Ansible Tower Upgrade and Migration

Release Ansible Tower 3.2.4

Red Hat, Inc.

Jul 01, 2020

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 3.2.4; April 27, 2018; https://access.redhat.com/

CHAPTER

ONE

RELEASE NOTES FOR ANSIBLE TOWER VERSION 3.2.4

1.1 Ansible Tower Version 3.2.4

- Added `UI_LIVE_UPDATES_ENABLED` setting for disabling websocket updates outside of job output
- Fixed organization admins to no longer be able to modify users by adding them to their organization (CVE-2018-1101)
- Fixed Tower to disable usage of Jinja templates in launch-time variables for security reasons (CVE-2018-1104). This release introduces the `ALLOW_JINJA_IN_EXTRA_VARS` configuration parameter for Tower. This parameter has three values: `template` to allow usage of Jinja saved directly on a job template definition (the default), `never` to disable all Jinja usage (recommended), and `always` to always allow Jinja (strongly discouraged, but an option for prior compatibility). Note that the `always` option is deprecated, and will be removed in a future Tower release.
- Fixed sanitization of module arguments with implicit `no_log`
- Fixed Smart Inventories to no longer run on hosts marked as disabled
- Fixed Fact Caching documentation to no longer refer to memcached
- Updated bundled python-saml for CVE-2017-11427
- Updated memcached to now listen on a local Unix socket instead of a TCP socket

1.2 Ansible Tower Version 3.2.3

- Added deprecation warning when installing on certain older operating systems, such as Ubuntu 14.04, which will be removed in a future release
- Fixed Inventory Updates to properly save `group_vars` inside of Tower group variables when used with Ansible 2.5 or later
- Fixed certain Inventory Updates to no longer fail when running against isolated nodes
- Fixed the ability to customize `ANSIBLE_LIBRARY` when Job Template fact caching is enabled
- Fixed fact cache data to no longer prematurely expire for Job Templates with large amounts of fact data
- Fixed isolated job runs to no longer fail when the playbook contained certain Unicode characters
- Fixed the installer to use the correct package version when running isolated Tower nodes
- Fixed Slack notification issues
- Fixed workflow artifacts to no longer periodically go missing in subsequent workflow nodes

- Fixed the Tower web interface to support large numbers of custom Credential Types
- Fixed the "Test" button when configuring UDP-based external logging
- Fixed the database restoration process that affected users with embedded PostgreSQL databases
- Fixed a few XSS vulnerabilities in the Tower web interface
- Fixed the ability to provide the admin password in the MOTD file for the Vagrant and AMI images

1.3 Ansible Tower Version 3.2.2

- Added support for Ansible Tower and Red Hat Virtualization credentials
- Added dynamic inventory scripts for Ansible Tower and Red Hat Virtualization
- Added `awx_*` extra variables to job runs in addition to `tower_*`
- Added a setting for maximum user interface job events to show to Tower configuration
- Added support for setting the Azure Cloud Environment in Azure credentials
- Added retry for cleaning up job artifacts from isolated nodes
- Added python-crypto requirement to RPM packaging for GCE inventory script
- Added rsync requirement to RPM packaging for isolated nodes
- Added error handling in installation for PostgreSQL 9.4 to 9.6 migration failures
- Removed unused CALLBACK_CONNECTION, CALLBACK_QUEUE, and JOB_CALLBACK_DEBUG environment variables from the job environment
- Fixed multiple issues where survey passwords were not properly encrypted in the database
- Fixed an issue where cleanup jobs could run slowly and exhaust system memory when large job output was present
- Fixed an issue where cleanup jobs could fail due to a race condition
- Fixed an issue where use of *remove: True* and *remove_users: True* in LDAP configuration would cause an excessive number of activity stream entries
- Fixed an issue where the GCE inventory script would erroneously cache information
- Fixed an issue when using Ipsilon as a SAML IdP
- Fixed an issue when using SAML authentication behind a load-balancer
- Fixed an issue where '+' in a search string was not handled properly
- Fixed an issue where non-alphanumeric characters were stripped from SAML usernames
- Fixed an issue where credential_type information appeared in `api/v1` output
- Fixed a styling issue for Host Config Key in the Job Template display
- Fixed an issue where it was impossible to remove an organization from a credential
- Fixed an issue where `overwrite_vars` on an inventory source would overwrite inventory toplevel variables
- Fixed an issue where some credential kinds were not properly shown in the user interface
- Fixed calculation of isolated instance capacity
- Fixed an issue where the 'Workflow Editor' and 'Survey Editor' buttons were incorrectly shown in some states

- Fixed navigation to additional pages of hosts in the Smart Inventory view
- Fixed an issue where CloudForms inventory would not work with process isolation
- Fixed an issue where job output would not properly word wrap
- Fixed a migration issue with unicode inventory source names
- Fixed an issue when launching an ad-hoc command with forbidden extra variables
- Fixed an issue with symlinked manual projects when used with process isolation
- Fixed an issue where some host_filter queries could not be removed
- Fixed an issue where non-ascii characters could not be used in a LDAP bind DN
- Fixed sizing of the ad-hoc command launch dialog
- Fixed an issue where https://github.com/ansible/ansible/issues/30064 would prevent project sync
- Fixed an issue where a Smart Inventory host_filter query would be improperly encoded when saved
- Fixed month name on dashboard chart
- Fixed scheduling error when browser is in UTC timezone
- Fixed autocompletion of SCM inventory file dropdown
- Fixed modal state handling when a modal dialog was closed by clicking outside of it
- Fixed assorted migration errors on upgrade
- Fixed a user interface error when rapidly deleting inventory groups
- Fixed an issue where the system auditor would get a 404 error when viewing job results
- Fixed assorted issues when cascading job cancellation to dependent jobs
- Fixed opacity of disabled 'Run Commands' and 'Smart Inventory' buttons
- Fixed 'total_hosts' field of Smart Inventories
- Fixed virtualenv paths in sosreport plugins
- Fixed installation with Ansible 2.2
- Fixed ownership on ha.py on installation
- Fixed django superuser check in installation
- Fixed setting of custom RabbitMQ AMQP ports during installation
- Fixed an issue where LDAP authentication could timeout or cause a Tower error
- Improved callback worker's ability to deal with idle or disconnected database connections
- Improved activity stream output for Tower configuration changes
- Improved deletion of inventory sources to properly delete imported hosts and groups
- Improved various error messages
- Improved initial zoom setting of workflow view
- Improved inline help popovers for credential types
- Improved configuration for SSH key handling for isolated nodes. This is now configurable during setup
- Improved preflight checks for cluster installation
- Improved backup/restore playbooks to be cluster-aware

- Improved error handling in backup/restore playbooks
- Updated translations for Dutch, French, Japanese, and Spanish

1.4 Ansible Tower Version 3.2.1

- Added support to enforce Tower software version consistency across clustered environments
- Fixed an issue where, when using Tower 3.2.0 + Ansible 2.4.0, creating a Job Template that used an inventory with fact caching enabled could cause the job to run against a host which should have been removed
- Fixed a problem where ad-hoc permissions could be used to run commands against the Tower server
- Fixed an issue where the migration of scan jobs failed due to an organization having a unicode character in the name
- Fixed an issue where database migrations failed for upgrades

1.5 Ansible Tower Version 3.2.0

- Removed system tracking data (historical facts) feature starting with Ansible Tower 3.2. However, you can collect facts by using the fact caching feature. Refer to Fact Caching for more detail.
- Removed system tracking views in favor of directly viewing facts on hosts. Comparisons are best done with external data analytics systems.
- Removed Rackspace as a supported inventory source type and credential type.
- Removed the storing of `ansible_env` in job event data.
- Removed Job launching capability from `/api/v2/jobs`. Job template launching and job relaunching are the only support launch options.
- Deprecated the `group` field for InventorySource, which has been renamed to `deprecated_group` and will be removed from InventorySource completely in Tower 3.3. As a result, the related field on Group, `inventory_source` has been renamed `deprecated_inventory_source` and will also be removed in Ansible Tower 3.3.
- Deprecated requirement that inventory sources be associated with a group.
- Deprecated the `/api/v1` heirarchy with the introduction of `/api/v2`. `/api/v1` will be removed in a future Ansible Tower release to be determined.
- Deprecated the `/api/v2/authtoken` endpoint, which will be removed in Ansible Tower 3.3.
- Updated the job environment variables for AWS credentials. Refer to Amazon Web Services section of the *Ansible Tower User Guide* for new variable names.
- Added support for connecting to external log aggregators via direct TCP and UDP connections.
- Added the ability to test logging configurations through the Configure Tower UI.
- Updated the Ansible Tower Rest API to version 2 which include added endpoints: `instances`, `instance_groups`, `credential_types`, and `inventory_sources`.
- Added ability to create inventory sources and create Smart Inventories.
- Added the ability to access Tower resources via resource-specific human-readable identifiers.
- Added the ability to create and modify credential types.

- Added ability to create and modify instance groups and isolated nodes.
- Added the ability to enable and disable SSL certification verification through the Configure Tower UI. You no longer have to manually set an environment variable in your local `settings.py` file to achieve this.
- Updated upstream Azure libraries will require users who use Ansible Tower with Azure to use Ansible 2.4 or later.
- Fixed an outstanding issue regarding variable precedence so that the variable value is derived from the survey (survey variables take precedence over Job Template variables).
- Added Insights project remediation, which allows you to run the Insights maintenance plan associated with an inventory.
- Added a new API endpoint - `/api/v2/settings/logging/test/` - for testing external log aggregrator connectivity.
- Updated passing `-e create_preload_data=False` to skip creating default organization/project/inventory/credential/job_template during Tower installation.
- Added support for sourcing inventory from a file inside of a source control project.
- Added support for custom cloud and network credential types, which give you the ability to modify environment variables, extra vars, and generate file-based credentials (such as file-based certificates or .ini files) at `ansible-playbook` runtime.
- Added support for assigning multiple cloud and network credential types on job templates. Job templates can now prompt for "extra credentials" at launch time in the same manner as promptable machine credentials.
- Updated custom inventory sources to now specify a `Credential`; you can store third-party credentials encrypted within Tower and use their values from within your custom inventory script (for example - by reading an environment variable or a file's contents).
- Added support for configuring groups of instance nodes to run tower jobs. Instance groups can be assigned to an organization, inventory, or job template.
- Fixed an issue installing Tower on multiple nodes where cluster internal node references are used.
- Updated Tower to now use a modified version of [Fernet](https://github.com/fernet/spec/blob/master/Spec.md) for encrypting sensitive fields such as credentials. Our *Fernet256* class uses *AES-256-CBC* instead of *AES-128-CBC* for all encrypted fields.
- Added the ability to set custom environment variables globally for all playbook runs, inventory updates, project updates, and notification sending, via AWX_TASK_ENV configuration setting.
- Added –diff mode to Job Templates and Ad-Hoc Commands. The diff can be found in the standard out when diff mode is enabled.
- Added support for accessing some Tower resources via their name-related unique identifiers apart from primary keys.
- Added support for authentication to Tower via TACACS+.
- Updated names of tower-mange commands `register_instance` -> `provision_instance`, `deprovision_node` -> `deprovision_instance`, and `instance_group_remove` -> `remove_from_queue`, with backward compatibility support for 3.1 command names.
- Improved handling of workflow logic errors.
- Updated Azure bindings, and therefore, removed support for the old Azure classic modules.
- Fixed system auditor permissions.
- Updated Tower to explicitly prevent non-json bodies from being accepted in the API.

- Improved handling of default values in Tower Configuration.
- Improved handling of sensitive environment variables in job details.
- Added the ability to set the system auditor with `AUTH_LDAP_USER_FLAGS_BY_GROUP`.
- Fixed some minor UTF-8 handling issues.
- Fixed the system to no longer allow using password fields with the `order_by` query parameter in the API.
- Improved censoring of Ansible `no_log` in job output.
- Fixed handling project repository URLs with spaces and special characters.
- Improved explanation when canceling jobs that are dependencies of other jobs.
- Updated the `ansible-playbook` parameters to pass through the `setup.sh` script.
- Added translations for Dutch; updated translations for Japanese, French, and Spanish.
- Improved ability to update org admin/member roles on the user detail page.
- Added force shutdown of cluster nodes that are not at the same version as the rest of the cluster.
- Added configuration options in Tower Configuration UI.
- Updated Postgres to 9.6.
- Updated Tower by separating Vault credentials from machine credentials.
- Added more prompting options to job templates.
- Added the ability to prevent IDP user from assuming a local admin role.
- Improved the display of SCM revision hashes by abbreviating them, and added ability to easily copy revision to clipboard.
- Fixed a potential issue showing encrypted values in the activity stream instead of obfuscation characters.
- Added the ability to set an enabled/disabled flag on all supported cloud inventory sources.
- Added support for vmware `host_filters` and `groupby_patterns`.
- Fixed an issue where Tower wouldn't redirect the user to the right URL after clicking a link and logging in.
- Fixed tower to preserve `stderr` from custom inventory scripts.
- Updated Tower to now act as a fact cache source for jobs.
- Improved handling of related resources when inventories are deleted.
- Added the ability to show an indicator during background inventory delete.
- Updated supported cloud regions for some inventories.
- Improved SAML configurations.
- Improved LDAP settings validation.
- Added support for providing SSL cert for log aggregator service.
- Added the ability to set proxy IP whitelists for trusted vs. untrusted load balancers.
- Improved the efficiency in generating entries in the activity stream.
- Added support for upgrading Ansible during setup playbook run (`-e upgrade_ansible_with_tower=1`).
- Fixed downloading ad-hoc command stdout.
- Fixed job launch dependency handling.

- Fixed some xss vulnerabilities.
- Added runas privilege escalation support.
- Improved handling of instance capacity calculation.
- Fixed SSL certificate handling for LDAP.
- Updated the Job detail event modals to now be resizeable.
- Improved yaml/json editor views.
- Improved job list performance.

CHAPTER

TWO

UPGRADING ANSIBLE TOWER

Topics:

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.

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 have been using Ansible version 2.5 with Ansible Tower version 3.2.0, 3.2.1, or 3.2.2, you may find that cloud inventory syncs have placed ordinary group or inventory variables inside of `hostvars` of member hosts. After upgrading to Ansible Tower 3.2.3, the variables will be returned inside of their respective groups or inside of the inventory variables. Since `hostvars` take precedence, variables that come from pre-migration updates may take precedence over their newer values, resulting in a variable precedence conflict. Refer to *Remediation Steps for Upgrade Variable Precedence Conflict* for a workaround.
- 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.

2.1.1 Remediation Steps for Upgrade Variable Precedence Conflict

If you encounter a variable precedence conflict after upgrading to Ansible Tower 3.2.3, you can resolve this conflict by temporarily setting the inventory source's `overwrite_vars` field to `True` and running an update. Apply this setting to all inventory sources that show the issue.

Inventories sourced from a project already require `overwrite_vars` to be set to `True` so they are not affected.

2.2 Obtaining Ansible Tower

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

Example Single Node Inventory File

```
[tower]
localhost ansible_connection=local

[database]

[all:vars]
admin_password='password'

pg_host=''
pg_port=''

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

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

# Needs to be true for fqdns and ip addresses
rabbitmq_use_long_name=false
# Needs to remain false if you are using localhost
```

Example Multi Node Cluster Inventory File

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

```
clusternode2.example.com
clusternode3.example.com

[database]
dbnode.example.com

[all:vars]
ansible_become=true

admin_password='password'

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

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

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

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

Example Inventory file for an external existing database

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

[database]

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


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

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

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.

CHAPTER

THREE

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.

CHAPTER

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.

CHAPTER

FIVE

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.

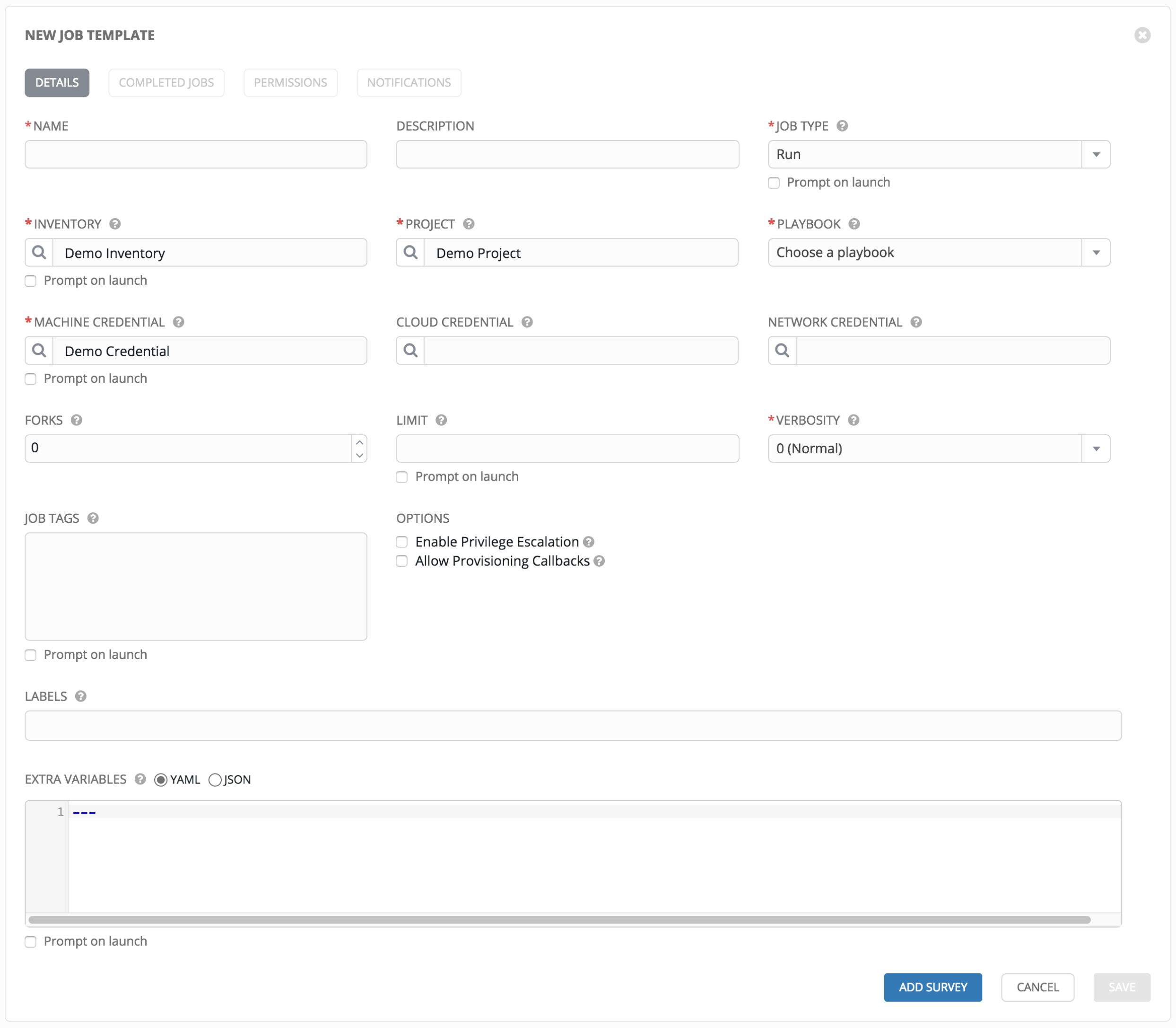
5.2 Prompt on Launch

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.

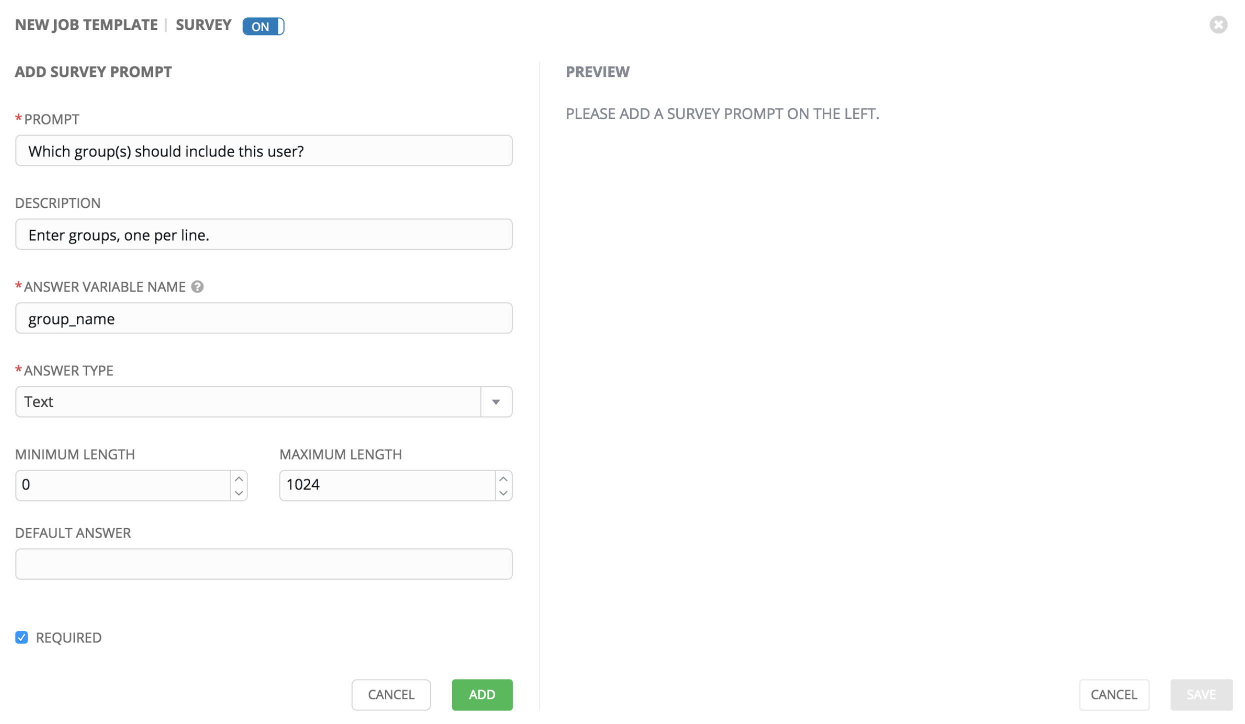
Users that are job template administrators can make changes to job templates; however, to make changes to the inventory, project, playbook, or credentials used in the job template, the user must also have the "Use" role for the project, inventory, and all credentials currently being used or being set.

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 simplier 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 reposnses.


NEW JOB TEMPLATE | SURVEY ON
ADD SURVEY PROMPT
PREVIEW
PLEASE ADD A SURVEY PROMPT ON THE LEFT.
CANCEL
ADD
CANCEL
SAVE

*PROMPT: Which group(s) should include this user?

DESCRIPTION: Enter groups, one per line.

*ANSWER VARIABLE NAME: group_name

*ANSWER TYPE: Text

MINIMUM LENGTH: 0

MAXIMUM LENGTH: 1024

DEFAULT ANSWER:

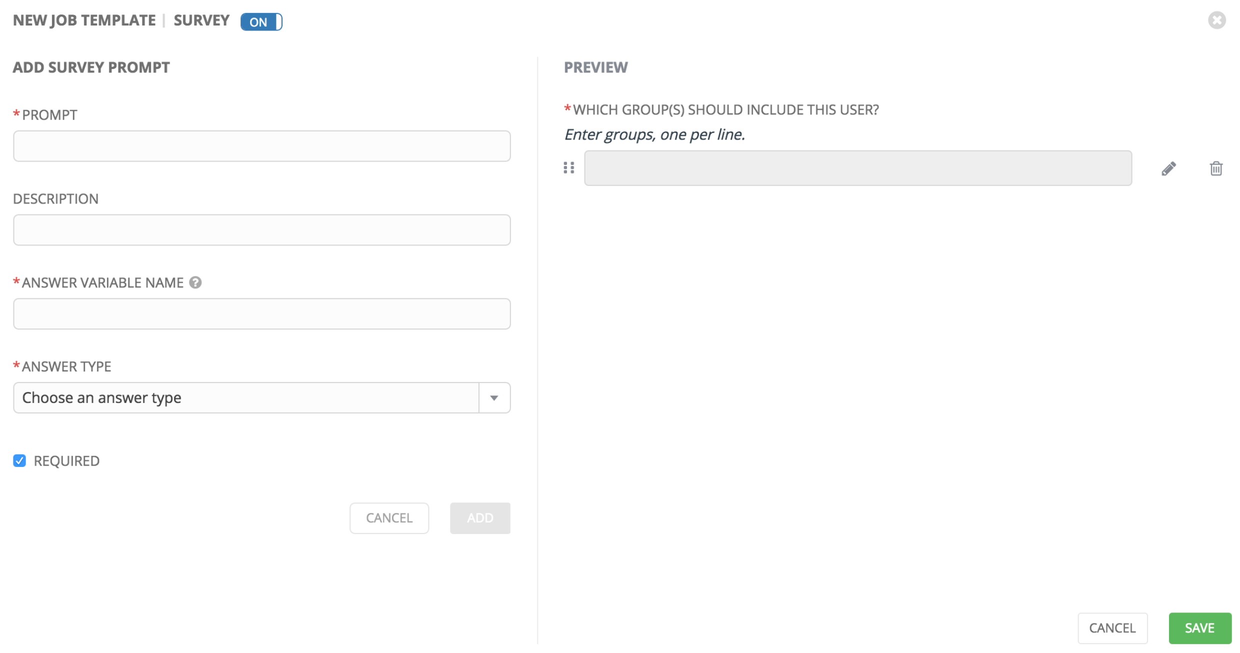
☑ REQUIRED

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.



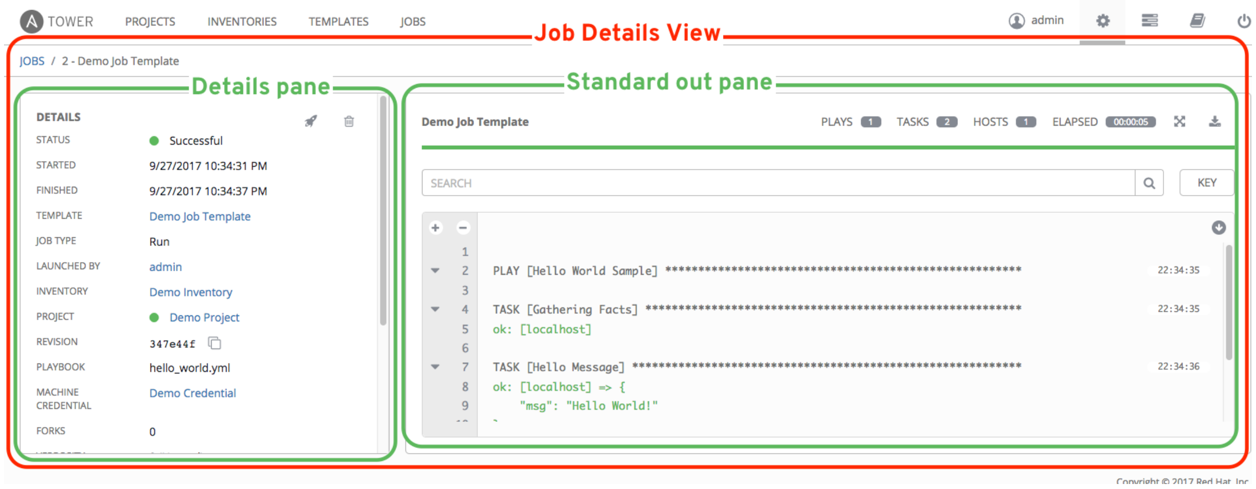
CHAPTER

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.



DETAILS	
STATUS	● Successful
STARTED	9/27/2017 10:34:31 PM
FINISHED	9/27/2017 10:34:37 PM
TEMPLATE	Demo Job Template
JOB TYPE	Run
LAUNCHED BY	admin
INVENTORY	Demo Inventory
PROJECT	● Demo Project
REVISION	347e44f
PLAYBOOK	hello_world.yml
MACHINE CREDENTIAL	Demo Credential
FORKS	0

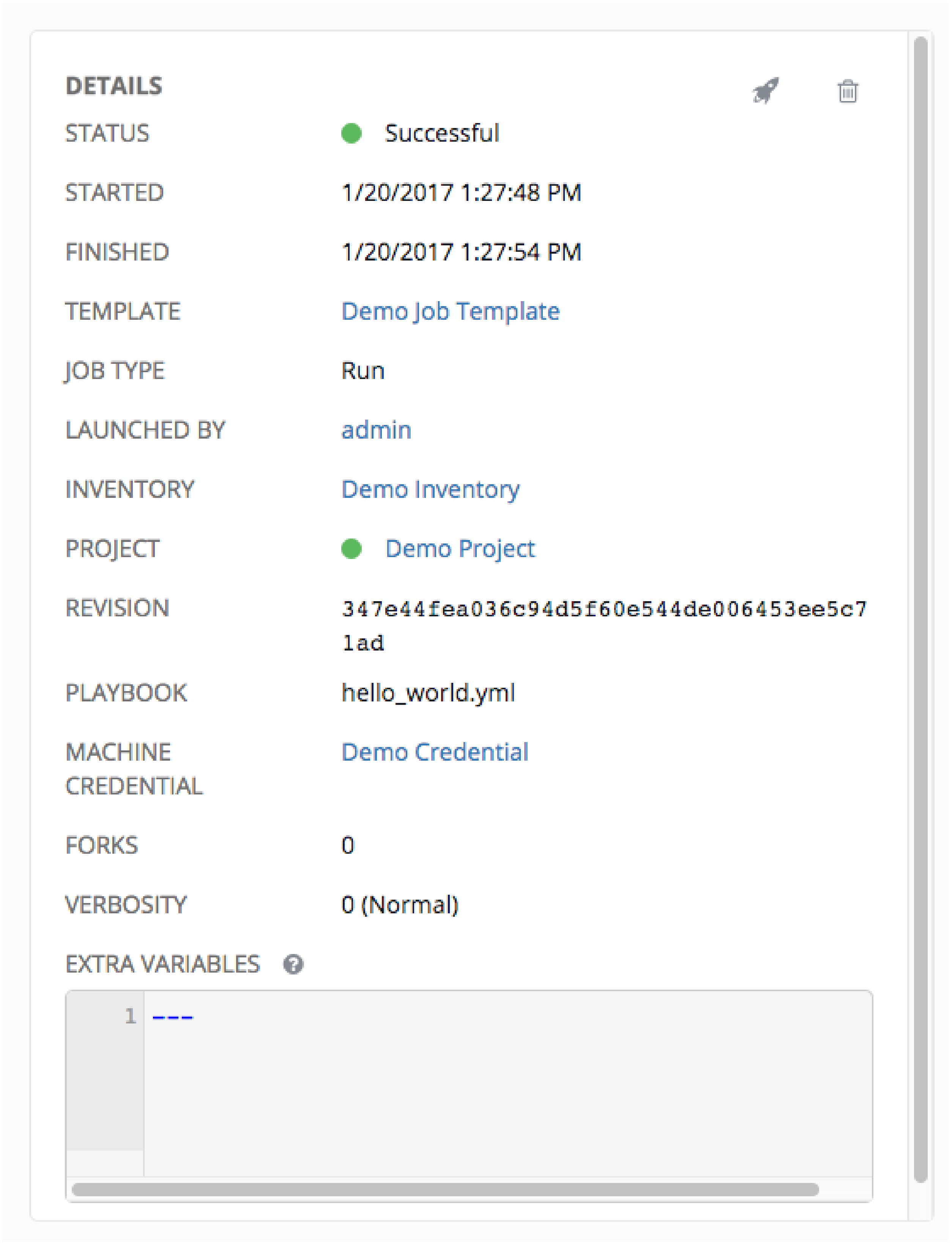
```

PLAY [Hello World Sample] *****************************************************     22:34:35

TASK [Gathering Facts] ********************************************************     22:34:35
ok: [localhost]

TASK [Hello Message] **********************************************************     22:34:36
ok: [localhost] => {
    "msg": "Hello World!"
```

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

```
---
```

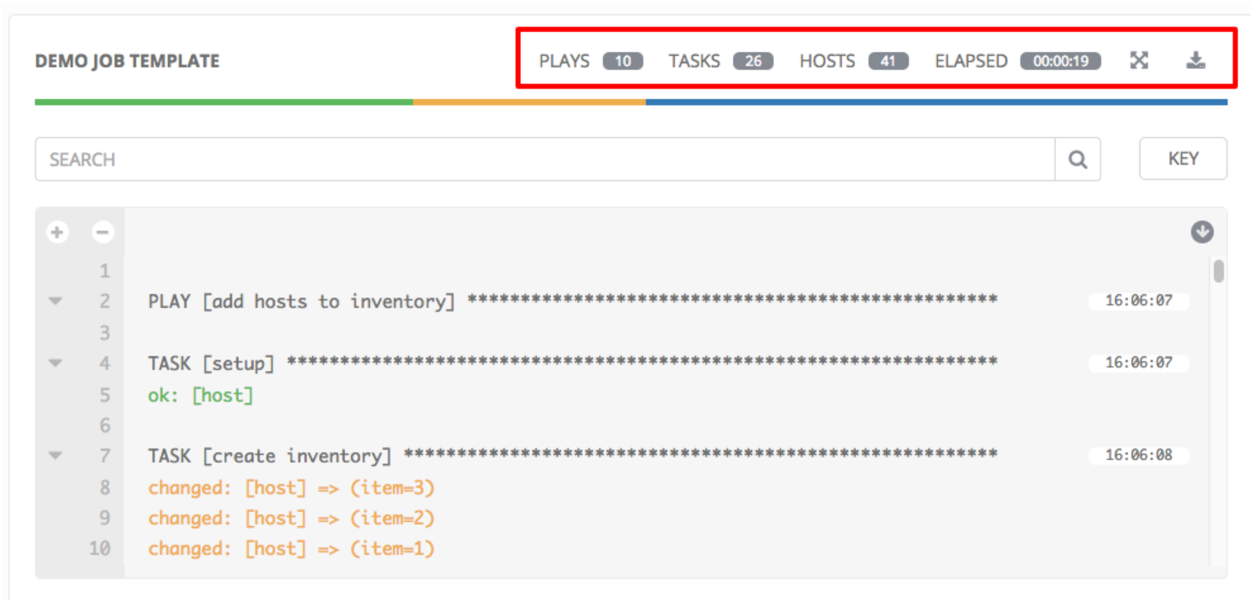
6.2 Standard Out Pane

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:

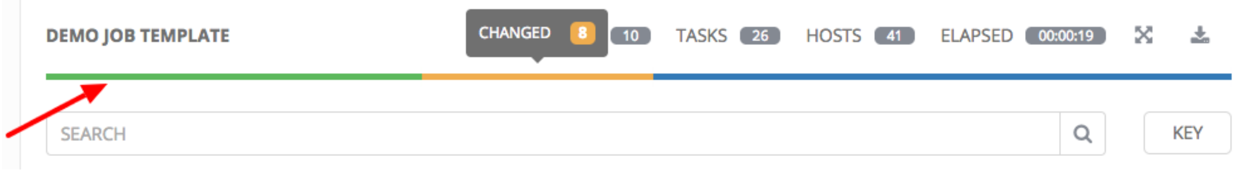
- the number of plays
- the number of tasks
- the number of hosts
- the elapsed time to run the job template



```
PLAY [add hosts to inventory] *****************************************************   16:06:07

TASK [setup] **********************************************************************   16:06:07
ok: [host]

TASK [create inventory] ***********************************************************   16:06:08
changed: [host] => (item=3)
changed: [host] => (item=2)
changed: [host] => (item=1)
```

6.2.2 Host Status Bar

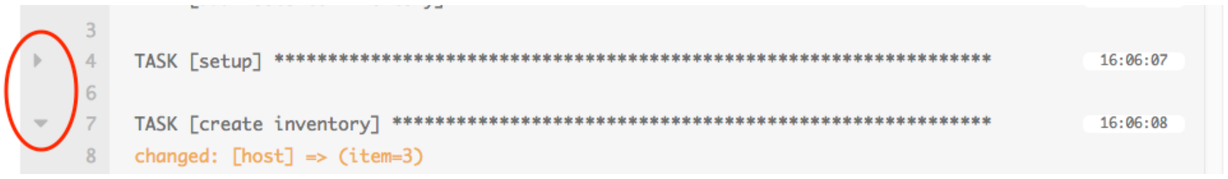
The host status bar runs across the top of the **Standard Out** pane. Hover over a section of the host status bar and the number of hosts associated with that particular status displays.



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.

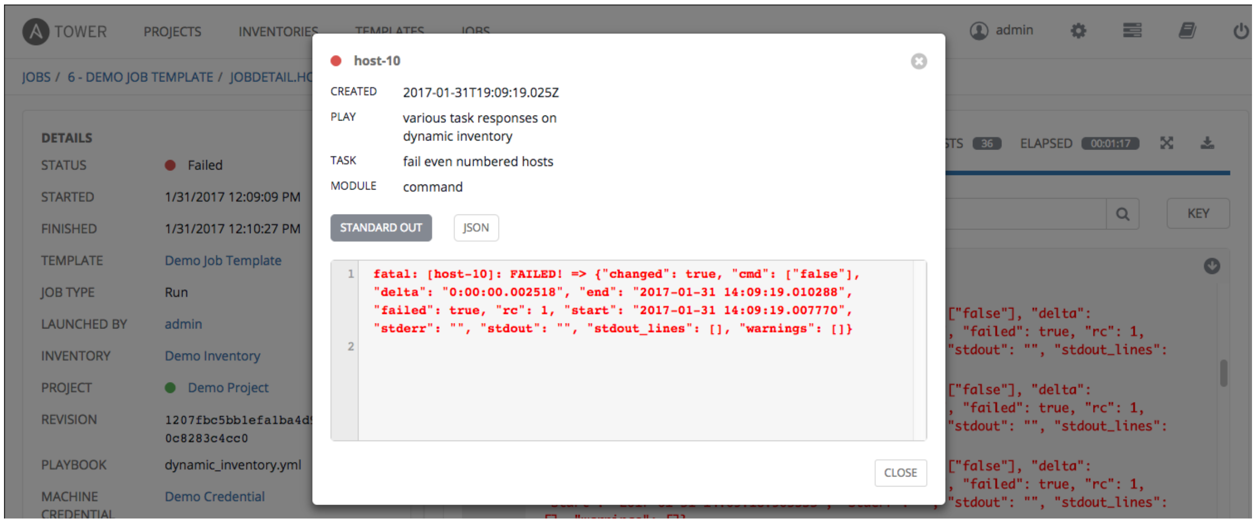


```
TASK [setup] *********************************************************************    16:06:07

TASK [create inventory] **************************************************************    16:06:08
changed: [host] => (item=3)
```

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:



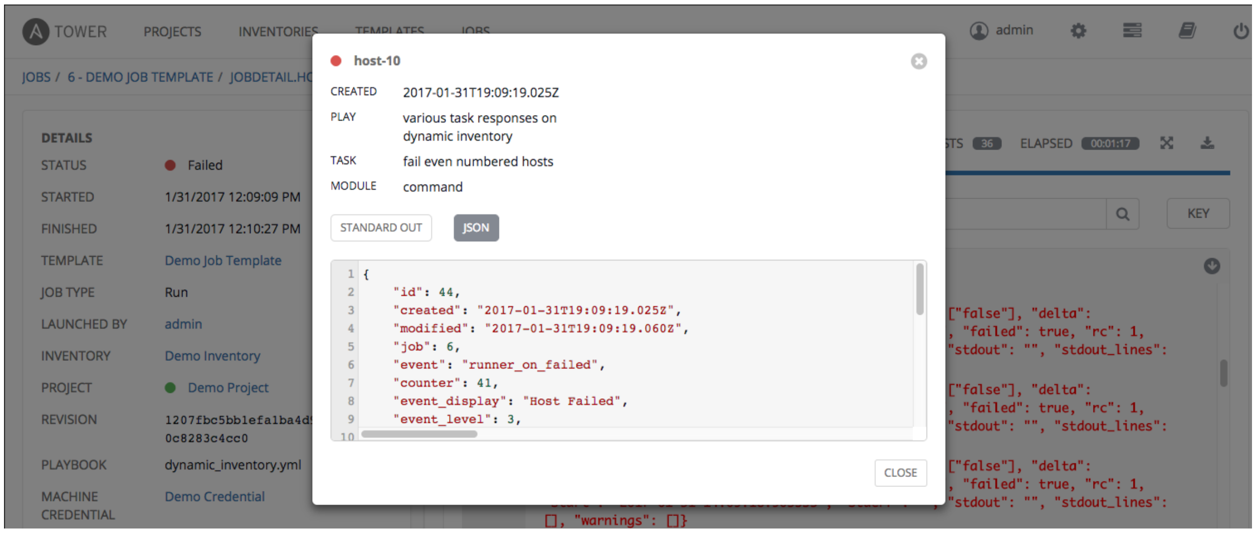
DETAILS	
STATUS	Failed
STARTED	1/31/2017 12:09:09 PM
FINISHED	1/31/2017 12:10:27 PM
TEMPLATE	Demo Job Template
JOB TYPE	Run
LAUNCHED BY	admin
INVENTORY	Demo Inventory
PROJECT	Demo Project
REVISION	1207fbc5bb1efa1ba4d 0c8283c4cc0
PLAYBOOK	dynamic_inventory.yml
MACHINE CREDENTIAL	Demo Credential

host-10

CREATED	2017-01-31T19:09:19.025Z
PLAY	various task responses on dynamic inventory
TASK	fail even numbered hosts
MODULE	command

```
fatal: [host-10]: FAILED! => {"changed": true, "cmd": ["false"],
"delta": "0:00:00.002518", "end": "2017-01-31 14:09:19.010288",
"failed": true, "rc": 1, "start": "2017-01-31 14:09:19.007770",
"stderr": "", "stdout": "", "stdout_lines": [], "warnings": []}
```

To view the results in JSON format, click on the **JSON** tab.



DETAILS

STATUS	● Failed
STARTED	1/31/2017 12:09:09 PM
FINISHED	1/31/2017 12:10:27 PM
TEMPLATE	Demo Job Template
JOB TYPE	Run
LAUNCHED BY	admin
INVENTORY	Demo Inventory
PROJECT	● Demo Project
REVISION	1207fbc5bb1efa1ba4d 0c8283c4cc0
PLAYBOOK	dynamic_inventory.yml
MACHINE CREDENTIAL	Demo Credential

● host-10

CREATED	2017-01-31T19:09:19.025Z
PLAY	various task responses on dynamic inventory
TASK	fail even numbered hosts
MODULE	command

```
{
    "id": 44,
    "created": "2017-01-31T19:09:19.025Z",
    "modified": "2017-01-31T19:09:19.060Z",
    "job": 6,
    "event": "runner_on_failed",
    "counter": 41,
    "event_display": "Host Failed",
    "event_level": 3,
```

CHAPTER

SEVEN

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.

Note: It is highly recommended that you run `umask` 0022 before installing any packages to the virtual environment, such a python package after initial install. Failure to properly configure permissions can result in Tower service failures. An example as follows:

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

7.1 Modifying the virtualenv

Modifying the virtualenv used by Tower is unsupported and not recommended. Instead, you can add modules to the virtualenv that Tower uses to run Ansible.

To do so, activate the Ansible virtualenv:

```
. /var/lib/awx/venv/ansible/bin/activate
```

...and then install whatever you need using `pip`:

```
pip install mypackagename
```

CHAPTER

EIGHT

INDEX

- genindex

CHAPTER

NINE

COPYRIGHT © 2018 RED HAT, INC.

INDEX