# Ansible Tower Administration Guide

*Release Ansible Tower 2.4.0*

Red Hat, Inc.

Jun 06, 2017

# CONTENTS

Thank you for your interest in Ansible Tower, the open source IT orchestration engine. Whether sharing operations tasks with your team or integrating with Ansible through the Tower REST API, Tower provides powerful tools to make your automation life easier.

The *Ansible Tower Administration Guide* documents the administration of Ansible Tower through custom scripts, management jobs, and more. Written for DevOps engineers and administrators, the *Ansible Tower Administration Guide* assumes a basic understanding of the systems requiring management with Tower's easy-to-use graphical interface. This document has been updated to include information for the latest release of Ansible Tower 2.4.0.

Ansible Tower Version 2.4.0; November 18, 2015; https://access.redhat.com/

CHAPTER

ONE

# TOWER LICENSING, UPDATES, AND SUPPORT

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

Ansible is an open source software project and is licensed under the GNU General Public License version 3, as detailed in the Ansible source code: https://github.com/ansible/ansible/blob/devel/COPYING

## 1.1 Support

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

If you are using Ansible core and are having issues, you should reach out to the "ansible-devel" mailing list or file an issue on the Github project page at https://github.com/ansible/ansible/issues/.

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

## 1.2 Trial Licenses

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

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

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

## 1.3 License Types

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

- **Basic**
  - Manage smaller environments (up to 250 nodes)
  - 30 days of initial web support through https://access.redhat.com/, for installation and setup
  - Maintenance and upgrades included
- **Enterprise**

  - Manage any size environment
  - Enterprise 8x5 support and SLA (4 hour critical incident response)
  - Phone and web support
  - Support for Ansible core included
  - Maintenance and upgrades included

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

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

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

## 1.4 License Features

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

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

- Custom rebranding for login (*added in Ansible Tower 2.4.0*)
- SAML and RADIUS Authentication Support (*added in Ansible Tower 2.4.0*)
- Multi-Organization Support
- Activity Streams
- Surveys
- LDAP Support
- High Availability
- System Tracking (*added in Ansible Tower 2.2.0*)

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

## 1.5 Tower Component Licenses

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

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

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

CHAPTER

TWO

# STARTING, STOPPING, AND RESTARTING TOWER

Ansible Tower now ships with an *admin utility script*, `ansible-tower-service`, that can `start`, `stop`, and `restart` the full tower infrastructure (including the database and message queue components). The services script resides in `/usr/bin/ansible-tower-service` and can be invoked as follows:

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

You can also invoke it via distribution-specific service management commands. Distribution packages often provide a similar script, sometimes as an init script, to manage services. Refer to your distribution-specific service management system for more information.

**Note:** Beginning with version 2.2.0, Ansible Tower has moved away from using an init script in favor of using an admin utility script. Previous versions of Ansible Tower shipped with a standard `ansible-tower` init script that could be used to `start`, `stop`, and `query` the full Tower infrastructure. It was evoked via the service command: `/etc/init.d/ansible-tower script`. For those using a 2.2.0 or later version of Ansible Tower, the new admin utility script, `ansible-tower-service`, should be used instead.

CHAPTER

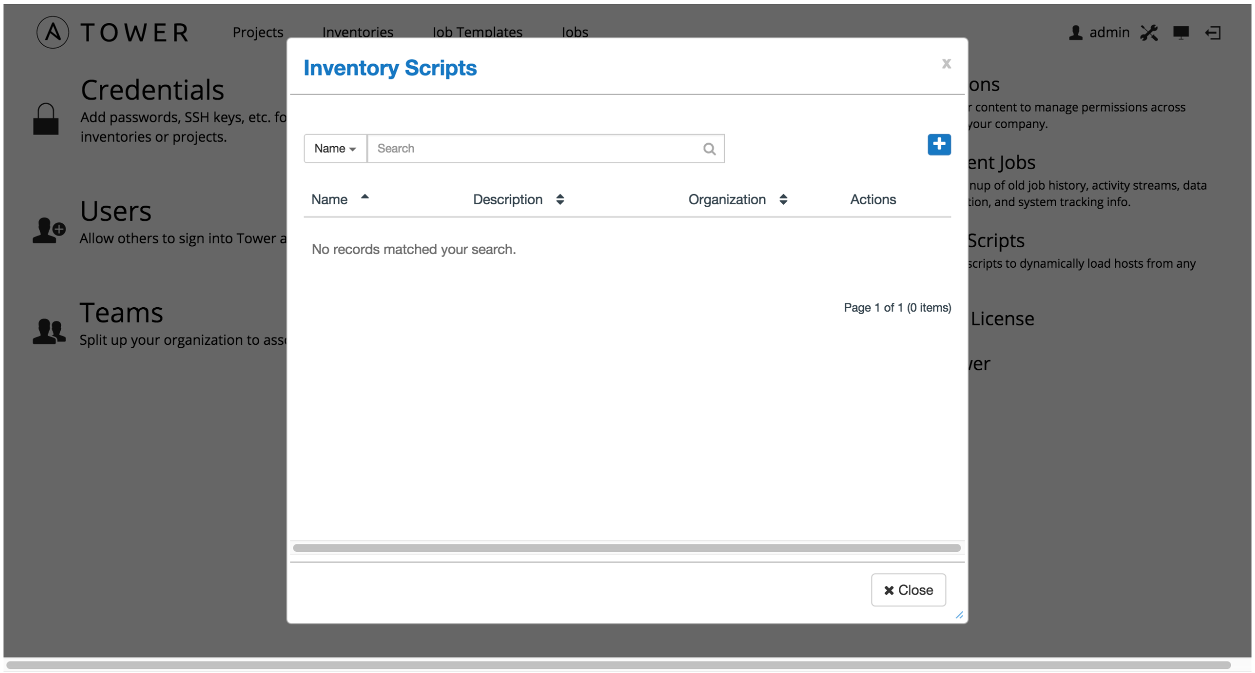
THREE

# CUSTOM INVENTORY SCRIPTS

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

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

To manage the custom inventory scripts available in Tower, choose **Inventory Scripts** from the Setup ( 🛠 ) menu.


TOWER
Projects
Inventories
Job Templates
Jobs
admin
Credentials
Add passwords, SSH keys, etc. fo
inventories or projects.
Users
Allow others to sign into Tower a
Teams
Split up your organization to ass
ons
r content to manage permissions across
your company.
ent Jobs
nup of old job history, activity streams, data
tion, and system tracking info.
Scripts
scripts to dynamically load hosts from any
License
ver

Inventory Scripts

Name ▾ | Search

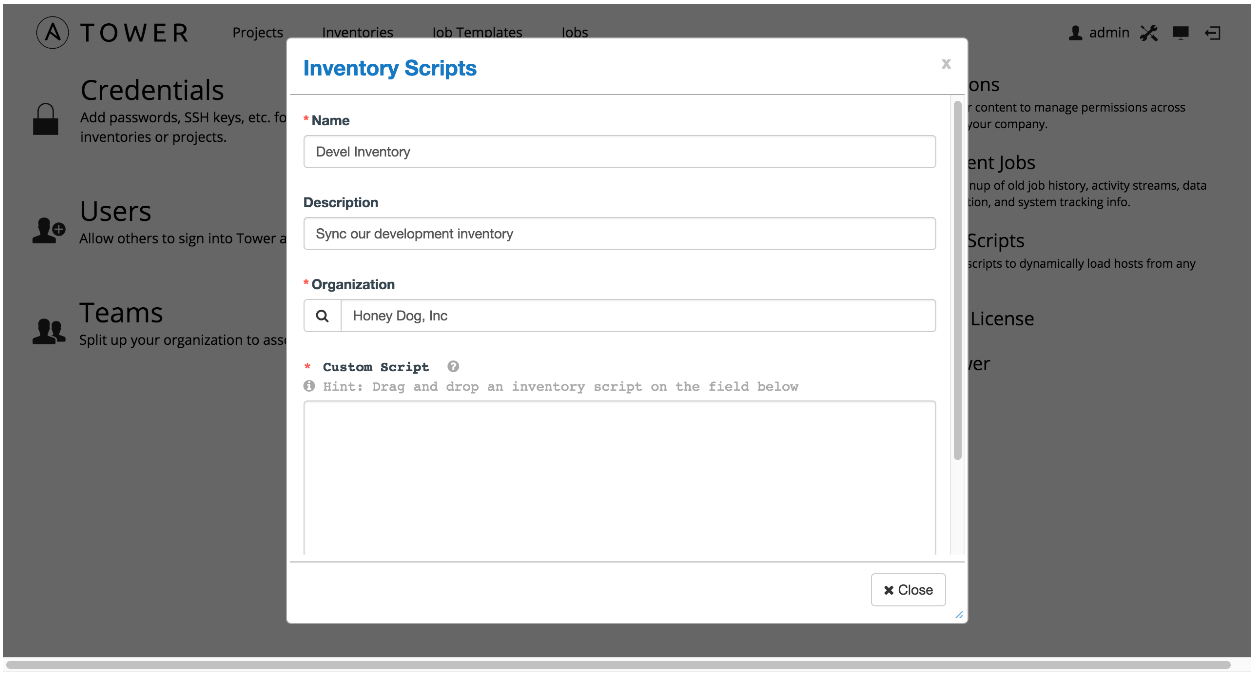
| Name ▲ | Description ⬍ | Organization ⬍ | Actions |
|---|---|---|---|

No records matched your search.

Page 1 of 1 (0 items)

✖ Close

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


TOWER
Projects
Inventories
Job Templates
Jobs
admin
Credentials
Add passwords, SSH keys, etc. fo
inventories or projects.
Users
Allow others to sign into Tower a
Teams
Split up your organization to ass
ons
r content to manage permissions across
your company.
ent Jobs
nup of old job history, activity streams, data
tion, and system tracking info.
Scripts
scripts to dynamically load hosts from any
License
ver
✖ Close

Inventory Scripts

* Name
Devel Inventory

Description
Sync our development inventory

* Organization
Honey Dog, Inc

* Custom Script
Hint: Drag and drop an inventory script on the field below

Enter the name for the script, plus 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.

## 3.1 Writing Inventory Scripts

You can write inventory scripts 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 invokes with `'--list'` to list the inventory, which returns in a JSON hash/dictionary.

Generally, they connect to the network to retrieve the inventory from other sources. When enabling multi-tenancy security (refer to Security for details), the inventory script will not be able to access most of the Tower machine. If this access to the local Tower machine is necessary, configure it in `/etc/tower/settings.py`.

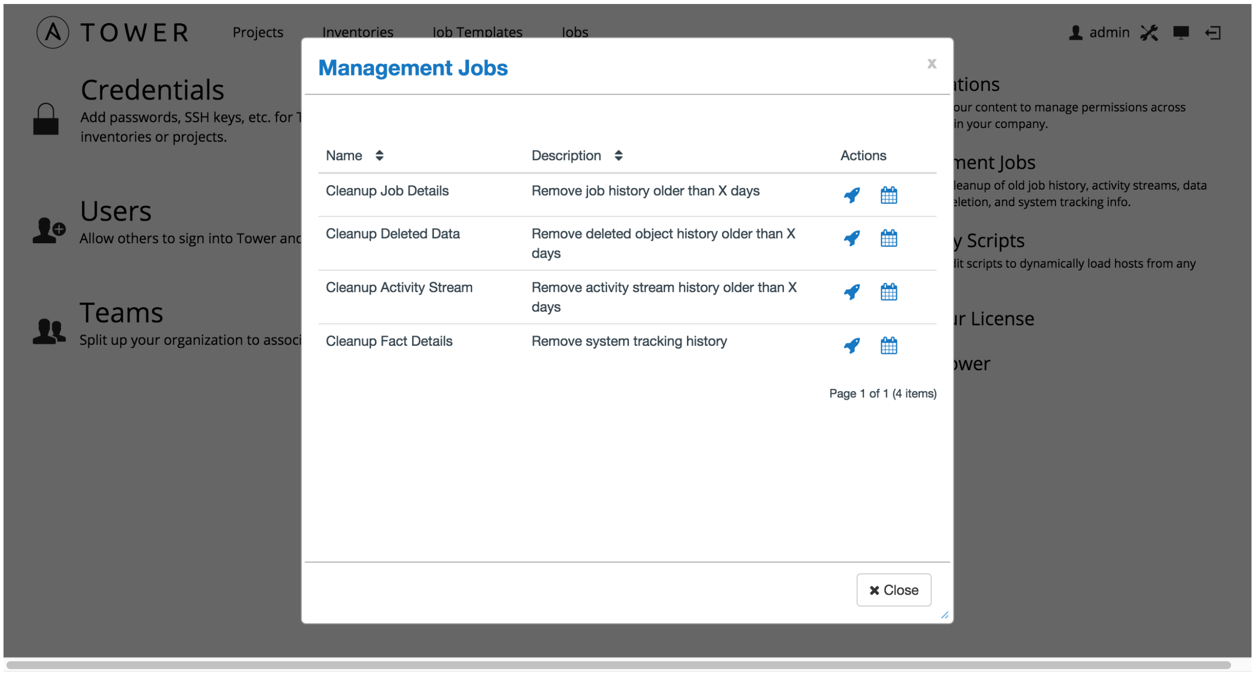
For more information on dynamic inventory scripts and how to write them, refer to the Intro to Dynamic Inventory and Developing Dynamic Inventory Sources sections of the Ansible documentation, or review the example dynamic inventory scripts on GitHub.

CHAPTER

FOUR

# MANAGEMENT JOBS

**Management Jobs** assist in the cleaning of old data, old system tracking information, old job histories, and old activity streams from Tower. You can use this if you have specific retention policies or need to decrease the storage used by your Tower database. From the Setup ( 🛠 ) menu, click on **Management Jobs**.


TOWER
Projects
Inventories
Job Templates
Jobs
admin
Credentials
Add passwords, SSH keys, etc. for T
inventories or projects.
Users
Allow others to sign into Tower and
Teams
Split up your organization to associ
tions
our content to manage permissions across
in your company.
ment Jobs
leanup of old job history, activity streams, data
eletion, and system tracking info.
y Scripts
lit scripts to dynamically load hosts from any
r License
wer
×
✖ Close

**Management Jobs**

| Name | Description | Actions |
|---|---|---|
| Cleanup Job Details | Remove job history older than X days | 🚀 📅 |
| Cleanup Deleted Data | Remove deleted object history older than X days | 🚀 📅 |
| Cleanup Activity Stream | Remove activity stream history older than X days | 🚀 📅 |
| Cleanup Fact Details | Remove system tracking history | 🚀 📅 |

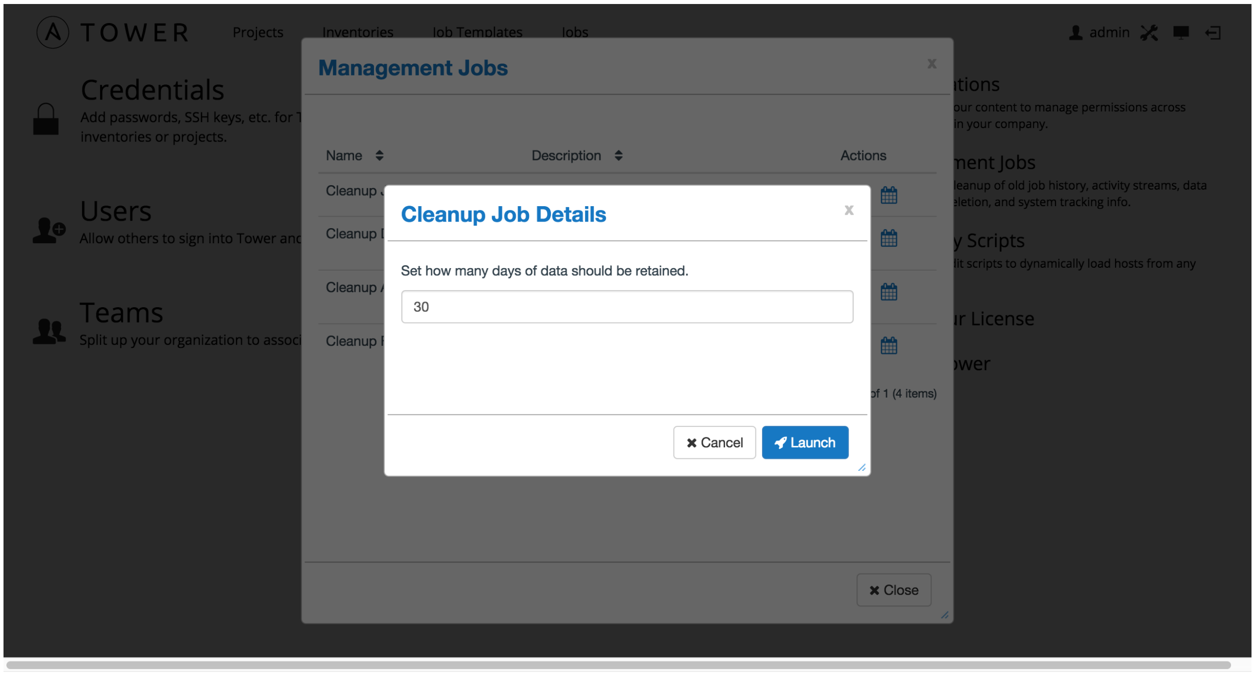
Page 1 of 1 (4 items)

Several job types are available for you to schedule and launch:

- **Cleanup Job Details**: Remove job history older than a specified number of days
- **Cleanup Deleted Data**: Remove deleted object history older than a specified number days
- **Cleanup Activity Stream**: Remove activity stream history older than a specified number of days
- **Cleanup Fact Details**: Remove system tracking history

## 4.1 Removing Old Job History

To remove job history older than a specified number of days, click on the 🚀 button beside **Cleanup Job Details**.



**Cleanup Job Details**

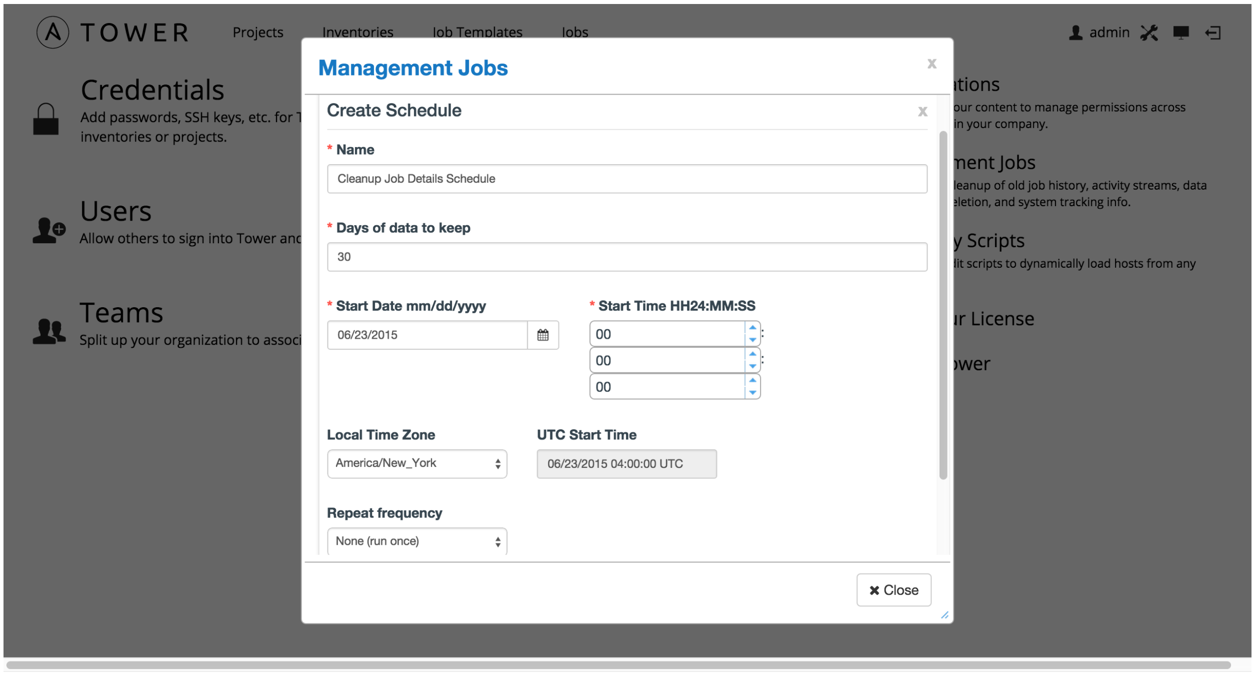
Set how many days of data should be retained.

30

✖ Cancel | Launch

Enter the number of days of data you would like to save and click **Launch**.

To set a schedule for cleaning up old job history information, click on the 📅 button.



**Create Schedule**

* Name: Cleanup Job Details Schedule

* Days of data to keep: 30

* Start Date mm/dd/yyyy: 06/23/2015

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

Local Time Zone: America/New_York

UTC Start Time: 06/23/2015 04:00:00 UTC

Repeat frequency: None (run once)

Enter the appropriate details for:

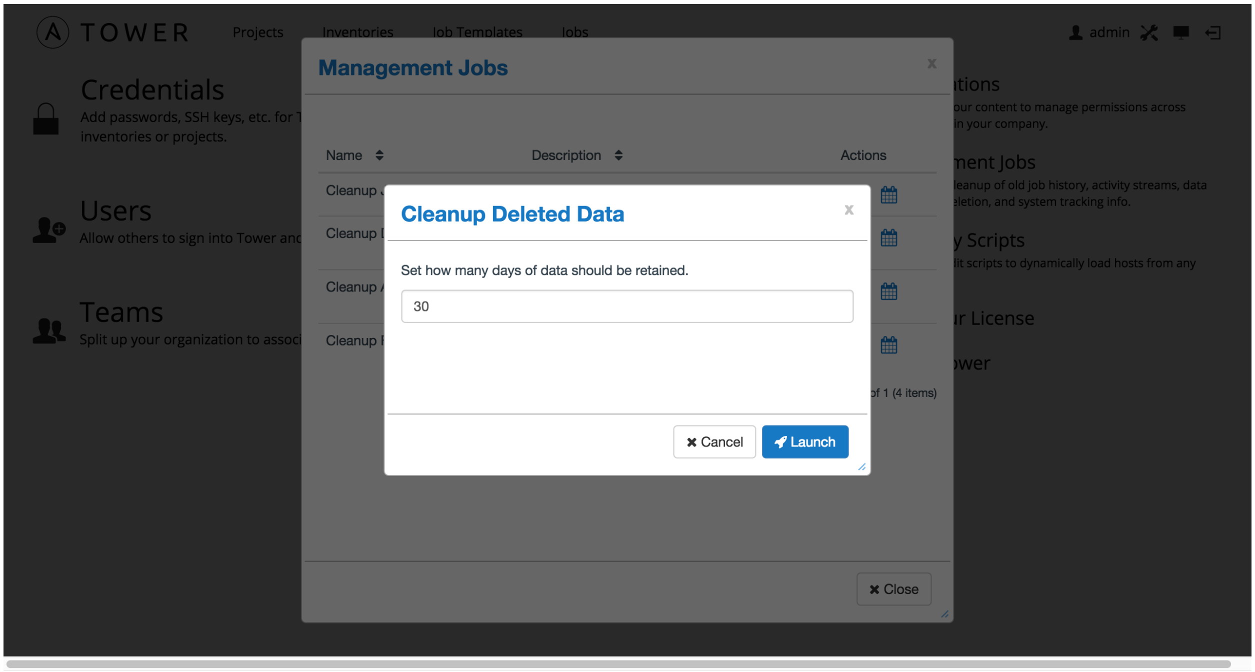
- **Name**: populated by default to match the management job type and can be left alone
- **Days of data to keep**: Enter the number of days of data you would like to retain.
- **Start Date**: Enter the date for which you want data cleanup to start (month/day/year format)
- **Start Time**: Enter the time to start data cleanup in hours, minutes, and seconds.

- **Local Time Zone**: Select your preferred time zone.
- **UTC Start Time**: Review the UTC Start time which is based on the Start Time and Local Time Zone you select.
- **Repeat Frequency**: Select how often you want the cleanup job to run. You can enter cleanups to occur once or every *X* minutes, hours, days, weeks, months, or years.

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

## 4.2 Removing Data Scheduled for Deletion

To remove object history which has been marked for deletion (much like emptying your trashcan in a desktop environment), click on the 🚀 button beside **Cleanup Deleted Data**.



Management Jobs

| Name | Description | Actions |
|---|---|---|
| Cleanup J | | 📅 |
| Cleanup D | | 📅 |
| Cleanup A | | 📅 |
| Cleanup F | | 📅 |

of 1 (4 items)

Cleanup Deleted Data

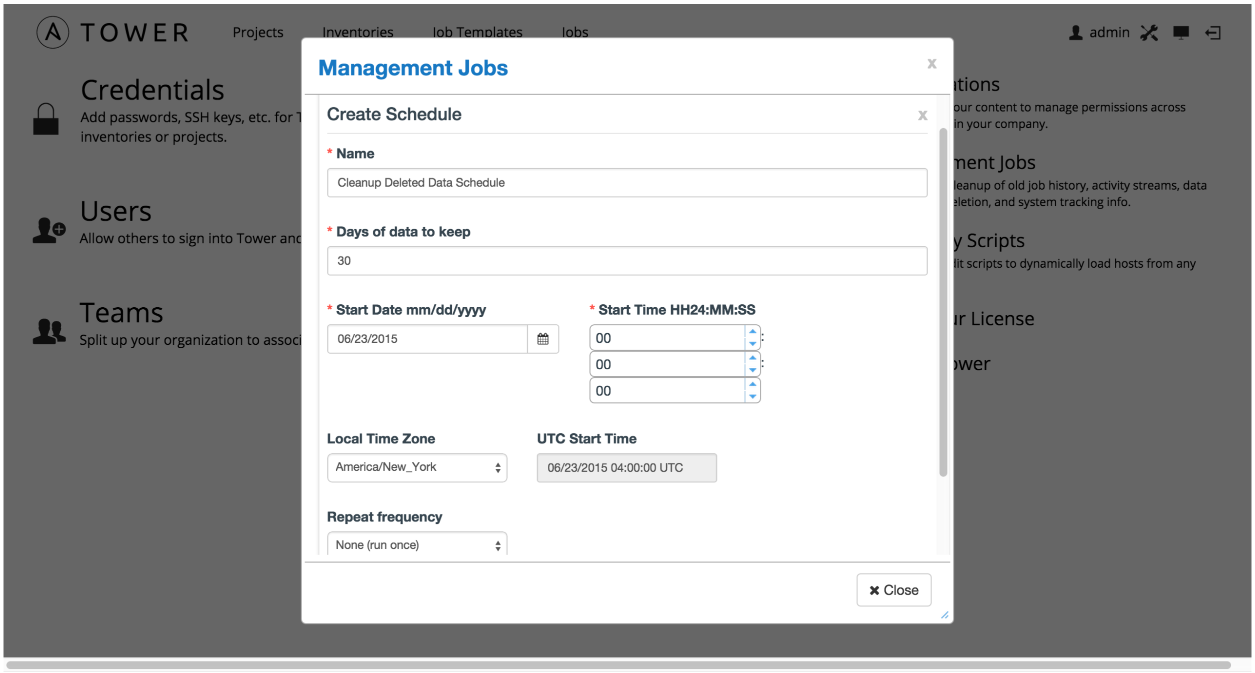
Set how many days of data should be retained.

30

TOWER
Projects
Inventories
Job Templates
Jobs
admin
Credentials
Add passwords, SSH keys, etc. for T
inventories or projects.
Users
Allow others to sign into Tower and
Teams
Split up your organization to assoc
tions
our content to manage permissions across
in your company.
ment Jobs
leanup of old job history, activity streams, data
letion, and system tracking info.
y Scripts
it scripts to dynamically load hosts from any
r License
wer
✖ Cancel
Launch
✖ Close

Enter the number of days of data you would like to save and click **Launch**.

To set a schedule for purging data marked for deletion, click on the 📅 button.


TOWER
Projects
Inventories
Job Templates
Jobs
admin
Credentials
Add passwords, SSH keys, etc. for T
inventories or projects.
Users
Allow others to sign into Tower and
Teams
Split up your organization to associ
tions
our content to manage permissions across
in your company.
ment Jobs
leanup of old job history, activity streams, data
eletion, and system tracking info.
y Scripts
lit scripts to dynamically load hosts from any
r License
ower
✖ Close

Management Jobs

Create Schedule

* Name: Cleanup Deleted Data Schedule

* Days of data to keep: 30

* Start Date mm/dd/yyyy: 06/23/2015

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

Local Time Zone: America/New_York

UTC Start Time: 06/23/2015 04:00:00 UTC

Repeat frequency: None (run once)

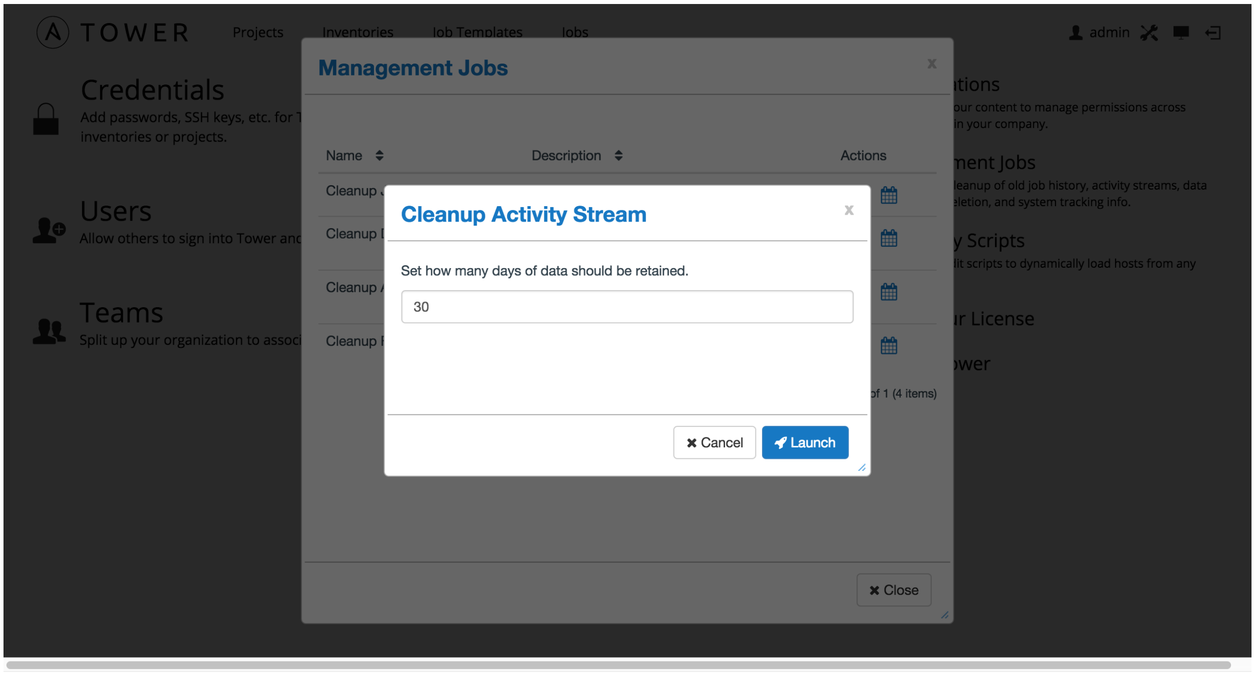
Enter the appropriate details for:

- **Name**: populated by default to match the management job type and can be left alone
- **Days of data to keep**: Enter the number of days of data you would like to retain.
- **Start Date**: Enter the date for which you want data cleanup to start (month/day/year format)
- **Start Time**: Enter the time to start data cleanup in hours, minutes, and seconds.
- **Local Time Zone**: Select your preferred time zone.
- **UTC Start Time**: Review the UTC Start time which is based on the Start Time and Local Time Zone you select.
- **Repeat Frequency**: Select how often you want the cleanup job to run. You can enter cleanups to occur once or every *X* minutes, hours, days, weeks, months, or years.

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

## 4.3 Removing Old Activity Stream Data

To remove older activity stream data, click on the 🚀 button beside **Cleanup Activity Stream**.


TOWER
Projects
Inventories
Job Templates
Jobs
admin
Credentials
Add passwords, SSH keys, etc. for T
inventories or projects.
Users
Allow others to sign into Tower and
Teams
Split up your organization to associ
tions
our content to manage permissions across
in your company.
ment Jobs
leanup of old job history, activity streams, data
eletion, and system tracking info.
y Scripts
lit scripts to dynamically load hosts from any
r License
wer
Management Jobs
Name
Description
Actions
Cleanup J
Cleanup D
Cleanup A
Cleanup F
of 1 (4 items)
Close

Cleanup Activity Stream

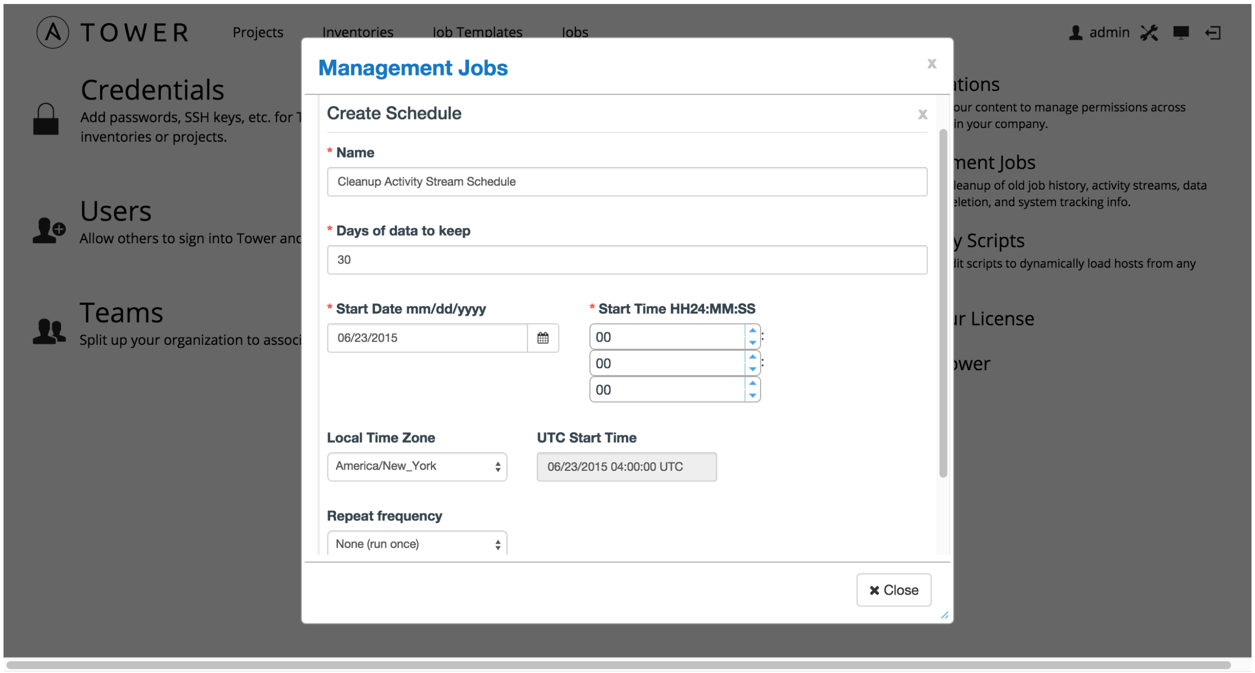
Set how many days of data should be retained.

30

✖ Cancel | Launch

Enter the number of days of data you would like to save and click **Launch**.

To set a schedule for purging data marked for deletion, click on the 📅 button.


TOWER
Projects
Inventories
Job Templates
Jobs
admin
Credentials
Add passwords, SSH keys, etc. for T
inventories or projects.
Users
Allow others to sign into Tower and
Teams
Split up your organization to associ
tions
our content to manage permissions across
in your company.
ment Jobs
leanup of old job history, activity streams, data
eletion, and system tracking info.
y Scripts
lit scripts to dynamically load hosts from any
r License
wer
Management Jobs
✖ Close

Create Schedule

* Name: Cleanup Activity Stream Schedule

* Days of data to keep: 30

* Start Date mm/dd/yyyy: 06/23/2015

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

Local Time Zone: America/New_York

UTC Start Time: 06/23/2015 04:00:00 UTC

Repeat frequency: None (run once)

Enter the appropriate details for:

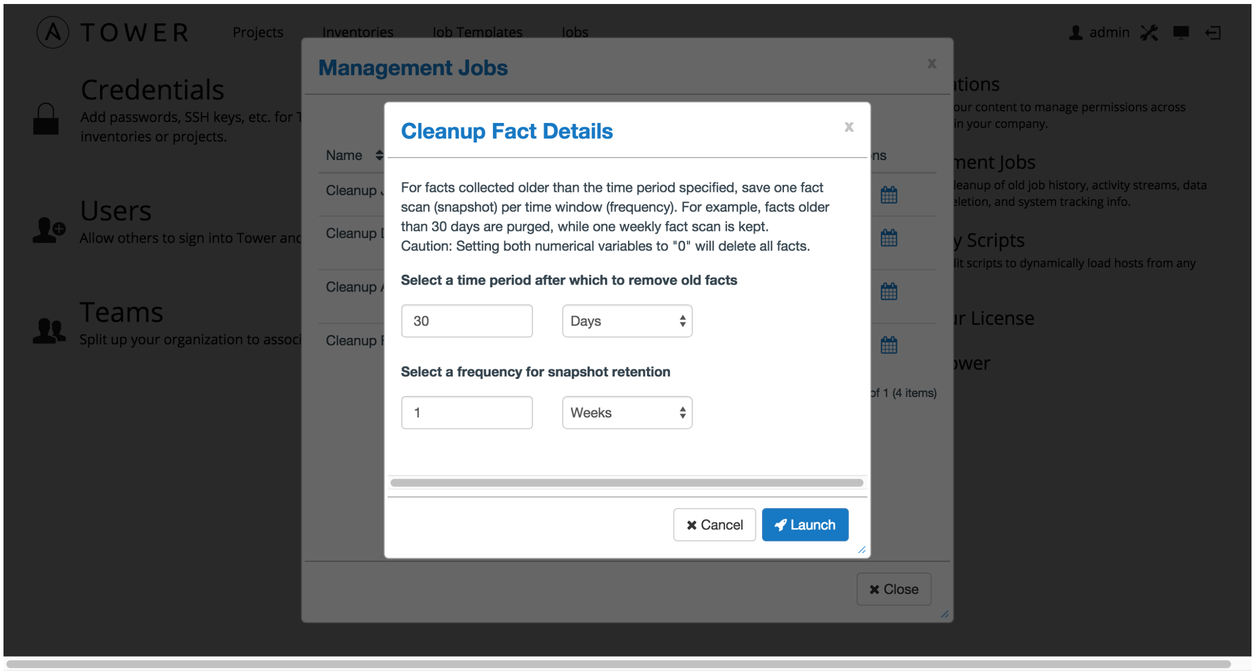
- **Name**: populated by default to match the management job type and can be left alone
- **Days of data to keep**: Enter the number of days of data you would like to retain.
- **Start Date**: Enter the date for which you want data cleanup to start (month/day/year format)
- **Start Time**: Enter the time to start data cleanup in hours, minutes, and seconds.

- **Local Time Zone**: Select your preferred time zone.
- **UTC Start Time**: Review the UTC Start time which is based on the Start Time and Local Time Zone you select.
- **Repeat Frequency**: Select how often you want the cleanup job to run. You can enter cleanups to occur once or every *X* minutes, hours, days, weeks, months, or years.

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

## 4.4 Removing Old System Tracking Data

To remove system tracking data, click on the 🚀 button beside **Cleanup Fact Details**.


TOWER
Projects
Inventories
Job Templates
Jobs
admin
Credentials
Add passwords, SSH keys, etc. for T
inventories or projects.
Users
Allow others to sign into Tower and
Teams
Split up your organization to associ
Management Jobs
Name
Cleanup J
Cleanup D
Cleanup A
Cleanup F
ns
of 1 (4 items)
✖ Close
itions
our content to manage permissions across
in your company.
ment Jobs
leanup of old job history, activity streams, data
eletion, and system tracking info.
y Scripts
lit scripts to dynamically load hosts from any
r License
wer

Cleanup Fact Details

For facts collected older than the time period specified, save one fact scan (snapshot) per time window (frequency). For example, facts older than 30 days are purged, while one weekly fact scan is kept.
Caution: Setting both numerical variables to "0" will delete all facts.

Select a time period after which to remove old facts: 30 Days

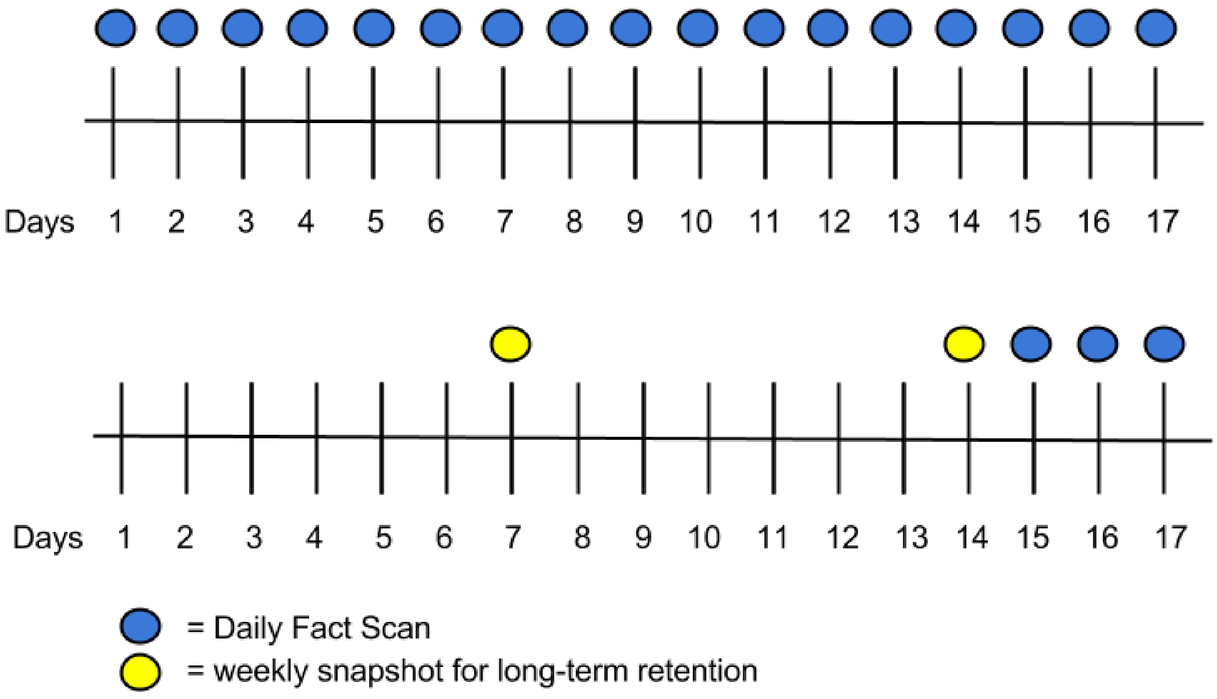
Select a frequency for snapshot retention: 1 Weeks

Select the **time period** after which you want to remove old data as well as the **frequency** for snapshot retention.

For facts collected older than the time period specified, you can choose to save one fact scan (or snapshot) per period of time(frequency). For example, facts older than 30 days could be purged, while one weekly fact scan is retained.

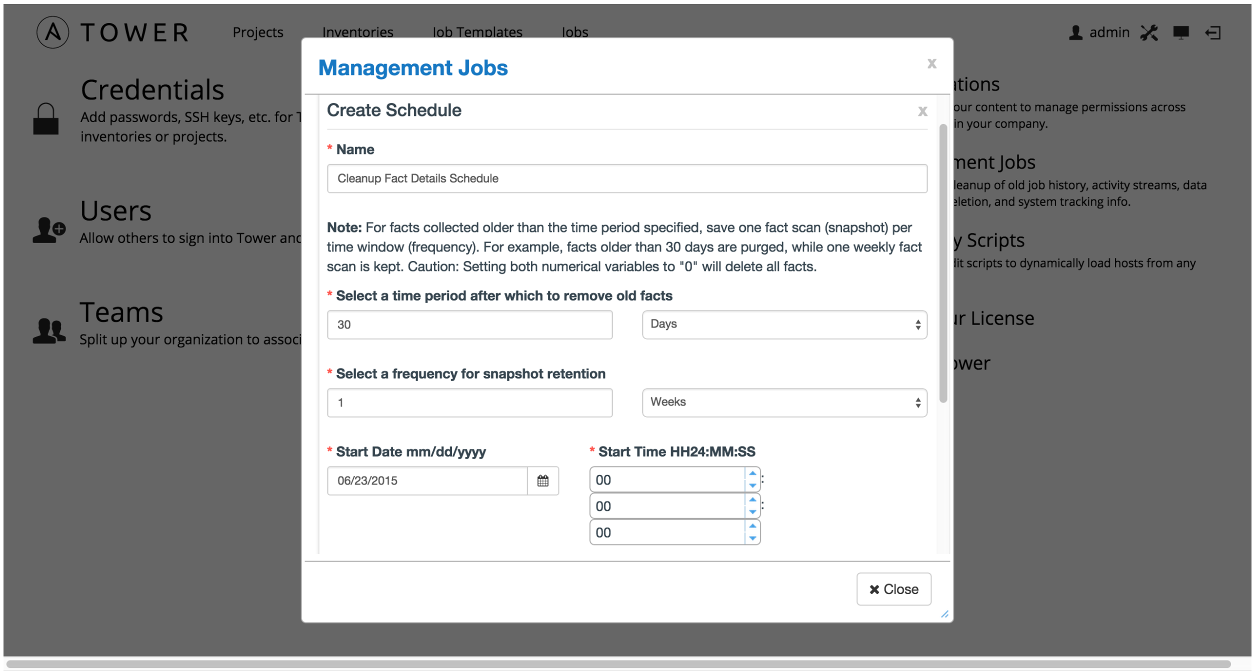
> **Warning:** Setting both numerical variables to "0" will delete all facts.

To help clarify this purge and retention schedule, consider the following timeline:


Days 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
Days 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
= Daily Fact Scan
= weekly snapshot for long-term retention

For this timeline example, consider that you have been running Tower for 17 days (since Jan 1st) and have collected 17 days of fact scans . On Jan 17, you decide to remove all fact scans older than 3 days while keeping a weekly snapshot. The most recent scan and a scan from one week earlier remains, along with the most recent data to be kept.

To set a schedule for cleaning up system tracking information, click on the 📅 button.


TOWER
Projects
Inventories
Job Templates
Jobs
admin
Credentials
Add passwords, SSH keys, etc. for T
inventories or projects.
Users
Allow others to sign into Tower and
Teams
Split up your organization to associ
ations
our content to manage permissions across
in your company.
ment Jobs
leanup of old job history, activity streams, data
eletion, and system tracking info.
y Scripts
lit scripts to dynamically load hosts from any
ur License
ower
✖ Close

**Management Jobs**

Create Schedule

* **Name**

Cleanup Fact Details Schedule

**Note:** For facts collected older than the time period specified, save one fact scan (snapshot) per time window (frequency). For example, facts older than 30 days are purged, while one weekly fact scan is kept. Caution: Setting both numerical variables to "0" will delete all facts.

* **Select a time period after which to remove old facts**

30 Days

* **Select a frequency for snapshot retention**

1 Weeks

* **Start Date mm/dd/yyyy**

06/23/2015

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

00 : 00 : 00

Enter the appropriate details for:

- **Name**: populated by default to match the management job type and can be left alone
- **Select a time period after which to remove old facts**: Select the number of days/weeks/years, after which old data will be removed.
- **Select a frequency for snapshot retention**: Select how often (days/weeks/years) to keep snapshots of your data.
- **Start Date**: Enter the date for which you want data cleanup to start (month/day/year format)
- **Start Time**: Enter the time to start data cleanup in hours, minutes, and seconds.
- **Local Time Zone**: Select your preferred time zone.
- **UTC Start Time**: Review the UTC Start time which is based on the Start Time and Local Time Zone you select.
- **Repeat Frequency**: Select how often you want the cleanup job to run. You can enter cleanups to occur once or every *X* minutes, hours, days, weeks, months, or years.

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

CHAPTER

FIVE

# HIGH AVAILABILITY

**Note:** High Availability support is only available to those with Enterprise-level licenses.

Tower installations can occur in a High Availability (HA) configuration. In this configuration, Tower runs with a single active node, called the Primary instance, and any number of inactive nodes, called Secondary instances. Secondary instances can become Primary at any time, with certain caveats. Running in a high-availability setup requires any database that Tower uses to be external–Postgres and MongoDB must be installed on a machine that is not one of the primary or secondary tower nodes.

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

For instructions on how to install into a HA configuration, refer to the Ansible Tower Installation and Reference Guide.

## 5.1 Setup Considerations

When creating a HA deployment of Tower, consider the following factors:

- Tower is designed as a unit.

  Only those parts explicitly mentioned as being supported as external services are swappable for external versions of those services. Just as Tower does not support swapping out Django for Flask, Apache for lighttpd, or PostgreSQL for Oracle or MSSQL, it does not support replacing MongoDB with a different component.

- Tower servers need isolation.

  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 Tower servers such that distribution occurs 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 require replication.

  If Tower runs in an HA mode, but the database is not run in an 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; instead, it prompts for database connection details to an existing database (which needs replication).

  Choose a database replication strategy that is appropriate for your deployment.

  - For the general case of PostgreSQL, refer to the PostgreSQL documentation or the PostgreSQL Wiki.

  - For deployments using Amazon's RDS, refer to the Amazon documentation.
  - For deployments using MongoDB, refer to the MongoDB Manual.

- Tower instances must maintain reasonable connections to the database.

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

- Source Control is necessary.

  To use playbooks stored locally on the Tower server (rather than set to check out from source control), you must ensure synchronization between the primary and secondary Tower instances. Using playbooks in source control alleviates this problem.

  When using SCM Projects, a best practices approach is setting the *Update on Launch* flag on the job template. This ensures that checkouts occur each time the playbook launches and that newly promoted secondary instances have up-to-date copies of the project content. When a secondary instance is promoted, a project_update for all SCM managed projects in the database is triggered. This provides Tower with copies of all project playbooks.

- A consistent Tower hostname for clients and users.

  Between Tower users' habits, Tower provisioning callbacks, and Tower API integrations, keep the Tower hostname that users and clients use constant. In a HA deployment, use a reverse proxy or a DNS CNAME. The CNAME is strongly preferred due to the websocket connection Tower uses for real-time output.

- When in HA mode, the remote Postgres and MongoDB version requirements are *Postgresql 9.4.x* and *mongodb 3.0.x*.

  Postgresql 9.4.x and Mongodb 3.0.x are also required if Tower is running locally. With local setups, Tower handles the installation of these services. You should ensure that these are setup correctly if working in a remote setup.

  For help allowing remote access to your Postgresql server, refer to: http://www.thegeekstuff.com/2014/02/enable-remote-postgresql-connection/

For example, an HA configuration for an infrastructure consisting of three datacenters places a Tower server and a replicated database in each datacenter. Clients accessing Tower use a DNS CNAME which points to the address of the current primary Tower instance.

For information determining size requirements for a MongoDB setup, refer to Requirements in the *Ansible Tower Installation and Reference Guide*.

**Note:** Instances have been reported where reusing the external DB during subsequent HA installations causes installation failures.

For example, say that you performed an HA installation. Next, say that you needed to do this again and performed a second HA installation reusing the same external database, only this subsequent installation failed.

When setting up an external HA database which has been used in a prior installation, the HA database must be manually cleared before any additional installations can succeed.

## 5.2 Differences between Primary and Secondary Instances

The Tower service runs on both primary and secondary instances. The primary instance accepts requests or run jobs, while the secondary instances do not.

Connection attempts to the web interface or API of a secondary Tower server redirect to the primary Tower instance.

## 5.3 Post-Installation Changes to Primary Instances

When changing the configuration of a primary instance after installation, apply these changes to the secondary instances as well.

Examples of these changes would be:

- Updates to /etc/tower/conf.d/ha.py

  If you have configured LDAP or customized logging in /etc/tower/conf.d/ldap.py, you will need to reflect these changes in /etc/tower/conf.d/ldap.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. Copy the license from the primary node at any time or install it via the normal license installation mechanism after the instance promotes to primary status.

**Note:** Users of older versions of Tower should update /etc/tower/settings.py instead of files within /etc/tower/conf.d/.

## 5.4 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": "2.1.4"
}
```

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 use

## 5.5 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 `update_instance` command of `tower_manage` as follows:

```
root@localhost:~$ tower-manage update_instance --primary
Successfully updated instance role (uuid="ec2dc2ac-7c4b-9b7e-b01f-0b7c30d0b0ab",
↪hostname="localhost",role="primary")
```

The current primary instance changes to be a secondary.

**Note:** Secondary nodes need a valid Tower license at `/etc/tower/license` to function as a proper primary instance. Copy the license from the primary node at any time or install it via the normal license installation mechanism after the instance promotes to primary status.

On failover, queued or running jobs in the database are 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. You should use an external monitoring or heartbeat tool combined with `tower_manage` for these system health checks. Use of the `/ping` API endpoint could help.

**Caution:** HA setup has been designed and tested only for LAN-connected configurations. If you are confident about the quality of your WAN, and research the downsides and possible failure scenarios of a replicated postgresql environment across a WAN, there is nothing technically stopping you from doing it this way. That said, this is outside of our tested configuration, it not recommended, and is unsupported.

## 5.6 Decommissioning Secondary instances

You cannot decommission 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 follows:

```
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.

CHAPTER

SIX

# PROXY SUPPORT

Proxy servers act as an intermediary for requests from clients seeking resources from other servers. A client connects to the proxy server, requesting some service or available resource from a different server, and the proxy server evaluates the request as a way to simplify and control its complexity.

Sessions in Tower associate an IP address upon creation. Tower policy requires that any use of the session match the original associated IP address.

To provide proxy server support, Tower handles proxied requests (such as ELB in front of Tower, HAProxy, Squid, and tinyproxy) via the `REMOTE_HOST_HEADERS` list variable in Tower settings (`/etc/tower/conf.d/remote_host_headers.py`). By default `REMOTE_HOST_HEADERS` is set to `['REMOTE_ADDR', 'REMOTE_HOST']`.

Tower determines the remote host's IP address by searching through the list of headers in `REMOTE_HOST_HEADERS` until the FIRST IP address is located.

**Note:** Header names are constructed using the following logic:

With the exception of `CONTENT_LENGTH` and `CONTENT_TYPE`, any HTTP headers in the request are converted to META keys by converting all characters to uppercase, replacing any hyphens with underscores, and adding an `HTTP_` prefix to the name. For example, a header called `X-Barkley` would be mapped to the META key `HTTP_X_Barkley`.

For more information on HTTP request and response objects, refer to: https://docs.djangoproject.com/en/1.8/ref/request-response/#django.http.HttpRequest.META

If you are behind a reverse proxy, you may want to setup a header field for `HTTP_X_FORWARDED_FOR`. The `X-Forwarded-For` (XFF) HTTP header field identifies the originating IP address of a client connecting to a web server through an HTTP proxy or load balancer.

# CHAPTER SEVEN

# TOWER LOGFILES

Tower logfiles have been consolidated and can be easily accessed from two centralized locations:

- /var/log/tower/
- /var/log/supervisor/

In the /var/log/tower/ directory, you can view logfiles related to:

- callback_receiver.log
- fact_receiver.log
- setup-XX-XX-XX-XX.log
- socketio_service.log
- task_system.log
- tower.log

In the /var/log/supervisor/ directory, you can view logfiles related to:

- awx-celery.log
- supervisord.log

The /var/log/supervisor/ directory include stdout files for all services as well.

```
"Mooving around: Consolidated logfiles for easier access!"
    \
     \   ^__^
      \  (oo)\_______
         (__)\       )\/\
             ||----w |
             ||     ||
```

CHAPTER

EIGHT

# THE *TOWER-MANAGE* UTILITY

The `tower-manage` (formerly awx-manage) utility is used to access detailed internal information of Tower. Commands for `tower-manage` should run as the `awx` or `root` user.

## 8.1 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 synchronizes a 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 (`--source`).

By default, inventory data already stored in Tower blends 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 adds any new variables from the external source, overwriting keys that do not already exist, but preserves any variables that were not sourced from the external data source.

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

## 8.2 Cleanup of old data

`tower-manage` has a variety of commands used to clean old data from Tower. Tower administrators can use the Tower Management Jobs interface for access or use the command line.

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

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

- `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.

## 8.3 HA management

Refer to 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.

CHAPTER

NINE

# SETTING UP AUTHENTICATION

Ansible Tower version 2.4.0 added authentication methods to help simplify logins for end users–offering single sign-ons using existing login information to sign into a third party website rather than creating a new login account specifically for that website.

Social authentication in Ansible Tower can be configured to centrally use OAuth2, while enterprise-level authentication can be configured for SAML, RADIUS, or even LDAP as a source for authentication information.

For websites, such as Google or GitHub, that offer social functionality to users, social login is often implemented using the OAuth standard. OAuth is a secure authorization protocol which is commonly used in conjunction with authentication to grant 3rd party applications a "session token" allowing them to make API calls to providers on the user's behalf.

SAML (Security Assertion Markup Language) is an XML-based, open-standard data format for exchanging authentication and authorization data between an identity provider and a service provider.

The RADIUS distributed client/server system allows you to secure networks against unauthorized access and can be implemented in network environments requiring high levels of security while maintaining network access for remote users.

## 9.1 Basic Authentication Settings:

To enable or disable HTTP basic authentication as used in the API browser, edit the `sessions.py` file, setting the following line as either `True` or `False`:

```
AUTH_BASIC_ENABLED = False
```

After saving your changes, run the `ansible-tower-service restart` command to ensure your changes take effect.

## 9.2 Google OAuth2 Settings

Create a project at https://console.developers.google.com/ and obtain an OAuth2 key and secret for a web application. You will also need to provide the following callback URL for your application, replacing "tower.example.com" with the FQDN to your Tower server: `https://tower.example.com/sso/complete/google-oauth2/`

Ensure that the Google+ API is enabled.

Edit the `/etc/tower/conf.d/social_auth.py` file and enter in the appropriate values:

```
SOCIAL_AUTH_GOOGLE_OAUTH2_KEY = ''
SOCIAL_AUTH_GOOGLE_OAUTH2_SECRET = ''
```

To restrict the domains who are allowed to login using Google OAuth2, uncomment the following line.

```
SOCIAL_AUTH_GOOGLE_OAUTH2_WHITELISTED_DOMAINS = ['example.com']
```

To only allow a single domain to authenticate using Google OAuth2, uncomment the following line; Google will not display any other accounts if the user is logged in with multiple Google accounts.

```
SOCIAL_AUTH_GOOGLE_OAUTH2_AUTH_EXTRA_ARGUMENTS = {'hd': 'example.com'}
```

Refer to the Python Social Auth documentation for advanced settings: https://python-social-auth.readthedocs.org/en/latest/backends/google.htmlgoogle-oauth2

Run `ansible-tower-service restart` on the Tower server to restart the instance and apply your changes.

## 9.3 Github OAuth2 Settings

Create a developer application at https://github.com/settings/developers and obtain an OAuth2 key (Client ID) and secret (Client Secret). You will also need to provide the following callback URL for your application, replacing "tower.example.com" with the FQDN to your Tower server: `https://tower.example.com/sso/complete/github/`

Edit the `/etc/tower/conf.d/social_auth.py` file and enter in the appropriate values:

```
SOCIAL_AUTH_GITHUB_KEY = ''
SOCIAL_AUTH_GITHUB_SECRET = ''
```

When defining authentication with either an organization or a team within an organization, you should use the specific organization and team settings. Authentication can be limited by an organization as well as by a team within an organization.

You can also choose to allow all by specifying non-organization or non-team based settings (as shown above).

To limit users who can login to Tower, in this case, to only those in an organization or on a team within an organization, rather than using:

```
SOCIAL_AUTH_GITHUB_KEY
```

Use the following instead (depending on whether you are setting up for an organization or a team):

```
SOCIAL_AUTH_GITHUB_ORG_KEY

SOCIAL_AUTH_GITHUB_TEAM_KEY
```

To setup authentication for your organization, create an organization-owned application at `https://github.com/organizations/<yourorg>/settings/applications` and obtain an OAuth2 key (Client ID) and secret (Client Secret). Each key and secret must belong to a unique application and cannot be shared or reused between different social authentication backends. Provide the following callback URL for your application, replacing "tower.example.com" with the FQDN to your Tower server: `https://tower.example.com/sso/complete/github-org/`

```
SOCIAL_AUTH_GITHUB_ORG_KEY = ''
SOCIAL_AUTH_GITHUB_ORG_SECRET = ''
SOCIAL_AUTH_GITHUB_ORG_NAME = ''
```

To setup authentication for your team, create a team-owned application at `https://github.com/organizations/<yourorg>/settings/applications` and obtain an OAuth2 key (Client ID) and secret

(Client Secret). Each key and secret must belong to a unique application and cannot be shared or reused between different social authentication backends. You will also need to provide the following callback URL for your application, replacing "tower.example.com" with the FQDN to your Tower server: `https://tower.example.com/sso/complete/github-team/`

Find the numeric team ID using the Github API: http://fabian-kostadinov.github.io/2015/01/16/how-to-find-a-github-team-id/

```
SOCIAL_AUTH_GITHUB_TEAM_KEY = ''
SOCIAL_AUTH_GITHUB_TEAM_SECRET = ''
SOCIAL_AUTH_GITHUB_TEAM_ID = ''
```

Refer to Python Social Auth documentation for advanced settings: https://python-social-auth.readthedocs.org/en/latest/backends/github.html

Run `ansible-tower-service restart` on the Tower server to restart the instance and apply your changes.

## 9.4 SAML Authentication Settings

**Note:** SAML authentication is a feature specific to Enterprise-level license holders.

To setup SAML authentication, edit the `/etc/tower/conf.d/social_auth.py` file and enter in the appropriate values.

Set the `SOCIAL_AUTH_SAML_SP_ENTITY_ID` to a URL for a domain name you own (does not need to be a valid URL as this value is only used as a unique ID).

```
SOCIAL_AUTH_SAML_SP_ENTITY_ID = 'https://tower.example.com'
```

Create a keypair for Tower to use as a service provider (SP) and include the certificate and private key contents:

```
SOCIAL_AUTH_SAML_SP_PUBLIC_CERT = ''
SOCIAL_AUTH_SAML_SP_PRIVATE_KEY = ''
```

As an example for public certs:

~~~
```SOCIAL_AUTH_SAML_SP_PUBLIC_CERT = '''

-----BEGIN CERTIFICATE----
... cert text ...
-----END CERTIFICATE----
~~~

As an example for private keys:

~~~
```SOCIAL_AUTH_SAML_SP_PRIVATE_KEY = '''
-----BEGIN PRIVATE KEY--
... key text ...
-----END PRIVATE KEY----
'''
~~~

Configure the following settings with information about your app and contact information:

```
SOCIAL_AUTH_SAML_ORG_INFO = {
    'en-US': {
        'name': 'example',
```

```
        'displayname': 'Example',
        'url': 'http://www.example.com',
    },
}
SOCIAL_AUTH_SAML_TECHNICAL_CONTACT = {
    'givenName': 'Some User',
    'emailAddress': 'suser@example.com',
}
SOCIAL_AUTH_SAML_SUPPORT_CONTACT = {
    'givenName': 'Some User',
    'emailAddress': 'suser@example.com',
}
```

Configure the entity ID, SSO URL and certificate for each identity provider (IdP) in use. Multiple SAML IdPs are supported.

Some IdPs may provide user data using attribute names that differ from the default OIDs (https://github.com/omab/python-social-auth/blob/master/social/backends/saml.pyL16). Attribute names may be overridden for each IdP as shown below.

```
SOCIAL_AUTH_SAML_ENABLED_IDPS = {
    'myidp': {
        'entity_id': 'https://idp.example.com',
        'url': 'https://myidp.example.com/sso',
        'x509cert': '',
    },
    'onelogin': {
        'entity_id': 'https://app.onelogin.com/saml/metadata/123456',
        'url': 'https://example.onelogin.com/trust/saml2/http-post/sso/123456',
        'x509cert': '',
        'attr_user_permanent_id': 'name_id',
        'attr_first_name': 'User.FirstName',
        'attr_last_name': 'User.LastName',
        'attr_username': 'User.email',
        'attr_email': 'User.email',
    },
}
```

Run `ansible-tower-service restart` on the Tower server to restart the instance and apply your changes.

Once configuration is complete, you will need to register your SP with each IdP. Provide the entity ID and the following callback URL for your application, replacing "tower.example.com" with the FQDN to your Tower server: `https://tower.example.com/sso/complete/saml/`

If your IdP allows uploading an XML metadata file, you can download one from your Tower installation customized with the settings above: `https://tower.example.com/sso/metadata/saml/`

## 9.5 RADIUS Authentication Settings

**Note:** RADIUS authentication is a feature specific to Enterprise-level license holders.

Ansible Tower can be configured to centrally use RADIUS as a source for authentication information. If backed by Active Directory, user access can be configured for RADIUS.

Edit the /etc/tower/conf.d/radius.py file and enter in the appropriate RADIUS server settings (skipped when RADIUS_SERVER is blank):

```
RADIUS_SERVER = ''
RADIUS_PORT = 1812
RADIUS_SECRET = ''
```

Run `ansible-tower-service restart` on the Tower server to restart the instance and apply your changes.

## 9.6 Using LDAP with Tower

**Note:** LDAP authentication is a feature specific to Enterprise-level license holders.

Administrators use LDAP as a source for authentication information for Tower users. User authentication is provided, but not the synchronization of user permissions and credentials. Organization membership (as well as the organization admin) and team memberships can be synchronized.

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

Users created via an LDAP login 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.

LDAP integration for Tower is configured in the file /etc/tower/conf.d/ldap.py. No configuration is accessible via the Tower user interface. Review the comments in that file for information on LDAP configuration and contact Ansible support via the Red Hat Customer Portal if you need help: https://access.redhat.com/

**Note:** Users of older versions of Tower (prior to Tower version 2.3) should update /etc/tower/settings.py instead of files within /etc/tower/conf.d/.

### 9.6.1 Enabling Logging for LDAP

To enable logging for LDAP, you must set the level to DEBUG in the LDAP configuration file, /etc/tower/conf/ldap.py:

```
LOGGING['handlers']['tower_warnings']['level'] =  'DEBUG'
```

## 9.7 Organization and Team Mapping

Next, you will need to control which users are placed into which Tower organizations based on their username and email address (mapping out your organization admins/users from social or enterprise-level authentication accounts).

Dictionary keys are organization names. Organizations will be created, if not already present and if the license allows for multiple organizations. Otherwise, the single default organization is used regardless of the key.

Values are dictionaries defining the options for each organization's membership. For each organization, it is possible to specify which users are automatically users of the organization and also which users can administer the organization.

**admins**: None, True/False, string or list/tuple of strings.

- If None, organization admins will not be updated.
- If True, all users using social authentication will automatically be added as admins of the organization.
- If False, no social authentication users will be automatically added as admins of the organization.
- If a string or list of strings, specifies the usernames and emails for users who will be added to the organization. Compiled regular expressions may also be used instead of string literals.

**remove_admins**: True/False. Defaults to False.

- If True, a user who does not match will be removed from the organization's administrative list.

**users**: None, True/False, string or list/tuple of strings. Same rules apply as for admins.

**remove_users**: True/False. Defaults to False. Same rules as apply for remove_admins.

```
SOCIAL_AUTH_ORGANIZATION_MAP = {
 Add all users to the default organization.
'Default': {
    'users': True,
},
'Test Org': {
    'admins': ['admin@example.com'],
    'users': True,
},
'Test Org 2': {
    'admins': ['admin@example.com', re.compile(r'^tower-[^@]+*?@.*$],
    'users': re.compile(r'^[^@].*?@example\.com$'),
},
}
```

Organization mappings may be specified separately for each social authentication backend. If defined, these configurations will take precedence over the global configuration above.

```
SOCIAL_AUTH_GOOGLE_OAUTH2_ORGANIZATION_MAP = {}
SOCIAL_AUTH_GITHUB_ORGANIZATION_MAP = {}
SOCIAL_AUTH_GITHUB_ORG_ORGANIZATION_MAP = {}
SOCIAL_AUTH_GITHUB_TEAM_ORGANIZATION_MAP = {}
SOCIAL_AUTH_SAML_ORGANIZATION_MAP = {}
```

Mapping of team members (users) from social auth accounts. Keys are team names (will be created if not present). Values are dictionaries of options for each team's membership, where each can contain the following parameters:

**organization**: string. The name of the organization to which the team belongs. The team will be created if the combination of organization and team name does not exist. The organization will first be created if it does not exist. If the license does not allow for multiple organizations, the team will always be assigned to the single default organization.

**users**: None, True/False, string or list/tuple of strings.

- If None, team members will not be updated.
- If True/False, all social auth users will be added/removed as team members.
- If a string or list of strings, specifies expressions used to match users. User will be added as a team member if the username or email matches. Compiled regular expressions may also be used instead of string literals.

**remove**: True/False. Defaults to False. If True, a user who does not match the rules above will be removed from the team.

```
SOCIAL_AUTH_TEAM_MAP = {
'My Team': {
    'organization': 'Test Org',
    'users': ['re.compile(r'^[^@]+?@test\.example\.com$')'],
    'remove': True,
},
'Other Team': {
    'organization': 'Test Org 2',
    'users': re.compile(r'^[^@]+?@test2\.example\.com$'),
    'remove': False,
},
}
```

Team mappings may be specified separately for each social authentication backend, based on which of these you setup. If defined, these configurations will take precedence over the the global configuration above.

```
SOCIAL_AUTH_GOOGLE_OAUTH2_TEAM_MAP = {}
SOCIAL_AUTH_GITHUB_TEAM_MAP = {}
SOCIAL_AUTH_GITHUB_ORG_TEAM_MAP = {}
SOCIAL_AUTH_GITHUB_TEAM_TEAM_MAP = {}
SOCIAL_AUTH_SAML_TEAM_MAP = {}
```

Uncomment the line below (i.e. set `SOCIAL_AUTH_USER_FIELDS` to an empty list) to prevent new user accounts from being created. Only users who have previously logged in to Tower using social or enterprise-level authentication or have a user account with a matching email address will be able to login.

```
SOCIAL_AUTH_USER_FIELDS = []
```

CHAPTER

TEN

# CHANGING THE DEFAULT TIMEOUT FOR AUTHENTICATION

Introduced in Ansible Tower 2.4 is a feature which adds an `Auth-Token-Timeout` to every response that includes a valid user-supplied token. The value of `Auth-Token-Timeout` is determined by the configuration (time expressed in seconds) of the `AUTH_TOKEN_EXPIRATION`.

The value of `Auth-Token-Timeout` indicates the length of time, in seconds, that the supplied token is valid, from the moment the request was initiated.

Create an API settings file (`/etc/tower/conf.d/session.py`) with the appropriately defined time variable:

```
AUTH_TOKEN_EXPIRATION = <seconds>  # default 1800
```

Create a `local_settings.json` file in `/var/lib/awx/public/static/js/local_settings.json` with any necessary settings.

The change from using a `local_config.js` file, which would overwrite all settings in the `config.js` file, to using a `local_settings.json` file, which only overwrites specific settings in the `config.js` file, was introduced in Ansible Tower version 2.4.

**Note:** When including a `local_settings.json` file with specifically configured variables, it will overwrite specific settings in the `config.js` file.

Tower is designed to look for the `config.js` file first, ensuring all preset configuration properties are set properly. Once loaded, Tower looks for a file called `local_settings.json` and checks to see which, if any, settings it should overwrite from `config.js`. Users can now specify only the properties they want to change.

**Note:** If you are using the `local_settings.js` file to configure some of your settings for the UI, you must switch to using the new `local_settings.json` file. If you do not, your Tower instance will be loaded with the default settings. Custom settings will not appear until you switch to using the new `local_settings.json` file.

For example, to turn the console debugger on, `local_settings.json` could contain the following object (no additional variables are needed):

```
{ "debug_mode" : true}
```

The variables you can use within the `local_settings.json` file are as follows:

- `tooltip_delay: {show: 500, hide: 100}` – Default number of milliseconds to delay displaying/hiding tooltips
- `debug_mode: false` – Enable console logging messages
- `password_length: 8` – Minimum user password length. Set to 0 to not set a limit

- `password_hasLowercase: true` – Requires a lowercase letter in the password
- `password_hasUppercase: false` – Requires an uppercase letter in the password
- `password_hasNumber: true` – Requires a number in the password
- `password_hasSymbol: false` – Requires one of these symbols to be in the password: -!$%^&*()_+|~=‘{}[]:”;’<>?,./
- `variable_edit_modes: {yaml, json}` – Options passed to ControlMirror for editing YAML/JSON variables (see below)

```
variable_edit_modes: {
           yaml: {
               mode:"text/x-yaml",
               matchBrackets: true,
               autoCloseBrackets: true,
               styleActiveLine: true,
               lineNumbers: true,
               gutters: ["CodeMirror-lint-markers"],
               lint: true
           },
           json: {
               mode: "application/json",
               styleActiveLine: true,
               matchBrackets: true,
               autoCloseBrackets: true,
               lineNumbers: true,
               gutters: ["CodeMirror-lint-markers"],
               lint: true
           }
       }
```

**Note:** If you are having trouble getting your authentication to stay, in that you have to continuously keep logging in over and over, and you are accessing Tower directly, you may just need to clear your web browser’s cache. In situations like this, often the authentication token has cached in the browser session and needs to be cleared.

CHAPTER

ELEVEN

# WORKING WITH SESSION LIMITS

Setting a session limit allows administrators to limit the number of simultaneous sessions per user or per IP.

In Ansible Tower, a session is created for each browser that a user uses to log in, which forces the user to log out any extra sessions after they exceed the administrator-defined maximum.

Session limits may be important, depending on your particular setup. For example, perhaps you only want a single user on your system with a single login per device (where the user could log in on his work laptop, phone, or home computer). In such a case, you would want to create a session limit equal to 1 (one). If the user logs in on his laptop, for example, then logs in using his phone, his laptop session expires (times out) and only the login on the phone persists.

While session counts can be very limited, they can also be expanded to cover as many session logins as are needed by your organization.

When a user logs in and their login results in other users being logged out, the session limit has been reached and those users who are logged out are notified as to why the logout occurred.

To make changes to your session limits, edit the `sessions.py` file. The settings you should change are detailed below:

```
# Seconds before auth tokens expire.
AUTH_TOKEN_EXPIRATION = 1800

# Maximum number of per-user valid, concurrent tokens.
# -1 is unlimited
AUTH_TOKEN_PER_USER = -1

# Enable / Disable HTTP Basic Authentication used in the API browser
# Note: Session limits are not enforced when using HTTP Basic Authentication.
AUTH_BASIC_ENABLED = True
```

**Note:** To truly make the best use of session limits, disable `AUTH_BASIC_ENABLED` by changing the value to `False` as it falls outside of the scope of session limit enforcement.

**Caution:** Proactive session limits will kick the user out when the session is idle. It is strongly recommended that you do not set the session limit to anything less than 1 minute, as doing so will break your Ansible Tower instance.

CHAPTER

TWELVE

# BACKING UP AND RESTORING TOWER

The ability to backup and restore your system(s) has been integrated into the Tower setup playbook, making it easy for you to backup and replicate your Tower instance as needed.

**Note:** The new backup/restore mechanism only works on systems which have been upgraded or installed as Ansible Tower 2.2. If you have an instance on an earlier version of Tower that you want to backup, for example, you must perform this process manually. For additional information, refer to: https://support.ansible.com/hc/en-us/articles/203295497-Tower-Manual-Backup-Restore

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 for backing up and restoring:

- `-b` Perform a database backup rather than an installation.
- `-r BACKUP_FILE` Perform a database restore rather than an installation.

As the root user, call `setup.sh` with the appropriate parameters and Tower backup or restored as configured.

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

```
root@localhost:~$ ./setup.sh -r BACKUP_FILE
```

## 12.1 Backup/Restore Playbooks

In addition to the `install.yml` file included with your `setup.sh` setup playbook, there are also `backup.yml` and `restore.yml` files for your backup and restoration needs.

These playbooks serve two functions:

- To backup the configuration files, keys, and other relevant files, plus the database of the Tower installation.
- To restore the backed up files and data to a freshly installed and working second instance of Tower.

The `backup.yml` file looks like the following:

```
---
- hosts: primary
  gather_facts: yes
  roles:
    - backup
```

And is called within `setup.sh` as:

```
b)
    PLAYBOOK="backup.yml"
    TEMP_LOG_FILE="backup.log"
    OPTIONS="$OPTIONS --force-handlers"
    ;;
```

The `restore.yml` file looks like the following:

```
---
- hosts: primary
  gather_facts: yes
  roles:
    - restore
```

And is called within `setup.sh` as:

```
r)
    PLAYBOOK="restore.yml"
    TEMP_LOG_FILE="restore.log"
    BACKUP_FILE=`realpath $OPTARG`
    OPTIONS="$OPTIONS --force-handlers"
    ;;
```

When restoring your system, Tower checks to see that the backup file exists before beginning the restoration. If the backup file is not available, your restoration will fail.

**Note:** Ensure your Tower host(s) are properly set up with SSH keys or user/pass variables in the hosts file, and that the user has sudo access.

## 12.2 Backup and Restoration Considerations

- Disk Space: Review your disk space requirements to ensure you have enough room to backup configuration files, keys, and other relevant files, plus the database of the Tower installation
- System Credentials: Confirm you have the system credentials you need when working with a local database or a remote database. On local systems, you may need root or `sudo` access, depending on how credentials were setup. On remote systems, you may need different credentials to grant you access to the remote system you are trying to backup or restore.

CHAPTER

THIRTEEN

# USING CUSTOM LOGOS IN ANSIBLE TOWER

**Note:** Custom rebranding was added to Ansible Tower with version 2.4.0 and is available to Enterprise-level license holders.

To set up a custom logo, you must upload a file called `custom_console_logo.png` into the `/var/lib/awx/public/static/assets/` directory. For the custom logo to look its best, use a `.png` file with a transparent background.

Next, either create or add the following variable to `/var/lib/awx/public/static/local_settings.json`:

```
{ "custom_logo": true }
```

Setting the `custom_logo` variable to `false` (or not setting it at all), results in the appearance of the standard Ansible Tower logo.

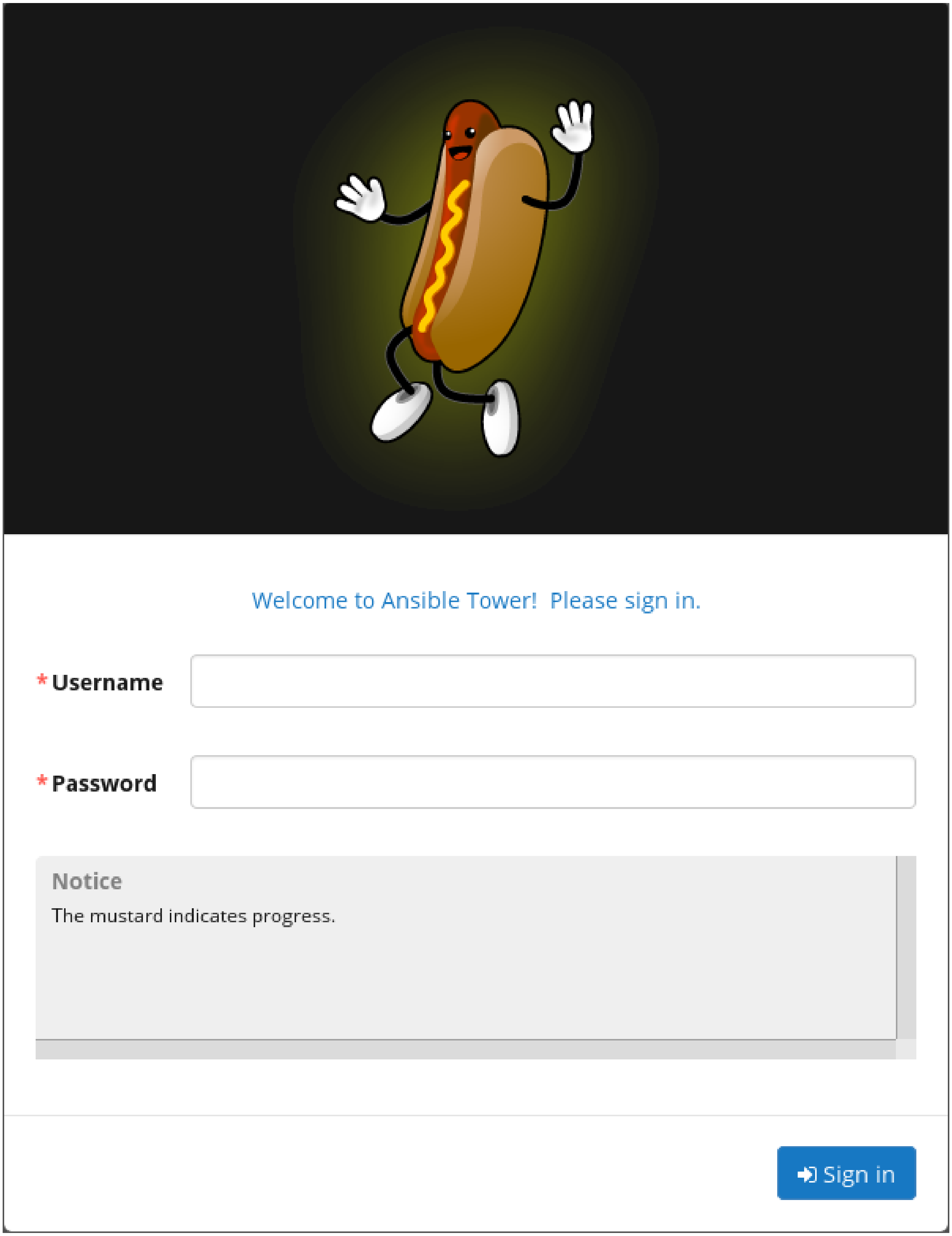
**Note:** If you cannot see the image in the Ansible Tower login modal after moving it into the `assets` directory, you must `chown` this file as the "awx" user, `chmod` this file as `755` for the correct permissions, and fix any SELinux labeling (if you are on Red Hat Enterprise Linux or CentOS, or another OS that uses SELinux). You can do this by using the `wget` or `scp` commands and directly accessing the `assets` directory (`/var/lib/awx/public/static/assets/`).

If needed, you can add specific information (such as a legal notice or a disclaimer) to a text box in the login modal by adding `"custom_login_info": "your notice content here"` to the object (`/var/lib/awx/public/static/local_settings.json`). Any content added must be in plain text, as custom HTML or other markup languages are not supported. If multiple paragraphs of text are needed, new lines (paragraphs) must be escaped as `\n` within the block of text, instead of being entered as a specific lines within the block of text in the `.json` file. Invalid `.json` files will error out if the syntax/formatting is incorrect.

For example, if you customized `/var/lib/awx/public/static/local_settings.json` as follows, and added the appropriate logo:

```
{  "custom_logo": true, "custom_login_info": "The mustard indicates progress." }
```

The Tower login dialog would look like this:



Welcome to Ansible Tower! Please sign in.

Notice

The mustard indicates progress.

With regard to upgrades of Ansible Tower, both of the `local_settings.json` and `custom_console_logo.png` files should persist without any special work.

CHAPTER

FOURTEEN

# TROUBLESHOOTING, TIPS, AND TRICKS

## 14.1 Error logs

Tower server errors are logged in `/var/log/tower`. Supervisors logs can be found in `/var/log/supervisor/`. Apache web server errors are logged in the httpd error log. Configure other Tower logging needs in `/etc/tower/conf.d/`.

Explore client-side issues using the JavaScript console built into most browsers and report any errors to https://access.redhat.com/.

## 14.2 Problems connecting to your host

If you are unable to run the `helloworld.yml` example playbook from the Quick Start Guide or other playbooks due to host connection errors, try the following:

- Can you `ssh` to your host? Ansible depends on SSH access to the servers you are managing.
- Are your hostnames and IPs correctly added in your inventory file? (Check for typos.)

## 14.3 Problems running a playbook

If you are unable to run the `helloworld.yml` example playbook from the Quick Start Guide or other playbooks due to playbook errors, try the following:

- Are you authenticating with the user currently running the commands? If not, check how the username has been setup or pass the `--user=username` or `-u username` commands to specify a user.
- Is your YAML file correctly indented? You may need to line up your whitespace correctly. Indentation level is significant in YAML. You can use `yamlint` to check your playbook. For more information, refer to the YAML primer at: http://docs.ansible.com/YAMLSyntax.html
- Items beginning with a `-` are considered list items or plays. Items with the format of `key: value` operate as hashes or dictionaries. Ensure you don't have extra or missing `-` plays.

## 14.4 Problems when running a job

If you are having trouble running a job from a playbook, you should review the playbook YAML file. When importing a playbook, either manually or via a source control mechanism, keep in mind that the host definition is controlled by Tower and should be set to `hosts: all`.

## 14.5 View a listing of all ansible_ variables

Ansible by default gathers "facts" about the machines under its management, accessible in Playbooks and in templates. To view all facts available about a machine, run the `setup` module as an ad hoc action:

```
ansible -m setup hostname
```

This prints out a dictionary of all facts available for that particular host. For more information, refer to: https://docs.ansible.com/ansible/playbooks_variables.html#information-discovered-from-systems-facts

## 14.6 Locate and configure the configuration file

While Ansible does not require a configuration file, OS packages often include a default one in `/etc/ansible/ansible.cfg` for possible customization. You can also install your own copy in `~/.ansible.cfg` or keep a copy in a directory relative to your playbook named as `ansible.cfg`.

To learn which values you can use in this file, refer to the configuration file on github.

Using the defaults are acceptable for starting out, but know that you can configure the default module path or connection type here, as well as other things.

## 14.7 Playbooks aren't showing up in the "Job Template" drop-down

If your playbooks are not showing up in the Job Template drop-down list, here are a few things you can check:

- Make sure that the playbook is valid YML and can be parsed by Ansible.
- Make sure the permissions and ownership of the project path (/var/lib/awx/projects) is set up so that the "awx" system user can view the files. You can run this command to change the ownership:

```
chown awx -R /var/lib/awx/projects/
```

## 14.8 Playbook stays in pending

If you are attempting to run a playbook Job and it stays in the "Pending" state indefinitely, try the following:

- Ensure all supervisor services are running via `supervisorctl status`.
- Check to ensure that the `/var/` partition has more than 1 GB of space available. Jobs will not complete with insufficient space on the `/var/` partition.
- Run `ansible-tower-service restart` on the Tower server.

If you continue to have problems, run `sosreport` as root on the Tower server, then file a support request with the result.

## 14.9 Cancel a Tower job

When issuing a `cancel` request on a currently running Tower job, Tower issues a `SIGINT` to the `ansible-playbook` process. While this does cause Ansible to exit, Ansible is designed to finish tasks before it exits and only does so *after* the currently running play has completed.

With respect to software dependencies, if a running job is canceled, the job is essentially removed but the dependencies will remain.

## 14.10 View Ansible outputs for JSON commands when using Tower

When working with Ansible Tower, you can use the API to obtain the Ansible outputs for commands in JSON format.

To view the Ansible outputs, browse to:

```
https://<tower server name>/api/v1/jobs/<job_id>/job_events/
```

## 14.11 Reusing an external HA database causes installations to fail

Instances have been reported where reusing the external DB during subsequent HA installations causes installation failures.

For example, say that you performed an HA installation. Next, say that you needed to do this again and performed a second HA installation reusing the same external database, only this subsequent installation failed.

When setting up an external HA database which has been used in a prior installation, the HA database must be manually cleared before any additional installations can succeed.

## 14.12 PRoot functionality and variables

The PRoot functionality in Ansible Tower limits which directories on the Tower file system are available for playbooks to see and use during playbook runs. You may find that you need to customize your PRoot settings in some cases. To fine tune your usage of PRoot, there are certain variables that can be set:

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

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

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

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

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

## 14.13 Changing the WebSockets port for live events

Ansible Tower uses port 8080 on the Tower server to stream live updates of playbook activity and other events to the client browser. If this port is already in use or is blocked by your firewall, you can reconfigure Tower to use a different port.

1. Create a `local_settings.json` file, add an entry for `websocket_port`, and set the value to the desired port, such as: `{"websocket_port": 8080}`.
2. Edit `/etc/awx/settings.py` and add a new line like the following (in this example, 8081 is your new desired port): `SOCKETIO_LISTEN_PORT=8081`
3. Make sure your firewall allows traffic through this port.
4. Restart Tower by running the admin utility script, `ansible-tower-service restart`.

Note that `local_settings.json` is removed when upgrading Tower to a new release. You must recreate and reapply the change on each upgrade of Tower.

## 14.14 Private EC2 VPC Instances in Tower Inventory

By default, Tower only shows instances in a VPC that have an Elastic IP (EIP) associated with them. To see all of your VPC instances, perform the following steps:

1. In the Tower interface, select your inventory.
2. Click on the group that has the Source set to AWS, and click on the Source tab.
3. In the `Source Variables` box, enter:

```
vpc_destination_variable: private_ip_address
```

Next, save and then trigger an update of the group. Once this is done, you should be able to see all of your VPC instances.

**Note:** Tower must be running inside the VPC with access to those instances if you want to configure them.

## 14.15 Using the Tower CLI Tool

Ansible Tower has a full-featured command line interface. It communicates with Tower via Tower's REST API. You can install it from any machine with access to your Tower machine, or on Tower itself.

Installation can be done using the `pip` command:

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

Refer to api_towercli and https://github.com/ansible/tower-cli/blob/master/README.md for configuration and usage instructions.

## 14.16 Changing the Tower Admin Password

During the installation process, you are prompted to enter an administrator password which is used for the `admin` superuser/first user created in Tower. If you log into the instance via SSH, it will tell you the default admin password in the prompt. If you need to change this password at any point, run the following command as root on the Tower server:

```
tower-manage changepassword admin
```

Next, enter a new password. After that, the password you have entered will work as the admin password in the web UI.

## 14.17 Setting up a jump host to use with Tower

Credentials supplied by Tower will not flow to the jump host via ProxyCommand. They are only used for the end-node once the tunneled connection is set up.

To make this work, configure a fixed user/keyfile in the AWX user's SSH config in the ProxyCommand definition that sets up the connection through the jump host. For example:

```
Host tampa
Hostname 10.100.100.11
IdentityFile [privatekeyfile]

Host 10.100..
Proxycommand ssh -W [jumphostuser]@%h:%p tampa
```

**Note:** You must disable PRoot by default if you need to use a jump host. You can disable PRoot by navigating to the `/etc/tower/settings.py` file, setting `AWX_PROOT_ENABLED=False`, then restarting services with the `ansible-tower-service restart` command.

CHAPTER

FIFTEEN

# USABILITY ANALYTICS AND DATA COLLECTION

Introduced in Ansible Tower version 2.4.0 is a behind the scenes functionality that allows Ansible, Inc. to collect usability data with regard to Tower specifically. This software was added to help Ansible, Inc. understand how you or your users are accessing Tower, to help enhance future releases, and to help streamline your user experience.

Only users installing a trial of Tower or a fresh installation of Tower are opted-in for this data collection.

If you want to change how you participate in this analytics collection, you can opt out or change your settings.

To opt out, navigate to the `/etc/tower/` directory and set the following in `settings.py`:

```
PENDO_TRACKING_STATE = 'off'
```

Once set, you must restart your instance of Tower using the `ansible-tower-service restart` command, re-authenticate, and force-reload your browser session.

To re-enable data collection, navigate to the `/etc/tower/` directory and set the following in `settings.py`:

```
PENDO_TRACKING_STATE = 'detailed'
```

Once set, you must restart your instance of Tower using the `ansible-tower-service restart` command, re-authenticate, and force-reload your browser session.

To enable data collection without your specific user data, navigate to the `/etc/tower/` directory and set the following in `settings.py`:

```
PENDO_TRACKING_STATE = 'anonymous'
```

Once set, you must restart your instance of Tower using the `ansible-tower-service restart` command, re-authenticate, and force-reload your browser session.

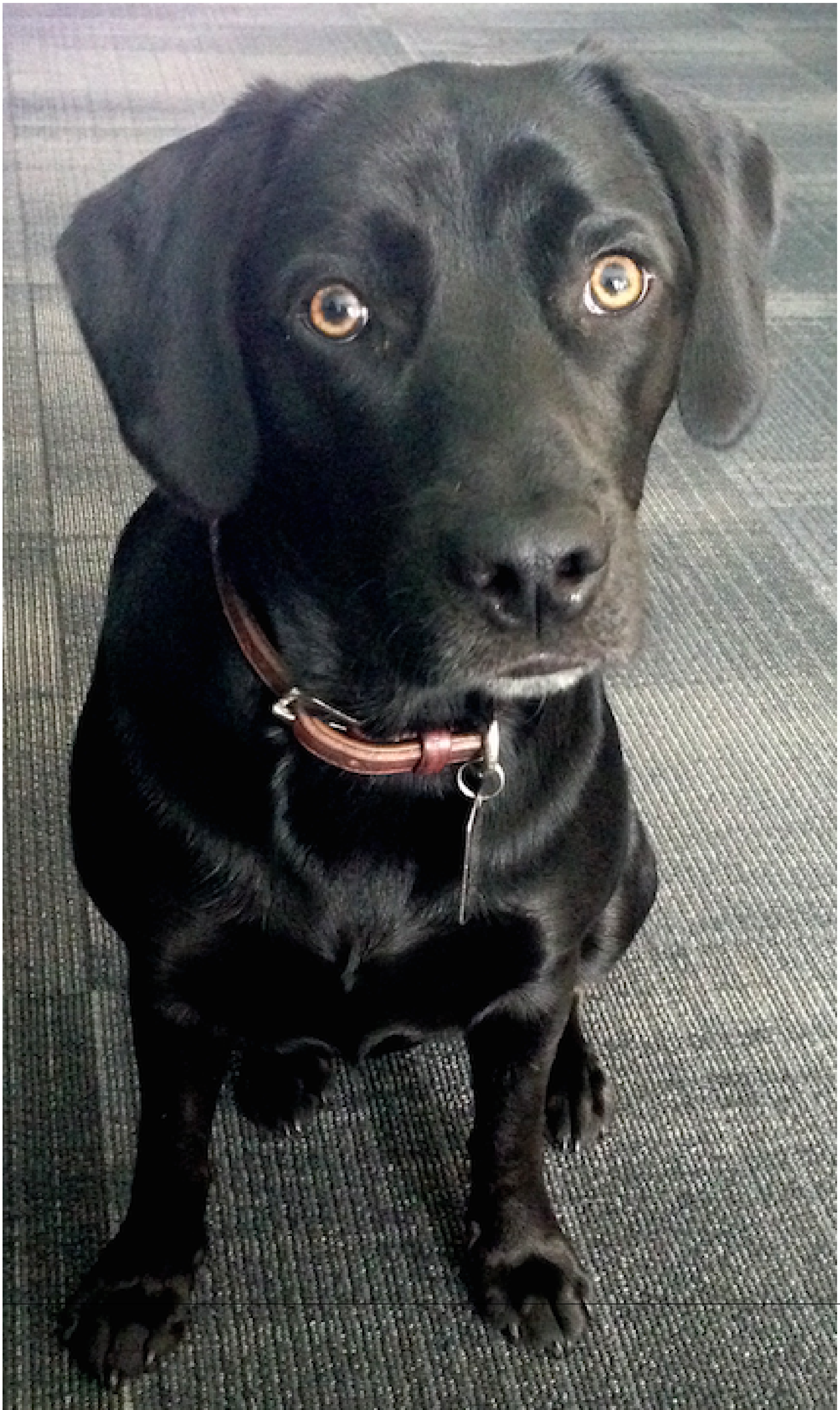
## CHAPTER SIXTEEN

# POSTFACE

Through community efforts, rigorous testing, dedicated engineers, enterprising sales teams, imaginative marketing, and outstanding professional services and support teams, the growing but always impressive group of individuals that make the Ansible-branded products can feel proud in saying:

Ansible, Ansible Tower, Tower CLI, and Ansible Galaxy are all, as Doge would say, “much approved.”[1]

[1] http://knowyourmeme.com/memes/doge



Josie Tested - Doge Approved.

CHAPTER

# SEVENTEEN

# INDEX

- genindex

**Third Party Rights**

# INDEX