ANSIBLE

TOWER

Ansible Tower User Guide

Version 1.4.10

April 28, 2014

support@ansible.com

Table of Contents

Overview .... 3.
Licensing .... 4.
Updates and Support .... 4.
Requirements .... 4.
Release Notes .... 6.
Known Issues .... 7.
Release History .... 8.

Getting Started .... 9.
Installation and Setup .... 9.
Upgrade an Existing Tower Installation .... 13.
Quick Start .... 14.

User Guide .... 43.
Logging In .... 43.
Home .... 44.
Organizations .... 45.
Users .... 46.
Teams .... 57.
Permissions .... 64.
Credentials .... 65.
Projects .... 68.
Inventories .... 78.
Job Templates .... 94.
Jobs .... 101.
Best Practices .... 107.
Security .... 108.
Troubleshooting .... 108.

Glossary .... 110.
Using LDAP with Tower .... 111.
Administration of Tower .... 112.
API .... 113.

Overview

Tower

Ansible Tower is a web-based user interface and 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.

True "Push Button" Automation

Access your favorite projects and re-trigger execution from the web interface with a minimum of clicking. Tower will let you supply input variables, let you pick your credentials, will kick off and monitor the job, and allows you many views into the results and the history of your hosts over time.

Role Based Access Control

Ansible Tower allows you to delegate 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 dry run or live mode.

Cloud & Autoscaling Flexibility

Tower features a powerful 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 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.

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 (http://servername/api), showing off every resource and relation. Everything that can be done in the user interface can be done in the API - and more.

Licensing

Tower is a proprietary software product and is licensed on an annual subscription basis. There is no license fee for managing up to 10 hosts. Should you wish to acquire a license for additional servers or get support for the ones you have, please visit http://www.ansible.com/pricing/ for details or contact support@ansible.com for assistance.

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

Updates and Support

Tower is licensed as an annual subscription, which includes:

- Limited, Standard, or Premium (24x7) Support via web, email, and telephone with SLA
- All regular updates and releases of Tower and Ansible

For more information, please contact Ansible at support@ansible.com or at http://www.ansible.com/pricing/.

Requirements

Ansible Tower has the following minimum requirements:

- Supported Operating Systems:
    - Red Hat Enterprise Linux 6 64-bit
    - CentOS 6 64-bit
    - Ubuntu 12.04 LTS 64-bit
- Ansible (latest stable release)
- 2 GB RAM
- 20 GB hard disk
- For Amazon EC2:
    - Instance size of m1.large or larger
    - an instance size of m1.xlarge or larger is suggested if there are more than 100 hosts

While other operating systems may technically function, currently only this list are supported to host an Ansible Tower installation. If you have a firm requirement to run Tower on an unsupported operating system, please contact support@ansible.com. Management of other operating systems (nodes) is as documented by the Ansible project itself, and allows for a wider list.

Actual RAM requirements for will vary based on how many hosts Tower will manage simultaneously (which is controlled by the forks parameter in the system). To avoid possible resource conflicts, the following is recommended:

- 2 GB RAM - forks values up to 100
- 4 GB RAM - forks values up to 200
- 8 GB RAM - forks values up to 400

A larger number of hosts can of course be addressed, though if the fork number is less than the total host count, more passes across the hosts will be required. These RAM limitations are avoided when using rolling updates or when using the provisioning callback system built into Tower, where each system requesting configuration enters a queue and is processed as quickly as possible; or in cases where Tower is producing or deploying images such as AMIs. All of these are great approaches to managing larger environments. For further questions, please contact support@ansible.com.

NOTE: *For users of Red Hat Enterprise Linux or CentOS systems, SELinux can be set to disabled, permissive, or enforcing, but is only supported in "targeted" mode.*

NOTE: *Although Tower and Ansible are written in Python, they are full applications and not a simple Python library. Therefore Tower cannot be installed in a Python virtualenv or similar; you must install it as described in the installation instructions below.*

NOTE: *It is recommended to use the latest stable release of Ansible for best performance and to ensure the latest bugfixes are available. However, Ansible version 1.3 is supported for Ansible Tower 1.4.X*

The requirements for systems managed by Tower are the same as for Ansible at: http://docs.ansible.com/intro_getting_started.html

Release Notes

- Changes from 1.4.9
    - Correctly handle schedule creation when browser timezone cannot be detected.
    - Corrected pagination on job_events page.
- Changes from 1.4.8
    - Corrected a provisioning callback issue on Enterprise Linux.
    - Added a sample provisioning callback script.
    - Various backend and UI improvements.
- Changes from 1.4.5
    - Scheduling for Jobs, SCM updates, and Inventory synchronization has been added. The UI for each of these objects has changed to accommodate this new scheduling feature.
        - The jobs page has been overhauled to show completed, active, queued, and scheduled jobs.
        - Inventory and project synchronization jobs are now also shown on the jobs page.
    - Added support for Ansible Vault to Credentials. For more information on how to use Ansible Vault, please visit: http://docs.ansible.com/playbooks_vault.html.
- Changes from 1.4
    - AWX has been renamed to Tower!
    - The hosts and groups pages have been merged into a single view
    - Hosts may be copied into groups via "drag and drop"
    - Activity Stream is now a fully supported feature. The activity stream is accessed by an icon in the top right of most screens and shows what actions have been performed and by which users.
    - Performance of EC2 synchronization has been improved
    - Performance of event subsystems have been improved for larger volumes of host activity
- Changes from 1.3.1
    - Added new Home tab with dashboard view of job and host status
    - Added user interface for inventory synchronization with Amazon Web Services and Rackspace Cloud Servers. Configure this in the groups editor of the Inventories tab.
    - Moved all credentials to the Credentials tab, including SSH, SCM, and cloud management. You can now create and manage all credentials from the Credentials tab. Previously, credentials were

    owned by a project, not a particular user. Any existing SCM synchronization jobs will be migrated such that the credentials will be owned by the admin user. If you find you can no longer synchronize SCM-based projects, please review the credentials assigned to the admin user and change their ownership to the appropriate team.
  - SCM integration dialogs are simplified
  - The hosts and groups pages have a more unified the look and feel
  - Various pages have new red and green light icons to indicate status
  - Added the Activity Stream as a beta feature available only to admin users in this release. The activity stream is accessed by an icon in the top right of most screens and shows what actions have been performed and by which users.
- Changes from 1.2.2
  - Added integration with Source Code Management systems for importing and managing Tower project playbooks
  - Added integration with LDAP and Active Directory for Tower user management. Please, see the section Using LDAP with Tower for more information.
  - The inventory display has been revised to improve the user experience

Known Issues

1. The EPEL ansible package has an implicit dependency on python-setuptools, which is no longer explicitly required by the distro package. The distro packagers are fixing this problem. However, it may take a few days after this release for the mirrors to update. See the section Installation and Setup for a solution.

2. Installation of Tower through an HTTP proxy requires setting the environment variable "HTTP_PROXY" accordingly, before running setup.sh.

3. Ansible Tower implements a role based access control system. You may appear to be able to edit objects that do not belong to you (like being able to pull up an edit dialog on your team mates whom you already have permission to view). Don't worry, when you try to edit something, you'll get a 403 error, and you can't see any information you shouldn't already have access to as defined in the system.

4. If a job template is deleted while jobs that depend on it are running, the system may be left in a somewhat indeterminate state with some queued jobs remaining in the list. Simply delete these queued jobs via the delete button in the jobs view.

5. When changing a source control type for a project to a different type or URL, the project directory is not always automatically deleted. In order to change the source control type, you may need to explicitly check the "delete on update" flag and then run an update of the project.

Release History

The release history for this documentation is as follows:

Version	Date	Changes
1.4.9	April 17, 2014	Updated for release of Tower 1.4.9
1.4.8	April 7, 2014	Updated for release of Tower 1.4.8
1.4.5	February 10, 2014	Updated for release of Tower 1.4.5
1.4.0	November 21, 2013	Updated for release of Tower 1.4
1.3.0	September 13, 2013	Updated for release of Tower 1.3
1.2.2	July 31, 2013	Initial Release

Getting Started

Welcome to Ansible Tower!

To get started, first follow the installation instructions in the section entitled Installation and Setup. Then, either walk through the quick start below to quickly get up and running with Tower or browse through the documentation and design an implementation plan that works for you.

We value your feedback. Please contact us at support@ansible.com and let us know what you think and your ideas for future features!

Installation and Setup

You can expect the installation of Tower to take less than fifteen minutes, depending on the speed of your network connection. (This installation will require that the Tower server be able to access the Internet.)

At the end of the installation, you will use your web browser to access Tower and utilize all of its capabilities.

NOTE: *Although Tower and Ansible are written in Python, they are full applications and not a simple Python library. Therefore Tower cannot be installed in a Python virtualenv or similar; you must install it as described in the installation instructions below.*

1. Install Ansible

Use 1.3.x or later as detailed in the Ansible documentation at:

http://docs.ansible.com/intro_installation.html

For convenience, we'll summarize those installation instructions here:

For Red Hat Enterprise Linux 6 and CentOS 6:

- Configure the EPEL repository

```
root@localhost:~$ yum install \
   http://mirror.oss.ou.edu/epel/6/x86_64/epel-release-6-8.noarch.rpm
```

- Install Ansible

```
root@localhost:~$ yum install ansible
```

For Ubuntu 12.04:

- Install Ansible dependencies

```
root@localhost:~$ apt-get install python-yaml python-paramiko python-jinja2 python-pip
```

- Install Ansible

```
root@localhost:~$ pip install ansible
```

2. Download Tower:

Download Ansible Tower by filling out the form at http://www.ansible.com/tower. After completing the form, you will receive an email containing the link to the Tower installation tarball.

3. Extract the tarball

Then `cd` into the setup directory. Replace the string `VERSION` in the commands below with the version of Tower that you are installing e.g., "1.4.8".

```
root@localhost:~$ tar xvzf awx-setup-latest.tar.gz
root@localhost:~$ cd awx-setup-VERSION
```

4. Change the default database password

Edit the file `group_vars/all`. Modify the variable `pg_password` to change the default database password.

NOTE: *The password should not contain quotes.*

5. LDAP / Active Directory (optional)

If you wish to setup LDAP / Active Directory authentication for Tower, please review the section Using LDAP with Tower.

6. Run the installation script

From the awx-setup-VERSION directory, run setup.sh

<<<<<<< Local Changes ***NOTE:*** *For users of Red Hat Enterprise Linux or CentOS, PackageKit can frequently interfere with the update mechani=======* ***NOTE:*** *For users of Red Hat Enterprise Linux or CentOS, PackageKit can frequently interfere with the update mechanism. Consider disabling or removing PackageKit if installed prior to running the setup process.*

NOTE: *Installation of Tower through an HTTP proxy requires setting the environment variable "HTTP_PROXY" accordingly, before running setup.sh.*

NOTE: *The EPEL ansible package has an implicit dependency on python-setuptools, which is no longer explicitly required by the distro package. The distro packagers are fixing this problem. However, it may take a few days for the mirrors to update. As a result, when attempting to install tower, you may encounter the following error:*

```
$ ./setup.sh
Traceback (most recent call last):
  File "/usr/bin/ansible-playbook", line 22, in
    import pkg_resources
ImportError: No module named pkg_resources
```

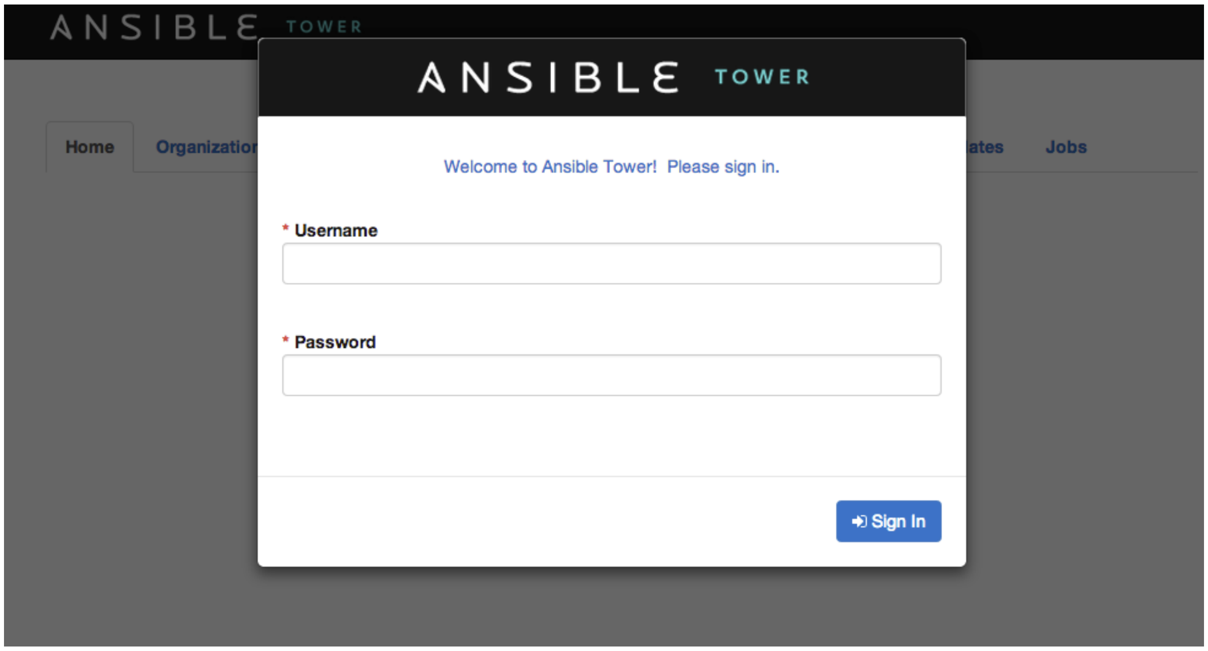
The workaround for this issue is to install the package python-setuptools:

```
yum install python-setuptools
```

```
root@localhost:~$ ./setup.sh
```

Setup will install Tower from RPM or Deb packages using repos hosted on ansible.com.

When setup completes successfully, you should be able to point your web browser to the Tower server and see the Tower login screen.



If the installation of Tower fails or if you need assistance, please contact us at support@ansible.com. Ansible subscription customers will receive a faster response by filing a support issue.

Upgrade an Existing Tower Installation

You can upgrade your existing Tower installation the latest version by running the setup playbook for the new version of Tower. All data will be preserved. However, it is important that if you changed any of the parameters in the file `group_vars/all` (e.g. `pg_password`) that you make the same changes to the new `group_vars/all` file.

You can expect the upgrade of Tower to take less than fifteen minutes, depending on the speed of your network connection. (This installation will require that the Tower server be able to access the Internet.)

At the end of the upgrade, you will use your web browser to access the Tower server and utilize all of its capabilities.

This upgrade procedure assumes that you have a working installation of Ansible and Tower.

1. Backup the existing Tower database

```
root@localhost:~$ awx-manage dumpdata > backup.json
```

2. Download Tower

Download Ansible Tower by filling out the form at http://www.ansible.com/tower. After completing the form, you will receive an email containing the link to the Tower installation tarball.

3. Extract the tarball

Then `cd` into the setup directory. Replace the string `VERSION` in the commands below with the version of Tower that you are installing e.g., "1.4.0".

```
root@localhost:~$ tar xvzf awx-setup-latest.tar.gz
root@localhost:~$ cd awx-setup-VERSION
```

4. Change the default database password

Edit the file `group_vars/all`. Modify the variable `pg_password` to change the default database password.

NOTE: *The password should not contain quotes.*

5. LDAP / Active Directory (optional)

If you wish to setup LDAP / Active Directory authentication for Tower, please review the section Using LDAP with Tower.

6. Run the installation script

From the awx-setup-VERSION directory, run setup.sh

NOTE: *For users of Red Hat Enterprise Linux or CentOS, PackageKit can frequently interfere with the update mechanism. Consider disabling or removing PackageKit if installed prior to running the setup process.*

```
root@localhost:~$ ./setup.sh
```

Setup will install Ansible from RPM or Deb packages using repos hosted on Ansible.com.

When setup completes successfully, you should be able to point your web browser to the Tower server and see the Tower login screen.

If the upgrade of Tower fails or if you need assistance, please contact us at support@ansible.com. Ansible subscription customers will receive a faster response by filing a support issue.

Quick Start

After the installation of Tower is complete, we'll complete the following tasks to quickly set up and launch our first Ansible playbook using Tower. This first playbook launch will execute simple Ansible tasks to teach you how to use Tower and also ensure Tower is setup properly.

Here's a summary of the tasks we'll need to accomplish:

1. Login as Super User
2. Create an Organization
3. Add a new User to the organization
4. Add an Inventory to the organization
5. Create a set of Credentials
6. Create a Project
7. Create a new Job Template using an Ansible example playbook
8. Launch it!

You can expect the Quick Start to take less than thirty minutes, from beginning to end. At the end of the Quick Start, you'll have a functioning Tower that you can use to launch more sophisticated playbooks.

For the Quick Start, you will need to have completed the Tower installation and you will also need a target system to deploy the playbook to. This can be any sort of system that can be managed by Ansible.

NOTE: *The requirements for a system to be managed by Ansible are at http://docs.ansible.com/intro_installation.html.*

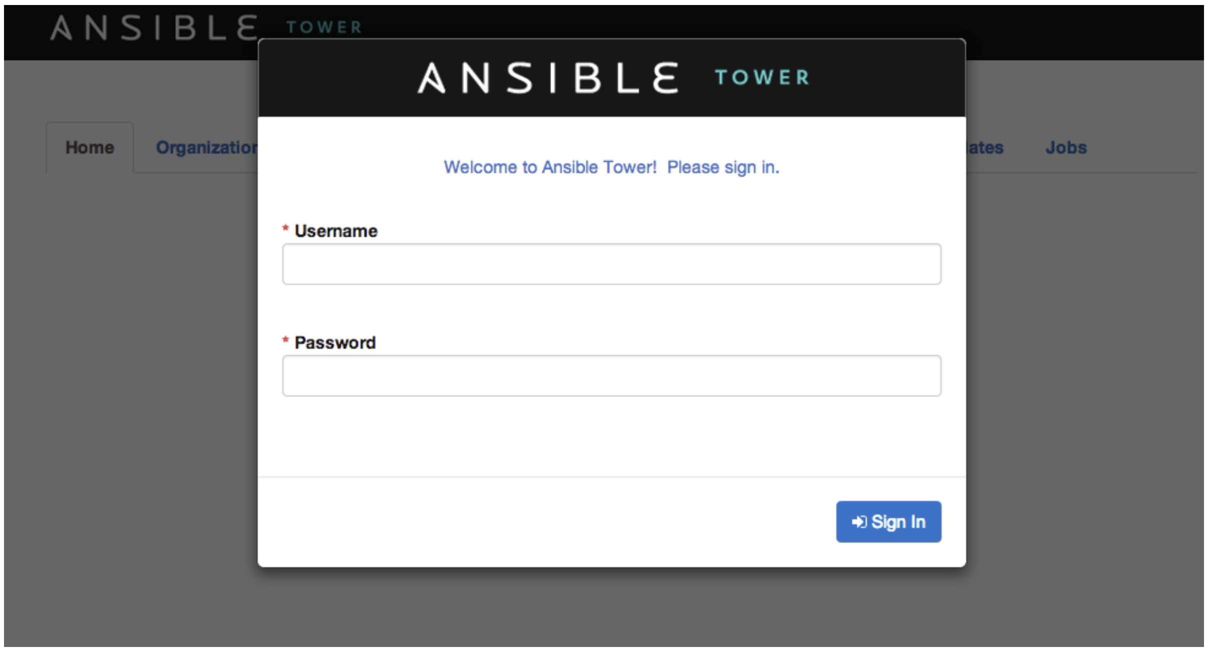
Ready? Let's go!

1. Login as Super User

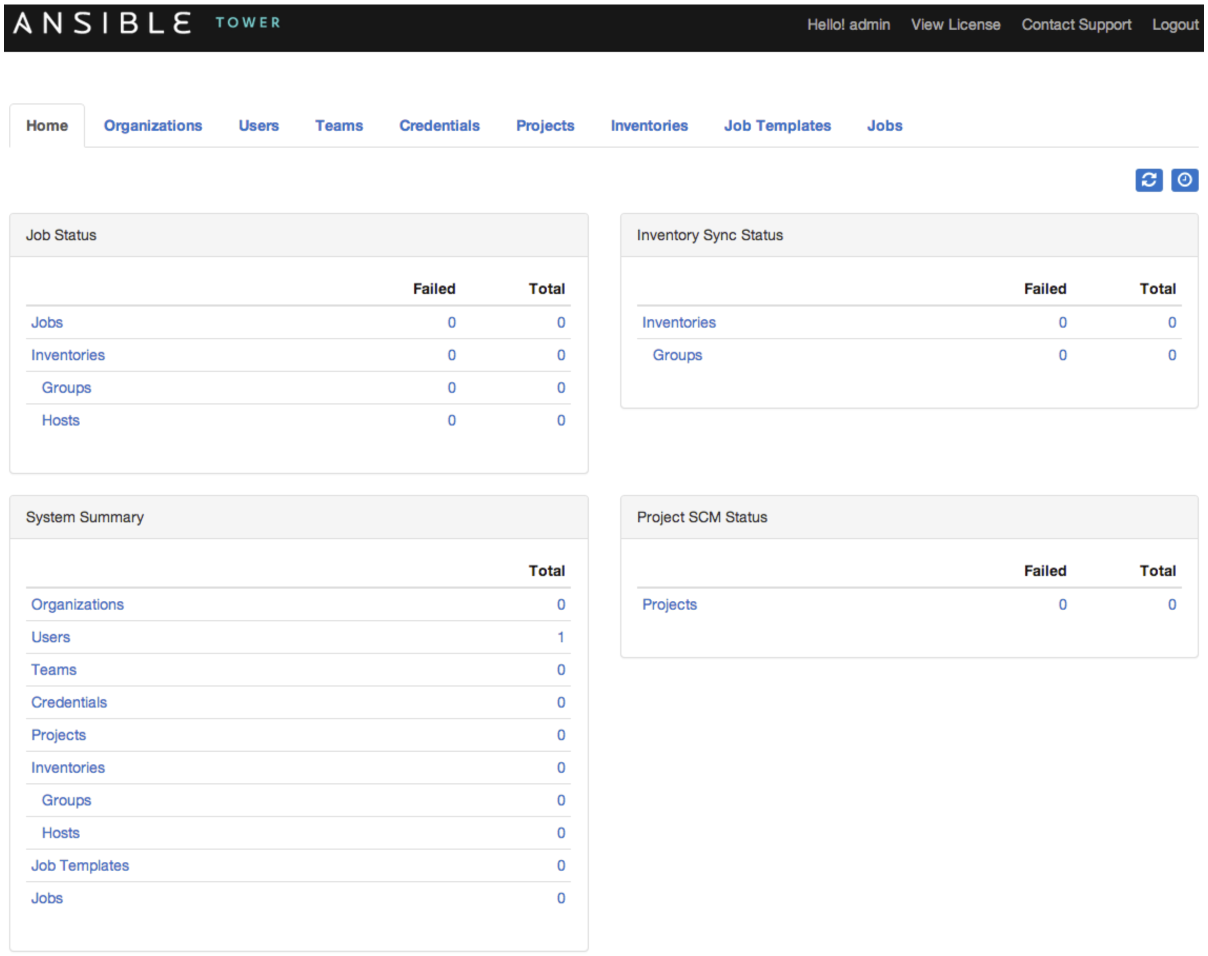
First, log in to Tower by browsing to the Tower server URL at `http://<Tower server name>/`

Log in using the username and password set during the installation process. By default, this will be username: "admin" and password: "password". You can change this by clicking on the "admin" account on the users tab.

NOTE: *We'll get into the details of the differences between a normal user, superuser, and organization administrator in the section Users.*



From this main interface, we can access all aspects of Tower, including **Organizations**, **Users**, **Teams**, **Projects**, **Inventories**, **Credentials**, **Job Templates**, and **Jobs**.



Job Status

	Failed	Total
Jobs	0	0
Inventories	0	0
Groups	0	0
Hosts	0	0

Inventory Sync Status

	Failed	Total
Inventories	0	0
Groups	0	0

System Summary

	Total
Organizations	0
Users	1
Teams	0
Credentials	0
Projects	0
Inventories	0
Groups	0
Hosts	0
Job Templates	0
Jobs	0

Project SCM Status

	Failed	Total
Projects	0	0

Keep in mind that the goal of this Quick Start is to launch a simple playbook. In order to do so, we'll need to set up a number of configuration options, but doing so now will ensure Tower is configured properly and allow us to easily execute more involved playbooks later while taking advantage of all the flexible role-based access control that Tower provides. You'll also get to know more about Tower along the way.

Tower provides multiple levels of role-based access, providing delegation of responsibility, but with fine - grained control over who can do what. We'll talk about that in more detail later in this document. For now, here's a simplified outline that shows the hierarchy of Tower's role based access control and the relationship between each element.

Tower Hierarchy

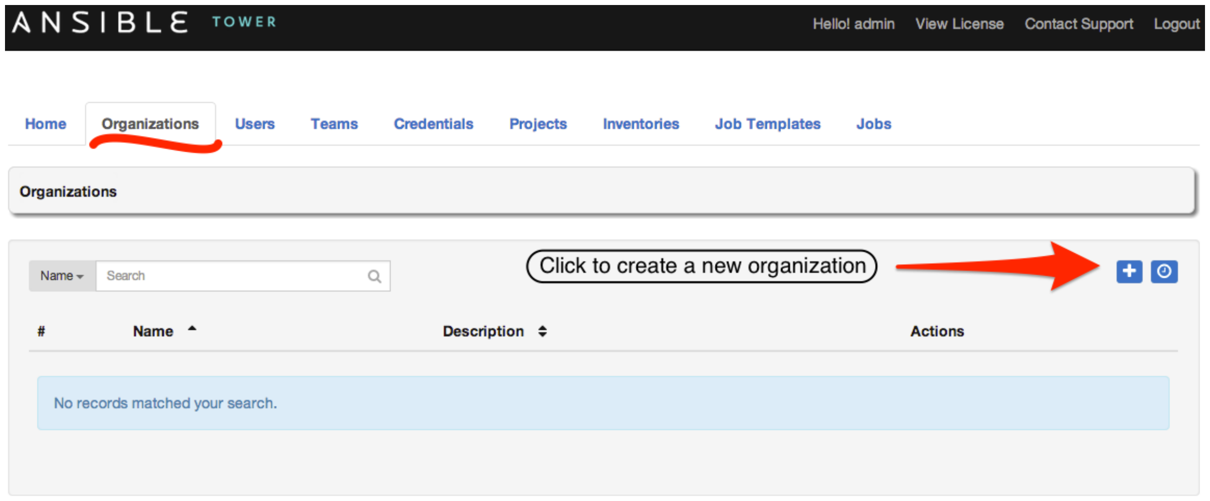
- Organization
    - Inventories
        - Groups
            - Hosts
    - Teams
        - Credentials
        - Permissions
        - Users
            - Credentials
            - Permissions
- Projects
    - Playbooks
    - Job Templates
- Jobs

Now, let's create a new organization within which we can create our first user, detail our inventory of hosts, and store SSH credentials for those hosts.

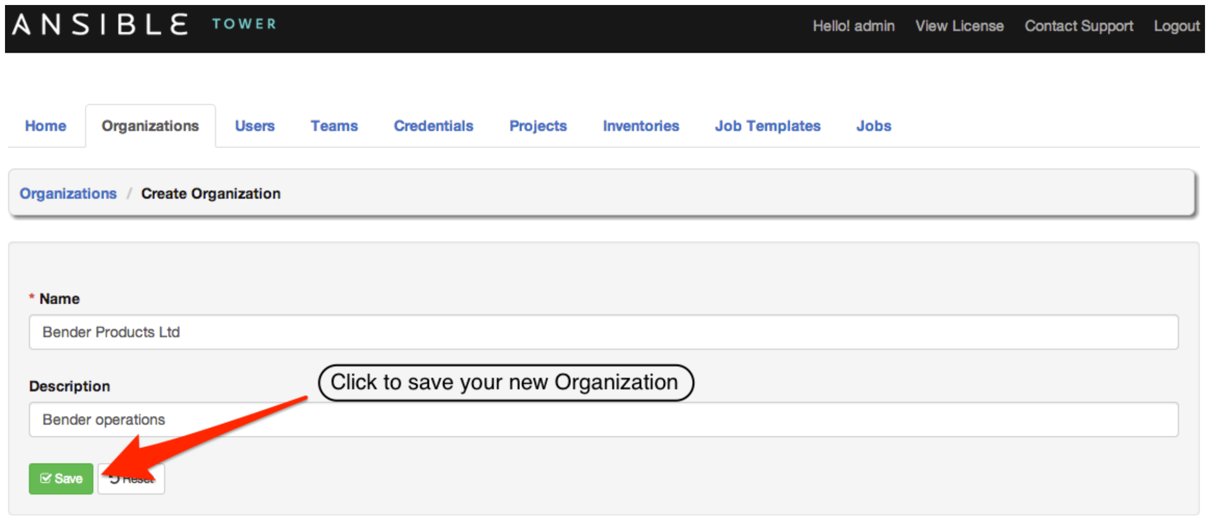
2. Create an Organization

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

Then click the ➕ button.



Enter a simple name and description for the organization. You can edit both of these fields later, so the values aren't critical. For our example, we will create an organization for a fictitious company called Bender Products Ltd.



* Name

Bender Products Ltd

Description

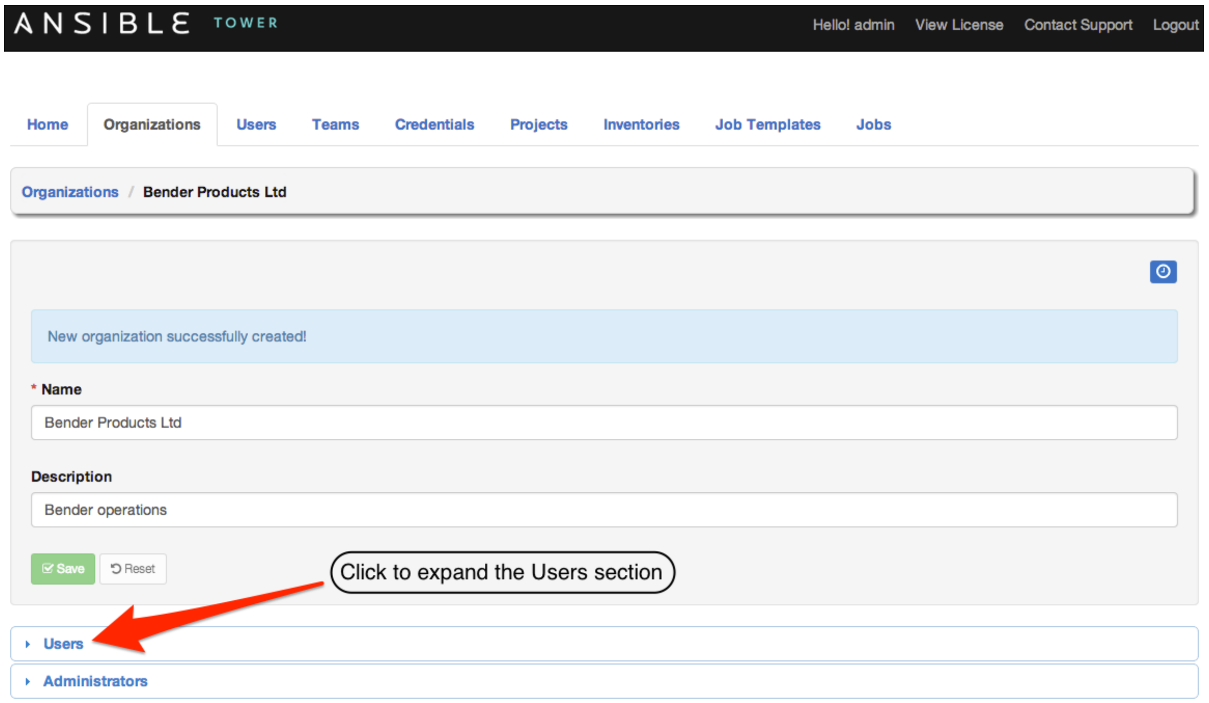
Bender operations

Organizations have both normal users and organization administrators. Organization Administrators are able to modify the membership and other properties of the organization, whereas normal users cannot. They are essentially super users but only within the scope of that organization. For more about the differences between users and administrators, see the section on Users.

The "admin" user is a Super User account -- a de-facto administrator for all organizations, so let's use our admin powers to create a new user and add it to our new organization. When creating a new user, the checkbox **Superuser?** corresponds to this level of access. Only Super Users can create other Super Users or promote existing users to this level.

3. Create a new user and add the user to the organization

Expand the **Users** section (not the Users tab!) as shown here:



New organization successfully created!

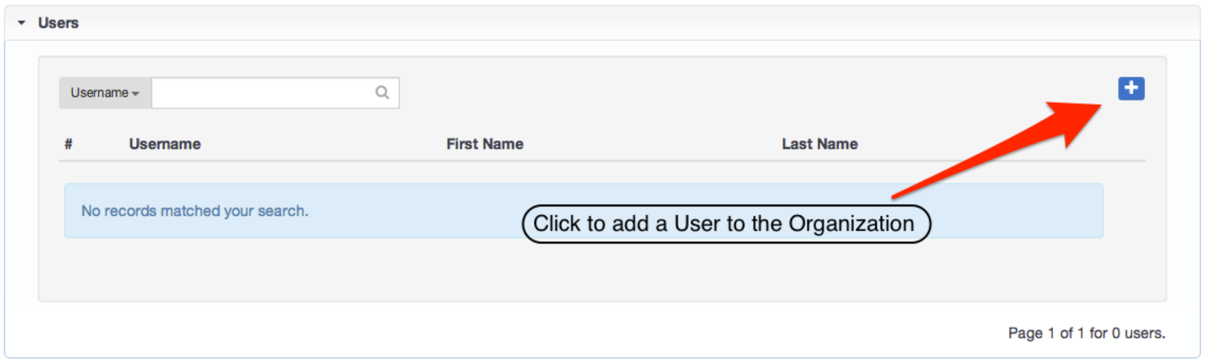
* Name

Bender Products Ltd

Description

Bender operations

Add a user by clicking the **+Add** button.

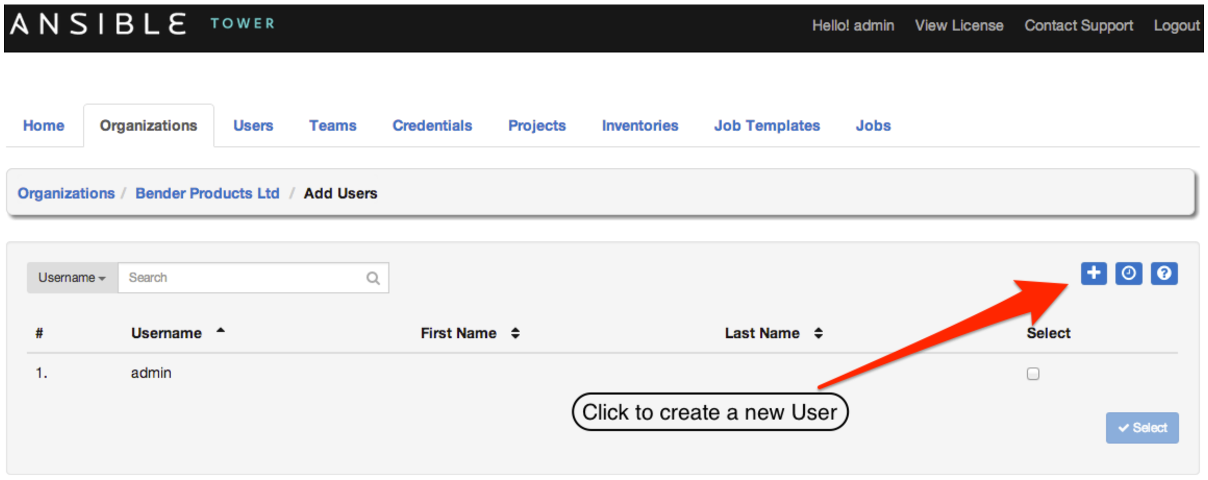


#	Username	First Name	Last Name

No records matched your search.

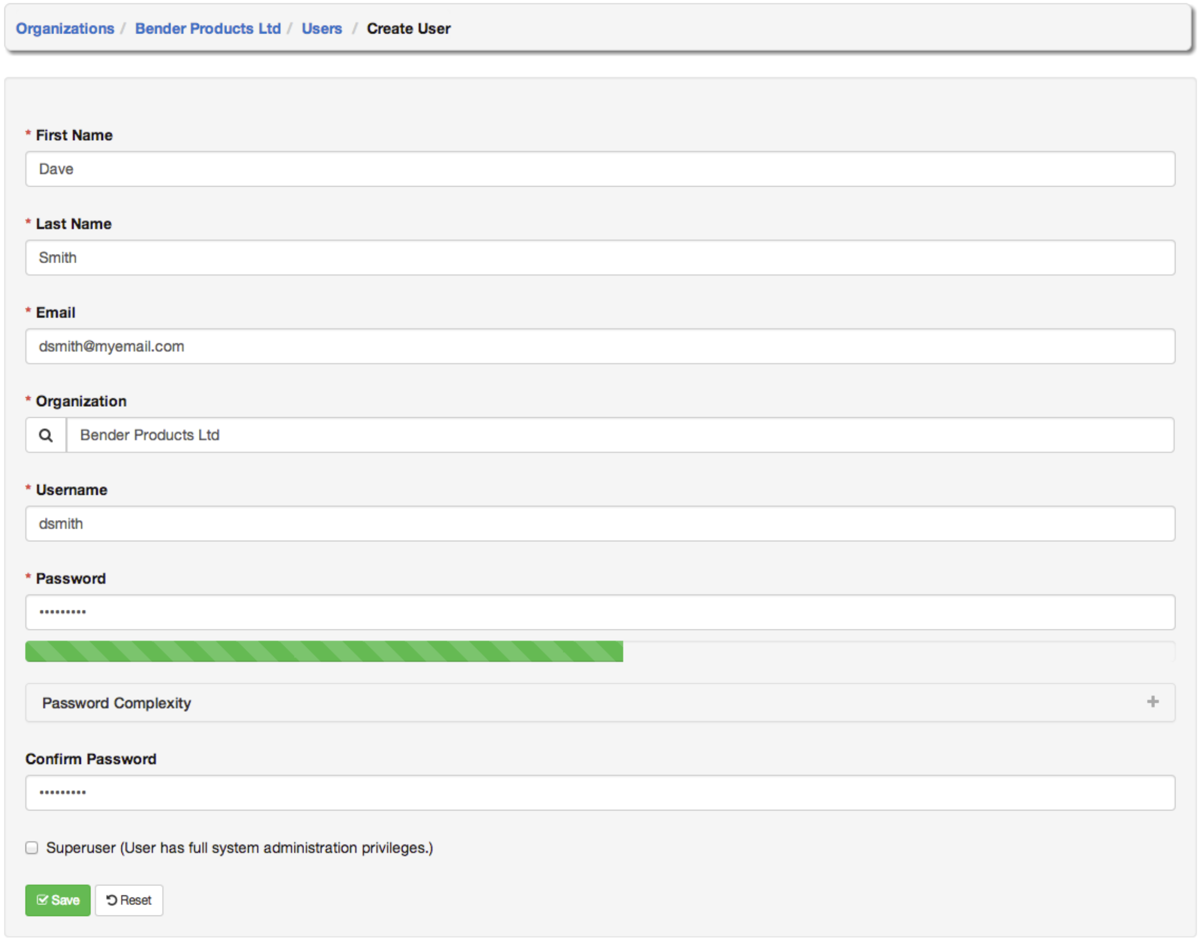
Page 1 of 1 for 0 users.

A list of all existing users will be presented. Since we have not created any users, the only user listed is "admin". Click the ✚ button to create a brand new user.



#	Username	First Name	Last Name	Select
1.	admin			☐

Enter the user's details.

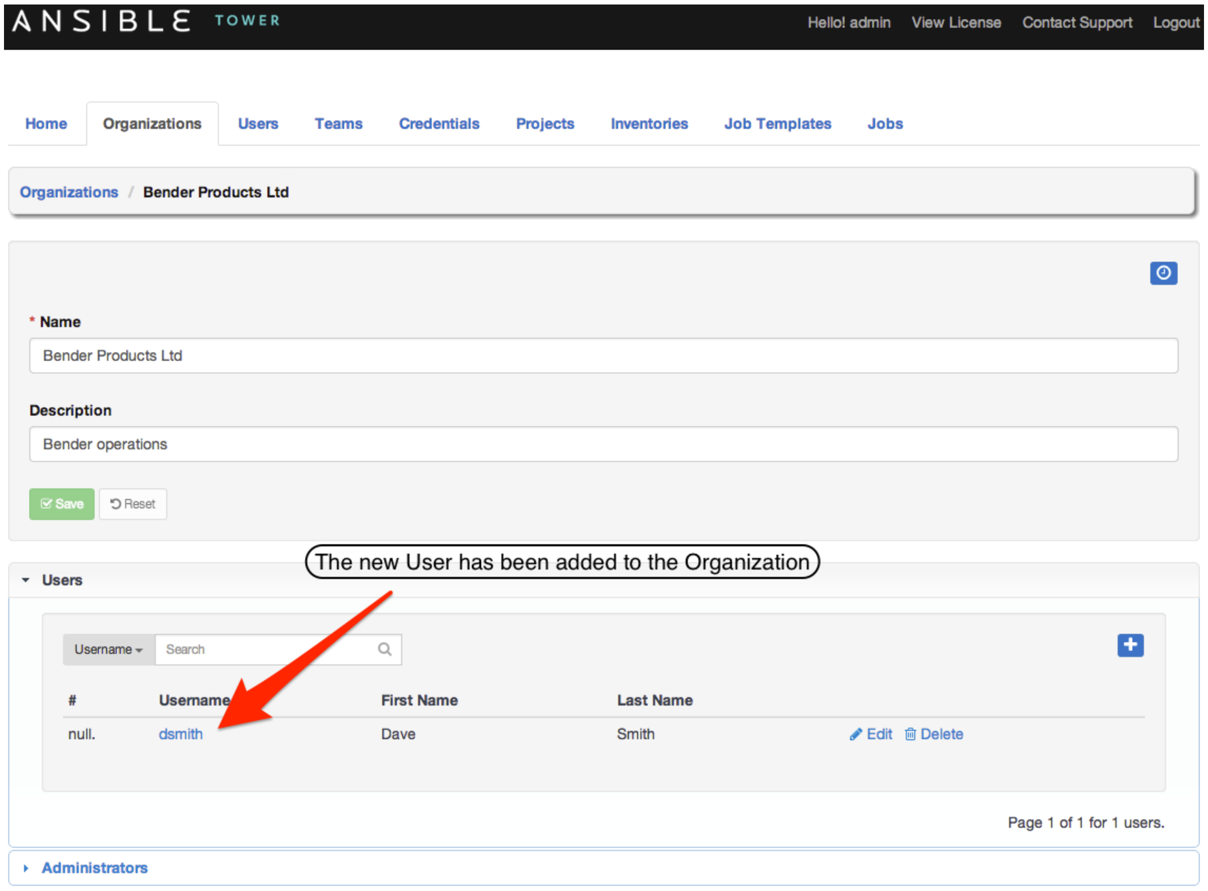


* First Name: Dave
* Last Name: Smith
* Email: dsmith@myemail.com
* Organization: Bender Products Ltd
* Username: dsmith
* Password: •••••••••

Confirm Password: •••••••••

☐ Superuser (User has full system administration privileges.)

Click the **Save** button to save the user. You will be taken back to the organization details, where the new user we just created now appears on the list.



* Name: Bender Products Ltd

Description: Bender operations

#	Username	First Name	Last Name
null.	dsmith	Dave	Smith

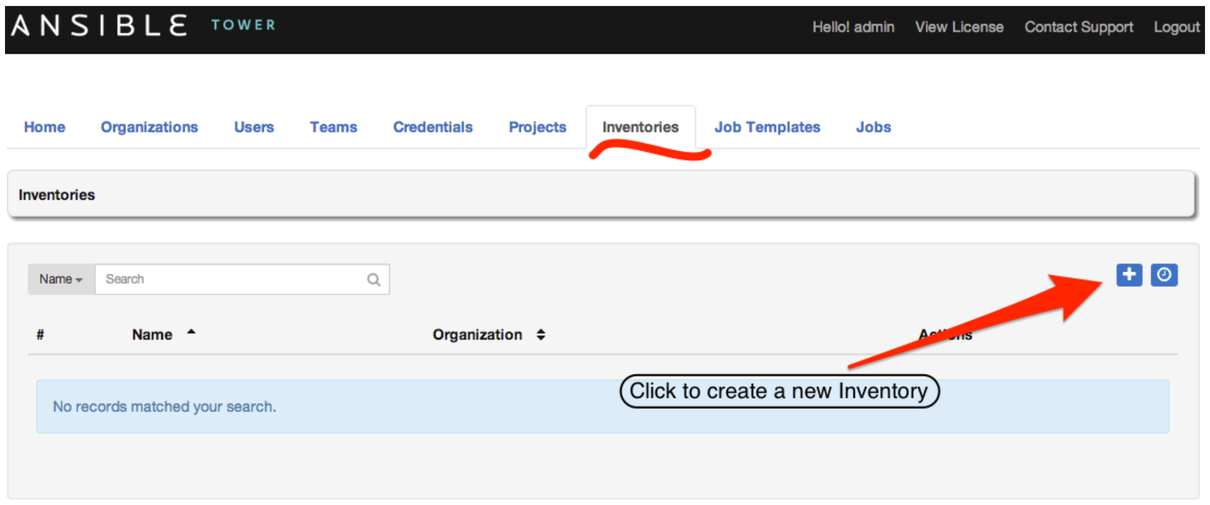
Page 1 of 1 for 1 users.

Now, we have an organization and a user. Let's add an inventory of hosts we'll be managing for Bender Products.

4. Create a new inventory and add it to the organization

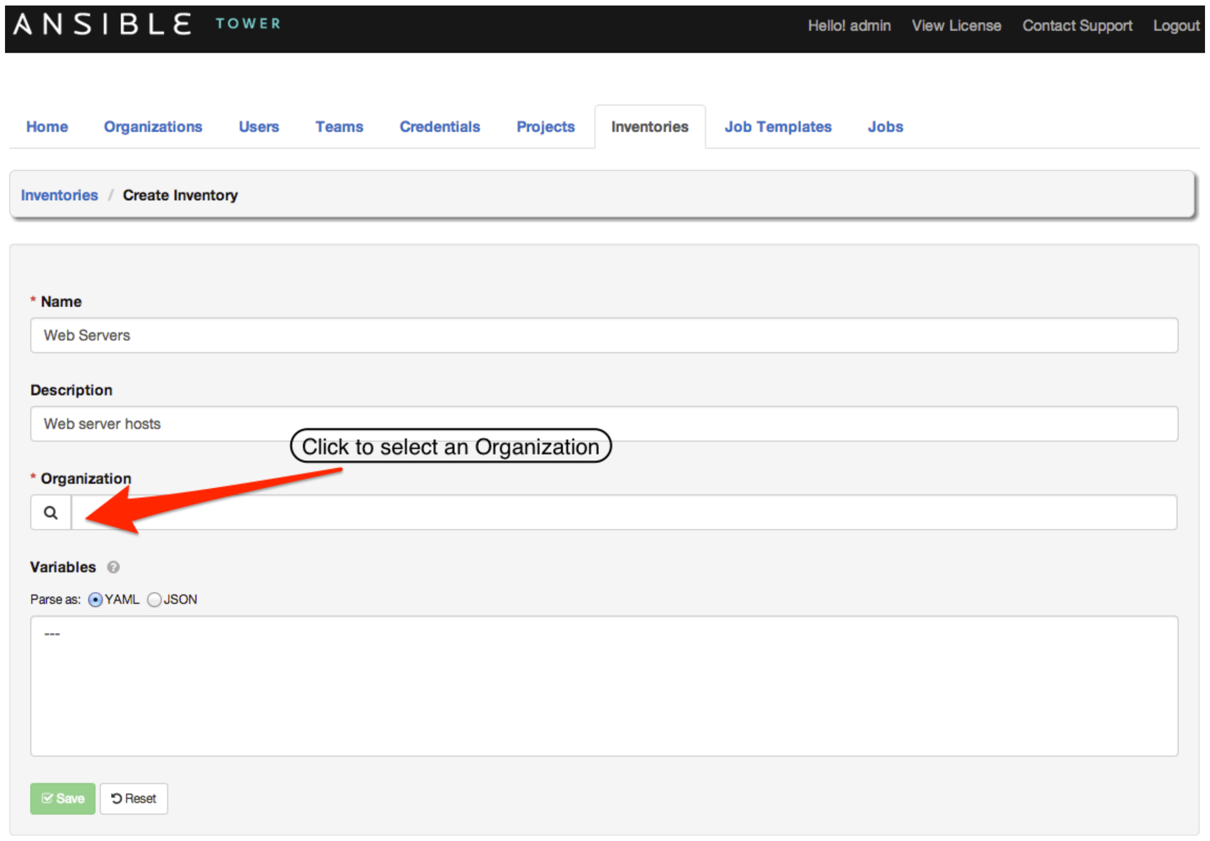
An inventory is a collection of hosts that can be managed with Tower. Inventories are assigned to organizations and permission to launch playbooks against inventories is controlled at the user and team level. More information can be found in the Inventories and Permissions sections.

Create a new inventory by browsing to the **Inventories** tab and clicking the ➕ button.



No records matched your search.

Enter the values for **Name** and **Description**. For this example, the name of our inventory will be Web Servers. Then click the look-up button to the left of the **Organization** field to select the organization that this inventory should belong to.



* Name
Web Servers

Description
Web server hosts

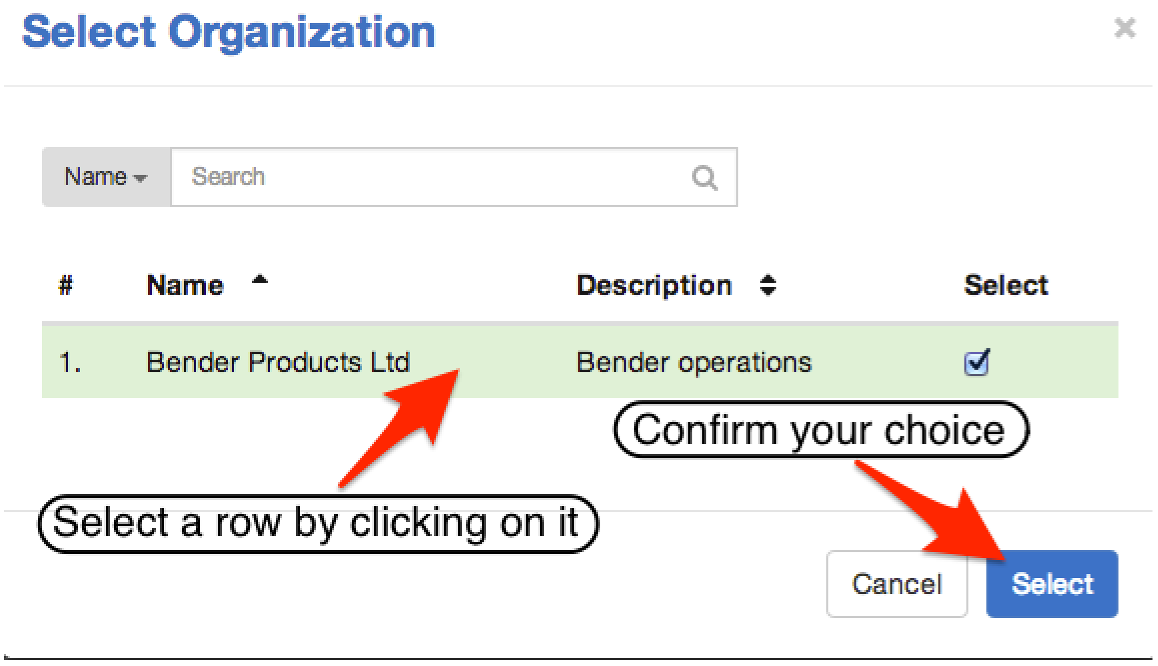
* Organization

Variables

Parse as: [✓ YAML] JSON

```
---
```

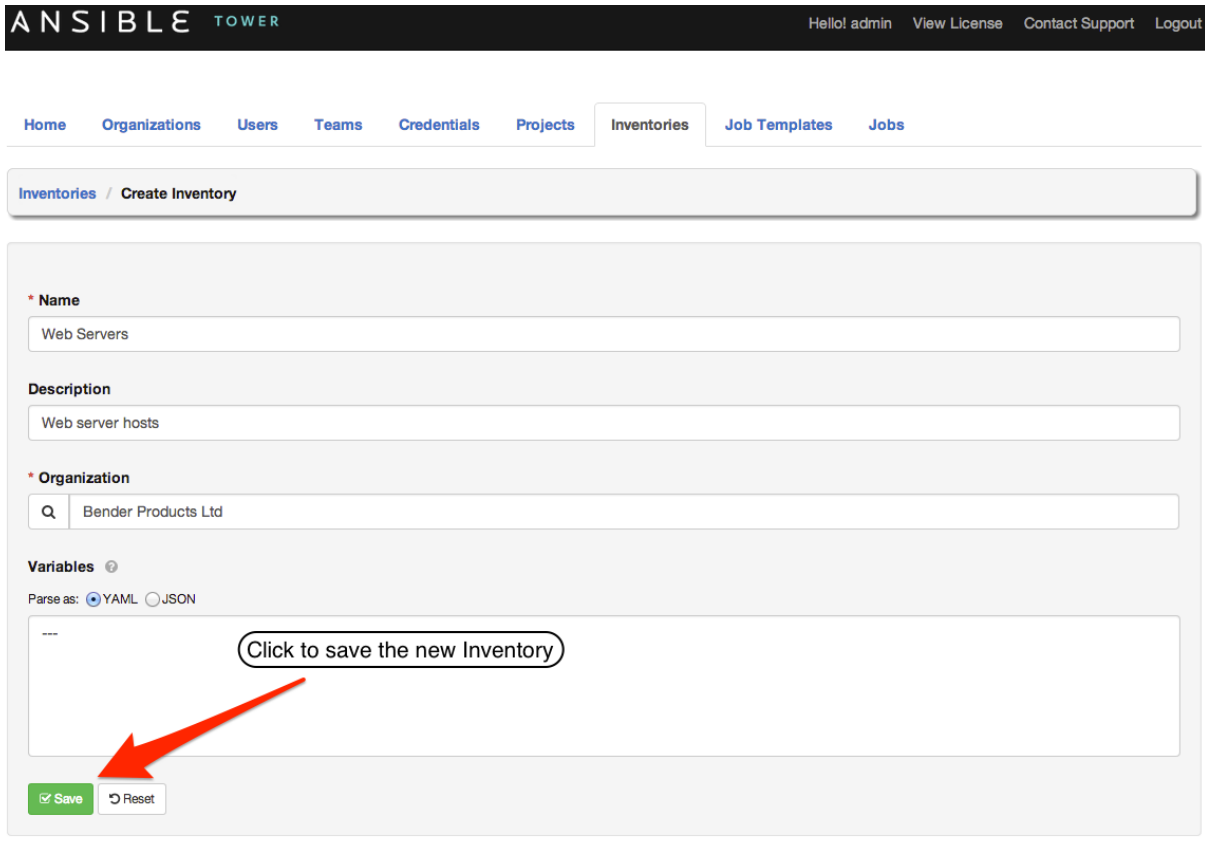
For this example we'll use the organization we created earlier. Select the row from the list by clicking on it. The selected row will be highlighted. Click the **Select** button to confirm your choice.



Select Organization

#	Name	Description	Select
1.	Bender Products Ltd	Bender operations	✓

We will discuss variables in more detail later. For now, leave the **Variables** field alone. Click the **Save** button at the bottom of the page to create the inventory.

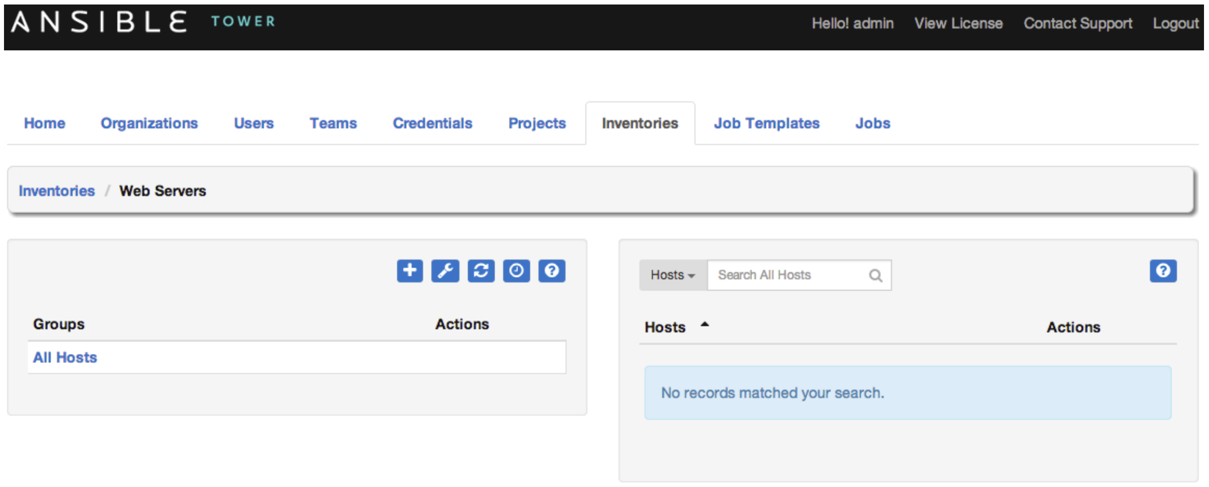


Inventories / Create Inventory

* Name	Web Servers
Description	Web server hosts
* Organization	Bender Products Ltd
Variables	Parse as: [✓ YAML] JSON

```
---
```

After clicking **Save**, you will the **Inventories** screen for the Web Servers inventory.

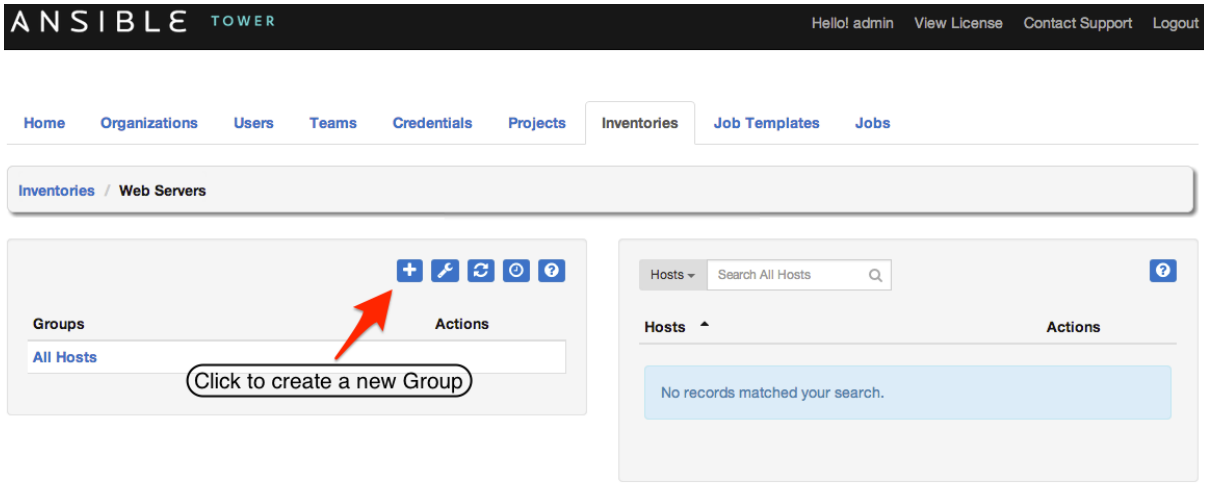


No records matched your search.

Inventories are divided into groups and groups contain hosts. A group might represent a particular environment (e.g. "Datacenter 1" or "Stage Testing"), a type of server (e.g. "Application Servers" or "DB Servers"), or any representation of your environment.

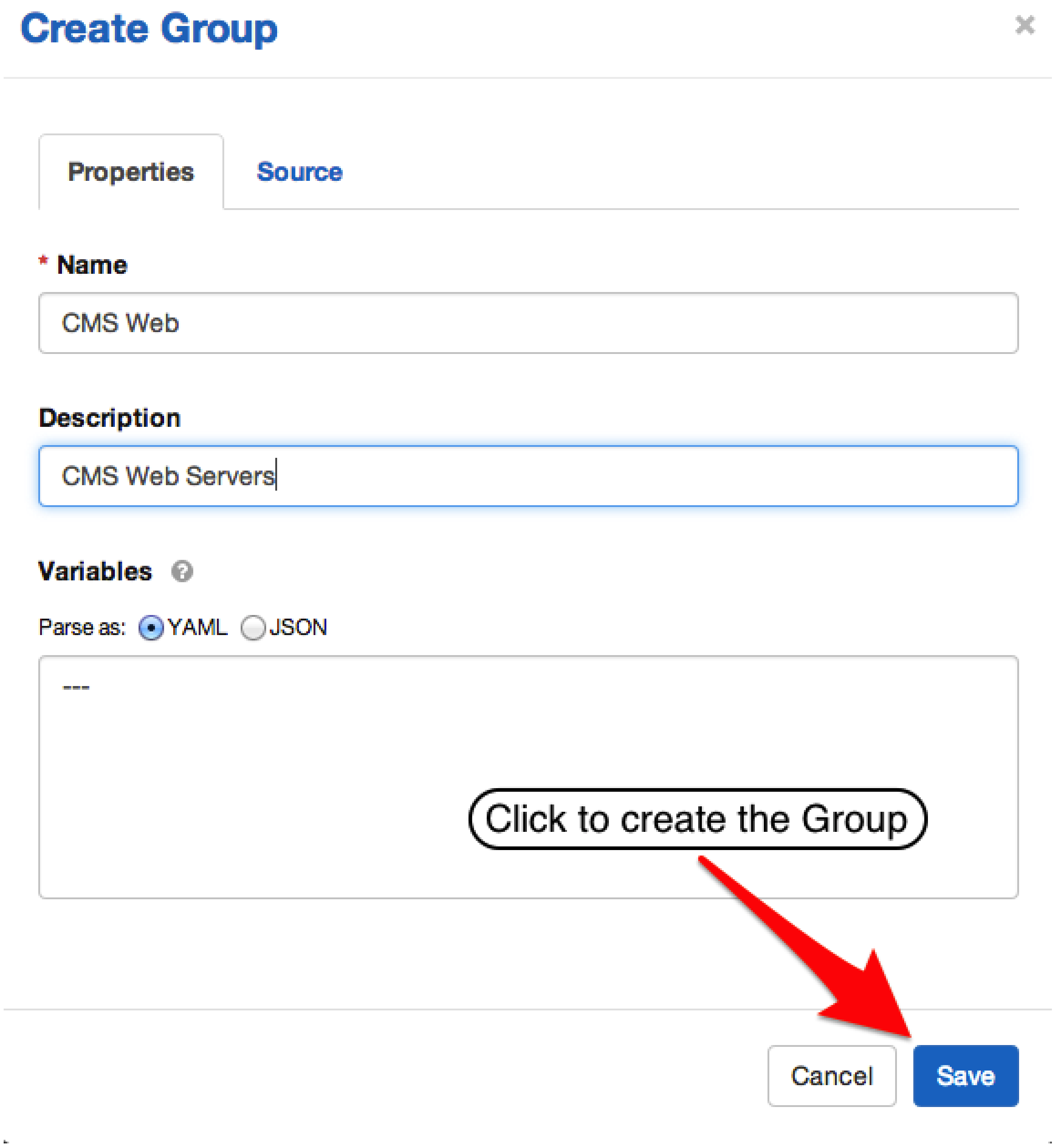
The left side of the screen displays the groups that belong to the Web Servers inventory. The groups list is empty at this point. The right side displays hosts.

Hosts are added to groups. They cannot be added directly to the inventory root. So to begin adding hosts to the Web Servers inventory, we first need to add a group. Click the ➕ button.



No records matched your search.

Bender Products has a group of web server hosts supporting the corporate CMS application. To add these hosts to the Web Servers inventory we'll create a "CMS Web" group. Again, we will defer a discussion of variables for later. Click the **Save** button to create the group.



Name: CMS Web

Description: CMS Web Servers

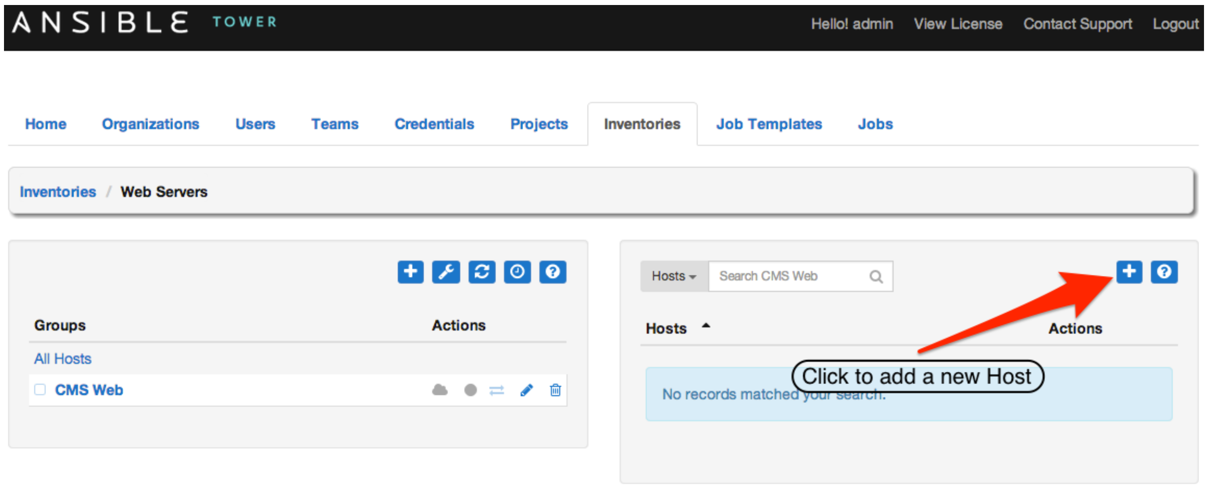
Variables

Parse as: [✓ YAML] JSON

```
---
```

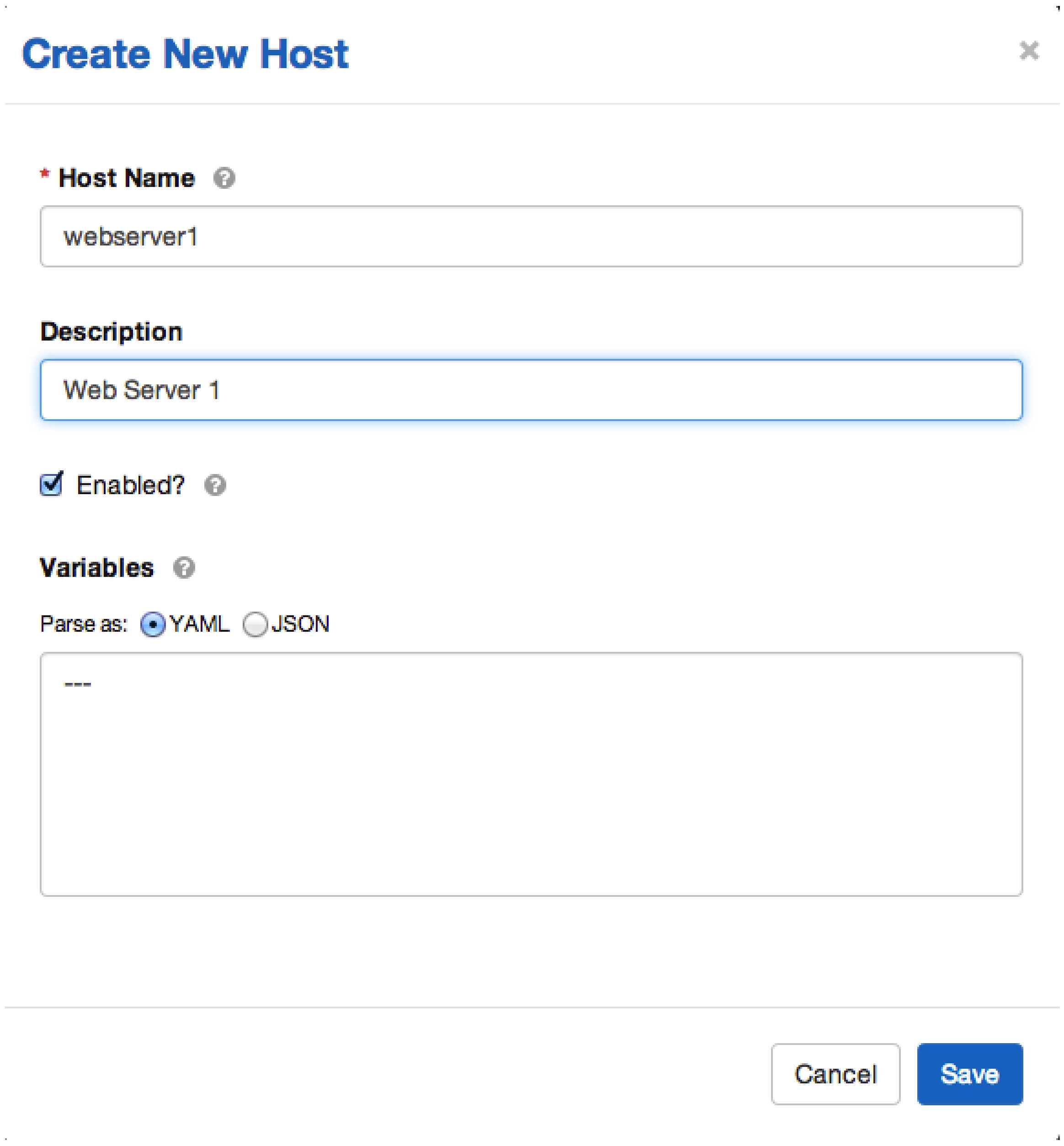
Finally, we'll add a host to the group.

Select ➕ to create the new host and add it to the group.



Enter the Host Name, which should either be the DNS resolvable name of the host or its IP address. This is how Tower will contact the host, so the host must be reachable using this hostname or IP address for Tower to function properly. The **Description** is arbitrary, as usual. (*Note, experienced Ansible users will know they could also set the* `ansible_ssh_host` *environment variable to use an alias, but that is not going to be covered here*).

For the purposes of this Quick Start, add a host that you can actually reach via SSH and manage using Ansible (i.e. that meets the Ansible requirements (http://docs.ansible.com/intro_installation.html)). We will launch a simple Ansible playbook that will not harm or modify the target in any way. Using a real target host allows us to ensure that Tower is setup properly.



Create New Host

* Host Name

webserver1

Description

Web Server 1

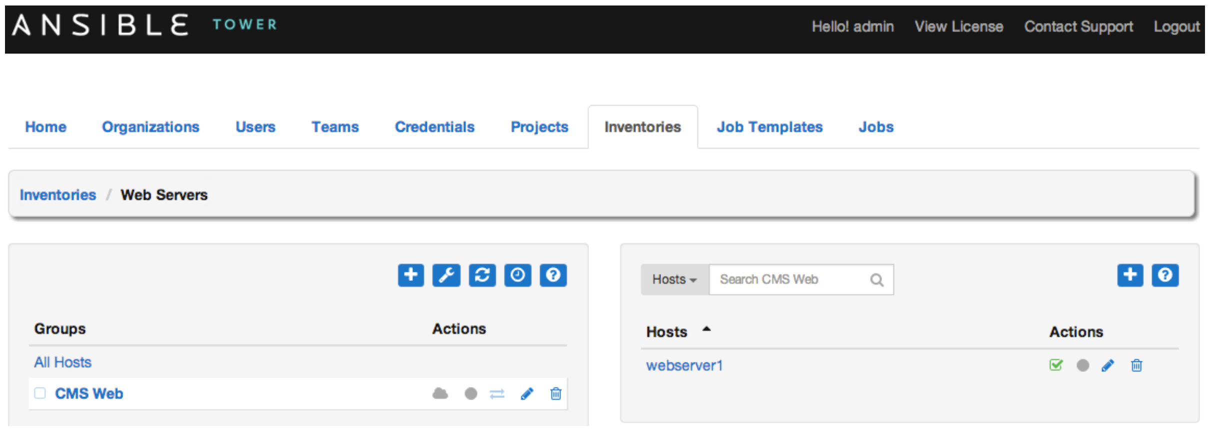
[✓] Enabled?

Variables

Parse as: (●) YAML ( ) JSON

```
---
```

Click **Save** to finish adding the host.

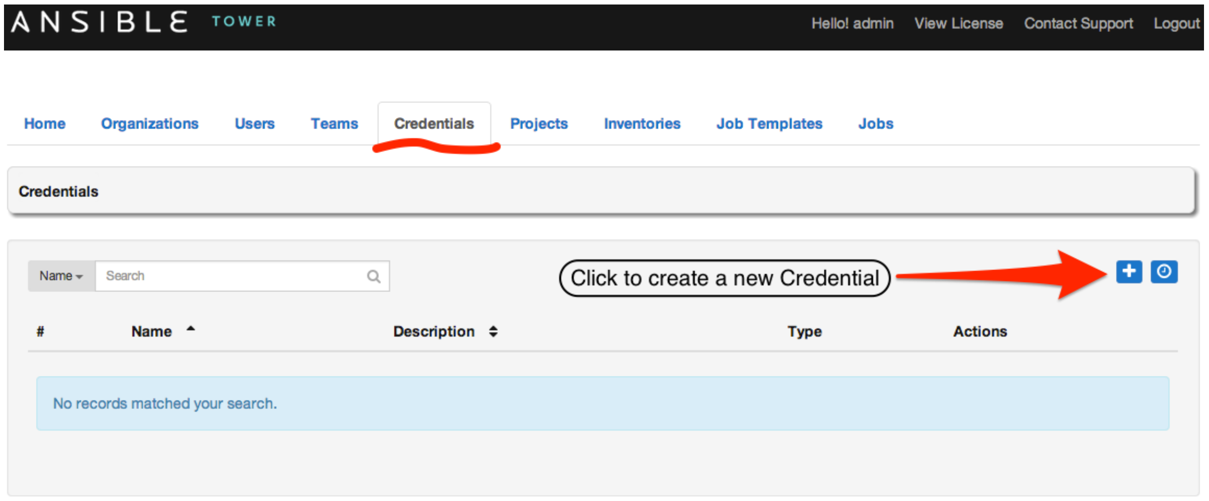


Next, we'll add credentials to our new user that Tower can use to access and launch Ansible playbooks for the host in our inventory.

5. Create a new Credential

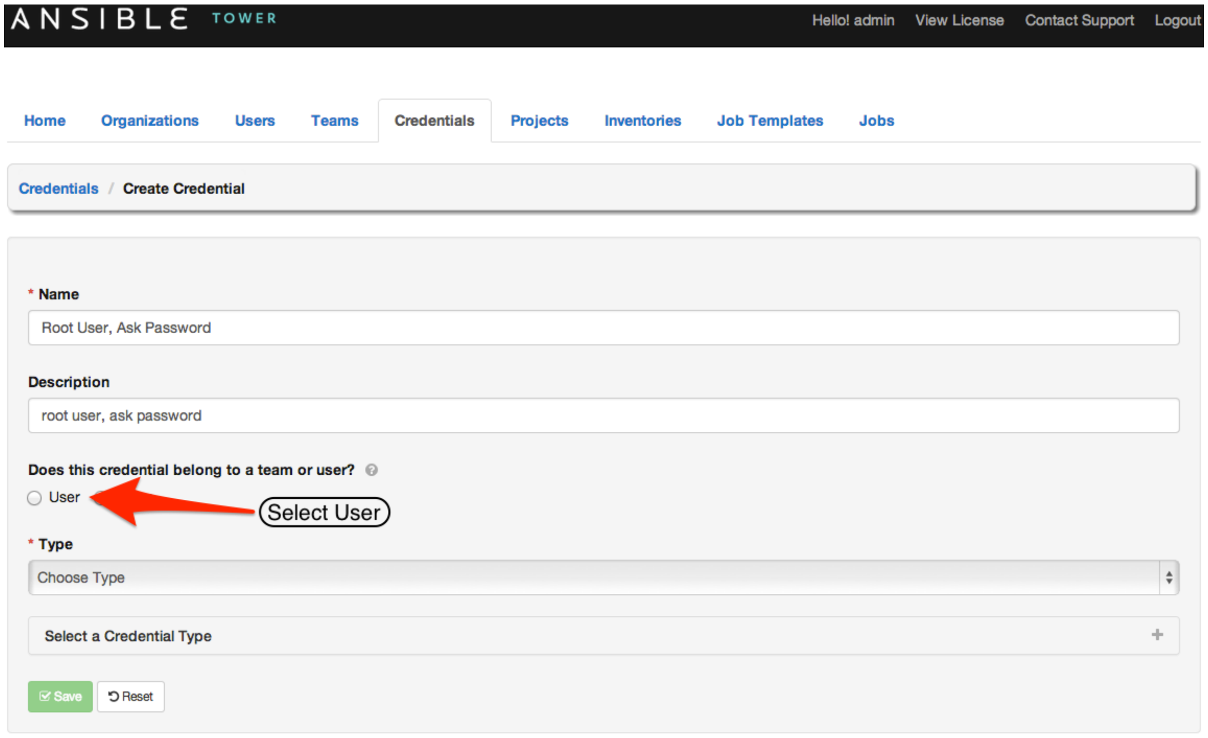
Credentials are used to authenticate the Tower user to launch Ansible playbooks against inventory hosts and can include passwords and SSH keys. You can also require the Tower user to enter a password or key phrase when a playbook is launched using the credentials feature of Tower.

Create a new credential by browsing to the **Credentials** tab. Click ➕ to create a new credential.



No records matched your search.

Enter an arbitrary **Name** and **Description** for this credential. Either an individual user or a team may own credentials. Let's associate this credential with the user we created in step #3. Select the "User" radio button.



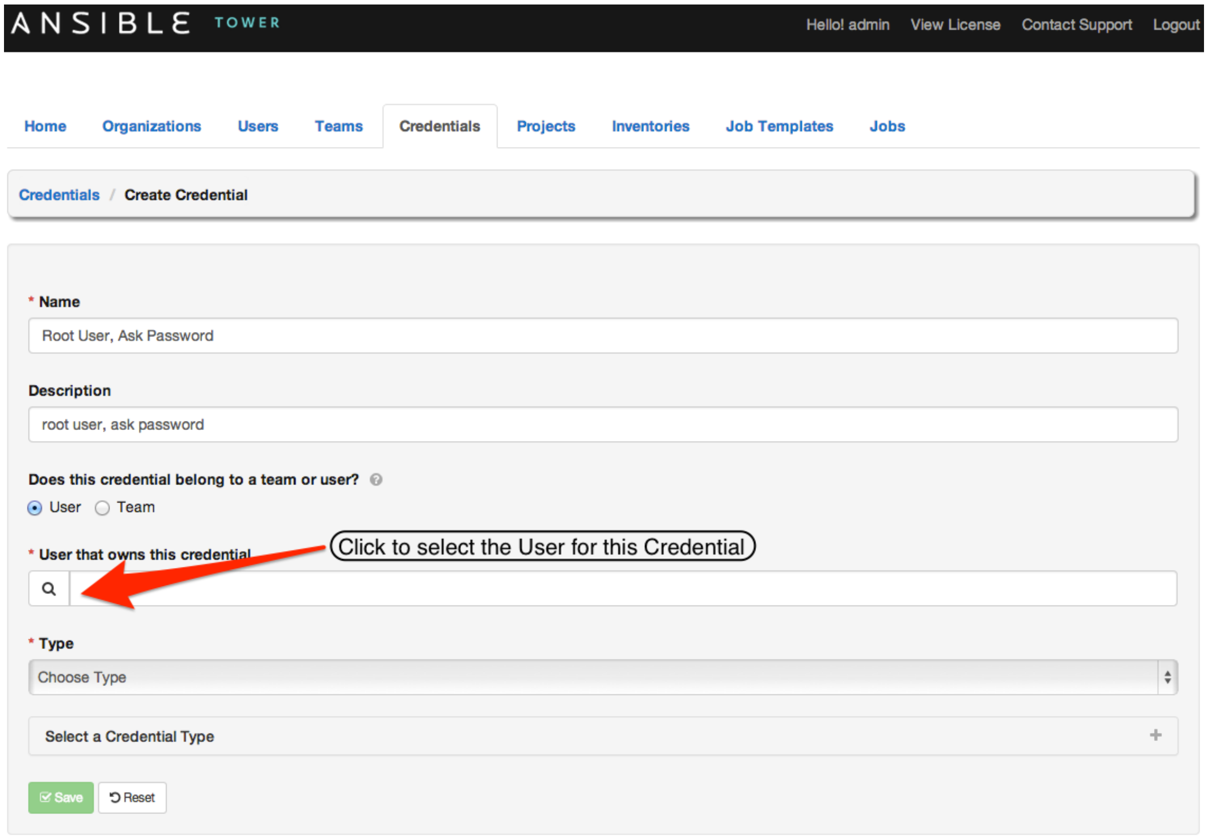
* Name: Root User, Ask Password

Description: root user, ask password

Does this credential belong to a team or user?
User

* Type: Choose Type

Then, select the look-up button to find the user that we created in step #3.



* Name: Root User, Ask Password

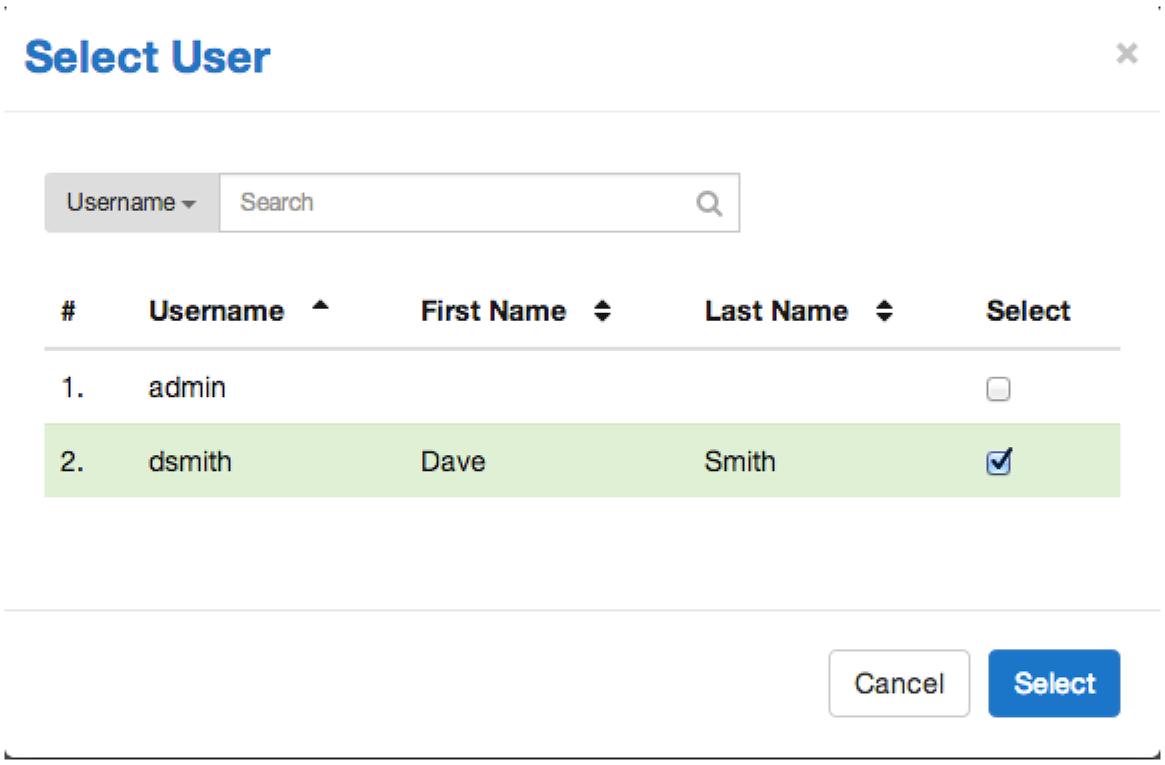
Description: root user, ask password

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

* User that owns this credential

* Type: Choose Type

Find and select the "dsmith" user.



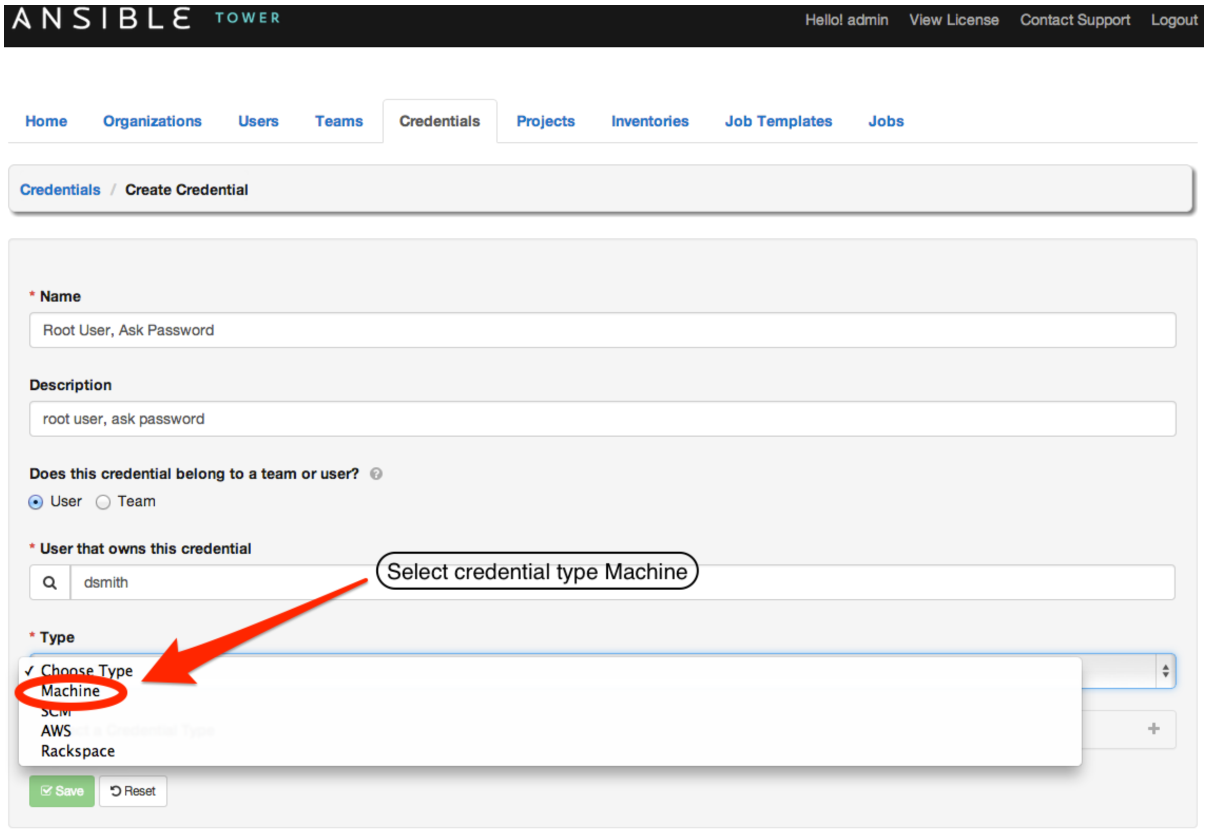
Select User

Username ▾ Search

#	Username	First Name	Last Name	Select
1.	admin			☐
2.	dsmith	Dave	Smith	☑

Cancel Select

Next, select credential type **Machine**.



* Name: Root User, Ask Password

Description: root user, ask password

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

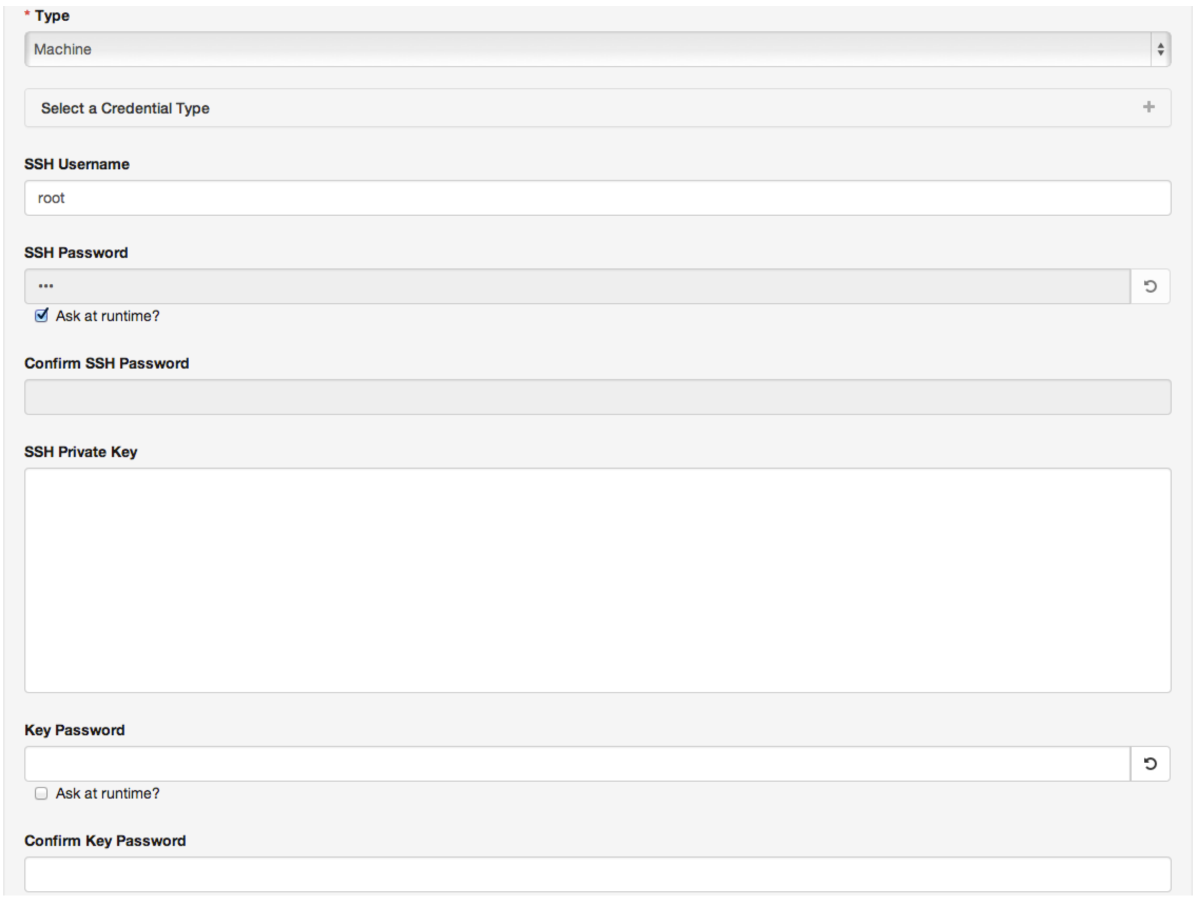
* User that owns this credential: dsmith

* Type: ✓ Choose Type, [✓ Machine], SCM, AWS, Rackspace

Now, we'll enter the details of the appropriate authentication mechanism to use for the host we added to Tower in step #3. Use the actual credentials for the real host. To keep things simple, we'll use an SSH password, but ask for it at runtime. So, rather than enter the password here, we'll enter it later when we launch a playbook using these credentials. To do so, check the box **Ask at runtime for SSH Password**, as shown here.

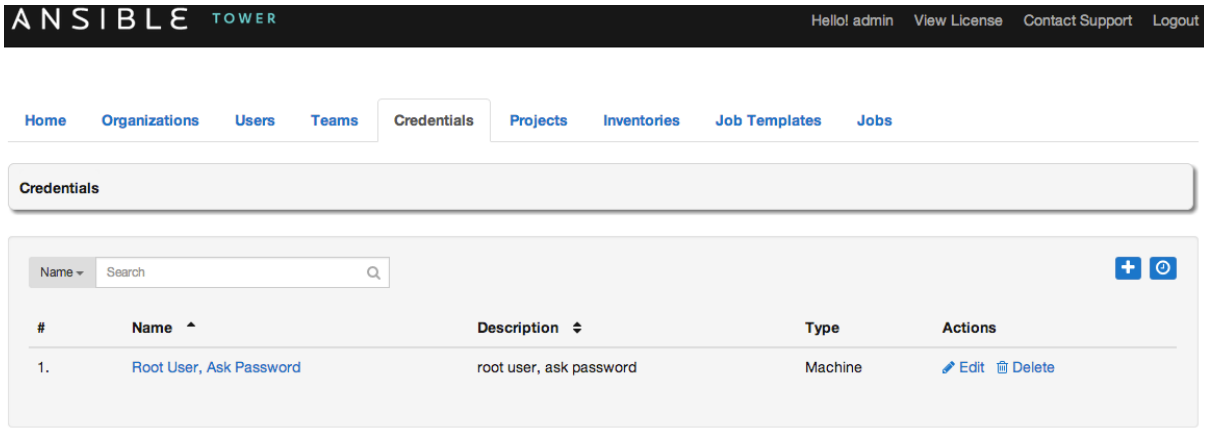
NOTE: *Tower supports various different options for what you want to store for credentials in this box. Uploading a locked SSH key is recommended, and Tower can prompt you for the SSH unlock password for use with ssh-agent when launching the job.*

Tower encrypts passwords and key information in the Tower database and never makes secret information visible via the API.



* Type	Machine
Select a Credential Type	
SSH Username	root
SSH Password	•••
Ask at runtime?	✓
Confirm SSH Password	
SSH Private Key	
Key Password	
Ask at runtime?	☐
Confirm Key Password	

Click **Save**.



#	Name	Description	Type	Actions
1.	Root User, Ask Password	root user, ask password	Machine	Edit Delete

Now, we'll create a new project and a job template with which to launch a simple playbook.

6. Create a new Project

Before we create this project, we'll need to create a subdirectory for it on the Tower server filesystem, where we will store the Ansible playbooks for this project.

NOTE: *This will require you to log into the Tower server on the command line console. In a future version of Tower, this will be done without leaving the Web interface.*

Create a new project directory by creating a directory on the Tower filesystem underneath the **Project Base Path**, by default "/var/lib/awx/projects".

```
root@localhost:~$ cd /var/lib/awx/projects
root@localhost:~$ mkdir helloworld
```

While we're here, let's go ahead and create a simple Ansible playbook. Use your favorite editor to create a file called "helloworld.yml" inside the directory we just created, "/var/lib/awx/projects".

```
root@localhost:~$ cd helloworld
root@localhost:~$ vi helloworld.yml
```

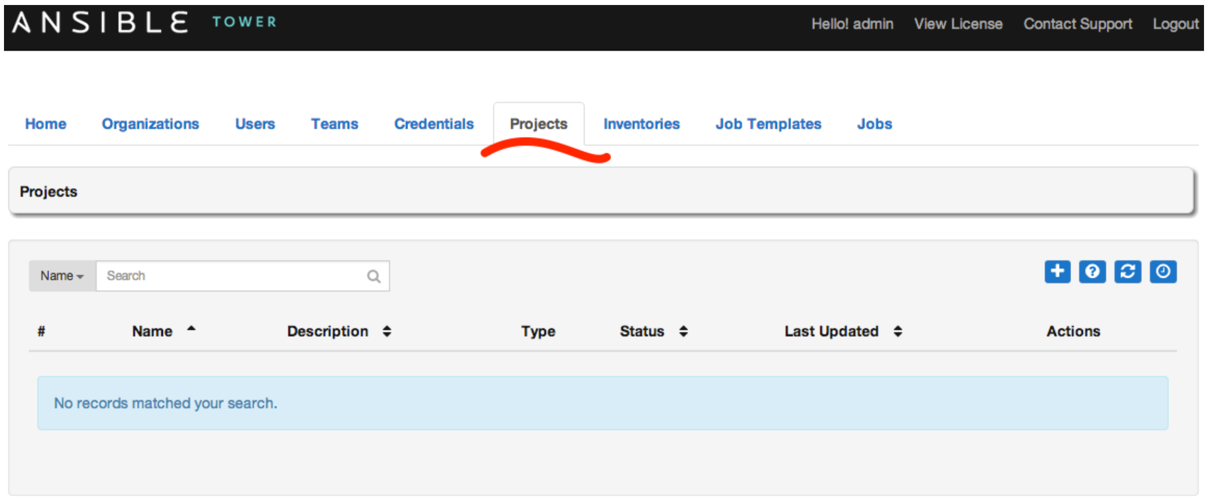
The contents of the file are below:

```
---
- name: Hello World!
  hosts: all
  user: root

  tasks:

  - name: Hello World!
    shell: echo "Hi! Tower is working"
```

Save this playbook file and we'll use it to test Tower running a playbook against the host in our inventory.

***NOTE:** Ansible playbooks utilize the YAML language. More information about Ansible playbooks may be found at: http://docs.ansible.com/playbooks.html. More information on YAML can be found at: http://yaml.org/.*

Now, create the new project by browsing to the **Projects** tab. Click the ➕ button.



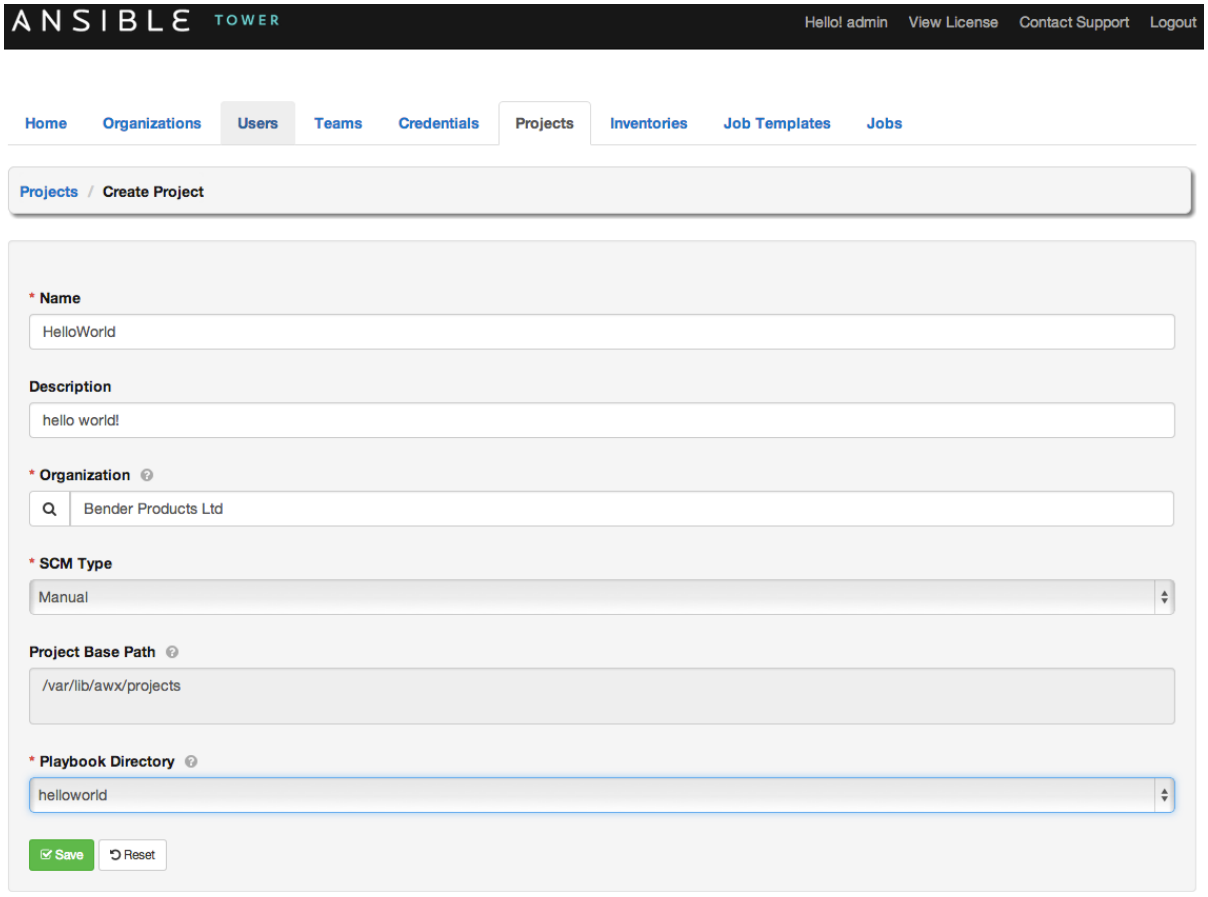
No records matched your search.

Enter a **Name** and **Description** for the project.

The **Project Base Path** will display the value entered when Tower was installed and cannot be edited from this dialog. (See the section Administration of Tower for more information on how to modify this value.)

Leave **SCM Type** set to Manual, for now.

For the **Playbook Directory** select a value that corresponds to the subdirectory we just created.



* Name: HelloWorld

Description: hello world!

* Organization: Bender Products Ltd

* SCM Type: Manual

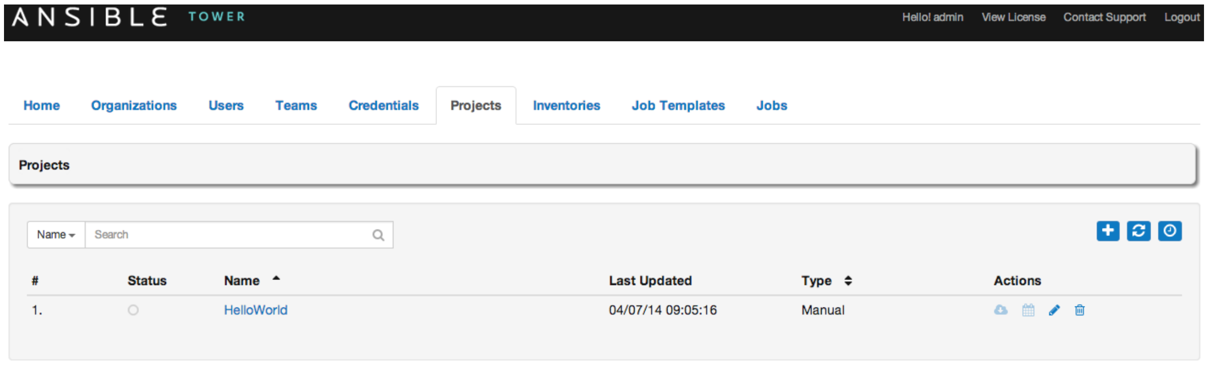
Project Base Path: /var/lib/awx/projects

* Playbook Directory: helloworld

***NOTE:** If you see the following warning, double check that the helloworld project directory and file were created correctly and that the permissions are correct. Use* `chown -R awx` *on the project directory if necessary. If SE Linux is enabled, check the directory and file context.*

*"**WARNING**: There are no unassigned playbook directories in the base project path /var/lib/awx/projects. Either the projects directory is empty, or all of the contents are already assigned to other projects. New projects can be checked out from source control by changing the SCM type option rather than specifying checkout paths manually. To continue with manual setup, log into the Tower server and ensure content is present in a subdirectory under /var/lib/awx/projects. Run "chown -R awx" on the content directory to ensure awx can read the playbooks."*

Select **Save** and the new project will be displayed.

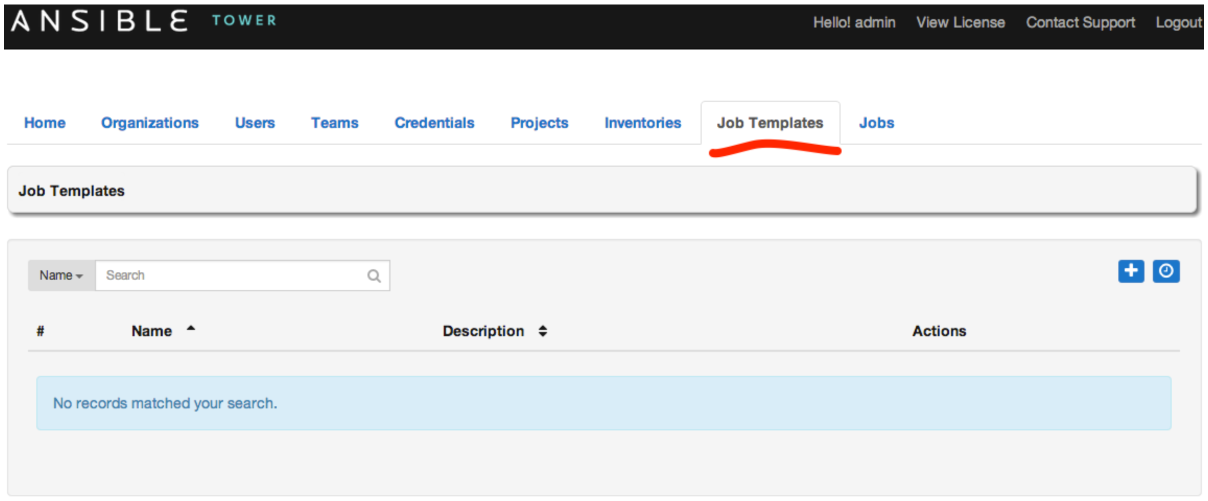


#	Status	Name	Last Updated	Type	Actions
1.	○	HelloWorld	04/07/14 09:05:16	Manual	

Finally, let's create a job template for this new playbook and launch it.

7. Create a new Job Template using an Ansible example playbook

A job template combines an Ansible playbook from a project and the settings required to launch it. Create a new job template by browsing to the **Job Templates** tab and clicking the button.

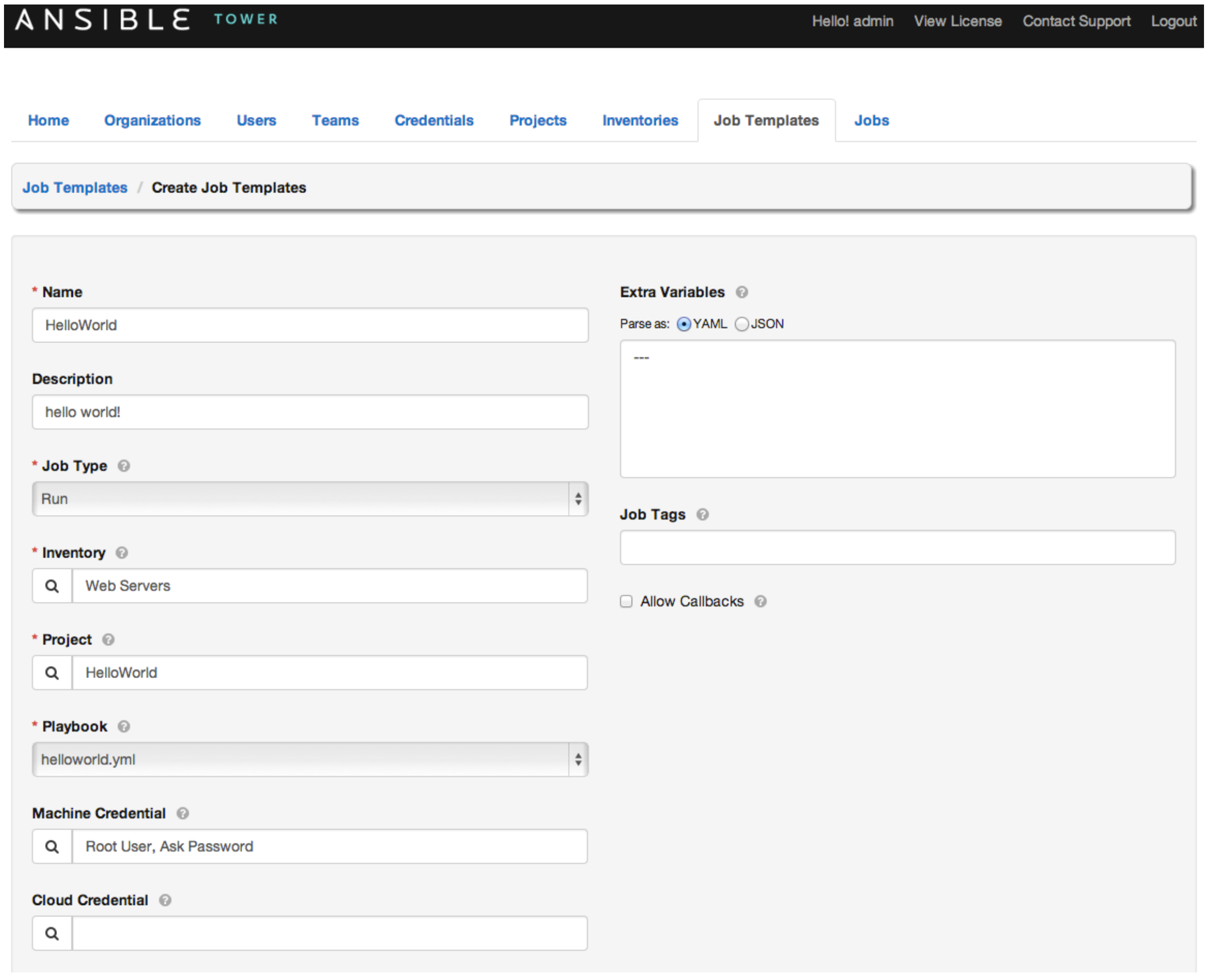


#	Name	Description	Actions

No records matched your search.

Enter values for the **Name** and **Description**. Jobs can be of type **Run** or **Check**. Select **Run** for this Quick Start (check corresponds to "dry run" mode.) Choose the **Inventory**, **Project**, and **Credential** from those we have created during this exercise.

The playbook drop - down menu will automatically populate from the project path and playbook we created in step #5. Choose the "helloworld" playbook.



* Name: HelloWorld

Description: hello world!

* Job Type: Run

* Inventory: Web Servers

* Project: HelloWorld

* Playbook: helloworld.yml

Machine Credential: Root User, Ask Password

Cloud Credential:

Extra Variables

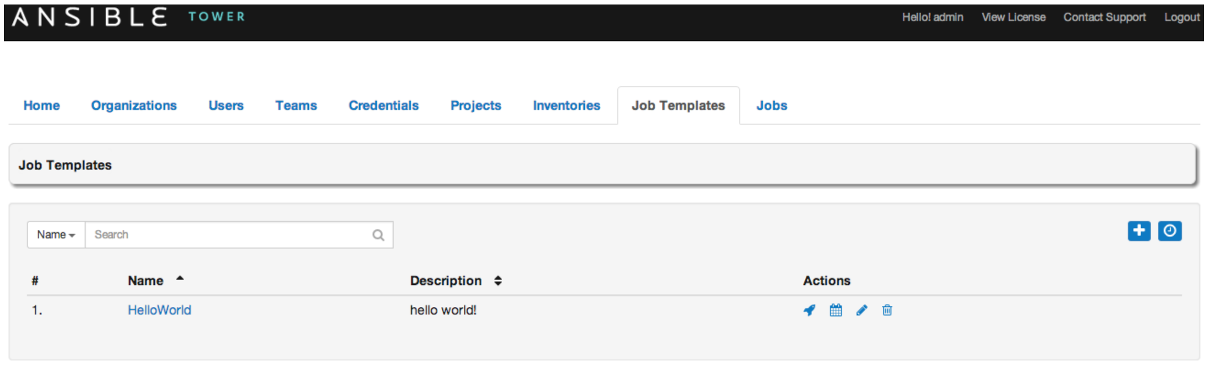
Parse as: [✓ YAML] JSON

```
---
```

Job Tags:

Allow Callbacks

Click **Save**.

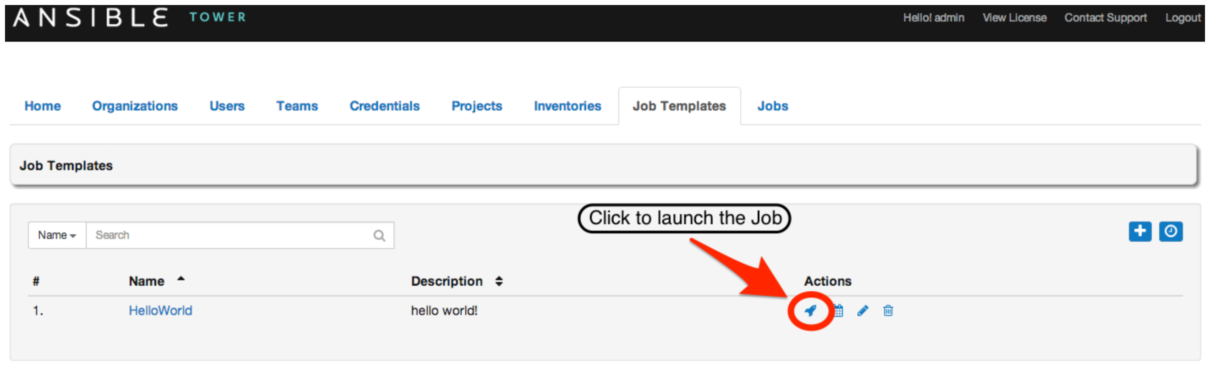


#	Name	Description	Actions
1.	HelloWorld	hello world!	

Now, let's launch the playbook and watch it all come together.

9. Launch it!

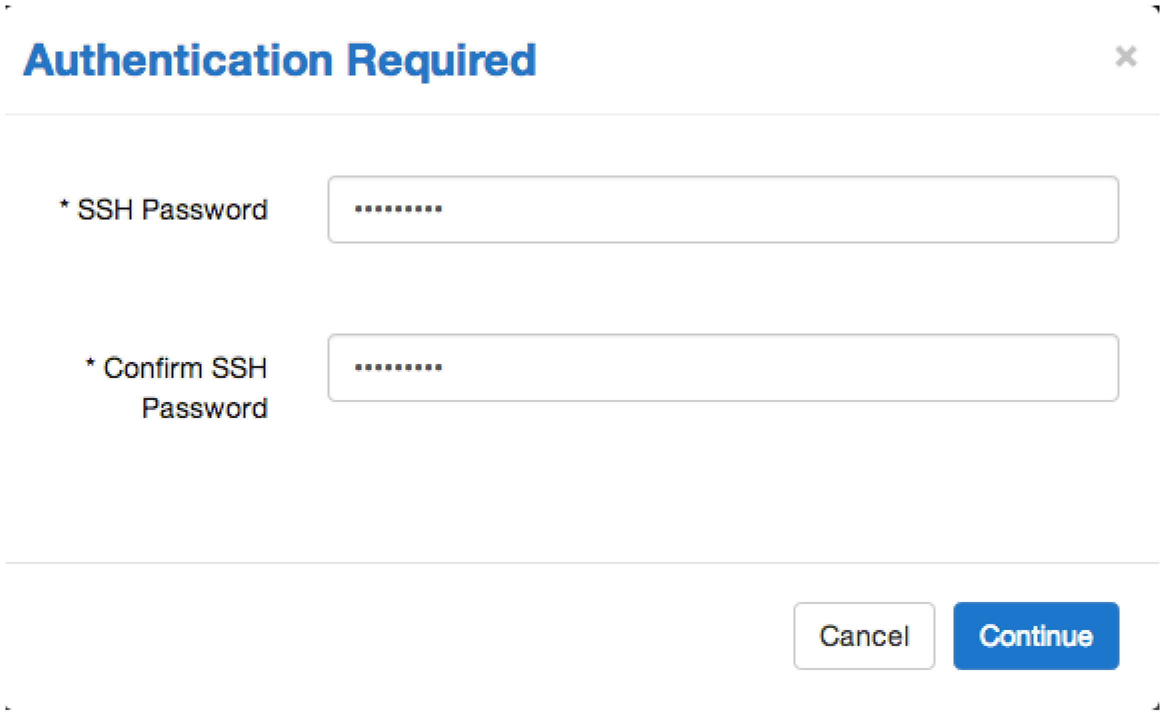
To launch the playbook, browse to the **Job Templates** tab and click **Launch** on the template.



Job Templates

#	Name	Description	Actions
1.	HelloWorld	hello world!	

Tower will ask you for the SSH password, as we configured the credential.

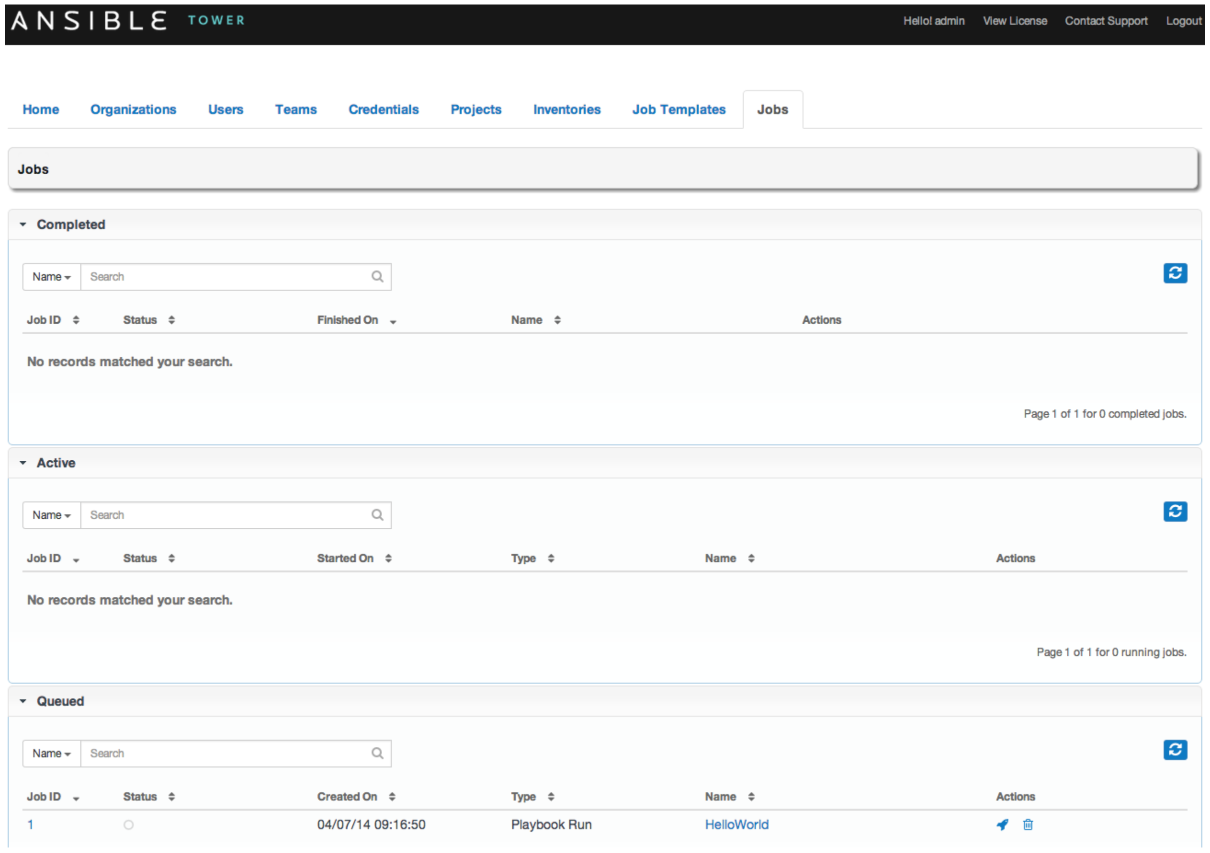


Authentication Required

* SSH Password: ••••••••••

* Confirm SSH Password: ••••••••••

Tower will then redirect the browser to the **Jobs** tab, where you can see the list of all jobs, separated into **Completed**, **Active**, **Queued**, and **Scheduled**.



Jobs

Completed

Job ID	Status	Finished On	Name	Actions

No records matched your search.

Page 1 of 1 for 0 completed jobs.

Active

Job ID	Status	Started On	Type	Name	Actions

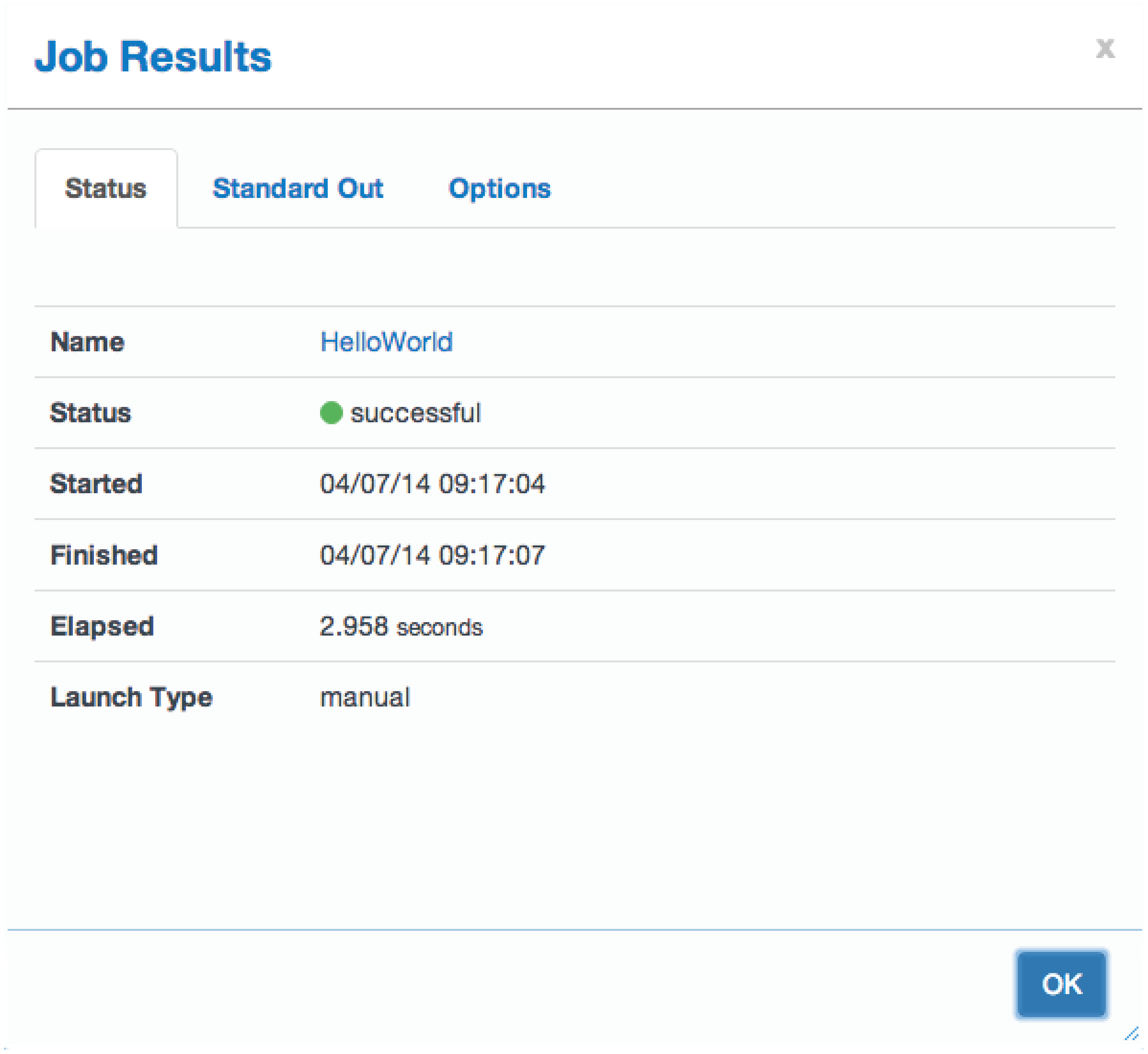
No records matched your search.

Page 1 of 1 for 0 running jobs.

Queued

Job ID	Status	Created On	Type	Name	Actions
1	○	04/07/14 09:16:50	Playbook Run	HelloWorld	

Refresh this page until the job moves from **Queued** to **Completed**. Then, select the **Job ID** or the **Name** to inspect the details of the job.



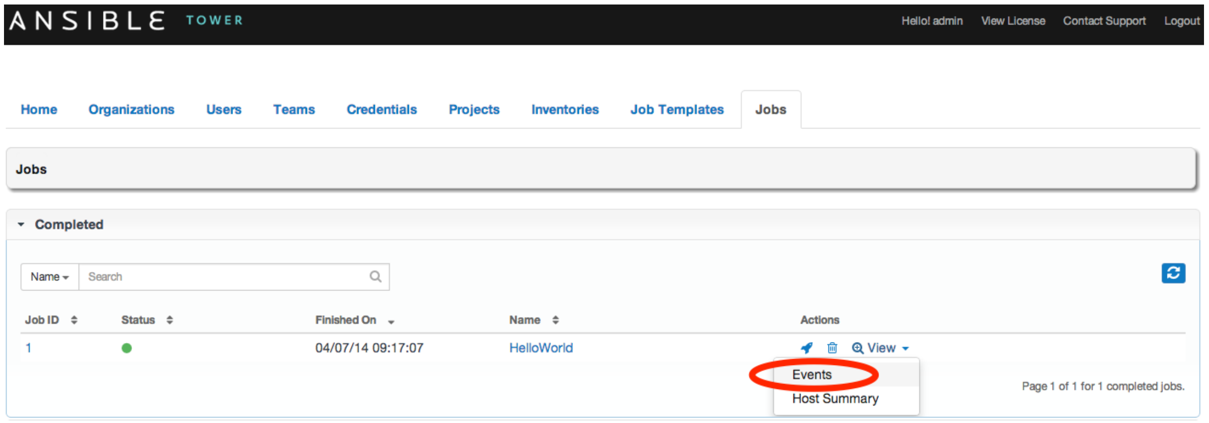
Job Results

Status | Standard Out | Options

Name	HelloWorld
Status	● successful
Started	04/07/14 09:17:04
Finished	04/07/14 09:17:07
Elapsed	2.958 seconds
Launch Type	manual

OK

Back at the **Jobs** tab, select **View** and then **Events**:



ANSIBLE TOWER — Hello! admin | View License | Contact Support | Logout

Home | Organizations | Users | Teams | Credentials | Projects | Inventories | Job Templates | Jobs

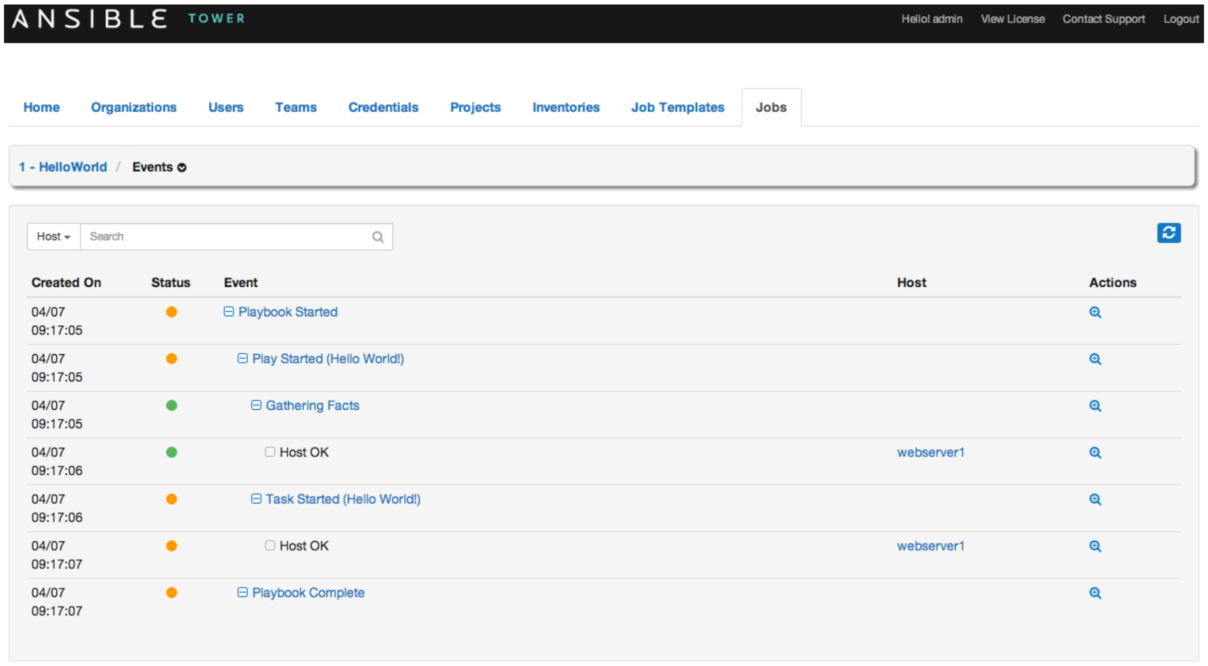
Jobs

Completed

Job ID	Status	Finished On	Name	Actions
1	●	04/07/14 09:17:07	HelloWorld	View ▾ (Events, Host Summary)

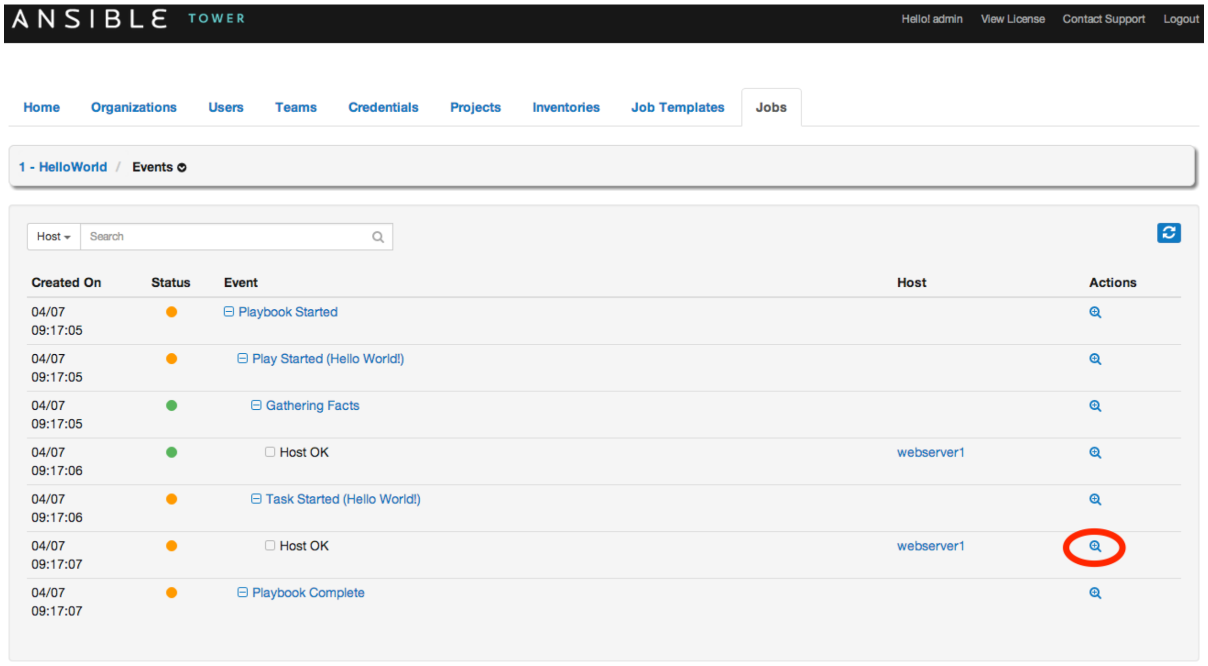
Page 1 of 1 for 1 completed jobs.

This screen will show us all of the events that resulted from running our playbook.


ANSIBLE TOWER
Hello! admin View License Contact Support Logout
Home Organizations Users Teams Credentials Projects Inventories Job Templates Jobs
1 - HelloWorld / Events
Host Search

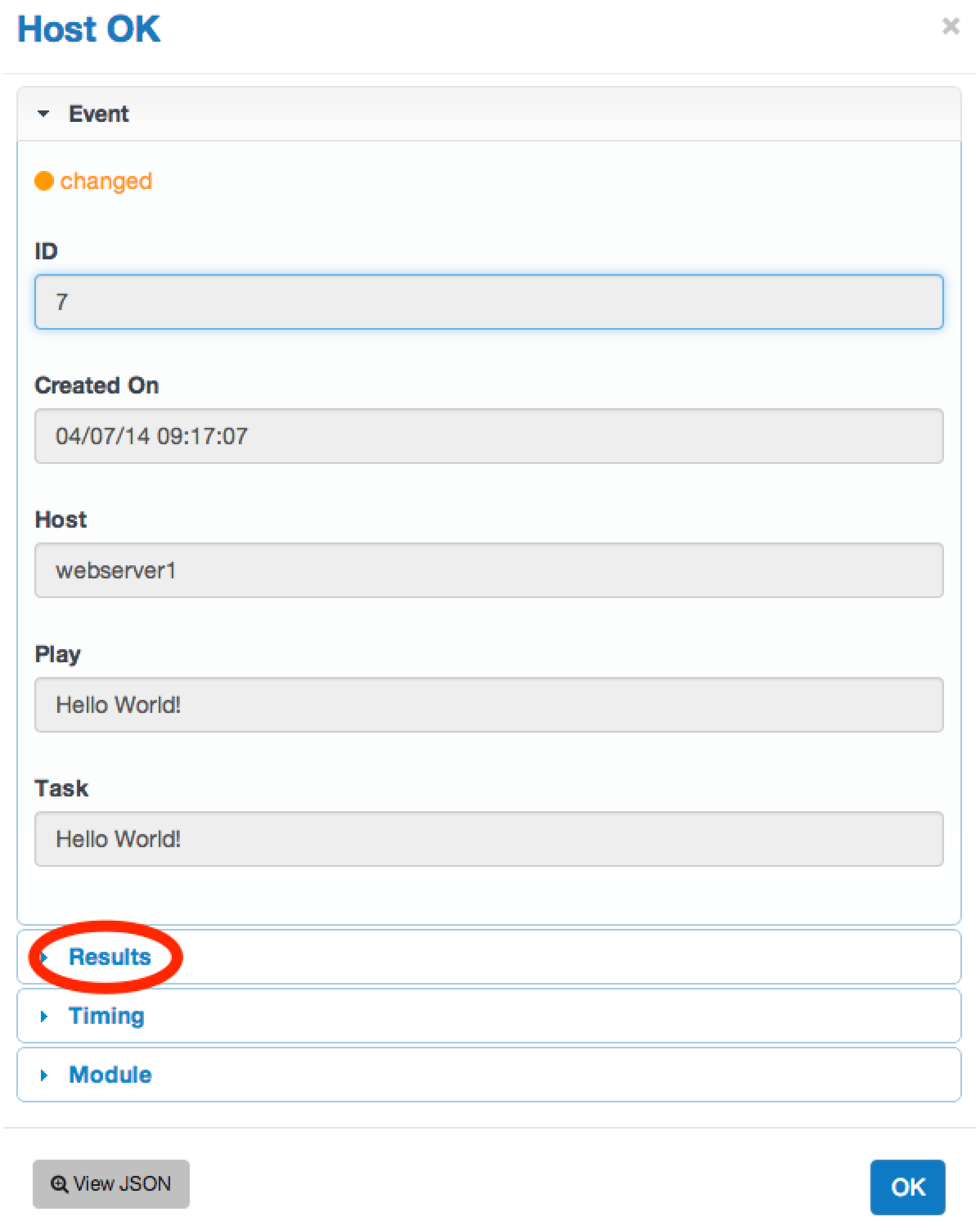
Created On	Status	Event	Host	Actions
04/07 09:17:05	●	⊟ Playbook Started		🔍
04/07 09:17:05	●	⊟ Play Started (Hello World!)		🔍
04/07 09:17:05	●	⊟ Gathering Facts		🔍
04/07 09:17:06	●	☐ Host OK	webserver1	🔍
04/07 09:17:06	●	⊟ Task Started (Hello World!)		🔍
04/07 09:17:07	●	☐ Host OK	webserver1	🔍
04/07 09:17:07	●	⊟ Playbook Complete		🔍

To show the event details, click **Actions** as shown below:


ANSIBLE TOWER
Hello! admin View License Contact Support Logout
Home Organizations Users Teams Credentials Projects Inventories Job Templates Jobs
1 - HelloWorld / Events
Host Search

Created On	Status	Event	Host	Actions
04/07 09:17:05	●	⊟ Playbook Started		🔍
04/07 09:17:05	●	⊟ Play Started (Hello World!)		🔍
04/07 09:17:05	●	⊟ Gathering Facts		🔍
04/07 09:17:06	●	☐ Host OK	webserver1	🔍
04/07 09:17:06	●	⊟ Task Started (Hello World!)		🔍
04/07 09:17:07	●	☐ Host OK	webserver1	🔍
04/07 09:17:07	●	⊟ Playbook Complete		🔍

Now, click **Results**:



Host OK

Event

changed

ID

7

Created On

04/07/14 09:17:07

Host

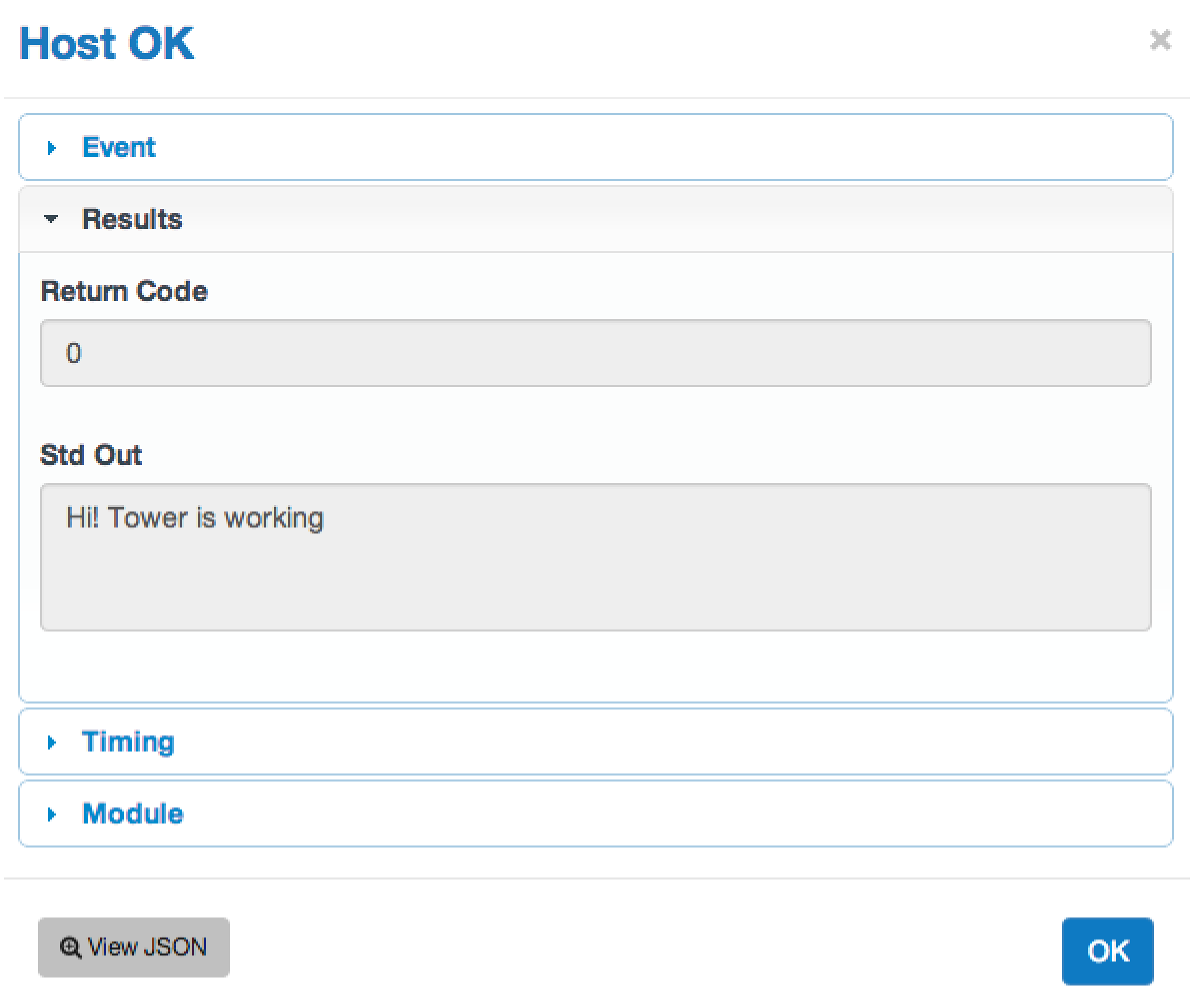
webserver1

Play

Hello World!

Task

Hello World!


Host OK

Event

Results

Return Code

0

Std Out

Hi! Tower is working

Timing

Module

Great work! Your Tower installation is up and running properly. Now, you can browse through the User Guide and learn about all of these features of Tower in more detail.

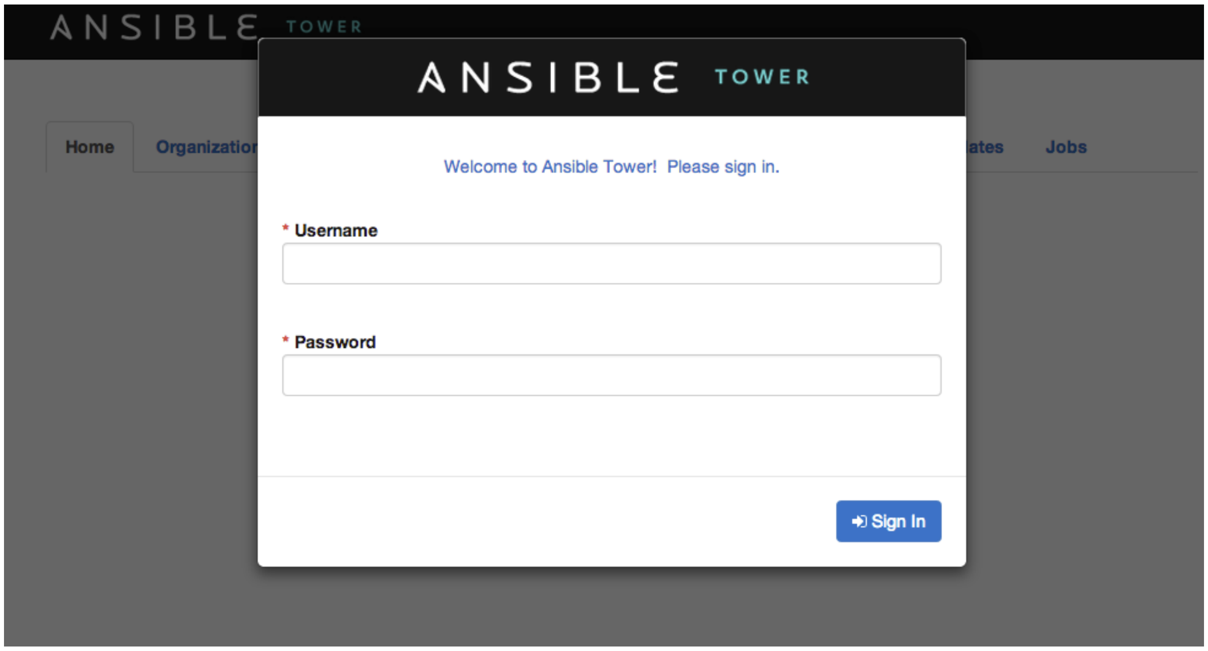
Don't hesitate to send your feedback to support@ansible.com. We appreciate your support!

User Guide

This section of the documentation will detail all of the functionality of Tower.

Logging In

To log in to Tower, browse to the Tower interface at `http://<Tower server hostname or IP Address>/`

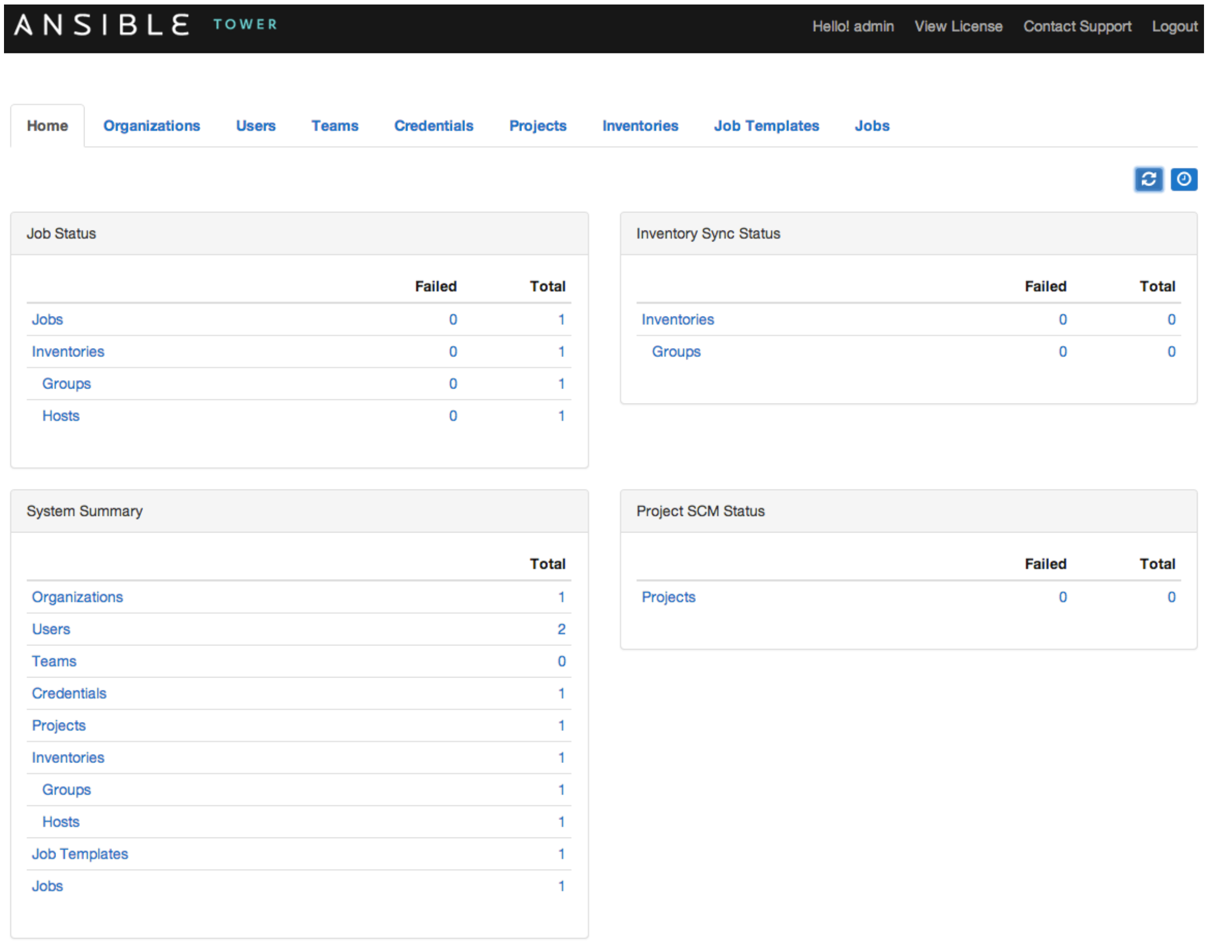


Log in using a valid Tower username and password.

***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 from the Users tab.*

Home

The central interface to Tower is the **Home** dashboard. This screen displays the status of Tower jobs, synchronization with inventory sources and source code management systems, and a summary of configured Tower objects.



Job Status

	Failed	Total
Jobs	0	1
Inventories	0	1
Groups	0	1
Hosts	0	1

Inventory Sync Status

	Failed	Total
Inventories	0	0
Groups	0	0

System Summary

	Total
Organizations	1
Users	2
Teams	0
Credentials	1
Projects	1
Inventories	1
Groups	1
Hosts	1
Job Templates	1
Jobs	1

Project SCM Status

	Failed	Total
Projects	0	0

All of the list items displayed on the Home dashboard are linked to their respective Tower objects for convenient access.

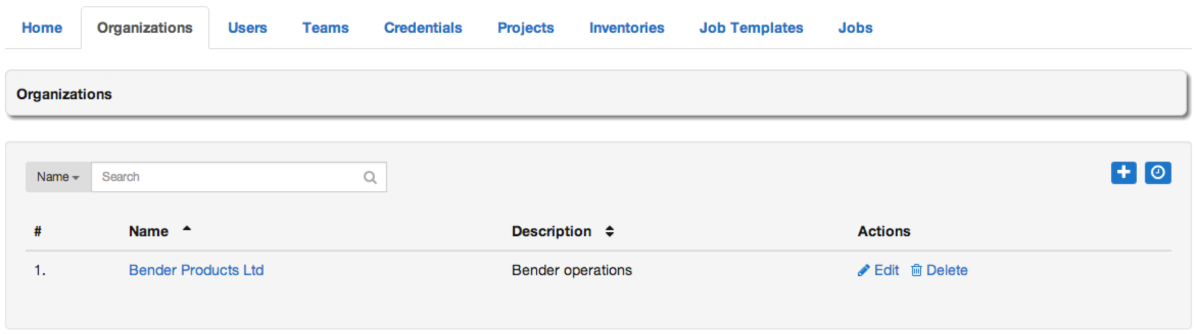
Buttons located in the upper right corner of the **Home** tab provide the following actions:

- Refresh the page
- View Activity Stream

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** tab 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.



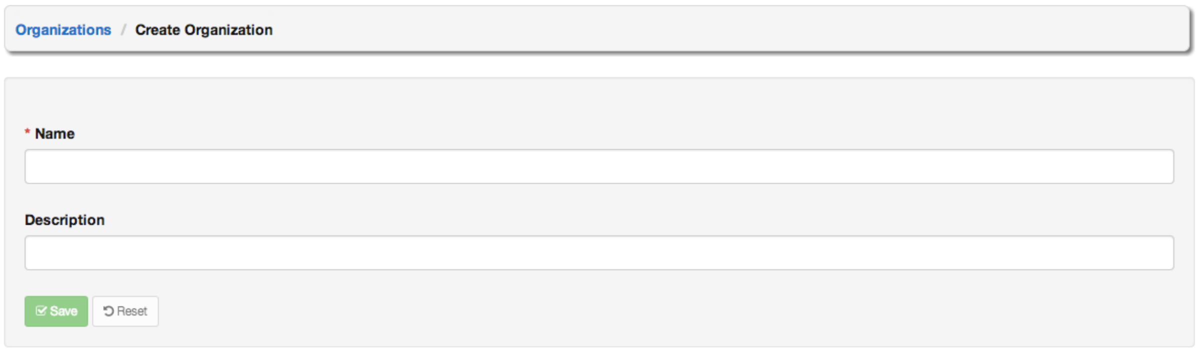
#	Name	Description	Actions
1.	Bender Products Ltd	Bender operations	Edit Delete

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

- Create a new organization
- View Activity Stream

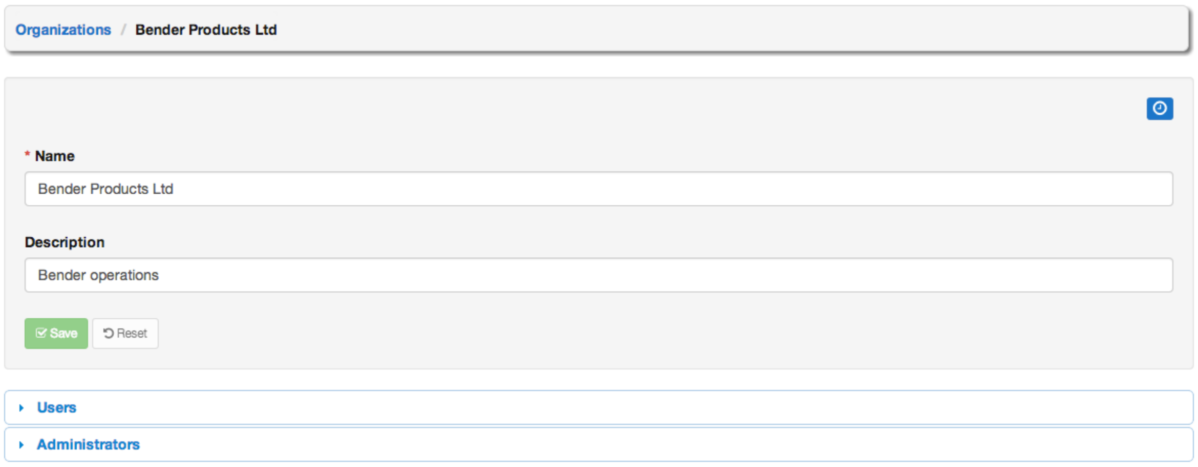
Create a new organization by selecting the ➕ button.

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 will display 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.


Organizations / Bender Products Ltd

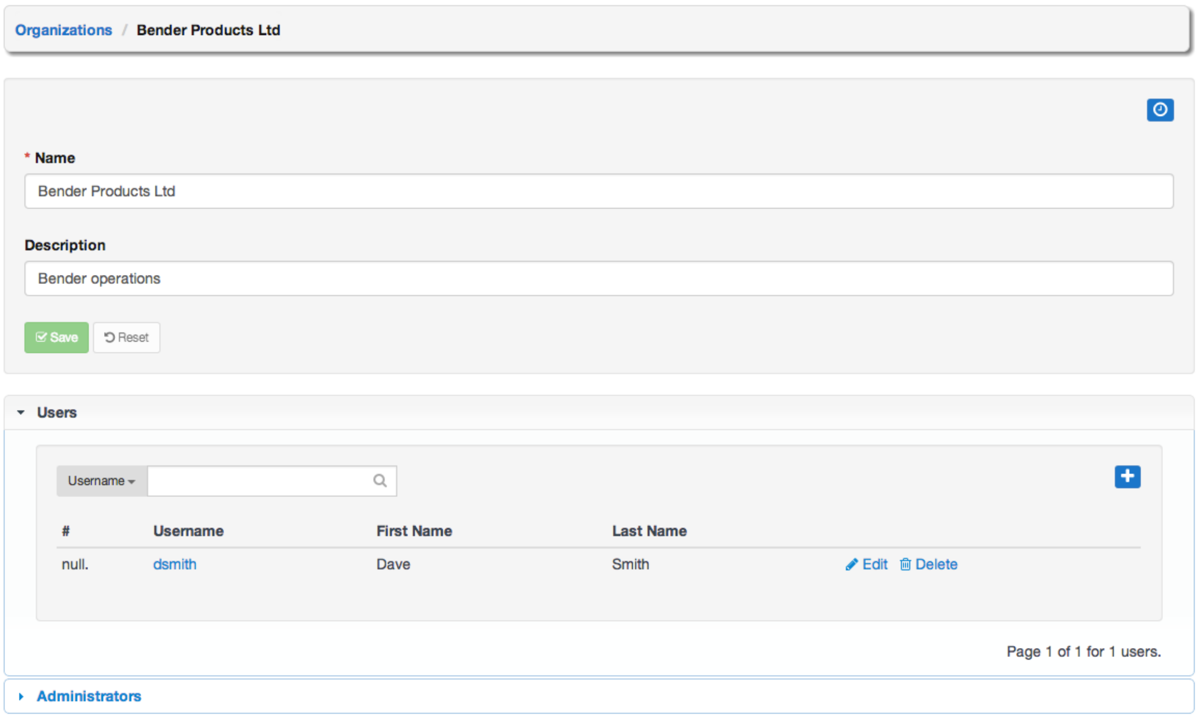
* Name
Bender Products Ltd

Description
Bender operations

Users

A user is someone with access to Tower with associated permissions and credentials. For more information, please see the section Users.

Expand the users menu by selecting **Users**.


Organizations / Bender Products Ltd

* Name
Bender Products Ltd

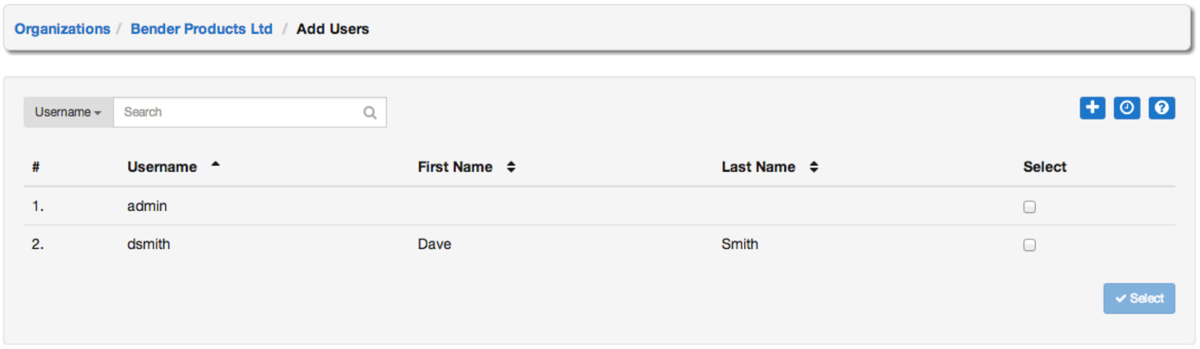
Description
Bender operations

#	Username	First Name	Last Name	
null.	dsmith	Dave	Smith	Edit Delete

Page 1 of 1 for 1 users.

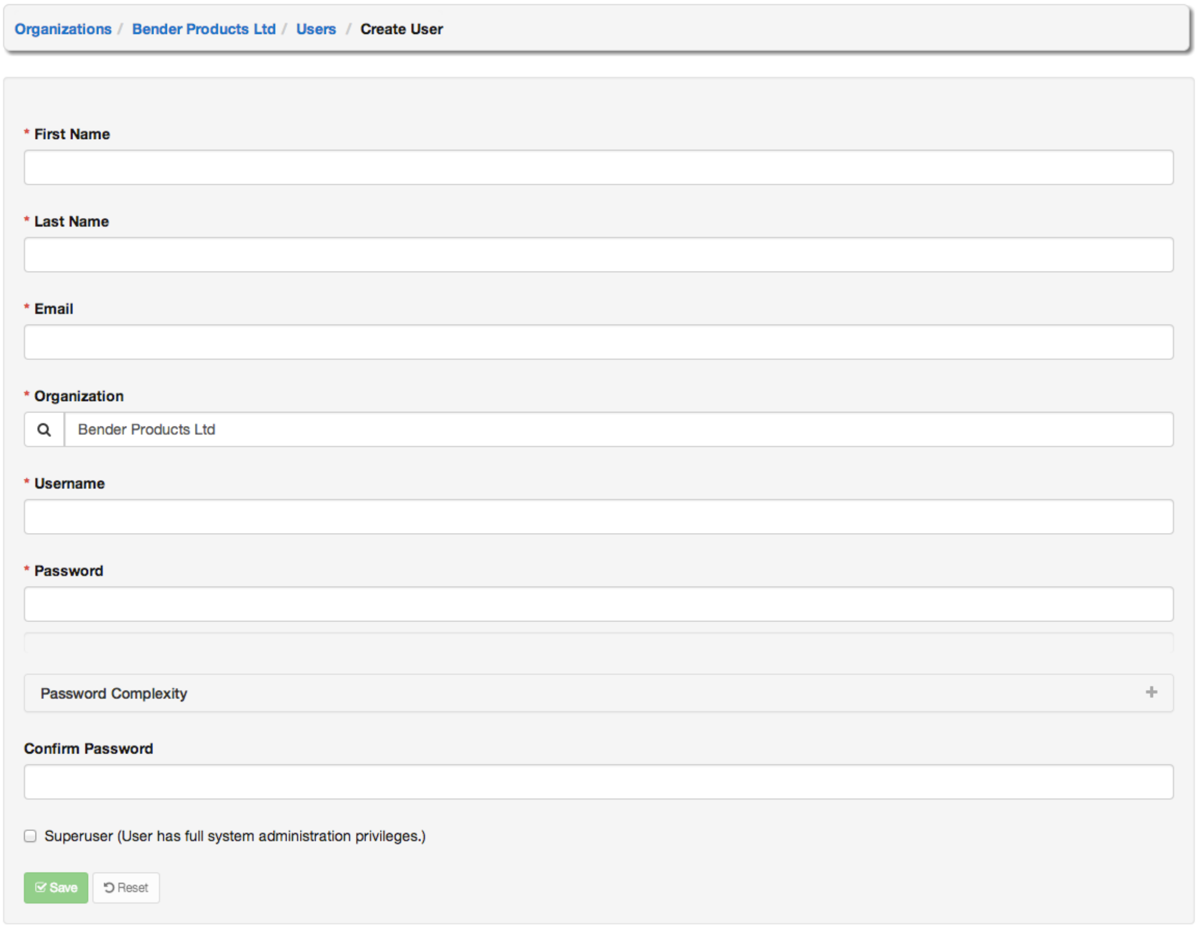
This menu allows you to manage the user membership for this organization. (User membership may also be managed on a per-user basis via the **Users** tab.) The user list may be sorted and searched by **Username**, **First Name**, or **Last Name**. Existing users may also be modified and removed using the **Edit** and **Delete** buttons.

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.



#	Username	First Name	Last Name	Select
1.	admin			☐
2.	dsmith	Dave	Smith	☐

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



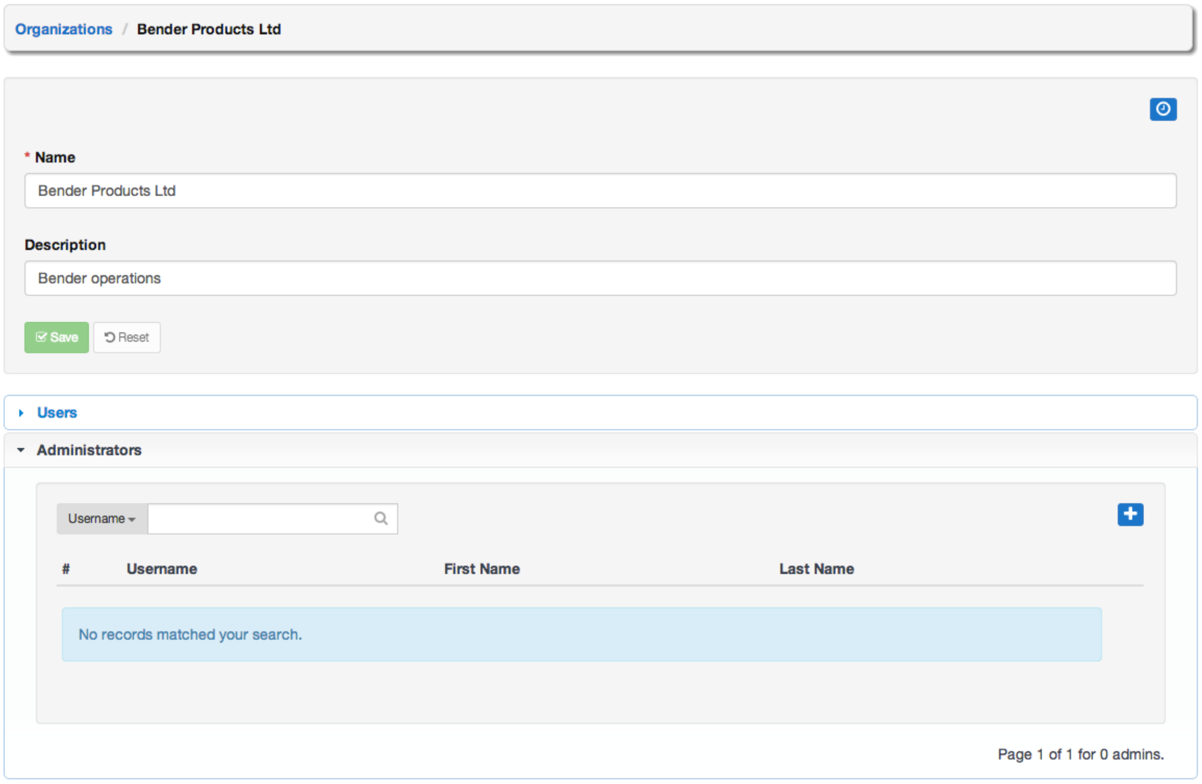
Enter the appropriate details into the following fields:

- First Name
- Last Name
- Email
- Organization
- Username
- Password
- Confirm Password
- Superuser? (Give this user Super User privileges for Tower. Caution!)

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

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**.



* Name: Bender Products Ltd

Description: Bender operations

#	Username	First Name	Last Name

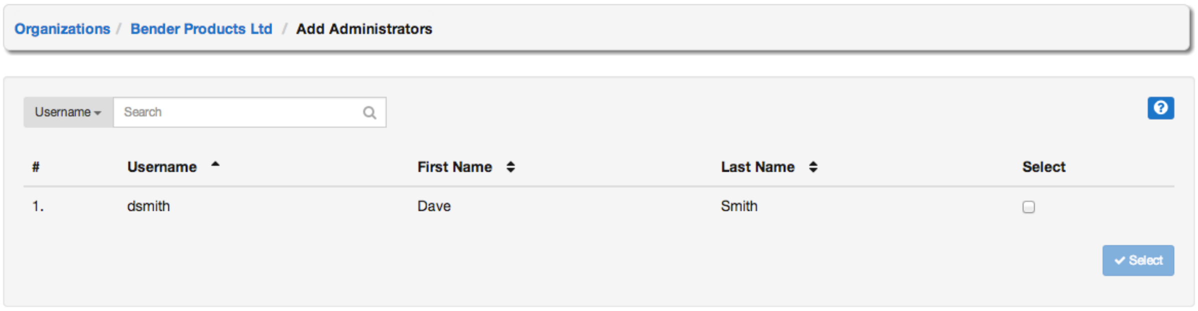
No records matched your search.

Page 1 of 1 for 0 admins.

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

To add an administrator 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.

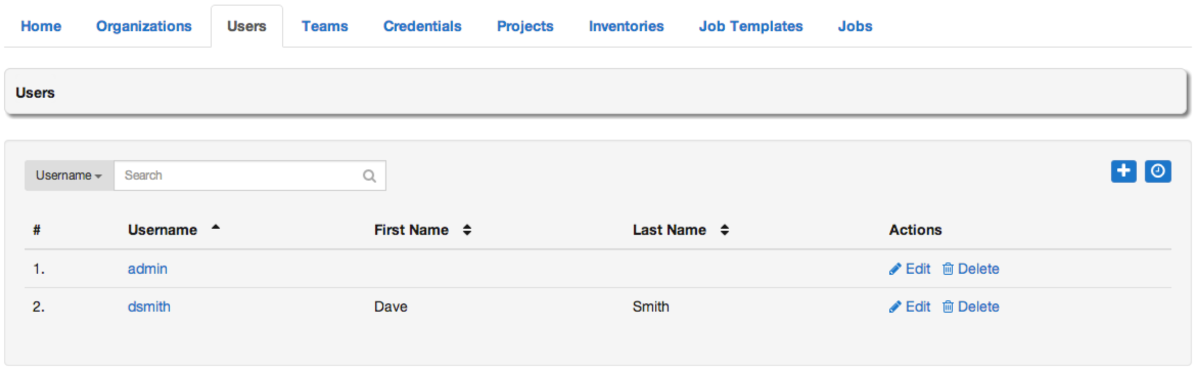

Organizations / Bender Products Ltd / Add Administrators

#	Username	First Name	Last Name	Select
1.	dsmith	Dave	Smith	☐

__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 tab allows you to manage the all Tower users. The user list may be sorted and searched by **Username**, **First Name**, or **Last Name**.


Home Organizations Users Teams Credentials Projects Inventories Job Templates Jobs

Users

#	Username	First Name	Last Name	Actions
1.	admin			Edit Delete
2.	dsmith	Dave	Smith	Edit Delete

There are three types of Tower Users:

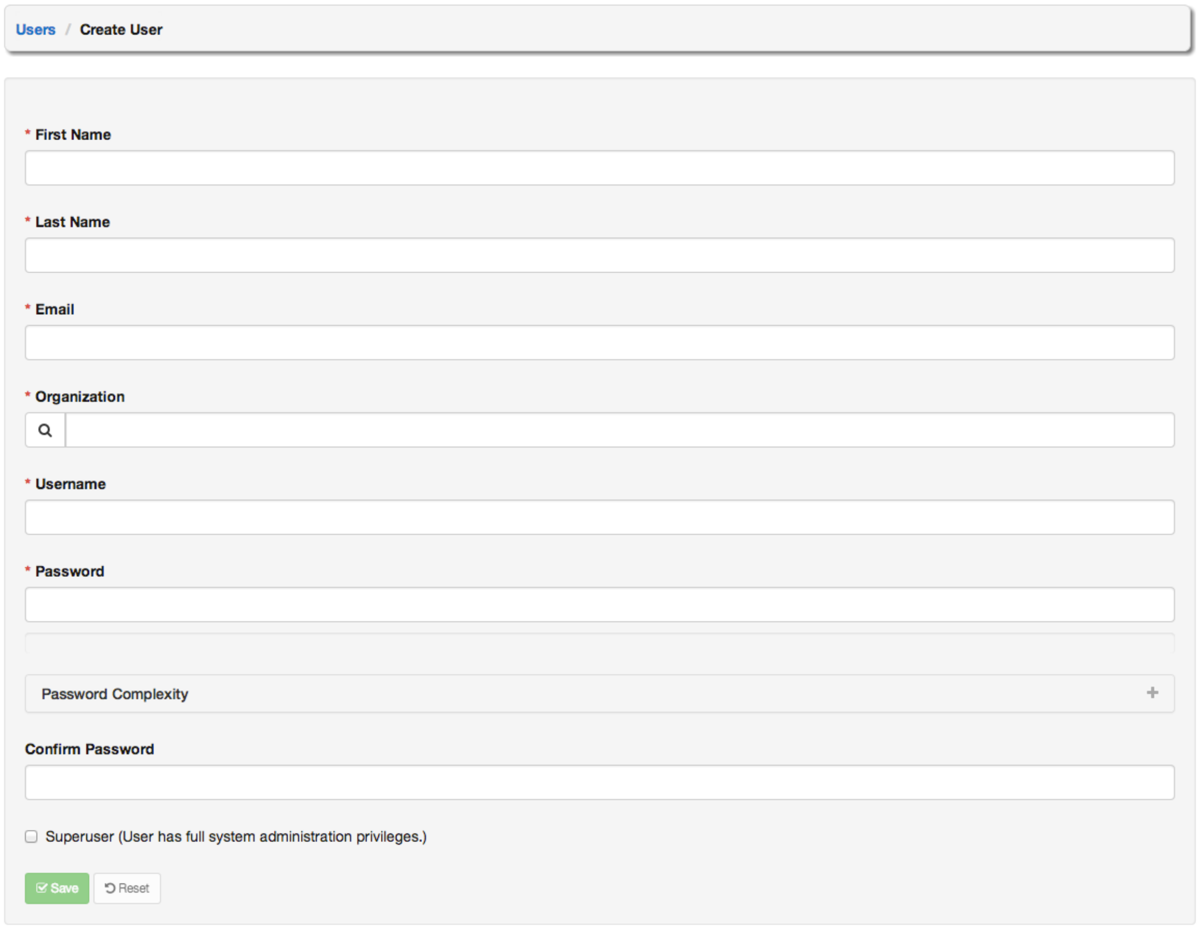
1. **Normal User**: read and write access is limited to the inventory and projects that the user has been granted the appropriate rights to.

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. **Super User**: a Tower Super User has admin, read, and write permissions over the entire Tower installation. A Super User 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 Super User. One Super User must always exist, so if you wish to delete "admin", first create another Super User account.*

To create a new user click the ➕ button, which takes us to the new user dialog.


Users / Create User

* First Name

* Last Name

* Email

* Organization

* Username

* Password

Password Complexity

Confirm Password

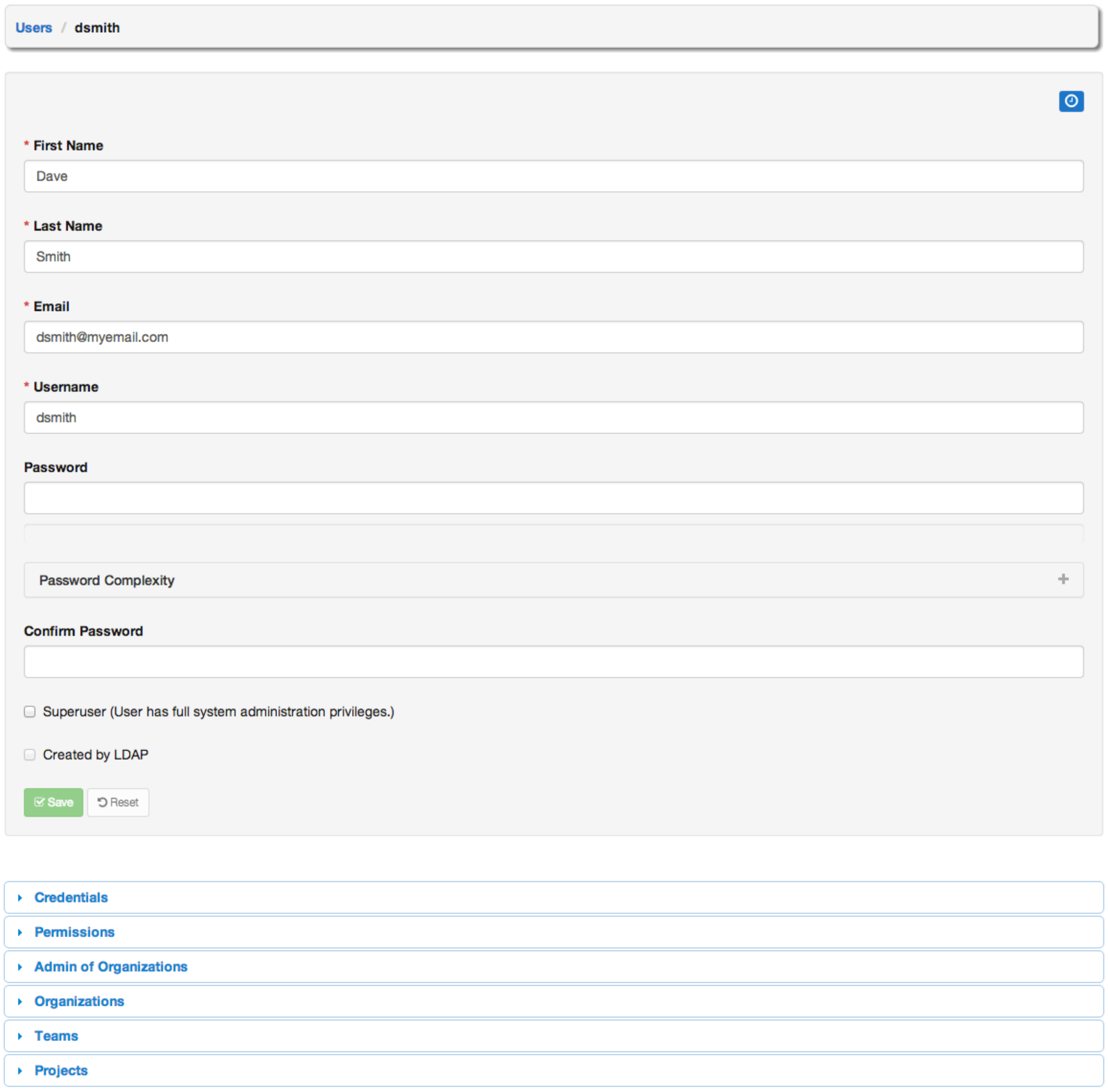
☐ Superuser (User has full system administration privileges.)

Enter the appropriate details into the following fields:

- First Name
- Last Name
- Email
- Organization (Choose from an existing organization)
- Username
- Password
- Confirm Password
- Superuser? (Gives this user admin privileges for Tower. Caution!)

All of these fields are required. Select **Save** when finished.

Once the user is successfully created, Tower will open the **Edit User** dialog. This is the same menu that is opened if the **Edit** button is clicked from the **Users** tab. Here, **User Setting**, **Credentials**, **Permissions**, and other user membership details may be reviewed and modified.


Users / dsmith

* First Name: Dave

* Last Name: Smith

* Email: dsmith@myemail.com

* Username: dsmith

Password:

Confirm Password:

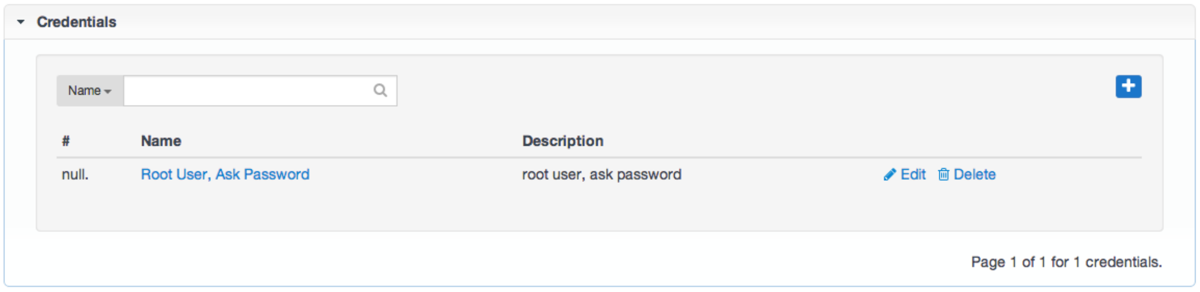
Superuser (User has full system administration privileges.): ☐

Created by LDAP: ☐

Users - Credentials

Credentials are utilized by Tower for authenticating when launching jobs against machines, to synchronize with inventory sources, and to import project content from version control systems. For details about how to use credentials, please see the section Credentials.

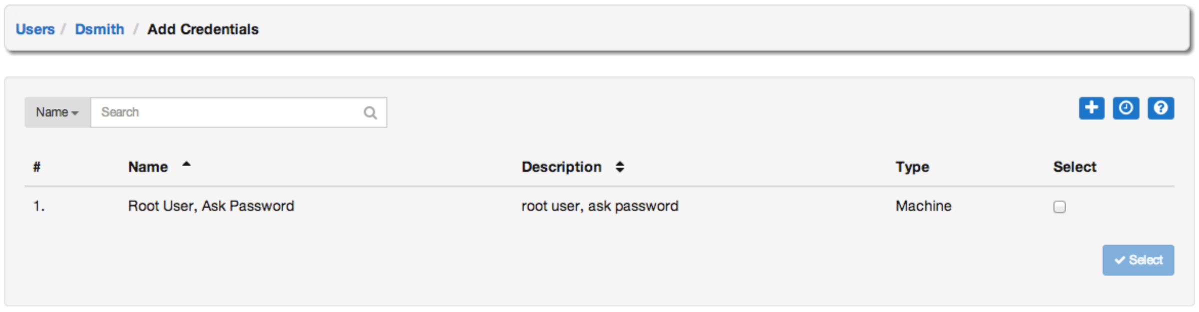
To add a credential to user, expand the credentials menu and click the ➕ button.


Credentials

#	Name	Description	
null.	Root User, Ask Password	root user, ask password	Edit Delete

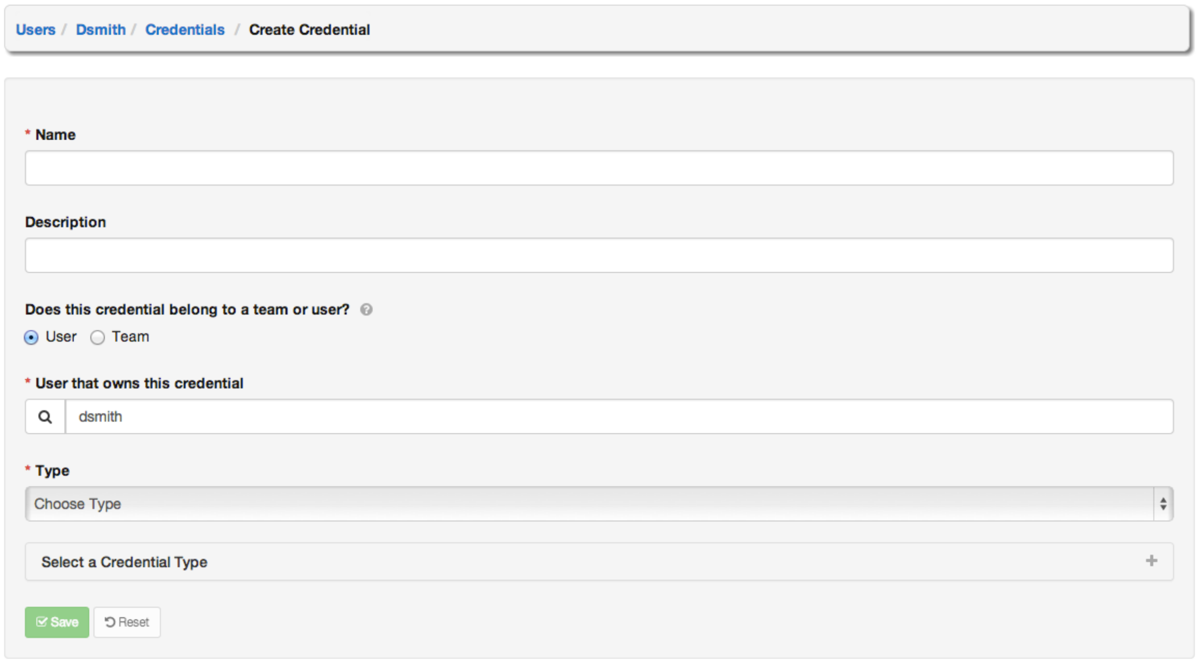
Page 1 of 1 for 1 credentials.

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


Users / Dsmith / Add Credentials

#	Name	Description	Type	Select
1.	Root User, Ask Password	root user, ask password	Machine	☐

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


Users / Dsmith / Credentials / Create Credential

* Name

Description

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

* User that owns this credential
dsmith

* Type
Choose Type

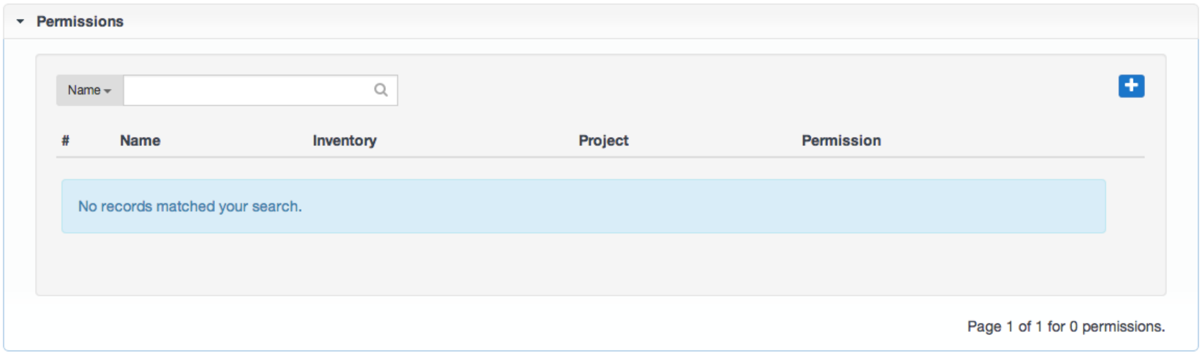
Select a Credential Type

Enter the appropriate details depending on the type of credential and select **Save**. (For details about credential types, please see the section Credentials.)

Users - Permissions

Permissions are the set of privileges assigned to users and teams that provide the ability to read, modify, and administer projects, inventories, and other Tower elements. For details about how to use permissions, please see the section Permissions.

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.

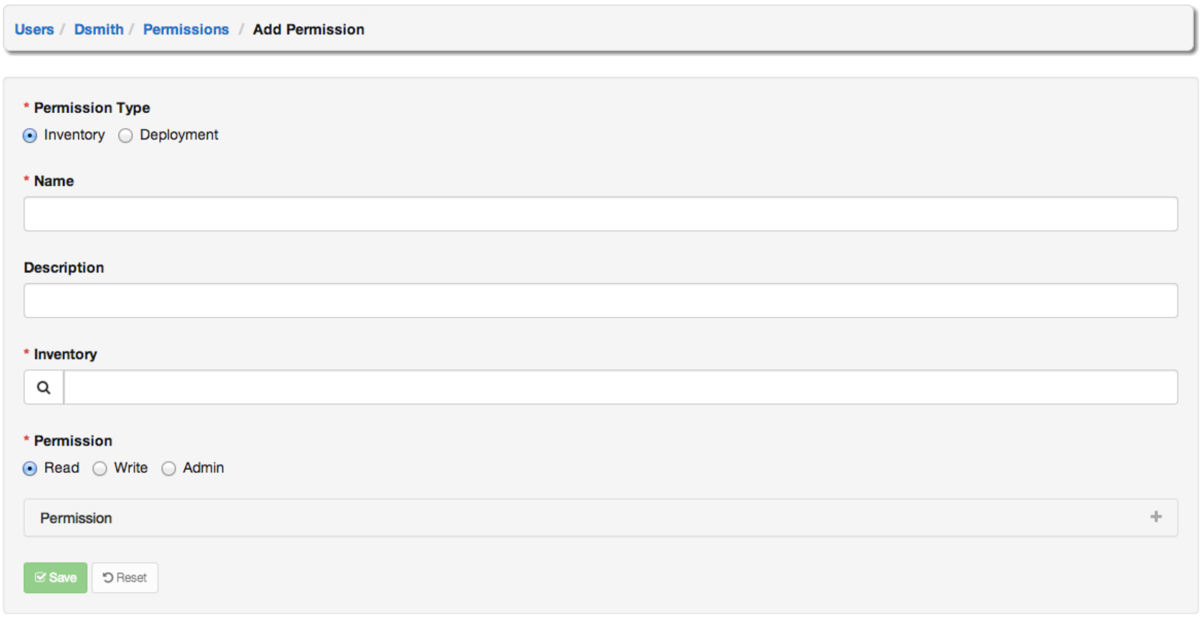


#	Name	Inventory	Project	Permission

No records matched your search.

Page 1 of 1 for 0 permissions.

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



* Permission Type
[✓ Inventory] Deployment

* Name

Description

* Inventory

* Permission
[✓ Read] Write Admin

Enter the appropriate details into the following fields:

- Permission Type
  - Inventory
  - Deployment
- Name

- Description

Selecting a **Permission Type** of either **Inventory** or **Deployment** will change 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
    - Admin
    - Read
    - Write

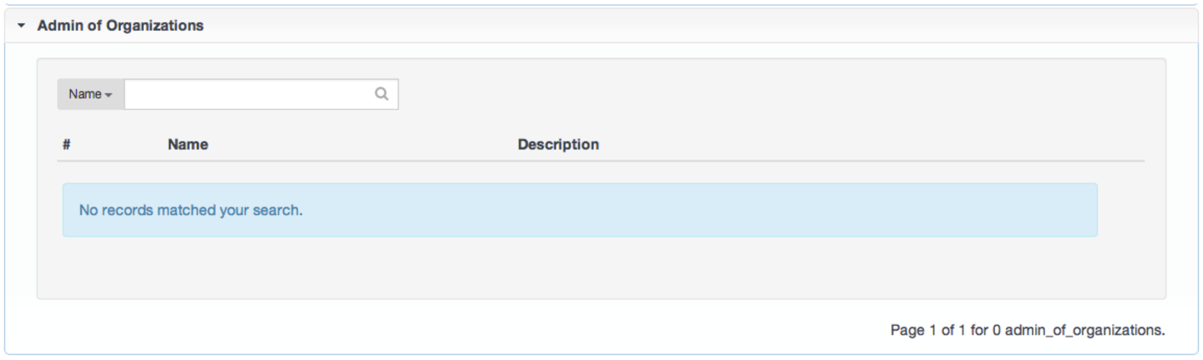
For a permission of type **Deployment**, enter the following details:

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

Select **Save**.

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.

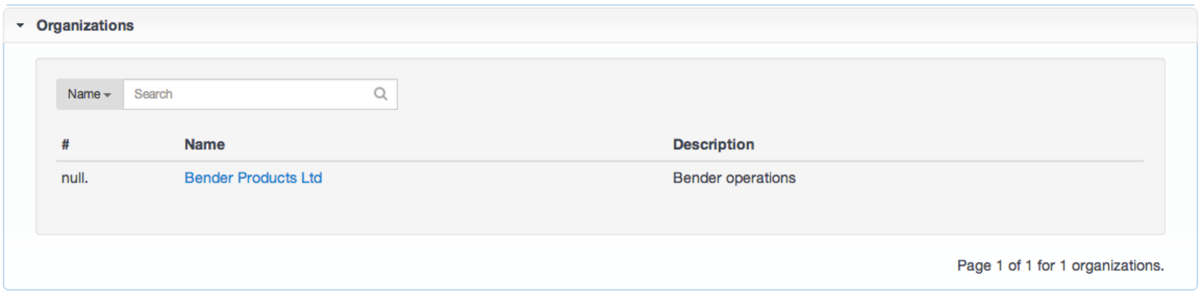


No records matched your search.

Page 1 of 1 for 0 admin_of_organizations.

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.

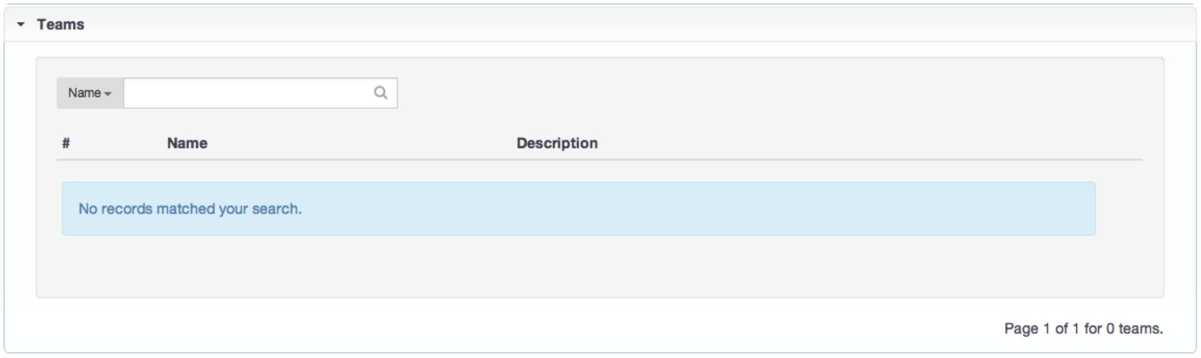


#	Name	Description
null.	Bender Products Ltd	Bender operations

Page 1 of 1 for 1 organizations.

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.



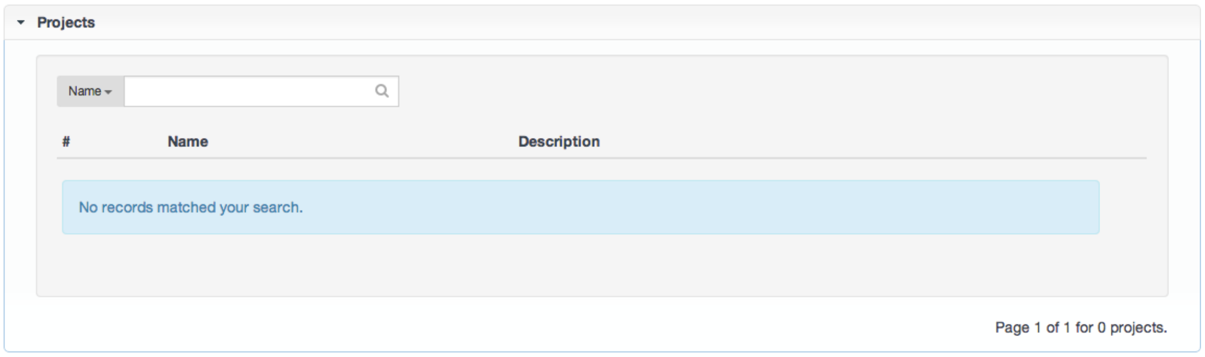
#	Name	Description

No records matched your search.

Page 1 of 1 for 0 teams.

Users - Projects

This displays the list of projects that this user has access to. This list may be searched by **Project Name** or **Description**. Project access cannot be modified from this display. For more information about projects, please see the section Projects.



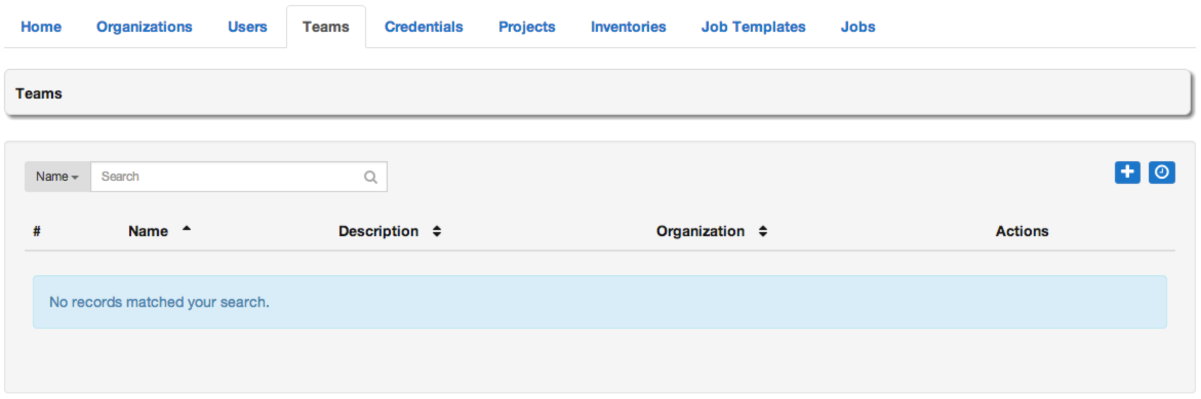
No records matched your search.

Page 1 of 1 for 0 projects.

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.

This tab allows you to manage the teams for Tower. The user list may be sorted and searched by **Username**, **Description**, or **Organization**.

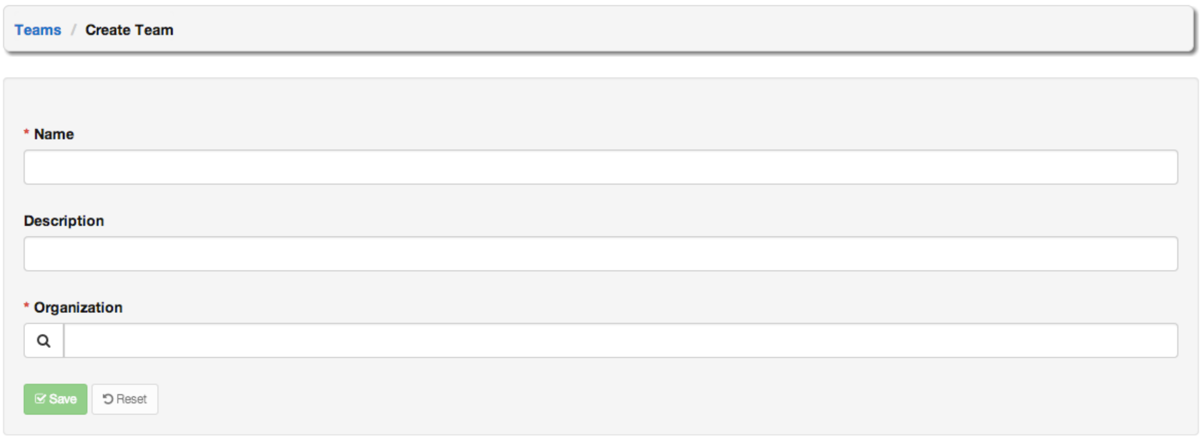


No records matched your search.

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

- Create a new team
- View Activity Stream

To create a new team, click the ✚ button.

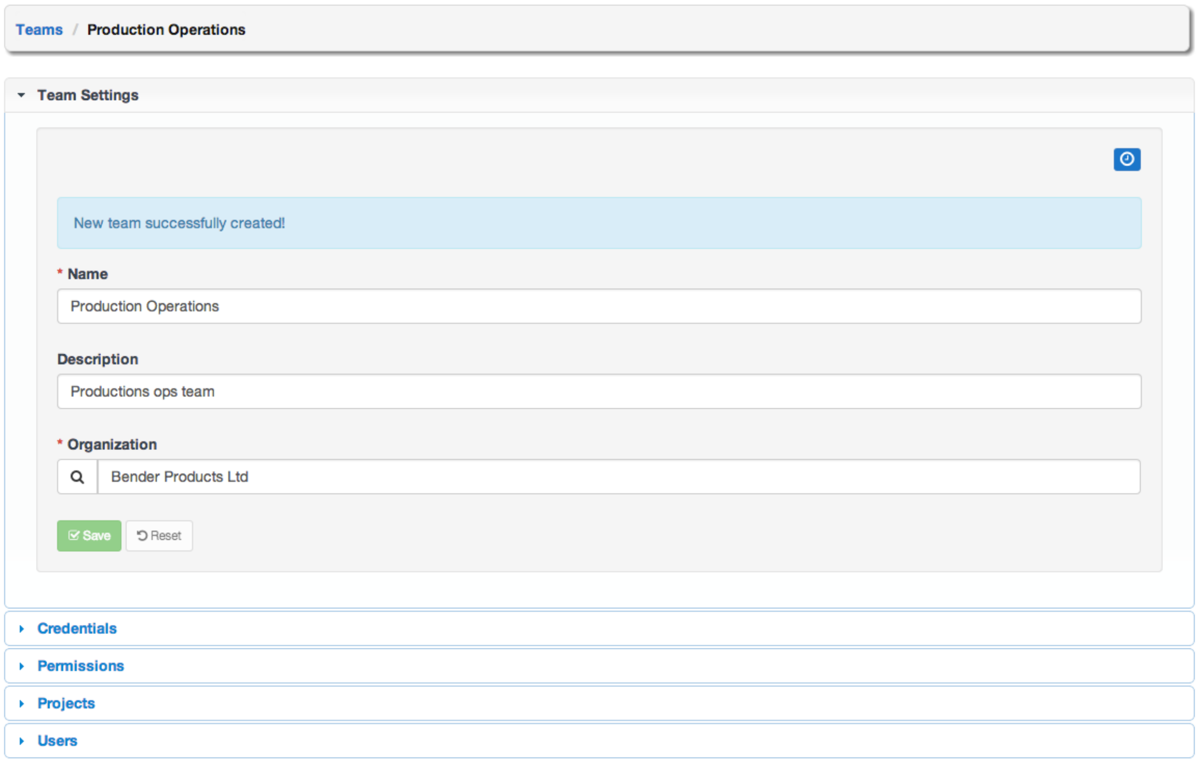


Enter the appropriate details into the following fields:

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

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

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

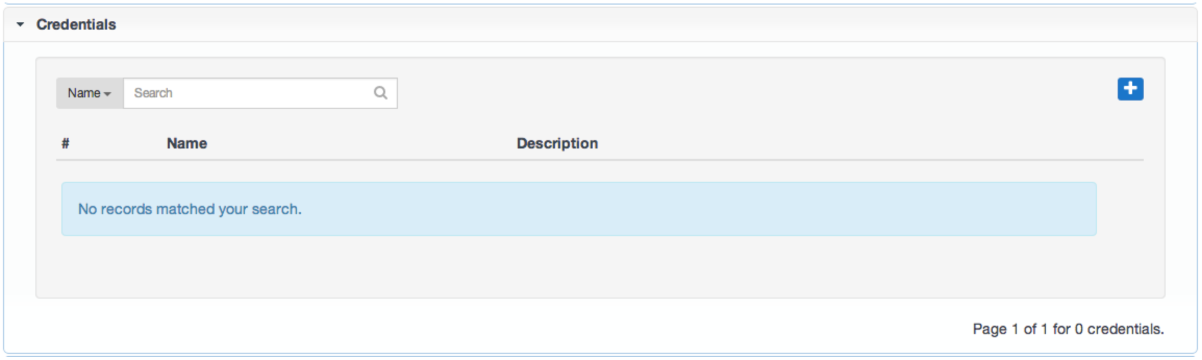


New team successfully created!

* Name: Production Operations
Description: Productions ops team
* Organization: Bender Products Ltd

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, please see the section Credentials.

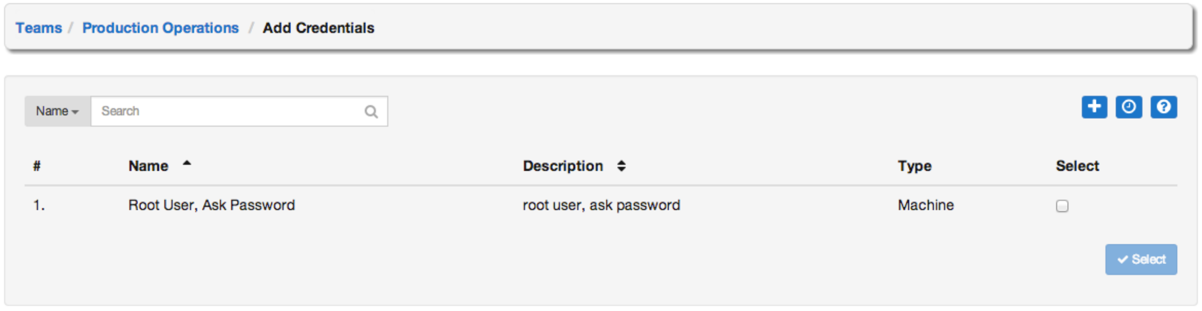


#	Name	Description

No records matched your search.

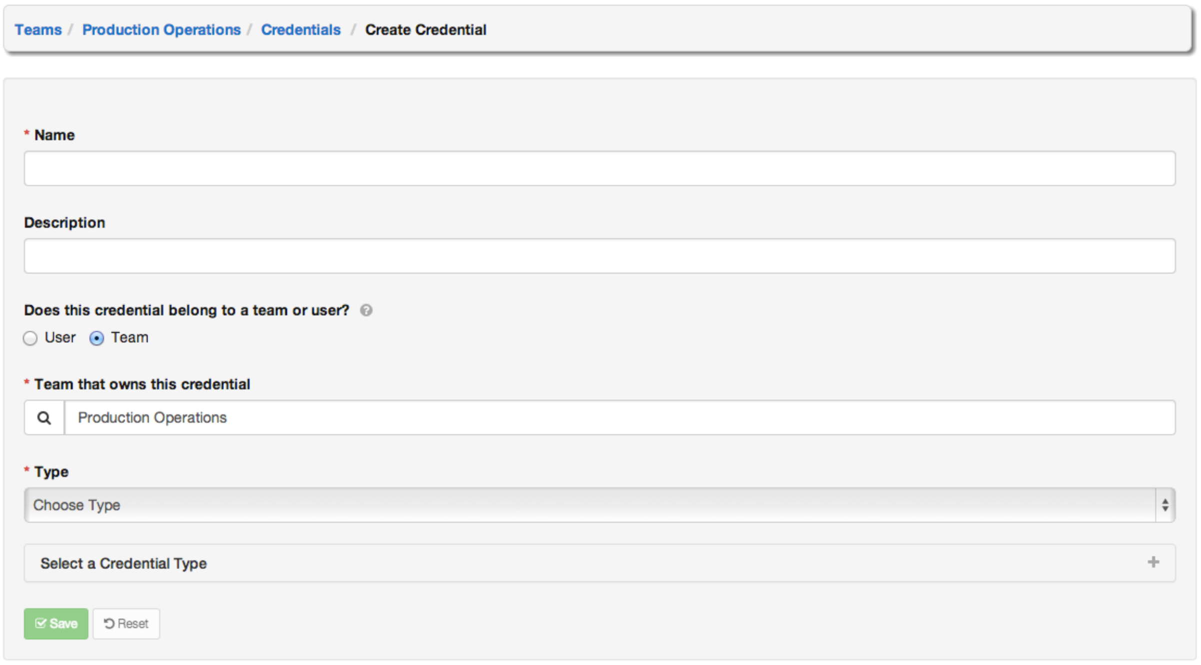
Page 1 of 1 for 0 credentials.

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.



#	Name	Description	Type	Select
1.	Root User, Ask Password	root user, ask password	Machine	☐

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


Teams / Production Operations / Credentials / Create Credential

* Name

Description

Does this credential belong to a team or user?

○ User [✓ Team]

* Team that owns this credential

Production Operations

* Type

Choose Type

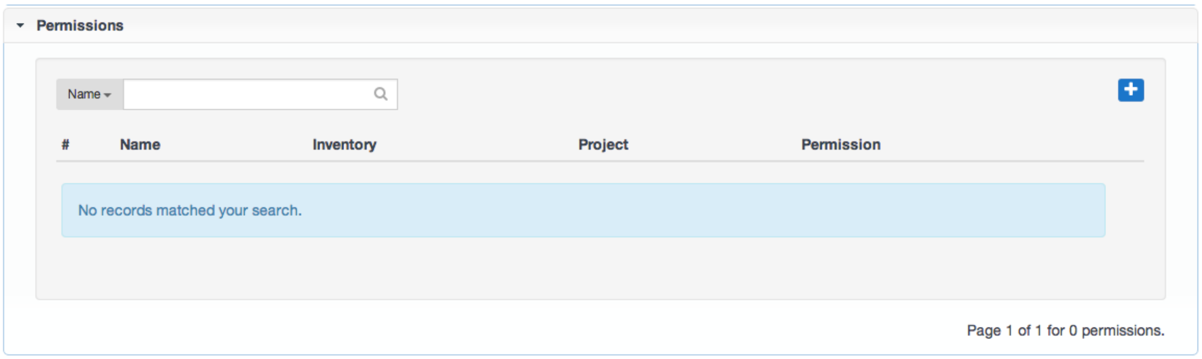
Select a Credential Type

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

Teams - Permissions

Permissions are the set of privileges assigned to users and teams that provide the ability to read, modify, and administer projects, inventories, and other Tower elements. For details about how to use permissions, please see the section Permissions.

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.

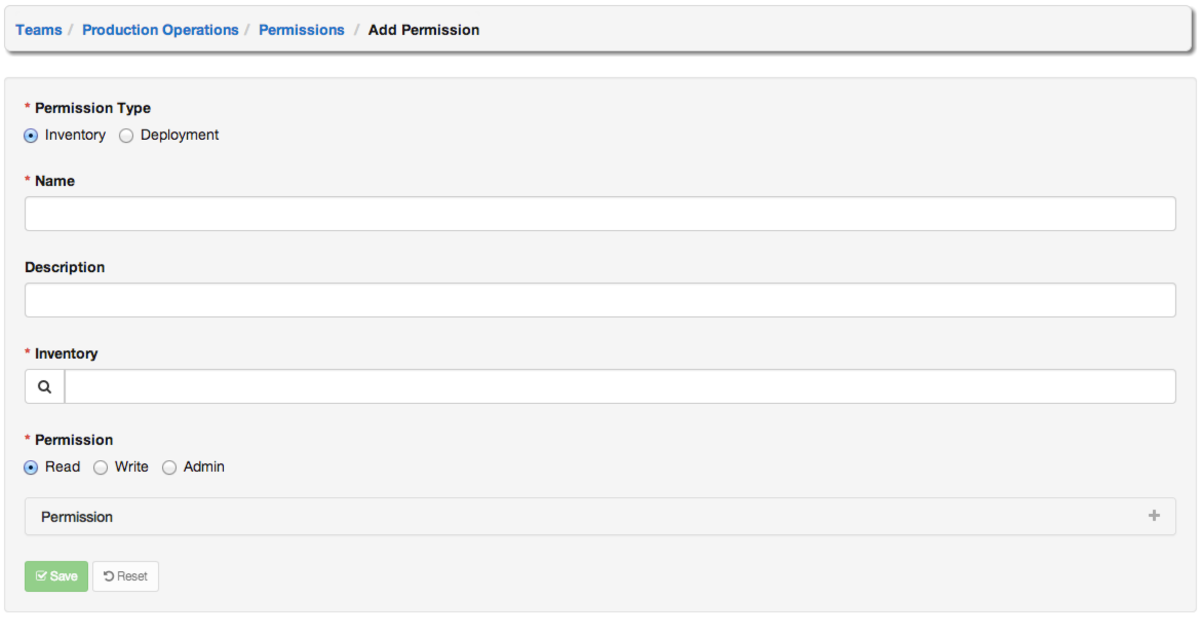


#	Name	Inventory	Project	Permission

No records matched your search.

Page 1 of 1 for 0 permissions.

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


Teams / Production Operations / Permissions / Add Permission

* Permission Type
[● Inventory] ○ Deployment

* Name

Description

* Inventory

* Permission
[● Read] ○ Write ○ Admin

Enter the appropriate details into the following fields:

- Permission Type
  - Inventory
  - Deployment
- Name
- Description

Selecting a **Permission Type** of either **Inventory** or **Deployment** will change 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
  - Admin
  - Read
  - Write

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

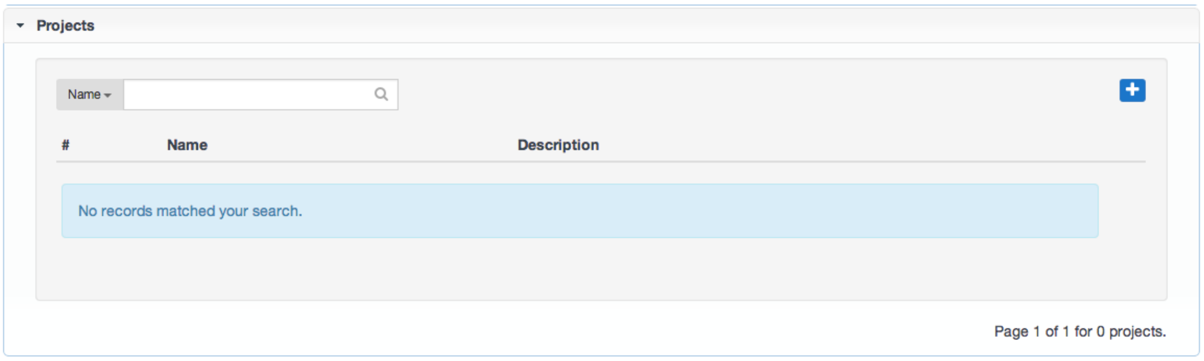
- Project (Select from the available projects)
- Inventory (Select from the available inventories)

- Permission
    - Run
    - Check

Select **Save**.

Teams - Projects

This displays the list of projects that this team has access to. This list may be searched by **Project Name** or **Description**. For more information about projects, please see the section Projects.



Projects

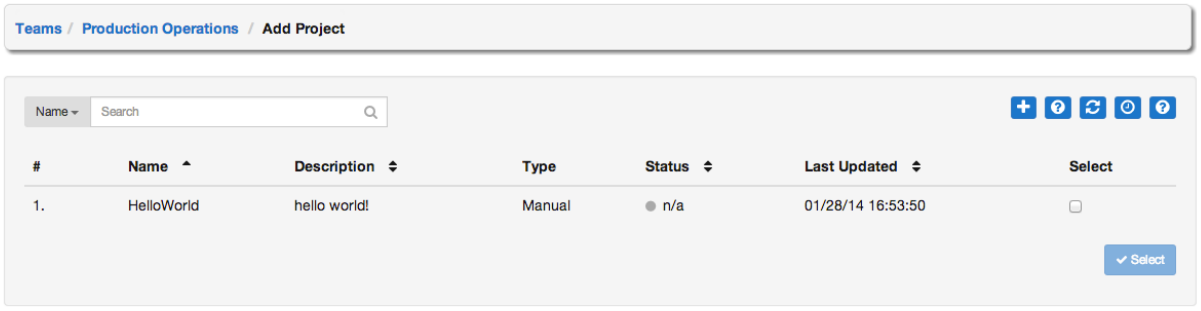
Name

#	Name	Description

No records matched your search.

Page 1 of 1 for 0 projects.

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

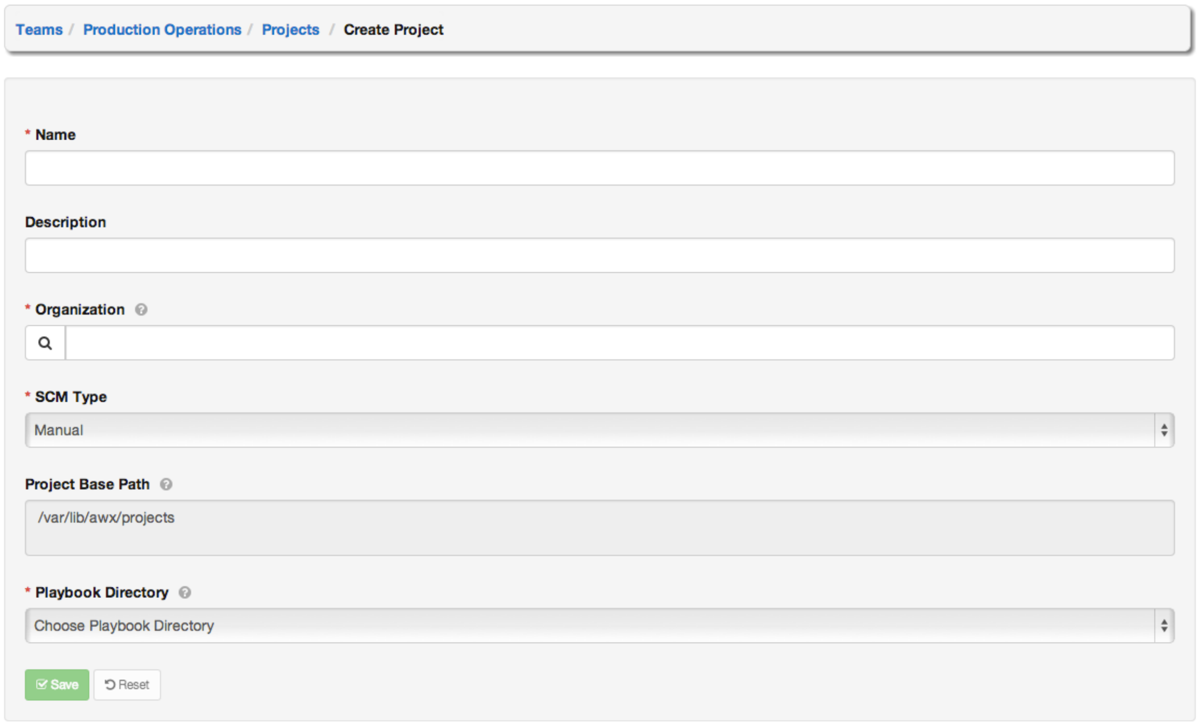


Teams / Production Operations / Add Project

Name Search

#	Name	Description	Type	Status	Last Updated	Select
1.	HelloWorld	hello world!	Manual	● n/a	01/28/14 16:53:50	☐

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


Teams / Production Operations / Projects / Create Project

* Name

Description

* Organization

* SCM Type
Manual

Project Base Path
/var/lib/awx/projects

* Playbook Directory
Choose Playbook Directory

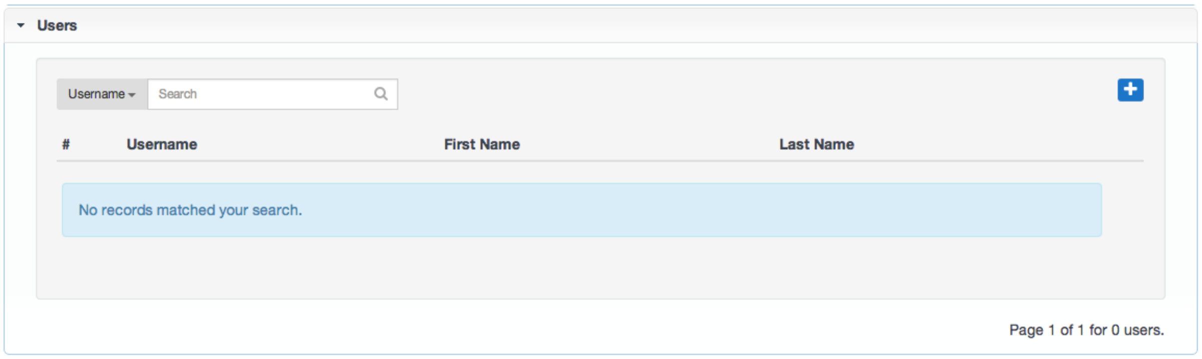
Enter the appropriate details into the following fields:

- Name
- Description
- Organization
- SCM Type (Select one of Manual, Git, SVN, or Mercurial.)
- Project Base Path (Shown here as a convenience.)
- Project Path

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

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 on users, please see the section Users.

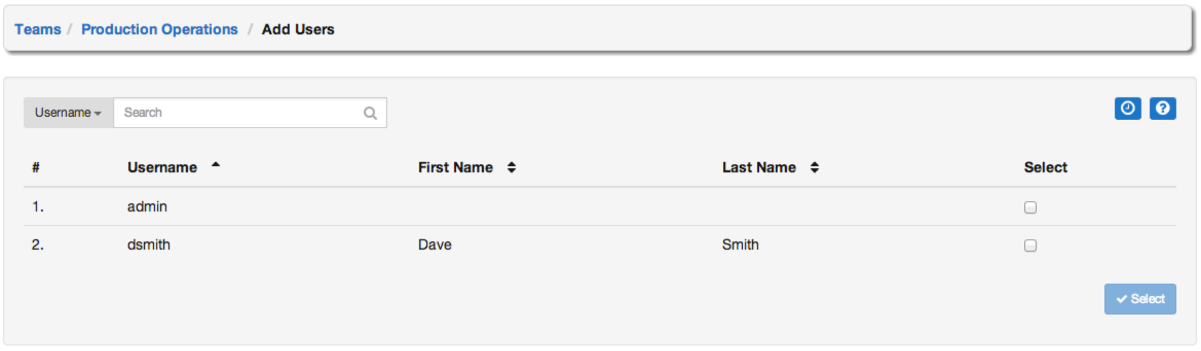


#	Username	First Name	Last Name

No records matched your search.

Page 1 of 1 for 0 users.

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.



#	Username	First Name	Last Name	Select
1.	admin			☐
2.	dsmith	Dave	Smith	☐

Permissions

Permissions are rights given to users to perform actions, including manage inventory and invoke Ansible playbooks / roles.

Permissions do not have their own tab in Tower, but may be managed from either or both of the **Users** and **Teams** tabs. (See those sections for information on how to modify, add, and delete 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

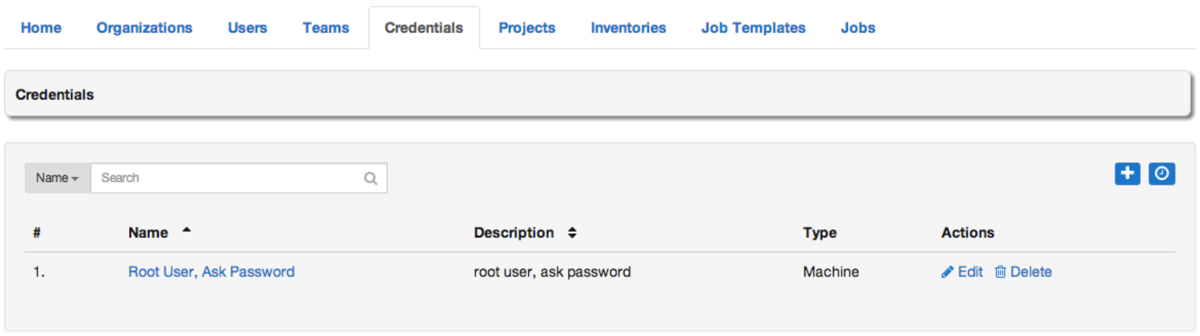
    - Admin: modify the settings for the specified inventory. This permission also grants Read and Write permissions.
    - Read: view groups and hosts within a specified inventory
    - Write: create, modify, and remove groups, and hosts within a specified inventory. Does not give permission to modify the inventory settings. This permission also grants the Read permission.
- Deployment: grants permission to launch jobs from the specified project against the specified inventory
    - Run: launch jobs of type Run. This permission also grants the Check permission.
    - Check: launch jobs of type Check.

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.

NOTE: *Tower encrypts passwords and key information in the Tower database and never makes secret information visible via the API.*

The **Credentials** tab displays a list of the credentials that are currently available. The credentials list may be sorted and searched by **Name**, **Description**, or **Type**.



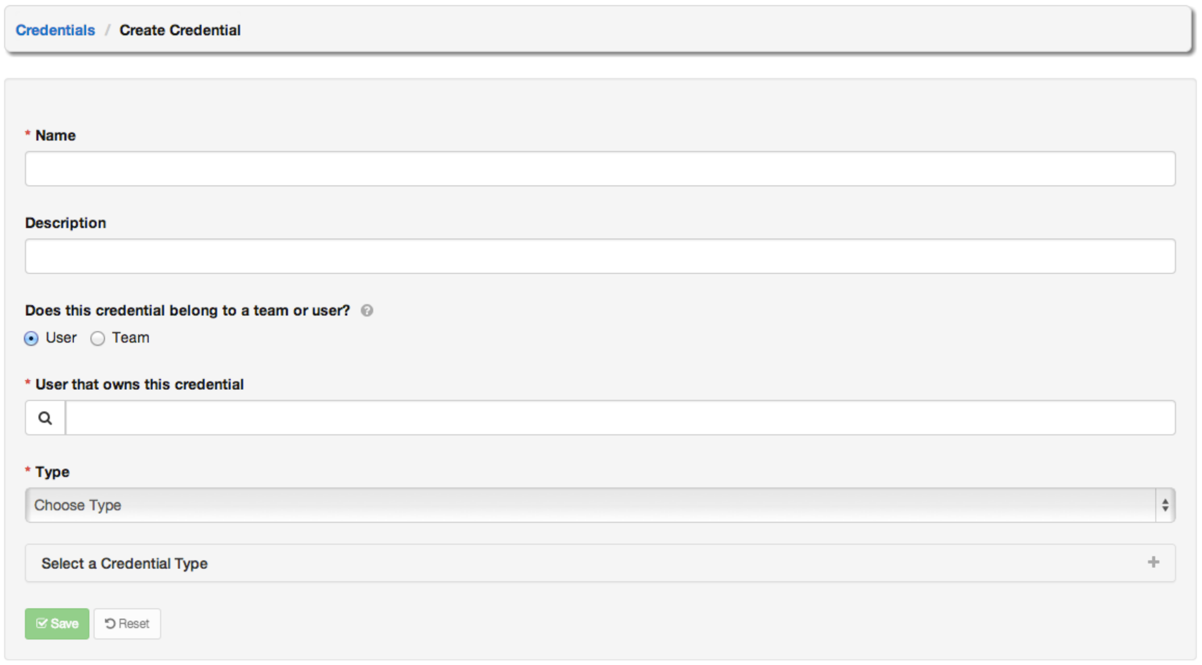
#	Name	Description	Type	Actions
1.	Root User, Ask Password	root user, ask password	Machine	Edit Delete

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

- Create a new credential
- View Activity Stream

Add a new credential

Create a new credential by selecting the ➕ button.


Credentials / Create Credential

* Name

Description

Does this credential belong to a team or user?

[✓ User] Team

* User that owns this credential

* Type

Choose Type

Select a Credential Type

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

There are four types of Credentials:

1. Machine

Define SSH and Sudo access for playbooks. Used when submitting jobs to run playbooks on a remote host.

Machine credentials have several attributes that may be configured:

- **SSH Password**

The actual password to be used to authenticate the user via SSH. 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 a job that uses this credential is launched by selecting Ask at runtime. In that case, a dialog will open when the job
is launched where the user may enter the password and password confirmation.

- **SSH 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.

- **SSH Private Key with Key Password**

In addition to using an SSH private key, you may configure a Key Password associated with the private key. 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 a job that uses this credential is launched by selecting Ask at runtime. In that case, a dialog will open when the job is launched where the user may enter the password and password confirmation.

- **Sudo Password**

The actual password to be used to authenticate the user via sudo. 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 a job that uses this credential is launched by selecting Ask at runtime. In that case, a dialog will open when the job
is launched where the user may enter the password and password confirmation.

Sudo Password must be used in combination with one of the other methods, since Tower must first establish an authenticated SSH connection with the host prior to invoking sudo to change to the sudo user.

- **Vault Password**

Ansible Vault may be used in conjunction with a Machine Credential. Enter the actual vault password into both fields. Alternatively, you may configure Tower to ask the user for the vault password when a job that uses this credential is launched by selecting "Ask at runtime." In that case, a dialog will open when the job is launched into which the user may enter the password and password confirmation.

For more information on how to use Ansible Vault, please visit: http://docs.ansible.com/playbooks_vault.html.

2. AWS

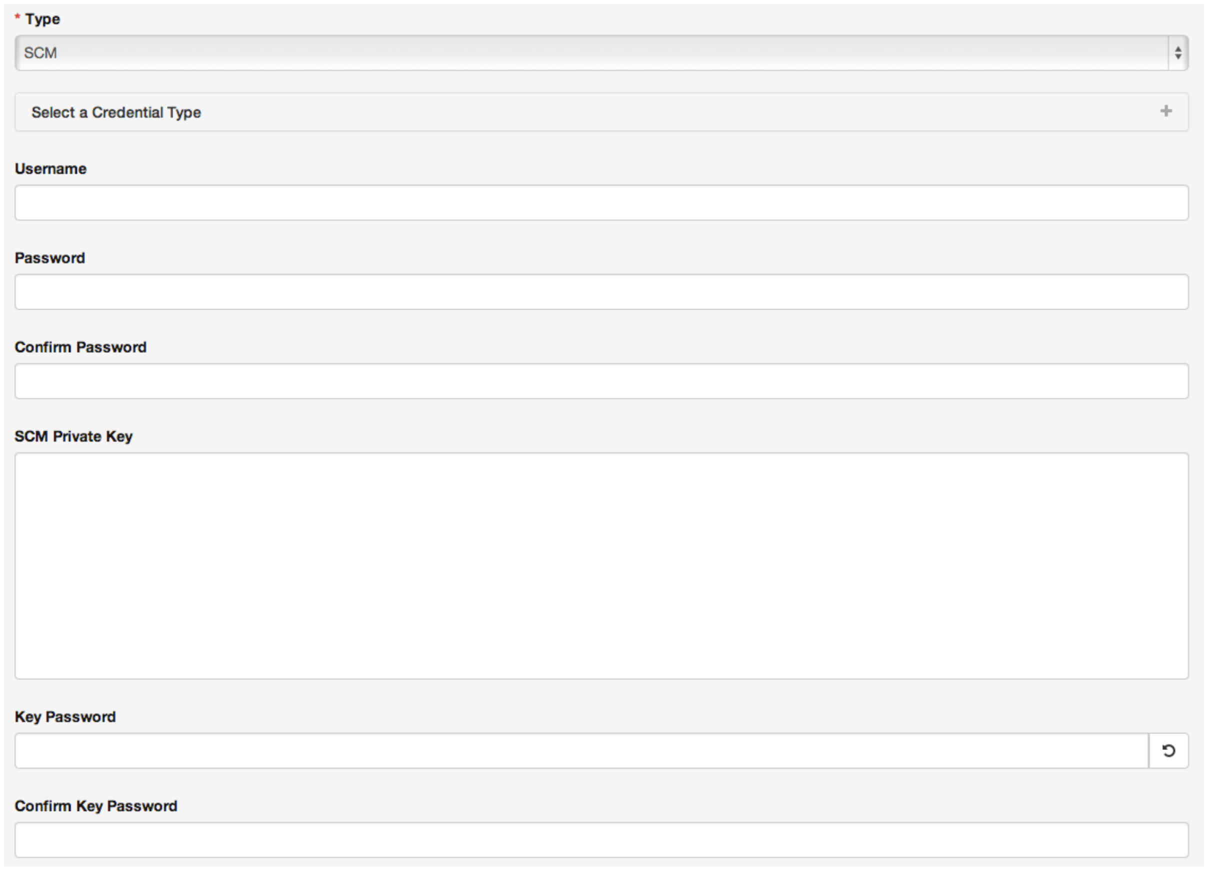
Enables synchronization of cloud inventory with Amazon Web Services. Requires the AWS Access Key and Secret Key.

3. Rackspace

Enables synchronization of cloud inventory with Rackspace. Requires the Rackspace Username and API Key.

4. SCM

Used on projects to clone and update local source code repositories from a remote revision control system such as Git, SVN or Mercurial.



Credentials may also be managed from either the **Teams** tab or the Users tab. To manage credentials for teams, please browse to the **Teams** tab and edit the appropriate team. Likewise, to manage credentials for a user, browse to the **Users** tab and edit the appropriate user.

Credentials added to a **Team** will be available to all members of the team, whereas credentials added to a user are only available to that user, by default.

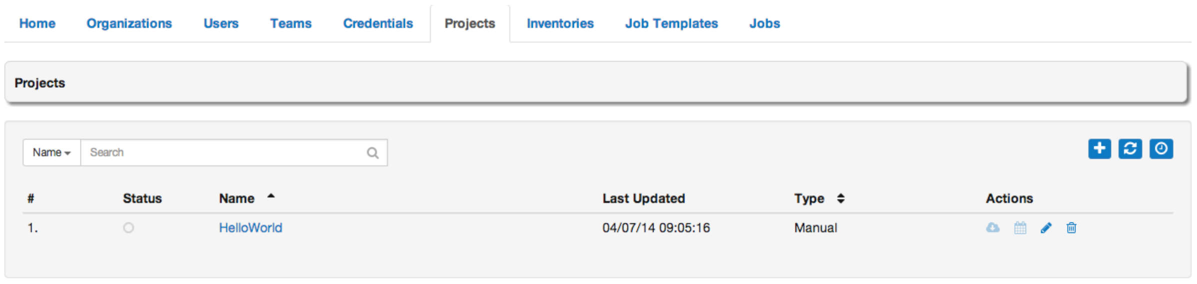
Projects

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

Add your Ansible projects to the filesystem of your Tower installation under the project base path. You can do this by either managing playbooks and playbook directories manually or by using a source code management (SCM) system supported by Tower, including Git, Subversion, and Mercurial.

NOTE: *By default, the* ***Base Project Path*** *is /var/lib/awx/projects, but this may have been modified by the Tower administrator. It is configured in /etc/awx/settings.py. Use caution when editing this file, as it is possible to 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 by **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
1.		HelloWorld	04/07/14 09:05:16	Manual	

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

- Create a new project
- Refresh
- View Activity Stream

For projects managed via Source Code Management (SCM), **Status** will display the update status for the project. Under **Actions**, the following actions are available:

- Invoke an immediate update from SCM, if configured for this project
- Schedule an update from SCM, if configured for this project
- Edit the project
- Delete the project

Status may be one of the following:

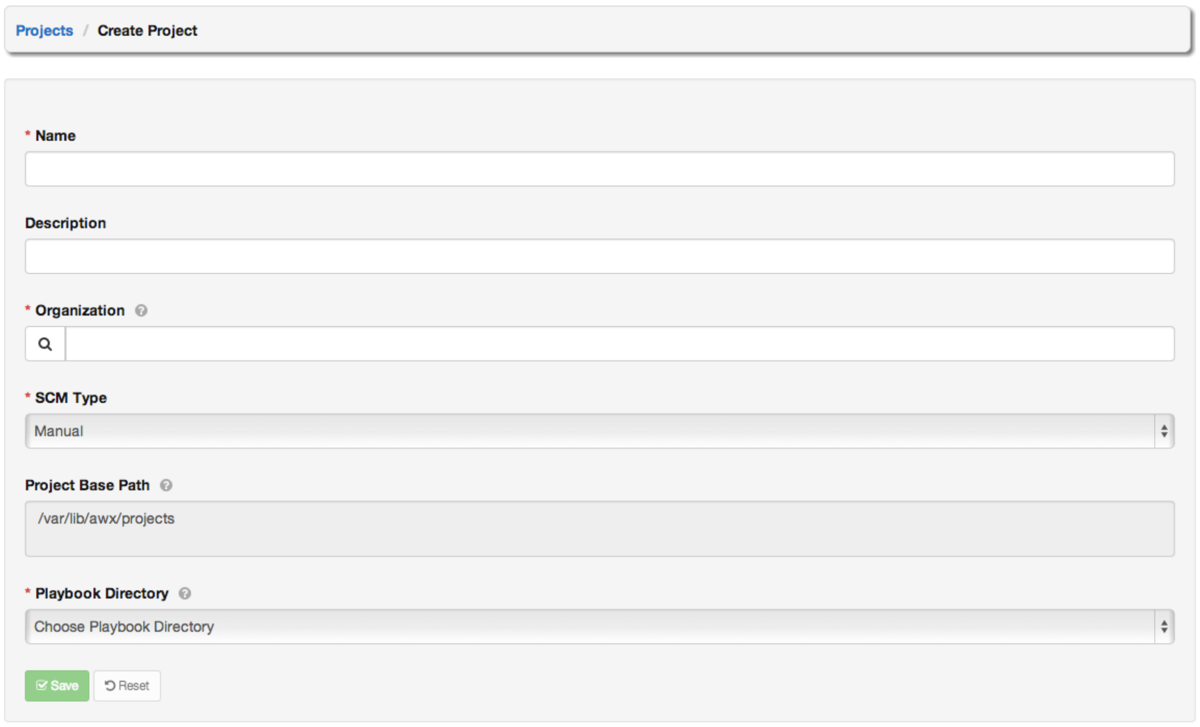
- Updating - an update is in progress
- Never updated - project has never been updated
- Failed - last update failed
- Successful - last updated succeeded

- Missing - project has a last update, but the project directory is missing, or project doesn't use SCM and the directory is missing

- Not configured for SCM

Add a new project

To create a new project, click the ➕ button, which takes us to the **Create Project** dialog.



Projects / Create Project

* Name

Description

* Organization

* SCM Type

Manual

Project Base Path

/var/lib/awx/projects

* Playbook Directory

Choose Playbook Directory

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

*"**WARNING**: There are no unassigned playbook directories in the base project path /var/lib/awx/projects. Either the projects directory is empty, or all of the contents are already assigned to other projects. New projects can be checked out from source control by changing the SCM type option rather than specifying checkout paths manually. To continue with manual setup, log into the Tower server and ensure content is present in a subdirectory under /var/lib/awx/projects. Run "chown -R awx" on the content directory to ensure awx can read the playbooks."*

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.

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, SVN, or Mercurial. (See the appropriate section below for more detail.)
- Project Base Path (Shown here as a convenience.)
- Project Path (The project paths show here are automatically read from the directory tree with a root of the project base path.)

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

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."

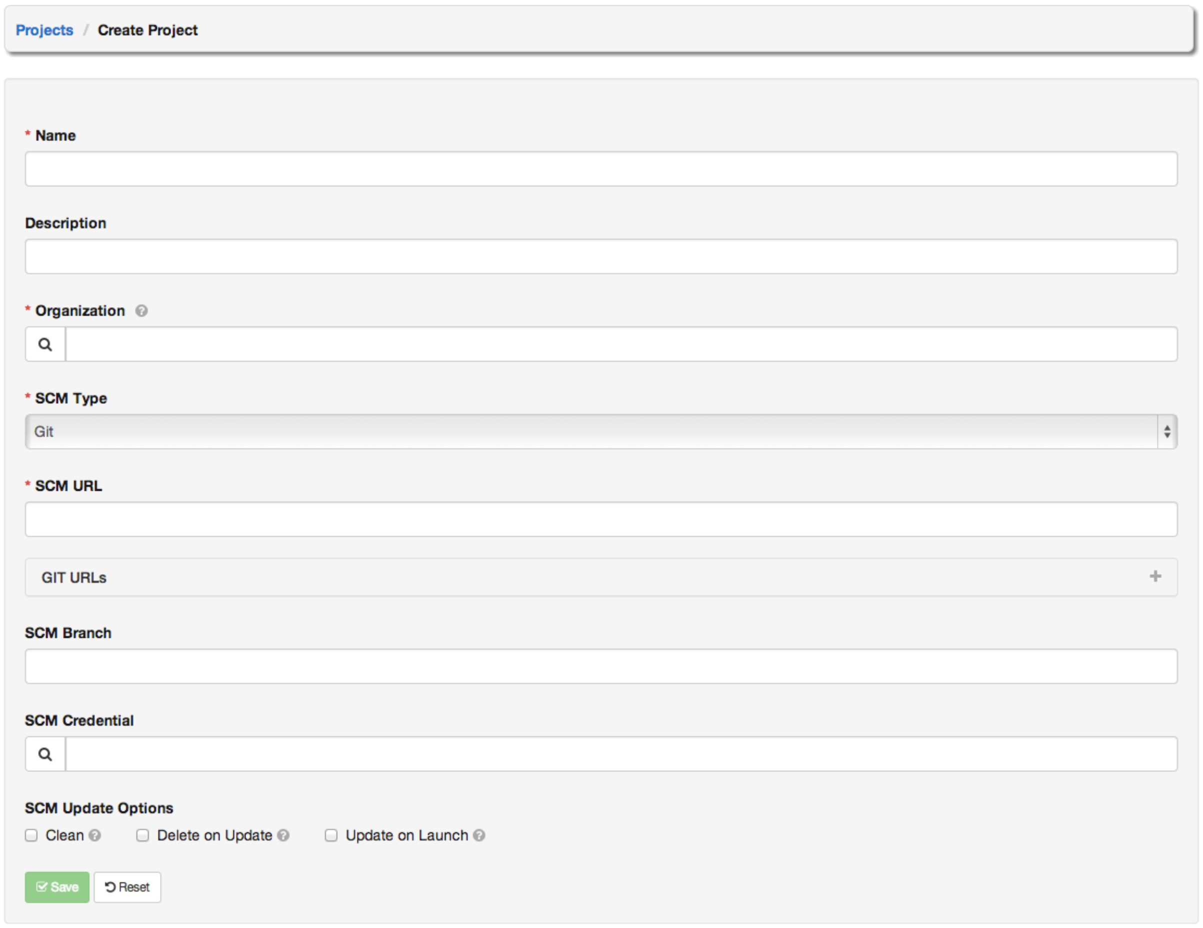
To manage playbooks manually

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

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

To manage playbooks using SCM

1. Select the appropriate **SCM Type**.



Projects / Create Project

* Name

Description

* Organization

* SCM Type

Git

* SCM URL

GIT URLs

SCM Branch

SCM Credential

SCM Update Options

2. Enter the appropriate details into the following fields:

- SCM URL
- SCM Branch

Optionally enter the SCM branch for Git or Mercurial

- Revision # (SVN only)

Optionally enter the Revision # for Subversion

- If authentication is required, select the appropriate SCM credentials.
- Clean

Remove any local modifications prior to performing an update.

- Delete on Update

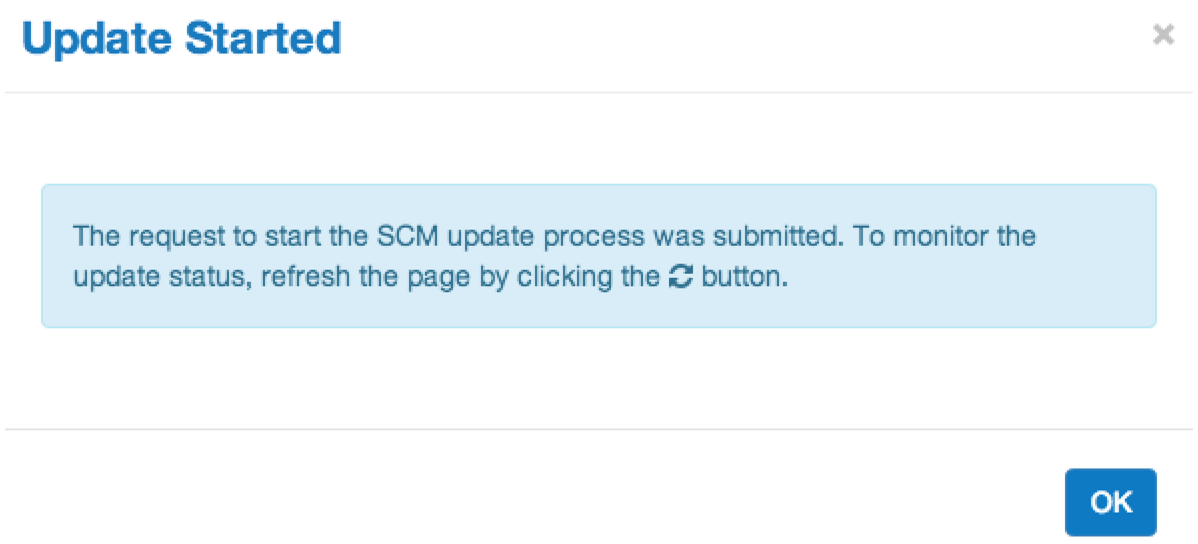
Delete the local repository in its entirety prior to performing an update. Depending on the size of the repository this may significantly increase the amount of time required to complete an update.

- Update on Launch

Each time a job runs using this project, perform an update to the local repository prior to starting the job.

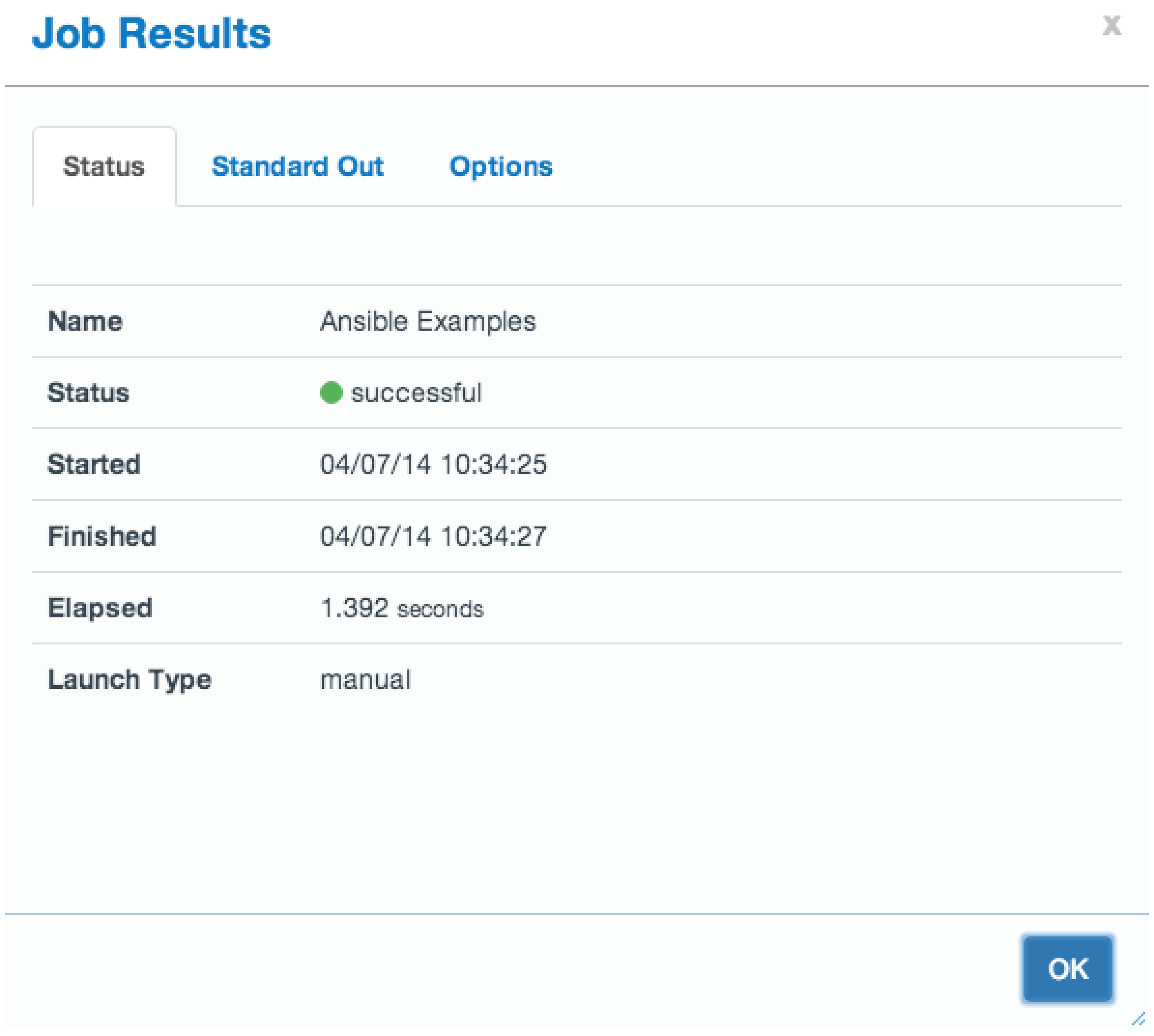
1. Click Save

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


Update Started

The request to start the SCM update process was submitted. To monitor the update status, refresh the page by clicking the button.

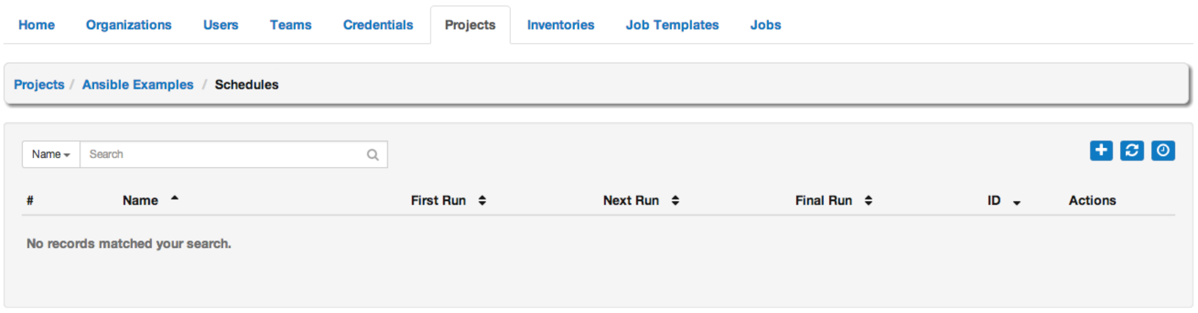
Click the **Status** icon to get further details about the update process:


Job Results

Status | Standard Out | Options

Name	Ansible Examples
Status	● successful
Started	04/07/14 10:34:25
Finished	04/07/14 10:34:27
Elapsed	1.392 seconds
Launch Type	manual

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



#	Name	First Run	Next Run	Final Run	ID	Actions

No records matched your search.

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** or **ID**.

The list of of schedules includes:

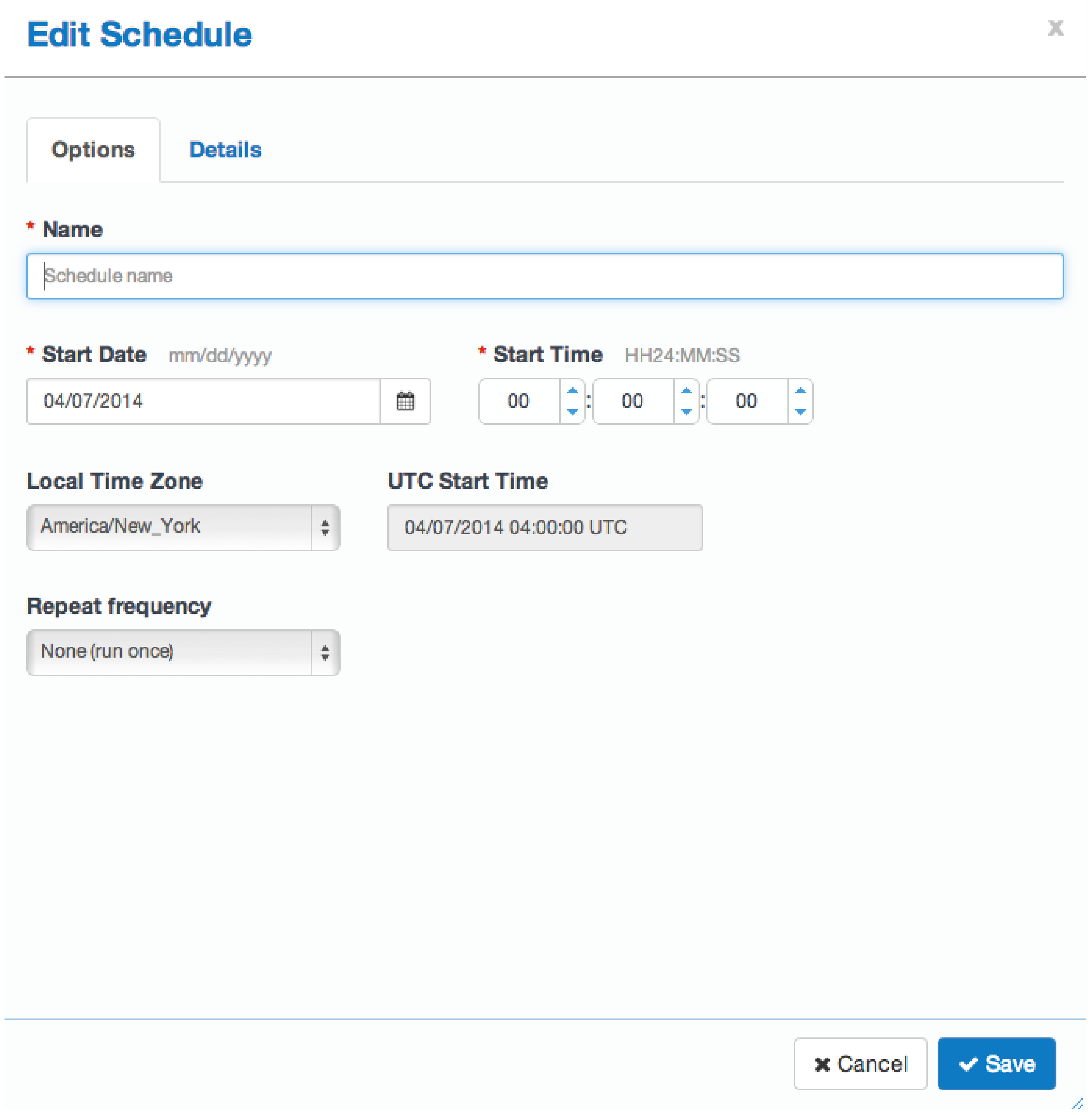
- Name - Clicking the schedule name will open the **Edit Schedule** dialog
- First Run
- Next Run
- Final Run
- ID - Clicking the schedule ID will open the **Edit Schedule** dialog

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

Add a new schedule

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



* Name

Schedule name

* Start Date mm/dd/yyyy

04/07/2014

* Start Time HH24:MM:SS

00 : 00 : 00

Local Time Zone

America/New_York

UTC Start Time

04/07/2014 04:00:00 UTC

Repeat frequency

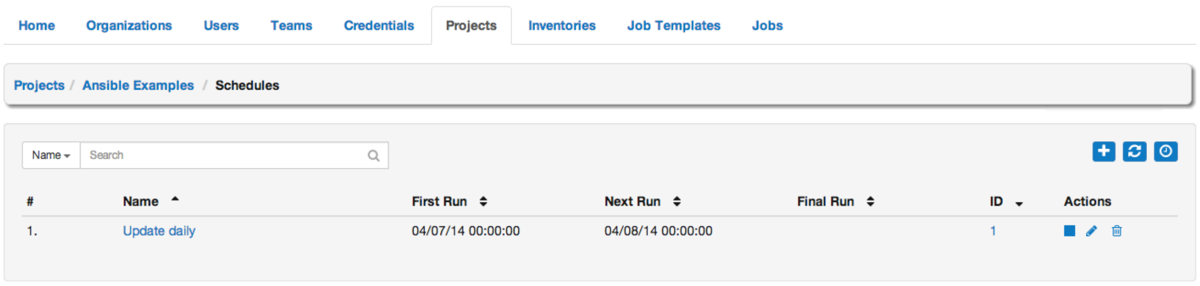
None (run once)

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

- Name (required)
- Start Date (required)
- Start Time (required)
- Local Time Zone
- UTC Start Time
- Repeat Frequency - the appropriate options will display as the update frequency is modified.

The **Details** tab will display a description of the schedule and a list of the scheduled occurrences in Local time or UTC, as selected.

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



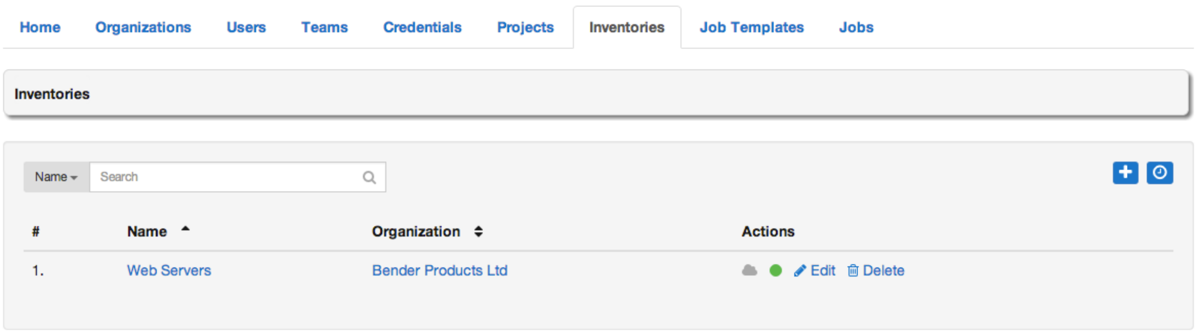
#	Name	First Run	Next Run	Final Run	ID	Actions
1.	Update daily	04/07/14 00:00:00	04/08/14 00:00:00		1	

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

Inventories

An inventory is a collection of hosts against which jobs may be launched. 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 cloud providers, including Amazon Web Services EC2 and Rackspace Cloud Servers.

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 source, by hosts with failed jobs, and inventories that have failed to update with an external source.



#	Name	Organization	Actions
1.	Web Servers	Bender Products Ltd	Edit Delete

The list of of inventories includes:

- Name - the inventory name. Clicking the Inventory name will navigate 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.)

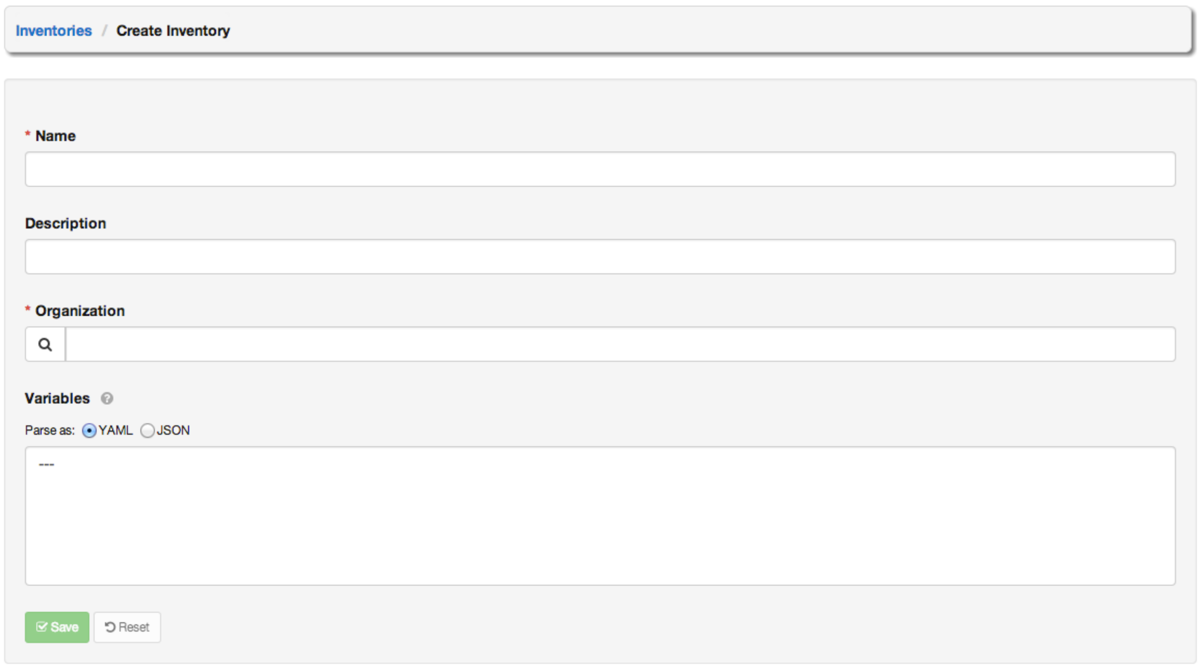
- Organization - the organization that the inventory belongs to
- Action - the following actions are available for the selected inventory:
    - Sync status - Show the status of inventory synchronization for inventories configured with cloud sources
    - Host status - Show the status of successful and failed jobs
    - 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

Add a new inventory

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


Inventories / Create Inventory

* Name

Description

* Organization

Variables

Parse as: [✓ YAML] JSON

```
---
```

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

- Name (required)
- Description
- Organization (Select from the available organizations)
- Variables

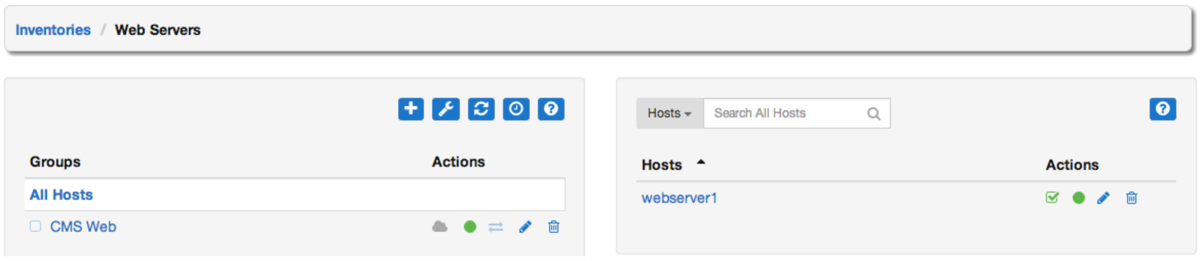
Enter variables using either JSON or YAML syntax. Use the radio button to toggle between the two.

Groups

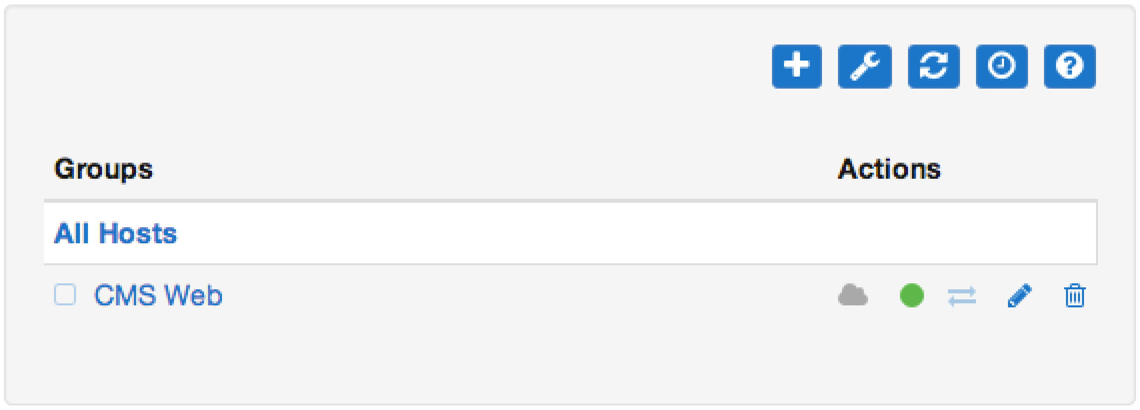
Inventories are divided into groups, which may contain hosts and other groups. *An inventory must contain at least one group*.

To add a group to an inventory or to manage an existing group, select **Edit** from the **Actions** menu for the selected inventory or click the inventory name.

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



There are several actions available for groups.



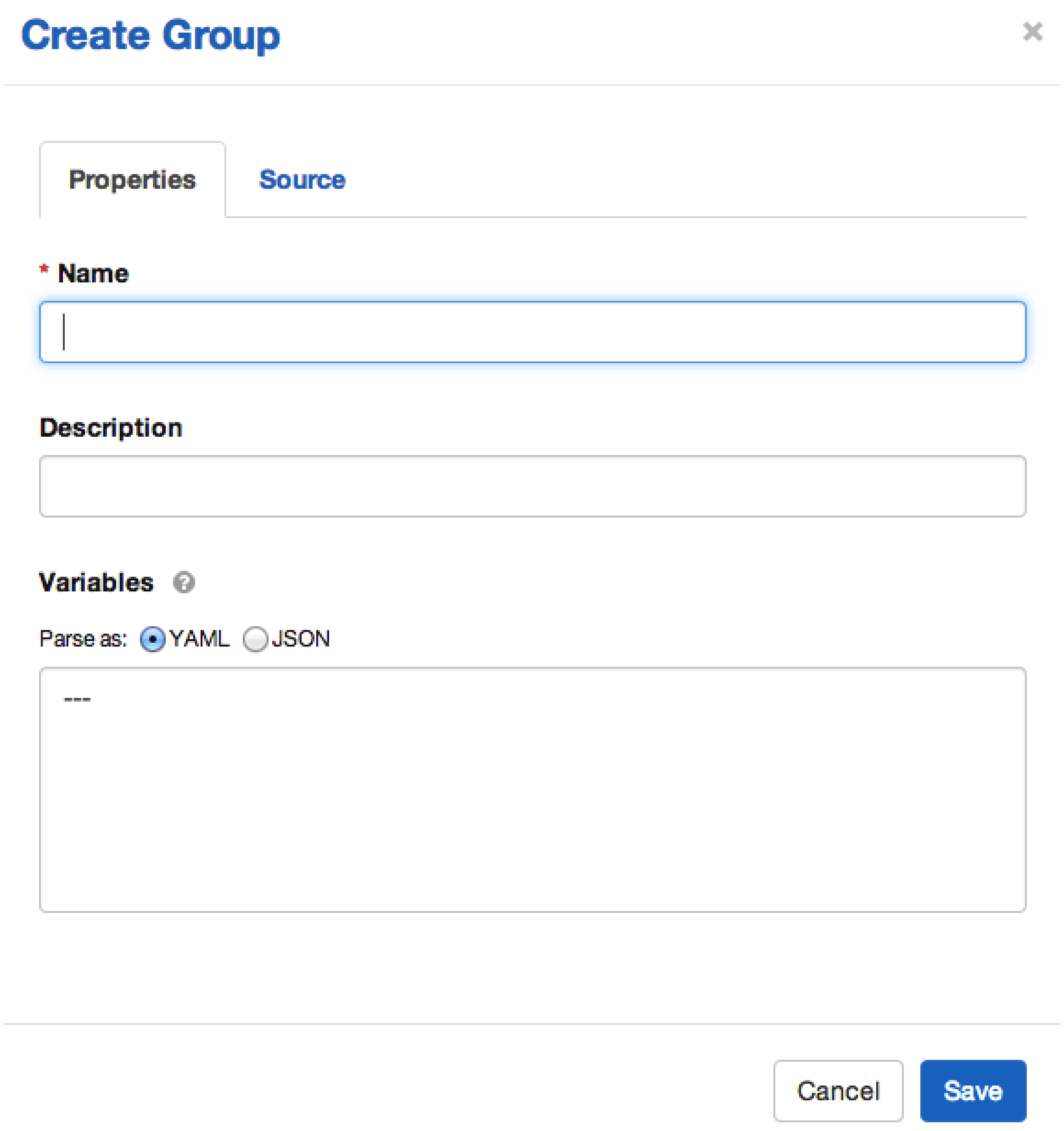
- Create a new Group
- Edit Inventory properties
- Refresh page
- View activity stream
- Help

To edit the group properties, click the 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 will show the synchronization log for the selected group.
- Update - Perform an update on the selected group from its source
- Host status - Show the status of successful and failed jobs for the selected group. Clicking this button will show 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
- Delete - Delete the selected group. This operation cannot be reversed!

Add a new group

Create a new group by clicking the ✚ button, which opens the **Create Group** window.



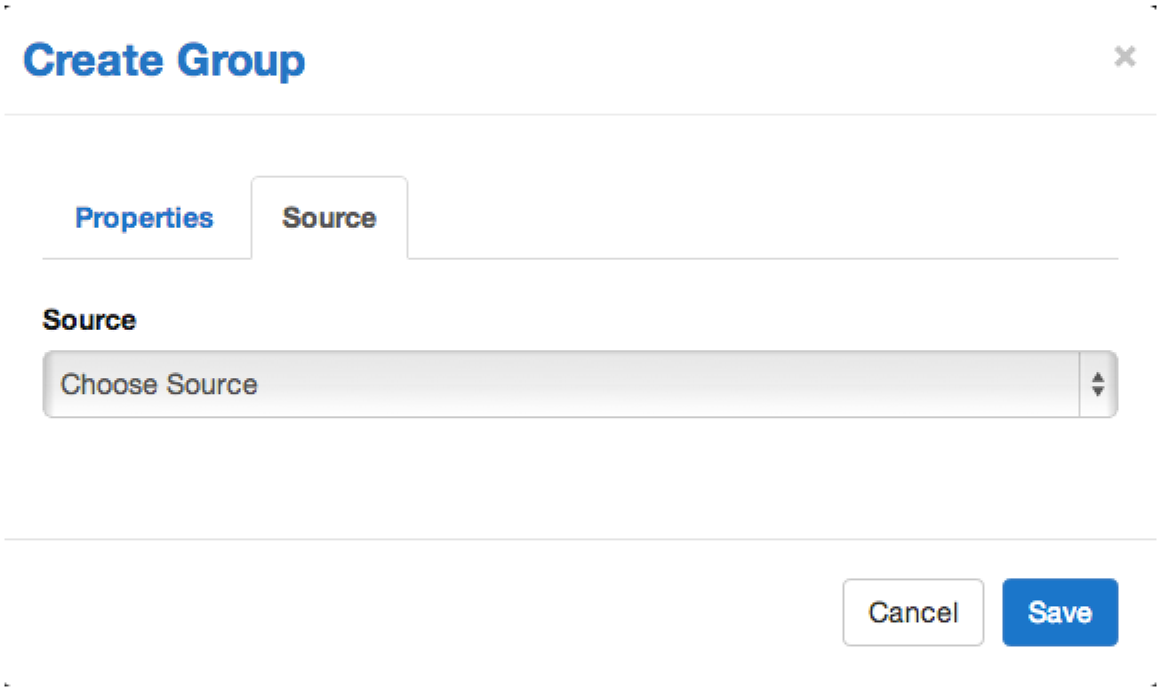
Enter the appropriate details into the following fields and click **Save**.

- Name (required)
- Description
- Variables

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

By default, the group **Source** is manual, which means that the hosts must be entered into Tower manually. (See Add a new host for more information on managing hosts individually.)

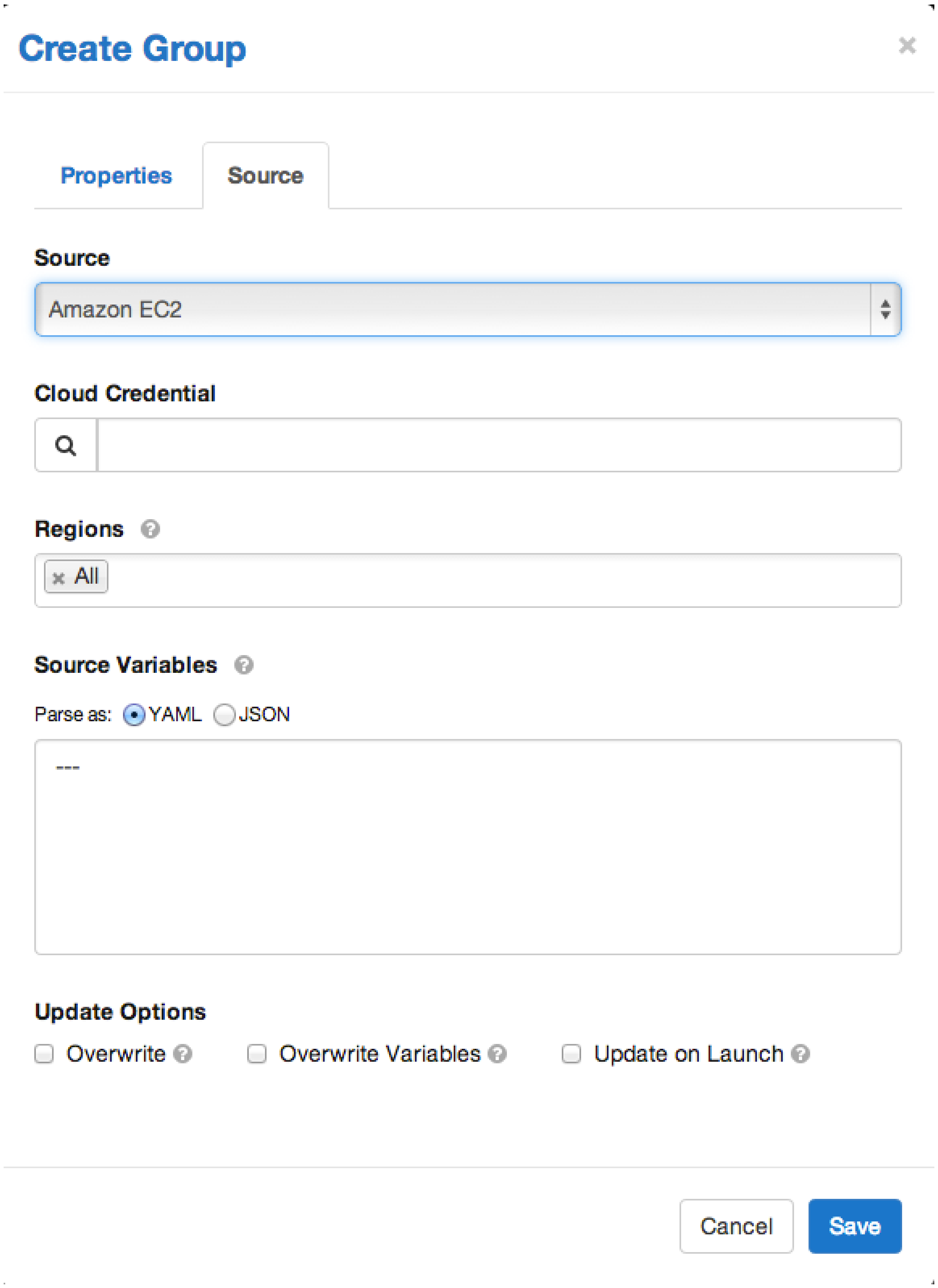
To synchronize the inventory group from a cloud source, select the **Source** tab and choose the appropriate source from the **Source** menu.



Tower 1.4 supports Amazon Web Services EC2 and Rackspace Cloud Servers.

Amazon Web Services EC2

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


Create Group

Properties | Source

Source: Amazon EC2

Cloud Credential:

Regions: All

Source Variables

Parse as: [✓ YAML] JSON

```
---
```

Update Options

☐ Overwrite ☐ Overwrite Variables ☐ Update on Launch

- Cloud Credential

Choose from an existing credential. For more information, see the Credentials section.

- 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.

- Source Variables

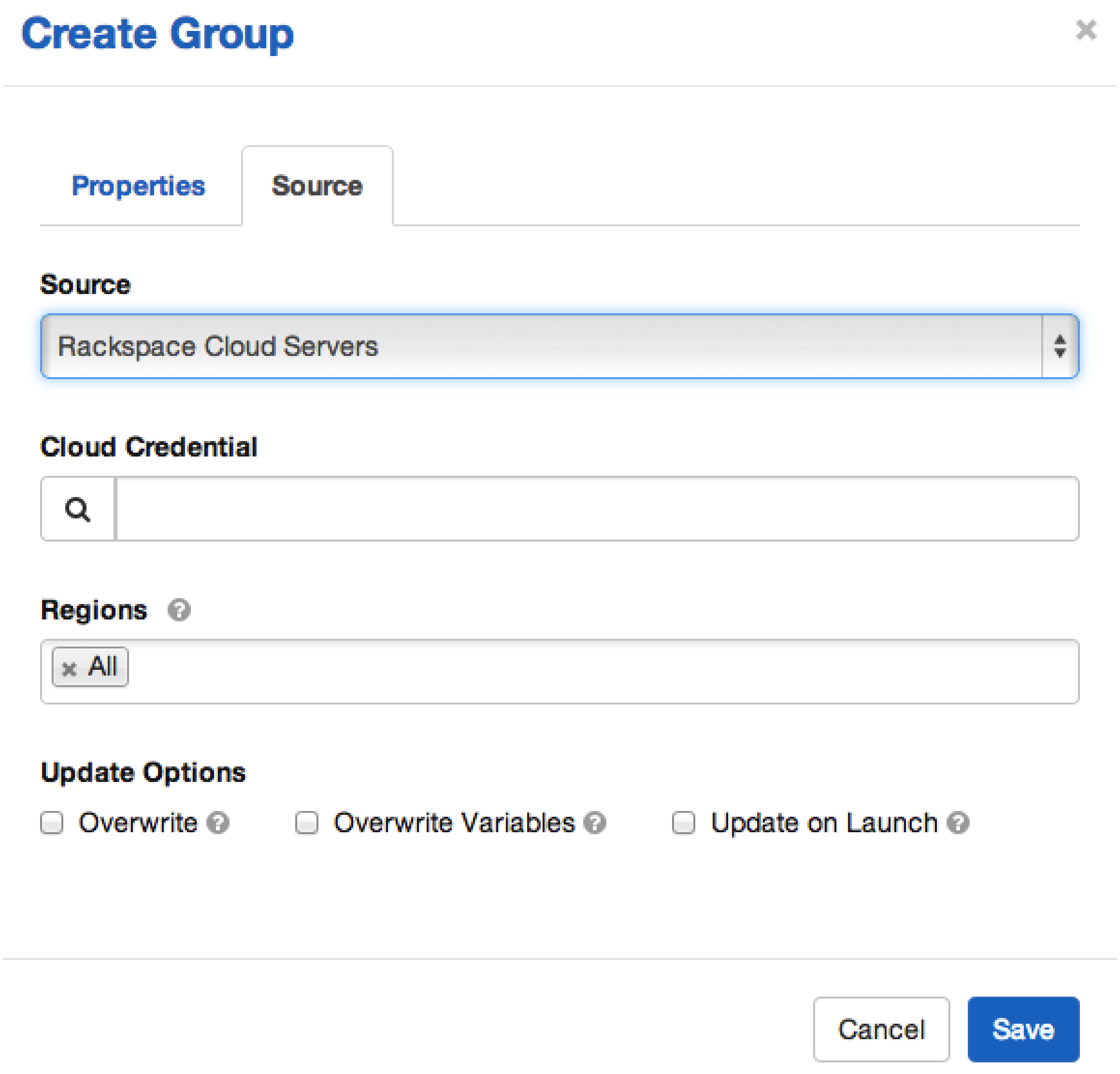
Override variables found in ec2.ini and used by the inventory update script. For a detailed description of these variables view ec2.ini in the Ansible Github repo (https://github.com/ansible/ansible/blob/devel/plugins/inventory/ec2.ini).

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

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

Rackspace Cloud Servers

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



Source: Rackspace Cloud Servers

Cloud Credential:

Regions: All

Update Options: [ ] Overwrite [ ] Overwrite Variables [ ] Update on Launch

- Cloud Credential

  Choose from an existing Credential. For more information, see the Credentials section.

- 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.

- Update Interval

  Instruct the Tower server to automatically run the inventory update process the selected number of minutes from the last run.

With a value set, task manager will periodically compare the amount of elapsed time from the last run. If enough time has elapsed, it will go ahead and start an inventory update process.

- Update Options

  - Overwrite

  When checked all child groups and hosts not found on the remote source will be deleted from the local inventory.

  Unchecked any local child hosts and groups not found on the external source will remain untouched by the inventory update process.

  - Overwrite Variables

  If checked, all variables for child groups and hosts will be removed and replaced by those found on the external source.

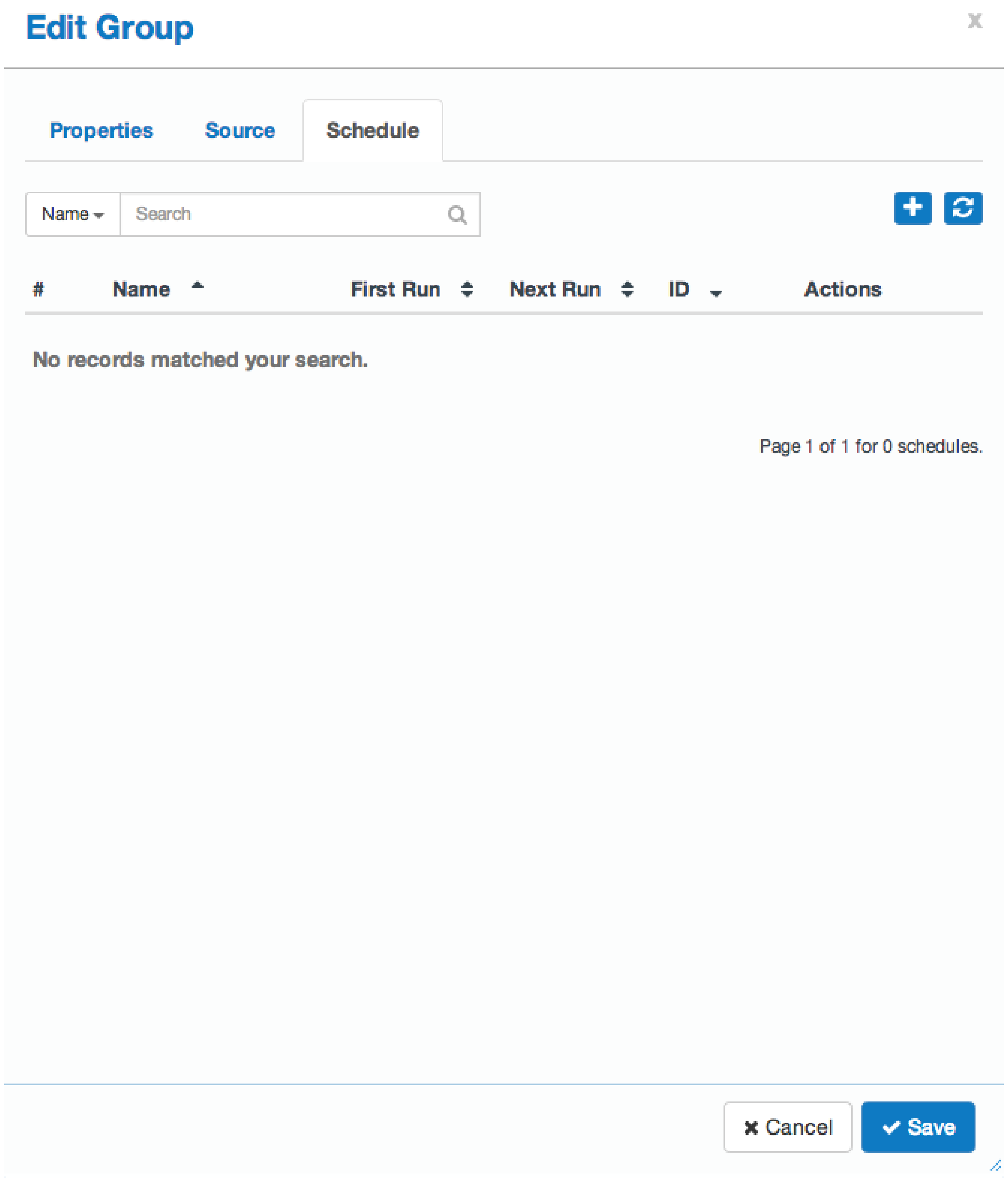
  When not checked a merge will be performed, combining local variables with those found on the external source.

  - Update on Launch

    Each time a job runs using this inventory, refresh the inventory from the selected source before executing job tasks.

Scheduling

For groups sourced from a cloud service, the inventory update process may be scheduled via the **Schedule** tab in the **Edit Group** dialog.



#	Name	First Run	Next Run	ID	Actions

No records matched your search.

Page 1 of 1 for 0 schedules.

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** or **ID**.

The list of schedules includes:

- Name - Clicking the schedule name will open the **Edit Schedule** dialog
- First Run
- Next Run
- Final Run

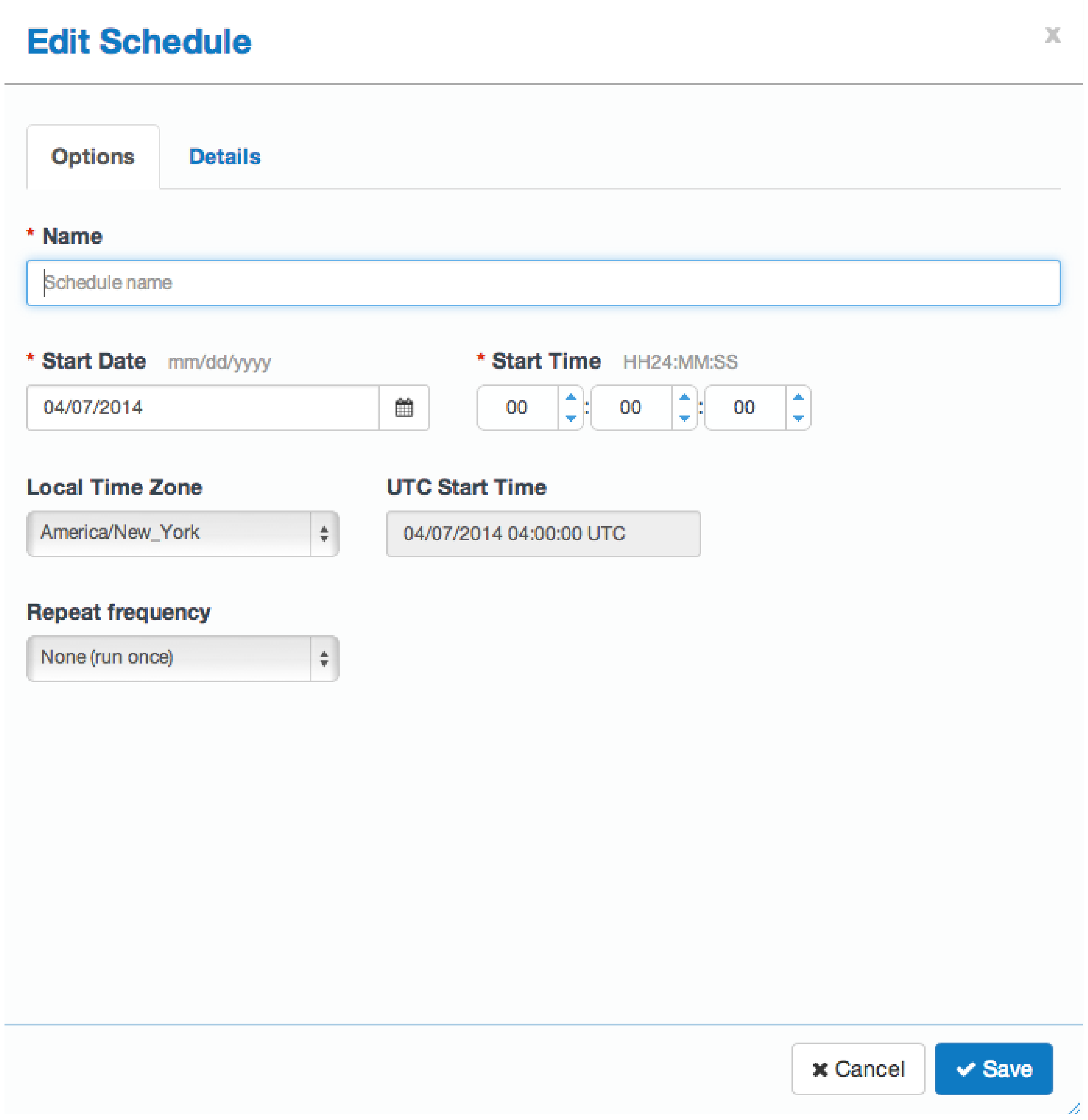
- ID - Clicking the schedule ID will open the **Edit Schedule** dialog

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.



Edit Schedule

Options | Details

* Name
Schedule name

* Start Date mm/dd/yyyy
04/07/2014

* Start Time HH24:MM:SS
00 : 00 : 00

Local Time Zone
America/New_York

UTC Start Time
04/07/2014 04:00:00 UTC

Repeat frequency
None (run once)

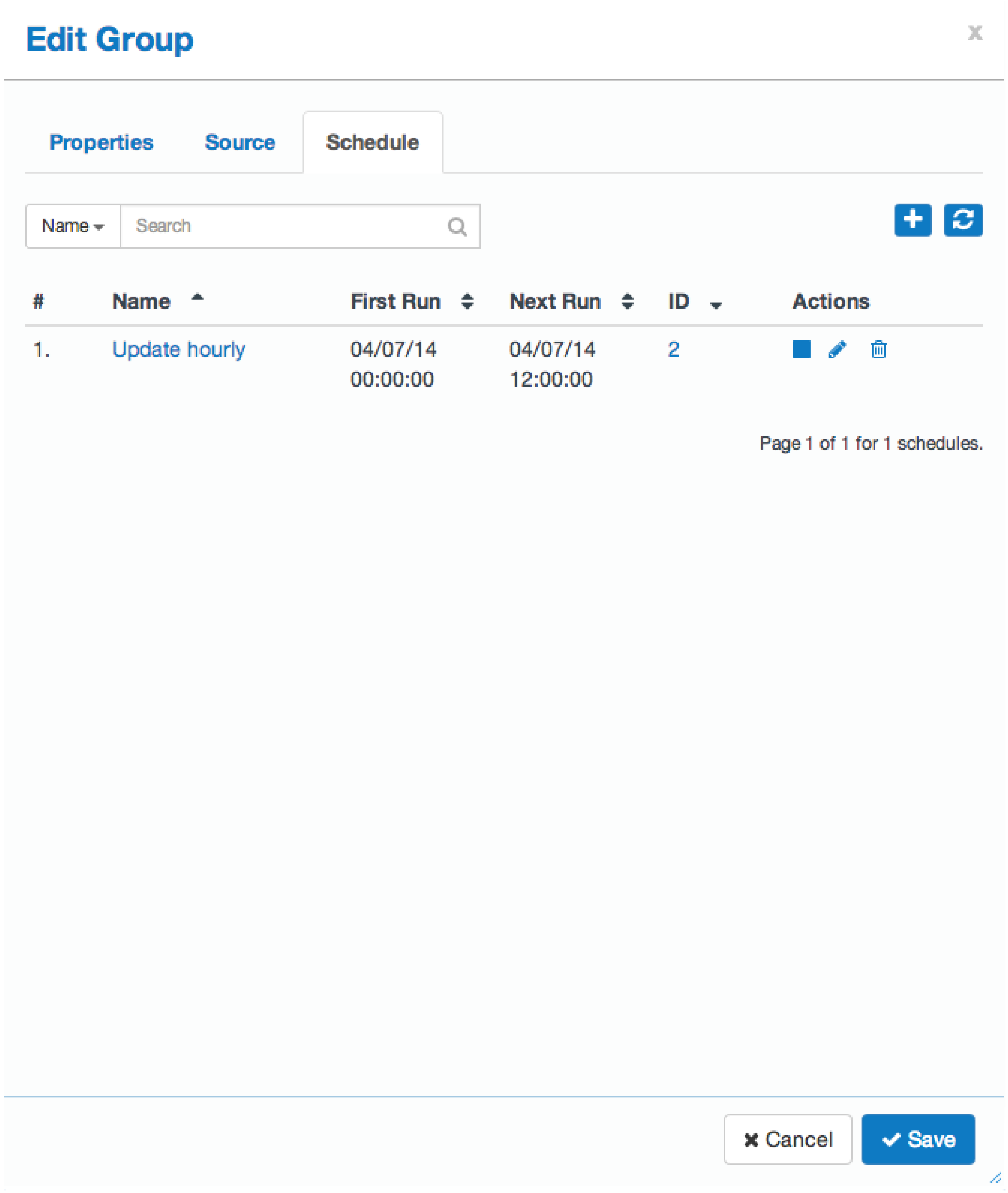
Enter the appropriate details into the following fields and select Save:

- Name (required)

- Start Date (required)
- Start Time (required)
- Local Time Zone
- UTC Start Time
- Repeat Frequency - the appropriate options will display as the update frequency is modified.

The **View Details** button at the bottom of the dialog will display a description of the schedule and a list of the scheduled occurences in Local time or UTC, as selected.

There are several actions available for schedules:



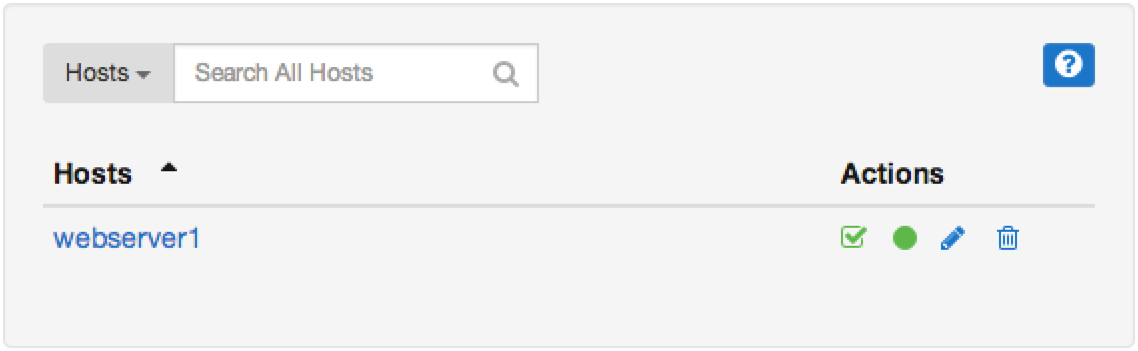
#	Name	First Run	Next Run	ID	Actions
1.	Update hourly	04/07/14 00:00:00	04/07/14 12:00:00	2	

Page 1 of 1 for 1 schedules.

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

Hosts

Hosts are listed on the right side of the Inventory 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 hosts synchronized with an external source.

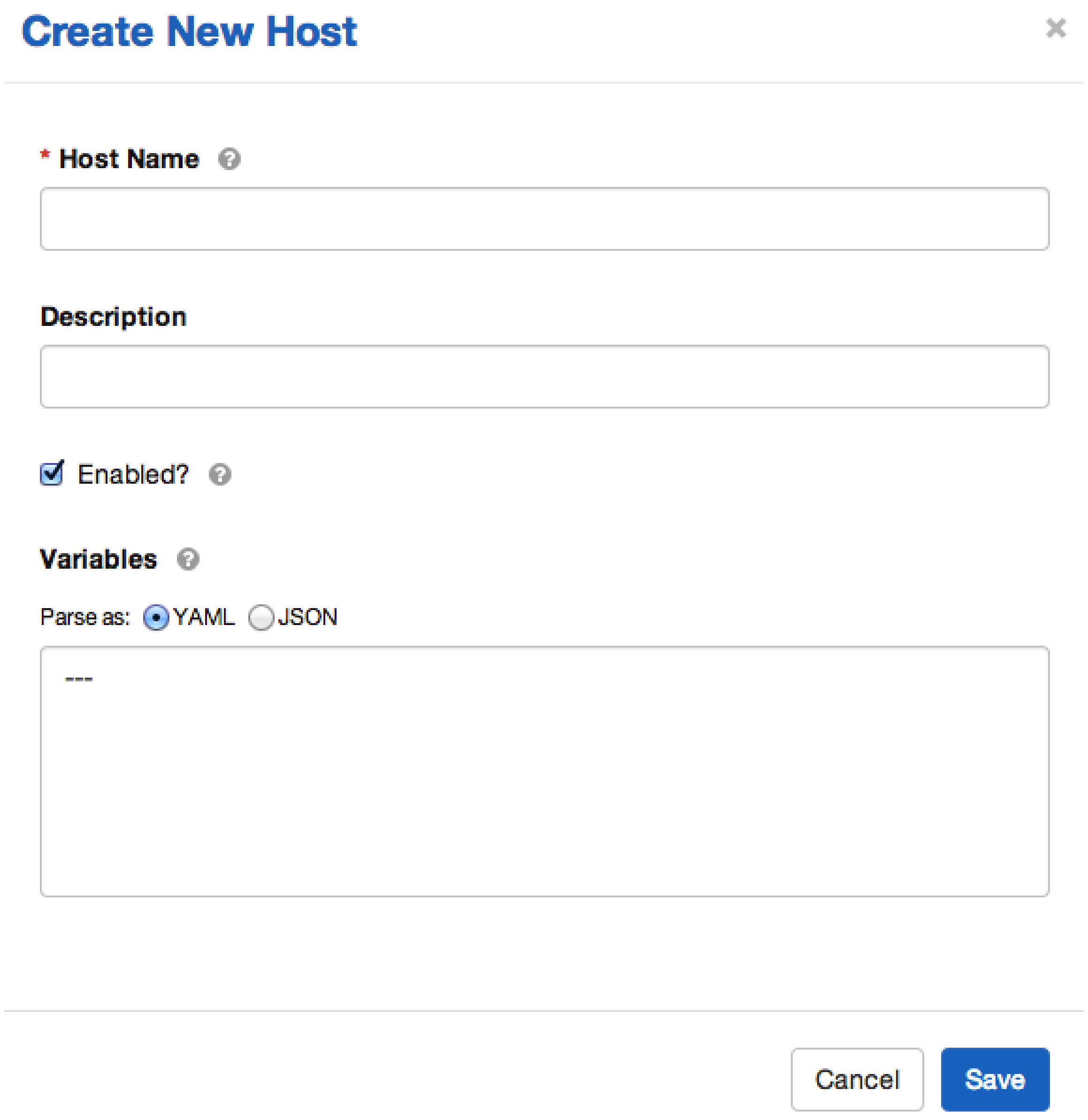
This list displays information about each host and provides for several actions:

- 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 will display a window showing the most recent jobs and their status.
- Edit host - Opens the **Host Properties** dialog
- Delete - Removed the host from Tower. *This operation is not reversible!*

Add a new host

To create a new host and add it to an existing group, click the ✚ button.

This will open to the **Create New Host** dialog.



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

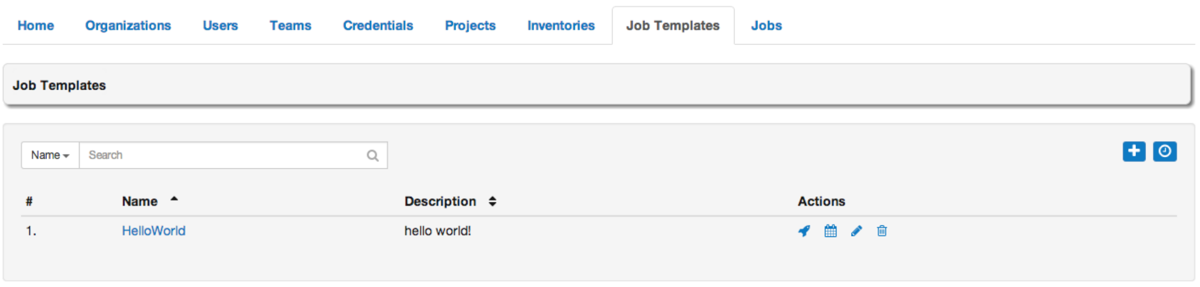
- Host Name - The hostname or IP address of the host
- Description
- Enabled? - Indicates if a host is available and should be included in running jobs. For hosts that are part of an external inventory, this flag cannot be changed. It will be set by the inventory sync process.
- Variables

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.

Job Templates

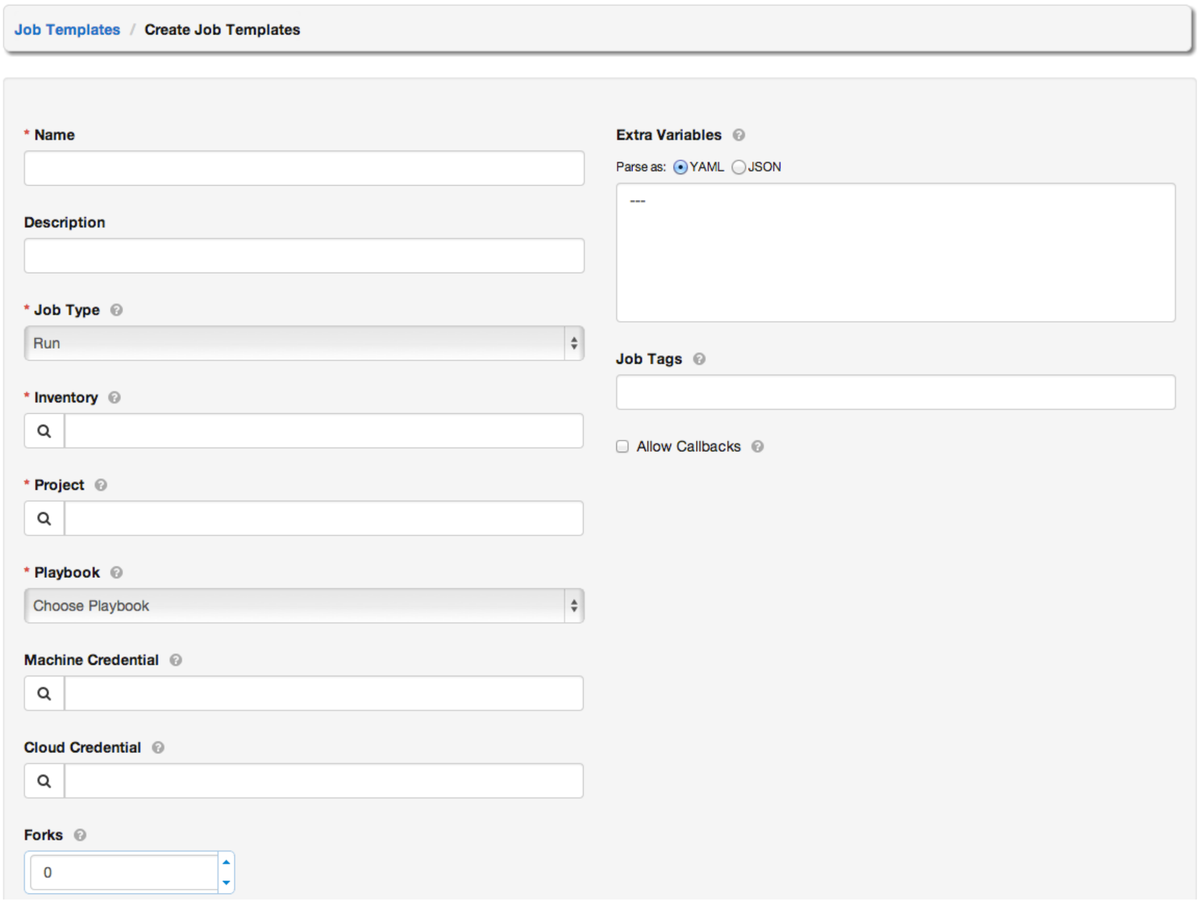
A job template is a definition and set of parameters for running an Ansible job. Job templates are useful to execute the same job many times. While the REST API allows executing jobs directly, Tower requires first creating a job template.

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 modify, launch, schedule, and remove a job template.



#	Name	Description	Actions
1.	HelloWorld	hello world!	

To create a new job template click the ➕ button.



* Name	
Description	
* Job Type	Run
* Inventory	
* Project	
* Playbook	Choose Playbook
Machine Credential	
Cloud Credential	
Forks	0
Extra Variables	Parse as: [✓ YAML] JSON; ---
Job Tags	
Allow Callbacks	☐

Enter the appropriate details into the following fields:

- Name (required)
- Description
- Job Type:
  - Run: Execute the playbook when launched, running Ansible tasks on the selected hosts
  - Check: Execute the playbook in dry-run mode, reporting "changed" when an item would be changed, but not actually making changes.

  More documentation on job types may be found in the Playbooks: Special Topics (http://docs.ansible.com/playbooks_special_topics.html) 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.
- 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 example, a playbook named "jboss.yml" in the project path will appear 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 will use 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 that will be 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 see Patterns (http://docs.ansible.com/intro_patterns.html) in the Ansible documentation.

- Verbosity: Control the level of output Ansible will produce as the playbook executes. Set the verbosity to any of Default, Verbose, or Debug. This only appears in the "details" report view. Verbose logging will include the output of all commands. Debug logging is exceedingly verbose and will include information on SSH operations that can be useful in certain support instances. Most users will 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 (http://docs.ansible.com/playbooks_variables.html#passing-variables-on-the-command-line). Provide key/value pairs using either YAML or JSON. These variables have a maximum value of precedence and will override other variables specified elsewhere. An example value might be:

```
---

git_branch: production
release_version: 1.5
```

- Job Tags: Provide a comma-separated list of playbook tags with which to filter this job template. More documentation on tags may be found in the Tags (http://docs.ansible.com/playbooks_tags.html) section of the Ansible documentation.
- 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.

  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. 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.

  Frequently this would be accessed via a firstboot type script, or from cron.

  To enable callbacks, check the Allow Callbacks checkbox. A unique host key will be displayed that corresponds to this job template. The host key may be reused across multiple hosts to apply this job template against multiple hosts.

The callback can be accessed via REST, though the suggested method of using the callback is to use an example script that ships with Tower.

To post manually via REST, look at the callback URL in the UI. The URL will look like the following:

`http://your.server.com:999/api/v1/job_templates/1/callback/`

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://your.server.com:999/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 will be denied. Note that if your host is not in inventory, Tower is smart enough to try to update cloud based inventory source before running the callback.

Successful requests will result in an entry on the Jobs tab, where the results and history can be viewed.

Should you wish to control what hosts are able to request configuration, the key may be changed at any time.

As we've mentioned, a better way to request configuration from a callback is to use the example script that ships in Tower (or a variant of it), located at /usr/share/awx/request_tower_configuration.sh. 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 will retry once per minute for up to ten minutes, which is amply conservative.

Most likely you will be using callbacks with dynamic inventory in Tower - such as pulling cloud inventory from AWS or Rackspace. In these cases, 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 will poll 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.

If attaching the request_tower_configuration script to cron, a suggested value for how often to run the configuration would be perhaps every 30 minutes, though 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 bake that into an autoscaling image.

Repeated configuration can be easily handled by scheduling in Tower, so the primary use of callbacks by most users will be to enable that a base image is bootstrapped into the latest configuration upon coming online.

This is a bit of an advanced feature, so if you have questions about ideal configuration of callbacks for autoscaling in your environment, feel free to reach out to support@ansible.com for some suggestions and advice.

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 will display a list of all of the jobs that have been launched from this template. Please see the section Jobs for more information about this interface.

Launching Jobs

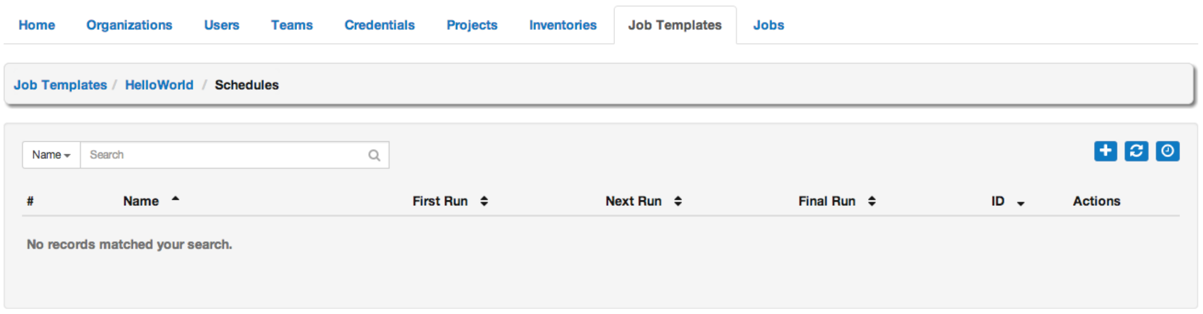
To launch a job template, click the 🚀 button.

If credentials require the user to enter additional information, such as a password or passphrase, a dialog will request this information.

Upon launch, Tower will automatically redirect the web browser to the **Jobs** tab.

Scheduling

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



#	Name	First Run	Next Run	Final Run	ID	Actions

No records matched your search.

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** or **ID**.

The list of schedules includes:

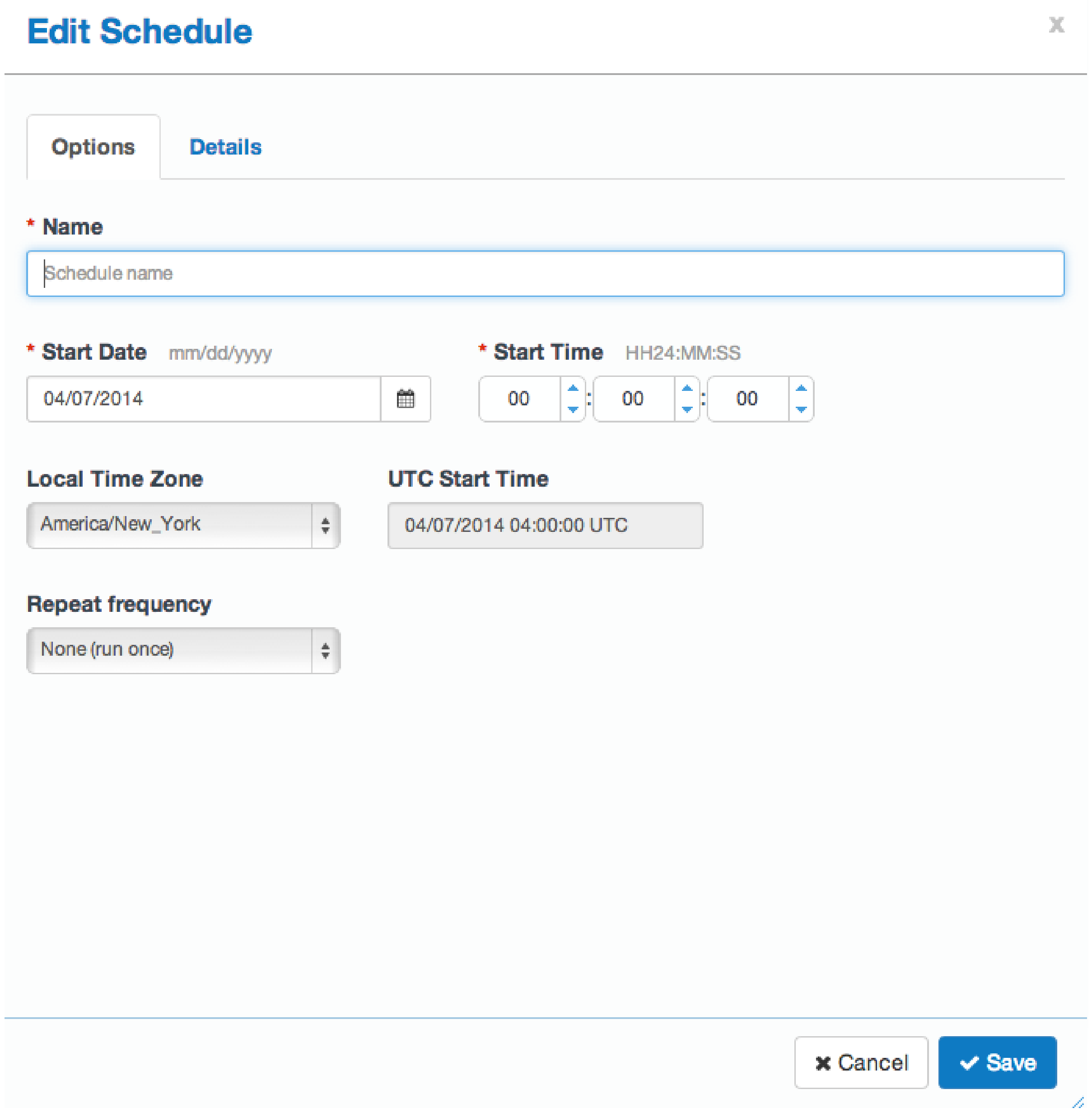
- Name - Clicking the schedule name will open the **Edit Schedule** dialog
- First Run
- Next Run
- Final Run
- ID - Clicking the schedule ID will open the **Edit Schedule** dialog

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.



Edit Schedule

Options | Details

* Name

Schedule name

* Start Date mm/dd/yyyy

04/07/2014

* Start Time HH24:MM:SS

00 : 00 : 00

Local Time Zone

America/New_York

UTC Start Time

04/07/2014 04:00:00 UTC

Repeat frequency

None (run once)

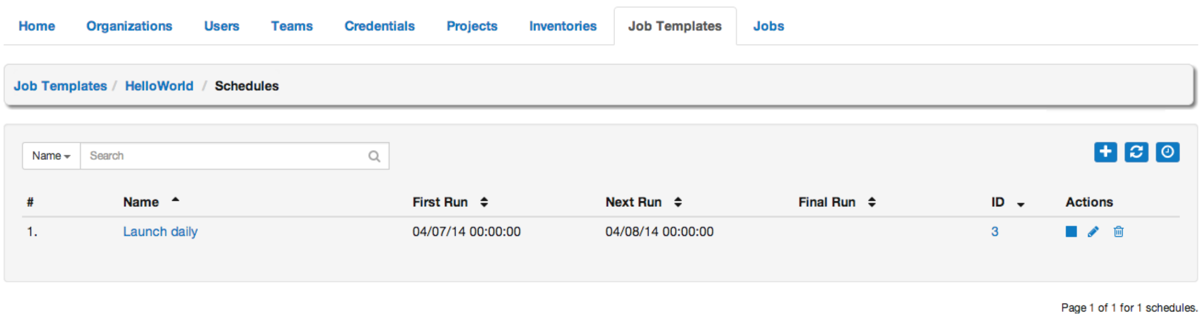
Enter the appropriate details into the following fields and select Save:

- Name (required)
- Start Date (required)
- Start Time (required)
- Local Time Zone
- UTC Start Time

- Repeat Frequency - the appropriate options will display as the update frequency is modified.

The **View Details** button at the bottom of the dialog will display a description of the schedule and a list of the scheduled occurences in Local time or UTC, as selected.

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



#	Name	First Run	Next Run	Final Run	ID	Actions
1.	Launch daily	04/07/14 00:00:00	04/08/14 00:00:00		3	

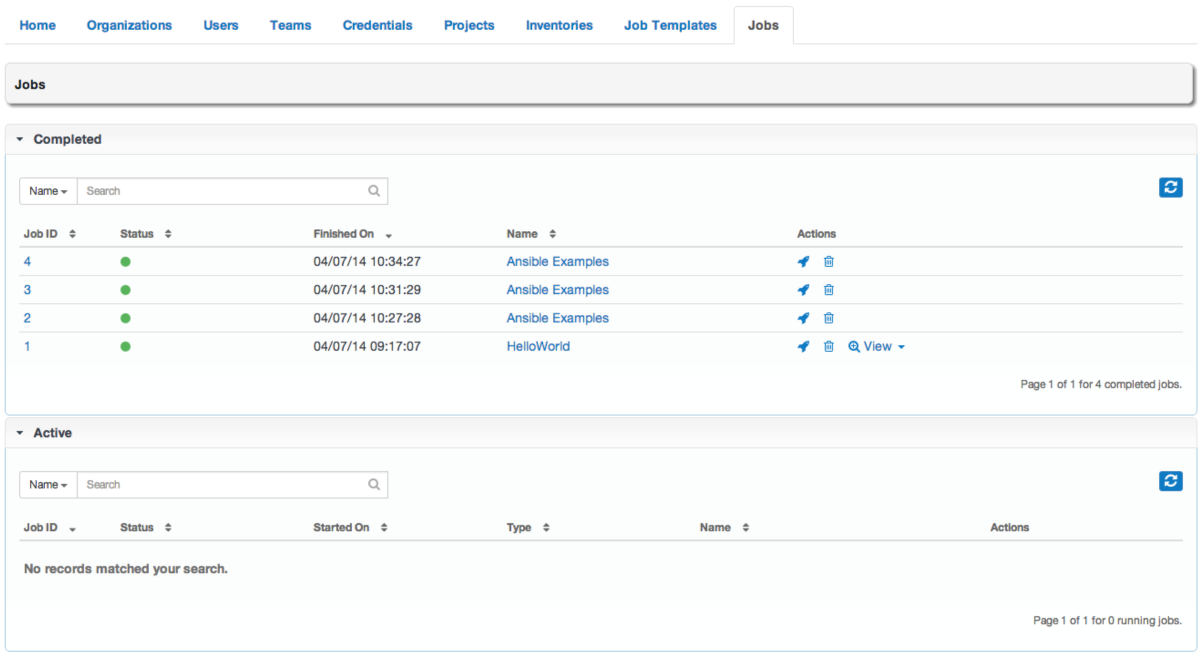
Page 1 of 1 for 1 schedules.

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

Jobs

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

The Jobs tab displays a list of jobs, including jobs that are completed, active, queued, and scheduled.



Jobs

Completed

Job ID	Status	Finished On	Name	Actions
4	●	04/07/14 10:34:27	Ansible Examples	
3	●	04/07/14 10:31:29	Ansible Examples	
2	●	04/07/14 10:27:28	Ansible Examples	
1	●	04/07/14 09:17:07	HelloWorld	View

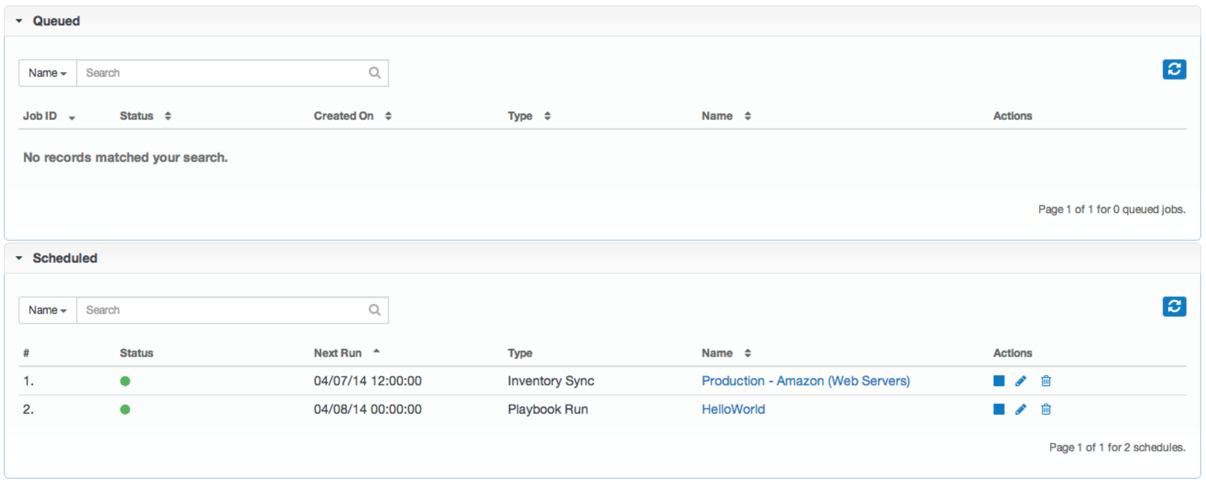
Page 1 of 1 for 4 completed jobs.

Active

Job ID	Status	Started On	Type	Name	Actions

No records matched your search.

Page 1 of 1 for 0 running jobs.



Queued

Job ID	Status	Created On	Type	Name	Actions

No records matched your search.

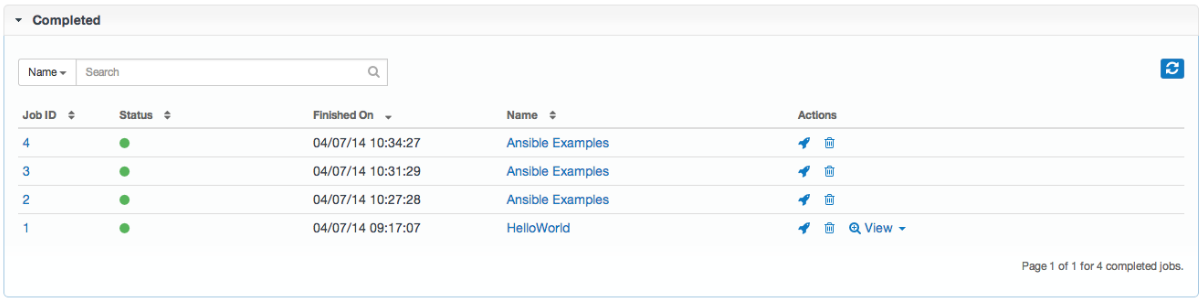
Page 1 of 1 for 0 queued jobs.

Scheduled

#	Status	Next Run	Type	Name	Actions
1.	●	04/07/14 12:00:00	Inventory Sync	Production - Amazon (Web Servers)	
2.	●	04/08/14 00:00:00	Playbook Run	HelloWorld	

Page 1 of 1 for 2 schedules.

The list of **Completed** jobs may be searched by **Job ID** or **Name**, and filtered by **Job failed?**, **Status**, or **Type**.



Completed

Job ID	Status	Finished On	Name	Actions
4	●	04/07/14 10:34:27	Ansible Examples	
3	●	04/07/14 10:31:29	Ansible Examples	
2	●	04/07/14 10:27:28	Ansible Examples	
1	●	04/07/14 09:17:07	HelloWorld	View

Page 1 of 1 for 4 completed jobs.

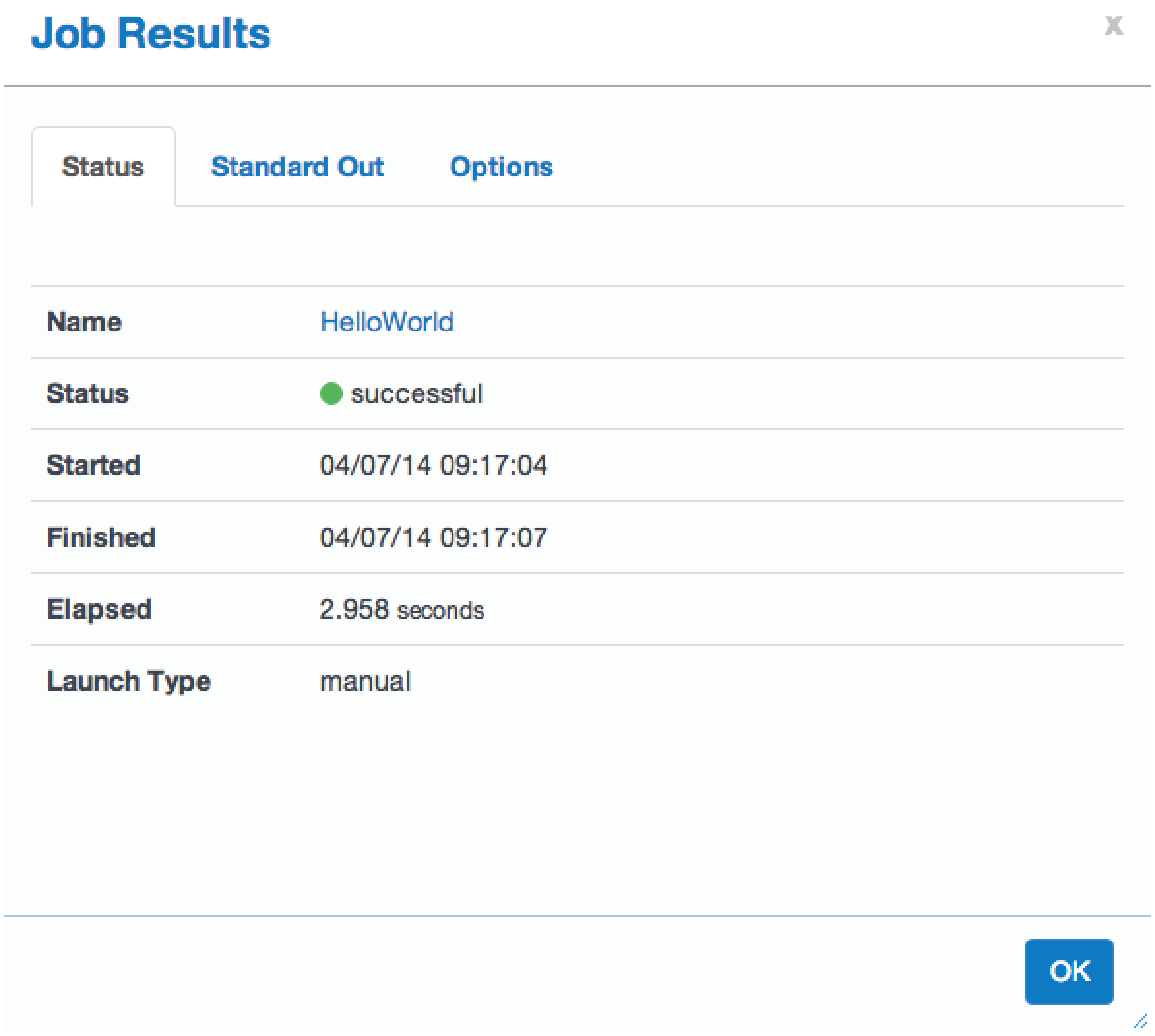
- **Job ID**: A unique integer that identifies a specific job.
- **Status**: Can be any of pending, running, successful, or failed.

- **Finished On**
- **Name**: For a job of type **SCM Update**, clicking the name will open the properties page for the appropriate project. For a job of type **Inventory Sync**, clicking the name will open the the associated inventory. For a job of type **Playbook Run**, clicking the name will open the **Job Results** window. See the section Job Results below.
- **Actions**: Depending on the job type, there are several actions available for each job:
  - **Relaunch**: Launch this job again, with the same parameters as it originally ran with. (Any modification to the job template after the job was launched will not be used during this re-launch.)
  - **Delete**: This button deletes the job from Tower.
  - **View**: For jobs of type **Playbook Run**, this menu opens the **Events** and **Host Summary** pages, as detailed below.

The Jobs tab does not automatically refresh to show the current status of any pending jobs. To refresh the job queue click the Refresh button.

Job Results

The **Job Results** window displays information about jobs of type **Playbook Run**.



Name	HelloWorld
Status	● successful
Started	04/07/14 09:17:04
Finished	04/07/14 09:17:07
Elapsed	2.958 seconds
Launch Type	manual

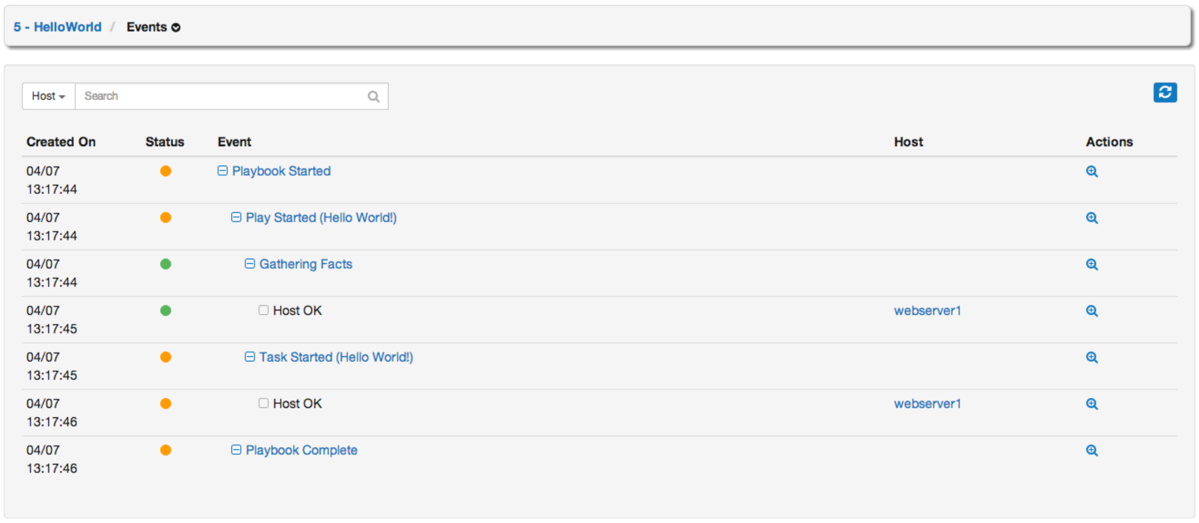
These details will include:

- **Name**: The name of the job template from which this job was launched.
- **Job Status**: can be any of pending, running, successful, or failed
- **Started**: the timestamp of when the job was initiated by Tower
- **Finished**: the timestamp of when the job was completed
- **Elapsed**
- **Launch Type**: manual or scheduled

- **Standard Out**: This tab displays the results of running the playbook from the standard out of the Tower server. This shows the same information you would see if you ran the Ansible playbook using Ansible from the command line.
- **Options**: The details of the job template that the job was launched with.

Events

Selecting **Events** will display details about each of the Tower events that comprised the job, including specific details on each playbook task.

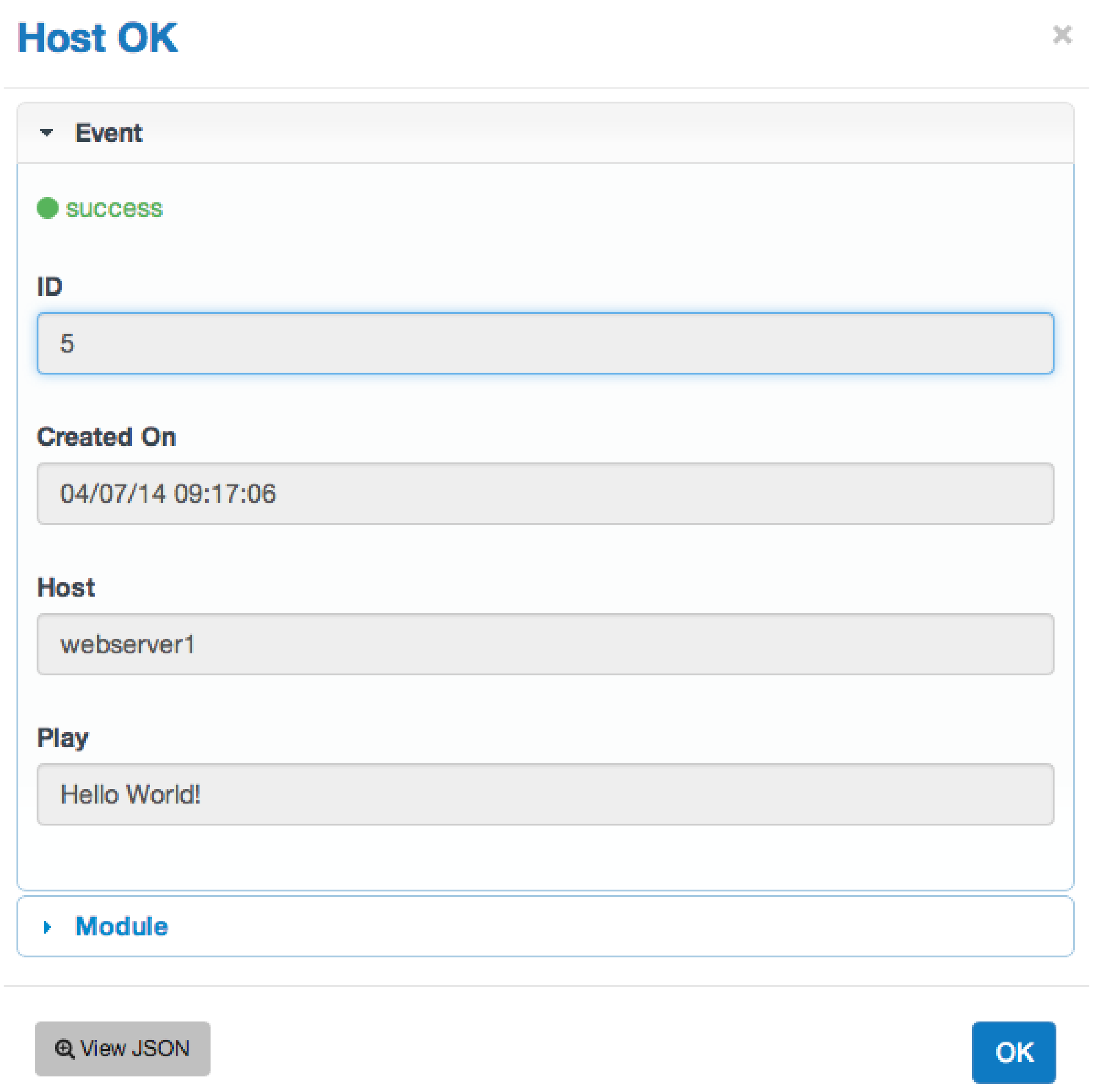


5 - HelloWorld / Events

Created On	Status	Event	Host	Actions
04/07 13:17:44	●	⊟ Playbook Started		
04/07 13:17:44	●	⊟ Play Started (Hello World!)		
04/07 13:17:44	●	⊟ Gathering Facts		
04/07 13:17:45	●	☐ Host OK	webserver1	
04/07 13:17:45	●	⊟ Task Started (Hello World!)		
04/07 13:17:46	●	☐ Host OK	webserver1	
04/07 13:17:46	●	⊟ Playbook Complete		

This information includes:

- **Created On**: the timestamp of when this job was initiated.
- **Status**: Can be any of changed, success, skipped, or failed.
- **Event**: the name of the event, including the playbook task name, if appropriate. Clicking the event name expands and collapses the tree of events and sub-events.
- **Host**: the host targeted by the event.
- **Actions**: Clicking the **Actions** button on any event will display additional information about the event.


Host OK

Event

success

ID

5

Created On

04/07/14 09:17:06

Host

webserver1

Play

Hello World!

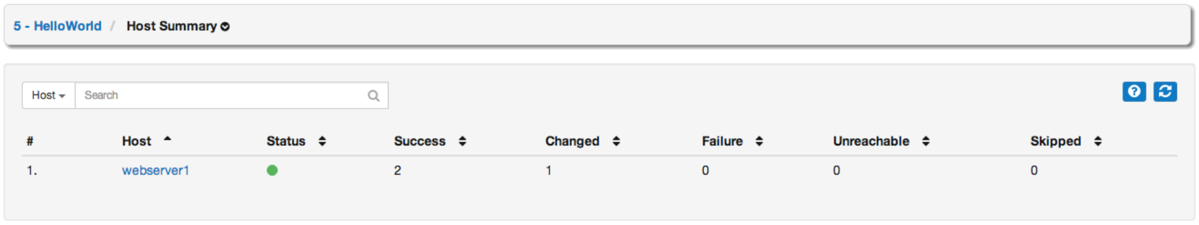
Module

This information includes:

- **Status**: can be any of changed, success, skipped, or failed
- **ID**: the Job ID
- **Created On**: the timestamp of when this job was initiated
- **Host**: the host targeted by the event
- **Play**: The name of the play
- **Module**: the name of the module invoked by the play

Host Summary

Selecting **Host Summary** will display a list of all of the hosts that this job was launched against, as well as useful summary information about each Ansible playbook task that was executed.


5 - HelloWorld / Host Summary

#	Host	Status	Success	Changed	Failure	Unreachable	Skipped
1.	webserver1	●	2	1	0	0	0

This information includes:

- **Host**: the host that the playbook was launched against. The host **Name** is a hot-link to the inventory containing this host.
- **Status**

The summary also display the number of playbook tasks that completed with an outcome of:

- **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.

Best Practices

Ansible file and directory structure

Please review the Ansible best practices from the Ansible documentation at http://docs.ansible.com/playbooks_best_practices.html.

Playbooks should not use the `vars_prompt` feature, as Tower does not interactively allow Q\&A for `vars_prompt` questions at this time.

Inventory Management

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 the "awx-manage inventory_import" command on an inventory source, it can sync such variables with the database.

Scaling

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

Consider setting "forks" on a job template to larger values to increase parallelism of execution runs.

CI integration / Continuous Deployment

For a Continuous Integration system, such as Jenkins, to spawn an Tower job, it should make a curl request to a job template. The credentials to the job template should not require prompting for any particular passwords. Using the API to spawn jobs is covered in the API section.

Security

The multi-tenancy features of Tower are sufficient to control who can run certain projects on what systems, but are not intended to hide project content from other teams. For instance, you could easily control that engineering could not push to production.

All playbooks are executed via the "awx" filesystem user. Users who have access to edit playbooks need to be trusted as playbooks do have access to the filesystem and all that implies.

Users concerned about credential security may choose to upload locked SSH keys and set the unlock password to "ask", or choose to have the system prompt them for SSH credentials or sudo passwords rather than having the system store them in the database.

Troubleshooting

Tower server errors are logged to syslog. Apache web server errors are logged to the httpd error log.

Client-side issues may be explored using the JavaScript console built into most browsers and should be reported to support@ansible.com.

If requested by support, super users may also edit the "Django Admin" browser at `http://<Tower server name>/admin` using a super user login. Do not do this unless requested by Ansible support as you may remove objects and bypass the business logic of the application. After logging in to the admin view, users may need to clear their cookies to successfully log back into the main application.

Glossary

Organization: A logical collection of Users, Teams, Projects, and Inventories. The highest level in the Tower object hierarchy. See this description of the Tower hierarchy.

User: An Tower operator with associated permissions and credentials.

Organization Administrator: An Tower user with the rights to modify the Organization's membership and settings, including making new users and projects within that organization. An organization admin can also grant permissions to other users within the organization.

Team: A sub-division of an Organization with associated Users, Projects, Credentials, and Permissions. Teams provide a means to implement role-based access control schemes and delegate responsibilities across Organizations.

Project: A logical collection of Ansible playbooks, represented in Tower.

Inventory: A collection of hosts against which Jobs may be launched.

Credentials: Authentication details that may be utilized by Tower to launch jobs against machines, to synchronize with inventory sources, and to import project content from a version control system.

Job Template: The combination of an Ansible playbook and the set of parameters required to launch it, designed to be reusable across hosts.

Job: The instantiation of a Job Template; the launch of an Ansible playbook.

Permissions: The set of privileges assigned to Users and Teams that provide the ability to read, modify, and administer Projects, Inventories, and other Tower objects.

Host: A system managed by Tower, which may include a physical, virtual, or cloud-based server, a network router, switch, or firewall, a storage device, or any unique system managed by Tower. Typically an operating system instance.

Playbook: An Ansible playbook.

Super User: An admin of the Tower server who has permission to edit any object in the system, whether associated to any organization. Super users can create organizations and other super users.

Using LDAP with Tower

As of the 1.3 release of Tower, administrators may utilize LDAP as a source for authentication information for Tower users. At this time, only user authentication is provided and not synchronization of user permissions, credentials, or team membership, however organization membership (and who is an organization admin) can be synchronized.

When so configured, a user who logs in with an LDAP username and password will automatically get an Tower account created for them and they can be automatically placed into multiple organizations as either regular users or organization administrators.

By default, if users are created via an LDAP login, by default they cannot change their username, firstname, lastname, or set a local password for themselves. This is also tunable to restrict editing of other field names.

Currently, LDAP integration for Tower is configured in the file "/etc/awx/settings.py." No configuration is accessible via the Tower user interface. Please, review the comments in that file for information on LDAP configuration and let us know at support@ansible.com if you need assistance.

Administration of Tower

Certain command line commands are available for management of Tower. In the future, some of these may be made available via GUI tools as well, and they may be augmented with other commands. Here is a useful subset. Do not run other awx-manage commands unless instructed by Ansible Support.

***NOTE:** These commands should be run as the 'awx' user.*

`awx-manage inventory_import [--help]`

The inventory_import command is used to synchronize an Tower inventory object with a text-based inventory file, dynamic inventory script, or a directory of one or more of the above as supported by core Ansible.

When running this command, specify either an `--inventory-id` or `--inventory-name`, and the path to the Ansible inventory source is given by `--source`.

By default, inventory data already stored in Tower will be blended with data from the external source. To use only the external data, specify `--overwrite`. To specify that any existing hosts get variable data exclusively from the `--source`, specify `--overwrite-vars`. The default behavior will add any new variables from the external source, overwriting keys that do not already exist, but preserving any variables that were not sourced from the external data source.

API

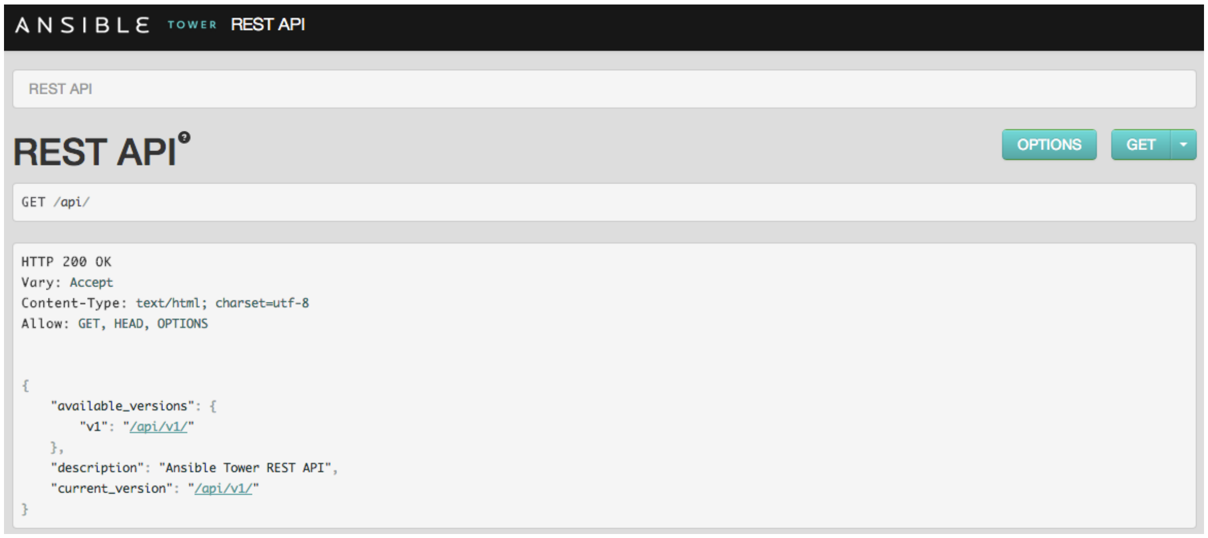
Firebug, Chrome, and Charles Proxy

This document gives a basic understanding of the API, though you may wish to see what API calls Tower makes in sequence. To do this, using the UI from Firebug or Chrome with developer plugins is useful, though Charles Proxy (http://www.charlesproxy.com/) is also an outstanding visualizer that you may wish to investigate. It is commercial software but can insert itself as, for instance, an OS X proxy and intercept both requests from web browsers but also curl and other API consumers.

Browseable API

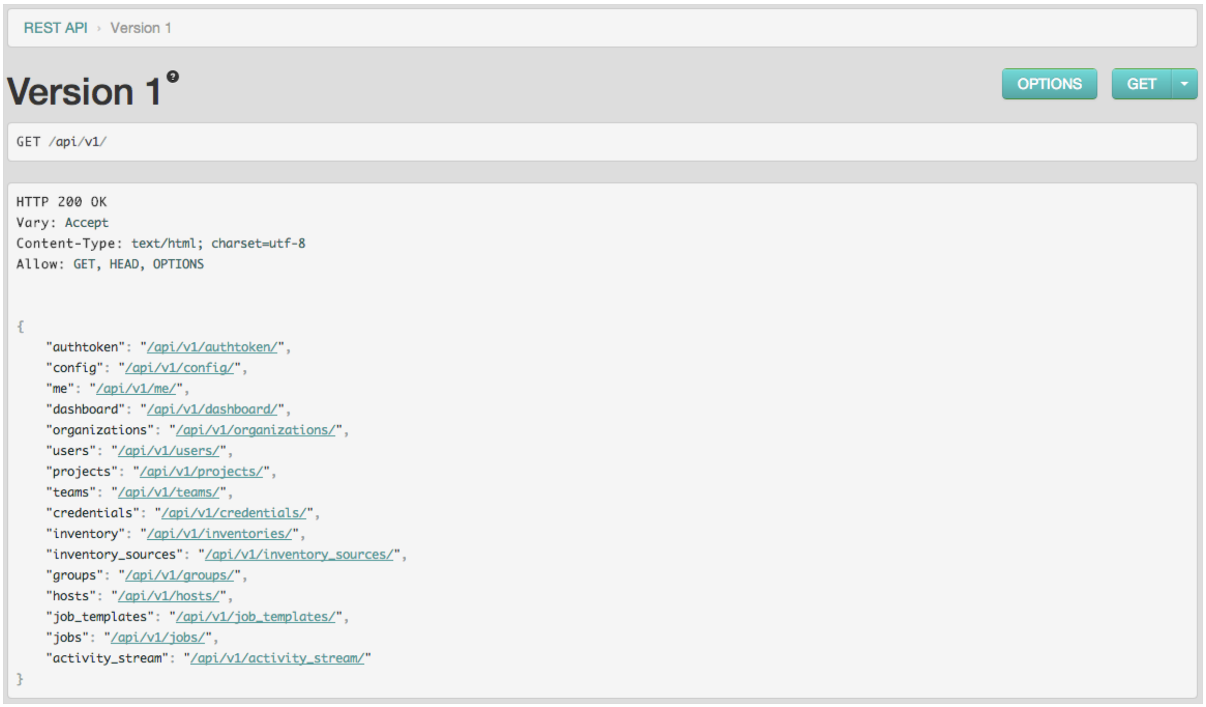
Tower features a browseable API feature.

You can visit the API in a browser at `http://<Tower server name>/api` and then click on various links in the API to explore related resources.



```
GET /api/
```

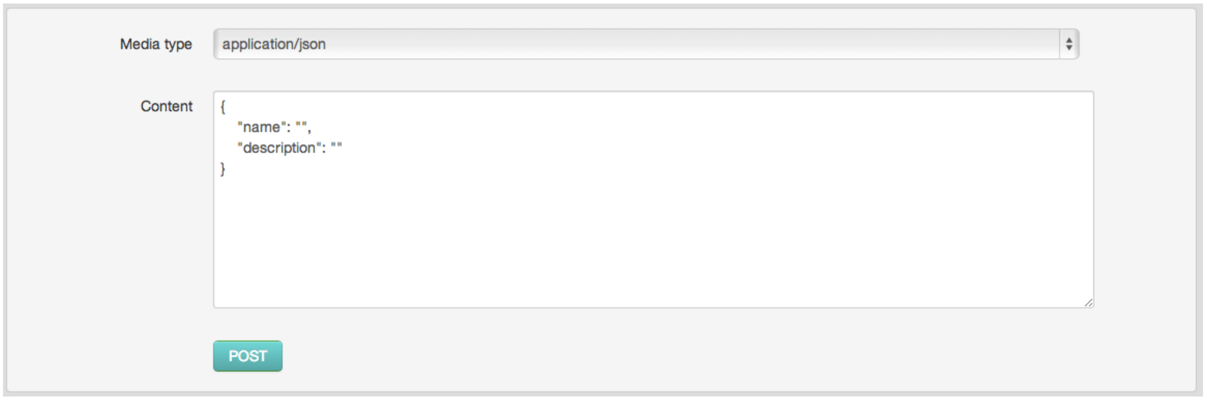
```
HTTP 200 OK
Vary: Accept
Content-Type: text/html; charset=utf-8
Allow: GET, HEAD, OPTIONS


{
    "available_versions": {
        "v1": "/api/v1/"
    },
    "description": "Ansible Tower REST API",
    "current_version": "/api/v1/"
}
```



```
GET /api/v1/
```

```
HTTP 200 OK
Vary: Accept
Content-Type: text/html; charset=utf-8
Allow: GET, HEAD, OPTIONS


{
    "authtoken": "/api/v1/authtoken/",
    "config": "/api/v1/config/",
    "me": "/api/v1/me/",
    "dashboard": "/api/v1/dashboard/",
    "organizations": "/api/v1/organizations/",
    "users": "/api/v1/users/",
    "projects": "/api/v1/projects/",
    "teams": "/api/v1/teams/",
    "credentials": "/api/v1/credentials/",
    "inventory": "/api/v1/inventories/",
    "inventory_sources": "/api/v1/inventory_sources/",
    "groups": "/api/v1/groups/",
    "hosts": "/api/v1/hosts/",
    "job_templates": "/api/v1/job_templates/",
    "jobs": "/api/v1/jobs/",
    "activity_stream": "/api/v1/activity_stream/"
}
```

You can also PUT and POST on the specific API pages if you so desire by formatting JSON in the various text fields.



Media type: application/json

Content:

```
{
    "name": "",
    "description": ""
}
```

Conventions

With all of the basics about how to explore the API and database objects out of the way, it's now time for some general API info.

Tower uses a standard REST API, rooted at /api/ on the server. The API is versioned for compatibility reasons but only /api/v1/ is presently available. By querying /api you can see information about what API versions are available.

All data is JSON by default. You may have to specify the content/type on POST or PUT requests accordingly.

All URIs should end in "/" or you will get a 301 redirect.

Sorting

Assume the following URL, `http://<Tower server name>/api/v1/groups/`

In order to sort the groups by name, access the following URL variation:

`http://<Tower server name>/api/v1/groups/?order_by=name`

You can order by any field in the object.

Filtering

Any collection is what the system calls a "queryset" and can be filtered via various operators.

For example, to find the groups that contain the name "foo":

`http://<Tower server name>/api/v1/groups/?name__contains=foo`

To do an exact match:

`http://<Tower server name>/api/v1/groups/?name=foo`

If a resource is of an integer type, you must add "__int" to the end to cast your string input value to an integer, like so:

`http://<Tower server name>/api/v1/arbitrary_resource/?x__int=5`

Related resources can also be queried, like so:

`http://<Tower server name>/api/v1/groups/?user__firstname__icontains=john`

This will return all groups with users with names that include the string "John" in them.

You can also filter against more than one field at once:

`http://<Tower server name>/api/v1/groups/?user__firstname__icontains=john&group__name__icontains__foo`

This will find all groups containing a user whose name contains John where the group contains the string foo.

For more about what types of operators are available, see:

https://docs.djangoproject.com/en/dev/ref/models/querysets/

You may also wish to watch the API as the UI is being used to see how it is filtering on various criteria.

Pagination

Responses for collections in the API are paginated. This means that while a collection may contain tens or hundreds of thousands of objects, in each web request, only a limited number of results are returned for API performance reasons.

When you get back the result for a collection you will see something like:

```
{'count': 25, 'next': 'http://testserver/api/v1/some_resource?page=2', 'previous': None, 'results': [ ... ] }
```

Where to get the next page, simply request the page given by the 'next' URL.

To request more items per page, pass the page size query string:

```
http://<Tower server name>/api/v1/some_resource?page_size=50
```

The serializer is quite efficient, but you should probably not request page sizes beyond a couple of hundred.

The user interface uses smaller values to avoid the user having to do a lot of scrolling.

Read Only Fields

Certain fields in the REST API are marked read only. These usually include the URL of a resource, the ID, and occasionally some internal fields. For instance, the 'created_by' attribute of each object indicates which user created the resource, and cannot be edited.

If you post some values and notice they are not changing, these fields may be read only.

API Example of Triggering A Job

For an example of how to launch an existing job template from a script, for integration with a Continuous Integration system or other program, please review the awx-cli project at https://github.com/ansible/awx-cli.

The awx-cli program is a command line utility used to send commands to the Tower API.

External Changes