Ansible Tower Upgrade and Migration

Release Ansible Tower 3.3.3

Red Hat, Inc.

Jun 21, 2019

CONTENTS

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

Note: You must upgrade your Ansible Tower 2.4.4 (or later) system to Ansible Tower 3.0 before you can upgrade to Ansible Tower 3.1.0.

The *Ansible Tower Upgrade and Migration Guide* discusses how to upgrade your Ansible Tower 2.4.4 (or later) system to the 3.0 version. The Ansible Tower 3.0 release introduced many updates and changes to Tower, including changes to the way upgrades run and a completely rewritten RBAC system. This guide covers these and other related changes that you should keep in mind as you plan to migrate your data and prepare for this upgrade.

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 3.3.3; December 19, 2018; https://access.redhat.com/

RELEASE NOTES FOR 3.3.X

1.1 Ansible Tower Version 3.3.3

- Fixed a RabbitMQ misconfiguration that decreased cluster stability and could allow for unauthorized access (CVE-2018-16879)
- Fixed RabbitMQ HA policy when deployed in OpenShift

1.2 Ansible Tower Version 3.3.2

- Fixed external loggers to send activity stream to send changes as raw JSON, instead of JSON-ified string
- Fixed `MANAGE_ORGANIZATION_AUTH` settings to allow superusers to make changes
- Fixed an XSS issue when viewing application tokens
- Fixed permissions to allow project admins to create projects
- Fixed permissions to allow job template admins to delegate permissions to certain users/teams
- Fixed Tower to allow selecting credential types when running in a non-English language
- Fixed the Activity Stream to properly note credentials and custom credentials
- Fixed an error resulting from a credential lookup where there are multiple custom credential types defined
- Fixed certain certificate validation issues
- Fixed templates to allow filtering by template type
- Fixed events with no output to no longer render excess blank lines in the job display
- Fixed detailed events to show hosts that were added during a playbook run
- Fixed the inability to properly mark Tower nodes as disabled when rabbitmq was offline
- Fixed the job output to correctly display certain job events
- Fixed the job events to properly line-wrap in the job output
- Fixed the pagination to no longer redirect to the dashboard when selecting multiple credentials
- Fixed a potential deadlock on inventory deletion
- Fixed the job relaunch operation to use the correct credentials when the defined credentials were changed
- Improved the Schedules view to more clearly denote which resource was being scheduled

- Updated Google Cloud credentials to be passed as `GCE_PEM_FILE_PATH` to `GCE_CREDENTIALS_FILE_PATH` in accordance with recent Ansible versions

1.3 Ansible Tower Version 3.3.1

- Fixed event callback error when in-line vaulted variables are used with `include_vars`
- Fixed HSTS and X-Frame-Options to properly be set in nginx configuration
- Fixed isolated node setup to no longer fail when `ansible_host` is used
- Fixed selection of custom virtual environments in job template creation
- Fixed display of `extra_vars` for scheduled jobs
- Fixed websockets for job details to properly work
- Fixed the `/api/v2/authtoken` compatibility shim. See the compatibility shim for detail.
- Fixed page size selection on the jobs screen
- Fixed instances in an instance group to properly be disabled in the user interface
- Fixed the job template selection in workflow creation to properly render
- Fixed `member_attr` to properly set on some LDAP configurations during upgrade, preventing login
- Fixed `PosixUIDGroupType` LDAP configurations
- Improved the RAM requirement in the installer preflight check
- Updated Tower to properly report an error when relaunch was used on a set of failed hosts that is too large
- Updated sosreport configuration to gather more python environment, nginx, and supervisor configuration

1.4 Ansible Tower Version 3.3.0

- Added support for container-based clusters using OpenShift
- Added support for multiple or no credential assignment in Job Templates
- Added support for multiple Vault credential assignment in Job Templates
- Added support for multiple LDAP server configurations
- Added support fact caching for isolated instances
- Added the ability to schedule configurations of a job template using job template prompts
- Added deprecation note to the Tower REST API for Version 1 (`/api/v1/`), which will be removed in a future release of Ansible Tower
- Added the ability to make a copy of existing Tower objects (inventory, project, etc.) as a template for creating a new one
- Added the ability to relaunch jobs on a subset of hosts by status
- Added validation to prevent string `"$encrypted$"` from becoming a literal survey question default
- Added support for more job template prompts at workflow node creation time
- Added `ask_variables_on_launch` to workflow job templates (WFJT)
- Added `diff_mode` and `verbosity` fields to WFJT nodes

- Added Saved Launch-time configurations feature - added WFJT node promptable fields to schedules, added `extra_data` to WFJT nodes, and added "schedule this job" endpoint.

- Added block creation of schedules when variables not allowed are given. Block similar cases for WFJT nodes.

- Added the ability to create instance groups and associate instances at runtime via the user interface and API

- Added the ability to group instances based on policy, such as such as "50% of instances" and "at least three instances"

- Added additional organizational roles for administration of projects, job templates, inventories, workflows and more

- Added support for custom virtual environments for customizing Ansible execution

- Added OAuth2 support for token based authentication

- Added support for OAuth2 applications and access token generation

- Added the ability to forcibly expire sessions through `awx-manage expire_sessions`

- Added support for making inventory parsing errors fatal, and only enable the `script` inventory plugin for job runs and vendored inventory updates

- Added `inventory` field to inventory updates

- Added related credentials endpoint for inventory updates to be more internally consistent with job templates, model changes

- Added the ability to show all teams to organization admins if the `ORG_ADMINS_CAN_SEE_ALL_USERS` setting is enabled

- Added the ability to create schedules and workflow nodes from job templates that use credentials which prompt for passwords if `ask_credential_on_launch` is set.

- Deprecated the `fact_versions` and `fact_view` endpoints from the API, including OPTIONS

- Deprecated fact tables

- Deprecated the `awx-manage cleanup_facts` command for fact cleanup

- Deprecated the `/api/v2/authtoken/` endpoint in the API and replaced it with `/api/v2/tokens/`. See the compatibility shim for detail.

- Fixed a conflict with Tower credential type by removing `TOWER_HOST` as a default environment variable in job running environments. Playbook authors should replace their use with `AWX_HOST`.

- Fixed a behavior in Tower to prevent it from deleting jobs when event processing is still ongoing

- Fixed a behavior in Tower to disallow relaunching jobs with `execute_role` if another user provided prompts

- Improved project updates so that previously synced git projects do not attempt to contact the server if they are already at the proper revision

- Improved WFJT node `credential` to many-to-many `credentials`

- Improved stricter criteria to admin users where organization admin role now necessary for all organizations of which the target user is a member. Additionally, removed unused `admin_role` associated with users

- Improved logs to consistently catch task exceptions

- Improved external loggers to passively create handler from settings on every log emission, replacing server restart, allowing use in OpenShift deployments

- Improved Tower to automatically run a project update if sensitive fields change like `scm_url`

- Improved queuing logic through setting `execution_node` in task manager and submitting waiting jobs to only the queue for the specific instance job is targeted to run on

- Updated the `auth-token-timeout` header name to `Session-Timeout`

- Updated the `AUTH_TOKEN_EXPIRATION` setting name to change to `SESSION_COOKIE_AGE` and `AUTH_TOKEN_PER_USER` changed to `SESSIONS_PER_USER`

- Updated source-control based inventory to allow for vaulted variable values

- Updated the minimum required version of Red Hat Enterprise Linux to 7.4

- Updated the minimum required RAM for standalone Tower to 4GB

- Updated Ansible Tower to set `ANSIBLE_DISPLAY_ARGS_TO_STDOUT` to `False` by default for all playbook runs to match Ansible's default behavior. See Jobs for more information.

- Updated all job and tasks to generate consistent output events and make job output available on all cluster nodes

- Updated external logging to default to HTTPS unless http:// is explicitly specified in the log aggregator hostname

- Updated the behavior of a job template to prohibit configuring callbacks on job templates without an inventory

- Updated the boolean fields for custom credential types to always default `extra_vars` and environment variables to `False` when a value is not provided

- Updated to disallow using HTTP PUT/PATCH methods to modify existing jobs in Job Details API endpoint

UPGRADING ANSIBLE TOWER

Topics:

- *Upgrade Planning*
- *Obtaining Ansible Tower*
- *Setting up the Inventory File*
- *The Setup Playbook*

Note: You must upgrade your Ansible Tower 2.4.4 (or later) system to Ansible Tower 3.0 before you can upgrade beyond Ansible Tower 3.0. Refer to the recommended upgrade path article from your customer portal.

2.1 Upgrade Planning

This section covers changes that you should keep in mind as you attempt to upgrade your Ansible Tower Instance

- If you are not yet using a 2.4.x version of Ansible Tower, **do not** attempt to upgrade directly to Ansible Tower 3.0.x or 3.1.0. You must start with a system which has a verison of Tower 2.4.x installed or the upgrade will fail.

- If you are not using using a 3.0.x version of Ansible Tower; you must upgrade to 3.0.x before you can upgrade beyond Ansible Tower 3.0.

- Ansible Tower 3.0 simplified installation and removed the need to run `./configure/` as part of the initial setup.

- The file `tower_setup_conf.yml` is no long used. Instead, you should now edit the **inventory** file in the `/ansible-tower-setup-<tower_version>/` directory.

- Earlier version of Tower used MongoDB when setting up an initial database; please note that Ansible Tower 3.0 has replaced the use of MongoDB with PostgreSQL.

- Clustered upgrades require special attention to instance and instance groups prior to starting the upgrade. Refer to the *Setting up the Inventory File* section.

- Changes in API authentication were made in Ansible Tower 3.3 to accommodate additional OAuth2 functionality. For more information, refer to Token-Based Authentication in the *Ansible Tower Administration Guide*.

2.2 Obtaining Ansible Tower

You may install standalone Tower or use the bundled installer:

- if you set up Tower on an environment with a direct Internet access, you can download the standalone Tower installer

- if you set up Tower on an environment without direct access to online repositories, or your environment enforces a proxy, you must use the bundled installer

Download and then extract the Ansible Tower installation/upgrade tool: http://releases.ansible.com/ansible-tower/setup/

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

To install or upgrade, start by editing the inventory file in the `ansible-tower-setup-<tower_version>` directory, replacing `<tower_version>` with the version number, such as `2.4.5` or `3.0.0`. directory.

2.3 Setting up the Inventory File

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

- The contents of the inventory file should be defined in `./inventory`, next to the `./setup.sh` installer playbook.

- For **installations and upgrades**: If you need to make use of external databases, you must ensure the database sections of your inventory file are properly setup. Edit this file and add your external database information before running the setup script.

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

- For **clustered installations**: If you are creating a clustered setup, you must replace `localhost` with the hostname or IP address of all instances. All nodes/instances must be able to reach any others using this hostname or address. In other words, you cannot use the `localhost ansible_connection=local` on one of the nodes *AND* all of the nodes should use the same format for the host names.

  Therefore, this will *not* work:

  ```
  [tower]
  localhost ansible_connection=local
  hostA
  hostB.example.com
  172.27.0.4
  ```

  Instead, use these formats:

  ```
  [tower]
  hostA
  hostB
  hostC
  ```

OR

```
hostA.example.com
hostB.example.com
hostC.example.com
```

OR

```
[tower]
172.27.0.2
172.27.0.3
172.27.0.4
```

- For **all standard installations**: When performing an installation, you must supply any necessary passwords in the inventory file.

Note: Changes made to the installation process now require that you fill out the all of the password fields in the inventory file. If you need to know where to find the values for these they should be:

`admin_password=''` <—— Tower local admin password

`pg_password=''` <—- Found in /etc/tower/conf.d/postgres.py

`rabbitmq_password=''` <—- create a new password here (alpha-numeric with no special characters)

Example Inventory file

- For **provisioning new nodes**: When provisioning new nodes add the nodes to the inventory file with all current nodes, make sure all passwords are included in the inventory file.

- For **upgrading a single node**: When upgrading, be sure to compare your inventory file to the current release version. It is recommended that you keep the passwords in here even when performing an upgrade.

Example Single Node Inventory File

```
[tower]
localhost ansible_connection=local

[database]

[all:vars]
admin_password='password'

pg_host=''
pg_port=''

pg_database='awx'
pg_username='awx'
pg_password='password'

rabbitmq_port=5672
rabbitmq_vhost=tower
rabbitmq_username=tower
rabbitmq_password='password'
rabbitmq_cookie=rabbitmqcookie

# Needs to be true for fqdns and ip addresses
```

```
rabbitmq_use_long_name=false
# Needs to remain false if you are using localhost
```

Example Multi Node Cluster Inventory File

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

[database]
dbnode.example.com

[all:vars]
ansible_become=true

admin_password='password'

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

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

rabbitmq_port=5672
rabbitmq_vhost=tower
rabbitmq_username=tower
rabbitmq_password=tower
rabbitmq_cookie=rabbitmqcookie

# Needs to be true for fqdns and ip addresses
rabbitmq_use_long_name=true
```

Example Inventory file for an external existing database

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

[database]

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


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

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

Example Inventory file for external database which needs installation

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

```
[database]
database.example.com

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

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

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

Once any necessary changes have been made, you are ready to run `./setup.sh`.

Note: Root access to the remote machines is required. With Ansible, this can be achieved in different ways:

- ansible_user=root ansible_ssh_password="your_password_here" inventory host or group variables

- ansible_user=root ansible_ssh_private_key_file="path_to_your_keyfile.pem" inventory host or group variables

- ANSIBLE_BECOME_METHOD='sudo' ANSIBLE_BECOME=True ./setup.sh

- ANSIBLE_SUDO=True ./setup.sh

2.4 The Setup Playbook

Note: Ansible Tower 3.0 simplifies installation and removes the need to run `./configure/` as part of the installation setup. Users of older versions should follow the instructions available in the v.2.4.5 (or earlier) releases of the Tower Documentation available at: http://docs.ansible.com/

The Tower setup playbook script uses the `inventory` file and is invoked as `./setup.sh` from the path where you unpacked the Tower installer tarball.

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

The setup script takes the following arguments:

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

- `-i INVENTORY_FILE` – Path to Ansible inventory file (default: `inventory`)

- `-e EXTRA_VARS` – Set additional Ansible variables as key=value or YAML/JSON (i.e. `-e bundle_install=false` forces an online installation)

- `-b` – Perform a database backup in lieu of installing

- `-r` – Perform a database restore in lieu of installing (a default restore path is used unless EXTRA_VARS are provided with a non-default path, as shown in the code example below)

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

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

If you need to back up or restore your Tower v3.0.0 or v3.0.1 installation, use the v3.0.2 installer to do so.

SYSTEM TRACKING MIGRATION

System tracking feature was deprecated starting with Ansible Tower 3.2. However, you can collect facts by using the fact caching feature. Refer to Fact Caching for more detail. If you upgrade from a version earlier than Ansible Tower 3.0, you will find that your system tracking data (historical facts) has been migrated from MongoDB to PostgreSQL.

If you want to delete the old data in MongoDB, you can do so manually. First, connect to your mongo database using the mongo command line client, then run the following commands:

```
$ use system_tracking
$ db.runCommand( { dropDatabase: 1 } )
```

At this point, you can also remove the MongoDB packages.

FOUR

ROLE-BASED ACCESS CONTROLS

Ansible Tower 3.0 has changed significantly around the way that the Role-Based Access Control (RBAC) system works. For the latest RBAC documentation, refer to the Role-Based Access Controls section in the Tower User Guide.

4.1 Enhanced and Simplified RBAC System

Based on user feedback, Ansible Tower both expands and simplifies its role-based access control. No longer is job template visibility configured via a combination of permissions on inventory, projects, and credentials. If you want to give any user or team permissions to use a job template, just assign permissions directly on the job template. Similarly, credentials are now full objects in Tower's RBAC system, and can be assigned to multiple users and/or teams for use.

A new 'Auditor' type has been introduced in Tower as well, who can see all aspects of the systems automation, but has no permission to run or change automation, for those that need a system-level auditor. (This may also be useful for a service account that scrapes automation information from Tower's API.)

4.2 Specific Changes to Note

There are a few changes you should keep in mind as you work with the RBAC system as redesigned for Ansible Tower:

- You no longer set the "team" or "user" for a credential. Instead, you use Tower's RBAC system to grant ownership, auditor, or usage roles.

- Deletion of job run data is now restricted to system and organization administrators.

- Projects no longer have multiple organizations. You *must* provide an organization when creating a new project through the API:

```
- projects/:id/organizations --> removed
```

- New Auditor type in Tower has been added which can see all aspects of the systems automation but does not have permission to run or change things.

JOB TEMPLATE CHANGES

Job templates have been updated in Tower to allow you more flexibility when creating and working with them.

5.1 Scan Jobs

If you maintained scan job templates in Ansible Tower 3.1.x and then upgrade to Ansible Tower 3.2, a new "Tower Fact Scan - Default" project is automatically created for you. This project contains the old scan playbook previously used in earlier versions of Ansible Tower.

5.2 Prompt on Launch

In prior versions of Ansible Tower, you could set "Prompt on Launch" against Extra Variables that you want to potentially pass through the job template. Starting with version 3.0, Ansible Tower allows you to prompt for an inventory selection, job type, and more.

Selecting "Prompt on Launch" means that even if a value is supplied at the time of the job template creation, the user launching the job will be prompted to supply new information or confirm what was entered in the job template originally.

The following job template settings allow for prompting at the time of launch:

- Job Type (run or check type jobs only, as scan jobs cannot be changed at the time of launch)
- Inventory
- Machine credential
- Limit
- Job Tags
- Extra variables

As you work with migrating your Tower 2.4.5 job templates to 3.x, please keep in mind the following:

- All "Prompt on Launch" fields are set to *False* by default after migrating to 3.x (new job templates created also have all "Prompt on Launch" fields set to *False* by default).

    - With one exception for those upgrading from 2.4.5 to 3.x: if a credential used in Tower 2.4.5 was null, the credential will be prompted for in 3.x.

- If you have Job Templates with a null credential, in the migration from 2.4.5 to 3.x, "ask_credential_on_launch" is set to *True*.

    - Note that there was no way to set a default credential in 2.4.5. However, in 3.x, you can set a default credential and select to prompt the user at launch time to confirm the default credential or change it to something new.

- All other "ask_xx_on_launch" prompts are set to *False*.

- Starting with Tower 3.x, if "ask_variables_on_launch" is set to *False*, extra variables passed at launch time (via UI or API) that are not part of an enabled survey are ignored.

- While there are no changes to how "ask_variables_on_launch" behaves, keep in mind that these variables combine with survey answers.

5.3 Permissions/RBAC Notes

Job template visibility is no longer configured via a combination of permissions on inventory, projects, and credentials. Admins who want to give any user or team permissions to use a job template can quickly assign permissions directly on the job template. Similarly, credentials are now full objects in Tower's RBAC system, and can be assigned to multiple users and/or teams for use.

If a job template a user has been granted execution capabilities on does not specify an inventory or credential, the user will be prompted at run-time to select among the inventory and credentials in the organization they own or have been granted usage capabilities.

Users that are job template administrators can make changes to job templates; however, to make changes to the inventory, project, playbook, or credentials used in the job template, the user must also have the "Use" role for the project, inventory, and all credentials currently being used or being set.

5.4 Surveys

In prior versions of Ansible Tower, you had to select a checkbox to "Enable Survey" on the Job Template before a button appeared allowing you to "Create Survey".

Enabling and creating surveys is much simpler in Ansible Tower.

At the bottom of each job template is a button (**ADD SURVEY**) which opens a new dialog where you can enter your survey questions and responses.

Use the **ON/OFF** toggle button to quickly activate or deactivate this survey prompt.

Once you have entered the question information, click **Add** to add the survey prompt.

A stylized preview of the survey is presented, along with a **New Question** button. Click this button to add additional questions.

For any question, you can click on the **Edit** button to edit the question, the **Delete** button to delete the question, and click on the Up and Down arrow buttons to rearrange the order of the questions. Click **Save** to save the survey.

SIX

JOB OUTPUT VIEW CHANGES

With the update of the overall Tower user interface, it is worth noting the changes to how job results are displayed.

Job results for inventory syncs and SCM updates only show the Results and Standard Out of the job recently Run. Job results for playbook runs consist of Results, Standard Out, Details, and the Event Summary.
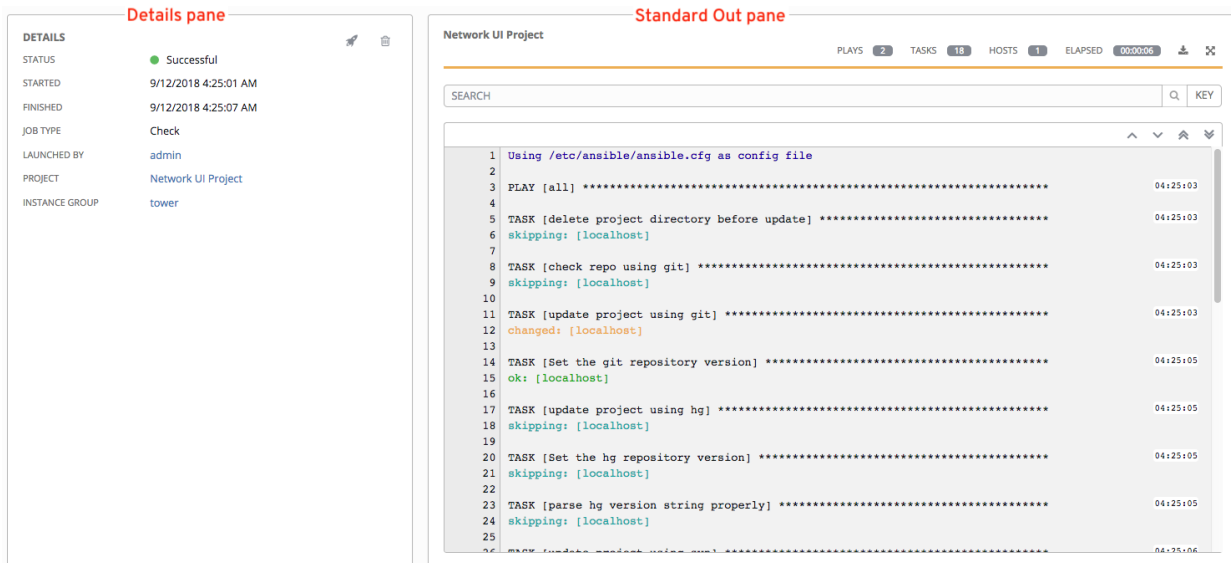
For more details regarding Job Results, refer to Jobs in the *Ansible Tower User Guide*.

6.1 Details

The **Details** area shows the basic status of the job (*Running*, *Pending*, *Successful*, or *Failed*), its start and end times, which template was used, how long the job run took, who launched it, and more. The buttons in the top right of the Details view allow you to relaunch or delete the job.

By clicking on these detail entries, where appropriate, you can view the corresponding job templates, projects, and other Tower objects.

DETAILS

STATUS	● Successful
STARTED	1/20/2017 1:27:48 PM
FINISHED	1/20/2017 1:27:54 PM
TEMPLATE	Demo Job Template
JOB TYPE	Run
LAUNCHED BY	admin
INVENTORY	Demo Inventory
PROJECT	● Demo Project
REVISION	347e44fea036c94d5f60e544de006453ee5c71ad
PLAYBOOK	hello_world.yml
MACHINE CREDENTIAL	Demo Credential
FORKS	0
VERBOSITY	0 (Normal)

EXTRA VARIABLES ❓

```
1 ---
```

6.2 Standard Out Pane

The **Standard Out** pane shows the full results of running the Ansible playbook. This shows the same information you would see if you ran it through the Ansible command line, and can be useful for debugging. You can view the event summary, host status, and the host events. The icons at the top right corner of the Standard Out pane allow you to toggle the output as a main view ( ) or to download the output ( ).

6.2.1 Events Summary

The events summary captures a tally of events that were run as part of this playbook:

- the number of plays
- the number of tasks
- the number of hosts
- the elapsed time to run the job template
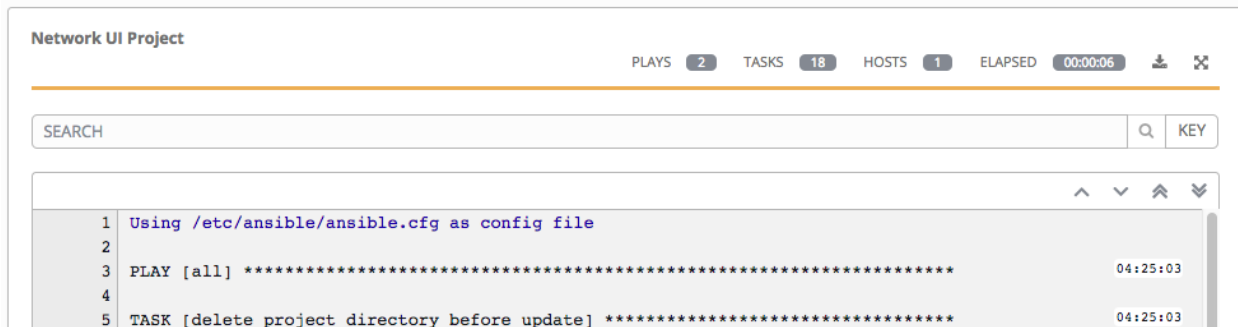
6.2.2 Host Status Bar

The host status bar runs across the top of the **Standard Out** pane. Hover over a section of the host status bar and the number of hosts associated with that particular status displays.
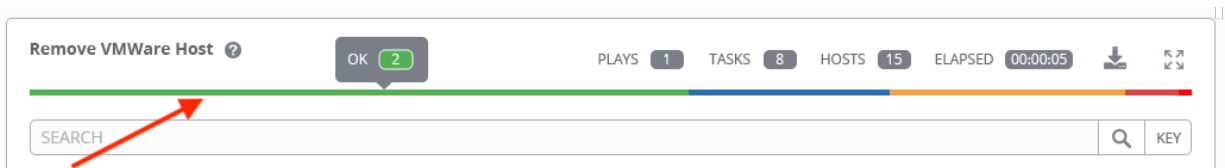
6.2.3 Standard output view

The standard output view displays all the events that occur on a particular job. By default, all rows are expanded so that all the details are displayed. Use the collapse-all button ( ) to switch to a view that only contains the headers for plays and tasks. Click the ( ) button to view all lines of the standard output.

Alternatively, you can display all the details of a specific play or task by clicking on the arrow icons next to them. Click an arrow from sideways to downward to expand the the lines associated with that play or task. Click the arrow back to the sideways position to collapse and hide the lines.

Click on a line of an event from the **Standard Out** pane and a **Host Events** dialog displays in a separate window. This window shows the host that was affected by that particular event.

6.2.4 Host Events

The **Host Events** dialog shows information about the host affected by the selected event and its associated play and task:
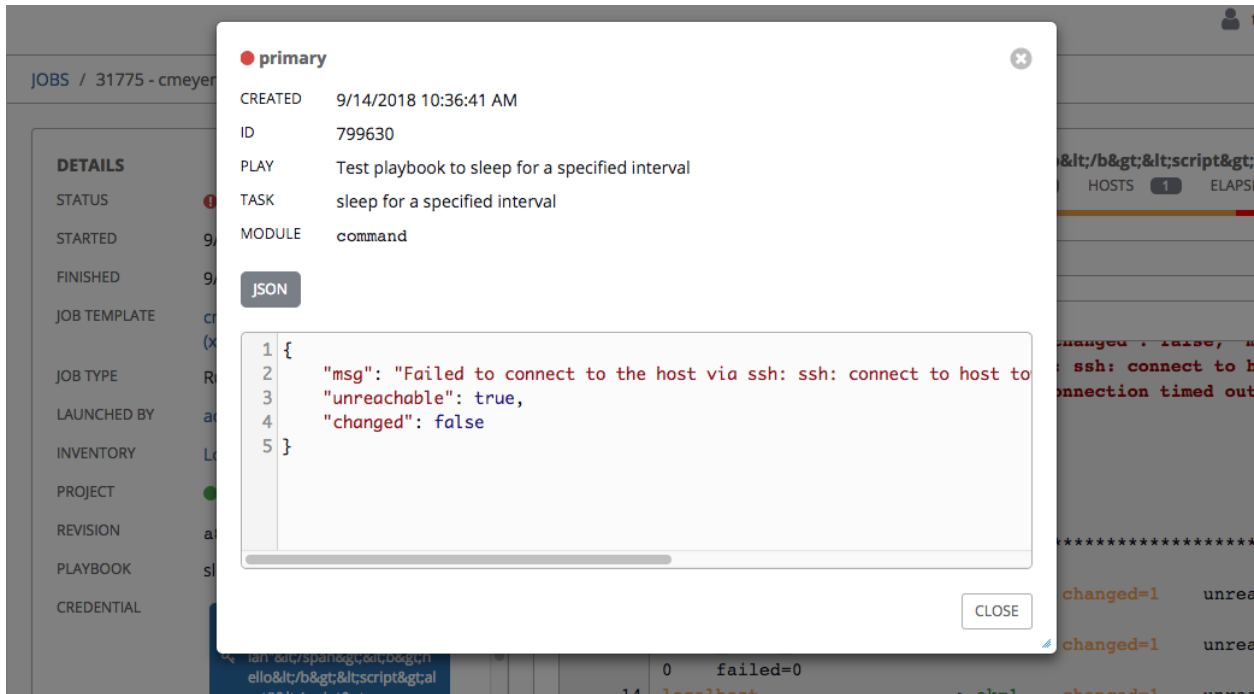
In the above example, this dialog box shows a single button for JSON, which is the only format given for the host event to view and display to you for this particular event. If Standard Out and Shell outputs are available, those buttons will display to allow you to see other output formats.

USING VIRTUALENV WITH ANSIBLE TOWER

Ansible Tower 3.0 and later uses *virtualenv*. Virtualenv creates isolated Python environments to avoid problems caused by conflicting dependencies and differing versions. Virtualenv works by simply creating a folder which contains all of the necessary executables and dependencies for a specific version of Python. Ansible Tower creates two virtualenvs during installation–one is used to run Tower, while the other is used to run Ansible. This allows Tower to run in a stable environment, while allowing you to add or update modules to your Ansible Python environment as necessary to run your playbooks. For more information on virtualenv, see the Python Guide to Virtual Environments and the *Python virtualenv* project itself.

By default, the virtualenv is located at `/var/lib/awx/venv/ansible` on the file system.

Tower also pre-installs a variety of third-party library/SDK support into this virtualenv for its integration points with a variety of cloud providers (such as EC2, OpenStack, Azure, etc.) Periodically, you may want to add additional SDK support into this virtualenv, which is described in further detail below.

Note: It is highly recommended that you run `umask 0022` before installing any packages to the virtual environment. Failure to properly configure permissions can result in Tower service failures. An example follows:

```
# source /var/lib/awx/venv/ansible/bin/activate
# umask 0022
# pip install --upgrade pywinrm
# deactivate
```

In addition to adding modules to the virtualenv that Tower uses to run Ansible, you can create new virtualenvs as described below.

7.1 Preparing a new custom virtualenv

Tower allows a different virtualenv to be specified and used on Job Template runs. To choose a custom virtualenv, first create one under `/var/lib/awx/venv/`:

```
$ sudo virtualenv /var/lib/awx/venv/my-custom-venv
```

Your newly created virtualenv needs a few base dependencies to properly run playbooks (Tower uses memcached as a placeholder for playbook artifacts):

```
$ sudo /var/lib/awx/venv/my-custom-venv/bin/pip install python-memcached psutil
```

From here, you can install additional Python dependencies that you care about, such as a per-virtualenv version of Ansible itself:

```
$ sudo /var/lib/awx/venv/my-custom-venv/bin/pip install -U "ansible == X.Y.Z"
```

Or you can add an additional third-party SDK that is not included with the base Tower installation:

```
$ sudo /var/lib/awx/venv/my-custom-venv/bin/pip install -U python-digitalocean
```

If you want to copy them, the libraries included in Tower's default virtualenv can be found using `pip freeze`:

```
$ sudo /var/lib/awx/venv/ansible/bin/pip freeze
```

In a clustered Tower installation, you need to ensure that the same custom virtualenv exists on every local file system at `/var/lib/awx/venv/`. Custom virtualenvs are supported on isolated instances. If you are using a custom virtual environment, it needs to also be copied or replicated on any isolated node you would be using, not just on the Tower node.

7.2 Assigning custom virtualenvs

Once you have created a custom virtualenv, you can assign it at the Organization, Project, or Job Template level:

```
PATCH https://awx-host.example.org/api/v2/organizations/N/
PATCH https://awx-host.example.org/api/v2/projects/N/
PATCH https://awx-host.example.org/api/v2/job_templates/N/


Content-Type: application/json
{
        'custom_virtualenv': '/var/lib/awx/venv/my-custom-venv'
}
```

An HTTP GET request to `/api/v2/config/` provides a list of detected installed virtualenvs:

```
{
        "custom_virtualenvs": [
                "/var/lib/awx/venv/my-custom-venv",
                "/var/lib/awx/venv/my-other-custom-venv",
        ],
...
}
```

You can also specify the virtual environment to assign to an Organization, Project, and Job Template from their respective edit screens in the Ansible Tower User Interface. Select the virtualenv from the **Ansible Environment** drop-down menu, as shown in the example below:

ORGANIZATIONS / Default

Default

| DETAILS | USERS | NOTIFICATIONS |

* NAME
DESCRIPTION

Default

ANSIBLE ENVIRONMENT

Use Default Environment

Use Default Environment

/var/lib/awx/venv/venv1/
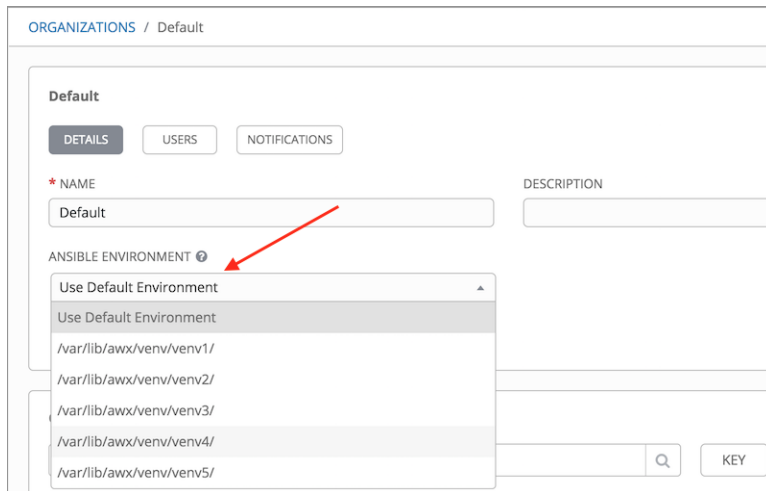
/var/lib/awx/venv/venv2/

/var/lib/awx/venv/venv3/

/var/lib/awx/venv/venv4/

/var/lib/awx/venv/venv5/

KEY

INDEX

- genindex

COPYRIGHT © 2018 RED HAT, INC.