
Ansible Automation Platform Upgrade and Migration

Release Automation Controller 4.1.0

Red Hat, Inc.

Feb 10, 2023

CONTENTS

1	Release Notes for Automation Controller Version 4.1.0	2
2	Upgrading to Ansible Automation Platform	4
2.1	Upgrade Planning	4
2.2	Obtaining the Installer	5
2.3	Setting up the Inventory File	5
2.4	Running the Setup Playbook	5
3	Upgrading to Execution Environments	6
3.1	Migrate legacy venvs to execution environments	6
3.2	Migrate isolated instances to execution nodes	8
3.3	View mesh topology	10
4	Index	11
5	Copyright © Red Hat, Inc.	12
	Index	13

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 4.1.0; December 02, 2021; <https://access.redhat.com/>

RELEASE NOTES FOR AUTOMATION CONTROLLER VERSION 4.1.0

Introduced

- Connected Receptor nodes to form a control plane and execution mesh configurations
- The special `controlplane` instance group to allow for the task manager code to target an OpenShift Controller node to run the project update
- The ability to render a configured mesh topology in a graph in the installer
- Controller 4.1 execution nodes can be remote
- Node types for Controller 4.1 (`control`, `hybrid`, `execution`, `hop`, `control`, `hybrid`, `execution`, `hop`) installed for different sets of services and provide different capabilities, allowing for scaling nodes that provide the desired capability such as job execution or serving of web requests to the API/UI.

Added

- The ability for the platform installer to allow users to install execution nodes and express receptor mesh topology in the inventory file. The platform installer will also be responsible for deprovisioning nodes.
- Work signing to the receptor mesh so that control plane nodes have the exclusive authority to submit receptor work to execution nodes over the mesh
- Support for pre-population of execution environment name, description, and image from query parameters when adding a new execution environment in the Controller User Interface
- Ability to trigger a reload of the topology configuration in Receptor without interrupting work execution
- Using Public Key Infrastructure (PKI) for securing the Receptor mesh
- Added importing execution environments from Automation Hub into the controller to improve the platform experience

Updated

- The controller to support new controller control plane and execution mesh
- Task manager will only run project updates and system jobs on nodes with `node_type` of “control” or “hybrid”
- Task manager will only run jobs, inventory updates, and ad hoc commands on nodes with `node_type` of “hybrid” or “execution”
- Heartbeat and capacity check to work with Receptor execution nodes
- Reaper to work with the addition of execution nodes
- Controller User Interface to not show control instances as an option to associate with instance groups
- The Associate pop-up screen to display host names when adding an existing host to a group
- Validators for editing miscellaneous authentication parameters

- Advanced search key options to be grouped
- SAML variables default values
- Survey validation on Prompt on Launch
- Login redirect

Deprecated

- None

Removed

- The ability to delete the default instance group through the User Interface

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:

- *Upgrade Planning*
- *Obtaining the Installer*
- *Setting up the Inventory File*
- *Running the Setup Playbook*

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 4.3, you must first be on version 4.1.x; i.e., there is no direct upgrade path from version 3.8.x or earlier. Refer to the [recommended upgrade path article](#) on the Red Hat customer portal.

In order to run automation controller 4.3, you must also have Ansible 2.12 at minimum.

To help you determine the right upgrade or migration path when moving from an old Ansible Automation Platform or Tower version to a new Ansible Automation Platform version, use the Upgrade Assistant at <https://access.redhat.com/labs/aapua/>. If prompted, use your Red Hat customer credentials to login.

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.

You can also automatically generate an inventory file based on your selections using a utility called the Inventory File Generator, which you can access at <https://access.redhat.com/labs/aapifg/>. If prompted, use your Red Hat customer credentials to login.

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
```

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 following 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 database, 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 while
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.

3.2 Migrate isolated instances to execution nodes

The move from isolated instance groups to execution nodes enables inbound or outbound connections. Contrast this with versions 3.8 and older where only outbound connections were allowed from controller nodes to isolated nodes.

Migrating legacy isolated instance groups to execution nodes in order to function properly in the automation controller mesh architecture in 4.1, is a preflight function of the installer that essentially creates an inventory file based on your old file. Even though both `.ini` and `.yml` files are still accepted formats, the generated file output is only an `.ini` file at this time.

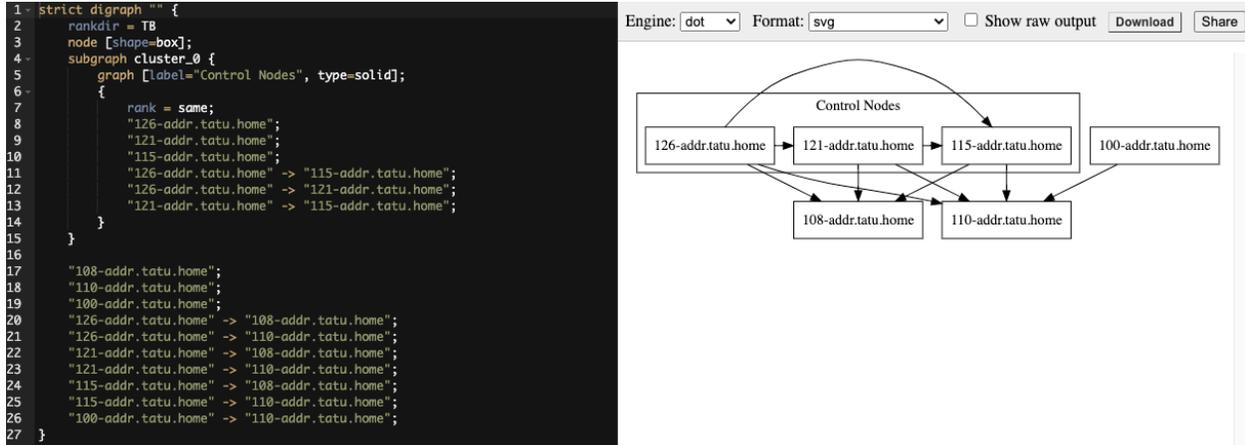
The preflight check leverages Ansible; and Ansible flattens the concept of children, this means that not every single inventory file can be replicated exactly, but it is very close. It will be functionally the same to Ansible, but may look different to you. The automated preflight processing does its best to create child relationships based on heuristics, but be aware that the tool lacks the nuance and judgment that human users have. Therefore, once the file is created, do **NOT** use it as-is. Check the file over and use it as a template to ensure that they work well for both you and the Ansible engine.

Here is an example of a before and after preflight check, demonstrating how Ansible flattens an inventory file and how the installer reconstructs a new inventory file. To Ansible, both of these files are essentially the same.

Old style (from Ansible docs)	New style (generated by installer)
<pre>[tower] localhost ansible_connection=local [database] [all:vars] admin_password='*****' pg_host='' pg_port='' pg_database='awx' pg_username='awx' pg_password='*****' rabbitmq_port=5672 rabbitmq_vhost=tower rabbitmq_username=tower rabbitmq_password='*****' rabbitmq_cookie=cookiemonster # Needs to be true for fqdns and ip_ ↪addresses rabbitmq_use_long_name=false [isolated_group_restrictedzone] isolated-node.c.towertest-188910.internal [isolated_group_restrictedzone:vars] controller=tower</pre>	<pre>[all:vars] admin_password='*****' pg_host='' pg_port='' pg_database='awx' pg_username='awx' pg_password='*****' rabbitmq_port=5672 rabbitmq_vhost='tower' rabbitmq_username='tower' rabbitmq_password='*****' rabbitmq_cookie='cookiemonster' rabbitmq_use_long_name='false' # In AAP 2.X [tower] has been renamed to_ ↪[automationcontroller] # Nodes in [automationcontroller] will_ ↪be hybrid by default, capable of_ ↪executing user jobs. # To specify that any of these nodes_ ↪should be control-only instead, give_ ↪them a host var of `node_type=control` [automationcontroller] localhost [automationcontroller:vars] # in AAP 2.X the controller variable has_ ↪been replaced with `peers` # which allows finer grained control_ ↪over node communication. # `peers` can be set on individual hosts, ↪to a combination of multiple groups_ ↪and hosts. peers='instance_group_restrictedzone' ansible_connection='local' # in AAP 2.X isolated groups are no_ ↪longer a special type, and should be_ ↪renamed to be instance groups [instance_group_restrictedzone] isolated-node.c.towertest-188910.internal [instance_group_restrictedzone:vars] # in AAP 2.X Isolated Nodes are_ ↪converted into Execution Nodes using_ ↪node_state=iso_migrate node_state='iso_migrate' # In AAP 2.X Execution Nodes have_ ↪replaced isolated nodes. All of these_ ↪nodes will be by default # `node_type=execution`. You can specify_ ↪new nodes that cannot execute jobs and_ ↪are intermediaries # between your control and execution_ ↪nodes by adding them to [execution_ ↪nodes] and setting a host var # `node_type=hop` on them. [execution_nodes] [execution_nodes:children] instance_group_restrictedzone</pre>

3.3 View mesh topology

If you configured a mesh topology, the installer can graphically validate your mesh configuration through a generated graph rendering tool. The graph is generated by reading the contents of the inventory file. See the [Red Hat Ansible Automation Platform automation mesh guide](#) for further detail.



Any given inventory file must include some sort of execution capacity that is governed by at least one control node. That is, it is unacceptable to produce an inventory file that only contains control-only nodes, execution-only nodes or hop-only nodes. There is a tightly coupled relationship between control and execution nodes that must be respected at all times. The installer will fail if the inventory files aren't properly defined. The only exception to this rule would be a single hybrid node, as it will satisfy the control and execution constraints.

In order to run jobs on an execution node, either the installer needs to pre-register the node, or user needs to make a PATCH request to `/api/v2/instances/N/` to change the enabled field to true.

- genindex

COPYRIGHT © RED HAT, INC.

Ansible, Ansible Automation Platform, 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

A

Ansible
 executing in a execution
 environment, 6

B

build
 execution environments, 6

E

executing in a execution environment
 Ansible, 6
execution environment, 6
execution environments
 build, 6
 mesh, 10

G

graph
 mesh, 10

I

installation script
 playbook setup, 5

M

mesh, 10
 execution environments, 10
 graph, 10
migrate to execution environments
 virtual environments, 6

P

playbook setup, 5
 installation script, 5
 setup.sh, 5

S

setup.sh
 playbook setup, 5

U

upgrade, 4
upgrade considerations, 4

V

virtual environments
 migrate to execution environments, 6