
Ansible Tower Quick Install

Release Ansible Tower 3.1.4

Red Hat, Inc.

Jul 12, 2017

CONTENTS

1	Preparing for the Tower Installation	2
1.1	Installation and Reference guide	2
1.2	Prerequisites and Requirements	2
1.3	Additional Notes on Tower Requirements	3
1.4	Ansible Software Requirements	4
1.5	Platform-specific Installation Notes	4
1.6	Tower Installation Scenarios	6
2	Download the Tower Installation Program	7
2.1	Using Vagrant/Amazon AMI Images	7
2.2	Using the Bundled Tower Installation Program	8
3	Installing Ansible Tower	9
3.1	Tower Installation Scenarios	9
3.2	Setting up the Inventory File	10
3.3	The Setup Playbook	13
3.4	Changing the Password	14
4	Congratulations	15
5	Index	16
6	Copyright © 2017 Red Hat, Inc.	17
	Index	18

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 Quick Installation Guide* covers basic installation instructions for installing Ansible Tower on Red Hat Enterprise Linux, CentOS, and Ubuntu systems. This document has been updated to include information for the latest release of Ansible Tower 3.1.4.

Ansible Tower Version 3.1.4; July 10, 2017; <https://access.redhat.com/>

PREPARING FOR THE TOWER INSTALLATION

This guide helps you get your Ansible Tower installation up and running as quickly as possible. At the end of the installation, using your web browser, you can access and fully utilize Tower.

1.1 Installation and Reference guide

While this guide covers the basics, you may find that you need the more detailed information available in the [Installation and Reference Guide](#).

You should also review the [General Installation Notes](#) before starting the installation.

1.2 Prerequisites and Requirements

For platform information, refer to [Installation Notes](#) in the *Ansible Tower Installation and Reference Guide*.

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: Ansible Tower requires Red Hat Enterprise Linux 7.2 or later.

- **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 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 System Tracking data storage:**
 - If you plan to utilize Tower's system tracking, 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:

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

1.3 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

1.4 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.

For convenience, summaries of those instructions are in the following sections.

1.5 Platform-specific Installation Notes

1.5.1 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.

1.5.2 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.

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.
-

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

For convenience, those installation instructions are summarized below.

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

1.5.3 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.

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
```

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

For convenience, those installation instructions are summarized below.

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

1.6 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 9.4.X 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 9.4.X*.

DOWNLOAD 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.

2.1 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.

2.1.1 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.

2.1.2 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.

2.2 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 (el7):

```
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` 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.

3.1 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 9.4.X 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 9.4.X*.

3.2 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 **redundant installations**: If you are creating a redundant 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 **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'
pg_password='password'
```

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'
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_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`
-

3.3 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)

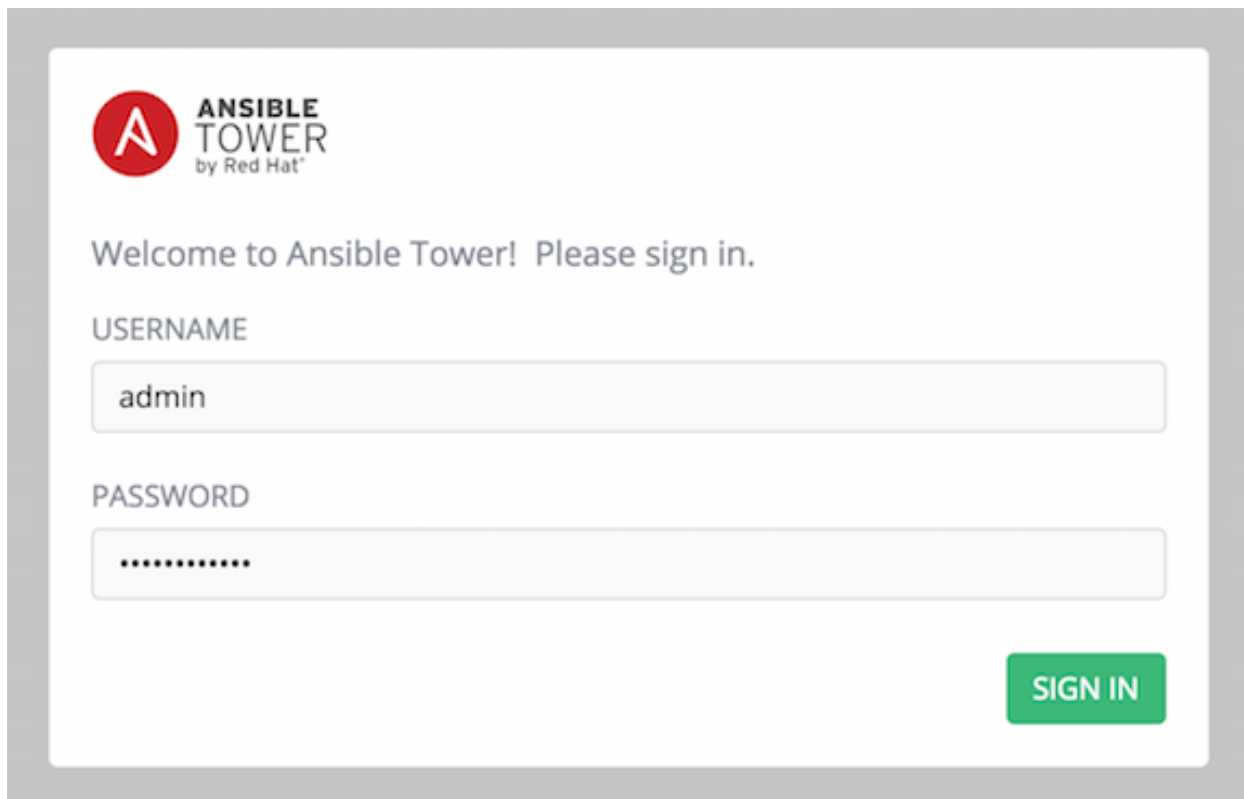
```
./setup.sh -e 'restore_backup_file=/path/to/nondefault/location' -r
```

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.

The image shows the Ansible Tower login interface. At the top left is the Ansible Tower logo, which consists of a red circle with a white 'A' inside, followed by the text 'ANSIBLE TOWER by Red Hat'. Below the logo, the text 'Welcome to Ansible Tower! Please sign in.' is displayed. Underneath this, there are two input fields. The first is labeled 'USERNAME' and contains the text 'admin'. The second is labeled 'PASSWORD' and contains a series of dots representing a masked password. To the right of these fields is a green button with the text 'SIGN IN' in white capital letters.

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/>.

3.4 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):

```
tower-manage changepassword admin
```

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

CONGRATULATIONS

Once the installation of Tower is complete, you are ready to set up and launch your first Ansible playbook using Tower.

If you are wondering what to do next, refer to the following list of Ansible documentation sets for information on getting started, administration, and more:

- [Ansible Tower Quick Setup Guide](#)
- [Ansible Tower Installation and Reference Guide](#)
- [Ansible Tower User Guide](#)
- [Ansible Tower Administration Guide](#)
- [Ansible Tower API Guide](#)
- <http://docs.ansible.com/>

INDEX

- `genindex`

COPYRIGHT © 2017 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, [6, 9](#)
- Amazon AMI image, [7](#)
- Ansible
 - prerequisites, [4](#)
- Ansible, 1.9.4, [4](#)
- Ansible, 2.0, [4](#)
- Ansible, configure repository access, [5](#)
- Ansible, installation, [5](#)
- Ansible, latest stable, [4](#)

B

- bundled installer, [8](#)

C

- CentOS, [5](#)

D

- download Ansible Tower, [7](#)

E

- external database
 - installation single machine, [6, 9](#)

I

- installation, [9](#)
 - multi-machine active/passive, external database, clustered, [6, 9](#)
 - platform-specific notes, [2, 4](#)
 - scenarios, [6, 9](#)
 - single machine external database, [6, 9](#)
 - single machine integrated, [6, 9](#)
- installation prerequisites, [2, 4](#)
- installation program, [7](#)
- installation requirements, [2](#)
- installation script
 - inventory file setup, [10](#)
 - playbook setup, [13](#)
- integrated
 - installation single machine, [6, 9](#)
- inventory file setup, [10](#)

M

- multi-machine
 - active/passive, external database, clustered, installation, [6, 9](#)

O

- operating system requirements, [2](#)

P

- password, changing, [14](#)
- platform-specific notes
 - CentOS, [4](#)
 - Red Hat Enterprise Linux, [4](#)
- platform-specific notes
 - CentOS, [2](#)
 - Red Hat Enterprise Linux, [2](#)
- playbook setup, [13](#)
 - setup.sh, [13](#)
- prerequisites, [2, 4](#)
 - Ansible, [4](#)

R

- Red Hat Enterprise Linux, [5](#)
- requirements, [2](#)
- requirements, Ansible, [2](#)
- resources, [2](#)

S

- single machine
 - external database, installation, [6, 9](#)
 - integrated, installation, [6, 9](#)

U

- Ubuntu, [5](#)

V

- Vagrant image, [7](#)