Red Hat Ansible Automation Platform 2.4

Ansible Automation Platform 1.2 to 2 Migration Guide

Last Updated: 2023-11-14

Red Hat Ansible Automation Platform 2.4 Ansible Automation Platform 1.2 to 2 Migration Guide

Anshul Behl

Roger Lopez
ansible-feedback@redhat.com

Abstract

This document provides a methodology to migrate from Ansible Automation Platform 1.2 to Ansible Automation Platform 2

Table of Contents

COMMENTS AND FEEDBACK 3

CHAPTER 1. OVERVIEW 4
1.1. ARCHITECTURAL OVERVIEW 5

CHAPTER 2. MIGRATION CONSIDERATIONS 10
2.1. TECHNICAL CONSIDERATIONS 10
2.2. ANSIBLE CONTENT MIGRATION CONSIDERATIONS 11
2.3. OPERATING MODEL CONSIDERATIONS 12

CHAPTER 3. PREREQUISITES 13
3.1. ENVIRONMENT SPECIFICATIONS 13
3.2. NETWORK REQUIREMENTS 15
3.3. VALIDATION CHECKLIST FOR NODES 16
3.4. AUTOMATION CONTROLLER CONFIGURATION DETAILS 17
3.4.1. OS Configuration 17
3.4.1.1. Chrony Configuration 17
3.4.1.2. Red Hat Subscription Manager 18
3.4.1.3. User Accounts 19
3.4.1.4. Copying SSH keys to all nodes 19
3.4.1.5. Configuring Firewall Settings 20
3.5. ANSIBLE AUTOMATION PLATFORM CONFIGURATION DETAILS 20
3.5.1. Configuring firewall settings for execution and hop nodes 20

CHAPTER 4. INFRASTRUCTURE MIGRATION 22
4.1. BACKUP ANSIBLE AUTOMATION PLATFORM 1.2 ON ENVIRONMENT A 22
4.2. IMPORT ANSIBLE AUTOMATION PLATFORM 1.2 DATABASE TO ENVIRONMENT B 23
4.3. UPGRADE ENVIRONMENT B TO ANSIBLE AUTOMATION PLATFORM 2 24
4.4. CONFIGURING INSTANCE AND INSTANCE GROUPS 28

CHAPTER 5. MIGRATE VIRTUAL ENVIRONMENTS TO EXECUTION ENVIRONMENTS 30
5.1. AUTOMATING THE MIGRATION OF VIRTUAL ENVIRONMENTS TO EXECUTION ENVIRONMENTS 30
5.1.1. Pushing to private automation hub 33

APPENDIX A. ABOUT THE AUTHORS 35

APPENDIX B. CONTRIBUTORS 36

APPENDIX C. CREATING AN ENCRYPTED CREDENTIALS.YML FILE 37

APPENDIX D. POST UPGRADE PLAYBOOK 39

APPENDIX E. REVISION HISTORY 40

COMMENTS AND FEEDBACK

In the spirit of open source, we invite anyone to provide feedback and comments on any reference architecture. Although we review our papers internally, sometimes issues or typographical errors are encountered. Feedback allows us to not only improve the quality of the papers we produce, but allows the reader to provide their thoughts on potential improvements and topic expansion to the papers. Feedback on the papers can be provided by emailing ansible-feedback@redhat.com. Please refer to the title within the email.

CHAPTER 1. OVERVIEW

The Ansible Automation Platform 1.2 to Ansible Automation Platform 2 migration guide provides an opinionated methodology on how to approach a side-by-side migration using the Ansible Automation Platform installer. Throughout this guide, you'll be provided steps to follow that ensure the process of backing up, importing and upgrading to Ansible Automation Platform 2 is a success. This reference architecture is best suited for system and platform administrators looking to migrate to the latest version of Ansible Automation Platform 2.

This side-by-side migration reference architecture consists of having two environments, *Environment A* and *Environment B*, where all the data from your *Environment A* Ansible Automation Platform 1.2 environment gets migrated and upgraded to a new *Environment B* Ansible Automation Platform 2 replacement environment.

The high level approach of the platform migration process:

- Creates a full Ansible Automation Platform 1.2 data backup using the Ansible Automation Platform installer on *Environment A*
- Imports the full Ansible Automation Platform data backup to the new and empty replacement Ansible Automation Platform 1.2 environment control plane (*Environment B*)
- Upgrades *Environment B* using the Ansible Automation Platform 2 installer from its Ansible Automation Platform 1.2 version
- Deploy automation mesh execution and hop nodes in *Environment B*

Once the infrastructure is successfully migrated, the focus shifts to migrating your Python virtual environments in your Ansible Automation Platform 1.2 environment from *Environment A* to automation execution environments that will be used within your newly available Ansible Automation Platform 2 environment on *Environment B*. This one-time effort opens the door to take advantage of the latest Ansible Automation Platform 2 capabilities and the ability to execute consistent automation across multiple platforms with reduced operational overhead.

The high level execution environment adoption process consists of:

- Exporting custom virtual environments from Ansible Automation Platform 1.2 on *Environment A*
- Comparing each exported virtual environment against Ansible 2.9 base execution environment on *Environment A*
- Creating new execution environments using the Ansible 2.9 execution environment plus the additional dependencies not included that were exported by the virtual environment
- Attaching the new execution environment(s) to corresponding Job Templates on *Environment B*

NOTE

This document shows an opinionated reference environment showing a holistic migration approach which helps customers understand how they can retain their job run history and platform objects between the two clusters in a complex architecture. Based upon your existing architecture and requirements, the migration process may be further simplified. In many cases, the default Ansible Automation Platform installer values are sufficient. The migration process documented here applies to both simple and complex environments.

NOTE

Customers migrating from Ansible Automation Platform 1.2 to version 2 may use the same manifest in both clusters/instances for upgrading as long as the managed node inventory is the same for both clusters/instances. The migration period may not exceed six months without an approved exception from the Ansible Business Unit by means of a formal BU Guidance request from the Red Hat Account Representative.

1.1. ARCHITECTURAL OVERVIEW

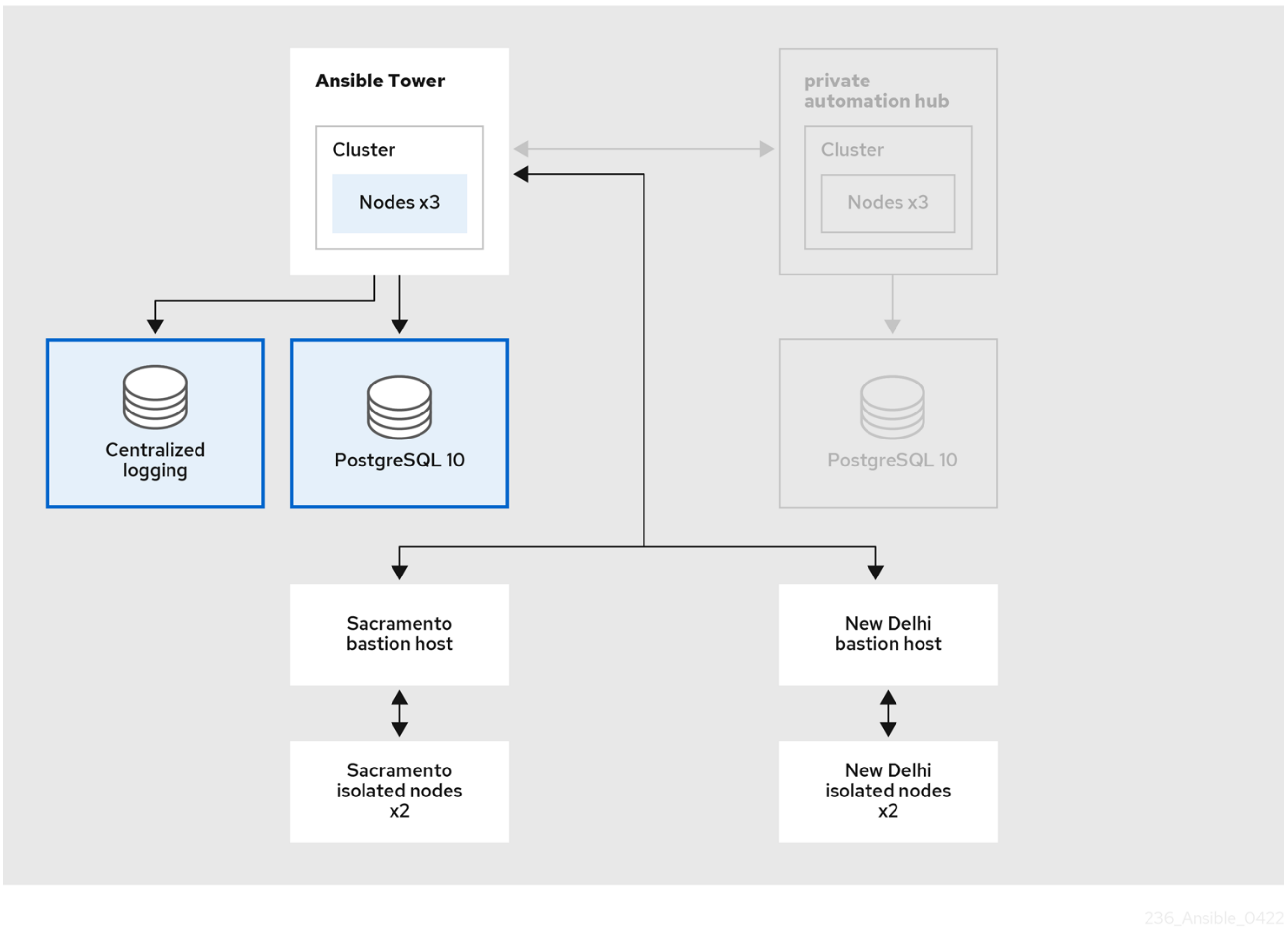
This section focuses on the architecture details of the two environments as they go through the side-by-side migration process.

The first environment, *Environment A*, consists of:

- 3 Ansible Tower 3.8.5 nodes running Red Hat Enterprise Linux 7 located in Raleigh, NC data center
- 1 Red Hat Enterprise Linux 7 database node
- 2 bastion hosts (jump hosts to access their corresponding isolated nodes)
- 2 isolated nodes located in Sacramento, CA data center
- 2 isolated nodes located in New Delhi, India data center

A pictorial representation of this environment is shown in:

Figure 1.1. Environment A Architecture Overview



The second environment, *Environment B*, is a new and empty Ansible Automation Platform 1.2 environment that will be used to import all the data from *Environment A* using the Ansible Automation Platform 1.2 installer from *Environment B* prior to upgrading to Ansible Automation Platform 2.

Initially, *Environment B* consists of:

- 3 Ansible Tower 3.8.5 nodes running Red Hat Enterprise Linux 8
- 1 Red Hat Enterprise Linux 8 database node

WARNING

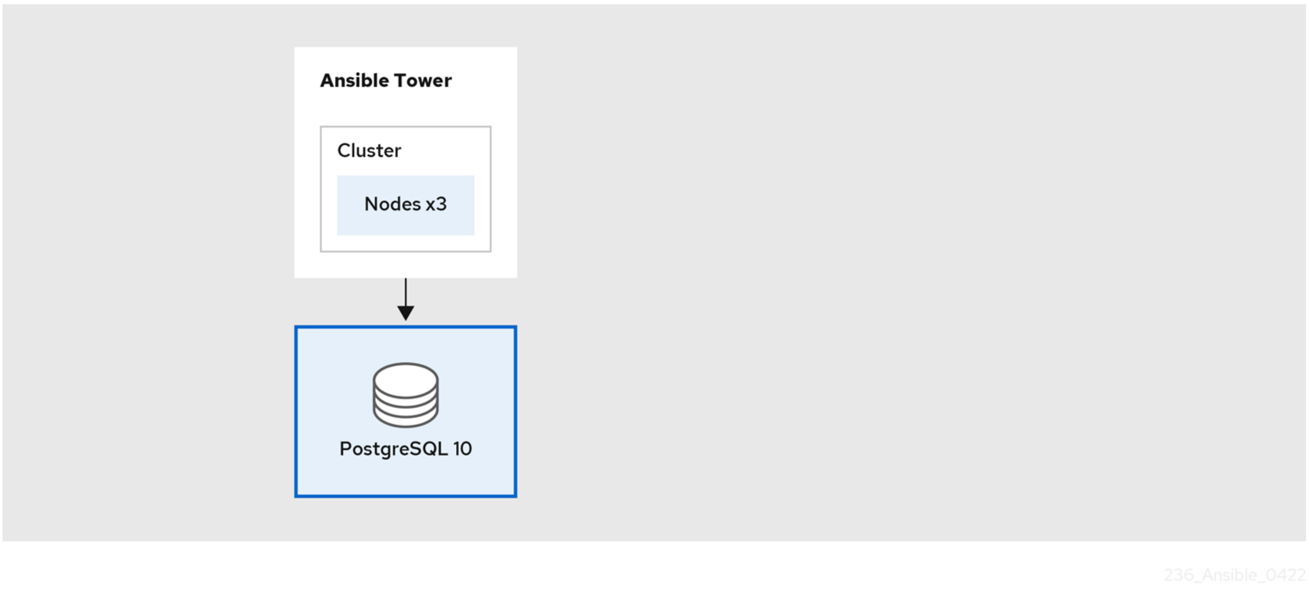
Ansible Automation Platform 2 does not support Red Hat Enterprise Linux 7. It is critical that Red Hat Enterprise Linux 8 be used as the base OS for your Ansible Automation Platform 1.2 *Environment B* prior to upgrading to Ansible Automation Platform 2.

NOTE

Throughout this reference architecture you will find references to Ansible Automation Platform 2.1. However, the Ansible Automation Platform migration process found within this document applies to version 2.1 or later.

A pictorial representation of the initial *Environment B* footprint is shown below:

Figure 1.2. Initial Environment B Architecture Overview



Once the data migration from *Environment A* to *Environment B* is successful, we expand the architecture of *Environment B* during the upgrade process by including:

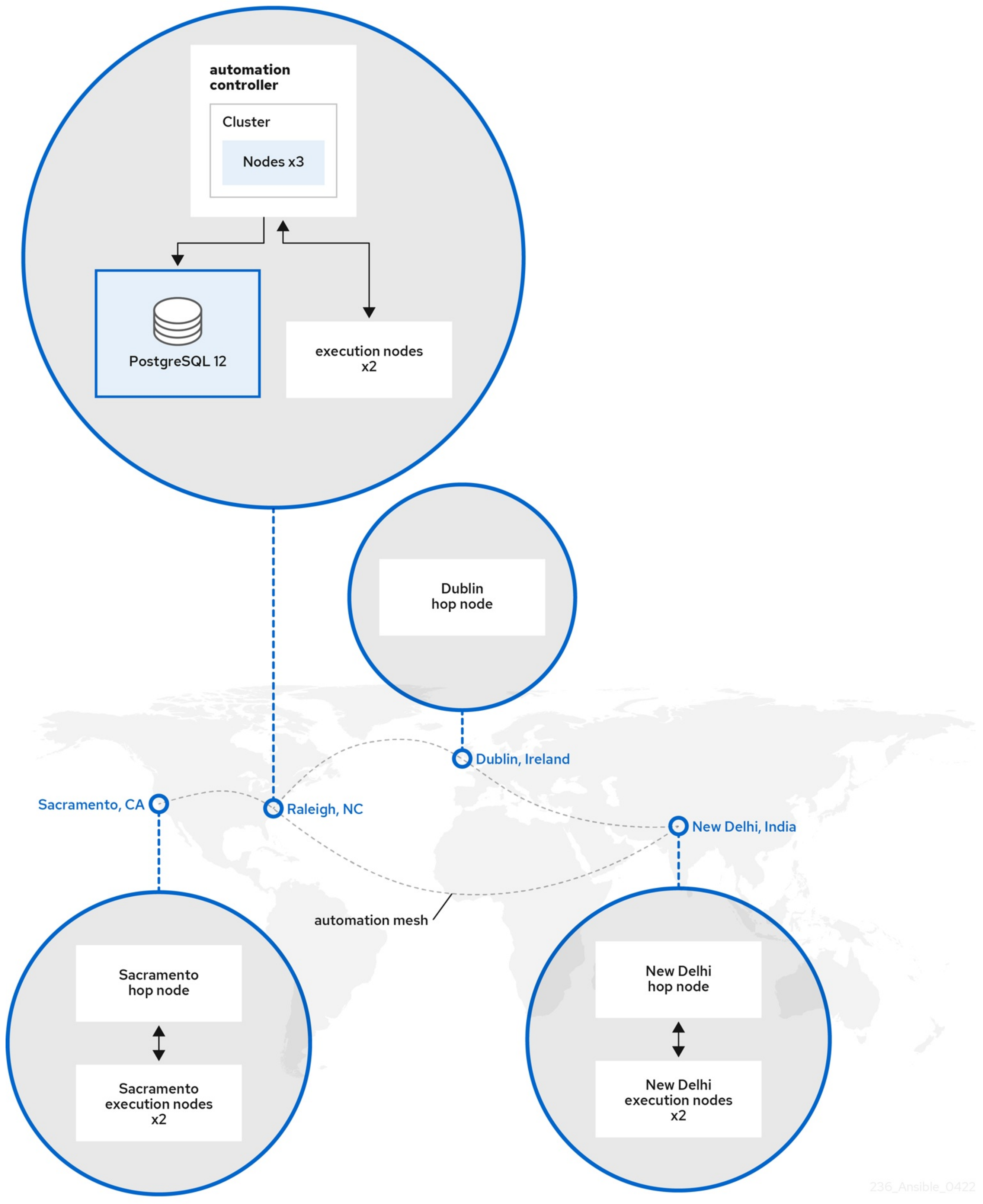
- 2 execution nodes accessible directly by the Ansible controllers
- 3 hop nodes (**sacramento-hop**, **dublin-hop new-delhi-hop**)
- 2 execution nodes accessible only via hop node **sacramento-hop**
- 2 execution nodes accessible via hop node **dublin-hop** and **new-delhi-hop**

NOTE

The **dublin-hop** provides another route via the automation mesh that can be used to access the execution nodes that reside in New Delhi, India.

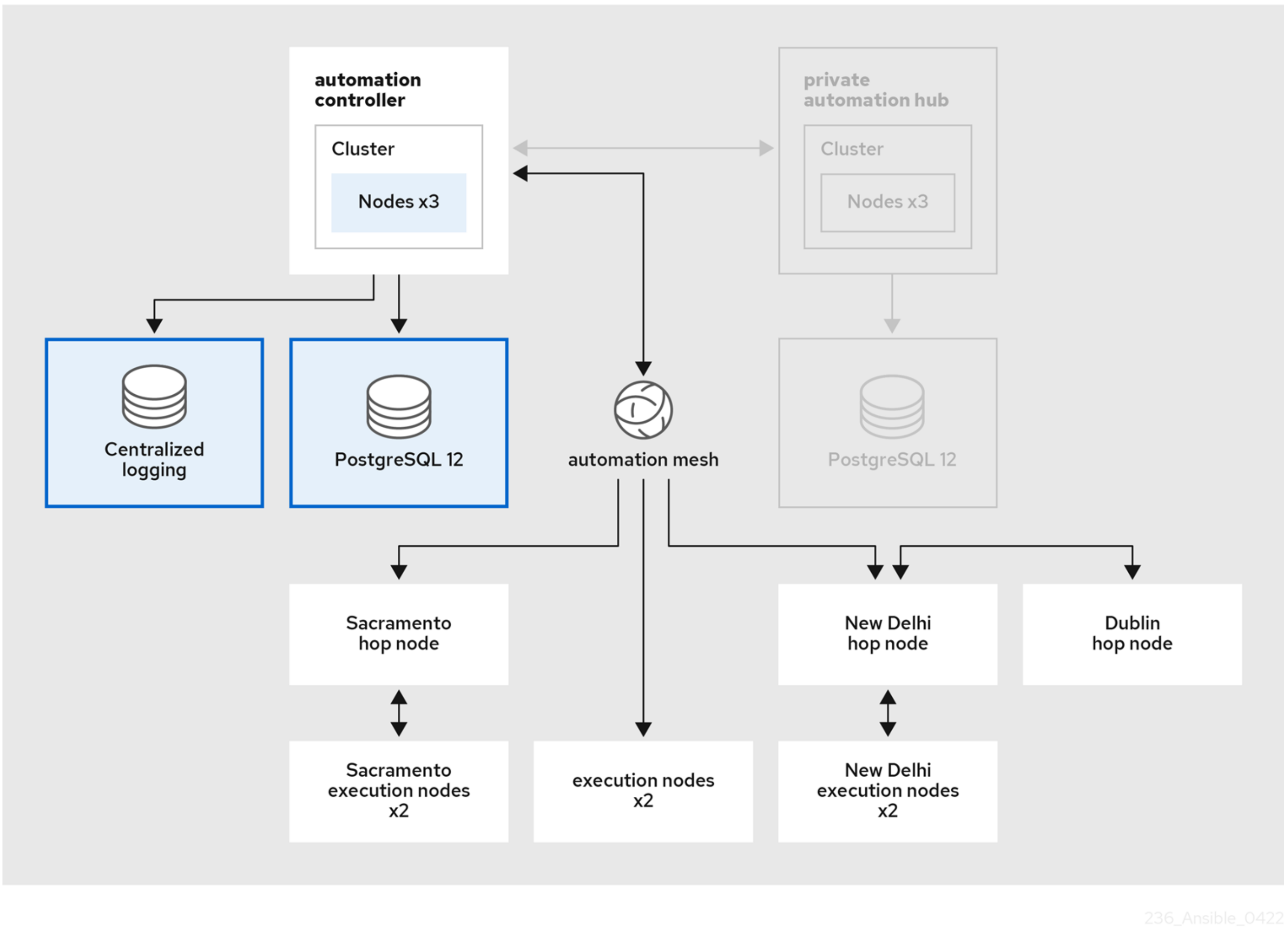
A world view of the expanded Ansible Automation Platform 2 *Environment B* is shown below:

Figure 1.3. World View of Environment B



A detailed representation of the expanded Ansible Automation Platform 2 *Environment B* architectural footprint is shown below:

Figure 1.4. Expanded Environment B Architecture Overview



NOTE

The process of disabling schedules from Job Templates is not covered within this reference architecture. It is important that once a successful migration is complete, to disable scheduling from any Job Templates from *Environment A* to ensure you do not have both *Environment A* and *Environment B* running the same jobs simultaneously.

CHAPTER 2. MIGRATION CONSIDERATIONS

With the introduction of Ansible Automation Platform 2, a re-imagined architecture has been created that expands the capabilities of automation. Ansible Automation Platform 2 now decouples the automation control plane and execution plane, which provides for a much more flexible architecture. This new capability alongside the introduction of automation mesh allows an organization to scale automation across the globe and allows automation to run as close to the endpoints as possible.

Prior to the step-by-step migration approach provided for this reference architecture, it is important to carefully assess and plan to determine your overall migration readiness to proceed without exceptions.

2.1. TECHNICAL CONSIDERATIONS

One of the key changes between Ansible Automation Platform 1.2 and Ansible Automation Platform 2 is the removal of isolated nodes in favor of hop nodes and execution nodes with automation mesh. Automation mesh is an overlay network that provides a simple, flexible and reliable way to scale automation of large inventories across diverse network topologies, platforms and regions. Automation mesh provides flexible design options to build resilient and fault-tolerant architectures while providing enhanced security to standardize and normalize automation across your entire IT estate.

In this reference environment, we capture the steps of migrating Ansible Automation Platform 1.2 that uses isolated nodes and third party tooling, such as SSH proxies and jump hosts, to upgrade to Ansible Automation Platform 2 using automation mesh hop and execution nodes.

The use of automation mesh and execution nodes will require additional ports be opened within your firewall.

Protocol	Port	Purpose
SSH	22/TCP	Ansible Automation Platform installation
HTTPS	443/TCP	Web UI, API, Execution Environment (EE) pulls
Receptor	27199/TCP	automation mesh

NOTE

The receptor TCP port can be customized during the Ansible Automation Platform upgrade process.

Another important consideration is database supportability. With the introduction of automation controller, the requirement of Postgres 12 database is introduced. If your existing Postgres 10 database was installed/managed by Ansible Automation Platform 1.2, the upgrade process to Ansible Automation Platform 2 handles the upgrade of the Postgres database to the new version 12.

NOTE

For this reference architecture, the Postgres database is managed by Ansible Automation Platform.

If you manage your own Postgres 10 database, you will want to:

- Install a new Postgres 10 database that will be part of the side-by-side migration
- Import your data from *Environment A* to the newly created Postgres 10 database that is to be used by *Environment B*
- Upgrade from Postgres 10 to Postgres 12 on the database that is to be used by *Environment B*
- With the upgraded Postgres 12, upgrade your Ansible Automation Platform 1.2 to Ansible Automation Platform 2 on *Environment B*

NOTE

Managing your own Postgres database is out of scope for this reference architecture.

NOTE

For more information regarding database supportability, visit the Database Scope of Coverage article.

Aside from the key considerations regarding isolated nodes and database supportability, there is a list of functionality that has been removed in Ansible Automation Platform 2. The list includes:

- The ability to delete the default instance group through the User Interface
- Support for deploying on CentOS (any version) and RHEL 7
- Support for Mercurial projects
- Support for custom inventory scripts stored in controller
- Resource profiling code (AWX_RESOURCE_PROFILING_*)
- Support for custom Python virtual environments in favor of execution environments
- Top-level **/api/v2/job_events/** API endpoint
- Job isolation is achieved using Execution environments and is no longer a feature of Ansible Tower.

NOTE

The replacement of custom Python virtual environments with user-built execution environments is described and done via an Ansible Playbook provided by this reference architecture in a later chapter.

2.2. ANSIBLE CONTENT MIGRATION CONSIDERATIONS

As you evaluate the process of migrating your environment to Ansible Automation Platform 1.2 to Ansible Automation Platform 2, there may be additional considerations to address when it comes to Ansible content (Collections, modules, roles, plugins, etc).

These considerations are not limited to this list, but include the following:

- Ansible Playbooks must run on Ansible Engine 2.9.10 or higher in order to be used in a compatibility execution environment (required)
- Ansible content in playbooks should utilize Ansible Collections (not required, but recommended)
- Ansible content in playbooks should utilize Fully Qualified Collection Name (FQCN) (not required, but recommended)
- Ansible Playbooks must be modified to address any references to localhost as using execution environments this refers to the pod itself and not the underlying host as previously done.

NOTE

User-built execution environments may be required if additional dependencies are needed to successfully execute your Ansible content.

2.3. OPERATING MODEL CONSIDERATIONS

With the adoption of a new Ansible Automation Platform architecture, it is vital to have a plan on how you will integrate your newly designed Ansible Automation Platform environment.

Key questions to have answers to include the following:

- What training and enablement is required in my organization to run Ansible Automation Platform 2?
- How will execution environments and Ansible Content Collections be managed?
- What execution environment container versioning best fits my organization?
  - Do I want different models for development and production?
- If user-built execution environments are needed, how will I manage and distribute those environments?
- Do we plan on building CI around execution environments?
- What is the security response plan to patch CVEs and remain compliant?
- How often will I upgrade my Ansible Automation Platform clusters?

While these are just a few examples, they are critical questions that need answers to ensure a successful strategy for your organization.

CHAPTER 3. PREREQUISITES

The side-by-side migration and upgrade to Ansible Automation Platform 2 requires *Environment A* and *Environment B*. *Environment A* should be your main cluster as shown in Figure 1.1, "Environment A Architecture Overview" consisting of:

- 3 Ansible Tower 3.8.5 nodes running Red Hat Enterprise Linux 7 located in Raleigh NC data center
- 1 Red Hat Enterprise Linux 7 database node.
- 2 bastion hosts(jump hosts to access their corresponding isolated nodes)
- 2 isolated nodes located in Sacramento, CA data center
- 2 isolated nodes located in New Delhi, India data center

Initially, *Environment B* is a simplified Ansible Automation Platform 1.2 architecture as shown in Figure 1.2, "Initial Environment B Architecture Overview". During the upgrade process to Ansible Automation Platform 2, a pool of Red Hat Enterprise Linux 8.4 servers expands the cluster to include:

- 2 execution nodes accessible directly by the control plane nodes
- 3 hop nodes (**sacramento-hop**, **dublin-hop** and **new-delhi-hop**)
- 2 execution nodes accessible only via hop node **sacramento-hop**
- 2 execution nodes accessible via hop nodes **dublin-hop** and **new-delhi-hop**

The final cluster post upgrade architecture can be seen within the Figure 1.4, "Expanded Environment B Architecture Overview"

NOTE

These nodes are not required to be physical servers.

3.1. ENVIRONMENT SPECIFICATIONS

Table 3.1. environment specifications

Node Type	Control	Execution	Hop	Database
CPU	4	4	4	4
RAM	16	16	16	16
Disk	40GB	40GB	40GB	150GB+
Notes	• Processes events and runs cluster jobs including	• Runs automation. Increase memory and CPU	• Serves to route traffic from one part of the	• Storage volume should be rated for a high baseline

	project updates and cleanup jobs. Increasing CPU and memory can help with job event processing. ● Dedicate a minimum of 20 GB to /**var**/` for file and working directory storage Storage volume should be rated for a minimum baseline of 1500 IOPS. ● Projects are stored on control and hybrid, and for the duration of jobs, also on execution nodes. If the cluster has many large projects, consider having twice the GB in /var/lib/awx/projects, to avoid disk space errors.	to increase capacity for running more forks	Automation Mesh to another (for example, could be a bastion host into another network). RAM could affect throughput, CPU activity is low. Network bandwidth and latency generally a more important factor than either RAM/CPU.	IOPS (1500 or more). ● Extra disk space is required for the /var/lib/pgsql directory. In the case you cannot expand the base OS partition, consider having 150GB+ for /var/lib/pgsql directory to avoid disk space errors.

NOTE

All automation controller data is stored in the PostgreSQL database. Database storage increases with the number of hosts managed, number of jobs run, number of facts stored in the fact cache, and number of tasks in any individual job. For example, a playbook run every hour (24 times a day) across 250, hosts, with 20 tasks will store over 800,000 events in the database every week.

If not enough space is reserved in the database, old job runs and facts will need to be cleaned on a regular basis. Refer to Management Jobs in the Automation Controller Administration Guide for more information

3.2. NETWORK REQUIREMENTS

Ansible Automation Platform 2 requires direct network connectivity between the automation controller instances and mesh worker nodes allocated to run control plane jobs.

NOTE

In case of hop nodes that route traffic to the execution nodes behind a DMZ or an isolated network, your Ansible Automation Platform cluster can span across multiple networks. A network administrator should ensure that **ssh** connectivity is available to all nodes for installation purposes. For this reference environment hop-nodes act as ssh proxies to get to their respective execution nodes.

In order to access the Ansible Automation Platform dashboard, a browser that can access the network that the control plane nodes reside on is required. If you wish to access the Ansible Automation Platform dashboard externally, ensure to add a public IP address to your control plane nodes.

It is recommended that network administrators provide a dedicated IP address to all nodes and an appropriate DNS record. Network administrators can assign a static IP address to the nodes and configure the DHCP server to either reserve the IP with an infinite lease, or network administrators should ensure the assigned IP addresses are part of an excluded range. This ensures each node’s IP address remains constant in the absence of a DHCP server. NOTE: For the purposes of this reference architecture, setup of a DHCP server, setting up DNS records and setting up a load balancer is out of scope.

NOTE

For the purposes of this reference architecture, setup of a DHCP server, setting up DNS records and setting up a load balancer is out of scope.

A network administrator should reserve at least the following number of IP addresses, including:

1. One IP address for each control plane node.
2. One IP address for each execution node.
3. One IP address for each hop node.
4. One IP address for the database node.

This reference environment reserves 13 IP addresses per site.

The following table provides an example of *Environment B* of the reference environment.

Usage	Host Name	IP
Automation Controller 1	envb_controller1.example.com	192.168.0.10
Automation Controller 2	envb_controller2.example.com	192.168.0.11
Automation Controller 3	envb_controller3.example.com	192.168.0.12
Control Plane Database	envb_database.example.com	192.168.0.13
Execution Node 1	envb_executionnode-1.example.com	192.168.0.14
Execution Node 2	envb_executionnode-2.example.com	192.168.0.15
Hop Node 1	envb_hopnode-sacramento.example.com	192.168.0.16
Hop Node 2	envb_hopnode-new-delhi.example.com	192.168.0.17
Hop Node 3	envb_hopnode-dublin.example.com	192.168.0.18
Execution Node 3	envb_executionnode-3.example.com	10.14.1.11
Execution Node 4	envb_executionnode-4.example.com	10.14.1.12
Execution Node 5	envb_executionnode-5.example.com	10.15.1.11
Execution Node 6	envb_executionnode-6.example.com	10.15.1.12

3.3. VALIDATION CHECKLIST FOR NODES

The following is a summary of all the requirements:

- ❏ 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 {rhel_version} or later 64-bit (x86) installed for all nodes

❑ Chrony configurationed for all nodes

NOTE

Check Section 3.1, "Environment Specifications" for more detailed notes on node requirements.

3.4. AUTOMATION CONTROLLER CONFIGURATION DETAILS

This reference architecture focuses on the migration and upgrade of Ansible Automation Platform 1.2 to Ansible Automation Platform 2. The configuration is intended to provide a comprehensive Ansible Automation Platform solution that covers isde-by-side migration scenarios. The key solution components covered within this reference archtiecture consists of:

- Ansible Automation Platform 1.2
- Ansible Automation Platform 2
- Ansible Automation Platform Installer
- Ansible Playbooks for custom Python virtual environment migration

3.4.1. OS Configuration

3.4.1.1. Chrony Configuration

Each Ansible Automation Platform node in the cluster must have access to an NTP server. The **chronyd** is a daemon for synchronization of the system clock. It can synchronize the clock with NTP servers. This ensures that when cluster nodes use SSL certificates that require validation, they don't fail if the date and time between the nodes are not in sync.

On all the nodes on *Environment B* including those to be used for the Ansible Automation Platform cluster expansion,

1. If not installed, install **chrony** as follows

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

2. Edit the **/etc/chrony.conf** file with a text editor such as **vi**.

```
# vi /etc/chrony.conf
```

3. Locate the following public server pool section, and modify it to include the appropriate servers. Only one server is required, but three is recommended. The **iburst** option is added to speed up the time that 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 all the changes within the **/etc/chrony.conf** file.
5. Start and enable that the **chronyd** daemon is started when the host is booted.

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

6. Verify the chronyd daemon status.

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

3.4.1.2. Red Hat Subscription Manager

The **subscription-manager** command registers a system to the Red Hat Network (RHN) and manages the subscription entitlements for a system. The **--help** option specifies on the command line to query the command for the available options. If the **--help** option is issued along with a command directive, then options available for the specific command directive are listed.

To use Red Hat Subscription Management for providing packages to a system, the system must first register with the service. In order to register a system, use the **subscription-manager** command and pass the **register** command directive. If the **--username** and **--password** options are specified, then the command does not prompt for the RHN Network authentication credentials.

An example of registering a system using **subscription-manager** is shown below.

NOTE

The following should be done on all the nodes within *Environment B* including those to be used for the Ansible Automation Platform cluster expansion.

```
# subscription-manager register --username [User] --password '[Password]'
The system has been registered with id: abcd1234-ab12-ab12-ab12-481ba8187f60
```

After a system is registered, it must be attached to an entitlement pool. For the purposes of this reference environment, the Red Hat Ansible Automation Platform is the pool chosen. Identify and subscribe to the Red Hat Ansible Automation Platform entitlement pool, the following command directives are required.

```
# subscription-manager list --available | grep -A8 "Red Hat Ansible Automation Platform"
---
Subscription Name:   Red Hat Ansible Automation Platform, Premium (5000 Managed Nodes)
Provides:            Red Hat Ansible Engine
                     Red Hat Single Sign-On
                     Red Hat Ansible Automation Platform
SKU:                 MCT3695
Contract:            <contract>
Pool ID:             <pool_id>
Provides Management: No
Available:           9990
Suggested:           1
Service Type:        L1-L3
Roles:
```

```
# subscription-manager attach --pool <pool_id>
Successfully attached a subscription for: Red Hat Ansible Automation Platform, Premium (5000
Managed Nodes)
```

```
# subscription-manager repos --enable=ansible-automation-platform-2.1-for-rhel-8-x86_64-rpms
```

NOTE

As part of the Ansible Automation Platform subscription, 10 Red Hat Enterprise Linux (RHEL) subscriptions are available to utilize RHEL for your automation controllers, private automation hubs and other Ansible Automation Platform components.

3.4.1.3. User Accounts

Prior to the installation of Ansible Automation Platform 2, 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

For the purposes of this reference environment, the user **ansible** was chosen, however, any user name would suffice.

On **all** the nodes on *Environment B* including those to be used for the Ansible Automation Platform cluster expansion, create a user named **ansible** and generate an **ssh** key.

1. Create a non-root user

   ```
   # useradd ansible
   ```

2. Set a password for your **ansible** user.

   ```
   # passwd ansible
   ```

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

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

4. Disable password requirements when using **sudo** as the **ansible** user

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

3.4.1.4. Copying SSH keys to all nodes

With the **ansible** user created, as the **ansible** user, copy the **ssh** key to all the nodes on *Environment B* including those to be used for the Ansible Automation Platform cluster expansion. This ensures that when the Ansible Automation Platform installation runs, it can **ssh** to all the nodes without a password.

This can be done using the **ssh-copy-id** command as follows:

```
$ ssh-copy-id ansible@hostname.example.com
```

NOTE

If running within a cloud provider, you may 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).

3.4.1.5. Configuring Firewall Settings

Firewall access and restrictions play a critical role in securing Ansible Automation Platform 2 environment. The use of Red Hat Enterprise Linux {rhel_version} defaults to using **firewalld**, a dynamic firewall daemon. **firewalld** works by assigning network zones to assign a level of trust to a network and its associated connections and interfaces.

It is recommended that firewall settings be configured to permit access to the appropriate services and ports for a success Ansible Automation Platform 2 installation.

On **all** the nodes within *Environment B* including those to be used for the Ansible Automation Platform cluster expansion, ensure that **firewalld** is installed, started and enabled.

1. Install the **firewalld** package

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

2. Start the **firewalld** service

   ```
   # systemctl start firewalld
   ```

3. Enable the **firewalld** service

   ```
   # systemctl enable firewalld
   ```

3.5. ANSIBLE AUTOMATION PLATFORM CONFIGURATION DETAILS

3.5.1. Configuring firewall settings for execution and hop nodes

For a successful Ansible Automation Platform upgrade to 2, one of the prerequisites is to enable the automation mesh port on the mesh nodes(execution and hop nodes). The default port used for the mesh networks on all the nodes is set to 27199/tcp, however it can also be configured to use a different port by specifying **receptor_listener_port** as each node's variable within your *inventory file*. More details on modifying the inventory file can be found within Section 4.3, "Upgrade *Environment B* to Ansible Automation Platform 2"

NOTE

For this reference environment all the Ansible Automation Platform 2 controller nodes are designated as node type control. If control nodes are designated as hybrid nodes (default node type), they require mesh port(default: 27199/tcp) to be enabled.

Within your hop and execution node, as the ansible user, set the firewalld port to be used for installation.

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

   ```
   $ sudo systemctl status firewalld
   ```

2. Add the **firewalld** port on 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
   ```

CHAPTER 4. INFRASTRUCTURE MIGRATION

To achieve a successful migration from Ansible Automation Platform 1.2 to Ansible Automation Platform 2, this reference environment takes advantage of the capabilities of the Ansible Automation Platform installer.

Using the Ansible Automation Platform installer, you'll be able to backup, import and upgrade to the latest Ansible Automation Platform 2 with a few simple commands.

The following sections provide a step-by-step of that process.

4.1. BACKUP ANSIBLE AUTOMATION PLATFORM 1.2 ON *ENVIRONMENT A*

As our Ansible Automation Platform 1.2 environment from *Environment A* contains all of our data, the following creates a backup using the Ansible Automation Platform installer on *Environment A*.

WARNING

Prior to taking a backup, ensure there are no current running jobs or future jobs scheduled to run. Any data collected after the backup is taken will be **LOST**.

Within *Environment A*,

1. Login as the **ansible** user

```
$ ssh ansible@enva_controller1.example.com
```

NOTE

This reference environment uses **enva_controller1** as the host that contains the Ansible Automation Platform installer directory and binaries.

2. Change to the **ansible-tower-setup-3.8.5-X** directory

```
$ cd /path/to/ansible-tower-setup-3.8.5-X
```

3. Run the Ansible Automation Platform installer to create a backup

   a. **backup_dest** provides the location where to store the backup of your Ansible Automation Platform database

   b. **use_compression** shrinks the size of the Ansible Automation Platform database backup

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

d. **-- --ask-vault-pass** asks for the password used to access the encrypted **credentials.yml** file

e. **-b** sets the create a backup option to True

```
$ ./setup.sh -e 'backup_dest=<mount_point>' -e 'use_compression=True' -e
@credentials.yml -b
```

NOTE

This reference environment takes advantage of encrypted credentials and does not include passwords in plain text. Details in *Appendix C, Creating an encrypted credentials.yml file* can be found on how to use **ansible-vault** to encrypt your credentials.

NOTE

The backup process may take some time to complete.

4.2. IMPORT ANSIBLE AUTOMATION PLATFORM 1.2 DATABASE TO *ENVIRONMENT B*

With the backup from *Environment A* created and available, the following imports the backed up Ansible Automation Platform database using the Ansible Automation Platform installer on *Environment B*.

Within *Environment B*,

1. Login as the **ansible** user

```
$ ssh ansible@envb_controller1.example.com
```

NOTE

This reference environment uses **envb_controller1** as the host that contains the Ansible Automation Platform installer directory and binaries.

2. Change to the **ansible-tower-setup-3.8.5-X** directory

```
$ cd /path/to/ansible-tower-setup-3.8.5-X
```

3. Run the Ansible Automation Platform installer to import the Ansible Automation Platform database

a. **restore_backup_file** provides the location of the backed up Ansible Automation Platform database

b. **use_compression** is set to True due to compression being used during the backup process

c. **-r** sets the restore database option to True

```
$ ./setup.sh -e ‘restore_backup_file=<mount_point>/tower-backup-latest.tar.gz -e
‘use_compression=True’ -e @credentials.yml -r -- --ask-vault-pass
```

NOTE

This reference environment takes advantage of encrypted credentials and does not include passwords in plain text. Details in *Appendix C, Creating an encrypted credentials.yml file* can be found on how to use **ansible-vault** to encrypt your credentials.

NOTE

The import process may take some time to complete.

4.3. UPGRADE *ENVIRONMENT B* TO ANSIBLE AUTOMATION PLATFORM 2

With the successful import of the Ansible Automation Platform database, the final step in the migration process is to upgrade the *Environment B* Ansible Automation Platform 1.2 environment to Ansible Automation Platform 2 and expand the architecture of *Environment B* as shown in Figure 1.4, "Expanded Environment B Architecture Overview".

Within *Environment B*,

1. Login as the **ansible** user

```
$ ssh ansible@envb_controller1.example.com
```

NOTE

This reference environment uses **envb_controller1** as the host that contains the Ansible Automation Platform installer directory and binaries.

2. Download Ansible Automation Platform 2.1.1 Setup tar ansible-automation-platform-setup-2.1.1-1.tar.gz

NOTE

For disconnected installs, download the Ansible Automation Platform 2.1.1 Setup Bundle

3. Untar the ansible-automation-platform-setup-2.1.1-1.tar.gz

```
$ tar zxvf ansible-automation-platform-setup-2.1.1-1.tar.gz
```

4. Change directory into ansible-automation-platform-setup-2.1.1-1

```
$ cd ansible-automation-platform-setup-2.1.1-1/
```

5. Copy the Ansible Automation Platform 1.2 inventory file to the ansible-automation-platform-setup-2.1.1-1 directory

```
$ cp /path/to/ansible-tower-setup-3.8.5-X/inventory .
```

6. Generate an Ansible Automation Platform 2 installation inventory proposal using the Ansible Automation Platform 1.2 inventory file copied over using the Ansible Automation Platform installer

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

NOTE

ansible-core is installed during this process if not already installed.

WARNING

Expect the Ansible Automation Platform installer to fail early in the process when creating the proposal *inventory.new.ini*.

Expected error task look as follows:

```
TASK [ansible.automation_platform_installer.check_config_static :
Detect pre-2.x inventory and offer a migration] ***
fatal: [172.16.58.48 -> localhost]: FAILED! => {"changed": false, "msg":
"The installer has detected that you are using an inventory format from a
version prior to 4.0. We have created an example inventory based on
your old style inventory. Please check the file `/home/ansible/aap_install-
2.1.1/ansible-automation-platform-setup-bundle-2.1.1-
2/inventory.new.ini` and make necessary adjustments so that the file can
be used by the installer."}
```

Proposed *inventory.new.ini*

```
[all:vars]
pg_host='10.0.188.133'
pg_port='5432'
pg_database='awx'
pg_username='awx'
pg_sslmode='prefer'
ansible_become='true'
ansible_user='ansible'
tower_package_name='automation-controller'
tower_package_version='4.1.1'
automationhub_package_name='automation-hub'
automationhub_package_version='4.4.1'
automation_platform_version='2.1.1'
automation_platform_channel='ansible-automation-platform-2.1-for-rhel-8-x86_64-rpms'
minimum_ansible_version='2.11'

# 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]
envb_controller1.example.com
envb_controller2.example.com
envb_controller3.example.com

[database]
envb_database.example.com
```

NOTE

The variables **admin_password**, **pg_password** and **registry_password** are not part of the *inventory.new.ini* file as it is not recommended to store passwords in plain text. An encrypted *credentials.yml* file is used instead.

7. With the proposed **inventory.new.ini** created, modify the file to include the expanded architecture of *Environment B* that includes hop nodes and execution nodes

Expanded *Environment B inventory.new.ini*

```
[all:vars]
pg_host='envb_database.example.com'
pg_port='5432'
pg_database='awx'
pg_username='awx'
pg_sslmode='prefer'
ansible_become='true'
ansible_user='ansible'
tower_package_name='automation-controller'
tower_package_version='4.1.1'
automationhub_package_name='automation-hub'
automationhub_package_version='4.4.1'
automation_platform_version='2.1.1'
automation_platform_channel='ansible-automation-platform-2.1-for-rhel-8-x86_64-rpms'
minimum_ansible_version='2.11'
registry_url='registry.redhat.io' ❶
registry_username='myusername' ❷

# 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]
envb_controller1.example.com
envb_controller2.example.com
envb_controller3.example.com

[database]
envb_database.example.com

[automationcontroller:vars]
node_type=control ❸
peers=envb_datacenter_execution_nodes,envb_datacenter_hop_nodes ❹
```

```
[execution_nodes]
envb_executionnode-1.example.com
envb_executionnode-2.example.com
envb_hopnnode-sacramento.example.com node_type=hop
peers=sacramento_execution_nodes ❺
envb_hopnode-new-delhi.example.com node_type=hop peers=new-delhi_execution_nodes
envb_hopnode-dublin.example.com node_type=hop peers=env_hopnode-new-
delhi.example.com
envb_executionnode-3.example.com
envb_executionnode-4.example.com
envb_executionnode-5.example.com
envb_executionnode-6.example.com

[envb_datacenter_execution_nodes] ❻
envb_executionnode-1.example.com
envb_executionnode-2.example.com

[envb_datacenter_hop_nodes] ❼
envb_hopnnode-sacramento.example.com
envb_hopnode-new-delhi.example.com
envb_hopnode-dublin.example.com

[sacramento_execution_nodes] ❽
envb_executionnode-3.example.com
envb_executionnode-4.example.com

[new-delhi_execution_nodes] ❾
envb_executionnode-5.example.com
envb_executionnode-6.example.com
```

❶ Execution Environment images are downloaded and included in your installation. Proper credentials required to download the images.

❷ User credential for access to registry_url.

❸ control nodes run project and inventory updates and system jobs, but not execution jobs. Execution capabilities are disabled on these nodes.

❹ Setting peer relationships between the execution nodes.

❺ Setting node type and peer relationships between the hop nodes and execution nodes.

❻ Group of execution nodes with direct connection access to the automation controller nodes.

❼ Group of hop nodes that route traffic to their corresponding execution nodes.

❽ Group of execution nodes accessible via *envb_hopnode-sacramento.example.com*

❾ Group of execution nodes accessible via *envb_hopnode-new-delhi.example.com*

8. Run the **setup.sh** to upgrade to Ansible Automation Platform 2 with the following options

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

–

9. Verify the Ansible Automation Platform dashboard UI is accessible across all automation controller nodes.

> **NOTE**
>
> If you experience 502 error or a Secure Connection Failed when accessing the the Ansible Automation Platform dashboard via any of your automation controllers, this is likely due to one or both of the following issues:
>
> - Certificate mismatch
> - Incorrect SELinux context for **nginx**
>
> The Appendix D, *Post upgrade playbook* provides a workaround to fix these issues. A fix is currently being implemented and should be fixed in an upcoming dot release.
>
> The cert mismatch issue is fixed in version 2.1.2 and later of Ansible Automation Platform. The incorrect SELinux context for **nginx** still requires the workaround Ansible Playbook. Check the Appendix D, *Post upgrade playbook* for more details.

This reference environment uses *credentials.yml* for the following variables: * **admin_password** * **registry_password** * **pg_password**

For more information regarding the different values that can be set within your inventory file, visit: Setting up the inventory file

4.4. CONFIGURING INSTANCE AND INSTANCE GROUPS

With the upgrade process complete, you'll need to associate your instances to their corresponding instance groups, e.g. **sacramento** and **new-delhi**.

1. Select **Administration→Instance Groups**
2. Click on the *sacramento* instance group
3. Select the **Instances** tab
4. Click the blue *Associate* button
5. Within the **Select Instances** window, select
   a. *envb_executionnode-3.example.com*
   b. *envb_executionnode-4.example.com*
6. Click **Save**

Repeat the process for the *new-delhi* instance group and associate the instances below with the *new-delhi* instance group:

- *envb_executionnode-5.example.com*
- *envb_executionnode-6.example.com*

Once complete, disassociate those instances within the *default* group.

1. Select **Administration→Instance Groups**
2. Click on the *default* instance group
3. Select the *Instances* tab
4. Select the checkbox to the following instances
    a. *envb_executionnode-3.example.com*
    b. *envb_executionnode-4.example.com*
    c. *envb_executionnode-5.example.com*
    d. *envb_executionnode-6.example.com*
5. Click the blue button labeled *Disassociate*
6. Confirm the dissociation via the red *Disassociate* button

The *default* instance group should only contain the following instances:

- *envb_executionnode-1.example.com*
- *envb_executionnode-2.example.com*

With the infrastructure migration complete, the focus shifts to migrating Python virtual environments to user-built execution environments.

CHAPTER 5. MIGRATE VIRTUAL ENVIRONMENTS TO EXECUTION ENVIRONMENTS

Ansible Automation Platform 2 comes with a re-imagined architecture that fully decouples the automation control plane and execution plane. The new capabilities enable easier to scale automation across the globe and allow you to run your automation as close to the source as possible without being bound to running automation in a single data center. It's more dynamic, scalable, resilient and flexible compared to Ansible Automation Platform 1.2.

With the introduction of automation execution environments, these container images allow for all the automation needed to be packaged and run including the key components such as Ansible Core, Ansible Content Collections, Python dependencies, Red Hat Enterprise Linux UBI 8, and any additional package dependencies.

This chapter focuses on migrating your custom Python virtual environments in your Ansible Automation Platform 1.2 cluster to user-built automation execution environments.

This one-time effort opens the door to take advantage of the latest Ansible Automation Platform 2 capabilities and the ability to execute consistent automation across multiple platforms with lower long-term maintenance.

WARNING

In order to access user-built execution environments, they are required to be hosted within private automation hub or a container registry. For more information on how to install private automation hub visit our Deploying Ansible Automation Platform 2.1 reference architecture.

5.1. AUTOMATING THE MIGRATION OF VIRTUAL ENVIRONMENTS TO EXECUTION ENVIRONMENTS

For simplicity, we are including supplemental Ansible Playbooks that automate the process by simply running an Ansible command.

For completeness, the manual process consists of:

1. An Ansible Automation Platform 1.2 environment with custom Python virtual environments
2. Using of the **awx-manage** command line utility to get a custom list of Python virtual environments
3. Running the **awx-manage export_custom_venv** command on each Python virtual environment to get the list of Python packages installed
4. Checking the association of a Python virtual environment using the **awx-manage custom_venv_associations** command
5. Filtering the above information to create execution environments using the **ansible-builder** tool

The automated process consists of:

1. Pulling a list of packages from each custom Python virtual environments present on the Ansible Automation Platform 1.2 environment

2. Comparing the package lists from the previous step with the package list of the Ansible-2.9[1] to find the packages that are not present in the base Ansible-2.9 execution environment

3. Create a new custom execution environment that uses the Ansible-2.9 execution environment as the base and including the missing dependencies from the list in the previous step

To run a scenario of what that may look like, let's take the following example.

In our existing Ansible Automation Platform 1.2, there are two custom Python virtual environments labeled *custom-venv1* and *custom-venv2*.

Using the role **virtualenv_migrate** packaged in the redhat_cop.ee_utilities collection we will run it against our Ansible Automation Platform 1.2 environment via **ssh** access to the Ansible Tower node to extract the the packages and their versions that are not currently part of the base execution environment we are comparing against (Ansible 2.9 execution environment).

NOTE

The *redhat_cop.ee_utilities* collection is a community project and officially not supported by Red Hat.

A sample playbook and inventory file respectively of the environment are found below:

playbook.yml

```
---
- name: Review custom virtualenvs and pull requirements
  hosts: enva_tower
  become: true
  tasks:
    - name: Include venv role
      include_role:
        name: redhat_cop.ee_utilities.virtualenv_migrate
```

Inventory

```
[tower]
ansibletower.example.com
ansible_ssh_private_key_file=/path/to/example.pem


[all:vars]
################################################################################
# Required configuration variables for migration from venv -> EE               #
################################################################################

# The default URL location to the execution environment (Default Ansible 2.9)
# If you want to use the newest Ansible base, change to: ee-minimal-rhel8:latest
venv_migrate_default_ee_url="registry.redhat.io/ansible-automation-platform-21/ee-29-rhel8:latest"
```

```
# User credential for access to venv_migrate_default_ee_url
registry_username='myusername'
```

NOTE

- Add ansible_user=<ANSIBLE_USER> based on the user needed to **ssh** into the Ansible Tower node.
- This reference environment takes advantage of encrypted credentials and does not include passwords in plain text. Details in Appendix C, *Creating an encrypted credentials.yml file* can be found on how to use **ansible-vault** to encrypt your registry credentials. An encrypted *credentials.yml* file is used to supply **registry_password**

WARNING

This role requires sudo privileges in order to run the **podman** commands.

The sample output of the Ansible playbook shows a list of the additional packages that are required for each custom Python virtual environment. In this case you'll notice that *custom-venv1* Python virtual environment requires the following packages in additional to what is already part of the Ansible 2.9 execution environment:

- certifi
- charset-normalizer
- enum34
- future
- solidfire-sdk-python

While *custom-venv2* Python virtual environment only required **zabbix-api** in addition to what is already part of the standard Ansible-2.9 execution environment.

NOTE

The Ansible 2.9 execution environment is used for comparison against the custom Python virtual environments because of the fact that most Ansible Automation Platform 1.2 environments are using Ansible 2.9. This ensures an easier migration transition due to backward compatibility.

```
TASK [redhat_cop.tower_utilities.virtualenv_migrate : diff | Show the packages that are extra from
default EEs in custom venvs.] ************************************************************************
ok: [3.228.23.40 -> localhost] => {
    "msg": [
        {
            "/opt/my-envs/custom-venv1/": [
                "certifi",
                "charset-normalizer",
                "enum34",
                "future",
                "solidfire-sdk-python"
            ]
        },
        {
            "/opt/my-envs/custom-venv2/": [
                "zabbix-api"
            ]
        }
    ]
}
```

Once the packages are captured for each custom Python virtual environment, the Ansible playbook uses the **ee_builder** role that is part of the redhat_cop.ee_utilities collection that automates the creation of execution environments on the local user environment.

Before running the supplied Ansible playbook, install ansible-builder on your localhost machine, the playbook run creates execution environments on your local machine based upon the package delta between the custom Python virtual environment and the base execution environment supplied.

```
$ podman images

REPOSITORY                  TAG         IMAGE ID
localhost/custom-venv2       latest      c017418d1919
localhost/custom-venv1       latest      7cbe3b49974d
localhost/custom-venv       latest      9d5d809f38b0
```

5.1.1. Pushing to private automation hub

With the local execution environments in place, you can push them to your private automation hub via:

NOTE

In this reference architecture, we keep the name of the automation execution environments as the name of the custom Python virtual environments for simplicity. If a change in name is required, use the podman tag command before pushing the execution environments to your private automation hub or a container registry of choice.

```
$ podman login [automation-hub-url]
# Enter the username and password to access Private Automation Hub.
$ podman tag [image-id] [automation-hub-url]/[container image name]
$ podman push [automation-hub-url]/[container image name]
```

TIP

For more information, visit: Managing containers in private automation hub

Once there, synchronize the execution environments to your automation controller by creating the registry credentials for your private automation hub inside the controller user interface.

To create your registry credentials within automation controller:

1. Select **Resources→Credentials**
2. Within **Credentials**, select the blue *Add* button
3. In the **Create New Credentials** window,
    a. provide a **Name** e.g. *My private automation hub credentials*
    b. under **Credential Type** select the drop down and select *Container Registry*
    c. under **Type Details**
        i. provide Authentication URL, e.g. *pah.example.com*
        ii. provide your private automation hub username within **Username** field
        iii. provide your private automation hub password or token within **Password or Token** field
        iv. select **Verify SSL** if your private automation hub environment supports SSL
4. Click **Save**

To make the execution environments available within automation controller, create a new execution environment that will pull the images from your private automation hub.

Within automation controller,

1. Select **Administration→Execution Environments**
2. Within **Execution Environments**, select the blue *Add* button
3. In the **Create new execution environment** window,
    a. provide a **Name**, e.g. *my execution environment*
    b. provide the image location of the execution environment, e.g. *repo/project/image-name:tag*
    c. select the **Registry credential** magnifying glass
        i. click the radio button for your private automation hub credentials, e.g. my private automation hub credentials

With the execution environments now available within automation controller, they can be used against any existing Job Templates or newly created Job Templates.

TIP

When creating new user-built execution environments not constrained to require backward compatibility, it is recommended to use the **ee-minimal** execution environment as your base execution environment to build your new images against.

[1] This reference architecture uses the **ansible-2.9** execution environment to best mimic the execution plane environment of Ansible Automation Platform 1.2.

APPENDIX A. ABOUT THE AUTHORS

Anshul Behl

Anshul is a Technical Marketing Manager at Red Hat, where he bring his software development and QE experience to increase Ansible Automation Platform's adoption experience for customers by producing technical content on all aspects of the product.

Roger Lopez

Roger Lopez is a Principal Technical Marketing Manager bringing 10+ years of computer industry experience delivering high-value solutions used by our sales, marketing and engineering teams to develop best practice documentation & methods for internal and external customers. He is a Red Hat Certified Engineer (RHCE) with experience building solutions around Ansible, OpenShift and OpenStack.

APPENDIX B. CONTRIBUTORS

We would like to thank the following individuals for their time and patience as we collaborated on this process. This document would not have been possible without their many contributions.

Contributor	Title	Contribution
Julen Landa Alustiza	Senior Software Quality Engineer	Technical Review
Craig Brandt	Principal Technical Marketing Manager	Content Review

APPENDIX C. CREATING AN ENCRYPTED *CREDENTIALS.YML* FILE

This section describes how to create an encrypted *credentials.yml* file that is passed to the Ansible Automation Platform installer.

WARNING

The passwords used must match between *Environment A* and *Environment B*.

Within your initial *Environment B* Ansible Automation Platform environment,

1. Create a *credentials.yml* file to store the encrypted credentials

```
$ cat credentials.yml
```

```
admin_password: my_long_admin_password
pg_password: my_long_pg_password
registry_password: my_long_registry_password
```

2. Encrypt the *credentials.yml* file using **ansible-vault**

```
$ ansible-vault encrypt credentials.yml
```

```
New Vault password:
Confirm New Vault password:
Encryption successful
```

WARNING

The **admin_password** and **pg_password** credentials need to match the value used in your Ansible Automation Platform 1.2 *Environment A*.

WARNING

Ensure to store your encrypted vault password in a safe location.

3. Verify the credentials.yml file is encrypted

```
$ cat credentials.yml
```

```
$ANSIBLE_VAULT;1.1;AES256
363836396535623865343163333339613833363064653364656138313534353135303764646
16539
376539306333030653234666633306462323630653166663103a3730623031333763396338313
03033
343135343839626136323037616366326239326530623438396136396536356433656162333
13365
3636616639313864300a35323937343331333961346532633931303563356535346435653866
53631
633464343835346432376638623533613666326136343332313163343639393964613265616
43336
34306335343039356462646330343839663362323033365383763
```

NOTE

The creation of encrypted credentials is optional but recommended as storing passwords within your inventory file should not be in plain text.

If you are already using encrypted credentails, use your credentials.yml file instead of creating a new one.

APPENDIX D. POST UPGRADE PLAYBOOK

This section provides an Ansible Playbook that you should run if you are not able to access the automation controller UI on all automation controller nodes after the migration. This addresses the SELinux context and the certificate mismatch issues that were described in the Chapter 4, *Infrastructure Migration* section.

Copy the below Ansible Playbook content and place it in a file called **post_upgrade_playbook.yml** inside the untarred installer directory.

NOTE

This directory consists of the installer inventory file and this playbook uses the installer inventory to make some changes to your automation controller nodes.

post_upgrade_playbook.yml

```
---
- name: Play to apply workaround to known issues in upgrade
  hosts: automationcontroller
  become: true
  tasks:
    - block:
      - name: Remove certs from all the controllers
        file:
          name: "{{ item }}"
          state: absent
        loop:
          - /etc/tower/tower.cert
          - /etc/tower/tower.key
      - name: Role to create new certs and copy to all controllers
        include_role:
          name: ansible.automation_platform_installer.nginx
      when:
        - automation_platform_version is version('2.1.1', '<=')
    - name: Add to targeted policy and apply selinux policy to controller dirs
      ansible.builtin.command: "{{ item }}"
      loop:
        - semodule -s targeted -i /usr/share/selinux/targeted/automation-controller.pp
        - /sbin/restorecon -R /var/lib/awx/venv /var/lib/awx/job_status /var/run/tower
    - name: Restart the controller service
      service:
        name: automation-controller
        state: restarted
```

Run the command to execute the playbook on the controller nodes.

```
$ ansible-playbook -i inventory.new.ini post_upgrade_playbook.yml
```

APPENDIX E. REVISION HISTORY

Revision 1.1-0 2022-04-29 Roger Lopez

- Added NOTE within Overview regarding using same subscription manifest when migrating between versions of Ansible Automation Platform

Revision 1.0-0 2022-03-31 Anshul Behl and Roger Lopez

- Initial Release