

---

# **Ansible Tower Installation and Reference Guide**

*Release Ansible Tower 3.2.4*

**Red Hat, Inc.**

**Jul 01, 2020**

# CONTENTS

<b>1</b>	<b>Tower Licensing, Updates, and Support</b>	<b>2</b>
1.1	Support	2
1.2	Trial / Evaluation	3
1.3	Subscription Types	3
1.4	Node Counting in Licenses	3
1.5	License Features	4
1.6	Tower Component Licenses	4
<b>2</b>	<b>Release Notes</b>	<b>5</b>
2.1	Ansible Tower Version 3.2.4	5
2.2	Ansible Tower Version 3.2.3	5
2.3	Ansible Tower Version 3.2.2	6
2.4	Ansible Tower Version 3.2.1	8
2.5	Ansible Tower Version 3.2.0	8
<b>3</b>	<b>Installation Notes</b>	<b>12</b>
3.1	Notes for Red Hat Enterprise Linux and CentOS setups	12
3.2	Configuration and Installation of Ansible with Red Hat Enterprise Linux and CentOS	12
3.3	Configuration and Installation of Ansible with Ubuntu	13
<b>4</b>	<b>Requirements</b>	<b>15</b>
4.1	Additional Notes on Tower Requirements	16
4.2	Ansible Software Requirements	16
<b>5</b>	<b>Obtaining the Tower Installation Program</b>	<b>18</b>
5.1	Using Vagrant/Amazon AMI Images	18
5.2	Using the Bundled Tower Installation Program	19
<b>6</b>	<b>Installing Ansible Tower</b>	<b>20</b>
6.1	Tower Installation Scenarios	20
6.2	Setting up the Inventory File	21
6.3	The Setup Playbook	24
6.4	Changing the Password	25
<b>7</b>	<b>Upgrading an Existing Tower Installation</b>	<b>26</b>
7.1	Requirements	26
7.2	Backing Up Your Tower Installation	26
7.3	Get the Tower Installer	27
7.4	The Setup Playbook	27
<b>8</b>	<b>Usability Analytics and Data Collection</b>	<b>29</b>

<b>9 Glossary</b>	<b>30</b>
<b>10 Index</b>	<b>33</b>
<b>11 Copyright © 2018 Red Hat, Inc.</b>	<b>34</b>
<b>Index</b>	<b>35</b>

Thank you for your interest in Red Hat Ansible Tower. Ansible Tower is a commercial offering that helps teams manage complex multi-tier deployments by adding control, knowledge, and delegation to Ansible-powered environments.

The *Ansible Tower Installation and Reference Guide* helps you to understand the installation requirements and processes behind installing Ansible Tower. This document has been updated to include information for the latest release of Ansible Tower 3.2.4.

### **We Need Feedback!**

If you spot a typo in this documentation, or if you have thought of a way to make this manual better, we would love to hear from you! Please send an email to: [docs@ansible.com](mailto:docs@ansible.com)

If you have a suggestion, try to be as specific as possible when describing it. If you have found an error, please include the manual's title, chapter number/section number, and some of the surrounding text so we can find it easily. We may not be able to respond to every message sent to us, but you can be sure that we will be reading them all!

Ansible Tower Version 3.2.4; April 27, 2018; <https://access.redhat.com/>

## TOWER LICENSING, UPDATES, AND SUPPORT

Ansible Tower by Red Hat (“**Ansible Tower**”) is a proprietary software product provided via an annual subscription entered into between you and Red Hat, Inc. (“**Red Hat**”).

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

### 1.1 Support

Red Hat offers support for paid **Enterprise: Standard** and **Enterprise: Premium** Subscription customers seeking help with the Ansible Tower product.

If you or your company has paid for Ansible Tower, you can contact the support team at <https://access.redhat.com>. To better understand the levels of support which match your Ansible Tower Subscription, refer to *Subscription Types*.

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

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

#### 1.1.1 Ansible Playbook Support

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

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

Playbook support does not consist of:

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

---

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

Notes:

## 1.2 Trial / Evaluation

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

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

## 1.3 Subscription Types

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

- **Enterprise: Standard (F.K.A. “Enterprise”)**
  - Manage any size environment
  - Enterprise 8x5 support and SLA
  - Maintenance and upgrades included
  - Review the SLA at: <https://access.redhat.com/support/offerings/production/sla>
  - Review the Red Hat Support Severity Level Definitions at: <https://access.redhat.com/support/policy/severity>
- **Enterprise: Premium (F.K.A. “Premium Enterprise”)**
  - Manage any size environment, including mission-critical environments
  - Premium 24x7 support and SLA
  - Maintenance and upgrades included
  - Review the SLA at: <https://access.redhat.com/support/offerings/production/sla>
  - Review the Red Hat Support Severity Level Definitions at: <https://access.redhat.com/support/policy/severity>

All Subscription levels include regular updates and releases of Ansible Tower.

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

## 1.4 Node Counting in Licenses

The Tower license defines the number of Managed Nodes that can be managed by Ansible Tower. A typical license will say ‘License Count: 500’, which sets the maximum number of Managed Nodes at 500.

Ansible Tower counts Managed Nodes by the number of hosts in inventory. If more Managed Nodes are in the Ansible Tower inventory than are supported by the license, you will be unable to start any Jobs in Ansible Tower. If a dynamic inventory sync causes Ansible Tower to exceed the Managed Node count specified in the license, the dynamic inventory sync will fail.

If you have multiple hosts in inventory that have the same name, such as “webserver1”, they will be counted for licensing purposes as a single node. Note that this differs from the ‘Hosts’ count in Tower’s dashboard, which counts hosts in separate inventories separately.

## 1.5 License Features

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

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

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

## 1.6 Tower Component Licenses

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

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

## RELEASE NOTES

The following list summarizes the additions, changes, and modifications which were made to Ansible Tower 3.2.4.

### 2.1 Ansible Tower Version 3.2.4

- Added `UI_LIVE_UPDATES_ENABLED` setting for disabling websocket updates outside of job output
- Fixed organization admins to no longer be able to modify users by adding them to their organization (CVE-2018-1101)
- Fixed Tower to disable usage of Jinja templates in launch-time variables for security reasons (CVE-2018-1104). This release introduces the `ALLOW_JINJA_IN_EXTRA_VARS` configuration parameter for Tower. This parameter has three values: `template` to allow usage of Jinja saved directly on a job template definition (the default), `never` to disable all Jinja usage (recommended), and `always` to always allow Jinja (strongly discouraged, but an option for prior compatibility). Note that the `always` option is deprecated, and will be removed in a future Tower release.
- Fixed sanitization of module arguments with implicit `no_log`
- Fixed Smart Inventories to no longer run on hosts marked as disabled
- Fixed Fact Caching documentation to no longer refer to memcached
- Updated bundled python-saml for CVE-2017-11427
- Updated memcached to now listen on a local Unix socket instead of a TCP socket

### 2.2 Ansible Tower Version 3.2.3

- Added deprecation warning when installing on certain older operating systems, such as Ubuntu 14.04, which will be removed in a future release
- Fixed Inventory Updates to properly save `group_vars` inside of Tower group variables when used with Ansible 2.5 or later
- Fixed certain Inventory Updates to no longer fail when running against isolated nodes
- Fixed the ability to customize `ANSIBLE_LIBRARY` when Job Template fact caching is enabled
- Fixed fact cache data to no longer prematurely expire for Job Templates with large amounts of fact data
- Fixed isolated job runs to no longer fail when the playbook contained certain Unicode characters
- Fixed the installer to use the correct package version when running isolated Tower nodes
- Fixed Slack notification issues



- Fixed workflow artifacts to no longer periodically go missing in subsequent workflow nodes
- Fixed the Tower web interface to support large numbers of custom Credential Types
- Fixed the “Test” button when configuring UDP-based external logging
- Fixed the database restoration process that affected users with embedded PostgreSQL databases
- Fixed a few XSS vulnerabilities in the Tower web interface
- Fixed the ability to provide the admin password in the MOTD file for the Vagrant and AMI images

## 2.3 Ansible Tower Version 3.2.2

- Added support for Ansible Tower and Red Hat Virtualization credentials
- Added dynamic inventory scripts for Ansible Tower and Red Hat Virtualization
- Added `awx_*` extra variables to job runs in addition to `tower_*`
- Added a setting for maximum user interface job events to show to Tower configuration
- Added support for setting the Azure Cloud Environment in Azure credentials
- Added retry for cleaning up job artifacts from isolated nodes
- Added python-crypto requirement to RPM packaging for GCE inventory script
- Added rsync requirement to RPM packaging for isolated nodes
- Added error handling in installation for PostgreSQL 9.4 to 9.6 migration failures
- Removed unused `CALLBACK_CONNECTION`, `CALLBACK_QUEUE`, and `JOB_CALLBACK_DEBUG` environment variables from the job environment
- Fixed multiple issues where survey passwords were not properly encrypted in the database
- Fixed an issue where cleanup jobs could run slowly and exhaust system memory when large job output was present
- Fixed an issue where cleanup jobs could fail due to a race condition
- Fixed an issue where use of `remove: True` and `remove_users: True` in LDAP configuration would cause an excessive number of activity stream entries
- Fixed an issue where the GCE inventory script would erroneously cache information
- Fixed an issue when using Ipsilon as a SAML IdP
- Fixed an issue when using SAML authentication behind a load-balancer
- Fixed an issue where ‘+’ in a search string was not handled properly
- Fixed an issue where non-alphanumeric characters were stripped from SAML usernames
- Fixed an issue where `credential_type` information appeared in `api/v1` output
- Fixed a styling issue for Host Config Key in the Job Template display
- Fixed an issue where it was impossible to remove an organization from a credential
- Fixed an issue where `overwrite_vars` on an inventory source would overwrite inventory toplevel variables
- Fixed an issue where some credential kinds were not properly shown in the user interface
- Fixed calculation of isolated instance capacity

- Fixed an issue where the ‘Workflow Editor’ and ‘Survey Editor’ buttons were incorrectly shown in some states
- Fixed navigation to additional pages of hosts in the Smart Inventory view
- Fixed an issue where CloudForms inventory would not work with process isolation
- Fixed an issue where job output would not properly word wrap
- Fixed a migration issue with unicode inventory source names
- Fixed an issue when launching an ad-hoc command with forbidden extra variables
- Fixed an issue with symlinked manual projects when used with process isolation
- Fixed an issue where some host\_filter queries could not be removed
- Fixed an issue where non-ascii characters could not be used in a LDAP bind DN
- Fixed sizing of the ad-hoc command launch dialog
- Fixed an issue where <https://github.com/ansible/ansible/issues/30064> would prevent project sync
- Fixed an issue where a Smart Inventory host\_filter query would be improperly encoded when saved
- Fixed month name on dashboard chart
- Fixed scheduling error when browser is in UTC timezone
- Fixed autocompletion of SCM inventory file dropdown
- Fixed modal state handling when a modal dialog was closed by clicking outside of it
- Fixed assorted migration errors on upgrade
- Fixed a user interface error when rapidly deleting inventory groups
- Fixed an issue where the system auditor would get a 404 error when viewing job results
- Fixed assorted issues when cascading job cancellation to dependent jobs
- Fixed opacity of disabled ‘Run Commands’ and ‘Smart Inventory’ buttons
- Fixed ‘total\_hosts’ field of Smart Inventories
- Fixed virtualenv paths in sosreport plugins
- Fixed installation with Ansible 2.2
- Fixed ownership on ha.py on installation
- Fixed django superuser check in installation
- Fixed setting of custom RabbitMQ AMQP ports during installation
- Fixed an issue where LDAP authentication could timeout or cause a Tower error
- Improved callback worker’s ability to deal with idle or disconnected database connections
- Improved activity stream output for Tower configuration changes
- Improved deletion of inventory sources to properly delete imported hosts and groups
- Improved various error messages
- Improved initial zoom setting of workflow view
- Improved inline help popovers for credential types
- Improved configuration for SSH key handling for isolated nodes. This is now configurable during setup
- Improved preflight checks for cluster installation

- Improved backup/restore playbooks to be cluster-aware
- Improved error handling in backup/restore playbooks
- Updated translations for Dutch, French, Japanese, and Spanish

## 2.4 Ansible Tower Version 3.2.1

- Added support to enforce Tower software version consistency across clustered environments
- Fixed an issue where, when using Tower 3.2.0 + Ansible 2.4.0, creating a Job Template that used an inventory with fact caching enabled could cause the job to run against a host which should have been removed
- Fixed a problem where ad-hoc permissions could be used to run commands against the Tower server
- Fixed an issue where the migration of scan jobs failed due to an organization having a unicode character in the name
- Fixed an issue where database migrations failed for upgrades

## 2.5 Ansible Tower Version 3.2.0

- Removed system tracking data (historical facts) feature starting with Ansible Tower 3.2. However, you can collect facts by using the fact caching feature. Refer to [Fact Caching](#) for more detail.
- Removed system tracking views in favor of directly viewing facts on hosts. Comparisons are best done with external data analytics systems.
- Removed Rackspace as a supported inventory source type and credential type.
- Removed the storing of `ansible_env` in job event data.
- Removed Job launching capability from `/api/v2/jobs`. Job template launching and job relaunching are the only support launch options.
- Deprecated the `group` field for `InventorySource`, which has been renamed to `deprecated_group` and will be removed from `InventorySource` completely in Tower 3.3. As a result, the related field on `Group`, `inventory_source` has been renamed `deprecated_inventory_source` and will also be removed in Ansible Tower 3.3.
- Deprecated requirement that inventory sources be associated with a group.
- Deprecated the `/api/v1` heirarchy with the introduction of `/api/v2`. `/api/v1` will be removed in a future Ansible Tower release to be determined.
- Deprecated the `/api/v2/authtoken` endpoint, which will be removed in Ansible Tower 3.3.
- Updated the job environment variables for AWS credentials. Refer to [Amazon Web Services](#) section of the *Ansible Tower User Guide* for new variable names.
- Added support for connecting to external log aggregators via direct TCP and UDP connections.
- Added the ability to test logging configurations through the Configure Tower UI.
- Updated the Ansible Tower Rest API to version 2 which include added endpoints: `instances`, `instance_groups`, `credential_types`, and `inventory_sources`.
- Added ability to create inventory sources and create Smart Inventories.
- Added the ability to access Tower resources via resource-specific human-readable identifiers.

- Added the ability to create and modify credential types.
- Added ability to create and modify instance groups and isolated nodes.
- Added the ability to enable and disable SSL certification verification through the Configure Tower UI. You no longer have to manually set an environment variable in your local `settings.py` file to achieve this.
- Updated upstream Azure libraries will require users who use Ansible Tower with Azure to use Ansible 2.4 or later.
- Fixed an outstanding issue regarding variable precedence so that the variable value is derived from the survey (survey variables take precedence over Job Template variables).
- Added Insights project remediation, which allows you to run the Insights maintenance plan associated with an inventory.
- Added a new API endpoint - `/api/v2/settings/logging/test/` - for testing external log aggregator connectivity.
- Updated passing `-e create_preload_data=False` to skip creating default organization/project/inventory/credential/job\_template during Tower installation.
- Added support for sourcing inventory from a file inside of a source control project.
- Added support for custom cloud and network credential types, which give you the ability to modify environment variables, extra vars, and generate file-based credentials (such as file-based certificates or .ini files) at `ansible-playbook` runtime.
- Added support for assigning multiple cloud and network credential types on job templates. Job templates can now prompt for “extra credentials” at launch time in the same manner as promptable machine credentials.
- Updated custom inventory sources to now specify a `Credential`; you can store third-party credentials encrypted within Tower and use their values from within your custom inventory script (for example - by reading an environment variable or a file’s contents).
- Added support for configuring groups of instance nodes to run tower jobs. Instance groups can be assigned to an organization, inventory, or job template.
- Fixed an issue installing Tower on multiple nodes where cluster internal node references are used.
- Updated Tower to now use a modified version of [Fernet](<https://github.com/fernet/spec/blob/master/Spec.md>) for encrypting sensitive fields such as credentials. Our `Fernet256` class uses `AES-256-CBC` instead of `AES-128-CBC` for all encrypted fields.
- Added the ability to set custom environment variables globally for all playbook runs, inventory updates, project updates, and notification sending, via `AWX_TASK_ENV` configuration setting.
- Added `-diff` mode to Job Templates and Ad-Hoc Commands. The diff can be found in the standard out when diff mode is enabled.
- Added support for accessing some Tower resources via their name-related unique identifiers apart from primary keys.
- Added support for authentication to Tower via TACACS+.
- Updated names of tower-mange commands `register_instance` -> `provision_instance`, `deprovision_node` -> `deprovision_instance`, and `instance_group_remove` -> `remove_from_queue`, with backward compatibility support for 3.1 command names.
- Improved handling of workflow logic errors.
- Updated Azure bindings, and therefore, removed support for the old Azure classic modules.
- Fixed system auditor permissions.

- Updated Tower to explicitly prevent non-json bodies from being accepted in the API.
- Improved handling of default values in Tower Configuration.
- Improved handling of sensitive environment variables in job details.
- Added the ability to set the system auditor with `AUTH_LDAP_USER_FLAGS_BY_GROUP`.
- Fixed some minor UTF-8 handling issues.
- Fixed the system to no longer allow using password fields with the `order_by` query parameter in the API.
- Improved censoring of Ansible `no_log` in job output.
- Fixed handling project repository URLs with spaces and special characters.
- Improved explanation when canceling jobs that are dependencies of other jobs.
- Updated the `ansible-playbook` parameters to pass through the `setup.sh` script.
- Added translations for Dutch; updated translations for Japanese, French, and Spanish.
- Improved ability to update org admin/member roles on the user detail page.
- Added force shutdown of cluster nodes that are not at the same version as the rest of the cluster.
- Added configuration options in Tower Configuration UI.
- Updated Postgres to 9.6.
- Updated Tower by separating Vault credentials from machine credentials.
- Added more prompting options to job templates.
- Added the ability to prevent IDP user from assuming a local admin role.
- Improved the display of SCM revision hashes by abbreviating them, and added ability to easily copy revision to clipboard.
- Fixed a potential issue showing encrypted values in the activity stream instead of obfuscation characters.
- Added the ability to set an enabled/disabled flag on all supported cloud inventory sources.
- Added support for vmware `host_filters` and `groupby_patterns`.
- Fixed an issue where Tower wouldn't redirect the user to the right URL after clicking a link and logging in.
- Fixed tower to preserve `stderr` from custom inventory scripts.
- Updated Tower to now act as a fact cache source for jobs.
- Improved handling of related resources when inventories are deleted.
- Added the ability to show an indicator during background inventory delete.
- Updated supported cloud regions for some inventories.
- Improved SAML configurations.
- Improved LDAP settings validation.
- Added support for providing SSL cert for log aggregator service.
- Added the ability to set proxy IP whitelists for trusted vs. untrusted load balancers.
- Improved the efficiency in generating entries in the activity stream.
- Added support for upgrading Ansible during `setup` `playbook` `run` (`-e upgrade_ansible_with_tower=1`).
- Fixed downloading ad-hoc command stdout.

- Fixed job launch dependency handling.
- Fixed some xss vulnerabilities.
- Added runas privilege escalation support.
- Improved handling of instance capacity calculation.
- Fixed SSL certificate handling for LDAP.
- Updated the Job detail event modals to now be resizeable.
- Improved yaml/json editor views.
- Improved job list performance.

For older version of the release notes, as well as other reference materials, refer to the [Ansible Tower Release Notes](#).

## 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/tower.cert` and `/etc/tower/tower.key` for HTTPS communication. These can be replaced after install with your own custom SSL certificates if you desire, but the filenames are required to be the same.
- If using Ansible version 1.8 or later, ensure that fact caching using Redis is not enabled in `ansible.cfg` on the Tower machine.
- 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 and the bundled installation program is not a solution for you (refer to *Using the Bundled Tower Installation Program*), please contact Ansible via the Red Hat Customer portal at <https://access.redhat.com/>.

### 3.1 Notes for Red Hat Enterprise Linux and CentOS setups

- 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.
- When performing a bundled install (refer to *Using the Bundled Tower Installation Program* for more information), Red Hat Enterprise Linux customers must enable the following repositories which are disabled by default:
  - Red Hat Enterprise Linux 7 users must enable the `extras` repositories.

### 3.2 Configuration and Installation of Ansible with Red Hat Enterprise Linux and CentOS

The following steps help you configure access to the repository as well as install Ansible on older versions of Tower.

#### 3.2.1 Configure Repository Access

Configure the EPEL repository and any others needed.

As the root user, for Red Hat Enterprise Linux 7 and CentOS 7

```
root@localhost:~$ yum install http://dl.fedoraproject.org/pub/epel/epel-release-
↳latest-7.noarch.rpm
```

---

**Note:**

- **You may also need to enable the `extras` repository specific for your environment:**
    - `extras` on CentOS 7
    - `rhel-7-server-extras-rpms` on Red Hat Enterprise Linux 7
    - `rhui-REGION-rhel-server-extras` when running in EC2.
  - When using the official Red Hat Enterprise Linux 7 marketplace AMI, ensure that the latest `rh-amazon-rhui-client` package that allows enabling the optional repository (named `rhui-REGION-rhel-server-optional` in EC2) is installed.
- 

### 3.2.2 Install Ansible

---

**Note:** Tower is installed using Ansible playbooks; therefore, Ansible is required to complete the installation of Tower. Beginning with Ansible Tower version 2.3.0, Ansible is installed automatically during the setup process.

If you are using an older version of Tower, prior to version 2.3.0, Ansible can be installed as detailed in the Ansible documentation at: [http://docs.ansible.com/intro\\_installation.html](http://docs.ansible.com/intro_installation.html)

For convenience, those installation instructions are summarized below.

---

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

## 3.3 Configuration and Installation of Ansible with Ubuntu

The following steps help you configure access to the repository as well as install Ansible on older versions of Tower.

### 3.3.1 Configure Repository Access

As the root user, configure Ansible PPA:

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

### 3.3.2 Install Ansible

---

**Note:** Tower is installed using Ansible playbooks; therefore, Ansible is required to complete the installation of Tower. Beginning with Ansible Tower version 2.3.0, Ansible is installed automatically during the setup process.

If you are using an older version of Tower, prior to version 2.3.0, Ansible can be installed as detailed in the Ansible documentation at: [http://docs.ansible.com/intro\\_installation.html](http://docs.ansible.com/intro_installation.html)

---



For convenience, those installation instructions are summarized below.

---

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

## REQUIREMENTS

---

**Note:** Tower is a full application and the installation process installs several dependencies such as PostgreSQL, Django, NGINX, and others. It is required 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 in this guide.

---

Ansible Tower has the following requirements:

- **Supported Operating Systems:**
  - Red Hat Enterprise Linux 7.2 or later 64-bit
  - CentOS 7.2 or later 64-bit
  - Ubuntu 14.04 LTS 64-bit
  - Ubuntu 16.04 LTS 64-bit

---

**Note:** Support for Ubuntu 14.04 as a Tower platform is deprecated and will be removed in a future release.

---

- **A currently supported version of Mozilla Firefox or Google Chrome**
  - Other HTML5 compliant web browsers may work but are not fully tested or supported.
- **2 CPUs minimum**
  - 2 CPUs recommended per 20 forks
- **2 GB RAM minimum** (*4+ GB RAM recommended*)
  - 2 GB RAM (minimum and recommended for Vagrant trial installations)
  - 4 GB RAM is recommended per 100 forks
- **20 GB of dedicated hard disk space** for Tower service nodes
  - 10 GB of the 20 GB requirement must be dedicated to `/var/`, where Tower stores its files and working directories
  - The storage volume should be rated for a minimum baseline of 750 IOPS.
- **20 GB of dedicated hard disk space** for nodes containing a database (*150 GB+ recommended*)
  - The storage volume should be rated for a high baseline IOPS (1000 or more.)

- All Tower data is stored in the database. Database storage increases with the number of hosts managed, number of jobs run, number of facts stored in the fact cache, and number of tasks in any individual job. For example, a playbook run every hour (24 times a day) across 250, hosts, with 20 tasks will store over 800000 events in the database every week.
- If not enough space is reserved in the database, old job runs and facts will need cleaned on a regular basis. Refer to [Management Jobs](#) in the *Ansible Tower Administration Guide* for more information
- **64-bit support required** (kernel and runtime)
- **PostgreSQL version 9.6 (at minimum) required** to run Ansible Tower 3.2 and later
- **Ansible version 2.2 (at minimum) required** to run Ansible Tower versions 3.2 and later

---

**Note:** You cannot use versions of PostgreSQL and Ansible older than those stated above and be able to run Ansible Tower 3.2 and later. Both are installed by the install script if they aren't already present.

---

- **For Amazon EC2:**
  - Instance size of m3.medium or larger
  - An instance size of m3.xlarge or larger if there are more than 100 hosts

## 4.1 Additional Notes on Tower Requirements

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 via the Red Hat Customer portal at <https://access.redhat.com/>. Management of other operating systems (nodes) is documented by the Ansible project itself and allows for a wider list.

Actual RAM requirements 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 4 GB of memory per 100 forks. For example, if `forks` is set to 100, 4 GB of memory is recommended; if `forks` is set to 400, 16 GB of memory is recommended.

For the hosts on which we install Ansible Tower, Tower checks whether or not `umask` is set to 0022. If not, the setup fails. Be sure to set `umask=0022` to avoid encountering this error.

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 are 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 Ansible via the Red Hat Customer portal at <https://access.redhat.com/>.

The requirements for systems managed by Tower are the same as for Ansible at: [http://docs.ansible.com/intro\\_getting\\_started.html](http://docs.ansible.com/intro_getting_started.html)

## 4.2 Ansible Software Requirements

While Ansible Tower depends on Ansible Playbooks and requires the installation of the latest stable version of Ansible before installing Tower, manual installations of Ansible are no longer required.

Beginning with Ansible Tower version 2.3, the Tower installation program attempts to install Ansible as part of the installation process. Previously, Tower required manual installations of the Ansible software release package before running the Tower installation program. Now, Tower attempts to install the latest stable Ansible release package.

If performing a bundled Tower installation, the installation program attempts to install Ansible (and its dependencies) from the bundle for you (refer to *Using the Bundled Tower Installation Program* for more information).

If you choose to install Ansible on your own, the Tower installation program will detect that Ansible has been installed and will not attempt to reinstall it. Note that you must install Ansible using a package manager like `yum` and that the latest stable version must be installed for Ansible Tower to work properly. At minimum, Ansible version 2.2 is required for Ansible Tower versions 3.2 and later.

## OBTAINING THE TOWER INSTALLATION PROGRAM

---

**Note:** To obtain a trial version of Ansible Tower, visit: <http://www.ansible.com/tower-trial>

For pricing information, visit: <http://www.ansible.com/pricing>

To download the latest version of Tower directly (note, you must also obtain a license before using this), visit: <https://releases.ansible.com/ansible-tower/setup/ansible-tower-setup-latest.tar.gz>

---

Download and then extract the Ansible Tower installation/upgrade tool: <http://releases.ansible.com/ansible-tower/setup/>

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

To install or upgrade, start by editing the inventory file in the `ansible-tower-setup-<tower_version>` directory, replacing `<tower_version>` with the version number, such as `2.4.5` or `3.0.0`. directory.

### 5.1 Using Vagrant/Amazon AMI Images

One easy way to try Ansible Tower is to use a Vagrant box or an Amazon EC2 instance, and launching a trial of Ansible Tower just takes a few minutes.

If you use the Vagrant box or Amazon AMI Tower images provided by Ansible, you can find the auto-generated admin password by connecting to the image and reading it from the *message of the day* (MOTD) shown at login.

#### 5.1.1 Vagrant

For Vagrant images, use the following commands to connect:

```
$ vagrant init ansible/tower
$ vagrant up --provider virtualbox
$ vagrant ssh
```

That last command provides your admin password and the Tower log-in URL. Upon login, you will receive directions on obtaining a trial license.

An up-to-date link to Ansible's Vagrant image is available from the [LAUNCH TOWER IN VAGRANT](#) section of Ansible's main website.

## 5.1.2 Amazon EC2

To launch the AMI, you must have an AMI ID (which varies based on your particular AWS region). A list of regions with links to AMI IDs is available in the [LAUNCH TOWER IN AMAZON EC2](#) section of Ansible's main website.

For Amazon AMI images, use the following command to connect:

```
ssh centos@<your amazon instance>
```

You must use the SSH key that you configured the instance to accept at launch time.

## 5.2 Using the Bundled Tower Installation Program

Beginning in Ansible Tower version 2.3.0, Tower installations can be performed using a bundled installation program. The bundled installation program is meant for customers who cannot, or would prefer not to, install Tower (and its dependencies) from online repositories. Access to Red Hat Enterprise Linux or CentOS repositories is still needed.

To download the latest version of the bundled Tower installation program directly (note, you must also obtain a license before using this), visit: <https://releases.ansible.com/ansible-tower/setup-bundle/>

---

**Note:** The bundled installer only supports Red Hat Enterprise Linux and CentOS. Ubuntu support has not yet been added.

---

Next, select the installation program which matches your distribution (e17):

```
ansible-tower-setup-bundle-latest.e17.tar.gz
```

---

**Note:** Red Hat Enterprise Linux customers must enable the following repositories which are disabled by default:

- Red Hat Enterprise Linux 7 users must enable the `extras` repository.

A list of package dependencies from Red Hat Enterprise Linux repositories can be found in the `bundle/base_packages.txt` file inside the setup bundle. Depending on what minor version of Red Hat Enterprise Linux you are running, the version and release specified in that file may be slightly different than what is available in your configured repository.

---

## INSTALLING ANSIBLE TOWER

Tower can be installed in various ways by choosing the best mode for your environment and making any necessary modifications to the inventory file.

### 6.1 Tower Installation Scenarios

Tower can be installed using one of the following scenarios:

#### Single Machine:

- **As an 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.

- **With an external database (2 options available):**

- Tower with remote DB configuration: This installs the Tower server on a single machine and configures it to talk to a remote instance of PostgreSQL 9.6 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 with a playbook install of a remote Postgres system: This installs the Tower server on a single machine and installs a remote Postgres database via the playbook installer (managed by Tower).

---

**Note:** 1). Tower will not configure replication or failover for the database that it uses, although Tower should work with any replication that you have. 2). The database server should be on the same network or in the same datacenter as the Tower server for performance reasons.

---

#### High Availability Multi-Machine Cluster:

Tower can be installed in a high availability cluster mode. In this mode, multiple Tower nodes are installed and active. Any node can receive HTTP requests and all nodes can execute jobs.

- **A Clustered Tower setup must be installed with an external database (2 options available):**

- Tower with remote DB configuration: 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 with a playbook install of a remote Postgres system: This installs the Tower server on a single machine and installs a remote Postgres database via the playbook installer (managed by Tower).

- For more information on configuring a clustered setup, refer to [Clustering](#).

---

**Note:** Running in a cluster setup requires any database that Tower uses to be external—Postgres must be installed on a machine that is not one of the primary or secondary tower nodes. When in a redundant setup, the remote Postgres version requirements is *PostgreSQL 9.6*.

---

## 6.2 Setting up the Inventory File

As you edit your inventory file, there are a few things you must keep in mind:

- The contents of the inventory file should be defined in `./inventory`, next to the `./setup.sh` installer playbook.
- For **installations and upgrades**: If you need to make use of external databases, you must ensure the database sections of your inventory file are properly setup. Edit this file and add your external database information before running the setup script.
- For **clustered installations**: If you are creating a clustered setup, you must replace `localhost` with the hostname or IP address of all instances. All nodes/instances must be able to reach any others using this hostname or address. In other words, you cannot use the `localhost ansible_connection: local` on one of the nodes *AND* all of the nodes should use the same format for the host names.

Therefore, this will *not* work:

```
[tower]
localhost ansible_connection: local
hostA
hostB.example.com
172.27.0.4
```

Instead, use these formats:

```
[tower]
hostA
hostB
hostC
```

OR

```
hostA.example.com
hostB.example.com
hostC.example.com
```

OR

```
[tower]
172.27.0.2
172.27.0.3
172.27.0.4
```

- For **all standard installations**: When performing an installation, you must supply any necessary passwords in the inventory file.

---

**Note:** Changes made to the installation process now require that you fill out the all of the password fields in the inventory file. If you need to know where to find the values for these they should be:



```
admin_password='' ← Tower local admin password
pg_password='' ← Found in /etc/tower/conf.d/postgres.py
rabbitmq_password='' ← create a new password here (alpha-numeric with no special characters)
```

### Example Inventory file

- For **provisioning new nodes**: When provisioning new nodes add the nodes to the inventory file with all current nodes, make sure all passwords are included in the inventory file.
- For **upgrades**: When upgrading, be sure to compare your inventory file to the current release version. It is recommended that you keep the passwords in here even when performing an upgrade.

### Example Single Node Inventory File

```
[tower]
localhost ansible_connection=local

[database]

[all:vars]
admin_password='password'

pg_host=''
pg_port=''

pg_database='awx'
pg_username='awx'
pg_password='password'

rabbitmq_port=5672
rabbitmq_vhost=tower
rabbitmq_username=tower
rabbitmq_password='password'
rabbitmq_cookie=rabbitmqcookie

# Needs to be true for fqdns and ip addresses
rabbitmq_use_long_name=false
# Needs to remain false if you are using localhost
```

### Example Multi Node Cluster Inventory File

```
[tower]
clusternode1.example.com
clusternode2.example.com
clusternode3.example.com

[database]
dbnode.example.com

[all:vars]
ansible_become=true

admin_password='password'

pg_host='dbnode.example.com'
pg_port='5432'
```

```
pg_database='tower'
pg_username='tower'
pg_password='password'

rabbitmq_port=5672
rabbitmq_vhost=tower
rabbitmq_username=tower
rabbitmq_password=tower
rabbitmq_cookie=rabbitmqcookie

# Needs to be true for fqdns and ip addresses
rabbitmq_use_long_name=true
```

### Example Inventory file for an external existing database

```
[tower]
node.example.com ansible_connection=local

[database]

[all:vars]
admin_password='password'
pg_password='password'
rabbitmq_password='password'

pg_host='database.example.com'
pg_port='5432'

pg_database='awx'
pg_username='awx'
```

### Example Inventory file for external database which needs installation

```
[tower]
node.example.com ansible_connection=local

[database]
database.example.com

[all:vars]
admin_password='password'
pg_password='password'
rabbitmq_password='password'

pg_host='database.example.com'
pg_port='5432'

pg_database='awx'
pg_username='awx'
```

Once any necessary changes have been made, you are ready to run `./setup.sh`.

**Note:** Root access to the remote machines is required. With Ansible, this can be achieved in different ways:

- `ansible_user=root ansible_ssh_password="your_password_here"` inventory host or group variables

- `ansible_user=root ansible_ssh_private_key_file="path_to_your_keyfile.pem"` inventory host or group variables
  - `ANSIBLE_BECOME_METHOD='sudo' ANSIBLE_BECOME=True ./setup.sh`
  - `ANSIBLE_SUDO=True ./setup.sh`
- 

## 6.3 The Setup Playbook

**Note:** Ansible Tower 3.0 simplifies installation and removes the need to run `./configure/` as part of the installation setup. Users of older versions should follow the instructions available in the v.2.4.5 (or earlier) releases of the Tower Documentation available at: <http://docs.ansible.com/>

---

The Tower setup playbook script uses the `inventory` file and is invoked as `./setup.sh` from the path where you unpacked the Tower installer tarball.

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

The setup script takes the following arguments:

- `-h` – Show this help message and exit
- `-i INVENTORY_FILE` – Path to Ansible inventory file (default: `inventory`)
- `-e EXTRA_VARS` – Set additional Ansible variables as `key=value` or YAML/JSON (i.e. `-e bundle_install=false` forces an online installation)
- `-b` – Perform a database backup in lieu of installing
- `-r` – Perform a database restore in lieu of installing (a default restore path is used unless `EXTRA_VARS` are provided with a non-default path, as shown in the code example below)

```
./setup.sh -e 'restore_backup_file=/path/to/nondefault/location' -r
```

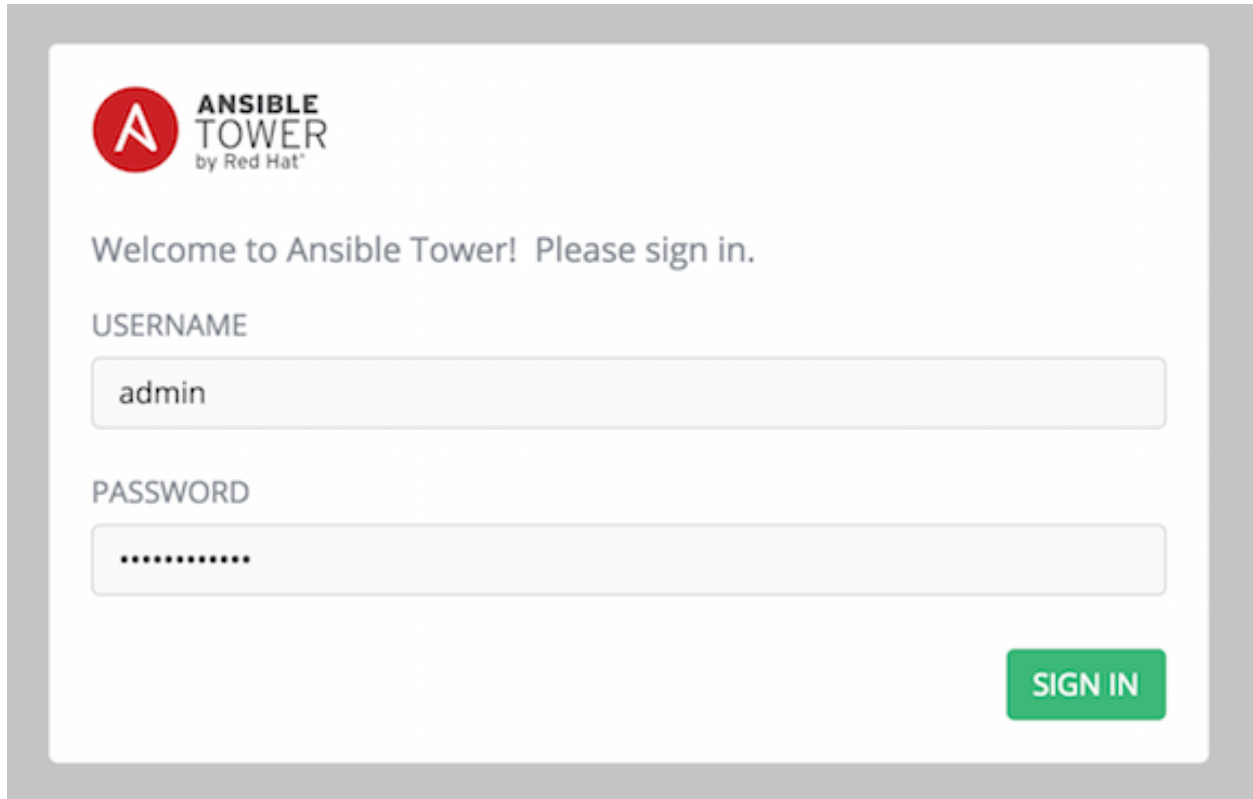
**Note:** Please note that a issue was discovered in Tower 3.0.0 and 3.0.1 that prevented proper system backups and restorations.

If you need to back up or restore your Tower v3.0.0 or v3.0.1 installation, use the v3.0.2 installer to do so.

---

After calling `./setup.sh` with the appropriate parameters, Tower is installed on the appropriate machines as has been configured. Setup installs Tower from RPM or Deb packages using repositories hosted on **ansible.com**.

Once setup is complete, use your web browser to access the Tower server and view the Tower login screen. Your Tower server is accessible from port 80 (<http://tower.company.com/>) but will redirect to port 443 so 443 needs to be available also.



If the installation of Tower fails and you are a customer who has purchased a valid license for Ansible Tower, please contact Ansible via the Red Hat Customer portal at <https://access.redhat.com/>.

## 6.4 Changing the Password

Once installed, if you log into the Tower instance via SSH, the default admin password is provided in the prompt. You can then change it with the following command (as root or as AWX user):

```
awx-manage changepassword admin
```

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

## UPGRADING AN EXISTING TOWER INSTALLATION

You can upgrade your existing Tower installation to the latest version easily. Tower looks for existing configuration files and recognizes when an upgrade should be performed instead of an installation.

As with installation, the upgrade process requires that the Tower server be able to access the Internet. The upgrade process takes roughly the same amount of time as a Tower installation, 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 Active/Passive Redundancy mode installation as part of an upgrade. Users who want to deploy Tower in a Redundant configuration should back up their Tower database, install a new Redundant configuration on a different VM or physical host, and then restore the database. It is possible to add a primary or secondary instance later on to Tower if it is already operating on an external database. Refer to the [Active/Passive Redundancy](#) chapter of the *Ansible Tower Administration Guide*.

---

### 7.1 Requirements

Before upgrading your Tower installation, refer to [Requirements](#) to ensure you have enough disk space and RAM as well as to review any software needs. For example, you should have the latest stable release of Ansible installed before performing an upgrade.

---

**Note:** If you are not yet using a 2.4.x version of Ansible Tower, **do not** attempt to upgrade directly to Ansible Tower 3.0. You must start with a system which has a version of Tower 2.4.x installed or the upgrade will fail.

You must upgrade your Ansible Tower 2.4.4 (or later) system to Ansible Tower 3.0 before you can upgrade to Ansible Tower 3.1.0.

---

### 7.2 Backing Up Your Tower Installation

It is advised that you create a backup before upgrading the system. After the backup process has been accomplished, proceed with OS/Ansible/Tower upgrades.

Refer to [Backing Up and Restoring Tower](#) in the *Ansible Tower Administration Guide*.

---

**Note:** Please note that an issue was discovered in Tower 3.0.0 and 3.0.1 that prevented proper system backups and restorations.

---

If you need to backup or restore your Tower v3.0.0 or v3.0.1 installation, use the v3.0.2 installer to do so.

---

## 7.3 Get the Tower Installer

Download and then extract the Ansible Tower installation/upgrade tool: <http://releases.ansible.com/ansible-tower/setup/>

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

To install or upgrade, start by editing the inventory file in the `ansible-tower-setup-<tower_version>` directory, replacing `<tower_version>` with the version number, such as `2.4.5` or `3.0.0`. directory.

---

**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 Ansible via the Red Hat Customer portal at <https://access.redhat.com/>.

## 7.4 The Setup Playbook

---

**Note:** Ansible Tower 3.0 simplifies installation and removes the need to run `./configure/` as part of the installation setup. Users of older versions should follow the instructions available in the v.2.4.5 (or earlier) releases of the Tower Documentation available at: <http://docs.ansible.com/>

---

The Tower setup playbook script uses the `inventory` file and is invoked as `./setup.sh` from the path where you unpacked the Tower installer tarball.

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

The setup script takes the following arguments:

- `-h` – Show this help message and exit
- `-i INVENTORY_FILE` – Path to Ansible inventory file (default: `inventory`)
- `-e EXTRA_VARS` – Set additional Ansible variables as `key=value` or YAML/JSON (i.e. `-e bundle_install=false` forces an online installation)
- `-b` – Perform a database backup in lieu of installing
- `-r` – Perform a database restore in lieu of installing (a default restore path is used unless `EXTRA_VARS` are provided with a non-default path, as shown in the code example below)

```
./setup.sh -e 'restore_backup_file=/path/to/nondefault/location' -r
```

---

**Note:** Please note that a issue was discovered in Tower 3.0.0 and 3.0.1 that prevented proper system backups and restorations.

---

If you need to back up or restore your Tower v3.0.0 or v3.0.1 installation, use the v3.0.2 installer to do so.

---


## USABILITY ANALYTICS AND DATA COLLECTION

In Ansible Tower version 2.4.0, a behind the scenes functionality was added to Tower to collect usability data. This software was introduced to better understand how Tower users specifically interact with Tower, to help enhance future releases, and to continue streamlining 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 using the Configure Tower user interface.

Ansible Tower collects user data automatically to help improve the Tower product. You can control the way Tower collects data by setting your participation level in the User Interface tab.

1. From the Settings (  ) Menu screen, click on **Configure Tower**.
2. Select the **User Interface** tab.
3. Select the desired level of data collection from the Analytics Tracking State drop-down list:
  - **Off**: Prevents any data collection.
  - **Anonymous**: Enables data collection without your specific user data.
  - **Detailed**: Enables data collection including your specific user data.
4. Click **Save** to apply the settings or **Cancel** to abandon the changes.

---

**Note:** This setting was previously configured via PENDO, which is no longer supported.

---



## GLOSSARY

**Ad Hoc** Refers to running Ansible to perform some quick command, using `/usr/bin/ansible`, rather than the orchestration language, which is `/usr/bin/ansible-playbook`. An example of an ad hoc command might be rebooting 50 machines in your infrastructure. Anything you can do ad hoc can be accomplished by writing a Playbook, and Playbooks can also glue lots of other operations together.

**Callback Plugin** Refers to some user-written code that can intercept results from Ansible and do something with them. Some supplied examples in the GitHub project perform custom logging, send email, or even play sound effects.

**Check Mode** Refers to running Ansible with the `--check` option, which does not make any changes on the remote systems, but only outputs the changes that might occur if the command ran without this flag. This is analogous to so-called “dry run” modes in other systems, though the user should be warned that this does not take into account unexpected command failures or cascade effects (which is true of similar modes in other systems). Use this to get an idea of what might happen, but it is not a substitute for a good staging environment.

**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.

**Facts** Facts are simply things that are discovered about remote nodes. While they can be used in playbooks and templates just like variables, facts are things that are inferred, rather than set. Facts are automatically discovered when running plays by executing the internal setup module on the remote nodes. You never have to call the setup module explicitly, it just runs, but it can be disabled to save time if it is not needed. For the convenience of users who are switching from other configuration management systems, the fact module also pulls in facts from the ‘ohai’ and ‘facter’ tools if they are installed, which are fact libraries from Chef and Puppet, respectively.

**Forks** Ansible and Tower talk to remote nodes in parallel and the level of parallelism can be set several ways—during the creation or editing of a Job Template, by passing `--forks`, or by editing the default in a configuration file. The default is a very conservative 5 forks, though if you have a lot of RAM, you can easily set this to a value like 50 for increased parallelism.

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

**Group Vars** The `group_vars/` files are files that live in a directory alongside an inventory file, with an optional filename named after each group. This is a convenient place to put variables that will be provided to a given group, especially complex data structures, so that these variables do not have to be embedded in the inventory file or playbook.

**Handlers** Handlers are just like regular tasks in an Ansible playbook (see Tasks), but are only run if the Task contains a “notify” directive and also indicates that it changed something. For example, if a config file is changed then the task referencing the config file templating operation may notify a service restart handler. This means services can be bounced only if they need to be restarted. Handlers can be used for things other than service restarts, but service restarts are the most common usage.

**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 Inventory. Sometimes referred to as a “node”.

**Host Specifier** Each Play in Ansible maps a series of tasks (which define the role, purpose, or orders of a system) to a set of systems. This “hosts:” directive in each play is often called the hosts specifier. It may select one system, many systems, one or more groups, or even some hosts that are in one group and explicitly not in another.

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

**Inventory Script** A very simple program (or a complicated one) that looks up hosts, group membership for hosts, and variable information from an external resource—whether that be a SQL database, a CMDB solution, or something like LDAP. This concept was adapted from Puppet (where it is called an “External Nodes Classifier”) and works more or less exactly the same way.

**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 its output and success/failure status.

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

**JSON** Ansible and Tower use JSON for return data from remote modules. This allows modules to be written in any language, not just Python.

**Notifier** An instance of a notification type (Email, Slack, Webhook, etc.) with a name, description, and a defined configuration.

**Notification** A manifestation of the notifier; for example, when a job fails a notification is sent using the configuration defined by the notifier.

**Notify** The act of a task registering a change event and informing a handler task that another action needs to be run at the end of the play. If a handler is notified by multiple tasks, it will still be run only once. Handlers are run in the order they are listed, not in the order that they are notified.

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

**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.

**Plays** A playbook is a list of plays. A play is minimally a mapping between a set of hosts selected by a host specifier (usually chosen by groups, but sometimes by hostname globs) and the tasks which run on those hosts to define the role that those systems will perform. There can be one or many plays in a playbook.

**Playbook** An Ansible playbook. Refer to <http://docs.ansible.com/> for more information.

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

**Roles** Roles are units of organization in Ansible and Tower. Assigning a role to a group of hosts (or a set of groups, or host patterns, etc.) implies that they should implement a specific behavior. A role may include applying certain variable values, certain tasks, and certain handlers—or just one or more of these things. Because of the file structure associated with a role, roles become redistributable units that allow you to share behavior among playbooks—or even with other users.

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

**Sudo** Ansible does not require root logins and, since it is daemonless, does not require root level daemons (which can be a security concern in sensitive environments). Ansible can log in and perform many operations wrapped

in a `sudo` command, and can work with both password-less and password-based `sudo`. Some operations that do not normally work with `sudo` (like `scp` file transfer) can be achieved with Ansible's *copy*, *template*, and *fetch* modules while running in `sudo` mode.

**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.

**Workflow Job Template** A set consisting of any combination of job templates, project syncs, and inventory syncs, linked together in order to execute them as a single unit.

**YAML** Ansible and Tower use YAML to define playbook configuration languages and also variable files. YAML has a minimum of syntax, is very clean, and is easy for people to skim. It is a good data format for configuration files and humans, but is also machine readable. YAML is fairly popular in the dynamic language community and the format has libraries available for serialization in many languages (Python, Perl, Ruby, etc.).

**INDEX**

- genindex

**COPYRIGHT © 2018 RED HAT, INC.**

Ansible, Ansible Tower, Red Hat, and Red Hat Enterprise Linux are trademarks of Red Hat, Inc., registered in the United States and other countries.

If you distribute this document, or a modified version of it, you must provide attribution to Red Hat, Inc. and provide a link to the original version.

**Third Party Rights**

Ubuntu and Canonical are registered trademarks of Canonical Ltd.

The CentOS Project is copyright protected. The CentOS Marks are trademarks of Red Hat, Inc. (“Red Hat”).

Microsoft, Windows, Windows Azure, and Internet Explore are trademarks of Microsoft, Inc.

VMware is a registered trademark or trademark of VMware, Inc.

Rackspace trademarks, service marks, logos and domain names are either common-law trademarks/service marks or registered trademarks/service marks of Rackspace US, Inc., or its subsidiaries, and are protected by trademark and other laws in the United States and other countries.

Amazon Web Services”, “AWS”, “Amazon EC2”, and “EC2”, are trademarks of Amazon Web Services, Inc. or its affiliates.

OpenStack™ and OpenStack logo are trademarks of OpenStack, LLC.

Chrome™ and Google Compute Engine™ service registered trademarks of Google Inc.

Safari® is a registered trademark of Apple, Inc.

Firefox® is a registered trademark of the Mozilla Foundation.

All other trademarks are the property of their respective owners.

## A

- active/passive, external database, clustered installation multi-machine, 20
- Ad Hoc, **30**
- Amazon AMI image, 18
- analytics collection, 29
- Ansible
  - prerequisites, 12
- Ansible, 1.9.4, 16
- Ansible, 2.0, 16
- Ansible, configure repository access, 12, 13
- Ansible, installation, 13
- Ansible, latest stable, 16

## B

- bundled installer, 19

## C

- Callback Plugin, **30**
- CentOS, 12
- Check Mode, **30**
- components
  - licenses, 4
- Credentials, **30**
- current
  - release notes, 5
- custom SSL certificates, 12

## D

- data collection, 29
- DEB files
  - licenses, 4
- download Ansible Tower, 18

## E

- evaluation, 3
- external database
  - installation single machine, 20

## F

- Facts, **30**
- features, 1

- Forks, **30**

## G

- glossary, 30
- Group, **30**
- Group Vars, **30**

## H

- Handlers, **30**
- Host, **30**
- Host Specifier, **31**
- http
  - proxy, 12

## I

- installation, 20
  - general notes, 12
  - multi-machine active/passive, external database, clustered, 20
  - platform-specific notes, 12
  - scenarios, 20
  - single machine external database, 20
  - single machine integrated, 20
- installation bundle
  - licenses, 4
- installation prerequisites, 12
- installation program, 18
  - upgrade, 26
- installation requirements, 15
- installation script
  - inventory file setup, 21
  - playbook setup, 24, 27
- integrated
  - installation single machine, 20
- Inventory, **31**
- inventory file setup, 21
- Inventory Script, **31**
- Inventory Source, **31**

## J

- Job, **31**
- Job Detail, **31**

Job Template, **31**  
 JSON, **31**

## L

license, 1, 2  
     features, 4  
     nodes, 3  
     trial, 3  
     types, 3  
 license features, 1  
 licenses  
     components, 4  
     DEB files, 4  
     installation bundle, 4  
     RPM files, 4

## M

multi-machine  
     active/passive, external database, clustered, installation, 20

## N

Notification, **31**  
 Notifier, **31**  
 Notify, **31**

## O

Organization, **31**  
 Organization Administrator, **31**

## P

password, changing, 25  
 Pendo, 29  
 PENDO\_TRACKING\_STATE, 29  
 Permissions, **31**  
 platform-specific notes  
     CentOS, 12  
     Red Hat Enterprise Linux, 12  
 Playbook, **31**  
 playbook setup, 24, 27  
     installation script, 24, 27  
     setup.sh, 24, 27  
 Plays, **31**  
 prerequisites, 12  
     Ansible, 12  
 Project, **31**

## R

Red Hat Enterprise Linux, 12, 13  
 release notes, 5  
     current, 5  
 requirements, 15  
 Roles, **31**

RPM files  
     licenses, 4

## S

Schedule, **31**  
 self-signed SSL certificate, 12  
 setup.sh  
     playbook setup, 24, 27  
 single machine  
     external database, installation, 20  
     integrated, installation, 20  
 Sudo, **31**  
 Superuser, **32**  
 support, 1–3  
 Survey, **32**

## T

Team, **32**  
 trial, 3

## U

Ubuntu, 13  
 updates, 3  
 upgrade, 26  
     installation program, 26  
 usability data collection, 29  
 User, **32**  
 user data tracking, 29

## V

Vagrant image, 18

## W

Workflow Job Template, **32**

## Y

YAML, **32**