

---

# **Ansible Tower Installation and Reference Guide**

*Release Ansible Tower 3.0*

**Red Hat, Inc.**

**Jul 07, 2020**

# CONTENTS

<b>1</b>	<b>Tower Licensing, Updates, and Support</b>	<b>2</b>
1.1	Support	2
1.2	Trial Licenses	2
1.3	License Types	2
1.4	Node Counting in Licenses	3
1.5	License Features	3
1.6	Tower Component Licenses	4
<b>2</b>	<b>Release Notes</b>	<b>5</b>
<b>3</b>	<b>Installation Notes</b>	<b>7</b>
3.1	Notes for Red Hat Enterprise Linux and CentOS setups	7
3.2	Configuration and Installation of Ansible with Red Hat Enterprise Linux and CentOS	7
3.3	Configuration and Installation of Ansible with Ubuntu	8
<b>4</b>	<b>Requirements</b>	<b>10</b>
4.1	Additional Notes on Tower Requirements	11
4.2	Ansible Software Requirements	12
<b>5</b>	<b>Obtaining the Tower Installation Program</b>	<b>13</b>
5.1	Using Vagrant/Amazon AMI Images	13
5.2	Using the Bundled Tower Installation Program	14
<b>6</b>	<b>Installing Ansible Tower</b>	<b>15</b>
6.1	Tower Installation Scenarios	15
6.2	The Setup Playbook	16
6.3	Changing the Password	17
<b>7</b>	<b>Upgrading an Existing Tower Installation</b>	<b>19</b>
7.1	Requirements	19
7.2	Backing Up Your Tower Installation	19
7.3	Get the Tower Installer	19
7.4	The Setup Playbook	20
<b>8</b>	<b>Usability Analytics and Data Collection</b>	<b>21</b>
<b>9</b>	<b>Glossary</b>	<b>22</b>
<b>10</b>	<b>Index</b>	<b>25</b>
<b>11</b>	<b>Copyright © 2016 Red Hat, Inc.</b>	<b>26</b>



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

The *Ansible Tower 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 3.0.

Ansible Tower Version 3.0; July 19, 2016; <https://access.redhat.com/>

## 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 your company has paid for a license of Ansible Tower, you can contact Ansible Support 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.

- **Self-Support**
  - Manage smaller environments (up to 250 nodes)
  - Maintenance and upgrades included
- **Enterprise: Standard**
  - Manage any size environment

- Enterprise 8x5 support and SLA (4 hour critical incident response, 24 hour response non-critical issues)
- Maintenance and upgrades included
- **Enterprise: Premium**
  - Manage any size environment, including mission-critical environments
  - Premium 24x7 support and SLA (2 hour critical incident response, 8 hour non-critical incident response)
  - Maintenance and upgrades included

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

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

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

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

## 1.5 License Features

---

**Note:** Ansible Tower version 2.2 introduced a separation of features for Basic (now called Self Support) versus Enterprise (now Enterprise: Standard) or Premium (now Enterprise: Premium) licenses.

---

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

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

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

## 1.6 Tower Component Licenses

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

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

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

## RELEASE NOTES

The following list summarizes the additions, changes, and modifications which were made to Ansible Tower 3.0.

- Added a notifications system for Tower which supports services like Slack, HipChat, IRC, etc.
- Added support for the new Azure inventory system and the latest Ansible Azure modules (legacy Azure inventory and credentials are still supported)
- Added support for keystone v3 which supports the latest Openstack versions
- Added counts and more detail to Organization endpoints (API)
- Added prompting for Job Templates
- Added labels for Job Templates
- Added support for user customization as Ansible tasks now run in their own environment
- Added support for new Ansible Network Credentials
- Added inventory support for Red Hat Cloudforms and Red Hat Satellite 6
- Added SUSE, OpenSuse, and Debian support for scan jobs
- Added a link to the schedule in the job detail view if the job was started as a result of a schedule
- Added survey spec management without requiring that surveys be enabled on job templates
- Added additional strict extra\_vars validation. extra\_vars passed to the job launch API are only honored if one of the following is true:
  - they correspond to variables in an enabled survey
  - ask\_variables\_on\_launch is set to True
- Added a deprecation notice for Ubuntu 12 and RHEL 6
- Changed how Projects are linked so that they now tie singularly to an Organization
- Changed how system tracking and scan data are stored—now in postgres. MongoDB dependency removed.
- Discovered an issue with ECDSA credentials—if your Tower server has a version of OpenSSH that predates 5.7, jobs will fail when launched jobs with ECDSA credentials.
- Fixed issues with scan jobs on RHEL5
- Fixed an issue with the websocket service when Tower is run on CentOS or RHEL 7.2
- Fixed issues with Ansible’s no\_log causing errors or not hiding data when running jobs
- Fixed the way setting a license is done so that it propagates to standby Tower nodes in an HA configuration
- Fixed GCE credential handling and inventory filtering



- Improved (through a complete rewrite to expand and simplify) the Role-Based Access Control system in Tower
- Improved job templates so that multiple invocations of the same job template will only block if the job templates used the same inventory
- Improved the setup playbook so that it now hides potentially sensitive information from stdout and the setup log
- Improved the Setup process now supports installing and configuring postgres on a remote system
- Removed MongoDB and changed view queries to use a Postgres implementation
- Removed soft-deletes. Tower now permanently deletes removed objects. Removed utilities to manage the cleanup of those soft-deleted objects
- Removed Munin monitoring
- Updated the look and feel of the entire Tower UI for a more approachable and intuitive user experience
- Updated and simplified the Tower setup process so that new Tower installs are now preloaded with Organization, Inventory, Project, and Job Template demo data
- Updated the setup process to support installing and configuring Postgres on a remote system
- Updated dependencies
- Updated RHEL 6/CentOS 6 to use python 2.7 (for Tower only)
- Updated the minimum open file descriptor check and configuration by raising it from 1024 to 4096

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 Support 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.
  - Red Hat Enterprise Linux 6 users must enable the `optional` repository.

### 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 6 and CentOS 6:

```
root@localhost:~$ yum install http://dl.fedoraproject.org/pub/epel/epel-release-
↳latest-6.noarch.rpm
```

---

**Note:**

- For users of Red Hat Enterprise Linux 6, you must enable the `optional` repository.

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, Apache, 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 6 64-bit
  - Red Hat Enterprise Linux 7 64-bit
  - CentOS 6 64-bit
  - CentOS 7 64-bit
  - Ubuntu 12.04 LTS 64-bit
  - Ubuntu 14.04 LTS 64-bit

---

**Note:** Starting with version 3.0 of Ansible Tower, support for running Tower on Red Hat Enterprise Linux 6 and CentOS 6 has been deprecated, and will be removed in a future release.

Starting with version 3.0 of Ansible Tower, support for running Tower on Ubuntu 12.04 has been deprecated and will be removed in a future release.

---

- **An HTML5 compliant web browser**
- **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**
  - 10 GB of the 20 GB requirement must be dedicated to `/var/`, where Tower stores its files and working directories (dedicating less space will cause the installation to fail)
- **64-bit support required** (kernel and runtime)
- **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

- **For Passive/Active Redundancy setups:**

- If you plan to run in a redundancy setup, the following guidelines provide a rough estimate for the amount of space required. The basic calculation is:

$$\frac{(\text{number of hosts in inventory}) * (\text{number of scans}) * ((\text{average module fact size}) * (\text{number of modules scanning}))}{3}$$

- For example, assuming a schedule of 1 scan per day for a year:

$$(\text{hosts} = 1,000) * (\text{number of scans} = 365) * ((\text{average module fact size} = 100 \text{ kb}) * (\text{number of modules} = 4) / 3) = 48 \text{ GB}$$

The default scan operation has the four (4) modules listed, but you can add your own. Depending on the kinds of modules and the size of the facts you are gathering, that size might be larger.

To help keep the size down, you can use a management job to purge old facts. Refer to [Management Jobs](#) in the *Ansible Tower Administration Guide* for more information

---

**Note:** Ansible Tower 3.0 moved away from MongoDB in favor of using Postgres. The new Postgres data-type consumes about one-third (1/3) less space than the equivalent human-readable JSON with no whitespace or newlines. If you are using an older version of Ansible Tower, you should use the following example to determine how much space may be required:

$$(\text{hosts} = 1,000) * (\text{number of scans} = 365) * (\text{average module fact size} = 100 \text{ kb}) * (\text{number of modules} = 4) = 146 \text{ GB}$$

---

## 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 Support 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.

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 Support 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.

## 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/awx/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
```

Replace `ansible-tower-2.4.X` with the version that you are trying to install. 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 root@<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 (el6 or el7):

```
ansible-tower-setup-bundle-latest.el6.tar.gz  
ansible-tower-setup-bundle-latest.el7.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` repositories.
- Red Hat Enterprise Linux 6 users must enable the `optional` 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

As 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 install 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).

---

**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.

---

#### Active/Passive Redundancy Multi-Machine:

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

- **An Active/Passive Tower setup can also be installed with an external database (2 options available):**

- Tower with remote DB configuration: This install 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).

**Note:** Running in a redundancy 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 9.4.x*.

---

If you are performing an installation, you can supply any necessary passwords here.

For those who are upgrading, your prior configuration will migrate over and filling out the passwords in the inventory file is unnecessary.

```
admin_password='password'  
redis_password='password'  
pg_password='password'
```

As an example:

```
[primary]  
localhost ansible_connection=local  
  
[secondary]  
  
[database]  
  
[all:vars]  
admin_password='password'  
redis_password='password'  
  
pg_host=''  
pg_port=''  
  
pg_database='awx'  
pg_username='awx'  
pg_password='password'
```

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

## 6.2 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.

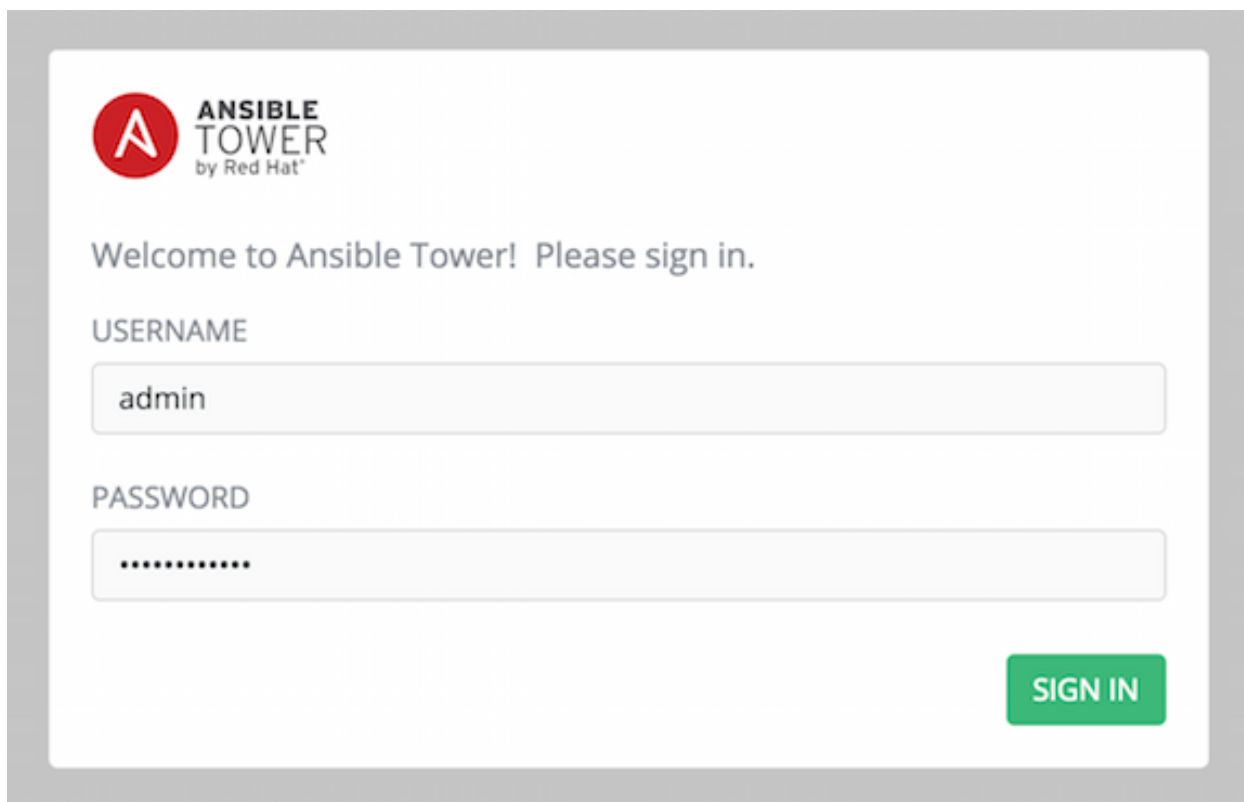
```
root@localhost:~$ ./setup.sh
```

The setup script takes the following arguments:

- `-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
- `-h` – Show this help message and exit

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/>).



If the installation of Tower fails, please contact Ansible Support via the Red Hat Customer Portal at <https://access.redhat.com/>.

## 6.3 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):

```
tower-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 upgrade.

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 *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.

---

### 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*.

### 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 Support 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:

- `-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
- `-h` – Show this help message and exit

## 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.

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.



## 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.

**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 © 2016 RED HAT, INC.**

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

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

**Third Party Rights**

Ubuntu and Canonical are registered trademarks of Canonical Ltd.

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

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

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

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

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

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

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

Safari® is a registered trademark of Apple, Inc.

Firefox® is a registered trademark of the Mozilla Foundation.

All other trademarks are the property of their respective owners.

**A**

- active/passive, external database
  - installation multi-machine, 15
- Ad Hoc, 22
- Amazon AMI image, 13
- analytics collection, 21
- Ansible
  - prerequisites, 7
- Ansible, 1.9.4, 12
- Ansible, 2.0, 12
- Ansible, configure repository access, 8, 9
- Ansible, installation, 8, 9
- Ansible, latest stable, 12

**B**

- bundled installer, 14

**C**

- Callback Plugin, 22
- CentOS, 8
- Check Mode, 22
- components
  - licenses, 4
- Credentials, 22
- custom SSL certificates, 7

**D**

- data collection, 21
- DEB files
  - licenses, 4
- download Ansible Tower, 13

**E**

- external database
  - installation single machine, 15

**F**

- Facts, 22
- features, 1
- Forks, 22

**G**

- glossary, 22
- Group, 22
- Group Vars, 22

**H**

- Handlers, 22
- Host, 22
- Host Specifier, 23

**I**

- installation, 15
  - general notes, 7
  - multi-machine active/passive, external database, 15
  - platform-specific notes, 7
  - scenarios, 15
  - single machine external database, 15
  - single machine integrated, 15
- installation bundle
  - licenses, 4
- installation prerequisites, 7
- installation program, 13
  - upgrade, 19
- installation requirements, 10
- installation script
  - playbook setup, 16, 20
- integrated
  - installation single machine, 15
- Inventory, 23
- Inventory Script, 23
- Inventory Source, 23

**J**

- Job, 23
- Job Detail, 23
- Job Template, 23
- JSON, 23

**L**

- license, 1, 2
  - features, 3
  - nodes, 3

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

## M

- multi-machine
  - active/passive, external database, installation, 15

## N

- Notification, 23
- Notifier, 23
- Notify, 23

## O

- Organization, 23
- Organization Administrator, 23

## P

- password, changing, 17
- Pendo, 21
- PENDO\_TRACKING\_STATE, 21
- Permissions, 23
- platform-specific notes
  - CentOS, 7
  - Red Hat Enterprise Linux, 7
- Playbook, 23
- playbook setup, 16, 20
  - setup.sh, 16, 20
- Plays, 23
- prerequisites, 7
  - Ansible, 7
- Project, 23

## R

- Red Hat Enterprise Linux, 8
- release notes, 5
- requirements, 10
- Roles, 23
- RPM files
  - licenses, 4

## S

- Schedule, 23
- self-signed SSL certificate, 7
- single machine
  - external database, installation, 15
  - integrated, installation, 15
- Sudo, 23

- Superuser, 24
- support, 1, 2
- Survey, 24

## T

- Team, 24

## U

- Ubuntu, 9
- updates, 2
- upgrade, 19
  - installation program, 19
- usability data collection, 21
- User, 24
- user data tracking, 21

## V

- Vagrant image, 13

## Y

- YAML, 24