

---

# **Ansible Tower User Guide**

*Release Ansible Tower 3.1.0*

**Red Hat, Inc.**

**Jul 12, 2017**

# CONTENTS

<b>1</b>	<b>Overview</b>	<b>2</b>
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
<b>2</b>	<b>Tower Licensing, Updates, and Support</b>	<b>5</b>
2.1	Support . . . . .	5
2.2	Trial / Evaluation . . . . .	6
2.3	Subscription Types . . . . .	6
2.4	Node Counting in Licenses . . . . .	6
2.5	License Features . . . . .	7
2.6	Tower Component Licenses . . . . .	7
<b>3</b>	<b>Logging In</b>	<b>8</b>
<b>4</b>	<b>Import a License</b>	<b>9</b>
4.1	Adding a Tower License Manually . . . . .	10
<b>5</b>	<b>The Tower Dashboard and Interface</b>	<b>12</b>
5.1	Tower Admin Menu . . . . .	13
5.2	Settings Menu . . . . .	14
5.3	My View – User Menu . . . . .	15
5.4	Dashboard Views . . . . .	16
5.5	Activity Streams . . . . .	17
<b>6</b>	<b>Search</b>	<b>19</b>
6.1	Sort . . . . .	20
<b>7</b>	<b>Organizations</b>	<b>22</b>
7.1	Organizations - Users . . . . .	24
7.2	Organizations - Notifications . . . . .	26

7.3	Organization - Summary . . . . .	26
<b>8</b>	<b>Users</b>	<b>27</b>
8.1	Create a User . . . . .	28
8.2	User Types - Quick View . . . . .	30
8.3	Users - Organizations . . . . .	30
8.4	Users - Teams . . . . .	31
8.5	Users - Permissions . . . . .	31
<b>9</b>	<b>Teams</b>	<b>35</b>
9.1	Create a Team . . . . .	35
<b>10</b>	<b>Credentials</b>	<b>43</b>
10.1	Understanding How Credentials Work . . . . .	43
10.2	Getting Started with Credentials . . . . .	44
10.3	Add a New Credential . . . . .	45
10.4	Credential Types . . . . .	45
<b>11</b>	<b>Projects</b>	<b>57</b>
11.1	Add a new project . . . . .	58
11.2	Updating projects from source control . . . . .	60
11.3	Add a new schedule . . . . .	62
11.4	Ansible Galaxy Support . . . . .	63
<b>12</b>	<b>Inventories</b>	<b>64</b>
12.1	Add a new inventory . . . . .	64
12.2	Groups and Hosts . . . . .	66
12.3	Running Ad Hoc Commands . . . . .	85
12.4	System Tracking . . . . .	87
<b>13</b>	<b>Job Templates</b>	<b>94</b>
13.1	Utilizing Cloud Credentials . . . . .	100
13.2	Surveys . . . . .	103
13.3	Scan Job Templates . . . . .	107
13.4	Provisioning Callbacks . . . . .	117
13.5	Launching Jobs . . . . .	119
13.6	Scheduling . . . . .	120
<b>14</b>	<b>Jobs</b>	<b>124</b>
14.1	Job Details - Inventory Sync . . . . .	125
14.2	Job Details - SCM . . . . .	127
14.3	Job Details - Playbook Run . . . . .	128
14.4	Job Concurrency . . . . .	132
<b>15</b>	<b>Notifications</b>	<b>135</b>
15.1	Notifier Hierarchy . . . . .	135
15.2	Workflow . . . . .	135
15.3	Create a Notification Template . . . . .	136
15.4	Notification Types . . . . .	136
15.5	Configuring the <code>towerhost</code> hostname . . . . .	141
<b>16</b>	<b>Workflows</b>	<b>143</b>
16.1	Extra Variables . . . . .	144
16.2	Role-Based Access Controls . . . . .	145

<b>17 Best Practices</b>	<b>146</b>
17.1 Use Source Control . . . . .	146
17.2 Ansible file and directory structure . . . . .	146
17.3 Use Dynamic Inventory Sources . . . . .	146
17.4 Variable Management for Inventory . . . . .	147
17.5 Autoscaling . . . . .	147
17.6 Larger Host Counts . . . . .	147
17.7 Continuous integration / Continuous Deployment . . . . .	147
<b>18 Security</b>	<b>148</b>
18.1 Playbook Access and Information Sharing . . . . .	148
18.2 PRoot functionality and variables . . . . .	149
18.3 Role-Based Access Controls . . . . .	149
<b>19 Index</b>	<b>155</b>
<b>20 Copyright © 2016 Red Hat, Inc.</b>	<b>156</b>
<b>Index</b>	<b>157</b>



Thank you for your interest in Ansible Tower by Red Hat. 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 3.1.0.

Ansible Tower Version 3.1.0; Feb 28, 2017; <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

Introduced in version 2.2.0, Tower's System Tracking brings a new level of visibility to your infrastructure—you can see exactly what is happening on your systems, comparing it to both the prior state of the system and to other systems in your cluster, which helps you to ensure compliance. The rich and extensible store of data available in System Tracking is accessible via Tower's REST API, enabling you to feed it into other tools and systems.

## 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

## TOWER LICENSING, UPDATES, AND SUPPORT

Ansible Tower by Red Hat (“**Ansible Tower**”) is a proprietary software product provided via 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 for paid **Enterprise: Standard** and **Enterprise: Premium** Subscription customers seeking help with the Ansible Tower product.

If you or your company has paid for Ansible Tower, 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*.

If you are experiencing Ansible software issues, you should reach out to the “ansible-devel” mailing list or file an issue on the Github project page at <https://github.com/ansible/ansible/issues/>.

All of Ansible’s community and OSS info can be found here: <https://docs.ansible.com/ansible/community.html>

#### 2.1.1 Ansible Playbook Support

For customers with a paid Enterprise: Standard or Enterprise: Premium Ansible Tower Subscription, Red Hat offers Ansible Playbook support<sup>1</sup>. Playbook support consists of support for:

- Runtime execution problems for Playbooks run via Tower
- Assistance with Playbook errors and tracebacks
- Limited best practice guidance in Ansible use from the Ansible Experts

Playbook support does not consist of:

- Enhancements and fixes for Ansible modules and the Ansible engine
- Assistance with the creation of Playbooks from anew
- Long-term maintenance of a specific Ansible or Ansible Tower version

---

<sup>1</sup> Playbook support is available for customers using the current or previous minor release of Ansible. For example, if the current version of Ansible is 2.2, Red Hat provides Ansible Playbook support for versions 2.2 and 2.1. In the event an Ansible Playbook workaround is not available, and an Ansible software correction is required, a version update will be required.

**Notes:**

## 2.2 Trial / Evaluation

While a license is required for Ansible Tower to run, there is no fee for managing up to 10 hosts. Additionally, trial licenses are available for exploring Ansible Tower with a larger number of hosts.

- Trial licenses for Ansible Tower are available at: <http://ansible.com/license>
- To acquire a license for additional Managed Nodes, visit: <http://www.ansible.com/pricing/>
- Ansible Playbook Support is not included in a trial license or during an evaluation of the Tower Software.

## 2.3 Subscription Types

Ansible Tower is provided at various levels of support and number of machines as an annual Subscription.

- **Self-Support**
  - Manage smaller environments (up to 250 Managed Nodes)
  - Maintenance and upgrades included
  - No support or SLA included
- **Enterprise: Standard (F.K.A. “Enterprise”)**
  - 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: Premium (F.K.A. “Premium Enterprise”)**
  - 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 hosts 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.

## 2.5 License Features

The following list of features are available for all new Enterprise: Standard or Enterprise: Premium Subscriptions:

- Workflows (*added in latl 3.1.0*)
- Clustering in Tower (*added in latl 3.1.0*)
- Custom re-branding for login (*added in Ansible Tower 2.4.0*)
- SAML and RADIUS Authentication Support (*added in Ansible Tower 2.4.0*)
- Multi-Organization Support
- Activity Streams
- Surveys
- LDAP Support
- Active/Passive Redundancy
- System Tracking (*added in Ansible Tower 2.2.0*)

Enterprise: Standard or Enterprise: Premium license users with versions of Ansible Tower prior to 2.2 must import a new license file to enable System Tracking.

## 2.6 Tower Component Licenses

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

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

## LOGGING IN


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



The screenshot shows the 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.

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.

---

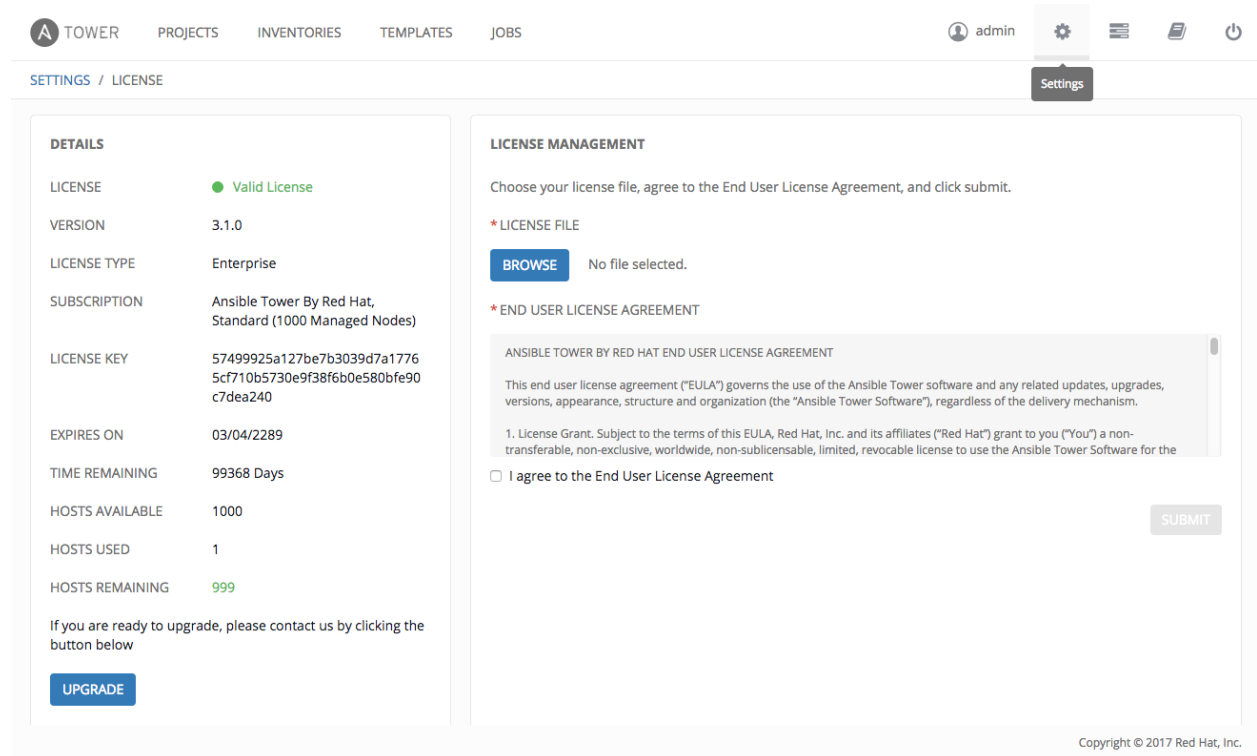
The screenshot shows the 'TOWER LICENSE' page in the Ansible Tower web interface. At the top left is the 'A TOWER' logo and at the top right is a power icon. The main content area is a white box with a light gray border. It has the title 'TOWER LICENSE' and a welcome message: 'Welcome to Ansible Tower! Please complete the steps below to acquire a license.' Below this are two numbered steps: 1. 'Please click the button below to visit Ansible's website to get a Tower license key.' with a blue 'REQUEST LICENSE' button. 2. 'Choose your license file, agree to the End User License Agreement, and click submit.' This step includes a '\* LICENSE FILE' section with a blue 'BROWSE' button and the text 'No file selected.' Below that is a '\* END USER LICENSE AGREEMENT' section with a scrollable text area containing the agreement text and a 'SUBMIT' button. At the bottom right of the page is the copyright notice 'Copyright © 2017 Red Hat, Inc.'

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 Settings (  ) Menu's 'VIEW YOUR LICENSE' link.



**DETAILS**

LICENSE ● Valid License

VERSION 3.1.0

LICENSE TYPE Enterprise

SUBSCRIPTION Ansible Tower By Red Hat, Standard (1000 Managed Nodes)

LICENSE KEY 57499925a127be7b3039d7a17765cf710b5730e9f38f6b0e580bfe90c7dea240

EXPIRES ON 03/04/2289

TIME REMAINING 99368 Days

HOSTS AVAILABLE 1000

HOSTS USED 1

HOSTS REMAINING 999

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

**UPGRADE**

**LICENSE MANAGEMENT**

Choose your license file, agree to the End User License Agreement, and click submit.

\* LICENSE FILE

**BROWSE** No file selected.

\* END USER LICENSE AGREEMENT

ANSIBLE TOWER BY RED HAT END USER LICENSE AGREEMENT

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

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

I agree to the End User License Agreement

**SUBMIT**

Copyright © 2017 Red Hat, Inc.

## 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/v1/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@hotdog.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 DASHBOARD AND INTERFACE

---

**Note:** Ansible Tower 3.0 provides a streamlined interface, with the Settings (  ) button offering 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.

---

The Tower Dashboard offers a friendly graphical framework for your IT orchestration needs. Across the top-left side of the Tower Dashboard, administrators can quickly navigate to their **Projects, Inventories, Job Templates, and Jobs.**

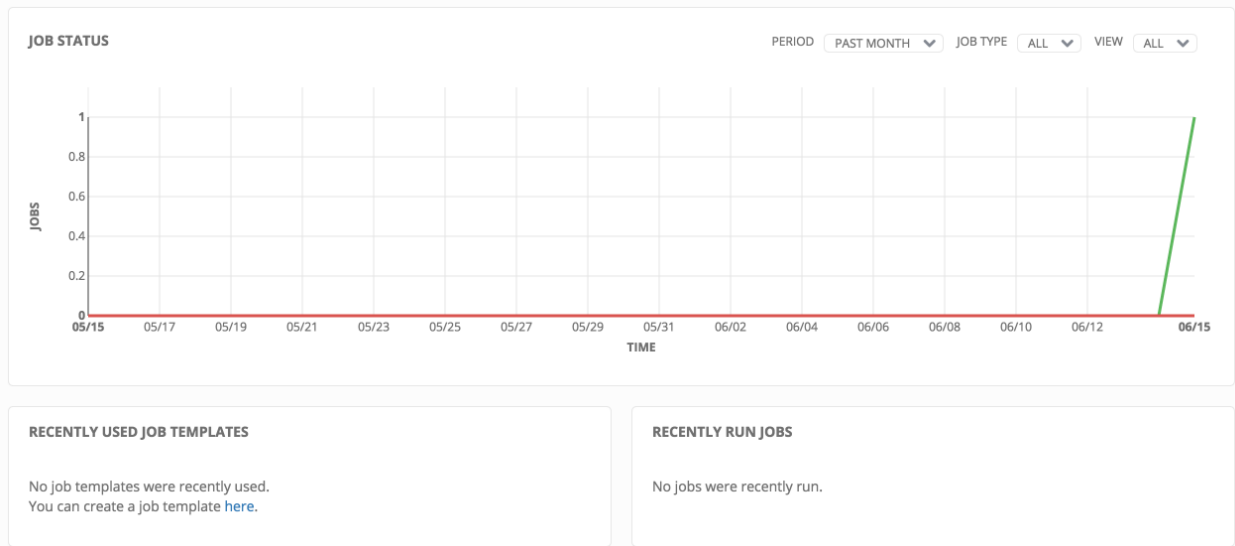
Across the top-right side of this interface, administrators can access the tools they need to configure organizations, users, groups, and permissions as well as view related documentation, access portal mode, and log out.



At the top of the Dashboard is a summary of your hosts, inventories, and projects. Each of these is linked to the corresponding object 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 Job Templates** and **Recently Run Jobs.**




**Note:** Clicking on the Ansible Tower logo at any time returns you to the Dashboard.

## 5.1 Tower Admin Menu

The Tower user menu is accessed by clicking  **admin**.

From here, you can:

- View/Edit the properties of the Tower admin account
- View the activity stream for that user (by clicking on the Activity Stream  button)
- View the **Organizations** which have been setup for the Tower user
- View the **Teams** to which the Tower user has been added
- View the **Permissions** for this Tower admin account


The screenshot shows the 'ADMIN' user configuration page in Ansible Tower. The breadcrumb trail is 'SETTINGS / USERS / ADMIN'. The user 'admin' is selected. The form has tabs for 'DETAILS', 'ORGANIZATIONS', 'TEAMS', and 'PERMISSIONS'. The 'DETAILS' tab is active. Fields include:
 

- \* FIRST NAME:
- \* LAST NAME:
- \* EMAIL:
- \* USERNAME:
- PASSWORD:
- CONFIRM PASSWORD:
- USER TYPE:

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

## 5.2 Settings Menu




To enter the Settings Menu screen for Ansible Tower, click the  button. This screen allows you to create your organizations, add credentials, add users and teams, schedule management jobs, modify your Tower’s configuration, and more. You can also view your license from the Settings Menu’s ‘View Your License’ link.

The screenshot shows the 'SETTINGS' menu in Ansible Tower. The breadcrumb trail is 'SETTINGS'. A 'Settings' dropdown menu is open, showing the gear icon. The menu contains the following items:
 


- ORGANIZATIONS**: Group all of your content to manage permissions across departments in your company.
- USERS**: Allow others to sign into Tower and own the content they create.
- TEAMS**: Split up your organization to associate content and control permissions for groups.
- CREDENTIALS**: Add passwords, SSH keys, etc. for Tower to use when launching jobs against machines, or when syncing inventories or projects.
- MANAGEMENT JOBS**: Manage the cleanup of old job history, activity streams, data marked for deletion, and system tracking info.
- INVENTORY SCRIPTS**: Create and edit scripts to dynamically load hosts from any source.
- NOTIFICATIONS**: Create templates for sending notifications with Email, HipChat, Slack, and SMS.
- VIEW YOUR LICENSE**: View and edit your license information.
- CONFIGURE TOWER**: Edit Tower's configuration.
- ABOUT TOWER**: View information about this version of Ansible Tower.

## 5.3 My View – User Menu

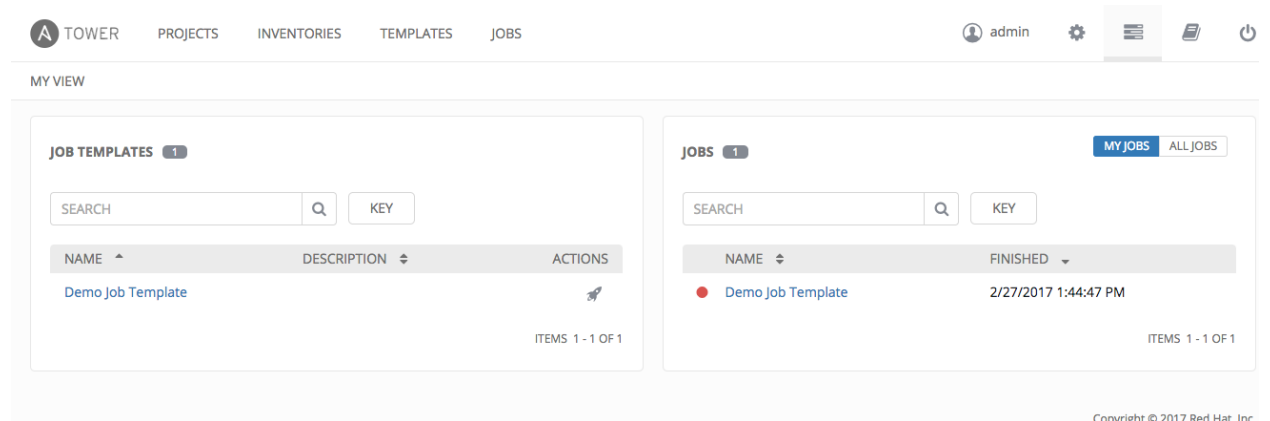
My View, a user’s single-page view of jobs and job templates, can be accessed by clicking the My View () button.

My View is a simplified interface for users that 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.

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




My View can be accessed in two ways:

- via the My View () button at the top-right of the Tower interface
- by navigating to `https://<Tower server name>/portal`

My View displays two main sections—*Job Templates* and *Jobs*.

### 5.3.1 Job Templates

This shows the job templates that are available for the user to run. This list can be searched by **Name** or **Description**,

and can be sorted by those keys as well. To launch a job template, click the  button. This launches the job, which can be viewed in **My Jobs**.

**Note:** Unlike Tower’s main interface, you are not automatically redirected to the Job view for the launched job. This view is still accessible via the **View Details** button for this job run in the **My Jobs** panel. This is useful for instances when a job fails and a non-technical user needs an Ansible expert look at what might have gone wrong.

### 5.3.2 Jobs

This shows the list of jobs that have run in the past.

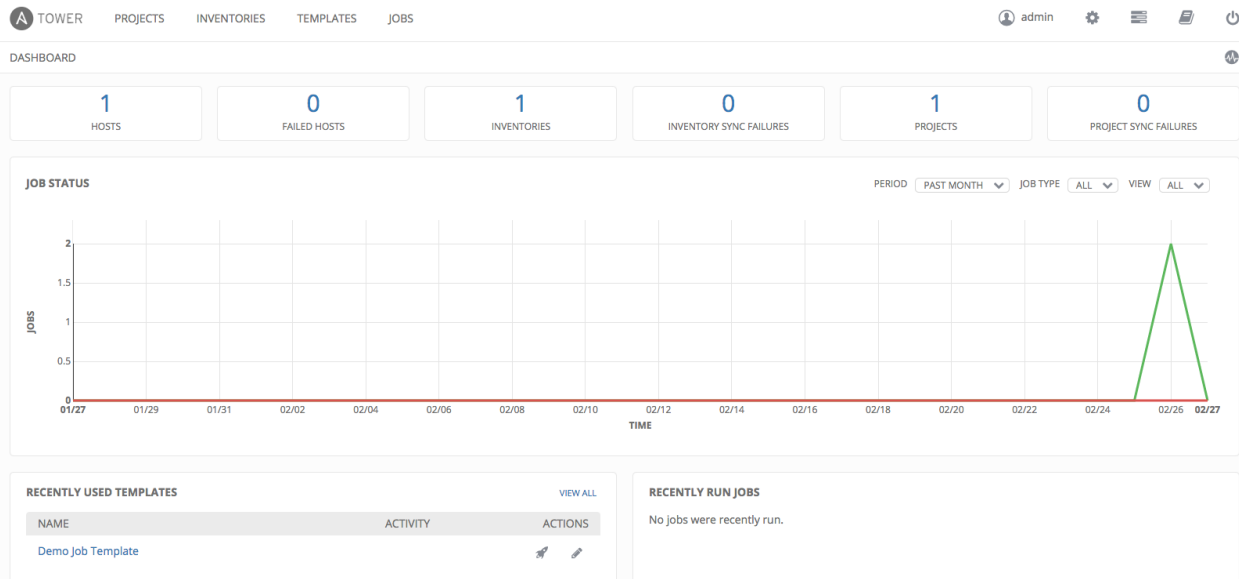
Sort for jobs specific to you by clicking on the **My Jobs** button or review all jobs you have access to view by clicking on the **All Jobs** button, next to the search bar.

- **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 the Job **ID**, the **Status** of the job (*Running, Pending, Successful, or Failed*), its start time, and the job **Name**. The job list can be sorted by any of these fields. Clicking on the *Details* button opens a new window with the **Job Details** for that job (refer to *Jobs* for more information).

## 5.4 Dashboard Views

The central interface to Tower is the Dashboard. You will use the dashboard to quickly view job statuses, recently run jobs, and recently used job templates.



Copyright © 2017 Red Hat, Inc.

### 5.4.1 Job Status

The Job Status graph displays the number of successful and failed jobs over a specified time period. You can choose to limit the job types that are viewed, and to change the time horizon of the graph.

### 5.4.2 Recently Used Job Templates

The Jobs section of this display shows a summary of the most recently used jobs. You can also access this summary by clicking on the **Jobs** entry in the main navigation menu.




### 5.4.3 Recently Run Jobs

The Recently Run Jobs section displays which jobs were most recently run, their status, and notes when they were run as well.

## 5.5 Activity Streams



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


ACTIVITY STREAM | ALL ACTIVITY

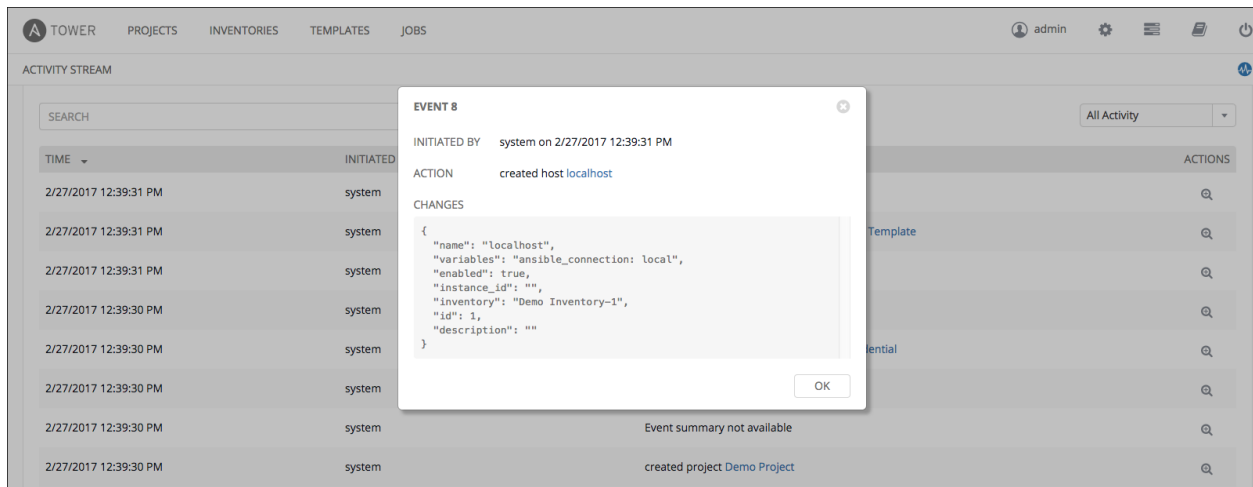
SEARCH   KEY  All Activity

TIME	INITIATED BY	EVENT	ACTIONS
6/29/2016 10:20:41 AM	admin	created job Demo Job Template	<input type="button" value="Q"/>
6/29/2016 9:12:08 AM	admin	updated team Production Operations	<input type="button" value="Q"/>
6/29/2016 9:11:15 AM	admin	associated Production Operations member_role to jdoge	<input type="button" value="Q"/>
6/29/2016 9:11:15 AM	admin	associated Production Operations admin_role to gdoge	<input type="button" value="Q"/>
6/29/2016 9:10:54 AM	admin	created team Production Operations	<input type="button" value="Q"/>
6/29/2016 9:10:12 AM	admin	associated cdoge member_role to Honey Dog, Inc.	<input type="button" value="Q"/>
6/29/2016 9:10:12 AM	admin	updated user cdoge	<input type="button" value="Q"/>
6/29/2016 9:10:12 AM	admin	created user cdoge	<input type="button" value="Q"/>
6/29/2016 9:09:39 AM	admin	associated jdoge member_role to Honey Dog, Inc.	<input type="button" value="Q"/>
6/29/2016 9:09:39 AM	admin	updated user jdoge	<input type="button" value="Q"/>
6/29/2016 9:09:39 AM	admin	created user jdoge	<input type="button" value="Q"/>
6/29/2016 9:09:12 AM	admin	associated gdoge member_role to Honey Dog, Inc.	<input type="button" value="Q"/>
6/29/2016 9:09:12 AM	admin	updated user gdoge	<input type="button" value="Q"/>
6/29/2016 9:09:12 AM	admin	created user gdoge	<input type="button" value="Q"/>
6/29/2016 9:08:40 AM	admin	associated admin member_role to Honey Dog, Inc.	<input type="button" value="Q"/>
6/29/2016 9:08:25 AM	admin	created organization Honey Dog, Inc.	<input type="button" value="Q"/>
6/29/2016 9:08:25 AM	admin	associated system_auditor to Honey Dog, Inc.	<input type="button" value="Q"/>
6/29/2016 9:08:25 AM	admin	associated system_administrator to Honey Dog, Inc.	<input type="button" value="Q"/>
6/29/2016 9:08:02 AM	admin	Event summary not available	<input type="button" value="Q"/>
6/29/2016 9:08:02 AM	admin	Event summary not available	<input type="button" value="Q"/>

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.



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.

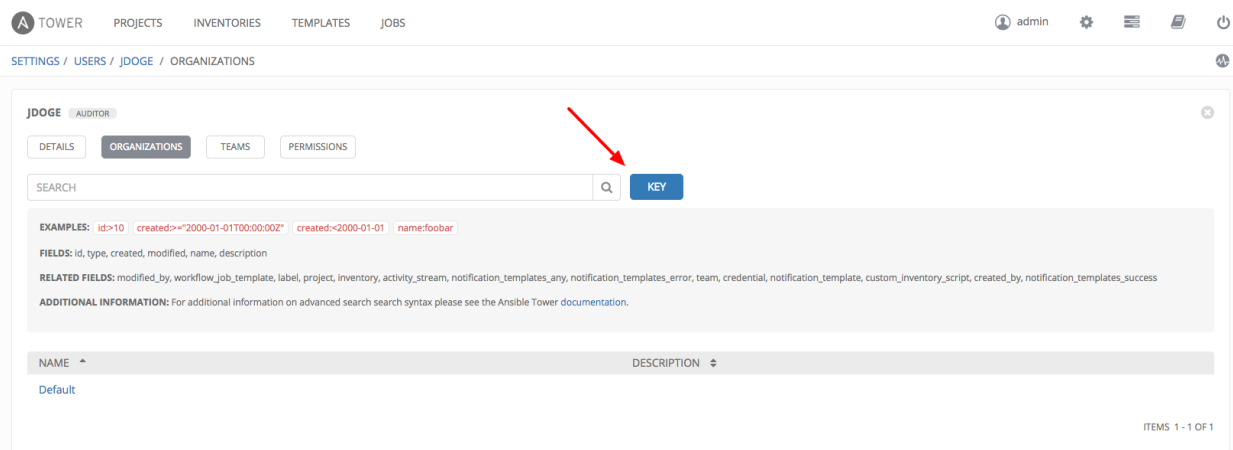
The Activity Stream on the main Dashboard shows the Activity Stream for the entire Tower instance. Most pages in Tower allow viewing an activity stream filtered for that specific object.

## SEARCH

Ansible Tower release 3.1 introduces 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.

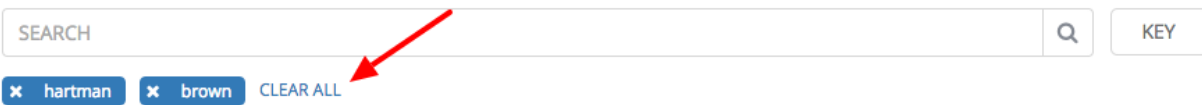


The screenshot shows the Ansible Tower interface. At the top, there is a navigation bar with 'TOWER', 'PROJECTS', 'INVENTORIES', 'TEMPLATES', and 'JOBS'. On the right, there is a user profile 'admin' and several utility icons. Below the navigation bar, there is a breadcrumb trail: 'SETTINGS / USERS / JDOGE / ORGANIZATIONS'. The main content area is titled 'JDOGE' and has a sub-tab 'AUDITOR'. There are three tabs: 'DETAILS', 'ORGANIZATIONS' (which is selected), and 'PERMISSIONS'. Below the tabs is a search bar with the word 'SEARCH' and a magnifying glass icon. To the right of the search bar is a blue button labeled 'KEY'. A red arrow points to this 'KEY' button. Below the search bar, there is a section titled 'EXAMPLES' with the text: 'id>10 | created>="2000-01-01T00:00:00Z" | created:<2000-01-01 | name:foobar'. Below this, there is a section titled 'FIELDS' with the text: 'id, type, created, modified, name, description'. Below that, there is a section titled 'RELATED FIELDS' with the text: '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'. Below that, there is a section titled 'ADDITIONAL INFORMATION' with the text: 'For additional information on advanced search syntax please see the Ansible Tower documentation.' At the bottom of the search bar area, there is a table with two columns: 'NAME' and 'DESCRIPTION'. The 'NAME' column has a dropdown arrow and the text 'Default'. The 'DESCRIPTION' column has a dropdown arrow. At the bottom right of the search bar area, there is a small text 'ITEMS 1 - 1 OF 1'.

A few rules to note about searching and filtering, and what the syntax equivalent is for each:

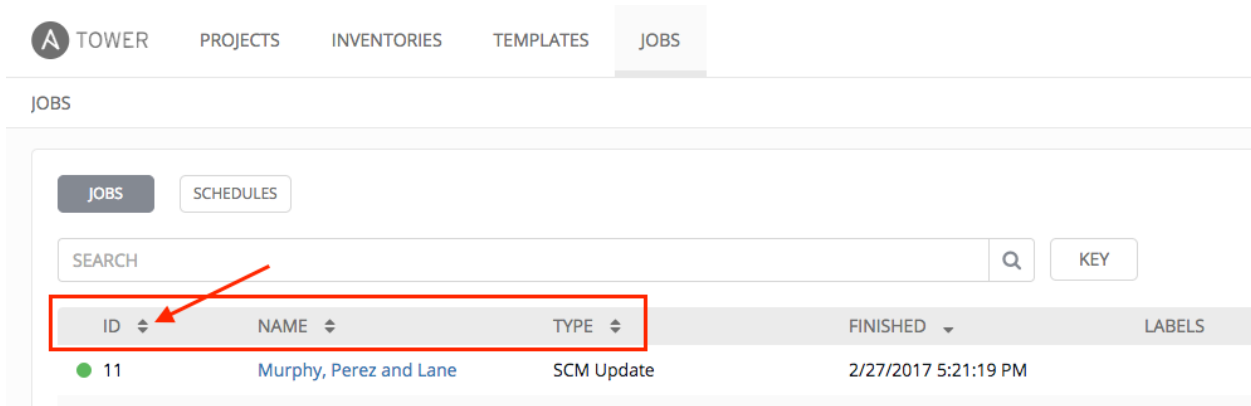
Rule	Syntax Equivalent
Searches are not case-sensitive. Example: FooBar	name:icontains:foobar
Entering key terms in the search will find all instances of that term across all fields. Example 1: mySearchTerm Example 2: In a Projects List, organization:foo searches for name AND description.	name:icontains:mySearchTerm
Specific name searches will search against the API name. Example: Management job in the user interface is system_job in the API.	
Placing a search string inside quotes will search for all instances that match exactly to that search string. Example: "foo bar baz"	AND foo AND bar AND baz
Searching terms without quotes will search for any instances that match any of the terms. Example: foo bar baz	OR foo OR bar OR baz

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



## 6.1 Sort

Use the arrows in each column to sort by ascending or descending order.



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

JOBS

JOBS SCHEDULES

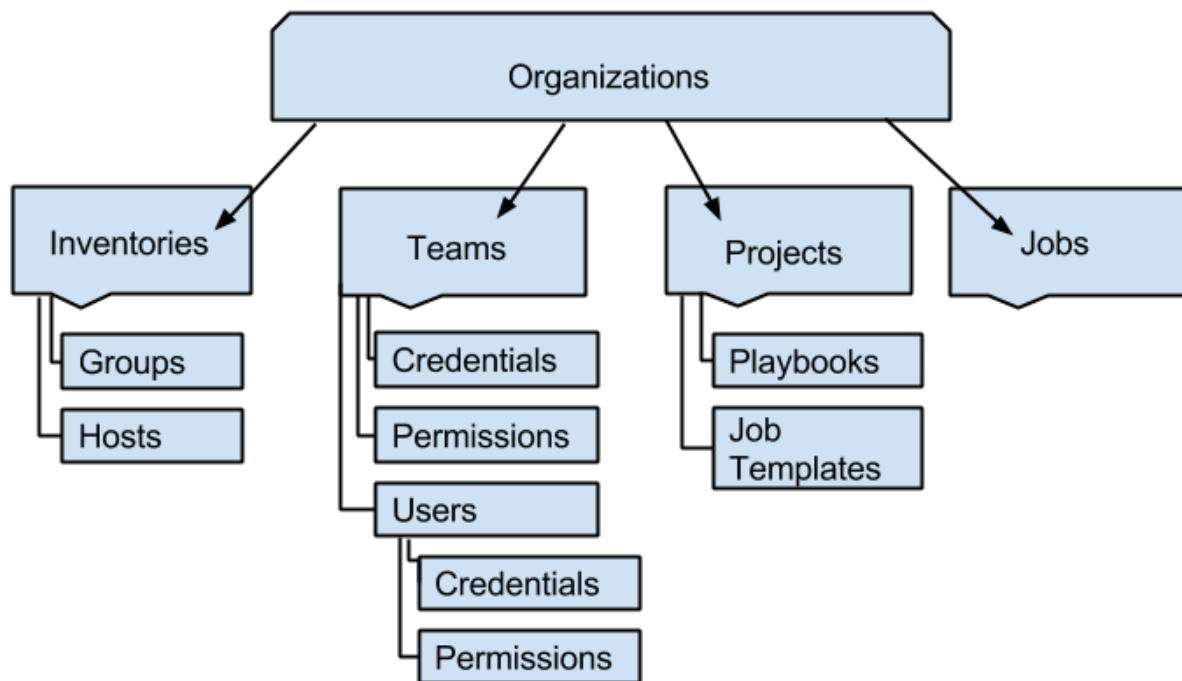
SEARCH [Q] KEY


ID	NAME	TYPE	FINISHED	LABELS	ACTIONS
3	Demo Project	SCM Update	2/27/2017 1:43:30 PM		[Search] [Refresh] [Delete]
5	Harmon Group	SCM Update	2/27/2017 5:21:11 PM		[Search] [Refresh] [Delete]
8	Martinez-Duarte	SCM Update	2/27/2017 5:21:16 PM		[Search] [Refresh] [Delete]
10	Moore Group	SCM Update	2/27/2017 5:21:18 PM		[Search] [Refresh] [Delete]
11	Murphy, Perez and Lane	SCM Update	2/27/2017 5:21:19 PM		[Search] [Refresh] [Delete]
6	Sanchez-Rodriguez	SCM Update	2/27/2017 5:21:12 PM		[Search] [Refresh] [Delete]
9	Ward, Hale and Peterson	SCM Update	2/27/2017 5:21:17 PM		[Search] [Refresh] [Delete]
7	Warren-johnson	SCM Update	2/27/2017 5:21:13 PM		[Search] [Refresh] [Delete]

ITEMS 1 - 8 OF 8

## ORGANIZATIONS

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

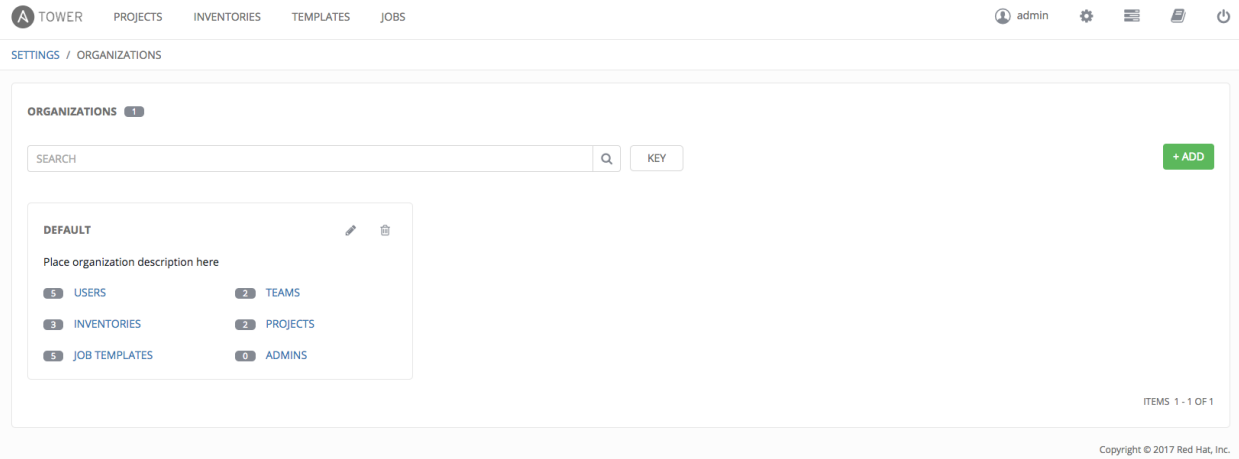



The **Organizations** link from the Settings (  ) menu displays all of the existing organizations for your installation of Tower. Organizations can be searched by **Name** or **Description**. Modify and remove organizations using the **Edit** and **Delete** buttons.

---

**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.

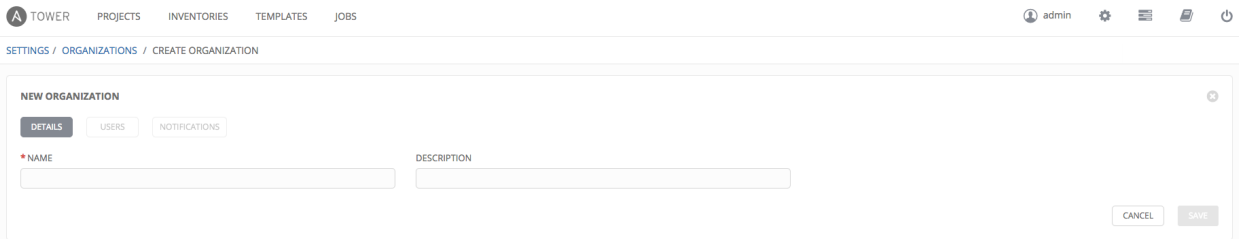
---



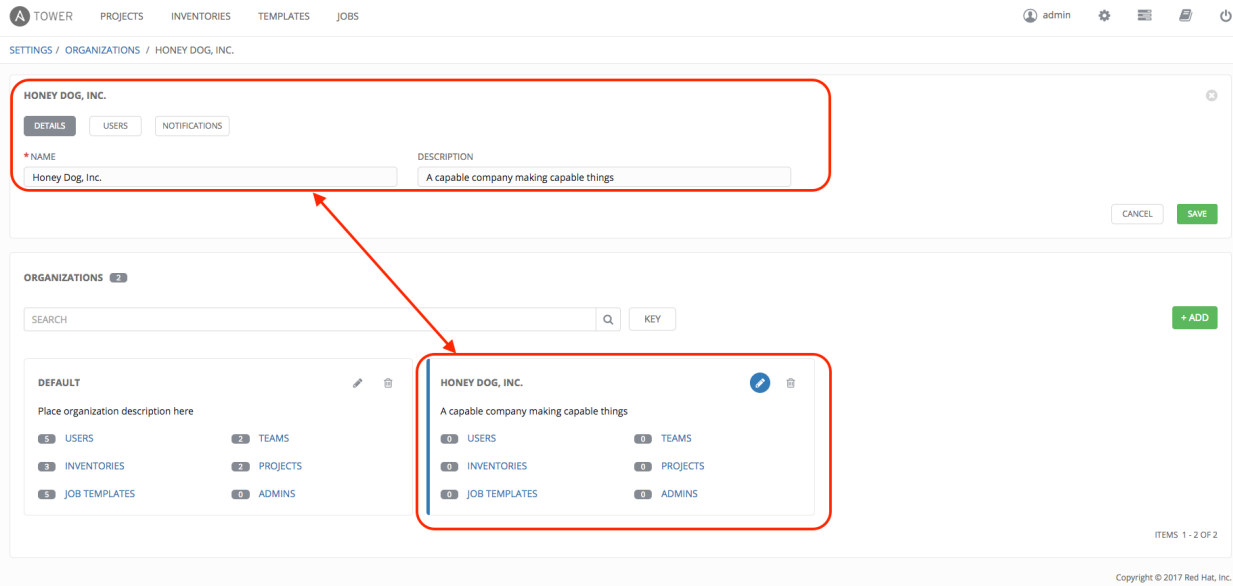
Enterprise: Standard and Enterprise: Premium Tower licenses allow you to create a new Organization by selecting the  button.

**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 Enterprise: Standard or Enterprise: Premium Tower licenses have the ability to add new Organizations beyond the default.

1. Enter the **Name** for your Organization.
2. Optionally, enter a **Description** for the Organization.
3. Click **Save** to finish creating the Organization.

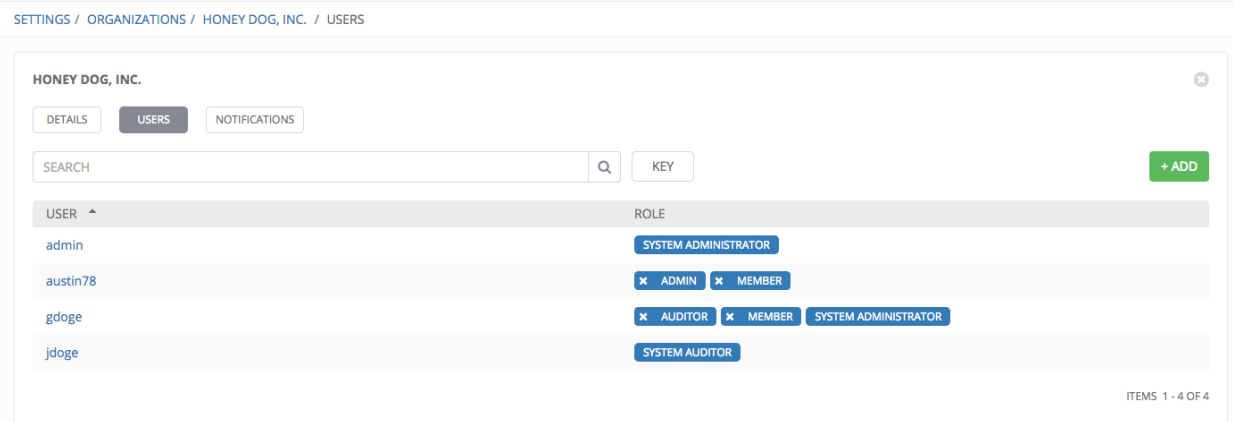



Once created, Tower displays the Organization details, and allows for the managing of users and administrators for the Organization.



## 7.1 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 via the **Users** link available from the Settings  menu.) The user list may be sorted by username and role. Use the Tower Search to search for users by various attributes. Refer to the [Search](#) chapter for more information.

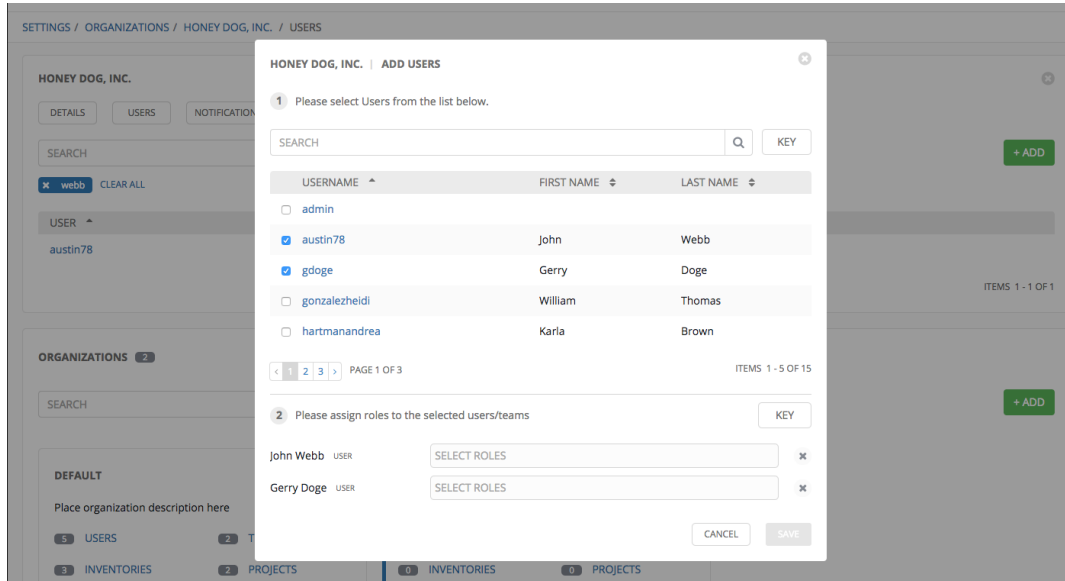
Clicking on a user brings up that user's details, allowing you to review, grant, edit, and remove associated permissions for that user. For more information, refer to [Users](#).

### 7.1.1 Add a User

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



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

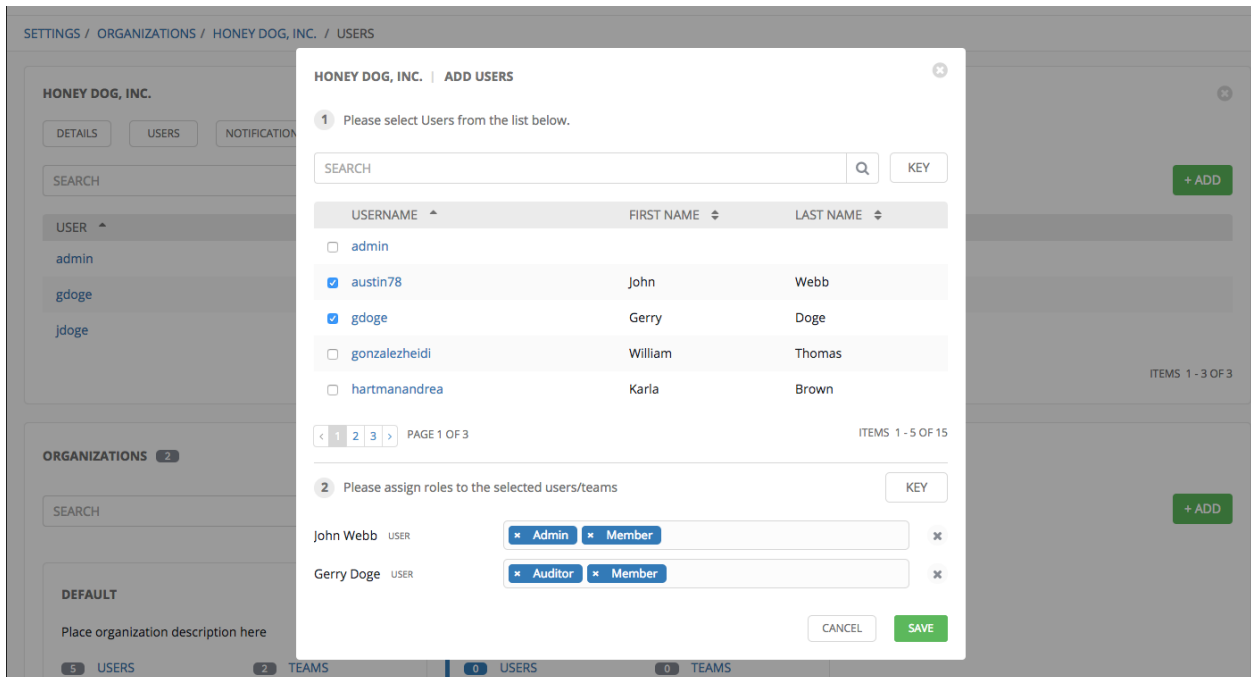


3. For each user, click from the drop-down menu to select one or more roles for that user.

---

**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.

## 7.2 Organizations - Notifications

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

**+ ADD NOTIFICATION TEMPLATE**

Click on the **+ ADD NOTIFICATION TEMPLATE** button to create a notification.

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

SETTINGS / NOTIFICATIONS / CREATE NOTIFICATION TEMPLATE

**NEW NOTIFICATION TEMPLATE**

\* NAME  DESCRIPTION  \* ORGANIZATION

\* TYPE   
 Choose a type

CANCEL SAVE

## 7.3 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.



**ORGANIZATIONS** 2

SEARCH  KEY  **+ ADD**


Organization	Users	Teams	Inventories	Projects	Job Templates	Admins
DEFAULT	5	2	3	2	5	0
HONEY DOG, INC.	2	0	0	0	0	1

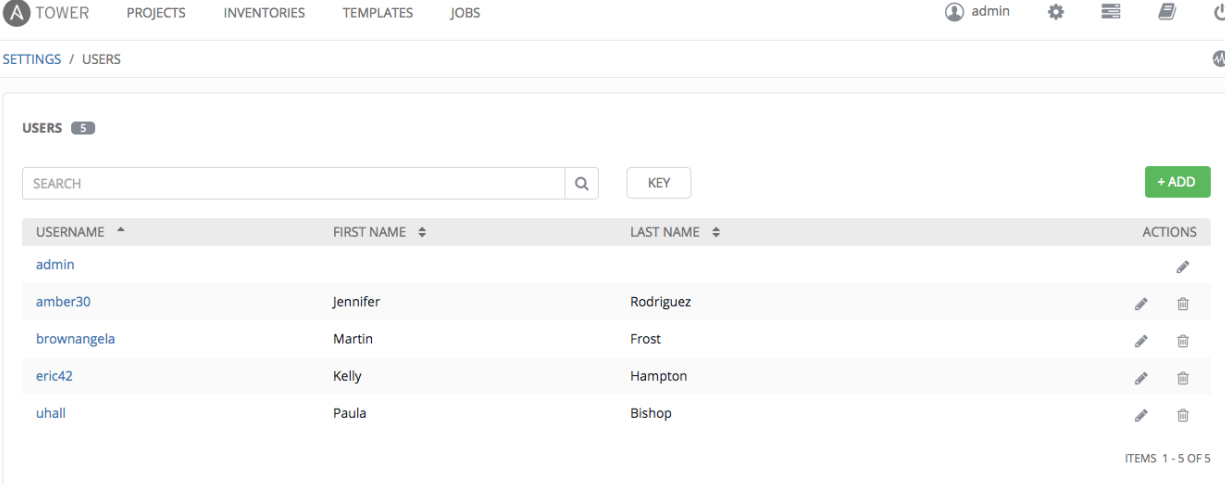
ITEMS 1 - 2 OF 2

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 (  ).

## USERS

A **User** is someone who has access to Tower with associated permissions and credentials. The Users link (found by clicking on the Settings (  ) menu and selecting **Users**) allows you to manage all Tower users. The User list may be sorted and searched by **Username**, **First Name**, or **Last Name** headers to toggle your sorting preference).



USERNAME	FIRST NAME	LAST NAME	ACTIONS
admin			
amber30	Jennifer	Rodriguez	
brownangela	Martin	Frost	
eric42	Kelly	Hampton	
uhall	Paula	Bishop	

ITEMS 1 - 5 OF 5

There are three types of Tower Users that can assigned from the Create User screen:

- **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 **admin, 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.

SETTINGS / USERS / CREATE USER

**NEW USER**

DETAILS ORGANIZATIONS TEAMS PERMISSIONS

\* FIRST NAME  \* LAST NAME  \* EMAIL

\* USERNAME  \* ORGANIZATION  \* 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.

## 8.1 Create a User

To create a new user:

1. Click the  button, which opens the Create User dialog.

TOWER PROJECTS INVENTORIES TEMPLATES JOBS admin

SETTINGS / USERS / CREATE USER

**NEW USER**

DETAILS ORGANIZATIONS TEAMS PERMISSIONS

\* FIRST NAME  \* LAST NAME  \* EMAIL

\* USERNAME  \* ORGANIZATION  \* PASSWORD  SHOW

\* CONFIRM PASSWORD  SHOW

USER TYPE

Normal User

CANCEL SAVE

**USERS** 5 + ADD


SEARCH  KEY


USERNAME	FIRST NAME	LAST NAME	ACTIONS
admin			

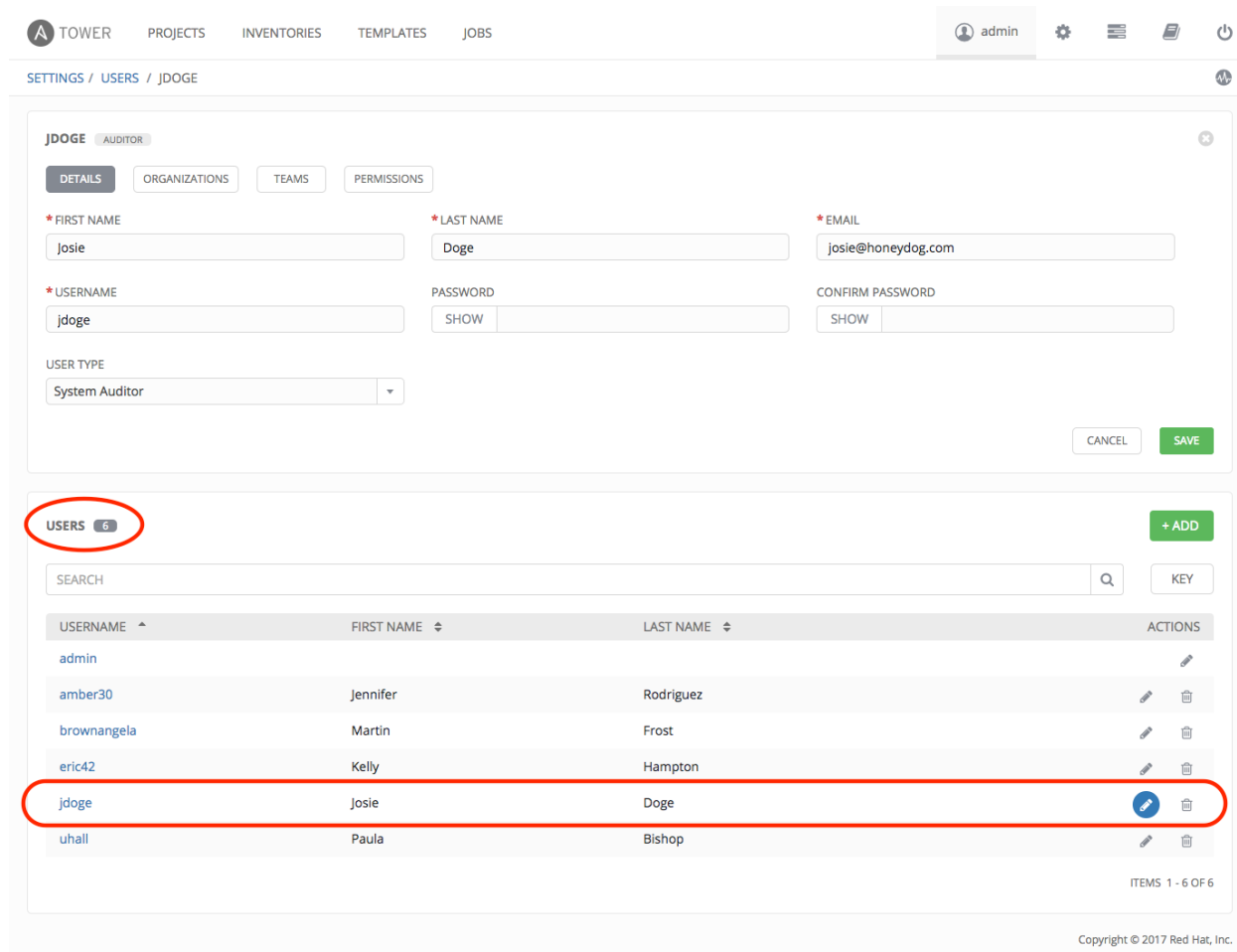
2. Enter the appropriate details into the following fields:
  - First Name
  - Last Name

- Email
  - Username
  - Organization (Choose from an existing organization—this is the default organization if you are using a Self-Supported level license.)
  - Password
  - Confirmation Password
  - User Type (The System Administrator, superuser, has full system administration privileges for Tower. Assign with caution!)
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. This

is the same dialog that is opened if the Edit (  ) button beside a User is clicked from the **Users** link within Tower's

Settings (  ). Here, the User's **Organizations**, **Teams** and **Permissions**, as well as other user membership details, may be reviewed and modified.

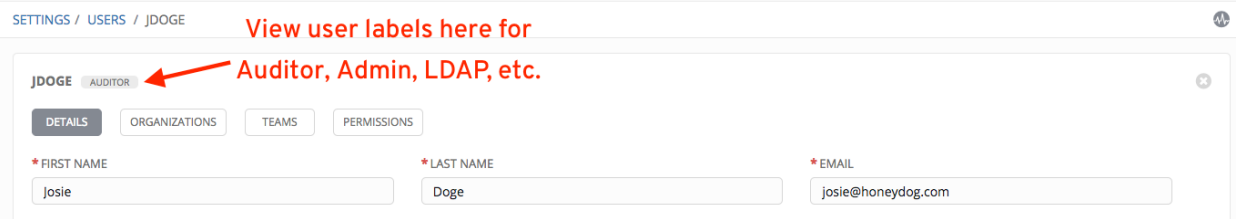


The screenshot displays the Ansible Tower user management interface. At the top, there is a navigation bar with 'TOWER' and menu items like 'PROJECTS', 'INVENTORIES', 'TEMPLATES', and 'JOBS'. The breadcrumb trail indicates the current location: 'SETTINGS / USERS / JDoge'. The main content area shows the 'JDoge' user details form, which is currently in the 'DETAILS' tab. The form includes fields for 'FIRST NAME' (Josie), 'LAST NAME' (Doge), 'EMAIL' (josie@honeydog.com), 'USERNAME' (jdoge), 'PASSWORD', and 'CONFIRM PASSWORD'. The 'USER TYPE' is set to 'System Auditor'. Below the form, there is a 'USERS' list with 6 items. The 'jdoge' user is highlighted with a red circle. The 'USERS' link is also circled in red.

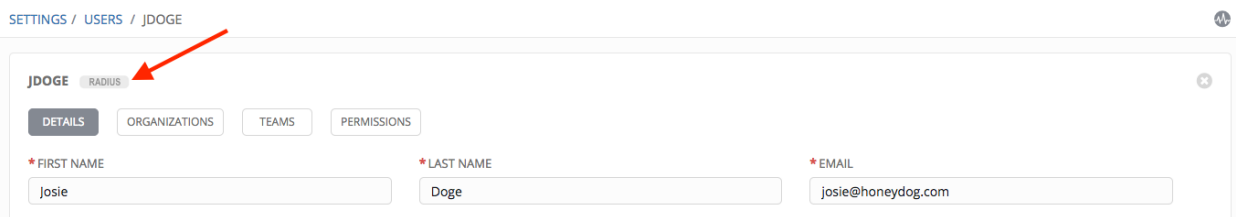
USERNAME	FIRST NAME	LAST NAME	ACTIONS
admin			
amber30	Jennifer	Rodriguez	 
brownangela	Martin	Frost	 
eric42	Kelly	Hampton	 
jdoge	Josie	Doge	 
uhall	Paula	Bishop	 

## 8.2 User Types - Quick View

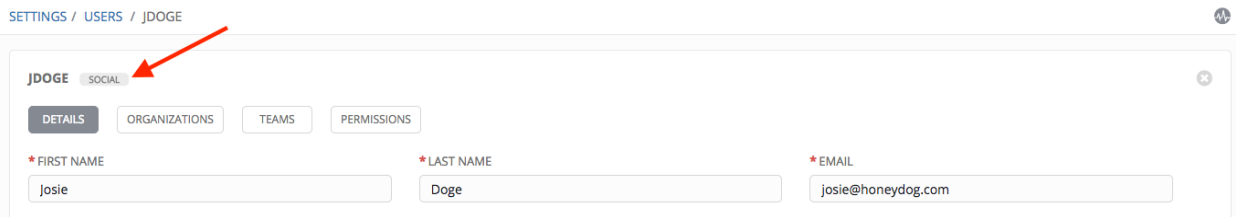
Once a user has been created, you can easily view permissions and user type information by looking beside their user name in the User overview screen.



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

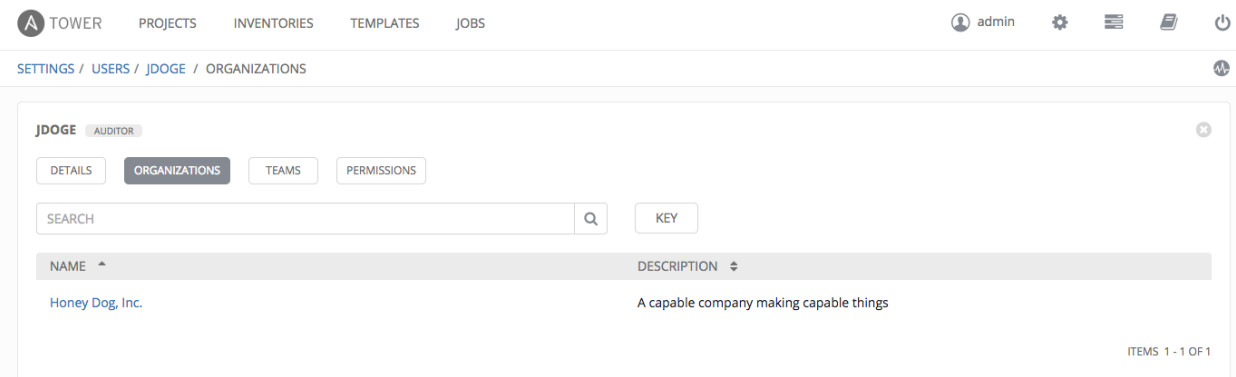


If the user account is associated with a social authentication method, the user type will look like:



## 8.3 Users - Organizations

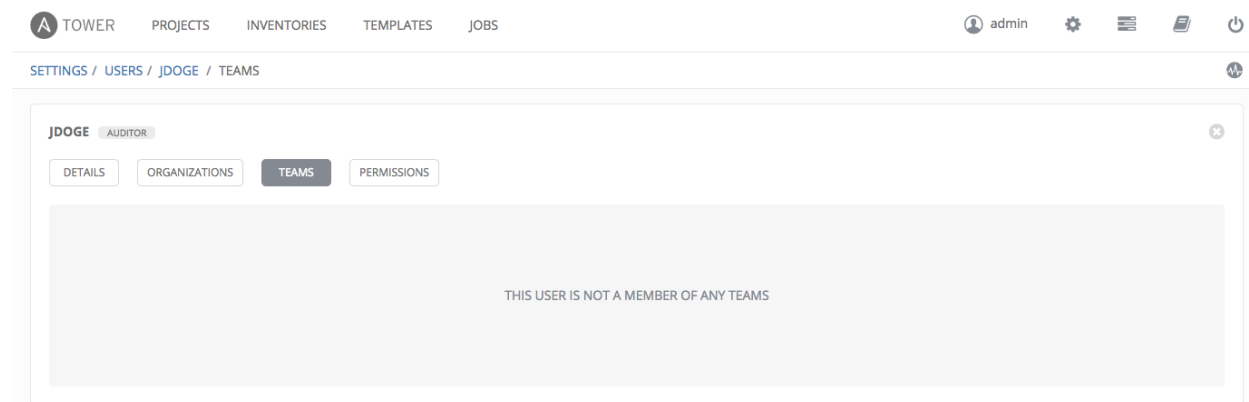
This displays the list of organizations of which that user is a member. This list may be searched by Organization Name or Description. Organization membership cannot be modified from this display panel.



## 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](#).

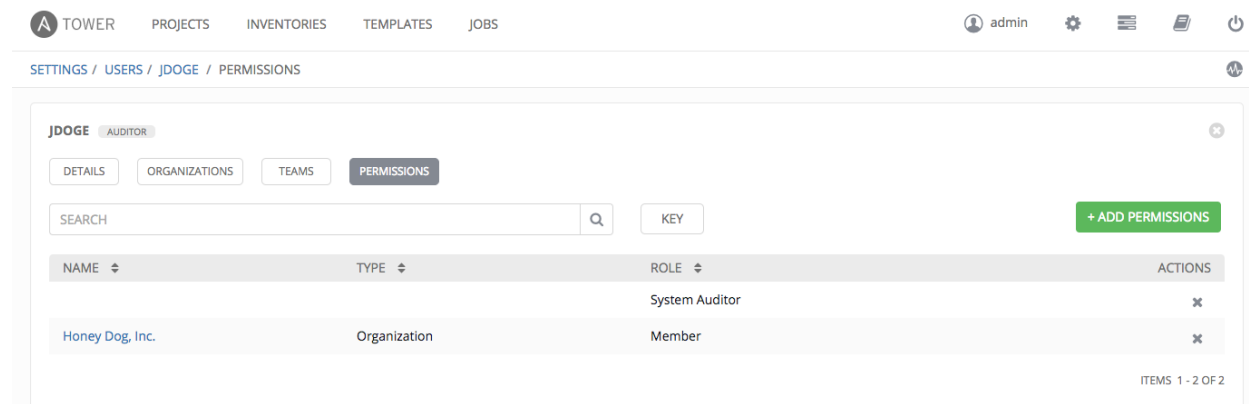
Until a Team has been created and the user has been assigned to that team, the assigned Teams Details for the User appears blank.



## 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.

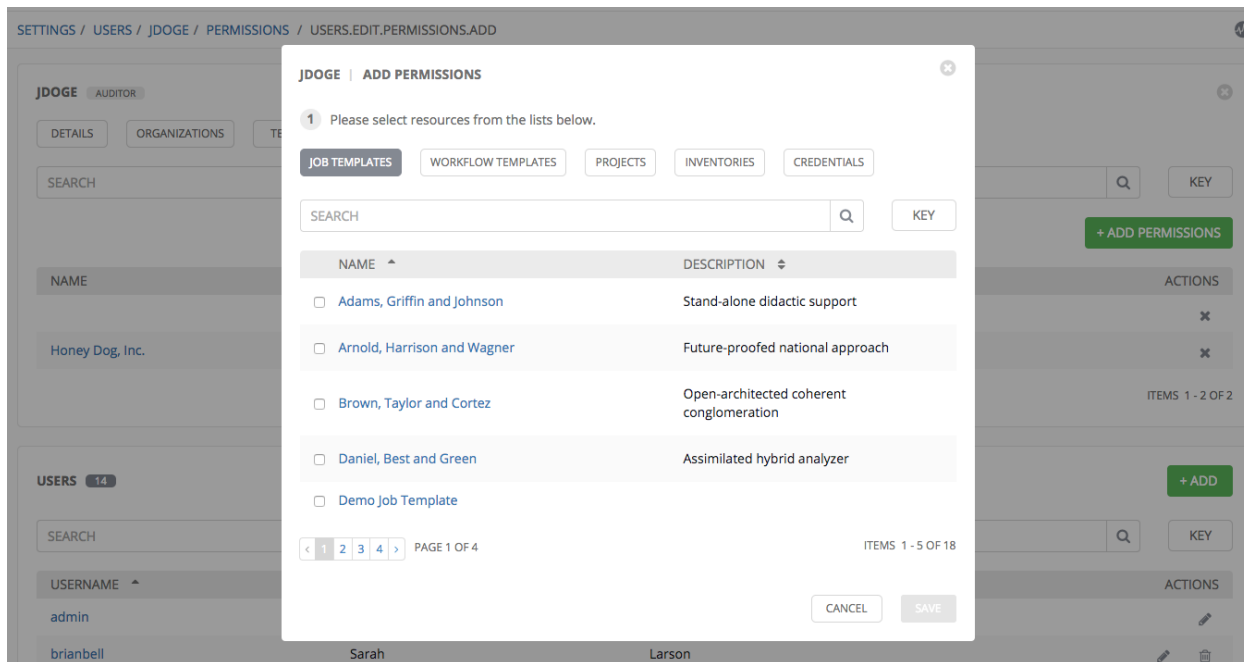
This screen displays a list of the privileges that are currently available for a selected User. The privileges list may be sorted and searched by **Name**, **Type**, or **Role**.



### 8.5.1 Add Permissions

To add permissions to a particular user:

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



2. Click to select the Tower object for which the user will have access:

- **Job Templates.** This is the default tab displayed in the Add Permissions Wizard.
- **Workflow Templates**
- **Projects**
- **Inventories**
- **Credentials**

**Note:** You can assign different roles to different resources all at once to avoid having to click the **+ ADD PERMISSIONS** 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:

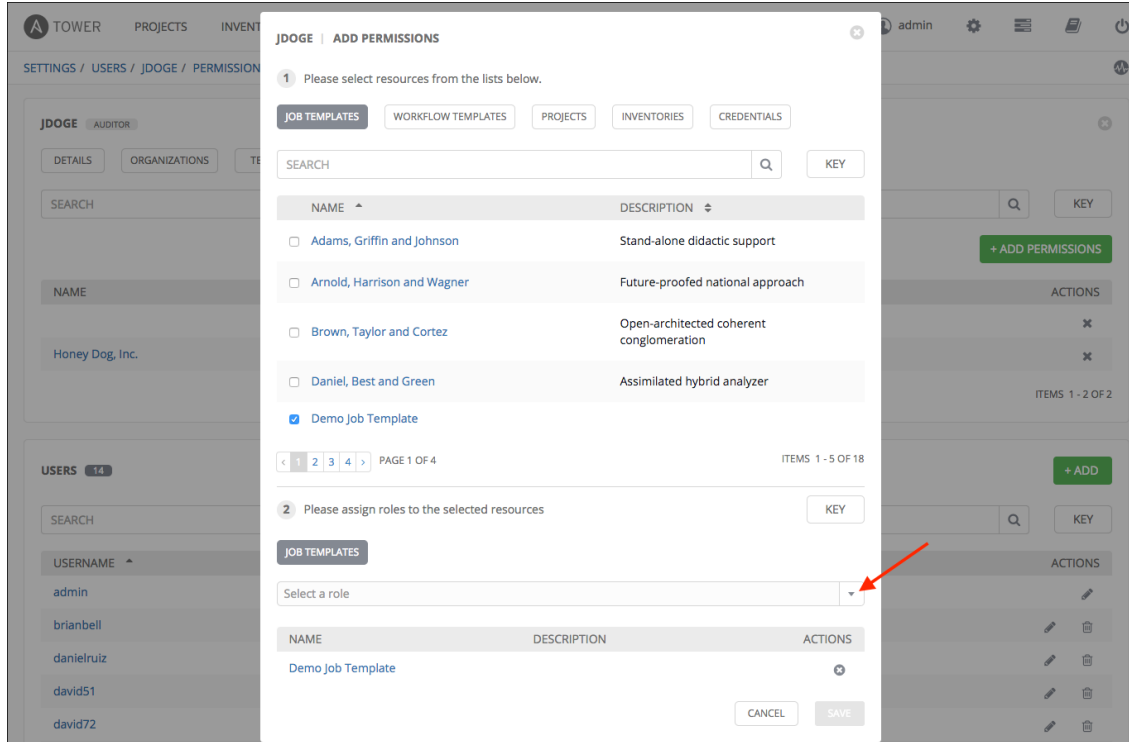
(a) In the desired tab, click the checkbox beside the name of the resource to select it.

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:

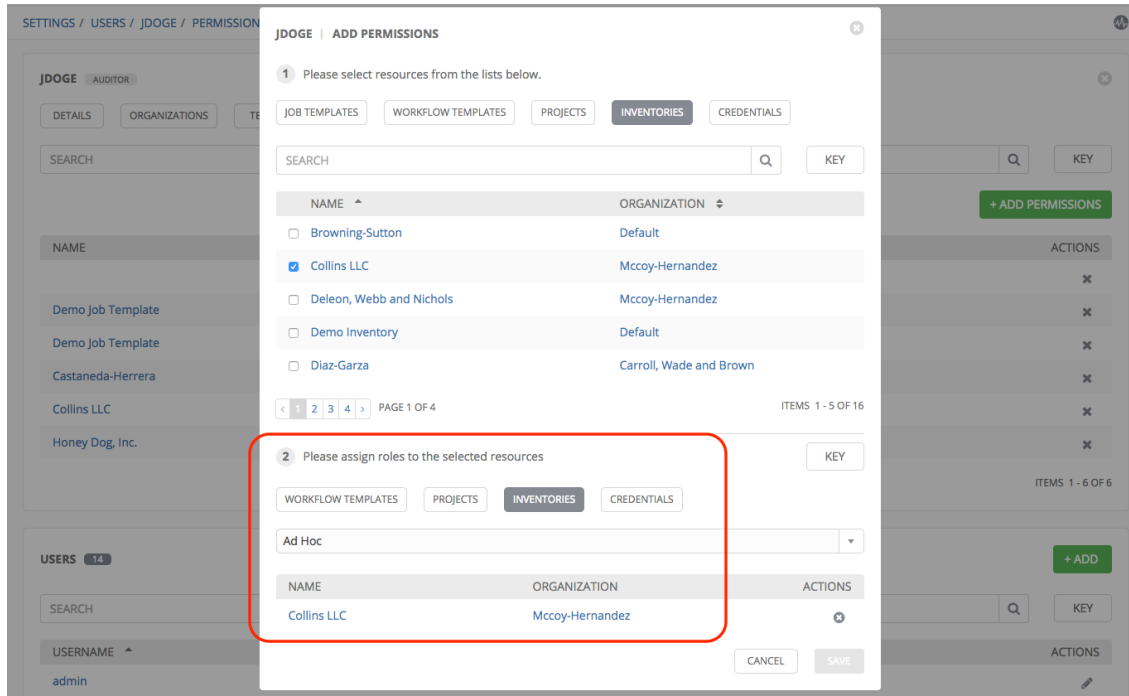
- **Admin** allows read, run, and edit privileges (applicable to all Tower objects)
- **Execute** allows read and run privileges (applicable to job templates and workflow templates)
- **Use** allows use of the project in a job template (applicable to projects, inventories, and credentials)
- **Update** allows updating of project, inventory, or group via the SCM Update (applicable to projects and inventories)
- **Ad Hoc** allows running of ad hoc commands (applicable to inventories)





**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.

SETTINGS / USERS / JDOGE / PERMISSIONS

JDOGE AUDITOR

DETAILS ORGANIZATIONS TEAMS PERMISSIONS

SEARCH

NAME	TYPE	ROLE	ACTIONS
		System Auditor	<input type="button" value="x"/>
Demo Job Template	Job Template	Admin	<input type="button" value="x"/>
Castaneda-Herrera	Project	Update	<input type="button" value="x"/>
Collins LLC	Inventory	Ad Hoc	<input type="button" value="x"/>
Caldwell and Sons	Credential	Use	<input type="button" value="x"/>
Blankenship Inc	Credential	Admin	<input type="button" value="x"/>
Rogers, Powell and Sweeney	Job Template	Execute	<input type="button" value="x"/>
Honey Dog, Inc.	Organization	Member	<input type="button" value="x"/>
Sample Workflow Job Template	Workflow Job Template	Admin	<input type="button" value="x"/>

ITEMS 1 - 9 OF 9

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 v3.1.0*.

## 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.

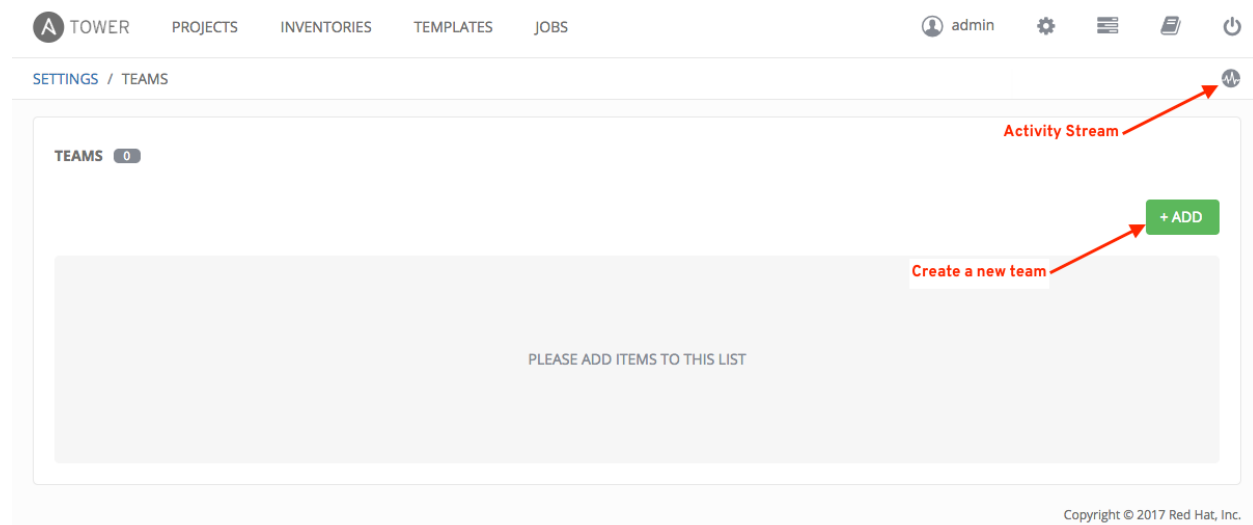
You can create as many Teams of users as make sense for your Organization. Each Team can be assigned permissions, just as with Users.

Teams can also scalably assign ownership for Credentials, preventing multiple Tower interface click-throughs to assign the same Credentials to the same user.

The Teams link, accessible by clicking on the Settings (  ) button and then selecting **Teams**, allows you to manage the teams for Tower. The team list may be sorted and searched by **Name**, **Description**, or **Organization**.

Buttons located in the upper right corner of the **Team** tab provide the following actions:

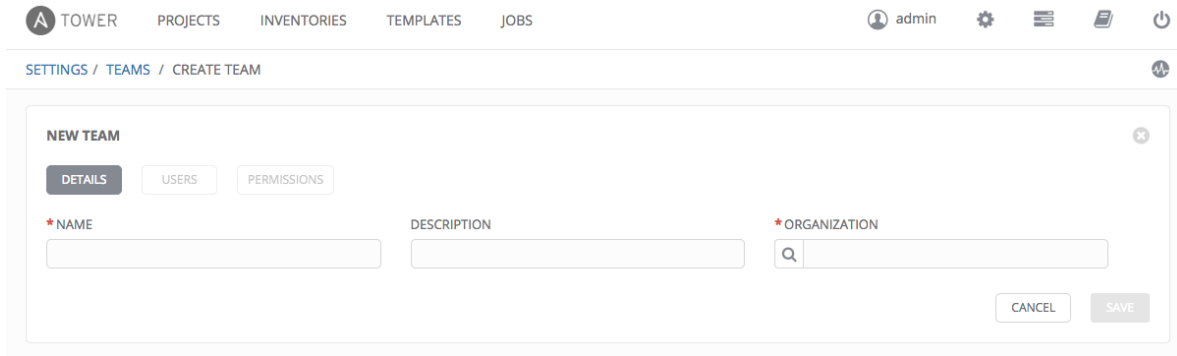
- View Activity Stream
- Create a new team




### 9.1 Create a Team

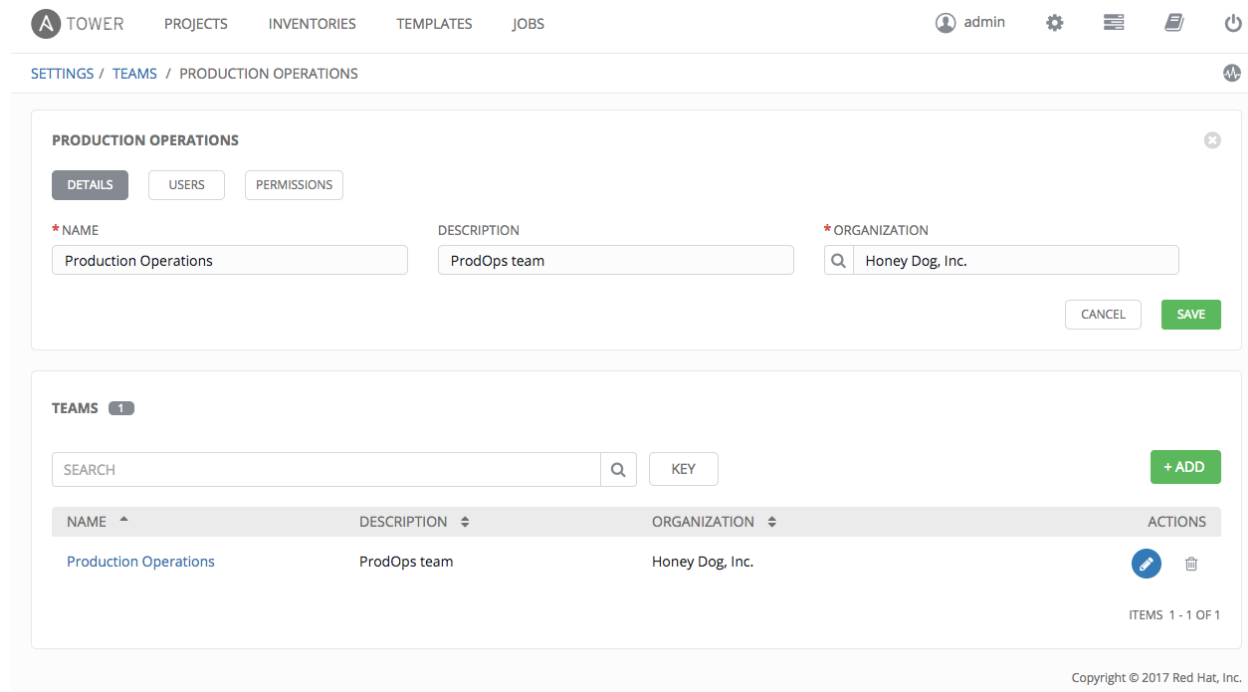
To create a new Team:

1. Click the  button.



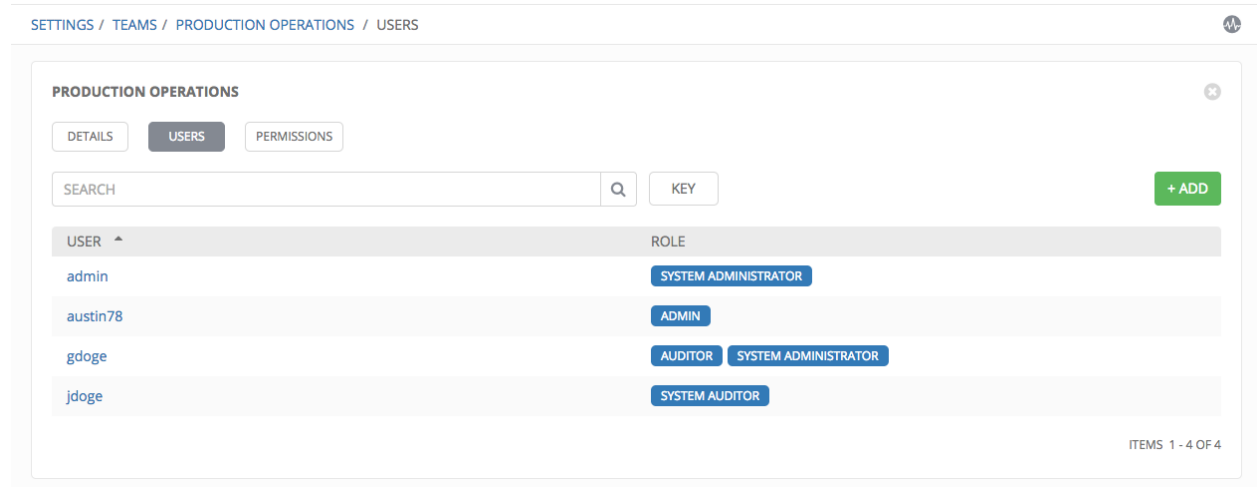
2. Enter the appropriate details into the following fields:
  - Name
  - Description (optional)
  - Organization (Choose from an existing organization)
3. Click **Save**.

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.




## 9.1.1 Teams - Users

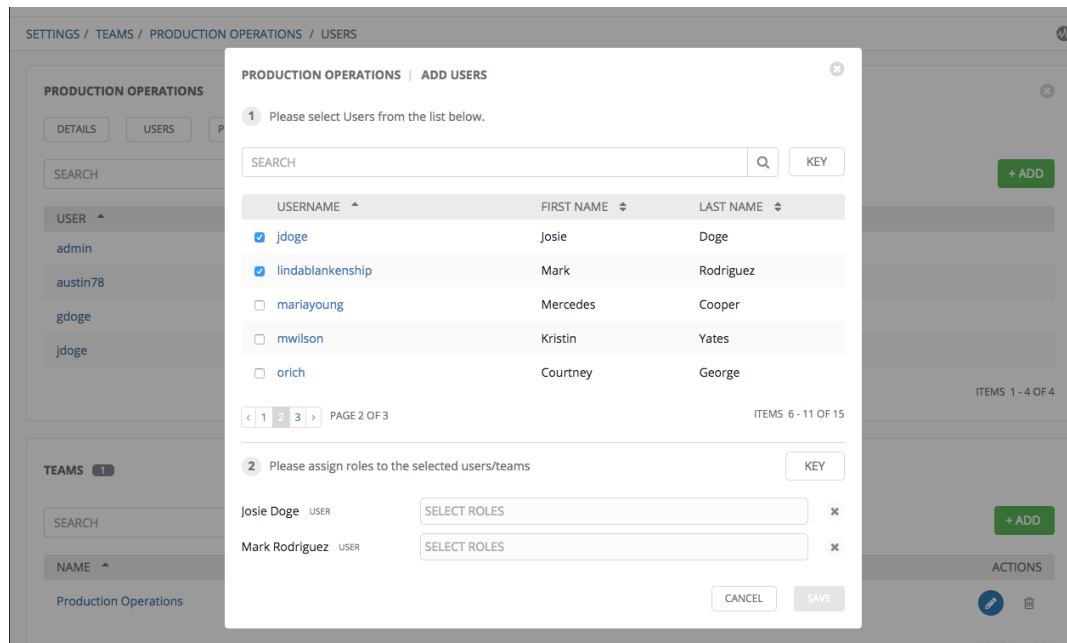
This menu displays the list of Users that are members of this Team. This list may be searched by **Username**, **First Name**, or **Last Name**. For more information, refer to *Users*.



### Add a User

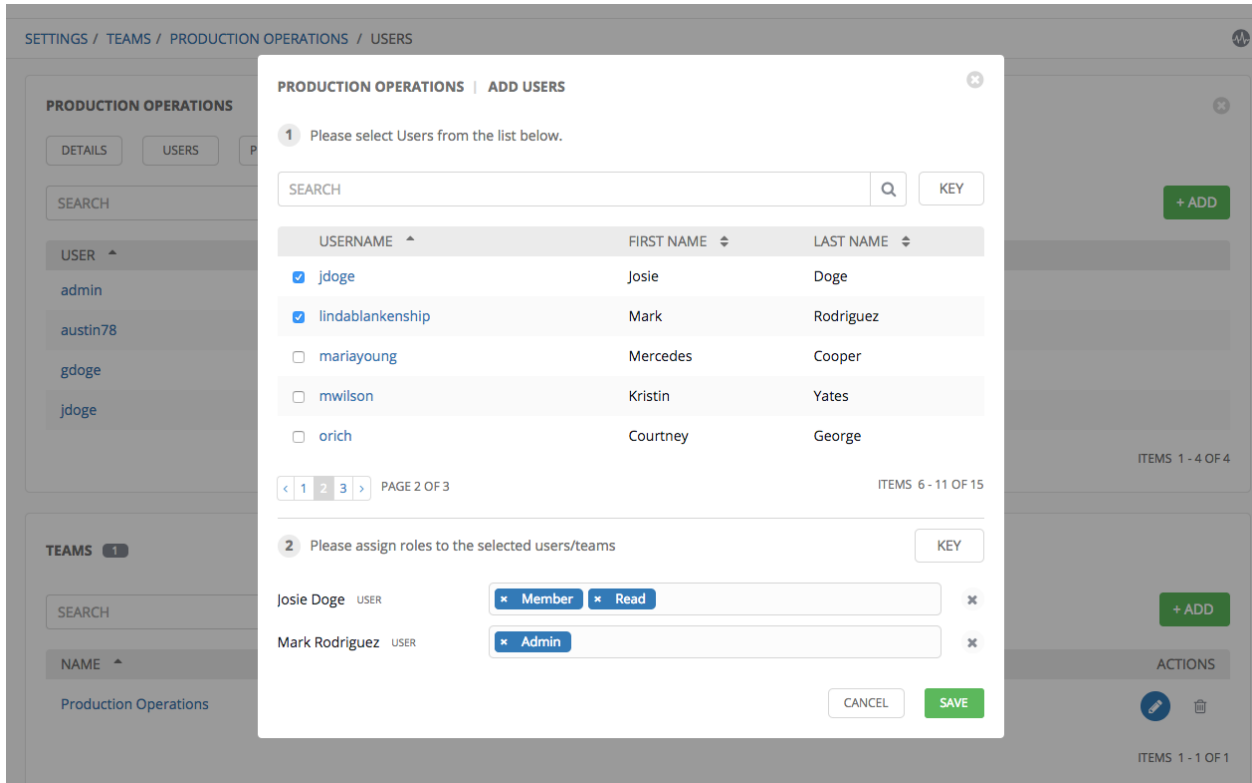
In order to add a user to a team, the user must already be created in Tower. Refer to *Create a User* to create a user. To add existing users to the Team:

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.

PRODUCTION OPERATIONS

DETAILS USERS PERMISSIONS

SEARCH Q KEY + ADD PERMISSIONS

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

ITEMS 1 - 3 OF 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.

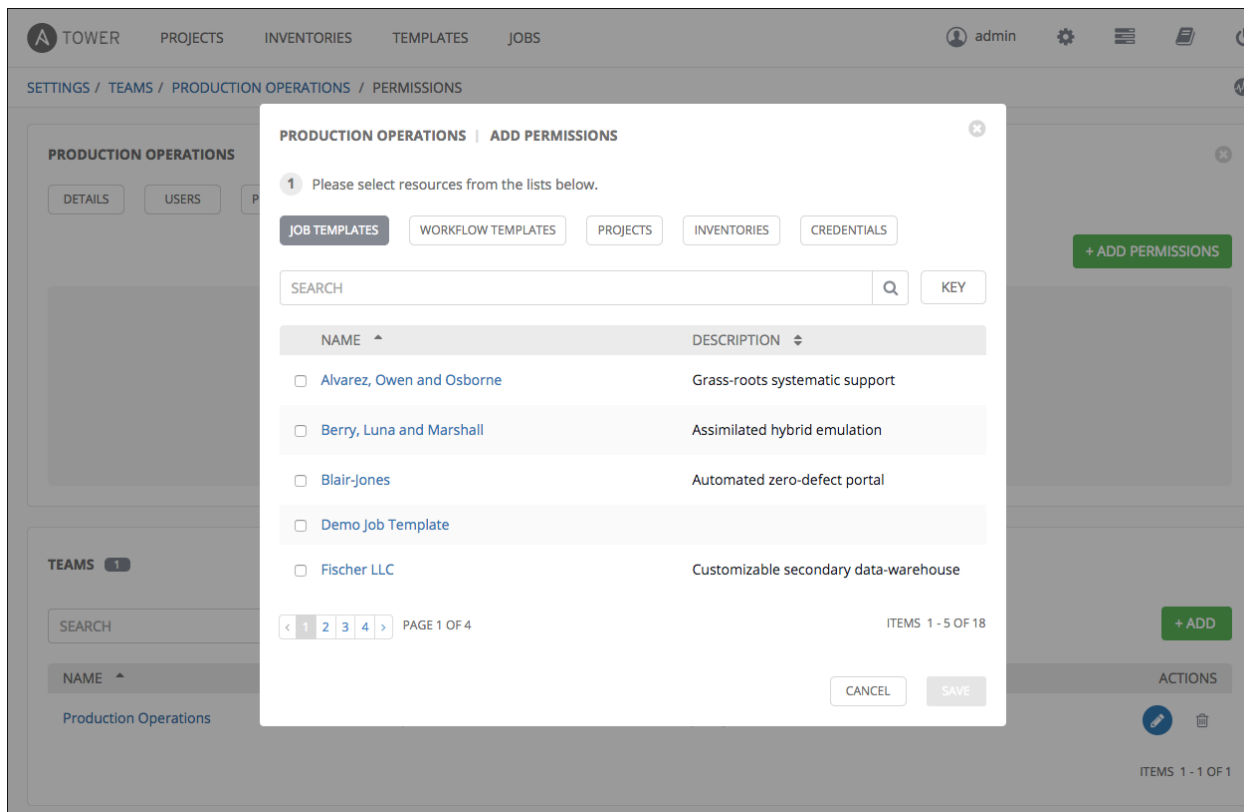
By default, the Team is given the “read” permission (also called a role).

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.



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**

---

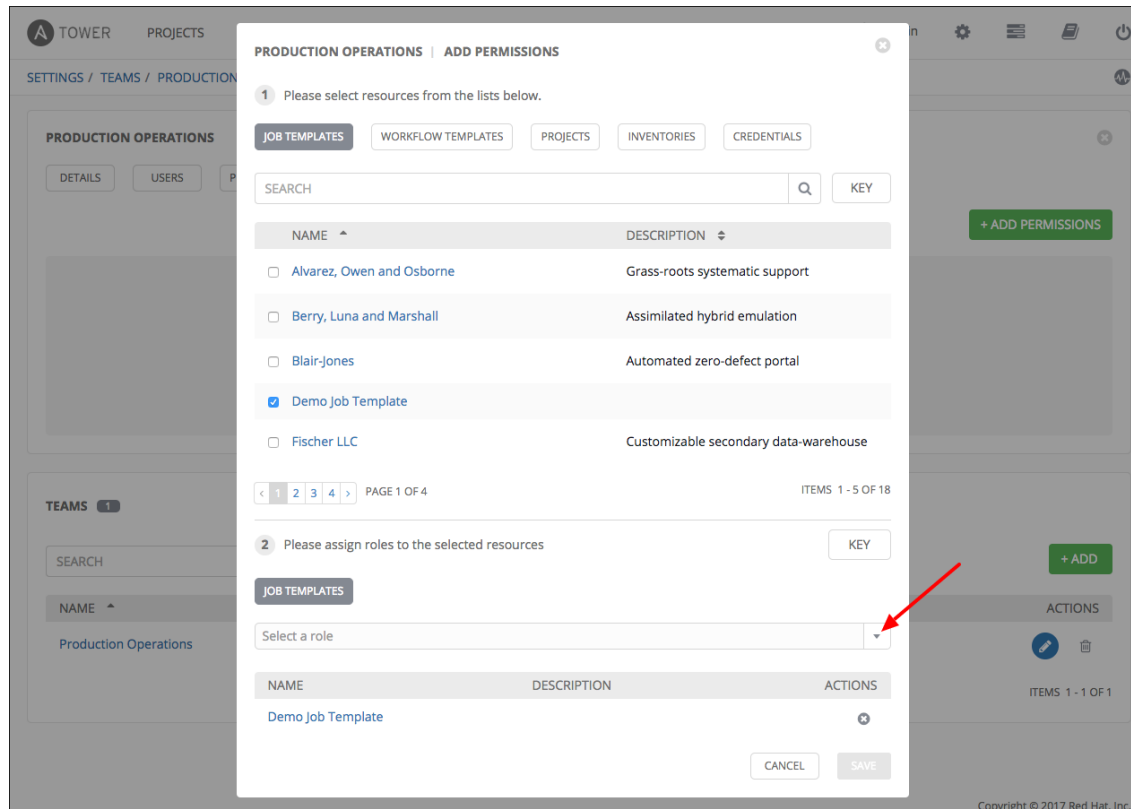
**Note:** You can assign different roles to different resources all at once to avoid having to click the **+ ADD PERMISSIONS** button. To do so, simply go from one tab to another after making your selections without saving.

---

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:
    - **Admin** allows read, run, and edit privileges (applicable to all Tower objects)
    - **Execute** allows read and run privileges (applicable to job templates and workflow templates)
    - **Use** allows use of the project in a job template (applicable to projects, inventories, and credentials)
    - **Update** allows updating of project, inventory, or group via the SCM Update (applicable to projects and inventories)

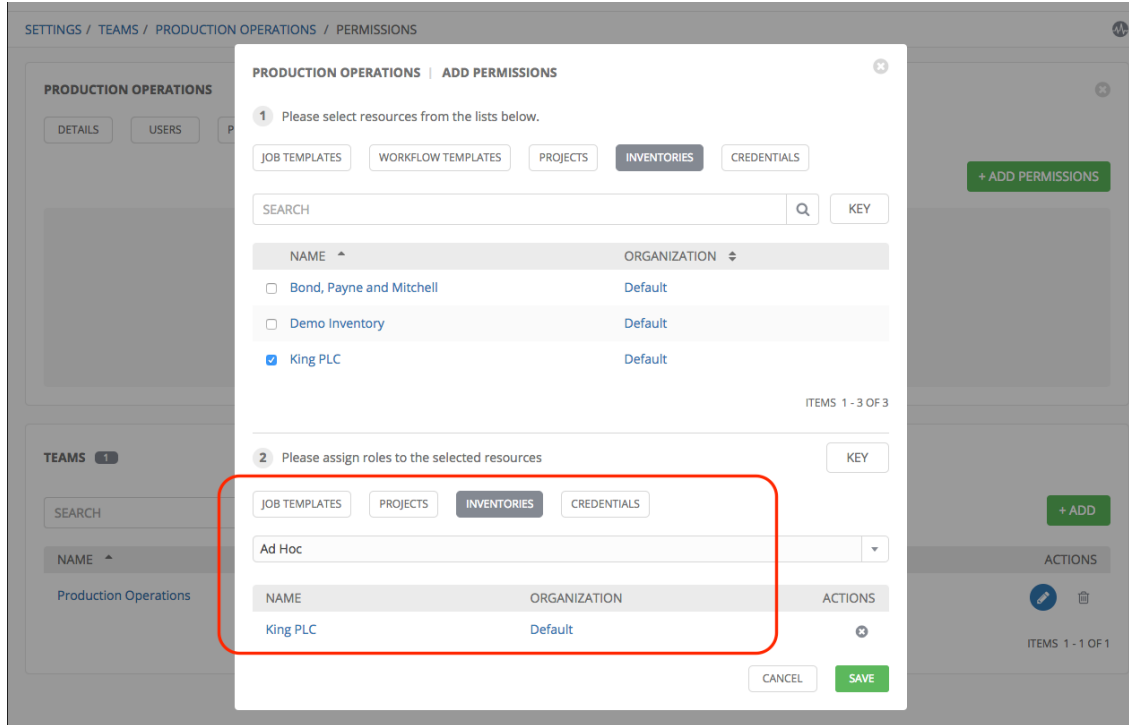


- **Ad Hoc** allows running of ad hoc commands (applicable to inventories)

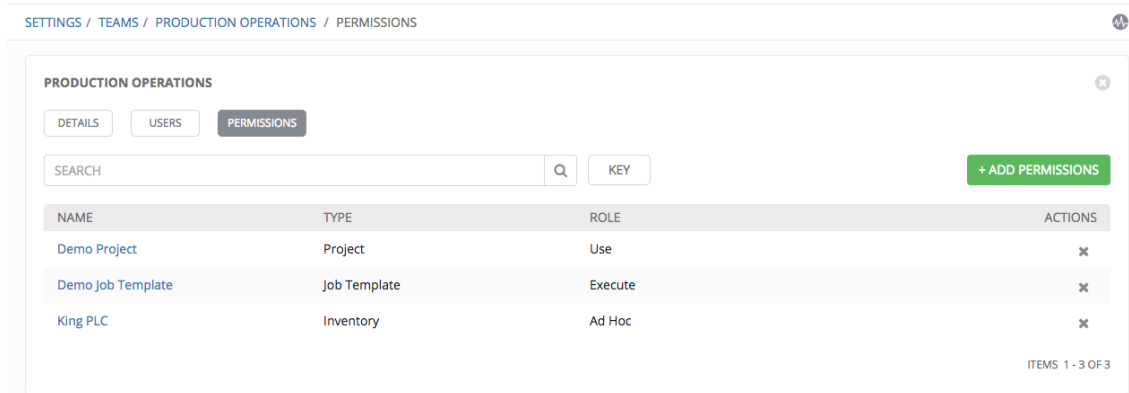


**Tip:** Use the **Key** button to display the help text for each of the roles applicable to the resource selected.

- 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 v3.1.0*.

## 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.

Tower credentials are imported and stored encrypted in Tower, and are not retrievable in plain text on the command line by any user. Once a password or key has been entered into the Tower interface, it is encrypted and inserted into the Tower database, and cannot be retrieved from Tower. 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 Electronic Code Book (ECB) as the mode of operation with AES-128 as the block cipher. The 128-bit AES 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-128 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.

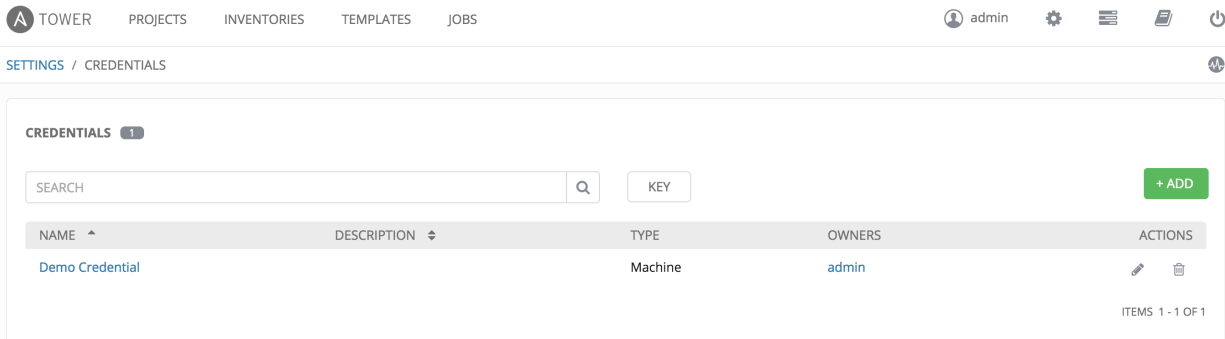
---



**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

The **Credentials** link, accessible from the Setting (  ) button, displays a list of all available Credentials. It can be sorted and searched by **Name**, **Description**, **Type**, or **Owners**.

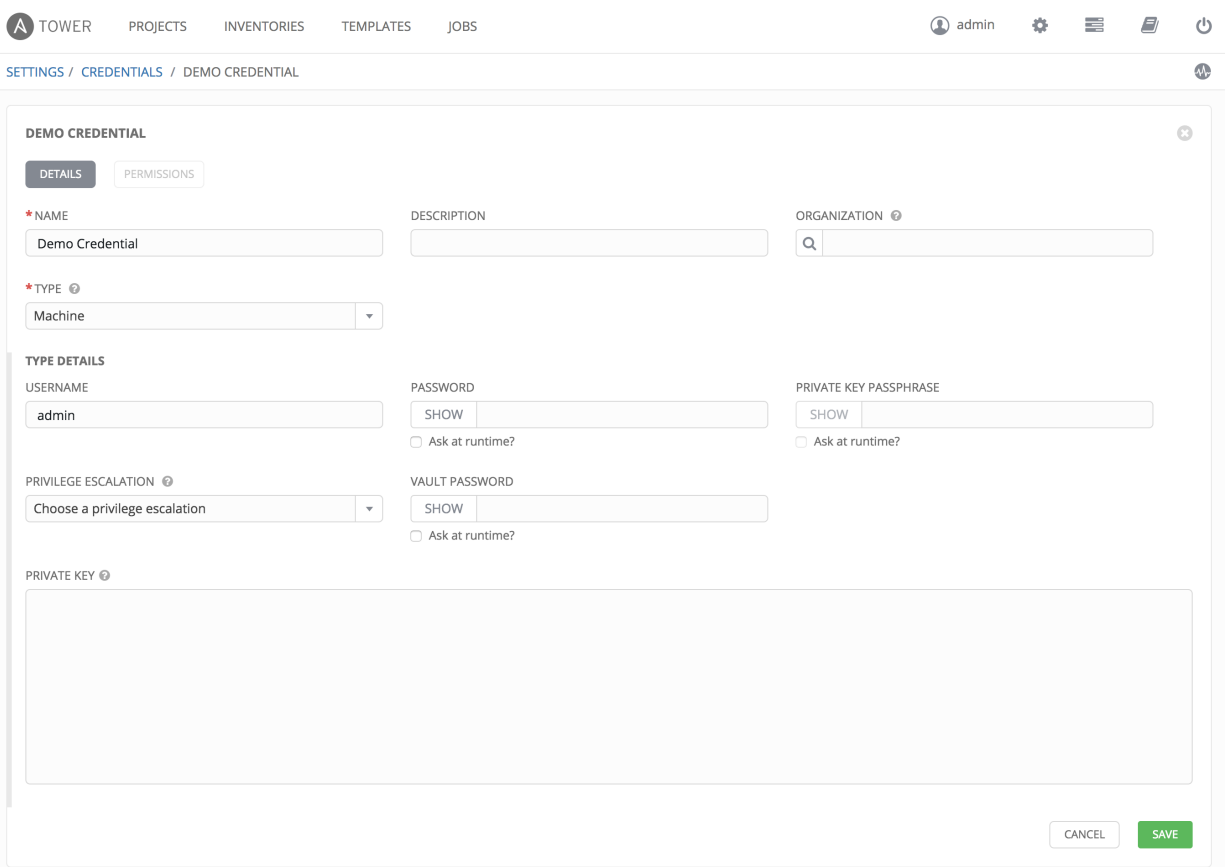


NAME	DESCRIPTION	TYPE	OWNERS	ACTIONS
Demo Credential		Machine	admin	 

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.

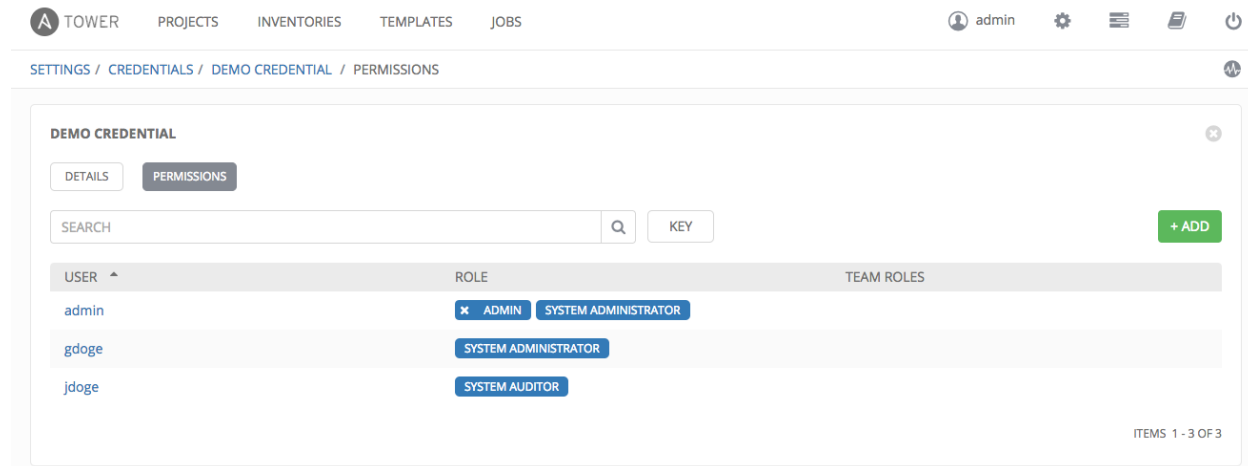
To help you get started, a Demo Credential has been created for your use.

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.)

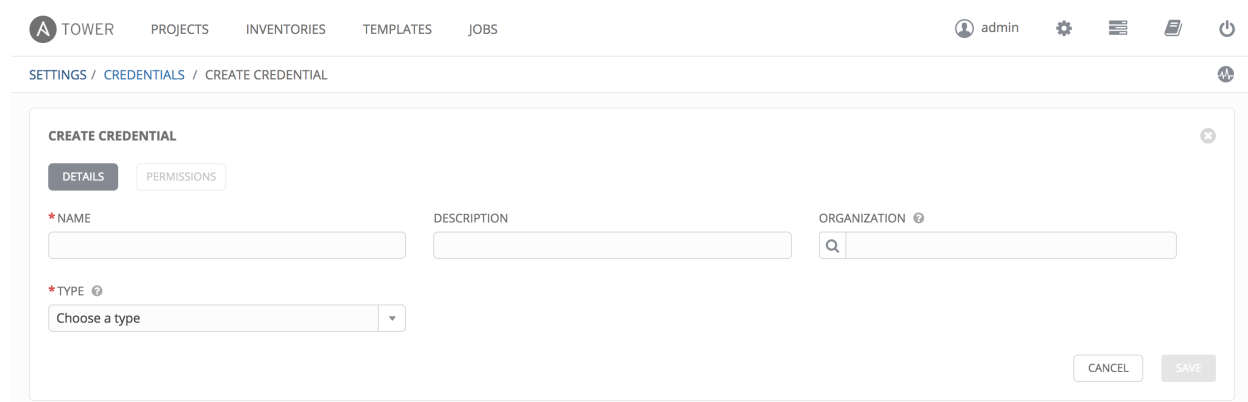


You can click the  button to assign this **Demo Credential** to additional Users or Teams.

## 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.



2. Enter the appropriate details depending on the type of credential as described in the following sections.
3. Click **Save** when done.

## 10.4 Credential Types

### Topics:

- *Machine*

- *Network*
- *Source Control*
- *Amazon Web Services*
- *Rackspace*
- *VMware vCenter*
- *Red Hat Satellite 6*
- *Red Hat CloudForms*
- *Google Compute Engine*
- *Microsoft Azure Classic (Deprecated)*
- *Microsoft Azure Resource Manager*
- *OpenStack*

### 10.4.1 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.

**CREATE CREDENTIAL** ✕

DETAILS
PERMISSIONS

\*NAME DESCRIPTION ORGANIZATION ?

\*TYPE ?

Machine

**TYPE DETAILS**

USERNAME PASSWORD PRIVATE KEY PASSPHRASE

Ask at runtime?  Ask at runtime?

PRIVILEGE ESCALATION ? VAULT PASSWORD

Choose a privilege escalation

Ask at runtime?

PRIVATE KEY ?

CANCEL
SAVE

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 can be stored encrypted in the Tower database, if entered. Alternatively, you can configure Tower to ask the user for the password when necessary by selecting “**Ask at runtime?**”. In these cases, a dialog opens when the job is launched, promoting the user to enter the password and password confirmation.
- **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 may be stored encrypted in the Tower database, if entered. Alternatively, you can configure Tower to ask the user for the password as necessary by selecting “**Ask at runtime?**”. In these cases, a dialog opens when the job is launched, prompting the user to enter the password and password confirmation.
- **Privilege Escalation:** 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`.
  - **none:** 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](#))
  - **pmrun:** Requests that an application is run in a controlled account (refer to [Privilege Manager for Unix 6.0](#))

- **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 may be stored encrypted in the Tower database, if entered. Alternatively, you may configure Tower to ask the user for the password when necessary by selecting “**Ask at runtime?**”. 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.

---

- **Vault Password:** If your playbook uses Ansible Vault, add the Vault password to your credentials here. Alternatively, you may configure Tower to ask the user for the vault password when necessary by selecting “**Ask at runtime?**”. In these cases, a dialog opens when the job is launched, promoting the user to enter the password and password confirmation.

For more information about Ansible Vault, refer to: [http://docs.ansible.com/playbooks\\_vault.html](http://docs.ansible.com/playbooks_vault.html)

**Warning:** Credentials which are used in *Scheduled Jobs* must not be configured as “Ask at runtime?”.

## 10.4.2 Network

Network credentials are used by Ansible networking modules to connect to and manage networking devices.

SETTINGS / CREDENTIALS / CREATE CREDENTIAL

**CREATE CREDENTIAL**

DETAILS | PERMISSIONS

\*NAME  DESCRIPTION  ORGANIZATION

\*TYPE

**TYPE DETAILS**

\*USERNAME  PASSWORD  PRIVATE KEY PASSPHRASE

AUTHORIZE

SSH KEY

CANCEL SAVE

Network credentials have several attributes that may be configured:

- **Username:** The username to use in conjunction with the network device.
- **Password:** The password to use in conjunction with the network device.
- **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 to add an **Authorize Password** which signs the RSA key with a password (the **Authorize Password** field is only seen if this option is selected).
- **SSH Key:** The actual SSH Private Key to be used to authenticate the user to the network via SSH.

## 10.4.3 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.



**CREATE CREDENTIAL** ✕

DETAILS
PERMISSIONS

\*NAME
DESCRIPTION
ORGANIZATION ?

\*TYPE ?

Source Control

**TYPE DETAILS**

USERNAME
PASSWORD
PRIVATE KEY PASSPHRASE

SHOW

SHOW

SCM PRIVATE KEY ?

CANCEL
SAVE

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.
- **Private Key Passphrase:** If the SSH Private Key used is protected by a passphrase, you may configure a Key Passphrase for the private key.
- **SCM Private Key:** The actual SSH Private Key to be used to authenticate the user to the source control system via SSH.

---

**Note:** Source Control credentials cannot be configured as “Ask at runtime?”.

---

#### 10.4.4 Amazon Web Services

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

SETTINGS / CREDENTIALS / CREATE CREDENTIALIAL

DETAILS PERMISSIONS

\* NAME DESCRIPTION ORGANIZATION

\* TYPE  
Amazon Web Services

TYPE DETAILS

\* ACCESS KEY \* SECRET KEY STS TOKEN

SHOW SHOW

CANCEL SAVE

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](http://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_temp.html)

AWS credentials consist of:

```
AWS_ACCESS_KEY
AWS_SECRET_KEY
AWS_SECURITY_TOKEN
```

**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.)

## 10.4.5 Rackspace

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

Rackspace credentials consist of the Rackspace **Username** and **API Key**.

---

**Note:** Rackspace inventory sync has been deprecated in Tower 3.1.0 and support for Rackspace will be removed in a future release.

---

## 10.4.6 VMware vCenter

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

VMware credentials have several attributes that may be configured:

- **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.

---

## 10.4.7 Red Hat Satellite 6

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

SETTINGS / CREDENTIALS / CREATE CREDENTIAL

DETAILS PERMISSIONS

\* NAME  DESCRIPTION  ORGANIZATION

\* TYPE

TYPE DETAILS

\* SATELLITE 6 URL  \* USERNAME  \* PASSWORD

CANCEL SAVE

Satellite credentials have several attributes that may be configured:

- **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.

## 10.4.8 Red Hat CloudForms

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

SETTINGS / CREDENTIALS / CREATE CREDENTIAL

DETAILS PERMISSIONS

\* NAME  DESCRIPTION  ORGANIZATION

\* TYPE

TYPE DETAILS

\* CLOUDFORMS URL  \* USERNAME  \* PASSWORD

CANCEL SAVE

CloudForms have several attributes that may be configured:

- **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/>.

## 10.4.9 Google Compute Engine

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

SETTINGS / CREDENTIALS / CREATE CREDENTIAL

**CREATE CREDENTIAL**

DETAILS PERMISSIONS

\*NAME DESCRIPTION ORGANIZATION

\*TYPE  
Google Compute Engine

**TYPE DETAILS**

\*SERVICE ACCOUNT EMAIL ADDRESS PROJECT

\*RSA PRIVATE KEY

CANCEL SAVE

Google Compute Engine credentials have several attributes that may be configured:

- **Service Account Email Address:** The email address assigned to the Google Compute Engine **service account**.
- **Project:** The GCE assigned identification. It is constructed as two words followed by a three digit number, such as: squeamish-ossifrage-123.
- **RSA Private Key:** The PEM file associated with the service account email.

## 10.4.10 Microsoft Azure Classic (Deprecated)

Selecting this credential type enables synchronization of cloud inventory with Windows Azure Classic.

SETTINGS / CREDENTIALS / CREATE CREDENTIAL ⌵

**CREATE CREDENTIAL** ✕

**DETAILS** | PERMISSIONS

\*NAME  DESCRIPTION  ORGANIZATION

\*TYPE

**TYPE DETAILS**

\*SUBSCRIPTION ID

\*MANAGEMENT CERTIFICATE

CANCEL SAVE

Microsoft Azure credentials have several attributes to configure:

- **Subscription ID:** The Subscription UUID for the Microsoft Azure Classic account.
- **Management Certificate:** The PEM file that corresponds to the certificate you uploaded in the Microsoft Azure Classic console.

### 10.4.11 Microsoft Azure Resource Manager

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

SETTINGS / CREDENTIALS / CREATE CREDENTIAL ⌵

**CREATE CREDENTIAL** ✕

**DETAILS** | PERMISSIONS

\*NAME  DESCRIPTION  ORGANIZATION

\*TYPE

**TYPE DETAILS**

*SUBSCRIPTION ID <input type="text"/>	USERNAME <input type="text"/>	PASSWORD <input type="text"/>
CLIENT ID <input type="text"/>	CLIENT SECRET <input type="text"/>	TENANT ID <input type="text"/>

CANCEL SAVE

Microsoft Azure Resource Manager credentials have several attributes to configure:

- **Subscription ID:** The Subscription UUID for the Microsoft Azure account.
- **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.

To pass service principal credentials, define the following variables:

```
AZURE_CLIENT_ID
AZURE_SECRET
AZURE_SUBSCRIPTION_ID
AZURE_TENANT
```

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
```

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

```
ad_user
password
subscription_id
```

### 10.4.12 OpenStack

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

SETTINGS / CREDENTIALS / CREATE CREDENTIAL 🔍

**CREATE CREDENTIAL** ✕

DETAILS PERMISSIONS

\*NAME  DESCRIPTION  ORGANIZATION

\*TYPE

**TYPE DETAILS**

\*HOST (AUTHENTICATION URL)  \*USERNAME  \*PASSWORD (API KEY)

\*PROJECT (TENANT NAME)  DOMAIN NAME

CANCEL SAVE

OpenStack credentials have several attributes that may be configured:

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



## 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, and Mercurial.

---

**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 **Status**, **Name**, or **Type**. For each project listed, you can edit project properties and delete the project, using the edit and delete icons.

The screenshot shows the Tower web interface. At the top, there is a navigation bar with tabs for TOWER, PROJECTS (selected), INVENTORIES, TEMPLATES, and JOBS. On the right, there are icons for user (admin), settings, help, and power. Below the navigation bar, the main content area is titled 'PROJECTS' and contains a search bar with a 'KEY' input field and a '+ ADD' button. A table lists two projects:

NAME	TYPE	REVISION	LAST UPDATED	ACTIONS
Demo Project	Git	b9502f756a27f072bda494505018c0d12 26df01d	2/27/2017 1:43:30 PM	[Icons for edit and delete]
Harmon Group	Git	997519cb18bc15ed2857fffa02fd6a4e b0bfe8e	2/27/2017 5:21:11 PM	[Icons for edit and delete]

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

**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):

- **New:** The source control update has been created, but not queued or started yet. (To be deprecated.)
- **Pending:** The source control update has been queued, but has not run yet. (To be deprecated.)
- **Waiting:** The source control update is waiting on an update/dependency.
- **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).

Under **Actions**, the following actions are available:

- **Update:** Invoke an immediate update from source control, if configured for this project
- **Schedule:** Schedule an update from source control, if configured for this project
- **Edit:** Edit the project
- **Delete:** Delete the project


---

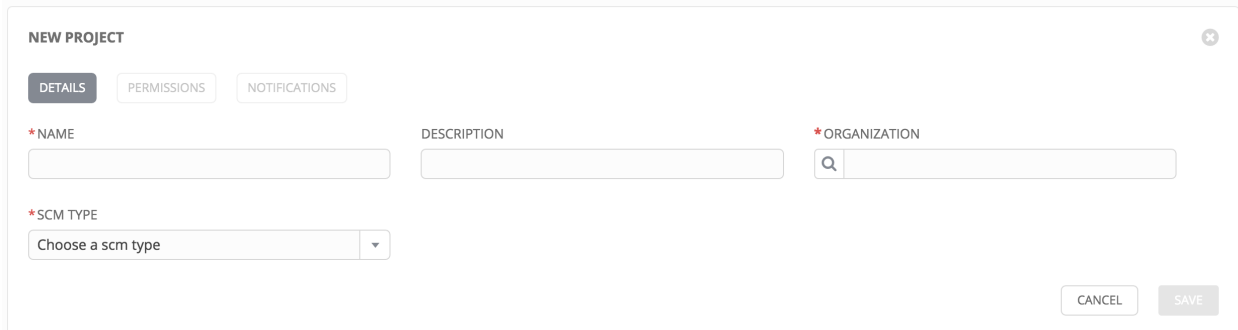
**Note:** Projects of credential type Manual cannot update or schedule source control-based actions without being reconfigured as an SCM type credential.

---

## 11.1 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 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.)
- **SCM Type** (Select one of Manual, Git, Subversion, or Mercurial. Refer to *Manage playbooks manually* and *Manage playbooks using Source Control* in this guide for more detail.)

---

**Note:** Each project path 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.

### 11.1.1 Manage playbooks manually

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

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

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

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

### 11.1.2 Manage playbooks using Source Control

1. Select the appropriate **SCM Type**.
2. Enter the appropriate details into the following fields:
  - SCM URL
  - SCM Branch (Optionally enter the SCM branch for Git or Mercurial.)
  - 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.









## 11.2 Updating projects from source control

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

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





**PROJECTS** 2

SEARCH  Q KEY + ADD



NAME ^	TYPE ^	REVISION ^	LAST UPDATED	ACTIONS
<span style="color: green;">●</span> Demo Project	Git	b9502f756a27f072bda494505018c0d12 26df01d	2/27/2017 1:43:30 PM	   
<span style="color: green;">●</span> Harmon Group	Git	997519cb18bc15ed2857ffaf02fd6a4e b0bfe8e	2/27/2017 5:21:11 PM	   

ITEMS 1 - 2 OF 2

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

TOWER PROJECTS INVENTORIES TEMPLATES JOBS admin    

JOBS / DEMO PROJECT

**RESULTS**  

NAME Demo Project

STATUS ● Successful



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

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

ELAPSED 3.775 seconds

LAUNCH TYPE Dependency

PROJECT Demo Project


**STANDARD OUT**  

```

Using /etc/ansible/ansible.cfg as config file
PLAY [all] *****
TASK [delete project directory before update] *****
skipping: [localhost]
TASK [update project using git and accept hostkey] *****
skipping: [localhost]
TASK [Set the git repository version] *****
skipping: [localhost]
TASK [update project using git] *****
ok: [localhost]
TASK [Set the git repository version] *****
ok: [localhost]
TASK [update project using hg] *****
skipping: [localhost]
TASK [Set the hg repository version] *****
skipping: [localhost]

```

Copyright © 2017 Red Hat, Inc.

3. To set a schedule for updating the project from SCM, click the  button. This will navigate to the **Schedules** screen.

EXAMPLE | SCHEDULES 0 + ADD

PLEASE ADD ITEMS TO THIS LIST

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.

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

- Create a new schedule
- Refresh this view
- View Activity Stream

## 11.3 Add a new schedule

To create a new schedule:

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

PROJECTS / EXAMPLE / SCHEDULES / CREATE SCHEDULE

* NAME Test Once	* START DATE (MM/DD/YYYY) 06/22/2016	* START TIME (HH24:MM:SS) 12:00:00
* LOCAL TIME ZONE America/New_York	* REPEAT FREQUENCY None (run once)	

**SCHEDULE DESCRIPTION**

every day for 1 time

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

06/22/2016 12:00:00 EDT

EXTRA VARIABLES  YAML     JSON

1 ---

CANCEL    SAVE

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.

There are several actions available for schedules, under the **Actions** column:

- Edit Schedule
- Delete schedule

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

## 11.4 Ansible Galaxy Support

At the end of a Project update, Tower searches for a file called `requirements.yml` in the roles directory, ‘<project-top-level-directory>/roles/requirements.yml’

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.

## 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 the [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** or **Organization**. This list may also be filtered by selecting **Cloud Sourced**, **Failed Hosts**, or **Sync Failures**.

NAME ^	ORGANIZATION ▾	ACTIONS
Bond, Payne and Mitchell	Default	
Demo Inventory	Default	
King PLC	Default	

ITEMS 1 - 3 OF 3


The list of Inventory details includes:

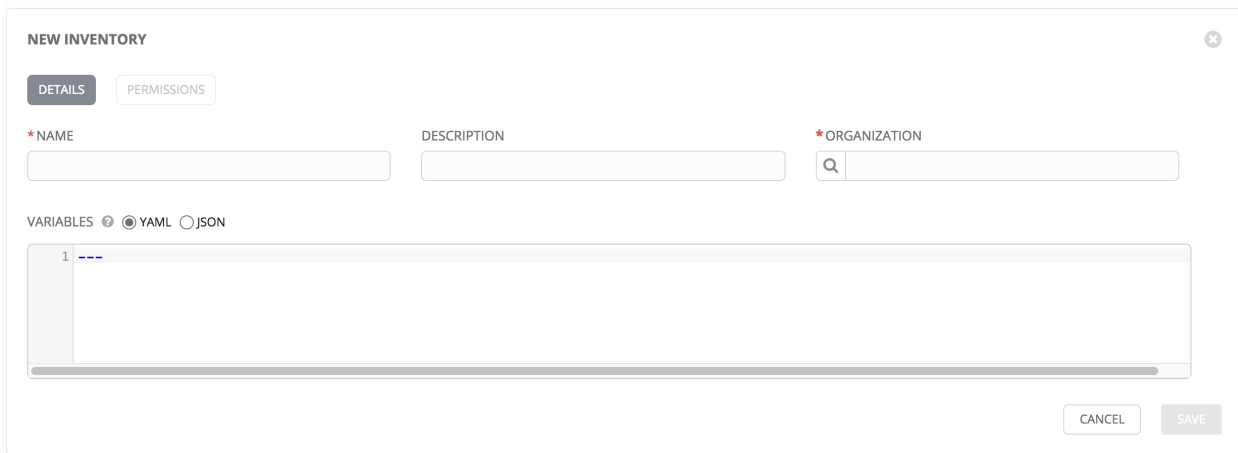
- **Inventory Sync:** If configured, you can “Sync” to fetch the project details from the configured source control.
- **Status Dot:** This shows the status of recent jobs for this inventory.
- **Name:** The inventory name. Clicking the Inventory name navigates to the properties screen for the selected inventory, which shows the inventory’s groups and hosts. (This view is also accessible from the **Action** menu.)
- **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
  - **Delete:** Delete the selected inventory. *This operation cannot be reversed!*

### 12.1 Add a new inventory

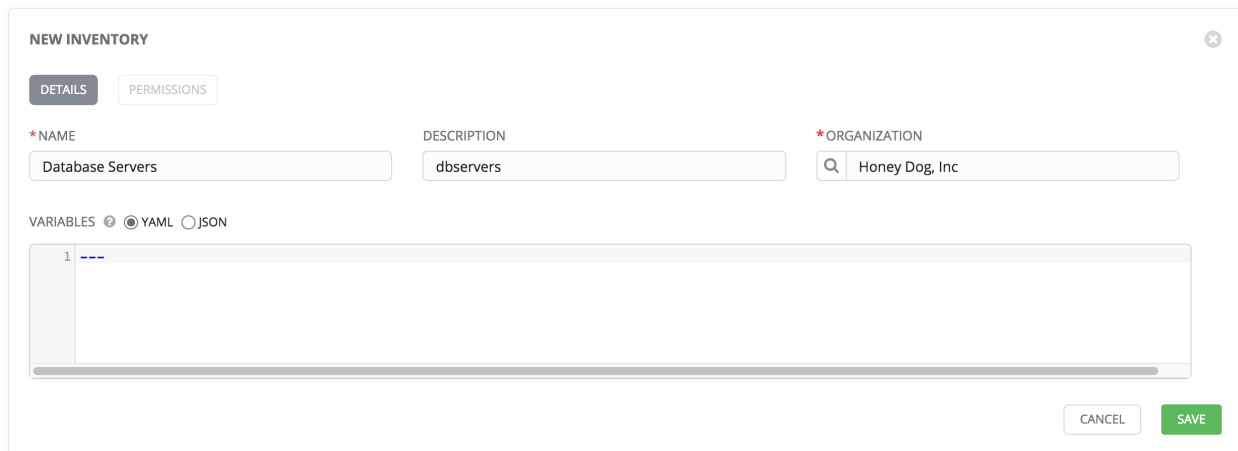
To create a new inventory:



1. Click the  button, which opens the **Create Inventory** window.



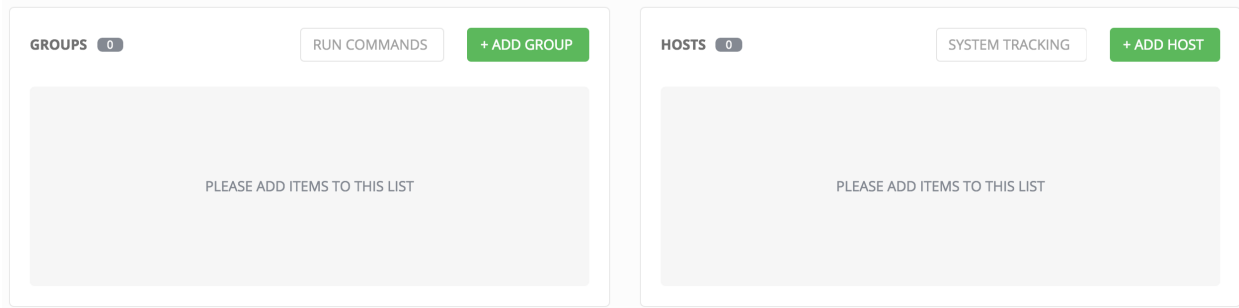
2. Enter the appropriate details into the following fields:
  - **Name:** Enter a name appropriate for this inventory.
  - **Description:** Enter an arbitrary description as appropriate (optional).
  - **Organization:** Choose among the available organizations.
  - **Variables:** Variable definitions 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 clicking save, the Groups and Hosts Management screen appears.

INVENTORIES / DATABASE SERVERS



## 12.2 Groups and Hosts

### Topics:

- *Groups*
  - *Add a new group*
  - *Credential Sources*
    - \* *Rackspace Cloud Servers*
    - \* *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*
    - \* *Custom Script*
  - *Scheduling*
    - \* *Add a new schedule*
- *Hosts*
  - *Add a new host*

Inventories are divided into groups, which may contain hosts and other groups, and hosts. There are several actions available for 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

## 12.2.1 Groups

Under Groups, you can view which groups belong to this inventory, easily filtered or searched by group name.

Additional actions may be performed on the group by selecting the buttons to the right of the group name:

- **Sync status:** Show the status of inventory synchronization for groups configured with cloud sources. If synchronization is configured, clicking this button shows the synchronization log for the selected group.
- **Host status:** Show the status of successful and failed jobs for the selected group. Clicking this button shows the list of hosts that are members of the selected group.
- **Start sync process:** Initiate a synchronization of the group with the configured cloud source. (A synchronization process that is in progress may be canceled by clicking the cancel button that appears here during synchronization.)
- **Edit Group:** Edit the properties for the selected group
- **Copy Group:** Groups can be nested. This allows you to copy or move the group to a different group.
- **Delete:** Delete the selected group. *This operation cannot be reversed!*

### Add a new group

To create a new group for an inventory:

**+ ADD GROUP**

1. Click the **+ ADD GROUP** button, which opens 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)

- **Source:** Choose a source which matches the credential type against which a host can be entered.
- **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.

In prior versions of Ansible Tower, the **Source** default was manual, meaning that the hosts must be entered into Tower manually. Beginning with Ansible Tower 3.0, host can be added via multiple methods/credential sources. (Refer to [Add a new host](#) for more information on managing hosts individually.)

3. To synchronize the inventory group from a cloud source, choose the appropriate source from the **Source** menu.

---

**Note:** Starting with version 2.2, Ansible Tower supports Amazon Web Services EC2, Rackspace Cloud Servers, Google Compute Engine, VMware vCenter, Microsoft Azure, OpenStack, and custom scripts added by the administrator. With Ansible Tower version 3.0, Microsoft Azure Classic (deprecated) and Microsoft Azure Resource Manager were added to expand upon the support offered for Microsoft Azure, and support for Red Hat Satellite 6 and RH Cloudforms credentials were also added.

---

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

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

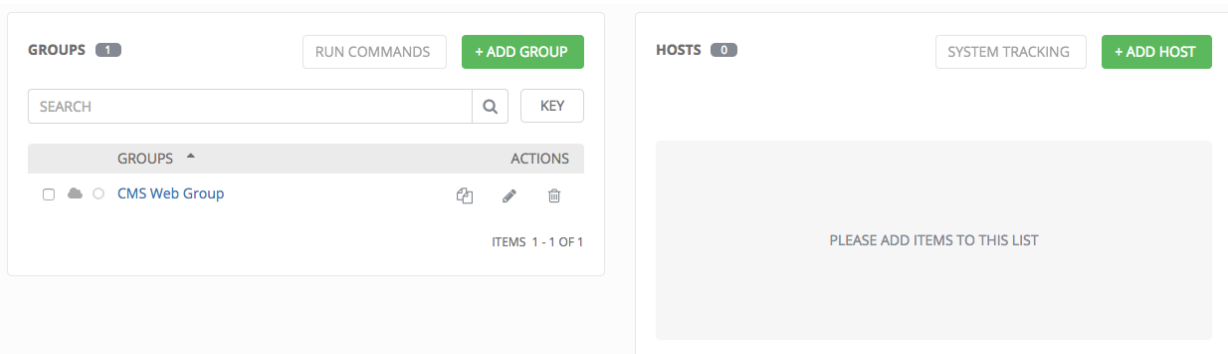
The “Update on Launch” setting refers to a dependency system for projects and inventory, and it will not specifically exclude two jobs from running at the same time. If a cache timeout is specified, then the dependencies for the second job is created and it uses the project and inventory update that the first job spawned. Both jobs then wait for that project and/or inventory update to finish before proceeding. If they are different job templates, they can then both start and run at the same time, if the system has the capacity to do so.

---

**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.

---

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



## Credential Sources

### Topics:

- *Rackspace Cloud Servers*
- *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*
- *Custom Script*

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

## Rackspace Cloud Servers

---

**Note:** Rackspace inventory sync has been deprecated in Tower 3.1.0 and support for Rackspace will be removed in a future release.

---

To configure a group for Rackspace, select **Rackspace Cloud Servers** and enter the following details:

- **Cloud Credential:** 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.
- **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.

You can also configure **Update Options**.

- **Overwrite:** If checked, all child groups and hosts not found on the external source are deleted from the local inventory. When not checked, local child hosts and groups not found on the external source remain untouched by the inventory update process.
- **Overwrite Variables:** If checked, all variables for child groups and hosts are removed and replaced by those found on the external source. When not checked, a merge is performed, combining local variables with those found on the external source.
- **Update on Launch:** Each time a job runs using this inventory, refresh the inventory from the selected source before executing job tasks.

**CREATE GROUP** ✕

**DETAILS** NOTIFICATIONS

<p><b>*NAME</b></p> <input style="width: 95%;" type="text"/>	<p>DESCRIPTION</p> <input style="width: 95%;" type="text"/>	<p>SOURCE</p> <input style="width: 95%;" type="text" value="Rackspace Cloud Servers"/>
<p><b>*CLOUD CREDENTIAL</b></p> <input style="width: 95%;" type="text"/>	<p>REGIONS <span style="font-size: 0.8em;">?</span></p> <input style="width: 95%;" type="text"/>	<p>UPDATE OPTIONS</p> <p><input type="checkbox"/> Overwrite <span style="font-size: 0.8em;">?</span></p> <p><input type="checkbox"/> Overwrite Variables <span style="font-size: 0.8em;">?</span></p> <p><input type="checkbox"/> Update on Launch <span style="font-size: 0.8em;">?</span></p>

VARIABLES ?  YAML  JSON

1 ---

CANCEL
SAVE

## Amazon Web Services EC2

To configure a group for AWS, select **Amazon EC2** and enter the following details:

**CREATE GROUP** ✕

**DETAILS** NOTIFICATIONS

<p><b>*NAME</b></p> <input style="width: 95%;" type="text" value="CMS Web Group - West"/>	<p>DESCRIPTION</p> <input style="width: 95%;" type="text" value="CMS Web West Coast"/>	<p>SOURCE</p> <input style="width: 95%;" type="text" value="Amazon EC2"/>
<p><b>*CLOUD CREDENTIAL</b></p> <input style="width: 95%;" type="text"/>	<p>REGIONS <span style="font-size: 0.8em;">?</span></p> <input style="width: 95%;" type="text"/>	<p>INSTANCE FILTERS <span style="font-size: 0.8em;">?</span></p> <input style="width: 95%;" type="text"/>

ONLY GROUP BY ?

UPDATE OPTIONS

Overwrite ?

Overwrite Variables ?

Update on Launch ?

VARIABLES ?  YAML  JSON

1 ---

SOURCE VARIABLES ?  YAML  JSON

1 ---

CANCEL
SAVE

- **Cloud 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 Type
  - Key Name
  - Region
  - Security Group
  - Tags (by name)
  - VPC ID

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.

- **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.
- **Source Variables:** 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](#).

You can also configure **Update Options**.

- **Overwrite:** If checked, all child groups and hosts not found on the external source are deleted from the local inventory. When not checked, local child hosts and groups not found on the external source remain untouched by the inventory update process.
- **Overwrite Variables:** If checked, all variables for child groups and hosts are removed and replaced by those found on the external source. When not checked, a merge is performed, combining local variables with those found on the external source.
- **Update on Launch:** Each time a job runs using this inventory, refresh the inventory from the selected source before executing job tasks.

## Google Compute Engine

To configure a group for Google Compute Engine, select **Google Compute Engine** and enter the following details:

- **Cloud Credential:** 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.
- **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.

You can also configure **Update Options**.

- **Overwrite:** If checked, all child groups and hosts not found on the external source are deleted from the local inventory. When not checked, local child hosts and groups not found on the external source remain untouched by the inventory update process.
- **Overwrite Variables:** If checked, all variables for child groups and hosts are removed and replaced by those found on the external source. When not checked, a merge is performed, combining local variables with those found on the external source.
- **Update on Launch:** Each time a job runs using this inventory, refresh the inventory from the selected source before executing job tasks.

The screenshot shows the 'CREATE GROUP' interface for Google Compute Engine. It features a 'DETAILS' tab and a 'NOTIFICATIONS' tab. The form includes the following fields and options:

- \* NAME:** CMS Web Group - West
- DESCRIPTION:** CMS Web West Coast
- SOURCE:** Google Compute Engine (dropdown menu)
- \* CLOUD CREDENTIAL:** Searchable input field
- REGIONS:** Searchable input field
- UPDATE OPTIONS:**
  - Overwrite
  - Overwrite Variables
  - Update on Launch
- VARIABLES:** Radio buttons for  YAML and  JSON, followed by a large text area for variable definitions.

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

## Microsoft Azure Classic (deprecated)

To configure a group for Microsoft Azure Classic, select **Microsoft Azure Classic (deprecated)** and enter the following details:

- **Cloud Credential:** 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.
- **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.



You can also configure **Update Options**.

- **Overwrite:** If checked, all child groups and hosts not found on the external source are deleted from the local inventory. When not checked, local child hosts and groups not found on the external source remain untouched by the inventory update process.
- **Overwrite Variables:** If checked, all variables for child groups and hosts are removed and replaced by those found on the external source. When not checked, a merge is performed, combining local variables with those found on the external source.
- **Update on Launch:** Each time a job runs using this inventory, refresh the inventory from the selected source before executing job tasks.

## Microsoft Azure Resource Manager

To configure a group for Microsoft Azure Resource Manager, select **Microsoft Azure Resource Manager** and enter the following details:

- **Cloud Credential:** 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.
- **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.

You can also configure **Update Options**.

- **Overwrite:** If checked, all child groups and hosts not found on the external source are deleted from the local inventory. When not checked, local child hosts and groups not found on the external source remain untouched by the inventory update process.
- **Overwrite Variables:** If checked, all variables for child groups and hosts are removed and replaced by those found on the external source. When not checked, a merge is performed, combining local variables with those found on the external source.
- **Update on Launch:** Each time a job runs using this inventory, refresh the inventory from the selected source before executing job tasks.

**CREATE GROUP**

DETAILS NOTIFICATIONS

\*NAME: CMS Web Group - West

DESCRIPTION: CMS Web West Coast

SOURCE: Microsoft Azure Resource Manager

\*CLOUD CREDENTIAL: [Search]

REGIONS: [?]

UPDATE OPTIONS:
 

- Overwrite [?]
- Overwrite Variables [?]
- Update on Launch [?]

VARIABLES [?]  YAML  JSON

1 ---

CANCEL SAVE

## VMware vCenter

To configure a group for VMware vCenter, select **VMware** and enter the following details:

- **Cloud Credential:** Choose from an existing Credential. For more information, refer to [Credentials](#).
- **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.
- **Source Variables:** Override variables found in `vmware.ini` and used by the inventory update script. For a detailed description of these variables [view vmware.ini in the Ansible GitHub repo](#). Enter variables using either JSON or YAML syntax. Use the radio button to toggle between the two.

You can also configure **Update Options**.

- **Overwrite:** If checked, all child groups and hosts not found on the external source are deleted from the local inventory. When not checked, local child hosts and groups not found on the external source remain untouched by the inventory update process.
- **Overwrite Variables:** If checked, all variables for child groups and hosts are removed and replaced by those found on the external source. When not checked, a merge is performed, combining local variables with those found on the external source.
- **Update on Launch:** Each time a job runs using this inventory, refresh the inventory from the selected source before executing job tasks.

**CREATE GROUP** ✕

DETAILS NOTIFICATIONS

\*NAME:  DESCRIPTION:  SOURCE:

\*CLOUD CREDENTIAL:  UPDATE OPTIONS:

Overwrite [?](#)  
 Overwrite Variables [?](#)  
 Update on Launch [?](#)

VARIABLES [?](#)  YAML  JSON

1 ---

SOURCE VARIABLES [?](#)  YAML  JSON

1 ---

CANCEL

## Red Hat Satellite 6

To configure a group for Red Hat Satellite 6, select **Red Hat Satellite 6** and enter the following details:

- **Cloud Credential:** Choose from an existing Credential. For more information, refer to [Credentials](#).
- **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.

You can also configure **Update Options**.

- **Overwrite:** If checked, all child groups and hosts not found on the external source are deleted from the local inventory. When not checked, local child hosts and groups not found on the external source remain untouched by the inventory update process.
- **Overwrite Variables:** If checked, all variables for child groups and hosts are removed and replaced by those found on the external source. When not checked, a merge is performed, combining local variables with those found on the external source.
- **Update on Launch:** Each time a job runs using this inventory, refresh the inventory from the selected source before executing job tasks.

## Red Hat CloudForms

To configure a group for Red Hat CloudForms, select **Red Hat CloudForms** and enter the following details:

- **Cloud Credential:** Choose from an existing Credential. For more information, refer to [Credentials](#).
- **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.

You can also configure **Update Options**.

- **Overwrite:** If checked, all child groups and hosts not found on the external source are deleted from the local inventory. When not checked, local child hosts and groups not found on the external source remain untouched by the inventory update process.
- **Overwrite Variables:** If checked, all variables for child groups and hosts are removed and replaced by those found on the external source. When not checked, a merge is performed, combining local variables with those found on the external source.
- **Update on Launch:** Each time a job runs using this inventory, refresh the inventory from the selected source before executing job tasks.

**CREATE GROUP** ✕

**DETAILS** **NOTIFICATIONS**

\*NAME:  DESCRIPTION:  SOURCE:

\*CLOUD CREDENTIAL:  UPDATE OPTIONS:

Overwrite ?

Overwrite Variables ?

Update on Launch ?

VARIABLES ?  YAML  JSON

1 ---

## OpenStack

To configure a group for OpenStack, select **OpenStack** and enter the following details:

- **Cloud Credential:** Choose from an existing Credential. For more information, refer to [Credentials](#).
- **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.
- **Source Variables:** Override variables found in `openstack.yml` and used by the inventory update script. For a detailed description of these variables [view openstack.yml in the Ansible GitHub repo](#). Enter variables using either JSON or YAML syntax. Use the radio button to toggle between the two.

You can also configure **Update Options**.

- **Overwrite:** If checked, all child groups and hosts not found on the external source are deleted from the local inventory. When not checked, local child hosts and groups not found on the external source remain untouched by the inventory update process.
- **Overwrite Variables:** If checked, all variables for child groups and hosts are removed and replaced by those found on the external source. When not checked, a merge is performed, combining local variables with those found on the external source.
- **Update on Launch:** Each time a job runs using this inventory, refresh the inventory from the selected source before executing job tasks.

**CREATE GROUP**

DETAILS NOTIFICATIONS

\*NAME: CMS Web Group - West      DESCRIPTION: CMS Web West Coast      SOURCE: OpenStack

\*CLOUD CREDENTIAL:       UPDATE OPTIONS:  Overwrite  Overwrite Variables  Update on Launch

VARIABLES  YAML  JSON

SOURCE VARIABLES  YAML  JSON

CANCEL SAVE

## Custom Script

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

To configure a group to use a Custom Inventory Script, select **Custom Script** and enter the following details:

- **Custom Inventory Script:** Choose from an existing Inventory Script.
- **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.

You can also configure **Update Options**.

- **Overwrite:** If checked, all child groups and hosts not found on the external source are deleted from the local inventory. When not checked, local child hosts and groups not found on the external source remain untouched by the inventory update process.
- **Overwrite Variables:** If checked, all variables for child groups and hosts are removed and replaced by those found on the external source. When not checked, a merge is performed, combining local variables with those found on the external source.
- **Update on Launch:** Each time a job runs using this inventory, refresh the inventory from the selected source before executing job tasks.

**NEW CUSTOM INVENTORY**

\*NAME:  DESCRIPTION:  \*ORGANIZATION:

\*CUSTOM SCRIPT

```
#!/usr/bin/env python

# Python
import json
import optparse
import os

nhosts = int(os.environ.get('NHOSTS', 100))

inv_list = {
```

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

## Scheduling

For groups sourced from a cloud service, the inventory update process may be scheduled via the **Schedule** view. To

access the **Schedule** view, click the Schedule (  ) button beside the Inventory Group name to open the **Edit Group** dialog.

**GROUPS** 1 RUN COMMANDS

NAME

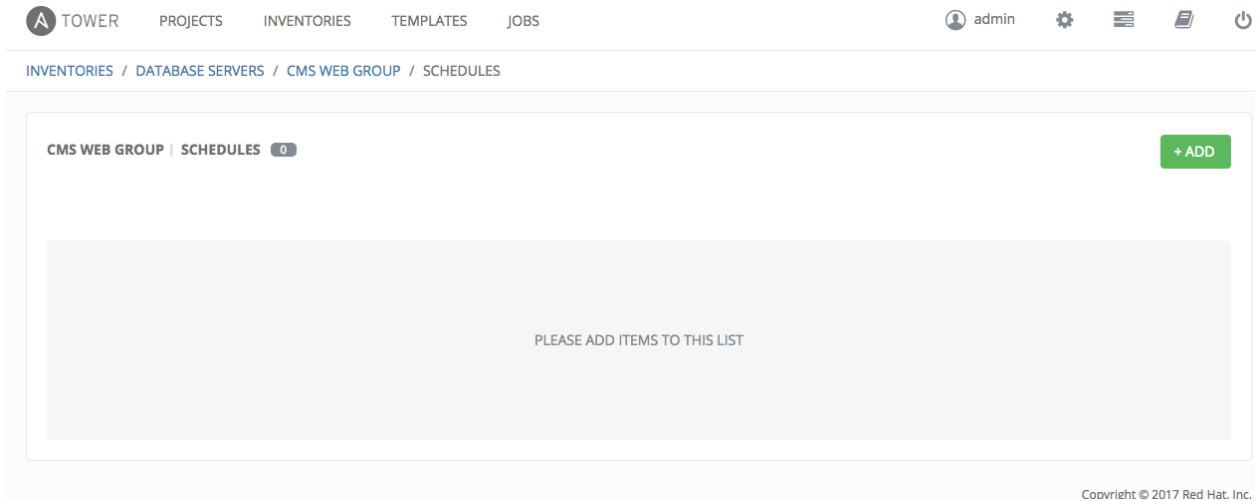
GROUPS	ACTIONS
<input type="checkbox"/> CMS Web Group West	

ITEMS 1-1 OF 1

**HOSTS** 0

PLEASE ADD ITEMS TO THIS LIST

This screen displays a list of the schedules that are currently available for the selected **Group**. 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
- Next Run

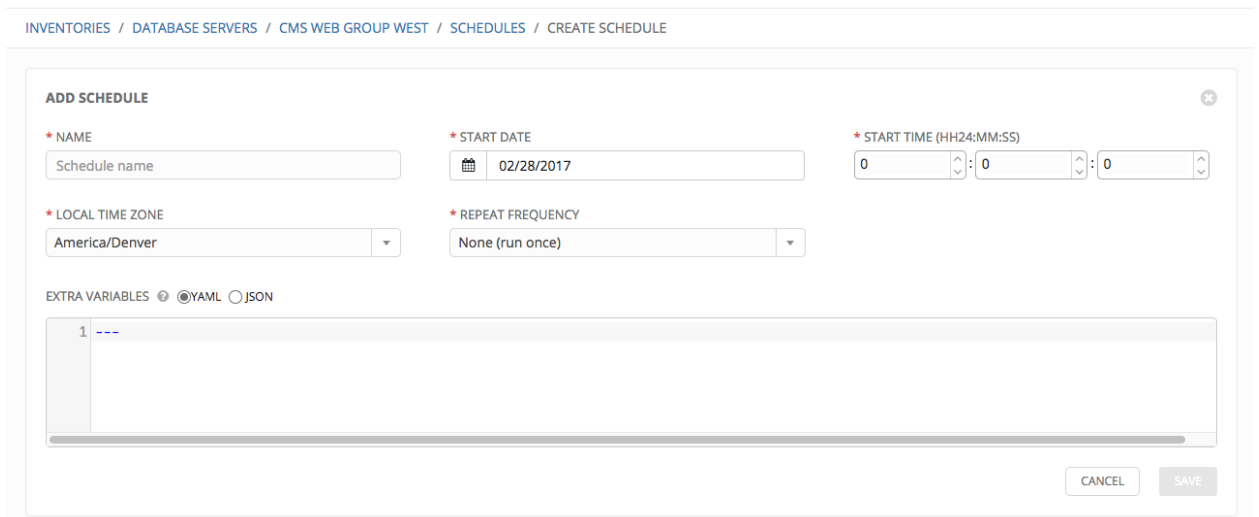
Buttons located in the upper right corner of the **Schedules** screen provide the following actions:

- Create a new schedule
- Refresh this view

### Add a new schedule

To create a new schedule:

1. Click the  button.



2. Enter the appropriate details into the following fields and select Save:



- **Name** (required)
- **Start Date** (required)
- **Start Time** (required)
- **Local Time Zone** (the entered Start Time should be in this timezone)
- **Repeat Frequency** (the appropriate options are displayed as the update frequency is modified)
  - None

QUICK SYNC +

\* NAME

\* START DATE (MM/DD/YYYY)

\* START TIME (HH24:MM:SS)

\* LOCAL TIME ZONE

\* REPEAT FREQUENCY

SCHEDULE DESCRIPTION

every day for 1 time

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

06/30/2016 00:00:00 EDT

- Minute, Hour, or Day

DAILY SYNC +

\* NAME

\* START DATE (MM/DD/YYYY)

\* START TIME (HH24:MM:SS)

\* LOCAL TIME ZONE

\* REPEAT FREQUENCY

FREQUENCY DETAILS

\* EVERY  DAYS

\* END

\* OCCURRENCE(S)

SCHEDULE DESCRIPTION

every day for 5 times

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

06/30/2016 00:00:00 EDT  
 07/01/2016 00:00:00 EDT  
 07/02/2016 00:00:00 EDT  
 07/03/2016 00:00:00 EDT  
 07/04/2016 00:00:00 EDT

- Week

**BI-WEEKLY SYNC**

\* NAME: Bi-Weekly Sync

\* START DATE (MM/DD/YYYY): 06/30/2016

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

\* LOCAL TIME ZONE: America/New\_York

\* REPEAT FREQUENCY: Week

**FREQUENCY DETAILS**

\* EVERY: 2 WEEKS

\* ON DAYS: SUN MON TUE **WED** THU FRI SAT

\* END: On Date

\* END DATE (MM/DD/YYYY): 08/26/2016

\* END TIME (HH24:MM:SS): 00:00:00

**SCHEDULE DESCRIPTION**

every 2 weeks on Monday, Wednesday, Thursday, Sunday until August 25, 2016

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

```
06/30/2016 00:00:00 EDT
07/03/2016 00:00:00 EDT
07/11/2016 00:00:00 EDT
07/13/2016 00:00:00 EDT
07/14/2016 00:00:00 EDT
07/17/2016 00:00:00 EDT
07/25/2016 00:00:00 EDT
07/27/2016 00:00:00 EDT
07/28/2016 00:00:00 EDT
07/31/2016 00:00:00 EDT
```

- Month

**MONTHLY SYNC**

\* NAME: Monthly Sync

\* START DATE (MM/DD/YYYY): 06/30/2016

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

\* LOCAL TIME ZONE: America/New\_York

\* REPEAT FREQUENCY: Month

**FREQUENCY DETAILS**

\* EVERY: 1 MONTHS

\* ON DAY: 1

\* ON THE: first Sunday

\* END: After

\* OCCURRENCE(S): 3

**SCHEDULE DESCRIPTION**

every month on Sunday for 3 times (~ approximate)

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

```
07/03/2016 00:00:00 EDT
08/07/2016 00:00:00 EDT
09/04/2016 00:00:00 EDT
```

- Year

YEARLY SYNC ✕

\* NAME  \* START DATE (MM/DD/YYYY)  \* START TIME (HH24:MM:SS)  :  :

\* LOCAL TIME ZONE  \* REPEAT FREQUENCY

FREQUENCY DETAILS

\* EVERY  YEARS  \* ON    \* ON THE

\* END  \* OCCURRENCE(S)

**SCHEDULE DESCRIPTION**


every January on the 1st for 4 times

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

01/01/2017 00:00:00 EST  
 01/01/2018 00:00:00 EST  
 01/01/2019 00:00:00 EST  
 01/01/2020 00:00:00 EST

The **Schedule Description** panel displays an detailed overview of the schedule and a list of the scheduled occurrences in the selected Local Time Zone or in UTC (be sure to select **Local Time** or **UTC** as based on your needs).

**Note:** When using UTC time settings, repeating jobs that runs at a specific time of day may move relative to a local timezone when Daylight Saving Time shifts occur.

Once you have saved the schedule, it can be viewed by clicking on the Schedule (  ) tab beside the group name.

TOWER PROJECTS INVENTORIES TEMPLATES JOBS
admin ⚙️ ☰ 📄 🔌

INVENTORIES / DATABASE SERVERS / CMS WEB GROUP WEST / SCHEDULES

CMS WEB GROUP WEST | SCHEDULES 1
+ ADD

NAME ^	FIRST RUN ↓	NEXT RUN ↓	FINAL RUN ↓	ACTIONS
<input checked="" type="checkbox"/> Weekly Sync	3/29/2017 11:30:00 PM	3/29/2017 11:30:00 PM	6/15/2017 9:30:01 PM	✎️ 🗑️

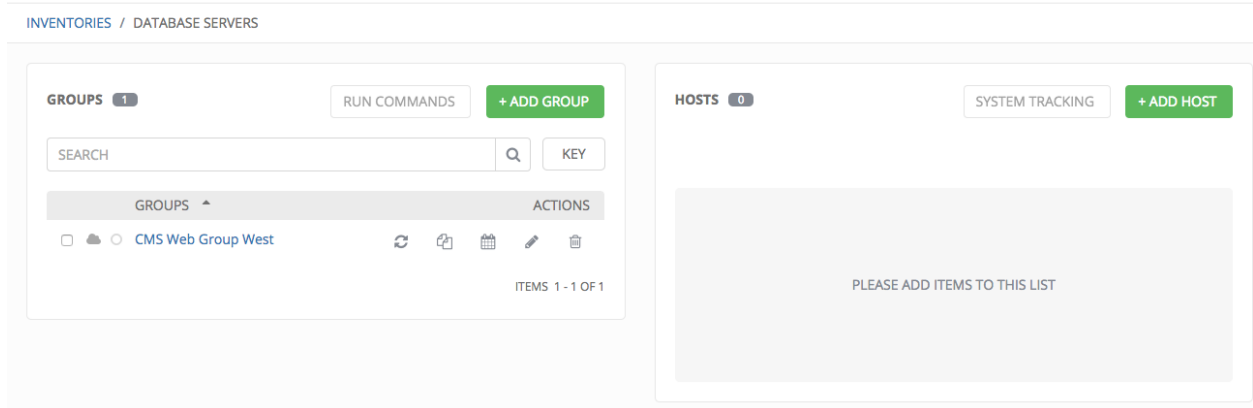
ITEMS 1 - 1 OF 1

There are server actions available for schedules:

- ON/OFF – Stop an active schedule or activate a stopped schedule by using the toggle button.
- Edit schedule
- Delete schedule

## 12.2.2 Hosts

Hosts are listed on the right side of the Inventory display screen.



The host list may be sorted and searched by **Name** or **Groups**, and filtered by hosts that are disabled, by hosts with failed jobs, and by hosts synchronized with an external source.

This list displays information about each host and provides for several actions:

- **ON/OFF** Indicates whether a host is available and should be included in running jobs. For hosts that are part of an external inventory, this flag is set by the inventory sync process and cannot be manually changed.
- **Name**: Opens the **Host Properties** dialog
- **Available**: A toggle indicating whether the host is enabled to receive jobs from Tower. Click to toggle this setting.
- **Jobs**: Shows the most recent Jobs run against this Host. Clicking this button displays a window showing the most recent jobs and their status.
- **Edit host**: Opens the **Host Properties** dialog
- **Copy host**: Copies or moves the host to a different group
- **Delete**: Removes the host from Tower. *This operation is not reversible!*

### Add a new host

Hosts can be added manually, by IP address, or hostname. Tower can also sync inventory directly from AWS EC2, Google Compute Engine, MS Azure, VMware, Rackspace Open Cloud, or OpenStack.

To create a new host and add it to an existing group:

1. Click the  button.

This opens the **Create Host** dialog.

2. Enter the appropriate details into the following fields:

- **Host Name:** The hostname or IP address of the host
- **Description:** Enter an arbitrary description as appropriate
- **Enabled?:** Indicates if a host is available and should be included in running jobs. For hosts that are part of an external inventory, this flag cannot be changed. It is set by the inventory sync process.
- **Variables:** Variable definitions and values to be applied to the selected host. Enter variables using either JSON or YAML syntax, using the radio button to toggle between JSON or YAML.

3. Click **Save** when done.

## 12.3 Running Ad Hoc Commands

To run an ad hoc command:

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

Then, click the  button.

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.
- **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.
- **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.



Click the button to run this ad hoc command.

**EXECUTE COMMAND**

\*MODULE: ping

ARGUMENTS: [ ]

LIMIT: CMS Web Group

\*MACHINE CREDENTIAL: Demo Credential

ENABLE PRIVILEGE ESCALATION

\*VERBOSITY: 0 (Normal)

\*FORKS: 0

RESET LAUNCH

**GROUPS** 1

RUN COMMANDS + ADD GROUP

SEARCH [ ] Q KEY

GROUPS	ACTIONS
<input checked="" type="checkbox"/> CMS Web Group	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>

ITEMS 1 - 1 OF 1

**HOSTS** 1

SYSTEM TRACKING + ADD HOST

SEARCH [ ] Q KEY

HOSTS	ACTIONS
<input type="checkbox"/> 127.0.0.1	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>

ITEMS 1 - 1 OF 1

## 12.4 System Tracking

**Note:** System Tracking, introduced as a new feature in Ansible Tower 2.2, is only available to those with Enterprise-level licenses.

System Tracking offers the ability to compare the results of two scan runs from different dates on one host or the same date on two hosts.

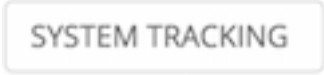
Data is grouped by fact modules:

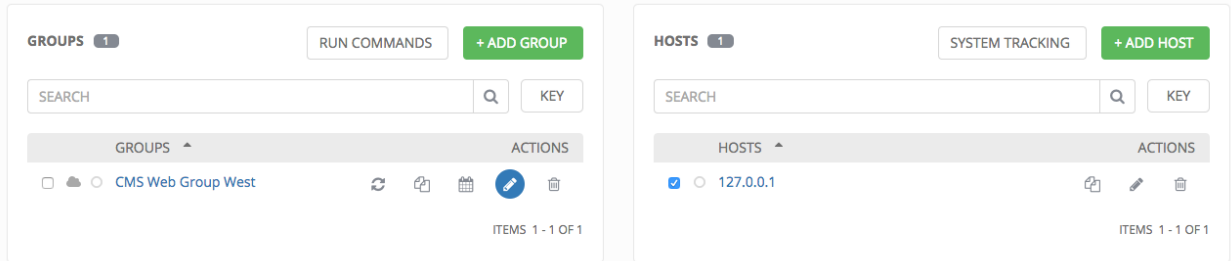
- Packages
- Services
- Files
- Ansible
- Custom

Tower is designed to make every attempt to find your data. If you select a date without any scan runs, Tower gathers the previous year's worth of scan runs to verify possible data to include. Successful comparisons display results from the available dates instead of the specified dates. Unsuccessful comparisons display a message indicating why they did not work.

**Note:** Service scan jobs should not run against an inventory with hosts that point to the same physical machine.

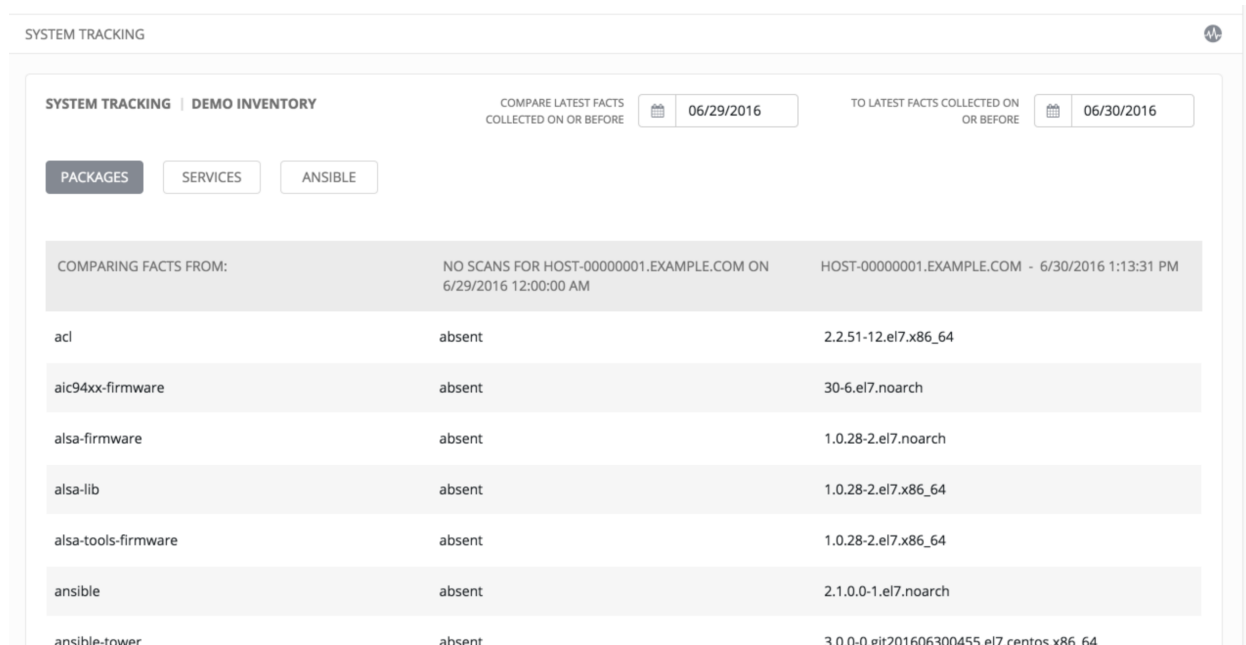
## 12.4.1 Single Host Workflow

Select a single host in an inventory to compare against two dates and click the  button.



**Note:** If you have not already created a job template set to scan, you will not be able to proceed until the correct job template has been created.

Select two dates on which you have scan data for the host, with the earliest date to compare on the left and the latest date to compare on the right.



Select the module (**Packages**, **Services**, or **Ansible**) for which you want to compare differences. To change modules, click on the module button with the button navigation to filter by different types of facts. Note that differences among the “ansible” and “files” modules changes are highlighted in red, while only changes for “packages” and “services” are shown.

You may also choose the same date in both date selectors if you want to compare multiple scan runs against a single date. If two or more scan jobs runs are discovered on a particular day, Tower compares the most recent and the second-most recent. If there is only one run for the selected date, Tower may display a message saying it could not find any scan job runs in one of the columns. (Also noted in [Known Issues](#) in the *Ansible Tower Release Notes* manual.)



SYSTEM TRACKING   DEMO INVENTORY		COMPARE LATEST FACTS COLLECTED ON OR BEFORE	TO LATEST FACTS COLLECTED ON OR BEFORE
		06/30/2016	06/30/2016
<div style="display: flex; justify-content: space-between;"> <span>PACKAGES</span> <span>SERVICES</span> <span style="background-color: #333; color: white; padding: 2px 5px;">ANSIBLE</span> </div>			
COMPARING FACTS FROM:		HOST-00000001.EXAMPLE.COM - 6/30/2016 1:13:31 PM	HOST-00000001.EXAMPLE.COM - 6/30/2016 1:20:11 PM
ansible_product_serial	NA	NA	NA
ansible_form_factor	Other	Other	Other
ansible_product_version	1.2	1.2	1.2
ansible_fips	false	false	false
ansible_service_mgr	systemd	systemd	systemd
ansible_user_id	awx	awx	awx
ansible_user_dir	/var/lib/awx	/var/lib/awx	/var/lib/awx
ansible_userspace_bits	64	64	64
ansible_ssh_host_key_rsa_public	AAAAB3NzaC1yc2EAAAADAQABAAQDU0nk47tYOcejwWgW6/LaRA3Xwfo2j6U+h5G49o6kUwQIKAcTGHAEuKYmhIRFGFwaEzpEI5Q3lynFOP1VskyGHdTq9i0BSDvbar2xM6Me/p0MNxKaYaloiH6X1t6DI4aQeO3RYHncPh7dm/W2KHs79hRhqaqucY24YmgJhQ32TeureeHq6Pqj8DIOFHH+r6UgwZS2Z2+x5mGja427yMLqnD82cz3aFRu3pcErdfxWj181q+Dg2KE1K7wjpUHEFeioaSreJxVDJ4clUQyb6MMDQkvNwORFcl7K48LejOSrcldlMrGisPF4cnn1iK74Bip1yMeCoWe1JQhAOPGPfZT	AAAAB3NzaC1yc2EAAAADAQABAAQDU0nk47tYOcejwWgW6/LaRA3Xwfo2j6U+h5G49o6kUwQIKAcTGHAEuKYmhIRFGFwaEzpEI5Q3lynFOP1VskyGHdTq9i0BSDvbar2xM6Me/p0MNxKaYaloiH6X1t6DI4aQeO3RYHncPh7dm/W2KHs79hRhqaqucY24YmgJhQ32TeureeHq6Pqj8DIOFHH+r6UgwZS2Z2+x5mGja427yMLqnD82cz3aFRu3pcErdfxWj181q+Dg2KE1K7wjpUHEFeioaSreJxVDJ4clUQyb6MMDQkvNwORFcl7K48LejOSrcldlMrGisPF4cnn1iK74Bip1yMeCoWe1JQhAOPGPfZT	AAAAB3NzaC1yc2EAAAADAQABAAQDU0nk47tYOcejwWgW6/LaRA3Xwfo2j6U+h5G49o6kUwQIKAcTGHAEuKYmhIRFGFwaEzpEI5Q3lynFOP1VskyGHdTq9i0BSDvbar2xM6Me/p0MNxKaYaloiH6X1t6DI4aQeO3RYHncPh7dm/W2KHs79hRhqaqucY24YmgJhQ32TeureeHq6Pqj8DIOFHH+r6UgwZS2Z2+x5mGja427yMLqnD82cz3aFRu3pcErdfxWj181q+Dg2KE1K7wjpUHEFeioaSreJxVDJ4clUQyb6MMDQkvNwORFcl7K48LejOSrcldlMrGisPF4cnn1iK74Bip1yMeCoWe1JQhAOPGPfZT
ansible_ssh_host_key_ecdsa_public	AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBcSd4s1dE22hUtoxQmF9N+ZQFcxllhRQIYSAfTC2Nlc+Q8lok/oKgphe9coUyUD3lWF9M9TZar7JFi87pGcj4=	AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBcSd4s1dE22hUtoxQmF9N+ZQFcxllhRQIYSAfTC2Nlc+Q8lok/oKgphe9coUyUD3lWF9M9TZar7JFi87pGcj4=	AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBcSd4s1dE22hUtoxQmF9N+ZQFcxllhRQIYSAfTC2Nlc+Q8lok/oKgphe9coUyUD3lWF9M9TZar7JFi87pGcj4=
ansible_distribution_version	7.2.1511	7.2.1511	7.2.1511
ansible_domain	localdomain	localdomain	localdomain
ansible_ssh_host_key_ed25519_public	AAAC3NzaC1lZDI1NTE5AAAAIQAqf3xMyHifXsRZUN680fFOMcmYYRD3h1FXXn1joY	AAAC3NzaC1lZDI1NTE5AAAAIQAqf3xMyHifXsRZUN680fFOMcmYYRD3h1FXXn1joY	AAAC3NzaC1lZDI1NTE5AAAAIQAqf3xMyHifXsRZUN680fFOMcmYYRD3h1FXXn1joY
ansible_processor_cores	2	2	2
ansible_virtualization_role	guest	guest	guest
ansible_processor_vcpus	2	2	2
ansible_system_capabilities_enforced	True	True	True
ansible_bios_version	VirtualBox	VirtualBox	VirtualBox
ansible_virtualization_type	virtualbox	virtualbox	virtualbox
ansible_memtotal_mb	1840	1840	1840
ansible_architecture	x86_64	x86_64	x86_64
ansible_swapfree_mb	2047	1691	1691
ansible_system_vendor	innotek GmbH	innotek GmbH	innotek GmbH

Please note that if the scans found for the selected date are identical, Tower displays a single result of all facts scanned.

As an example, say that a user selects “7/7/2015” for both dates and selects the “packages” module. And say that two runs occurred on this date, but there were no changes to packages on the selected host. The user sees a message indicating the scans were identical as well as a single column containing all package versions, instead of a two-column listing of differences.

SYSTEM TRACKING | DEMO INVENTORY

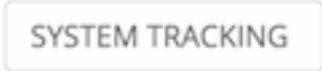
COMPARE LATEST FACTS COLLECTED ON OR BEFORE  TO LATEST FACTS COLLECTED ON OR BEFORE

**PACKAGES** SERVICES ANSIBLE

THE SELECTED FACT SCANS WERE IDENTICAL FOR THIS MODULE. SHOWING ALL FACTS FROM THE LATEST SELECTED SCAN INSTEAD.

COMPARING FACTS FROM:	HOST-00000001.EXAMPLE.COM - 6/30/2016 1:20:11 PM
acl	2.2.51-12.el7.x86_64
aic94xx-firmware	30-6.el7.noarch
alsa-firmware	1.0.28-2.el7.noarch
alsa-lib	1.0.28-2.el7.x86_64

### 12.4.2 Host to Host Workflow

To compare two hosts, select the hosts and click the  button.

INVENTORIES / WEB SERVERS

**GROUPS** 1 RUN COMMANDS + ADD GROUP

SEARCH  Q KEY

GROUPS ^ ACTIONS

● CMS Web Group West 22 ↻ 📄 📅 ✎ 🗑️

ITEMS 1-1 OF 1

**HOSTS** 101 SYSTEM TRACKING + ADD HOST

SEARCH  Q KEY

HOSTS ^	ACTIONS
<input type="checkbox"/> 127.0.0.1	<span>📄</span> <span>✎</span> <span>🗑️</span>
<input checked="" type="checkbox"/> host-00000000.example.com	<span>📄</span> <span>✎</span> <span>🗑️</span>
<input checked="" type="checkbox"/> host-00000001.example.com	<span>📄</span> <span>✎</span> <span>🗑️</span>
<input type="checkbox"/> host-00000002.example.com	<span>📄</span> <span>✎</span> <span>🗑️</span>
<input type="checkbox"/> host-00000003.example.com	<span>📄</span> <span>✎</span> <span>🗑️</span>
<input type="checkbox"/> host-00000004.example.com	<span>📄</span> <span>✎</span> <span>🗑️</span>
<input type="checkbox"/> host-00000005.example.com	<span>📄</span> <span>✎</span> <span>🗑️</span>
<input type="checkbox"/> host-00000006.example.com	<span>📄</span> <span>✎</span> <span>🗑️</span>
<input type="checkbox"/> host-00000007.example.com	<span>📄</span> <span>✎</span> <span>🗑️</span>
<input type="checkbox"/> host-00000008.example.com	<span>📄</span> <span>✎</span> <span>🗑️</span>
<input type="checkbox"/> host-00000009.example.com	<span>📄</span> <span>✎</span> <span>🗑️</span>
<input type="checkbox"/> host-00000010.example.com	<span>📄</span> <span>✎</span> <span>🗑️</span>
<input type="checkbox"/> host-00000011.example.com	<span>📄</span> <span>✎</span> <span>🗑️</span>
<input type="checkbox"/> host-00000012.example.com	<span>📄</span> <span>✎</span> <span>🗑️</span>
<input type="checkbox"/> host-00000013.example.com	<span>📄</span> <span>✎</span> <span>🗑️</span>
<input type="checkbox"/> host-00000014.example.com	<span>📄</span> <span>✎</span> <span>🗑️</span>
<input type="checkbox"/> host-00000015.example.com	<span>📄</span> <span>✎</span> <span>🗑️</span>
<input type="checkbox"/> host-00000016.example.com	<span>📄</span> <span>✎</span> <span>🗑️</span>
<input type="checkbox"/> host-00000017.example.com	<span>📄</span> <span>✎</span> <span>🗑️</span>
<input type="checkbox"/> host-00000018.example.com	<span>📄</span> <span>✎</span> <span>🗑️</span>

< 1 2 3 4 5 6 >
PAGE 1 OF 6
ITEMS 1-20 OF 101

Copyright © 2016 Red Hat, Inc.

Select a single date on which to compare the two hosts. Next, select the module for which you want to view differences.

**GROUPS** 1 RUN COMMANDS + ADD GROUP

SEARCH  Q KEY

GROUPS ^ ACTIONS

● CMS Web Group West 22

ITEMS 1-1 OF 1

**HOSTS** 101 SYSTEM TRACKING + ADD HOST

SEARCH  Q KEY

HOSTS ^	ACTIONS
<input type="checkbox"/> 127.0.0.1	
<input checked="" type="checkbox"/> host-00000000.example.com	
<input checked="" type="checkbox"/> host-00000001.example.com	
<input type="checkbox"/> host-00000002.example.com	
<input type="checkbox"/> host-00000003.example.com	
<input type="checkbox"/> host-00000004.example.com	
<input type="checkbox"/> host-00000005.example.com	
<input type="checkbox"/> host-00000006.example.com	
<input type="checkbox"/> host-00000007.example.com	
<input type="checkbox"/> host-00000008.example.com	
<input type="checkbox"/> host-00000009.example.com	
<input type="checkbox"/> host-00000010.example.com	
<input type="checkbox"/> host-00000011.example.com	
<input type="checkbox"/> host-00000012.example.com	
<input type="checkbox"/> host-00000013.example.com	
<input type="checkbox"/> host-00000014.example.com	
<input type="checkbox"/> host-00000015.example.com	
<input type="checkbox"/> host-00000016.example.com	
<input type="checkbox"/> host-00000017.example.com	
<input type="checkbox"/> host-00000018.example.com	

< 1 2 3 4 5 6 >
PAGE 1 OF 6
ITEMS 1-20 OF 101

Although Tower only supports picking a single date for both hosts, you may notice different dates in the results. Remember that Tower is designed to make every attempt to find your data. If a date is selected without any scan runs, Tower gathers the previous year's worth of scan runs to verify possible data to include. Note that differences among the "ansible" and "files" modules changes are highlighted in red, while only changes for "packages" and "services" are shown.

SYSTEM TRACKING | DEMO INVENTORY

COMPARE LATEST FACTS  
COLLECTED ACROSS BOTH  
HOSTS ON OR BEFORE

 06/30/2016

PACKAGES
SERVICES
ANSIBLE

COMPARING FACTS FROM:	HOST-00000001.EXAMPLE.COM - 6/30/2016 1:20:11 PM	HOST-00000003.EXAMPLE.COM - 6/30/2016 1:20:11 PM
ansible_product_serial	NA	NA
ansible_form_factor	Other	Other
ansible_product_version	1.2	1.2
ansible_fips	false	false
ansible_service_mgr	systemd	systemd
ansible_user_id	awx	awx
ansible_user_dir	/var/lib/awx	/var/lib/awx
ansible_userspace_bits	64	64
ansible_ssh_host_key_rsa_public	AAAAAB3NzaC1yc2EAAAADAQABAAQDU0nk47tYOcejwWgW6/LaRA3Xwfo2j6U+h5G49o6kUwQikAectGhAEukYmhlRfGFwaEzpEI5Q3iynFop1VskyGHdTq9i0BSDvbbbar2xM6Me/p0MNxKaYaloiH6X1t6DI4aQEo3RYHncPh7dm/W2KHs79hRhqaqucYZ4YmgJhQ32TeureeHq6Pqj8DIOFHH+r6UgWZS2Z2+x5mGja427yMLqnD82cz3aFRu3pcErdfxWJ181q+Dg2KE1K7wjpUHEFeioaSrejxVDJ4clUQyb6MMDQkvNwORFcl7K48LejOSrclIMrGisPF4cnn1ik74Bip1yMeCoWe1JQhAOPGPfZT	AAAAAB3NzaC1yc2EAAAADAQABAAQDU0nk47tYOcejwWgW6/LaRA3Xwfo2j6U+h5G49o6kUwQikAectGhAEukYmhlRfGFwaEzpEI5Q3iynFop1VskyGHdTq9i0BSDvbbbar2xM6Me/p0MNxKaYaloiH6X1t6DI4aQEo3RYHncPh7dm/W2KHs79hRhqaqucYZ4YmgJhQ32TeureeHq6Pqj8DIOFHH+r6UgWZS2Z2+x5mGja427yMLqnD82cz3aFRu3pcErdfxWJ181q+Dg2KE1K7wjpUHEFeioaSrejxVDJ4clUQyb6MMDQkvNwORFcl7K48LejOSrclIMrGisPF4cnn1ik74Bip1yMeCoWe1JQhAOPGPfZT
ansible_ssh_host_key_ecdsa_public	AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBcSd4s1dE22hUtoxQmFt9N+ZQFcxllhRQIYSAfTC2Nlc+Q8lok/okGphfE9coUyUD3lWF9M9Tzar7JfI87pGcj4=	AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBcSd4s1dE22hUtoxQmFt9N+ZQFcxllhRQIYSAfTC2Nlc+Q8lok/okGphfE9coUyUD3lWF9M9Tzar7JfI87pGcj4=
ansible_distribution_version	7.2.1511	7.2.1511
ansible_domain	localdomain	localdomain
ansible_ssh_host_key_ed25519_public	AAAAC3NzaC1lZDI1NTE5AAAAIlsAqff3xMyHiFxsRZUN680fFOMcmYYRD3h1FXxn1joY	AAAAC3NzaC1lZDI1NTE5AAAAIlsAqff3xMyHiFxsRZUN680fFOMcmYYRD3h1FXxn1joY
ansible_processor_cores	2	2
ansible_virtualization_role	guest	guest
ansible_processor_vcpus	2	2
ansible_system_capabilities_enforced	True	True
ansible_bios_version	VirtualBox	VirtualBox
ansible_virtualization_type	virtualbox	virtualbox
ansible_memtotal_mb	1840	1840
ansible_architecture	x86_64	x86_64
ansible_swapfree_mb	1691	1691
ansible_system_vendor	innotek GmbH	innotek GmbH

## 12.4. System Tracking

ansible_os_family	RedHat	RedHat
ansible_distribution_release	Core	Core
ansible_userspace_architecture	x86_64	x86_64


## 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.

This menu opens a list of the job templates that are currently available. The job template list may be sorted and searched by **Name** or **Description**. The **Job Templates** tab also enables the user to launch, schedule, modify, and remove a job template.

The screenshot shows the 'TEMPLATES' page in Ansible Tower. At the top, there is a navigation bar with 'TOWER', 'PROJECTS', 'INVENTORIES', 'TEMPLATES', and 'JOBS'. The 'TEMPLATES' tab is active. On the right side of the navigation bar, there are icons for user profile (admin), settings, menu, and power. Below the navigation bar, the page title is 'TEMPLATES'. The main content area features a search bar with a 'SEARCH' input and a 'KEY' button. To the right of the search bar is a green '+ ADD' button. Below the search bar is a table with the following columns: NAME, TYPE, DESCRIPTION, ACTIVITY, LABELS, and ACTIONS. The table contains one row: 'Demo Job Template' (NAME), 'Job Template' (TYPE), and icons for launch, schedule, copy, edit, and delete (ACTIONS). At the bottom right of the table, it says 'ITEMS 1 - 1 OF 1'.

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: Setting the type to Check does not execute the playbook, but does check the syntax, test the environment setup, and report problems. Think of this as running the playbook in dry-run mode and having it report “changed” when an item would be changed, but not actually making changes.
  - Scan: Gather system tracking information. Users who have ‘Use’ permissions on a particular inventory have permission to create scan jobs. A default playbook has been created for your use. Custom written scan playbooks can use scan modules.
  - **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 (scan job types cannot be changed at the time of launch).

**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”.
- **Machine Credential:** Choose the machine credential to be used with this job template from the credentials available to the currently logged in Tower user.
- **Cloud Credential:** Choose the credential to be used with this job template from the credentials available to the currently logged in Tower user.
- **Network Credential** Choose the network credential to be used with this job template from the credentials available to the currently logged in Tower user.
- **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.

---

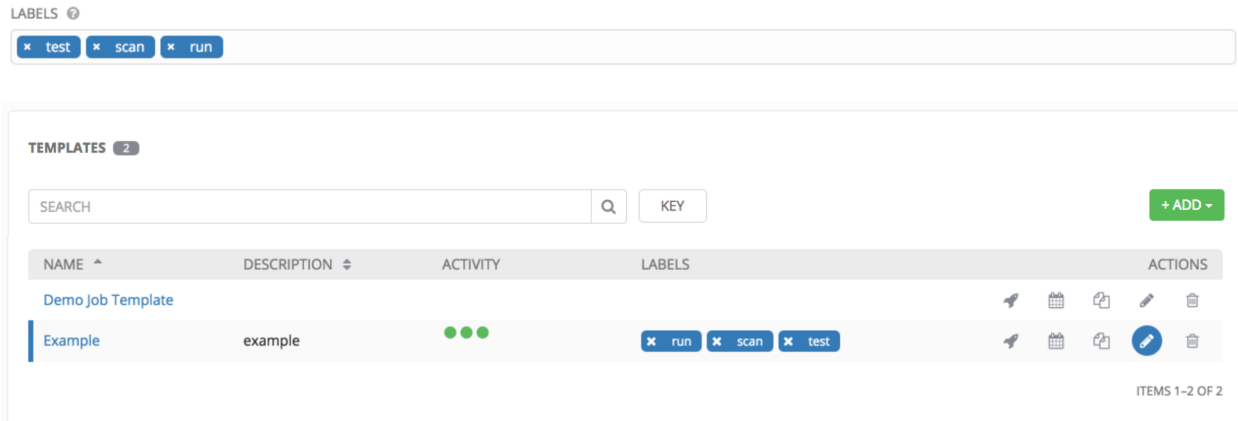
**Note:** For more information and examples refer to [Patterns](#) in the Ansible documentation.

---

- **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.
- **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 a job tag.



- **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 an 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.



- **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
```

- **Prompt on Launch** – If selected, even if a default value is supplied, you will be prompted upon launch to choose command line variables.
  - **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 [Job Concurrency](#) for additional information.
3. When you have completed configuring the job template, select **Save**.

When editing an existing job template, by clicking the job template name or the **Edit** button, the bottom of the screen displays a list of all of the jobs that have been launched from this template. Refer to the section [Jobs](#) for more information about this interface.

**Note:** Starting with Ansible Tower 3.0, Job Templates now have an option to be copied. If you choose to **Copy** a Job Template, you must understand that this **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.

The **Details** view of a saved job allows you to review, edit, and add a survey (if the job type is not a scan).

TEMPLATES / EXAMPLE

**EXAMPLE**

DETAILS | COMPLETED JOBS | PERMISSIONS | NOTIFICATIONS

\*NAME: Example

DESCRIPTION: example

\*JOB TYPE: Scan

\*INVENTORY: Demo Inventory

\*PROJECT: Default

\*PLAYBOOK: Default

\*MACHINE CREDENTIAL: Demo Credential

CLOUD CREDENTIAL:

NETWORK CREDENTIAL:

Prompt on launch:

FORKS: 0

LIMIT:

\*VERBOSITY: 0 (Normal)

Prompt on launch:

JOB TAGS:

SKIP TAGS:

Prompt on launch:

OPTIONS

- Enable Privilege Escalation
- Allow Provisioning Callbacks
- Enable Concurrent Jobs

LABELS: scan test

EXTRA VARIABLES:  YAML  JSON

1 ---

Prompt on launch:

CANCEL SAVE

The **Completed Jobs** view provides details of how this job 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.

EXAMPLE

DETAILS COMPLETED JOBS PERMISSIONS NOTIFICATIONS

STATUS FILTER

ID	NAME	TYPE	FINISHED	ACTIONS
10	Example	Playbook Run	6/30/2016 1:23:06 PM	
7	Example	Playbook Run	6/30/2016 1:16:19 PM	
4	Example	Playbook Run	6/30/2016 12:52:21 PM	

ITEMS 1-3 OF 3

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

EXAMPLE

DETAILS COMPLETED JOBS PERMISSIONS NOTIFICATIONS

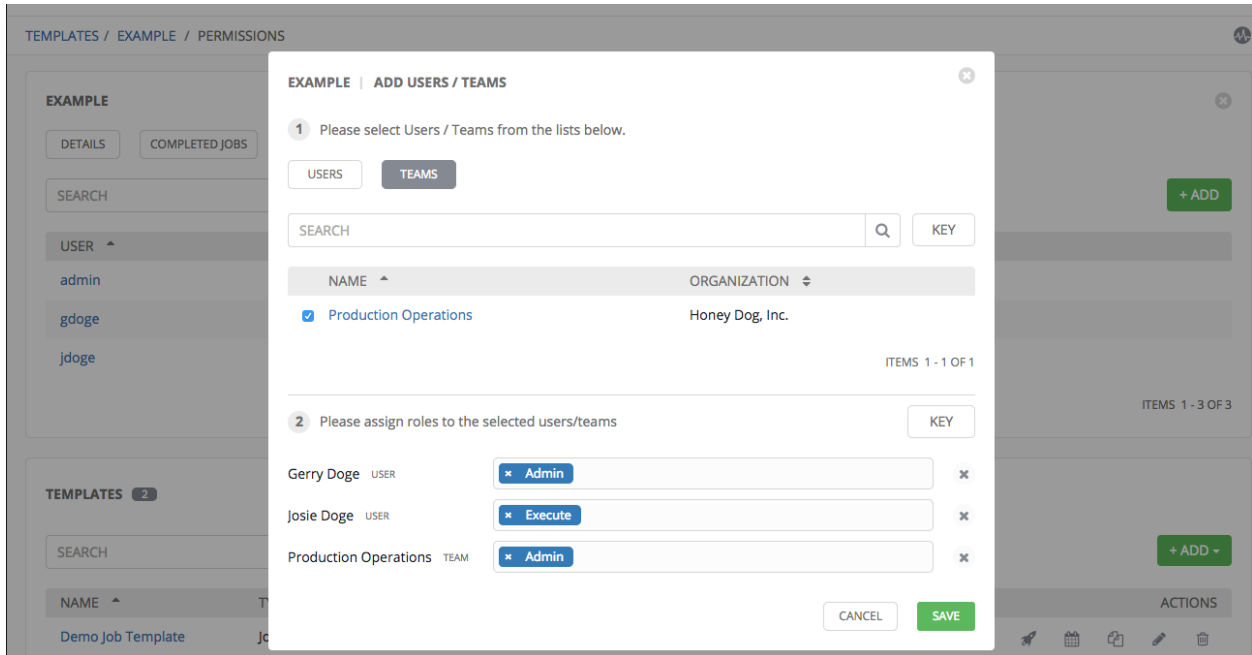
USER SEARCH

USER	ROLE	TEAM ROLES
admin	ADMIN	
cdoge		
gdoge	EXECUTE	
jdoge	ADMIN	

ITEMS 1-4 OF 4

Click the button to create new permissions for this Job Template.

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



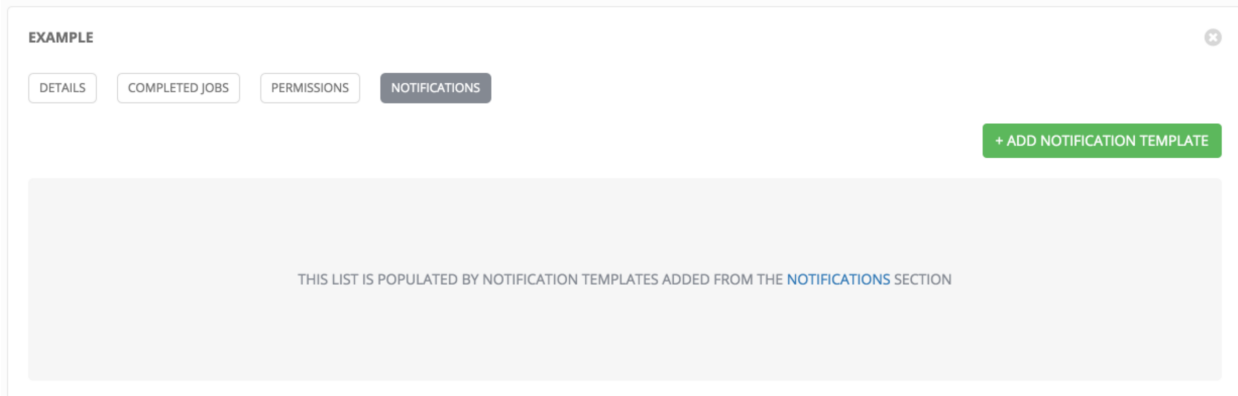
Note that you do not have to choose between teams or users, and that you can assign permissions to both at the same time.

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

**+ ADD NOTIFICATION TEMPLATE**

Click on the button to create a notification.

Refer to *Notifications* for more information.



## 13.1 Utilitizing 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.

### 13.1.1 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 }}"
```

### 13.1.2 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`
- `AWS_SECRET_KEY`

All of the AWS modules will implicitly use these credentials when run via Tower without having to set the `aws_access_key` or `aws_secret_key` module options.

### 13.1.3 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.

### 13.1.4 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_PEM_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.

### 13.1.5 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.

### 13.1.6 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

```

## 13.2 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 allows 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.

---

### 13.2.1 Creating a Survey

To create a survey:



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

Use the **ON/OFF** toggle button 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.

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.

- *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.



NEW JOB TEMPLATE | SURVEY ON
✕

**ADD SURVEY PROMPT**

\* PROMPT

DESCRIPTION

\* ANSWER VARIABLE NAME ?

\* ANSWER TYPE

REQUIRED

**PREVIEW**

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

☰

✎ ✕

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

### 13.2.2 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 ( "" ).

### 13.2.3 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.

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

### 13.2.4 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`.

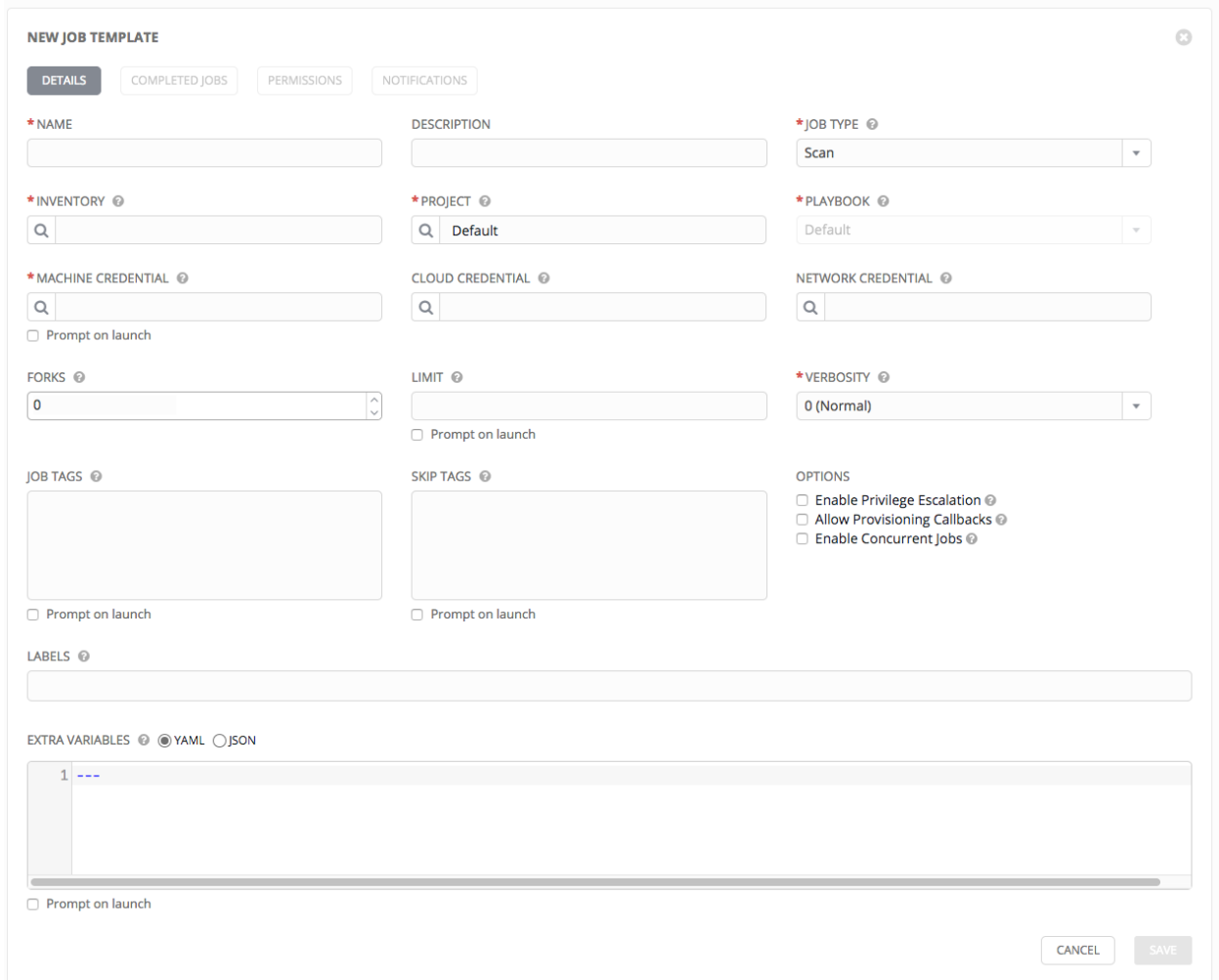
## 13.3 Scan Job Templates

Scan jobs are special Job Templates that only collect information about the host on which the job is running.

To create this special type of job template:

1. Click on **Templates**.

2. Click the  button then select **Job Template** from the menu list, which launches the **Create Job Template**



**NEW JOB TEMPLATE**

DETAILS | COMPLETED JOBS | PERMISSIONS | NOTIFICATIONS

\*NAME:

DESCRIPTION:

\*JOB TYPE:

\*INVENTORY:

\*PROJECT:

\*PLAYBOOK:

\*MACHINE CREDENTIAL:

CLOUD CREDENTIAL:

NETWORK CREDENTIAL:

Prompt on launch

FORKS:

LIMIT:

\*VERBOSITY:

Prompt on launch

JOB TAGS:

SKIP TAGS:

Prompt on launch

OPTIONS

Enable Privilege Escalation

Allow Provisioning Callbacks

Enable Concurrent Jobs

LABELS:

EXTRA VARIABLES:  YAML  JSON

1 ---

Prompt on launch

CANCEL SAVE

window.

3. Enter values for the following fields (required fields are marked with a red asterisk):

- **Name:** Enter a name appropriate for this inventory.
- **Description:** Enter an arbitrary description as appropriate.
- **Job Type:** Jobs can be of type Run, Check, or **Scan**.
- **Inventory:** Select the inventory containing the hosts you want this job to manage.
- **Project:** Select the project containing the playbook you want this job to execute. Use the default project included with Tower unless you have a specific project to scan.

- **Playbook:** Select the playbook to be executed by this job. Use the default playbook included with Tower unless you have a specific playbook to scan.
- **Machine Credential:** Select the credential you want the job to use when accessing the remote hosts. Choose the credential containing the username and SSH key or password that Ansible will need to log into the remote hosts.
- **Cloud Credential:** Selecting an optional cloud credential in the job template will pass along the access credentials to the running playbook, allowing provisioning into the cloud without manually passing parameters to the included modules.
- **Network Credential** Choose the network credential to be used with this job template from the credentials available to the currently logged in Tower user.
- **Forks:** The number of parallel or simultaneous processes to use while executing the playbook.
- **Limit:** Provide a host pattern to further constrain the list of hosts that will be managed or affected by the playbook.
  - **Prompt on Launch:** If selected, even if a default value is supplied, you will be prompted upon launch to choose a limit.
- **Verbosity:** Control the level of output Ansible produces as the playbook executes.
- **Job Tags:** Provide a comma separated list of tags to run a specific part of a play or task.
  - **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 a job tag.
- **Options:**
  - **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 [Job Concurrency](#) for additional information.
- **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 an 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.

LABELS

✕ test ✕ scan ✕ run

---

TEMPLATES 2

SEARCH

NAME ^	DESCRIPTION ⇅	ACTIVITY	LABELS	ACTIONS
<a href="#">Demo Job Template</a>				
Example	example	<span>● ● ●</span>	<span>✕ run</span> <span>✕ scan</span> <span>✕ test</span>	

ITEMS 1-2 OF 2

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

**NEW JOB TEMPLATE** ✕

DETAILS
COMPLETED JOBS
PERMISSIONS
NOTIFICATIONS

\*NAME 
DESCRIPTION 
\*JOB TYPE ?

\*INVENTORY ? 
\*PROJECT ? 
\*PLAYBOOK ?

\*MACHINE CREDENTIAL ? 
CLOUD CREDENTIAL ? 
FORKS ?

Prompt on launch

LIMIT ? 
\*VERBOSITY ? 
JOB TAGS ?

Prompt on launch

Prompt on launch

**OPTIONS**  
 Enable Privilege Escalation ?  
 Allow Provisioning Callbacks ?

**LABELS ?**

**EXTRA VARIABLES ?**  YAML  JSON

```

1 scan_file_paths:
2 - /root/
3 - /root/
4 scan_use_checksum: true
5 scan_use_recursive: true
        
```

Prompt on launch

**Note:** You cannot assign a new inventory at the time of launch to a scan job. Scan jobs must be tied to a fixed inventory.

**Note:** You cannot change the Job Type at the time of launch to or from the type of “scan”. The `ask_job_type_on_launch` option only enables you to toggle “run” versus “check” at launch time.

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

### 13.3.1 Supported OSES for Scan Job Templates

The following operating systems are supported for Scan Jobs:

- 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
- Fedora 22, 23, 24
- Amazon Linux 2016.03

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 that the scan modules depend on.

## 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.

### 13.3.2 Launching a Scan Job Template

NAME ^	DESCRIPTION ⇅	ACTIVITY	LABELS	ACTIONS
Demo Job Template				
Hello World				
Hello World 2 (Scan)		<span style="color: green;">●</span>		

You can **Launch**, **Schedule**, **Edit**, **Delete**, or **Copy** the scan job template using the buttons to the right.

Click on the  button. Enter any necessary credentials, passwords, passphrases, etc. that were setup for this job template.

The Jobs page shows details of all the tasks and events for that playbook run.



**RESULTS**

STATUS ● **Successful**    TEMPLATE **Hello World 2 (Scan)**

STARTED **6/27/2016 10:32:36 PM**    FINISHED **6/27/2016 10:32:45 PM**

LAUNCHED BY **admin**    ELAPSED **00:00:09**

INVENTORY **Demo Inventory**    MACHINE CREDENTIAL **Demo Credential**

VERBOSITY **Default**

EXTRA VARIABLES

```
1 ---
```

**STANDARD OUT**

```
SSH password:
PLAY [all] *****
TASK [setup] *****
ok: [localhost]
TASK [scan_packages] *****
ok: [localhost]
TASK [scan_services] *****
ok: [localhost]
TASK [scan_files] *****
skipping: [localhost]
PLAY RECAP *****
localhost : ok=3  changed=0  unreachable=0  failed=0
```

**DETAILS**

1 Please select from a play below to view its associated tasks.

PLAY NAME   ALL FAILED

PLAYS	STARTED	ELAPSED
<span style="color: green;">●</span> all	22:32:41	00:00:04

2 Please select a task below to view its associated hosts

TASK NAME   ALL FAILED

TASKS	STARTED	ELAPSED	HOST STATUS
<span style="color: green;">●</span> setup	22:32:41	00:00:01	<span style="color: green;">1</span>
<span style="color: green;">●</span> scan_packages	22:32:42	00:00:01	<span style="color: green;">1</span>
<span style="color: green;">●</span> scan_services	22:32:44	00:00:01	<span style="color: green;">1</span>
<span style="color: green;">●</span> scan_files	22:32:45	00:00:00	<span style="color: green;">1</span>

3 Please select a host below to view associated task details.

HOST NAME   ALL FAILED

HOSTS	ITEM	MESSAGE
<span style="color: green;">●</span> localhost		

**EVENT SUMMARY**

4 Please select a host below to view a summary of all associated tasks.

HOST NAME   ALL FAILED


HOSTS	COMPLETED TASKS
localhost	<span style="color: green;">3</span> <span style="color: blue;">1</span>

### 13.3. Scan Job Templates

HOST STATUS SUMMARY  
● OK: 100%

### 13.3.3 Scheduling a Scan Job Template



To access scheduling for your scan job, click the  button (most easily accessible from the **Job Templates** navigational link).

HELLO WORLD 2 (SCAN) | SCHEDULES 0 + ADD

PLEASE ADD ITEMS TO THIS LIST

Click the  button to add a schedule.

**DAILY SCAN** ✕

<b>* NAME</b> <input type="text" value="Daily Scan"/>	<b>* START DATE (MM/DD/YYYY)</b> <input type="text" value="06/27/2016"/>	<b>* START TIME (HH24:MM:SS)</b> <input type="text" value="22"/> : <input type="text" value="45"/> : <input type="text" value="00"/>
<b>* LOCAL TIME ZONE</b> <input type="text" value="America/New_York"/>	<b>* REPEAT FREQUENCY</b> <input type="text" value="Day"/>	

**FREQUENCY DETAILS**

<b>* EVERY</b> <input type="text" value="2"/> DAYS	<b>* END</b> <input type="text" value="On Date"/>	<b>* END DATE (MM/DD/YYYY)</b> <input type="text" value="7/19/2016"/>
<b>* END TIME (HH24:MM:SS)</b> <input type="text" value="23"/> : <input type="text" value="45"/> : <input type="text" value="00"/>		

**SCHEDULE DESCRIPTION**

every 2 days until July 19, 2016

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

```
06/27/2016 22:45:00 EDT
06/29/2016 22:45:00 EDT
07/01/2016 22:45:00 EDT
07/03/2016 22:45:00 EDT
07/05/2016 22:45:00 EDT
07/07/2016 22:45:00 EDT
07/09/2016 22:45:00 EDT
07/11/2016 22:45:00 EDT
07/13/2016 22:45:00 EDT
07/15/2016 22:45:00 EDT
```

**EXTRA VARIABLES**  YAML     JSON

1 ---

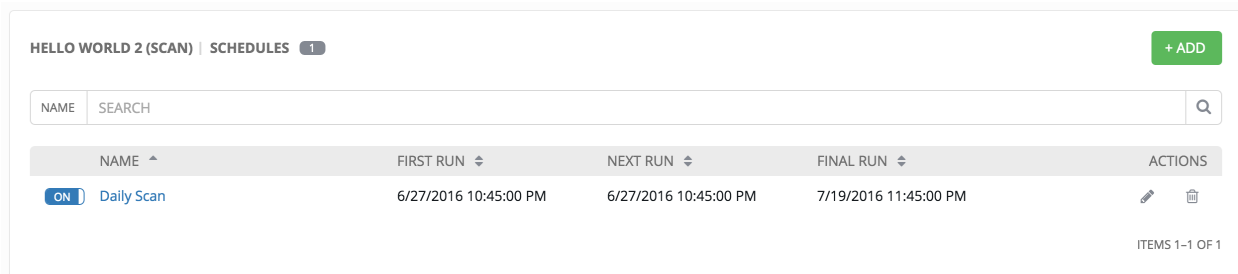
CANCEL
SAVE

Enter the appropriate details into the following fields and select **Save**:

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

The **Details** tab displays a description of the schedule and a list of the scheduled occurrences in the selected Local Time Zone.

**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.



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

There are several actions available for schedules, under the Actions column:

- Edit Schedule
- Delete schedule

### 13.3.4 Custom Scan Job Templates

Custom scan jobs are normal scan job templates which use a custom scan playbook with customized fact modules. Additional Ansible fact modules can be included easily via custom scan playbooks.

#### Fact Scan Playbooks

The default scan job playbook bundled with Tower, `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.

## 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"
  }
]
```

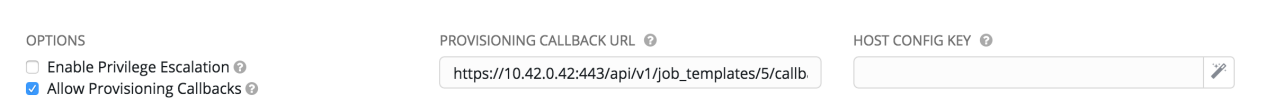
## 13.4 Provisioning Callbacks

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.



OPTIONS

Enable Privilege Escalation ⓘ


Allow Provisioning Callbacks ⓘ

PROVISIONING CALLBACK URL ⓘ

https://10.42.0.42:443/api/v1/job\_templates/5/callback

HOST CONFIG KEY ⓘ

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/v1/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 --data "host_config_key=5a8ec154832b780b9bdef1061764ae5a" \
    http://<Tower server name>/api/v1/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.

### 13.4.1 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/v1/job_templates/1/callback
```

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

## 13.5 Launching Jobs

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, 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

Here is an example job launch that prompts for Job Tags, and runs the example survey created in *Surveys*.

**LAUNCH JOB | HELLO WORLD** ✕

**OTHER PROMPTS** **SURVEY**

JOB TAGS

INVENTORY: Demo Inventory      CREDENTIAL: Demo Credential

**CANCEL** **NEXT**

**LAUNCH JOB | HELLO WORLD** ✕

OTHER PROMPTS SURVEY

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

INVENTORY      CREDENTIAL  
 Demo Inventory      Demo Credential

CANCEL      **LAUNCH**

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`: One of *manual*, *callback*, or *scheduled* to indicate how the job was started
- `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_user_id`: The user ID 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.

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

## 13.6 Scheduling



Launching job templates may also be scheduled via the  button. Clicking this button opens the **Schedules** page.



NAME	DESCRIPTION	ACTIVITY	LABELS	ACTIONS
Demo Job Template				Refresh, Calendar, Share, Edit, Delete
Hello World				Refresh, Calendar, Share, Edit, Delete
Hello World 2 (Scan)		● ● ●		Refresh, Calendar, Share, Edit, Delete


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 **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 run of the task

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

- Create a new schedule
- Refresh this view
- View Activity Stream

### 13.6.1 Add a new schedule

To create a new schedule click the  button.

**DAILY SCAN** ✕

\* NAME:       \* START DATE (MM/DD/YYYY):       \* START TIME (HH24:MM:SS):  :  :

\* LOCAL TIME ZONE:       \* REPEAT FREQUENCY:

**FREQUENCY DETAILS**

\* EVERY:  DAYS      \* END:       \* END DATE (MM/DD/YYYY):

\* END TIME (HH24:MM:SS):  :  :

**SCHEDULE DESCRIPTION**

every 2 days until July 19, 2016

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

06/27/2016 22:45:00 EDT  
 06/29/2016 22:45:00 EDT  
 07/01/2016 22:45:00 EDT  
 07/03/2016 22:45:00 EDT  
 07/05/2016 22:45:00 EDT  
 07/07/2016 22:45:00 EDT  
 07/09/2016 22:45:00 EDT  
 07/11/2016 22:45:00 EDT  
 07/13/2016 22:45:00 EDT  
 07/15/2016 22:45:00 EDT

EXTRA VARIABLES  YAML     JSON

1 ---

Enter the appropriate details into the following fields and select **Save**:

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

HELLO WORLD 2 (SCAN) | SCHEDULES + ADD

NAME SEARCH

NAME ^	FIRST RUN ↕	NEXT RUN ↕	FINAL RUN ↕	ACTIONS
<span style="background-color: #007bff; color: white; padding: 2px;">ON</span> Daily Scan	6/27/2016 10:45:00 PM	6/27/2016 10:45:00 PM	7/19/2016 11:45:00 PM	<input type="button" value="✎"/> <input type="button" value="✖"/>

ITEMS 1-1 OF 1

The **Details** tab displays a description of the schedule and a list of the scheduled occurrences in the selected Local Time Zone.

**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.

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

There are several actions available for schedules, under the Actions column:

- Edit Schedule
- Delete schedule

HELLO WORLD 2 (SCAN)   SCHEDULES 1					+ ADD
NAME	SEARCH				Q
NAME ^	FIRST RUN ↕	NEXT RUN ↕	FINAL RUN ↕	ACTIONS	
<span>ON</span> Daily Scan	6/27/2016 10:45:00 PM	6/27/2016 10:45:00 PM	7/19/2016 11:45:00 PM		

ITEMS 1-1 OF 1

## 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.

The screenshot shows the Tower web interface for the 'JOBS' section. At the top, there is a navigation bar with 'TOWER', 'PROJECTS', 'INVENTORIES', 'TEMPLATES', and 'JOBS' (selected). On the right, there are user and system icons. Below the navigation, the 'JOBS' page title is displayed. A search bar is present with a 'SEARCH' label and a 'KEY' button. Below the search bar is a table with the following columns: ID, NAME, TYPE, FINISHED, LABELS, and ACTIONS. The table contains five rows of job data, all with a green status indicator. The 'ACTIONS' column for each row contains three icons: a magnifying glass, a refresh symbol, and a trash can. At the bottom right of the table, it says 'ITEMS 1 - 5 OF 5'. A copyright notice 'Copyright © 2017 Red Hat, Inc.' is visible at the bottom right of the screenshot.

ID	NAME	TYPE	FINISHED	LABELS	ACTIONS
17	Demo Job Template	Playbook Run	1/17/2017 2:57:30 PM		
18	Demo Project	SCM Update	1/17/2017 2:57:24 PM		
16	Montgomery-Anderson	SCM Update	1/17/2017 2:29:09 PM		
8	Lee-Daniels	SCM Update	1/17/2017 1:32:26 PM		
5	Terrell, Thomas and Chapman	SCM Update	1/17/2017 1:32:21 PM		

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

**Job Details View**

Jobs / 8 - DEMO JOB TEMPLATE

**DETAILS**

- STATUS: Successful
- STARTED: 1/20/2017 1:27:48 PM
- FINISHED: 1/20/2017 1:27:54 PM
- TEMPLATE: Demo Job Template
- JOB TYPE: Run
- LAUNCHED BY: admin
- INVENTORY: Demo Inventory
- PROJECT: Demo Project
- REVISION: 347e44fea036c94d5f60e544de006453ee5c71ad
- PLAYBOOK: hello\_world.yml
- MACHINE CREDENTIAL: Demo Credential
- FORKS: 0
- VERBOSITY: 0 (Normal)
- EXTRA VARIABLES: 1

**DEMO JOB TEMPLATE**

PLAYS 1 | TASKS 2 | HOSTS 1 | ELAPSED 00:00:05

```

1
2 PLAY [Hello World Sample] ***** 13:27:53
3
4 TASK [setup] ***** 13:27:53
5 ok: [localhost]
6
7 TASK [Hello Message] ***** 13:27:54
8 ok: [localhost] => {
9   "msg": "Hello World!"
10 }
11
12 PLAY RECAP ***** 13:27:54
13 localhost          : ok=2  changed=0  unreachable=0  failed=0
14
    
```

**Standard out pane**

**Details pane**

Copyright © 2017 Red Hat, Inc.

## 14.1 Job Details - Inventory Sync

Jobs / GROUP

**DETAILS**

- NAME: Group
- STATUS: Successful
- LICENSE ERROR: False
- STARTED: 1/20/2017 1:30:35 PM
- FINISHED: 1/20/2017 1:30:38 PM
- ELAPSED: 3.191 seconds
- LAUNCH TYPE: Manual
- GROUP: Group
- SOURCE: Custom Script
- OVERWRITE: False
- OVERWRITE VARS: False



**STANDARD OUT**

```

1.134 INFO  Updating inventory 1: Demo Inventory
1.146 INFO  Reading executable JSOW source: /tmp/ansible_tower_launch_CGwEbl/tmpKK;
1.205 INFO  Loaded 22 groups, 100 hosts
1.212 INFO  Group "group" variables unmodified
1.219 INFO  Group "eights.example.com" added
1.225 INFO  Group "events.example.com" added
1.230 INFO  Group "fives.example.com" added
1.235 INFO  Group "fours.example.com" added
1.239 INFO  Group "group-000000X.example.com" added
1.244 INFO  Group "group-000001X.example.com" added
1.249 INFO  Group "group-000002X.example.com" added
1.253 INFO  Group "group-000003X.example.com" added
1.260 INFO  Group "group-000004X.example.com" added
1.266 INFO  Group "group-000005X.example.com" added
1.271 INFO  Group "group-000006X.example.com" added
1.275 INFO  Group "group-000007X.example.com" added
1.281 INFO  Group "group-000008X.example.com" added
1.285 INFO  Group "group-000009X.example.com" added
1.290 INFO  Group "group-00000XX.example.com" added
1.294 INFO  Group "group-00000XX.example.com" added
1.299 INFO  Group "nines.example.com" added
1.304 INFO  Group "odds.example.com" added
1.308 INFO  Group "sevens.example.com" added
1.313 INFO  Group "sixes.example.com" added
1.318 INFO  Group "tens.example.com" added
    
```

Copyright © 2017 Red Hat, Inc.

### 14.1.1 Details



The **Details** pane shows the basic status of the job—*Running*, *Pending*, *Successful*, or *Failed*—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 *Pending*, *Running*, *Successful*, or *Failed*.
- **License Error:** Only shown for **Inventory Sync** jobs. If this is *True*, the hosts added by the inventory sync caused Tower to exceed the licensed number of managed hosts.
- **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*.
- **Group:** The group being synced.
- **Source:** The type of cloud inventory.
- **Overwrite:** The value of *Overwrite* for this Inventory Sync. Refer to *Inventories* for details.
- **Overwrite Vars:** The value of *Overwrite Vars* for this Inventory Sync. Refer to *Inventories* for details.

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

### 14.1.2 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 (  ).

## 14.2 Job Details - SCM

The screenshot shows the Ansible Tower interface. At the top, there are navigation tabs: TOWER, PROJECTS, INVENTORIES, TEMPLATES, and JOBS. The user is logged in as 'admin'. The main content area is titled 'JOBS / DEMO PROJECT'. It is divided into two panels:



- DETAILS:** A table showing job information:
 

NAME	Demo Project
STATUS	Successful
STARTED	1/20/2017 2:43:52 PM
FINISHED	1/20/2017 2:43:56 PM
ELAPSED	3.641 seconds
LAUNCH TYPE	Manual
PROJECT	Demo Project
- STANDARD OUT:** A scrollable text area showing the execution output:
 

```
Using /etc/ansible/ansible.cfg as config file
PLAY [all] *****
TASK [delete project directory before update] *****
skipping: [localhost]
TASK [update project using git and accept hostkey] *****
skipping: [localhost]
TASK [Set the git repository version] *****
skipping: [localhost]
TASK [update project using git] *****
ok: [localhost]
TASK [Set the git repository version] *****
ok: [localhost]
TASK [update project using hg] *****
skipping: [localhost]
TASK [Set the hg repository version] *****
skipping: [localhost]
TASK [update project using svn] *****
```

At the bottom right of the screenshot, there is a small copyright notice: "Copyright © 2017 Red Hat, Inc."

### 14.2.1 Details



The **Details** pane shows the basic status of the job—*Running*, *Pending*, *Successful*, or *Failed*—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 *Pending*, *Running*, *Successful*, or *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.

### 14.2.2 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 (  ).

## 14.3 Job Details - Playbook Run

The screenshot displays the Ansible Tower interface for a job titled 'DEMO JOB TEMPLATE'. The left sidebar contains a 'DETAILS' pane with the following information:

- STATUS:** Successful
- STARTED:** 1/20/2017 1:27:48 PM
- FINISHED:** 1/20/2017 1:27:54 PM
- TEMPLATE:** Demo Job Template
- JOB TYPE:** Run
- LAUNCHED BY:** admin
- INVENTORY:** Demo Inventory
- PROJECT:** Demo Project
- REVISION:** 347e44f... (truncated)
- PLAYBOOK:** hello\_world.yml
- MACHINE CREDENTIAL:** Demo Credential
- FORKS:** 0
- VERBOSITY:** 0 (Normal)
- EXTRA VARIABLES:** (empty)

The main execution log on the right shows the following steps:

```

1
2 PLAY [Hello World Sample] ***** 13:27:53
3
4 TASK [setup] ***** 13:27:53
5 ok: [localhost]
6
7 TASK [Hello Message] ***** 13:27:54
8 ok: [localhost] => {
9   "msg": "Hello World!"
10 }
11
12 PLAY RECAP ***** 13:27:54
13 localhost : ok=2  changed=0  unreachable=0  failed=0
14
  
```

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

### 14.3.1 Details

The **Details** pane shows the basic status of the job—*Running*, *Pending*, *Successful*, or *Failed*—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 *Pending*, *Running*, *Successful*, or *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.



### 14.3.2 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

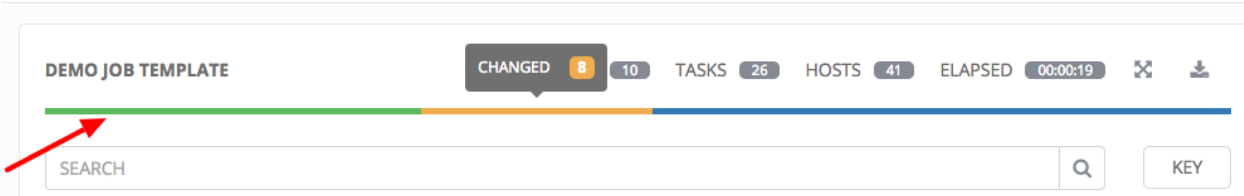
The events summary captures a tally of events that were run as part of this playbook:

- 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.

DEMO JOB TEMPLATE      PLAYS 10   TASKS 26   HOSTS 41   ELAPSED 00:00:19



SEARCH [ ]    Q    KEY

or.stdout.contains:failed    CLEAR ALL

Line	Task	Duration
1		
2	PLAY [add hosts to inventory] *****	16:06:07
3		
4	TASK [setup] *****	16:06:07
6		
7	TASK [create inventory] *****	16:06:08
18		
19	RUNNING HANDLER [single host handler] *****	16:06:08

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

## Standard output view

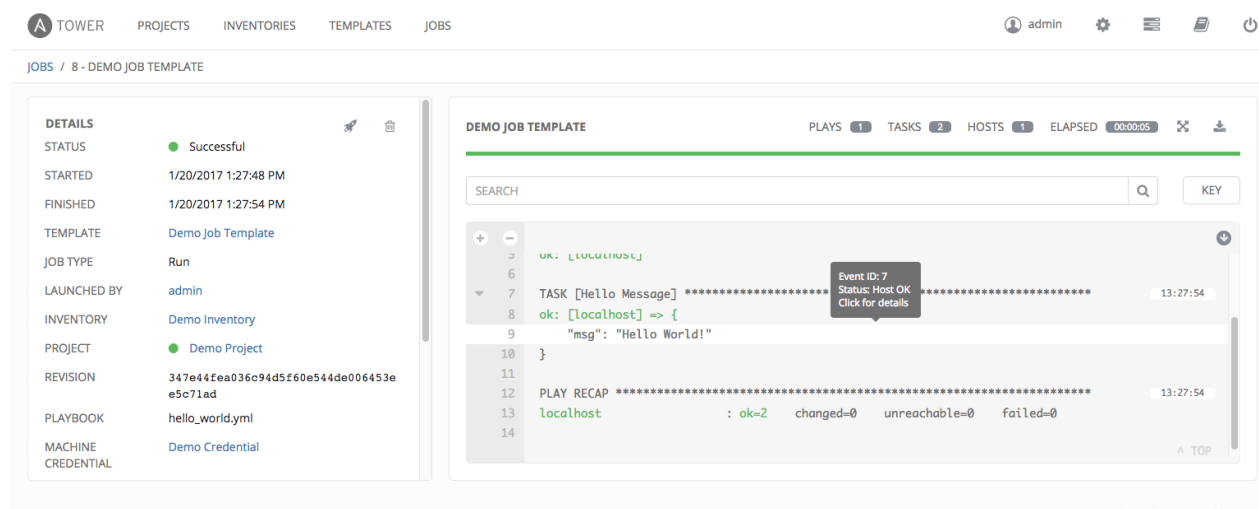
The standard output view displays all the events that occur on a particular job. By default, all rows are expanded so that all the details are displayed. Use the collapse-all button (  ) to switch to a view that only contains the headers for plays and tasks. Click the (  ) button to view all lines of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

3		
4	TASK [setup] *****	16:06:07
6		
7	TASK [create inventory] *****	16:06:08
8	changed: [host] => (item=3)	

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.



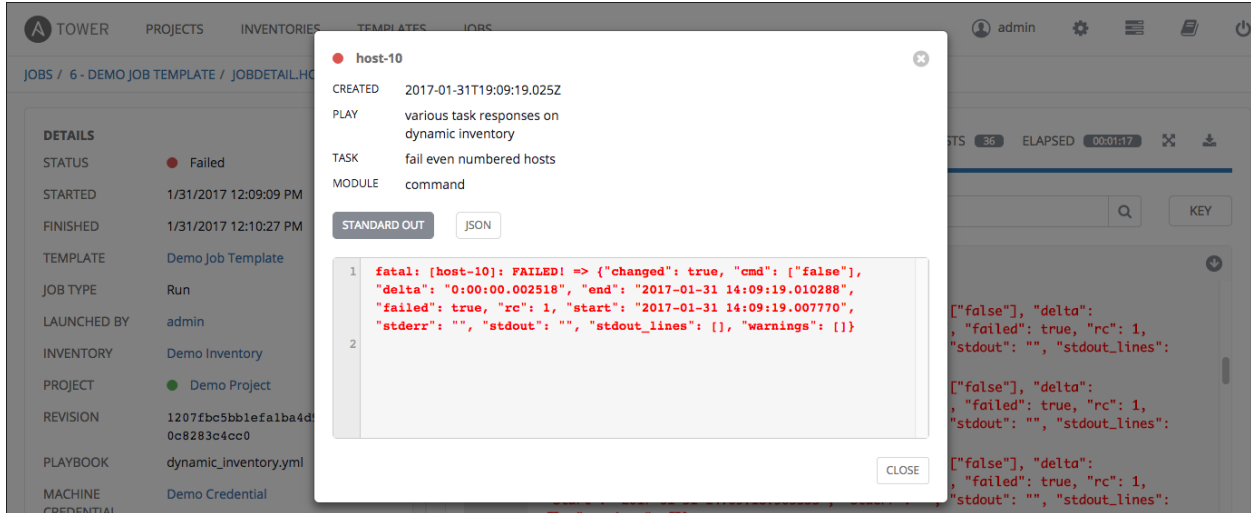
Copyright © 2017 Red Hat, Inc.

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.

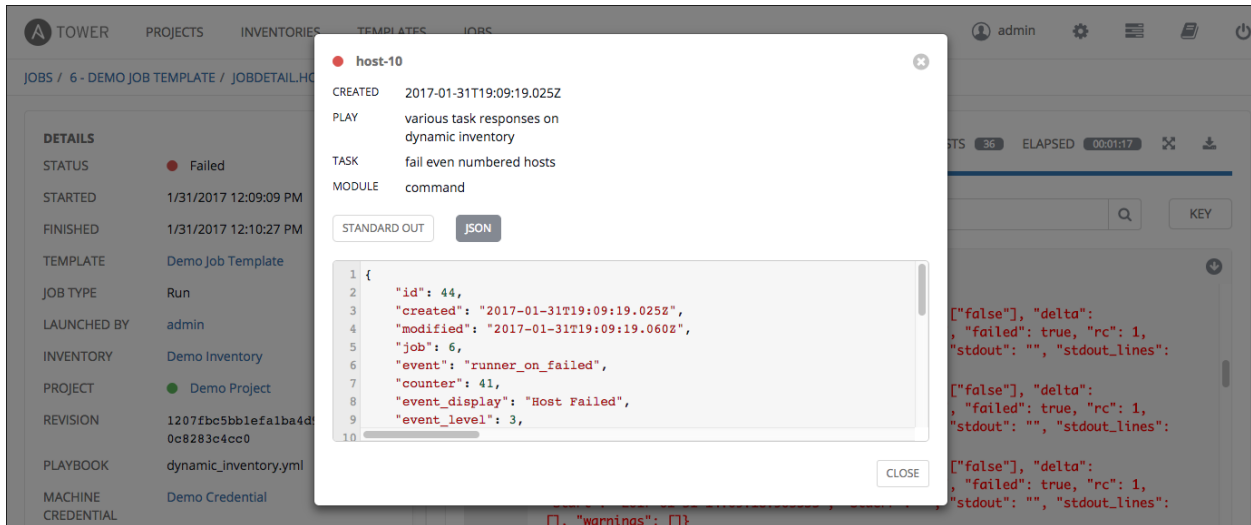
## Host Events

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



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



## 14.4 Job Concurrency

Tower limits the number of simultaneous jobs that can run based on the amount of physical memory and the complexity of the playbook. Ansible Tower 3.1.0 adds additional job capacity by having N additional Tower hosts in a cluster. You will be able to view Tower’s capacity for running jobs in the Ping API Endpoint. The job itself also displays the node the task is running on.

If the “Update on Launch” setting is checked, job templates that rely on the inventory or project *also* trigger an update on them if it is within the cache timeout. If multiple jobs are launched within the cache timeout that depend on the same project or inventory, only one of those project or inventory updates is created (instead of one for each job that relies on it).

If you are having trouble, try setting the cache timeout on the project or inventory source to 60 seconds.

The restriction related to the amount of RAM on your Tower server and the size of your inventory equates to the total number of machines from which facts can be gathered and stored in memory. The algorithm used is:

```
50 + ((total memory in megabytes) / 1024) - 2) * 75
```

With 50 as the baseline.

Each job that runs is:

```
(number of forks on the job) * 10
```

Which defaults to 50 if the limit is set Tower's default value of 0.

Forks determine the default number of parallel processes to spawn when communicating with remote hosts. The fork number is automatically limited to the number of possible hosts, so this is really a limit of how much network and CPU load you can handle. Many users set this to 50, while others set it to 500 or more. If you have a large number of hosts, higher values will make actions across all of those hosts complete faster. You can edit the `ansible.cfg` file to change this value.

The Ansible fork number default is extremely conservative and is set to five (5). When you do not pass a forks value in Tower (leaving it as 0), Ansible uses 5 forks (the default). If you set your forks value to one (1) in Tower, Ansible uses the value entered and one (1) fork is created. Non-zero inputs are used as instructed.

As an example, if you have a job with 0 forks (the Tower default) on a system with 2 GB of memory, your algorithm would look like the following:

```
50 + ((2048 / 1024) - 2) * 75 = 50
```

If you have a job with 0 forks (the Tower default) on a system with 4 GB of memory then you can run four (4) tasks simultaneously which includes callbacks.

```
50 + ((4096 / 1024) - 2) * 75 = 200
```

This can be changed by setting a value in the Tower settings file (`/etc/tower/settings.py`) such as:

```
SYSTEM_TASK_CAPACITY = 300
```

If you want to override the setting, use caution, as you may run out of RAM if you set this value too high. You can determine what the calculated setting is by reviewing `/var/log/tower/task_system.log` and looking for a line similar to:

```
Running Nodes: []; Capacity: 50; Running Impact: 0; Remaining Capacity: 50
```

The `Capacity: 50` is the current calculated setting.

As long as you have the capacity to do so, Tower attempts to queue and run the most number of jobs possible. There are some blockers and exceptions worth noting, however.

- A Job Template will block the same instance of another Job Template launched with the same inventory (this is a change in behavior – prior to Ansible Tower 3.0, a Job Template would block the same instance of another Job Template).
- A project update, an SCM update, or an inventory update will block another project requiring the same update(s).
- Job Templates which launch via provisioning callbacks can run, just not as an instance on the same host. This allows running two (2) templates on the same inventory. However, if the inventory requires an update, they will not run. Callbacks are special types of job templates which receive “push requests” from the host to the inventory. They run on one host only and can run parallel with other jobs as long as they are different callbacks and different hosts.
- System Jobs can only run one at a time. They block all other jobs and must be run on their own to avoid conflict. System jobs will finish behind jobs schedule ahead of them, but will finish ahead of those jobs scheduled behind it.

- Jobs may be queued if they are waiting on spare job running capacity.
- Ad hoc jobs are blocked by any inventory updates running against the inventory for that ad hoc job as specified.

## 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/v1/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/v1/job_templates/n/notifiers_error` API endpoint.

### 15.1 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


### 15.2 Workflow

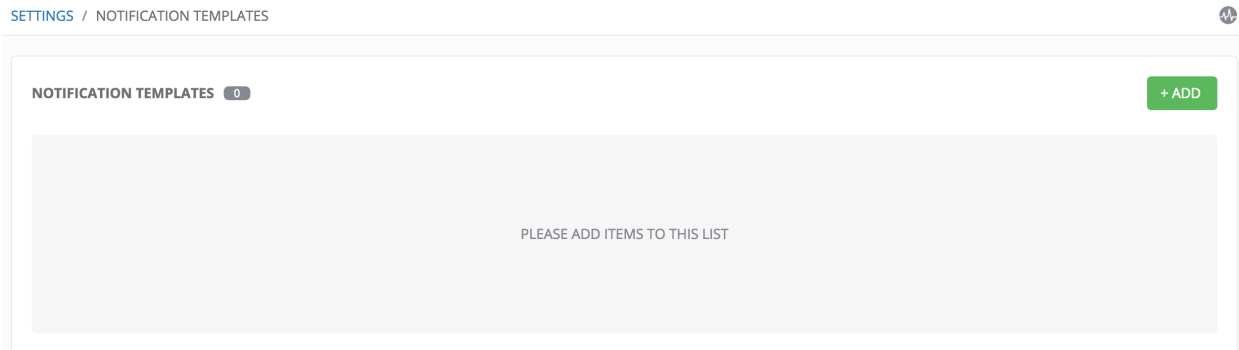
When a job succeeds or fails, the error or success handler will pull a list of relevant notifiers using the procedure defined above. It will then create a Notification object for each one containing relevant details about the job and then sends it to the destination (email addresses, slack channel(s), sms numbers, etc). These Notification objects are available as

related resources on job types (jobs, inventory updates, project updates), and also at `/api/v1/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/v1/notifications/<n>`).

## 15.3 Create a Notification Template

To create a Notification Template, click the  button and select **Notifications**.



## 15.4 Notification Types

### Topics:

- *Email*
- *Slack*
- *Twilio*
- *PagerDuty*
- *HipChat*
- *Webhook*
- *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.

### 15.4.1 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** ✕

<p><b>* NAME</b></p> <input type="text" value="Tell Me"/>	<p><b>DESCRIPTION</b></p> <input type="text" value="notification test"/>	<p><b>* ORGANIZATION</b></p> <input type="text" value="Honey Dog, Inc."/>
<p><b>* TYPE</b></p> <input type="text" value="Email"/>		
<p><b>TYPE DETAILS</b></p>		
<p><b>* USERNAME</b></p> <input type="text" value="luckydog"/>	<p><b>* HOST</b></p> <input type="text" value="honeydog.com"/>	<p><b>* SENDER EMAIL</b></p> <input type="text" value="info@honeydog.com"/>
<p><b>* RECIPIENT LIST</b> ⓘ</p> <div style="border: 1px solid #ccc; padding: 5px; min-height: 40px;"> <p>engineering@honeydog.com</p> <p>release-managers@honeydog.com</p> </div>	<p><b>* PASSWORD</b></p> <input style="width: 100%;" type="text" value="SHOW ....."/>	<p><b>* PORT</b></p> <input type="text" value="1"/>
<p><b>OPTIONS</b></p> <p><input type="checkbox"/> Use TLS</p> <p><input type="checkbox"/> Use SSL</p>		
		<p><input type="button" value="CANCEL"/> <input type="button" value="SAVE"/></p>

**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.

## 15.4.2 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.

**NEW NOTIFICATION TEMPLATE** ✕

<p><b>* NAME</b></p> <input type="text" value="Tell Me"/>	<p><b>DESCRIPTION</b></p> <input type="text" value="notification test"/>	<p><b>* ORGANIZATION</b></p> <input type="text" value="Honey Dog, Inc."/>
<p><b>* TYPE</b></p> <input type="text" value="Slack"/>		
<p><b>TYPE DETAILS</b></p>		
<p><b>* DESTINATION CHANNELS</b> ⓘ</p> <div style="border: 1px solid #ccc; padding: 5px; min-height: 40px;"> <p>#engineering</p> <p>#helpdesk</p> <p>#support</p> </div>	<p><b>* TOKEN</b></p> <input style="width: 100%;" type="text" value="SHOW ....."/>	
		<p><input type="button" value="CANCEL"/> <input type="button" value="SAVE"/></p>

### 15.4.3 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

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

- \* NAME:** Tell Me
- DESCRIPTION:** notification test
- \* ORGANIZATION:** Honey Dog, Inc.
- \* TYPE:** Twilio
- TYPE DETAILS:**
  - \* ACCOUNT TOKEN:** SHOW [REDACTED]
  - \* SOURCE PHONE NUMBER:** 9109876555
  - \* DESTINATION SMS NUMBER:** 9109676565
- \* ACCOUNT SID:** Aa54e6dg4345d3t676f60

Buttons: CANCEL, SAVE

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

## 15.4.5 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.).

**NEW NOTIFICATION TEMPLATE**

\*NAME:  DESCRIPTION:  \*ORGANIZATION:

\*TYPE:

**TYPE DETAILS**

\*DESTINATION CHANNELS:  \*TOKEN:  \*LABEL TO BE SHOWN WITH NOTIFICATION:

\*API URL:  \*NOTIFICATION COLOR:   NOTIFY CHANNEL

### 15.4.6 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:  DESCRIPTION:  \*ORGANIZATION:

\*TYPE:

**TYPE DETAILS**

\*TARGET URL:

\*HTTP HEADERS: 2"/>

## 15.4.7 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

The screenshot shows the 'NEW NOTIFICATION TEMPLATE' form. It has the following fields and values:

- \* NAME:** Tell Me
- DESCRIPTION:** notification test
- \* ORGANIZATION:** Honey Dog, Inc.
- \* TYPE:** IRC
- TYPE DETAILS:**
  - \* IRC SERVER PASSWORD:** SHOW [REDACTED]
  - \* IRC SERVER PORT:** 6667
  - \* IRC SERVER ADDRESS:** irc.testirc.net
  - \* IRC NICK:** helpbot
  - \* DESTINATION CHANNELS OR USERS:** #engineering, #release-engineers
  - SSL CONNECTION:**

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

## 15.5 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.

### 15.5.1 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 'VIEW YOUR LICENSE' link 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.

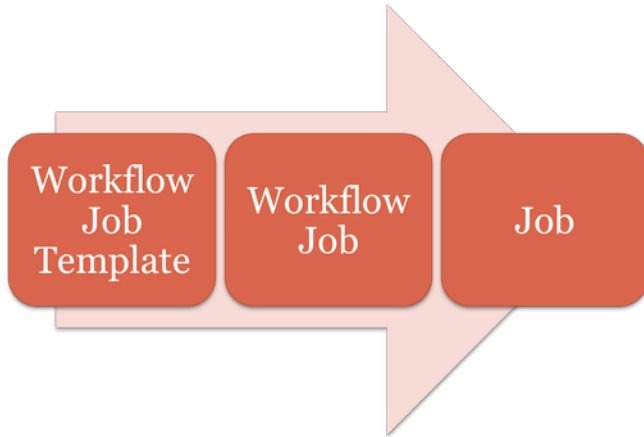
As the workflow runs, jobs are spawned from the node’s linked template. Nodes linking to a job template which has prompt-driven fields (`job_type`, `job_tags`, `skip_tags`, `limit`, and associated credential and inventory) can contain those fields, and will substitute them into the job on launch. A job template that is associated with each workflow node will run based on the success or failure scenario as it proceeds. The successful completion of a job triggers the execution of other job(s), and failure of a job potentially triggers the execution of different job(s).

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.

## 16.1 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:

Ansible	Tower
	set_stats (i.e. artifacts)
	Job Artifacts
custom facts	<div style="border: 2px solid red; border-radius: 15px; padding: 5px;">                     Workflow Job Template extra variables                      Workflow Job Template Survey (defaults)                      Workflow Job Launch extra variables                 </div>

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.

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.

## 16.2 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](#).

## BEST PRACTICES

### 17.1 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.

### 17.2 Ansible file and directory structure

Please review the Ansible best practices from the Ansible documentation at [http://docs.ansible.com/playbooks\\_best\\_practices.html](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` 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.

---

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.

### 17.3 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.

---

## 17.4 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.

## 17.5 Autoscaling

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

## 17.6 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](#).

## 17.7 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 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.

### 18.1 Playbook Access and Information Sharing

By default, Tower’s multi-tenant security prevents playbooks from reading files outside of their project directory. To share information between playbooks or to read files on the file system outside of their project directory, you must edit `/etc/tower/settings.py` and add the directories that are available to the `AWX_PROOT_SHOW_PATHS` setting.

The following paths, plus any user specified paths, are hidden by `AWX_PROOT_HIDE_PATHS`:

- `/etc/tower`
- `/var/lib/awx`
- `/var/log`
- `/tmp`
- `/var/lib/awx/projects`
- `/var/lib/awx/job_status`

The following paths, plus any user specified paths, are shown by `AWX_PROOT_SHOW_PATHS`:

- `/var/lib/awx/projects/<current_project>`
- `/tmp/ansible_tower_XXXXX`

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.

## 18.2 PRoot functionality and variables

The PRoot 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 PRoot settings in some cases. To fine tune your usage of PRoot, there are certain variables that can be set:

```
# Enable proot support for running jobs (playbook runs only).
AWX_PROOT_ENABLED = False

# Command/path to proot.
AWX_PROOT_CMD = 'proot'

# Additional paths to hide from jobs using proot.
AWX_PROOT_HIDE_PATHS = []

# Additional paths to show for jobs using proot.
AWX_PROOT_SHOW_PATHS = []
```

To customize your PRoot settings, navigate to the `/etc/tower/settings.py` file. Once your changes have been saved, restart services with the `ansible-tower-service restart` command.

## 18.3 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.

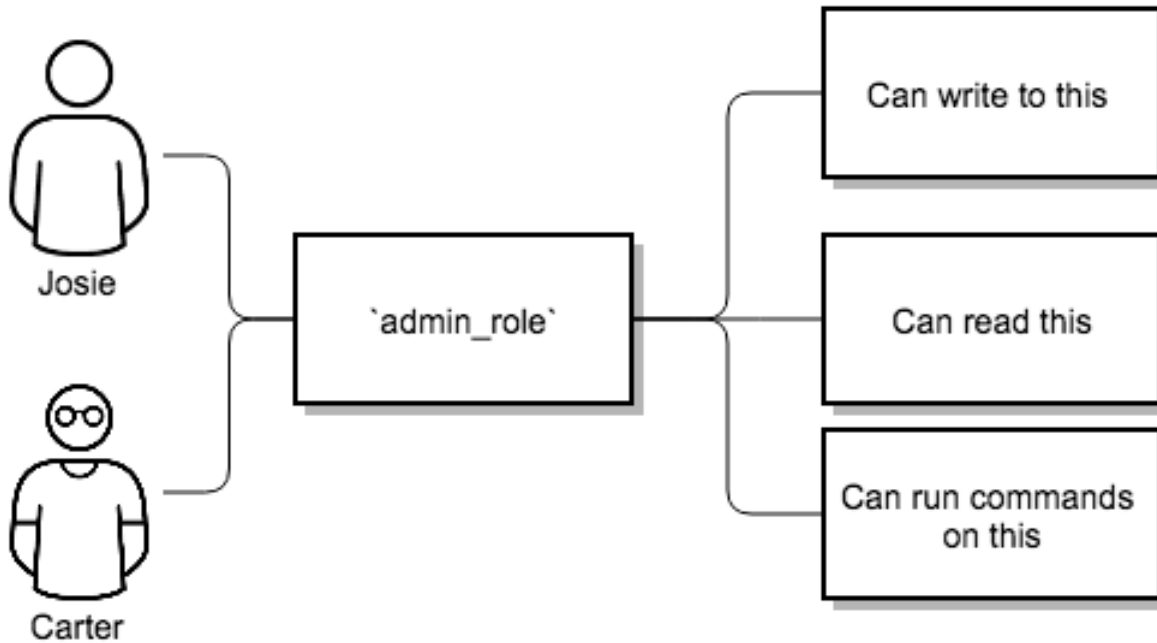
There are a few main concepts that you should become familiar with regarding Tower’s RBAC design—roles, resources, and users. Users can be members of a role, which gives them certain access to any resources associated with that role, or any resources associated with “descendant” roles.

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.

### 18.3.1 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 made both people members of the organization’s `admin_role`.

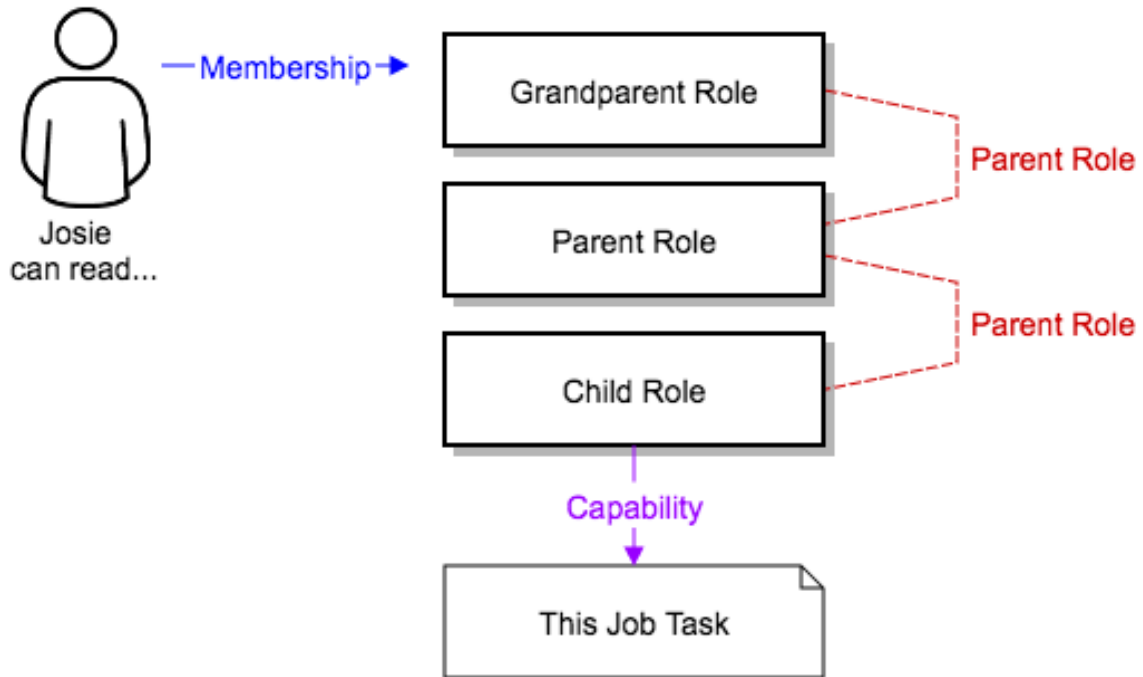


Often, you will have many Roles in a system and you will want some roles to include all of the capabilities of other roles. For example, you may want a System Administrator to have access to everything that an Organization Administrator has access to, who has everything that a Project Administrator has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

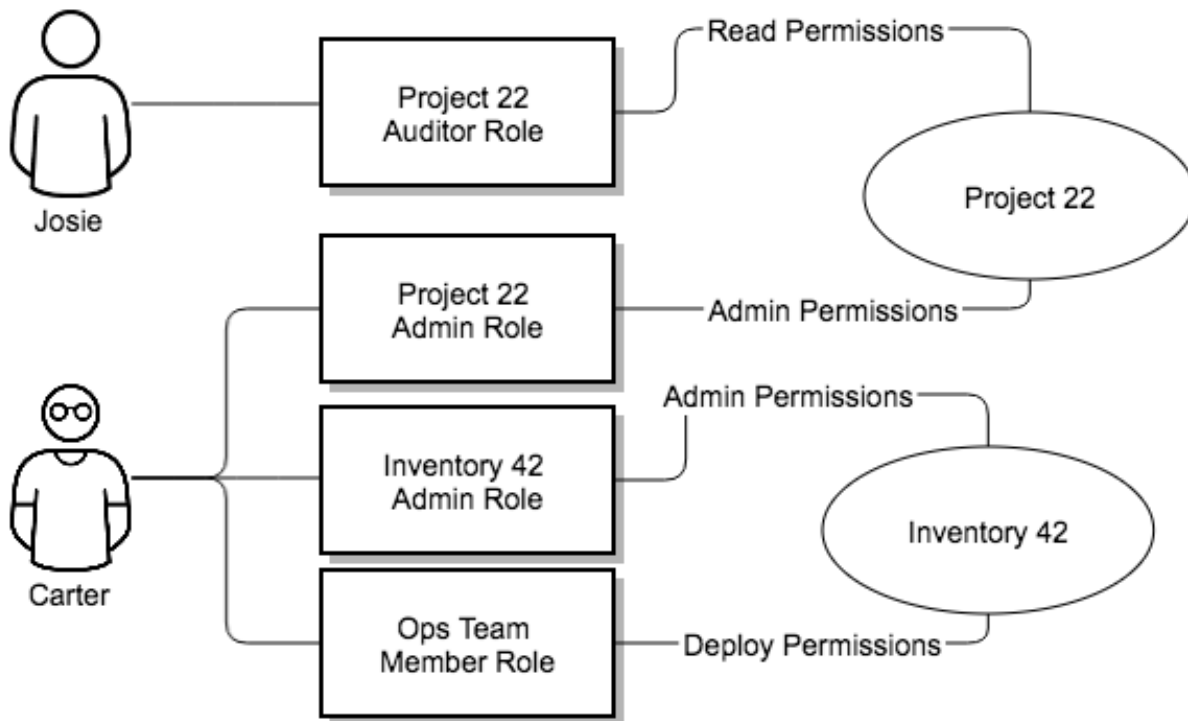
This concept is referred to as the ‘Role Hierarchy’:

- 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.

### 18.3.2 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 an *Auditor*.

- System administrators implicitly inherit all capabilities for all objects (read/write/execute) within the Tower environment.
- 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.

### **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.

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.

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 make changes to the inventory, project, playbook, or credentials used in the job template, the user must also have the “Use” role for the project, inventory, and all credentials currently being used or being set.

### 18.3.3 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, Projects, Inventory, Projects, Job Templates	Manages all aspects of a defined Organization, Project, Inventory, or Job Template
Auditor Role - Organizations, Projects, Inventory, Projects, Job Templates	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 - All	Views settings for a defined Organization, Project, Inventory, or Job Template
SCM 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 you can create and is intended for system wide roles.

**INDEX**

- `genindex`

**COPYRIGHT © 2016 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.

## A

- about Tower
  - settings menu, 14
- activity streams, 17
- ad hoc commands, 85
  - inventories, 85
- add new
  - inventories, 64
  - scan job, 107
- adding new
  - credentials, 45
- admin menu, 13
- Amazon Web Services
  - credential types, 49
  - inventories, 70
- Ansible Galaxy, 63
- Ansible Galaxy integration
  - features, 3
- automation
  - features, 2
- autoscaling
  - best practices, 147
- autoscaling flexibility
  - features, 3
- AWS
  - cloud credentials, 101

## B

- backup and restore
  - features, 3
- best practices, 146
  - autoscaling, 147
  - deployment, continuous, 147
  - dynamic inventory sources, 146
  - file and directory structure, 146
  - host counts, larger, 147
  - integration, continuous, 147
  - source control, 146
  - variable inventory management, 147

## C

- callbacks

- extra variables, 118
- check
  - job types, 94
- cloud credentials
  - AWS, 101
  - Google, 102
  - job templates, 100
  - MS Azure, 102
  - OpenStack, 101
  - Rackspace, 102
  - VMware, 102
- cloud flexibility
  - features, 3
- CloudForms
  - credential types, 52
- components
  - licenses, 7
- concurrency
  - jobs, 132
- configure Tower
  - settings menu, 14
- create template
  - notifications, 136
- credential types, 45
  - Amazon Web Services, 49
  - CloudForms, 52
  - Google Compute Engine, 53
  - machine, 46
  - Microsoft Azure Classic, 53
  - Microsoft Azure Resource Manager, 54
  - network, 48
  - OpenStack, 55
  - rackspace, 50
  - Red Hat Satellite, 52
  - source control, 48
  - VMware, 51
- credentials, 43
  - adding new, 45
  - getting started, 44
  - how they work, 43
  - settings menu, 14
  - types, 45

- custom
  - fact scan job, 116
  - scan job, 115
- custom fact scans
  - playbook, 116
  - system tracking, 116
- custom script
  - inventories, 78
- D**
- dashboard, 16
  - host count, 17
  - job status, 16
  - jobs tab, 16
  - main menu, 12
  - schedule status, 16
- DEB files
  - licenses, 7
- deployment, continuous
  - best practices, 147
- dynamic inventory sources
  - best practices, 146
- E**
- Email
  - notifications types, 136
- evaluation, 6
- extra variables
  - callbacks, 118
  - provisioning callbacks, 118
  - surveys, 105, 144
- extra\_vars, 106
- F**
- fact scan job
  - custom, 116
  - playbook, 115
- fact scan playbook
  - system tracking, 115
- features, 4
  - Ansible Galaxy integration, 3
  - automation, 2
  - autoscaling flexibility, 3
  - backup and restore, 3
  - cloud flexibility, 3
  - inventory sources, Red Hat CloudForms, 4
  - inventory sources, Red Hat Satellite 6, 4
  - notifications, 4
  - OpenStack inventory support, 3
  - overview, 2
  - 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, 146
- forks
  - jobs, 132
- functionality
  - PRoot, 149
- G**
- Galaxy support, 63
- getting started
  - credentials, 44
- Google
  - cloud credentials, 102
- Google Compute Engine
  - credential types, 53
  - inventories, 72
- groups
  - notifications, 135
- H**
- Hipchat
  - notifications types, 136
- host count
  - dashboard, 17
- host counts, larger
  - best practices, 147
- host to host
  - scan job, 90
  - system tracking, 90
- hostname configuration
  - notifications, 141
- hosts
  - inventories, 83
- hosts, add new
  - inventories, 84
- how they work
  - credentials, 43
- I**
- installation bundle
  - licenses, 7
- integration, continuous
  - best practices, 147
- inventories, 64
  - ad hoc commands, 85
  - add new, 64
  - Amazon Web Services, 70
  - custom script, 78
  - Google Compute Engine, 72
  - groups, 66, 67
    - add new, 67
  - groups and hosts, 66

- hosts, 83
- hosts, add new, 84
- Microsoft Azure Classic (deprecated), 72
- Microsoft Azure Resource Manager, 73
- OpenStack, 77
- Rackspace Cloud Servers, 69
- Red Hat CloudForms, 76
- Red Hat Satellite 6, 75
- scan job, 87
- scan job templates, creating, 107
- scan job templates, custom, 115
- scan job templates, launching, 112
- scan job templates, scheduling, 114
- scheduling, 79
- scheduling, add new, 80
- system tracking, 87
- VMware vCenter, 74
- inventory scripts
  - settings menu, 14
- inventory sources
  - notifications, 135
- inventory sources, Red Hat CloudForms
  - features, 4
- inventory sources, Red Hat Satellite 6
  - features, 4
- inventory sync
  - job results, 125
- IRC
  - notifications types, 136

**J**

- job results, 124
  - inventory sync, 125
- job status
  - dashboard, 16
- job templates, 94
  - cloud credentials, 100
  - job variables, 106
  - jobs, launching, 119
  - portal mode, 15
  - provisioning callbacks, 117
  - relaunch, 106
  - scheduling, 120, 121
  - survey creation, 103
  - survey extra variables, 105
  - survey optional questions, 105
  - surveys, 103
- job templates, hierarchy, 106
- job templates, overview, 106
- job types
  - check, 94
  - run, 94
  - scan, 94
- job variables

- job templates, 106
- jobs, 124
  - concurrency, 132
  - event summary, 129
  - events summary, 129
  - forks, 132
  - host events, 131
  - host status bar, 129
  - host summary, 129
  - job summary, 129
  - notifications, 135
  - portal mode, 16
  - results, 124
- jobs results
  - playbook run, 128
  - SCM, 127
- jobs tab
  - dashboard, 16
- jobs, launching
  - job templates, 119

**L**

- launching
  - scan job, 112
- license, 4, 5
  - features, 7
  - nodes, 6
  - trial, 6
  - troubleshooting, 10
  - types, 6
- license features, 4
- license, add manually, 10
- license, import, 9
- license, viewing, 14
- licenses
  - components, 7
  - DEB files, 7
  - installation bundle, 7
  - RPM files, 7
- logging in, 8

**M**

- machine
  - credential types, 46
- main menu
  - dashboard, 12
- management jobs
  - settings menu, 14
- Microsoft Azure Classic
  - credential types, 53
- Microsoft Azure Classic (deprecated)
  - inventories, 72
- Microsoft Azure Resource Manager
  - credential types, 54

- inventories, 73
- MS Azure
  - cloud credentials, 102
- my view, 15

## N

- network
  - credential types, 48
- new schedule addition
  - projects, 62
- notifications
  - create template, 136
  - features, 4
  - groups, 135
  - hostname configuration, 141
  - inventory sources, 135
  - jobs, 135
  - notifier, 135
  - notifier hierarchy, 135
  - notifier workflow, 135
  - organizations, 26
  - resetting the TOWER\_URL\_BASE, 141
  - template, 136
  - troubleshooting TOWER\_URL\_BASE, 141
  - types, 136
  - types Email, 136
  - types Hipchat, 136
  - types IRC, 136
  - types pagerduty, 136
  - types Slack, 136
  - types Twilio, 136
  - types Webhook, 136
- notifier
  - notifications, 135
- notifier hierarchy
  - notifications, 135
- notifier workflow
  - notifications, 135

## O

- OpenStack
  - cloud credentials, 101
  - credential types, 55
  - inventories, 77
- OpenStack inventory support
  - features, 3
- ordering
  - sorting, 20
- organization
  - settings menu, 14
  - summary, 26
- organizations, 22
  - notifications, 26
  - users, 24, 30

- overview
  - features, 2

## P

- pagerduty
  - notifications types, 136
- permissions
  - teams, 38
  - users, 31
- playbook
  - custom fact scans, 116
  - fact scan job, 115
- playbook run
  - jobs results, 128
- playbooks
  - manage manually, 59
  - projects, 59
  - PRoot settings, 148
  - sharing access, 148
  - sharing content, 148
  - source control, 59
- portal mode, 15
  - job templates, 15
  - jobs, 16
- projects, 57
  - add new, 58
  - new schedule addition, 62
  - playbooks, 59
  - source control update, 60
- PRoot
  - functionality, 149
  - troubleshooting, 149
  - variables, 149
- PRoot settings
  - playbooks, 148
- provisioning callbacks
  - extra variables, 118
  - job templates, 117

## R

- Rackspace
  - cloud credentials, 102
- rackspace
  - credential types, 50
- Rackspace Cloud Servers
  - inventories, 69
- RBAC
  - security, 149
- real-time playbook
  - features, 2
- Red Hat CloudForms
  - inventories, 76
- Red Hat Satellite
  - credential types, 52



- Red Hat Satellite 6
  - inventories, 75
- relaunch
  - job templates, 106
- remote command execution
  - features, 3
- resetting the TOWER\_URL\_BASE
  - notifications, 141
- RESTful API
  - features, 3
- role-based access control
  - features, 2
- role-based access controls, 149
- RPM files
  - licenses, 7
- run
  - job types, 94
- run-time job customization
  - features, 4
- S**
- scan
  - job types, 94
- scan job
  - add new, 107
  - custom, 115
  - host to host, 90
  - inventories, 87
  - launching, 112
  - scheduling, 114
  - single host, 88
- scan job templates, creating
  - inventories, 107
- scan job templates, custom
  - inventories, 115
- scan job templates, launching
  - inventories, 112
- scan job templates, scheduling
  - inventories, 114
- schedule status
  - dashboard, 16
- scheduling
  - add new, 121
  - inventories, 79
  - job templates, 120, 121
  - scan job, 114
- scheduling, add new
  - inventories, 80
- SCM
  - jobs results, 127
- searching, 19
- security, 148
  - RBAC, 149
- settings menu
  - about Tower, 14
  - configure Tower, 14
  - credentials, 14
  - inventory scripts, 14
  - management jobs, 14
  - organization, 14
  - teams, 14
  - users, 14
  - view license, 14
- sharing access
  - playbooks, 148
- sharing content
  - playbooks, 148
- single host
  - scan job, 88
  - system tracking, 88
- Slack
  - notifications types, 136
- sorting
  - ordering, 20
- source control
  - best practices, 146
  - credential types, 48
- source control update
  - projects, 60
- summary
  - organization, 26
- support, 4–6
- survey extra variables
  - job templates, 105
  - workflows, 144
- surveys
  - creation, 103
  - extra variables, 105, 144
  - job templates, 103
  - optional questions, 105
- system tracking
  - custom fact scans, 116
  - fact scan playbook, 115
  - features, 4
  - host to host, 90
  - inventories, 87
  - scan job, 94
  - single host, 88
- T**
- teams, 35
  - permissions, 38
  - settings menu, 14
  - users, 31, 37
- template
  - notifications, 136
- Tower admin menu, 13
- Tower settings menu, 14

- trial, 6
- troubleshooting
  - license, 10
  - PRoot, 149
- troubleshooting TOWER\_URL\_BASE
  - notifications, 141
- Twilio
  - notifications types, 136
- types
  - Email, notifications, 136
  - Hipchat, notifications, 136
  - IRC, notifications, 136
  - notifications, 136
  - pagerduty, notifications, 136
  - Slack, notifications, 136
  - Twilio, notifications, 136
  - Webhook, notifications, 136

## U

- updates, 6
- user menu, 15
- users, 27
  - organizations, 24, 30
  - permissions, 31
  - settings menu, 14
  - teams, 31, 37

## V

- variable inventory management
  - best practices, 147
- variable precedence, 106
- variables
  - PRoot, 149
- view license
  - settings menu, 14
- VMware
  - cloud credentials, 102
  - credential types, 51
- VMware vCenter
  - inventories, 74

## W

- Webhook
  - notifications types, 136
- workflows, 143
  - survey extra variables, 144