Ansible Tower Installation and Reference Guide

Release Ansible Tower 2.2.0

Ansible, Inc.

Jun 06, 2017

CONTENTS

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

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 2.2.0.

Ansible Tower Version 2.2.0; July 14, 2015; https://access.redhat.com/

CHAPTER

ONE

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 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/

1.2 License Types

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

- **Basic**
  - Manage smaller environments (up to 250 nodes)
  - 30 days of initial web support through the Red Hat Customer Portal at https://access.redhat.com/, for installation and setup
  - Maintenance and upgrades included
- **Enterprise**
  - Manage any size environment
  - Enterprise 8x5 support and SLA (4 hour critical incident response)
  - Phone and web support
  - Support for Ansible core included
  - Maintenance and upgrades included
- **Premium**
  - Manage any size environment, including mission-critical environments
  - Premium 24x7 support and SLA (4 hour critical incident response, 8 hour non-critical incident response)

  - Phone and web support
  - Support for Ansible core included
  - Maintenance and upgrades included

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

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

1.3 License Features

Ansible Tower 2.2 introduces a separation of features for Basic versus Enterprise or Premium licenses. The following list of features are available for all new Enterprise or Premium license users:

- Multi-Organization Support
- Activity Streams
- Surveys
- LDAP Support
- High Availability
- System Tracking (*new to Ansible Tower 2.2*)

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

CHAPTER

TWO

RELEASE NOTES

- Version 2.2
  - Added System Tracking job scan (available for Enterprise and Premium licenses only)
  - Simplified Dashboard and Interface with new Setup Menu
  - Added inventory support for OpenStack
  - Added data cleanup and snapshot retention scheduling
  - Added Ansible Galaxy integration
  - Added support for Remote Command Execution
  - Added Status widget for easily viewing the 10 most recent jobs run on a job template
  - Added integration for easier backups and restorations into the Tower setup playbook
  - Adjusted dates to display in the user's locale format
  - Simplified password/passphrase entry
  - Added more configurable verbosity levels for job templates
  - Assorted other bugfixes and enhancements
  - API change: Formatting of `extra_vars` attached to Job Template records is preserved. Previously, YAML would be converted to JSON and returned as JSON. In 2.2.0 and newer, YAML is returned as YAML with formatting and comments preserved, and JSON is returned as JSON.
- Version 2.1.4
  - Corrected Tower's Live Events feature, again. Really.
- Version 2.1.3
  - Corrected an issue where Tower Live Events would attempt to endlessly reconnect
  - Corrected issues when running with Ansible 1.9.0.1
- Version 2.1.2
  - Corrected multiple issues with Tower's Live Events feature
  - Corrected an issue where Tower would become stuck if a job was killed due to memory exhaustion
  - Improved the response time of Project queries
  - Corrected an error that caused users to be unable to relaunch jobs
- Version 2.1.1
  - Multi-tenancy security enabled by default for new installs

  - Added support for setting VPC id for RDS instances to EC2 dynamic inventory
  - Added the ability for organization admins to create surveys
  - Added support for scheduling of custom inventory scripts
  - Corrected an error when parsing extra_vars as YAML
  - Corrected an error when configuring a remote database
  - Added EULA agreement when updating license
  - Corrected the sending of live events in some cases
  - Corrected a potential XSS issue
- Version 2.1
  - New simplified Portal Mode view for users, access at `https://<Tower server name>/portal/`
  - New surveys on job templates allow easy prompting of users for job parameters
  - Tower can now use an external PostgreSQL instance as the Tower database, including Amazon's RDS
  - Added support for active/passive High Availability Tower deployments
  - Custom dynamic inventory scripts can be pasted in using the admin user menu
  - Limit Amazon EC2 inventory imports into Tower based on tags, keys, and more
  - Tower data cleanup jobs can now be scheduled and run directly from the Tower interface versus logging into the Tower instance
  - The /etc/awx Tower configuration directory has moved to /etc/tower
  - Non-admin api users must now use the /launch endpoint for a job template and can no longer call a job's /start endpoint directly.
  - Many assorted improvements and fixes
- Version 2.0.5
  - Ensured websocket connection uses user's RBAC credentials
  - Corrected a potential CSRF issue when using the REST API graphical browser
- Version 2.0.4
  - Corrected a privilege escalation related to user account levels
- Version 2.0.2
  - Further corrections for job execution with certain 0mq library versions
  - Changes to AMI license logic to allow bring-your-own-license usage
- Version 2.0.1
  - Corrected a job execution issue due to 0mq library versions on certain platforms
  - Reduced logfile verbosity and retention for some Tower subcomponents
  - Adjusted setup playbook for the release of EPEL 7
- Version 2.0
  - New dashboard that provides at-a-glance status of your Ansible deployment
  - Completely redesigned job status page featuring real-time playbook output and progress updates
  - Added support for multiple new cloud providers - Azure, Google Compute Engine, and VMware vSphere

  - New user interface look and feel
  - Integrated monitoring support for checking the health of your Tower install
  - Tower now requires a license to run. 10 machine free licenses, as well as free large trial licenses, are available at http://ansible.com/license
  - Support added for Red Hat Enterprise Linux 7 and CentOS 7
  - Upgrades will reuse password information, not requiring reentry in 'group_vars/all' of setup playbook
  - Many assorted improvements and fixes
- Version 1.4.12
  - Corrected an issue handling Unicode output from ansible-playbook
  - Corrected an issue displaying job details for some jobs
- Version 1.4.11
  - Performance improvements to inventory import and deletion
    * Groups UI under inventory tab is now paginated
    * Updated UI options for moving and copying groups (and host contents)
  - Added the ability to optionally prompt for job variables when launching jobs to the job template detail pages
- Version 1.4.10
  - Correctly handle schedule creation when browser timezone cannot be detected.
  - Corrected pagination on job_events page.
- Version 1.4.9
  - Corrected a provisioning callback issue on Enterprise Linux.
  - Added a sample provisioning callback script.
  - Various backend and UI improvements.
- Version 1.4.8
  - Scheduling for Jobs, SCM updates, and Inventory synchronization has been added. The UI for each of these objects has changed to accommodate this new scheduling feature.
    * The jobs page has been overhauled to show completed, active, queued, and scheduled jobs.
    * Inventory and project synchronization jobs are now also shown on the jobs page.
  - Added support for Ansible Vault to Credentials. For more information on how to use Ansible Vault, please visit: http://docs.ansible.com/playbooks_vault.html.

CHAPTER

THREE

KNOWN ISSUES

3.1 Installation failure related to MongoDB

During the releases of Tower 2.x, the upstream URL for the MongoDB repository changed. This change in the upstream URL causes installations to fail.

The Ansible Tower 2.4.5 release has been corrected with the new upstream URL for MongoDB. For those that require MongoDB, use the latest Tower 2.4.5 release. If MongoDB is not a requirement, consider upgrading to the latest version of Ansible Tower.

3.2 Errors when editing objects

Ansible Tower implements a role based access control system. It may appear that you can edit objects that do not belong to you–like being able to pull up an edit dialog of your teammates, which you have been granted permission to view–but when you try to edit something, you will receive a 403 error and you cannot view any information you should not already have access to as defined in the system.

3.3 Host comparisons against a single host

When performing a host comparison against a single host, if there is only one (1) run for the selected date, Tower may display a message saying it could not find any scan job runs in one of the columns.

3.4 Host comparisons against a two hosts

When performing a host comparison against two hosts, you can only select from a single page of hosts.

3.5 Live events status indicators

Live events status dots are either seen as a red or orange dot at the top of the Tower Dashboard when something goes wrong, or they are not seen at all to indicate a healthy system state. If you encounter a red or orange live events status indicator, even when your system seems fine, the following suggestions may offer a solution:

- Try manually refreshing/reloading your browser page.
- Try changing web browsers, as Firefox and Safari have been reported to have issues trusting self-signed certificates.

- Try creating a self-signed certificate that matches your DNS and import it into your trust manually.
- Try using an incognito or private browsing session.
- Try disabling your browser plugins to ensure none are blocking the service.

Live event status dots are used for troubleshooting problems with your Tower instance and the socketio logs can point out anything problematic. You can collect troubleshooting help by running a sosreport. As root, run the command `sosreport` from your system to automatically generate a diagnostic tar file, then contact Ansible's Support team with the collected information for further assistance.

Note: Starting with Ansible Tower 2.2.0, live events status indicators only appear if Tower detects a problem. In earlier releases, a green status dot was shown to indicate a healthy system.

3.6 Playbooks missing access to necessary data due to PRoot issues

When running a playbook that reads and writes information in certain prohibited directories, users may encounter issues with PRoot. PRoot runs the `ansible-playbook` command within a `chroot` jail. In cases like these, the running playbook cannot see other playbooks or sensitive data on disk and should the playbook expect to have access to that information, problems will occur. To fine tune your usage of PRoot, there are certain variables that can be set:

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

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

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

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

3.7 Slower than expected performance

Slow performance issues have been reported against Ansible Tower 2.2.0. You may experience slower than expected system performance if:

- You have a large number of job templates, users, credentials, permissions, projects, and/or inventories. If you find that loading the job template page for normal users is unacceptably slow, run the "Cleanup Deleted Data" management job with `days = 0`. Refer to Management Jobs in the *Ansible Tower Administration Guide* for more information.
- You have a job that produces a lot of playbook output (several megabytes or more). Performance can often be improved by switching to "Normal" verbosity, assuming that your Job Template is set to a higher verbosity level.

CHAPTER

FOUR

GENERAL 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/awx.cert` and `/etc/tower/awx.key` for HTTPS communication. These can be replaced after install with your own certificates if you desire, but the filenames are required to be the same.
- If using Ansible 1.8 or later, ensure that fact caching using Redis is not enabled in `ansible.cfg` on the Tower machine.

CHAPTER

FIVE

SUPPORTED PLATFORMS AND REQUIREMENTS

As you prepare for the Tower installation, keep in mind any *Requirements* and *Prerequisites* that may be necessary.

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, please contact Ansible via the Red Hat Customer Portal at https://access.redhat.com/.

CHAPTER

SIX

PLATFORM-SPECIFIC NOTES

6.1 Red Hat Enterprise Linux and CentOS

- 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.

CHAPTER

SEVEN

REQUIREMENTS

Ansible Tower has the following 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 below.

- 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
- The latest stable release of Ansible
- 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 hard disk
- 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 HA MongoDB setups:
  - If you plan to run MongoDB, the following guidelines provide a rough estimate for the amount of space required. The basic calculation is:

```
(number of hosts in inventory) * (number of scans) *
(average module fact size) * (number of modules scanning)
```

- For example:

  hosts = 1,000

  number of scans = 365 (1 scan per day for a year)

  average module fact size = 100 kb

  number of modules = 4 (ansible, packages, services, files)

  = 146 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: Tower depends on Ansible playbooks and requires the installation of the latest stable version of Ansible before installing Tower.

Use the latest stable release of Ansible for best performance and to ensure the latest bug fixes are available. Versions of Ansible 1.8 or later are supported for Ansible Tower 2.2.

Detailed instructions on installing Ansible are available at: http://docs.ansible.com/intro_installation.html

---

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 as 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 via the Red Hat Customer Portal at https://access.redhat.com/.

Note: It is strongly recommended to use the latest stable release of Ansible for best performance and to ensure the latest bugfixes are available. However, any version of Ansible 1.8 or later is supported for Ansible Tower 2.2.

The requirements for systems managed by Tower are the same as for Ansible at: http://docs.ansible.com/intro_getting_started.html

CHAPTER

EIGHT

PREREQUISITES

Tower is installed using Ansible playbooks. Therefore, you need Ansible to install Tower. Ansible Tower requires Ansible version 1.8 or later.

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 here:

8.1 Configuration and Installation for Ansible with Red Hat Enterprise Linux and CentOS

The following steps help you configure access to the repository as well as install Ansible.

8.1.1 Configure access to the repository for Ansible with Red Hat Enterprise Linux and CentOS (version 6 or later):

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://download.fedoraproject.org/pub/epel/6/x86_64/
↪epel-release-6-8.noarch.rpm
```

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

```
root@localhost:~$ yum install http://download.fedoraproject.org/pub/epel/7/x86_64/e/
↪epel-release-7-5.noarch.rpm
```

Note: You may also need to enable the "extras" repository, named "extras" on CentOS 7, "rhel-7-server-extras-rpms" on Red Hat Enterprise Linux 7, and "rhui-REGION-rhel-server-extras" when running in EC2.

Note: For users of Red Hat Enterprise Linux 7, you also need to enable the "optional" repository. When using the official Red Hat Enterprise Linux 7 marketplace AMI, be sure you install the latest "rh-amazon-rhui-client" package that allows enabling the optional repository (named "rhui-REGION-rhel-server-optional" in EC2).

8.1.2 Install Ansible on Red Hat Enterprise Linux and CentOS (version 6 or later):

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

8.2 Configuration and Installation for Ansible with Ubuntu

The following steps help you configure access to the repository as well as install Ansible.

8.2.1 Configure access to the repository for Ansible with Ubuntu 12.04 and Ubuntu 14.04:

As the root user, configure Ansible PPA:

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

8.2.2 Install Ansible on Ubuntu 12.04 and Ubuntu 14.04:

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

CHAPTER

NINE

TOWER INSTALLATION SCENARIOS

Tower can be installed in three scenarios.

- Single Machine 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.

- Single Machine with an external database

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 will not configure replication or failover for the database that it uses, although Tower should work with any replication that you have.

Note: 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 with an external database

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

Note: Running in a high-availability setup requires any database that Tower uses to be external–Postgres and MongoDB must be installed on a machine that is not one of the primary or secondary tower nodes. When in High Availability mode, the remote Postgres and MongoDB version requirements are *Postgresql 9.4.x* and *mongodb 3.0.x*.

Each of these scenarios can be configured through the Tower Installation Wizard.

CHAPTER

TEN

GET THE TOWER INSTALLATION PROGRAM

Using the link provided in the email you received based on your interest in Ansible Tower, download and extract the Tower installation tarball.

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: http://releases.ansible.com/awx/setup/ansible-tower-setup-latest.tar.gz

Once extracted, `cd` into the setup directory using a command line console. In the following commands, replace the string `VERSION` with the version of Tower that you are installing (e.g., "2.1.4").

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

CHAPTER

ELEVEN

THE TOWER INSTALLATION WIZARD

The Tower setup process consists of two parts–an installation wizard that determines your Tower configuration and a setup playbook that uses that information to install Tower.

The Tower Installation Wizard and the Tower setup playbook do not need to be run from the system that will run Tower, although they can. The Tower Installation Wizard asks for credentials needed to access external systems where necessary.

The Tower Installation Wizard is invoked as `configure` from the path where you unpacked the Tower installation tarball. It writes a file called `tower_setup_conf.yml` which contains the configuration for Tower.

```
username@localhost:~$ ./configure
```

11.1 Installation Arguments

The wizard takes the following arguments:

- `-h`, `--help` Displays a brief usage summary.
- `-l`, `--local` Assumes that you are installing Tower on the local machine where you are running `configure`. This implies an internal embedded PostgreSQL database as well. This option skips some questions in the wizard.
- `--no-secondary-prompt` Assumes you are not installing in a high-availability setup. This option skips some questions in the wizard.
- `-A`, `--no-autogenerate` Do not autogenerate random passwords for PostgreSQL or Redis–prompt the user for them instead.
- `-o FILE`, `--options-file=FILE` Use the file `FILE` as a source of answers. This can be the `tower_setup_conf.yml` file from a previous run of the wizard. Depending on the contents of the file, this option skips some questions in the wizard. The contents of the `tower_setup_conf.yml` file may look similar to the following:

```
admin_password: password
database: internal
munin_password: password
pg_password: cmTM4eaCpnDS54ReXGv34szoHQiXccFno3atfBij
primary_machine: localhost
redis_password: 8nG2TRpSDnpr69eWbqFwXTbryCUW64r76VjBqsKx
```

Once you invoke the Tower Installation Wizard, you are asked about the configuration of a few different items.

11.2 Primary Tower machine configuration

First, the Tower wizard asks about where you intend to place the primary (or only) Tower instance.

```
root@localhost:~$ ./configure
-----------------------------------------
Welcome to the Ansible Tower Install Wizard
-----------------------------------------

PRIMARY TOWER MACHINE
Tower can be installed (or upgraded) on this machine, or onto a remote machine
that is reachable by SSH.

Note: If using the High Availability features of Tower, you must use DNS
resolvable hostnames or IP addresses (do not use "localhost").

Enter the hostname or IP to configure Ansible Tower
(default: localhost):
```

If you are installing on the current machine, enter `localhost` or `127.0.0.1` for the current machine. If you are installing on a different machine, enter the IP address or hostname of the machine. This machine must be running and accessible via SSH when running the setup playbook later.

11.2.1 Configuring the Database

Tower can be setup as an internal database installed on the primary Tower machine or as an external PostgreSQL database.

Enter `i` for an internal database on the same machine as Tower, or `e` for an external database. To run Tower in a high-availability configuration, you must use an external database.

```
DATABASE
Tower can use an internal database installed on the Tower machine, or an
external PostgreSQL database. An external database could be a hosted database,
such as Amazon's RDS.

An internal database is fine for most situations. However, to use the High
Availability features of Tower, an external database is required.

If using an external database, the database (but not the necessary tables) must
already exist.

Will this installation use an (i)nternal or (e)xternal database?
```

If you choose to use an external database, the wizard prompts you for the following additional database parameters:

- Database host to connect to
- Database name
- PostgreSQL user to use to access the database
- Password for the above PostgreSQL user
- Port to connect to the PostgreSQL database on (hit enter for the default PostgreSQL port)

The wizard will attempt to verify these parameters if your system has the PostgreSQL client libraries installed.

11.3 Secondary Installation (if applicable)

At this time, if you chose to setup an external database, you can configure any neccessary secondary Tower instances.

```
SECONDARY MACHINES
You may optionally elect to add any number of secondary machines, on which
Ansible Tower will also be installed (in secondary mode).
Add secondary machines (y/n)?
```

Enter `y` to configure additional secondary Tower instances.

Enter the hostnames or IP addresses of machines you want to configure as secondary Tower instances, one at a time. Enter a blank line to end the list. These machines must be running and accessible via SSH when running the setup playbook later.

11.4 Passwords

You are then prompted for the passwords you need for various Tower services.

```
PASSWORDS
For security reasons, since this is a new install, you must specify the
following application passwords.
```

The installation wizard prompts you to provide an **Admin Password**. This 'admin' password is used for the first user (and superuser) created upon installation. You must have this password for your initial login to Tower.

If you passed the `-A` or `--no-autogenerate` parameters to the Installation Wizard, you are prompted for a PostgreSQL password and a Redis password. These are used internally to Tower and are not needed by the admin at runtime. These additional passwords are normally auto-generated as a random value.

11.4.1 Password Strength Configuration

This feature allows you to define minimum strengths for passwords. Password strength configuration for Tower specifies policies and security mechanisms for providing rules to specify user passwords.

To make use of this feature, set your preferred password strength configuration options in the `local_config.js` file.

- Copy `config.js` in its entirety to the file `local_config.js`.
- Set the password_* variables to `true`/`false` based on the rules you want to enforce.

The password strength configuration feature allows you to create passwords of any combination of upper and lowercase characters, numbers, and special characters that include "!", "@", "#", "$", "%","^", "&", "*", "(", and ")".

You can also set the password's minimum (1) and maximum (64) length. The recommended minimum password length is eight (8) characters.

11.5 Connection Information

If you chose to install on machines other than the current machine you are running the installation wizard on, you must also enter details on how to connect to those machines.

```
CONNECTION INFORMATION
Enter the SSH user to connect with (default: root):
```

First, you are prompted for the user to SSH to the remote hosts with. If this user is not root, you are prompted for how you will escalate privileges.

```
Root access is required to install Tower.
Will you use (1) sudo or (2) su?
```

Choose either `1` or `2` to configure sudo or su access. If you enter that you need a password for sudo or su access, this will be prompted for during the setup playbook run.

You are then prompted for SSH key information. If you are using a SSH key to access this host, you are prompted for the path to the SSH private key to use.

The same connection and su/sudo information will be used for all machines that are configured by the setup playbook, whether primary or secondary nodes. If you need different connection information for different machines, this can be configured by manually modifying the `inventory` file generated by the Installation Wizard.

11.6 Review and Confirm

You are then asked to review the settings you entered. An example would be:

```
REVIEW
You selected the following options:

The primary Tower machine is: tower.example.com
Tower will operate on an EXTERNAL database.
  host: database.example.com
  database: mydb
  user: db_admin
  password: ********
  port: 5432
Additional secondary machines:
  - tower-backup.example.com
  - tower-backup2.example.com
Using SSH user: jdoe

Are these settings correct (y/n)?
```

Select `y`, and you will then be given some information on running the setup playbook.

```
FINISHED!
You have completed the setup wizard. You may execute the installation of
Ansible Tower by issuing the following command:

# Add your SSH key to SSH agent.
# You may be asked to enter your SSH unlock key password to do this.
ssh-agent bash
ssh-add ~/.ssh/id_my-example-key
./setup.sh -s
```

11.7 Reviewing the Tower Configuration

The Tower configuration is written into two files by the Tower Installation Wizard.

- `tower_setup_conf.yml` This Tower configuration file contains needed Tower passwords, database connection information, and machine connection information.
- `inventory` This includes the machines that the setup playbook will operate on, grouped into the `primary` and `secondary` groups of nodes.

11.8 The Setup Playbook

The Tower setup playbook is invoked as `setup.sh` from the path where you unpacked the Tower installer tarball. It uses the `tower_setup_conf.yml` and `inventory` files written by the Tower Installation Wizard. The setup script takes the following arguments:

- `-h`, `--help` Displays a brief usage summary.
- `-c FILE` Use the specified FILE as the Tower configuration file rather than `tower_setup_conf.yml` in the current directory.
- `-i FILE` Use the specified FILE as the inventory for the setup playbook rather than `inventory` in the current directory.
- `-p` Set ansible to prompt for a SSH password when connecting to remote machines
- `-s` Set ansible to prompt for a sudo password on remote machines when installing Tower.
- `-u` Set ansible to prompt for a su password on remote machines when installing Tower.
- `-e` Set additional ansible variables for the playbook to use either in key=value or YAML/JSON form. This should not be needed in normal operation.
- `-b` Perform a database backup in lieu of installing.
- `-r BACKUP_FILE` Perform a database restore in lieu of installing.

Depending on the configuration you entered when running the Tower Installation Wizard, it may have prompted you to run the setup playbook with some combination of `-p`, `-s`, or `-u`.

After calling `setup.sh` with the appropriate parameters, Tower is installed on the appropriate machines as has been configured.

CHAPTER

TWELVE

UPGRADING AN EXISTING TOWER INSTALLATION

You can upgrade your existing Tower installation to the latest version easily.

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 High Availability installation as part of an upgrade. Users who want to deploy Tower in a High Availability configuration should back up their Tower database, install a new HA 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 High Availability chapter of the *Tower Administration Guide*.

12.1 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.

12.2 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*.

12.3 Get the Tower Installer

Download the Ansible Tower install/upgrade tool: http://releases.ansible.com/ansible-tower/setup/

Extract it, then `cd` into the setup directory. Replace the string `VERSION` in the commands below with the version of Tower that you are installing e.g., "2.1.4".

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

12.4 Run the Tower Installation Wizard

To configure your upgrade, you run the same Tower Installation Wizard as you would for an installation.

12.4.1 Simplified Tower Upgrade

If you are upgrading a Tower instance on a local machine with an internal database, you can bypass many of the questions by invoking the *configure* script as `./configure --local`

```
root@localhost:~$ ./configure --local
-----------------------------------------
Welcome to the Ansible Tower Install Wizard
-----------------------------------------

This wizard will guide you through the setup process.

LOCAL INSTALLATION
You are installing Ansible Tower on this machine, using an internal database.

REVIEW
You are UPGRADING an existing Tower installation on localhost.


Are these settings correct (y/n)?


Confirm that the settings are correct.
```

12.4.2 Upgrade Using an existing Settings File

If you have the `tower_setup_conf.yml` file from when you installed Tower, you can pass it to the `configure` script:

```
root@localhost:~$ ./configure -o tower_setup_conf.yml
-----------------------------------------
Welcome to the Ansible Tower Install Wizard
-----------------------------------------

This wizard will guide you through the setup process.

The configuration provided in /home/tower/ansible-tower-setup-2.1.4/tower_setup_conf.
↪yml appears complete.

FINISHED!
You have completed the setup wizard. You may execute the installation of
Ansible Tower by issuing the following command:

sudo ./setup.sh
```

12.4.3 Upgrade Interactively

Alternatively, you can walk through the upgrade process. Invoke the `configure` script:

```
root@localhost:~$ ./configure
----------------------------------------
----------------------------------------
Welcome to the Ansible Tower Install Wizard
----------------------------------------

This wizard will guide you through the setup process.

PRIMARY TOWER MACHINE
Tower can be installed (or upgraded) on this machine, or onto a remote machine
that is reachable by SSH.

Note: If using the High Availability features of Tower, you must use DNS
resolvable hostnames or IP addresses (do not use "localhost").

Enter the hostname or IP to configure Ansible Tower
(default: localhost):
```

Once you enter the host, the Tower Installation Wizard contacts it to determine if Tower is installed. If it is a functioning Tower installation, it determines the current Tower configuration, including the location of any secondary nodes that need upgrading.

The Installation Wizard then asks you for connection details for connecting to any remote machines. Enter any needed SSH user, SSH key location, and whether `sudo` or `su` is in use.

12.5 The Setup Playbook

The Tower setup playbook script is invoked as `./setup.sh` from the path where you unpacked the Tower installer tarball. It uses the `tower_setup_conf.yml` and `inventory` files written by the Tower Installation Wizard. The setup script takes the following arguments:

- `-h`, `--help` Displays a brief usage summary.
- `-c FILE` Use the specified FILE as the Tower configuration file rather than `tower_setup_conf.yml` in the current directory.
- `-i FILE` Use the specified FILE as the inventory for the setup playbook rather than `inventory` in the current directory.
- `-p` Set ansible to prompt for a SSH password when connecting to remote machines
- `-s` Set ansible to prompt for a sudo password on remote machines when upgrading Tower.
- `-u` Set ansible to prompt for a su password on remote machines when upgrading Tower.
- `-e` Set additional ansible variables for the playbook to use either in key=value or YAML/JSON form. This should not be needed in normal operation.
- `-b` Perform a database backup in lieu of installing.
- `-r BACKUP_FILE` Perform a database restore in lieu of installing.

Depending on the configuration you entered when running the Tower Installation Wizard, it may have prompted you to run the setup playbook with some combination of `-p`, `-s`, or `-u`.

As the root user, call `setup.sh` with the appropriate parameters and Tower is upgraded on the appropriate machines as configured.

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

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 at http://support.ansible.com/.

CHAPTER

THIRTEEN

SUPPORTED LOCALES

Ansible Tower supports the following locales for UTC-friendly date and time information.

Tower automatically sets the locale preference based on the user's browser settings. For Safari, Internet Explorer, and older versions of Chrome as well as FireFox, this is handled automatically.

For newer versions of Chrome (v32 and later) and FireFox (v32 and later), Tower uses the language preferences set from your browser's language settings. The browser lists the user's preferred languages and selects the first in the array as the user's top choice, which Tower uses as the preferred locale. This means that you can change your browser's language and change your Tower locale preferences (although you may need to reload/refresh Tower in your browser to see this change.)

- `az` – Cyrillic
- `bg` – Bulgarian
- `bs` – Bosnian
- `ca` – Catalan
- `cs` – Czech
- `da` – Danish
- `de` – German
- `el` – Greek
- `en-gb` – English (United Kingdom)
- `es` – Spanish
- `et` – Estonian
- `eu` – Basque
- `fa` – Persian
- `fi` – Finnish
- `fo` – Faroese
- `fr` – French
- `gl` – Galician
- `he` – Hebrew
- `hr` – Croatian
- `hu` – Hungarian
- `id` – Indonesian

- `is` – Icelandic
- `it` – Italian
- `ja` – Japanese
- `ka` – Georgian
- `lt` – Lithuanian
- `lv` – Latvian
- `mk` – Macedonian
- `nb` – Norwegian
- `nl` – Dutch
- `pl` – Polish
- `pt-br` – Portuguese (Brazil)
- `pt` – Portuguese
- `ro` – Romanian
- `ru` – Russian
- `sk` – Slovak
- `sl` – Slovenian
- `sq` – Albanian
- `sr` – Serbian
- `sv` – Swedish
- `th` – Thai
- `tr` – Turkish
- `uk` – Ukrainian
- `vi` – Vietnamese
- `zh-cn` – Chinese (simplified)
- `zh-tw` – Chinese (traditional)

CHAPTER

FOURTEEN

TOWER COMPONENT LICENSES

The following components have been integrated into Ansible, Inc.'s distribution of Ansible Tower. 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.

Component	License	Upstream Source
babel	BSD	http://babel.pocoo.org/
IPy	BSD	https://github.com/autocracy/python-ipy
Markdown	BSD	http://packages.python.org/Markdown/
PrettyTable	BSD	http://code.google.com/p/prettytable/
South	Apache 2.0	http://south.aeracode.org/
amqp	LGPL 2.1	http://github.com/celery/py-amqp
angular	MIT	https://github.com/angular/angular.js.git
angular-animate	MIT	https://github.com/angular/angular.js.git
angular-codemirror	MIT	https://github.com/chouseknecht/angular-codemirror
angular-cookies	MIT	https://github.com/angular/angular.js.git
angular-filters	Apache 2.0	https://github.com/frapontillo/angular-filters
angular-md5	MIT	https://github.com/gdi2290/angular-md5.git
angular-mocks	MIT	https://github.com/angular/angular.js.git
angular-moment	MIT	http://github.com/urish/angular-moment
angular-resource	MIT	https://github.com/angular/angular.js.git
angular-route	MIT	https://github.com/angular/angular.js.git
angular-sanitize	MIT	https://github.com/angular/angular.js.git
angular-scheduler	MIT	https://github.com/chouseknecht/angular-scheduler
angular-tz-extensions	MIT	https://github.com/chouseknecht/angular-tz-extensions
ansiconv	MIT	https://bitbucket.org/dhrrgn/ansiconv
anyjson	BSD	http://bitbucket.org/runeh/anyjson/
apache-libcloud	Apache 2.0	http://libcloud.apache.org/
argparse	PSF	http://code.google.com/p/argparse/
azure	Apache 2.0	https://github.com/WindowsAzure/azure-sdk-for-python
billiard	BSD	http://github.com/celery/billiard
bootstrap	MIT	https://github.com/twbs/bootstrap.git
bootstrap-datepicker	Apache 2.0	https://github.com/eternicode/bootstrap-datepicker
boto	MIT	https://github.com/boto/boto/
celery	BSD	http://celeryproject.org
codemirror	MIT	https://github.com/codemirror/CodeMirror.git
components-font-awesome	SIL, MIT*	http://fortawesome.github.io/Font-Awesome/

Continue

Table 14.1 – continued from previous page

Component	License	Upstream Source
d2to1	BSD	http://pypi.python.org/pypi/d2to1
d3	BSD	https://github.com/mbostock/d3.git
distribute	PSF or ZPL	http://packages.python.org/distribute
django-auth-ldap	BSD	http://bitbucket.org/psagers/django-auth-ldap/
django-celery	BSD	http://celeryproject.org
django-crum	BSD	https://projects.ninemoreminutes.com/projects/django-crum/
django-extensions	MIT	http://github.com/django-extensions/django-extensions
django-jsonfield	BSD	http://bitbucket.org/schinckel/django-jsonfield/
django-qsstats-magic	MIT	http://bitbucket.org/kmike/django-qsstats-magic/
django-rest-framework-mongoengine	MIT	https://github.com/umutbozkurt/django-rest-framework-mongoe
django-split-settings	BSD	http://github.com/2general/django-split-settings
django-taggit	BSD	http://github.com/alex/django-taggit/tree/master
django_polymorphic	BSD	https://github.com/chrisglass/django_polymorphic
djangorestframework	BSD	http://www.django-rest-framework.org
dogpile.cache	BSD	http://bitbucket.org/zzzeek/dogpile.cache
dogpile.core	BSD	http://bitbucket.org/zzzeek/dogpile.core
ember-cli-test-loader	MIT	https://github.com/rjackson/ember-cli-test-loader
gevent-socketio	BSD	https://github.com/abourget/gevent-socketio
gevent-websocket	Apache 2.0	https://bitbucket.org/Jeffrey/gevent-websocket
httplib2	MIT	https://github.com/jcgregorio/httplib2
importlib	PSF	https://pypi.python.org/pypi/importlib
iso8601	MIT	https://bitbucket.org/micktwomey/pyiso8601
isodate	BSD	http://cheeseshop.python.org/pypi/isodate
jQuery.dotdotdot	MIT, GPL**	https://github.com/BeSite/jQuery.dotdotdot
javascript-detect-element-resize	MIT	https://github.com/sdecima/javascript-detect-element-resize
jquery	MIT	https://github.com/jquery/jquery
jquery-ui	MIT	http://jqueryui.com/
jqueryui	MIT	http://jqueryui.com/
js-yaml	MIT	https://github.com/nodeca/js-yaml
jsonlint	MIT	https://github.com/zaach/jsonlint.git
kapusta-jquery.sparkline,	BSD	http://omnipotent.net/jquery.sparkline/
keyring	PSF, MIT	http://bitbucket.org/kang/python-keyring-lib
kombu	BSD	http://kombu.readthedocs.org
loader.js	MIT	https://github.com/stefanpenner/loader.js
lodash	MIT	https://github.com/lodash/lodash
lrInfiniteScroll	MIT	https://github.com/lorenzofox3/lrInfiniteScroll
mock	BSD	http://www.voidspace.org.uk/python/mock/
moment	MIT	http://momentjs.com/
mongoengine	MIT	http://mongoengine.org/
netaddr	BSD	https://github.com/drkjam/netaddr/
nvd3	Apache 2.0	http://www.nvd3.org/
ordereddict	MIT	https://pypi.python.org/pypi/ordereddict
os-client-config	Apache 2.0	http://www.openstack.org/
os_diskconfig_python_novaclient_ext	Apache 2.0	https://github.com/rackerlabs/os_diskconfig_python_novaclient_
os_networksv2_python_novaclient_ext	Apache 2.0	https://github.com/rackerlabs/os_networksv2_python_novaclien
os_virtual_interfacesv2_python_novaclient_ext	Apache 2.0	https://github.com/rackerlabs/os_virtual_interfacesv2_ext
oslo.config	Apache 2.0	https://launchpad.net/oslo
oslo.i18n	Apache 2.0	http://launchpad.net/oslo
oslo.serialization	Apache 2.0	http://launchpad.net/oslo

Continue

Table 14.1 – continued from previous page

Component	License	Upstream Source
oslo.utils	Apache 2.0	http://launchpad.net/oslo
pbr	Apache 2.0	http://pypi.python.org/pypi/pbr
pexpect	ISC	http://pexpect.readthedocs.org/
pip	MIT	http://www.pip-installer.org
psphere	Apache 2.0	https://github.com/jkinred/psphere
pyrax	Apache 2.0	https://github.com/rackspace/pyrax
python-cinderclient	Apache 2.0	http://www.openstack.org/
python-dateutil	BSD	https://dateutil.readthedocs.org
python-glanceclient	Apache 2.0	http://www.openstack.org/
python-ironicclient	Apache 2.0	http://www.openstack.org/
python-keystoneclient	Apache 2.0	http://www.openstack.org/
python-neutronclient	Apache 2.0	http://www.openstack.org/
python-novaclient	Apache 2.0	https://git.openstack.org/cgit/openstack/python-novaclient
python-swiftclient	Apache 2.0	http://www.openstack.org/
python-troveclient	Apache 2.0	http://www.openstack.org/
pytz	MIT	http://pythonhosted.org/pytz
pywinrm	MIT	http://github.com/diyan/pywinrm/
rackspace-auth-openstack	Apache 2.0	https://github.com/rackerlabs/rackspace-auth-openstack
rackspace-novaclient	Apache 2.0	https://github.com/rackerlabs/rackspace-novaclient
rax_default_network_flags_python_novaclient_ext	Apache 2.0	https://github.com/rackerlabs/rax_default_network_flags_python
rax_scheduled_images_python_novaclient_ext	Apache 2.0	https://github.com/rackspace-titan/rax_scheduled_images_pytho
redis	MIT	http://github.com/andymccurdy/redis-py
requests	Apache 2.0	http://python-requests.org
rrule	BSD	https://github.com/jakubroztocil/rrule.git
scrollto	MIT	https://github.com/balupton/jquery-scrollto
select2	MIT	https://github.com/chrisjbaik/select2.git
setuptools	PSF or ZPL	https://bitbucket.org/pypa/setuptools
shade	Apache 2.0	http://ci.openstack.org/
simplejson	MIT	http://github.com/simplejson/simplejson
six	MIT	http://pypi.python.org/pypi/six/
sizzle	MIT	https://github.com/jquery/sizzle.git
socket.io-client	MIT	https://github.com/Automattic/socket.io-client.git
stevedore	Apache 2.0	https://github.com/dreamhost/stevedore
suds	LGPL 3	https://fedorahosted.org/suds
timezone-js	Apache 2.0	https://github.com/mde/timezone-js.git
twitter	MIT	http://getbootstrap.com
underscore	MIT	https://github.com/jashkenas/underscore
xmltodict	MIT	https://github.com/martinblech/xmltodict

- *SIL Open Font License and MIT
- **MIT and GPL (Ansible licenses via MIT)

CHAPTER

FIFTEEN

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.

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 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.

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.

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.).

CHAPTER

SIXTEEN

INDEX

- genindex

This document is Copyright © 2015 Ansible, Inc. All rights reserved.

Ansible and Ansible Tower are trademarks of Ansible, Inc.

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

Third Party Rights

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

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

M

O

P

R

S

U