# Ansible Tower Installation and Reference Guide

*Release Ansible Tower 3.4.2*

**Red Hat, Inc.**

**May 05, 2021**

# CONTENTS


**1 Tower Licensing, Updates, and Support** **2**
- 1.1 Support . . . 2
- 1.2 Trial / Evaluation . . . 2
- 1.3 Subscription Types . . . 2
- 1.4 Node Counting in Licenses . . . 3
- 1.5 Tower Component Licenses . . . 3

**2 Release Notes** **4**
- 2.1 Ansible Tower Version 3.4.2 . . . 4

**3 Installation Notes** **5**
- 3.1 Flags and extra vars passed with Tower . . . 5
- 3.2 Installing Tower on Systems with FIPS Mode Enabled . . . 6
- 3.3 Notes for Red Hat Enterprise Linux and CentOS setups . . . 7
- 3.4 Configuration and Installation of Ansible with Red Hat Enterprise Linux and CentOS . . . 7
- 3.5 Configuration and Installation of Ansible with Ubuntu . . . 8
- 3.6 Configuration and Installation on OpenShift . . . 8

**4 Requirements** **9**
- 4.1 Additional Notes on Tower Requirements . . . 10
- 4.2 Ansible Software Requirements . . . 11

**5 Obtaining the Tower Installation Program** **12**
- 5.1 Using the Bundled Tower Installation Program . . . 12
- 5.2 Using Vagrant/Amazon AMI Images . . . 13

**6 Installing Ansible Tower** **14**
- 6.1 Tower Installation Scenarios . . . 14
- 6.2 Setting up the Inventory File . . . 15
- 6.3 The Setup Playbook . . . 18
- 6.4 Changing the Password . . . 19

**7 Upgrading an Existing Tower Installation** **20**
- 7.1 Requirements . . . 20
- 7.2 Backing Up Your Tower Installation . . . 20
- 7.3 Get the Tower Installer . . . 21
- 7.4 The Setup Playbook . . . 21

**8 Usability Analytics and Data Collection** **23**

**9 Glossary** **24**

10 Index 27

11 Copyright © 2019 Red Hat, Inc. 28

Index 29

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 3.4.2.

**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

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 3.4.2; March 06, 2019; https://access.redhat.com/

CHAPTER

ONE

# TOWER LICENSING, UPDATES, AND SUPPORT

Red Hat Ansible Tower ("**Ansible Tower**") is a software product provided as part of 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 to paid Red Hat Ansible Automation customers.

If you or your company has purchased a subscription for Ansible Automation, 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*. For details of what is covered under an Ansible Automation subscription, please see the Scopes of Support at: https://access.redhat.com/support/policy/updates/ansible-tower#scope-of-coverage-4 and https://access.redhat.com/support/policy/updates/ansible-engine.

## 1.2 Trial / Evaluation

While a license is required for Ansible Tower to run, there is no fee for a trial license.

- Trial licenses for Red Hat Ansible Automation are available at: http://ansible.com/license
- Support is not included in a trial license or during an evaluation of the Tower Software.

## 1.3 Subscription Types

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

- **Standard (F.K.A. "Enterprise: Standard")**
  - 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 (F.K.A. "Enterprise: Premium")**
  - 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 nodes 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.

For more information on managed node requirements for licensing, please see https://access.redhat.com/articles/3331481.

## 1.5 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.

CHAPTER

TWO

# RELEASE NOTES

The following list summarizes the additions, changes, and modifications which were made to Ansible Tower 3.4.2.

## 2.1 Ansible Tower Version 3.4.2

- Added a number of notable improvements to translations
- Fixed the Notifications tab to properly appear when viewing a Job Template
- Fixed the total host count to be correct when viewing job output
- Fixed the `awx_workflow_job_name` extra variable to be correct for jobs launched via nested workflows
- Fixed scheduled jobs to no longer run twice in clustered Tower installations
- Fixed the display of playbooks that utilize Ansible's `serial` keyword
- Fixed instance group assignments to restore correctly after Tower is backed up and restored
- Fixed a number of workflow builder user interface issues
- Fixed the Job Template credential selection to display all available options when large numbers of Credential Types are in use
- Fixed the stdout display to no longer delay when Smart Inventories are in use
- Fixed a bug that prevented Ansible to upgrade when Tower is updated with `-e upgrade_ansible_with_tower=1`
- Fixed the installer to properly upgrade Tower dependencies on upgrade
- Fixed a performance issue in Ansible Tower 3.4.0 which caused a notable delay in job start time for large inventories (>1000 hosts).
- Fixed Tower logs to now properly record failed logins
- Updated `settings.SCHEDULE_MAX_JOBS` so that it also limits workflow Job Templates

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

CHAPTER

THREE

# 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/.
- If installing Tower on OpenShift, refer to OpenShift Deployment and Configuration.

## 3.1 Flags and extra vars passed with Tower

Flags and/or extra variables that you can use with the Ansible Tower installer include (but are not limited to) the following:

```
Usage: setup.sh [Options] [-- Ansible Options]
Options:
        -i INVENTORY_FILE     Path to ansible inventory file (default: ${INVENTORY_
↪FILE})
        -e EXTRA_VARS         Set additional ansible variables as key=value or YAML/
↪JSON
                                i.e. -e bundle_install=false will force an online␣
↪install
        -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
Ansible Options:
        Additional options to be passed to ansible-playbook can be added following␣
↪the -- separator.
```

Use the `--` separator to add any Ansible arguments you wish to apply. For example: `./setup.sh -i my_awesome_inventory.yml -e matburt_is_awesome=True -- -K`

### 3.1.1 Examples

The following are examples of common scenarios - be sure to supply your own values appropriate to your specific case.

- **To upgrade core**:

```
./setup.sh -e upgrade_ansible_with_tower=1
```

- **To disable https handling at nginx**:

```
./setup.sh -e nginx_disable_https=true
```

- **To specify a non-default path when restoring from a backup file**:

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

- **To override an inventory file used by passing it as an argument to the setup script**:

```
setup.sh -i <inventory file>
```

## 3.2 Installing Tower on Systems with FIPS Mode Enabled

Tower can run on systems where FIPS mode is enabled, though there are a few limitations to keep in mind:

- Only Enterprise Linux 7+ is supported. The standard python that ships with RHEL must be used for Ansible Tower to work in FIPS mode. Using any non-standard, non-system python for Tower is therefore, unsupported.
- By default, Tower configures PostgreSQL using password-based authentication, and this process relies on the usage of `md5` when `CREATE USER` is run at install time. If you intend to run the Tower installer from a FIPS-enabled system, you'll need to pre-compute an `md5` hash yourself at install time and add its value to the inventory file described in *Setting up the Inventory File*.

  Specifically, you must pre-compute the hashed value on a system *without* FIPS enabled:

```
$ python -c 'from hashlib import md5; print("md5" + md5("choose-a-password" + "awx
↪").hexdigest())'
md57d08dde7e95e862eaadfff09565e92e6
```

  ...and specify **both** `pg_password` and `pg_hashed_password` in your inventory file:

```
pg_password='choose-a-password'
pg_hashed_password='md57d08dde7e95e862eaadfff09565e92e6'
```

  The "awx" implies the Postgres username. If you have an alternative username, provide that rather than "awx".

- The `ssh-keygen` command generates keys in a format (RFC4716) which uses the `md5` digest algorithm at some point in the process (as part of a transformation performed on the input passphrase). On a FIPS-enforcing system, `md5` is completely disabled, so these types of encrypted SSH keys (RFC4716 private keys protected by a passphrase) will not be usable. When FIPS mode is enabled, any encrypted SSH key you import into Ansible Tower **must** be a `PKCS8`-formatted key. Existing `AES128` keys can be converted to `PKCS8` by running the following `openssl` command:

```
$ openssl pkcs8 -topk8 -v2 aes128 -in <INPUT_KEY> -out <NEW_OUTPUT_KEY>
```

  For more details, see: https://access.redhat.com/solutions/1519083

- Use of Ansible features that use the `paramiko` library will not be FIPS compliant. This includes setting `ansible_connection=paramiko` as a transport and using network modules that utilize the `ncclient` NETCONF library.
- The TACACS+ protocol uses `md5` to obfuscate the content of authorization packets; TACACS+ Authentication is not supported for systems where FIPS mode is enabled.
- The RADIUS protocol uses `md5` to encrypt passwords in `Access-Request` queries; RADIUS Authentication is not supported for systems where FIPS mode is enabled.

## 3.3 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.

## 3.4 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.

### 3.4.1 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.

---

### 3.4.2 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

For convenience, those installation instructions are summarized below.

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

## 3.5 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.

### 3.5.1 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
```

### 3.5.2 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

For convenience, those installation instructions are summarized below.

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

## 3.6 Configuration and Installation on OpenShift

For OpenShift-based deployments, refer to OpenShift Deployment and Configuration.

CHAPTER

FOUR

# 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 or any similar subsystem; you must install it as described in the installation instructions in this guide. For OpenShift-based deployments, refer to OpenShift Deployment and Configuration.

Ansible Tower has the following requirements:

- **Supported Operating Systems**:
  - Red Hat Enterprise Linux 7.4 or later 64-bit
  - CentOS 7.4 or later 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** for Tower installations. Refer to the capacity algorithm section of the *Ansible Tower User Guide* for determining the CPU capacity required for the number of forks in your particular configuration.
- **4 GB RAM minimum** for Tower installations
  - 4 GB RAM (minimum and recommended for Vagrant trial installations)
  - 4 GB RAM (minimum for external standalone Postgres databases)
  - For specific RAM needs, refer to the capacity algorithm section of the *Ansible Tower User Guide* for determining capacity required based on the number of forks in your particular configuration
- **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.X 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 m4.large or larger
  - An instance size of m4.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 1 GB of memory per 10 forks + 2GB reservation for Tower, see the capacity algorithm for further details. If `forks` is set to 400, 40 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

### 4.1.1 Postgres Configurations

Optionally, you can configure the Postgres database as separate nodes that are not managed by the Tower installer. When the Tower installer manages the database server, it configures the server with defaults that are generally recommended for most workloads. However, you can adjust these PostgreSQL settings for standalone database server node where `ansible_memtotal_mb` is the total memory size of the database server:

```
max_connections ==1024
shared_buffers == ansible_memtotal_mb*0.3
work_mem == ansible_memtotal_mb*0.03
maintenance_work_mem == ansible_memtotal_mb*0.04
```

Refer to PostgreSQL documentation for more detail on tuning your PostgresSQL server.

## 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.

CHAPTER

FIVE

# 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

For the OpenShift installer, go to http://releases.ansible.com/ansible-tower/setup_openshift

You may install standalone Tower or use the bundled installer:

- if you set up Tower on an environment with a direct Internet access, you can download the standalone Tower installer
- if you set up Tower on an environment without direct access to online repositories, or your environment enforces a proxy, you must use the bundled 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 `3.4.2` or `3.4.0` directory.

## 5.1 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.

## 5.2 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.

### 5.2.1 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.

### 5.2.2 Amazon EC2

To launch the AMI, you must have an AMI ID (which varies based on you 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.

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

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

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

CHAPTER

SIX

# 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. For OpenShift-based deployments, refer to OpenShift Deployment and Configuration.

## 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 **upgrading an existing cluster**: When upgrading a cluster, you may decide that you want to also reconfigure your cluster to omit existing instances or instance groups. Omitting the instance or the instance group from the inventory file will not be enough to remove them from the cluster. In addition to omitting instances or instance groups from the inventory file, you must also deprovision instances or instance groups before starting the upgrade. Otherwise, omitted instances or instance groups will continue to communicate with the cluster, which can cause issues with tower services during the upgrade.
- 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 **upgrading a single node**: 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
```

(continues on next page)

(continued from previous page)

```
[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'
```

(continues on next page)

(continued from previous page)

```
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
```

---

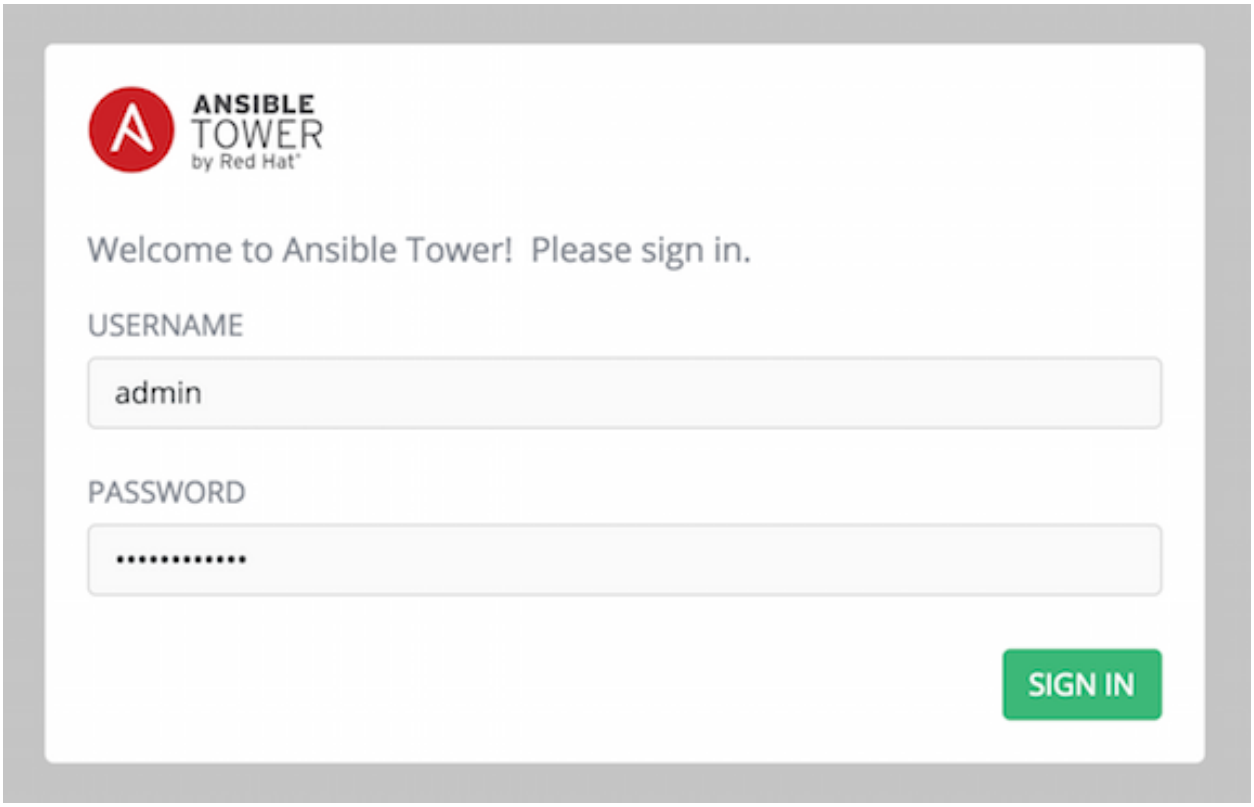
**Note:** Please note that a issue was discovered in Tower 3.0.0 and 3.0.1 that prevented proper system backups and restorations.

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.

CHAPTER

SEVEN

# 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

You may install standalone Tower or use the bundled installer:

- if you set up Tower on an environment with a direct Internet access, you can download the standalone Tower installer
- if you set up Tower on an environment without direct access to online repositories, or your environment enforces a proxy, you must use the bundled 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 `3.4.2` or `3.4.0` 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.

CHAPTER

EIGHT

# 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. 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.

2. 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.

CHAPTER

NINE

# 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.

**Distributed Job** A job that consists of a job template, an inventory, and slice size. When executed, a distributed job slices each inventory into a number of "slice size" chunks, which are then used to run smaller job slices.

**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 serveral 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.

**Instance Group** A group that contains instances for use in a clustered environment. An instance group provides the ability to group instances based on policy.

**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 Slice** See *Distributed Job*.

**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.

**Policy** Policies dictate how instance groups behave and how jobs are executed.

**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.

**Sliced Job** See *Distributed Job*.

**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.).

CHAPTER

TEN

# INDEX

- genindex

CHAPTER

# ELEVEN

# COPYRIGHT © 2019 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.

# INDEX

## Symbols


2.2
    Ansible, 11


## A


active/passive, external database, clustered
    installation multi-machine, 14
Ad Hoc, **24**
Amazon AMI image, 13
analytics collection, 23
Ansible
    2.2, 11
    latest, 11
    prerequisites, 7
    stable, 11
Ansible, configure repository access, 7, 8
Ansible, installation, 8


## B


bundled installer, 12


## C


Callback Plugin, **24**
CentOS, 7
Check Mode, **24**
components
    licenses, 3
configuration
    Postgres, 11
Credentials, **24**
current
    release notes, 4
custom SSL certificates, 5


## D


data collection, 23
database
    Postgres, 11
DEB files
    licenses, 3
Distributed Job, **24**
download Ansible Tower, 12


## E


evaluation, 2
external database
    installation single machine, 14
extra vars
    flags, 5
    installation, 5


## F


Facts, **24**
features, 1
flags
    extra vars, 5
    installation, 5
Forks, **24**


## G


glossary, 24
Group, **24**
Group Vars, **24**


## H


Handlers, **24**
Host, **25**
Host Specifier, **25**
http
    proxy, 5


## I


installation, 14
    extra vars, 5
    flags, 5
    general notes, 5
    multi-machine active/passive, external database, clustered, 14
    platform-specific notes, 6
    scenarios, 14
    single machine external database, 14

    single machine integrated, 14
installation bundle
    licenses, 3
installation prerequisites, 7
installation program, 12
    upgrade, 20
installation requirements, 9
installation script
    inventory file setup, 15
    playbook setup, 18, 21
Instance Group, **25**
integrated
    installation single machine, 14
Inventory, **25**
inventory file setup, 15
Inventory Script, **25**
Inventory Source, **25**

## J

Job, **25**
Job Detail, **25**
Job Slice, **25**
Job Template, **25**
JSON, **25**

## L

latest
    Ansible, 11
license, 1, 2
    nodes, 3
    trial, 2
    types, 2
license features, 1
licenses
    components, 3
    DEB files, 3
    installation bundle, 3
    RPM files, 3

## M

multi-machine
    active/passive, external database, clustered, installation, 14

## N

Notification, **25**
Notifier, **25**
Notify, **25**

## O

Organization, **25**
Organization Administrator, **25**

## P

password, changing, 19
Pendo, 23
PENDO_TRACKING_STATE, 23
Permissions, **25**
platform-specfic notes
    CentOS, 6
    Red Hat Enterprise Linux, 6
Playbook, **25**
playbook setup, 18, 21
    installation script, 18, 21
    setup.sh, 18, 21
Plays, **25**
Policy, **25**
Postgres
    configuration, 11
    database, 11
    tuning, 11
prerequisites, 7
    Ansible, 7
Project, **25**

## R

Red Hat Enterprise Linux, 7, 8
release notes, 4
    current, 4
requirements, 9
Roles, **26**
RPM files
    licenses, 3

## S

Schedule, **26**
self-signed SSL certificate, 5
setup.sh
    playbook setup, 18, 21
single machine
    external database, installation, 14
    integrated, installation, 14
Sliced Job, **26**
stable
    Ansible, 11
Sudo, **26**
Superuser, **26**
support, 1, 2
Survey, **26**

## T

Team, **26**
trial, 2
tuning
    Postgres, 11

## U

Ubuntu, 8
updates, 2
upgrade, 20
    installation program, 20
usability data collection, 23
User, **26**
user data tracking, 23

## V

Vagrant image, 13

## W

Workflow Job Template, **26**

## Y

YAML, **26**