Ansible Automation Platform Upgrade and Migration

Release Automation Controller 4.0.0

Red Hat, Inc.

Dec 09, 2022

CONTENTS

Thank you for your interest in Red Hat Ansible Automation Platform controller. automation controller 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 automation controller to automation controller 3.8 before you can upgrade to automation controller 4.0.

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!

Automation Controller Version 4.0.0; July 15, 2021; https://access.redhat.com/

CHAPTER

ONE

RELEASE NOTES FOR AUTOMATION CONTROLLER VERSION 4.0.0

Introduced

- Support for automation execution environments. All automation now runs in execution environments via containers, either directly via OpenShift, or locally via podman
- New PatternFly 4 based user-interface for increased performance, security, and consistency with other Ansible Automation Platform components

Added

- Added identity provider support for GitHub Enterprise
- Support for RHEL system crypto profiles to nginx configuration
- The ability to disable local system users and only pull users from configured identity providers
- Additional Prometheus metrics for tracking job event processing performance
- New `awx-manage` command for dumping host automation information
- Red Hat Insights as an inventory source
- Ability to set server-side password policies using Django's `AUTH_PASSWORD_VALIDATORS` setting
- Support for Centrify Vault as a credential lookup plugin
- Support for namespaces in Hashicorp Vault credential plugin

Updated

- OpenShift deployment to be done via an Operator instead of a playbook
- Python used by application to Python 3.8
- Nginx used to version 1.18
- PostgreSQL used to PostgreSQL 12, and moved to partitioned databases for performance
- The "container groups" feature to general availability from Tech Preview; now fully utilizes execution environments
- Insights remediation to use new Red Hat Insights inventory source rather than utilizing scan playbooks with arbitrary inventory
- Subscriptions display to count hosts automated on instead of hosts imported
- Inventory source, credential, and Ansible content collection to reference *controller* instead of *tower*

Deprecated

- None

Removed

- Support for deploying on CentOS (any version) and RHEL 7
- Support for Mercurial projects
- Support for custom inventory scripts stored in controller (use `awx-manage export_custom_scripts` to export them)
- Resource profiling code (`AWX_RESOURCE_PROFILING_*`)
- Support for custom Python virtual environments for execution. Use new `awx-manage` tools for assisting in migration
- Top-level /api/v2/job_events/ API endpoint
- The ability to disable job isolation

CHAPTER

TWO

UPGRADING TO ANSIBLE AUTOMATION PLATFORM

Automation Hub acts as a content provider for automation controller, which requires both an automation controller deployment and an Automation Hub deployment running alongside each other. The Ansible Automation Platform installer contains both of these. This section covers each component of the upgrading process:

Note: All upgrades should be no more than two major versions behind what you are currently upgrading to. For example, in order to upgrade to automation controller 4.0, you must first be on version 3.8.x; i.e., there is no direct upgrade path from version 3.7.x or earlier. Refer to the recommended upgrade path article on the Red Hat customer portal.

In order to run automation controller 4.0, you must also have Ansible 2.10.

2.1 Upgrade Planning

This section covers changes that you should keep in mind as you attempt to upgrade your automation controller instance.

- Even if you already have a valid license from a previous version, you must still provide your credentials or a subscriptions manifest again upon upgrading to the latest automation controller. See Import a Subscription in the *Automation Controller User Guide*.
- If you need to upgrade Red Hat Enterprise Linux and automation controller, you will need to do a backup and restore of your controller data (from the automation controller). Refer to Backing Up and Restoring in the *Automation Controller Administration Guide* for further detail.
- Clustered upgrades require special attention to instance and instance groups prior to starting the upgrade. See Editing the Red Hat Ansible Automation Platform installer inventory file and Clustering for details.

2.2 Obtaining the Installer

Refer to Choosing and obtaining a Red Hat Ansible Automation Platform installer on the Red Hat Customer Portal for detail. Be sure to use your Red Hat customer login to access the full content.

2.3 Setting up the Inventory File

See Editing the Red Hat Ansible Automation Platform installer inventory file for information.

2.4 Running the Setup Playbook

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
```

CHAPTER

THREE

UPGRADING TO EXECUTION ENVIRONMENTS

If upgrading from older versions of automation controller to 4.0 or later, the controller has the ability to detect previous versions of virtual environments associated with Organizations, Inventory, and Job Templates; and inform you that you will need to migrate to the new execution environment model. A brand new installation of automation controller creates two virtualenvs during installation–one is used to run the controller itself, while the other is used to run Ansible. Like legacy virtual environments, execution environments allow the controller to run in a stable environment, while allowing you to add or update modules to your execution environment as necessary to run your playbooks. For more information, see Execution Environments in the *Automation Controller User Guide*.

3.1 Migrate legacy venvs to execution environments

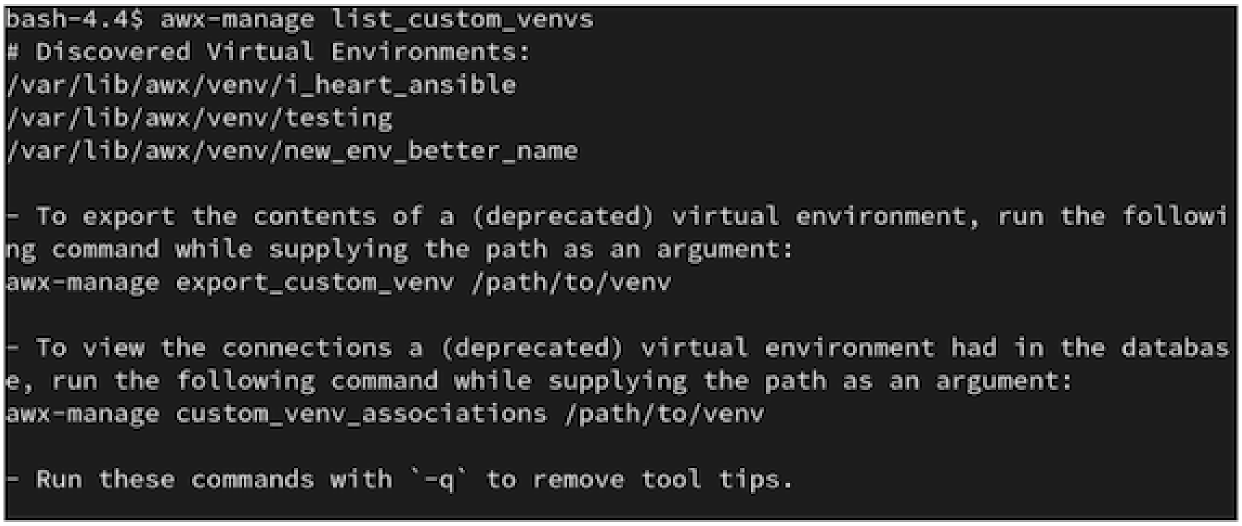
You can have the exact same setup in an execution environment that you had in a prior custom virtual environment by migrating them to the new execution environment. Use the `awx-manage` commands in this section to:

- list of all the current custom virtual environments and their paths (`list_custom_venvs`)
- view the resources that rely a particular custom virtual environment (`custom_venv_associations`)
- export a particular custom virtual environment to a format that can be used to migrate to an execution environment (`export_custom_venv`)

1. Before you migrate, it is recommended that you view all the custom virtual environments you currently have running by using the `awx-manage list` command:

```
$ awx-manage list_custom_venvs
```

Below is an example output when running this command:



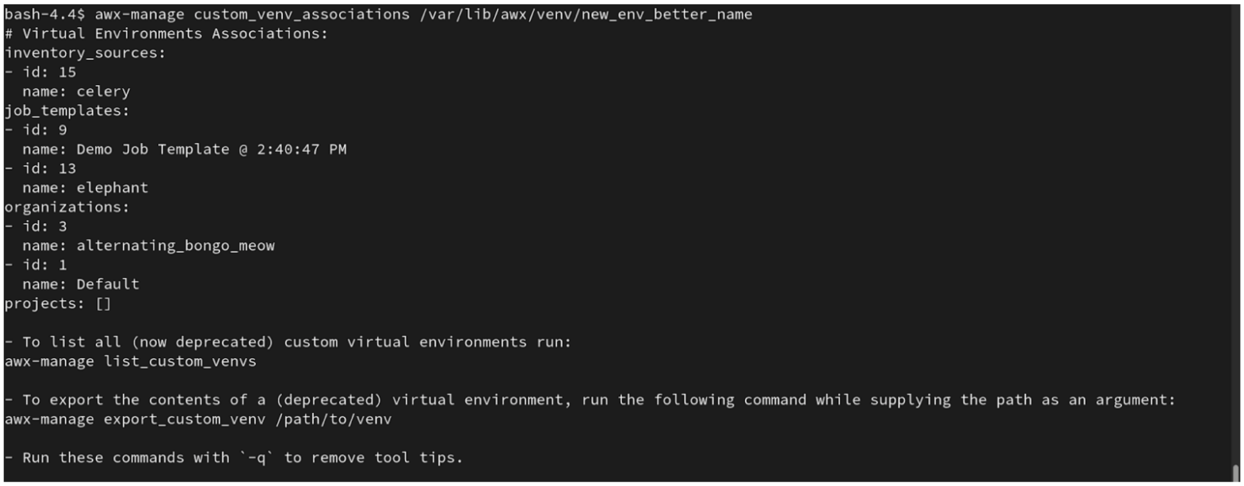
```
bash-4.4$ awx-manage list_custom_venvs
# Discovered Virtual Environments:
/var/lib/awx/venv/i_heart_ansible
/var/lib/awx/venv/testing
/var/lib/awx/venv/new_env_better_name

- To export the contents of a (deprecated) virtual environment, run the followi
ng command while supplying the path as an argument:
awx-manage export_custom_venv /path/to/venv

- To view the connections a (deprecated) virtual environment had in the databas
e, run the following command while supplying the path as an argument:
awx-manage custom_venv_associations /path/to/venv

- Run these commands with `-q` to remove tool tips.
```

The above output shows three custom virtual environments and their paths. If you have a custom virtual environment that is not located within the default `/var/lib/awx/venv/` directory path, it will not be included here.

2. Use the `_associations` command to view what organizations, jobs, and inventory sources a custom virtual environment is associated with in order to determine which resources rely on them:

```
$ awx-manage custom_venv_associations /this/is/the/path/
```

Below is an example output when running this command:

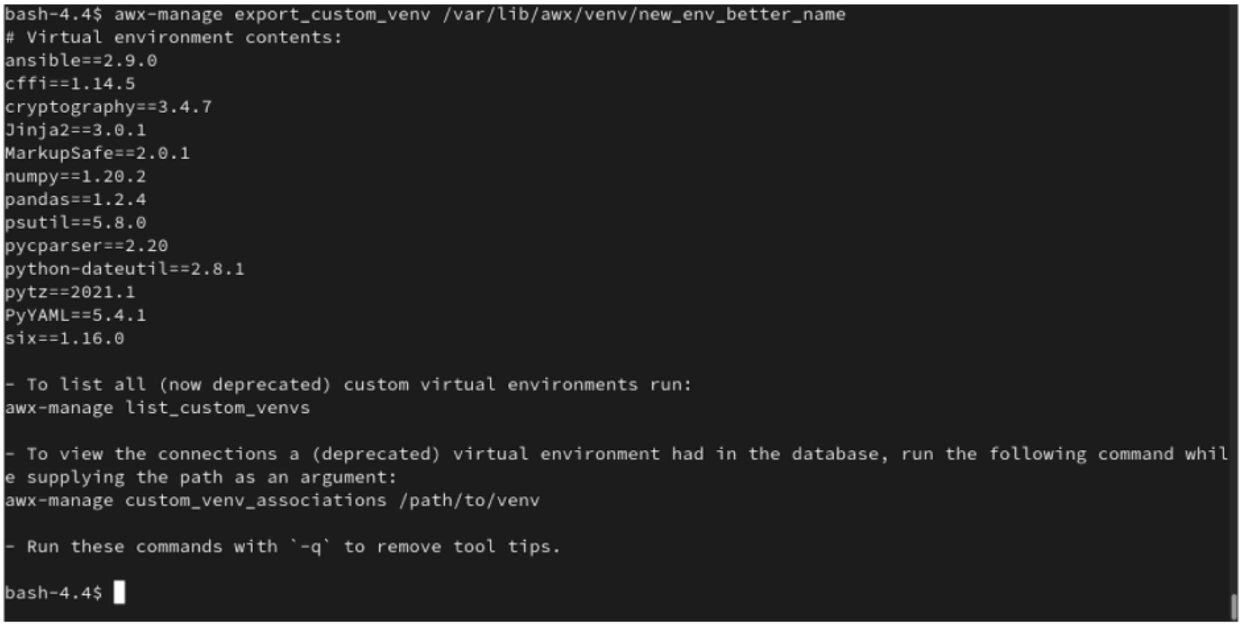


```
bash-4.4$ awx-manage custom_venv_associations /var/lib/awx/venv/new_env_better_name
# Virtual Environments Associations:
inventory_sources:
- id: 15
  name: celery
job_templates:
- id: 9
  name: Demo Job Template @ 2:40:47 PM
- id: 13
  name: elephant
organizations:
- id: 3
  name: alternating_bongo_meow
- id: 1
  name: Default
projects: []

- To list all (now deprecated) custom virtual environments run:
awx-manage list_custom_venvs

- To export the contents of a (deprecated) virtual environment, run the following command while supplying the path as an argument:
awx-manage export_custom_venv /path/to/venv

- Run these commands with `-q` to remove tool tips.
```

3. Select a path for the virtual environment that you want to migrate and specify it in the `awx-manage export` command:

```
$ awx-manage export_custom_venv /this/is/the/path/
```

The resulting output is essentially the results of executing a `pip freeze` command. The example shows the contents of the selected custom virtual environment:



```
bash-4.4$ awx-manage export_custom_venv /var/lib/awx/venv/new_env_better_name
# Virtual environment contents:
ansible==2.9.0
cffi==1.14.5
cryptography==3.4.7
Jinja2==3.0.1
MarkupSafe==2.0.1
numpy==1.20.2
pandas==1.2.4
psutil==5.8.0
pycparser==2.20
python-dateutil==2.8.1
pytz==2021.1
PyYAML==5.4.1
six==1.16.0

- To list all (now deprecated) custom virtual environments run:
awx-manage list_custom_venvs

- To view the connections a (deprecated) virtual environment had in the database, run the following command whil
e supplying the path as an argument:
awx-manage custom_venv_associations /path/to/venv

- Run these commands with `-q` to remove tool tips.

bash-4.4$ 
```

Note: All of these commands can be run with a `-q` option, which removes the instructional content provided on each output.

Now that you have the output from this `pip freeze` data, you can paste it into a definition file that can be used to spin up your new execution environment using ansible-builder. Anyone (both normal users and admins) can use ansible-builder to create an execution environment. See Building an Execution Environment in the *Automation Controller User Guide* for further detail.

CHAPTER

FOUR

INDEX

- genindex

CHAPTER

FIVE

COPYRIGHT © RED HAT, INC.

Third Party Rights

INDEX