web Services

credential types, 49

inventories, 101

Ansible Galaxy, 82

Ansible Galaxy integration

features, 3

Ansible Tower

inventories, 107

API considerations

credential types, 61

applications

adding new, 68

adding tokens, 68

authentication, 67

create, 68

getting started, 67

tokens, 67, 68

authentication, 67

applications, 67

automation

features, 2

autoscaling

best practices, 201

autoscaling flexibility

features, 3

AWS

cloud credentials, 136

B

backup and restore

features, 3

best practices, 200

autoscaling, 201

deployment, continuous, 201

dynamic inventory sources, 200

file and directory structure, 200

host counts, larger, 201

integration, continuous, 201

source control, 200

variable inventory management, 201

bubblewrap

functionality, 203

playbooks, 202

troubleshooting, 203

variables, 203

C

callbacks

extra variables, 139

capacity

jobs, 172

check

job types, 111

cloud credentials

AWS, 136

Google, 136

job templates, 135

MS Azure, 136

OpenStack, 135

Rackspace, 136

VMware, 136

cloud flexibility

features, 3

CloudForms

credential types, 56

components

licenses, 7

- configure Tower
 - settings menu, 19
- create
 - applications, 68
- create template
 - notifications, 177
- creating new
 - credential types, 63
- credential types, 48, 61
 - latl, 50
 - Amazon Web Services, 49
 - API considerations, 61
 - CloudForms, 56
 - creating new, 63
 - Google Compute Engine, 50
 - insights, 51
 - machine, 51
 - Microsoft Azure Resource Manager, 53
 - network, 54
 - OpenStack, 55
 - oVirt, 57
 - Red Hat Satellite, 57
 - Red Hat Virtualization, 57
 - rhv, 57
 - source control, 58
 - Vault, 59
 - VMware, 59
- credentials, 45
 - adding new, 47
 - getting started, 46, 62
 - how they work, 45
 - Insights, 189
 - types, 48
- custom
 - fact scan job, 132
- custom fact scans
 - playbook, 132
 - system tracking, 132
- custom script
 - inventories, 107

D

- dashboard, 14
 - host count, 14
 - job status, 14
 - jobs tab, 14
 - main menu, 12
 - schedule status, 14
- DEB files
 - licenses, 7
- deployment, continuous
 - best practices, 201
- dynamic inventory sources
 - best practices, 200

E

- Email
 - notifications types, 177
- evaluation, 6
- extra variables
 - callbacks, 139
 - provisioning callbacks, 139
 - surveys, 139, 157, 186
- extra_vars, 139, 157

F

- fact cache
 - features, 4
- fact caching
 - playbook, 133
- fact scan job
 - custom, 132
 - playbook, 130
- fact scan playbook
 - system tracking, 130
- facts
 - scan job templates, 133
- features, 5
 - Ansible Galaxy integration, 3
 - automation, 2
 - autoscaling flexibility, 3
 - backup and restore, 3
 - cloud flexibility, 3
 - fact cache, 4
 - inventory sources, Red Hat CloudForms, 4
 - inventory sources, Red Hat Satellite 6, 4
 - notifications, 4
 - OpenStack inventory support, 3
 - overview, 2
 - playbooks, Red Hat Insights, 4
 - real-time playbook, 2
 - remote command execution, 3
 - RESTful API, 3
 - role-based access control, 2
 - run-time job customization, 4
 - system tracking, 4
- file and directory structure
 - best practices, 200
- forks
 - jobs, 172
- functionality
 - bubblewrap, 203

G

- Galaxy support, 82
- getting started
 - applications, 67
 - credentials, 46, 62
- Google

- cloud credentials, 136
- Google Compute Engine
 - credential types, 50
 - inventories, 102
- groups
 - notifications, 176
- H**
- Hipchat
 - notifications types, 177
- host count
 - dashboard, 14
- host counts, larger
 - best practices, 201
- hostname configuration
 - notifications, 184
- how they work
 - credentials, 45
- I**
- Insights
 - credentials, 189
 - inventory, 192, 196
 - project, 191
 - projects, 189
- insights
 - credential types, 51
- installation bundle
 - licenses, 7
- instance groups, 159
- integration, continuous
 - best practices, 201
- inventories, 84
 - ad hoc commands, 108
 - add new, 87
 - Amazon Web Services, 101
 - Ansible Tower, 107
 - custom script, 107
 - Google Compute Engine, 102
 - groups, 92
 - add new, 92
 - Microsoft Azure Classic (deprecated), 103
 - Microsoft Azure Resource Manager, 103
 - OpenStack, 106
 - project-sourced, 100
 - Red Hat CloudForms, 106
 - Red Hat Satellite 6, 105
 - Red Hat Virtualization, 107
 - scan job templates, 130
 - smart, 86
 - VMware vCenter, 104
- inventory
 - Insights, 192, 196
- inventory sources
 - notifications, 176
 - inventory sources, Red Hat CloudForms
 - features, 4
 - inventory sources, Red Hat Satellite 6
 - features, 4
 - inventory sync
 - job results, 165
- IRC
 - notifications types, 177
- J**
- job results, 164
 - inventory sync, 165
- job status
 - dashboard, 14
- job templates, 111
 - cloud credentials, 135
 - job variables, 139
 - jobs, launching, 126
 - provisioning callbacks, 137
 - relaunch, 140
 - scheduling, 121, 122
 - survey creation, 124
 - survey extra variables, 139
 - survey optional questions, 126
 - surveys, 124
- job templates, hierarchy, 139
- job templates, overview, 139
- job types
 - check, 111
 - run, 111
 - scan, 111
- job variables
 - job templates, 139
 - workflow templates, 157
- jobs, 163
 - capacity, 172
 - event summary, 169
 - events summary, 169
 - forks, 172
 - host events, 171
 - host status bar, 169
 - host summary, 169
 - job summary, 169
 - notifications, 176
 - results, 164
 - views, 14
- jobs results
 - playbook run, 168
 - SCM, 166
- jobs tab
 - dashboard, 14
- jobs, launching
 - job templates, 126

workflow job templates, 155

L

license, 5, 6
 nodes, 7
 trial, 6
 troubleshooting, 10
 types, 6

license features, 5
 license, add manually, 10

license, import, 9
 license, viewing, 19

licenses
 components, 7
 DEB files, 7
 installation bundle, 7
 RPM files, 7

logging in, 8

M

machine
 credential types, 51

main menu
 dashboard, 12

Mattermost
 notifications types, 177

Microsoft Azure Classic (deprecated)
 inventories, 103

Microsoft Azure Resource Manager
 credential types, 53
 inventories, 103

MS Azure
 cloud credentials, 136

my view, 14

N

network
 credential types, 54

new schedule addition
 projects, 81

notifications
 create template, 177
 features, 4
 groups, 176
 hostname configuration, 184
 inventory sources, 176
 jobs, 176
 notifier, 176
 notifier hierarchy, 176
 notifier workflow, 176
 organizations, 29
 resetting the TOWER_URL_BASE, 184
 template, 177
 troubleshooting TOWER_URL_BASE, 184

types, 177
 types Email, 177
 types Hipchat, 177
 types IRC, 177
 types Mattermost, 177
 types pagerduty, 177
 types Rocket.Chat, 177
 types Slack, 177
 types Twilio, 177
 types Webhook, 177

notifier
 notifications, 176

notifier hierarchy
 notifications, 176

notifier workflow
 notifications, 176

O

OpenStack
 cloud credentials, 135
 credential types, 55
 inventories, 106

OpenStack inventory support
 features, 3

ordering
 sorting, 23

organization
 summary, 29

organizations, 25
 notifications, 29
 users, 27, 33

overview
 features, 2

oVirt
 credential types, 57

P

pagerduty
 notifications types, 177

permissions
 projects, 75
 teams, 42
 users, 34

playbook
 custom fact scans, 132
 fact caching, 133
 fact scan job, 130
 scan job, 130

playbook run
 jobs results, 168

playbooks
 bubblewrap, 202
 manage manually, 73
 process isolation, 202

- projects, 73, 74
 - PRoot settings, 202
 - sharing access, 202
 - sharing content, 202
 - source control, 74
- playbooks, Red Hat Insights
 - features, 4
- process isolation
 - playbooks, 202
- project
 - Insights, 191
 - Scan, 193
- project-sourced
 - inventories, 100
- projects, 71
 - add new, 72
 - Insights, 189
 - new schedule addition, 81
 - permissions, 75
 - playbooks, 73, 74
 - source control update, 75
- PRoot settings
 - playbooks, 202
- provisioning callbacks
 - extra variables, 139
 - job templates, 137

R

- Rackspace
 - cloud credentials, 136
- RBAC
 - security, 204
- real-time playbook
 - features, 2
- Red Hat CloudForms
 - inventories, 106
- Red Hat Satellite
 - credential types, 57
- Red Hat Satellite 6
 - inventories, 105
- Red Hat Virtualization
 - credential types, 57
 - inventories, 107
- relaunch
 - job templates, 140
- remote command execution
 - features, 3
- resetting the TOWER_URL_BASE
 - notifications, 184
- RESTful API
 - features, 3
- rhv
 - credential types, 57
- Rocket.Chat

- notifications types, 177
- role-based access control
 - features, 2
- role-based access controls, 204
- RPM files
 - licenses, 7
- run
 - job types, 111
- run-time job customization
 - features, 4

S

- Scan
 - project, 193
- scan
 - job types, 111
- scan job
 - playbook, 130
- scan job templates
 - facts, 133
 - inventories, 130
- schedule
 - views, 14
- schedule status
 - dashboard, 14
- scheduling
 - add new, 122, 146
 - job templates, 121, 122
 - workflow job template, 146
 - workflow job templates, 146
- SCM
 - jobs results, 166
- searching, 21
- security, 202
 - RBAC, 204
- settings menu
 - configure Tower, 19
 - view license, 19
- sharing access
 - playbooks, 202
- sharing content
 - playbooks, 202
- Slack
 - notifications types, 177
- smart
 - inventories, 86
- smart inventories
 - add new, 87
- sorting
 - ordering, 23
- source control
 - best practices, 200
 - credential types, 58
- source control update

- projects, 75
- summary
 - organization, 29
- support, 5, 6
- survey extra variables
 - job templates, 139
 - workflow templates, 157
 - workflows, 186
- surveys
 - creation, 124, 148
 - extra variables, 139, 157, 186
 - job templates, 124
 - optional questions, 126, 150
 - workflow job templates, 148
- system tracking
 - custom fact scans, 132
 - fact scan playbook, 130
 - features, 4
 - scan job, 111

T

- teams, 39
 - permissions, 42
 - users, 34, 40
- template
 - notifications, 177
- token authentication, 67
- tokens
 - applications, 67, 68
- Tower admin menu, 18
- Tower settings menu, 19
- trial, 6
- troubleshooting
 - bubblewrap, 203
 - license, 10
- troubleshooting TOWER_URL_BASE
 - notifications, 184
- Twilio
 - notifications types, 177
- types
 - Email, notifications, 177
 - Hipchat, notifications, 177
 - IRC, notifications, 177
 - Mattermost, notifications, 177
 - notifications, 177
 - pagerduty, notifications, 177
 - Rocket.Chat, notifications, 177
 - Slack, notifications, 177
 - Twilio, notifications, 177
 - Webhook, notifications, 177

U

- updates, 6
- users, 31

- organizations, 27, 33
- permissions, 34
- teams, 34, 40

V

- variable inventory management
 - best practices, 201
- variable precedence, 139, 157
- variables
 - bubblewrap, 203
- Vault
 - credential types, 59
- view license
 - settings menu, 19
- views
 - jobs, 14
 - schedule, 14
- visualizer
 - workflow, 150
- VMware
 - cloud credentials, 136
 - credential types, 59
- VMware vCenter
 - inventories, 104

W

- Webhook
 - notifications types, 177
- workflow
 - visualizer, 150
- workflow job template
 - scheduling, 146
- workflow job templates, 142
 - jobs, launching, 155
 - scheduling, 146
 - survey creation, 148
 - survey optional questions, 150
 - surveys, 148
- workflow templates
 - job variables, 157
 - survey extra variables, 157
 - workflow visualizer, 150
- workflow templates, hierarchy, 157
- workflow templates, overview, 157
- workflow visualizer
 - workflow templates, 150
- workflows, 185
 - survey extra variables, 186