

---

# **Ansible Tower User Guide**

*Release Ansible Tower 2.4.5*

**Red Hat, Inc.**

**Jun 06, 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	Role Based Access Control and Auditing . . . . .	2
1.4	Cloud & Autoscaling Flexibility . . . . .	2
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 . . . . .	3
<b>2</b>	<b>Tower Licensing, Updates, and Support</b>	<b>4</b>
2.1	Support . . . . .	4
2.2	Trial Licenses . . . . .	4
2.3	License Types . . . . .	4
2.4	Node Counting in Licenses . . . . .	5
2.5	License Features . . . . .	5
2.6	Tower Component Licenses . . . . .	6
<b>3</b>	<b>Logging In</b>	<b>7</b>
<b>4</b>	<b>Exploring the Dashboard and Tower Interface</b>	<b>8</b>
4.1	Tower User Menu . . . . .	8
4.2	Setup Menu . . . . .	9
4.3	Portal Mode . . . . .	10
4.4	The Dashboard . . . . .	11
4.5	Activity Streams . . . . .	13
<b>5</b>	<b>Organizations</b>	<b>14</b>
5.1	Organizations - Users . . . . .	16
5.2	Organization - Administrators . . . . .	18
<b>6</b>	<b>Users</b>	<b>20</b>
6.1	Users - Credentials . . . . .	22
6.2	Users - Permissions . . . . .	24
6.3	Users - Admin of Organizations . . . . .	26
6.4	Users - Organizations . . . . .	26
6.5	Users - Teams . . . . .	27
<b>7</b>	<b>Teams</b>	<b>28</b>

7.1	Teams - Credentials	29
7.2	Teams - Permissions	31
7.3	Teams - Projects	33
7.4	Teams - Users	34
<b>8</b>	<b>Credentials</b>	<b>36</b>
8.1	Understanding How Credentials Work	36
8.2	Getting Started with Credentials	36
8.3	Add a New Credential	37
8.4	Credential Types	38
<b>9</b>	<b>Projects</b>	<b>47</b>
9.1	Add a new project	48
9.2	Updating projects from source control	50
9.3	Ansible Galaxy Support	52
9.4	Add a new schedule	52
<b>10</b>	<b>Inventories</b>	<b>54</b>
10.1	Add a new inventory	55
10.2	Scan Job Templates	56
10.3	Groups and Hosts	63
10.4	Running Ad Hoc Commands	77
10.5	System Tracking	79
<b>11</b>	<b>Job Templates</b>	<b>85</b>
11.1	Utilizing Cloud Credentials	88
11.2	Surveys	90
11.3	Provisioning Callbacks	93
11.4	Launching Jobs	95
11.5	Scheduling	96
<b>12</b>	<b>Jobs</b>	<b>99</b>
12.1	Job Results	99
12.2	Job Status	101
12.3	Job Concurrency	104
<b>13</b>	<b>Best Practices</b>	<b>106</b>
13.1	Use Source Control	106
13.2	Ansible file and directory structure	106
13.3	Use Dynamic Inventory Sources	106
13.4	Variable Management for Inventory	107
13.5	Autoscaling	107
13.6	Larger Host Counts	107
13.7	Continuous integration / Continuous Deployment	107
<b>14</b>	<b>Security</b>	<b>108</b>
14.1	Playbook Access and Information Sharing	108
14.2	PRoot functionality and variables	109
14.3	Role-Based Access Controls	109
<b>15</b>	<b>Index</b>	<b>115</b>
<b>16</b>	<b>Copyright © 2016 Red Hat, Inc.</b>	<b>116</b>
	<b>Index</b>	<b>117</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 2.4.5.

Ansible Tower Version 2.4.5; June 2, 2016; <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 Role Based Access Control and Auditing**

Ansible Tower allows delegating specific authority to different teams or explicit users. Keep some projects private. Allow some users to edit inventory and others to run playbooks against only certain systems—either in check (dry run) or live mode. 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.

### **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.

## TOWER LICENSING, UPDATES, AND SUPPORT

Tower is a proprietary software product and is licensed on an annual subscription basis.

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

Ansible offers support for paid Enterprise customers seeking help with the Tower product. If you or your company has paid for a license of Ansible Tower, you can contact Ansible via the Red Hat Customer Portal at <https://access.redhat.com/>. To better understand the levels of support which match your Tower license, refer to *License Types*.

If you are using Ansible core and are having 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>

### 2.2 Trial Licenses

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

Trial licenses for Tower are available at: <http://ansible.com/license>

To acquire a license for additional servers, visit: <http://www.ansible.com/pricing/>

### 2.3 License Types

Tower is licensed at various levels as an annual subscription. Whether you have a small business or a mission-critical environment, Ansible is ready to simplify your IT work-flow.

- **Self-Support**
  - Manage smaller environments (up to 250 nodes)
  - Maintenance and upgrades included
- **Enterprise: Standard**
  - Manage any size environment
  - Enterprise 8x5 support and SLA (4 hour critical incident response)

- Phone and web support
- Maintenance and upgrades included
- **Enterprise: Premium**
  - Manage any size environment, including mission-critical environments
  - Premium 24x7 support and SLA (4 hour critical incident response, 8 hour non-critical incident response)
  - Phone and web support
  - Maintenance and upgrades included

All subscriptions include regular updates and releases of both Ansible Tower and Ansible core.

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 nodes that can be managed by Tower. A typical license will say ‘Enterprise Tower Up To 250 Nodes’, which sets the maximum number of nodes that can be managed at 250.

Tower counts nodes by the number of hosts in inventory. If more nodes are in the Tower inventory than are supported by the license, you will be unable to start any Jobs in Tower. If a dynamic inventory sync will cause Tower to exceed the 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

---

**Note:** Ansible Tower version 2.2 introduced a separation of features for Basic versus Enterprise or Premium licenses.

---

The following list of features are available for all new Enterprise or Premium license users:

- Custom rebranding 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 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

Ansible Tower includes some open source components. Ansible, Inc. supports Tower's use of and interactions with these components for both development and production purposes, subject to applicable terms and conditions. Unless otherwise agreed to in writing, the use of Ansible Tower is subject to the Ansible Software Subscription and Services Agreement located at <http://www.ansible.com/subscription-agreement>. Ansible Tower is a proprietary product offered by Ansible, Inc. and its use is not intended to prohibit the rights under any open source license.

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>/`



Log in using a valid Tower username and password.

**Note:** 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 Setup (  ) Menu.

---

## EXPLORING THE DASHBOARD AND TOWER INTERFACE

---

**Note:** Ansible Tower 2.2 provides a streamlined interface, with the setup button offering access to administrative configuration needs. Users of older versions of Ansible Tower (pre-2.2) can access most of these through the top-level navigational menu.

---



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, permissions, and more.


On the main Tower Dashboard screen, a summary appears listing your current *Hosts*, *Inventories*, and *Projects*. You can view charts and graphs for **Job Status** and **Host Status** by clicking on their tabs. Also available for review are summaries of **Recent Used Job Templates** and **Recent Run Jobs**.

---

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

---

### 4.1 Tower User Menu

The Tower user menu is accessed by clicking  admin .

From here, you can:

- Edit the properties of the Tower user account and view the activity stream for that user
- Add credentials to the Tower user account
- Add and setup permission types for your inventories for your users (read/write/admin as well as the ability execute commands if not an admin)
- Review the Admin of Organizations
- Review the organizations which have been setup for the Tower user
- Review the teams to which the Tower user has been added

**TOWER** Projects Inventories Job Templates Jobs admin

Setup > Users > admin

**Properties**

\* First Name

\* Last Name

\* Email

\* Username

**Password**  Show

**Confirm Password**  Show

Superuser (User has full system administration privileges.)

Created by LDAP

[Credentials](#)


[Permissions](#)

[Admin of Organizations](#)


[Organizations](#)


[Teams](#)


## 4.2 Setup Menu

To enter the Setup Menu screen for Ansible Tower, click the  button. This screen allows you to create your organization, add credentials, add users and teams, schedule management jobs, and more. You can also view your license from the Setup Menu's 'View License' link.

**TOWER** Projects Inventories Job Templates Jobs admin

 **Credentials**  
Add passwords, SSH keys, etc. for Tower to use when launching jobs against machines, or when syncing inventories or projects.

 **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.

**Organizations**  
Group all of your content to manage permissions across departments in your company.


**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.

**View Your License**


**About Tower**

## 4.3 Portal Mode

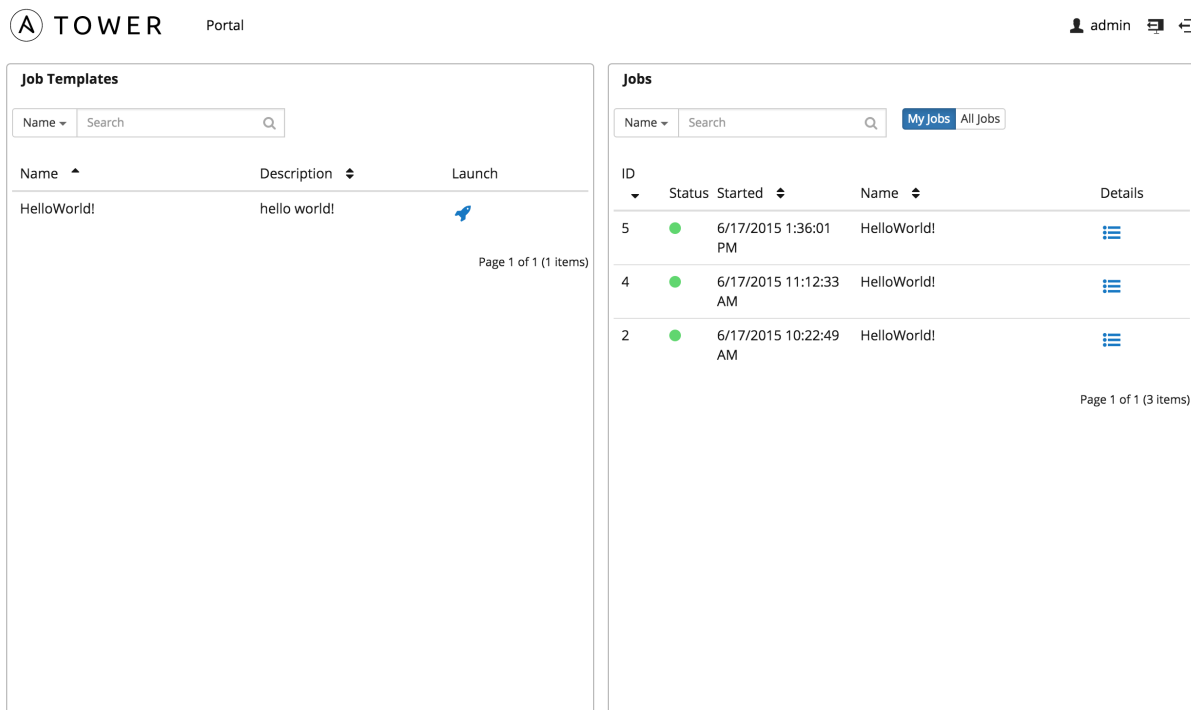
Portal mode, a single-page view of jobs and job templates, can be accessed by clicking the  button.

Portal mode is a simplified interface for users that need to run Ansible jobs, but that don't need an advanced knowledge of Ansible or Tower. Portal mode could be used by, for instance, development teams, or even departmental users in non-technical fields.


Portal mode 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 portal mode launches it, potentially asking some survey questions.

Other portions of the interface are hidden from view until portal mode is exited.




Portal mode can be accessed in two ways:

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

In Portal mode, the top bar of Tower only has the Tower user button, an **Exit Portal** button to exit to the main interface, and the **Logout** button. Portal mode displays two main sections: *Job Templates* and *Jobs*

### 4.3.1 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.

---

### 4.3.2 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.

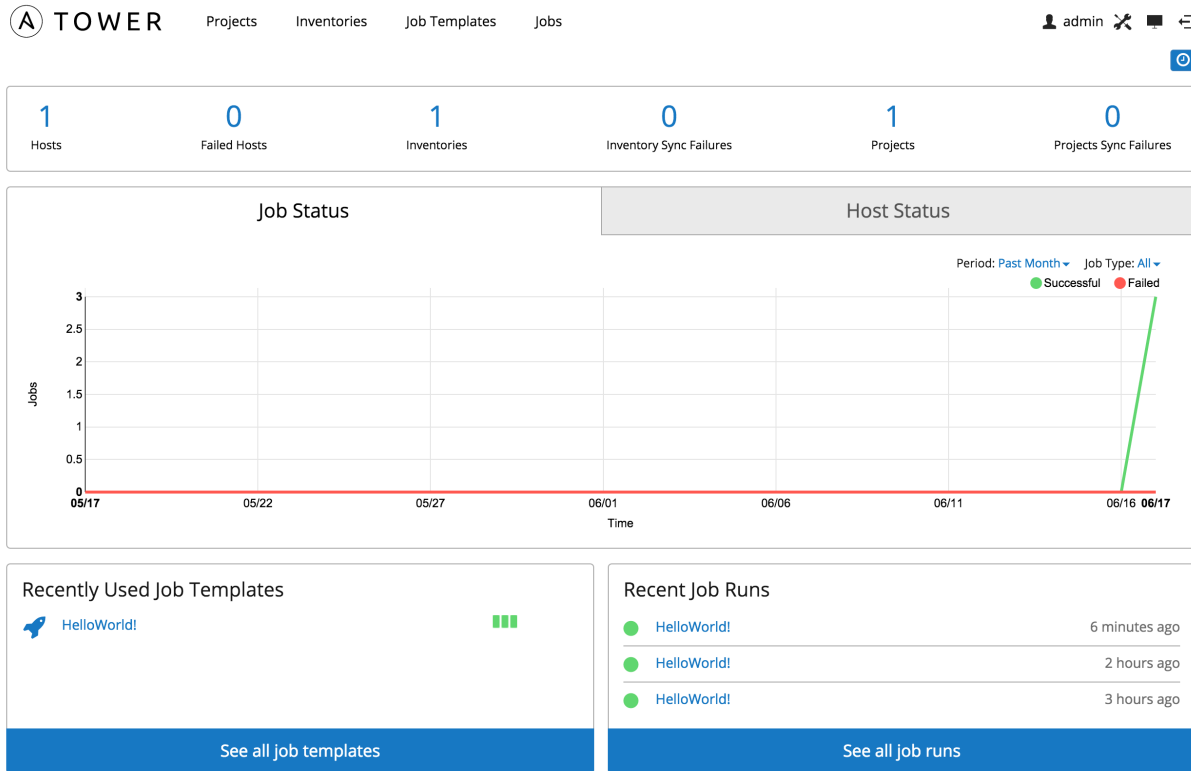
- **My Jobs:** View jobs that you (as the user) ran .
- **All Jobs:** View your team members' completed jobs, viewable based on your RBAC permissions.

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).

## 4.4 The Dashboard

The central interface to Tower is the Dashboard.

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.



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.

The Dashboard contains four graphs.

#### 4.4.1 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.

#### 4.4.2 Host Status

The Host Status graph displays, as of the most recent job run, how many of the configured hosts in your inventory have been marked as ‘Successful’.


#### 4.4.3 Recently Used Job Templates

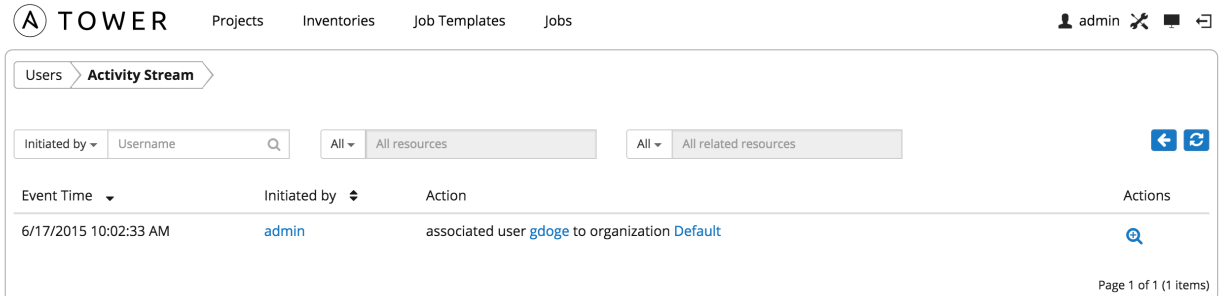
The Jobs section of this display shows a summary of the most recently used jobs. You can also access this summary by clicking on the **Jobs** entry in the main navigation menu.

#### 4.4.4 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.

## 4.5 Activity Streams

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




Users > Activity Stream

Initiated by Username   All All resources All All related resources

Event Time	Initiated by	Action	Actions
6/17/2015 10:02:33 AM	admin	associated user <a href="#">gdoge</a> to organization <a href="#">Default</a>	<input type="button" value="Q"/>

Page 1 of 1 (1 items)

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  button shows the event log for the change.

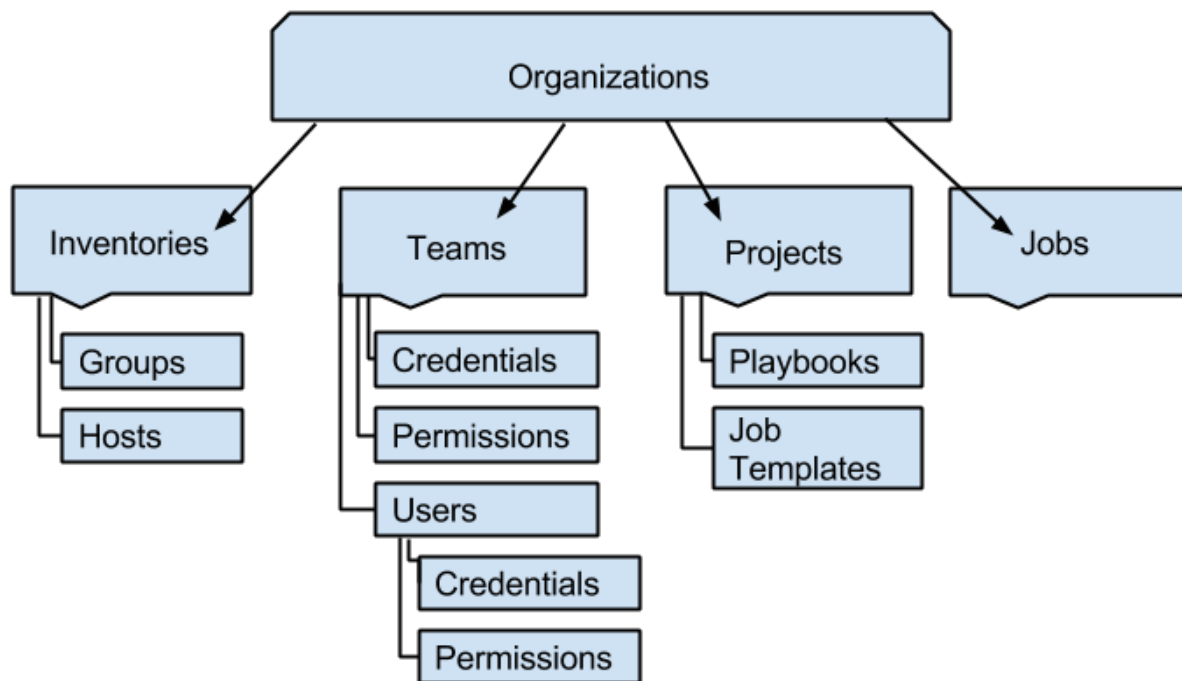
The Activity Stream can be filtered by the initiating user (or the system, if it was system initiated), and by any related Tower object, such as a particular credential, job template, or schedule.


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.



## ORGANIZATIONS

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

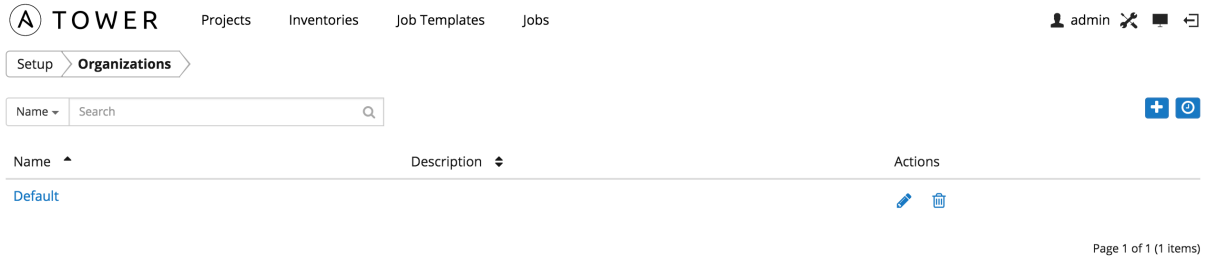


The **Organizations** link from the Setup (  ) menu displays all of the existing organizations for your installation of Tower. Organizations can be searched by **Name** or **Description**. Modify and remove organizations using the **Edit** and **Delete** buttons.

---


**Note:** Starting with version 2.2.0, Tower creates a default organization automatically. Users of older versions of Tower will not see this default organization. Users of Tower with a Basic-level license only have the default organization available and should **not** delete it.

---



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

- Create a new organization
- View Activity Stream

Create a new organization by selecting the  button.

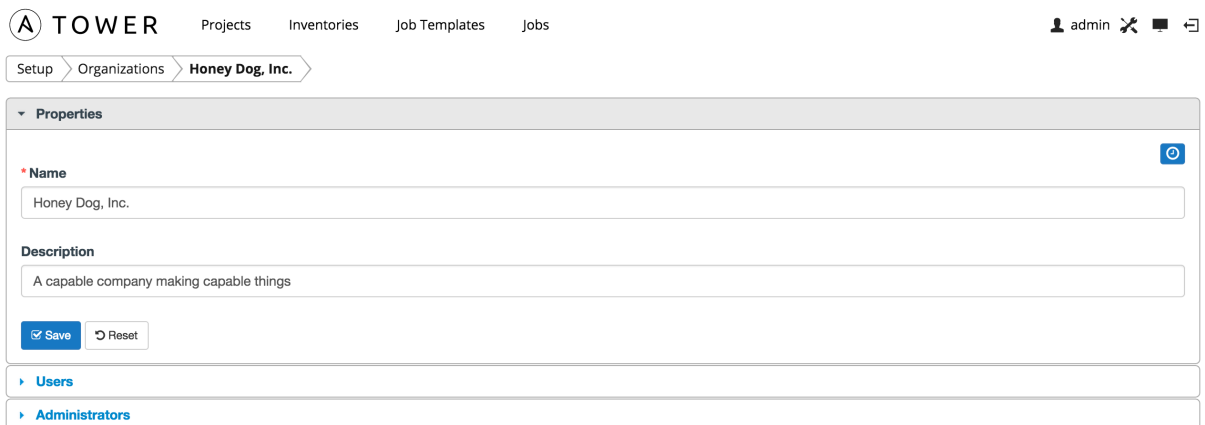
**Note:** If you are using Ansible Tower with a Basic license, you must use the default organization. Do not delete it and try to add a new organization, or you will break your Tower setup. Only Enterprise or Premium Tower licenses have the ability to add new organizations beyond the default.

1. Enter the **Name** for your organization.
2. Optionally, enter a **Description** for the organization.

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

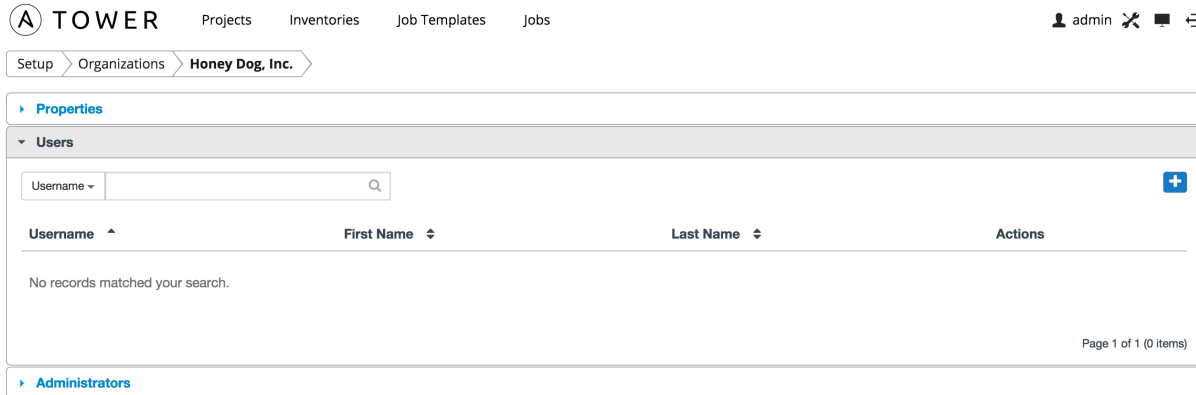



Once created, Tower displays the organization details, including two accordion-style menus below the organization name and description details, that provide for managing users and administrators for the organization.




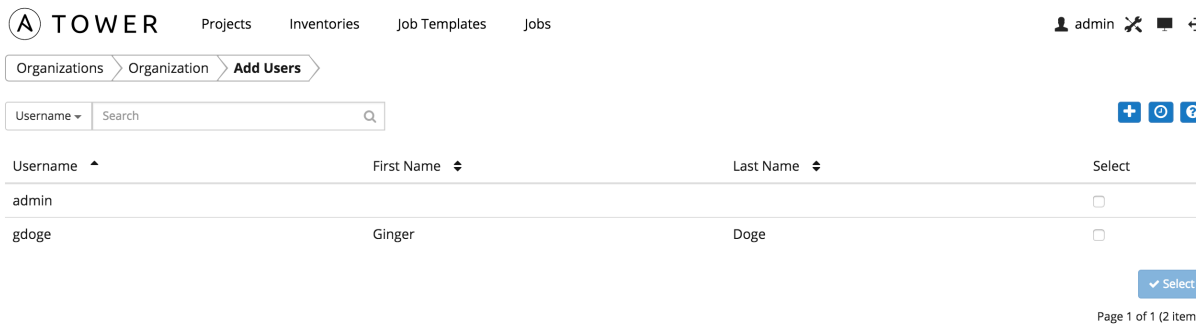
## 5.1 Organizations - Users

The **Users** menu of an Organization displays all the Users associated with this organization. A user is someone with access to Tower with associated permissions and credentials. Expand the users menu by selecting **Users**.



This menu allows you to manage the user membership for this organization. (User membership may also be managed on a per-user basis via the **Users** link available from the Setup  menu.) The user list may be sorted and searched by **Username**, **First Name**, or **Last Name**. Existing users may also be modified and removed using the **Edit** and **Delete** buttons. Clicking on a user brings up that user's details, which can then be edited. For more information, refer to *Users*.

To add existing users to the organization, click the  button. Then, select one or more users from the list of available users by clicking the **Select** checkbox or clicking anywhere on the user row. Click the **Select** button when done.



To create a new user and add it to the organization, click the  button from the **Add Users** screen, which takes you to the new user dialog.

**A TOWER** Projects Inventories Job Templates Jobs admin

---

**\* First Name**

**\* Last Name**

**\* Email**

**\* Organization**

**\* Username**

**\* Password**

**Confirm Password**

Superuser (User has full system administration privileges.)

Enter the appropriate details into the following fields:

- First Name
- Last Name
- Email
- Organization (prefilled with the current organization—or the default organization if you are using a Basic license)
- Username
- Password
- Confirm Password
- Superuser (Gives this user full system administration privileges. Set with caution!)

**TOWER** Projects Inventories Job Templates Jobs admin

**\* First Name**

**\* Last Name**

**\* Email**

**\* Organization**

**\* Username**

**\* Password**

**Confirm Password**

Superuser (User has full system administration privileges.)

All of these fields are required. Select **Save** when finished and the user is added to the organization.

**TOWER** Projects Inventories Job Templates Jobs admin

Setup > Organizations > **Honey Dog, Inc.**

**Properties**

**Users**

Username

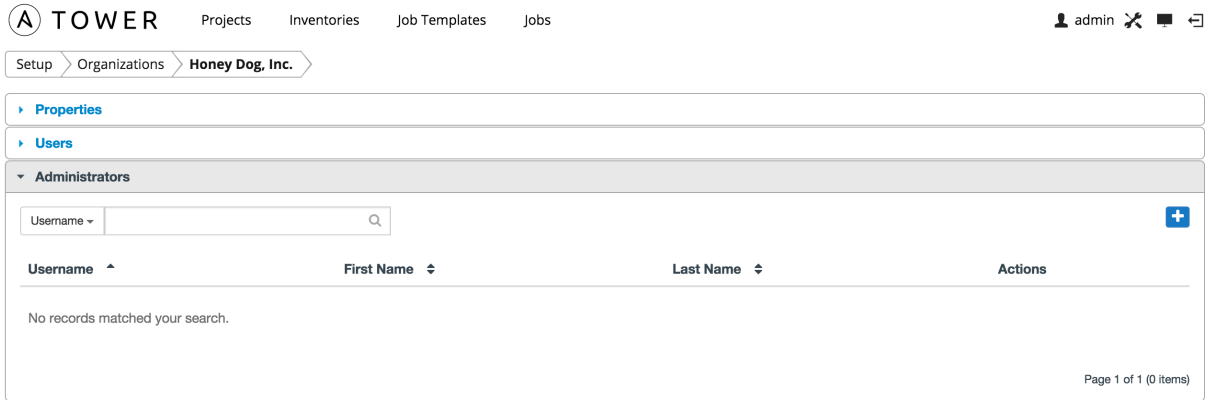
Username	First Name	Last Name	Actions
jdoge	Josie	Doge	<input type="button" value="edit"/> <input type="button" value="delete"/>

Page 1 of 1 (1 Items)

**Administrators**

## 5.2 Organization - Administrators

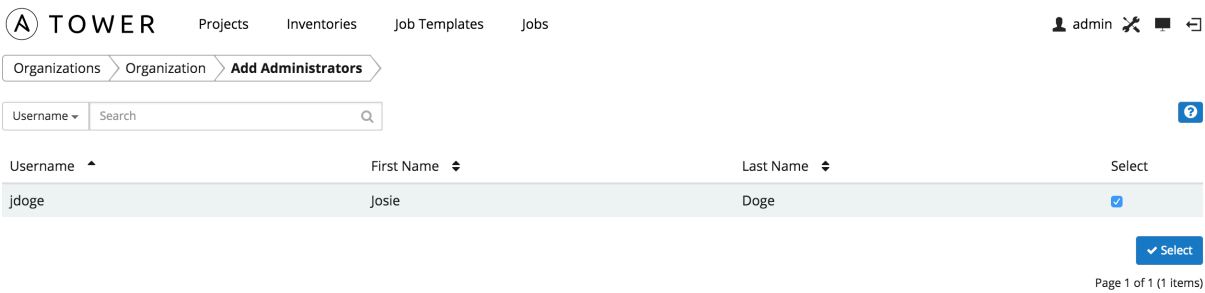
An organization administrator is a type of user that has the rights to create, modify, or delete objects in the organization, including projects, teams, and users in that organization. Expand the **Administrators** menu by selecting **Administrators**.



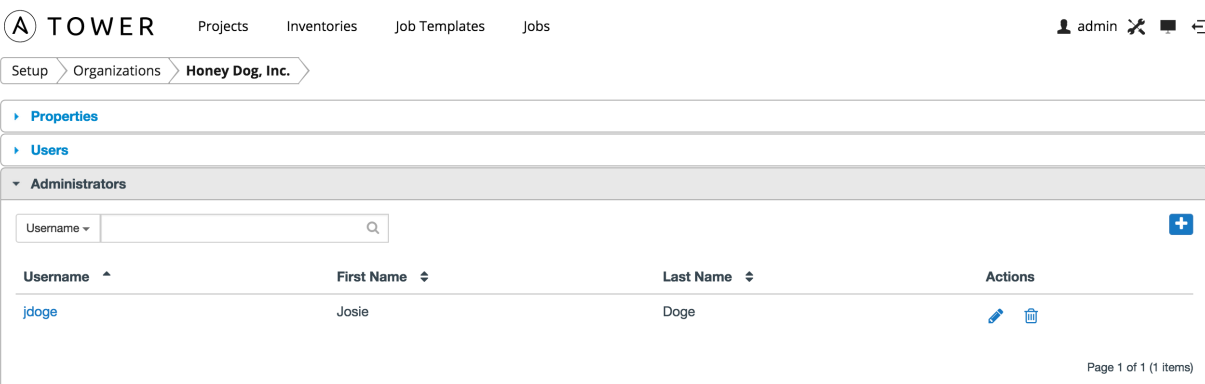
This menu displays a list of the users that are currently setup as an administrator of the organization. The administrator list may be sorted and searched by **Username**, **First Name**, or **Last Name**.

**Note:** Any user marked as a ‘Superuser’ is implicitly an administrator of all organizations, and is not displayed here.


To add an administrator to the organization, click the  button. Select one or more users from the list of available users by clicking the **Select** checkbox or clicking anywhere on the user row. Click the **Select** button when done.

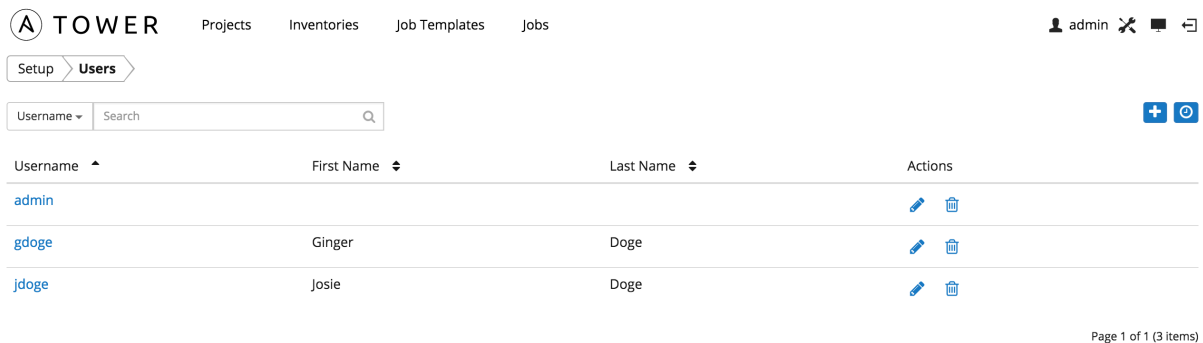








**Note:** A user must first be added to the organization before it can be added to the list of administrators for that organization.



## USERS

A user is someone who has access to Tower with associated permissions and credentials. The Users link (accessible from the Setup [  ] menu) allows you to manage the all Tower users. The user list may be sorted and searched by **Username**, **First Name**, or **Last Name**.



Username	First Name	Last Name	Actions
admin			 
gdoge	Ginger	Doge	 
jdoge	Josie	Doge	 

There are three types of Tower Users:

1. **Normal User:** read and write access is limited to the inventory and projects for which that user has been granted the appropriate rights.
2. **Organization Administrator:** the administrator of an organization has all of the rights of a normal user, as well as admin, read, and write permission over the entire organization and all of its inventories and projects, but does not have those levels of access on content belonging to other organizations. This level of user can create and manage users.
3. **Superuser:** a Tower Superuser has admin, read, and write permissions over the entire Tower installation. A Superuser is typically a systems administrator responsible for managing Tower and will delegate responsibilities for day-to-day work to various Organization Administrators.

---

**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, first create another Superuser account.

---

To create a new user click the  button, which opens the new user dialog.

The screenshot shows the 'Create User' form in Ansible Tower. At the top, there is a navigation bar with 'TOWER' and links for 'Projects', 'Inventories', 'Job Templates', and 'Jobs'. On the right, there is a user profile for 'admin' with icons for settings, notifications, and help. Below the navigation is a breadcrumb trail: 'Setup > Users > Create User'. The form itself contains several input fields:
 

- \* First Name: A text input field.
- \* Last Name: A text input field.
- \* Email: A text input field.
- \* Organization: A text input field with a search icon (magnifying glass) on the left.
- \* Username: A text input field.
- \* Password: A text input field with a 'Show' button on the left.
- Confirm Password: A text input field with a 'Show' button on the left.


 At the bottom of the form, there is a checkbox labeled 'Superuser (User has full system administration privileges.)' and two buttons: 'Save' (with a checkmark icon) and 'Reset' (with a circular arrow icon).

Enter the appropriate details into the following fields:

- First Name
- Last Name
- Email
- Organization (Choose from an existing organization–this is the default organization if you are using a Basic license)
- Username
- Password
- Confirm Password
- Superuser (The superuser has full system administration privileges for Tower. Use with caution!)

Select **Save** when finished.

Once the user is successfully created, the **Edit User** dialog opens. This is the same menu that is opened if the **Edit**

button is clicked from the **Users** link from the  button. Here, the user's **Properties**, **Credentials**, **Permissions**, and other user membership details may be reviewed and modified.



**TOWER** Projects Inventories Job Templates Jobs admin

Setup > Users > **ldoge**

**Properties**

\* **First Name**

\* **Last Name**

\* **Email**

\* **Username**

**Password**

**Confirm Password**

Superuser (User has full system administration privileges.)

Created by LDAP

[Credentials](#)

[Permissions](#)

[Admin of Organizations](#)

[Organizations](#)

[Teams](#)

## 6.1 Users - Credentials

Credentials are utilized by Tower for authentication when launching jobs against machines, for synchronization with inventory sources, and when importing project content from version control systems. For more information, refer to *Credentials*.

**TOWER** Projects Inventories Job Templates Jobs admin

Setup > Users > **ldoge**

[Properties](#)

**Credentials**

Name

Name	Description	Actions
No records matched your search.		


Page 1 of 1 (0 items)

[Permissions](#)

[Admin of Organizations](#)

[Organizations](#)

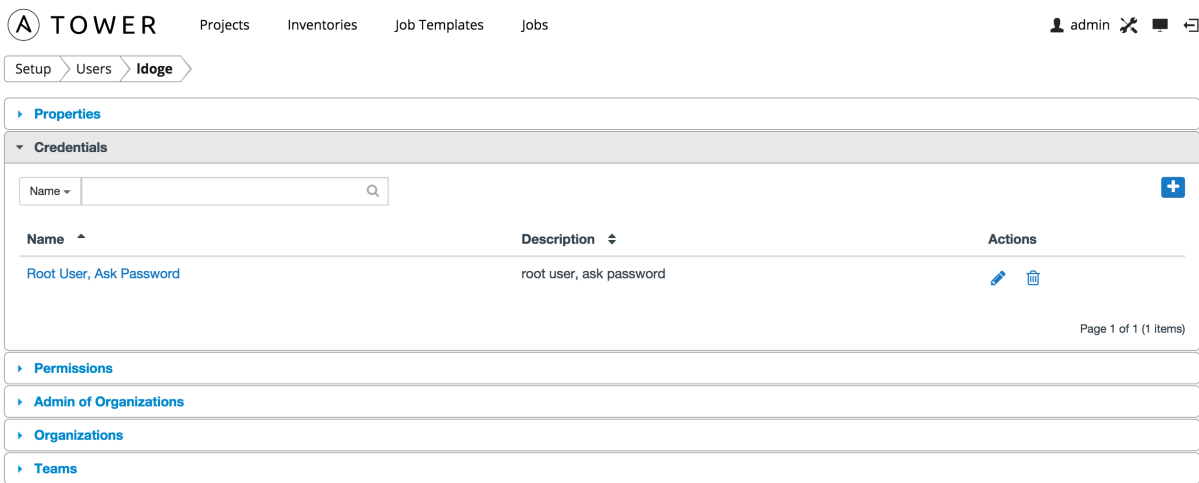
[Teams](#)

To add a credential to user, expand the credentials menu and click the  button.




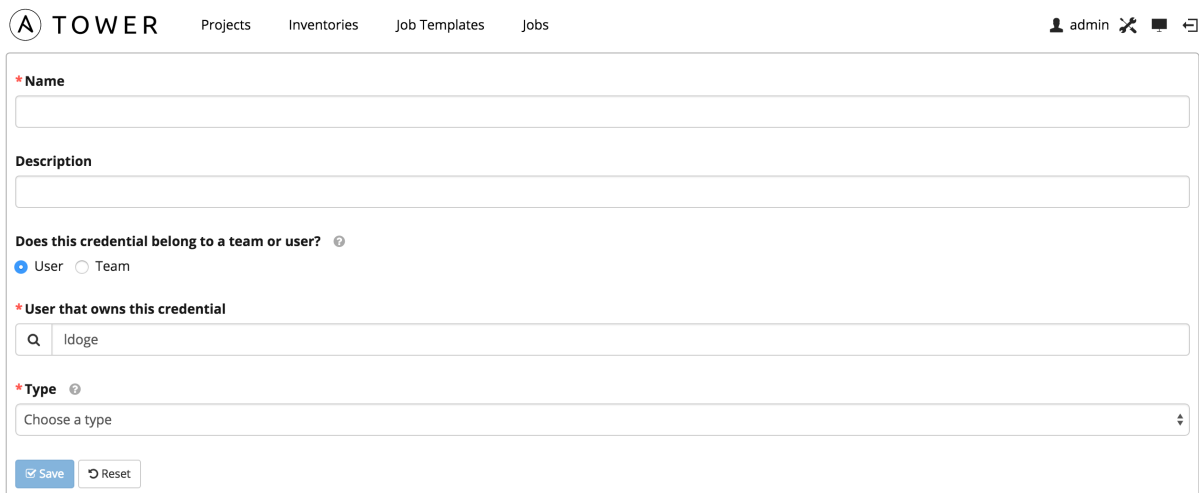
The screenshot shows the 'Add Credentials' interface. At the top, there's a navigation bar with 'TOWER' and 'Users' selected. Below that, there's a search bar and a table of credentials. The table has columns for Name, Description, Type, and Select. One credential is listed: 'Root User, Ask Password' with description 'root user, ask password' and type 'Machine'. A 'Select' checkbox is checked for this credential. A 'Select' button is at the bottom right.

Then, select one or more credentials from the list of available credentials by clicking the **Select** checkbox. Click the **Select** button when done.



The screenshot shows the 'Idoge' user profile page. At the top, there's a navigation bar with 'TOWER' and 'Users' selected. Below that, there's a search bar and a list of credentials. The list has columns for Name, Description, and Actions. One credential is listed: 'Root User, Ask Password' with description 'root user, ask password'. There are edit and delete icons in the Actions column. A 'Select' checkbox is checked for this credential.

To create a new credential and add it to the user, click the  button from the **Add Credentials** screen, which opens the **Create Credential** dialog.



The screenshot shows the 'Create Credential' dialog. It has fields for Name, Description, Does this credential belong to a team or user? (User selected), User that owns this credential (Idoge), and Type (Choose a type). There are Save and Reset buttons at the bottom.

Enter the appropriate details depending on the type of credential and select **Save**. For more information, refer to

*Credentials.*

## 6.2 Users - Permissions

The set of privileges assigned to users and teams (role-based access control) that provide the ability to read, modify, and administer projects, inventories, job templates, and other Tower elements are permissions.

There are two permission types available to be assigned to users and teams, each with its own set of permissions available to be assigned:

- **Inventory:** grants permission to act on inventories, groups, and hosts
  - **Read:** view groups and hosts within a specified inventory
  - **Write:** create, modify, and remove groups, and hosts within a specified inventory. Does not give permission to modify the inventory settings. This permission also grants the Read permission.
  - **Admin:** modify the settings for the specified inventory. This permission also grants Read and Write permissions.
  - **Execute commands:** Allow the user to execute commands on the inventory.
- **Job Template:** grants permission to launch jobs from the specified project against the specified inventory
  - **Create:** Allow the user or team to create job templates. This implies that they have the Run and Check permissions
  - **Run:** launch jobs of type Run. This permission also grants the Check permission.
  - **Check:** launch jobs of type Check.

This menu displays a list of the permissions that are currently available. The permissions list may be sorted and searched by **Name**, **Inventory**, **Project**, or **Permission** type.

The screenshot shows the Ansible Tower web interface. At the top, there is a navigation bar with 'TOWER' and links for 'Projects', 'Inventories', 'Job Templates', and 'Jobs'. A user profile for 'admin' is visible in the top right. Below the navigation, a breadcrumb trail shows 'Setup > Users > Idoge'. The main content area is titled 'Permissions' and contains a search bar with the text 'Name' and a magnifying glass icon. Below the search bar is a table with columns: 'Name', 'Inventory', 'Project', 'Permission', and 'Actions'. The table currently displays 'No records matched your search.' and 'Page 1 of 1 (0 items)'. There is a blue '+' button in the top right corner of the table area. Below the table, there are links for 'Admin of Organizations', 'Organizations', and 'Teams'.

To add new permissions to the user, click the  button, which opens the **Add Permission** dialog.

**\* Permission Type**  
 Inventory  Job Template

**\* Name**

**Description**

**\* Inventory**

**\* Permission**  
 Read  Write  Admin  
 Execute commands

**Permission**

**Read**  
 Only allow the user or team to view the inventory.

**Write**  
 Allow the user or team to modify hosts and groups contained in the inventory, add new hosts and groups, and perform inventory sync operations.

**Admin**  
 Allow the user or team full access to the inventory. This includes reading, writing, deletion of the inventory, inventory sync operations, and the ability to execute commands on the inventory.

**Execute commands**  
 Allow the user to execute commands on the inventory.

Enter the appropriate details into the following fields:

- Permission Type
  - Inventory
  - Job Template
- Name
- Description

**Note:** Before you can select an Inventory, you must first create it and make it available. Refer to *Inventories* for more information.

Selecting a **Permission Type** of either **Inventory** or **Job Template** changes the appearance of the **Add Permission** dialog to present appropriate options for each type of permission.

For a permission of type **Inventory**, enter the following details:

- Inventory (Select from the available inventories)
- Permission
  - Read
  - Write
  - Admin
  - Execute commands

For a permission of type **Job Template**, enter the following details:

- Project (Select from the available projects)
- Inventory (Select from the available inventories)
- Permission
  - Create
  - Run
  - Check

Select **Save**.





## 6.3 Users - Admin of Organizations

This displays the list of organizations that this user is an administrator of. This list may be searched by **Organization Name** or **Description**. A user cannot be made an organization administrator from this interface panel.

The screenshot shows the Ansible Tower web interface. At the top, there is a navigation bar with 'TOWER' and links for 'Projects', 'Inventories', 'Job Templates', and 'Jobs'. The user 'admin' is logged in. The breadcrumb trail is 'Setup > Users > Idoge'. The 'Admin of Organizations' section is expanded, showing a search bar with 'Name' selected. Below the search bar is a table with columns 'Name', 'Description', and 'Actions'. The table is currently empty, displaying 'No records matched your search.' and 'Page 1 of 1 (0 items)'.

## 6.4 Users - Organizations

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


**TOWER**
Projects
Inventories
Job Templates
Jobs
admin   

Setup > Users > **Idoge**

- [Properties](#)
- [Credentials](#)
- [Permissions](#)
- [Admin of Organizations](#)
- [Organizations](#)
- [Teams](#)





Name

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

Page 1 of 1 (1 items)

## 6.5 Users - Teams

This displays the list of teams that this user is a member of. 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*.


**TOWER**
Projects
Inventories
Job Templates
Jobs
admin   

Setup > Users > **Idoge**

- [Properties](#)
- [Credentials](#)
- [Permissions](#)
- [Admin of Organizations](#)
- [Organizations](#)
- [Teams](#)

Name

Name	Description	Actions
No records matched your search.		

Page 1 of 1 (0 items)


## 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  button, 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:

- Create a new team
- View Activity Stream

The screenshot shows the Tower interface with the 'Teams' tab selected. The navigation bar includes 'TOWER', 'Projects', 'Inventories', 'Job Templates', and 'Jobs'. The user 'admin' is logged in. The 'Teams' tab is active, and a search bar is visible with the text 'No records matched your search.' The table headers are 'Name', 'Description', 'Organization', and 'Actions'. There are two buttons in the top right corner: a blue plus sign and a blue circle with a slash.

Page 1 of 1 (0 items)



To create a new team, click the  button.

The screenshot shows the 'Create Team' form in Ansible Tower. At the top, there is a navigation bar with 'TOWER' logo and links for 'Projects', 'Inventories', 'Job Templates', and 'Jobs'. A user profile 'admin' is visible in the top right. Below the navigation, a breadcrumb trail shows 'Setup > Teams > Create Team'. The form itself has three main sections:
 

- Name:** A text input field containing 'Production Operations'.
- Description:** A text input field containing 'prod ops team'.
- Organization:** A dropdown menu with a search icon and the text 'Honey Dog, Inc.'.

 At the bottom of the form, there are two buttons: a blue 'Save' button and a 'Reset' button.

Enter the appropriate details into the following fields:

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

Select **Save**.

Once the team is successfully created, Tower opens the **Edit Team** dialog. This is the same menu that is opened if the **Edit** button is clicked from the **Teams** link. Here, **Team Settings**, **Credentials**, **Permissions**, **Projects**, and **Users** associated with this team may be reviewed and modified.

The screenshot shows the 'Edit Team' dialog in Ansible Tower for the team 'Production Operations'. The layout is similar to the 'Create Team' form, but with additional features:
 


- The breadcrumb trail is 'Setup > Teams > Production Operations'.
- The form fields for Name, Description, and Organization are identical to the previous screenshot.
- Below the form, there are four expandable sections: 'Credentials', 'Permissions', 'Projects', and 'Users', each with a right-pointing arrow.
- The 'Save' and 'Reset' buttons are still present at the bottom of the form area.

## 7.1 Teams - Credentials


Credentials are utilized by Tower for authenticating when launching jobs against machines, to synchronize with inventory sources, and to import project content from a version control system. For details about how to use credentials, refer to [Credentials](#).



The screenshot shows the 'Production Operations' team page in Ansible Tower. The 'Credentials' section is expanded, showing a search bar and a table with columns for Name, Description, and Actions. The table is currently empty, displaying 'No records matched your search.'

To add credentials to the team, click the  button. Then, select one or more credentials from the list of available credentials by clicking the **Select** checkbox. Click the **Select** button when done.

The screenshot shows the 'Add Credentials' screen in Ansible Tower. It features a search bar and a table with columns for Name, Description, Type, and Select. One credential, 'Root User, Ask Password', is listed with a checked 'Select' checkbox. A 'Select' button is visible at the bottom right.

To create new credentials and add them to the team, click the  button from the **Add Credentials** screen.

The screenshot shows the 'Add Credentials' form in Ansible Tower. The form includes fields for Name, Description, a radio button to select 'Team' as the owner, a dropdown for 'Team that owns this credential' (set to 'Production Operations'), and a dropdown for 'Type'. 'Save' and 'Reset' buttons are at the bottom.

Enter the appropriate details depending on the type of credential and select **Save**. (For details about credential types, refer to [Credentials](#).)

## 7.2 Teams - Permissions

The set of privileges assigned to users and teams that provide the ability to read, modify, and administer projects, inventories, and other Tower elements are permissions.

There are two permission types available to be assigned to users and teams, each with its own set of permissions available to be assigned:

- **Inventory:** grants permission to act on inventories, groups, and hosts
  - **Read:** View groups and hosts within a specified inventory.
  - **Write:** Create, modify, and remove groups, and hosts within a specified inventory. Does not give permission to modify the inventory settings. This permission also grants the Read permission.
  - **Admin:** Modify the settings for the specified inventory. This permission also grants Read and Write permissions.
  - **Execute commands:** Allow the user to execute commands on the inventory.
- **Job Template:** grants permission to launch jobs from the specified project against the specified inventory
  - **Create:** Allow the user or team to create job templates. This implies that they have the Run and Check permissions.
  - **Run:** Launch jobs of type Run. This permission also grants the Check permission.
  - **Check:** Launch jobs of type Check.

This menu displays a list of the permissions that are currently available. The permissions list may be sorted and searched by **Name**, **Inventory**, **Project** or **Permission** type.

The screenshot shows the Ansible Tower web interface. At the top, there is a navigation bar with 'TOWER' and links for 'Projects', 'Inventories', 'Job Templates', and 'Jobs'. A user profile for 'admin' is visible on the right. Below the navigation bar, there is a breadcrumb trail: 'Setup > Teams > Production Operations'. The main content area is titled 'Permissions' and contains a search bar with the text 'Name' and a magnifying glass icon. Below the search bar is a table with the following columns: 'Name', 'Inventory', 'Project', 'Permission', and 'Actions'. The table is currently empty, displaying the message 'No records matched your search.' At the bottom right of the table, it says 'Page 1 of 1 (0 items)'. There is a blue '+ Add' button in the top right corner of the table area. Below the table, there are links for 'Projects' and 'Users'.

To add new permissions to the team, click the  button, which launches the **Add Permission** dialog.

Enter the appropriate details into the following fields:

- Permission Type
  - Inventory
  - Job Template
- Name
- Description

**Note:** Before you can select an Inventory, you must first create it and make it available. Refer to [Inventories](#) for more information.

Selecting a **Permission Type** of either **Inventory** or **Job Template** changes the appearance of the **Add Permission** dialog to present appropriate options for each type of permission.

For a permission of type **Inventory**, enter the following details:

- Inventory (Select from the available inventories)
- Permission
  - Read
  - Write
  - Admin

For a permission of type **Job Template**, enter the following details:

- Project (Select from the available projects)


- Inventory (Select from the available inventories)
- Permission
  - Create
  - Run
  - Check

Select **Save**.

## 7.3 Teams - Projects


This displays the list of projects that this team has access to. This list may be searched by **Project Name** or **Description**. Before a project can be added, it must first be created. For more information about projects, refer to *Projects*.

The screenshot shows the 'Production Operations' team page in Ansible Tower. The 'Projects' section is expanded, showing a search bar and a table with columns for Name, Description, and Actions. The table currently displays 'No records matched your search.'

To add an existing project to the team, click the  button. Then select one or more available projects from the list by clicking the **Select** checkbox or by clicking anywhere on the user row. Select **Finished** when done.

The screenshot shows the 'Add Project' screen in Ansible Tower. A table lists a project named 'Hello World!' with a 'Select' checkbox checked. A 'Select' button is visible at the bottom right.

Status	Name	Last Updated	Type	Select
<input type="checkbox"/>	Hello World!	6/18/2015 12:52:47 PM	Manual	<input checked="" type="checkbox"/>

To create a new project and it to the team, click the  button from the **Add Project** screen, which launches the **Create Project** dialog.

**TOWER** Projects Inventories Job Templates Jobs admin ✕ 📄 🏠

Teams > Team > Projects > **Create Project**

**\*Name**

**Description**

**\*Organization** ⓘ

**\*SCM Type**

Enter the appropriate details into the following fields:

- Name
- Description
- Organization
- SCM Type (Select one of Manual, Git, Subversion, or Mercurial.)
  - If the SCM type is Manual, select the Project Base Path and Playbook Directory.
  - If the SCM type is Git, Subversion, or Mercurial, select the SCM URL.
    - \* You can also enter optional information for the SCM branch, SCM credential, and preferred SCM update options.
    - \* Revision # (Subversion only–optionally enter the Revision # for Subversion.)

All fields are required. Select **Save**.

## 7.4 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*.

**TOWER** Projects Inventories Job Templates Jobs admin ✕ 📄 🏠

Setup > Teams > **Production Operations**

▶ Properties

▶ Credentials

▶ Permissions

▶ Projects


▼ **Users**





Username  +

Username	First Name	Last Name	Actions
No records matched your search.			



Page 1 of 1 (0 items)



To add users to the team, click the  button. Then, select one or more users from the list of available users by clicking the **Select** checkbox or clicking anywhere on the user row. Click the **Select** button when done.

 **TOWER**
Projects
Inventories
Job Templates
Jobs
admin   

Teams >
Team >
Add Users

Username ▾   

Username ▾	First Name ▾	Last Name ▾	Select
admin			<input type="checkbox"/>
gdoge	Ginger	Doge	<input type="checkbox"/>
jdoge	Josie	Doge	<input checked="" type="checkbox"/>
ldoge	Leo	Doge	<input checked="" type="checkbox"/>

Select

Page 1 of 1 (4 items)

## 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.

---

### 8.1 Understanding How Credentials Work

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 primary key. 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. Refer to [Unified Job List API Endpoint](#) in the *Ansible Tower API Guide* for more information.

---

### 8.2 Getting Started with Credentials

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


Credentials can also be managed from either the **Teams** link or the **Users** link from the Setup (  ) menu. To manage credentials for teams, browse to the **Teams** tab and edit the appropriate team. To manage credentials for a user, browse to the **Users** tab and edit the appropriate user.

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.

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

- Create a new credential
- View Activity Stream

### 8.3 Add a New Credential

Create a new credential by selecting the  button.

Enter the appropriate details depending on the type of credential and select **Save**.



## 8.4 Credential Types

### 8.4.1 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.

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:** The actual SSH Private Key to be used to authenticate the user via SSH. This key is stored encrypted in the Tower database.
- **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`.
  - **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.
- **Privilege Escalation Username:** The username to use with escalation privileges on the remote system.
- **Privilege Escalation Password:** 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.

**A TOWER** Projects Inventories Job Templates Jobs admin

Setup > Credentials > **Create Credential**

---

**\* Name**

**Description**

**Does this credential belong to a team or user?**

User  Team

**\* User that owns this credential**

**\* Type**

Machine

**Username**

**Password**

Show

Ask at runtime?

**Private Key**

Hint: drag and drop an SSH private key file on the field below

**Private Key Passphrase**

Show

Ask at runtime?

**Privilege Escalation**

Sudo

**Privilege Escalation Username**

**Privilege Escalation Password**

Show

Ask at runtime?

**Vault Password**

Show

Ask at runtime?

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?”.

## 8.4.2 Source Control

Used with Projects to clone and update local source code repositories from a remote revision control system such as Git, Subversion, or Mercurial.

The screenshot shows the 'Create Credential' form in Ansible Tower. The breadcrumb trail is 'Setup > Credentials > Create Credential'. The form contains the following fields and controls:

- Name:** A required text input field.
- Description:** A text input field.
- Does this credential belong to a team or user?:** Radio buttons for 'User' (selected) and 'Team'.
- User that owns this credential:** A search input field with a magnifying glass icon.
- Type:** A dropdown menu currently showing 'Source Control'.
- Username:** A text input field.
- Password:** A text input field with a 'Show' button.
- SCM Private Key:** A large text area with a hint: 'Hint: drag and drop an SSH private key file on the field below'.
- Private Key Passphrase:** A text input field with a 'Show' button.
- Buttons:** 'Save' and 'Reset' buttons at the bottom left.

Source Control credentials have several attributes that may be configured:

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

---

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

---

## 8.4.3 Amazon Web Services

Enables synchronization of cloud inventory with Amazon Web Services.

The screenshot shows the 'Create Credential' form in Ansible Tower. The breadcrumb trail is 'Setup > Credentials > Create Credential'. The form has the following fields and controls:

- \* Name:** A text input field with a copy icon.
- Description:** A text input field.
- Does this credential belong to a team or user?** Radio buttons for 'User' (selected) and 'Team'.
- \* User that owns this credential:** A search input field with a magnifying glass icon.
- \* Type:** A dropdown menu currently showing 'Amazon Web Services'.
- \* Access Key:** A text input field.
- \* Secret Key:** A text input field with a 'Show' button and a copy icon.
- STS Token:** A text input field with a 'Show' button and a copy icon.
- Buttons:** 'Save' and 'Reset' buttons at the bottom left.

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.)

## 8.4.4 Rackspace

Enables synchronization of cloud inventory with Rackspace.

The screenshot shows the 'Create Credential' form in Ansible Tower. The breadcrumb trail is 'Setup > Credentials > Create Credential'. The form contains the following fields and controls:

- \* Name**: A text input field.
- Description**: A text input field.
- Does this credential belong to a team or user?**: Radio buttons for 'User' (selected) and 'Team'.
- \* User that owns this credential**: A search input field with a magnifying glass icon.
- \* Type**: A dropdown menu with 'Rackspace' selected.
- \* Username**: A text input field.
- \* API Key**: A text input field with a 'Show' button to toggle visibility.
- Buttons: 'Save' and 'Reset' at the bottom left.

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

## 8.4.5 VMware

Enables synchronization of inventory with VMware vCenter.

**A TOWER** Projects Inventories Job Templates Jobs admin

Setup > Credentials > **Create Credential**

**\* Name**

**Description**

Does this credential belong to a team or user? ?

User  Team

**\* User that owns this credential**

**\* Type** ?

VMware vCenter ▾

**\* vCenter Host** ?

**\* Username**

**\* Password**

Show

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.

---

## 8.4.6 Google Compute Engine

Enables synchronization of cloud inventory with Google Compute Engine.

**A TOWER** Projects Inventories Job Templates Jobs admin

Setup > Credentials > Create Credential

**\* Name**

Description

Does this credential belong to a team or user? ?  
 User  Team

**\* User that owns this credential**

**\* Type** ?  
 Google Compute Engine

**\* Service Account Email Address** ?

**\* RSA Private Key** ?  
 Hint: drag and drop a private key file on the field below

**Project** ?

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

## 8.4.7 Microsoft Azure

Enables synchronization of cloud inventory with Windows Azure.

**A TOWER** Projects Inventories Job Templates Jobs admin ✕ 📄 ↶

Setup > Credentials > **Create Credential**

**\* Name**

**Description**

**Does this credential belong to a team or user?** ⓘ

User  Team

**\* User that owns this credential**

**\* Type** ⓘ

Microsoft Azure

**\* Subscription ID** ⓘ

**\* Management Certificate** ⓘ

ⓘ Hint: drag and drop a management certificate file on the field below

Microsoft Azure credentials have several attributes that may be configured:

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

## 8.4.8 OpenStack

Enables synchronization of cloud inventory with OpenStack.



**A TOWER** Projects Inventories Job Templates Jobs admin

Setup > Credentials > **Create Credential**

**\* Name**

**Description**

**Does this credential belong to a team or user?**

User  Team

**\* User that owns this credential**

**\* Type**

**\* Host (Authentication URL)**

**\* Username**

**\* Password (API Key)**

**\* Project (Tenet Name/ID)**

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 (Tenet Name/ID):** The tenant name or tenant id used for OpenStack. This value is usually the same as the username.

If you are interested in using OpenStack Cloud Credentials, refer to [Utilizing Cloud Credentials](#) 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 **Name**, **Type**, or **Status**. For each project listed, you can edit project properties and delete the project, using the edit and delete icons.

Status	Name	Last Updated	Type	Actions
OK	Hello World!	6/18/2015 12:52:47 PM	Manual	

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

- Create a new project
- View Activity Stream

**Status** indicates the state of the project and may be one of the following:

- Running: Source control update is currently in progress.
- Never updated: Project is configured for source control, but has never been updated.
- Failed: The last source control update for this project failed.
- Successful: The last source control update for this project succeeded.
- Missing: Projects are absent from the project base path of `/var/lib/awx/projects` (applicable for manual or source control managed projects).
- OK: The project is not configured for source control, and is correctly in place.


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
- Cancel: Cancel a scheduled update from source control (only available when scheduling a source control update)

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

## 9.1 Add a new project

To create a new project, click the  button, which launches the **Create Project** dialog.



Enter the appropriate details into the following fields:

- Name
- Description
- 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* for more detail.)

All fields are required.

**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.”

### 9.1.1 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:

\*SCM Type

Manual

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

Project Base Path

`/var/lib/awx/projects`

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.

### 9.1.2 Manage playbooks using Source Control

Select the appropriate **SCM Type**. Then, 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.)

**A TOWER** Projects Inventories Job Templates Jobs admin

**Create Project**

**\* Name**  
Examples

**Description**  
Ansible example playbook

**\* Organization**  
Honey Dog, Inc.

**\* SCM Type**  
Git

**\* SCM URL**  
https://github.com/ansible/ansible-examples.git

**GIT URLs**

**SCM Branch**

**SCM Credential**


**SCM Update Options**  
 Clean  Delete on Update  Update on Launch

Click **Save** to save your project.

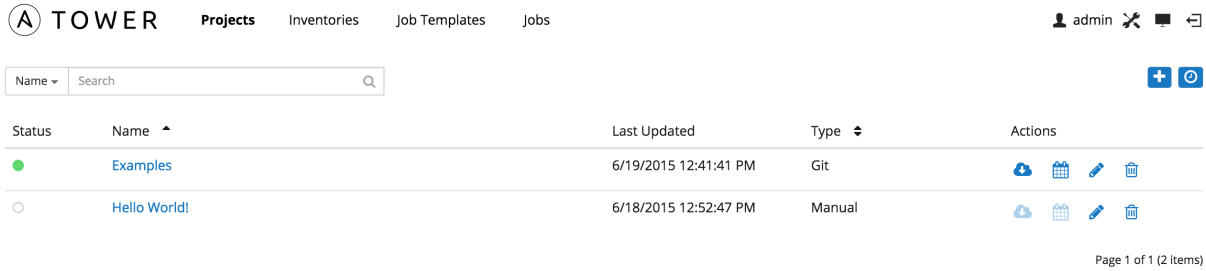
**Tip:** Using a Github link offers an easy way to use a playbook. To help get you started, use the `helloworld.yml` file available at: <https://github.com/ansible/tower-example.git>

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.

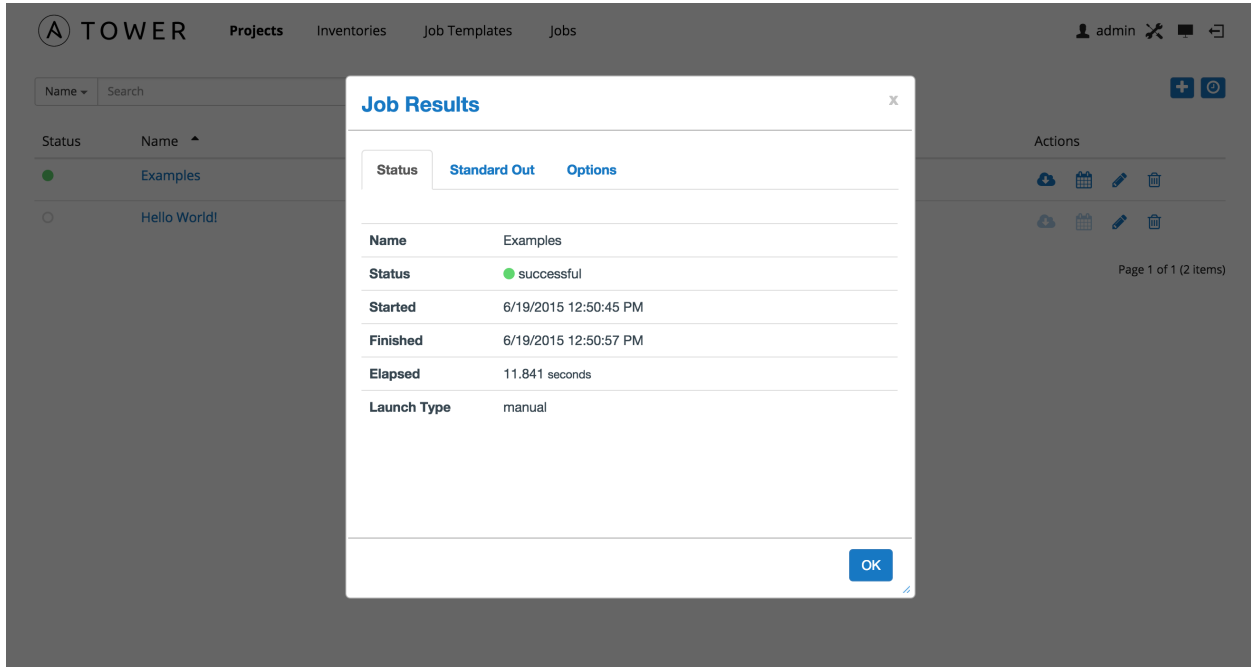
## 9.2 Updating projects from source control


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

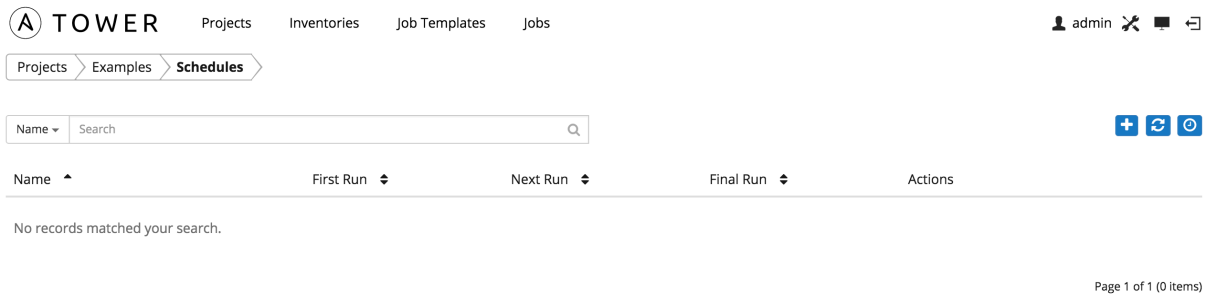
**Note:** Please note that immediately after adding a project setup to use source control, a “Sync” starts that fetches the project details from the configured source control.



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



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



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

## 9.3 Ansible Galaxy Support

At the end of a Project update, Tower searches for a file called `roles/requirements.yml` in the top level of the Project directory. If this file is found, the following command automatically runs:

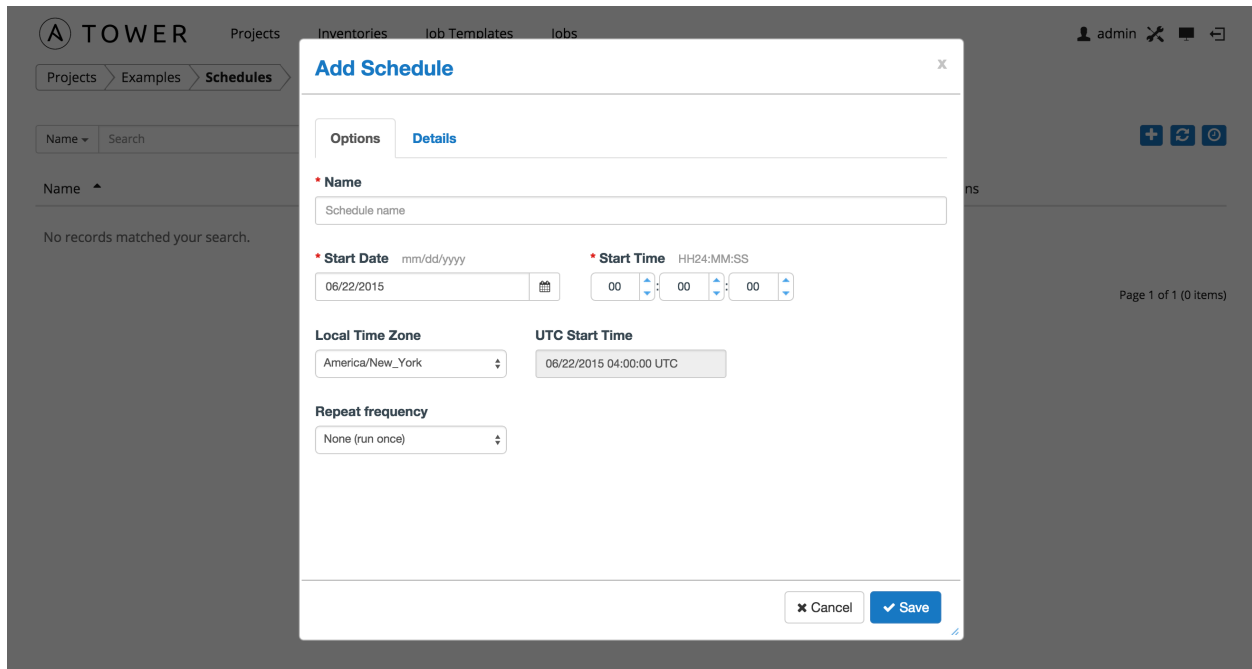
```
ansible-galaxy install -r requirements.yml -p ./roles/ --force
```

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.

## 9.4 Add a new schedule

To create a new schedule click the  button, which opens the **Add Schedule** dialog.



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)
- UTC Start Time (calculated from Start Time + Local Time Zone)
- Repeat Frequency (appropriate scheduling options are displayed depending on the frequency you select)

The **Details** tab displays a description of the 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.

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

- Stop an active schedule or activate a stopped schedule
- Edit Schedule
- Delete schedule

The screenshot shows the Ansible Tower interface for managing schedules. At the top, there's a navigation bar with 'TOWER' and several menu items: 'Projects', 'Inventories', 'Job Templates', and 'Jobs'. A user profile 'admin' is visible on the right. Below the navigation, there's a breadcrumb trail: 'Projects > Examples > Schedules'. A search bar is located below the breadcrumb. The main content area features a table with the following columns: 'Name', 'First Run', 'Next Run', 'Final Run', and 'Actions'. The table contains one entry with the following data:

Name	First Run	Next Run	Final Run	Actions
Schedule name	6/19/2015 12:00:00 AM	6/20/2015 12:00:00 AM	6/20/2015 12:00:00 AM	[Stop] [Edit] [Delete]

The page footer indicates 'Page 1 of 1 (1 items)'.



## INVENTORIES

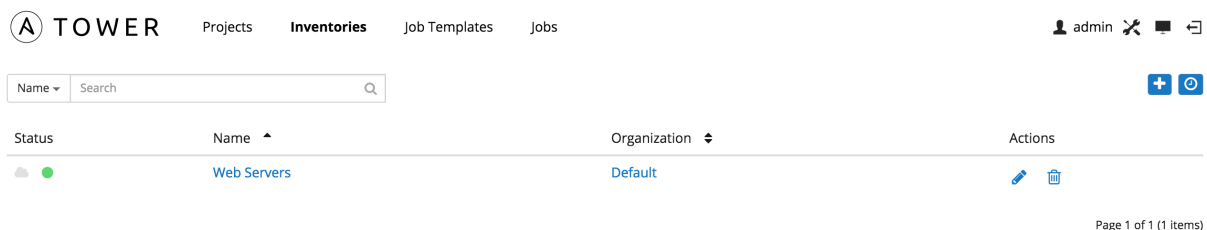
An inventory is a collection of hosts against which jobs may be launched, the same as an Ansible inventory file. Inventories are divided into groups and these groups contain the actual hosts. Groups may be sourced manually, by entering host names into Tower, or from one of Ansible Tower’s supported cloud 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**, and filtered by inventories with external sources, inventories with external sources that have failed to update, and inventories whose hosts have failed jobs.




The list of inventories includes:

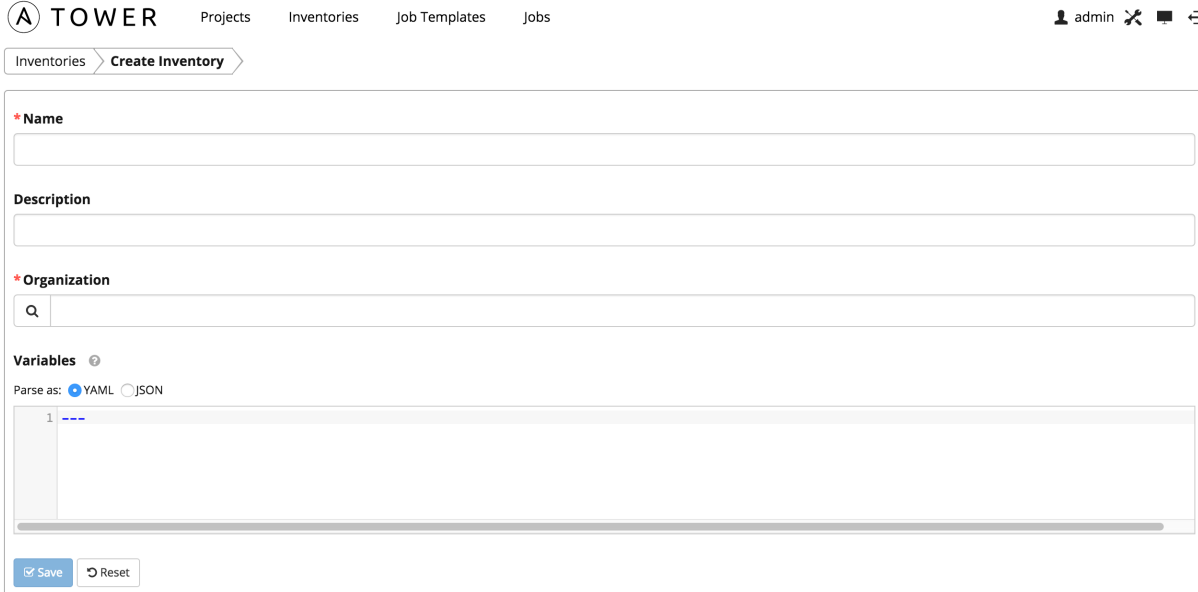
- **Status:** This includes the status of inventory synchronization for inventories configured with cloud sources, and the status of recent jobs for this inventory.
- **Name:** The inventory name. Clicking the Inventory name navigates to the properties screen for the selected inventory, which shows the 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!*

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

- Create a new inventory
- View Activity Stream

## 10.1 Add a new inventory

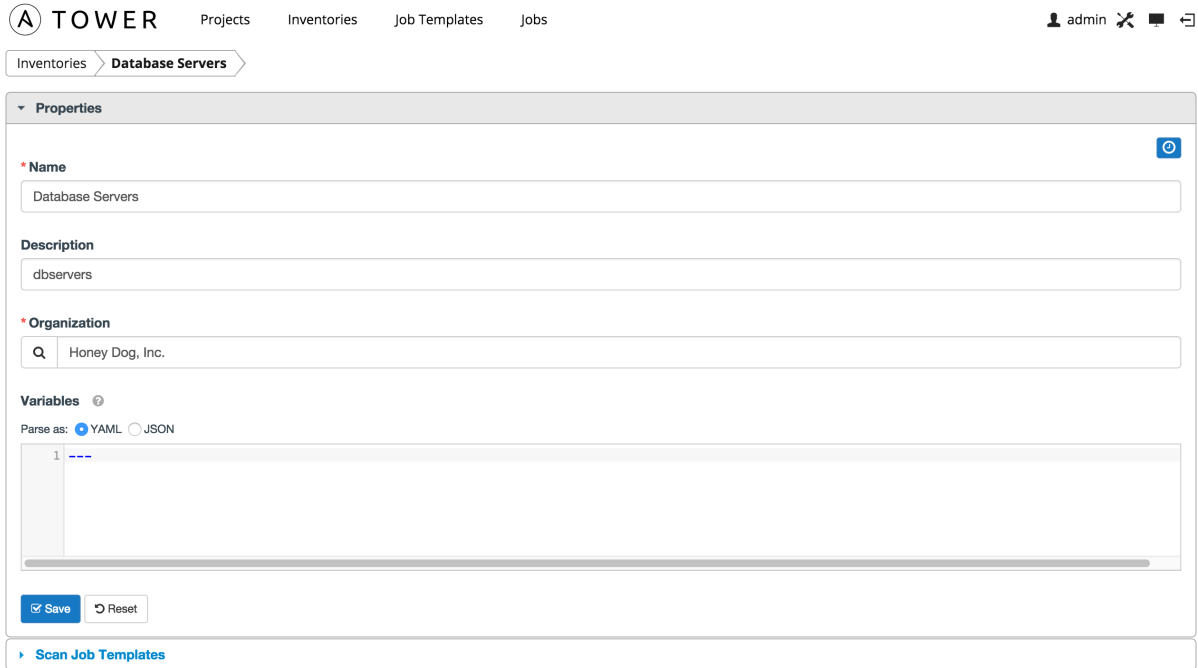
To create a new inventory click the  button, which opens the **Create Inventory** window.



The screenshot shows the 'Create Inventory' form in the Ansible Tower interface. The form is titled 'TOWER' and has navigation links for 'Projects', 'Inventories', 'Job Templates', and 'Jobs'. The user is logged in as 'admin'. The form is divided into several sections: 'Name' (required), 'Description', 'Organization' (with a search icon), 'Variables' (with a help icon), and 'Parse as' (radio buttons for 'YAML' and 'JSON'). The 'Variables' section contains a text area with a single line of text '1 ---'. At the bottom of the form are 'Save' and 'Reset' buttons.

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

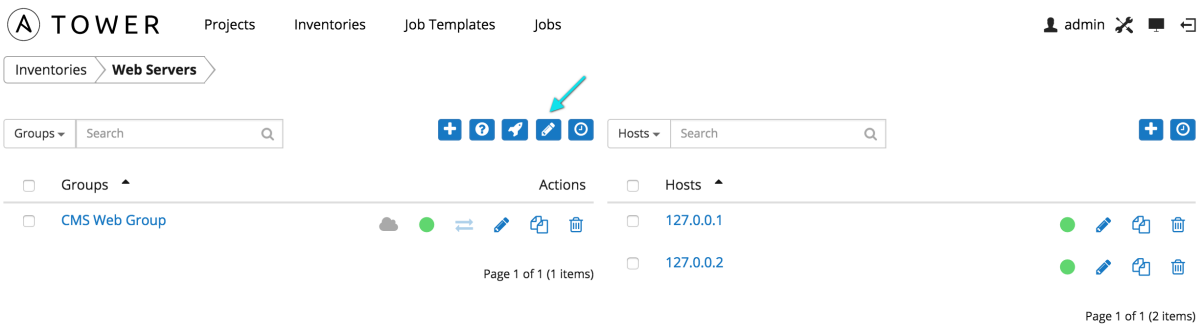
- **Name:** Enter a name appropriate for this inventory.
- **Description:** Enter an arbitrary description as appropriate.
- **Organization:** Choose among the available organizations.
- **Variables:** Variable definitions and values to be applied to all hosts in this inventory. Enter variables using either JSON or YAML syntax. Use the radio button to toggle between the two.




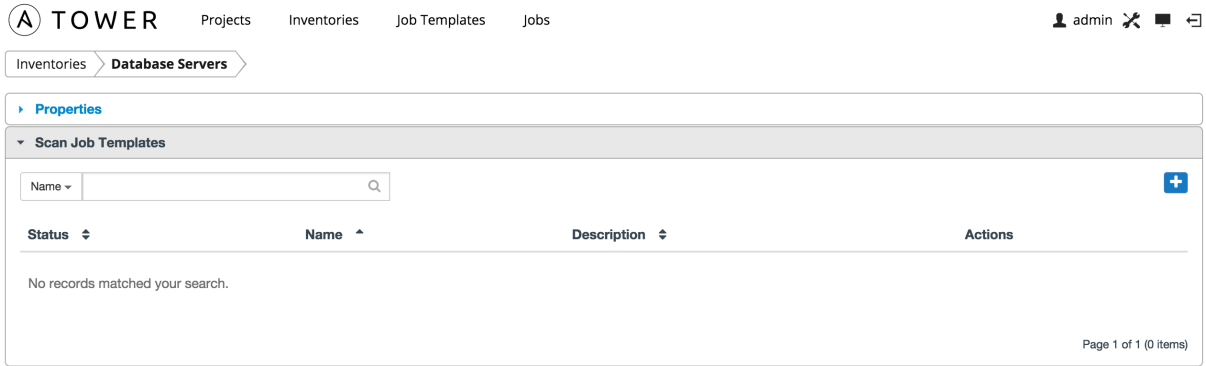
After clicking save, an accordion-style menu appears for **Scan Job Templates**. Expand this menu to view current scan job templates.

## 10.2 Scan Job Templates

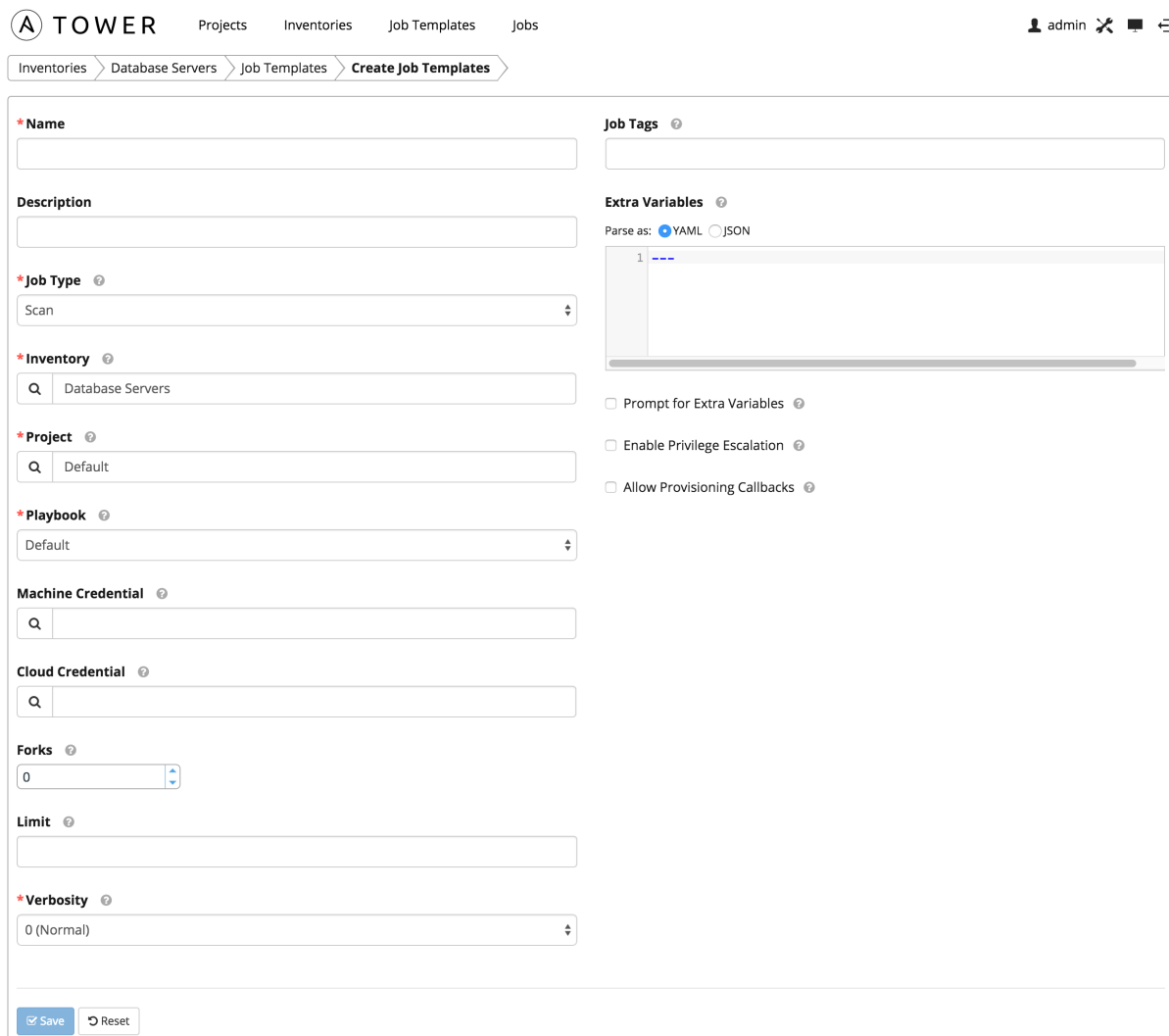
Scan jobs are special Job Templates that only collect information about the host on which the job is running.



To access scan job templates directly, click on the  button of a selected Inventory to edit the inventory's properties (above the display of groups and hosts). An accordion-style menu appears for **Scan Job Templates** under the inventory's **Properties** window. Expand this menu to view current scan job templates.



To create a new job template click the  button, which opens the **Create Job Templates** window.



Enter values for the following fields and select **Save**.

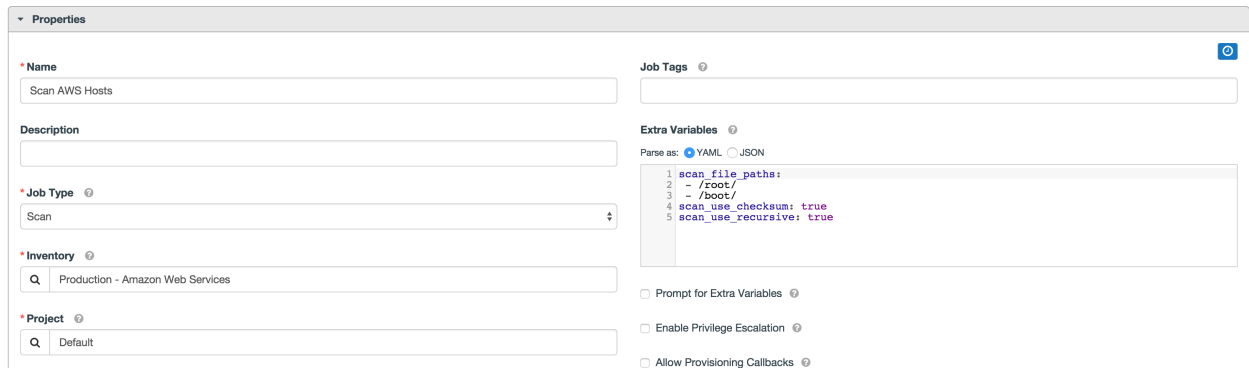
- **Name:** Enter a name appropriate for this inventory. (Required.)
- **Description:** Enter an arbitrary description as appropriate.

- **Job Type:** Jobs can be of type Run, Check, or Scan. (Required.)
- **Inventory:** Select the inventory containing the hosts you want this job to manage. (Required.)
- **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. (Required.)
- **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. (Required.)
- **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.
- **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.
- **Verbosity:** Control the level of output ansible will produce as the playbook executes. (Required.)
- **Job Tags:** Provide a comma separated list of tags to run a specific part of a play or task.
- **Extra Variables:** Variable definitions and values to be applied to all hosts in this job template. Enter variables using either JSON or YAML syntax. Use the radio button to toggle between the two.

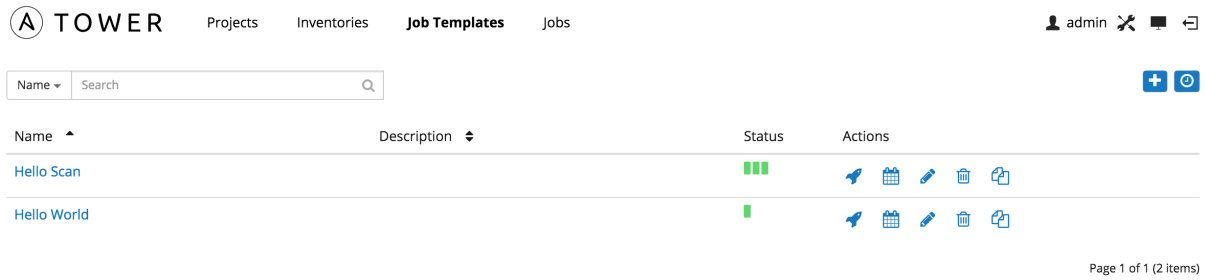
Extra variables can be passed as command line variables to the playbook. This is the “-e” or “--extra-vars” command line parameter for ansible-playbook that is documented in the Ansible documentation at [Passing Variables on the Command Line](#). Example commands might include:

```
scan_file_paths:
- /root/
- /root/
scan_use_checksum: true
scan_use_recursive: true
```


Extra variables can also be provided by key/value pairs using either YAML or JSON. These variables have a maximum value of precedence and overrides other variables specified elsewhere.

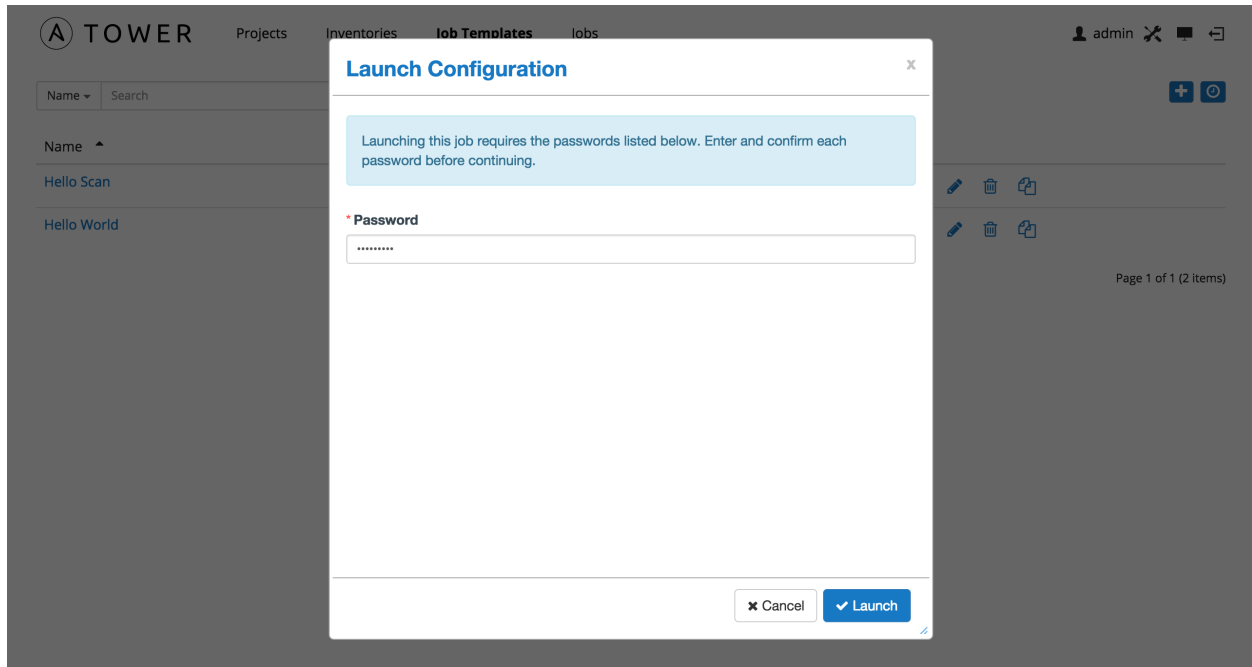


## 10.2.1 Launching a Scan Job Template



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.

**TOWER** Projects Inventories Job Templates Jobs admin

Jobs > 7 - Hello Scan

**Status** ● Successful

**Timing** Started 06/30/15 14:50:55 Finished 06/30/15 14:51:05 Elapsed 00:00:09

**Plays**

Started	Elapsed	Status	Name
14:50:58	00:00:07	●	all

**Tasks**

Started	Elapsed	Status	Name	Host Status
14:50:58	00:00:03	●	Gathering Facts	2
14:51:01	00:00:01	●	scan_packages	2
14:51:03	00:00:01	●	scan_services	2
14:51:04	00:00:00	●	scan_files	2

**Host Events**

Status	Host	Item	Message
●	104.130.169.209		
●	127.0.0.1		

**Events Summary**

● OK ● Changed ● Unreachable ● Failed

Host	Completed Tasks
104.130.169.209	3
127.0.0.1	3

**Host Summary**

OK

## 10.2.2 Scheduling a Scan Job Template

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

**TOWER** Projects Inventories Job Templates Jobs admin

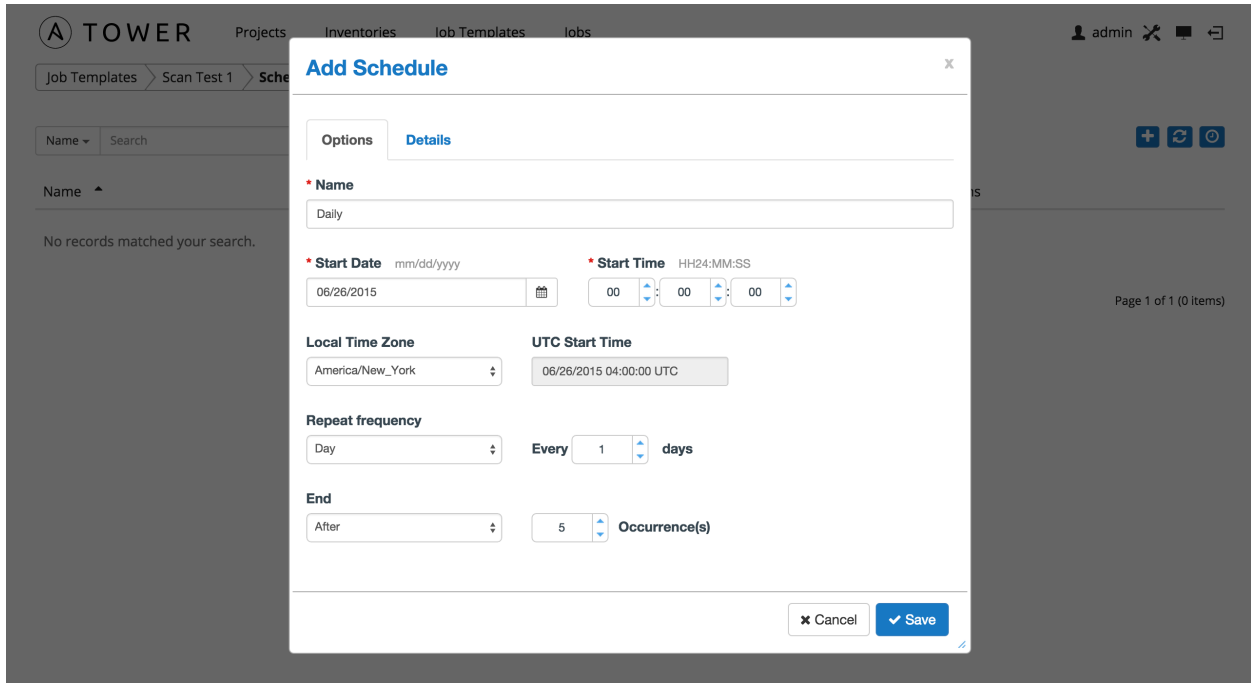
Job Templates > Scan Test 1 > Schedules

Name  Search + ↺ ⌂

Name	First Run	Next Run	Final Run	Actions
No records matched your search.				

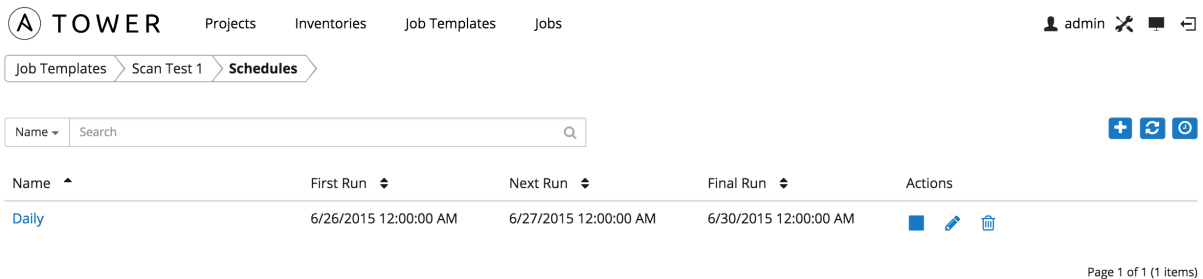
Page 1 of 1 (0 items)

Click the button to add a schedule.



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)
- UTC Start Time (calculated from Start Time + Local Time Zone)
- Repeat Frequency (the appropriate options displays as the update frequency is modified.)



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

- Stop an active schedule or activate a stopped schedule
- Edit Schedule
- Delete schedule

### 10.2.3 Custom Scan Job Templates

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



## Fact Scan Playbook

The `scan_facts.yml` playbook contains invocations of, potentially, multiple file fact scan modules. The default playbook bundled with Tower invokes four (4) fact scan modules—`ansible`, `packages`, `services`, and `files`. 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 fact `file scan` module is the only module that accepts parameters:

```
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

The playbook for custom fact scans is similar to the example of the Fact Scan Playbook above. It differs in that it *only* invokes the custom fact scan module, `scan_foo`.

*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()
```

The custom fact scan module lives in the `/library/` subdirectory of the Ansible project. The fact scan module is very simple, returning a hard-coded set of facts:

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

## 10.3 Groups and Hosts

Inventories are divided into groups, which may contain hosts and other groups, and hosts. To add a group or host to an inventory or to manage an existing group or host, click on the inventory name.

This screen displays groups and hosts that belong to the selected Inventory.

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

### 10.3.1 Groups

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

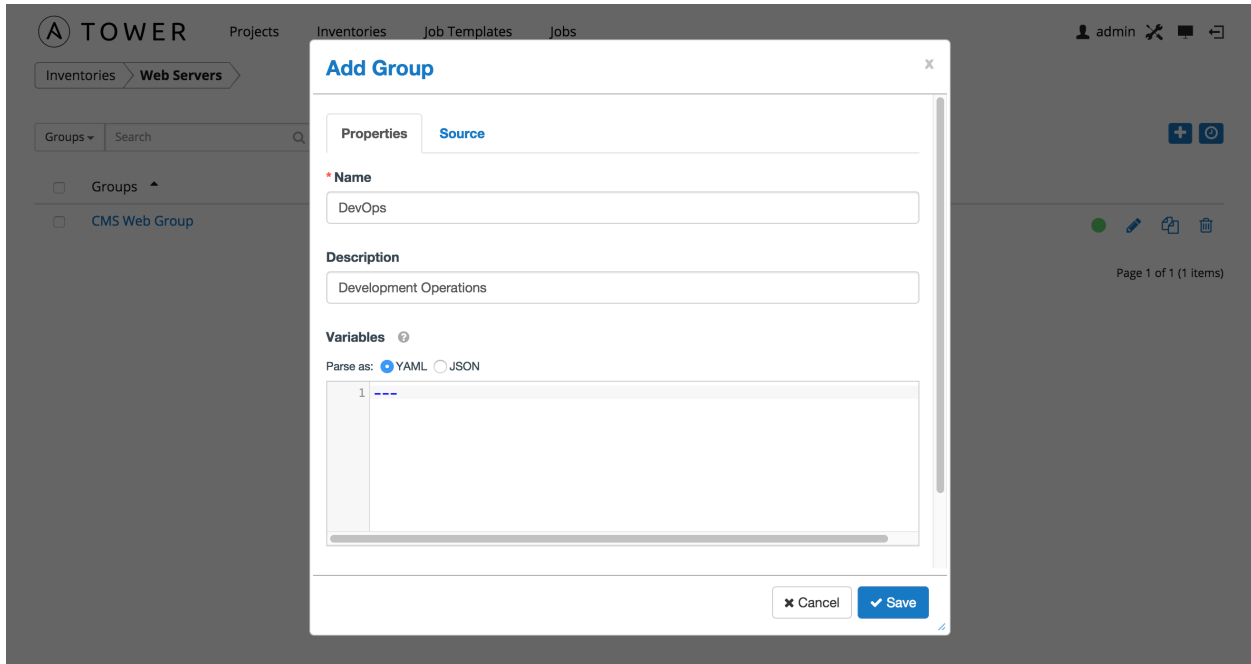
Additional actions may be performed on the group by selecting the buttons to the right of the group name:

- **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

Create a new group for an inventory by clicking the  button, which opens the **Create Group** window.

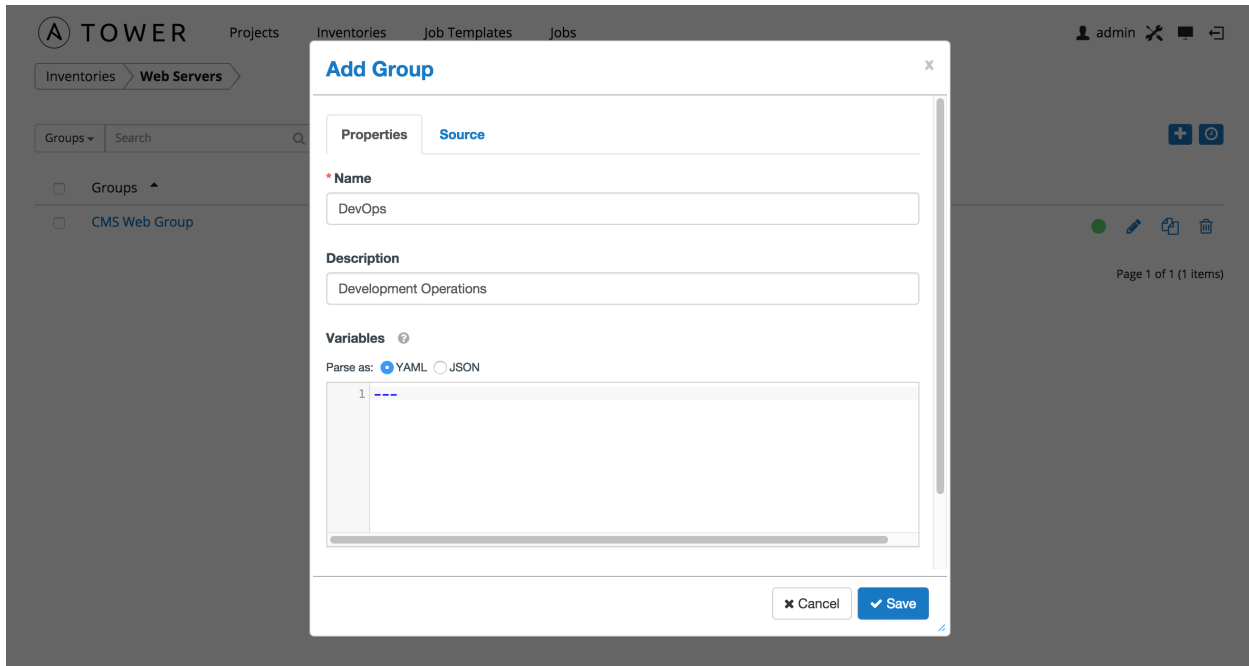


Enter the appropriate details into the following fields and click **Save**.

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

By default, the group **Source** is manual, meaning that the hosts must be entered into Tower manually. (Refer to [Add a new host](#) for more information on managing hosts individually.)

To synchronize the inventory group from a cloud source, select the **Source** tab and choose the appropriate source from the **Source** menu. Tower 2.2 supports Amazon Web Services EC2, Rackspace Cloud Servers, Google Compute Engine, VMware vCenter, Microsoft Azure, OpenStack, and custom scripts added by the administrator.



All cloud inventory sources have the following update options:

- **Overwrite:** When checked all child groups and hosts not found on the remote source is 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.

---

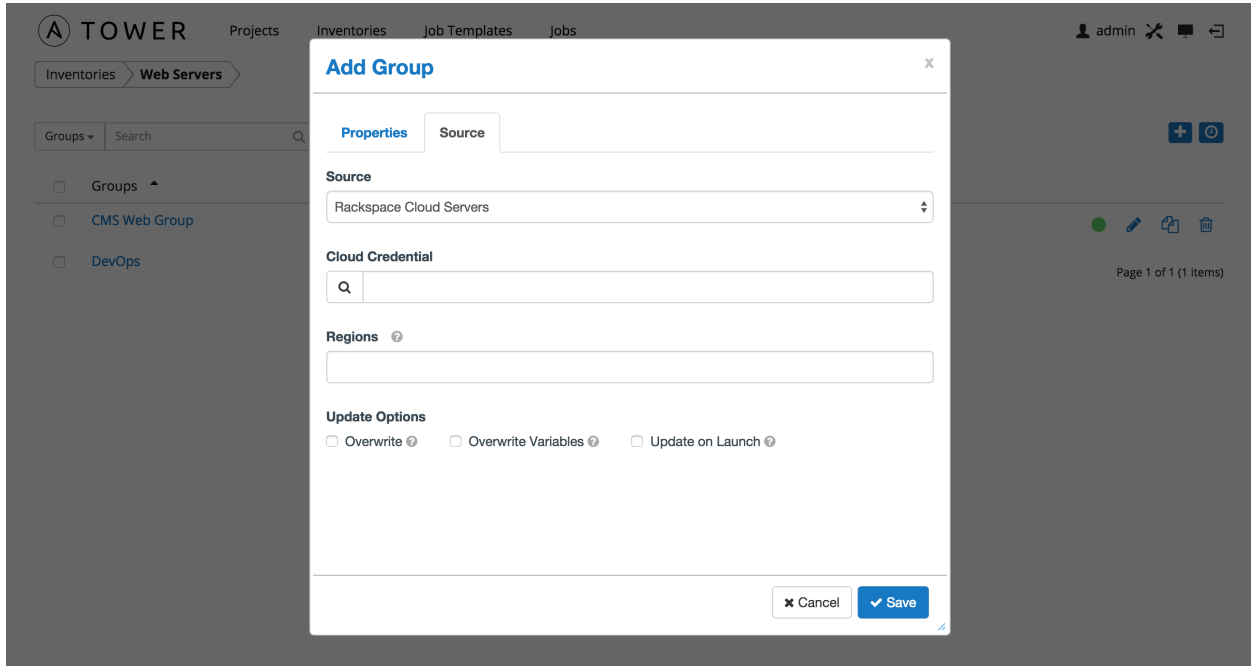
## Rackspace Cloud Servers

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.

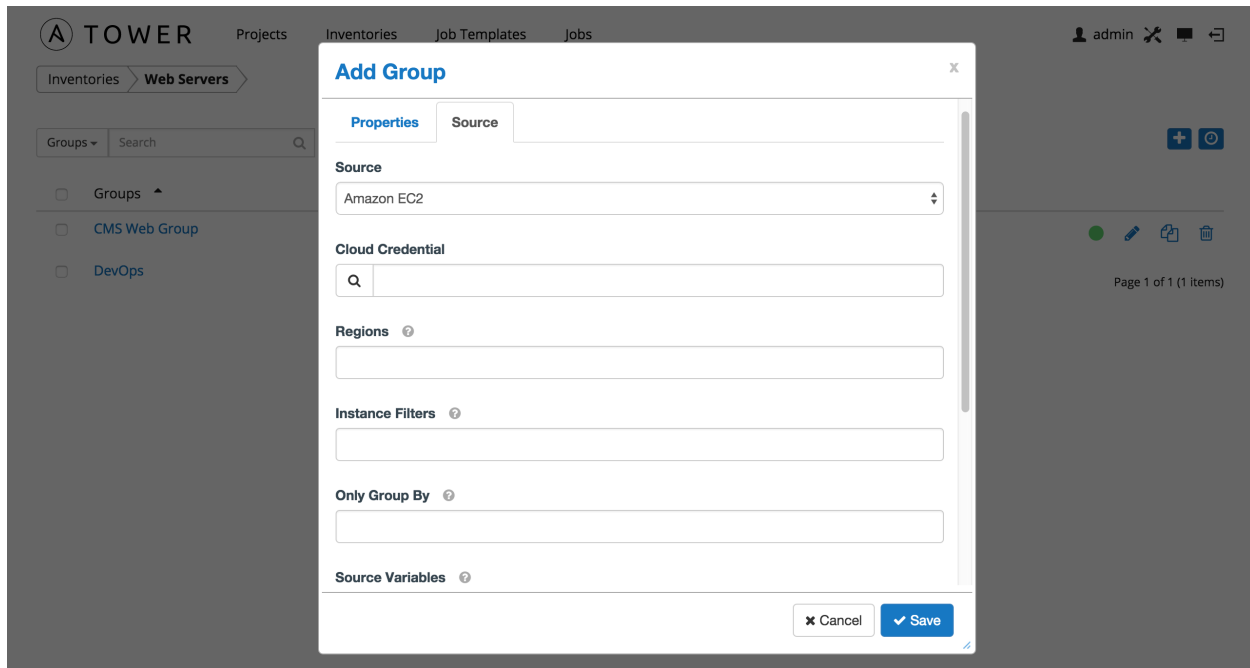
You can also configure **Update Options**.

- **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.



## Amazon Web Services EC2

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



- **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 [DescribeInstances](#) 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.

### Source Variables

Override variables found in `ec2.ini` and used by the inventory update script. For a detailed description of these variables [view `ec2.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.

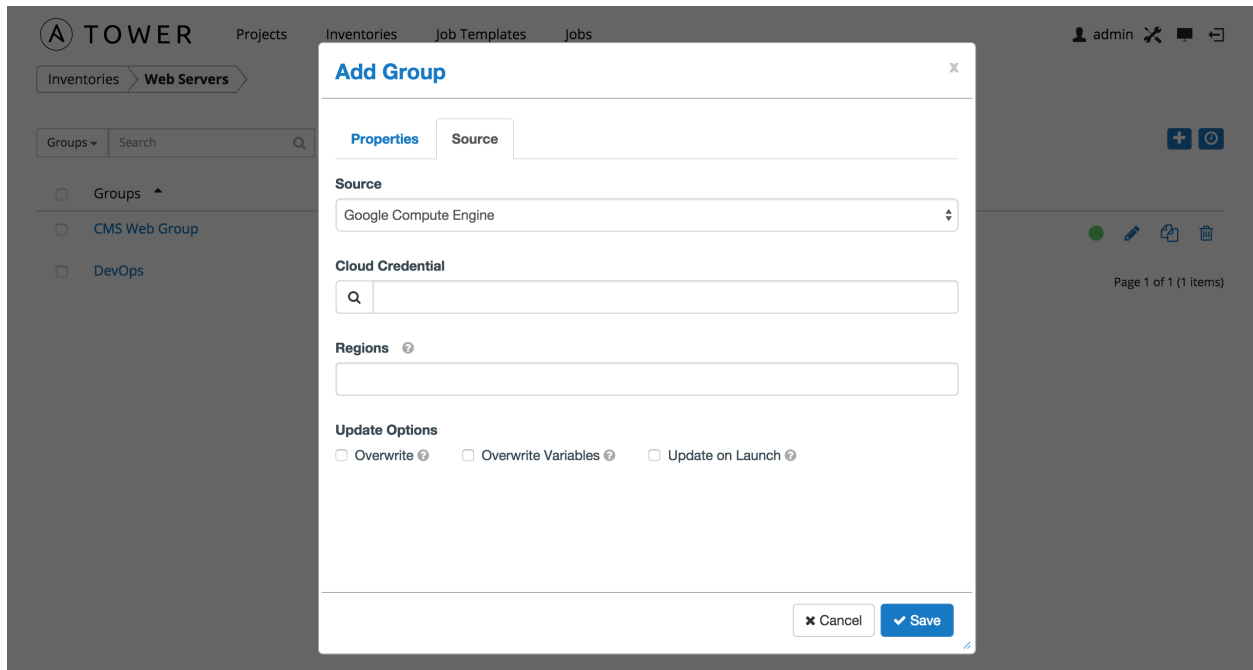
## Google Compute Engine

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.

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

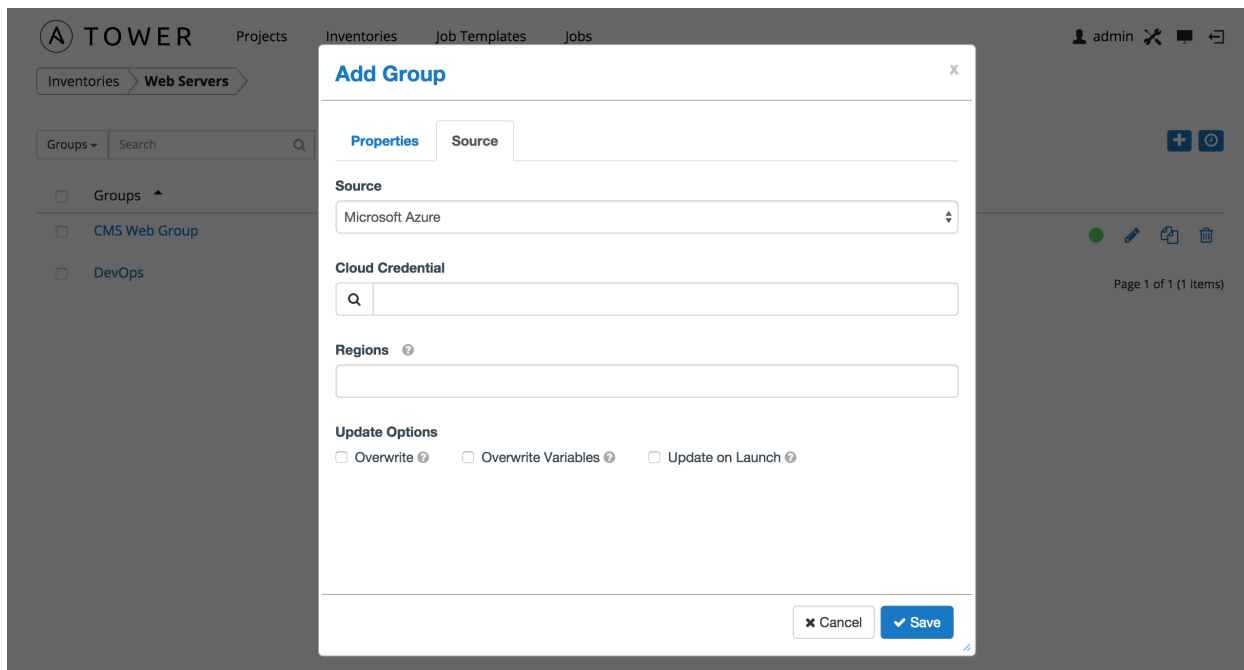
To configure a group for Microsoft Azure, select **Microsoft Azure** and enter the following details:

- **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.

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.





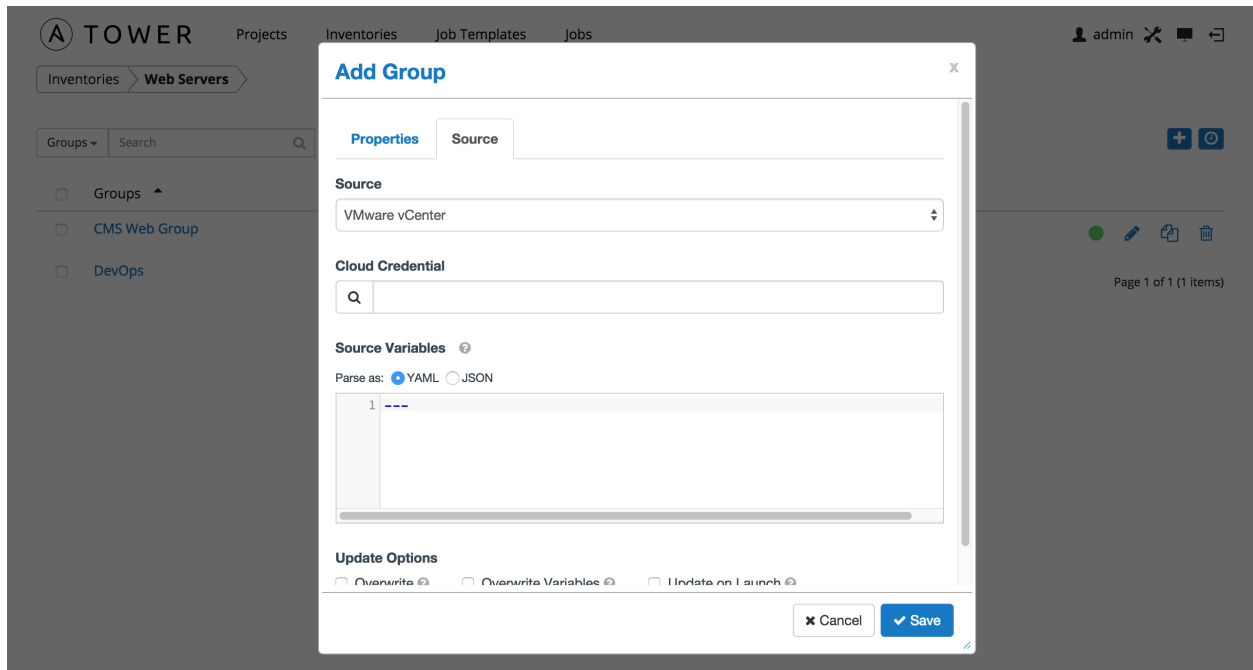
## VMware vCenter

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](#).
- **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.



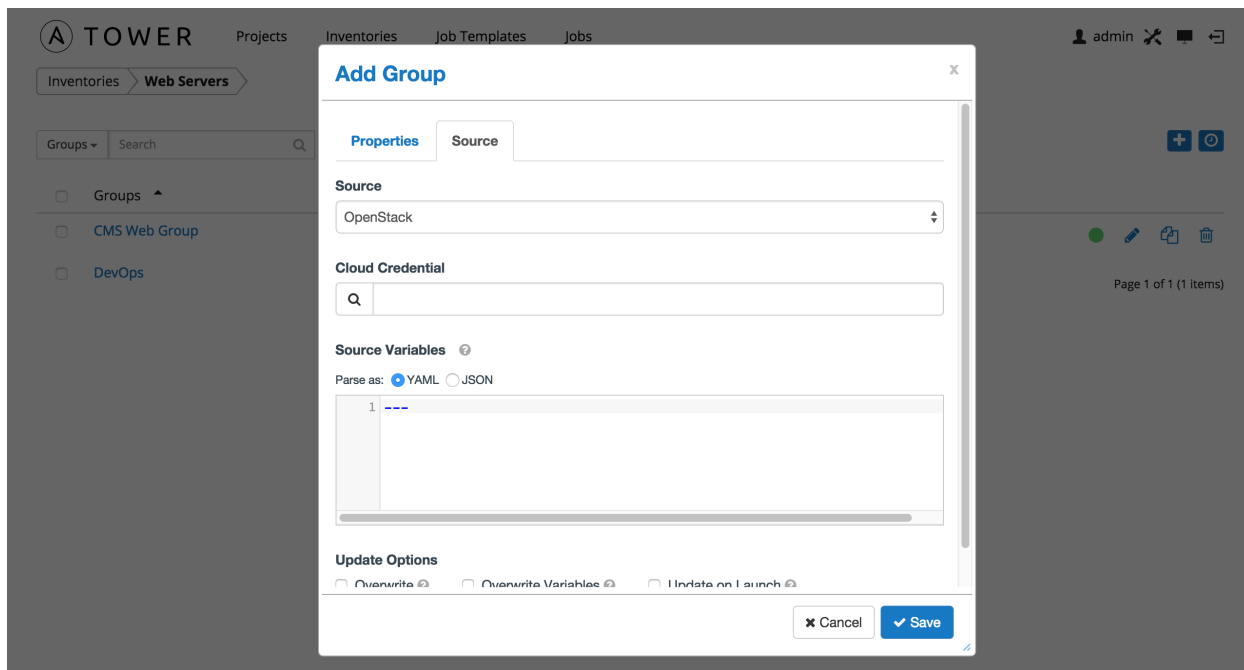
## OpenStack

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](#).
- **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.



### Custom Script

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

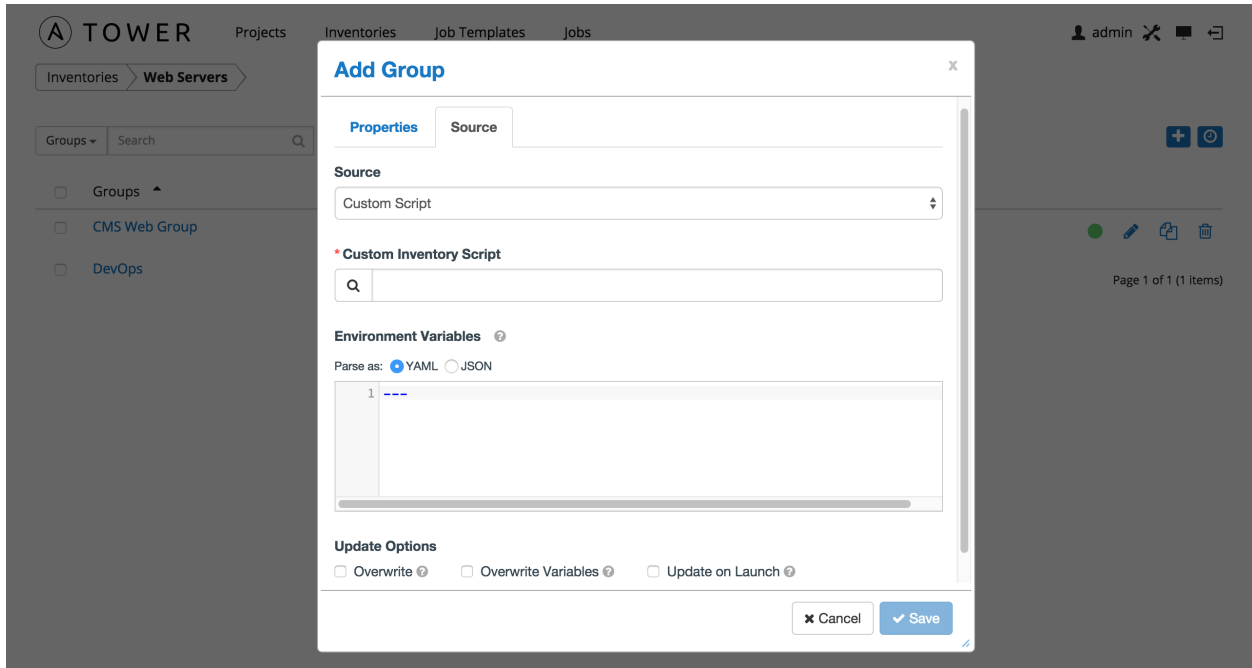
To configure a group to use a Custom Inventory Script, select **Custom Script** and enter the following details:

- **Custom Inventory Script:** Choose from an existing Inventory Script. For information on how to add custom inventory scripts to Tower, refer to [Custom Inventory Scripts](#).
- **Environment Variables:** Set variables in the environment to be used by the inventory update script. The variables would be specific to the script that you have written.


Enter variables using either JSON or YAML syntax. Use the radio button to toggle between the two.

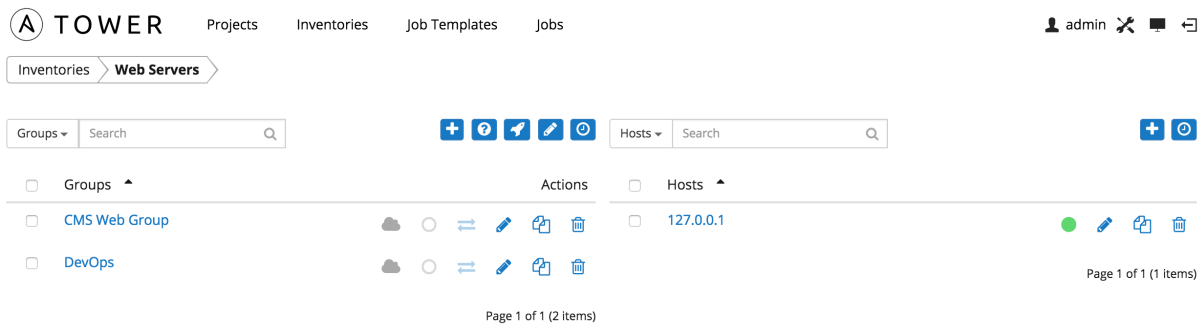
You can also configure **Update Options**.

- **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.

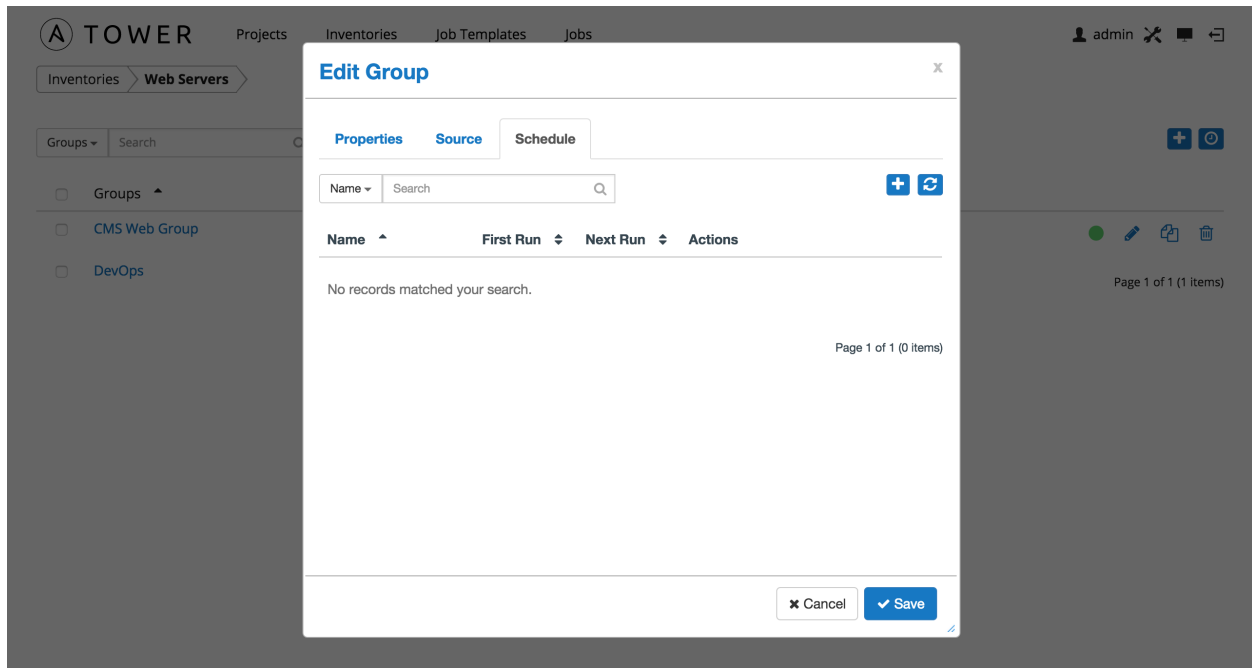


## Scheduling

For groups sourced from a cloud service, the inventory update process may be scheduled via the **Schedule** tab. To access the **Schedule** tab, click the  button beside the Inventory Group name to open the **Edit Group** dialog.



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:

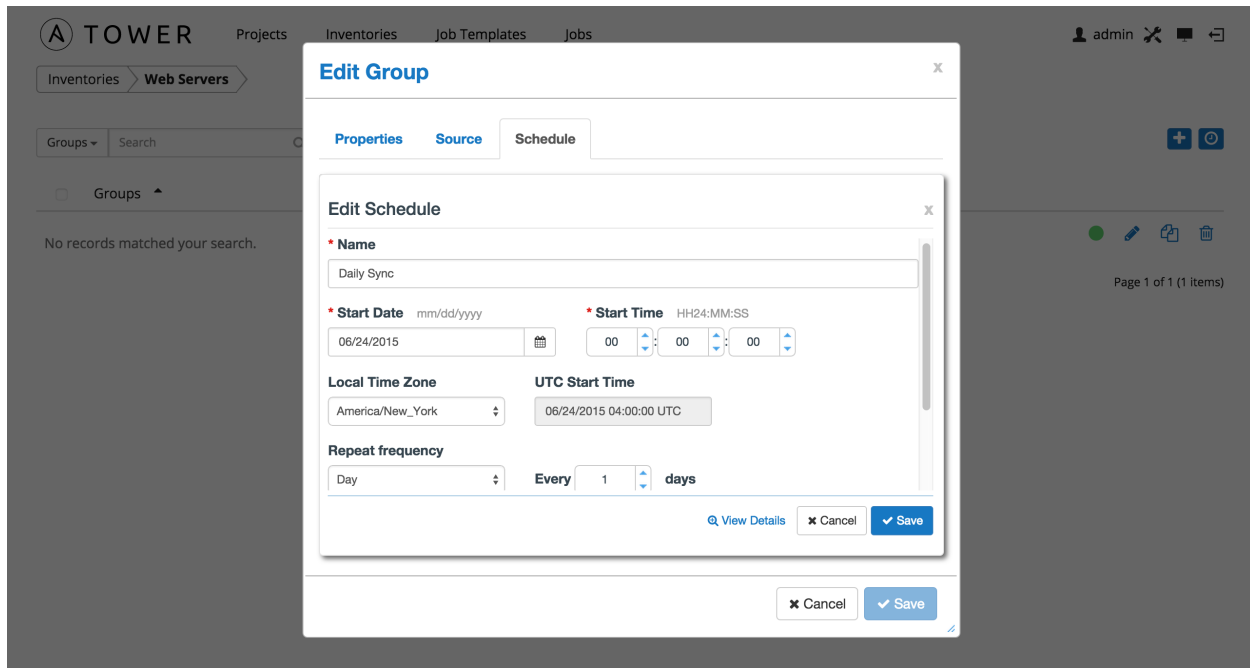
- Name (Clicking the schedule name opens the **Edit Schedule** dialog)
- First Run
- Next Run

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 click the  button.



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)
- **UTC Start Time** (calculated from Start Time + Local Time Zone)
- **Repeat Frequency** (the appropriate options are displayed as the update frequency is modified).

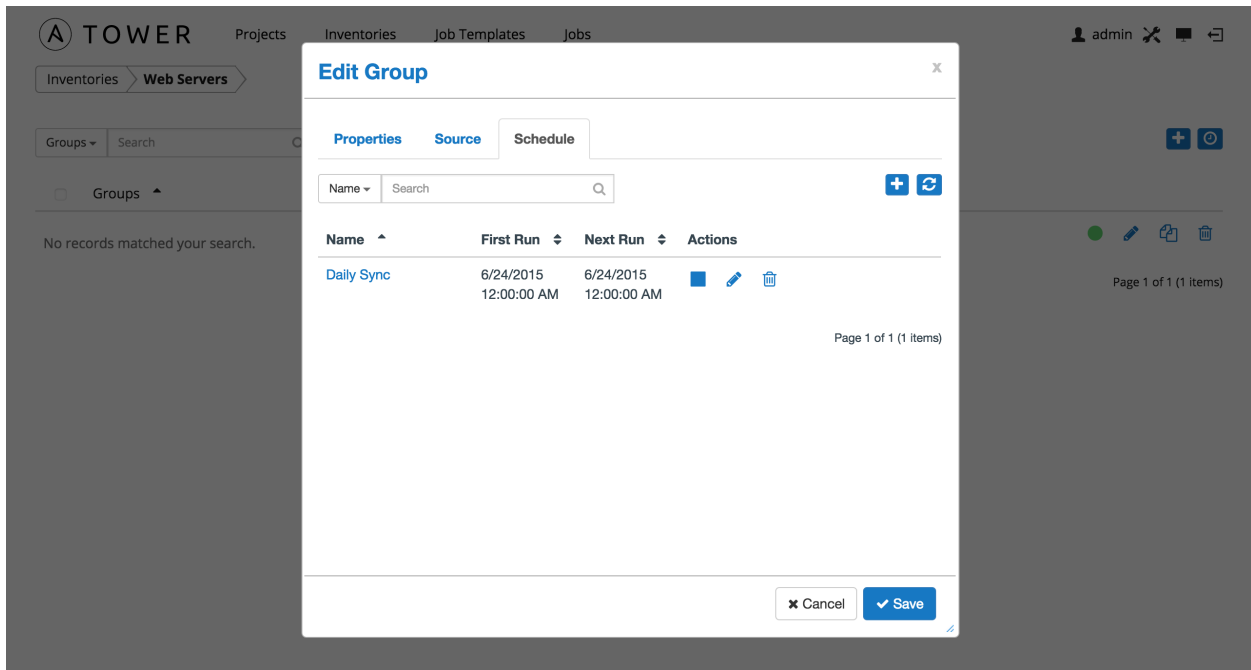
The **View Details** link at the bottom displays a description of the schedule and a list of the scheduled occurrences in the selected Local Time Zone.

---

**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.

---

Once you have saved the schedule, it can be viewed on the **Schedule** tab.

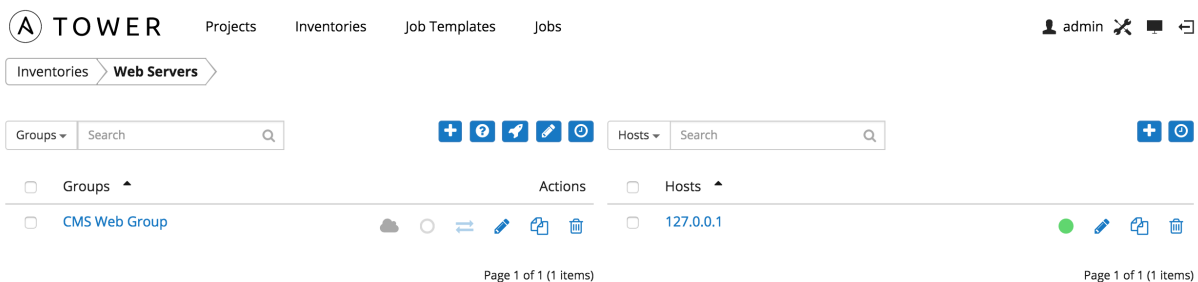


There are server actions available for schedules:

- Stop an active schedule or activate a stopped schedule
- Edit schedule
- Delete schedule

## 10.3.2 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:


- **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, click the  button.

This opens the **Create New Host** dialog.

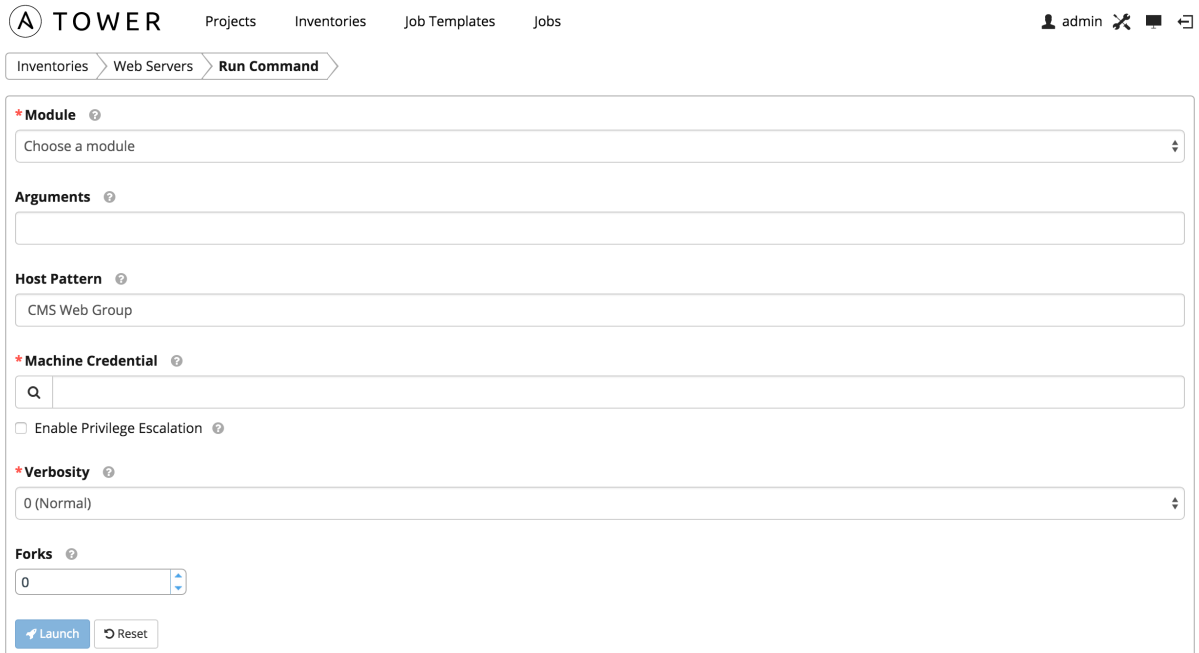
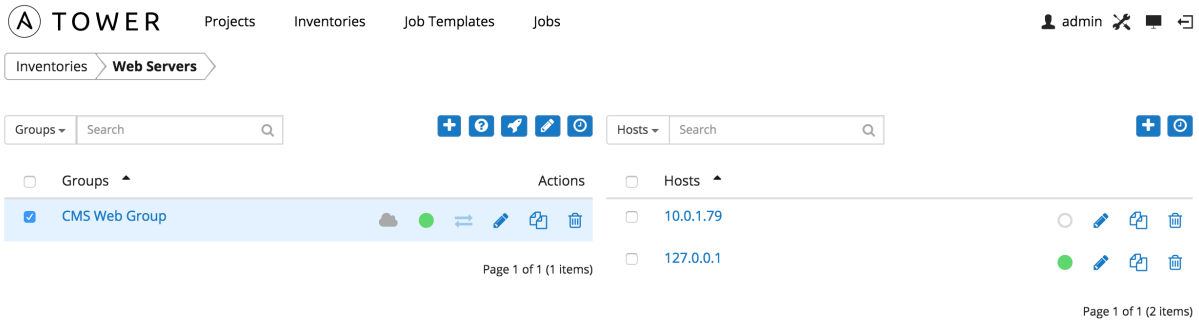
Enter the appropriate details into the following fields and click **Save**:

- **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.

## 10.4 Running Ad Hoc Commands

To run an ad hoc command, select an inventory source and click the  button. The inventory source can be a single group or host, a selection of multiple hosts, or a selection of multiple groups.





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.
- **Host Pattern:** Enter the pattern 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 **Launch** button to run this ad hoc command.

## 10.5 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:

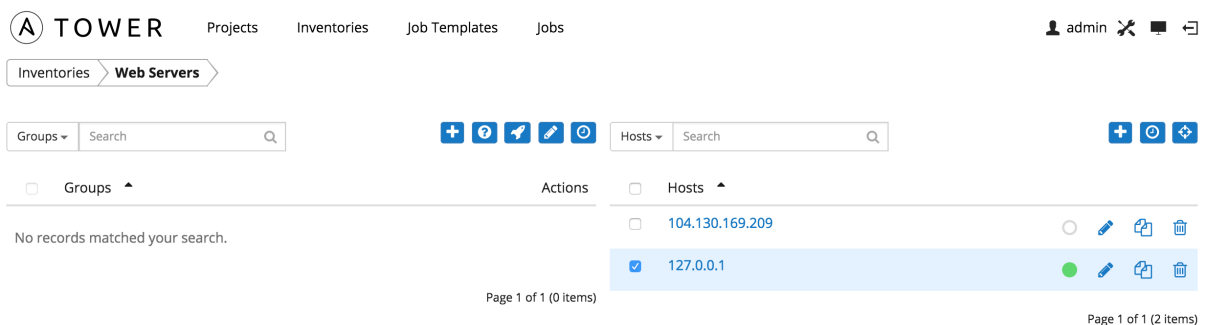
- Packages
- Services
- Files
- Ansible
- Custom

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.

### 10.5.1 Single Host Workflow

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



The screenshot shows the Ansible Tower web interface. At the top, there's a navigation bar with 'TOWER' and links for 'Projects', 'Inventories', 'Job Templates', and 'Jobs'. The user is logged in as 'admin'. Below the navigation, there's a breadcrumb trail: 'Inventories > Web Servers'. A search bar is present. The main content area is divided into two sections: 'Groups' and 'Hosts'. The 'Groups' section shows 'No records matched your search.' The 'Hosts' section shows a list of hosts. The host '127.0.0.1' is selected, and the 'Compare' button (a blue square with a white cross) is visible next to it. The page number 'Page 1 of 1 (2 Items)' is shown at the bottom right.

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.

**A TOWER** Projects Inventories Job Templates Jobs admin 🔧 🗨️ 🏠

Inventories > Web Servers > **System Tracking**

Compare latest facts collected on or before To latest facts collected on or before

📅 06/30/2015 📅 06/30/2015

« June 2015 »						
Su	Mo	Tu	We	Th	Fr	Sa
31	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	1	2	3	4
5	6	7	8	9	10	11

Services

Ansible

The two fact scans were identical for this module.




Select the module 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, while only changes for “packages” and “services” are shown.

Compare latest facts collected on or before To latest facts collected on or before

Packages	Services	Ansible
<b>Comparing facts collected from:</b>		
	<b>127.0.0.1 scanned on 6/30/2015 2:35:49 PM</b>	<b>127.0.0.1 scanned on 6/30/2015 2:50:56 PM</b>
ansible_product_serial	0	0
ansible_form_factor	Other	Other
ansible_product_version	1.2	1.2
ansible_fips	false	false
ansible_user_id	root	root
ansible_user_dir	/root	/root
ansible_memtotal_mb	1840	1840
ansible_product_uuid	415F08E2-BF64-465C-872F-8285C706DDF5	415F08E2-BF64-465C-872F-8285C706DDF5
ansible_architecture	x86_64	x86_64
ansible_distribution_version	7.1.1503	7.1.1503
ansible_domain	localdomain	localdomain
ansible_user_shell	/bin/bash	/bin/bash
ansible_virtualization_type	virtualbox	virtualbox
ansible_processor_cores	2	2
ansible_virtualization_role	guest	guest
ansible_processor_vcpus	2	2
ansible_bios_version	VirtualBox	VirtualBox
ansible_userspace_bits	64	64
ansible_ssh_host_key_ecdsa_public	AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyN...	AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyN...
ansible_swapfree_mb	2047	2047
ansible_distribution_release	Core	Core
ansible_system_vendor	innotek GmbH	innotek GmbH
ansible_os_family	RedHat	RedHat
ansible_userspace_architecture	x86_64	x86_64
ansible_swaptotal_mb	2047	2047
ansible_product_name	VirtualBox	VirtualBox
ansible_pkg_mgr	yum	yum
ansible_memfree_mb	545	461

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 (2) 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 (1) 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 *Tower Installation and Reference*

Guide.)

**A** TOWER
Projects Inventories Job Templates Jobs
admin   

Inventories > Web Servers > **System Tracking**

**Compare latest facts collected on or before**

**To latest facts collected on or before**


Packages
Services
Ansible

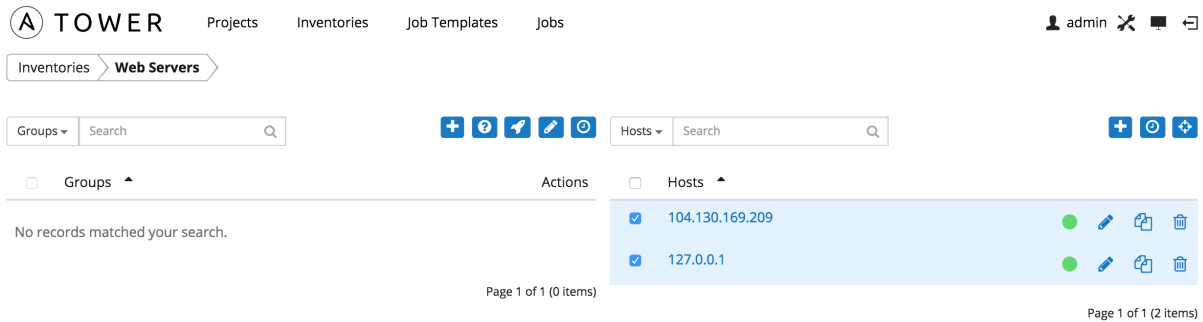
Comparing facts collected from:	127.0.0.1 scanned on 6/30/2015 2:35:49 PM	127.0.0.1 scanned on 6/30/2015 2:50:56 PM
ansible_product_serial	0	0
ansible_form_factor	Other	Other
ansible_product_version	1.2	1.2
ansible_fips	false	false
ansible_user_id	root	root
ansible_user_dir	/root	/root
ansible_memtotal_mb	1840	1840
ansible_product_uuid	415F08E2-BF64-465C-872F-8285C706DDF5	415F08E2-BF64-465C-872F-8285C706DDF5
ansible_architecture	x86_64	x86_64
ansible_distribution_version	7.1.1503	7.1.1503
ansible_domain	localdomain	localdomain
ansible_user_shell	/bin/bash	/bin/bash
ansible_virtualization_type	virtualbox	virtualbox
ansible_processor_cores	2	2
ansible_virtualization_role	guest	guest
ansible_processor_vcpus	2	2
ansible_bios_version	VirtualBox	VirtualBox
ansible_userspace_bits	64	64
ansible_ssh_host_key_ecdsa_public	AAAAE2VjZHNhLXNoYTItbmlzdHAYNTYAAAAIbmlzdHAYN...	AAAAE2VjZHNhLXNoYTItbmlzdHAYNTYAAAAIbmlzdHAYN...
ansible_swapfree_mb	2047	2047
ansible_distribution_release	Core	Core
ansible_system_vendor	innotek GmbH	innotek GmbH
ansible_os_family	RedHat	RedHat
ansible_userspace_architecture	x86_64	x86_64
ansible_swaptotal_mb	2047	2047
ansible_product_name	VirtualBox	VirtualBox
ansible_pkg_mgr	yum	yum
ansible_memfree_mb	545	461
ansible_user_uid	0	0

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.

## 10.5.2 Host to Host Workflow


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



The screenshot shows the Ansible Tower web interface. At the top, there's a navigation bar with 'TOWER' and tabs for 'Projects', 'Inventories', 'Job Templates', and 'Jobs'. The user is logged in as 'admin'. The breadcrumb trail is 'Inventories > Web Servers'. Below this, there are two search boxes: 'Groups' and 'Hosts'. The 'Groups' section shows 'No records matched your search.' The 'Hosts' section shows two hosts selected: '104.130.169.209' and '127.0.0.1'. Each host has a green status indicator, a pencil icon for editing, a document icon for details, and a trash icon for deletion. The page numbers are 'Page 1 of 1 (0 items)' for groups and 'Page 1 of 1 (2 items)' for hosts.

You can currently only select from a single page of hosts. This is a [known issue](#).

Select a single date on which to compare the two (2) hosts. Next, select the module for which you want to view differences.



The screenshot shows the Ansible Tower web interface. The breadcrumb trail is 'Inventories > Web Servers > System Tracking'. Below this, there are two date selection boxes. The first box is labeled 'Compare latest facts collected on or before' and has a date of '06/30/2015'. The second box is labeled 'To latest facts collected on or before' and also has a date of '06/30/2015'.

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, while only changes for “packages” and “services” are shown.

Compare latest facts collected across both hosts on or before

 06/30/2015

Packages	Services	Ansible
----------	----------	---------


Comparing facts collected from:	104.130.169.209 scanned on 6/30/2015 2:50:56 PM	127.0.0.1 scanned on 6/30/2015 2:50:56 PM
abrt	2.1.11-22.el7.centos.0.1.x86_64	absent
abrt-addon-ccpp	2.1.11-22.el7.centos.0.1.x86_64	absent
abrt-addon-kerneloops	2.1.11-22.el7.centos.0.1.x86_64	absent
abrt-addon-pstoreoops	2.1.11-22.el7.centos.0.1.x86_64	absent
abrt-addon-python	2.1.11-22.el7.centos.0.1.x86_64	absent
abrt-addon-vmcore	2.1.11-22.el7.centos.0.1.x86_64	absent
abrt-addon-xorg	2.1.11-22.el7.centos.0.1.x86_64	absent
abrt-cgi	2.1.11-22.el7.centos.0.1.x86_64	absent
abrt-console-notification	2.1.11-22.el7.centos.0.1.x86_64	absent
abrt-libs	2.1.11-22.el7.centos.0.1.x86_64	absent
abrt-python	2.1.11-22.el7.centos.0.1.x86_64	absent
abrt-retrace-client	2.1.11-22.el7.centos.0.1.x86_64	absent
abrt-tui	2.1.11-22.el7.centos.0.1.x86_64	absent
ansible	absent	1.9.1-1.el7.noarch
ansible-tower	absent	2.2.0-0.git201506260554.el7.centos.noarch
apr	absent	1.4.8-3.el7.x86_64
apr-util	absent	1.5.2-6.el7.x86_64
at	3.1.13-17.el7_0.1.x86_64	absent
attr	2.4.46-12.el7.x86_64	absent
augeas-libs	1.1.0-17.el7.x86_64	absent
bash-completion	1:2.1-6.el7.noarch	absent

## 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.

Name	Description	Status	Actions
Hello World!	hello world!	running	

To create a new job template click the  button.



**A TOWER** Projects Inventories Job Templates Jobs admin

Job Templates > **Create Job Templates**

**\* Name**

**Description**

**\* Job Type**

Run

**\* Inventory**

Web Servers

**\* Project**

**\* Playbook**

Choose a playbook

**Machine Credential**

**Cloud Credential**

**Forks**

0

**Limit**

**\* Verbosity**

0 (Normal)

**Job Tags**

**Extra Variables**

Parse as:  YAML  JSON

1

Prompt for Extra Variables

Enable Survey

Enable Privilege Escalation

Allow Provisioning Callbacks

Save
Reset

Enter the appropriate details into the following fields:

- **Name:** Required
- **Description:** Enter an arbitrary description as appropriate.
- **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. Only Superusers and Admins have permission to create scan jobs. A default playbook has been created for your use. Custom written scan playbooks can use scan modules.
  - 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.

- **Project:** Choose the project to be used with this job template from the projects available to the currently logged in Tower user.
- **Playbook:** Choose the playbook to be launched with this job template from the available playbooks. This menu is automatically populated with the names of the playbooks found in the project base path for the selected project. For example, a playbook named “jboss.yml” in the project path appears in the menu as “jboss”.
- **Credential:** Choose the 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.
- **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”.
  - For more information and examples refer to [Patterns](#) in the Ansible documentation.
- **Job Tags:**
  - A comma-separated list of playbook tags to constrain what parts of the playbooks are executed.
  - For more information and examples refer to [Tags](#) 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.
- **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** for Extra Variables: If this is checked, the user is prompted for Extra Variables at job execution. The set of extra variables defaults to any Extra Variables already configured for the job template.
- **Enable Survey:** Survey the user on job launch. Refer to [Surveys](#) for additional information.
- **Create Survey:** Creates a survey, if the survey is enabled.
- **Edit Survey:** Edits the existing survey for this job template.
- **Delete Survey:** Deletes the existing survey for this job template.
- **Allow 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.

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.

## 11.1 Utilizing Cloud Credentials

Cloud Credentials can be used when syncing a respective cloud inventory. Cloud Credentials may also be associated with a Job Template and included in the runtime environment for use by a playbook. The use of Cloud Credentials was introduced in Ansible Tower version 2.4.0.

### 11.1.1 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. 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 }}"
```

### 11.1.2 Amazon Web Services

Amazon Web Services cloud credentials are exposed as the following environment variables during playbook execution:

- 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.

### 11.1.3 Rackspace

Rackspace cloud credentials are exposed as the following environment variables during playbook execution:

- 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.

### 11.1.4 Google

Google cloud credentials are exposed as the following environment variables during playbook execution:

- 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.

### 11.1.5 Azure

Azure cloud credentials are exposed as the following environment variables during playbook execution:

- 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.

## 11.1.6 VMware

VMware cloud credentials are exposed as the following environment variables during playbook execution:

- 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
```

## 11.2 Surveys

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. If **Enable Survey** is checked, you can see a button to **Create 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.

---

### 11.2.1 Creating a Survey

Clicking on **Create Survey** brings up the **Add Survey** window.

A survey can consist of any number of questions. For each question, enter the following information:

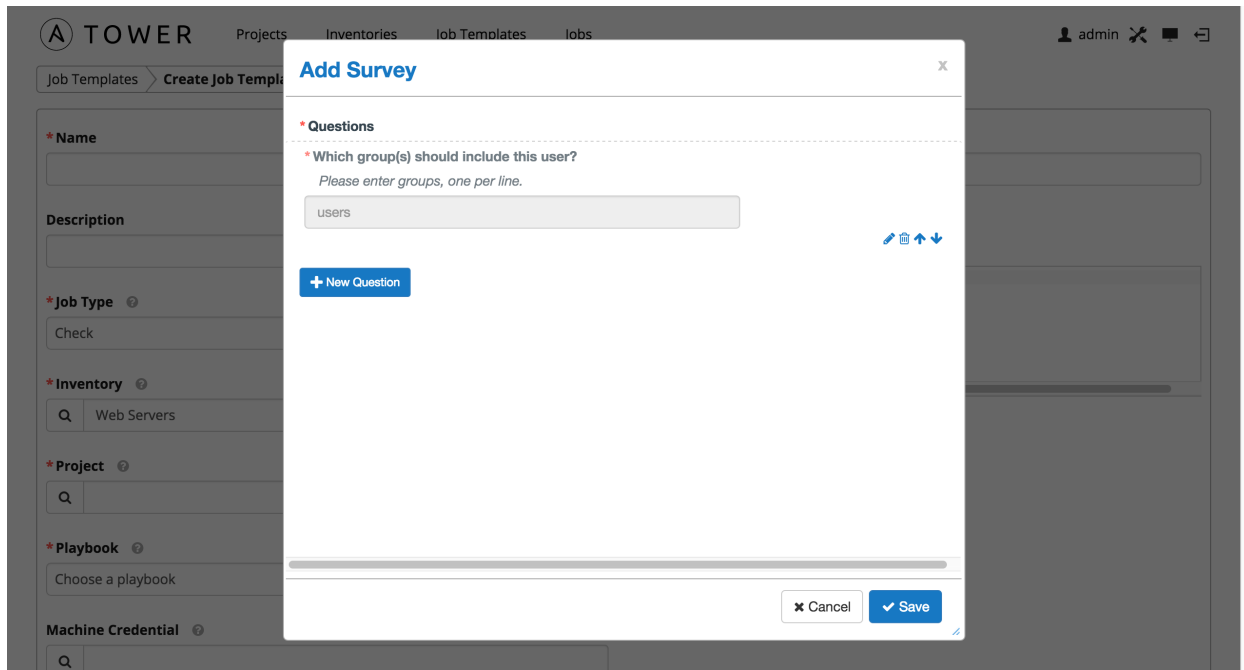
- **Name:** The question to ask the user
- **Description:** (optional) A description of what’s being asked of the user.
- **Answer Variable Name:** The Ansible variable name to store the user’s response in. This is the variable to be used by the playbook. Variable names cannot contain spaces.
- **Answer Type:** Choose from the following question types.
  - *Text:* A single line of text. You can set the minimum and maximum length (in characters) for this answer.
  - *Textarea:* A multi-line text field. You can set the minimum and maximum length (in characters) for this answer.

- *Password*: Responses are treated as sensitive information, much like an actual password is treated. You can set the minimum and maximum length (in characters) for this answer.
- *Multiple Choice (single select)*: A list of options, of which only one can be selected at a time. Enter the options, one per line, in the **Multiple Choice Options** box.
- *Multiple Choice (multiple select)*: A list of options, any number of which can be selected at a time. Enter the options, one per line, in the **Multiple Choice Options** box.
- *Integer*: An integer number. You can set the minimum and maximum length (in characters) for this answer.
- *Float*: A decimal number. You can set the minimum and maximum length (in characters) for this answer.
- **Default Answer**: The default answer to the question. This value is pre-filled in the interface and is used if the answer is not provided by the user.
- **Required**: Whether or not an answer to this question is required from the user.

Once you have entered the question information, click **Add Question** to add the question.

A stylized version of the survey is presented, along with a **New Question** button. Click this button to add additional questions.

For any question, you can click on the **Edit** button to edit the question, the **Delete** button to delete the question, and click on the Up and Down arrow buttons to rearrange the order of the questions. Click **Save** to save the survey.



Click **Save** to save the survey.

## 11.2.2 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 ( `""` ).

## 11.2.3 Extra Variables

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 variables and include variables	
block variables	
task variables	
extra variables	Job Template extra variables Job Template Survey (defaults) Job Launch extra_vars

### 11.2.4 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 that is being relaunched.

For example, say that you launched a Job Template with no `extra_vars` which resulted in the creation of **j1**. Next, say that you changed the Job Template and all the `extra_vars` “{ “hello”: “world” }”. Relaunching **j1** results in the creation of **j2**, but **j2** will not have any `extra_vars`.

To continue upon this example, if you were to then launch the Job Template (with the `extra_vars` you added after the creation of **j1**), the relaunched job (**j3**) will include the `extra_vars` “{ “hello”: “world” }”. Relaunching **j3**, resulting in the creation of **j4**, would also include `extra_vars` “{ “hello”: “world” }”.

## 11.3 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 that phone home to be configured, 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 Callbacks* checkbox. 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.

---

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://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. 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, which is amply conservative.

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 abusive 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 schedul-

ing 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` script and make that into an autoscaling image.

### 11.3.1 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.)

## 11.4 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.

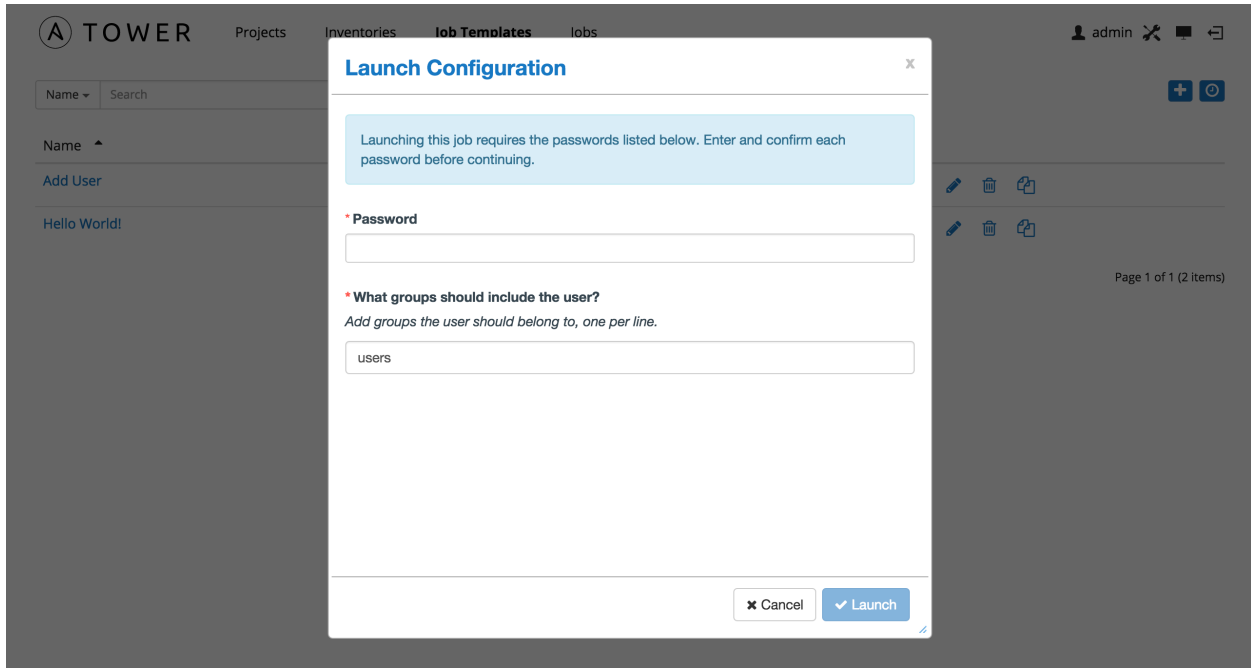
Easier deployments drive consistency, by running your playbooks the same way each time, and allow you to delegate responsibilities—even users who aren't Ansible experts can run Tower playbooks written by others.

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 a password, and runs the example survey created in *Surveys*.



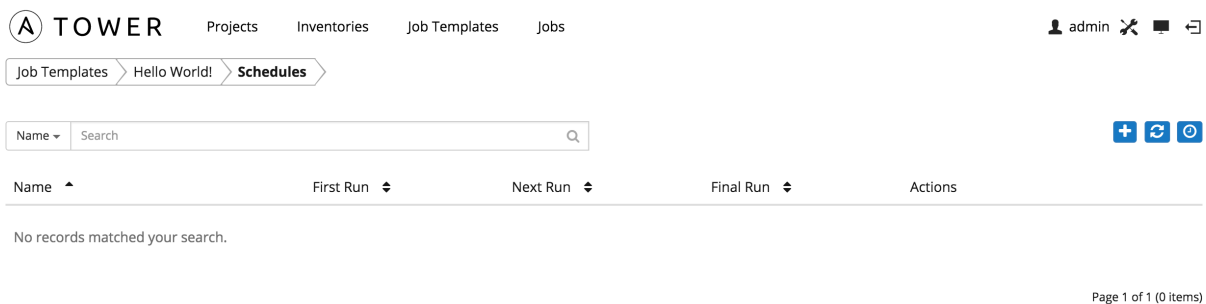
Along with any extra variables set in the job template and survey, Tower automatically adds the following variables to the job environment:

- `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.

## 11.5 Scheduling

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




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**.

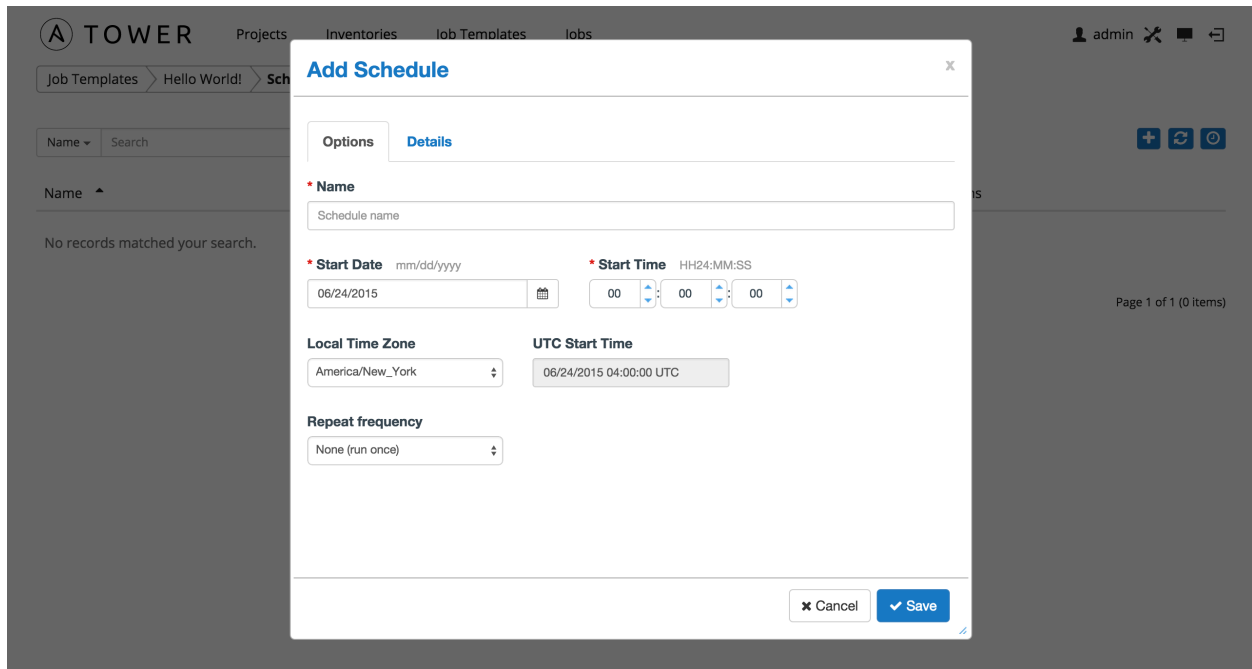
The list of schedules includes: - **Name**: Clicking the schedule name opens the **Edit Schedule** dialog - **First Run**: The first scheduled run of this task - **Next Run**: The next scheduled run of this task - **Final Run**: If the task has an end date, this is the last run of the task

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

### 11.5.1 Add a new schedule

To create a new schedule click the  button.



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)
- UTC Start Time (calculated from Start Time + Local Time Zone)
- Repeat Frequency (the appropriate options displays 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.

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

- Stop an active schedule or activate a stopped schedule
- Edit Schedule
- Delete schedule

(A) TOWER
Projects
Inventories
Job Templates
Jobs
admin

Job Templates > Hello World! > **Schedules**

Name 
+ ↻ ⏹

Name ^	First Run ↕	Next Run ↕	Final Run ↕	Actions
Daily	6/24/2015 12:00:00 AM	6/25/2015 12:00:00 AM	6/26/2015 12:00:00 AM	<span>■</span> <span>✎</span> <span>🗑</span>

Page 1 of 1 (1 items)

## 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.

The screenshot shows the Tower web interface. At the top, there is a navigation bar with 'TOWER' and links for 'Projects', 'Inventories', 'Job Templates', and 'Jobs'. A user profile 'admin' is visible in the top right. Below the navigation, there are tabs for 'Jobs' and 'Schedule'. A search bar is present with a dropdown for 'Name' and a search icon. The main content is a table with columns: ID, Status, Finished, Type, Name, and Actions. There are three rows of job data. The bottom right of the table area shows 'Page 1 of 1 (3 items)'.

ID	Status	Finished	Type	Name	Actions
9	Successful	6/22/2015 4:24:54 PM	SCM Update	Examples	
8	Successful	6/22/2015 4:24:26 PM	SCM Update	Examples	
5	Successful	6/22/2015 10:45:56 AM	Playbook Run	Hello World!	

- Jobs can be searched by **Job Failed?**, job **ID**, **Name**, **Status**, or **Type**.
- Jobs can be filtered by job **ID**, **Finished**, **Type**, or **Name**.

From this screen, you can relaunch jobs, remove jobs, or view the standard output of a particular job.

Clicking on a **Name** for an **SCM Update** job opens the **Job Results** screen, while clicking on a manually created **Playbook Run** job takes you to the **Job Status** page for this job (also accessible after launching a job from the **Job Templates** link in the main Tower navigational menu).

### 12.1 Job Results

The **Job Results** window displays information about jobs of type **Inventory Sync** and **SCM Update**.

This display consists of three tabs. The **Status** tab includes details on the job execution:

- **Name:** The name of the job template from which this job was launched.
- **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*.

The **Standard Out** tab shows the full results of running the SCM Update or Inventory Sync playbook. This shows the same information you would see if you ran the Ansible playbook using Ansible from the command line, and can be useful for debugging.

The **Options** tab describes the details of this job:

- For *SCM Update* jobs, this consists of the **Project** associated with the job.
- For *Inventory Sync* jobs, this consists of:
  - **Credential:** The cloud credential for the job
  - **Group:** The group being synced
  - **Source:** The type of cloud inventory
  - **Regions:** Any region filter, if set
  - **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.

## 12.2 Job Status

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

The **Jobs** page consists of multiple areas: **Status**, **Plays**, **Tasks**, **Host Events**, **Events Summary**, and **Hosts Summary**.

The screenshot displays the Ansible Tower interface for a job named "11 - Hello World!". The main status is "Successful". The job started at 06/24/15 13:40:17 and finished at 06/24/15 13:40:24, with an elapsed time of 00:00:06. The "Plays" section shows one play: "Hello World!" with a status of "Changed" (yellow dot). The "Tasks" section shows two tasks: "Gathering Facts" (status "OK", green dot) and "Hello World!" (status "Changed", yellow dot). The "Host Events" section shows one event for host "127.0.0.1" with status "OK". The "Events Summary" section shows a legend for "OK" (green), "Changed" (yellow), "Unreachable" (red), and "Failed" (red). Below the legend, the "Host Summary" section shows a large yellow ring and a legend for "Changed" (yellow square).

### 12.2.1 Status

The **Status** area shows the basic status of the job—*Running*, *Pending*, *Successful*, or *Failed*—and its start time. The buttons in the top right of the status page allow you to view the standard output of the job run, delete the job run, or relaunch the job.

Clicking on **more** gives the basic settings for this job:

- the job **Template** for this job
- the **Job Type** of *Run*, *Check*, or *Scan*
- the **Launched by** username associated with this job
- the **Inventory** associated with this job
- the **Project** associated with this job
- the **Playbook** that is being run
- the **Credential** in use
- the **Verbosity** setting for this job



- any **Extra Variables** that this job used

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

## 12.2.2 Plays

The **Plays** area shows the plays that were run as part of this playbook. The displayed plays can be filtered by **Name**, and can be limited to only failed plays.

For each play, Tower shows the start time for the play, the elapsed time of the play, the play **Name**, and whether the play succeeded or failed. Clicking on a specific play filters the **Tasks** and **Host Events** area to only display tasks and hosts relative to that play.

## 12.2.3 Tasks

The **Tasks** area shows the tasks run as part of plays in the playbook. The displayed tasks can be filtered by **Name**, and can be limited to only failed tasks.


For each task, Tower shows the start time for the task, the elapsed time of the task, the task **Name**, whether the task succeeded or failed, and a summary of the host status for that task. The host status displays a summary of the hosts status for all hosts touched by this task. Host status can be one of the following:

- **Success**: the playbook task returned “Ok”.
- **Changed**: the playbook task actually executed. Since Ansible tasks should be written to be idempotent, tasks may exit successfully without executing anything on the host. In these cases, the task would return Ok, but not Changed.
- **Failure**: the task failed. Further playbook execution was stopped for this host.
- **Unreachable**: the host was unreachable from the network or had another fatal error associated with it.
- **Skipped**: the playbook task was skipped because no change was necessary for the host to reach the target state.

Clicking on a specific task filters the **Host Events** area to only display hosts relative to that task.

## 12.2.4 Host Events

The **Host Events** area shows hosts affected by the selected play and task. For each host, Tower shows the host’s status, its name, and any **Item** or **Message** set by that task.

Click the  button to edit the host’s properties (hostname, description, enabled or not, and variables).

Clicking on the linked hostname brings up the **Host Event** dialog for that host and task.

The **Host Event** dialog shows the events for this host and the selected play and task:

- the **Host**
- the **Status**
- a unique **ID**
- a **Created On** time stamp
- the **Role** for this task
- the name of the **Play**

- the name of the **Task**
- if applicable, the Ansible **Module** for the task, and any *arguments* for that module

There is also a **JSON** tab which displays the result in JSON format.

The screenshot shows the Ansible Tower interface. A 'Host Event' dialog box is open, displaying the following information:

- Event:** JSON
- Host:** 127.0.0.1
- Status:** ok
- ID:** 30
- Created On:** 6/24/2015 2:08:21 PM
- Play:** Hello World!
- Task:** Gathering Facts
- Module:** setup


The background interface shows the 'Host Events' table with the following data:

Status	Host	Item
ok	127.0.0.1	

## 12.2.5 Events Summary

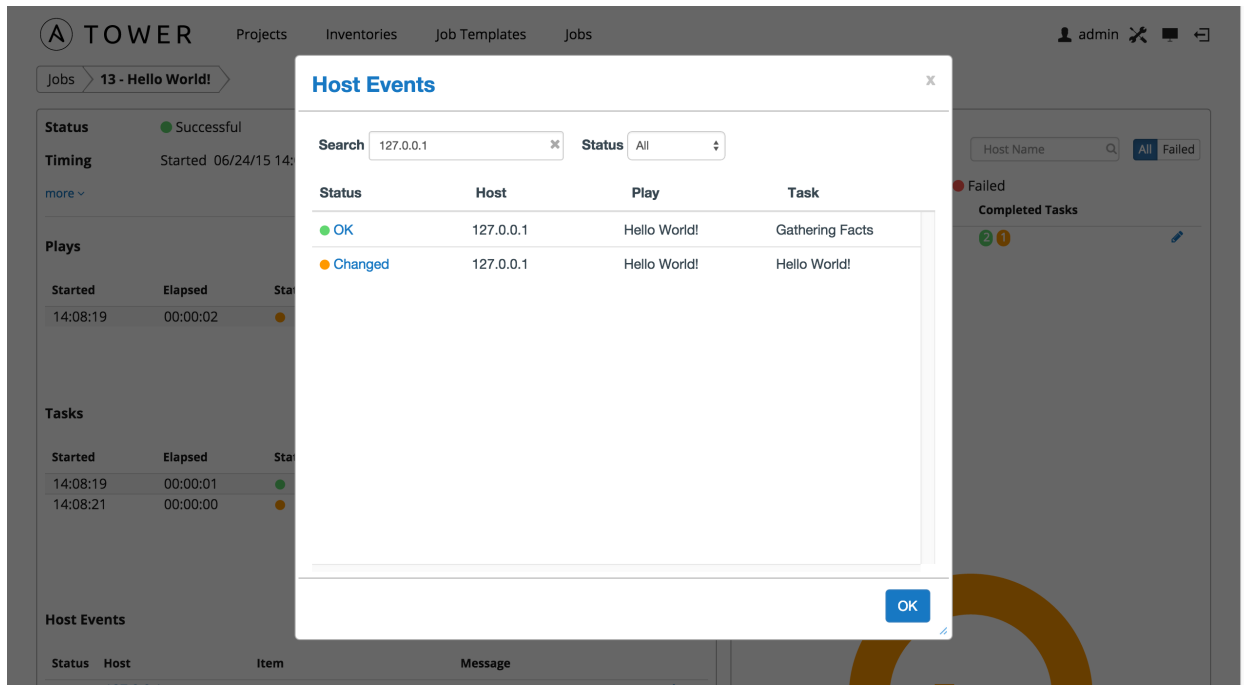
The **Events Summary** area shows a summary of events for all hosts affected by this playbook. Hosts can be filtered by their hostname, and can be limited to showing only changed, failed, OK, and unreachable hosts.

For each host, the **Events Summary** area shows the hostname and the number of completed tasks for that host, sorted by status.

Click the  button to edit the host's properties (hostname, description, enabled or not, and variables).

Clicking on the hostname brings up a **Host Events** dialog, displaying all tasks that affected that host.

This dialog can be filtered by the result of the tasks, and also by the hostname. For each event, Tower displays the status, the affected host, the play name, and the task name. Clicking on the status brings up a the same **Host Event** dialog that would be shown for that host and event from the Host Events area.



## 12.2.6 Host Summary

The **Host Summary** area shows a graph summarizing the status of all hosts affected by this playbook run.

## 12.3 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.

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 to 0 in Tower, the default value.

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 reorder 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.
- A project update will block for another project requiring the same update.
- 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.
- Ad hoc jobs are blocked by any inventory updates running against the inventory for that ad hoc job as specified.

## BEST PRACTICES

### 13.1 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.

### 13.2 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.

### 13.3 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.

---

## 13.4 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.

## 13.5 Autoscaling

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

## 13.6 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](#).

## 13.7 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 multi-tenancy RBAC features of Tower are sufficient to control who can run certain projects on what systems. For instance, you could easily control that engineering could not push to production.

All playbooks are executed via the `awx` filesystem 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 projection (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 filesystem 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.

### 14.1 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 filesystem 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.

## 14.2 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.

## 14.3 Role-Based Access Controls

A role is essentially a collection of permissions and all users receive permissions only through the roles to which they are assigned or through roles they inherit through the role hierarchy. Within an organization, roles are relatively stable, while users and permissions are both numerous and may change rapidly.

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.

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 that to which they are granted permission. 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 permission is being set.

### 14.3.1 Users

#### Views

User records can be viewed by the user who owns the user record, by the Organization Admin (if the user is a part of the organization they are associated with), and by the Super User.

#### Changes



The user who owns the user record can make changes to that user record, as can the Organization Admin (if the user is a part of the organization they are associated with), and the Super User. Organization Admins can add users to their organizations. Super Users can add users to any organization.

#### **Deletions**

The Organization Admin can remove a user if the user is a part of their organization. Super Users can remove any user from any organization as needed.

### **14.3.2 Organizations**

#### **Views**

Users who can see an organization must be a member of that organization, must be the Organization Admin for that organization, or a Super User.

#### **Changes**

Only Organization Admins (for the particular organization they are associated with) and Super Users can make changes to an organization.

#### **Deletions**

Only Organization Admins (for the particular organization they are associated with) and Super Users can remove an organization.

### **14.3.3 Inventories**

#### **Views**

Super Users can view any inventory and Organization Admins (for the particular organization they are associated with) can view inventories for their organization. Users or Teams associated with the inventory who have Read, Write, or Administrator privileges can also view the inventory for which they have explicit permission granted.

#### **Changes**

Super Users can edit any inventory and Organization Admins (for the particular organization they are associated with) can edit inventories for their organization. Users or Teams associated with the inventory who have Write or Administrator privileges granted can also edit an inventory.

#### **Deletions**

Super Users can remove any inventory and Organization Admins (for the particular organization they are associated with) can remove inventories for their organization. Users or Teams associated with the inventory who have the Administrator privilege granted can also remove an inventory.

#### **Ad Hoc Commands**

Super Users and Organization Admins (for the particular organization they are associated with) can run ad hoc commands against inventories for their organization. Users or Teams associated with the inventory who have Read, Write, or Administrator privileges granted, with the “Run Ad Hoc Commands” checkmark selected can also remove an inventory.

### **14.3.4 Hosts**

#### **Views**

Anyone who can view an inventory can view hosts assigned to that inventory.

**Changes**

Super Users can edit any inventory host and Organization Admins (for the particular organization they are associated with) can edit inventory hosts for their organization. Users or Teams associated with the inventory who have Write or Administrator privileges granted can also edit an inventory host.

**Deletions**

Super Users can remove any inventory host and Organization Admins (for the particular organization they are associated with) can remove inventory hosts for their organization. Users or Teams associated with the inventory who have the Administrator privilege granted can also remove an inventory host.

### 14.3.5 Groups

**Views**

Anyone who can view an inventory can view hosts assigned to that inventory.

**Changes**

Super Users can edit any inventory group and Organization Admins (for the particular organization they are associated with) can edit inventory groups for their organization. Users or Teams associated with the inventory who have Write or Administrator privileges granted can also edit an inventory group.

**Deletions**

Super Users can remove any inventory group and Organization Admins (for the particular organization they are associated with) can remove inventory groups for their organization. Users or Teams associated with the inventory who have the Administrator privilege granted can also remove an inventory group.

### 14.3.6 Inventory Updates

**Views**

Super Users can view any inventory update and Organization Admins (for the particular organization they are associated with) can view inventory updates for their organization. Users or Teams associated with the inventory with Administrator privileges granted can also see inventory updates.

**Deletions**

Super Users can remove any inventory update and Organization Admins (for the particular organization they are associated with) can remove inventory updates for their organization. Users or Teams associated with the inventory with Administrator privileges granted can also remove inventory updates.

### 14.3.7 Credentials

**Views**

Super Users and Organization Admins (for the particular organization they are associated with) can view credentials. The User or Team which owns the credential can also view it.

**Changes**

Super Users and Organization Admins (for the particular organization they are associated with) can edit credentials. The User or Team which owns the credential can also edit it.

**Deletions**

Super Users and Organization Admins (for the particular organization they are associated with) can remove credentials. The User or Team which owns the credential can also remove it.

### **14.3.8 Teams**

#### **Views**

Super Users and Organization Admins (for the particular organization they are associated with) can view Teams. The users associated with that Team can also view the Team.

#### **Changes**

Super Users and Organization Admins (for the particular organization they are associated with) can edit Teams.

#### **Deletions**

Super Users and Organization Admins (for the particular organization they are associated with) can remove Teams or individual team members.

### **14.3.9 Projects**

#### **Views**

Super Users and Organization Admins (for projects linked to the organization they are associated with) can view projects. Users or Teams associated with a project can also view that project. Users or Teams granted explicit permission for a project can also view that project, even if they are outside of the organization for which the project is associated.

#### **Changes**

Super Users and Organization Admins (for projects linked to the organization they are associated with) can edit projects.

#### **Deletions**

Super Users and Organization Admins (for projects linked to the organization they are associated with) can remove projects

### **14.3.10 Project Updates**

#### **Views**

Super Users and Organization Admins (for project updates linked to the organization they are associated with) can view projects. Users or Teams associated with a project can also view that project. Users or Teams granted explicit permission for a project can also view that project, even if they are outside of the organization for which the project is associated.

#### **Changes**

Super Users and Organization Admins (for project updates linked to the organization they are associated with) can edit projects.

#### **Deletions**

Super Users and Organization Admins (for project updates linked to the organization they are associated with) can remove projects

### 14.3.11 Job Templates

Job Templates have three associated actions—Check, Run, and Create.

- **Check:** Users or Team members with Check level permissions can run Check type jobs. Check is great for dry-runs and testing the Ansible playbook.
- **Run:** Users with Run level permissions can run Check and Run type jobs.
- **Create:** Users or Team members with Create level permissions can create new Job Templates. Super Users and Organization Admins (for job templates linked to the organization they are associated with) can create Job Templates.

#### Views

Super Users and Organization Admins (for job templates linked to the organization they are associated with) can view job templates. Users granted access to view the inventory, project, and credential can view an associated job template. Users with explicit team-based permissions granted during the job template setup can also view an associated job template. If you can see jobs run on the Job Template, you can view the job template.

- **System Job Templates:** Only Super Users can view.
- **Ad hoc commands:** Super Users and Organization Admins (for job templates linked to the organization they are associated with) can view ad hoc command job templates. Users and team members with explicit team-based Read permissions granted, along with the ad hoc command flag set, can also view ad hoc command job templates.

#### Launch

Super Users and Organization Admins (for job templates linked to the organization they are associated with) can launch job templates. Users granted team-based Run or Check level permissions or Users with team-based Create level permissions can launch job templates for which they are associated. If you can view it, you can start it.

- **System Job Templates:** Only Super Users can launch.
- **Ad hoc commands:** Super Users and Organization Admins (for job templates linked to the organization they are associated with) can launch ad hoc command job templates. Users and team members with explicit team-based Read permissions granted, along with the ad hoc command flag set, can also launch ad hoc command job templates.

#### Create

Super Users and Organization Admins (for job templates linked to the organization they are associated with) can create job templates. Users granted a team-based Create level permission can also create a job template.

- **System Job Templates:** Only Super Users can create.

#### Deletions

Super Users and Organization Admins (for job templates linked to the organization they are associated with) can remove job templates. Users granted team-based Create level permissions can also remove a job template. Anyone with permission to create a job template also has permission to remove it.

- **System Job Templates:** Only Super Users can remove.

### 14.3.12 Activity Stream

#### View

Activity Stream information is available as read-only. Users can only see activity on objects for which they have been granted permission to view. If a user cannot see the job template or the organization, they cannot view anything in the activity stream for those two events.

**Cleanup Job**

Only Super Users can access this activity stream.

---

**Note:** Activity Streams are only available to those with Enterprise-level licenses.

---

**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
  - setup menu, 9
- activity streams, 13
- ad hoc commands, 77
  - inventories, 77
- add new
  - inventories, 55
  - scan job, 56
- adding new
  - credentials, 37
- admin of organization
  - users, 26
- administrators
  - organization, 18
- Amazon Web Services
  - credential types, 41
  - inventories, 66
- Ansible Galaxy, 52
- Ansible Galaxy integration
  - features, 3
- automation
  - features, 2
- autoscaling
  - best practices, 107
- autoscaling flexibility
  - features, 2
- AWS
  - cloud credentials, 89

**B**

- backup and restore
  - features, 3
- best practices, 106
  - autoscaling, 107
  - deployment, continuous, 107
  - dynamic inventory sources, 106
  - file and directory structure, 106
  - host counts, larger, 107
  - integration, continuous, 107
  - source control, 106
  - variable inventory management, 107

**C**

- callbacks
  - extra variables, 95
- check
  - job types, 85
- cloud credentials
  - AWS, 89
  - Google, 89
  - job templates, 88
  - MS Azure, 89
  - OpenStack, 88
  - Rackspace, 89
  - VMware, 90
- cloud flexibility
  - features, 2
- components
  - licenses, 6
- concurrency
  - jobs, 104
- credential types, 38
  - Amazon Web Services, 41
  - Google Compute Engine, 43
  - machine, 38
  - Microsoft Azure, 44
  - OpenStack, 45
  - rackspace, 42
  - source control, 40
  - VMware, 42
- credentials, 36
  - adding new, 37
  - getting started, 36
  - how they work, 36
  - setup menu, 9
  - teams, 29
  - types, 38
  - users, 22
- custom
  - fact scan job, 62
  - scan job, 61
- custom fact scans
  - playbook, 62
  - system tracking, 62



custom script  
inventories, 72

## D

dashboard, 11  
host count, 12  
host status, 12  
job status, 12  
jobs tab, 12  
main menu, 8  
schedule status, 12

DEB files  
licenses, 6

deployment, continuous  
best practices, 107

dynamic inventory sources  
best practices, 106

## E

extra variables  
callbacks, 95  
provisioning callbacks, 95  
surveys, 92

extra\_vars, 92

## F

fact scan job  
custom, 62  
playbook, 62  
fact scan playbook  
system tracking, 62

features, 3  
Ansible Galaxy integration, 3  
automation, 2  
autoscaling flexibility, 2  
backup and restore, 3  
cloud flexibility, 2  
OpenStack inventory support, 3  
overview, 2  
real-time playbook, 2  
remote command execution, 3  
RESTful API, 3  
role-based access control, 2  
system tracking, 3

file and directory structure  
best practices, 106

forks  
jobs, 104

functionality  
PRoot, 109

## G

Galaxy support, 52  
getting started

credentials, 36

Google  
cloud credentials, 89  
Google Compute Engine  
credential types, 43  
inventories, 68

## H

host count  
dashboard, 12  
host counts, larger  
best practices, 107  
host status  
dashboard, 12  
host to host  
scan job, 83  
system tracking, 83

hosts  
inventories, 76

hosts, add new  
inventories, 77

how they work  
credentials, 36

## I

installation bundle  
licenses, 6

integration, continuous  
best practices, 107

inventories, 54  
ad hoc commands, 77  
add new, 55  
Amazon Web Services, 66  
custom script, 72  
Google Compute Engine, 68  
groups, 63  
add new, 64  
groups and hosts, 63  
hosts, 76  
hosts, add new, 77  
Microsoft Azure, 69  
OpenStack, 71  
Rackspace Cloud Servers, 65  
scan job, 79  
scan job templates, creating, 56  
scan job templates, custom, 61  
scan job templates, launching, 59  
scan job templates, scheduling, 60  
scheduling, 73  
scheduling, add new, 74  
system tracking, 79  
VMware vCenter, 70

inventory scripts  
setup menu, 9

## J

- job status
  - dashboard, 12
- job templates, 85
  - cloud credentials, 88
  - job variables, 92
  - jobs, launching, 95
  - portal mode, 11
  - provisioning callbacks, 93
  - relaunch, 93
  - scheduling, 96, 97
  - survey creation, 90
  - survey extra variables, 92
  - survey optional questions, 92
  - surveys, 90
- job templates, hierarchy, 92
- job templates, overview, 92
- job types
  - check, 85
  - run, 85
  - scan, 85
- job variables
  - job templates, 92
- jobs, 99
  - concurrency, 104
  - events summary, 103
  - forks, 104
  - host events, 102
  - host summary, 104
  - job status, 101
  - plays, 102
  - portal mode, 11
  - results, 99
  - status, 101
  - tasks, 102
- jobs tab
  - dashboard, 12
- jobs, launching
  - job templates, 95

## L

- launching
  - scan job, 59
- license, 3, 4
  - features, 5
  - nodes, 5
  - trial, 4
  - types, 4
- license features, 3
- license, viewing, 9
- licenses
  - components, 6
  - DEB files, 6
  - installation bundle, 6

- RPM files, 6
- logging in, 7

## M

- machine
  - credential types, 38
- main menu
  - dashboard, 8
- management jobs
  - setup menu, 9
- Microsoft Azure
  - credential types, 44
  - inventories, 69
- MS Azure
  - cloud credentials, 89

## N

- new schedule addition
  - projects, 52

## O

- OpenStack
  - cloud credentials, 88
  - credential types, 45
  - inventories, 71
- OpenStack inventory support
  - features, 3
- organization
  - administrators, 18
  - setup menu, 9
- organizations, 14
  - users, 16, 26
- overview
  - features, 2

## P

- permissions
  - teams, 31
  - users, 24
- playbook
  - custom fact scans, 62
  - fact scan job, 62
- playbooks
  - manage manually, 49
  - projects, 49
  - PRoot settings, 108
  - sharing access, 108
  - sharing content, 108
  - source control, 49
- portal mode, 10
  - job templates, 11
  - jobs, 11
- projects, 47
  - add new, 48

- new schedule addition, 52
  - playbooks, 49
  - source control update, 50
  - teams, 33
- PRoot
  - functionality, 109
  - troubleshooting, 109
  - variables, 109
- PRoot settings
  - playbooks, 108
- provisioning callbacks
  - extra variables, 95
  - job templates, 93
- R**
- Rackspace
  - cloud credentials, 89
- rackspace
  - credential types, 42
- Rackspace Cloud Servers
  - inventories, 65
- RBAC
  - activity stream, 113
  - credentials, 111
  - groups, 111
  - hosts, 110
  - inventories, 110
  - inventory updates, 111
  - job templates, 113
  - organizations, 110
  - project updates, 112
  - projects, 112
  - security, 109
  - teams, 112
  - users, 109
- real-time playbook
  - features, 2
- relaunch
  - job templates, 93
- remote command execution
  - features, 3
- RESTful API
  - features, 3
- role-based access control
  - features, 2
- role-based access controls, 109
  - activity stream, 113
  - credentials, 111
  - groups, 111
  - hosts, 110
  - inventories, 110
  - inventory updates, 111
  - job templates, 113
  - organizations, 110
  - project updates, 112
  - projects, 112
  - teams, 112
  - users, 109
- RPM files
  - licenses, 6
- run
  - job types, 85
- S**
- scan
  - job types, 85
- scan job
  - add new, 56
  - custom, 61
  - host to host, 83
  - inventories, 79
  - launching, 59
  - scheduling, 60
  - single host, 79
- scan job templates, creating
  - inventories, 56
- scan job templates, custom
  - inventories, 61
- scan job templates, launching
  - inventories, 59
- scan job templates, scheduling
  - inventories, 60
- schedule status
  - dashboard, 12
- scheduling
  - add new, 97
  - inventories, 73
  - job templates, 96, 97
  - scan job, 60
- scheduling, add new
  - inventories, 74
- security, 108
  - RBAC, 109
- setup menu
  - about Tower, 9
  - credentials, 9
  - inventory scripts, 9
  - management jobs, 9
  - organization, 9
  - teams, 9
  - users, 9
  - view license, 9
- sharing access
  - playbooks, 108
- sharing content
  - playbooks, 108
- single host
  - scan job, 79

- system tracking, 79
- source control
  - best practices, 106
  - credential types, 40
- source control update
  - projects, 50
- support, 3, 4
- survey extra variables
  - job templates, 92
- surveys
  - creation, 90
  - extra variables, 92
  - job templates, 90
  - optional questions, 92
- system tracking
  - custom fact scans, 62
  - fact scan playbook, 62
  - features, 3
  - host to host, 83
  - inventories, 79
  - scan job, 85
  - single host, 79

## T

- teams, 28
  - credentials, 29
  - permissions, 31
  - projects, 33
  - setup menu, 9
  - users, 27, 34
- Tower portal mode, 10
- Tower setup menu, 9
- Tower user menu, 8
- troubleshooting
  - PRoot, 109

## U

- updates, 4
- user menu, 8
- users, 20
  - admin of organization, 26
  - credentials, 22
  - organizations, 16, 26
  - permissions, 24
  - setup menu, 9
  - teams, 27, 34

## V

- variable inventory management
  - best practices, 107
- variable precedence, 92
- variables
  - PRoot, 109
- view license

- setup menu, 9
- VMware
  - cloud credentials, 90
  - credential types, 42
- VMware vCenter
  - inventories, 70