Ansible Tower Quick Install

Release Ansible Tower 3.8.0

Red Hat, Inc.

Feb 11, 2023

CONTENTS

Thank you for your interest in 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 Automation Platform Quick Installation Guide* covers basic installation instructions for installing Ansible Tower on Red Hat Enterprise Linux and CentOS systems. This document has been updated to include information for the latest release of Ansible Tower v3.8.0.

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 3.8.0; November 18, 2020; https://access.redhat.com/

CHAPTER

ONE

PREPARING FOR THE ANSIBLE AUTOMATION PLATFORM INSTALLATION

This guide helps you get your Ansible Automation Platform installation up and running as quickly as possible.

At the end of the installation, using your web browser, you can access and fully utilize the Automation Platform.

1.1 Installation and Reference Guide

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

You should also review the General Installation Notes before starting the installation.

1.2 Prerequisites and Requirements

For platform information, refer to *Platform-specific Installation Notes*.

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. Also, **DO NOT** change the default `alternative` for Python 3.

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

Ansible Tower has the following requirements:

Starting with Ansible Tower 3.8, you **must** have valid subscriptions attached before installing the Ansible Automation Platform. See Attaching Subscriptions for detail.

- **Supported Operating Systems**:
  - Red Hat Enterprise Linux 8.2 or later 64-bit (x86)
  - Red Hat Enterprise Linux 7.7 or later 64-bit (x86)
  - CentOS 7.7 or later 64-bit (x86)

Note: For Automation Hub, selinux-policy package version greater or equal to `3.13.1-268.el7_9.2` is required. If your setup has `rhel-7-server-rpms` repository enabled, the `_9.2` version will be pulled automatically along with Automation Hub. Also, RHUI subscriptions **cannot** be used to install Automation Hub.

Note: The next major release of Ansible Tower will not support Red Hat Enterprise Linux 7 or CentOS (any version) as an installation platform.

- **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 Automation Platform 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 Automation Platform installations
  - 4 GB RAM (minimum and recommended for Vagrant trial installations)
  - 4 GB RAM (minimum for external standalone PostgreSQL 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 10 required to run Ansible Tower 3.7 and later. Backup and restore will *only* work on PostgreSQL versions supported by your current Ansible Tower version.
- **Ansible version 2.9 required** to run Ansible Tower versions 3.8 and later

Note: You cannot use versions of PostgreSQL and Ansible older than those stated above and be able to run Ansible Tower 3.7 and later. Both are installed by the install script if they aren't already present.

- **For Automation Hub**: Starting with Ansible Tower 3.8, Automation Hub will act as a content provider for Ansible Tower, which requires both an Ansible Tower deployment and an Automation Hub deployment running alongside each other. Tower and Automation Hub can run on either RHEL 7 or 8, but only Tower (not Automation Hub) is supported on an OpenShift Container Platform (OCP).

- **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

1.2.1 Additional Notes on Automation Platform Requirements

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 Ansible Automation Platform are the same as for Ansible at: http://docs.ansible.com/intro_getting_started.html

Notable PostgreSQL Changes

Automation Platform uses PostgreSQL 10, which is an SCL package on RHEL 7 and an app stream on RHEL8. Some changes worth noting when upgrading to PostgreSQL 10 are:

- PostgreSQL user passwords will now be hashed with SCRAM-SHA-256 secure hashing algorithm before storing in the database.
- You will no longer need to provide a `pg_hashed_password` in your inventory file at the time of installation because PostgreSQL 10 can now store the user's password more securely. If users supply a password in the inventory file for the installer (`pg_password`), that password will be SCRAM-SHA-256 hashed by PostgreSQL as part of the installation process. **DO NOT** use special characters in `pg_password` as it may cause the setup to fail.
- Since Ansible Tower and Automation Hub are using a Software Collections version of PostgreSQL in 3.8, the *rh-postgresql10* scl must be enabled in order to access the database. Administrators can use the `awx-manage dbshell` command, which will automatically enable the PostgreSQL SCL.
- If you just need to determine if your Tower instance has access to the database, you can do so with the command, `awx-manage check_db`.

PostgreSQL Configurations

Optionally, you can configure the PostgreSQL database as separate nodes that are not managed by the Automation Platform installer. When the Automation Platform 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 PostgreSQL server.

1.2.2 Ansible Software Requirements

While Automation Platform 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.

Upon new installations, Tower installs the latest release package of Ansible 2.9.

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

If you choose to install Ansible on your own, the Automation Platform 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 Automation Platform to work properly. Ansible version 2.9 is required for Ansible Tower versions 3.8 and later.

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

1.2.3 Platform-specific Installation Notes

Installing Automation Platform on Systems with FIPS Mode Enabled

Ansible Automation Platform 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. To run the Tower installer from a FIPS-enabled system, specify `pg_password` in your inventory file. **DO NOT** use special characters in `pg_password` as it may cause the setup to fail:

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

  For further detail, see *Setting up the Inventory File*.

  If you supply a password in the inventory file for the installer (`pg_password`), that password will be SCRAM-SHA-256 hashed by PostgreSQL as part of the installation process.

- 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.

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 Ansible Automation Platform Installer* for more information.
- When installing Ansible Tower, you only need to run `setup.sh`, any repositories needed by Tower are installed automatically.
- The latest version of Ansible is installed automatically during the setup process. No additional installation or configuration is required.

Notes for Ubuntu setups

Ansible Tower no longer supports Ubuntu. Refer to previous versions of the *Ansible Automation Platform Installation and Reference Guide* for details on Ubuntu.

Configuration and Installation on OpenShift

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

Installing Satellite instances on Tower

Satellite users will need to install the Katello RPM for your Satellite instance on the Tower node prior to installing Tower. This RPM automatically configures Subscription Manager to use Satellite as its content source, and the `hostname` value gets updated in `/etc/rhsm/rhsm.conf`.

Note: If you were to install the Katello RPM *after* installing Tower, Tower would not have access to the `rhsm.conf`– which it relies on for applying a subscription from Satellite. This is because the Tower installer sets an ACL rule on the `rhsm.conf` file, therefore, a subscription is unable to be applied if that file does not exist, gets later overwritten, or the user does not have the right permissions to access the file.

For detail on how to register a host with a Satellite server, refer to the Registration section of the Satellite documentation.

1.3 Automation Platform Installation Scenarios

Ansible Automation Platform can be installed using one of the following scenarios:

Single Machine:

- Standalone Tower with database on the same node as Tower or non-installer managed database. 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.

```
[tower]
host
```

- Standalone Tower with an external managed database. This installs the Tower server on a single machine and configures it to talk to a remote instance of PostgreSQL 10 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]
host

[database]
host2
```

- Standalone Automation Hub with a database on the same node as Automation Hub or non-installer managed database:

```
[automationhub]
host
```

- Standalone Automation Hub with an external managed database. This installs the Automation Hub server on a single machine and installs a remote PostgreSQL database via the playbook installer (managed by Automation Platform Installer).

```
[automationhub]
host

[database]
host2
```

Platform Installation:

Platform installation involves Tower and Automation Hub. The Platform installer allows you to deploy 1 and only 1 Automation Hub per inventory. Given the installer can be used as an Automation Hub standalone installer, you can run the installer any number of times with any number of different inventories if you want to deploy multiple Automation Hub nodes. The 2 options supported for the Platform installation are:

- Platform (Tower + Automation Hub) with a database on the same node as Tower or non-installer managed database:

```
[tower]
host1

[automationhub]
host2
```

- Platform (Tower + Automation Hub) with an external managed database:

```
[tower]
host1

[automationhub]
host2

[database]
host3
```

Multi-Machine Cluster

This scenario involves Platform (Clustered Tower + Automation Hub) installation with an external managed database. In this mode, multiple Tower nodes are installed and active. Any node can receive HTTP requests and all nodes can execute jobs. This installs the Platform 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]
host1
host11
host12

[automationhub]
host2

[database]
host3
```

Note: Running in a cluster setup requires any database that Tower uses to be external–PostgreSQL must be installed on a machine that is not one of the primary or secondary tower nodes. When in a redundant setup, the remote PostgreSQL version requirements is *PostgreSQL 10*.

For more information on configuring a clustered setup, refer to Clustering.

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. 3). Tower and Automation Hub can not run on the same node, this is

a scenario that is not supported. It means that any deployment of the Platform becomes at least a 2-node deployment topology.

Settings available for an Automation Platform install:

- `automationhub_importer_settings`: Dictionary of settings/configuration to pass to `galaxy-importer`. It will end up in `/etc/galaxy-importer/galaxy-importer.cfg`
- `automationhub_require_content_approval`: Whether or not Automation Hub enforces the approval mechanism before collections are made available
- `automationhub_disable_https`: Whether or not Automation Hub should be deployed with TLS enabled
- `automationhub_disable_hsts`: Whether or not Automation Hub should be deployed with the HTTP Strict Transport Security (HSTS) web-security policy mechanism enabled
- `automationhub_ssl_validate_certs`: Whether or not Automation Hub should validate certificate when requesting itself (default = False) because by default, Platform deploys with self-signed certificates
- `automationhub_ssl_cert`: Same as `web_server_ssl_cert` but for Automation Hub UI and API
- `automationhub_ssl_key`: Same as `web_server_ssl_key` but for Automation Hub UI and API
- `automationhub_backup_collections`: Automation Hub provides artifacts in `/var/lib/pulp`. By default, this is set to `true` so Tower automatically backs up the artifacts by default. If a partition (e.g., LVM, NFS, CephFS, etc.) was mounted there, an enterprise organization would ensure it is always backed up. If this is the case, you can set `automationhub_backup_collections = false` and the backup/restore process will not have to backup/restore `/var/lib/pulp`.

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

CHAPTER

TWO

DOWNLOAD THE ANSIBLE AUTOMATION PLATFORM INSTALLATION PROGRAM

Starting with Ansible Tower 3.8, Automation Hub will act as a content provider for Ansible Tower, which requires both an Ansible Tower deployment and an Automation Hub deployment running alongside each other. Tower and Automation Hub can run on either RHEL 7 or 8, but only Tower (not Automation Hub) is supported on an OpenShift Container Platform (OCP). Before obtaining the Automation Hub installer, determine how you will be installing your environment.

Both installers have Ansible Tower and Automation Hub; and provide the exact same feature set.

- Installing *with* internet access. Download the latest version of the online installer directly from https://releases.ansible.com/ansible-tower/setup/ansible-tower-setup-latest.tar.gz and then extract the installation/upgrade tool:

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

  Installing with internet access will retrieve the latest required repositories, packages, and dependencies. You must have a valid subscription to activate the Ansible Automation Platform programs.

  After installation or upgrade, start by editing the inventory file in the `ansible-tower-setup-<tower_version>` directory, replacing `<tower_version>` with the version number, i.e., `3.8.0`.

- Installing **without** internet. Download the *bundled installer*. Use this method when you do not have direct access to online repositories, or your environment enforces a proxy.

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

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

---

2.1 Using the Bundled Ansible Automation Platform Installer

The bundled Ansible Automation Platform installer is meant for customers who are unable to access the internet, or would prefer not to install separate components (and its dependencies) from online repositories. Access to Red Hat Enterprise Linux or CentOS repositories is still needed. All other dependencies are included within the tar archive.

1. Access the latest version of the bundled Ansible Automation Platform installation program directly from https://access.redhat.com/downloads/content/480 (note, you must have a Red Hat customer account to access the downloads).

Note: The Ansible Automation Platform installer only supports Red Hat Enterprise Linux and CentOS.

2. Next, select the latest installation program to download. The single `.tar.gz` works with both RHEL 7 and RHEL 8 distributions:

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

Note: On Red Hat Enterprise Linux 7, Ansible Tower 3.8.0 requires some packages from RHSCL repo. If you are installing Tower offline, you need either CentOS-SCL or RH-SCL repositories enabled through a local mirror:

- Red Hat Subscription Manager: `rhel-server-rhscl-7-rpms`
- Red Hat UI: `rhui-rhel-server-rhui-rhscl-7-rpms`
- CentOS: `centos-release-scl`

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.

3. On your chosen machine, extract the installation/upgrade tool:

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

4. Upon installation or upgrade, start by editing the inventory file in the `ansible-tower-setup-<tower_version>` directory, replacing `<tower_version>` with the version number, i.e., `3.8.0`.

CHAPTER

THREE

INSTALLING ANSIBLE AUTOMATION PLATFORM

Ansible Automation Platform 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, you can only deploy Tower on OpenShift. Deployment of Automation Hub on OpenShift is not supported. However, you can deploy Automation Hub in a virtual environment that points to OpenShift. For more detail, refer to OpenShift Deployment and Configuration.

Note: A database server will still be installed even if a node does not have a database. It is a known issue and will be addressed in a future release.

3.1 Ansible Automation Platform Installation Scenarios

Ansible Automation Platform can be installed using one of the following scenarios:

Single Machine:

- Standalone Tower with database on the same node as Tower or non-installer managed database. 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.

```
[tower]
host
```

- Standalone Tower with an external managed database. This installs the Tower server on a single machine and configures it to talk to a remote instance of PostgreSQL 10 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]
host

[database]
host2
```

- Standalone Automation Hub with a database on the same node as Automation Hub or non-installer managed database:

```
[automationhub]
host
```

- Standalone Automation Hub with an external managed database. This installs the Automation Hub server on a single machine and installs a remote PostgreSQL database via the playbook installer (managed by Automation Platform Installer).

```
[automationhub]
host

[database]
host2
```

Platform Installation:

Platform installation involves Tower and Automation Hub. The Platform installer allows you to deploy 1 and only 1 Automation Hub per inventory. Given the installer can be used as an Automation Hub standalone installer, you can run the installer any number of times with any number of different inventories if you want to deploy multiple Automation Hub nodes. The 2 options supported for the Platform installation are:

- Platform (Tower + Automation Hub) with a database on the same node as Tower or non-installer managed database:

```
[tower]
host1

[automationhub]
host2
```

- Platform (Tower + Automation Hub) with an external managed database:

```
[tower]
host1

[automationhub]
host2

[database]
host3
```

Multi-Machine Cluster

This scenario involves Platform (Clustered Tower + Automation Hub) installation with an external managed database. In this mode, multiple Tower nodes are installed and active. Any node can receive HTTP requests and all nodes can execute jobs. This installs the Platform 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]
host1
host11
host12

[automationhub]
host2

[database]
host3
```

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

PostgreSQL version requirements is *PostgreSQL 10*.

For more information on configuring a clustered setup, refer to Clustering.

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. 3). Tower and Automation Hub can not run on the same node, this is a scenario that is not supported. It means that any deployment of the Platform becomes at least a 2-node deployment topology.

Settings available for an Automation Platform install:

- `automationhub_importer_settings`: Dictionary of settings/configuration to pass to `galaxy-importer`. It will end up in `/etc/galaxy-importer/galaxy-importer.cfg`
- `automationhub_require_content_approval`: Whether or not Automation Hub enforces the approval mechanism before collections are made available
- `automationhub_disable_https`: Whether or not Automation Hub should be deployed with TLS enabled
- `automationhub_disable_hsts`: Whether or not Automation Hub should be deployed with the HTTP Strict Transport Security (HSTS) web-security policy mechanism enabled
- `automationhub_ssl_validate_certs`: Whether or not Automation Hub should validate certificate when requesting itself (default = False) because by default, Platform deploys with self-signed certificates
- `automationhub_ssl_cert`: Same as `web_server_ssl_cert` but for Automation Hub UI and API
- `automationhub_ssl_key`: Same as `web_server_ssl_key` but for Automation Hub UI and API
- `automationhub_backup_collections`: Automation Hub provides artifacts in `/var/lib/pulp`. By default, this is set to `true` so Tower automatically backs up the artifacts by default. If a partition (e.g., LVM, NFS, CephFS, etc.) was mounted there, an enterprise organization would ensure it is always backed up. If this is the case, you can set `automationhub_backup_collections = false` and the backup/restore process will not have to backup/restore `/var/lib/pulp`.

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

3.2 Setting up the Inventory File

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

- The contents of the inventory file should be defined in `./inventory`, next to the `./setup.sh` installer playbook.
- For **installations and upgrades**: If you need to make use of external databases, you must ensure the database sections of your inventory file are properly setup. Edit this file and add your external database information before running the setup script.
- For **Ansible Automation Platform** or **Automation Hub**: Be sure to add an automation hub host in the [automationhub] group (Tower and Automation Hub cannot be installed on the same node).
- Tower will not configure replication or failover for the database that it uses, although Tower should work with any replication that you have.
- The database server should be on the same network or in the same data center as the Tower server for performance reasons.

- 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 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

Warning: Do not use special characters in `pg_password` as it may cause the setup to fail.

3.2.1 Example Inventory files

- 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 Standalone Automation Hub Inventory File

```
[automationhub]
automationhub.acme.org
[all:vars]
automationhub_admin_password='<password>'
automationhub_pg_host=''
automationhub_pg_port=''
automationhub_pg_database='automationhub'
automationhub_pg_username='automationhub'
automationhub_pg_password='<password>'
automationhub_pg_sslmode='prefer'
# The default install will deploy a TLS enabled Automation Hub.
# If for some reason this is not the behavior wanted one can
# disable TLS enabled deployment.
#
# automationhub_disable_https = False
# The default install will generate self-signed certificates for the Automation
# Hub service. If you are providing valid certificate via automationhub_ssl_cert
# and automationhub_ssl_key, one should toggle that value to True.
#
# automationhub_ssl_validate_certs = False
# SSL-related variables
# If set, this will install a custom CA certificate to the system trust store.
# custom_ca_cert=/path/to/ca.crt
# Certificate and key to install in Automation Hub node
# automationhub_ssl_cert=/path/to/automationhub.cert
# automationhub_ssl_key=/path/to/automationhub.key
```

Example Platform Inventory File

```
[tower]
tower.acme.org
[automationhub]
automationhub.acme.org
[database]
database-01.acme.org
[all:vars]
admin_password='<password>'
pg_host='database-01.acme.org'
pg_port='5432'
pg_database='awx'
pg_username='awx'
pg_password='<password>'
pg_sslmode='prefer'  # set to 'verify-full' for client-side enforced SSL
# Automation Hub Configuration
#
```

(continues on next page)

(continued from previous page)

```
automationhub_admin_password='<password>'
automationhub_pg_host='database-01.acme.org'
automationhub_pg_port='5432'
automationhub_pg_database='automationhub'
automationhub_pg_username='automationhub'
automationhub_pg_password='<password>'
automationhub_pg_sslmode='prefer'
# The default install will deploy a TLS enabled Automation Hub.
# If for some reason this is not the behavior wanted one can
# disable TLS enabled deployment.
#
# automationhub_disable_https = False
# The default install will generate self-signed certificates for the Automation
# Hub service. If you are providing valid certificate via automationhub_ssl_cert
# and automationhub_ssl_key, one should toggle that value to True.
#
# automationhub_ssl_validate_certs = False
# Isolated Tower nodes automatically generate an RSA key for authentication;
# To disable this behavior, set this value to false
# isolated_key_generation=true
# SSL-related variables
# If set, this will install a custom CA certificate to the system trust store.
# custom_ca_cert=/path/to/ca.crt
# Certificate and key to install in nginx for the web UI and API
# web_server_ssl_cert=/path/to/tower.cert
# web_server_ssl_key=/path/to/tower.key
# Certificate and key to install in Automation Hub node
# automationhub_ssl_cert=/path/to/automationhub.cert
# automationhub_ssl_key=/path/to/automationhub.key
# Server-side SSL settings for PostgreSQL (when we are installing it).
# postgres_use_ssl=False
# postgres_ssl_cert=/path/to/pgsql.crt
# postgres_ssl_key=/path/to/pgsql.key
```

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'
```

Warning: Do not use special characters in `pg_password` as it may cause the setup to fail.

Example Multi Node Cluster Inventory File

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

[database]
dbnode.example.com

[all:vars]
ansible_become=true

admin_password='password'

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

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

Warning: Do not use special characters in `pg_password` as it may cause the setup to fail.

Example Inventory file for an external existing database

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

[database]

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


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

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

Warning: Do not use special characters in `pg_password` as it may cause the setup to fail.

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'

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

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

Warning: Do not use special characters in `pg_password` as it may cause the setup to fail.

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_pass="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 (Only applies to Ansible 2.7)

The `DEFAULT_SUDO` Ansible configuration parameter was removed in Ansible 2.8, which causes the `ANSIBLE_SUDO=True ./setup.sh` method of privilege escalation to no longer work. For more information on `become` plugins, refer to Understanding Privilege Escalation and the list of become plugins.

3.3 Playbook setup

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
```

After calling `./setup.sh` with the appropriate parameters, Tower is installed on the appropriate machines as has been configured. Setup installs Tower from RPM 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 (*https://<TOWER_SERVER_NAME>/*) 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/.

3.4 Changing the Password

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

```
awx-manage changepassword admin
```

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

CHAPTER

FOUR

IMPORT A SUBSCRIPTION

Starting with 3.8, Ansible Tower uses available subscriptions or a subscription manifest to authorize the use of Tower. Previously, Tower used a license key and a JSON dictionary of license metadata. Even if you already have valid licenses from previous versions, you must still provide your credentials or a subscriptions manifest again upon upgrading to Ansible Tower 3.8. To obtain your Tower subscription, you can either:

1. Provide your Red Hat or Satellite username and password on the license page.
2. Obtain a subscriptions manifest from your Subscription Allocations page on the customer portal. See Obtaining a subscriptions manifest for more detail.

If you **have** a Red Hat Ansible Automation Platform subscription, use your Red Hat customer credentials when you launch Tower to access your subscription information (see instructions below).

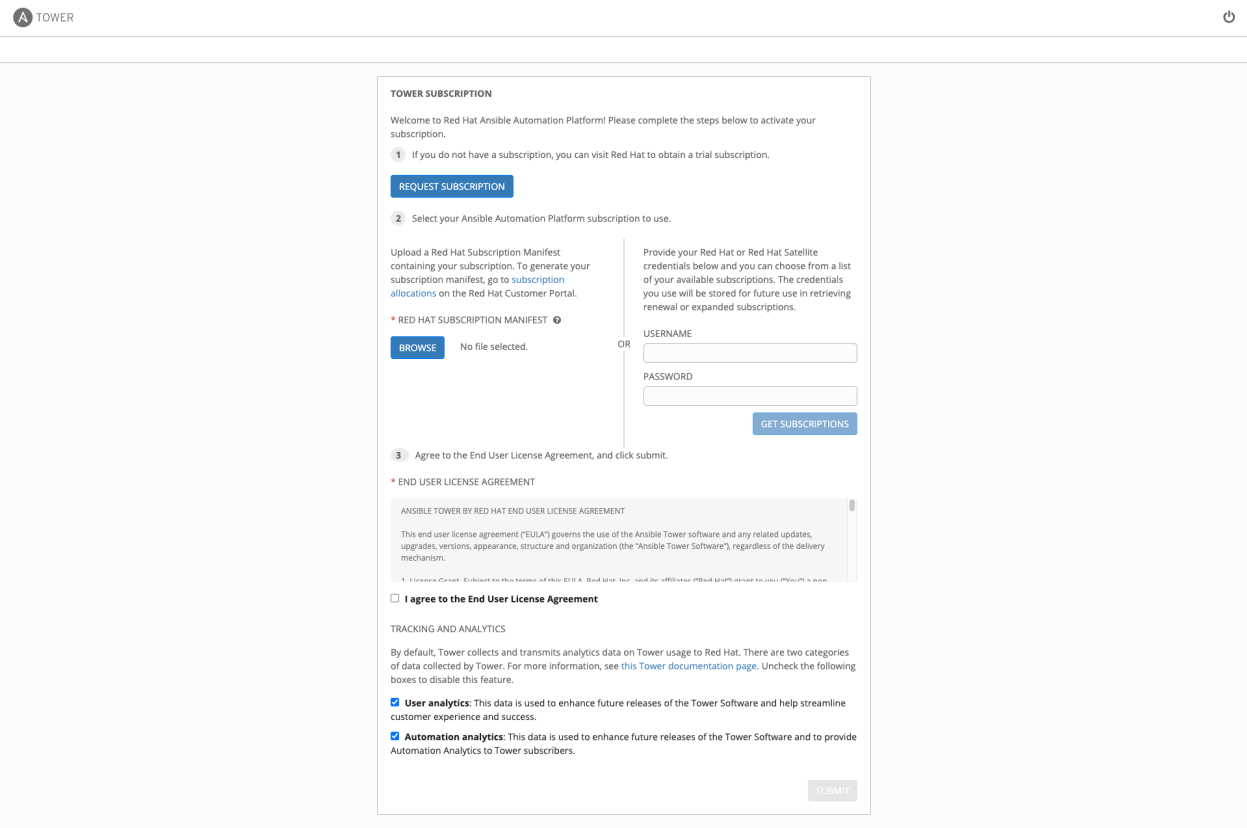
If you **do not** have a Red Hat Ansible Automation Platform subscription, you can request a trial subscription here or click **Request Subscription** and follow the instructions to request one.

Disconnected environments with Satellite will be able to use the login flow on vm-based installations if they have configured subscription manager on the Tower instance to connect to their Satellite instance. Recommended workarounds for disconnected environments **without Satellite** include [1] downloading a manifest from access.redhat.com in a connected environment, then uploading it to the disconnected Tower instance, or [2] connecting to the Internet through a proxy server.

Note: In order to use a disconnected environment, it is necessary to have a valid Ansible Tower entitlement attached to your Satellite organization's manifest. This can be confirmed by using `hammer subscription list \--organization <org_name>`.

If you have issues with the subscription you have received, please contact your Sales Account Manager or Red Hat Customer Service at https://access.redhat.com/support/contact/customerService/.

When Tower launches for the first time, the Tower Subscription screen automatically displays.



Use your Red Hat credentials (username and password) to retrieve and import your subscription, or upload a subscription manifest you generate from https://access.redhat.com/management/subscription_allocations.

1. Enter your Red Hat customer credentials (username and password) and click **Get Subscriptions**. Use your Satellite username/password if your Tower cluster nodes are registered to Satellite via Subscription Manager. See *Installing Satellite instances on Tower* for more information.

Alternatively, if you have a subscriptions manifest, you can upload it by browsing to the location where the file is saved to upload it (the subscription manifest is the complete .zip file, not its component parts). See Obtaining a subscriptions manifest for more detail.

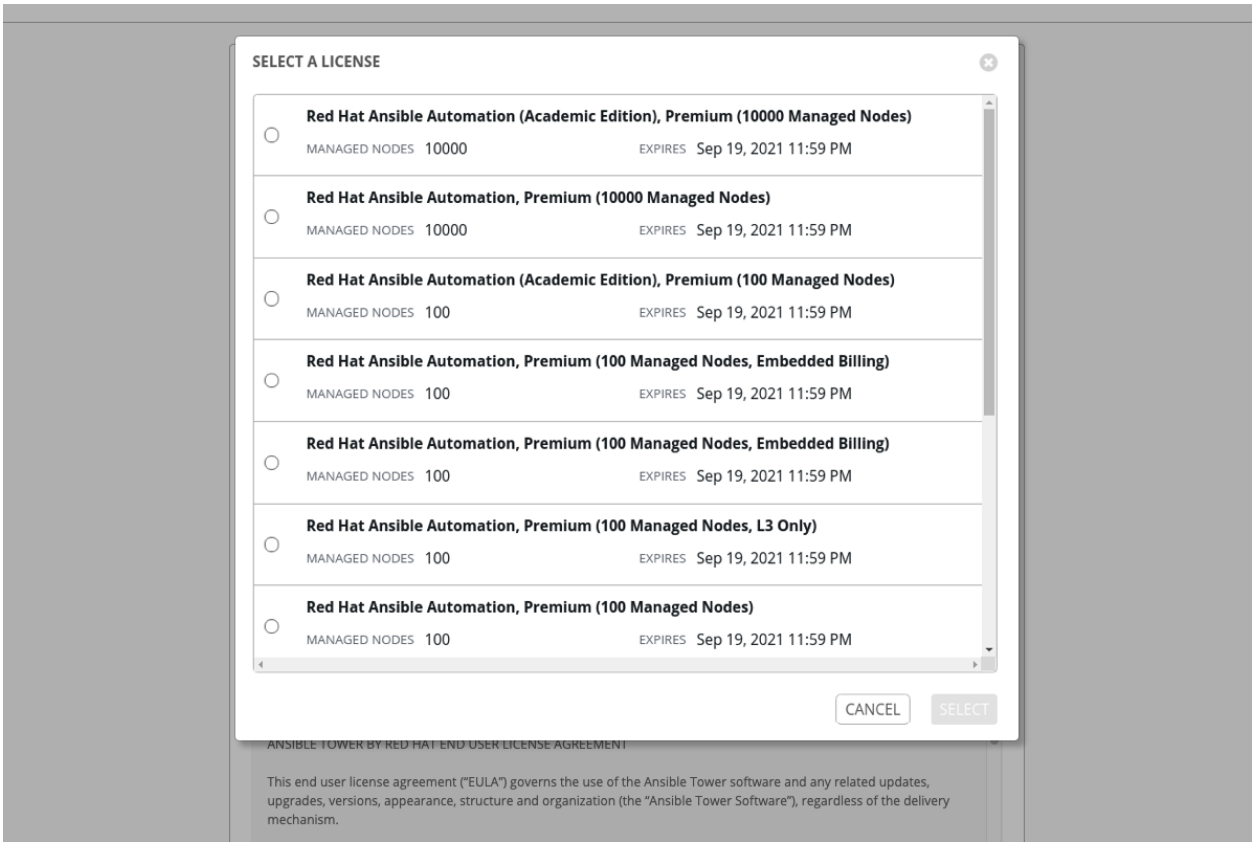
Note: If the **Browse** button is grayed-out, clear the username and password fields to enable the **Browse** button.

---

2. The subscription metadata is then retrieved from the RHSM/Satellite API, or from the manifest provided.

- If it is a subscription manifest, Tower will use the first valid subscription included in your manifest file. This is why it is important to only include the subscription you want applied to the Tower installation.
- If you entered your credential information (username/password), Tower retrieves your configured subscription service. Then it prompts you to choose the subscription you want to run (the example below shows multiple subscriptions) and entitles Tower with that metadata. You can log in over time and retrieve new subscriptions if you have renewed.

Note: When your subscription expires (you can check this on the License settings in the Configure Tower screen of the UI), you will need to renew it in Tower by one of these two methods.



If you encounter the following error message, you will need the proper permissions required for the Satellite user with which the Tower admin uses to apply a subscription.

The Satellite username/password is used to query the Satellite API for existing subscriptions. From the Satellite API, Tower gets back some metadata about those subscriptions, then filter through to find valid subscriptions that you could apply, which are then displayed as valid subscription options in the UI.

The following Satellite roles grant proper access:

- Custom with `view_subscriptions` and `view_organizations` filter
- Viewer
- Administrator

- Organization Admin
- Manager

As the *Custom* role is the most restrictive of these, this is the recommend role to use for your Tower integration. Refer to the Satellite documentation on managing users and roles for more detail.

Note: The System Administrator role is not equivalent to the Administrator user checkbox, and will not provide sufficient permissions to access the subscriptions API page.

---

3. Proceed by checking the **End User License Agreement**.
4. The bottom half of the license screen involves analytics data collection. This helps Red Hat improve the product by delivering you a much better user experience. For more information about data collection, refer to Usability Analytics and Data Collection. This option is checked by default, but you may opt out of any of the following:

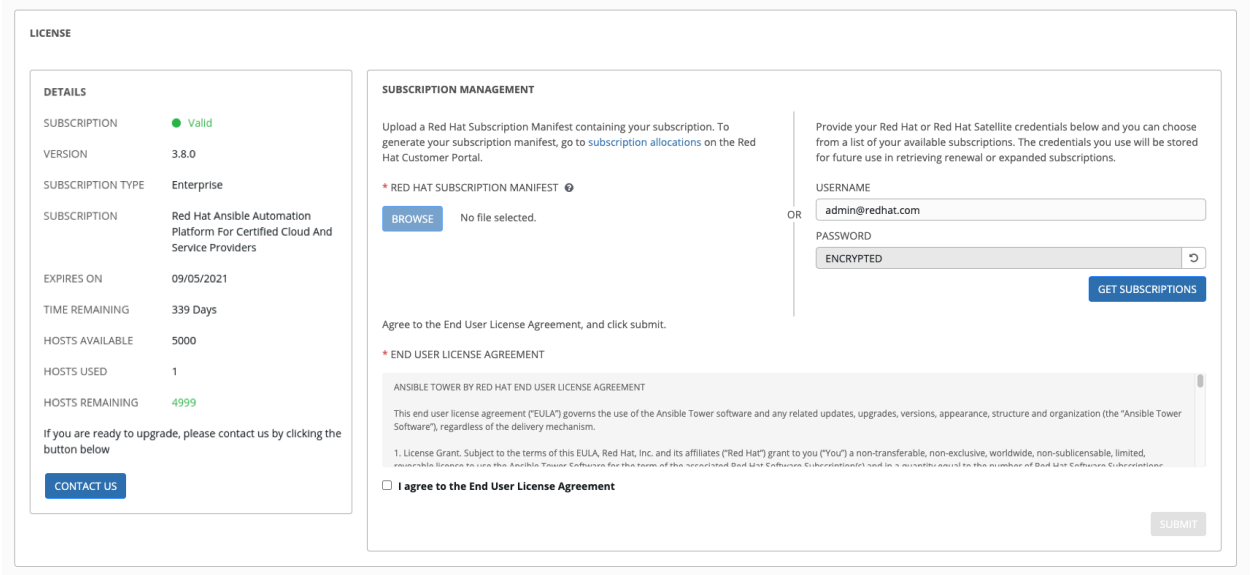
- **User analytics** collects data from the Tower User Interface.
- **Automation analytics** provides a high level analysis of your automation with Ansible Tower, which is used to help you identify trends and anomalous use of Tower. For opt-in of Automation Analytics to have any effect, your instance of Ansible Tower **must** be running on Red Hat Enterprise Linux. See instructions described in the Automation Analytics section.

Note: At this time, Automation Insights is not supported when Ansible Tower is running in the OpenShift Container Platform. You may change your analytics data collection preferences at any time, as described in the Usability Analytics and Data Collection section.

---

5. After you have specified your tracking and analytics preferences, click **Submit**.

Once your subscription has been accepted, Tower briefly displays the license screen and navigates you to the Dashboard of the Ansible Tower interface. For later reference, you can return to the license screen by clicking the Settings () icon from the left navigation bar and select the **License** tab from the Settings screen.



CHAPTER

FIVE

CONGRATULATIONS

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

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

- Ansible Tower Quick Setup Guide
- Ansible Tower Installation and Reference Guide
- Ansible Tower User Guide
- Ansible Tower Administration Guide
- Ansible Tower API Guide
- http://docs.ansible.com/

CHAPTER

SIX

INDEX

- genindex

CHAPTER

SEVEN

COPYRIGHT © 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.

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

A

B

C

D

E

F

I

L

M

O

P

R

S

T