ANSIBLE

TOWER

Ansible Tower User Guide

Version 2.1

January 7, 2015

http://support.ansible.com/

Table of Contents

Overview

Tower

Ansible Tower is a web 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.

Realtime Playbook Output and Exploration

Watch playbooks run in real time, seeing each host as they check in. Easily go back and explore the results for specific tasks and hosts in great detail. Search for specific plays or hosts and see just those results, or quickly zero in on errors that need to be corrected.

"Push Button" Automation

Access your favorite projects and re-trigger execution from the web interface with a minimum of clicking. Tower will ask for input variables, prompt for your credentials, kick off and monitor the job, and display results and host history over time.

Role Based Access Control and Auditing

Ansible Tower allows delegating specific authority to different teams or explicit users. Keep some projects private. Allow some users to edit inventory and others to run playbooks against only certain systems - either in check (dry run) or live mode. Allow certain users to use credentials without exposing the credentials to them. Regardless of what you do, tower records the history of operations and who made them - including objects edited and jobs launched.

Cloud & Autoscaling Flexibility

Tower features a powerful provisioning callback feature that allows nodes to request configuration on demand. While optional, this is an ideal solution for a cloud auto-scaling scenario, integrating with provisioning servers like Cobbler, or when dealing with managed systems with unpredictable uptimes. Requiring no management software to be installed on remote nodes, the callback solution can be triggered via a simple call to 'curl' or 'wget', and is easily embeddable in init scripts, kickstarts, or preseeds. Access is controlled such that only machines in inventory can request configuration.

The Ideal RESTful API

The Tower REST API is the ideal RESTful API for a systems management application, with all resources fully discoverable, paginated, searchable, and well modeled. A styled API browser allows API exploration from the API root at `http://<Tower server name>/api/`, showing off every resource and relation. Everything that can be done in the user interface can be done in the API - and more.

Licensing

Tower is a proprietary software product and is licensed on an annual subscription basis. While Tower does require a license to run, there is no fee for managing up to 10 hosts. Additionally, trial licenses are available for exploring Tower with a larger number of hosts.

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 http://support.ansible.com/ for assistance.
Trial licenses are available at http://ansible.com/license.
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:

- Basic (Web-Only), Enterprise (8x5), or Premium (24x7) Support, available via web, email, and telephone with SLA
- All regular updates and releases of Tower and Ansible

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

Release Notes

- Version 2.1
  - New simplified Portal Mode view for users, access at `https://<Tower server name>/portal/`
  - New surveys on job templates allow easy prompting of users for job parameters
  - Tower can now use an external PostgreSQL instance as the Tower database, including Amazon's RDS
  - Added support for active/passive High Availability Tower deployments
  - Custom dynamic inventory scripts can be pasted in using the admin user menu
  - Limit Amazon EC2 inventory imports into Tower based on tags, keys, and more
  - Tower data cleanup jobs can now be scheduled and run directly from the Tower interface versus logging into the Tower instance
  - The /etc/awx Tower configuration directory has moved to /etc/tower
  - Many assorted improvements and fixes
- Version 2.0.5
  - Ensured websocket connection uses user's RBAC credentials
  - Corrected a potential CSRF issue when using the REST API graphical browser
- Version 2.0.4
  - Corrected a privilege escalation related to user account levels
- Version 2.0.2
  - Further corrections for job execution with certain 0mq library versions
  - Changes to AMI license logic to allow bring-your-own-license usage
- Version 2.0.1
  - Corrected a job execution issue due to 0mq library versions on certain platforms
  - Reduced logfile verbosity and retention for some Tower subcomponents
  - Adjusted setup playbook for the release of EPEL 7
- Version 2.0
  - New dashboard that provides at-a-glance status of your Ansible deployment
  - Completely redesigned job status page featuring real-time playbook output and progress updates

    - Added support for multiple new cloud providers - Azure, Google Compute Engine, and VMware vSphere
    - New user interface look and feel
    - Integrated monitoring support for checking the health of your Tower install
    - Tower now requires a license to run. 10 machine free licenses, as well as free large trial licenses, are available at http://ansible.com/license
    - Support added for Red Hat Enterprise Linux 7 and CentOS 7
    - Upgrades will reuse password information, not requiring reentry in 'group_vars/all' of setup playbook
    - Many assorted improvements and fixes
- Version 1.4.12
    - Corrected an issue handling Unicode output from ansible-playbook
    - Corrected an issue displaying job details for some jobs
- Version 1.4.11
    - Performance improvements to inventory import and deletion
        - Groups UI under inventory tab is now paginated
        - Updated UI options for moving and copying groups (and host contents)
    - Added the ability to optionally prompt for job variables when launching jobs to the job template detail pages
- Version 1.4.10
    - Correctly handle schedule creation when browser timezone cannot be detected.
    - Corrected pagination on job_events page.
- Version 1.4.9
    - Corrected a provisioning callback issue on Enterprise Linux.
    - Added a sample provisioning callback script.
    - Various backend and UI improvements.
- Version 1.4.8
    - 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.

Known Issues

1. 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.
2. 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.
3. On Red Hat Enterprise Linux 7 and CentOS 7, you will need to either disable the `firewalld` service (if active), or modify the `firewalld` configuration to allow incoming connections on port 80, 443, and 8080.

Getting Started

Welcome to Ansible Tower!

To get started, you can follow the quick installation instructions below. Detailed installation instructions are in the section entitled Installation and Setup Reference. 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 http://support.ansible.com/ and let us know what you think and your ideas for future features!

Requirements

Ansible Tower has the following minimum requirements:

- Supported Operating Systems:
  - Red Hat Enterprise Linux 6 64-bit
  - Red Hat Enterprise Linux 7 64-bit
  - CentOS 6 64-bit
  - CentOS 7 64-bit
  - Ubuntu 12.04 LTS 64-bit
  - Ubuntu 14.04 LTS 64-bit
- Ansible (1.7.X or later)
- 4 GB RAM
- 20 GB hard disk
- For Amazon EC2:
  - Instance size of m3.large or larger
  - an instance size of m3.xlarge or larger is suggested if there are more than 100 hosts

While other operating systems may technically function, currently only the above list is supported to host an Ansible Tower installation. If you have a firm requirement to run Tower on an unsupported operating system, please contact Ansible at http://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 job template or the system `ansible.cfg` file). To avoid possible resource conflicts, Ansible recommends 4GB of memory per 100 forks. For example, if `forks` is set to 100, 4GB of memory is recommended; if `forks` is set to 400, 16GB of memory is recommended.

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 http://support.ansible.com/.

NOTE: *It is strongly recommended to use the latest stable release of Ansible for best performance and to ensure the latest bugfixes are available. However, any version of Ansible 1.7 or later is supported for Ansible Tower 2.X.*

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

Prerequisites

Tower is installed using Ansible playbooks. Therefore, you need Ansible to install Tower. Ansible Tower requires Ansible version 1.7.x or later.

Ansible can be installed as detailed in the Ansible documentation at:

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

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

Configure access to the repository for Ansible

For Red Hat Enterprise Linux and CentOS (version 6 or later):

Configure the EPEL repository and any additional repositories.

- For Red Hat Enterprise Linux 6 and CentOS 6:

```
root@localhost:~$ yum install \
   http://download.fedoraproject.org/pub/epel/6/x86_64/epel-release-6-8.noarch.rpm
```

- For Red Hat Enterprise Linux 7 and CentOS 7

```
root@localhost:~$ yum install \
   http://download.fedoraproject.org/pub/epel/7/x86_64/e/epel-release-7-5.noarch.rpm
```

NOTE: *You may also need to enable the "extras" repository. This is named "extras" on CentOS 7, "rhel-7-server-extras-rpms" on Red Hat Enterprise Linux 7, and "rhui-REGION-rhel-server-extras" when running in EC2.*

NOTE: *For users of Red Hat Enterprise Linux 7, you also need to enable the "optional" repository. When using the official Red Hat Enterprise Linux 7 marketplace AMI, be sure you install the latest "rh-amazon-rhui-client" package that allows enabling the optional repo (named "rhui-REGION-rhel-server-optional" in EC2).*

For Ubuntu 12.04 and Ubuntu 14.04:

- Configure Ansible PPA

```
root@localhost:~$ apt-get install software-properties-common
root@localhost:~$ apt-add-repository ppa:ansible/ansible
```

Install Ansible

For Red Hat Enterprise Linux and CentOS (version 6 or later):

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

For Ubuntu 12.04 and Ubuntu 14.04:

```
root@localhost:~$ apt-get update
root@localhost:~$ apt-get install ansible
```

Guided Installation

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: *Tower is a full application and the installation process installs several dependencies such as PostgreSQL, Django, Apache, and others. We require that you install Tower on a standalone VM or cloud instance and do not co-locate any other applications on that machine (beyond possible monitoring or logging software). Although Tower and Ansible are written in Python, they are not just simple Python libraries. Therefore Tower cannot be installed in a Python virtualenv, a Docker container, or any similar subsystem; you must install it as described in the installation instructions below.*

Get the Tower Installer

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.

Download this tarball, and extract it. 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., "2.1".

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

Run the Tower Install Wizard

Ansible Tower includes a simple text-based wizard that configures your Tower installation.

This section of the User Guide assumes a Tower installation utilizing a built-in database on the host you are running the installation from. For other install scenarios, including installing Tower on a remote host, Tower in a High Availability configuration, or Tower using an external database, please see the Installation and Setup Reference section.

To install Tower on the local machine with a local database, invoke the configure script as `./configure --local`. This will simplify the installer questionaire process. If you wish to use an external database or a HA configuration, do not pass `--local` and refer to the detailed instructions.

```
root@localhost:~$ ./configure --local
-----------------------------------------
Welcome to the Ansible Tower Install Wizard
-----------------------------------------

This wizard will guide you through the setup process.

LOCAL INSTALLATION
You are installing Ansible Tower on this machine, using an internal database.

PASSWORDS
For security reasons, since this is a new install, you must specify the
following application passwords.
```

The installation wizard will ask you for the following passwords:

- Admin Password

  This is the password for 'admin', the first user (and superuser) created on installation. You'll need this password for your initial login to Tower.

- Munin Password

  This password is used by Tower superusers to access the Munin-based monitoring of your Tower server.

Once you've entered the required passwords, wizard will confirm your Tower install selection (in our case, on `localhost` with an internal database). Enter `y` to verify the installation. The wizard then saves your configuration information to the file `tower_setup_conf.yml`. You can use this file to bypass the configuration step if you want to install tower using an identical configuration and passwords in the future.

Run the installation script

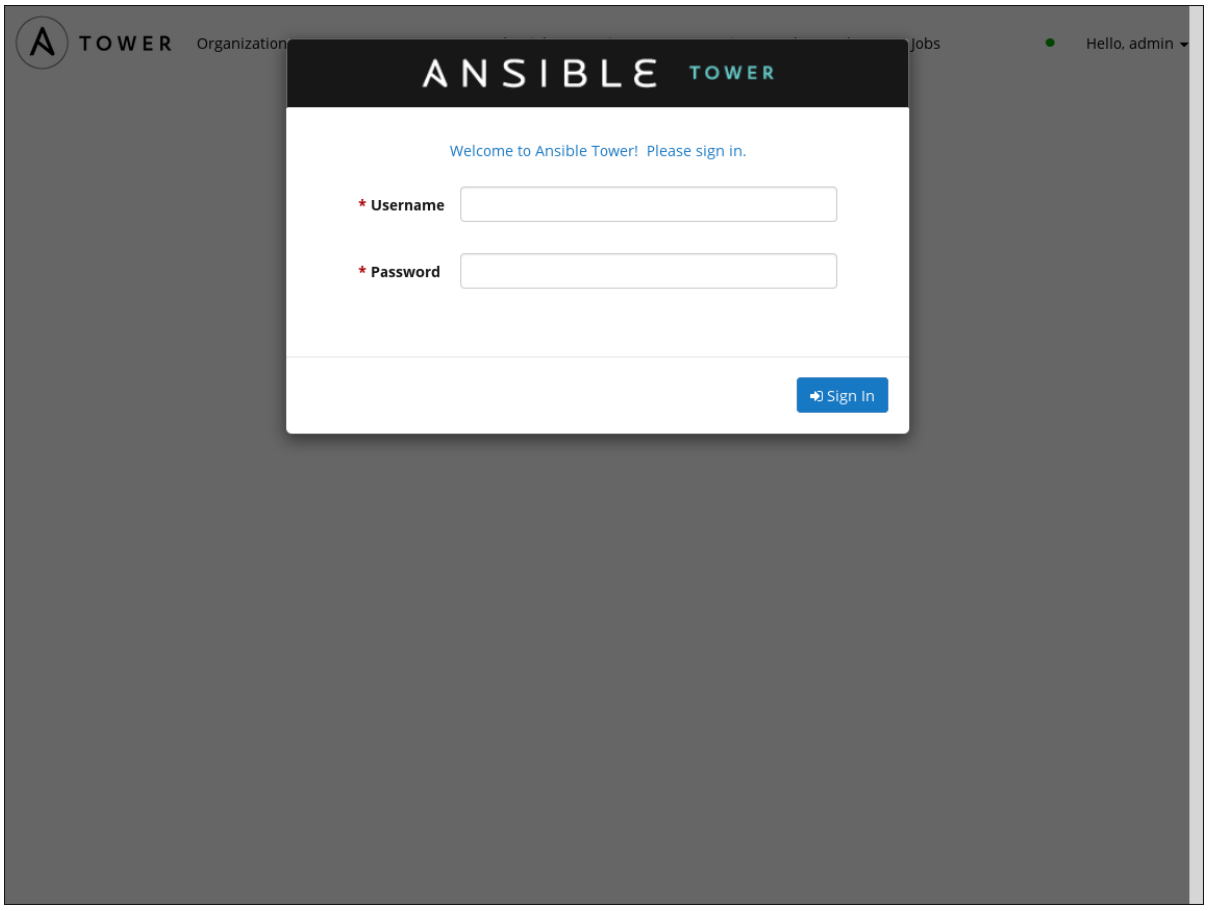
Once you've run through the configuration wizard, invoke the installation script `setup.sh` as indicated by the setup wizard, for example:

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

If the wizard suggests another command, type this command instead.

Setup will install Tower from RPM or Deb packages using repositories 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 http://support.ansible.com/. Tower subscription customers will receive a faster response by filing a support issue.

Configure LDAP / Active Directory (optional)

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

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 a Superuser
2. Import a License
3. Examine the Tower Dashboard
4. Configure Live Events
5. Create an Organization
6. Add a new User to the Organization
7. Add an Inventory to the Organization
8. Create a Credential
9. Create a Project
10. Create a new Job Template
11. 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, as described at http://docs.ansible.com/intro_installation.html.

Ready? Let's go!

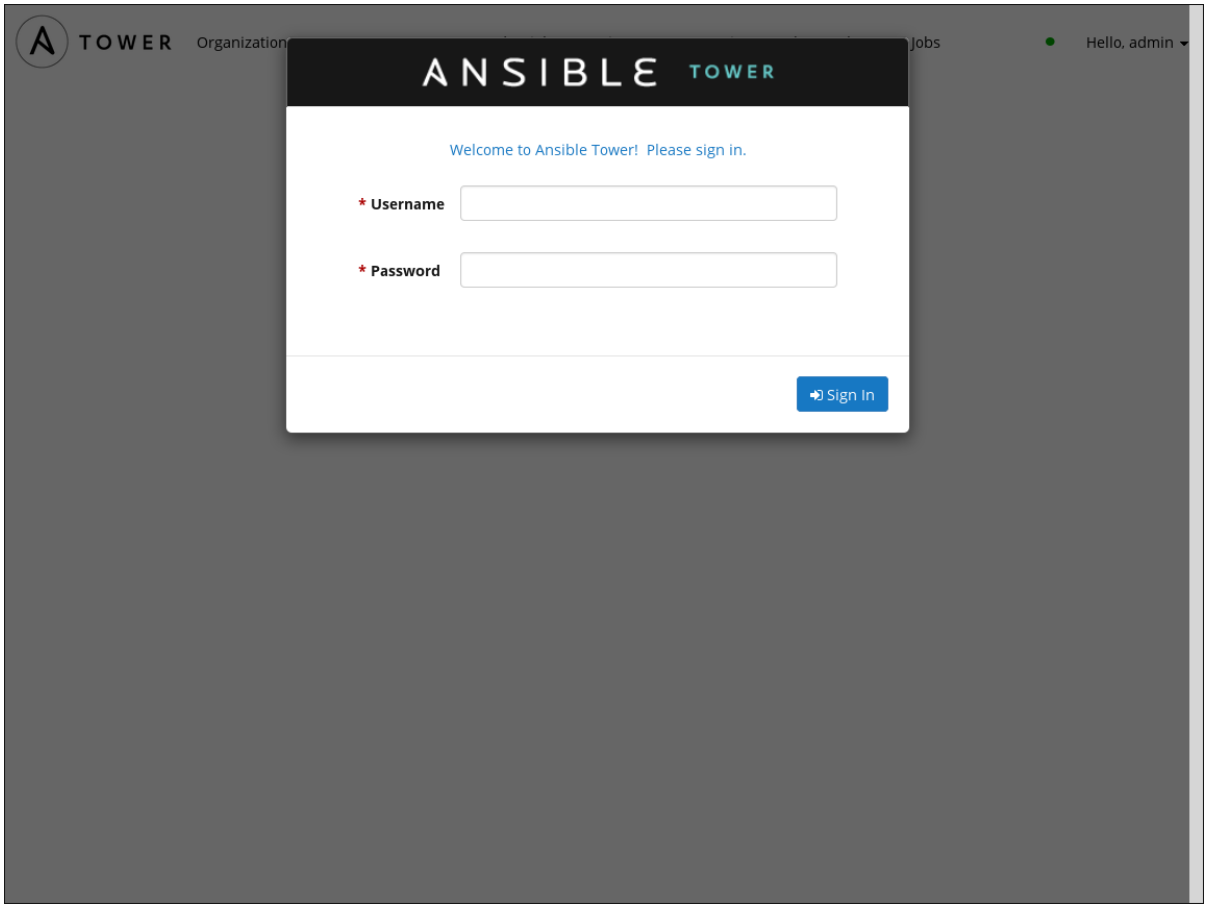
1. Login as a Superuser

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

> ***NOTE:*** *Out of the box, Tower installs a self-signed certificate for HTTPS communication. You may need to accept this certificate in your browser. See the notes in Installation and Setup on replacing this cert if needed.*

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.

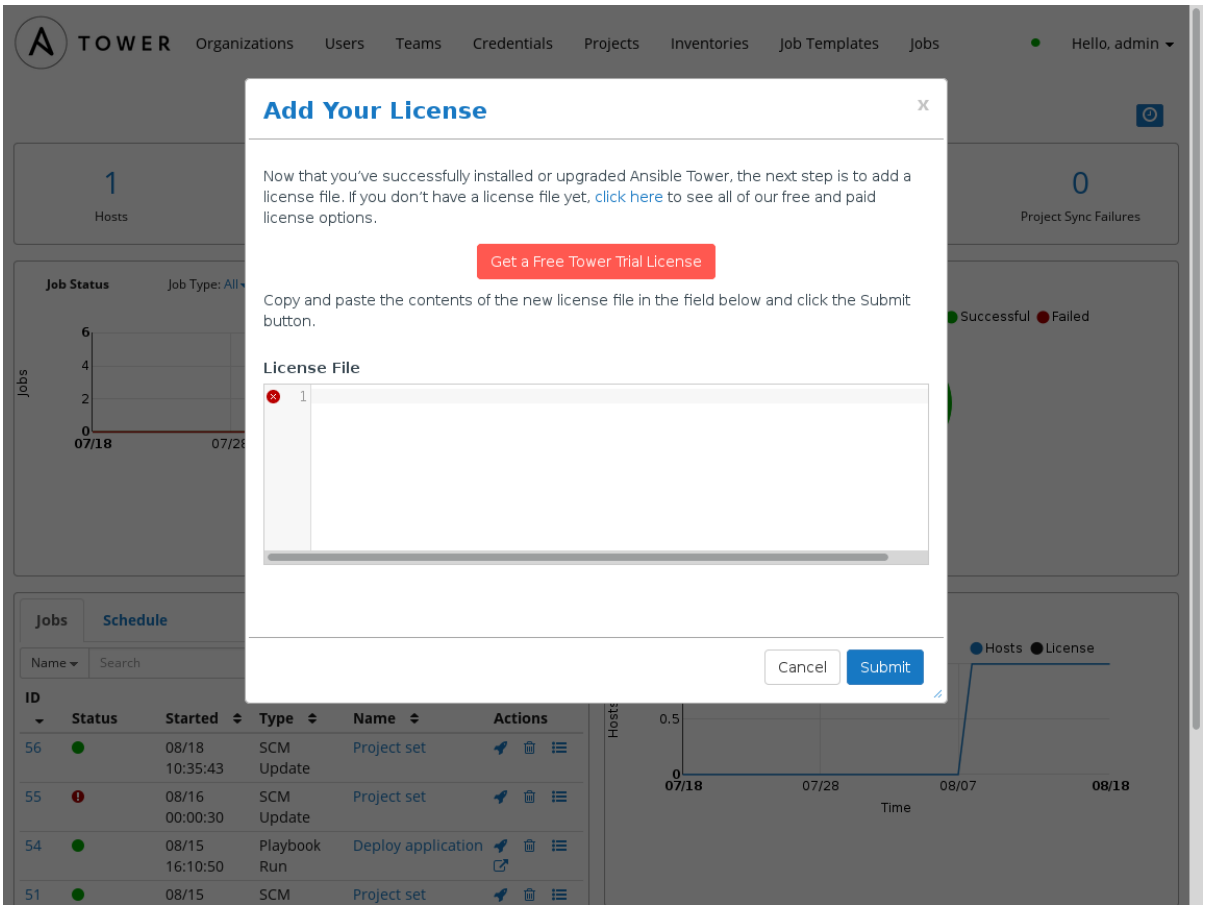
NOTE: *We'll get into the details of the differences between a normal user, superuser, and organization administrator in the section Users.*



2. Import a License

Tower requires a valid license to run. When you entered your information to download Tower, you should have received an e-mail that contains your license; you may also have received a license direct from Ansible. If you did not receive a license, or have issues with the license you received, please visit http://ansible.com/license to see our free and paid license options (including free trial licenses) or contact Ansible Support at http://support.ansible.com/.

When you start up Tower without a valid license, you'll see the following dialog.



Paste in the license you received from Ansible, and click 'Submit'. Your license should be accepted, and you can continue to the main Ansible interface.
This license screen can be viewed later from the 'View License' dropdown of the Tower user's menu on the top right screen.

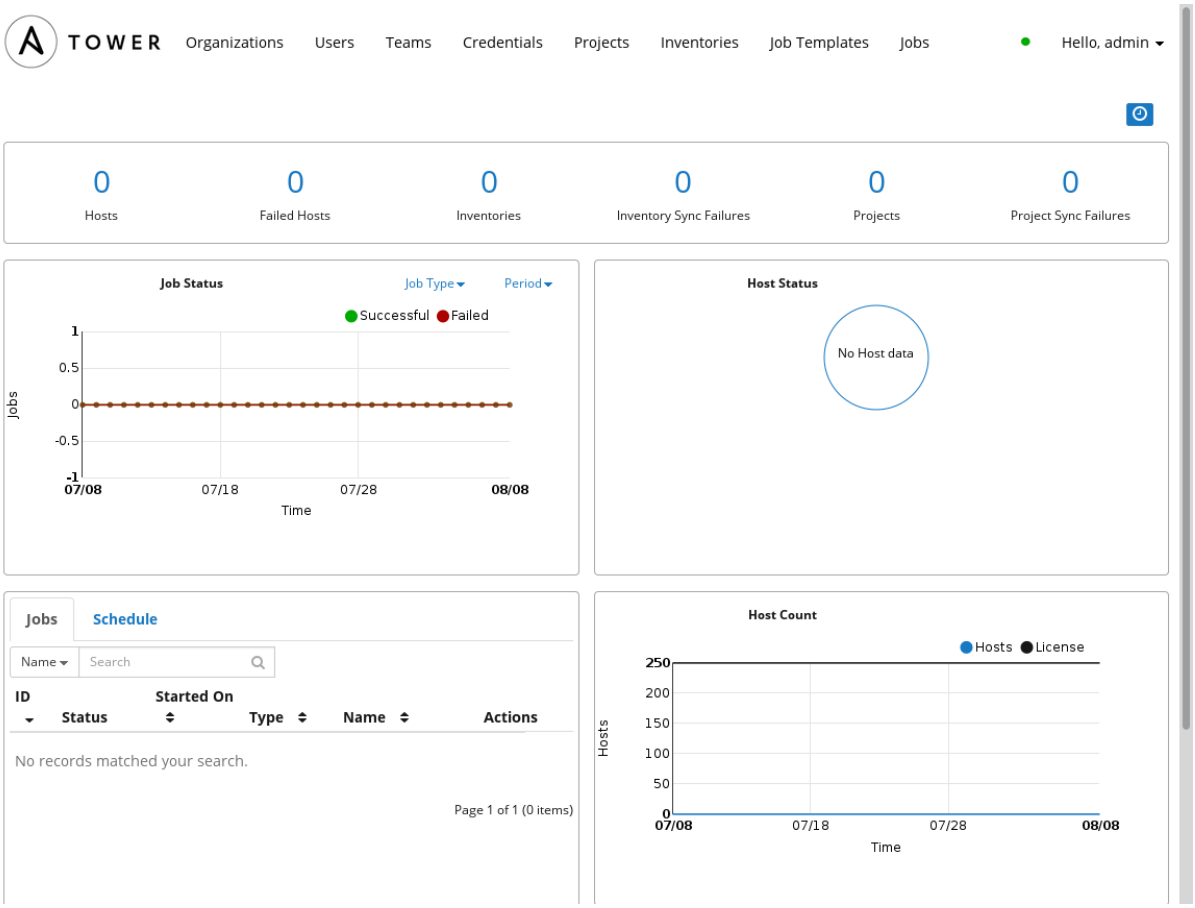
NOTE: *Only a superuser can update the license.*

NOTE: *You can also save the license file to /etc/tower/license on the Tower server.*

3. Examine the Tower Dashboard

We are now at the Tower Dashboard. On this screen, we can see a summary of your current **Hosts**, **Inventories**, and **Projects**. There is a time-based graph of job status, a summary chart of host status, a summary of both recently completed jobs and scheduled jobs, and for superusers, a summary of the host count.

Across the top of this interface, we have navigation to access all aspects of Tower, including **Organizations**, **Users**, **Teams**, **Credentials**, **Projects**, **Inventories**, **Job Templates**, and **Jobs**.



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.

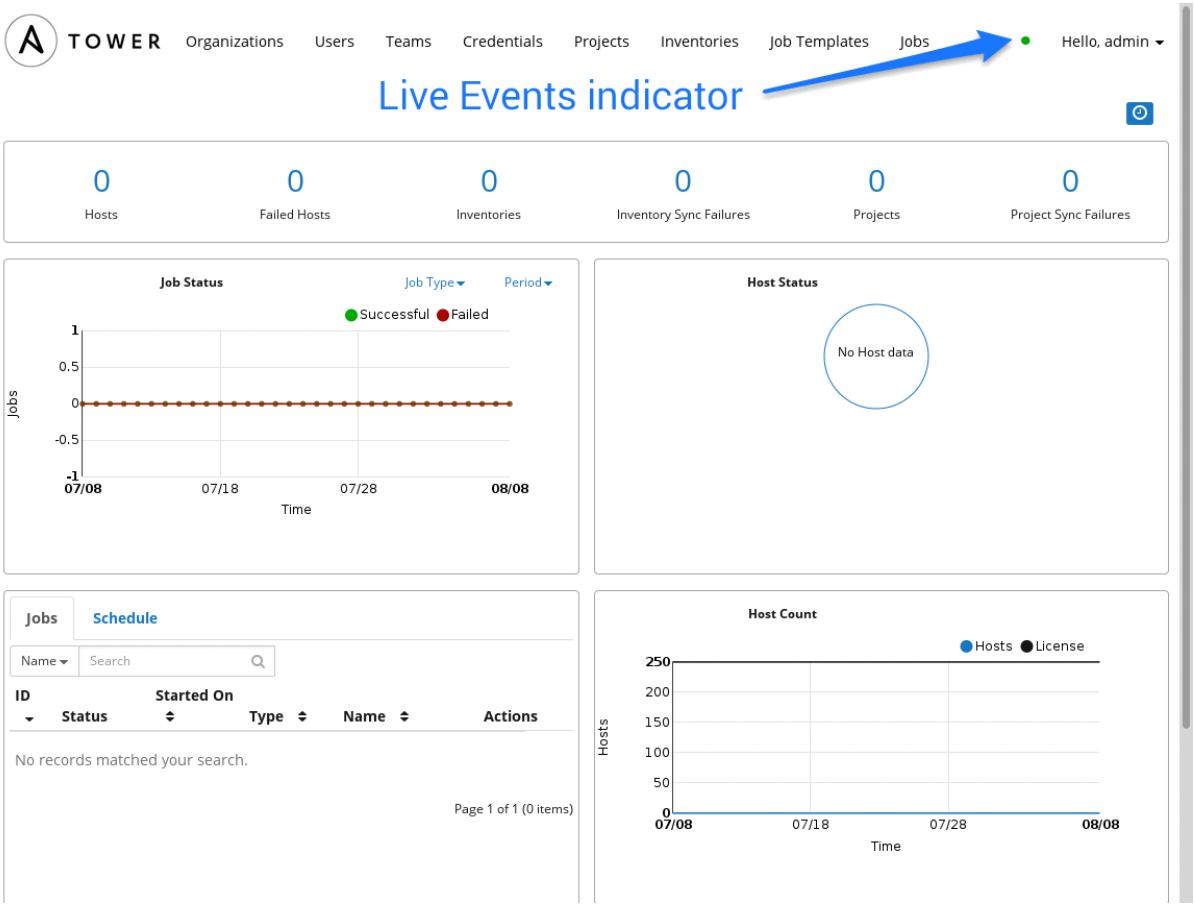
Tower provides multiple levels of role-based access, providing delegation of responsibility, but with fine - grained control over who can do what. We'll talk about that in more detail later in this document. For now, here's a simplified outline that shows the hierarchy of Tower's role based access control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

Tower Hierarchy

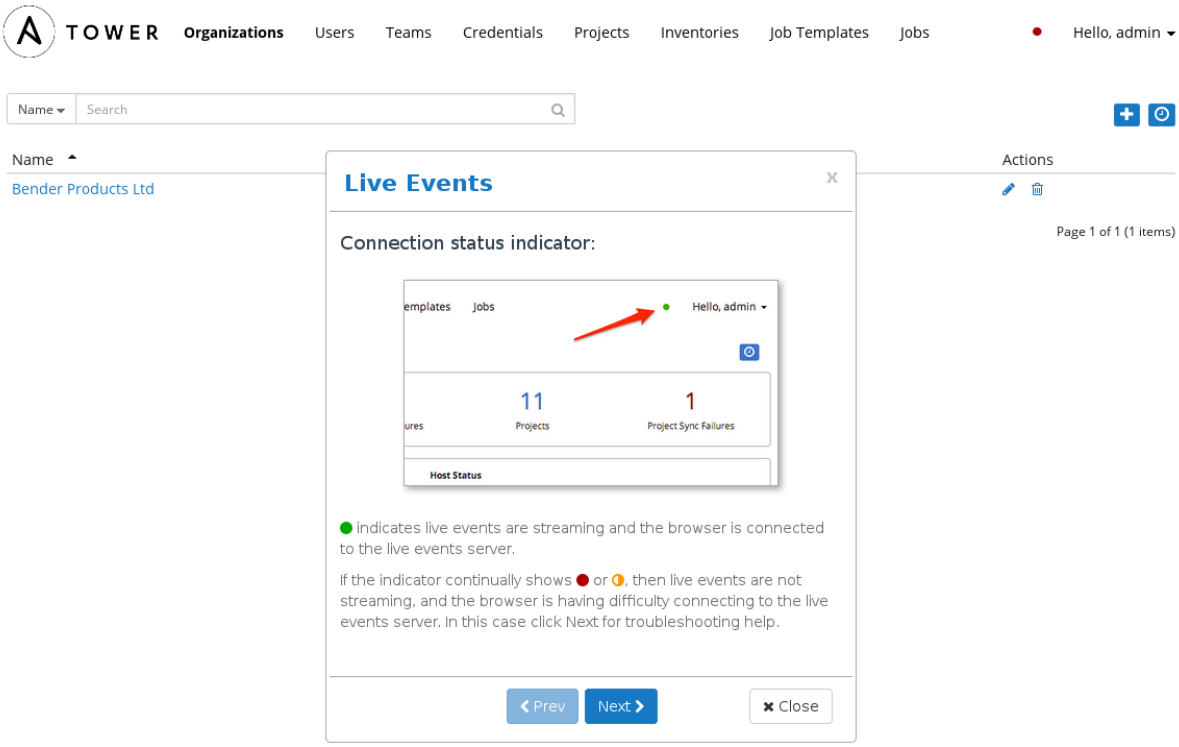
- Organization
  - Inventories
    - Groups
      - Hosts
  - Teams
    - Credentials
    - Permissions
    - Users
      - Credentials
      - Permissions
- Projects
  - Playbooks
  - Job Templates
- Jobs

4. Configure Live Events

In the Tower menu you will see a colored dot next to the Tower User's menu. This dot shows the status of Tower's *Live Events* functionality.



If this dot is green, all is well. If this dot is red or orange, Live Events are not working. In this case, click the dot to bring up the Live Events troubleshooting wizard. Follow the instructions there to configure Live Events.



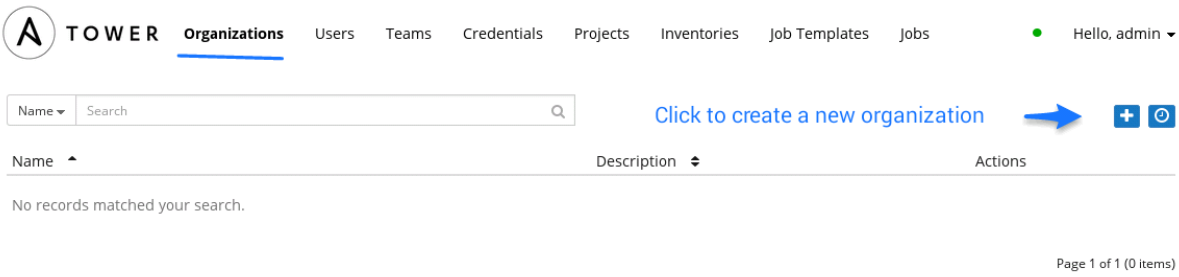
If Live Events is not working, many pages will have a button, which can be used to refresh their contents.

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.

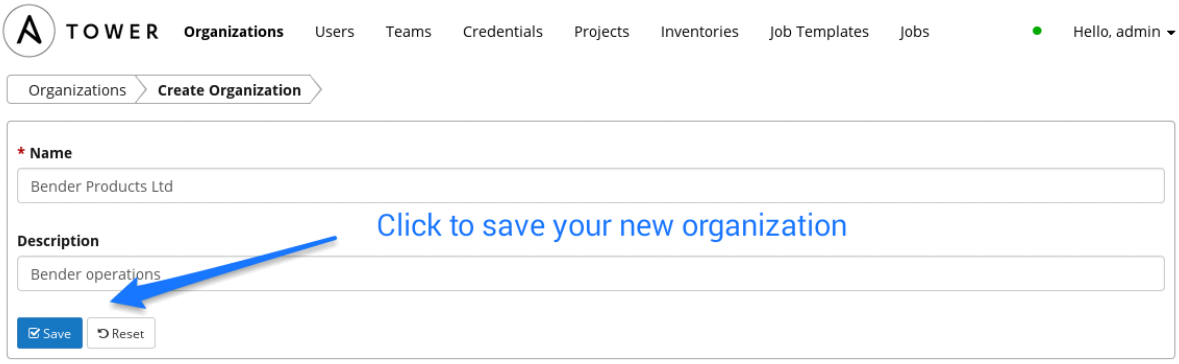
5. Create an Organization

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. Click the **Save** button to save the organization.

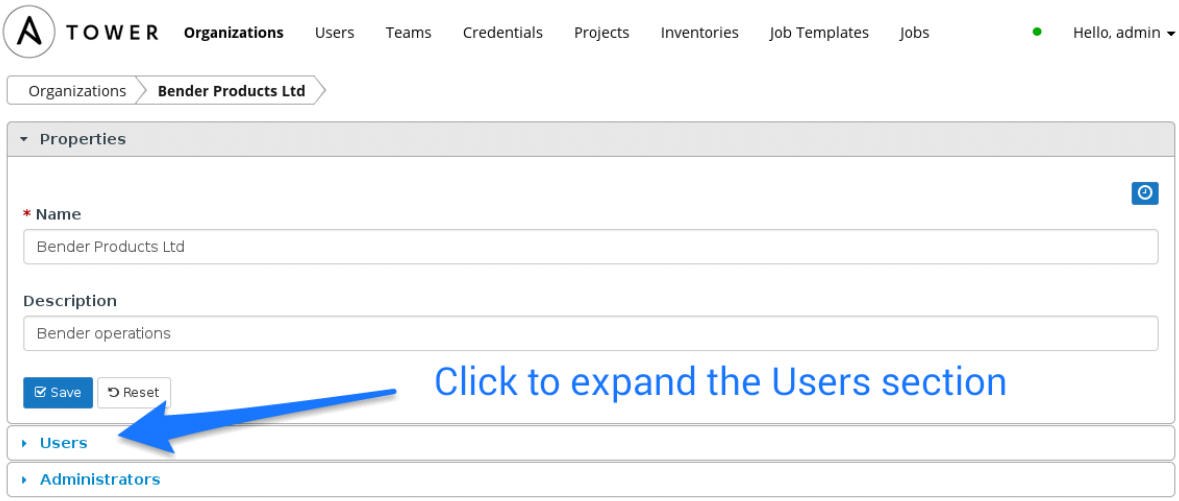


Organizations have both normal users and organization administrators. Organization Administrators are able to modify the membership and other properties of the organization, whereas normal users cannot. They are essentially superusers but only within the scope of that organization. For more about the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

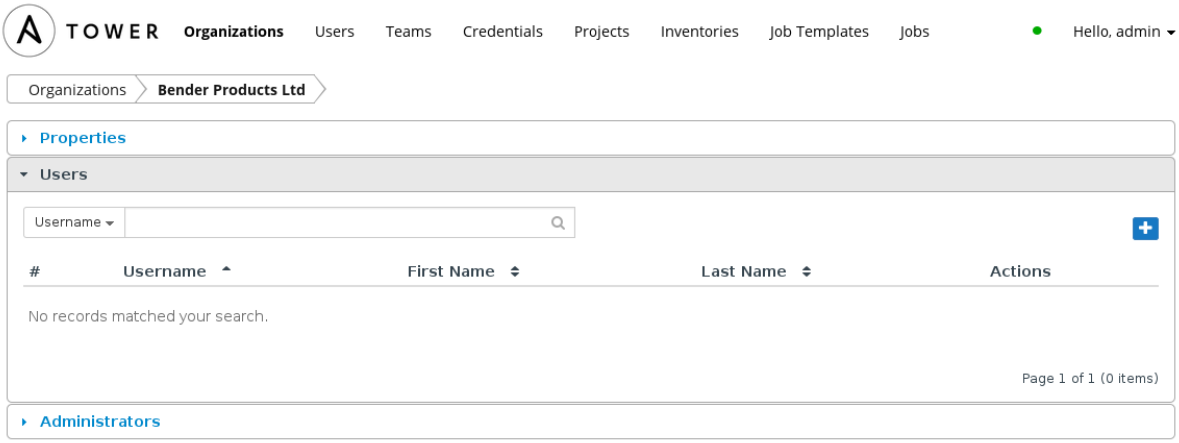
The "admin" user is a Superuser account -- a de-facto administrator for all organizations, so let's use our admin powers to create a new user and add it to our new organization. When creating a new user, the checkbox **Superuser?** corresponds to this level of access. Only Superusers can create other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

6. Create a new User and add the user to the Organization

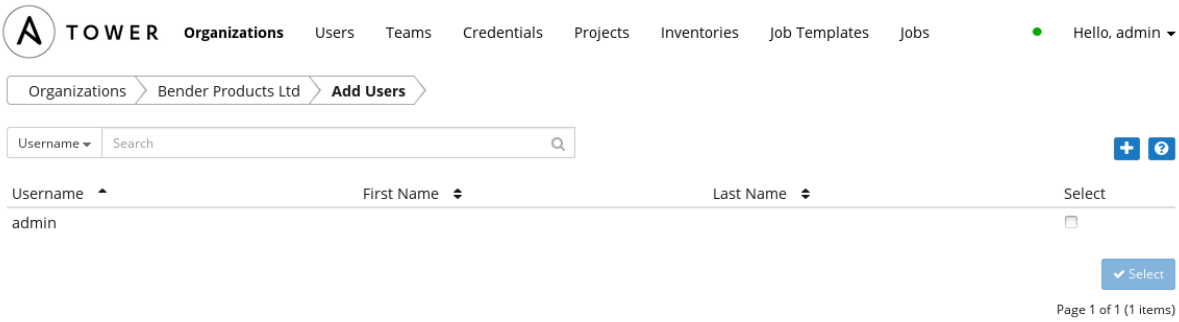
Expand the **Users** section (not the Users tab!) as shown here:



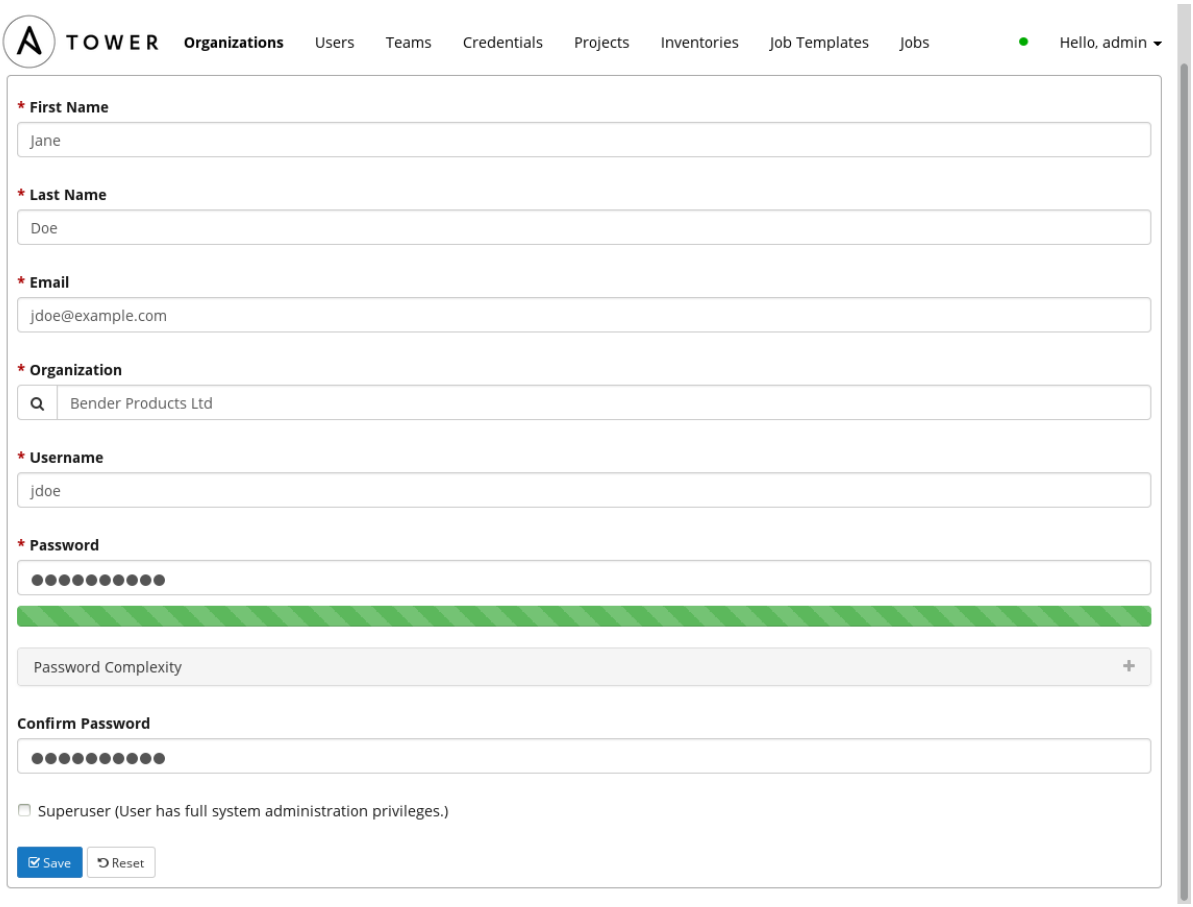
Add a user by clicking the button.



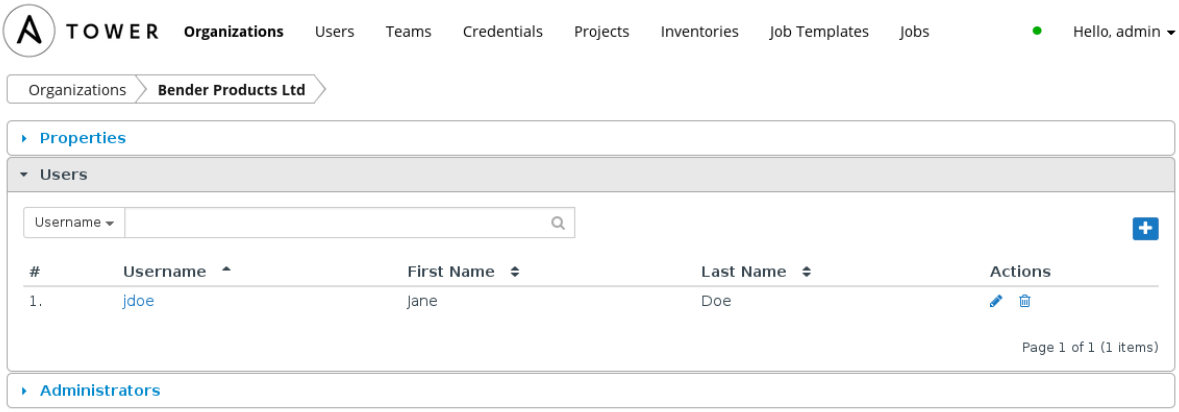
A list of all existing users will be presented. Since we have not created any users, the only user listed is "admin". Click the + button to create a brand new user.



Enter the user's details.



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.

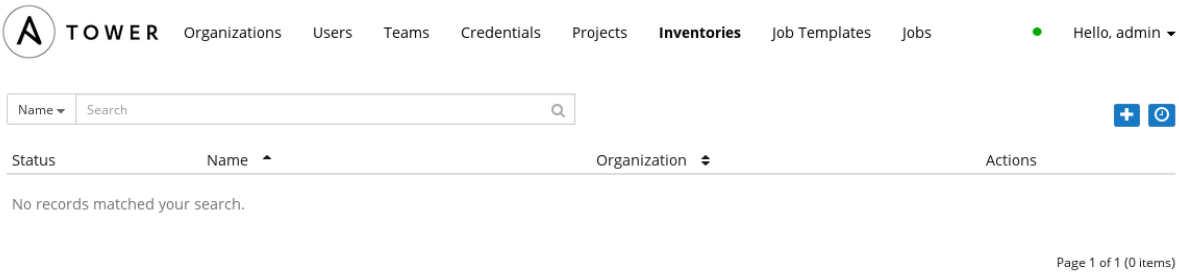


Now, we have an organization and a user. Let's add an inventory of hosts we'll be managing for Bender Products.

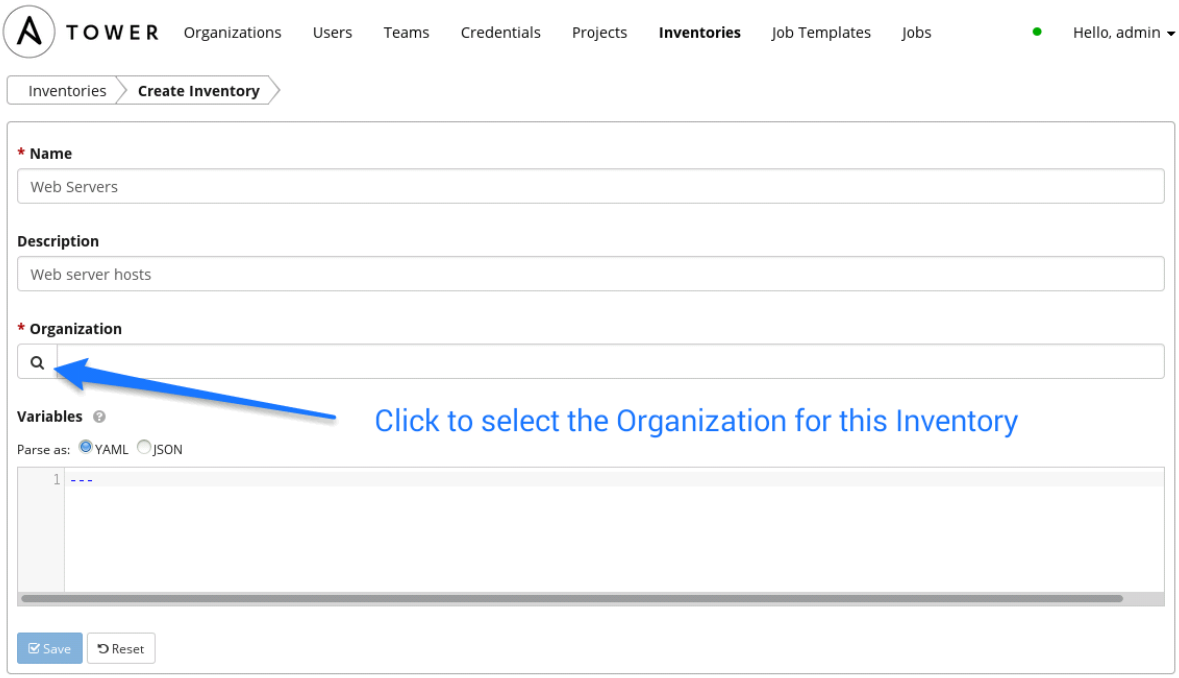
7. Create a new Inventory and add it to the Organization

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.

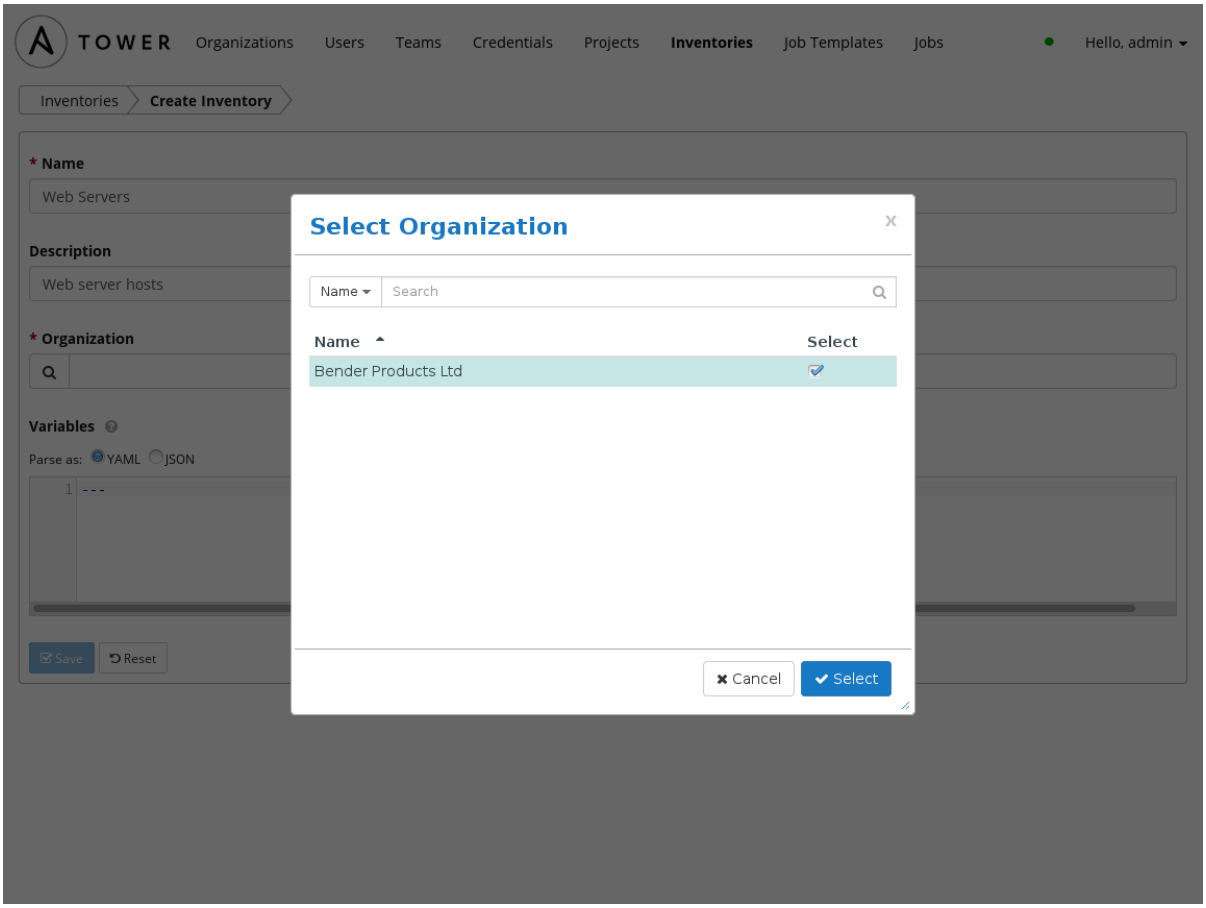
Create a new inventory by browsing to the **Inventories** tab and clicking the  button.



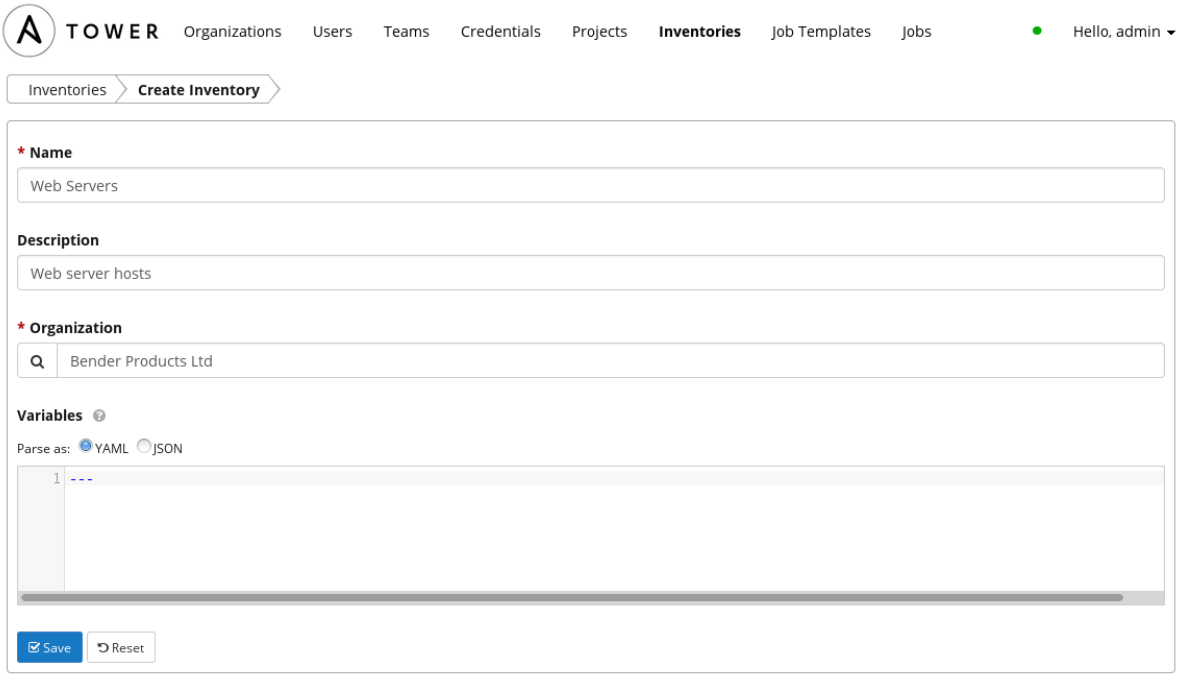
Enter the values for **Name** and **Description**. For this example, the name of our inventory will be Web Servers. Then click the  button to the left of the **Organization** field to select the organization that this inventory should belong to.



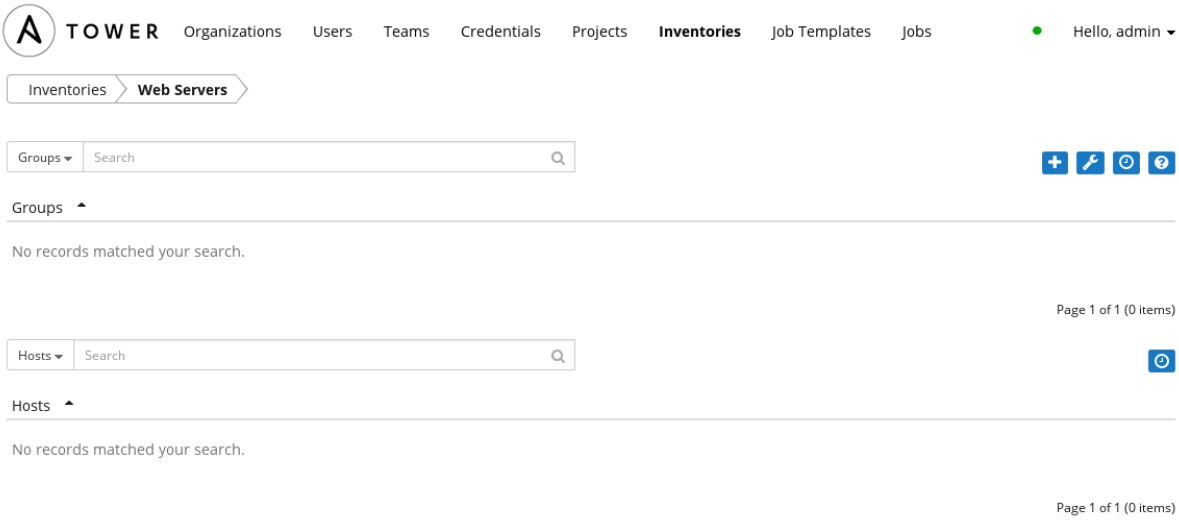
For this example we'll use the organization we created earlier. Select the row from the list by clicking on it. The selected row will be highlighted. Click the **Select** button to confirm your choice.



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.



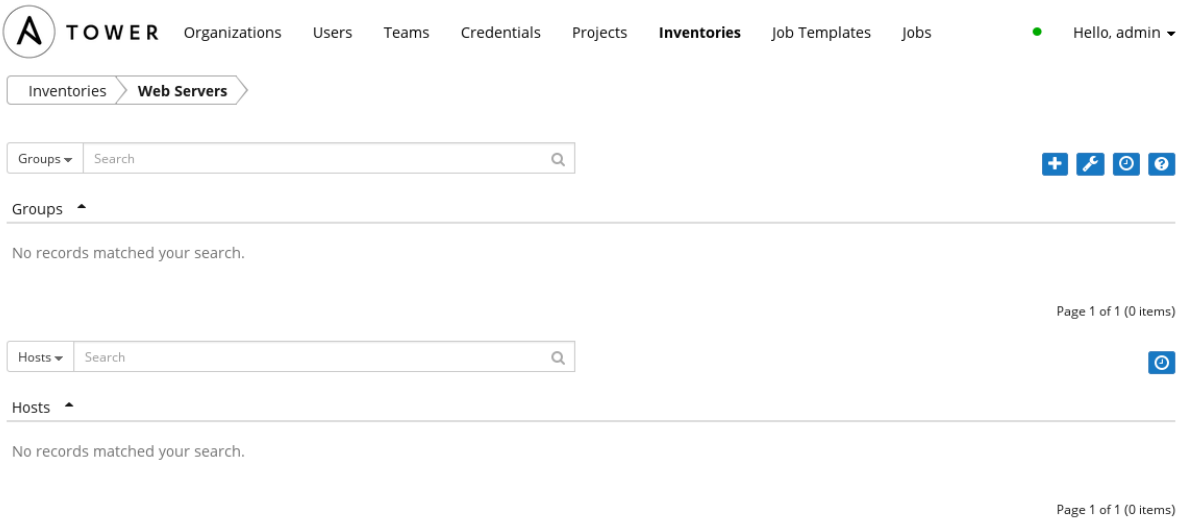
After clicking **Save**, you will see the **Inventories** screen for the Web Servers inventory.



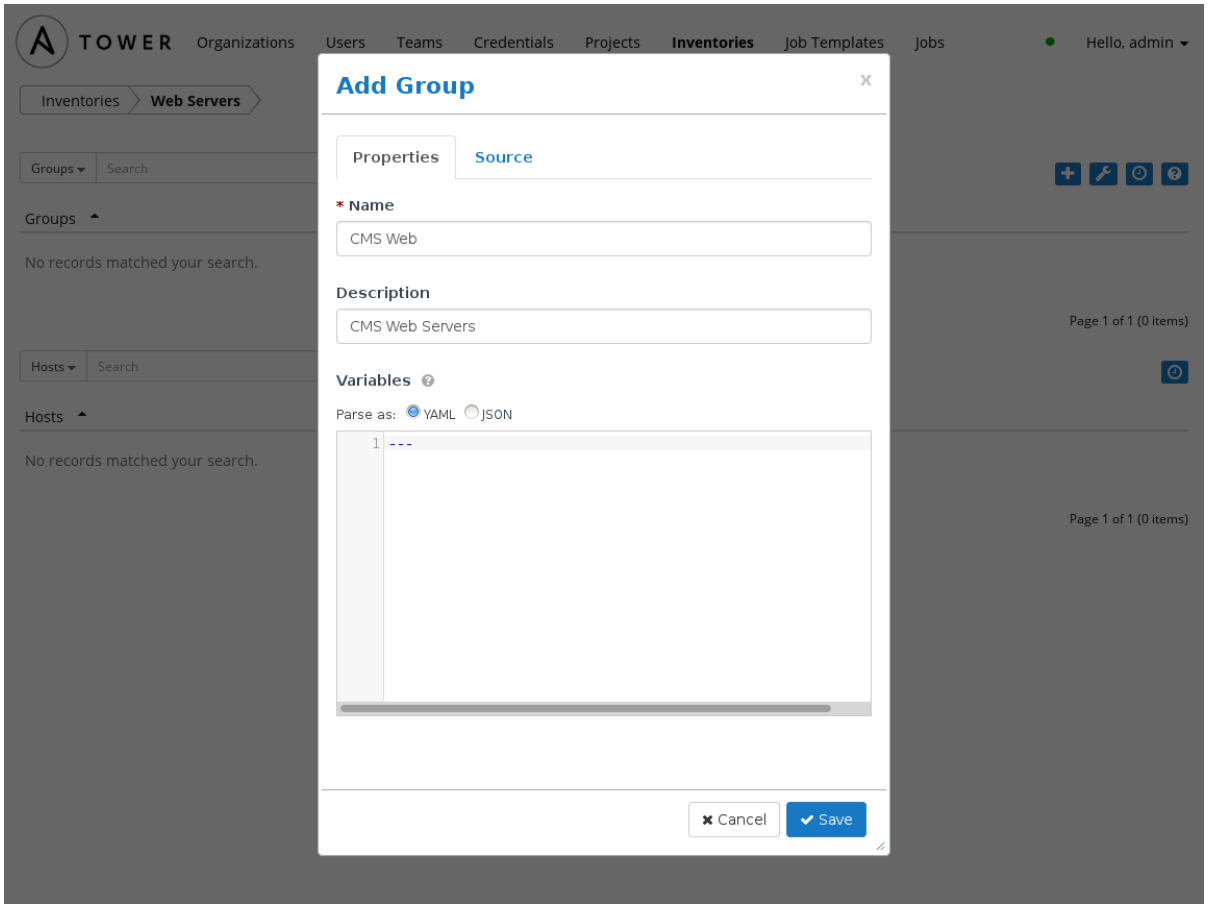
Inventories are divided into groups and groups contain hosts. A group might represent a particular environment (e.g. "Datacenter 1" or "Stage Testing"), a type of server (e.g. "Application Servers" or "DB Servers"), or any representation of your environment.

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

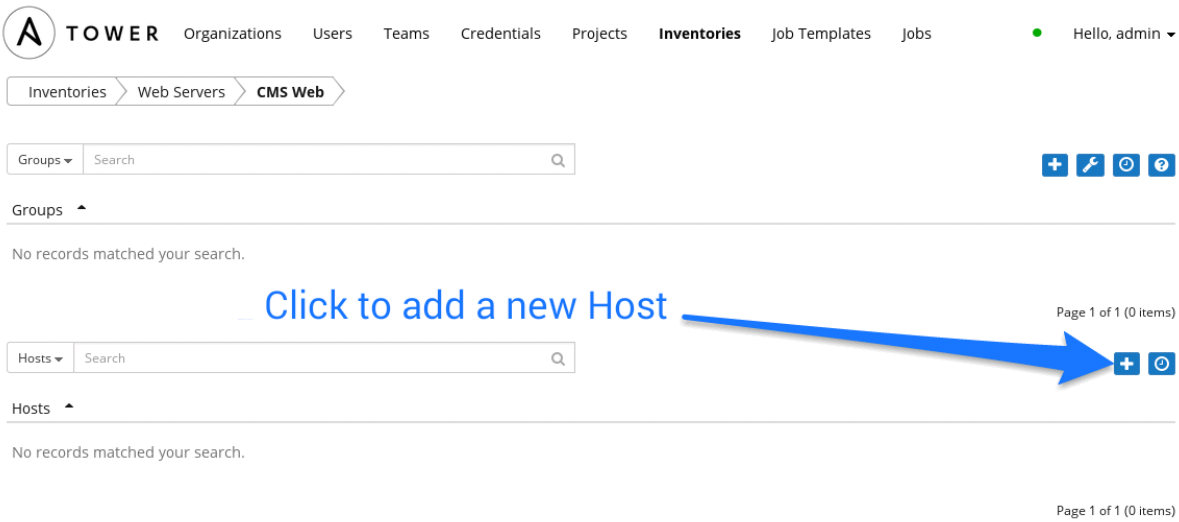


Bender Products has a group of web server hosts supporting the corporate CMS application. To add these hosts to the Web Servers inventory we'll create a "CMS Web" group. Again, we will defer a discussion of variables for later. Click the **Save** button to create the group.



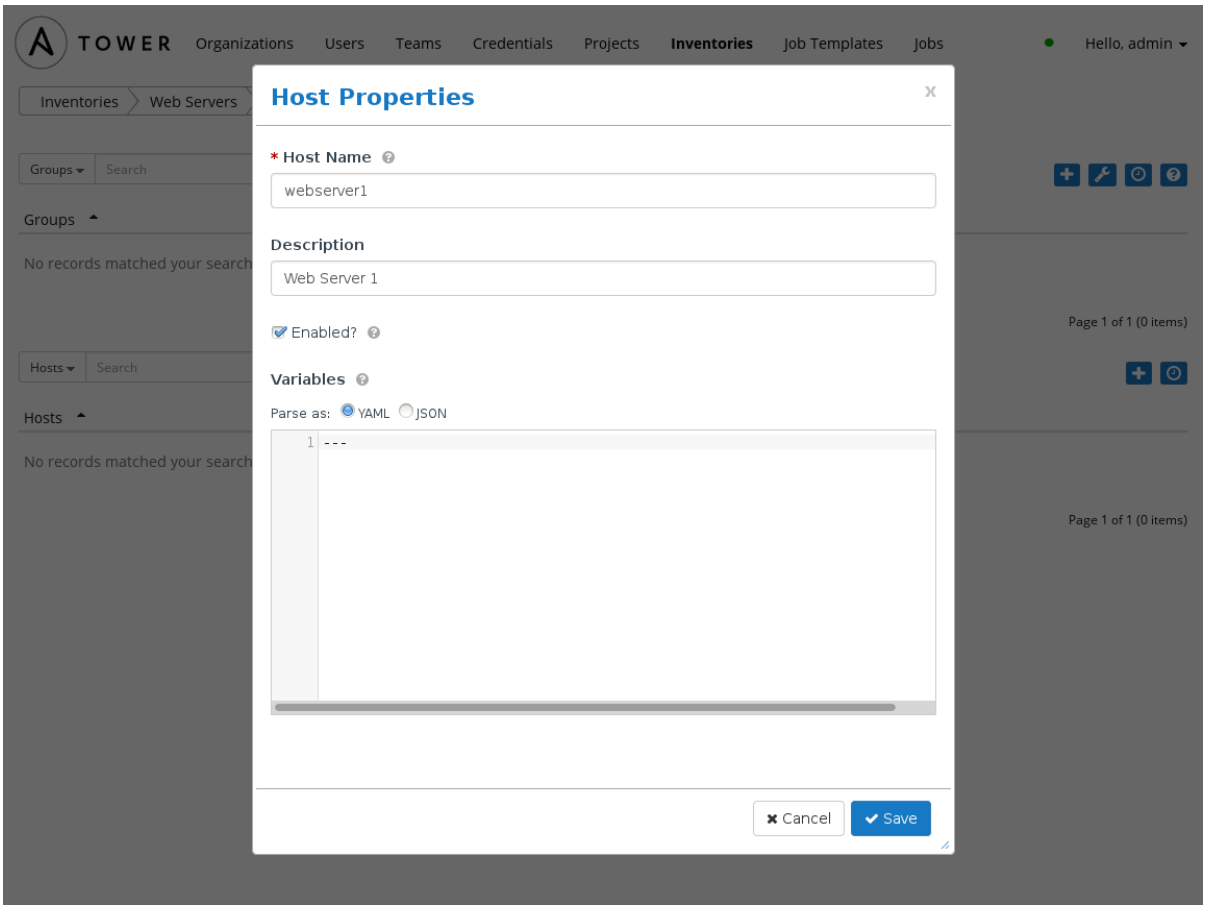
Finally, we'll add a host to the group.

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

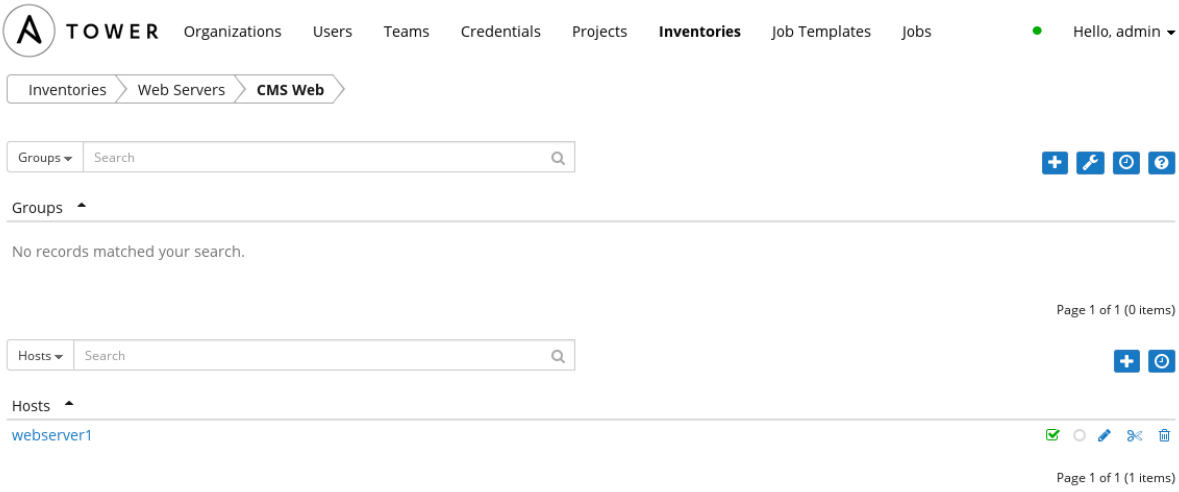


Enter the Host Name, which should either be the DNS resolvable name of the host or its IP address. This is how Tower will contact the host, so the host must be reachable using this hostname or IP address for Tower to function properly. The **Description** is arbitrary, as usual. (*Note, experienced Ansible 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.



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

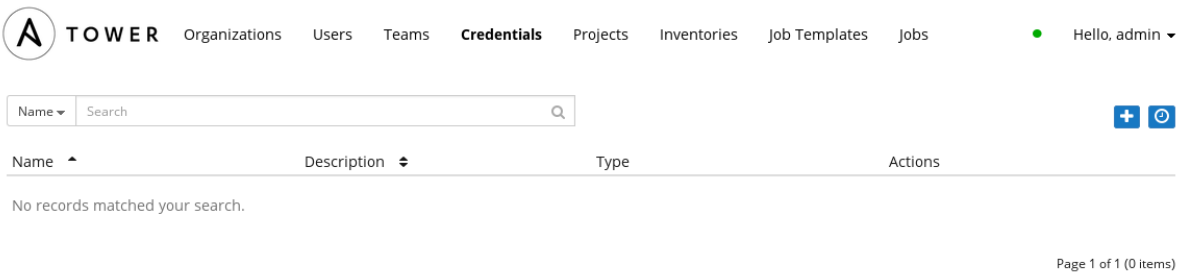


Next, we'll add credentials to our new user that Tower can use to access and launch Ansible playbooks for the host in our inventory.

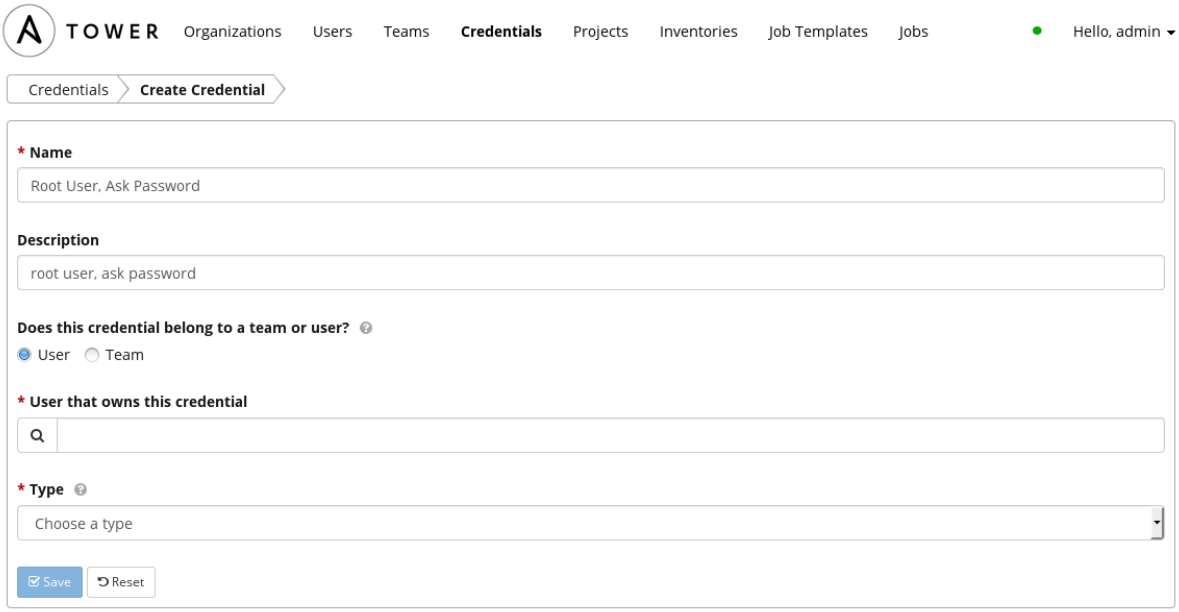
8. Create a new Credential

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.

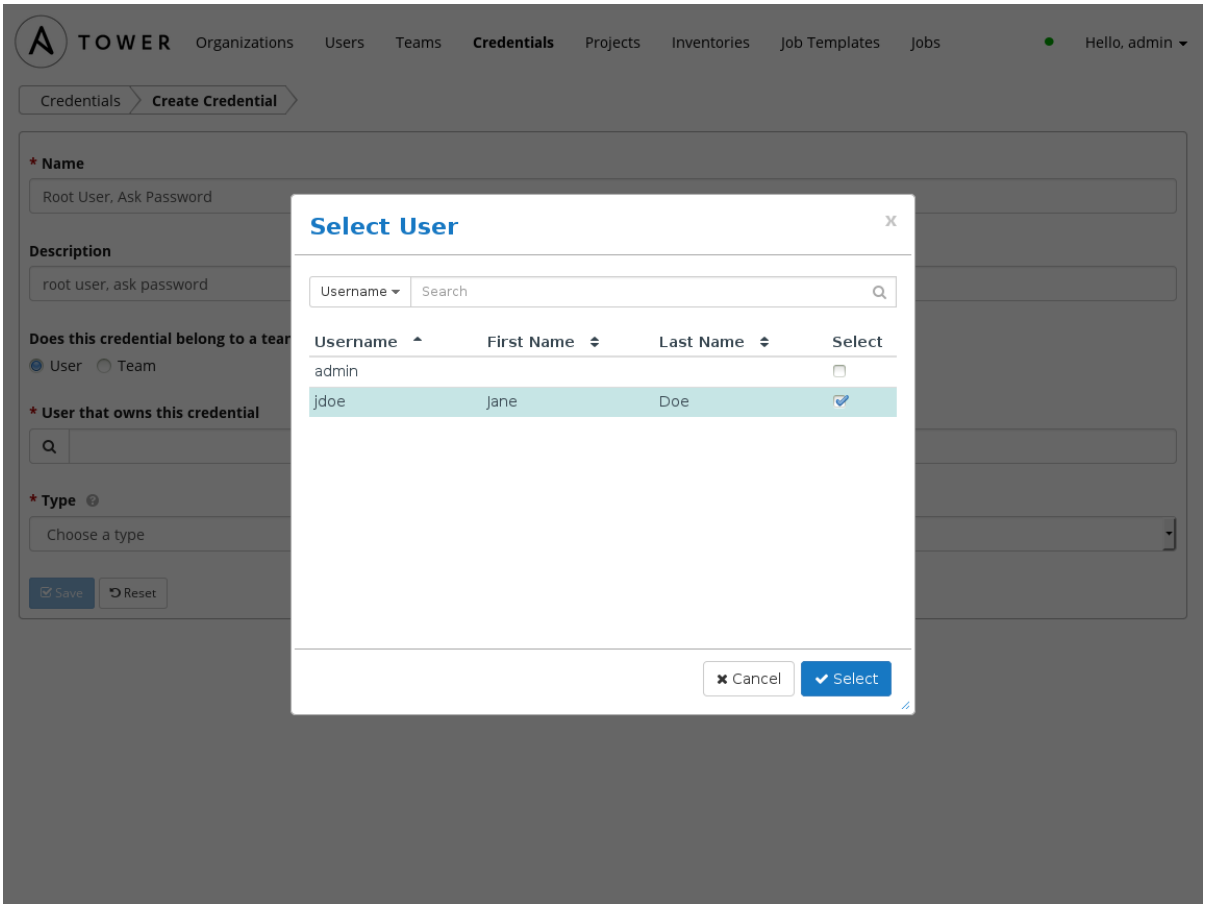


Enter an arbitrary **Name** and **Description** for this credential. Either an individual user or a team may own credentials. Let's associate this credential with the user we created in step #3. Select the "User" radio button.

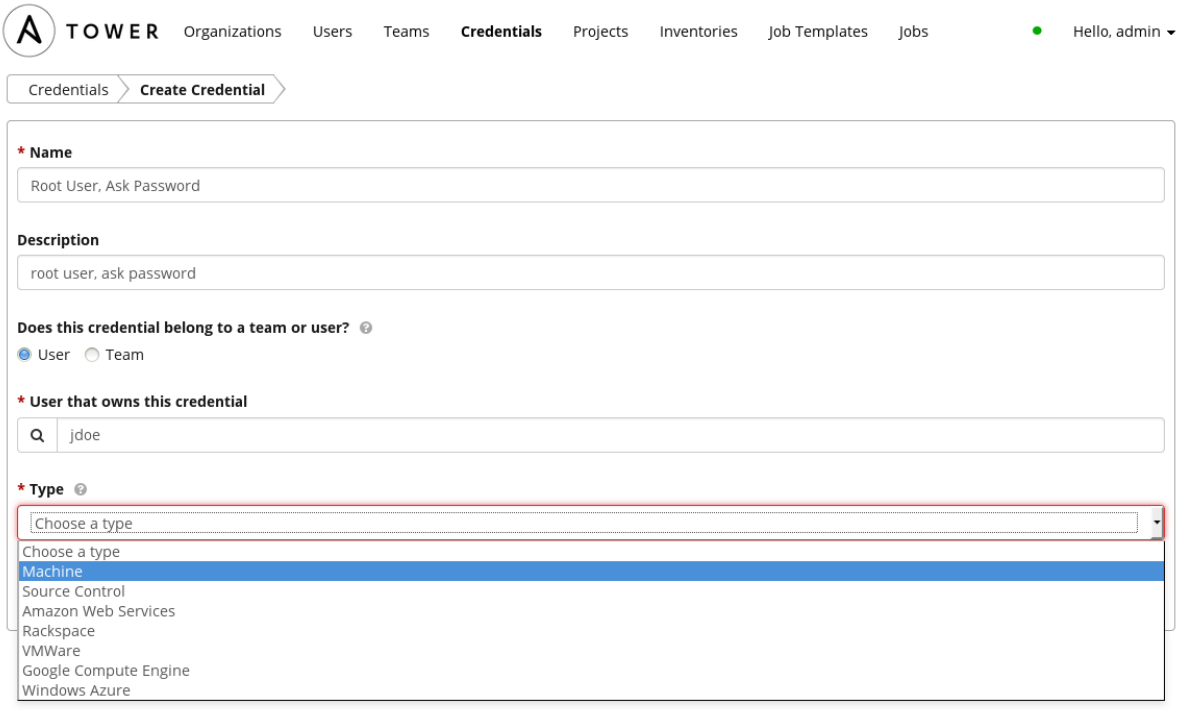


Then, select the  button to find the user that we created in step #3.

Find and select the "jdoe" user.



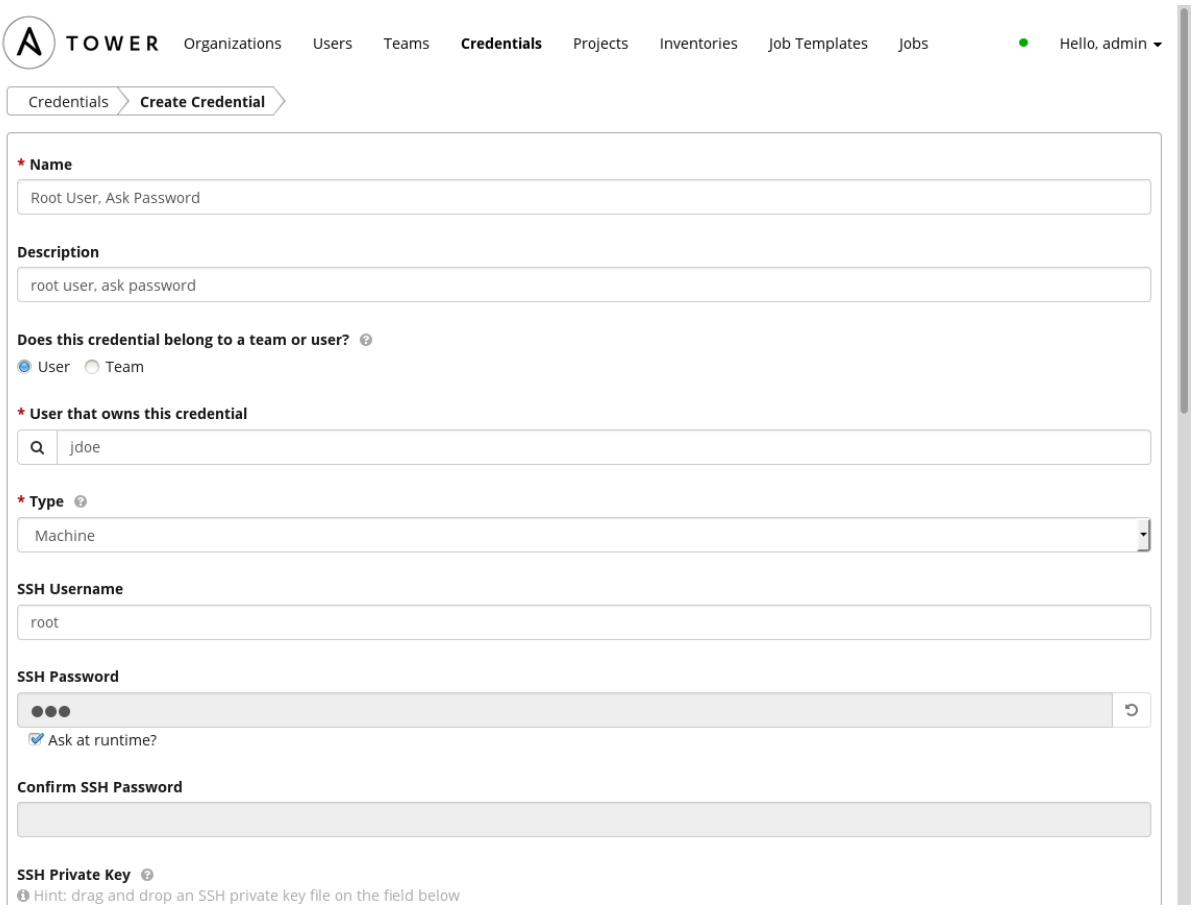
Next, select credential type **Machine**.



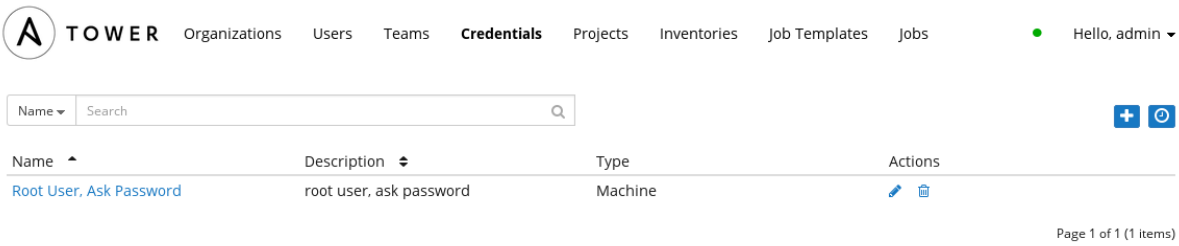
Now, we'll enter the details of the appropriate authentication mechanism to use for the host we added to Tower in step #3. Use the actual credentials for the real host. To keep things simple, we'll use an SSH password, but ask for it at runtime. So, rather than enter the password here, we'll enter it 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.



Click **Save**.



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

9. 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
```

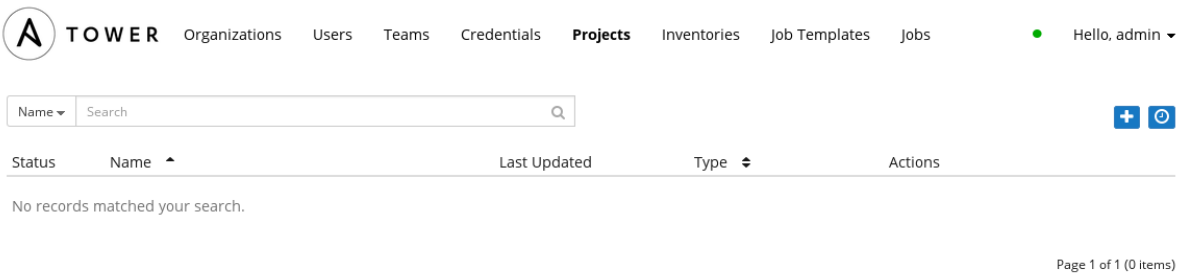
The contents of the file are below:

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

  tasks:

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

Note the user: root line. If you need to log into the target machine as another user, for instance, "ubuntu", change that here. Also note the indentation - it is important. 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://docs.ansible.com/YAMLSyntax.html and http://yaml.org/.*

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

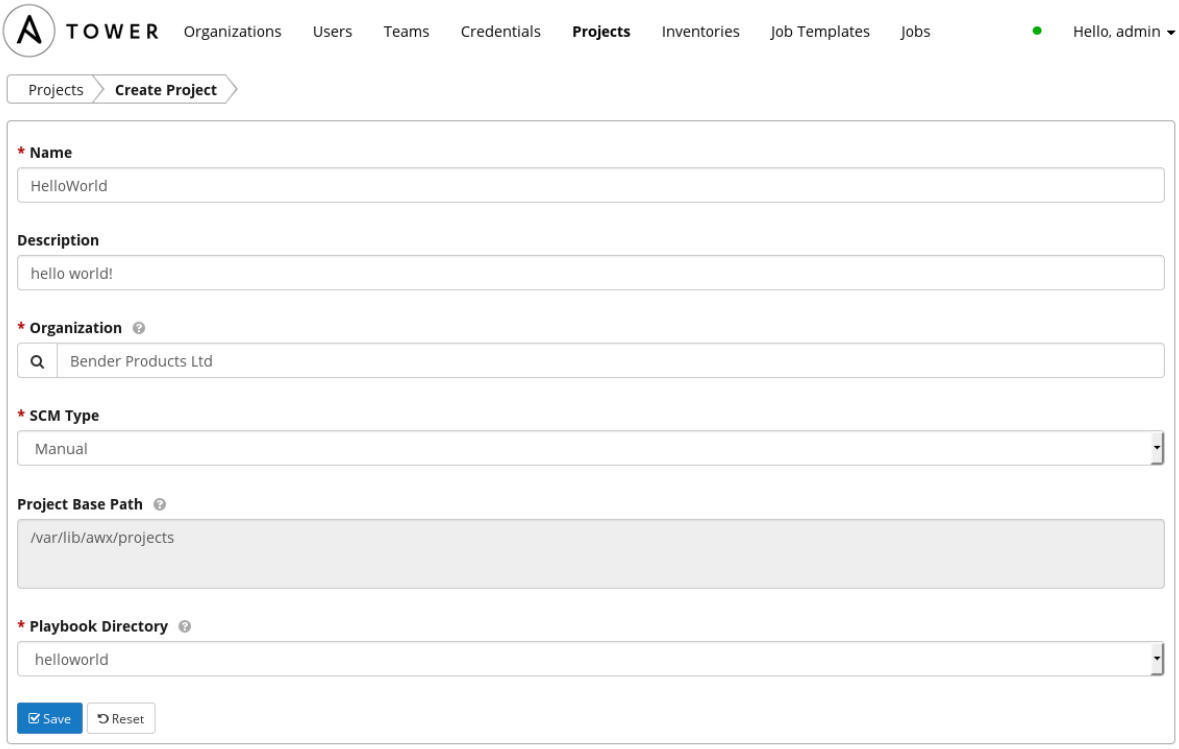


Enter a **Name** and **Description** for the project.

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.

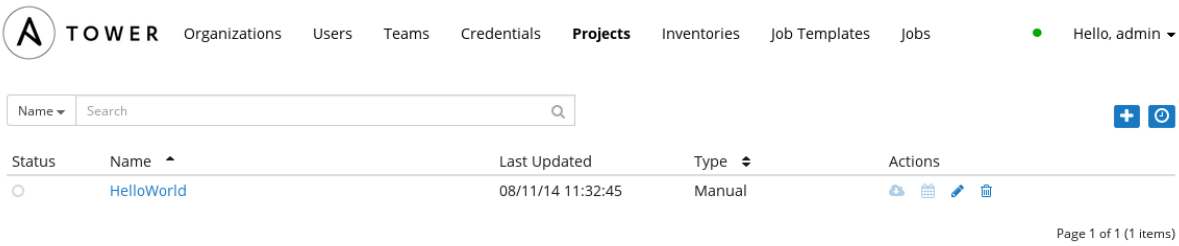


NOTE: *If you see the following warning:*

*"**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."*

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.*

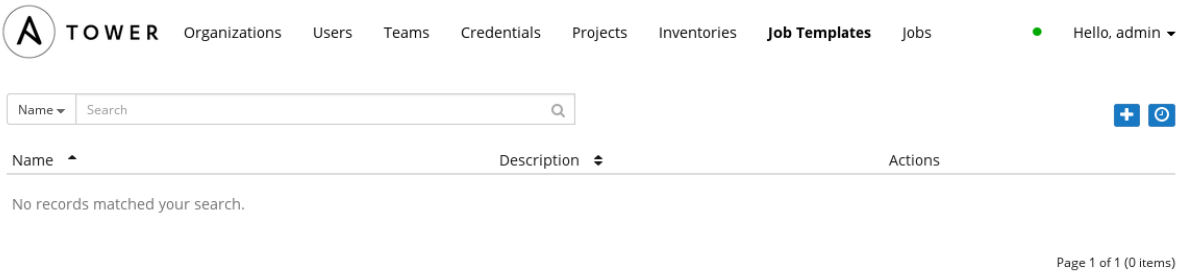
Select **Save** and the new project will be displayed.



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

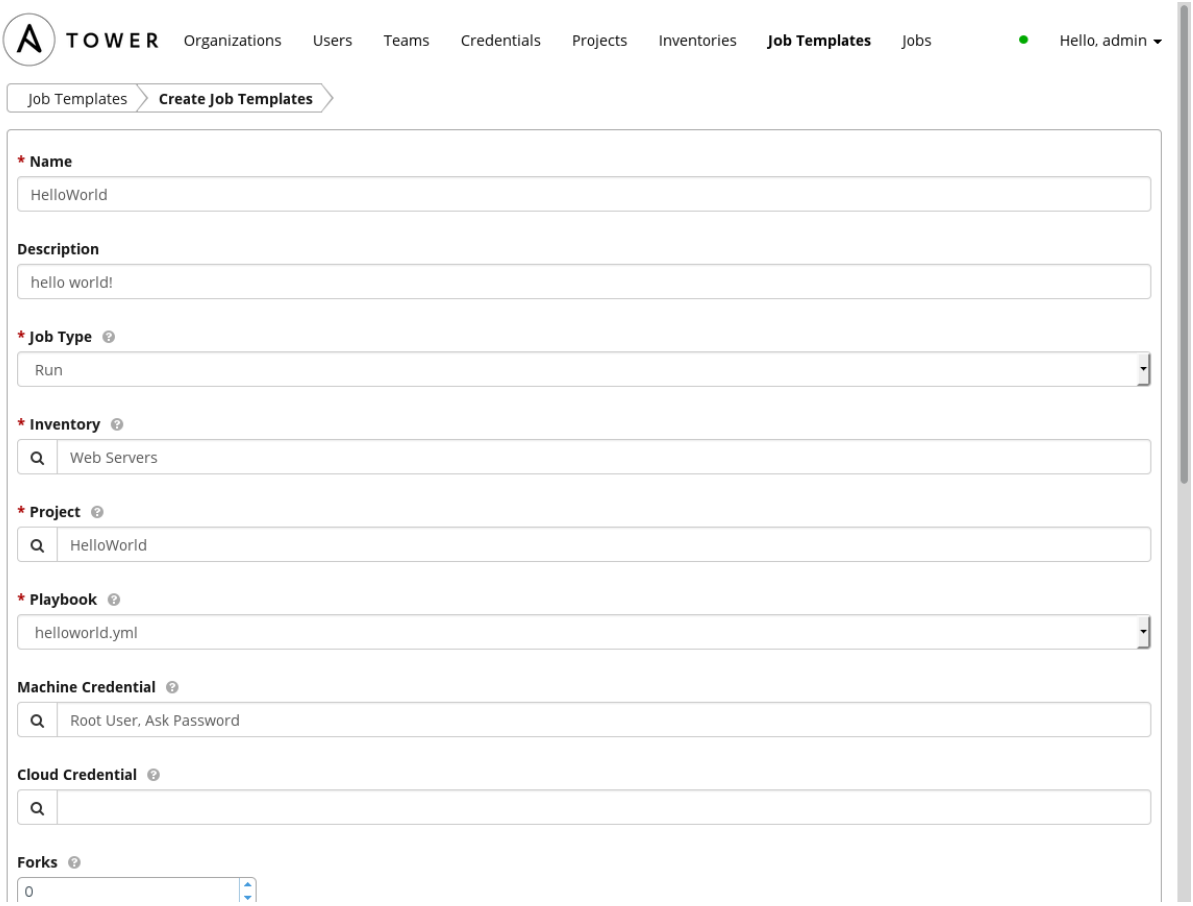
10. Create a new Job Template

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.

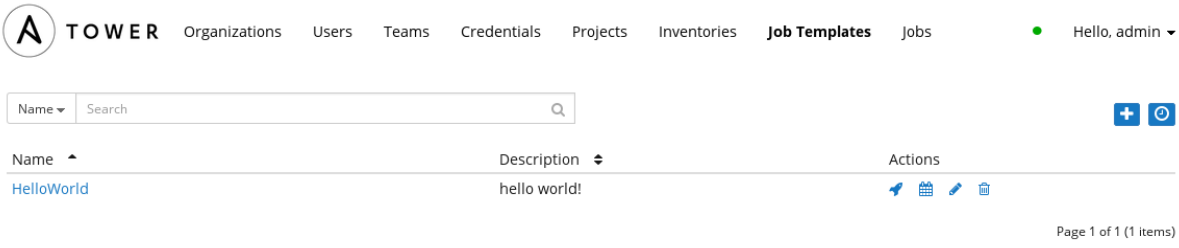


Enter values for the **Name** and **Description**. Jobs can be of type **Run** or **Check**. Select **Run** for this Quick Start (check corresponds to "dry run" mode.) Choose the **Inventory**, **Project**, and **Credential** from those we have created during this exercise. The playbook drop - down menu will automatically populate from the project path and playbook we created in step #5. Choose the **"helloworld"** playbook.

You can leave the other values, such as **Forks** and **Job Tags** set to their default values or blank. They'll be covered later under Job Templates.



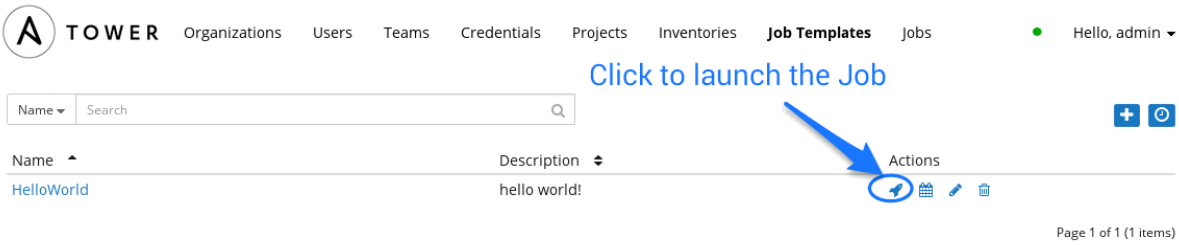
Click **Save**.



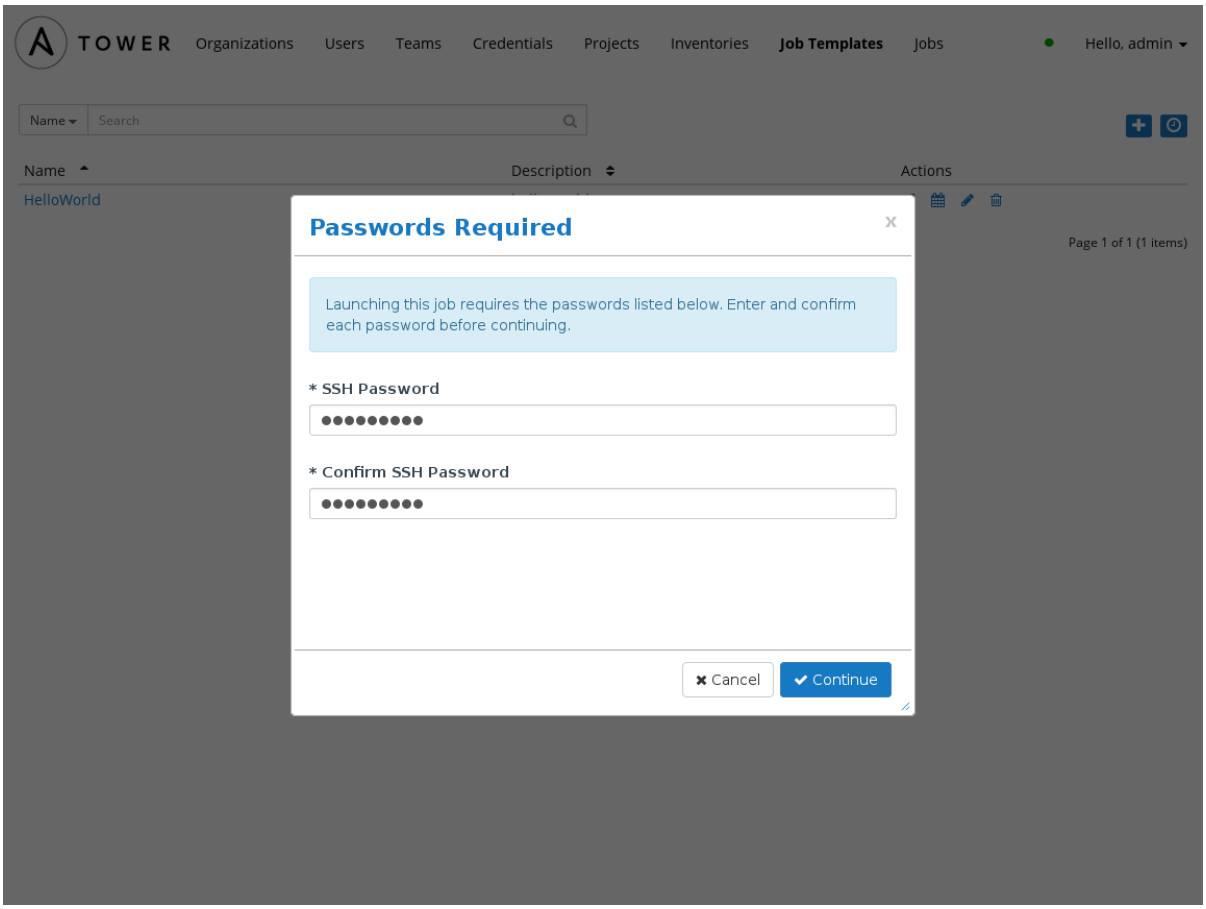
Now, let's launch the playbook and watch it all come together.

11. Launch it!

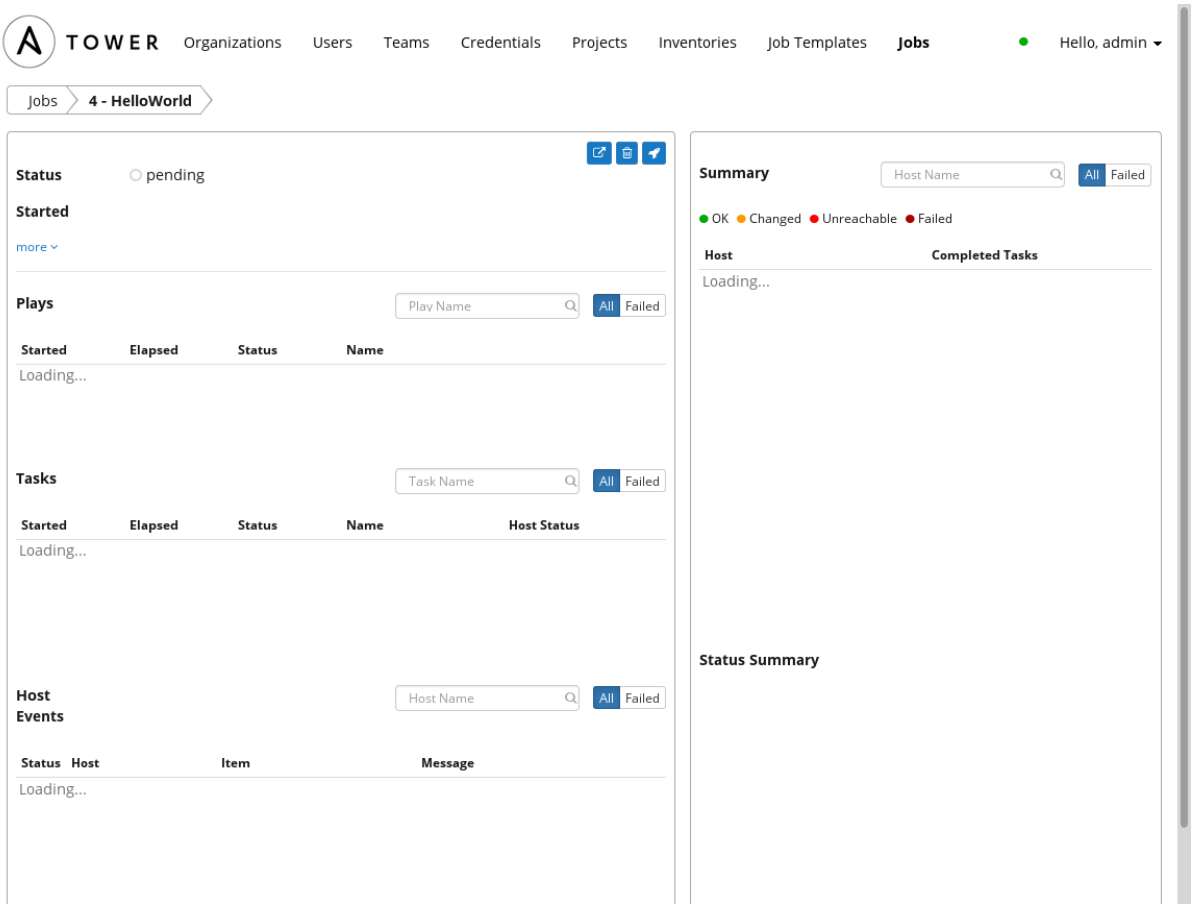
To launch the playbook, browse to the **Job Templates** tab and click **Launch** on the template.



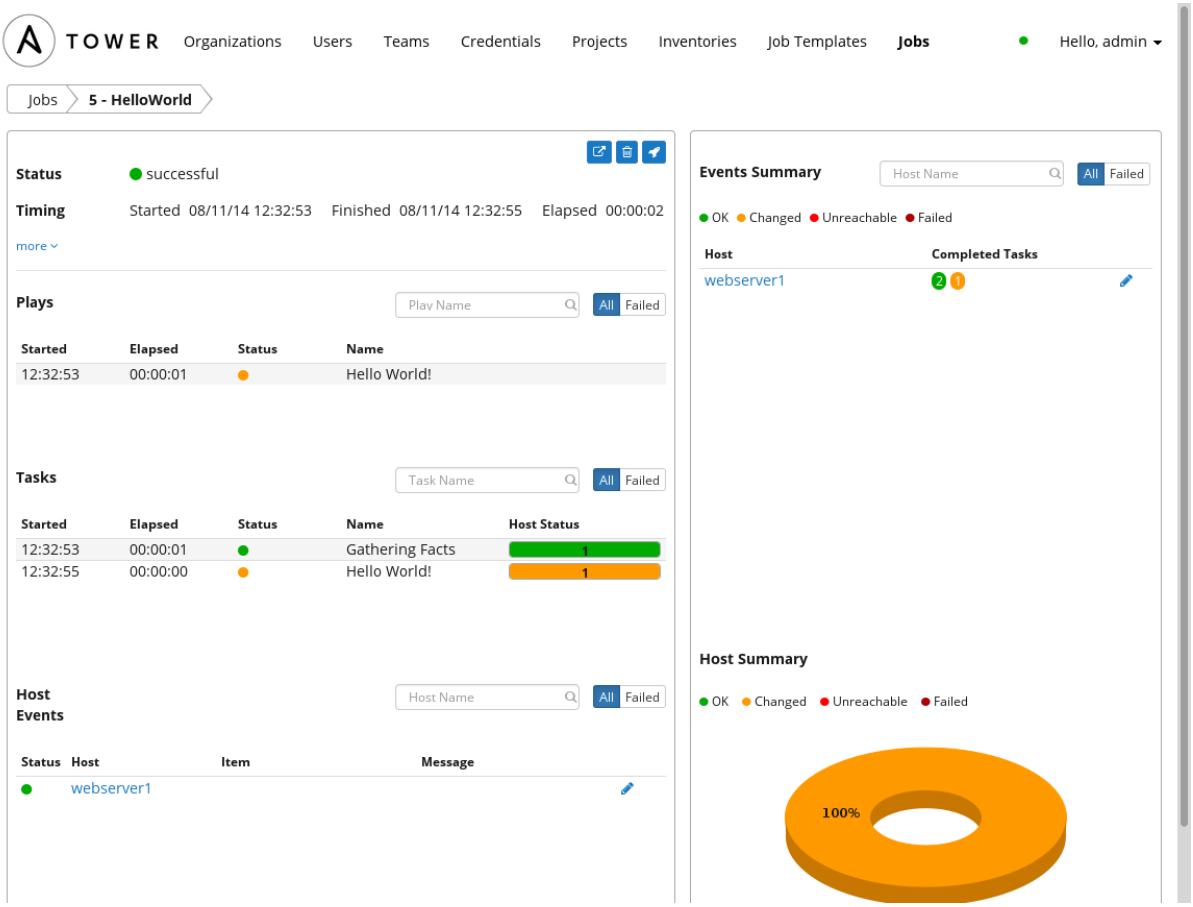
Tower will ask you for the SSH password, as we configured the credential.



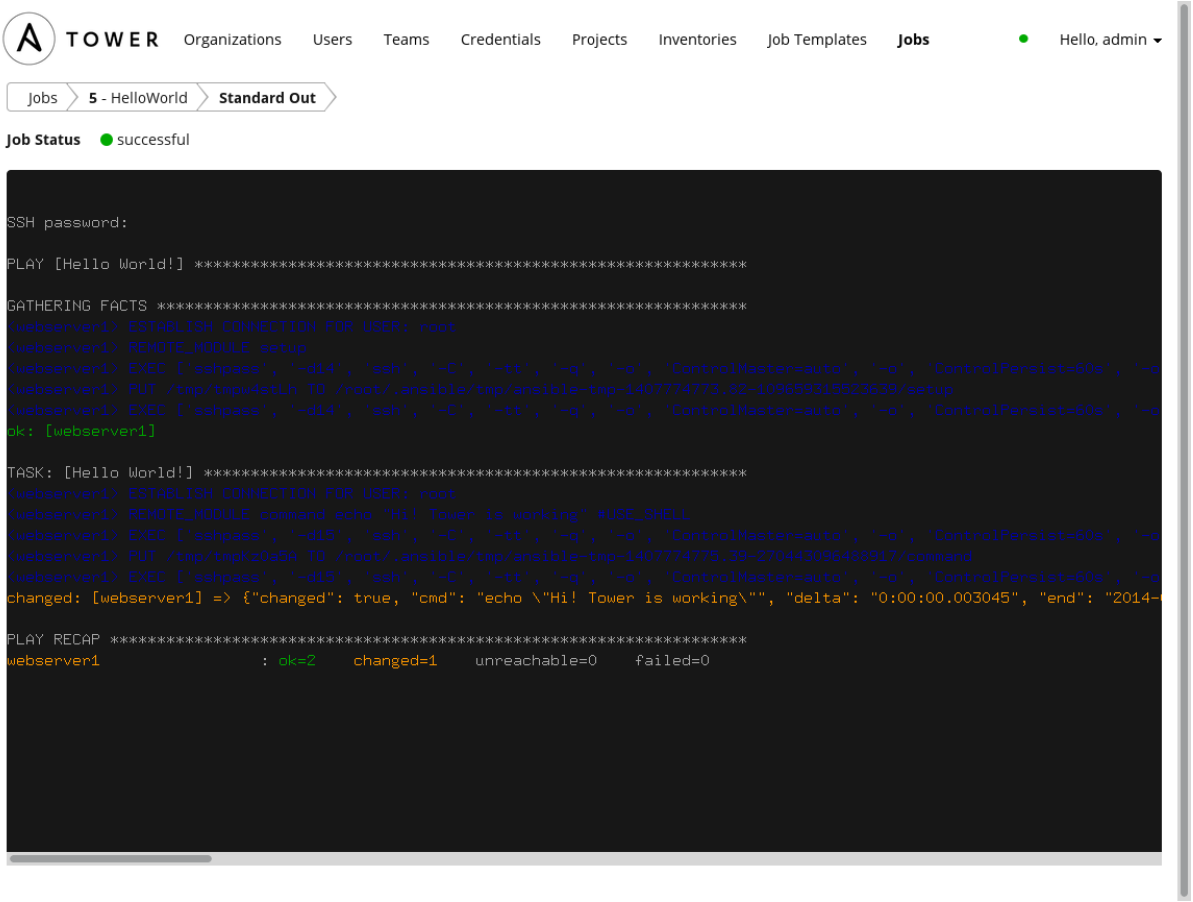
Tower will then redirect the browser to the status page for this job under **Jobs** tab, where you can watch this job as it runs.



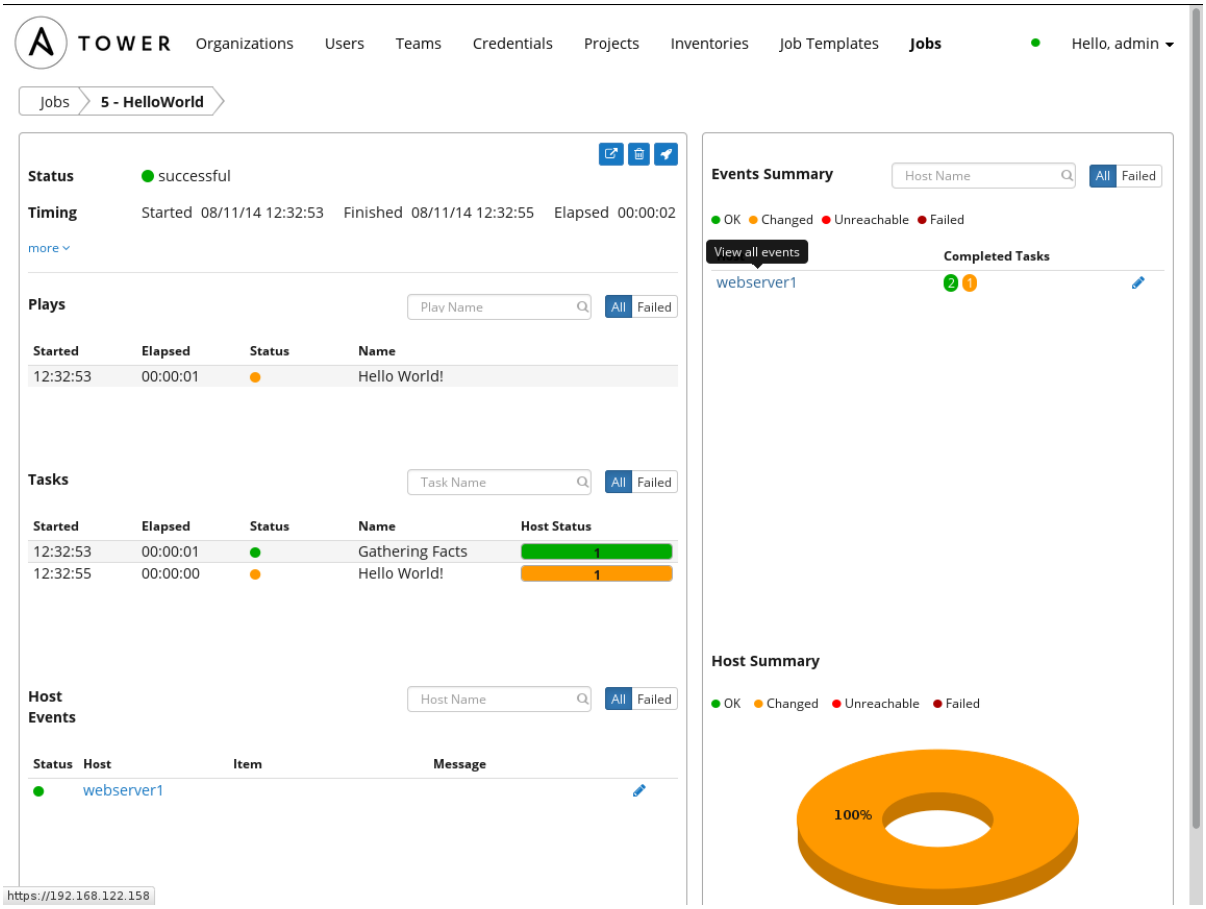
This page will automatically refresh using Tower's Live Event feature until the job is complete.



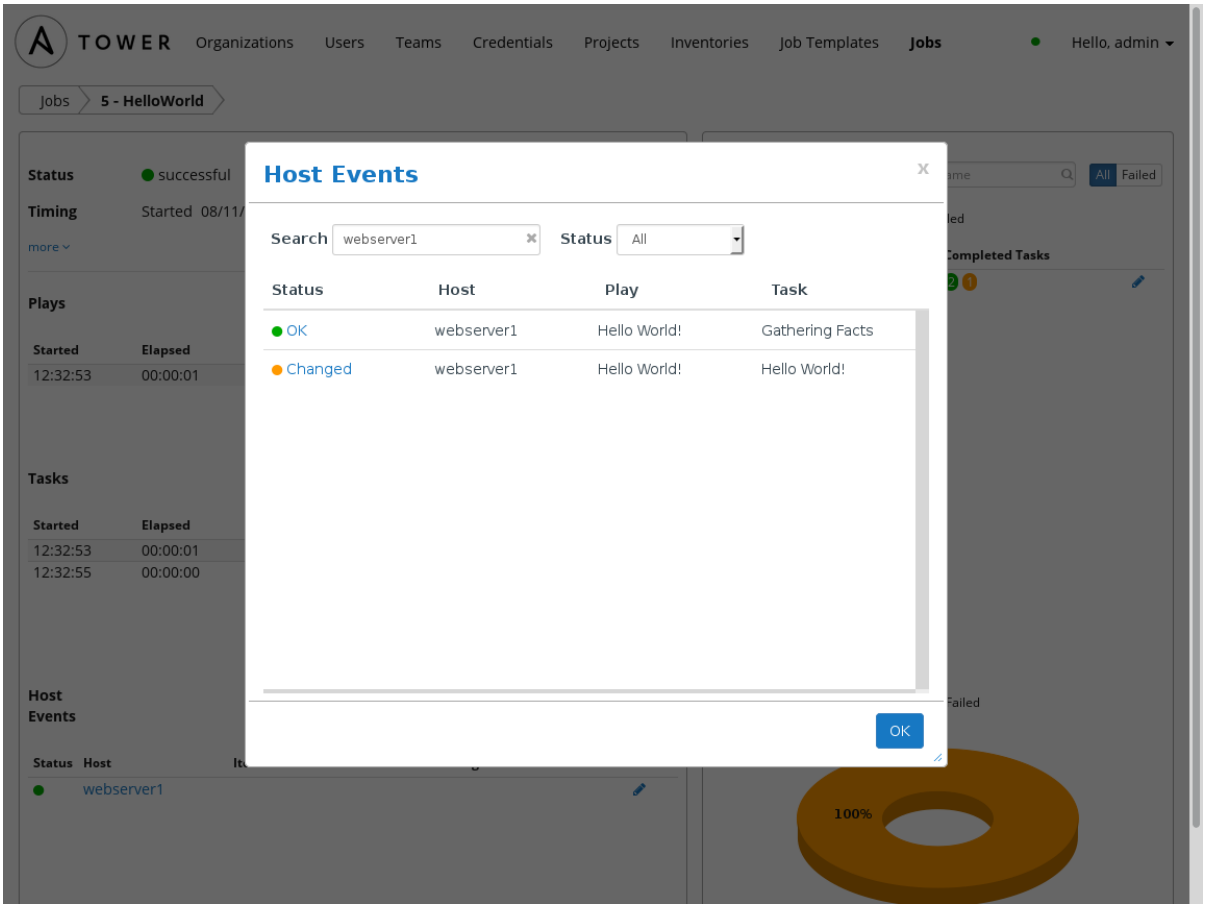
When the job has finished, click the button to view the standard output for the job.



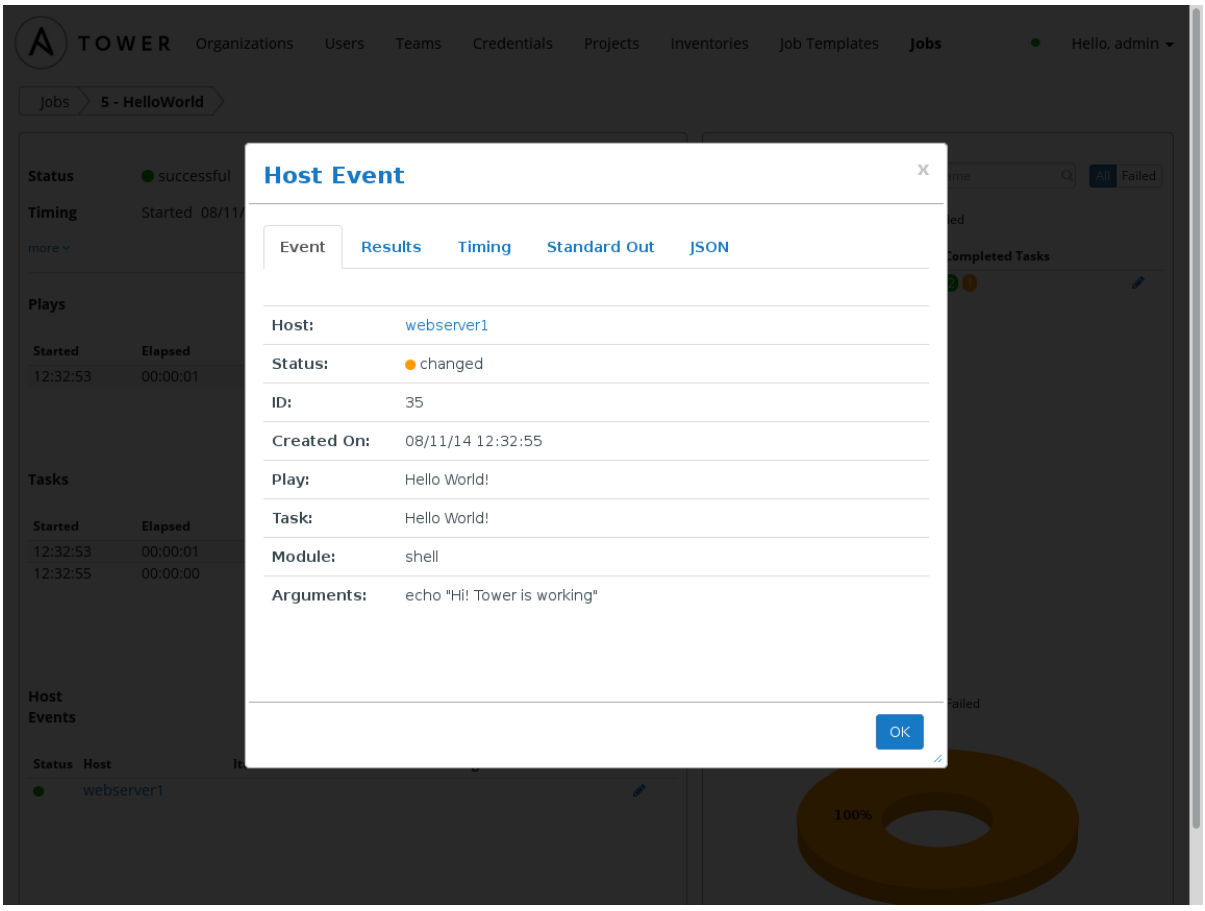
You can also drill down into individual tasks. Select the `webserver` host under **Summary**, to see job events for that host:



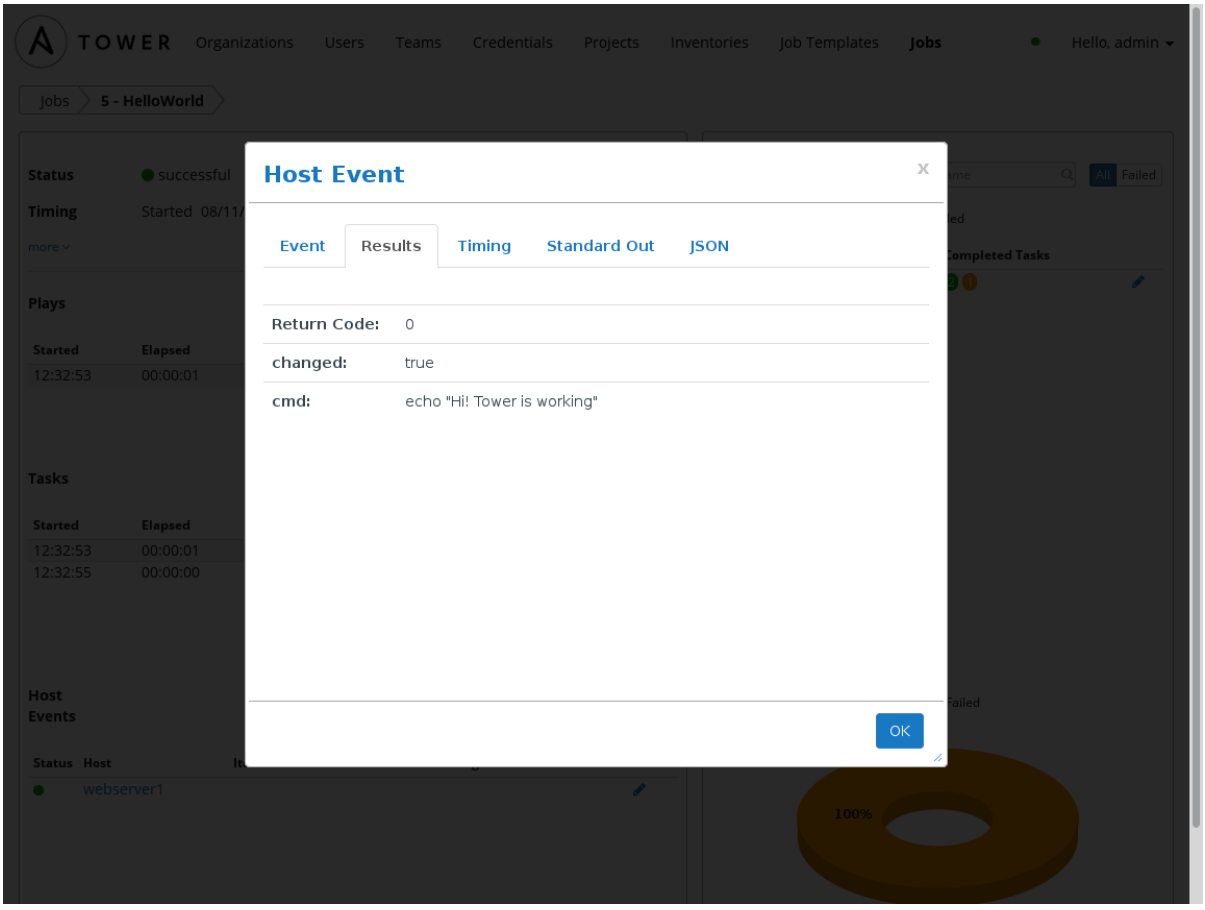
This screen will show us all of the events that resulted from running our playbook.



To show the event details, click on a particular event in the list:



Now, click **Results**:



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 http://support.ansible.com/. We appreciate your support!

User Guide

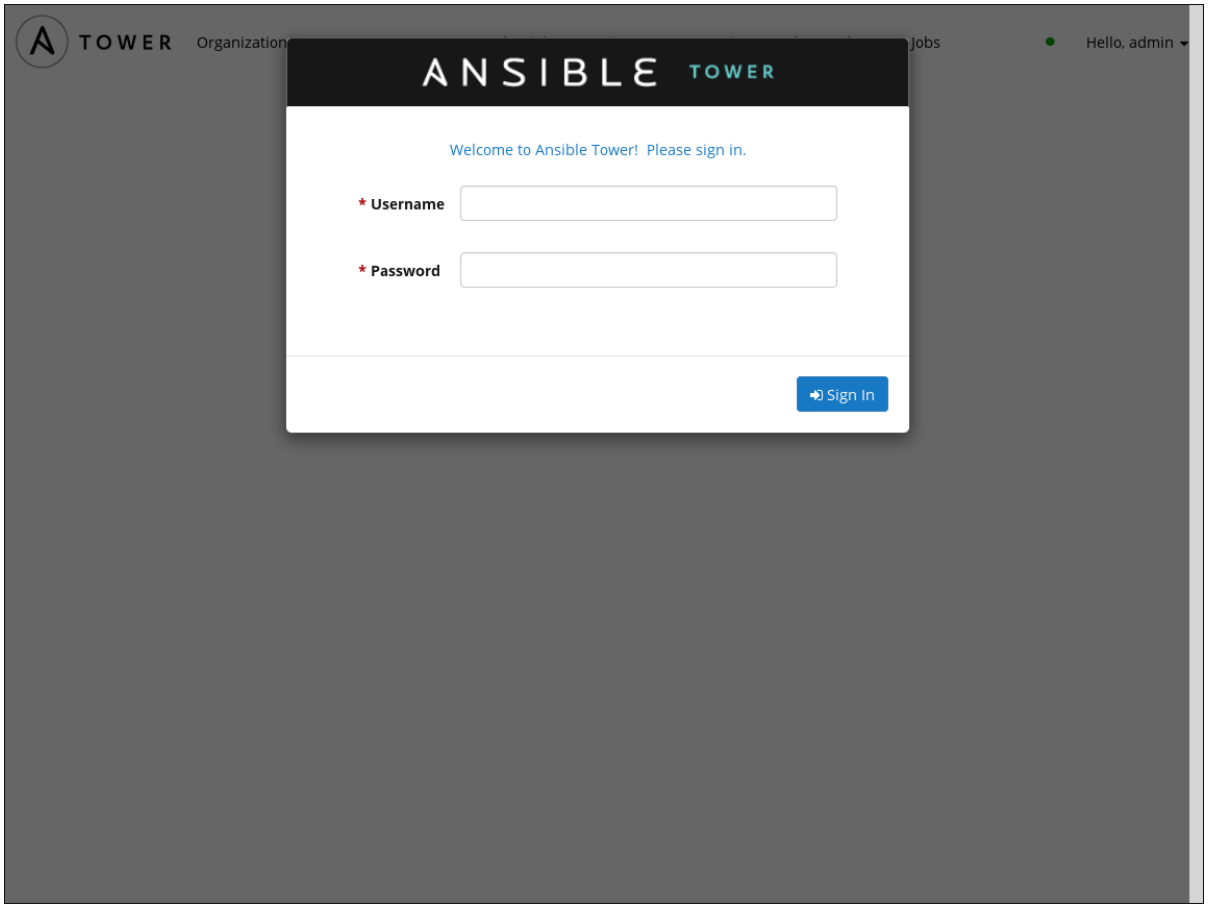
This section of the documentation will detail all of the functionality of Tower.

Ansible

This user guide assumes moderate familiarity with Ansible, including concepts such as **Playbooks**, **Variables**, and **Tags**. For more information on these and other Ansible concepts, please see the Ansible documentation at http://docs.ansible.com/.

Logging In

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



Log in using a valid Tower username and password.

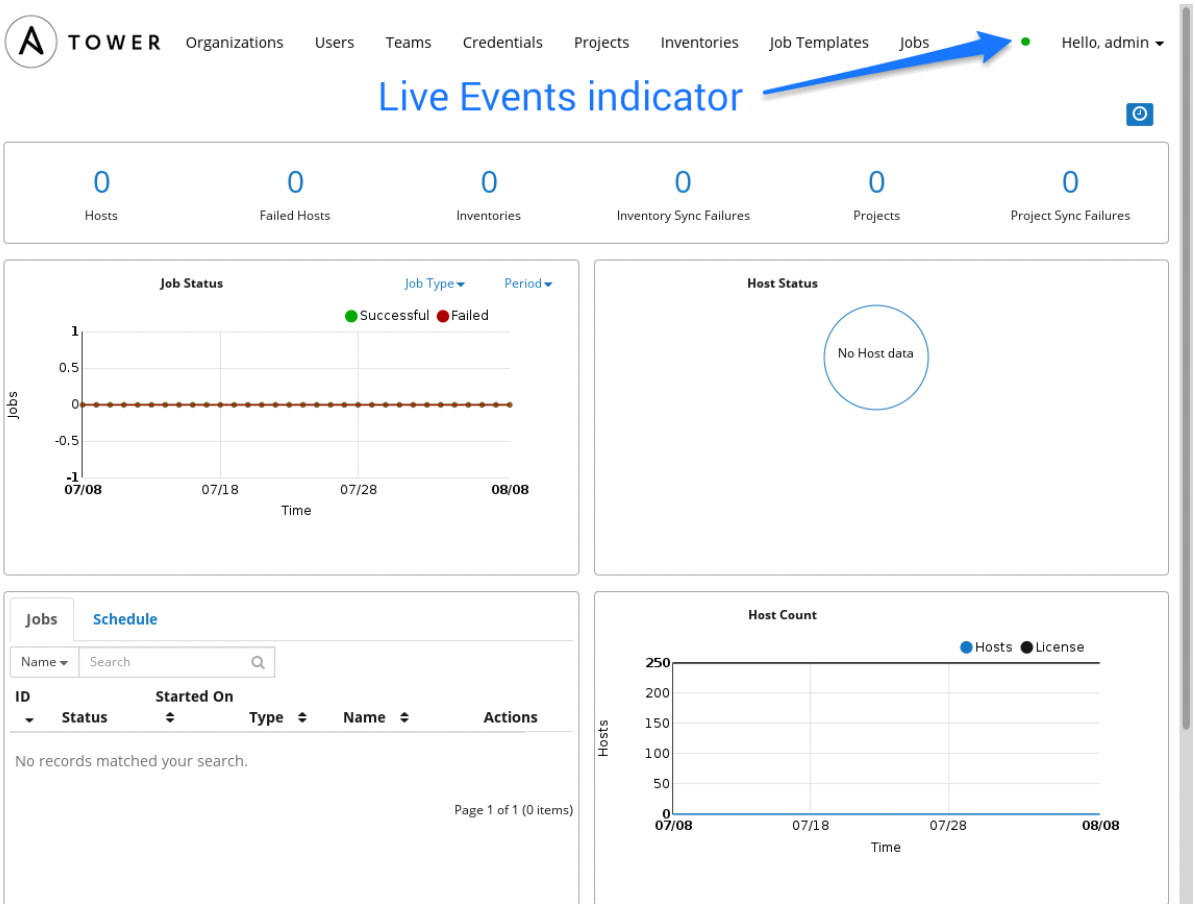
NOTE: *The default username and password set during installation are "admin" and "password", but the Tower administrator may have changed these settings during installation. If the default settings have not been changed, you can do so from the Users tab.*

Main Menu

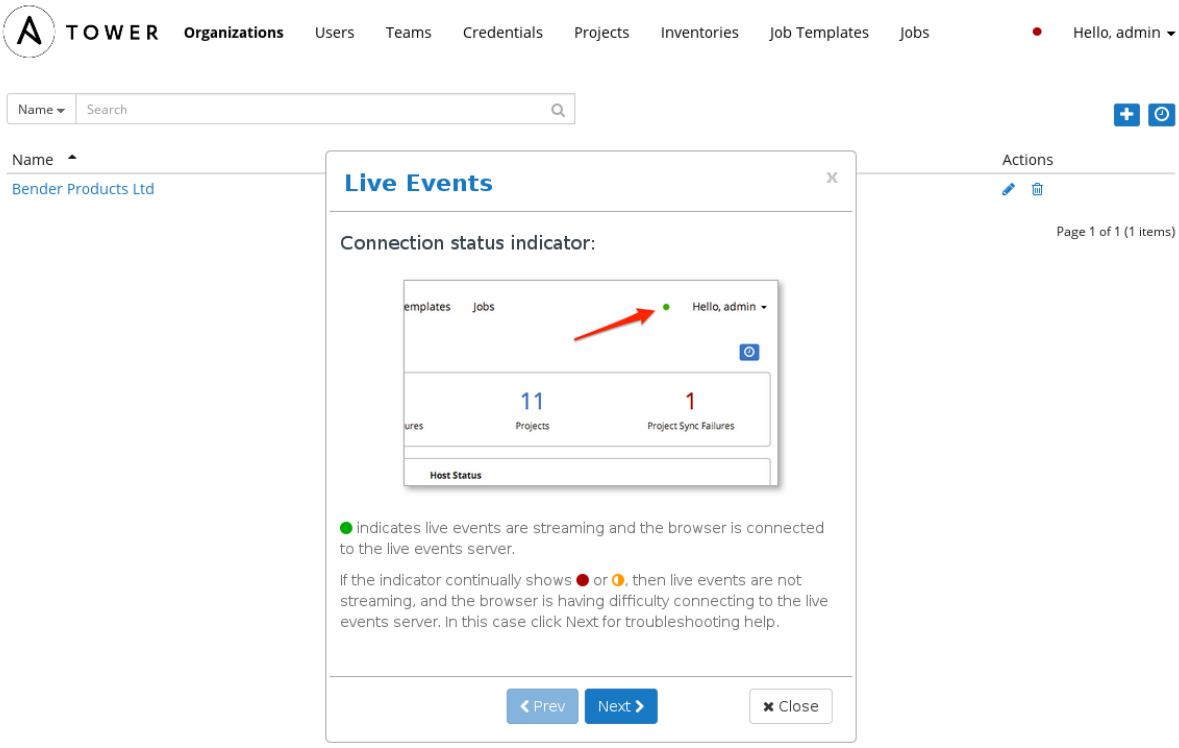
The top menu of Ansible provides quick links to the main aspects of Tower - **Organizations**, **Users**, **Teams**, **Credentials**, **Projects**, **Inventories**, **Job Templates**, **Jobs**. Each of these will be described in more detail below.
Clicking on the Ansible Tower logo at any time returns you to the Dashboard. There is also a status icon that displays the status of Tower Live Events, and the Tower User menu.

Live Events

Most pages in Tower, such as listing pages, or the job status, will automatically update as events happen, via Tower's *Live Events* feature. The status of Live Events is shown by a colored dot next to the Tower User's menu. This dot shows the status of Tower's *Live Events* functionality.



If this dot is green, all is well. If this dot is red or orange, Live Events are not available. Live Events are implemented by a "WebSocket" connection over HTTPS on port 8080. The most common cause of Live Events failures is the need to accept the certificate by visiting the web server via https:// on port 8080. If Live Events are not available because the red dot is displayed, click the dot to bring up the Live Events troubleshooting wizard. Follow the instructions there to configure Live Events.



If Live Events are not available, many pages will have a button, which can be used to refresh their contents.

Tower User Menu

The Tower User menu allows the Tower User to:

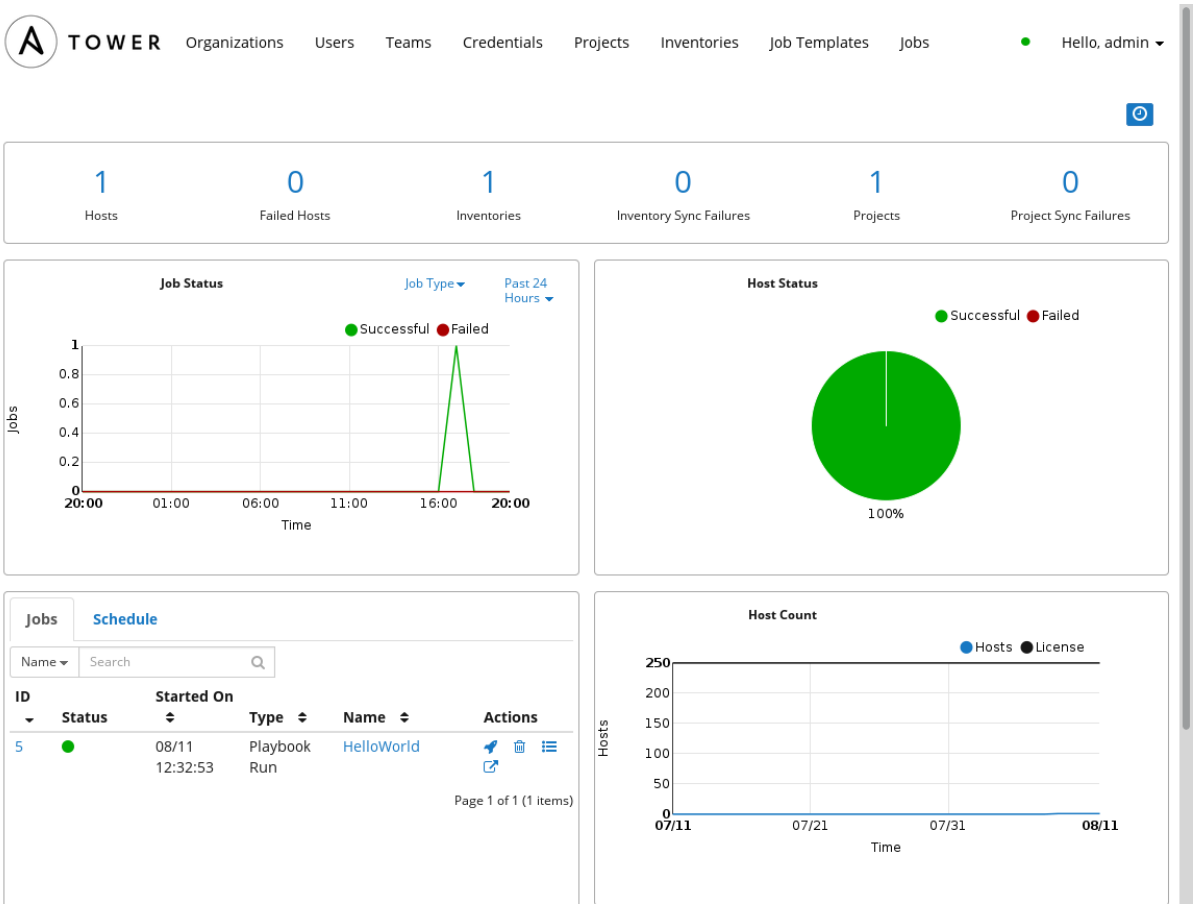
- See some basic information in **About Tower**
- Change the user's information in **Account Settings**
- Search the Ansible KnowledgeBase and file any issues via **Contact Support**
- Enter Tower's **Portal Mode**
- **View License** to view (and for Superusers, update) the Tower license
- **Logout**

If the user is a superuser, the user menu will also allow the Tower administrator to.

- Set up custom **Inventory Scripts** that can be used to dynamically sync their inventory.
- Run assorted **Management Jobs** to maintain their Tower instance.
- Examine assorted Tower runtime stats with **Monitor Tower**.

Dashboard

The central interface to Tower is the Dashboard.



In the upper right corner of the Dashboard is the button, which can be used to view the activity stream of all actions for the Tower installation. Most pages in Tower allow viewing an activity stream filtered for that specific object.

At the top of the Dashboard is a summary of your hosts, inventories, and projects. Each of these is linked to the corresponding object in Tower, for easy access.

The Dashboard contains four graphs.

Job Status

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

Host Status

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

Job and Schedule Status

The Jobs tab of this display shows a summary of the most recently completed jobs. It is the same summary you will see if you click on the **Jobs** entry in the menu.

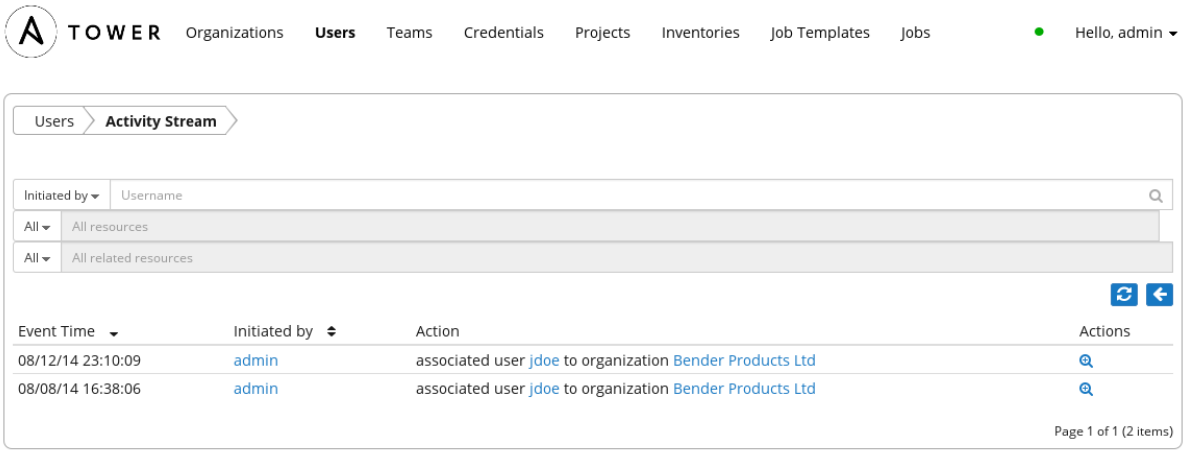
The Schedule tab of this display shows upcoming scheduled jobs. It is the same summary you will see if you click on the **Jobs** entry in the menu and look at the Scheduled table.

Host Count

The Host Count graph shows the number of managed hosts, over time. It can be used to determine whether your Tower installation is coming close to its license capacity. This graph is only shown to Superusers.

Activity Streams

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



An Activity Stream shows all changes for a particular object. For each change, the Activity Stream shows the time of the event, the user that initiated the event, and the action. Clicking on the button shows the event log for the change.

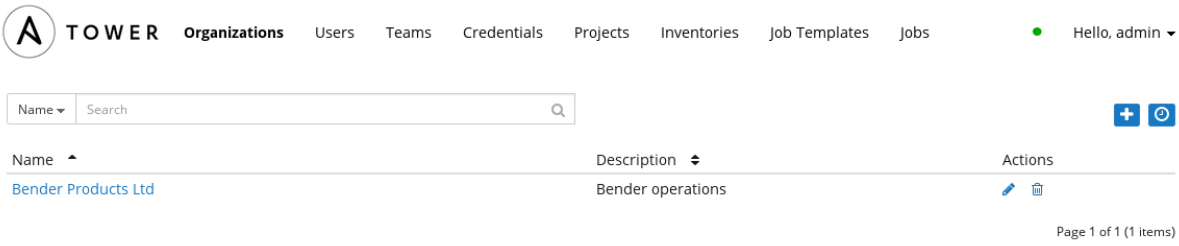
The Activity Stream can be filtered by the initiating user (or the system, if it was system initiated), and by any related Tower object, such as a particular credential, job template, or schedule.

The Activity Stream on the Dashboard shows the Activity Stream for the entire Tower instance.

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.

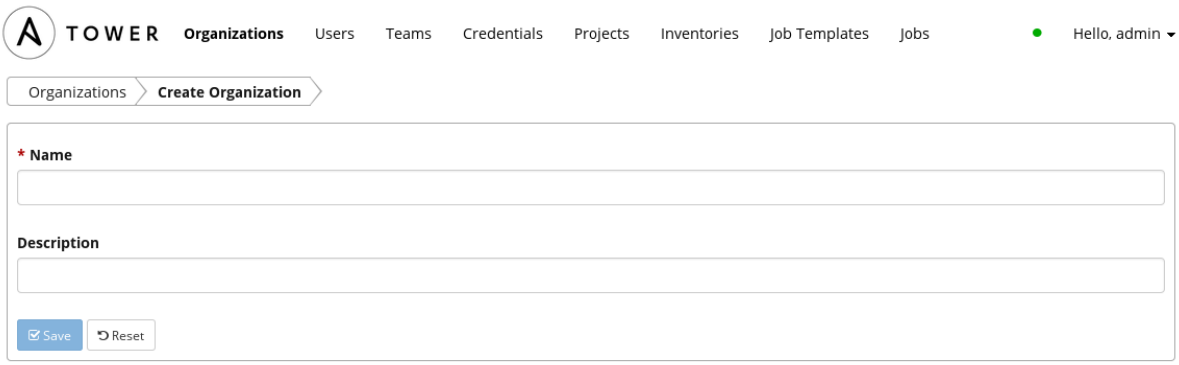


Buttons located in the upper right corner of the **Organizations** tab provide the following actions:

- Create a new organization
- View Activity Stream

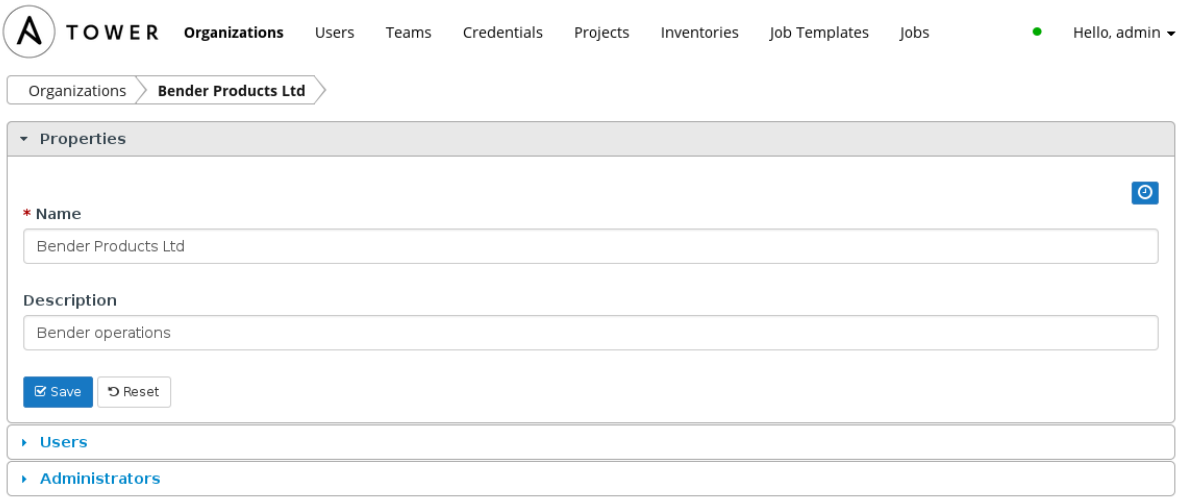
Create a new organization by selecting the  button.

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



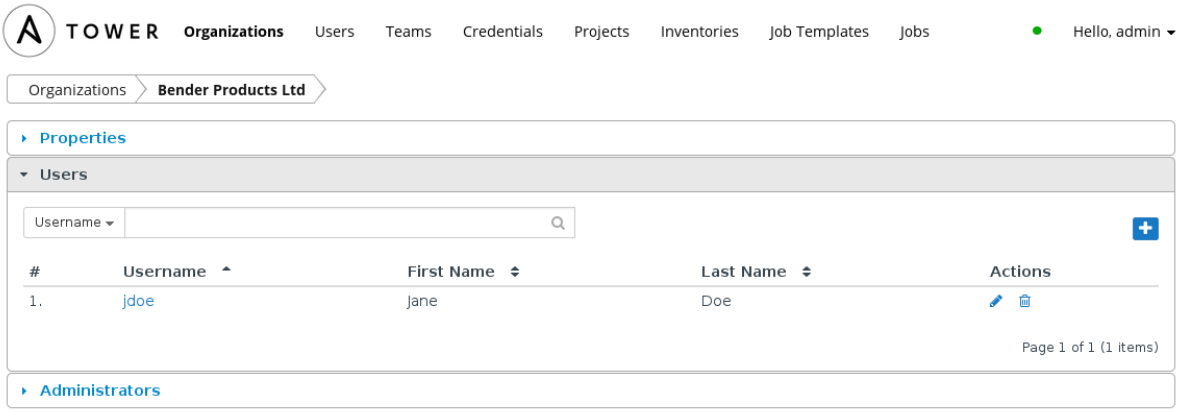
Click **Save** to finish creating the organization.

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.



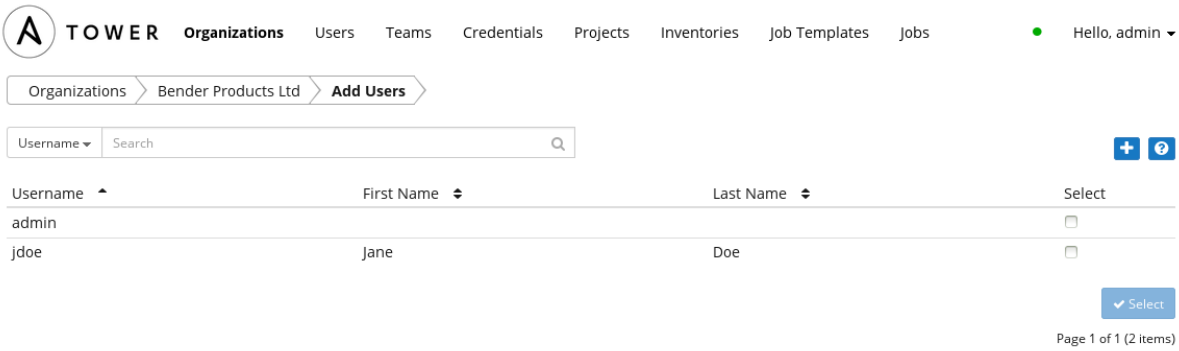
Organizations - Users

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

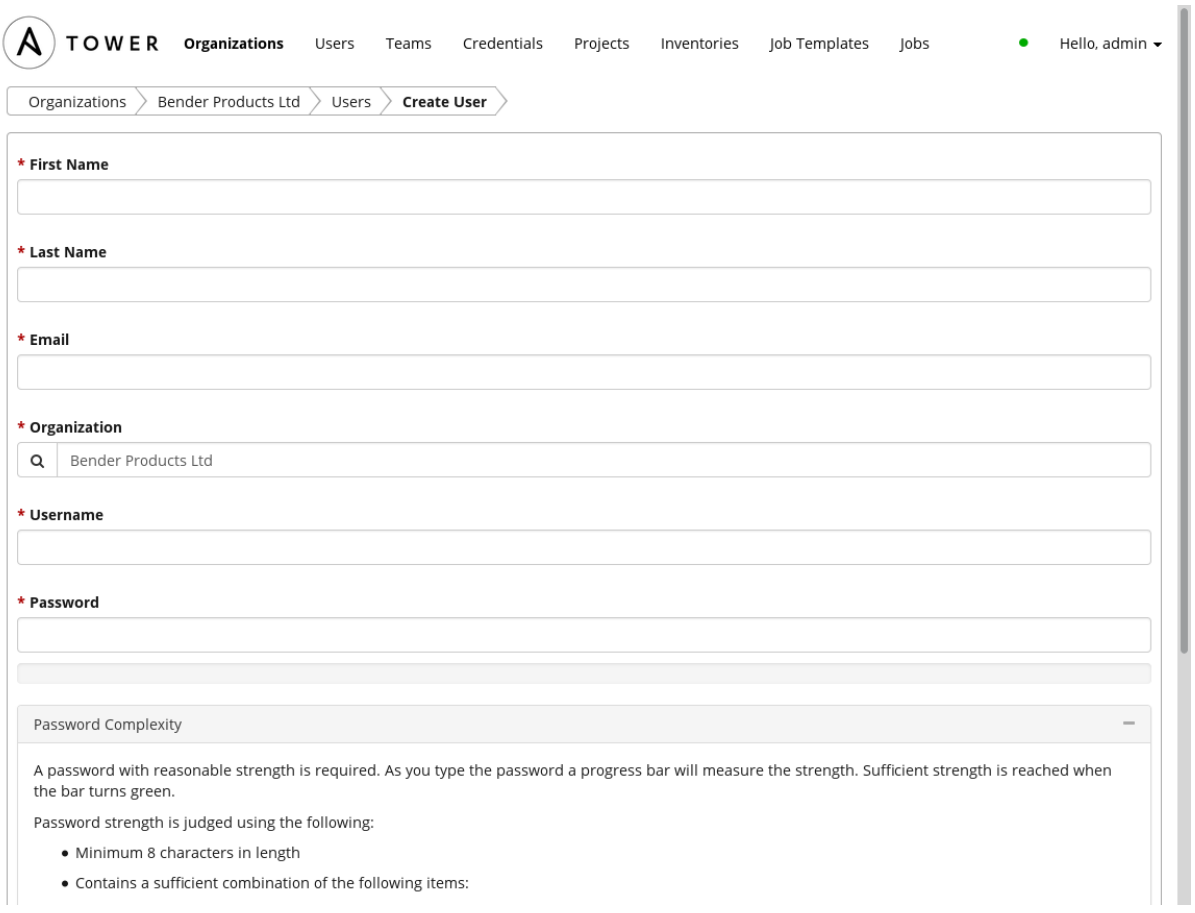


This menu allows you to manage the user membership for this organization. (User membership may also be managed on a per-user basis via the **Users** tab.) The user list may be sorted and searched by **Username**, **First Name**, or **Last Name**. Existing users may also be modified and removed using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

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



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



.

Enter the appropriate details into the following fields:

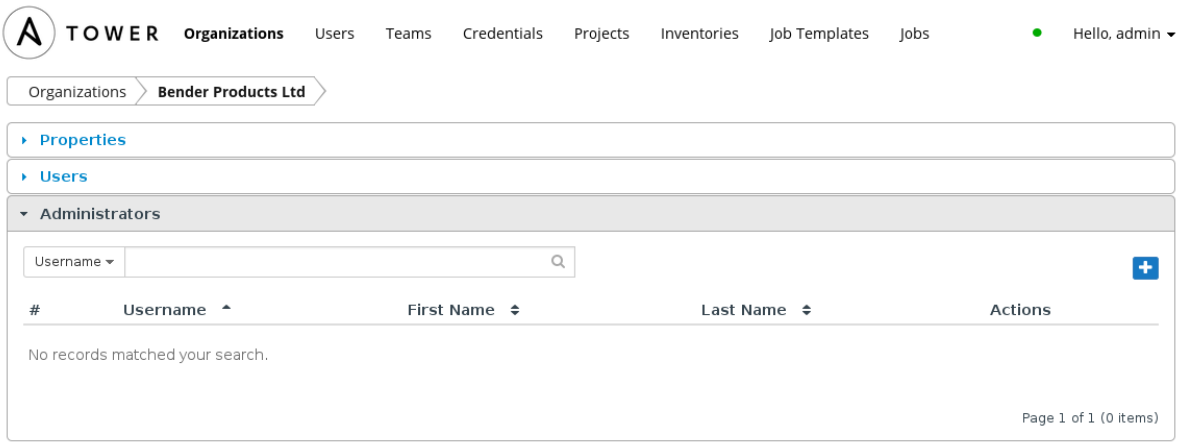
- First Name

- Last Name
- Email
- Organization (will be prefilled with the current Organization)
- Username
- Password
- Confirm Password
- Superuser? (Give this user Superuser 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

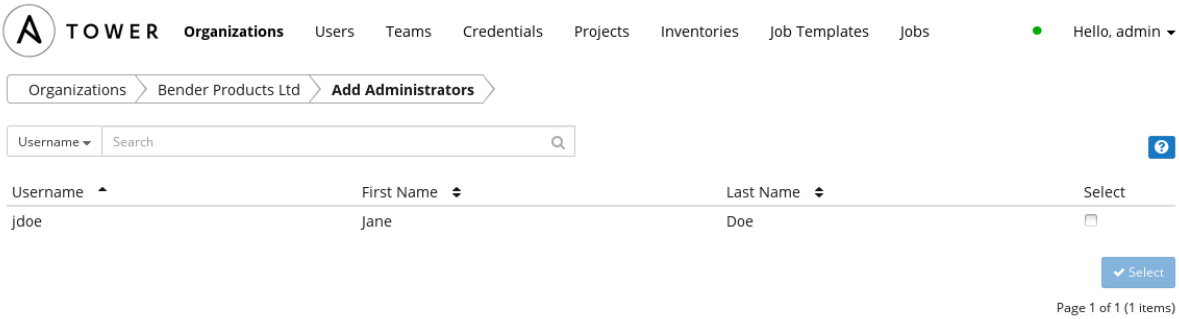
An organization administrator is a type of user that has the rights to create, modify, or delete objects in the organization, including projects, teams, and users in that organization. Expand the **Administrators** menu by selecting **Administrators**.



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**. Note that any user marked as a 'Superuser' is implicitly an administrator of all organizations, and is not displayed here.

To add an administrator to the organization, click the  button.

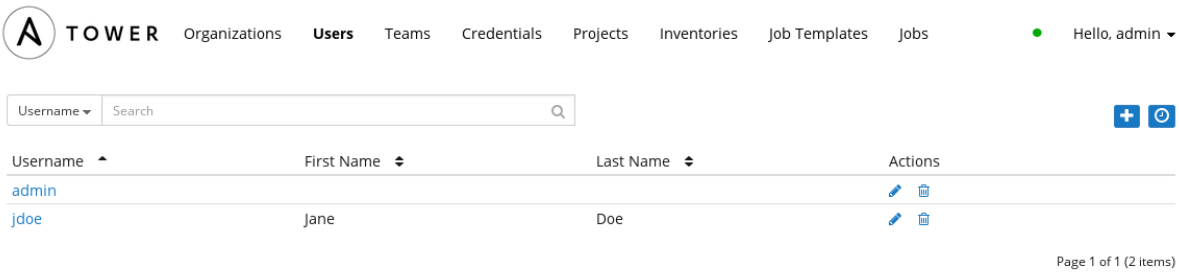
Then, select one or more users from the list of available users by clicking the **Select** checkbox or clicking anywhere on the user row. Click the **Select** button when done.



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**.



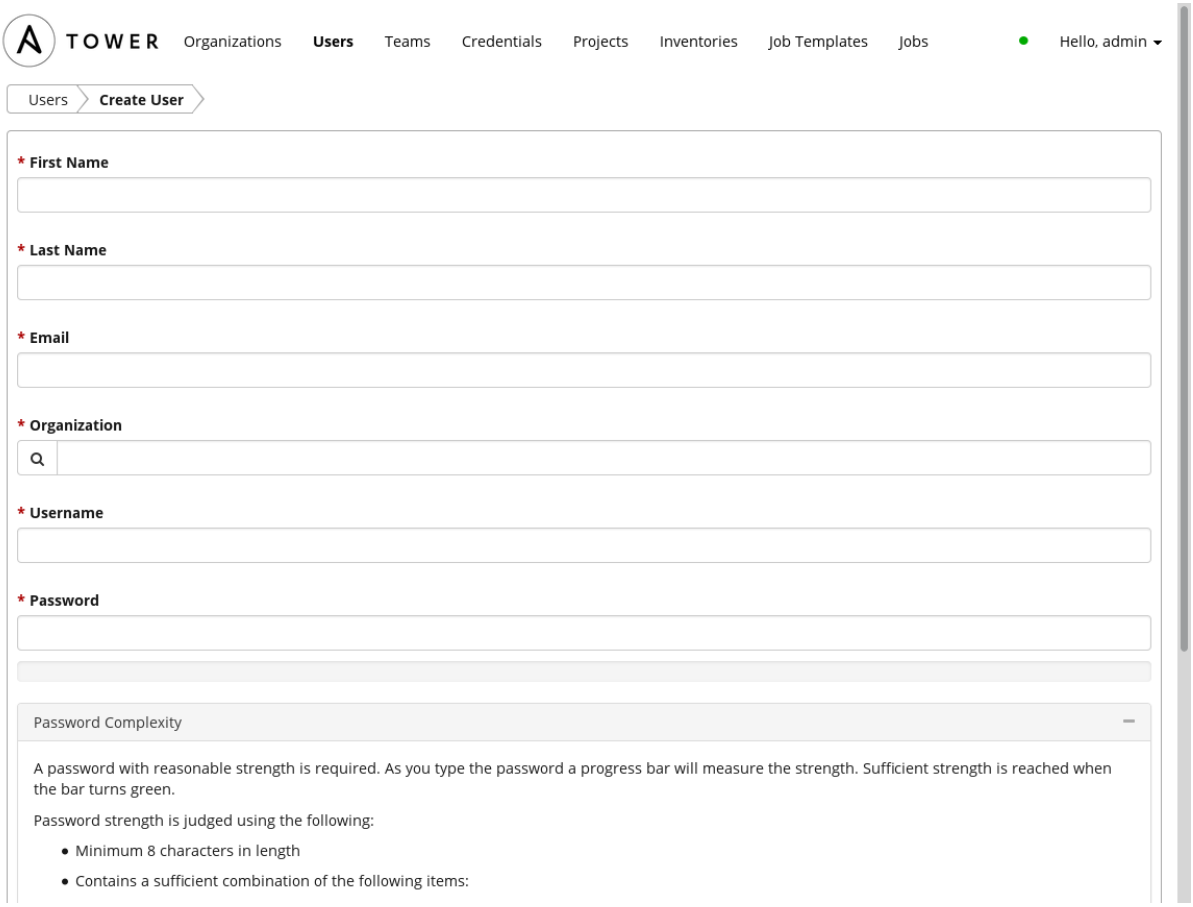
There are three types of Tower Users:

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. **Superuser**: a Tower Superuser has admin, read, and write permissions over the entire Tower installation. A Superuser is typically a systems administrator responsible for managing Tower and will delegate responsibilities for day-to-day work to various Organization Administrators.

NOTE: *The initial user (usually "admin") created by the Tower installation process is a Superuser. One Superuser must always exist, so if you wish to delete "admin", first create another Superuser account.*

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

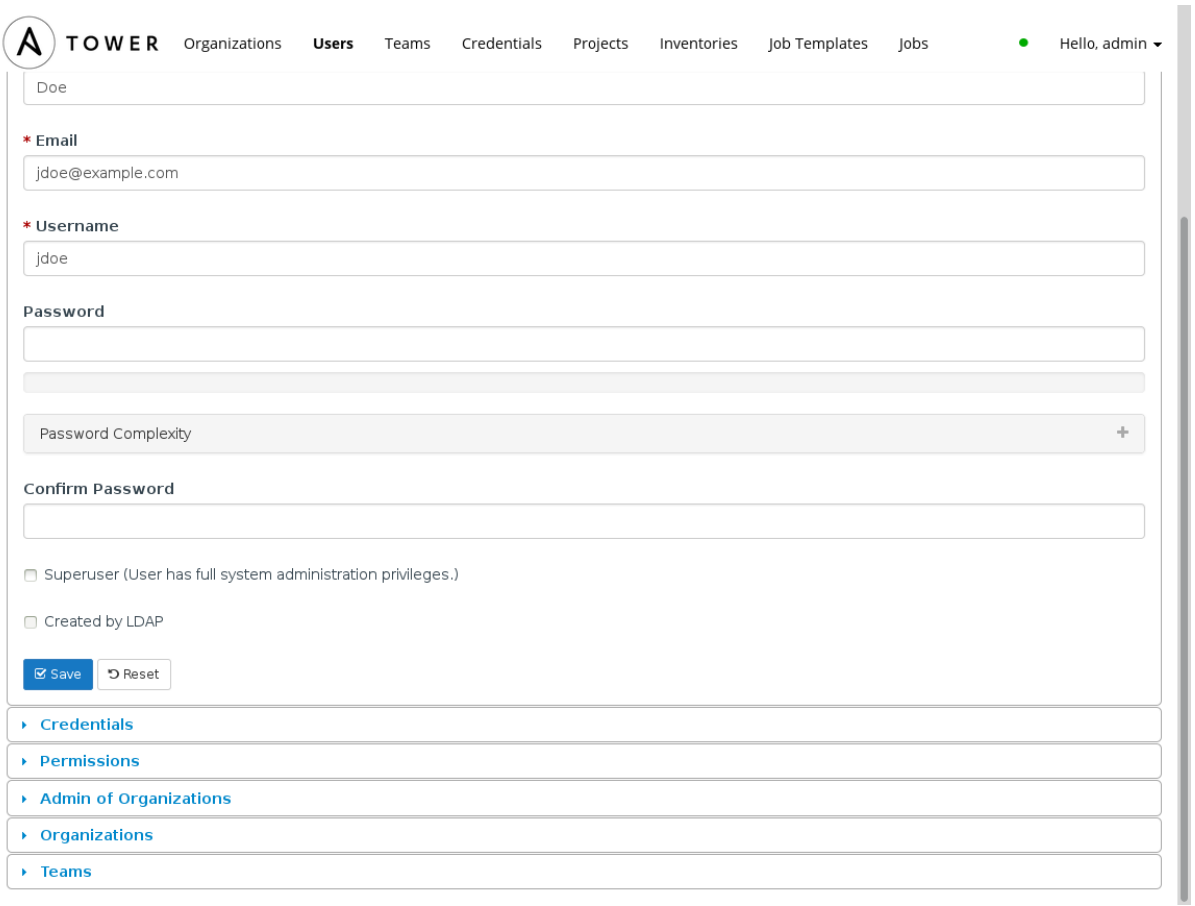


Enter the appropriate details into the following fields:

- 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.

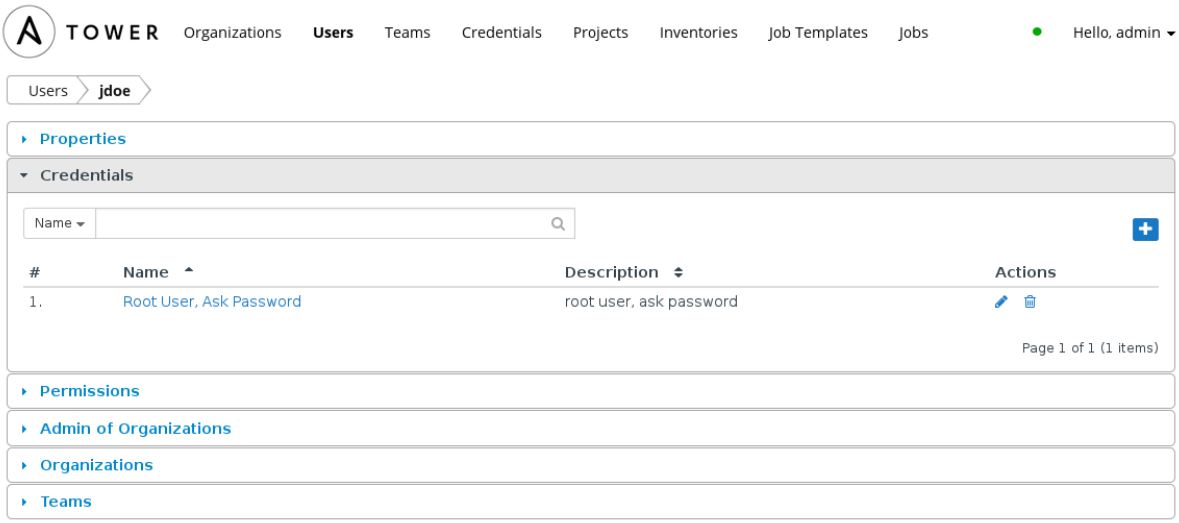
Once the user is successfully created, Tower will open the **Edit User** dialog. This is the same menu that is opened if the **Edit** button is clicked from the **Users** tab. Here, **User Setting**, **Credentials**, **Permissions**, and other user membership details may be reviewed and modified.



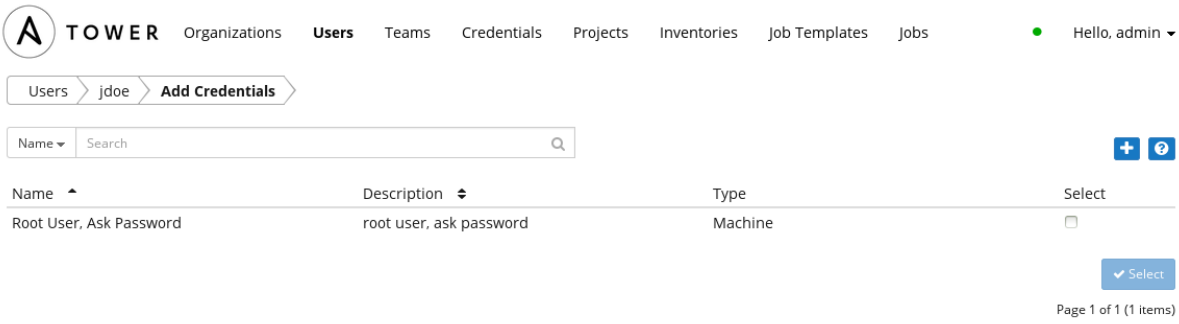
Users - Credentials

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.

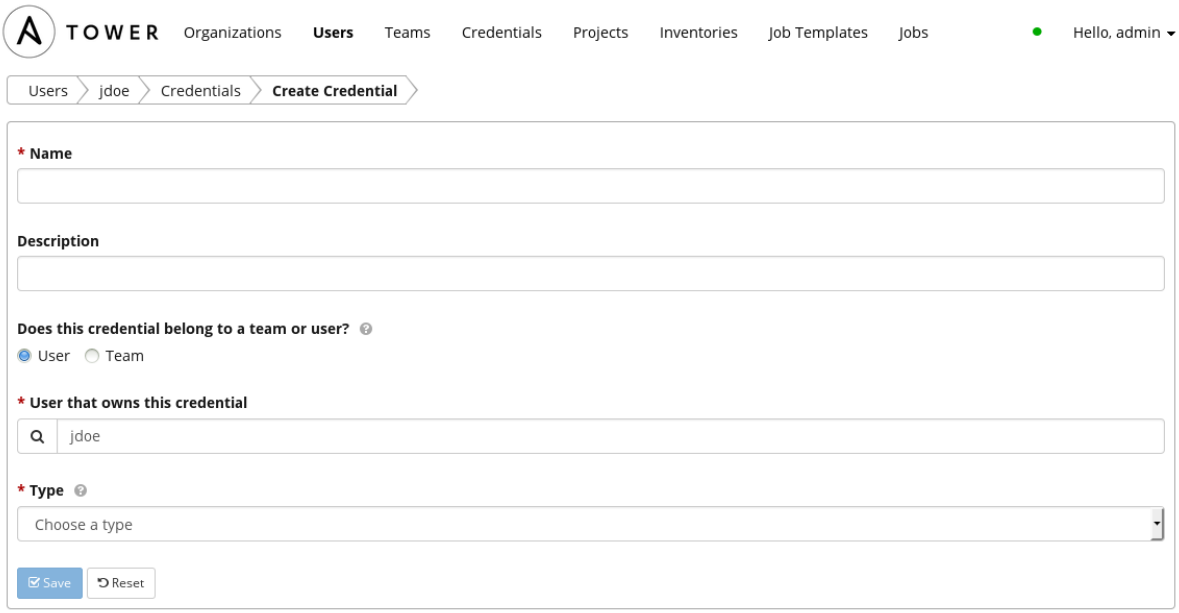
To add a credential to user, expand the credentials menu and click the + button.



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



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.



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

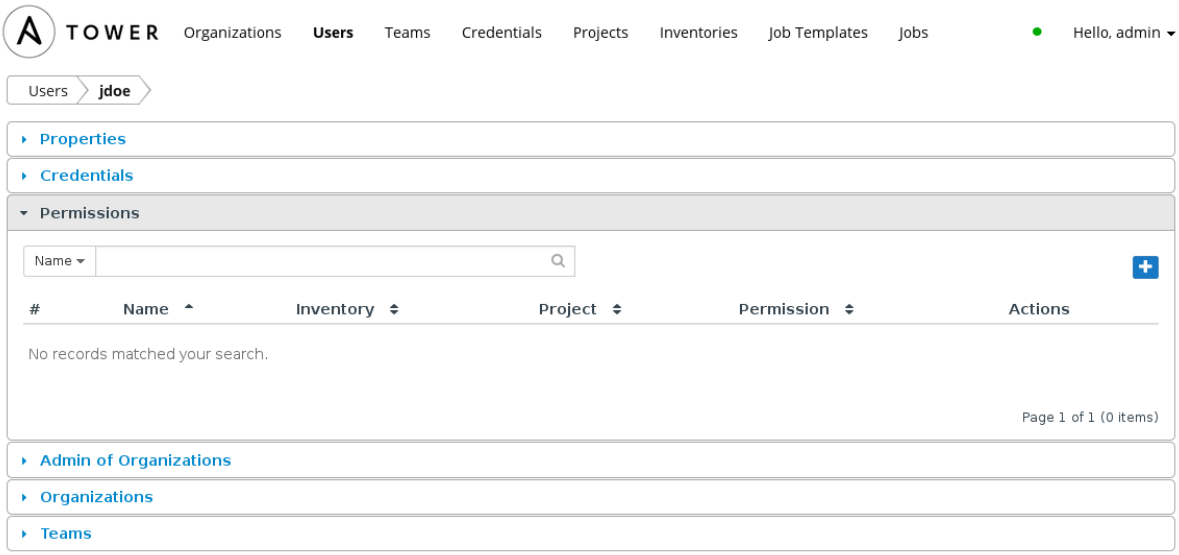
Permissions are the set of privileges assigned to users and teams that provide the ability to read, modify, and administer projects, inventories, and other Tower elements.

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

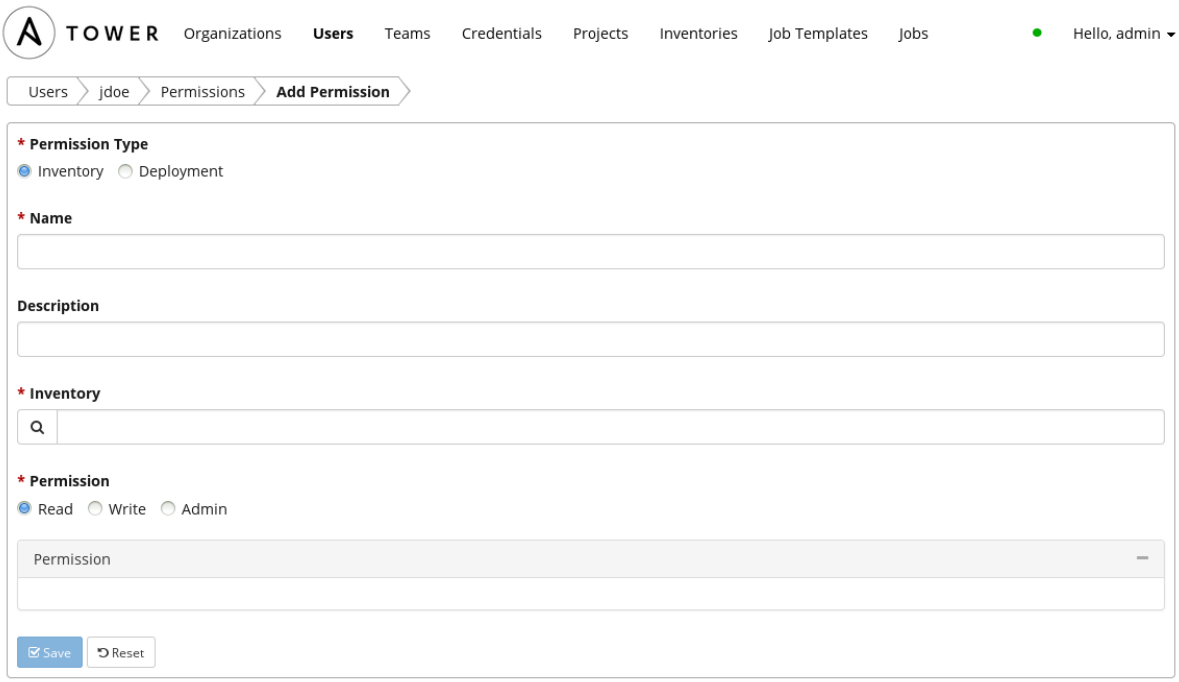
- Inventory: grants permission to act on inventories, groups, and hosts
  - Admin: modify the settings for the specified inventory. This permission also grants Read and Write permissions.
  - Read: view groups and hosts within a specified inventory
  - Write: create, modify, and remove groups, and hosts 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.

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.



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



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

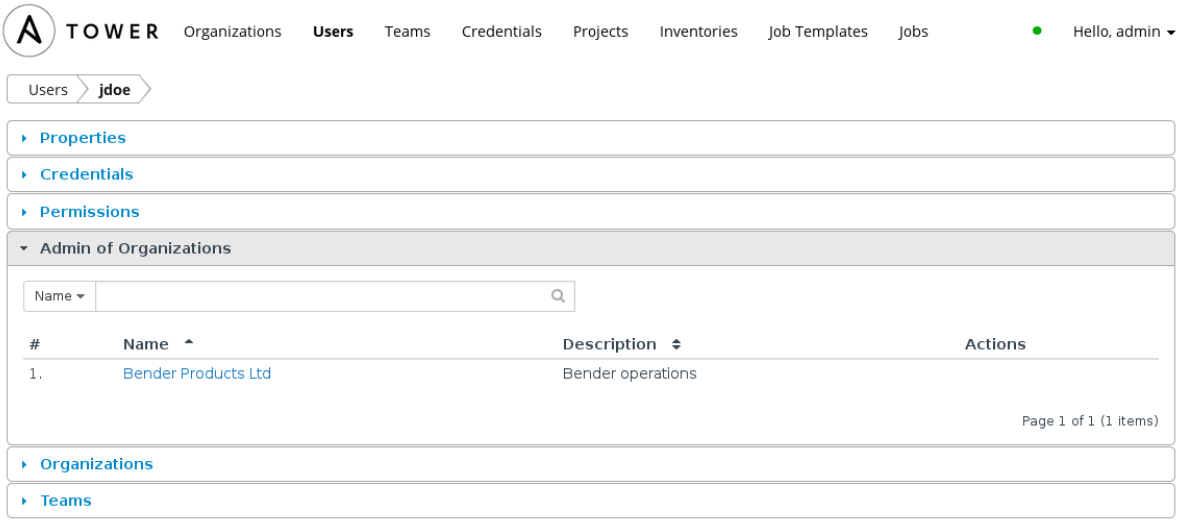
For a permission of type **Deployment**, enter the following details:

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

Select **Save**.

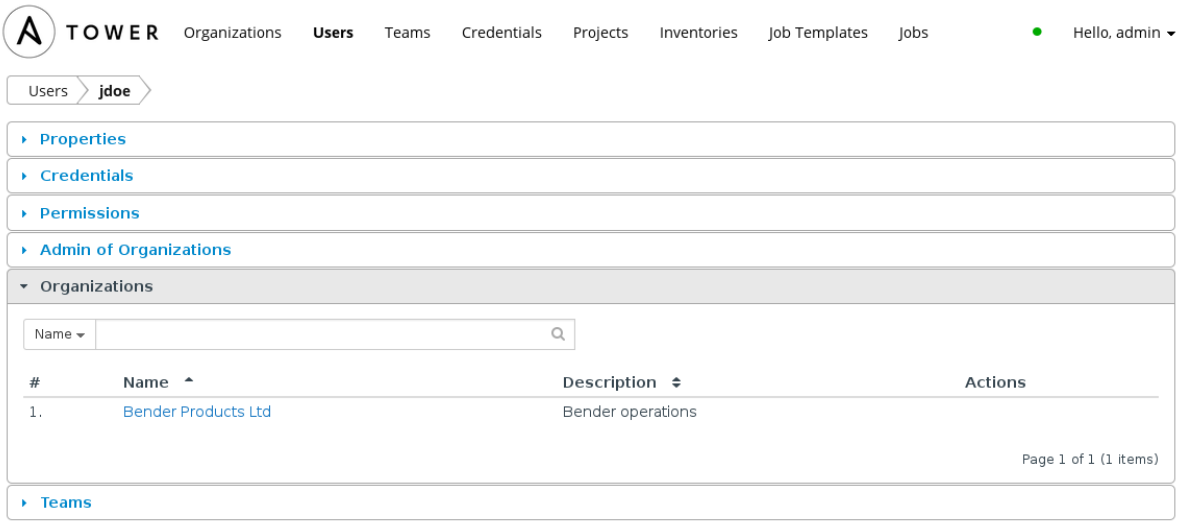
Users - Admin of Organizations

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.



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.



Users - Teams

This displays the list of teams that this user is a member of. This list may be searched by **Team Name** or **Description**. Team membership cannot be modified from this display panel. For more information, see the Teams section.

Teams

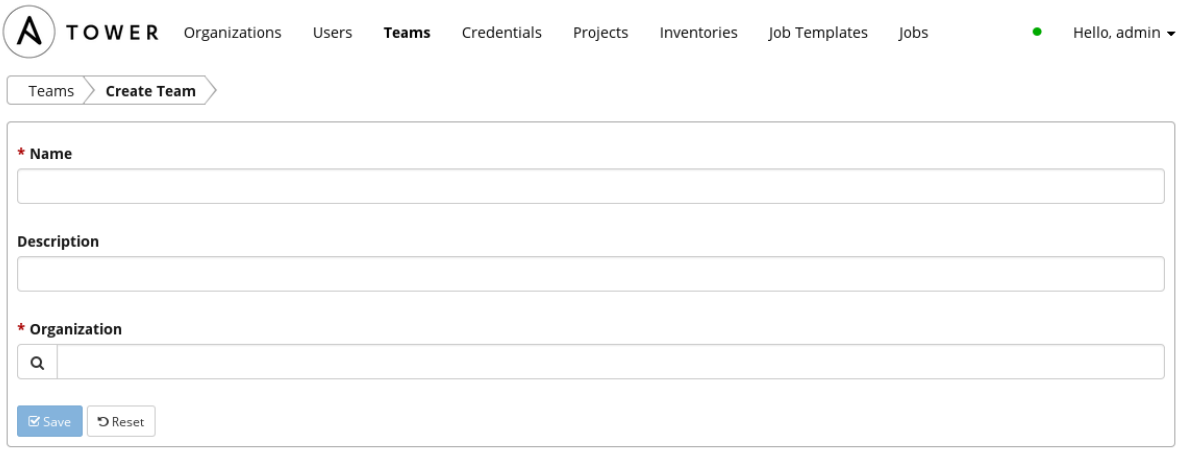
A team is a subdivision of an organization with associated users, projects, credentials, and permissions. Teams provide a means to implement role-based access control schemes and delegate responsibilities across organizations. For instance, permissions may be granted to a whole team rather than each user on the team.

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

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

- Create a new team
- View Activity Stream

To create a new team, click the button.



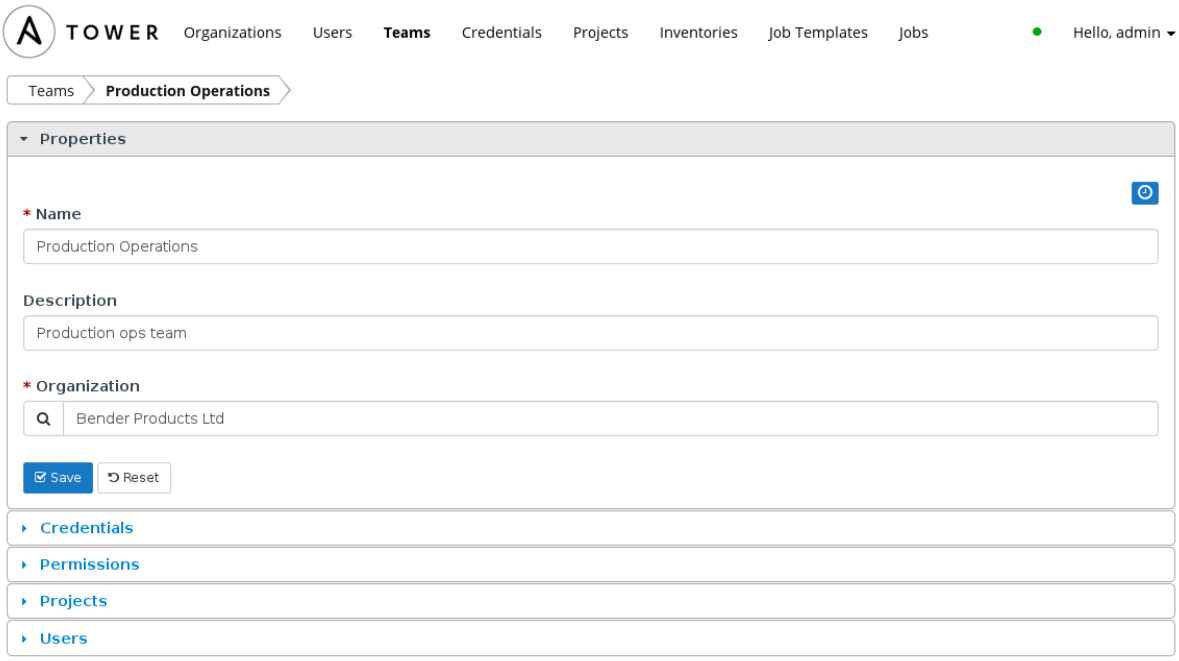
Enter the appropriate details into the following fields:

- Name
- Description

- Organization (Choose from an existing organization)

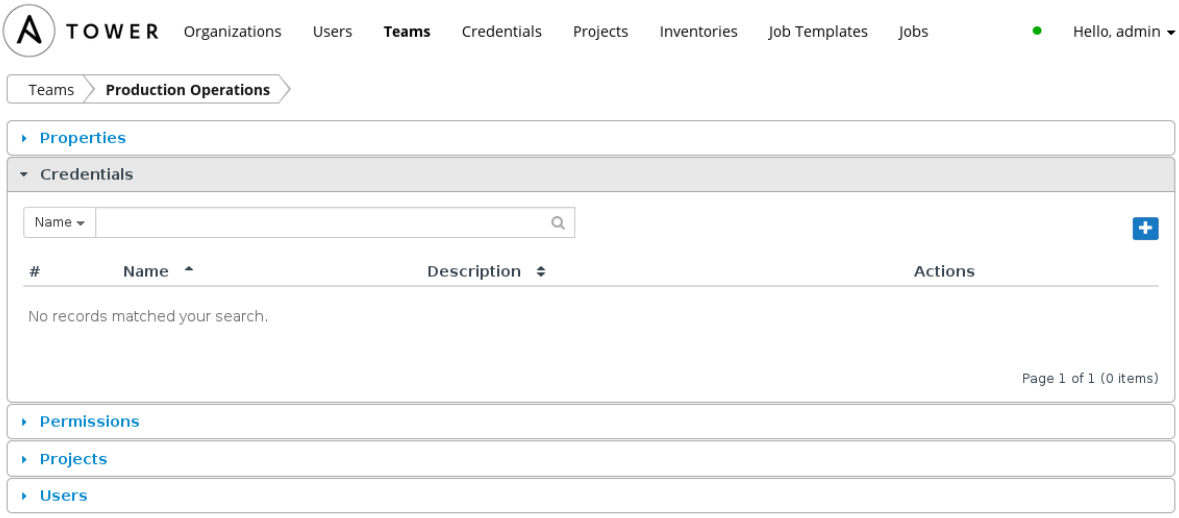
All fields are required. Select **Save**.

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.

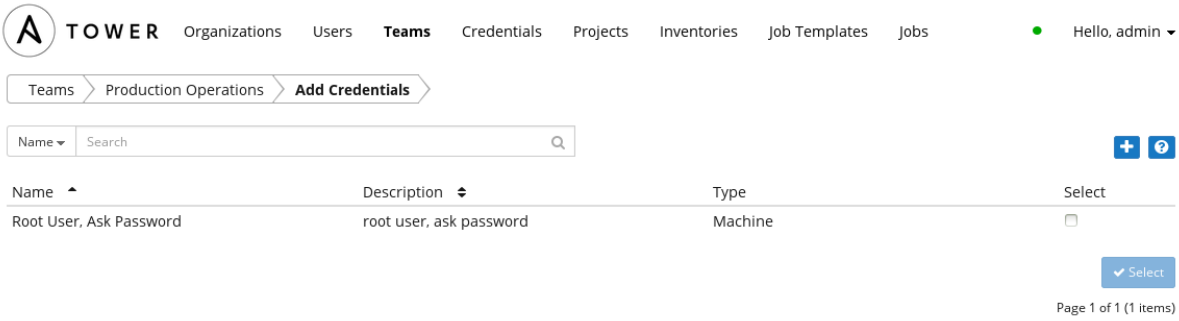


Teams - Credentials

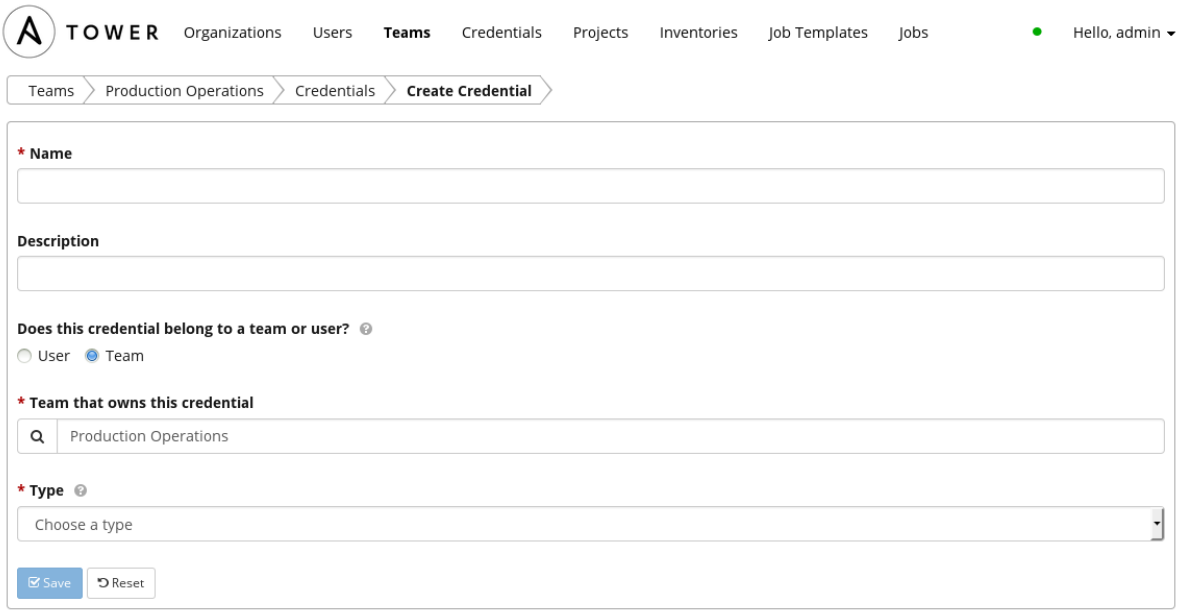
Credentials are utilized by Tower for authenticating when launching jobs against machines, to synchronize with inventory sources, and to import project content from a version control system. For details about how to use credentials, please see the section Credentials.



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.



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



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

Teams - Permissions

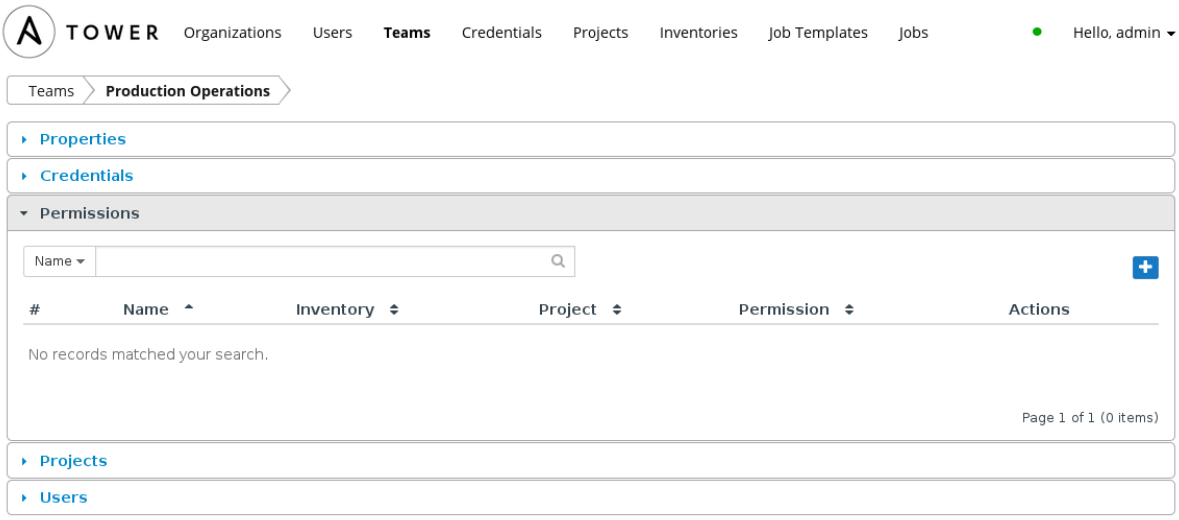
Permissions are the set of privileges assigned to users and teams that provide the ability to read, modify, and administer projects, inventories, and other Tower elements.

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

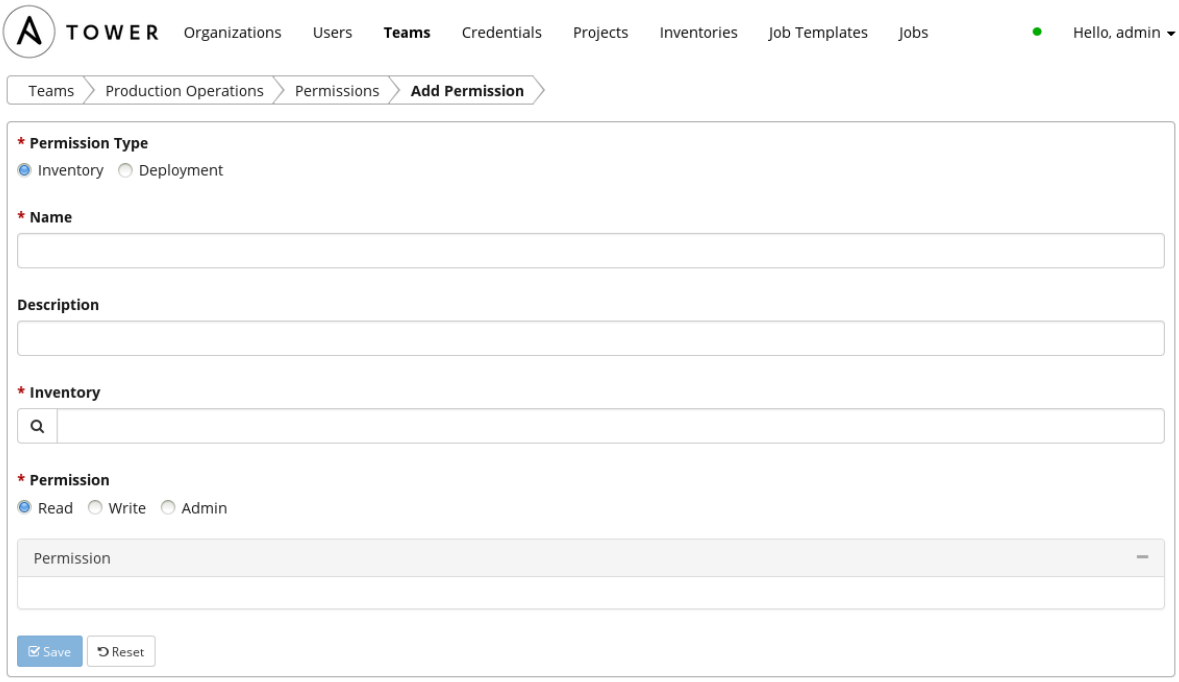
- Inventory: grants permission to act on inventories, groups, and hosts
  - Admin: modify the settings for the specified inventory. This permission also grants Read and Write permissions.
  - Read: view groups and hosts within a specified inventory
  - Write: create, modify, and remove groups, and hosts 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.

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.



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



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

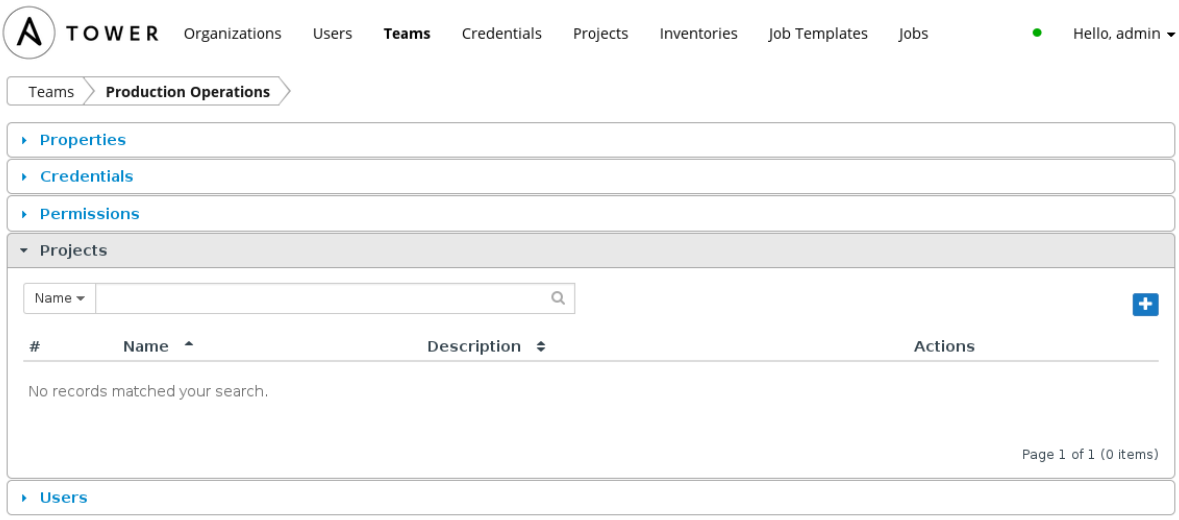
For a permission of type **Deployment**, enter the following details:

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

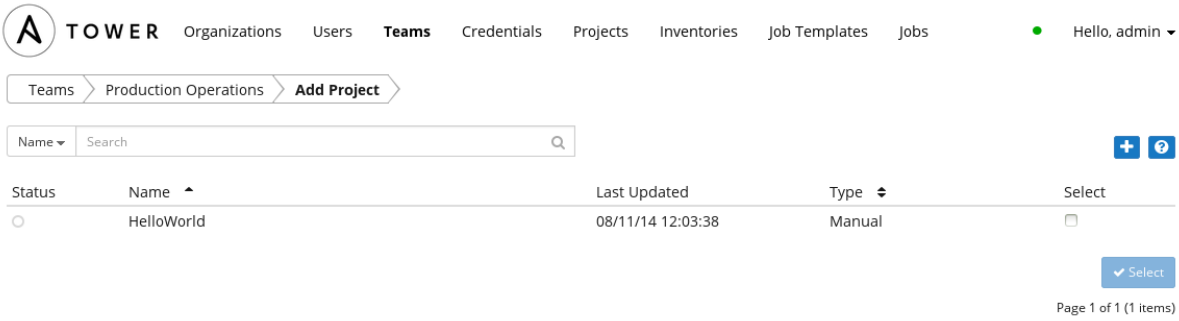
Select **Save**.

Teams - Projects

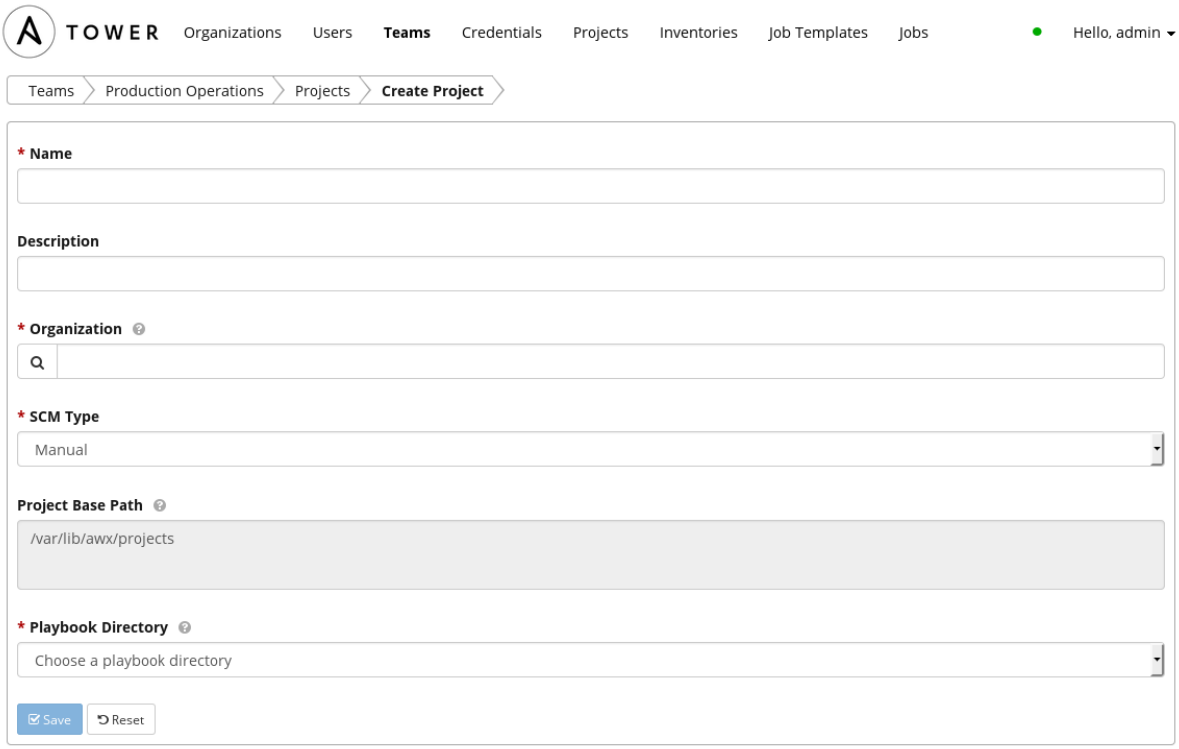
This displays the list of projects that this team has access to. This list may be searched by **Project Name** or **Description**. For more information about projects, please see the section Projects.



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



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.



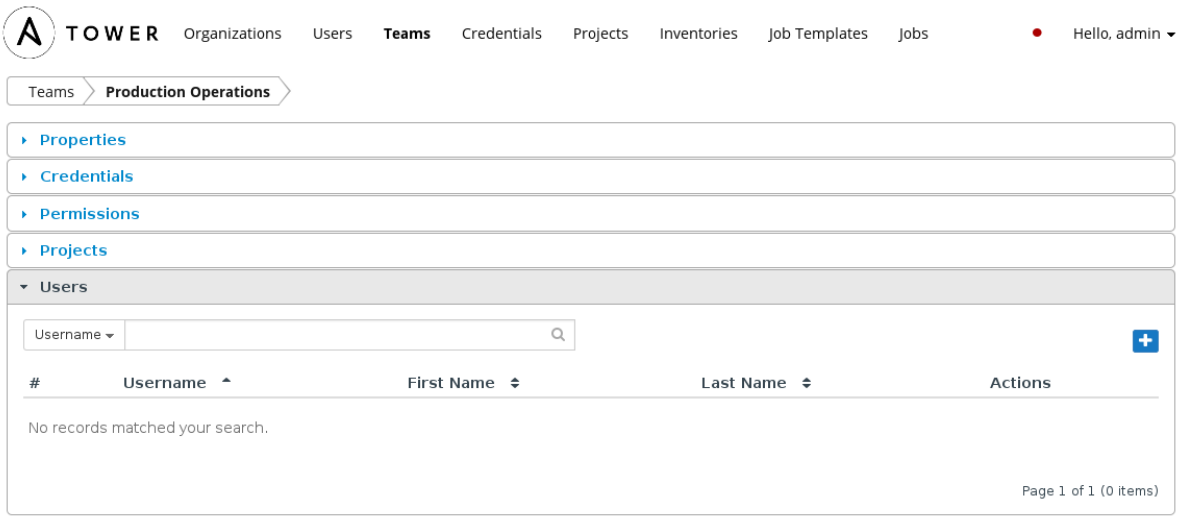
Enter the appropriate details into the following fields:

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

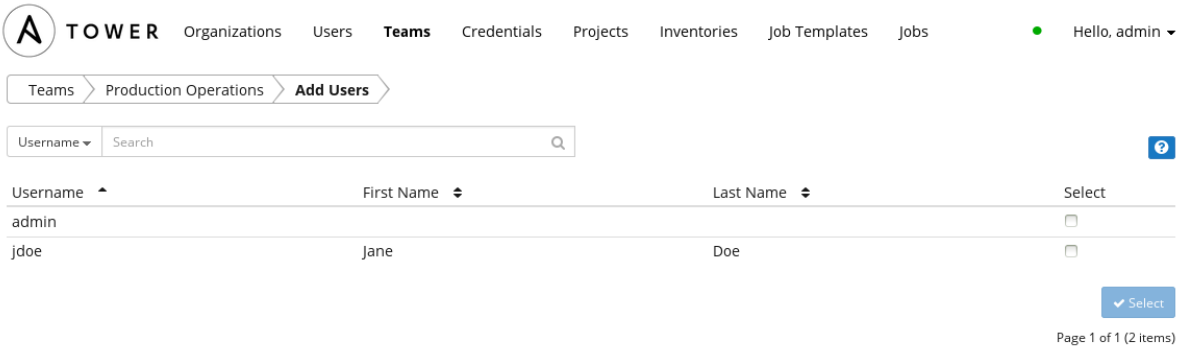
All fields are required. Select **Save**.

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.



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.

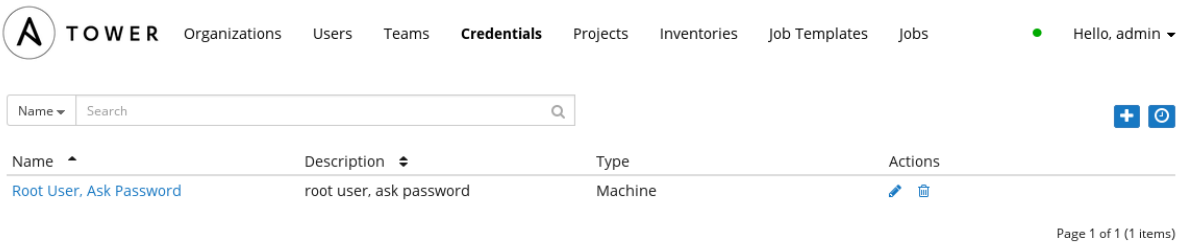


Credentials

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**.



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.

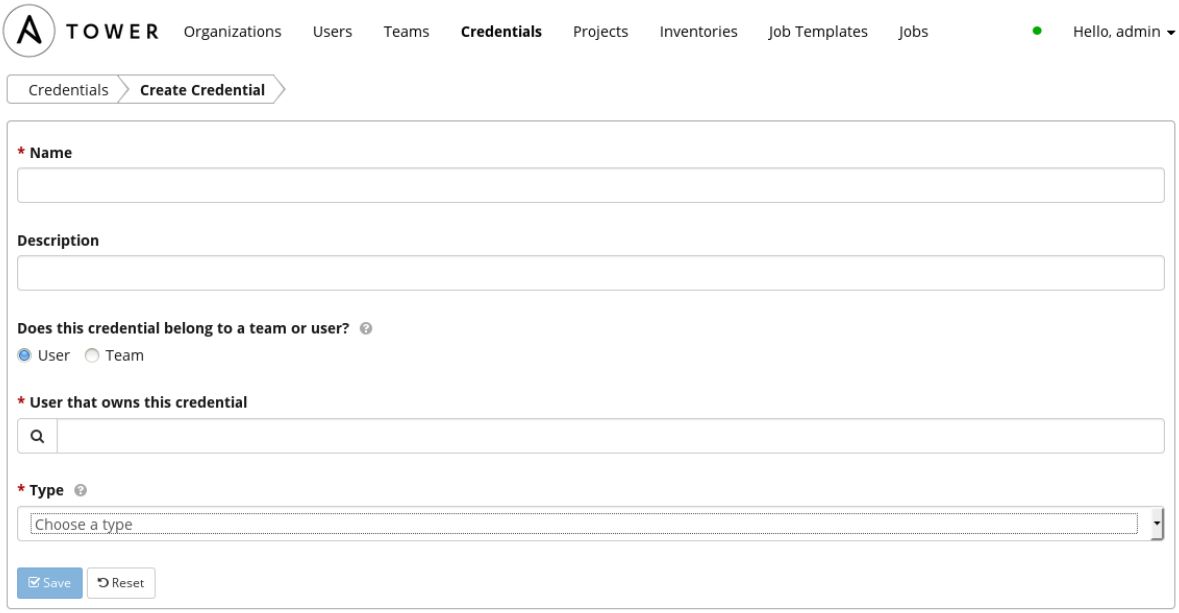
Credentials added to a **Team** will be available to all members of the team, whereas credentials added to a user are only available to that user by default.

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.



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

There are many types of Credentials:

Machine

Machine credentials define SSH and Sudo access for playbooks. They are used when submitting jobs to run playbooks on a remote host.

Machine credentials have several attributes that may be configured:

- **SSH Username**

The username to be used to authenticate the user via SSH.

- **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 necessary 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.

- **Key Password**

If the SSH Private Key used is protected by a password, you may configure a Key Password for 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 necessary 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 Username**

The username to sudo to on the remote system.

- **Sudo Password**

The actual password to be used to authenticate the user via sudo on the remote system. This password may be stored encrypted in the Tower database, if entered. Alternatively, you may configure Tower to ask the user for the password when necessary by selecting **Ask at runtime?**. In 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 SSH passwords or SSH Private Keys, since Tower must first establish an authenticated SSH connection with the host prior to invoking sudo to change to the sudo user.

- **Vault Password**

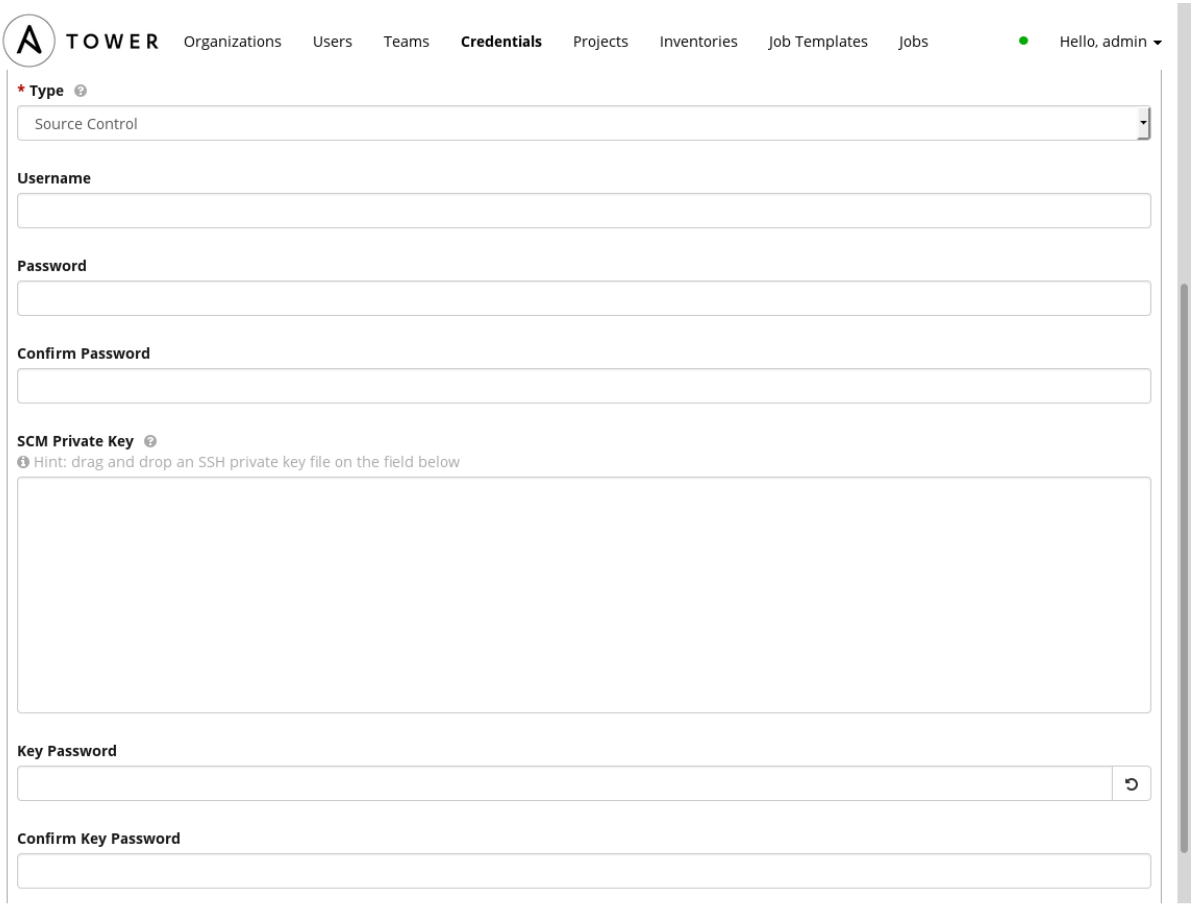
If your playbook uses Ansible Vault, add the Vault password to your credential here. Alternatively, you may configure Tower to ask the user for the vault password when necessary by selecting "**Ask at runtime?**." In that case, a dialog will open when the job is launched into which the user may enter the password and password confirmation.

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

NOTE: *Any credentials that will be used in Scheduled jobs must not be configured as* ***Ask at runtime?****.*

Source Control

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



Source Control credentials have several attributes that may be configured:

- **Username**

The username to use in conjunction with the source control system.

- **Password**

The password to use in conjunction with the source control system.

- **SCM Private Key**

The actual SSH Private Key to be used to authenticate the user to the source control system via SSH.

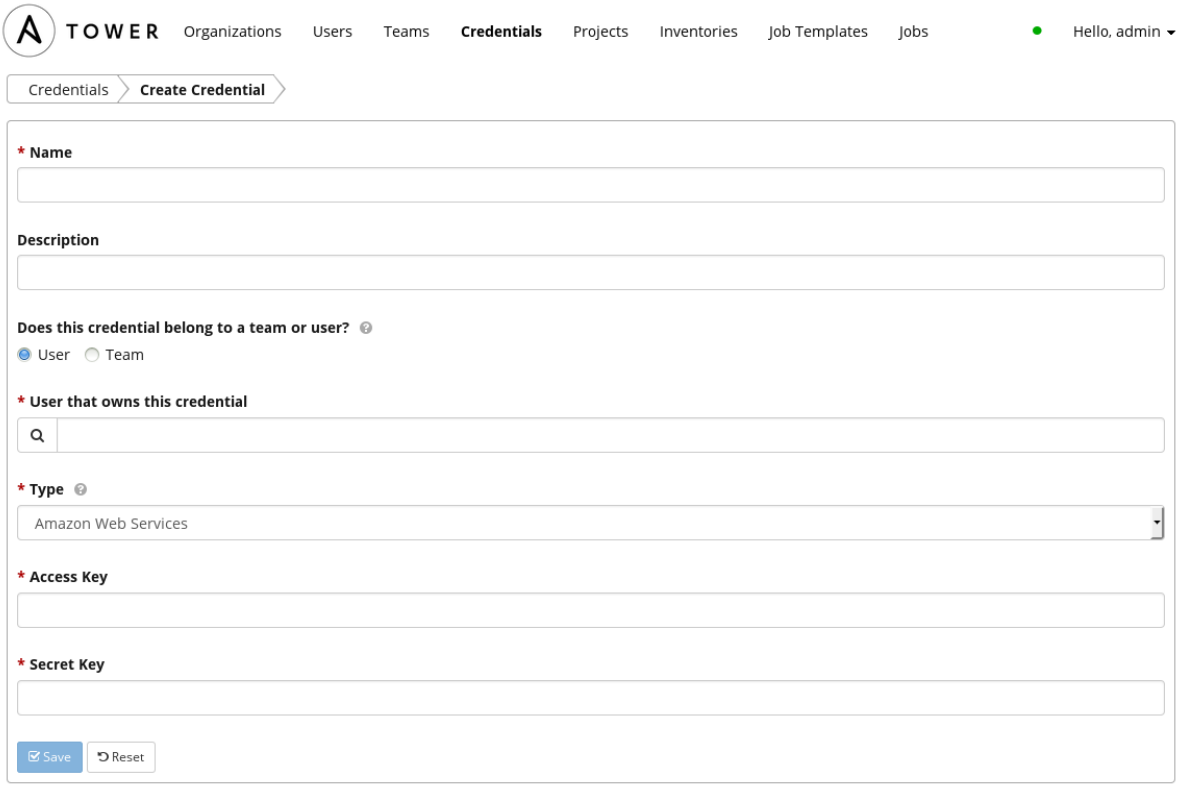
- **Key Password***

If the SSH Private Key used is protected by a password, you may configure a Key Password for the private key.

NOTE: *Source Control credentials cannot be configured as* ***Ask at runtime?****.*

Amazon Web Services

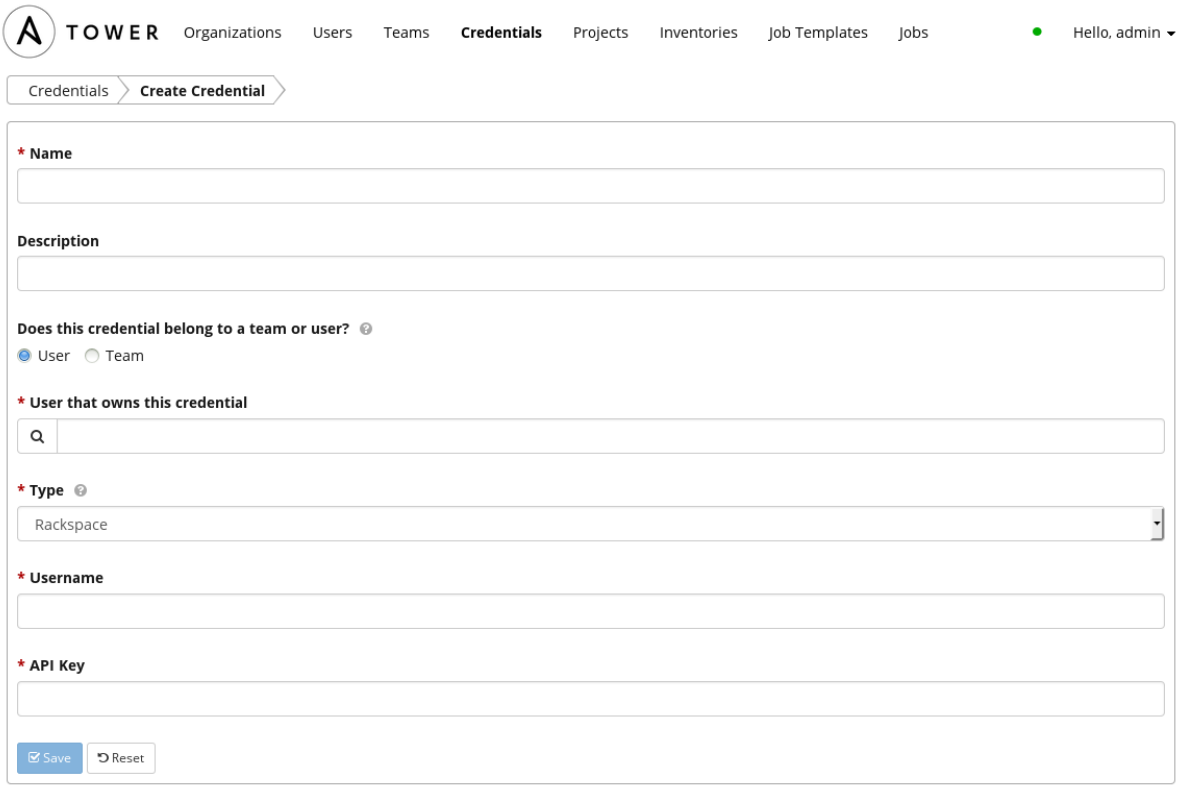
Enables synchronization of cloud inventory with Amazon Web Services.



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

Rackspace

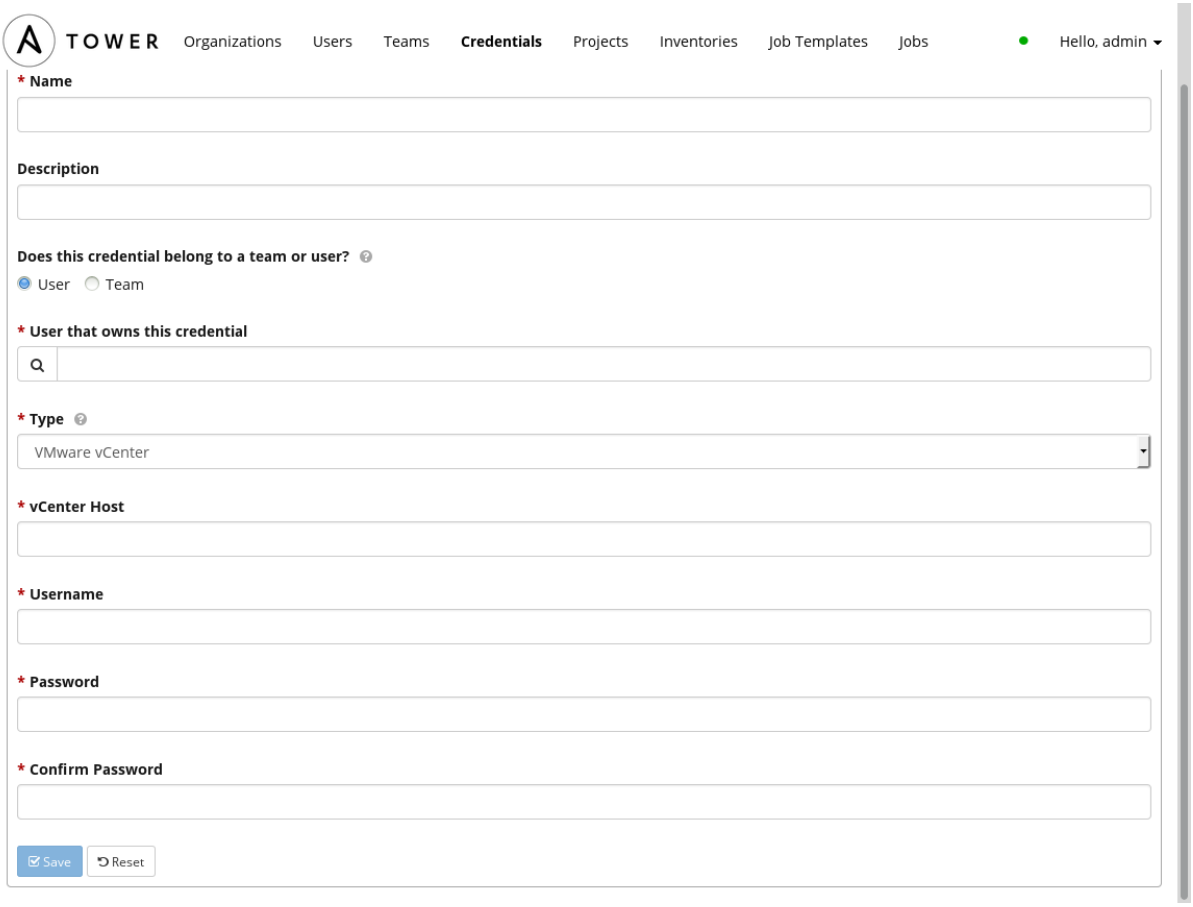
Enables synchronization of cloud inventory with Rackspace.



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

VMware

Enables synchronization of inventory with VMware vCenter.



VMware credentials have several attributes that may be configured:

- **vCenter Host**

The vCenter hostname or IP address to connect to.

- **Username**

The username to use to connect to vCenter.

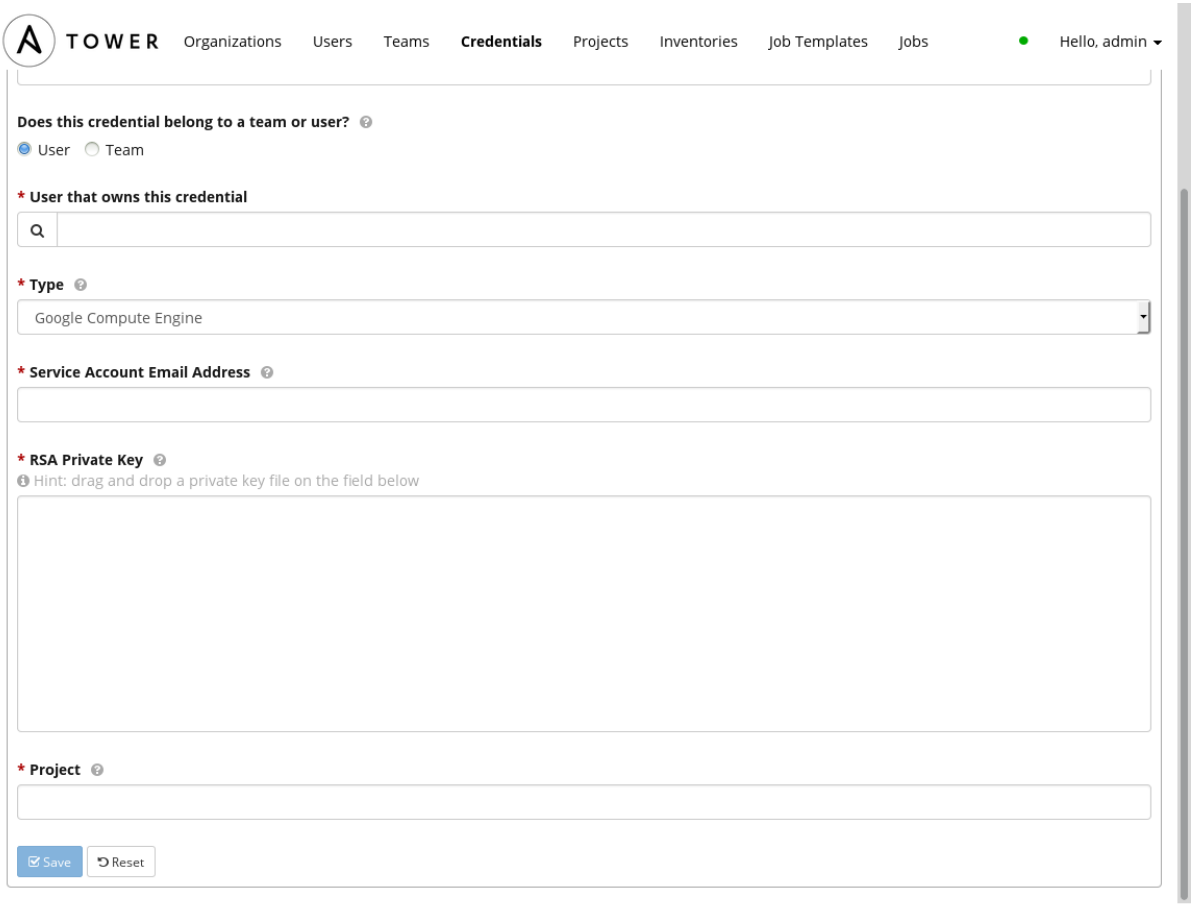
- **Password**

The password to use to connect to vCenter.

> ***NOTE:*** *If the VMware guest tools are not running on the instance, VMware inventory sync may not return an IP address for that instance.*

Google Compute Engine

Enables synchronization of cloud inventory with Google Compute Engine.



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

- **Service Account Email Address**

The email address assigned to the Google Compute Engine **service account**.

- **RSA Private Key**

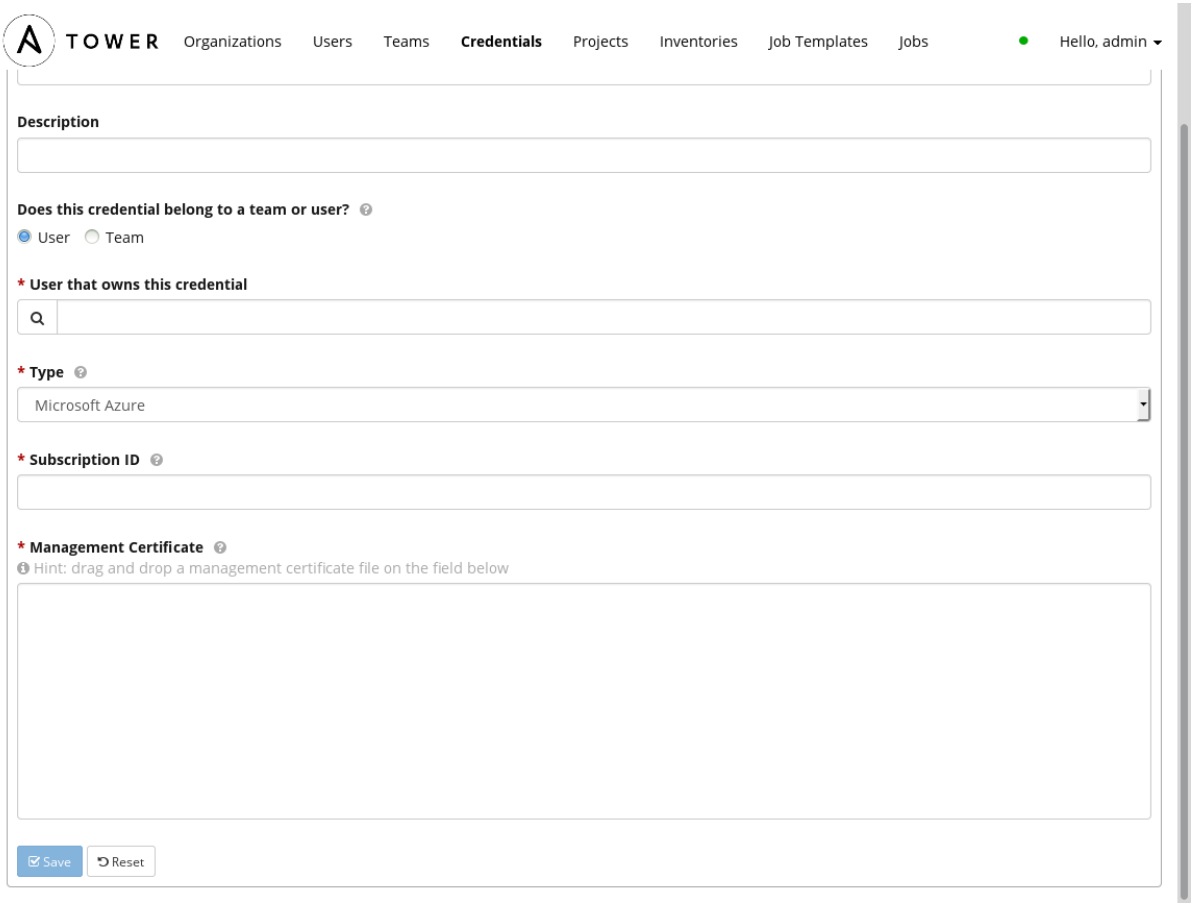
The PEM file associated with the service account email.

- **Project**

The GCE assigned identification. It is constructed as two words followed by a three digit number, such as: squeamish-ossifrage-123.

Microsoft Azure

Enables synchronization of cloud inventory with Windows Azure.



Microsoft Azure credentials have several attributes that may be configured:

- **Subscription ID**

The Subscription UUID for the Microsoft Azure account.

- **Management Certificate**

The PEM file that corresponds to the certificate you uploaded in the Microsoft Azure console.

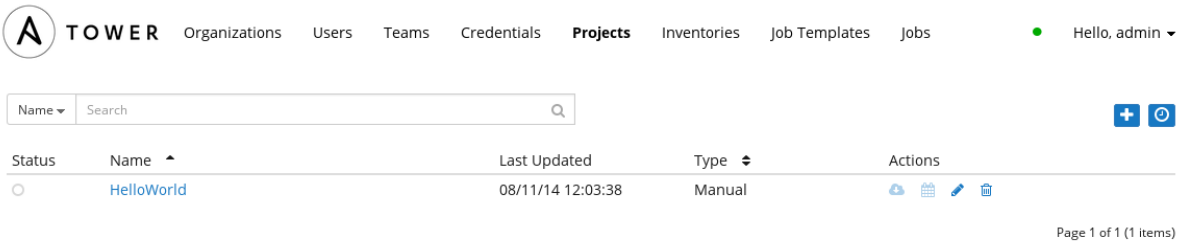
Projects

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

You can manage playbooks and playbook directories by either placing manually them under the Project Base Path on your Tower server, or by placing your playbooks into a source code management (SCM) system supported by Tower, including Git, Subversion, and Mercurial.

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

This menu displays a list of the projects that are currently available. The list of projects may be sorted and searched by **Name**, **Type**, or by **Status**. For each project listed, you can edit project properties and delete the project, using the edit and delete icons.



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

- Create a new project
- View Activity Stream

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

- Running - Source control update is currently in progress
- Never updated - Project is configured for source control, but has never been updated
- Failed - The last source control update for this project failed
- Successful - The last source control update for this project succeeded succeeded
- Missing - The project directory is missing (valid for both manual or source control managed projects) project has a last update, but the project directory is missing, or project doesn't use SCM and the directory is missing
- OK - The project is not configured for source control, and is correctly in place.

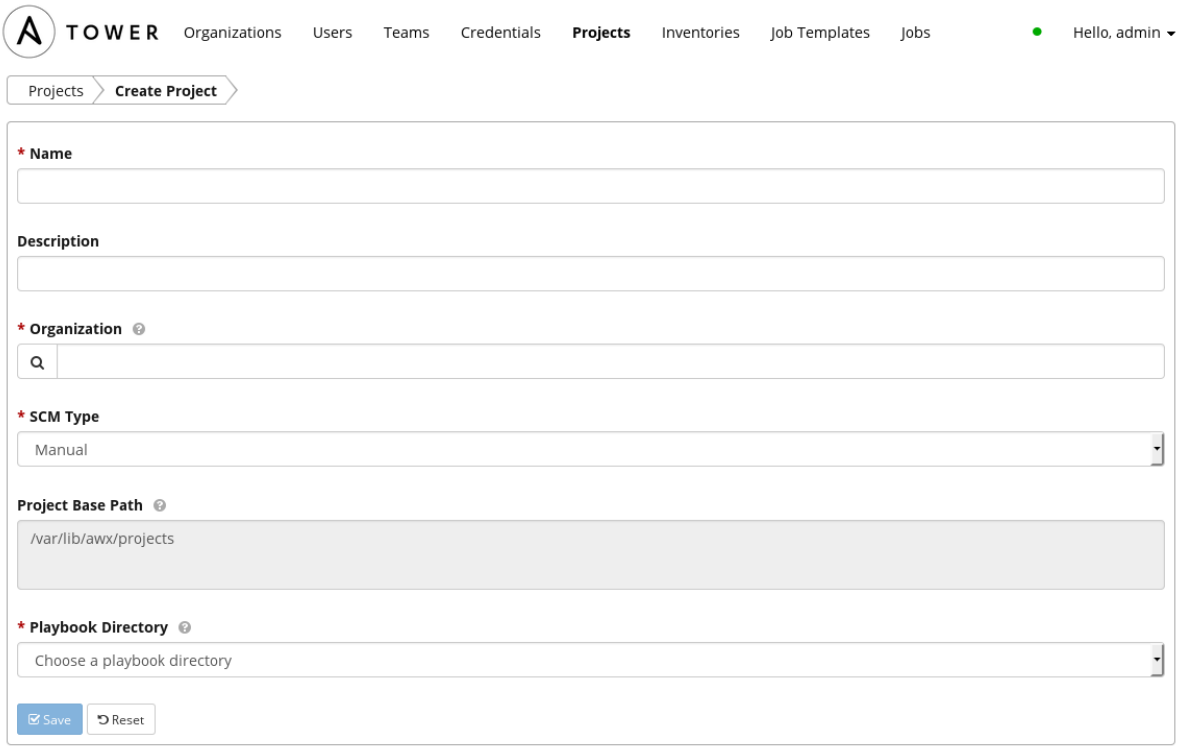
Under **Actions**, the following actions are available:

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

- Delete the project
- Cancel a running or scheduled update from source control, if configured for this project

Add a new project

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



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, Subversion, 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.

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."

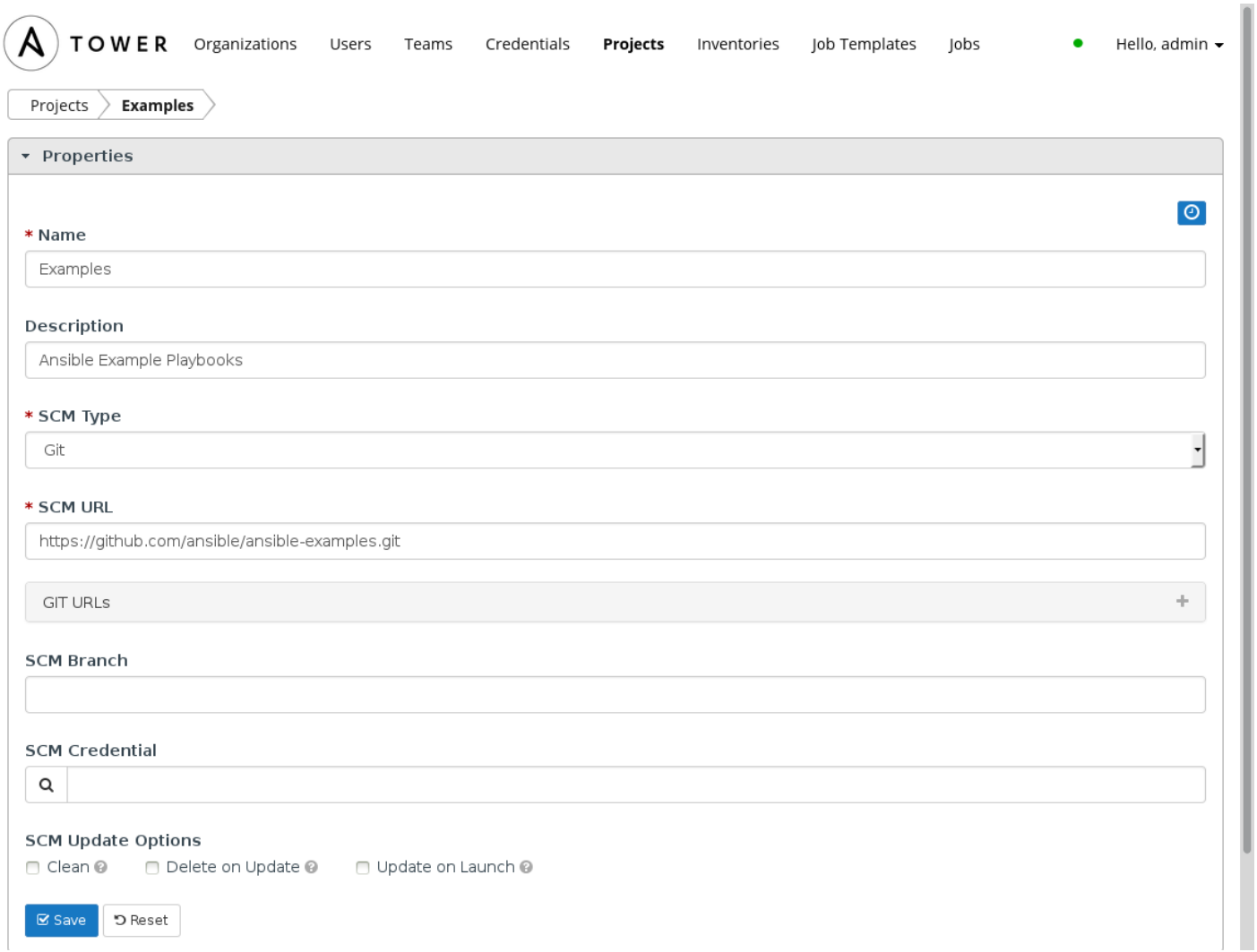
To manage playbooks manually

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

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

To manage playbooks using Source Control

- Select the appropriate **SCM Type**.



- Enter the appropriate details into the following fields:
- SCM URL
- SCM Branch

  Optionally enter the SCM branch for Git or Mercurial.
- Revision # (Subversion only)

  Optionally enter the Revision # for Subversion.
- If authentication is required, select the appropriate SCM credential.
- Clean

  Remove any local modifications prior to performing an update.

- Delete on Update

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

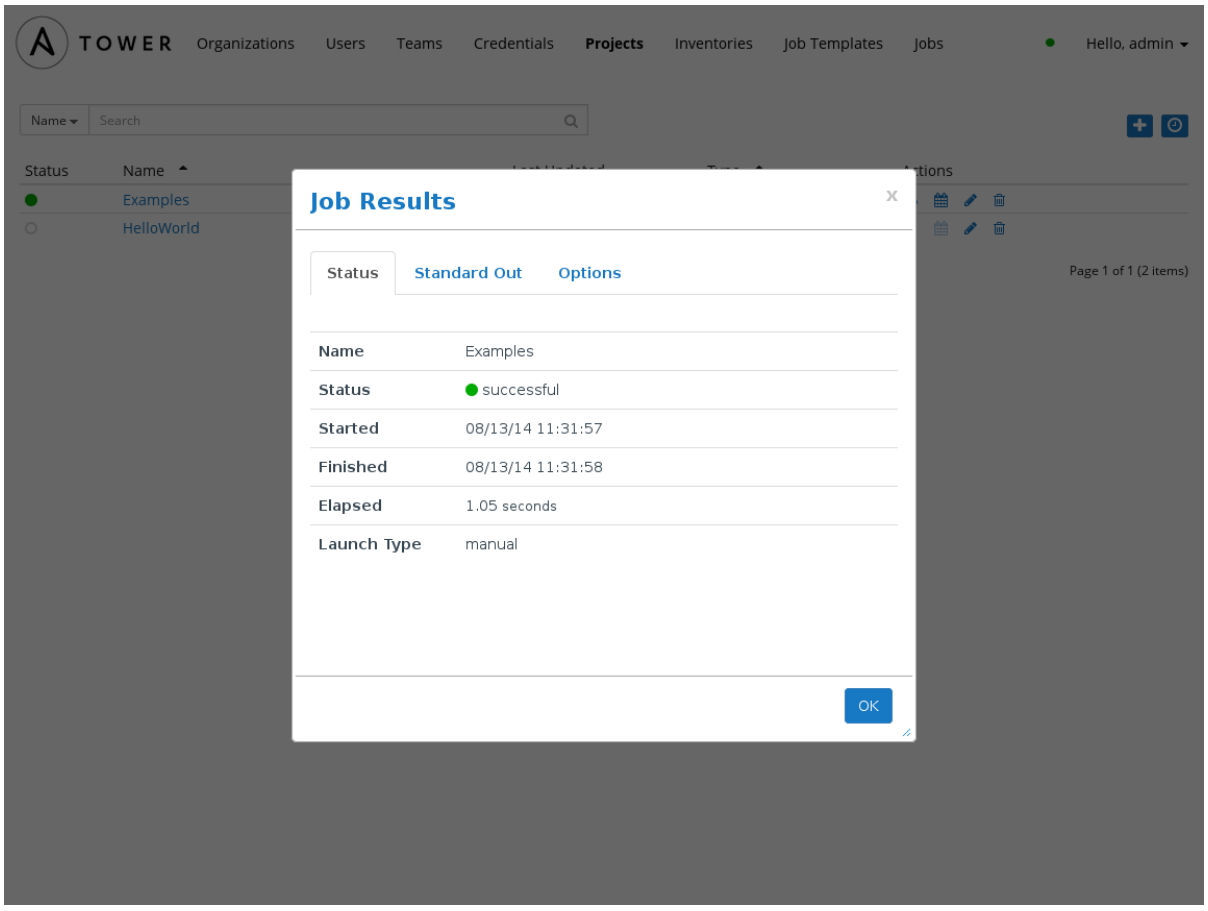
  Each time a job runs using this project, perform an update to the local repository prior to starting the job. To avoid job overflows if jobs are spawned faster than the project can sync, selecting this allows you to configure a Cache Timeout to cache prior project syncs for a certain number of seconds.

Click **Save** to save your project.

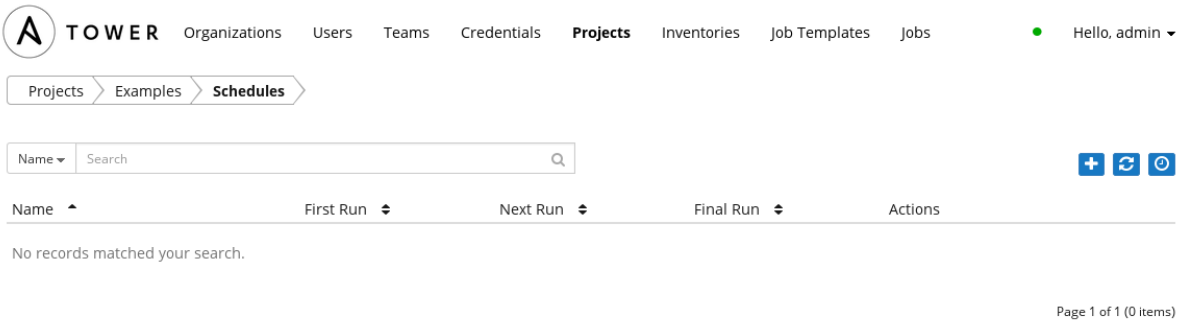
Updating projects from source control

Update an existing SCM-based project by clicking the button. This starts an update task.

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



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



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

The list of of schedules includes:

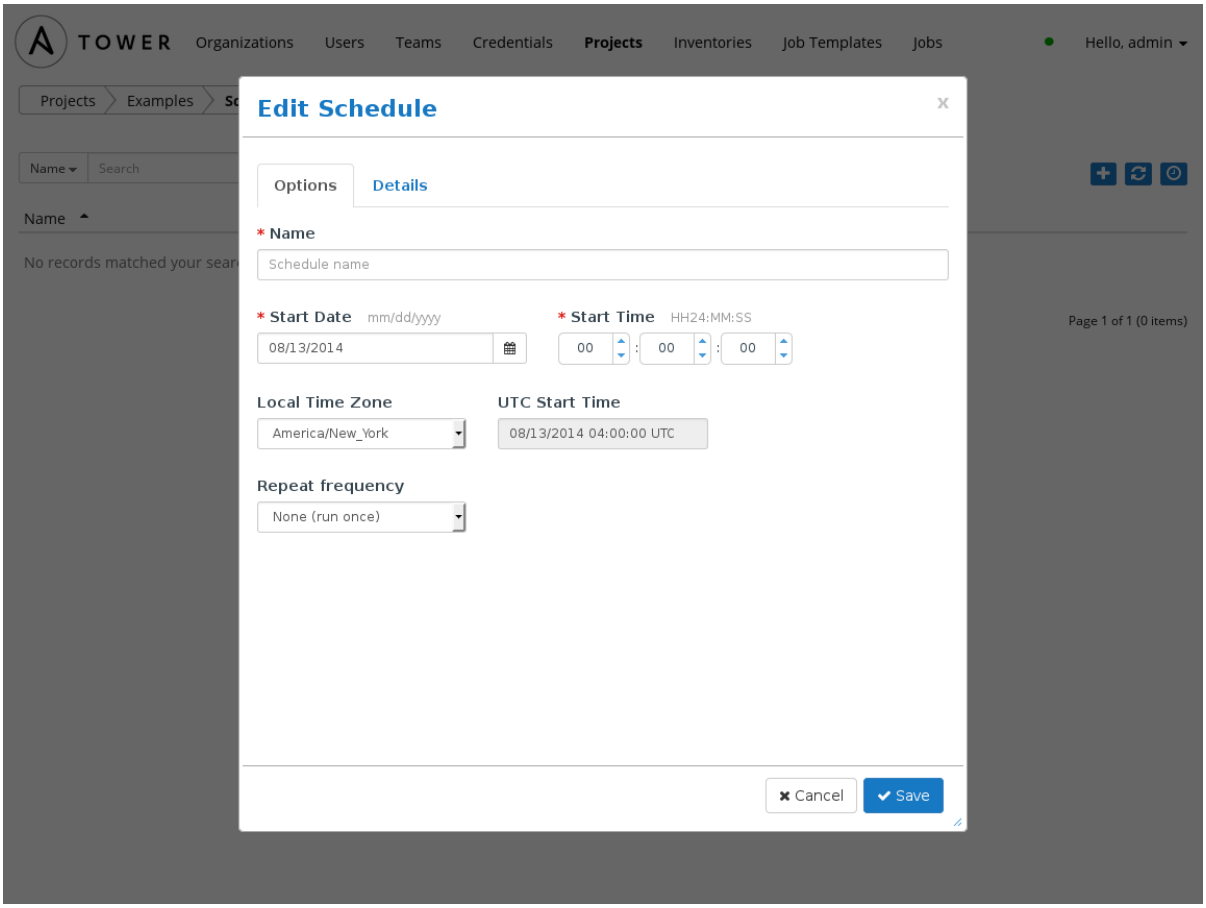
- Name - Clicking the schedule name will open the **Edit Schedule** dialog
- First Run - the first scheduled run of this task
- Next Run - the next scheduled run of this task
- Final Run - If the task has an end date, this is the last run of the task

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

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

Add a new schedule

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



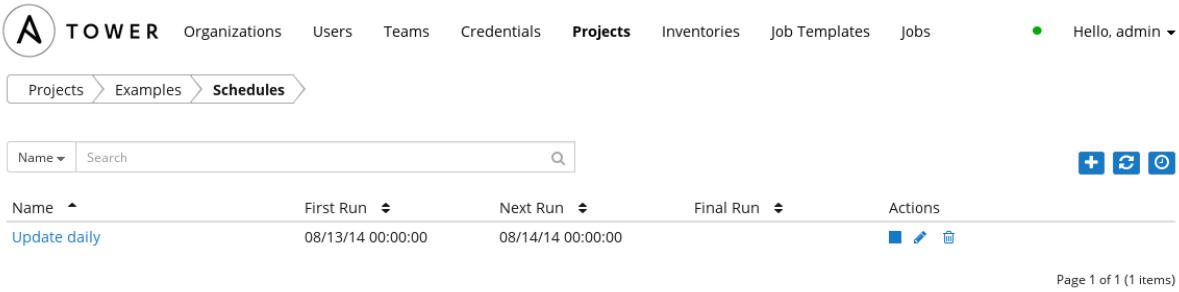
Enter the appropriate details into the following fields and select Save:

- Name (required)
- Start Date (required)
- Start Time (required)
- Local Time Zone (the entered Start Time should be in this timezone)
- UTC Start Time (calculated from Start Time + Local Time Zone)
- Repeat Frequency - 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 the selected Local Time Zone.

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

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



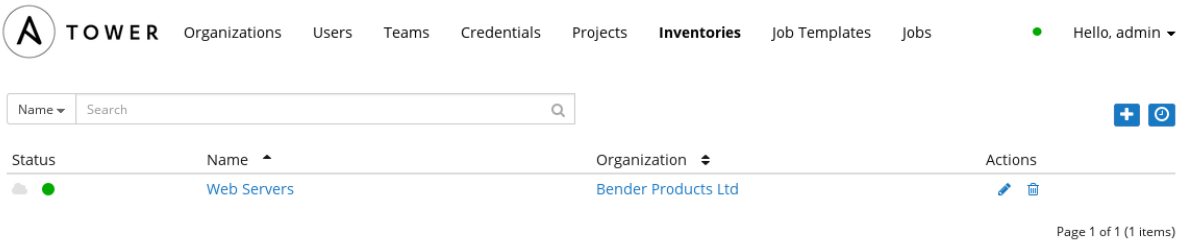
- Stop an active schedule or activate a stopped schedule
- Edit Schedule
- Delete schedule

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 one of Ansible Tower's supported cloud providers.

NOTE: *If you have a custom dynamic inventory script, or a cloud provider that is not yet supported natively in Tower, you can also import that into Tower. Please see the section on [Administration of Tower](#administration-of-tower)*

This tab displays a list of the inventories that are currently available. The inventory list may be sorted and searched by **Name** or **Organization** and filtered by inventories with external sources, inventories with external sources that have failed to update, and inventories whose hosts have failed jobs.



The list of of inventories includes:

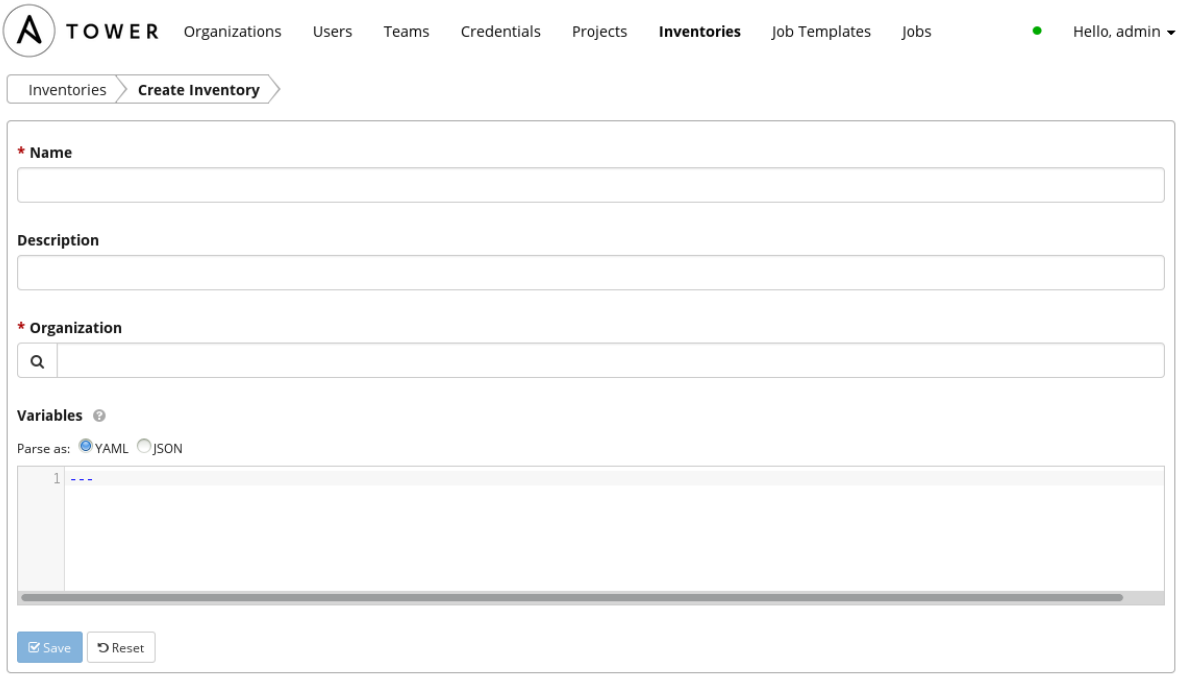
- *Status* - this includes the status of inventory synchronization for inventories configured with cloud sources, and the status of recent jobs for this inventory
- *Name* - the inventory name. Clicking the Inventory name will navigate to the properties screen for the selected inventory, which shows the inventory's groups and hosts. (This view is also accessible from the **Action** menu.)
- *Organization* - the organization that the inventory belongs to
- *Actions* - the following actions are available for the selected inventory:
    - Edit - Edit the properties for the selected inventory
    - Delete - Delete the selected inventory. This operation cannot be reversed!

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

- Create a new inventory
- View Activity Stream

Add a new inventory

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



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

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

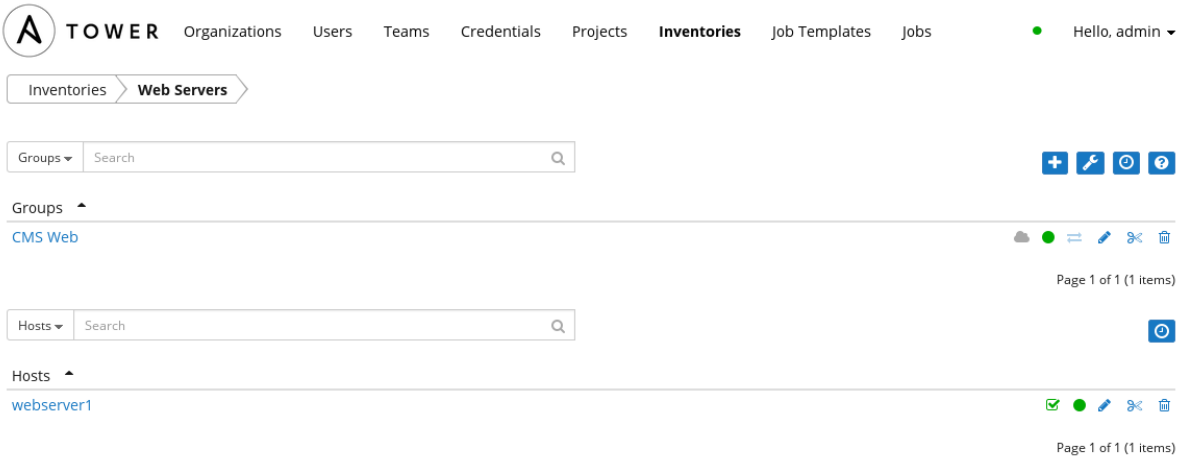
  Variable definitions and values to be applied to all hosts in this inventory. Enter variables using either JSON or YAML syntax. Use the radio button to toggle between the two.

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 inventories.

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

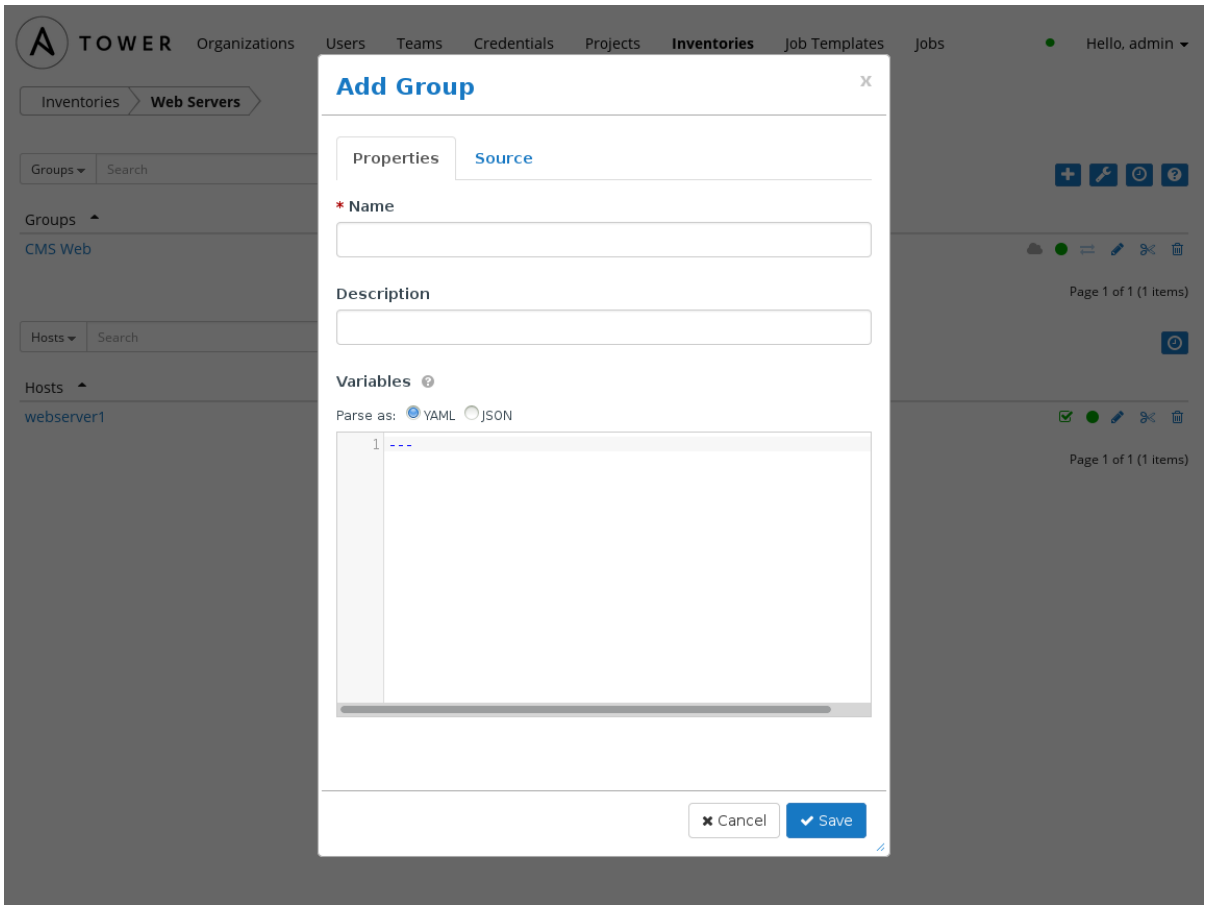
Under Groups, you can see the groups for this inventory. Groups can be filtered or searched by group name.

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

- Sync status - Show the status of inventory synchronization for groups configured with cloud sources. If synchronization is configured, clicking this button will show the synchronization log for the selected group.
- 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
- Copy Group - Groups can be nested. This allows you to copy or move the group to a different group.
- Delete - Delete the selected group. This operation cannot be reversed!

Add a new group

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



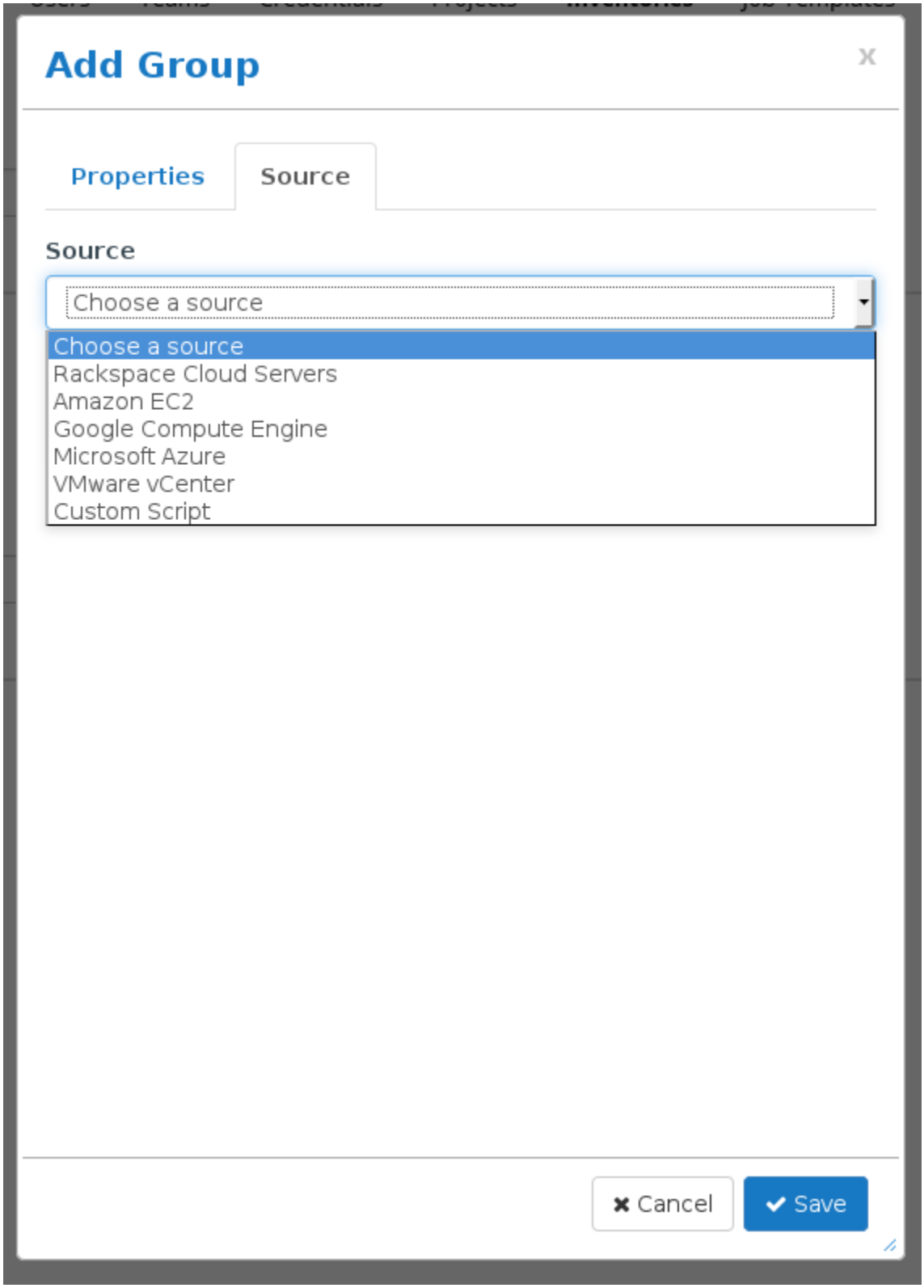
Enter the appropriate details into the following fields and click **Save**.

- Name (required)
- Description
- Variables

  Variable definitions and values to be applied to all hosts in this group. Enter variables using either JSON or YAML syntax. Use the radio button to toggle between the two.

By default, the group **Source** is manual, 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 2.1 supports Amazon Web Services EC2, Rackspace Cloud Servers, Google Compute Engine, VMware vCenter, Microsoft Azure, and custom scripts added by the administrator.

All cloud inventory sources have the following options:

- **Update Options**
  - Overwrite

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

    When not checked 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.

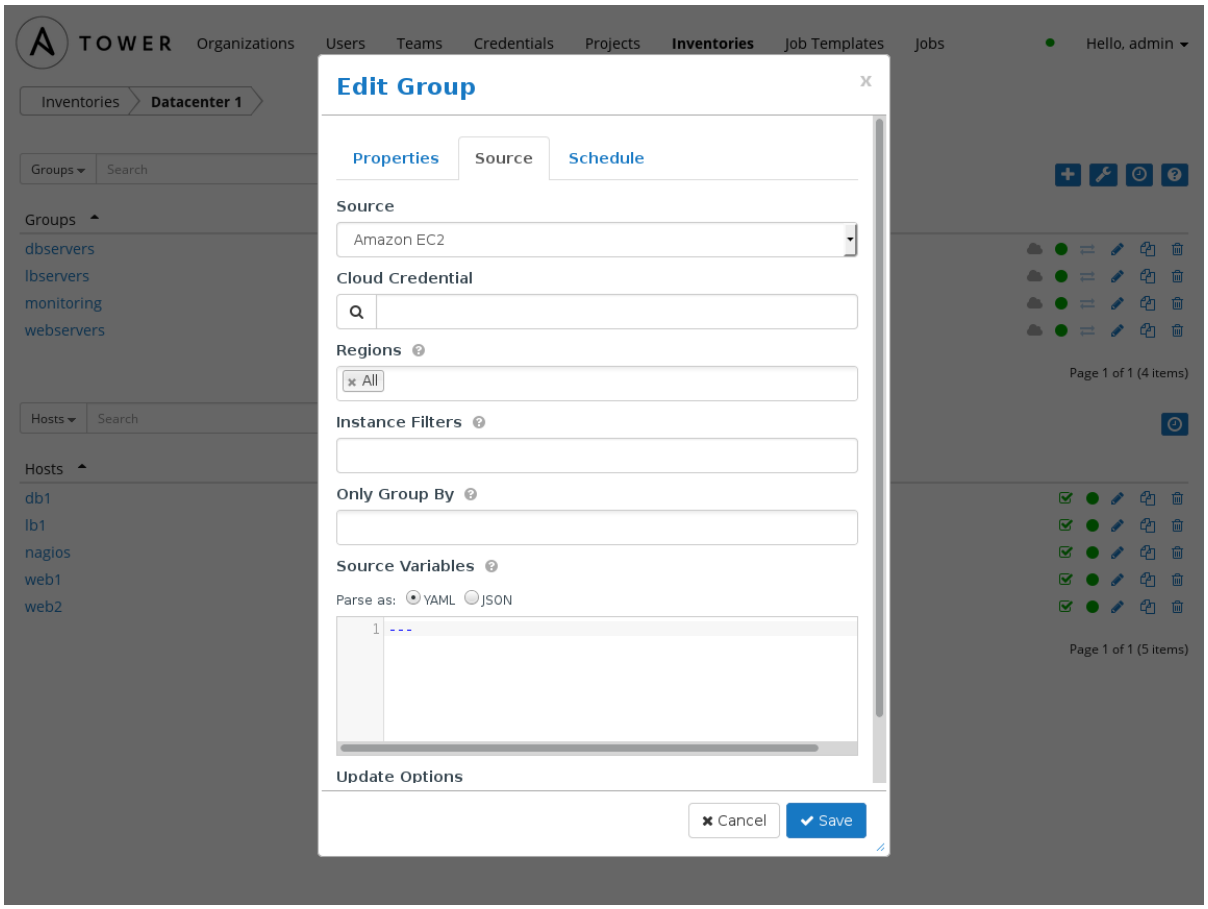
    When not checked a merge will be performed, combining local variables with those found on the external source.
  - Update on Launch

    Each time a job runs using this inventory, refresh the inventory from the selected source before executing job tasks. To avoid job overflows if jobs are spawned faster than the inventory can sync, selecting this allows you to configure a Cache Timeout to cache prior inventory syncs for a certain number of seconds.

NOTE: *If you intend to use Tower's provisioning callback feature with a dynamic inventory source,* ***Update on Launch*** *should be set for the inventory group.*

Amazon Web Services EC2

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



- **Cloud Credential**

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

If Tower is running on an EC2 instance with an assigned IAM Role, the credential may be omitted, and the security credentials from the instance metadata will be used instead. For more information on using IAM Roles, see the IAM Roles for Amazon EC2 (http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/iam-roles-for-amazon-ec2.html) documentation at Amazon.

- **Regions**

Click on the regions field to see a list of regions for your cloud provider. You can select multiple regions, or choose "All" to include all regions. Tower will only be updated with Hosts associated with the selected regions.

- **Instance Filters**

Rather than import your entire Amazon EC2 inventory, you can filter the instances returned by the inventory script based on a variety of metadata. Hosts will be imported if they match *any* of the filters entered here.

Examples:

- To limit to hosts having the tag `TowerManaged`:

  Enter `tag-key=TowerManaged`

- To limit to hosts using either the key-name `staging` or `production`:

  Enter `key-name=staging, key-name=production`

- To limit to hosts where the `Name` tag begins with `test`:

  Enter `tag:Name=test*`

For more information on the filters that can be used here, see the DescribeInstances (http://docs.aws.amazon.com/AWSEC2/latest/APIReference/API_DescribeInstances.html) documentation at Amazon.

- **Only Group By**

By default, Tower will create groups based on the following Amazon EC2 parameters:

- Availability Zones
- Image ID
- Instance Type
- Key Name
- Region
- Security Group
- Tags (by name)
- VPC ID

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

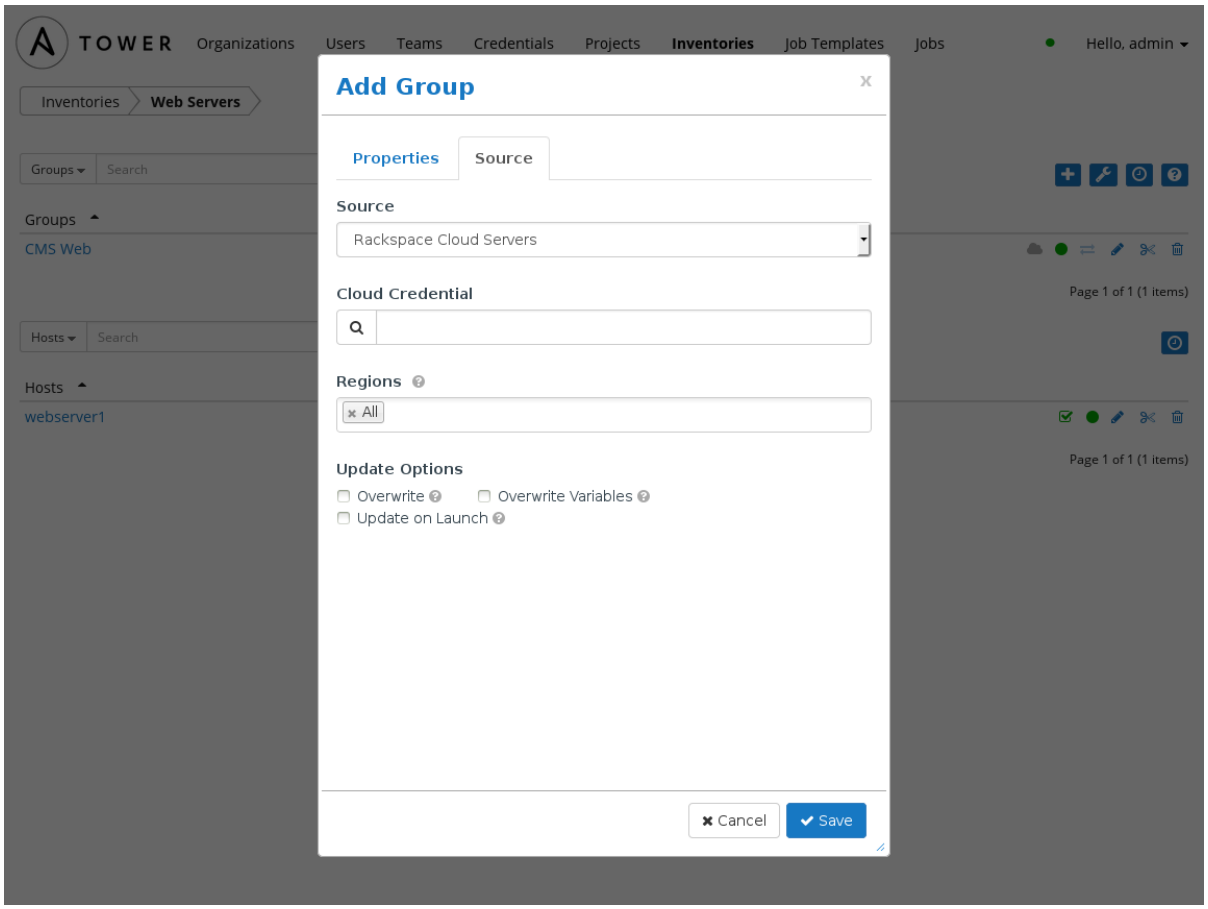
- **Source Variables**

Override variables found in ec2.ini and used by the inventory update script. For a detailed description of these variables view ec2.ini in the Ansible GitHub repo (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.

Rackspace Cloud Servers

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



- **Cloud Credential**

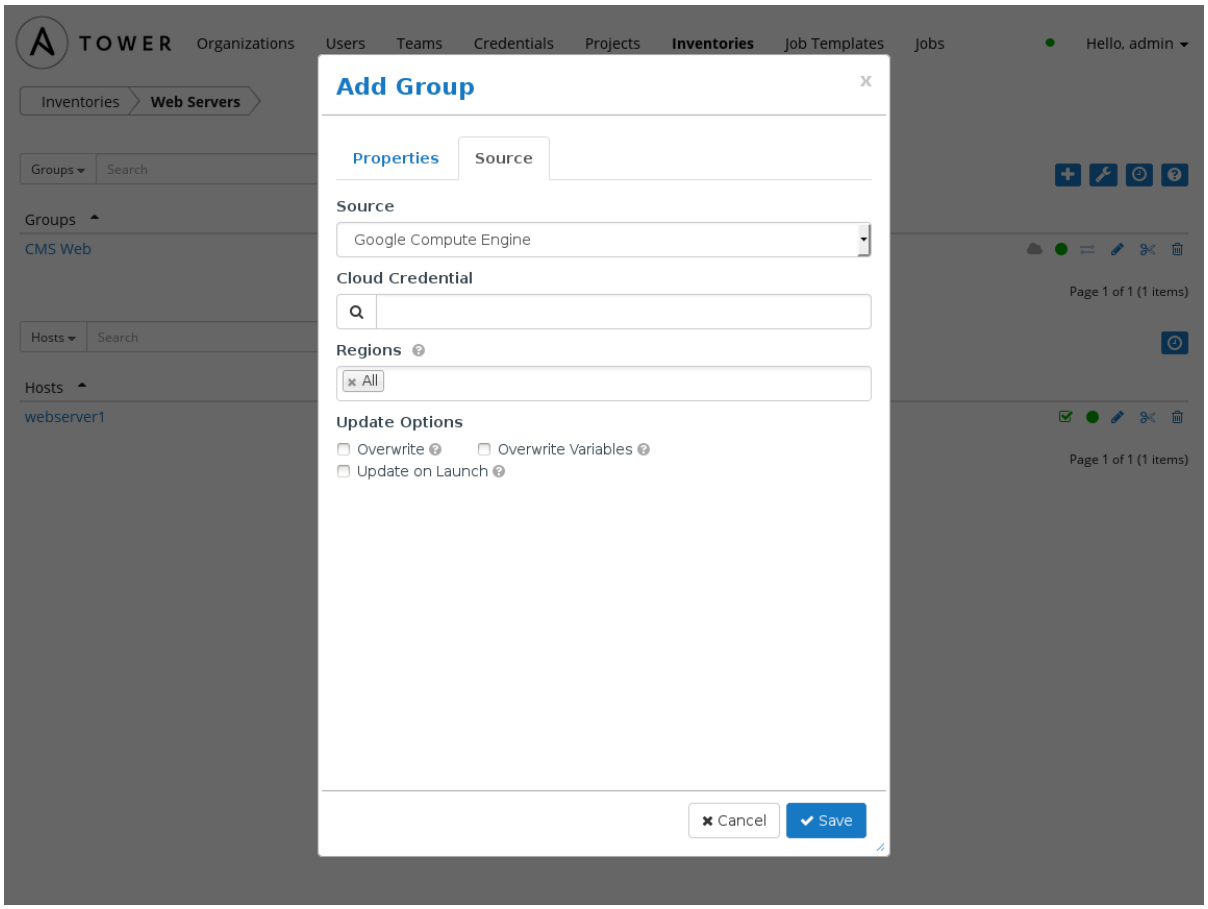
  Choose from an existing Credential. For more information, see the Credentials section.

- **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.

Google Compute Engine

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



- **Cloud Credential**

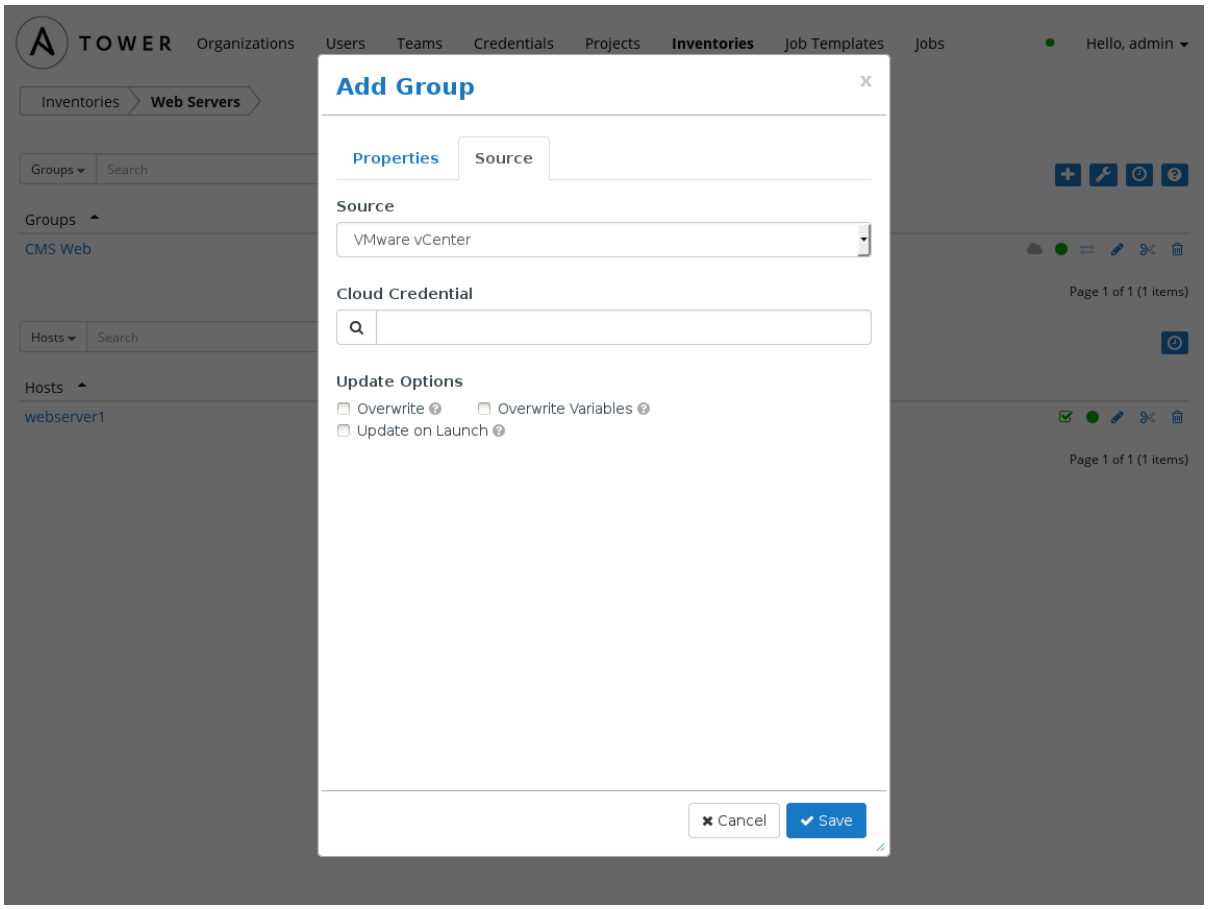
  Choose from an existing Credential. For more information, see the Credentials section.

- **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.

VMware vCenter

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



- **Cloud Credential**

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

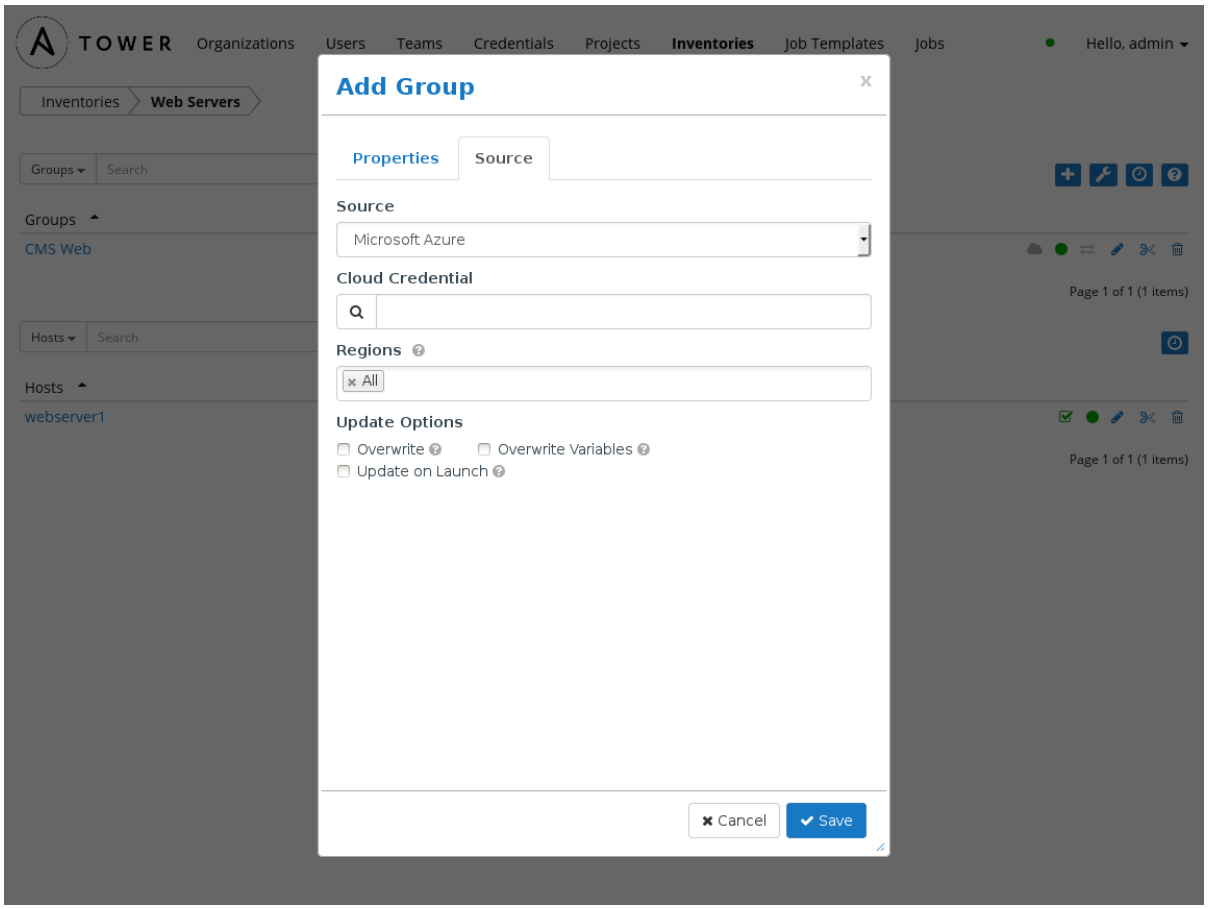
- **Source Variables**

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

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

Microsoft Azure

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



- **Cloud Credential**

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

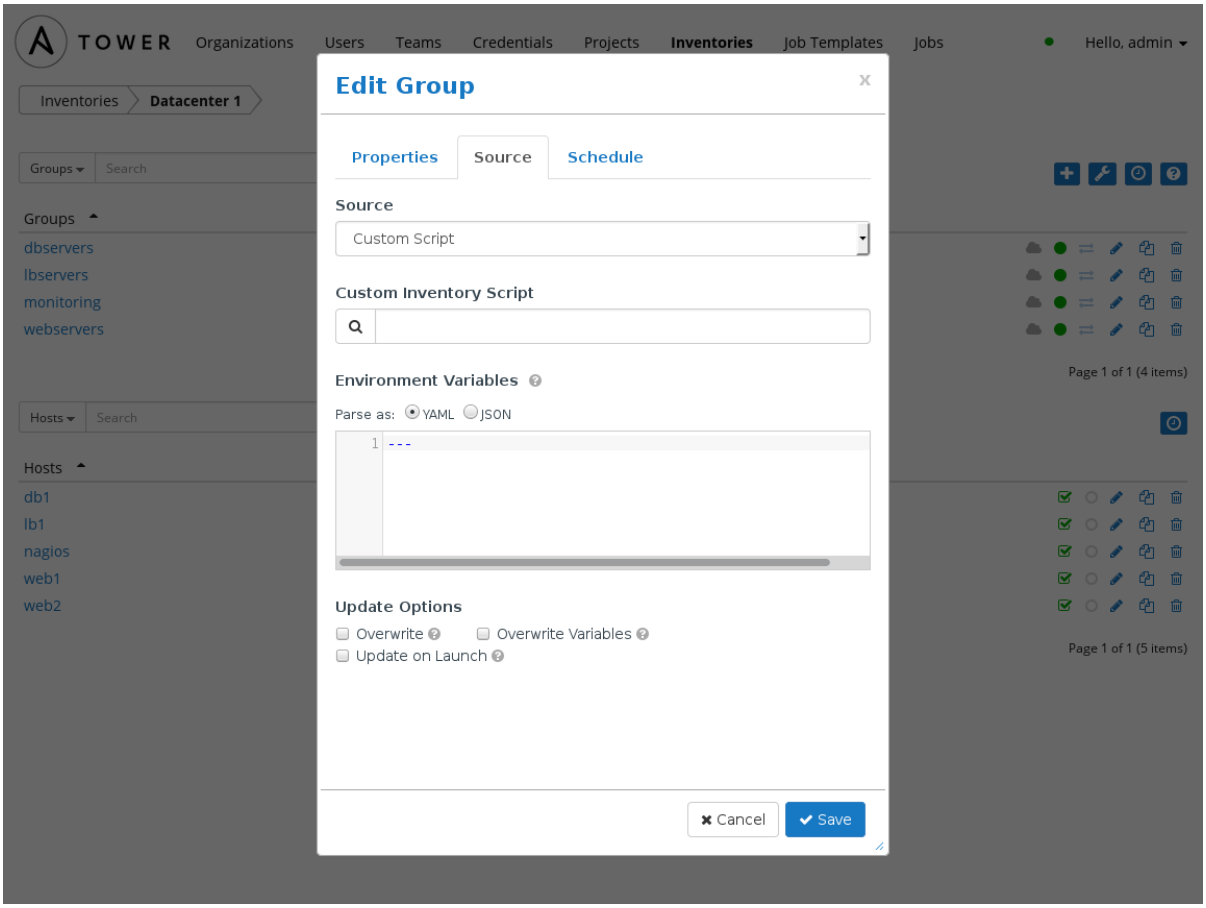
- **Regions**

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.

Custom Script

Tower allows you to use a custom dynamic inventory script, if your administrator has added one. For information on how to add custom inventory scripts to Tower, see Custom Inventory Scripts.

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



- **Custom Inventory Script**

  Choose from an existing Inventory Script. For more information, see the Custom Inventory Scripts section.

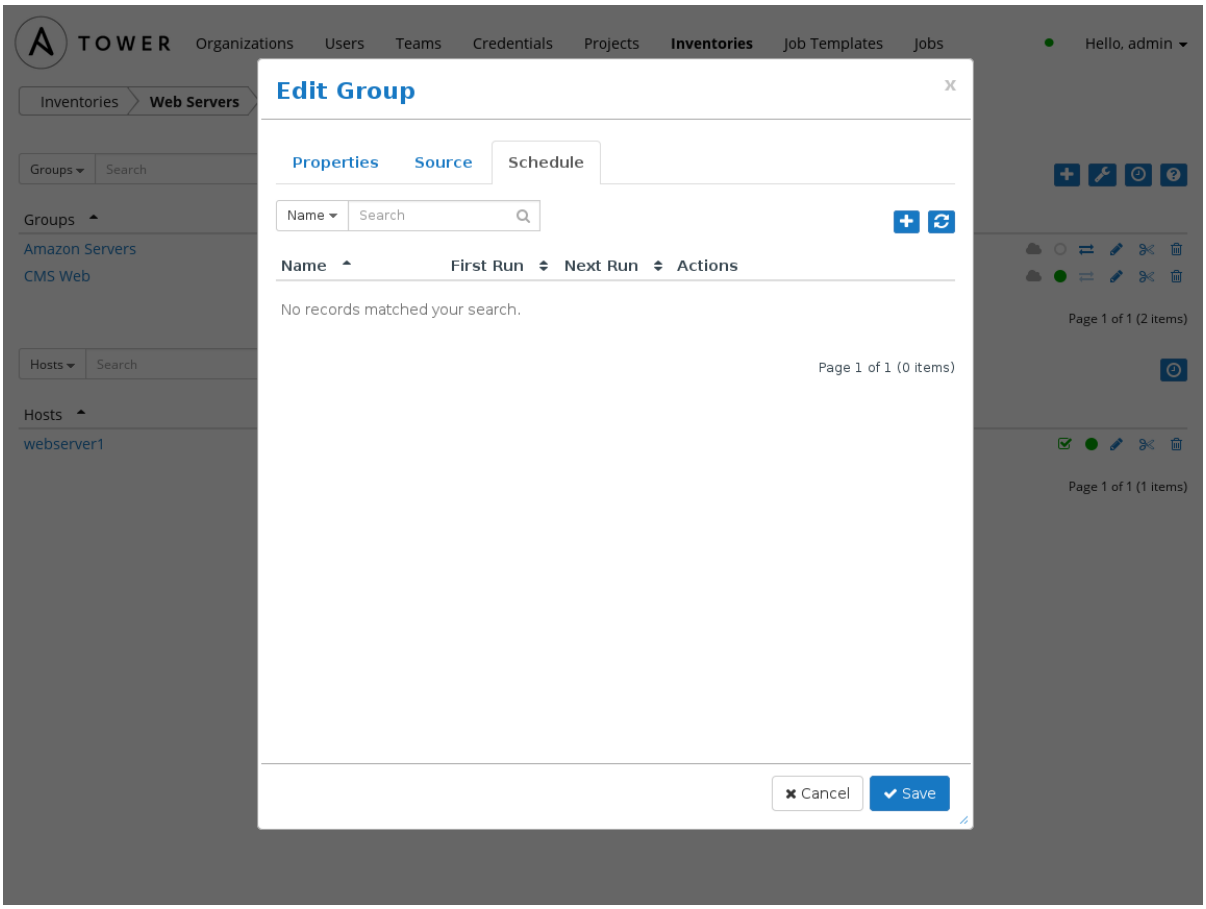
- **Environment Variables**

Set variables in the environment to be used by the inventory update script. The variables would be specific to the script that you have written.

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

Scheduling

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



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

The list of schedules includes:

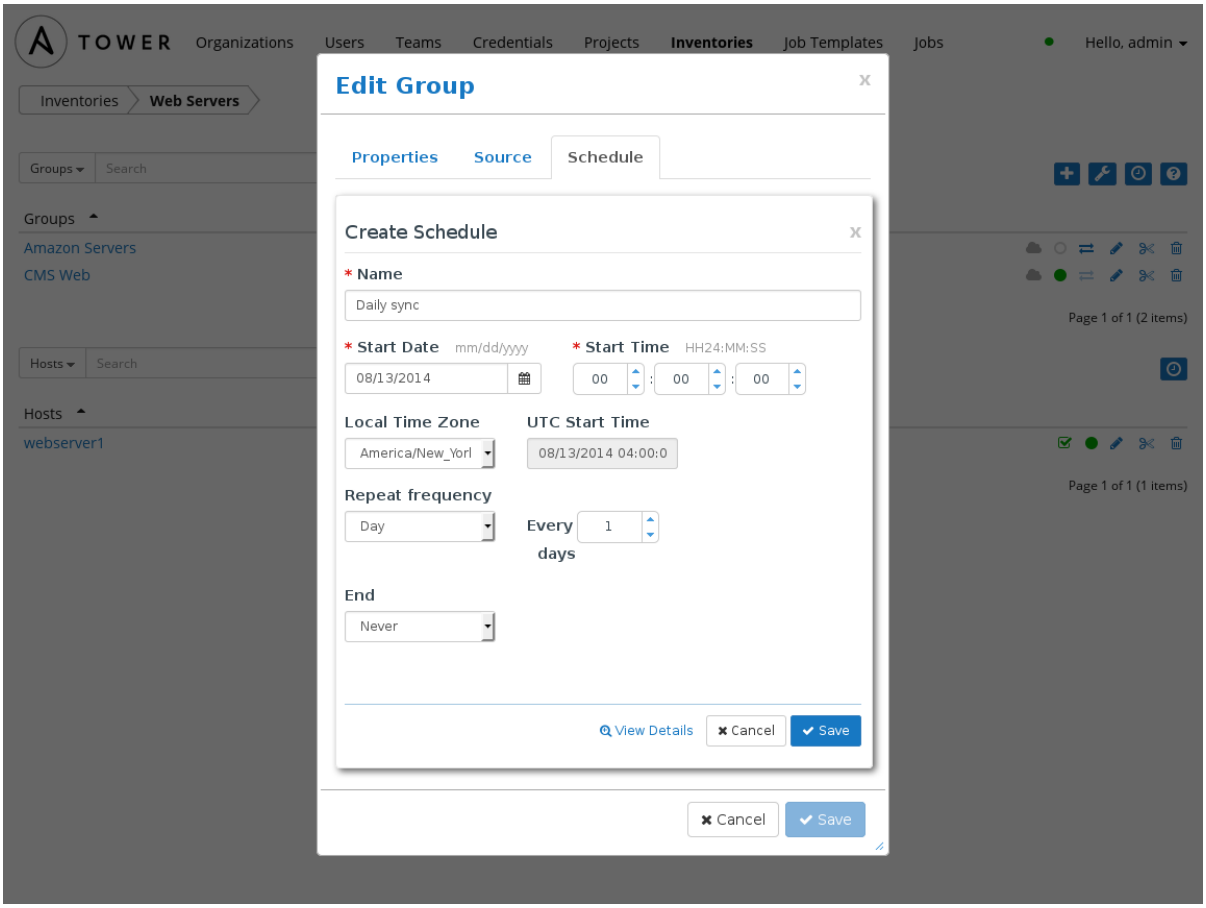
- Name - Clicking the schedule name will open the **Edit Schedule** dialog
- First Run
- Next Run

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

- Create a new schedule
- Refresh this view

Add a new schedule

To create a new schedule click the + button.



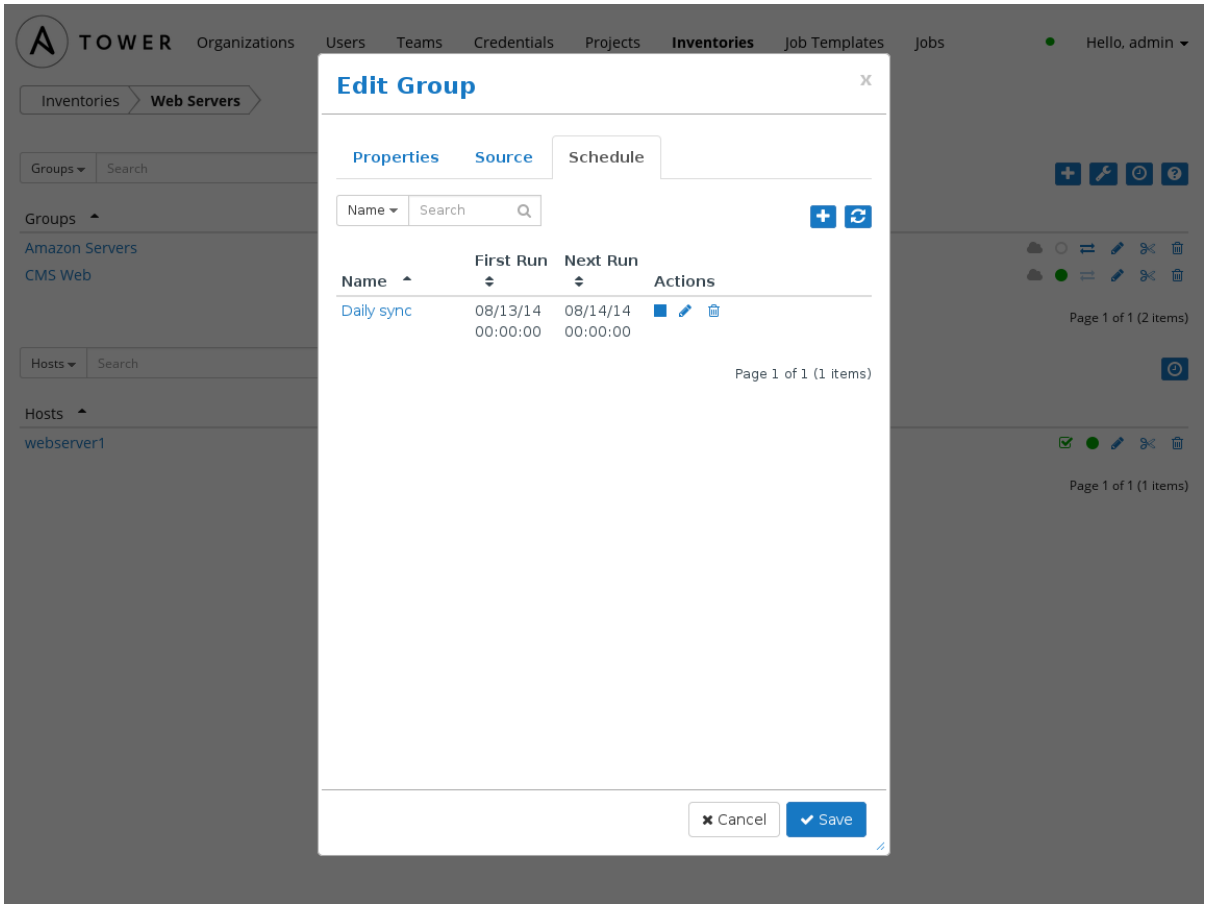
Enter the appropriate details into the following fields and select Save:

- Name (required)
- Start Date (required)
- Start Time (required)
- Local Time Zone (the entered Start Time should be in this timezone)
- UTC Start Time (calculated from Start Time + Local Time Zone)
- Repeat Frequency - the appropriate options 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 the selected Local Time Zone.

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

Once you've saved the schedule, it will be shown on the Schedule tab.

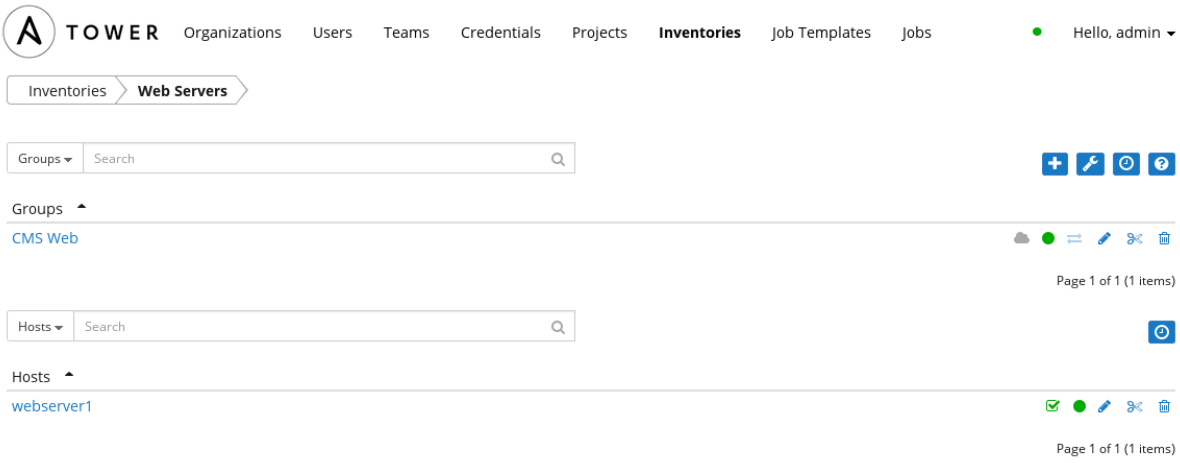


There are server actions available for schedules:

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

Hosts

Hosts are listed in the lower area of the Inventory display screen.



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

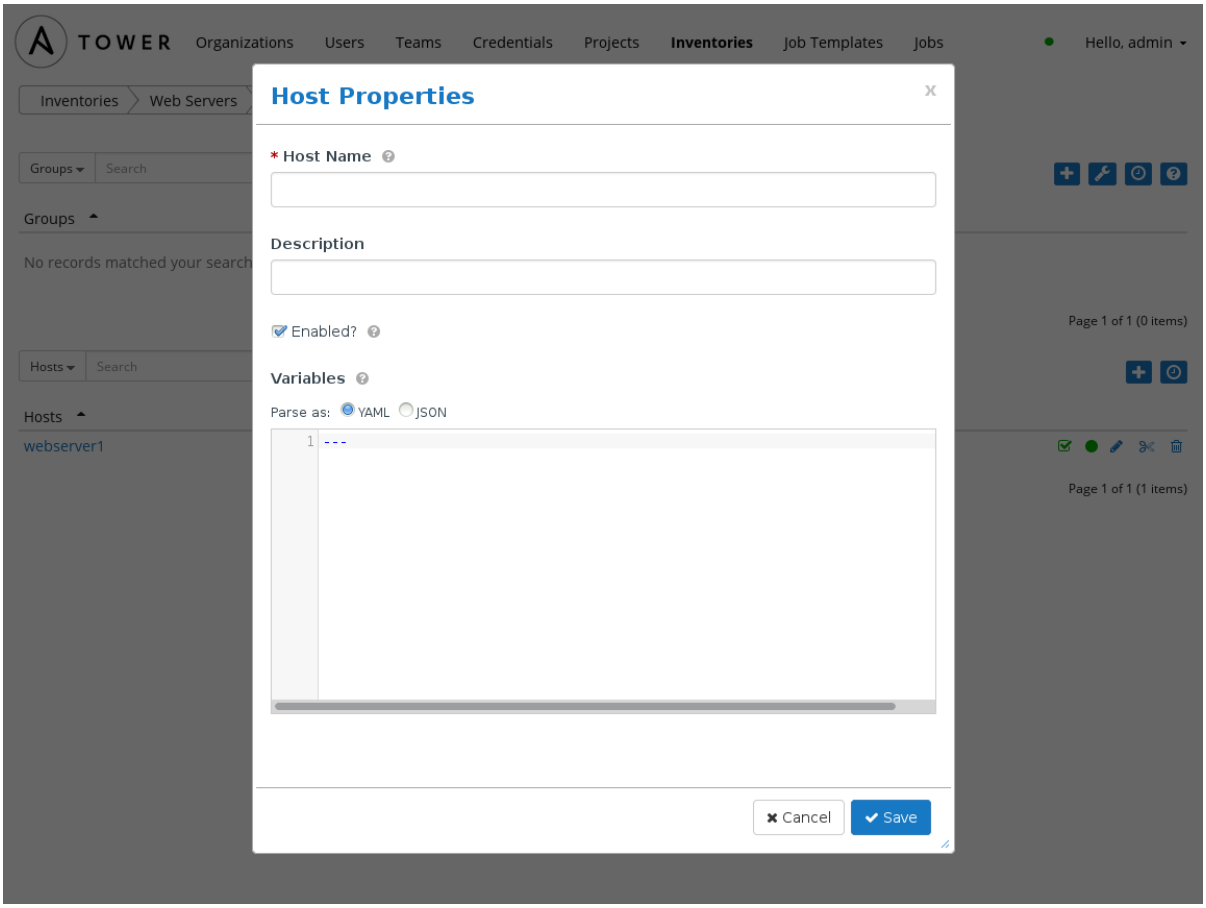
This list displays information about each host and provides for several actions:

- 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
- Copy host - Copies or moves the host to a different group
- 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**:

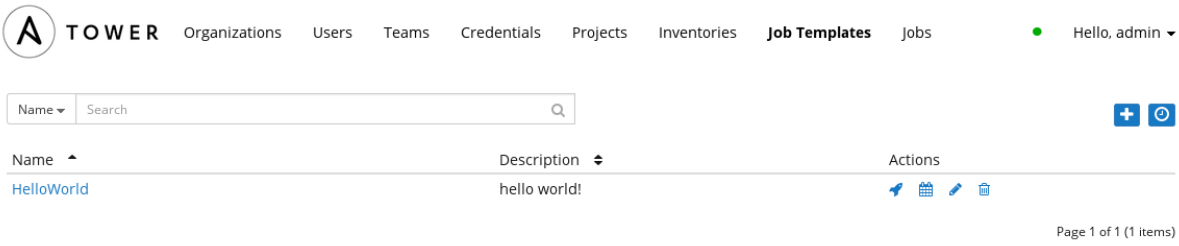
- Host Name - The hostname or IP address of the host
- Description
- Enabled? - Indicates if a host is available and should be included in running jobs. For hosts that are part of an external inventory, this flag cannot be changed. It will be set by the inventory sync process.
- Variables

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.

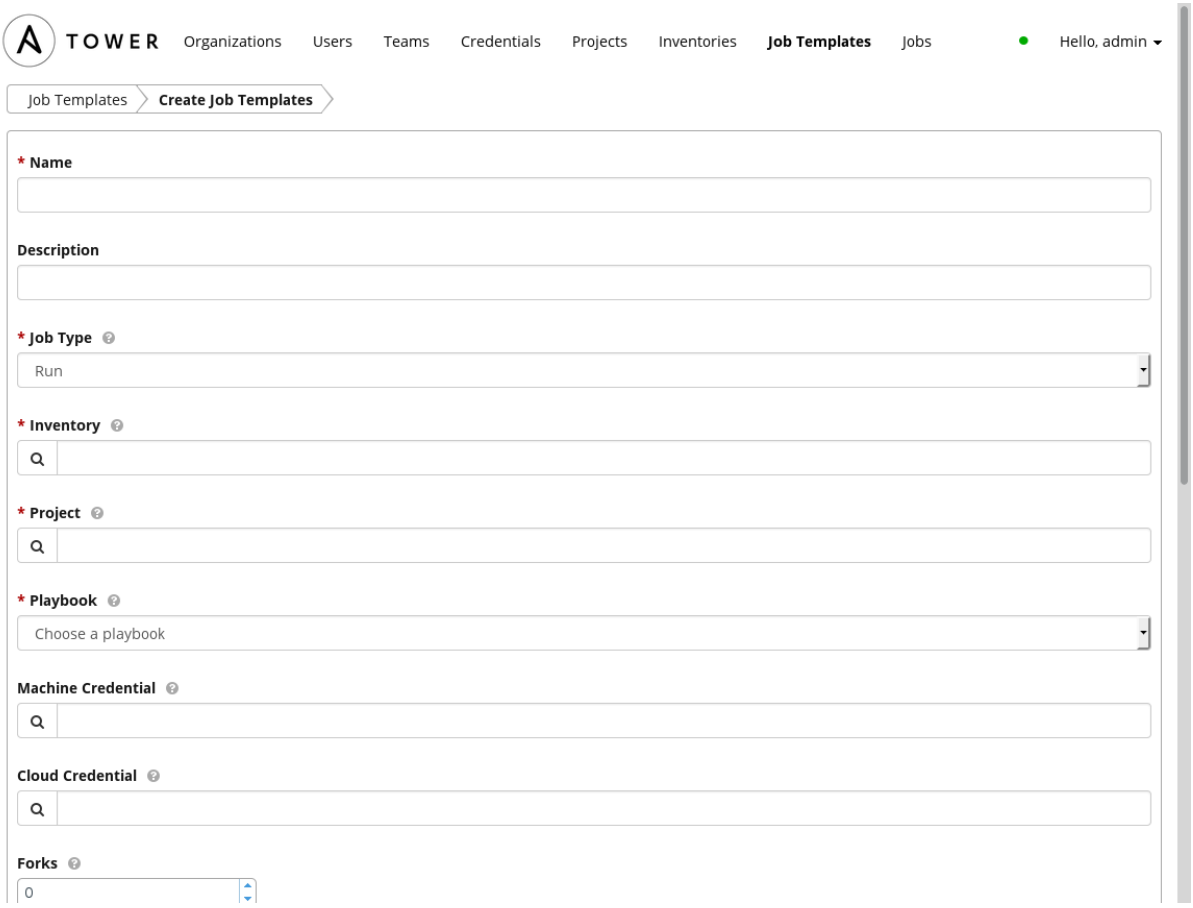
Job Templates

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



To create a new job template click the  button.



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.
- Project: Choose the project to be used with this job template from the projects available to the currently logged in Tower user.
- Playbook: Choose the playbook to be launched with this job template from the available playbooks. This menu is automatically populated with the names of the playbooks found in the project base path for the selected project. For example, a playbook named "jboss.yml" in the project path 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.

- Job Tags: A comma-separated list of playbook tags to constrain what parts of the playbooks will be executed.

  For more information and examples see Tags (http://docs.ansible.com/playbooks_tags.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
```

- Prompt for Extra Variables: If this is checked, the user will be prompted for Extra Variables at job execution. The set of extra variables will default to any Extra Variables already configured for the job template.
- Enable Survey: Survey the user on job launch. See Surveys below.
- Create Survey: Creates a survey, if the survey is enabled.
- Edit Survey: Edits the existing survey for this job template.
- Delete Survey: Deletes the existing survey for this job template.
- Allow Callbacks: Enable a host to call back to Tower via the Tower API and invoke the launch of a job from this job template. See Provisioning Callbacks, below.

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.

Surveys

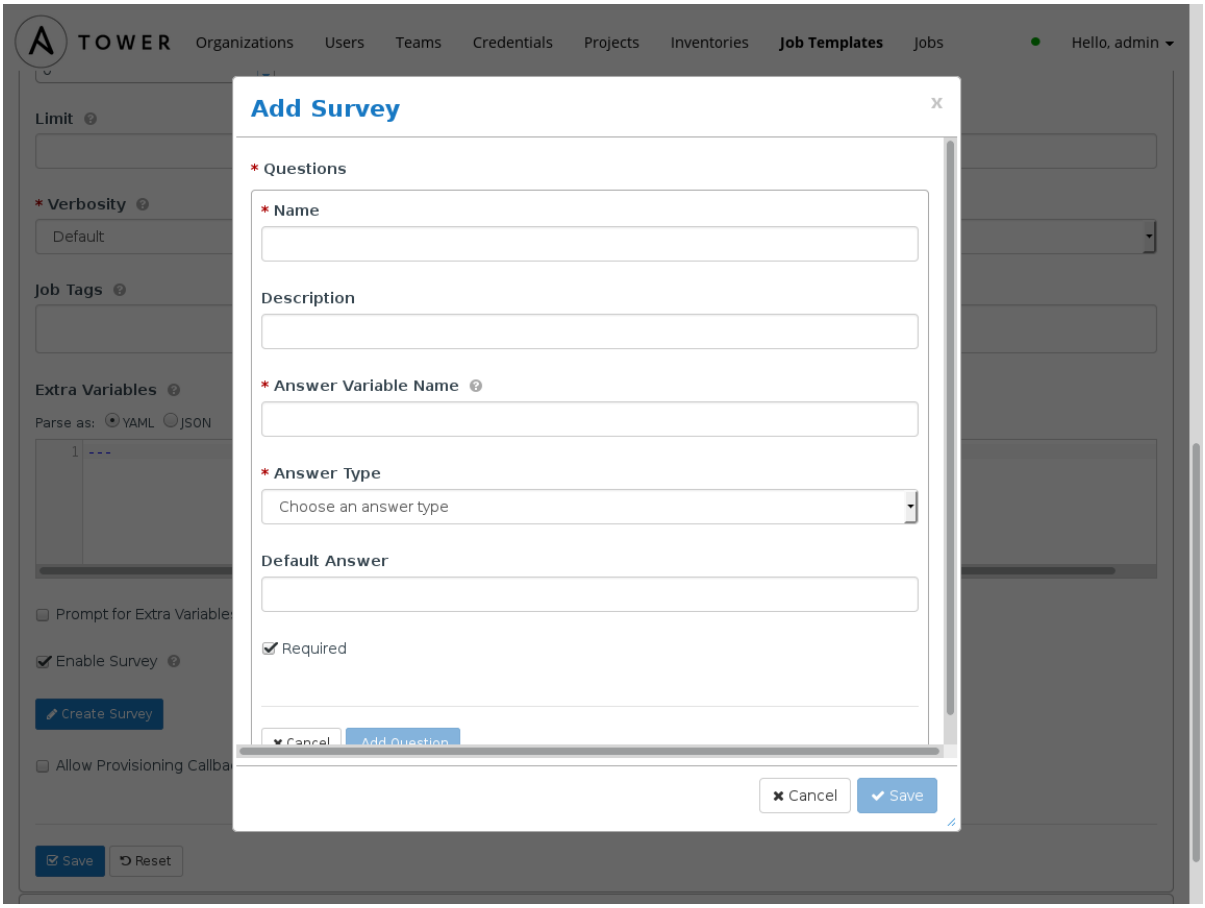
Surveys set extra variables for the playbook similar to 'Prompt for Extra Variables' does, but in a user-friendly question and answer way. Surveys also allows for validation of user input. If **Enable Survey** is checked, you will then see a button to **Create Survey**.

Use cases for surveys are numerous. An example might be if operations wanted to give developers a "push to stage" button they could run without advanced Ansible knowledge. This task, when launched could ask questions like "What tag should we release?".

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

Creating a Survey

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



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

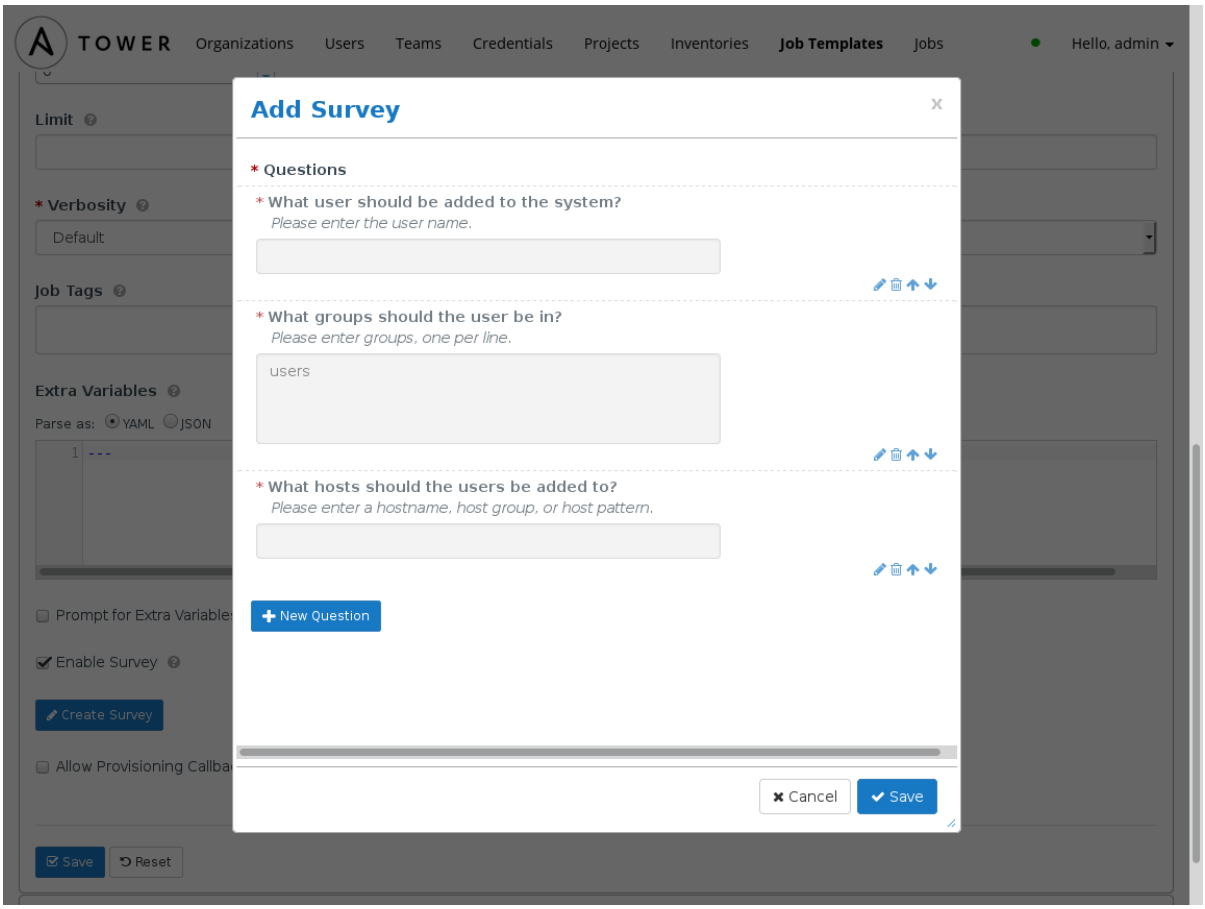
- **Name**: The question to ask the user
- **Description**: (optional) A description of what's being asked of the user.

- **Answer Variable Name**: The Ansible variable name to store the user's response in. This is the variable that will be used by the playbook. Variable names cannot contain spaces.
- **Answer Type**: Choose from the following question types.
  - *Text*: A single line of text. You can set the minimum and maximum length (in characters) that the answer can be.
  - *Textarea*: A multi-line text field. You can set the minimum and maximum length (in characters) that this answer can be.
  - *Multiple Choice (single select)*: A list of options, of which only one can be selected at a time. Enter the options, one per line, in the **Multiple Choice Options** box.
  - *Multiple Choice (multiple select)*: A list of options, any number of which can be selected at a time. Enter the options, one per line, in the **Multiple Choice Options** box.
  - *Integer*: An integer number. You can set the minimum and maximum value that the answer can be.
  - *Float*: A decimal number. You can set the minimum and maximum value that the answer can be.
- **Default Answer**: The default answer to the question. This value will be pre-filled in the interface, and will be used if the answer is not provided by the user.
- **Required**: Whether or not an answer to this question is required from the user.

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

You'll then see a stylized version of the survey, with a **New Question** button. Click this button if you want to add additional questions.

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



Click **Save** to save the survey.

Provisioning Callbacks

Provisioning callbacks are a feature of Tower that allow a host to initiate a playbook run against itself, rather than waiting for a user to launch a job to manage the host from the tower console. 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. This will display the **Provisioning Callback URL** for this job template.

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

Callbacks also require a Host Config Key, to ensure that foreign hosts with the URL cannot request configuration. Click the  button to create a unique host key for this callback, or enter your own key. The host key may be reused across multiple hosts to apply this job template against multiple hosts. Should you wish to control what hosts are able to request configuration, the key may be changed at any time.

To callback manually via REST, look at the callback URL in the UI, which is of the form: `http://<Tower server name>/api/v1/job_templates/1/callback/`. The '1' in this sample URL would be the job template ID in Tower.

The request from the host must be a POST. Here is an example using curl (all on a single line):

```
root@localhost:~$ curl --data "host_config_key=5a8ec154832b780b9bdef1061764ae5a" \
                    http:///api/v1/job_templates/1/callback/
```

The requesting host must be defined in your inventory for the callback to succeed. If Tower fails to locate the host either by name or IP address in one of your defined inventories, the request will be denied. Note that if your host is not in inventory, if `Update on Launch` is set for the inventory group, Tower will 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.

While the callback can be accessed via REST, the suggested method of using the callback is to use an example script that ships with Tower 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 one of the supported cloud providers. In these cases, along with setting *Update On Launch*, be sure to configure an inventory cache timeout for the inventory source, to avoid abusive hammering of your Cloud's API endpoints. Since the `request_tower_configuration.sh` script 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.

While we recommend against running the `request_tower_configuration.sh` script from a cron job, a suggested cron interval would be perhaps every 30 minutes. Repeated configuration can be easily handled by scheduling in Tower, so the primary use of callbacks by most users will be to enable that a base image is bootstrapped into the latest configuration upon coming online. To do so, running at first boot is a better practice. First boot scripts are just simple init scripts that typically self-delete, so you would set up an init script that called a copy of the request_tower_configuration script and bake that into an autoscaling image.

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 us at http://support.ansible.com/ for some suggestions and advice.

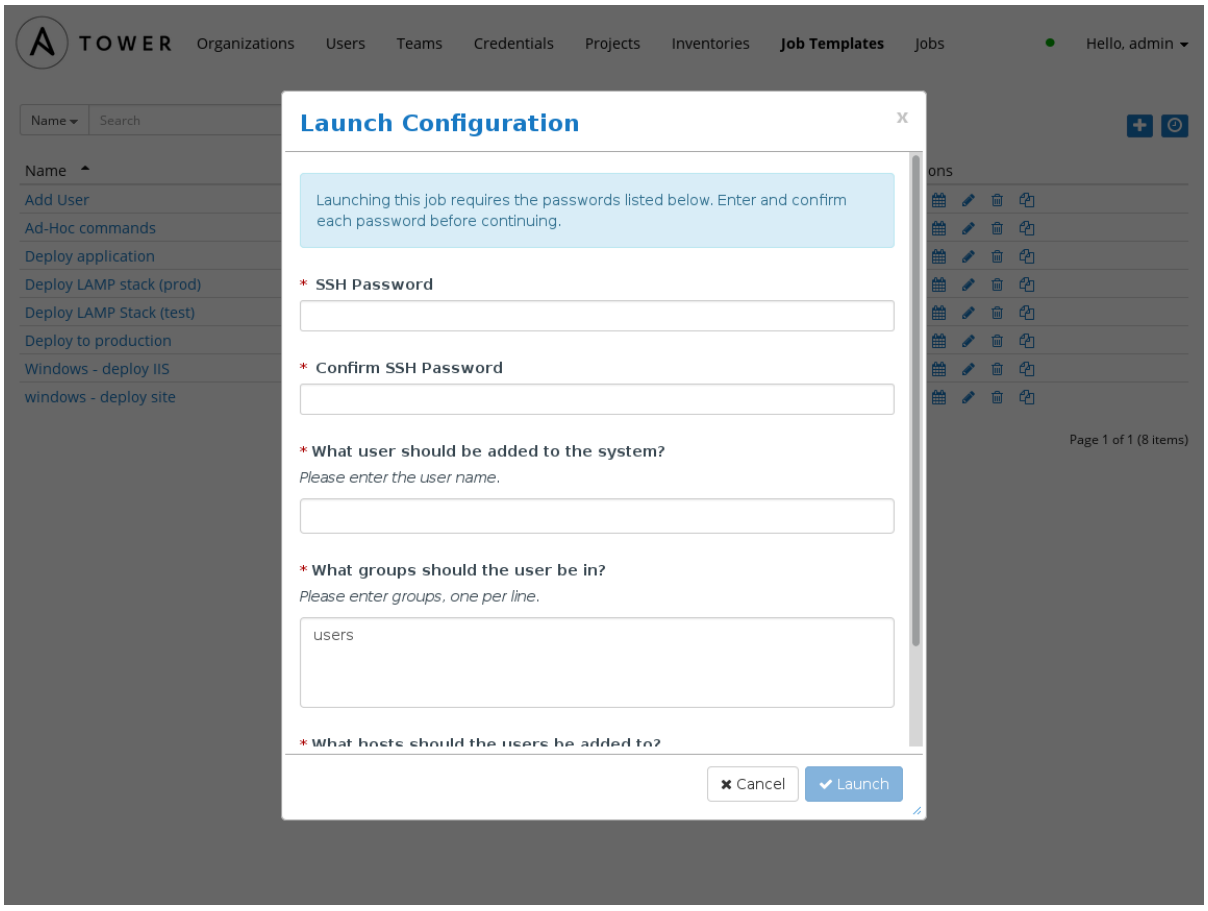
Launching Jobs

To launch a job template, click the button.

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

- Passwords or passphrases that have been set to **Ask**
- Survey, if one has been configured for the job templates
- Extra variables, if requested by the job template

Here is an example job launch that prompts for a SSH password, and runs the example survey created in Surveys.



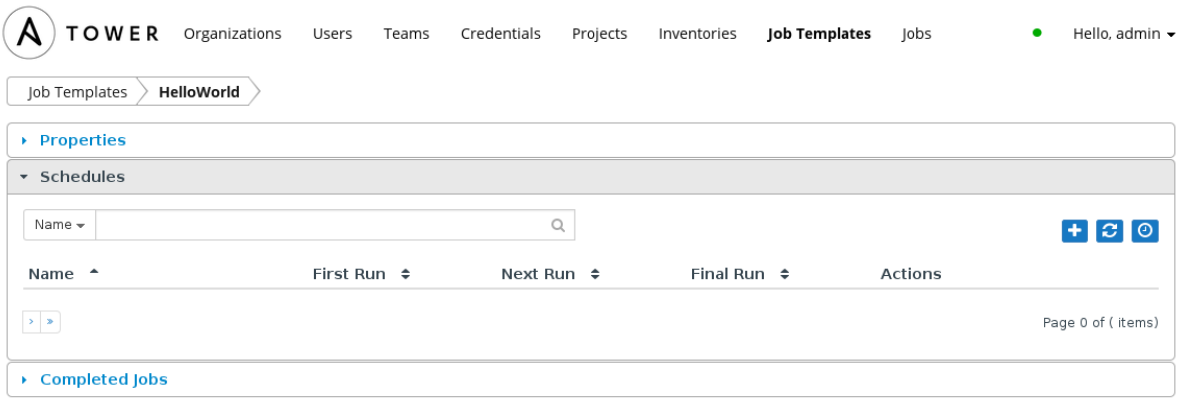
Along with any extra variables set in the job template and survey, Tower automatically adds the following variables to the job environment:

- `tower_job_id`: The Job ID for this job run
- `tower_job_launch_type`: One of *manual*, *callback*, or *scheduled* to indicate how the job was started
- `tower_job_template_id`: The Job Template ID that this job run uses
- `tower_job_template_name`: The Job Template name that this job uses
- `tower_user_id`: The user ID of the Tower user that started this job. This is not available for callback or scheduled jobs.
- `tower_user_name`: The user name of the Tower user that started this job. This is not available for callback or scheduled jobs.

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

Scheduling

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



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

The list of schedules includes:

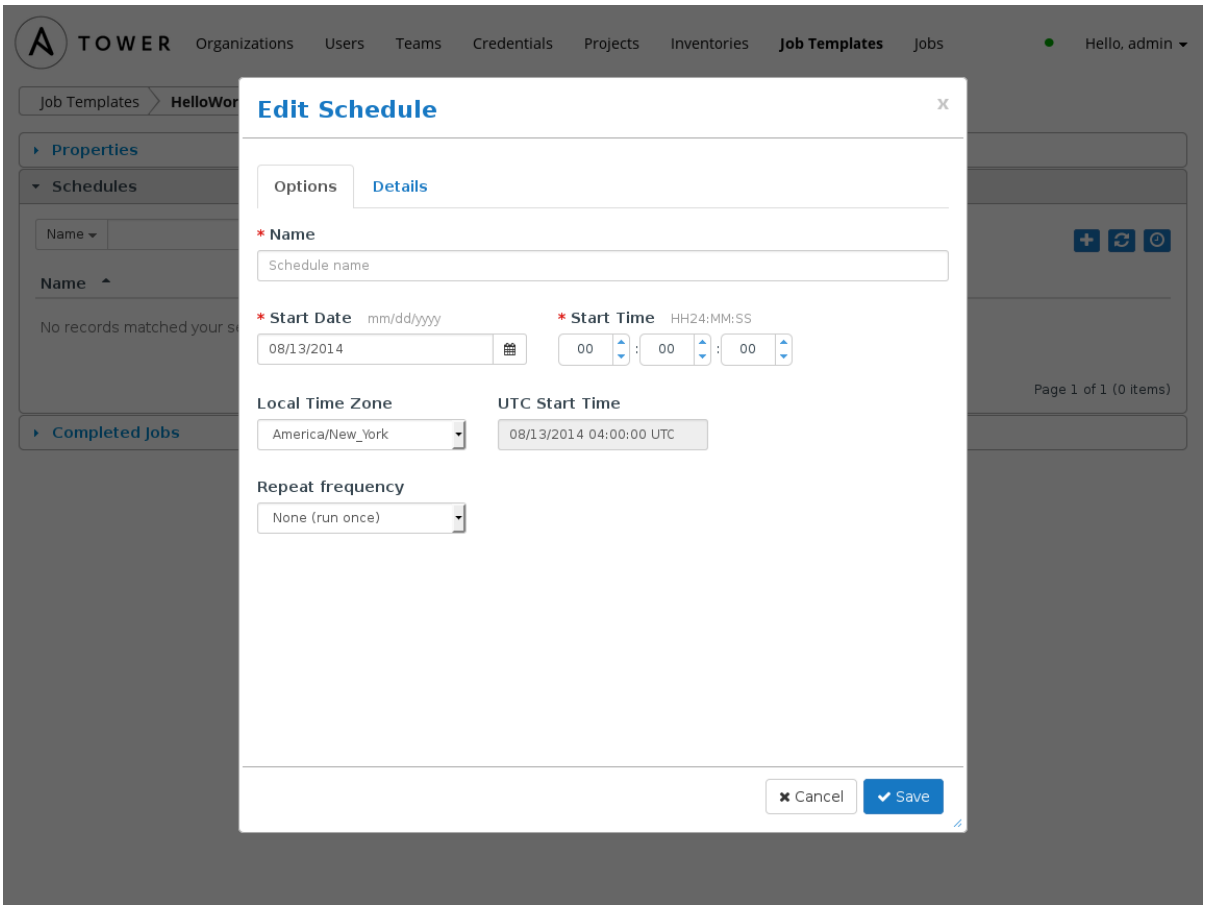
- Name - Clicking the schedule name will open the **Edit Schedule** dialog
- First Run - the first scheduled run of this task
- Next Run - the next scheduled run of this task
- Final Run - If the task has an end date, this is the last run of the task

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

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

Add a new schedule

To create a new schedule click the button.



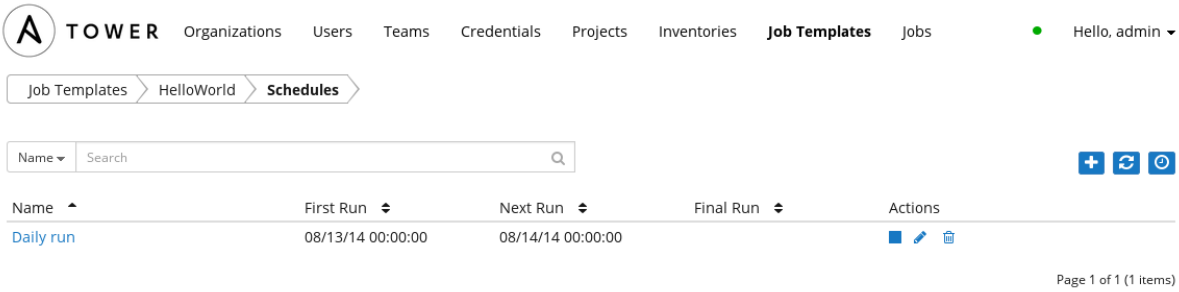
Enter the appropriate details into the following fields and select Save:

- Name (required)
- Start Date (required)
- Start Time (required)
- Local Time Zone (the entered Start Time should be in this timezone)
- UTC Start Time (calculated from Start Time + Local Time Zone)
- Repeat Frequency - the appropriate options 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 the selected Local Time Zone.

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

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

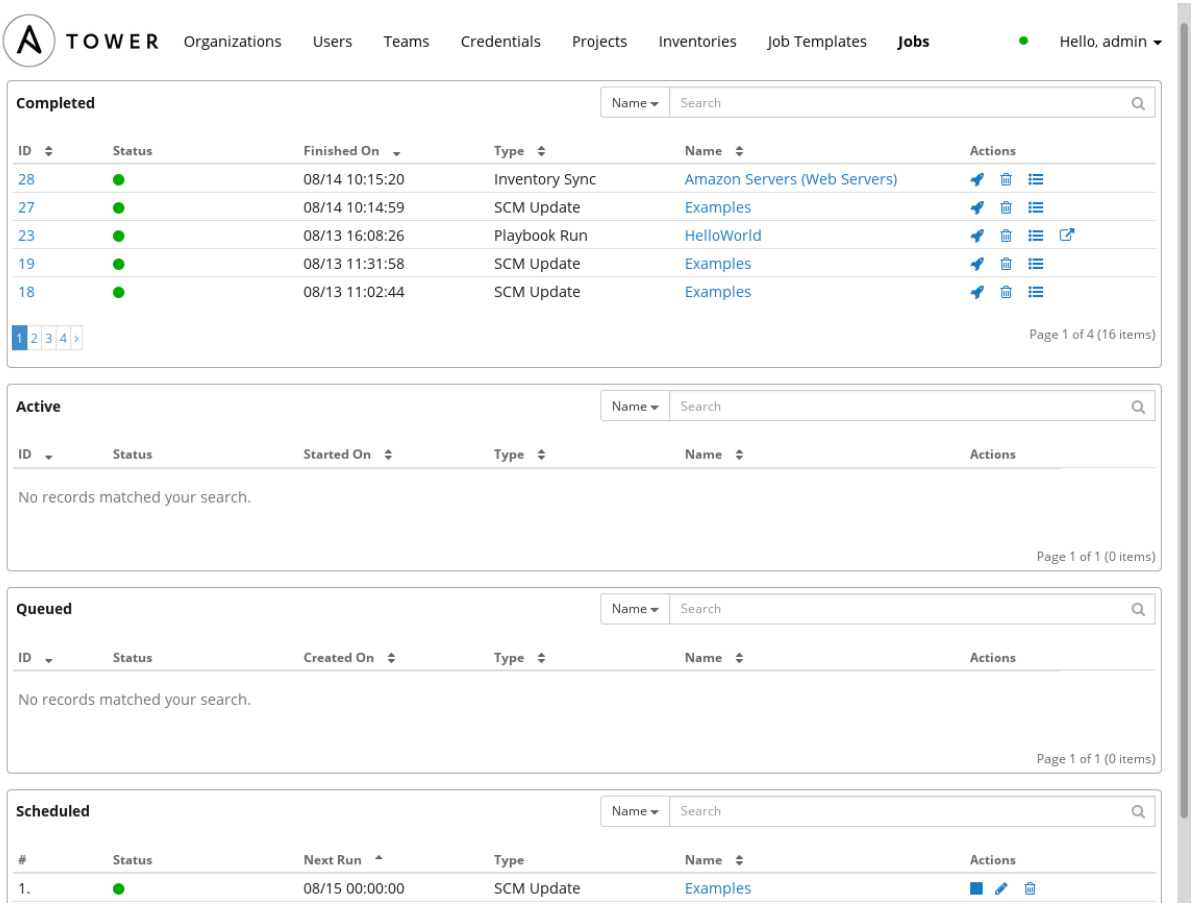


- Stop an active schedule or activate a stopped schedule
- Edit Schedule
- Delete schedule

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.



Completed Jobs

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

- **Job ID**: A unique integer that identifies a specific job.
- **Status**: Will be *Successful*, or *Failed* for completed jobs. Clicking this displays the job details. For a jobs of type **SCM Update** and **Inventory Sync**, clicking the name will open the Job Results window. For **Playbook Runs**, clicking the name will open the **Job** display for that job.
- **Finished On**: The date and time the job finished, in the server's local time.
- **Type**: The type of the job. Jobs can be **Inventory Sync** (for cloud inventory sources), **SCM Update** (for projects under source control), and **Playbook Run**.
- **Name**: Clicking this displays the job details.
- **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. This will not delete the job template, nor does it undo any changes an **Inventory Sync** or **SCM Update** job has done.
- **View Job Details**: This gives a view of the job details, in the same way as clicking on the **Name** or **Status** of the job.
  - **View Standard Output**: For jobs of type **Playbook Run**, this opens the Standard Out view of the Job Details page for that job.

Active Jobs

The list of **Active** jobs may be searched by **Job ID** or **Name**, and filtered by **Type**.

- **Job ID**: A unique integer that identifies a specific job.
- **Status**: Will be *Running* for active jobs. Clicking this displays the job details.
- **Started On**: The date and time the job started, in the server's local time.
- **Type**: The type of the job.
- **Name**: Clicking this displays the job details.
- **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.)
  - **Cancel**: This button cancels the job run.
  - **View Job Details**: This gives a view of the job details, in the same way as clicking on the **Name** or **Status** of the job.
    - **View Standard Output**: For jobs of type **Playbook Run**, this opens the Standard Out view of the Job Details page for that job.

Queued Jobs

Queued jobs are pending jobs that are queued to be run but are not currently active. The list of **Queued** jobs may be searched by **Job ID** or **Name**, and filtered by **Type**.

- **Job ID**: A unique integer that identifies a specific job.

- **Status**: Will be *pending* for queued jobs. Clicking this displays the job details.
- **Created On**: The date and time the job was created, in the server's local time.
- **Type**: The type of the job.
- **Name**: Clicking this displays the job details.
- **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.)
    - **Cancel**: This button cancels the job run.
    - **View Job Details**: This gives a view of the job details, in the same way as clicking on the **Name** or **Status** of the job.

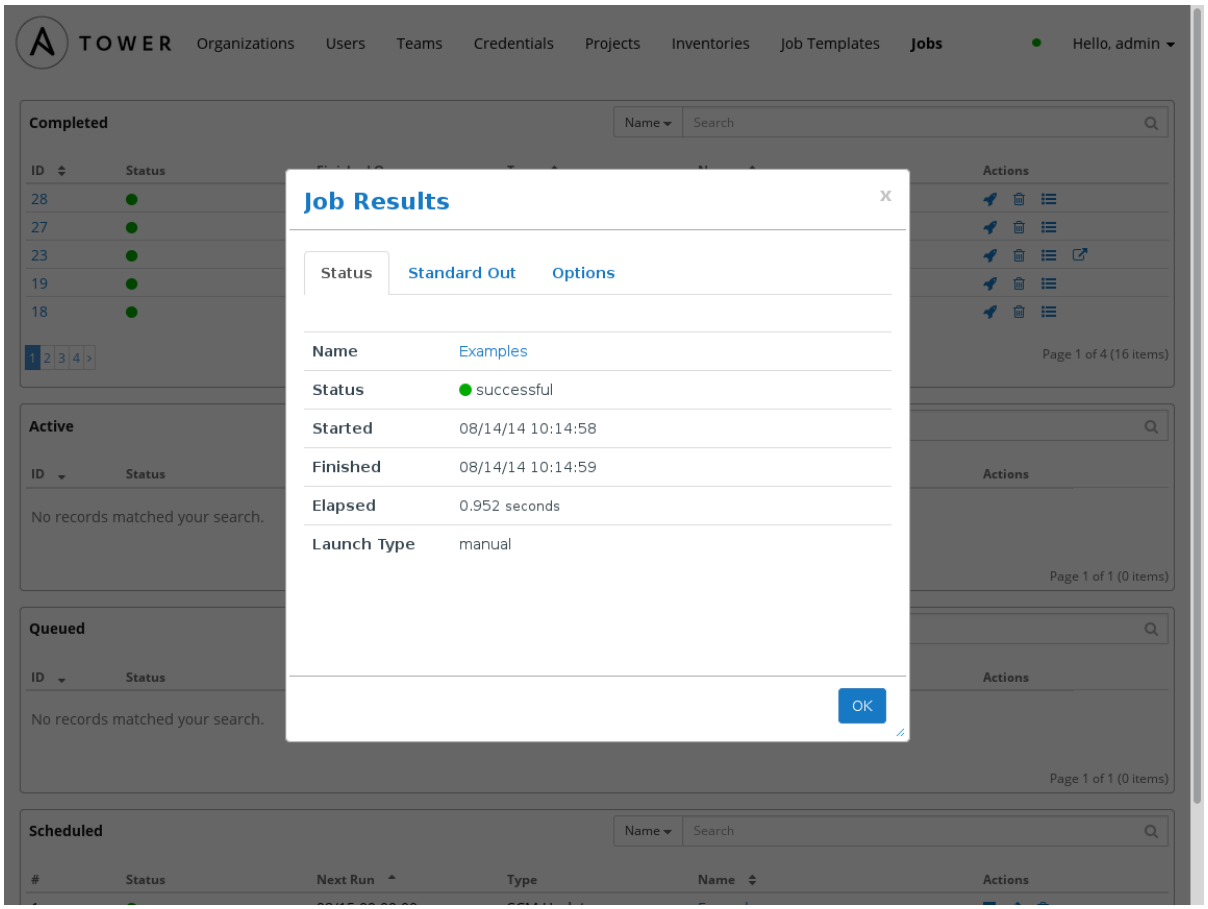
Scheduled Jobs

The list of **Scheduled** jobs may be searched by **Name**.

- **Number**: The order of the scheduled job.
- **Status**: Shows whether the schedule is active and will run. Click to change the status the scheduled job.
- **Next Run**: When the job is scheduled to run.
- **Type**: One of **SCM Update**, **Inventory Sync**, or **Playbook Run**.
- **Name**: Clicking the name will open the schedule for this job, where it can be updated.
- **Actions**:
  - **Play/Stop**: Activate or deactivate the scheduled job.
  - **Edit**: Edit the job schedule. To edit the job itself, you will need to edit it from the **Inventory**, **Project**, or **Job Template** views.
  - **Delete**: Delete the scheduled job.

Job Results

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



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

- **Name**: The name of the job template from which this job was launched.
- **Status**: Can be any of *Pending*, *Running*, *Successful*, or *Failed*.
- **License Error**: Only shown for **Inventory Sync** jobs. If this is *True*, the hosts added by the inventory sync caused Tower to exceed the licensed number of managed hosts.
- **Started**: The timestamp of when the job was initiated by Tower.
- **Finished**: The timestamp of when the job was completed.
- **Elapsed**: The total time the job took.
- **Launch Type**: *Manual* or *Scheduled*.

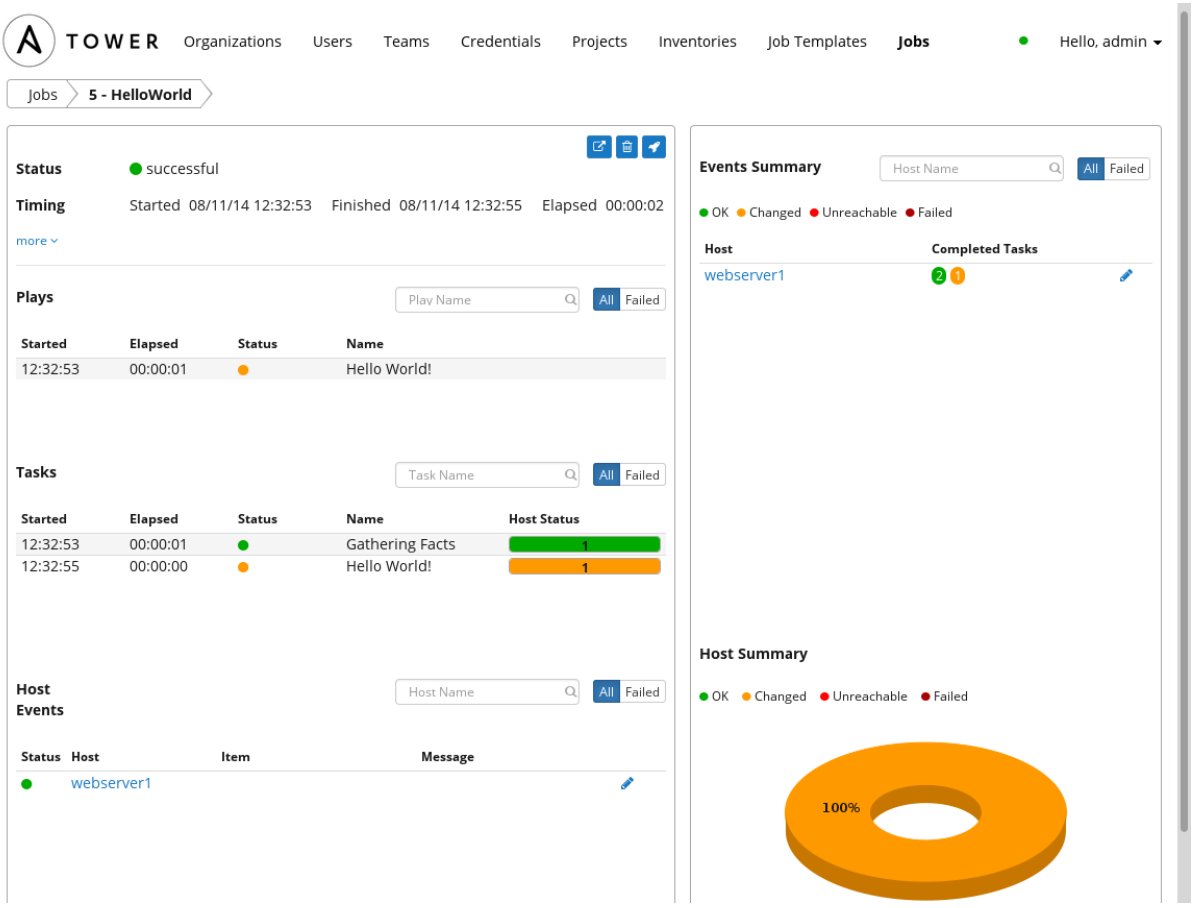
The **Standard Out** tab shows the full results of running the SCM Update or Inventory Sync playbook. This shows the same information you would see if you ran the Ansible playbook using Ansible from the command line, and can be useful for debugging.

The **Options** tab describes the details of this job. For SCM Update jobs, this consists of the **Project* associated with the job. For Inventory Sync jobs, this consists of:

- **Credential**: The cloud credential for the job
- **Group**: The group being synced
- **Source**: The type of cloud inventory
- **Regions**: Any region filter, if set
- **Overwrite**: The value of *Overwrite* for this Inventory Sync. See the Inventory section of this manual for details.
- **Overwrite Vars**: The value of *Overwrite Vars* for this Inventory Sync. See the Inventory section for this manual for details.

Job

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



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

Status

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

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

- the **Job Template** for this job
- the job **Type**: *Run* or *Check*
- the **Inventory** associated with this job
- the **Project** associated with this job
- the playbook that is being run
- the **Credential** in use
- any *Limit* settings for this job
- the *Verbosity* setting for this job
- any *Extra Variables* that this job used

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

Plays

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

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

Tasks

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

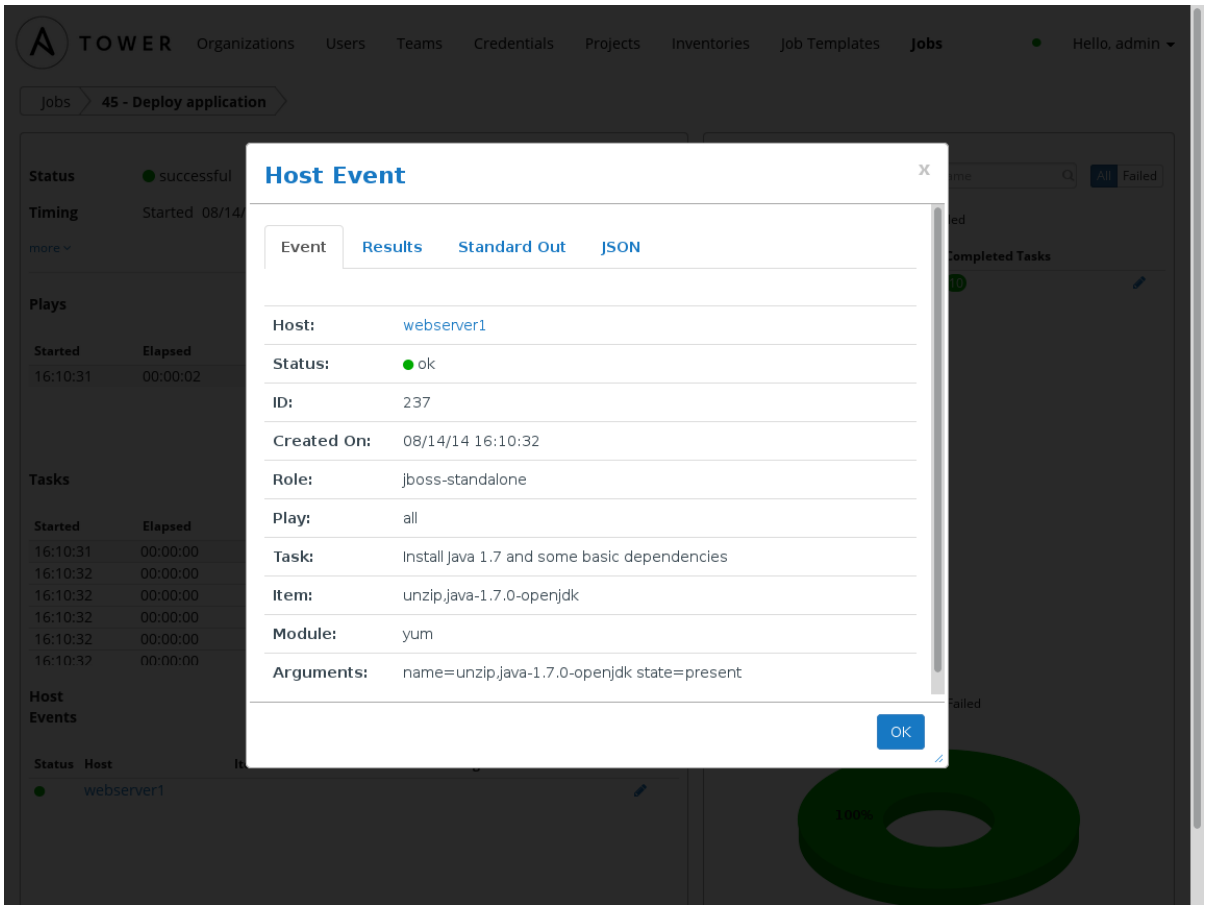
For each task, Tower shows the start time for the task, the elapsed time of the task, the task **Name**, whether the task succeeded or failed, and a summary of the host status for that task. The host status displays a summary of the hosts status for all hosts touched by this task. Host status can be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

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

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

Host Events

The **Host Events** area shows hosts affected by the selected play and task. For each host, Tower shows the host's status, its name, and any **Item** or **Message** set by that task. You can click on the button to edit the host. Clicking on the host brings up the **Host Event** dialog for that host and task.



.

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

- **Host**
- **Status**
- a unique **ID**
- the time this task was started
- the **Role** for this task
- the name of the **Play**
- the name of the **Task**
- if applicable, the Ansible *module* for the task, and any *arguments* for that module

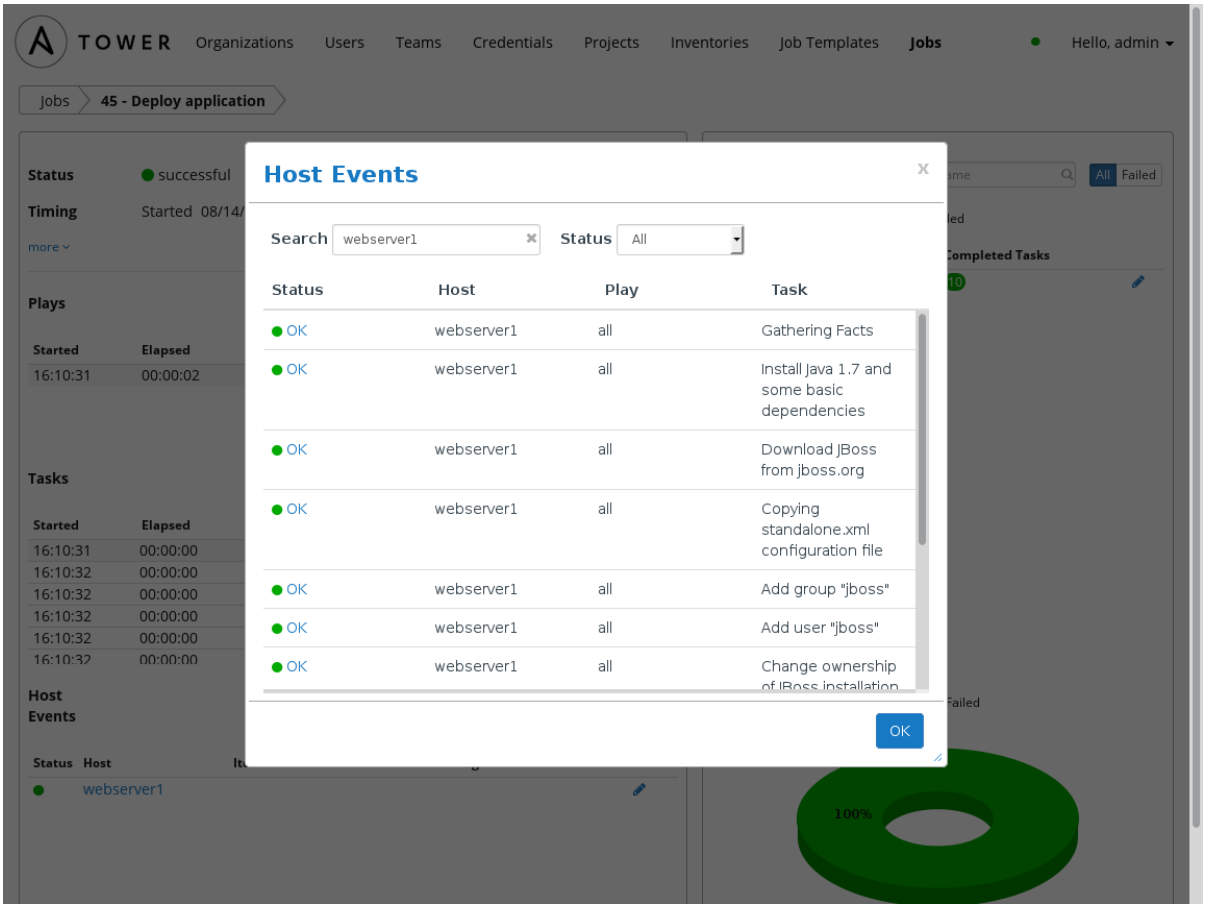
The **Host Event** dialog includes a **Results** tab that shows the results of this task. The fields displayed on this tab will be specific to the task and the module used. There is also a **JSON** tab that displays the result in JSON format.

Events Summary

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

For each host, the *Events Summary** area shows the host name and the number of completed tasks for that host, sorted by status. You can also click on the button to edit the host.

Clicking on the host name brings up a **Host Events** dialog that shows all tasks that affected that host.



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

Host Summary

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

Portal Mode

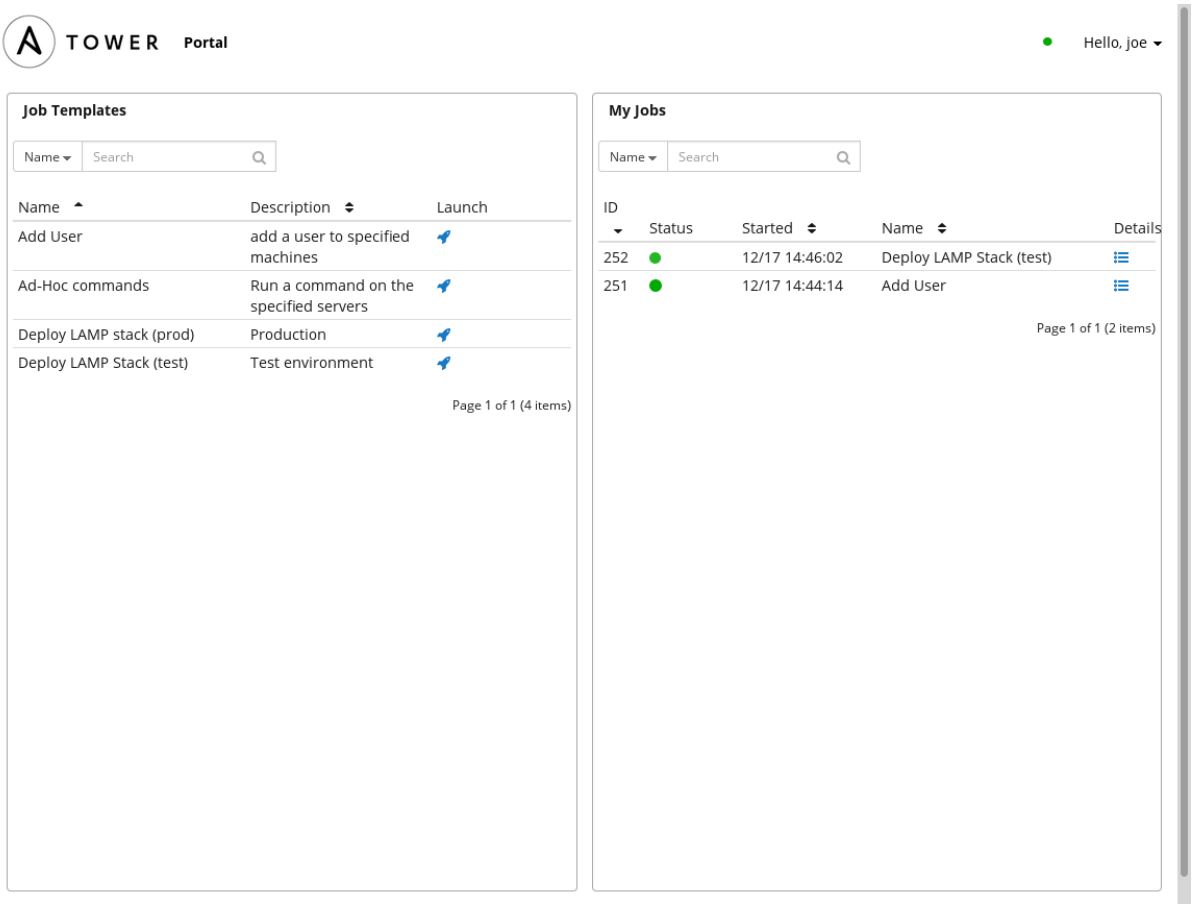
Portal mode is a simplified interface for users that need to run Ansible jobs, but that don't need an advanced knowledge of Ansible or Tower.

Portal mode could be used by, for instance, development teams, or even departmental users in non-technical fields.

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

Pressing the "rocket" beside a job in portal mode will launch it, potentially asking some survey questions.

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



Portal mode can be accessed in two ways

- via the **Portal Mode** option in the user menu at the top right of the Tower interface
- by navigating to `https://<Tower server name>/portal`

In Portal mode, the top bar of Tower only has the user menu, where the user can either **Exit Portal** to the main interface, or **Logout**. Portal mode shows two columns:

Job Templates

This shows the job templates that are available for the user to run. This list can be searched by **Name** or **Description**, and can be sorted by those keys as well.

To launch a job template, click the button. This will launch the job, which will then be seen in **My Jobs**.

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

My Jobs

This shows the list of jobs that this user has run in the past.

For each job, it lists the Job **ID**, the **Status** of the job (*Running*, *Pending*, *Successful*, or *Failed*), its start time, and the job **Name**. The job list can be sorted by any of these fields. Clicking on the Details button will open a new window with the **Job Details** for that job.

Best Practices

Use Source Control

While Tower supports playbooks stored directly on the Tower server, best practice is to store your playbooks, roles, and any associated details in source control. This way you have an audit trail describing when and why you changed the rules that are automating your infrastructure. Plus, it allows for easy sharing of playbooks with other parts of your infrastructure or team.

Ansible file and directory structure

Please review the Ansible best practices from the Ansible documentation at http://docs.ansible.com/playbooks_best_practices.html. If creating a common set of roles to use across projects, these should be accessed via source control submodules, or a common location such as `/opt`. Projects should not expect to import roles or content from other projects.

Playbooks should not use the `vars_prompt` feature, as Tower does not interactively allow for `vars_prompt` questions. If you need this functionality, use the Survey functionality of Tower.

Jobs run in Tower use the playbook directory as the current working directory, although jobs should be coded to use the `playbook_dir` variable rather than relying on this.

Use Dynamic Inventory Sources

If you have an external source of truth for your infrastructure, whether it's a cloud provider or a local CMDB, it is best to define an inventory sync process and use Tower's support for dynamic inventory (including cloud inventory sources and custom inventory scripts). This ensures your inventory is always up to date.

Variable Management for Inventory

Keeping variable data along with the objects in Tower (see the inventory editor) is encouraged, rather than using `group_vars/` and `host_vars/`. If you use dynamic inventory sources, Tower can sync such variables with the database as long as the **Overwrite Variables** option is not set.

Autoscaling

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

Larger Host Counts

Consider setting "forks" on a job template to larger values to increase parallelism of execution runs. For more information on tuning Ansible, see the Ansible blog (http://www.ansible.com/blog/ansible-performance-tuning).

Continuous integration / Continuous Deployment

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

Security Notes

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

For credential security, users 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. Additionally, uploaded credentials are kept encrypted in the database and are not surfaced to API or UI requestors.

Installation and Setup Reference

Supported Platforms and Requirements

Please note the Requirements and Prerequisites for Tower installation.

Note that the Tower installation must be run from an internet connected machine that can install software from trusted 3rd-party places such as Ansible's software repository, and your OS vendor's software repositories. In some cases, access to the Python Package Index (PyPI) is necessary as well. If you need to be able to install in a disconnected environment, please contact Ansible support at support.ansible.com (https://support.ansible.com/).

General Installation Notes

- If you need to access a HTTP proxy to install software from your OS vendor, ensure that the environment variable "HTTP_PROXY" is set accordingly before running `setup.sh`.
- The Tower installer creates a self-signed SSL certificate and keyfile at `/etc/tower/awx.cert` and `/etc/tower/awx.key` for HTTPS communication. These can be replaced after install with your own certificates if you desire, but the filenames are required to be the same.
- If using Ansible 1.8 or later, ensure that fact caching using Redis is not enabled in `ansible.cfg` on the Tower machine.

Platform-Specific Notes

Red Hat Enterprise Linux and CentOS

- PackageKit can frequently interfere with the installation/update mechanism. Consider disabling or removing PackageKit if installed prior to running the setup process.
- Only the "targeted" SELinux policy is supported. The targeted policy can be set to disabled, permissive, or enforcing.

- For users of Red Hat Enterprise Linux 7 or CentOS 7, you will need to either disable the 'firewalld' service (if active), or modify the firewalld configuration to allow incoming connections on ports 80, 443, and 8080. This will be fixed in a future release.

Tower Installation Scenarios

Tower can be installed in three scenarios.

- Single Machine integrated installation

This is a single machine install of Tower - the web frontend, REST API backend, and database are all on a single machine. This is the standard installation of Tower. It also installs PostgreSQL from your OS vendor repository, and configures the Tower service to use that as its database.

- Single Machine with an external database

This installs the Tower server on a single machine, and configures it to talk to a remote instance of PostgreSQL as its database. This remote PostgreSQL can be a server you manage, or can be provided by a cloud service such as Amazon RDS.

Tower will not configure replication or failover for the database that it uses, although Tower should work with any replication that you have.

NOTE: *The database server should be local to the Tower server for performance reasons.*

- High Availability Multi-Machine with an external database

Tower can run in an active-passive high-availability mode. In this mode, Tower will run with one 'primary' node active at any time, and any number of passive 'secondary' nodes that can be made active if necessary.

NOTE: *Running in a high-availability setup requires the use of an external database.*

Each of these scenarios can be configured through the Tower Installation Wizard.

Get the Tower Installer

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.

Download this tarball, and extract it. 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., "2.1".

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

The Tower Installation Wizard

The Tower setup process consists of two parts: an installation wizard that determines your Tower configuration, and then a setup playbook that uses that information to install Tower.

The Tower Installation Wizard and the Tower setup playbook do not need to be run from the system that will be running Tower, although they can. The Tower Installation Wizard will ask for credentials needed to access external systems where necessary.

The Tower Installation Wizard is invoked as `configure` from the path where you unpacked the Tower installer tarball. It writes a file called `tower_setup_conf.yml` that contains the configuration for Tower. The wizard takes the following arguments:

- `-h`, `--help`

  Displays a brief usage summary.
- `-l`, `--local`

  Assumes that you are installing Tower on the local machine where you are running `configure`. This implies an internal embedded PostgreSQL database as well. This option skips some questions in the wizard.
- `--no-secondary-prompt`

  Assumes you are not installing in a high-availability setup. This option skips some questions in the wizard.
- `-A`, `--no-autogenerate`

  Do not autogenerate random passwords for PostgreSQL or Redis - prompt the user for them instead.

- `-o FILE`, `--options-file=FILE`

  Use the file `FILE` as a source of answers. This can be the `tower_setup_conf.yml` file from a previous run of the wizard. Depending on the contents of the file, this option will skip some questions in the wizard.

Once you invoke the Tower Installation Wizard, you'll be asked about the configuration of a few different items.

Primary Tower machine configuration

First, the Tower wizard asks about where you intend to place the the primary (or only) Tower instance.

```
root@localhost:~$ ./configure
-------------------------------------------
Welcome to the Ansible Tower Install Wizard
-------------------------------------------

PRIMARY TOWER MACHINE
Tower can be installed (or upgraded) on this machine, or onto a remote machine
that is reachable by SSH.

Note: If using the High Availability features of Tower, you must use DNS
resolvable hostnames or IP addresses (do not use "localhost").

Enter the hostname or IP to configure Ansible Tower
(default: localhost):
```

If you are installing on the current machine, enter `localhost` or `127.0.0.1` for the current machine. If you are installing on a different machine, enter the IP address or hostname of the machine. This machine must be running and accessible via SSH when running the setup playbook later.

Configuring the Database

```
DATABASE
Tower can use an internal database installed on the Tower machine, or an
external PostgreSQL database. An external database could be a hosted database,
such as Amazon's RDS.

An internal database is fine for most situations. However, to use the High
Availability features of Tower, an external database is required.

If using an external database, the database (but not the necessary tables) must
already exist.

Will this installation use an (i)nternal or (e)xternal database?
```

Enter `i` for an internal database on the same machine as Tower, or `e` for an external database. To run Tower in a high-availability configuration, you must use an external database.

If you choose to use an external database, the wizard will prompt you for the following additional database parameters:

- Database host to connect to
- Database name
- PostgreSQL user to use to access the database
- Password for the above PostgreSQL user
- Port to connect to the PostgreSQL database on (hit enter for the default PostgreSQL port)

The wizard will attempt to verify these parameters if your system has the PostgreSQL client libraries installed.

Secondary Installation (if applicable)

At this time, if you've chosen an external database, you can configure any secondary Tower instances if you so desire.

```
SECONDARY MACHINES
You may optionally elect to add any number of secondary machines, on which
Ansible Tower will also be installed (in secondary mode).
Add secondary machines (y/n)?
```

Enter `y` to configure additional secondary Tower instances.

Enter the hostname or IP of secondary machines. If you are done adding
machines, enter an empty line.
Hostname or IP:

Enter the hostnames or IP addresses of machines you want to configure as secondary Tower instances, one at a time. Enter a blank line to end the list. These machines must be running and accessible via SSH when running the setup playbook later.

Passwords

You are then prompted for the passwords you need for various Tower services.

```
PASSWORDS
For security reasons, since this is a new install, you must specify the
following application passwords.
```

The installation wizard will ask you for the following passwords:

- Admin Password
  This is the password for 'admin', the first user (and superuser) created on installation. You'll need this password for your initial login to Tower.
- Munin Password
  This password is used by Tower superusers to access the Munin-based monitoring of your Tower server.

If you passed the `-A` or `--no-autogenerate` parameters to the Installation Wizard, you will additionally be prompted for a PostgreSQL password and a Redis password. These are used internally to Tower and not needed by the admin at runtime; therefore, they are normally autogenerated as a random value.

Connection Information

If you chose to install on machines other than the current machine you are running the installation wizard on, you are now prompted for details on how to connect to those machines.

```
CONNECTION INFORMATION
Enter the SSH user to connect with (default: root):
```

First, you are prompted for the user to SSH to the remote hosts with. If this user is not root, you are prompted for how you will escalate privileges.

```
Root access is required to install Tower.
Will you use (1) sudo or (2) su?
```

Choose either `1` or `2` to configure sudo or su access. If you enter that you need a password for sudo or su access, this will be prompted for during the setup playbook run.

You are then prompted for SSH key information. If you are using a SSH key to access this host, you are prompted for the path to the SSH private key to use.

The same connection and su/sudo information will be used for all machines that are configured by the setup playbook, whether primary or secondary nodes. If you need different connection information for different machines, this can be configured by manually modifying the `inventory` file generated by the Installation Wizard.

Review and Confirm

You are then asked to review the settings you entered. An example would be:

```
REVIEW
You selected the following options:

The primary Tower machine is: tower.example.com
Tower will operate on an EXTERNAL database.
  host: database.example.com
  database: mydb
  user: db_admin
  password: ********
  port: 5432
Additional secondary machines:
  - tower-backup.example.com
  - tower-backup2.example.com
Using SSH user: jdoe

Are these settings correct (y/n)?
```

Select `y`, and you will then be given some information on running the setup playbook.

```
FINISHED!
You have completed the setup wizard. You may execute the installation of
Ansible Tower by issuing the following command:

# Add your SSH key to SSH agent.
# You may be asked to enter your SSH unlock key password to do this.
ssh-agent bash
ssh-add ~/.ssh/id_my-example-key
./setup.sh -s
```

Reviewing the Tower configuration

The Tower configuration is written into two files by the Tower Installation Wizard.

- `tower_setup_conf.yml`
  This Tower configuration file contains needed Tower passwords, database connection information, and machine connection information.
- `inventory`
  This includes the machines that the setup playbook will operate on, grouped into the `primary` and `secondary` groups of nodes.

The Setup Playbook

The Tower setup playbook is invoked as `setup.sh` from the path where you unpacked the Tower installer tarball. It uses the `tower_setup_conf.yml` and `inventory` files written by the Tower Installation Wizard. The setup script takes the following arguments:

- `-h`, `--help`
  Displays a brief usage summary.
- `-c FILE`
  Use the specified FILE as the Tower configuration file rather than `tower_setup_conf.yml` in the current directory.
- `-i FILE`
  Use the specified FILE as the inventory for the setup playbook rather than `inventory` in the current directory.
- `-p`
  Set ansible to prompt for a SSH password when connecting to remote machines
- `-s`
  Set ansible to prompt for a sudo password on remote machines when installing Tower.
- `-u`
  Set ansible to prompt for a su password on remote machines when installing Tower.

- `-e`
  Set additional ansible variables for the playbook to use either in key=value or YAML/JSON form. This should not be needed in normal operation.

Depending on the configuration you entered when running the Tower Installation Wizard, it may have prompted you to run the setup playbook with some combination of `-p`, `-s`, or `-u`.

Call `setup.sh` with the appropriate parameters, and Tower will be installed on the appropriate machines as configured.

Upgrading an Existing Tower Installation

You can upgrade your existing Tower installation to the latest version easily.

As with installation, the upgrade process requires that the Tower server be able to access the Internet. The upgrade process will take roughly the same amount of time as a Tower install, plus any time needed for data migration.

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

NOTE: *You can not convert an embedded-database Tower to a High Available installation as part of an upgrade. Users who want to deploy Tower in a High Availability configuration should back up their Tower database, install a new HA configuration on a different VM or physical host, and then restore the database. It is possible to add a first or additional secondaries later to a Tower already operating on an external database.*

Get the Tower Installer

Download the Ansible Tower install/upgrade tool at http://releases.ansible.com/ansible-tower/setup/. Extract it, 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., "2.1".

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

Run the Tower Installation Wizard

To configure your upgrade, you will run the same Tower Installation Wizard that you would for installation.

Simplified Tower Upgrade

If you're upgrading a Tower instance on a local machine with an internal database, you can bypass many of the questions by invoking the configure script as `./configure --local`

```
root@localhost:~$ ./configure --local
-----------------------------------------
Welcome to the Ansible Tower Install Wizard
-----------------------------------------

This wizard will guide you through the setup process.

LOCAL INSTALLATION
You are installing Ansible Tower on this machine, using an internal database.

REVIEW
You are UPGRADING an existing Tower installation on localhost.


Are these settings correct (y/n)?
```

Confirm that the settings are correct.

Upgrade Using an existing Settings File

If you have the `tower_setup_conf.yml` file from when you installed Tower, you can pass it to the `configure` script:

```
root@localhost:~$ ./configure -o tower_setup_conf.yml
----------------------------------------
Welcome to the Ansible Tower Install Wizard
----------------------------------------

This wizard will guide you through the setup process.

The configuration provided in /home/tower/ansible-tower-
setup-2.1/tower_setup_conf.yml appears complete.

FINISHED!
You have completed the setup wizard. You may execute the installation of
Ansible Tower by issuing the following command:

sudo ./setup.sh
```

Upgrade Interactively

Alternatively, you can walk through the upgrade process. Invoke the `configure` script:

```
root@localhost:~$ ./configure
----------------------------------------
----------------------------------------
Welcome to the Ansible Tower Install Wizard
----------------------------------------

This wizard will guide you through the setup process.

PRIMARY TOWER MACHINE
Tower can be installed (or upgraded) on this machine, or onto a remote machine
that is reachable by SSH.

Note: If using the High Availability features of Tower, you must use DNS
resolvable hostnames or IP addresses (do not use "localhost").

Enter the hostname or IP to configure Ansible Tower
(default: localhost):
```

Once you enter the host, the Tower Installation Wizard will contact it to determine if Tower is installed. If it is a functioning Tower installation, it will determine the current Tower configuration, including the location of any secondary nodes that need upgraded.

The Installation Wizard will then ask you for connection details for connecting to any remote machines. Enter any needed SSH user, SSH key location, and whether sudo or su is in use.

The Setup Playbook

The Tower setup playbook is invoked as `setup.sh` from the path where you unpacked the Tower installer tarball. It uses the `tower_setup_conf.yml` and `inventory` files written by the Tower Installation Wizard. The setup script takes the following arguments:

- `-h`, `--help`
  Displays a brief usage summary.
- `-c FILE`
  Use the specified FILE as the Tower configuration file rather than `tower_setup_conf.yml` in the current directory.
- `-i FILE`
  Use the specified FILE as the inventory for the setup playbook rather than `inventory` in the current directory.
- `-p`
  Set ansible to prompt for a SSH password when connecting to remote machines
- `-s`
  Set ansible to prompt for a sudo password on remote machines when upgrading Tower.
- `-u`
  Set ansible to prompt for a su password on remote machines when upgrading Tower.
- `-e`
  Set additional ansible variables for the playbook to use either in key=value or YAML/JSON form. This should not be needed in normal operation.

Depending on the configuration you entered when running the Tower Installation Wizard, it may have prompted you to run the setup playbook with some combination of `-p`, `-s`, or `-u`.

Call `setup.sh` with the appropriate parameters, and Tower will be upgraded on the appropriate machines as configured.

> ***NOTE:*** *As part of the upgrade process, database schema migration may be done. Depending on the size of your Tower installation, this may take some time.*

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

Administration of Tower

Init script

Tower ships with a standard `ansible-tower` init script that can be used to start, stop, and query the full tower infrastructure (including the database and message queue components.)
You can invoke it via the `service` command:

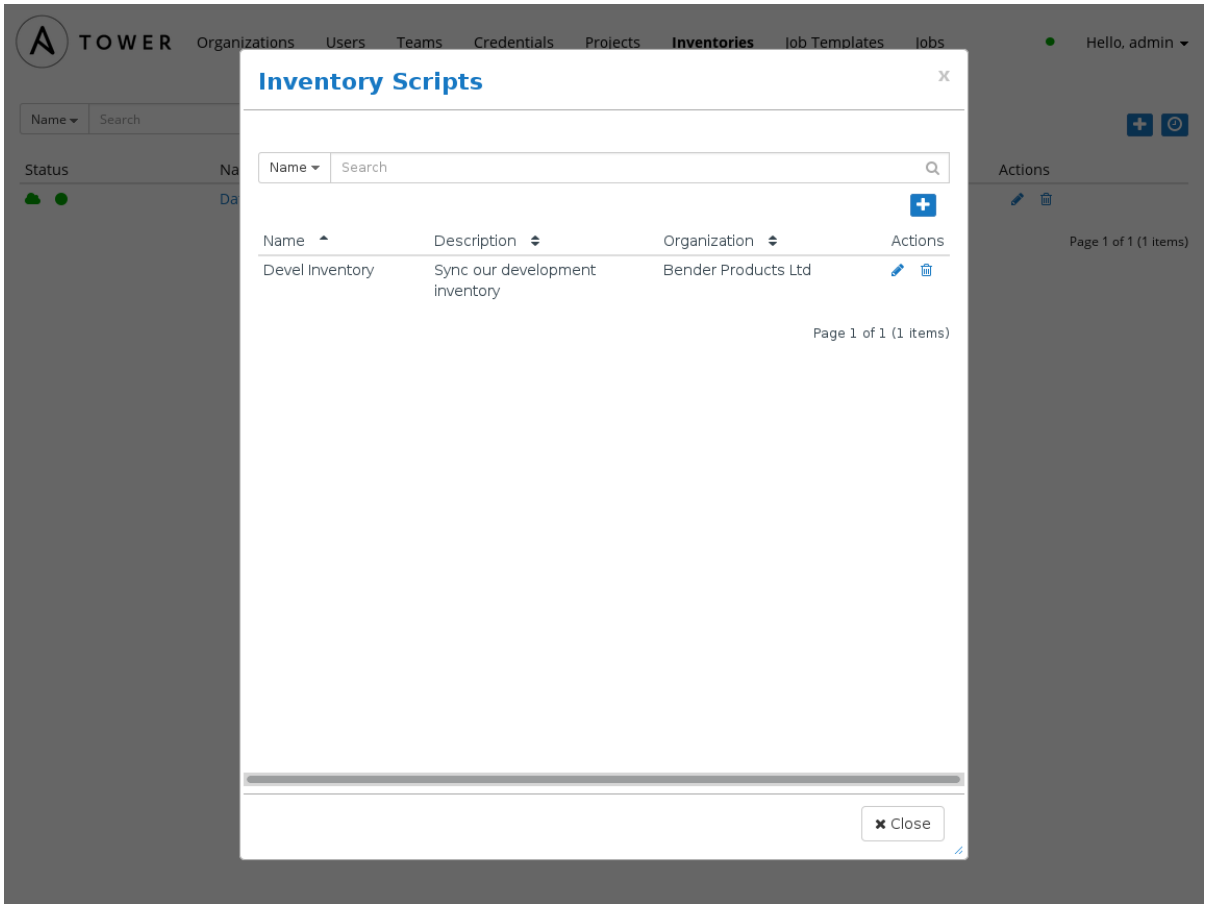
```
root@localhost:~$ service ansible-tower restart
```

or via distribution-specific service management commands.

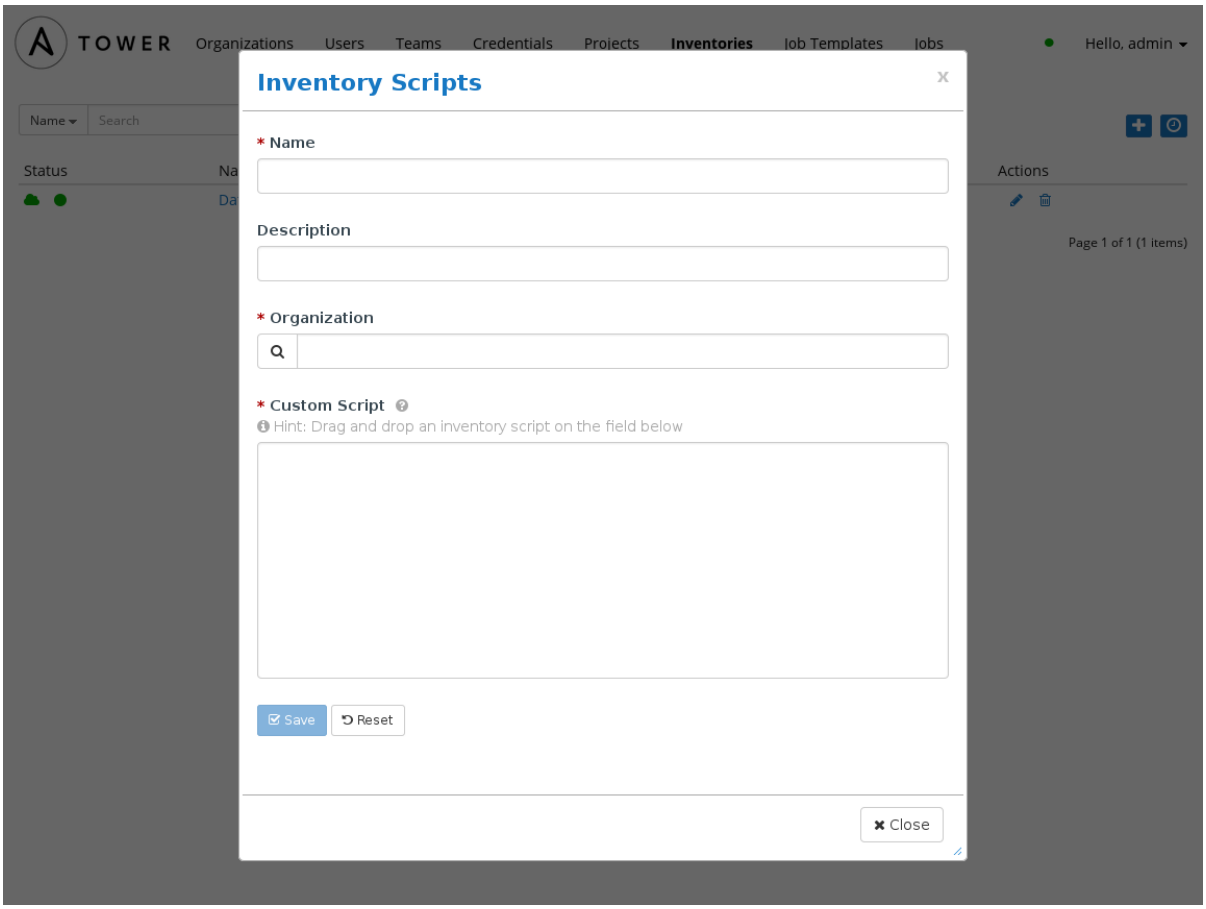
Custom Inventory Scripts

Tower includes built-in support for syncing dynamic inventory from cloud sources such as Amazon AWS, Google Compute Engine, Rackspace, and more. However, if you have your own inventory source, Tower also offers the ability to use a custom script to pull from it.

To manage the custom inventory scripts available in Tower, choose **Inventory Scripts** from the user menu at the top right of the Tower interface.



To add a new custom inventory script, click the button.



Enter the name for the script, and an optional description. Then select the **Organization** that this script belongs to.

You can then either drag and drop a script on your local system into the **Custom Script** text box, or cut and paste the contents of the inventory script there.

Writing Inventory Scripts

Inventory scripts can be written in any dynamic language that you have installed on the Tower machine (such as shell or python). They must start with a normal script shebang line such as `#!/bin/bash` or `#!/usr/bin/python`. They run as the `awx` user. The inventory script will be invoked with '--list' to list the inventory, which should be returned in a JSON hash/dictionary.

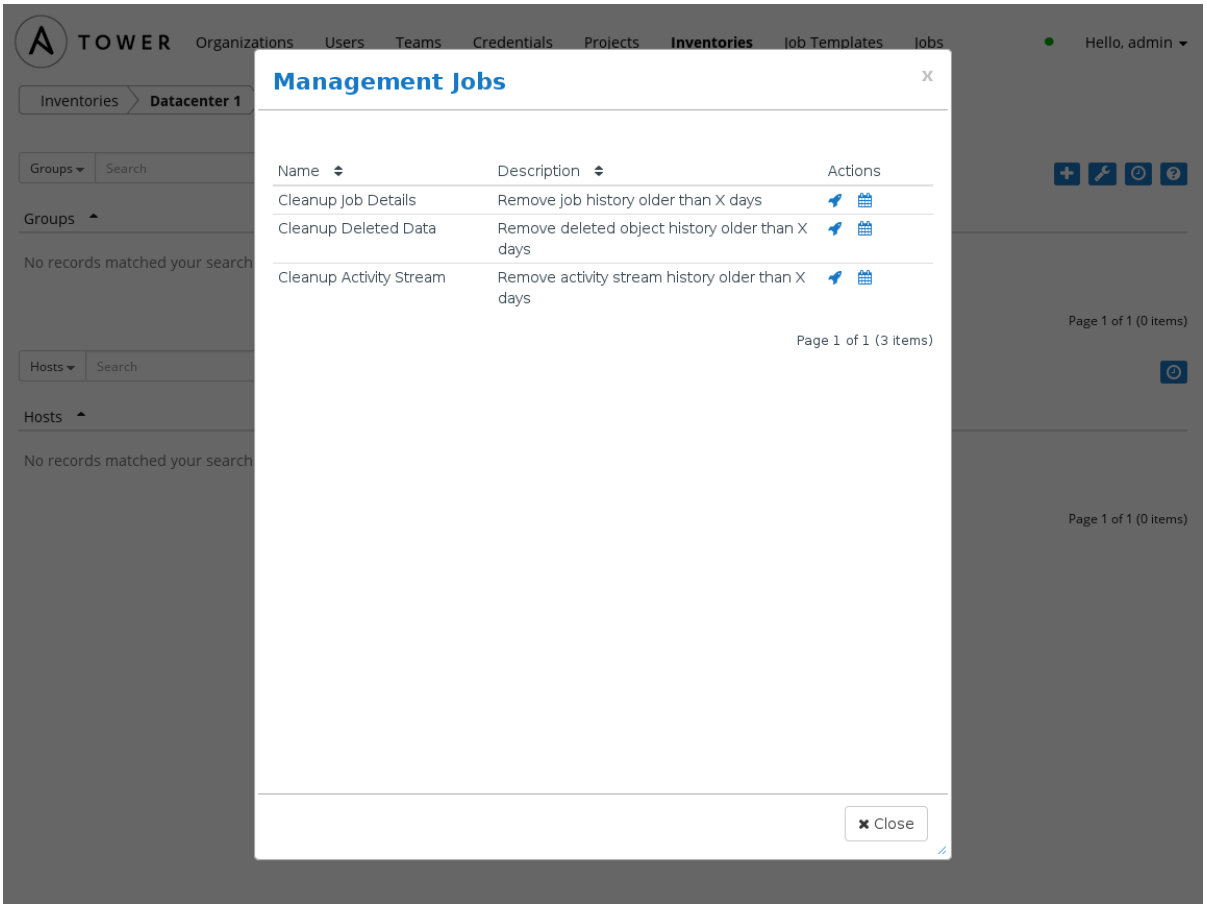
Generally, they connect to the network to retrieve the inventory from other sources.

For more information on dynamic inventory scripts and how to write them, see the Intro to Dynamic Inventory (http://docs.ansible.com/intro_dynamic_inventory.html) and Developing Dynamic Inventory Sources (http://docs.ansible.com/developing_inventory.html) section of the Ansible documentation, or see example dynamic inventory scripts on GitHub (https://github.com/ansible/ansible/tree/devel/plugins/inventory).

Tower Management Jobs

Tower includes the built-in ability to clean old data from Tower's database. You can use this if you have specific data retention policies, or need to decrease the storage used by your Tower database.

Tower Management Jobs are available via the **Management Jobs** entry of the user menu at the top right of the Tower interface. There are three categories of Management Job available:



- Cleanup Job Details

This permanently deletes the job details and job output for jobs older than a specified number of days.

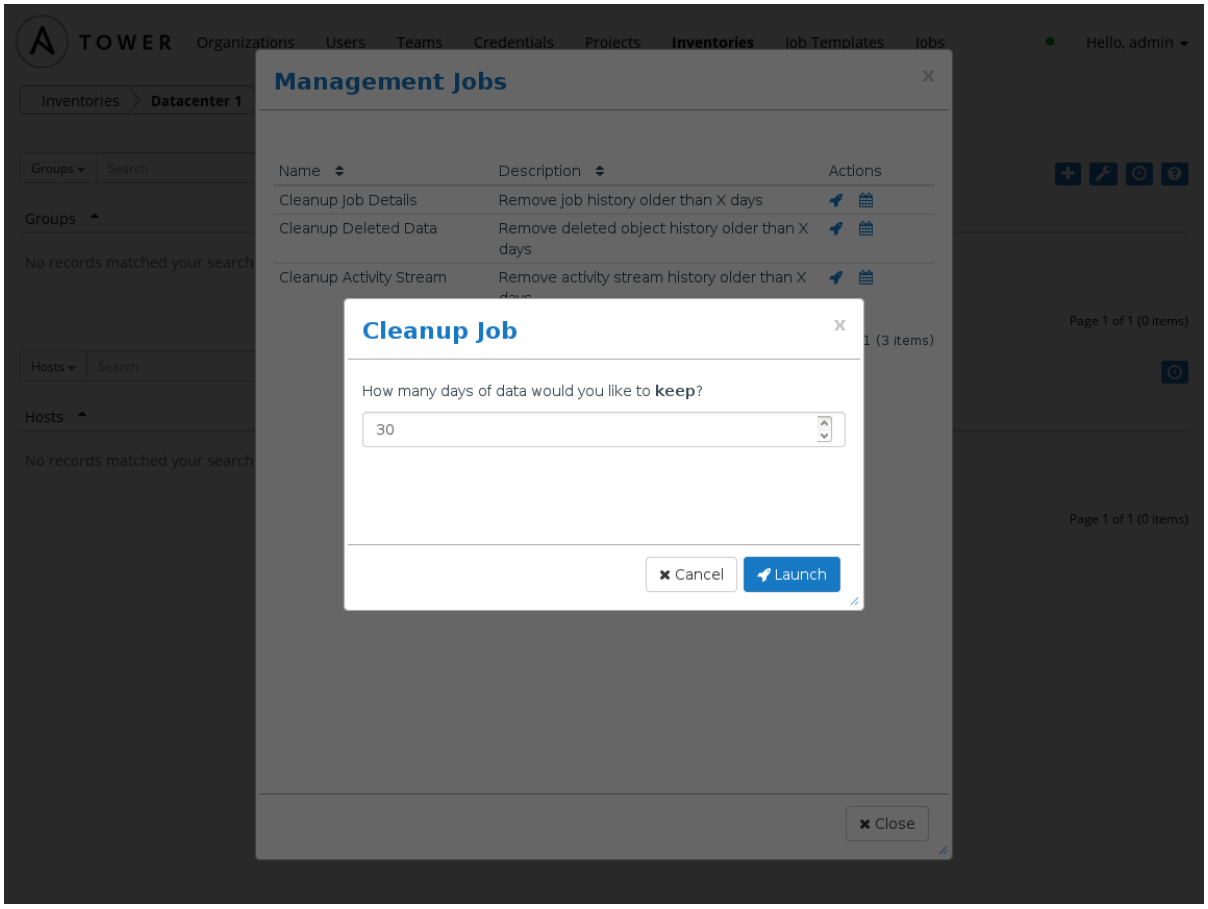
- Cleanup Deleted Data

This permanently deletes any deleted Tower objects that are older than a specified number of days.

- Cleanup Activity Stream

This permanently deletes any activity stream data older than a specific number of days.

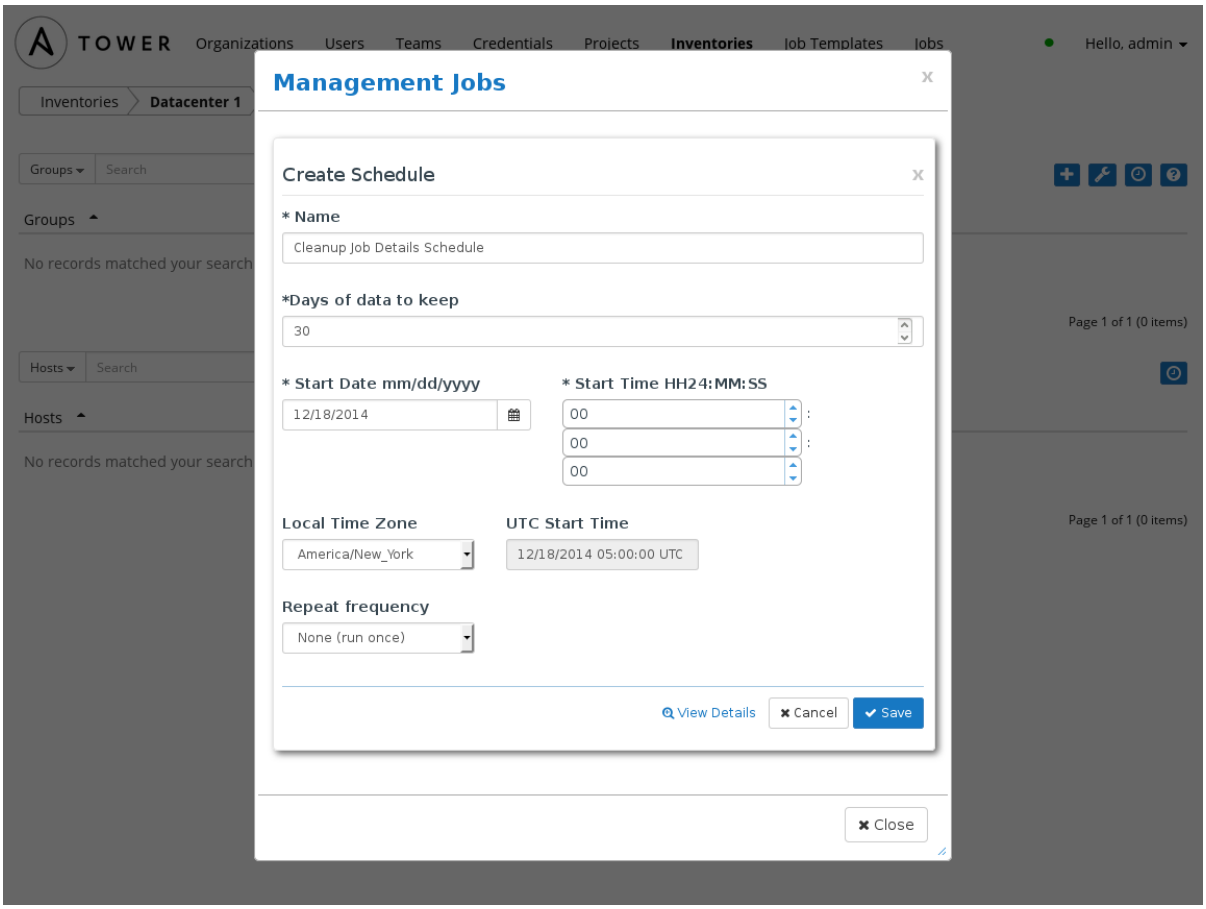
If you click on the button for any of these jobs, you will be prompted for the number of days of data to keep.



Enter how many days of data you would like to keep, and select **Launch**. Data older than that number of days will be deleted.

NOTE: *This action is not reversible.*

For any management job, you can also click on the button to create a schedule for this job. In this schedule you can set the number of days of data to keep, and when you would like this job to periodically run.



Monitoring

Tower includes its own munin instance for monitoring Tower. This interface can be accessed via the `Monitor Tower` dropdown of the Tower user's menu at the top right of the Tower interface, and also directly at `https://<Tower server name>/munin`.

The default munin user is `admin`, and the default password is set in `group_vars/all` at installation time.

This monitors general aspects of the Tower system, such as the apache webserver, networking, disk I/O and free storage, system processes, and CPU usage.

It also monitors aspects specific to Tower, under the **Tower** heading.

It monitors the health of the following key Tower services:

- Celery Service
- Redis Service
- SocketIO Service
- Task Manager

It also monitors the running jobs, and the number of processes for receiving Tower callbacks.

Using LDAP with 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, however organization membership (and who is an organization admin) and team memberships 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, first name, last name, 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/tower/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 http://support.ansible.com/ if you need assistance.

High Availability

Tower can be installed in a High Availability (HA) configuration. In this configuration, Tower is run with a single active node, called the Primary instance, and any number of inactive nodes, called Secondary instances. Secondary instances can be made Primary at any time, with certain caveats. When running in a HA configuration, Tower must be configured to use an external PostgreSQL database.

Tower's HA mode is designed for having a standby Tower infrastructure that can be made active in case of infrastructure failure, avoiding single points of failure. It is not meant to run in a active/active or multi-master mode, and is not a mechanism for horizontally scaling the Tower service. Further, failover to a secondary must be user triggered and is not automatic.

For instructions on how to install into a HA configuration, see the Installation and Setup Reference.

Setup Considerations

When creating a HA deployment of Tower, there are certain factors that should be considered.

- Tower servers should be isolated

  If the primary and secondary Tower services share a physical host, a network, or potentially a datacenter, your infrastructure has a single point of failure. You should locate the the Tower servers such that they are distributed in a manner consistent with other services that you make available across your infrastructure. If your infrastructure is already using features such as Availability Zones in your cloud provider, having Tower distributed across Zones as well makes sense.

- The database should be replicated.

  If Tower is run in a HA mode, but the database is not run in a HA or replicated mode, you still have a single point of failure for your Tower infrastructure. The Tower installer will not set up database replication, but merely prompt for database connection details to an existing database (which should be replicated). You should choose a database replication strategy that is appropriate for your deployment. For the general case of PostgreSQL, see the PostgreSQL documentation (http://www.postgresql.org/docs/current/static/high-availability.html). For deployments using Amazon's RDS, see Amazon's documentation (http://aws.amazon.com/rds/postgresql/).

- Tower instances should maintain reasonable connections to the database.

  Tower both queries and write to the database frequently; good locality between the Tower server and the database replicas is critical to ensure performance.

- Use Source Control

  If you're using playbooks stored locally on the Tower server (rather than set to check out from source control), you will need to ensure they are synchronized between the primary and secondary Tower instances. Using playbooks in source control alleviates this problem. When using SCM Projects, ensure that the 'Update on Launch' flag is set on the job template so that a checkout is made every time the playbook is launched - this ensures that a newly promoted secondary has an up-to-date copy of the project content.

- Present a consistent Tower hostname to users

  Between Tower user's habits, Tower provisioning callbacks, and Tower API integrations, it is best to keep the Tower hostname that users and clients use constant. In a HA deployment, this would be done via the use of a reverse proxy or a DNS CNAME. The CNAME is strongly preferred due to the websocket connection Tower uses for realtime output.

So, an example HA configuration for an infrastructure that consists of three datacenters would place a Tower server and a replicated database in each datacenter. There is a DNS CNAME that is used by clients accessing Tower, and it is pointed to the address of the current primary Tower instance.

Differences between Primary and Secondary Instances

The Tower service runs on both primary and secondary instances. However, only the primary instance will accept requests or run jobs.

If you attempt to connect to the web interface or API of a secondary Tower server, you will be redirected to the primary Tower instance.

Post-Installation Changes to Primary Instances

If you change the configuration of a primary instance after installation, these changes will need to be applied to the secondary instances as well.

Examples of these changes would be:

- Updates to `/etc/tower/settings.py`

  If you have configured LDAP or customized logging in `/etc/tower/settings.py`, you will need to reflect these changes in `/etc/tower/settings.py` on your secondary instances as well.

- Updating the Tower license

  Any secondary instance of Tower requires a valid license to run properly when promoted to a primary instance. This can be copied from the primary node at any time, or be installed via the normal license installation mechanism after the instance is promoted to primary status.

Examining the HA configuration of Tower

To see the HA configuration of Tower, you can query the **ping** endpoint of the Tower REST API. To do this via Tower's built in API browser, go to `https://<Tower server name>/api/v1/ping`. You can go to this specific URL on either the primary or secondary nodes.

An example return from this API call would be (in JSON format):

```
HTTP 200 OK
Content-Type: application/json
Vary: Accept
Allow: GET, HEAD, OPTIONS
X-API-Time: 0.008s
{
    "instances": {
        "primary": "192.168.122.158",
        "secondaries": [
            "192.168.122.109",
            "192.168.122.26"
        ]
    },
    "ha": true,
    "role": "primary",
    "version": "2.1.0"
}
```

It contains the following fields.

- Instances
  - Primary: The primary Tower instance (hostname or IP address)
  - Secondaries: The secondary Tower instances (hostname or IP address)
- HA: Whether Tower is running in HA mode
- Role: Whether this specific instance is a primary or secondary
- Version: The Tower version in uses

Promoting a Secondary Instance/Failover

To promote a secondary instance to be the new primary instance (also known as initiating failover), use the `tower_manage` command.

To make a running secondary node the primary node, log on to the desired new primary node, and run the `register_instance` command of `tower_manage` as so:

```
root@localhost:~$ tower-manage register_instance --primary
Instance 480edf29-778e-47d6-be8d-7363b82d7da9 registered (changed: True).
```

The current primary instance will be changed to be a secondary.

> ***NOTE:*** *Secondary nodes need a valid Tower license at /etc/tower/license to function as a proper primary instance. This can be copied from the primary node at any time, or be installed via the normal license installation mechanism after the instance is promoted to primary status.*

On failover, any jobs in the database that are currently queued or running will be marked as failed.

Tower does not attempt any health checks between primary or secondary nodes to do automatic failover in case of the loss of the primary node. We recommend the use of an external monitoring or heartbeat tool combined with `tower_manage` if this is desired. Use of the "/ping" API endpoint could be used for this purpose.

Deregistering Secondary instances

You cannot deregister a current primary Tower instance without first selecting a new primary.

If you need to decommission a secondary instance of Tower, log onto the secondary node and run the `remove_instance` command of `tower_manage` as so:

```
root@localhost:~$ tower-manage remove_instance --hostname tower2.example.com
Instance removed (changed: True).
```

Replace `tower2.example.com` with the registered name (IP address or hostname) of the Tower instance you are removing from the list of secondaries.

You can then shutdown the Tower service on the decommissioned secondary node.

tower-manage

tower-manage (formerly awx-manage) is a utility that can be used to access detailed internal information of Tower. tower-manage commands should be run as the `awx` or `root` user.

Inventory Import

tower-manage is a mechanism by which a Tower administrator can import inventory directly into Tower, for those who cannot use Custom Inventory Scripts.

```
tower-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.

Cleanup of old data

tower-manage has a variety of commands that can be used to clean old data from Tower. While it is recommended that Tower administrators use the Tower Management Jobs interface, these commands are available.

- `tower-manage cleanup_jobs [--help]`

This permanently deletes the job details and job output for jobs older than a specified number of days.

- Cleanup Activity Stream
- `tower-manage cleanup-deleted [--help]`

This permanently deletes any deleted Tower objects that are older than a specified number of days.

- `tower-manage cleanup_activitystream [--help]`

This permanently deletes any activity stream data older than a specific number of days.

HA management

See the High Availability section for details on the `tower-manage register_instance` and `tower-manage remove_instance` commands.

NOTE: *Do not run other `tower-manage` commands unless instructed by Ansible Support.*

Troubleshooting

Tower server errors are logged to syslog. Apache web server errors are logged to the httpd error log. Additional Tower logging can be configured in `/etc/tower/settings.py`.

Client-side issues may be explored using the JavaScript console built into most browsers and any errors should be reported to http://support.ansible.com/.

API

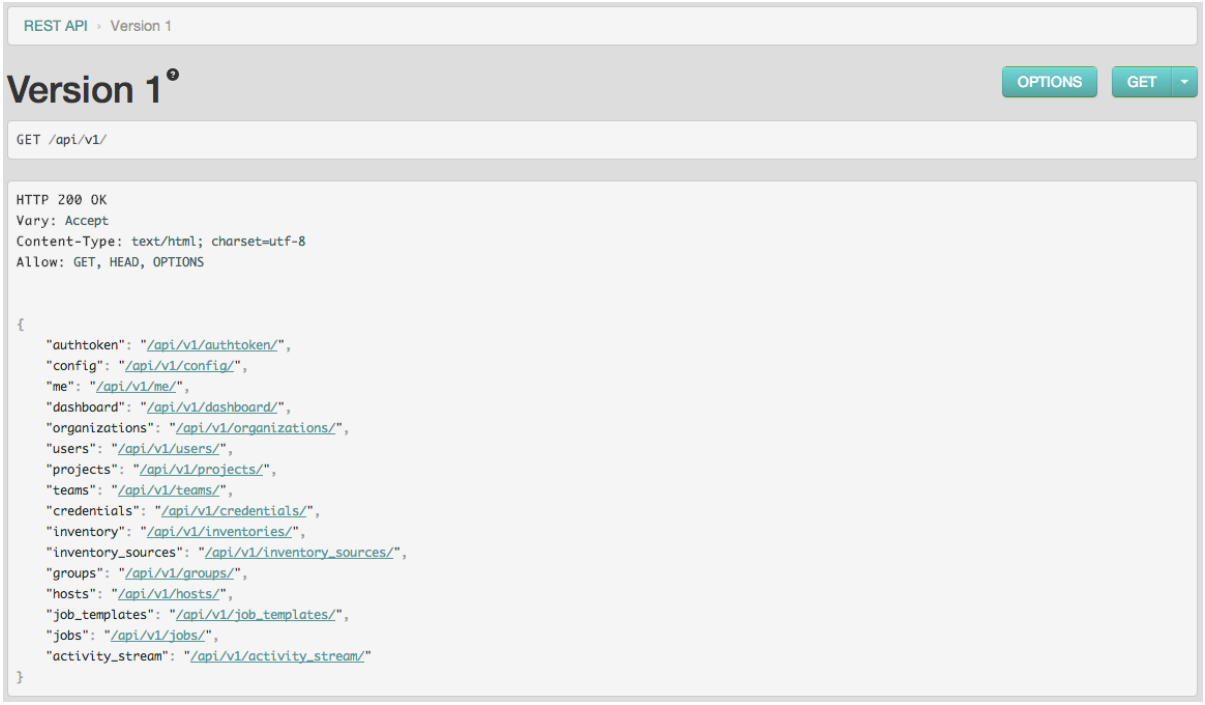
Tools

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.

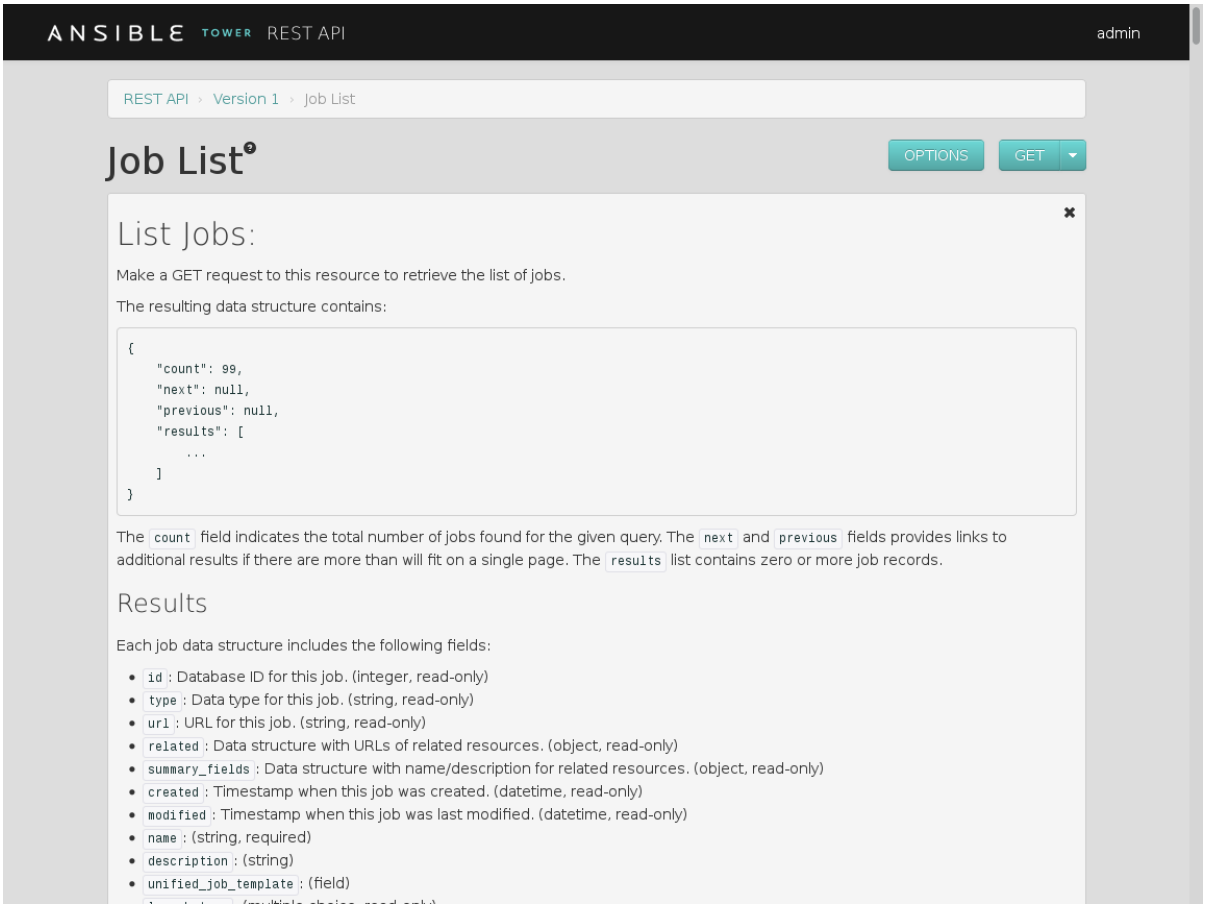
Browseable API

Tower features a browseable API feature.

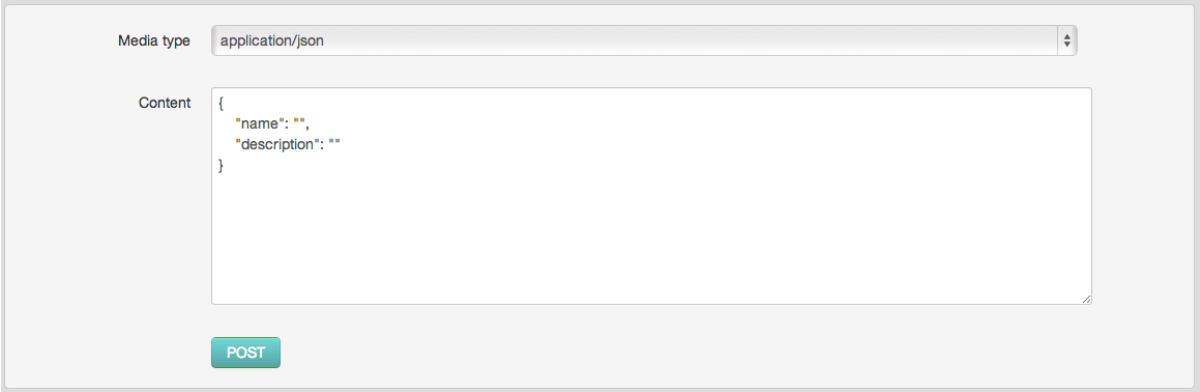
You can visit the API in a browser at `http://<Tower server name>/api` and then click on various links in the API to explore related resources.



Clicking on the '?' next to the page name for an API endpoint will give you documentation on the access methods for that particular API endpoint and what data is returned when using those methods.



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



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.

tower-cli

tower-cli is a command line tool for Ansible Tower. It allows Tower commands to be easily run from the Unix command-line. It can also be used as a client library for other python apps, or as a reference for others developing API interactions with Tower's REST API.

Capabilities

tower-cli sends commands to the Tower API. It is capable of retrieving, creating, modifying, and deleting most objects within Tower.

A few potential uses include:

- Launching playbook runs (for instance, from Jenkins, TeamCity, Bamboo, etc)
- Checking on job statuses
- Rapidly creating objects like organizations, users, teams, and more

Installation

Tower CLI is available as a package on PyPI (https://pypi.python.org/pypi/ansible-tower-cli)

The preferred way to install is through pip:

```
$ pip install ansible-tower-cli
```

The main branch of this project may also be consumed directly from source.

For more information on tower-cli, see the project page at https://github.com/ansible/tower-cli.

Glossary

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.

Group: A set of hosts in Ansible that can be addressed as a set, of which many may exist within a single Inventory.

Host: A system managed by Tower, which may include a physical, virtual, cloud-based server, or other device. Typically an operating system instance. Hosts are contained in Groups, which are in turn contained in Inventory. Sometimes referred to as a "node".

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

Inventory Source: Information about a cloud or other script that should be merged into the current inventory group, resulting in the automatic population of Groups, Hosts, and variables about those groups and hosts.

Job: One of many background tasks launched by Tower, this is usually the instantiation of a Job Template; the launch of an Ansible playbook. Other types of jobs include inventory imports, project synchronizations from source control, or administrative cleanup actions.

Job Detail: The history of running a particular job, including it's output and success/failure status.

Job Template: The combination of an Ansible playbook and the set of parameters required to launch it.

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.

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.

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.

Playbook: An Ansible playbook. See docs.ansible.com for more information.

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

Schedule: The calendar of dates and times for which a job should run automatically.

Superuser: An admin of the Tower server who has permission to edit any object in the system, whether associated to any organization. Superusers can create organizations and other superusers.

Survey: Questions asked by a job template at job launch time, configurable on the job template.

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.

User: An Tower operator with associated permissions and credentials.
