Red Hat Ansible Automation Platform 2.1

Red Hat Ansible Automation Platform Upgrade and Migration Guide

Upgrading to the latest version of Ansible Automation Platform and migrating legacy virtual environments to automation execution environments

Last Updated: 2022-05-13

Red Hat Ansible Automation Platform 2.1 Red Hat Ansible Automation Platform Upgrade and Migration Guide

Upgrading to the latest version of Ansible Automation Platform and migrating legacy virtual environments to automation execution environments

Legal Notice

Copyright © 2022 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at
http://creativecommons.org/licenses/by-sa/3.0/
. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux ® is the registered trademark of Linus Torvalds in the United States and other countries.

Java ® is a registered trademark of Oracle and/or its affiliates.

XFS ® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL ® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js ® is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack ® Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

Abstract

Providing Feedback: If you have a suggestion to improve this documentation, or find an error, please contact technical support at to create an issue on the Ansible Automation Platform Jira project using the Docs component.

Table of Contents

CHAPTER 1. UPGRADING ISOLATED NODES TO EXECUTION NODES

Upgrading from version 1.x to the latest version of the Red Hat Ansible Automation Platform requires platform administrations to migrate data from isolated legacy nodes to execution nodes. This migration is necessary to deploy the automation mesh.

This guide explains how to perform a side-by-side migration. This ensures that the data on your original automation environment remains untouched during the migration process.

The migration process involves the following steps:

1. Verify upgrade configurations.
2. Backup original instance.
3. Deploy new instance for a side-by-side upgrade.
4. Recreate instance groups in the new instance using ansible controller.
5. Restore original backup to new instance.
6. Set up execution nodes and upgrade instance to Red Hat Ansible Automation Platform 2.1.
7. Configure upgraded controller instance.

1.1. PREREQUISITES FOR UPGRADING ANSIBLE AUTOMATION PLATFORM

Before you begin to upgrade Ansible Automation Platform, ensure your environment meets the following node and configuration requirements.

1.1.1. Node requirements

The following specifications are required for the nodes involved in the Ansible Automation Platform upgrade process:

- 16 GB of RAM for controller nodes, database node, execution nodes and hop nodes.
- 4 CPUs for controller nodes, database nodes, execution nodes, and hop nodes.
- 150 GB+ disk space for database node.
- 40 GB+ disk space for non-database nodes.
- DHCP reservations use infinite leases to deploy the cluster with static IP addresses.
- DNS records for all nodes.
- Red Hat Enterprise Linux 8 or later 64-bit (x86) installed for all nodes.
- Chrony configured for all nodes.

1.1.2. Automation controller configuration requirements

The following automation controller configurations are required before you proceed with the Ansible Automation Platform upgrade process:

Configuring NTP server using Chrony

Each Ansible Automation Platform node in the cluster must have access to an NTP server. Use the **chronyd** to synchronize the system clock with NTP servers. This ensures that cluster nodes using SSL certificates that require validation do not fail if the date and time between nodes are not in sync.

This is required for all nodes used in the upgraded Ansible Automation Platform cluster:

1. Install **chrony**:

```
# dnf install chrony --assumeyes
```

2. Open /**etc**/**chrony.conf** using a text editor.

3. Locate the public server pool section and modify it to include the appropriate NTP server addresses. Only one server is required, but three are recommended. Add the **iburst** option to speed up the time it takes to properly sync with the servers:

```
# Use public servers from the pool.ntp.org project.
# Please consider joining the pool (http://www.pool.ntp.org/join.html).
server <ntp-server-address> iburst
```

4. Save changes within the /**etc**/**chrony.conf** file.

5. Start the host and enable the **chronyd** daemon:

```
# systemctl --now enable chronyd.service
```

6. Verify the **chronyd** daemon status:

```
# systemctl status chronyd.service
```

Attaching Red Hat subscription on all nodes

Red Hat Ansible Automation Platform requires you to have valid subscriptions attached to all nodes. You can verify that your current node has a Red Hat subscription by running the following command:

```
# subscription-manager list --consumed
```

If there is not a Red Hat subscription attached to the node, see attaching your Ansible Automation Platform subscription for more information.

Creating non-root user with sudo privileges

Before you upgrade Ansible Automation Platform, it is recommended to create a non-root user with sudo privileges for the deployment process. This user is used for:

- SSH connectivity.
- Passwordless authentication during installation.

- Privilege escalation (sudo) permissions.

The following example uses **ansible** to name this user. On all nodes used in the upgraded Ansible Automation Platform cluster, create a non-root user named **ansible** and generate an ssh key:

1. Create a non-root user:

```
# useradd ansible
```

2. Set a password for your user:

```
# passwd ansible 1
Changing password for ansible.
Old Password:
New Password:
Retype New Password:
```

1 Replace **ansible** with the non-root user from step 1, if using a different name

3. Generate an **ssh** key as the user:

```
$ ssh-keygen -t rsa
```

4. Disable password requirements when using **sudo**:

```
# echo "ansible ALL=(ALL) NOPASSWD:ALL" | sudo tee -a /etc/sudoers.d/ansible
```

Copying SSH keys to all nodes

With the **ansible** user created, copy the **ssh** key to all the nodes used in the upgraded Ansible Automation Platform cluster. This ensures that when the Ansible Automation Platform installation runs, it can **ssh** to all the nodes without a password:

```
$ ssh-copy-id ansible@node-1.example.com
```

NOTE

If running within a cloud provider, you might need to instead create an **~/.ssh/authorized_keys** file containing the public key for the **ansible** user on all your nodes and set the permissions to the **authorized_keys** file to only the owner (**ansible**) having read and write access (permissions 600).

Configuring firewall settings

Configure the firewall settings on all the nodes used in the upgraded Ansible Automation Platform cluster to permit access to the appropriate services and ports for a successful Ansible Automation Platform upgrade. For Red Hat Enterprise Linux 8 or later, enable the **firewalld** daemon to enable the access needed for all nodes:

1. Install the **firewalld** package:

```
# dnf install firewalld --assumeyes
```

2. Start the **firewalld** service:

```
# systemctl start firewalld
```

3. Enable the **firewalld** service:

```
# systemctl enable --now firewalld
```

1.1.3. Ansible Automation Platform configuration requirements

The following Ansible Automation Platform configurations are required before you proceed with the Ansible Automation Platform upgrade process:

Configuring firewall settings for execution and hop nodes

After upgrading your Red Hat Ansible Automation Platform instance, add the automation mesh port on the mesh nodes (execution and hop nodes) to enable automation mesh functionality. The default port used for the mesh networks on all nodes is **27199/tcp**. You can configure the mesh network to use a different port by specifying **recptor_listener_port** as the variable for each node within your inventory file.

Within your hop and execution node set the **firewalld** port to be used for installation.

1. Ensure that **firewalld** is running:

```
$ sudo systemctl status firewalld
```

2. Add the **firewalld** port to your controller database node (e.g. port 27199):

```
$ sudo firewall-cmd --permanent --zone=public --add-port=27199/tcp
```

3. Reload **firewalld**:

```
$ sudo firewall-cmd --reload
```

4. Confirm that the port is open:

```
$ sudo firewall-cmd --list-ports
```

1.2. BACK UP YOUR ANSIBLE AUTOMATION PLATFORM INSTANCE

Back up an existing Ansible Automation Platform instance by running the **.setup.sh** script with the **backup_dir** flag, which saves the content and configuration of your current environment:

1. Navigate to your **ansible-tower-setup-latest** directory.

2. Run the **./setup.sh** script following the example below:

```
$ ./setup.sh -e ‘backup_dir=/ansible/mybackup’ -e ‘use_compression=True’ @credentials.yml
-b (1) (2)
```

(1) **backup_dir** specifies a directory to save your backup to.

2 **@credentials.yml** passes the password variables and their values encrypted via **ansible-vault**.

With a successful backup, a backup file is created at **/ansible/mybackup/tower-backup-latest.tar.gz** .

This backup will be necessary later to migrate content from your old instance to the new one.

1.3. DEPLOY A NEW INSTANCE FOR A SIDE-BY-SIDE UPGRADE

To proceed with the side-by-side upgrade process, deploy a second instance of Ansible Tower 3.8.x with the same instance group configurations. This new instance will receive the content and configuration from your original instance, and will later be upgraded to Red Hat Ansible Automation Platform 2.1.

1.3.1. Deploy a new instance of Ansible Tower

To deploy a new Ansible Tower instance, do the following:

1. Download the Tower installer version that matches your original Tower instance by navigating to the Ansible Tower installer page .
2. Navigate to the installer, then open the **inventory** file using a text editor to configure the **inventory** file for a Tower installation:
   a. In addition to any Tower configurations, remove any fields containing **isolated_group** or **instance_group**.

NOTE

For more information about installing Tower using the Ansible Automation Platform installer, see the Ansible Automation Platform Installation Guide for your specific installation scenario.

3. Run the **setup.sh** script to begin the installation.

Once the new instance is installed, configure the Tower settings to match the instance groups from your original Tower instance.

1.3.2. Recreate instance groups in the new instance

To recreate your instance groups in the new instance, do the following:

NOTE

Make note of all instance groups from your original Tower instance. You will need to recreate these groups in your new instance.

1. Log in to your new instance of Tower.
2. On the side panel, navigate to **Administration → Instance groups**.
3. Click the + button, then click **Create instance group**.
4. Enter a **Name** that matches an instance group from your original instance, then click **Save**.

5. Repeat until all instance groups from your original instance have been recreated.

1.4. RESTORE BACKUP TO NEW INSTANCE

Running the **./setup.sh** script with the **restore_backup_file** flag migrates content from the backup file of your original 1.x instance to the new instance. This effectively migrates all job histories, templates, and other Ansible Automation Platform related content.

Procedure

1. Run the following command:

```
$ ./setup.sh -r -e 'restore_backup_file=/ansible/mybackup/tower-backup-latest.tar.gz' -e
'use_compression=True' -e @credentials.yml -r -- --ask-vault-pass 1 2 3
```

1. **restore_backup_file** specifies the location of the Ansible Automation Platform backup database
2. **use_compression** is set to **True** due to compression being used during the backup process
3. **-r** sets the restore database option to **True**

2. Log in to your new RHEL 8 Tower 3.8 instance to verify whether the content from your original instance has been restored:
   a. Navigate to **Administration** > **Instance groups**. The recreated instance groups should now contain the **Total Jobs** from your original instance.
   b. Using the side navigation panel, check that your content has been imported from your original instance, including Jobs, Templates, Inventories, Credentials, and Users.

You now have a new instance of Ansible Tower with all the Ansible content from your original instance.

You will upgrade this new instance to Ansible Automation Platform 2.1 so that you keep all your previous data without overwriting your original instance.

1.5. UPGRADING TO ANSIBLE AUTOMATION PLATFORM 2.1

To upgrade your instance of Ansible Tower to Ansible Automation Platform 2.1, copy the **inventory** file from your original Tower instance to your new Tower instance and run the installer. The Red Hat Ansible Automation Platform installer detects a pre-2.1 inventory file and offers an upgraded inventory file to continue with the upgrade process:

1. Download the latest installer for Red Hat Ansible Automation Platform from the Red Hat Customer Portal.
2. Extract the files:

```
$ tar xvzf ansible-automation-platform-setup-<latest-version>.tar.gz
```

3. Navigate into your Ansible Automation Platform installation directory:

```
$ cd ansible-automation-platform-setup-<latest-version>/
```

4. Copy the **inventory** file from your original instance into the directory of the latest installer:

```
$ cp ansible-tower-setup-3.8.x.x/inventory ansible-automation-platform-setup-<latest-version>
```

5. Run the **setup.sh** script:

```
$ ./setup.sh
```

The setup script pauses and indicates that a "pre-2.x" inventory file was detected, but offers a new file called **inventory.new.ini** allowing you to continue to upgrade your original instance.

6. Open **inventory.new.ini** with a text editor.

NOTE

By running the setup script, the Installer modified a few fields from your original inventory file, such as renaming [tower] to [automationcontroller].

7. Modify the newly generated **inventory.new.ini** file to configure your automation mesh by assigning relevant variables, nodes, and relevant node-to-node peer connections:

NOTE

The design of your automation mesh topology depends on the automation needs of your environment. The example below offers one possible scenario for automation mesh design, and the design of your automation mesh topology depends on the automation needs of your environment. Review the full Ansible Automation Platform automation mesh guide for information on designing it for your needs.

Example inventory file with a standard control plane consisting of three nodes utilizing hop nodes:

```
[automationcontroller]
control-plane-1.example.com
control-plane-2.example.com
control-plane-3.example.com

[automationcontroller:vars]
node_type=control (1)
peers=execution_nodes (2)


[execution_nodes]
execution-node-1.example.com peers=execution-node-2.example.com
execution-node-2.example.com peers=execution-node-3.example.com
execution-node-3.example.com peers=execution-node-4.example.com
execution-node-4.example.com peers=execution-node-5.example.com node_type=hop
execution-node-5.example.com peers=execution-node-6.example.com node_type=hop (3)
execution-node-6.example.com peers=execution-node-7.example.com
execution-node-7.example.com
```

```
[execution_nodes:vars]
node_type=execution
```

1. Specifies a control node that runs project and inventory updates and system jobs, but not regular jobs. Execution capabilities are disabled on these nodes.
2. Specifies peer relationships for node-to-node connections in the **[execution_nodes]** group.
3. Specifies hop nodes that route traffic to other execution nodes. Hop nodes cannot execute automation.

8. Once you have finished configuring your **inventory.new.ini** for automation mesh, run the setup script using **inventory.new.ini**:

```
$ ./setup.sh -i inventory.new.ini -e @credentials.yml -- --ask-vault-pass
```

9. Once the installation completes, verify that your Ansible Automation Platform has been installed successfully by logging in to the Ansible Automation Platform dashboard UI across all automation controller nodes.

Additional resources

- For general information on using the Ansible Automation Platform installer, see the Red Hat Ansible Automation Platform installation guide.
- For more information about automation mesh, see the Ansible Automation Platform automation mesh guide

1.6. CONFIGURING YOUR UPGRADED ANSIBLE AUTOMATION PLATFORM

1.6.1. Configuring automation controller instance groups

After upgrading your Red Hat Ansible Automation Platform instance, associate your original instance with its corresponding instance groups by configuring settings in the automation controller UI:

1. Log into the new Controller instance.
2. Content from the old instance, such as credentials, jobs, inventories should now be visible on your Controller instance.
3. Navigate to **Administration → Instance Groups**.
4. Associate execution nodes by clicking on an instance group, then click the **Instances** tab.
5. Click **Associate**. Select the node(s) to associate to this instance group, then click **Save**.
6. You can also modify the default instance to disassociate your new execution nodes.

CHAPTER 2. MIGRATING TO AUTOMATION EXECUTION ENVIRONMENTS

2.1. WHY UPGRADE TO AUTOMATION EXECUTION ENVIRONMENTS?

Red Hat Ansible Automation Platform 2.1 introduces automation execution environments. Automation execution environments are container images that allow for easier administration of Ansible by including everything needed to run Ansible automation within a single container. Automation execution environments include:

- RHEL UBI 8
- Ansible 2.9 or Ansible Core 2.11
- Python 3.8
- Any Ansible Content Collections
- Collection python or binary dependencies

By including these elements, Ansible provides platform administrators a standardized way to define, build, and distribute the environments the automation runs in.

Due to the new automation execution environment, it is no longer necessary for administrators to create custom plugins and automation content. Administrators can now spin up smaller automation execution environments in less time to create their content.

All custom dependencies are now defined in the development phase instead of the administration and deployment phase. Decoupling from the control plane enables faster development cycles, scalability, reliability, and portability across environments. Automation execution environments enables the Ansible Automation Platform to move to a distributed architecture allowing administrators to scale automation across their organization.

2.2. ABOUT MIGRATING LEGACY VENVS TO AUTOMATION EXECUTION ENVIRONMENTS

When upgrading from older versions of automation controller to version 4.0 or later, the controller can detect previous versions of virtual environments associated with Organizations, Inventory and Job Templates and informs you to migrate to the new automation execution environments model. A new installation of automation controller creates two virtualenvs during the installation; one runs the controller and the other runs Ansible. Like legacy virtual environments, automation execution environments allow the controller to run in a stable environment, while allowing you to add or update modules to your automation execution environments as necessary to run your playbooks.

You can duplicate your setup in an automation execution environment from a previous custom virtual environment by migrating it to the new automation 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 automation execution environment (**export_custom_venv**)

The below workflow describes how to migrate from legacy venvs to automation execution environments using the **awx-manage** command.

2.3. MIGRATING VIRTUAL ENVS TO AUTOMATION EXECUTION ENVIRONMENTS

Use the following sections to assist with additional steps in the migration process once you have upgraded to Red Hat Ansible Automation Platform 2.0 and automation controller 4.0.

2.3.1. Listing custom virtual environments

You can list the virtual environments on your automation controller instance using the **awx-manage** command.

Procedure

1. SSH into your automation controller instance and run:

```
$ awx-manage list_custom_venvs
```

A list of discovered virtual environments will appear.

```
# Discovered virtual environments:
/var/lib/awx/venv/testing
/var/lib/venv/new_env

To export the contents of a virtual environment, re-run while supplying the path as an argument:
awx-manage export_custom_venv /path/to/venv
```

2.3.2. Viewing objects associated with a custom virtual environment

View the organizations, jobs, and inventory sources associated with a custom virtual environment using the **awx-manage** command.

Procedure

1. SSH into your automation controller instance and run:

```
$ awx-manage custom_venv_associations /path/to/venv
```

A list of associated objects will appear.

```
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: []
```

2.3.3. Selecting the custom virtual environment to export

Select the custom virtual environment you wish to export using **awx-manage export_custom_venv** command.

Procedure

1. SSH into your automation controller instance and run:

```
$ awx-manage export_custom_venv /path/to/venv
```

The output from this command will show a **pip freeze** of what is in the specified virtual environment. This information can be copied into a **requirements.txt** file for Ansible Builder to use for creating a new automation execution environments image

```
numpy==1.20.2
pandas==1.2.4
python-dateutil==2.8.1
pytz==2021.1
six==1.16.0

To list all available custom virtual environments run:
awx-manage list_custom_venvs
```

NOTE

Pass the **-q** flag when running **awx-manage list_custom_venvs** to reduce output.

2.4. ADDITIONAL RESOURCES

- See the Red Hat Ansible Automation Platform Creator Guide for more information of migrating to automation execution environments.