Red Hat Ansible Automation Platform 2.5

Containerized installation

Install the containerized version of Ansible Automation Platform

Last Updated: 2025-09-24

Red Hat Ansible Automation Platform 2.5 Containerized installation

Install the containerized version of Ansible Automation Platform

Legal Notice

Abstract

This guide helps you to understand the installation requirements and processes behind our containerized version of Ansible Automation Platform.

Table of Contents

PROVIDING FEEDBACK ON RED HAT DOCUMENTATION

If you have a suggestion to improve this documentation, or find an error, you can contact technical support at https://access.redhat.com to open a request.

CHAPTER 1. MANAGING ANSIBLE AUTOMATION PLATFORM LICENSING, UPDATES AND SUPPORT

Ansible is an open source software project and is licensed under the GNU General Public License version 3, as described in the Ansible Source Code .

You must have valid subscriptions attached before installing Ansible Automation Platform.

1.1. TRIAL AND EVALUATION

A license is required to run Ansible Automation Platform. You can start by using a free trial license.

- Trial licenses for Ansible Automation Platform are available at the Red Hat product trial center .
- Support is not included in a trial license or during an evaluation of the Ansible Automation Platform.

1.2. COMPONENT LICENSES

To view the license information for the components included in Ansible Automation Platform, see **/usr/share/doc/automation-controller-<version>/README**.

where **<version>** refers to the version of automation controller you have installed.

To view a specific license, see **/usr/share/doc/automation-controller-<version>/*.txt**.

where * is the license file name to which you are referring.

1.3. NODE COUNTING IN LICENSES

The Ansible Automation Platform license defines the number of Managed Nodes that can be managed as part of your subscription.

A typical license says "License Count: 500", which sets the maximum number of Managed Nodes at 500.

For more information about managed node requirements for licensing, see How are "managed nodes" defined as part of the Red Hat Ansible Automation Platform offering.

NOTE

Ansible does not recycle node counts or reset automated hosts.

1.4. SUBSCRIPTION TYPES

Red Hat Ansible Automation Platform is provided at various levels of support and number of machines as an annual subscription.

- **Standard:**
  - Manage any size environment
  - Enterprise 8x5 support and SLA

- Maintenance and upgrades included
- Review the SLA at Product Support Terms of Service
- Review the Red Hat Support Severity Level Definitions

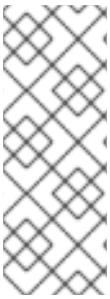
- **Premium:**
  - Manage any size environment, including mission-critical environments
  - Premium 24x7 support and SLA
  - Maintenance and upgrades included
  - Review the SLA at Product Support Terms of Service
  - Review the Red Hat Support Severity Level Definitions

All subscription levels include regular updates and releases of automation controller, Ansible, and any other components of the Ansible Automation Platform.

For more information, contact Ansible through the Red Hat Customer Portal or at the Ansible site.

1.5. ATTACHING YOUR RED HAT ANSIBLE AUTOMATION PLATFORM SUBSCRIPTION

You **must** have valid subscriptions on all nodes before installing Red Hat Ansible Automation Platform.



NOTE

Simple Content Access (SCA) is now the default subscription method for all Red Hat accounts. With SCA, you only need to register your systems to Red Hat Subscription Management (RHSM) or Satellite to access content. Traditional pool-based subscription attachment commands (such as **subscription-manager attach --pool** or **subscription-manager attach --auto**) are no longer required. For more information, see Simple Content Access.

Procedure

1. Register your system with Red Hat Subscription Management:

```
$ sudo subscription-manager register --username <$INSERT_USERNAME_HERE> --password <$INSERT_PASSWORD_HERE>
```

With Simple Content Access (SCA), registration is the only step required to access Ansible Automation Platform content.

NOTE

For accounts still using legacy subscription pools, you might need to manually attach subscriptions using the commands shown in the troubleshooting section.

Verification

1. Refresh the subscription information on your system:

```
$ sudo subscription-manager refresh
```

2. Verify your registration:

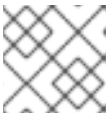
```
$ sudo subscription-manager identity
```

This command displays your system identity, name, organization name, and organization ID, confirming successful registration.

Troubleshooting

- For legacy accounts not using SCA, you might need to manually attach subscriptions:

```
$ sudo subscription-manager list --available --all | grep "Ansible Automation Platform" -B 3 -A
6
$ sudo subscription-manager attach --pool=<pool_id>
```



NOTE

Do not use MCT4022 as a **pool_id** as it can cause subscription attachment to fail.

- If you are unable to locate certain packages that came bundled with the Ansible Automation Platform installer, or if you are seeing a ***Repositories disabled by configuration*** message, try enabling the repository by using the command:
Ansible Automation Platform 2.5 for RHEL 9

```
$ sudo subscription-manager repos --enable ansible-automation-platform-2.5-for-rhel-9-
x86_64-rpms
```

Ansible Automation Platform 2.5 for RHEL 10

```
$ sudo subscription-manager repos --enable ansible-automation-platform-2.5-for-rhel-10-
x86_64-rpms
```

1.6. OBTAINING A MANIFEST FILE

You can obtain a subscription manifest in the Subscription Allocations section of Red Hat Subscription Management. After you obtain a subscription allocation, you can download its manifest file and upload it to activate Ansible Automation Platform.

To begin, login to the Red Hat Customer Portal using your administrator user account and follow the procedures in this section.

1.6.1. Create a subscription allocation

Creating a new subscription allocation allows you to set aside subscriptions and entitlements for a system that is currently offline or air-gapped. This is necessary before you can download its manifest and upload it to Ansible Automation Platform.

Procedure

1. From the Subscription Allocations page, click **New Subscription Allocation**.
2. Enter a name for the allocation so that you can find it later.
3. Select **Type: Satellite 6.16** as the management application.
4. Click **Create**.

Next steps

- Add the subscriptions.

1.6.2. Adding subscriptions to a subscription allocation

Once an allocation is created, you can add the subscriptions you need for Ansible Automation Platform to run properly. This step is necessary before you can download the manifest and add it to Ansible Automation Platform.

Procedure

1. From the Subscription Allocations page, click on the name of the **Subscription Allocation** to which you would like to add a subscription.
2. Click the **Subscriptions** tab.
3. Click **Add Subscriptions**.
4. Enter the number of Ansible Automation Platform Entitlement(s) you plan to add.
5. Click **Submit**.

Next steps

- Download the manifest file.

1.6.3. Downloading a manifest file

After an allocation is created and has the appropriate subscriptions on it, you can download the manifest from Red Hat Subscription Management.

Procedure

1. From the Subscription Allocations page, click on the name of the **Subscription Allocation** to which you would like to generate a manifest.
2. Click the **Subscriptions** tab.
3. Click **Export Manifest** to download the manifest file.
   This downloads a file *manifest*<allocation name>_<date>.zip_ to your default downloads folder.

Next steps

- Upload the manifest file.

1.7. ACTIVATING RED HAT ANSIBLE AUTOMATION PLATFORM

If you are an organization administrator, you must create a service account and use the client ID and client secret to activate your subscription.

If you do not have administrative access, you can enter your Red Hat username and password in the Client ID and Client secret fields, to locate and add your subscription to your Ansible Automation Platform instance.

NOTE

If you enter your client ID and client secret but cannot locate your subscription, you might not have the correct permissions set on your service account. For more information and troubleshooting guidance for service accounts, see Configure Ansible Automation Platform to authenticate through service account credentials.

For Red Hat Satellite, input your Satellite username and Satellite password in the fields below.

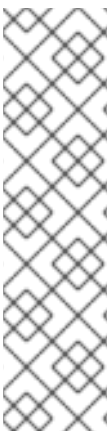
Red Hat Ansible Automation Platform uses available subscriptions or a subscription manifest to authorize the use of Ansible Automation Platform. To obtain a subscription, you can do either of the following:

1. Use your Red Hat username and password, service account credentials, or Satellite credentials when you launch Ansible Automation Platform.
2. Upload a subscriptions manifest file either using the Red Hat Ansible Automation Platform interface or manually in an Ansible Playbook.

1.7.1. Activate with credentials

When Ansible Automation Platform launches for the first time, the Ansible Automation Platform Subscription screen automatically displays. If you are an organization administrator, you can use your Red Hat service account to retrieve and import your subscription directly into Ansible Automation Platform.

If you do not have administrative access, you can enter your Red Hat username and password in the Client ID and Client secret fields, respectively, to locate and add your subscription to your Ansible Automation Platform instance.



NOTE

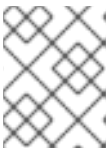
You are opted in for Automation Analytics by default when you activate the platform on first time log in. This helps Red Hat improve the product by delivering you a much better user experience. You can opt out, after activating Ansible Automation Platform, by doing the following:

1. From the navigation panel, select **Settings → Automation Execution → System**.
2. Clear the **Gather data for Automation Analytics**option.
3. Click **Save**.

Procedure

1. Log in to Red Hat Ansible Automation Platform.

2. Select **Service Account / Red Hat Satellite**
3. Enter your **Client ID / Satellite username** and **Client secret / Satellite password**
4. Select your subscription from the **Subscription** list.



NOTE

You can also use your Satellite username and password if your cluster nodes are registered to Satellite through Subscription Manager.

5. Review the End User License Agreement and select **I agree to the End User License Agreement**.
6. Click **Finish**.

Verification

After your subscription has been accepted, subscription details are displayed. A status of *Compliant* indicates your subscription is in compliance with the number of hosts you have automated within your subscription count. Otherwise, your status will show as *Out of Compliance*, indicating you have exceeded the number of hosts in your subscription. Other important information displayed include the following:

Hosts automated

Host count automated by the job, which consumes the license count

Hosts imported

Host count considering all inventory sources (does not impact hosts remaining)

Hosts remaining

Total host count minus hosts automated

1.7.2. Activate with a manifest file

If you have a subscriptions manifest, you can upload the manifest file either by using the Red Hat Ansible Automation Platform interface.

You are opted in for Automation Analytics by default when you activate the platform on first time log in. This helps Red Hat improve the product by delivering you a much better user experience. However, you can opt out of Automation Analytics after the Ansible Automation Platform is activated. To opt out, select **Settings → Automation Execution → System** from the navigation panel, uncheck the **Gather data for Automation Analytics** option, and then click **Save**.

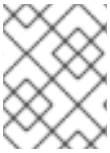
Prerequisites

You must have a Red Hat Subscription Manifest file exported from the Red Hat Customer Portal. For more information, see Obtaining a manifest file .

Procedure

1. Log in to Red Hat Ansible Automation Platform.
2. If you are not immediately prompted for a manifest file, go to **Settings → Subscription**.
3. Select **Subscription manifest**

4. Click **Browse** and select the manifest file.
5. Review the End User License Agreement and select **I agree to the End User License Agreement**.
6. Click **Finish**.



NOTE

If the **BROWSE** button is disabled on the License page, clear the **USERNAME** and **PASSWORD** fields.

Verification

After your subscription has been accepted, subscription details are displayed. A status of *Compliant* indicates your subscription is in compliance with the number of hosts you have automated within your subscription count. Otherwise, your status will show as *Out of Compliance*, indicating you have exceeded the number of hosts in your subscription. Other important information displayed include the following:

Hosts automated

Host count automated by the job, which consumes the license count

Hosts imported

Host count considering all inventory sources (does not impact hosts remaining)

Hosts remaining

Total host count minus hosts automated

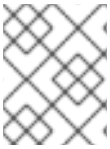
Next steps

- You can return to the license screen by selecting **Settings → Subscription** from the navigation panel and clicking **Edit subscription**.

CHAPTER 2. ANSIBLE AUTOMATION PLATFORM CONTAINERIZED INSTALLATION

Ansible Automation Platform is a commercial offering that helps teams manage complex multi-tier deployments by adding control, knowledge, and delegation to Ansible-powered environments.

This guide helps you to understand the installation requirements and processes behind the containerized version of Ansible Automation Platform.



NOTE

Upgrades from 2.4 Containerized Ansible Automation Platform Tech Preview to 2.5 Containerized Ansible Automation Platform are not supported.

2.1. TESTED DEPLOYMENT MODELS

Red Hat tests Ansible Automation Platform 2.5 with a defined set of topologies to give you opinionated deployment options. The supported topologies include infrastructure topology diagrams, tested system configurations, example inventory files, and network ports information.

For containerized Ansible Automation Platform, there are two infrastructure topology shapes:

1. Growth - (All-in-one) Intended for organizations that are getting started with Ansible Automation Platform. This topology allows for smaller footprint deployments.

2. Enterprise - Intended for organizations that require Ansible Automation Platform deployments to have redundancy or higher compute for large volumes of automation. This is a more future-proofed scaled out architecture.

For more information about the tested deployment topologies for containerized Ansible Automation Platform, see Container topologies in *Tested deployment models*.

2.2. SYSTEM REQUIREMENTS

Use this information when planning your installation of containerized Ansible Automation Platform.

2.2.1. Prerequisites

- Ensure a dedicated non-root user is configured on the Red Hat Enterprise Linux host.
    - This user requires **sudo** or other Ansible supported privilege escalation (**sudo** is recommended) to perform administrative tasks during the installation.
    - This user is responsible for the installation of containerized Ansible Automation Platform.
    - This user is also the service account for the containers running Ansible Automation Platform.
- For managed nodes, ensure a dedicated user is configured on each node. Ansible Automation Platform connects as this user to run tasks on the node. For more information about configuring a dedicated user on each node, see Preparing the managed nodes for containerized installation.

- For remote host installations, ensure SSH public key authentication is configured for the non-root user. For guidelines on setting up SSH public key authentication for the non-root user, see How to configure SSH public key authentication for passwordless login .
- Ensure internet access is available from the Red Hat Enterprise Linux host if you are using the default online installation method.
- Ensure the appropriate network ports are open if a firewall is in place. For more information about the ports to open, see Container topologies in *Tested deployment models*.

IMPORTANT

Storing container images on an NFS share is not supported by Podman. To use an NFS share for the user home directory, set up the Podman storage backend path outside of the NFS share. For more information, see Rootless Podman and NFS.

2.2.2. Ansible Automation Platform system requirements

Your system must meet the following minimum system requirements to install and run Red Hat Ansible Automation Platform.

Table 2.1. System configuration

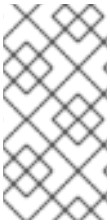
Type	Description	Notes
Subscription	• Valid Red Hat Ansible Automation Platform subscription • Valid Red Hat Enterprise Linux subscription (to consume the BaseOS and AppStream repositories)	
Operating system	• Red Hat Enterprise Linux 9.2 or later minor versions of Red Hat Enterprise Linux 9. • Red Hat Enterprise Linux 10 or later minor versions of Red Hat Enterprise Linux 10.	
CPU architecture	x86_64, AArch64, s390x (IBM Z), ppc64le (IBM Power)	

Type	Description	Notes
ansible-core	• RHEL 9: installation program uses **ansible-core** 2.14, Ansible Automation Platform operation uses **ansible-core** 2.16. • RHEL 10: installation program uses **ansible-core** 2.16, Ansible Automation Platform operation uses **ansible-core** 2.16.	• The installation program uses the **ansible-core** package from the RHEL AppStream repository. • Ansible Automation Platform bundles **ansible-core** 2.16 for operation, so you do not need to install it manually.
Browser	A currently supported version of Mozilla Firefox or Google Chrome.	
Database	PostgreSQL 15	External (customer supported) databases require ICU support.

Each virtual machine (VM) has the following system requirements:

Table 2.2. Virtual machine requirements

Requirement	Minimum requirement
RAM	16 GB
CPUs	4
Local disk	• Total available disk space: 60 GB • Installation directory: 15 GB (if on a dedicated partition) • **/var/tmp** for online installations: 1 GB • **/var/tmp** for offline or bundled installations: 3 GB • Temporary directory (defaults to **/tmp**) for offline or bundled installations: 10GB
Disk IOPS	3000



NOTE

If performing a bundled installation of the growth topology with **hub_seed_collections=true**, then 32 GB RAM is recommended. Note that with this configuration the install time is going to increase and can take 45 or more minutes alone to complete seeding the collections.

2.2.3. Database requirements

Ansible Automation Platform can work with two varieties of database:

1. Database installed with Ansible Automation Platform - This database consists of a PostgreSQL installation done as part of an Ansible Automation Platform installation using PostgreSQL packages provided by Red Hat.
2. Customer provided or configured database - This is an external database that is provided by the customer, whether on bare metal, virtual machine, container, or cloud hosted service.

Ansible Automation Platform requires customer provided (external) database to have ICU support.

Additional resources

- Red Hat Ansible Automation Platform Database Scope of Coverage
- Setting up a customer provided (external) database

2.3. PREPARING THE RED HAT ENTERPRISE LINUX HOST FOR CONTAINERIZED INSTALLATION

Containerized Ansible Automation Platform runs the component services as Podman based containers on top of a Red Hat Enterprise Linux host. Prepare the Red Hat Enterprise Linux host to ensure a successful installation.

Procedure

1. Log in to the Red Hat Enterprise Linux host as your non-root user.
2. Ensure the hostname associated with your host is set as a fully qualified domain name (FQDN).
   a. To check the hostname associated with your host, run the following command:

   ```
   hostname -f
   ```

   Example output:

   ```
   aap.example.org
   ```

   b. If the hostname is not a FQDN, you can set it with the following command:

   ```
   $ sudo hostnamectl set-hostname <your_hostname>
   ```

3. Register your Red Hat Enterprise Linux host with **subscription-manager**:

   ```
   $ sudo subscription-manager register
   ```

4. Verify that only the BaseOS and AppStream repositories are enabled on the host:

```
$ sudo dnf repolist
```

Example output for RHEL 9:

```
Updating Subscription Management repositories.
repo id                                 repo name
rhel-9-for-x86_64-appstream-rpms        Red Hat Enterprise Linux 9 for x86_64 -
AppStream (RPMs)
rhel-9-for-x86_64-baseos-rpms           Red Hat Enterprise Linux 9 for x86_64 -
BaseOS (RPMs)
```

Example output for RHEL 10:

```
Updating Subscription Management repositories.
repo id                                 repo name
rhel-10-for-x86_64-appstream-rpms       Red Hat Enterprise Linux 10 for x86_64 -
AppStream (RPMs)
rhel-10-for-x86_64-baseos-rpms          Red Hat Enterprise Linux 10 for x86_64 -
BaseOS (RPMs)
```

- For disconnected installations follow the steps in Obtaining and configuring RPM source dependencies to access these repositories.

5. Ensure the host can resolve host names and IP addresses using DNS. This is essential to ensure services can talk to one another.

6. Install **ansible-core**:

```
$ sudo dnf install -y ansible-core
```

7. Optional: You can install additional utilities that can be useful for troubleshooting purposes, for example **wget**, **git-core**, **rsync**, and **vim**:

```
$ sudo dnf install -y wget git-core rsync vim
```

8. Optional: To have the installation program automatically pick up and apply your Ansible Automation Platform subscription manifest license, follow the steps in Obtaining a manifest file .

Additional resources

- Attaching your Red Hat Ansible Automation Platform Subscription
- Setting up an unbound DNS server
- Setting up and configuring a BIND DNS server
- Ansible Core Documentation

2.4. PREPARING THE MANAGED NODES FOR CONTAINERIZED INSTALLATION

Managed nodes, also referred to as hosts, are the devices that Ansible Automation Platform is configured to manage.

To ensure a consistent and secure setup of containerized Ansible Automation Platform, create a dedicated user on each host. Ansible Automation Platform connects as this user to run tasks on the host.

Once configured, you can define the dedicated user for each host by adding **ansible_user=<username>** in your inventory file, for example: **aap.example.org ansible_user=aap**.

Complete the following steps for each host:

Procedure

1. Log in to the host as the root user.
2. Create a new user. Replace **<username>** with the username you want, for example **aap**.

```
$ adduser <username>
```

3. Set a password for the new user. Replace **<username>** with the username you created.

```
$ passwd <username>
```

4. Configure the user to run sudo commands.
   a. To do this open the sudoers file:

```
$ vi /etc/sudoers
```

   b. Add the following line to the file (replacing **<username>** with the username you created):

```
<username> ALL=(ALL) NOPASSWD: ALL
```

   c. Save and exit the file.

2.5. DOWNLOADING ANSIBLE AUTOMATION PLATFORM

Choose the installation program you need based on your Red Hat Enterprise Linux environment internet connectivity and download the installation program to your Red Hat Enterprise Linux host.

Prerequisites

- You have logged in to the Red Hat Enterprise Linux host as your non-root user.

Procedure

1. Download the latest version of containerized Ansible Automation Platform from the Ansible Automation Platform download page.
   a. For online installations: **Ansible Automation Platform 2.5 Containerized Setup**
   b. For offline or bundled installations: **Ansible Automation Platform 2.5 Containerized Setup Bundle**

2. Copy the installation program **.tar.gz** file and the optional manifest **.zip** file onto your Red Hat Enterprise Linux host.

   a. You can use the **scp** command to securely copy the files. The basic syntax for **scp** is:

   ```
   scp [options] <path_to_source_file> <path_to_destination>
   ```

   Use the following **scp** command to copy the installation program **.tar.gz** file to an AWS EC2 instance with a private key (replace the placeholder **<>** values with your actual information):

   ```
   scp -i <path_to_private_key> ansible-automation-platform-containerized-setup-<version_number>.tar.gz ec2-user@<remote_host_ip_or_hostname>:<path_to_destination>
   ```

3. Decide where you want the installation program to reside on the file system. This is referred to as your installation directory.

   a. Installation related files are created under this location and require at least 15 GB for the initial installation.

4. Unpack the installation program **.tar.gz** file into your installation directory, and go to the unpacked directory.

   a. To unpack the online installer:

   ```
   $ tar xfvz ansible-automation-platform-containerized-setup-<version_number>.tar.gz
   ```

   b. To unpack the offline or bundled installer:

   ```
   $ tar xfvz ansible-automation-platform-containerized-setup-bundle-<version_number>-<arch_name>.tar.gz
   ```

Additional resources

- scp(1) - Linux manual page

2.6. CONFIGURING THE INVENTORY FILE

You can control the installation of Ansible Automation Platform with inventory files. Inventory files define the information needed to customize the installation. For example, host details, certificate details, and various component-specific settings.

Example inventory files are available in this document that you can copy and change to quickly get started.

Additionally, growth topology and enterprise topology inventory files are available in the following locations:

- In the downloaded installation program package:
  - The default inventory file, named **inventory**, is for the enterprise topology pattern.
  - To deploy the growth topology (all-in-one) pattern, use the **inventory-growth** file instead.

- In Container topologies in *Tested deployment models*.

To use the example inventory files, replace the **< >** placeholders with your specific variables, and update the host names.

Refer to the **README.md** file in the installation directory or Inventory file variables for more information about optional and required variables.

2.6.1. Inventory file for online installation for containerized growth topology (all-in-one)

Use the example inventory file to perform an online installation for the containerized growth topology (all-in-one):

```
# This is the Ansible Automation Platform installer inventory file intended for the container growth deployment topology.
# This inventory file expects to be run from the host where Ansible Automation Platform will be installed.
# Consult the Ansible Automation Platform product documentation about this topology's tested hardware configuration.
#
https://docs.redhat.com/en/documentation/red_hat_ansible_automation_platform/2.5/html/tested_deployment_models/container-topologies
#
# Consult the docs if you are unsure what to add
# For all optional variables consult the included README.md
# or the Ansible Automation Platform documentation:
#
https://docs.redhat.com/en/documentation/red_hat_ansible_automation_platform/2.5/html/containerized_installation

# This section is for your platform gateway hosts
# -----------------------------------------------------
[automationgateway]
aap.example.org

# This section is for your automation controller hosts
# -----------------------------------------------------
[automationcontroller]
aap.example.org

# This section is for your automation hub hosts
# -----------------------------------------------------
[automationhub]
aap.example.org

# This section is for your Event-Driven Ansible controller hosts
# -----------------------------------------------------
[automationeda]
aap.example.org

# This section is for the Ansible Automation Platform database
# -----------------------------------------------------
[database]
aap.example.org
```

```
[all:vars]
# Ansible
ansible_connection=local

# Common variables
# https://docs.redhat.com/en/documentation/red_hat_ansible_automation_platform/2.5/html/containerized_installation/appendix-inventory-files-vars#general-variables
# ------------------------------------------------------
postgresql_admin_username=postgres
postgresql_admin_password=<set your own>

registry_username=<your RHN username>
registry_password=<your RHN password>

redis_mode=standalone

# Platform gateway
# https://docs.redhat.com/en/documentation/red_hat_ansible_automation_platform/2.5/html/containerized_installation/appendix-inventory-files-vars#platform-gateway-variables
# ------------------------------------------------------
gateway_admin_password=<set your own>
gateway_pg_host=aap.example.org
gateway_pg_password=<set your own>

# Automation controller
# https://docs.redhat.com/en/documentation/red_hat_ansible_automation_platform/2.5/html/containerized_installation/appendix-inventory-files-vars#controller-variables
# ------------------------------------------------------
controller_admin_password=<set your own>
controller_pg_host=aap.example.org
controller_pg_password=<set your own>
controller_percent_memory_capacity=0.5

# Automation hub
# https://docs.redhat.com/en/documentation/red_hat_ansible_automation_platform/2.5/html/containerized_installation/appendix-inventory-files-vars#hub-variables
# ------------------------------------------------------
hub_admin_password=<set your own>
hub_pg_host=aap.example.org
hub_pg_password=<set your own>
hub_seed_collections=false

# Event-Driven Ansible controller
# https://docs.redhat.com/en/documentation/red_hat_ansible_automation_platform/2.5/html/containerized_installation/appendix-inventory-files-vars#event-driven-ansible-variables
# ------------------------------------------------------
eda_admin_password=<set your own>
eda_pg_host=aap.example.org
eda_pg_password=<set your own>
```

- **ansible_connection=local** - Used for all-in-one installations where the installation program is run on the same node that hosts Ansible Automation Platform.
    - If the installation program is run from a separate node, do not include **ansible_connection=local**. In this case, use an SSH connection instead.

Additional resources

- Container growth topology

2.6.2. Inventory file for online installation for containerized enterprise topology

Use the example inventory file to perform an online installation for the containerized enterprise topology:

```
# This is the Ansible Automation Platform enterprise installer inventory file
# Consult the docs if you are unsure what to add
# For all optional variables consult the included README.md
# or the Red Hat documentation:
#
https://docs.redhat.com/en/documentation/red_hat_ansible_automation_platform/2.5/html/containerized_installation

# This section is for your platform gateway hosts
# -----------------------------------------------------
[automationgateway]
gateway1.example.org
gateway2.example.org

# This section is for your automation controller hosts
# -----------------------------------------------------
[automationcontroller]
controller1.example.org
controller2.example.org

# This section is for your Ansible Automation Platform execution hosts
# -----------------------------------------------------
[execution_nodes]
hop1.example.org receptor_type='hop'
exec1.example.org
exec2.example.org

# This section is for your automation hub hosts
# -----------------------------------------------------
[automationhub]
hub1.example.org
hub2.example.org

# This section is for your Event-Driven Ansible controller hosts
# -----------------------------------------------------
[automationeda]
eda1.example.org
eda2.example.org

[redis]
```

```
gateway1.example.org
gateway2.example.org
hub1.example.org
hub2.example.org
eda1.example.org
eda2.example.org

[all:vars]

# Common variables
#
https://docs.redhat.com/en/documentation/red_hat_ansible_automation_platform/2.5/html/containerized_installation/appendix-inventory-files-vars#general-variables
# ------------------------------------------------------
postgresql_admin_username=<set your own>
postgresql_admin_password=<set your own>
registry_username=<your RHN username>
registry_password=<your RHN password>

# Platform gateway
#
https://docs.redhat.com/en/documentation/red_hat_ansible_automation_platform/2.5/html/containerized_installation/appendix-inventory-files-vars#platform-gateway-variables
# ------------------------------------------------------
gateway_admin_password=<set your own>
gateway_pg_host=externaldb.example.org
gateway_pg_database=<set your own>
gateway_pg_username=<set your own>
gateway_pg_password=<set your own>

# Automation controller
#
https://docs.redhat.com/en/documentation/red_hat_ansible_automation_platform/2.5/html/containerized_installation/appendix-inventory-files-vars#controller-variables
# ------------------------------------------------------
controller_admin_password=<set your own>
controller_pg_host=externaldb.example.org
controller_pg_database=<set your own>
controller_pg_username=<set your own>
controller_pg_password=<set your own>

# Automation hub
#
https://docs.redhat.com/en/documentation/red_hat_ansible_automation_platform/2.5/html/containerized_installation/appendix-inventory-files-vars#hub-variables
# ------------------------------------------------------
hub_admin_password=<set your own>
hub_pg_host=externaldb.example.org
hub_pg_database=<set your own>
hub_pg_username=<set your own>
hub_pg_password=<set your own>

# Event-Driven Ansible controller
#
https://docs.redhat.com/en/documentation/red_hat_ansible_automation_platform/2.5/html/containerized_installation/appendix-inventory-files-vars#event-driven-ansible-variables
```

```
# ----------------------------------------------------
eda_admin_password=<set your own>
eda_pg_host=externaldb.example.org
eda_pg_database=<set your own>
eda_pg_username=<set your own>
eda_pg_password=<set your own>
```

Additional resources

- Container enterprise topology
- Caching and queueing system

2.6.3. Setting registry_username and registry_password

When using the **registry_username** and **registry_password** variables for an online non-bundled installation, you need to create a new registry service account.

Registry service accounts are named tokens that can be used in environments where credentials will be shared, such as deployment systems.

Procedure

1. Go to https://access.redhat.com/terms-based-registry/accounts.
2. On the **Registry Service Accounts** page click **New Service Account**.
3. Enter a name for the account using only the allowed characters.
4. Optionally enter a description for the account.
5. Click **Create**.
6. Find the created account in the list by searching for your name in the search field.
7. Click the name of the account that you created.
8. Alternatively, if you know the name of your token, you can go directly to the page by entering the URL:

   ```
   https://access.redhat.com/terms-based-registry/token/<name-of-your-token>
   ```

9. A **token** page opens, displaying a generated username (different from the account name) and a token.
   a. If no token is displayed, click **Regenerate Token**. You can also click this to generate a new username and token.
10. Copy the username (for example "1234567|testuser") and use it to set the variable **registry_username**.
11. Copy the token and use it to set the variable **registry_password**.

2.7. ADVANCED CONFIGURATION OPTIONS

Advanced configuration options, such as external database set up and the use of custom TLS certs, are available for more complex deployments of containerized Ansible Automation Platform.

If you are not using these advanced configuration options, go to Installing containerized Ansible Automation Platform to continue with your installation.

2.7.1. Adding a safe plugin variable to Event-Driven Ansible controller

When using **redhat.insights_eda** or similar plugins to run rulebook activations in Event-Driven Ansible controller, you must add a safe plugin variable to a directory in Ansible Automation Platform. This ensures connection between Event-Driven Ansible controller and the source plugin, and displays port mappings correctly.

Procedure

1. Create a directory for the safe plugin variable:

```
mkdir -p ./group_vars/automationeda
```

2. Create a file within that directory for your new setting (for example, **touch ./group_vars/automationeda/custom.yml**)

3. Add the variable **eda_safe_plugins** with a list of plugins to enable. For example:

```
eda_safe_plugins: ['ansible.eda.webhook', 'ansible.eda.alertmanager']
```

2.7.2. Adding execution nodes

Containerized Ansible Automation Platform can deploy remote execution nodes.

You can define remote execution nodes in the **[execution_nodes]** group of your inventory file:

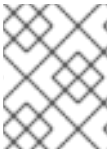
```
[execution_nodes]
<fqdn_of_your_execution_host>
```

By default, an execution node is configured with the following settings which can be modified as needed:

```
receptor_port=27199
receptor_protocol=tcp
receptor_type=execution
```

- **receptor_port** - The port number that receptor listens on for incoming connections from other receptor nodes.
- **receptor_type** - The role of the node. Valid options include **execution** or **hop**.
- **receptor_protocol** - The protocol used for communication. Valid options include **tcp** or **udp**.

By default, all nodes in the **[execution_nodes]** group are added as peers for the controller node. To change the peer configuration, use the **receptor_peers** variable.



NOTE

The value of **receptor_peers** must be a comma-separated list of host names. Do not use inventory group names.

Example configuration:

```
[execution_nodes]
# Execution nodes
exec1.example.com
exec2.example.com
# Hop node that peers with the two execution nodes above
hop1.example.com receptor_type=hop
receptor_peers='["exec1.example.com","exec2.example.com"]'
```

2.7.3. Configuring storage for automation hub

Configure storage backends for automation hub including Amazon S3, Azure Blob Storage, and Network File System (NFS) storage.

2.7.3.1. Configuring Amazon S3 storage for automation hub

Amazon S3 storage is a type of object storage that is supported in containerized installations. When using an AWS S3 storage backend, set **hub_storage_backend** to **s3**. The AWS S3 bucket needs to exist before running the installation program.

Procedure

1. Ensure your AWS S3 bucket exists before proceeding with the installation.
2. Add the following variables to your inventory file under the **[all:vars]** group to configure S3 storage:
   - **hub_s3_access_key**
   - **hub_s3_secret_key**
   - **hub_s3_bucket_name**
   - **hub_s3_extra_settings**
     You can pass extra parameters through an Ansible **hub_s3_extra_settings** dictionary. For example:

     ```
     hub_s3_extra_settings:
       AWS_S3_MAX_MEMORY_SIZE: 4096
       AWS_S3_REGION_NAME: eu-central-1
       AWS_S3_USE_SSL: True
     ```

Additional resources

- django-storages documentation - Amazon S3

2.7.3.2. Configuring Azure Blob Storage for automation hub

Azure Blob storage is a type of object storage that is supported in containerized installations. When using an Azure blob storage backend, set **hub_storage_backend** to **azure**. The Azure container needs to exist before running the installation program.

Procedure

1. Ensure your Azure container exists before proceeding with the installation.
2. Add the following variables to your inventory file under the **[all:vars]** group to configure Azure Blob storage:
   - **hub_azure_account_key**
   - **hub_azure_account_name**
   - **hub_azure_container**
   - **hub_azure_extra_settings**
     You can pass extra parameters through an Ansible **hub_azure_extra_settings** dictionary. For example:

     ```
     hub_azure_extra_settings:
       AZURE_LOCATION: foo
       AZURE_SSL: True
       AZURE_URL_EXPIRATION_SECS: 60
     ```

Additional resources

- django-storages documentation - Azure Storage

2.7.3.3. Configuring Network File System (NFS) storage for automation hub

NFS is a type of shared storage that is supported in containerized installations. Shared storage is required when installing more than one instance of automation hub with a **file** storage backend. When installing a single instance of the automation hub, shared storage is optional.

Procedure

1. To configure shared storage for automation hub, set the **hub_shared_data_path** variable in your inventory file:

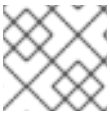
   ```
   hub_shared_data_path=<path_to_nfs_share>
   ```

   The value must match the format **host:dir**, for example **nfs-server.example.com:/exports/hub**.
2. (Optional) To change the mount options for your NFS share, use the **hub_shared_data_mount_opts** variable. The default value is **rw,sync,hard**.

2.7.4. Configuring a HAProxy load balancer

To configure a HAProxy load balancer in front of platform gateway with a custom CA cert, set the following inventory file variables under the **[all:vars]** group:

```
custom_ca_cert=<path_to_cert_crt>
gateway_main_url=<https://load_balancer_url>
```



NOTE

HAProxy SSL passthrough mode is not supported with platform gateway.

2.7.5. Enabling automation content collection and container signing

Automation content signing is disabled by default. To enable it, the following installation variables are required in the inventory file:

```
# Collection signing
hub_collection_signing=true
hub_collection_signing_key=<full_path_to_collection_gpg_key>

# Container signing
hub_container_signing=true
hub_container_signing_key=<full_path_to_container_gpg_key>
```

The following variables are required if the keys are protected by a passphrase:

```
# Collection signing
hub_collection_signing_pass=<gpg_key_passphrase>

# Container signing
hub_container_signing_pass=<gpg_key_passphrase>
```

The **hub_collection_signing_key** and **hub_container_signing_key** variables require the set up of keys before running an installation.

Automation content signing currently only supports GnuPG (GPG) based signature keys. For more information about GPG, see the GnuPG man page.

NOTE

The algorithm and cipher used is the responsibility of the customer.

Procedure

1. On a RHEL9 server run the following command to create a new key pair for collection signing:

```
gpg --gen-key
```

2. Enter your information for "Real name" and "Email address":
   Example output:

```
gpg --gen-key
gpg (GnuPG) 2.3.3; Copyright (C) 2021 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Note: Use "gpg --full-generate-key" for a full featured key generation dialog.
```

```
GnuPG needs to construct a user ID to identify your key.

Real name: Joe Bloggs
Email address: jbloggs@example.com
You selected this USER-ID:
    "Joe Bloggs <jbloggs@example.com>"

Change (N)ame, (E)mail, or (O)kay/(Q)uit? O
```

If this fails, your environment does not have the necessary prerequisite packages installed for GPG. Install the necessary packages to proceed.

3. A dialog box will appear and ask you for a passphrase. This is optional but recommended.
4. The keys are then generated, and produce output similar to the following:

```
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
gpg: key 022E4FBFB650F1C4 marked as ultimately trusted
gpg: revocation certificate stored as '/home/aapuser/.gnupg/openpgp-
revocs.d/F001B037976969DD3E17A829022E4FBFB650F1C4.rev'
public and secret key created and signed.

pub   rsa3072 2024-10-25 [SC] [expires: 2026-10-25]
      F001B037976969DD3E17A829022E4FBFB650F1C4
uid                 Joe Bloggs <jbloggs@example.com>
sub   rsa3072 2024-10-25 [E] [expires: 2026-10-25]
```

Note the expiry date that you can set based on company standards and needs.

5. You can view all of your GPG keys by running the following command:

```
gpg --list-secret-keys --keyid-format=long
```

6. To export the public key run the following command:

```
gpg --export -a --output collection-signing-key.pub <email_address_used_to_generate_key>
```

7. To export the private key run the following command:

```
gpg -a --export-secret-keys <email_address_used_to_generate_key> > collection-signing-key.priv
```

8. If a passphrase is detected, you will be prompted to enter the passphrase.
9. To view the private key file contents, run the following command:

```
cat collection-signing-key.priv
```

Example output:

```
-----BEGIN PGP PRIVATE KEY BLOCK-----

lQWFBGcbN14BDADTg5BsZGbSGMHypUJMuzmIffzzz4LULrZA8L/I616lzpBHJvEs
sSN6KuKY1TcIwIDCCa/U5Obm46kurpP2Y+vNA1YSEtMJoSeHeamWMDd99f49ltBp

<snippet>

j920hRy/3wJGRDBMFa4mlQg=
=uYEF
-----END PGP PRIVATE KEY BLOCK-----
```

10. Repeat steps 1 to 9 to create a key pair for container signing.

11. Add the following variables to the inventory file and run the installation to create the signing services:

```
# Collection signing
hub_collection_signing=true
hub_collection_signing_key=/home/aapuser/aap/ansible-automation-platform-containerized-setup-<version_number>/collection-signing-key.priv
# This variable is required if the key is protected by a passphrase
hub_collection_signing_pass=<password>

# Container signing
hub_container_signing=true
hub_container_signing_key=/home/aapuser/aap/ansible-automation-platform-containerized-setup-<version_number>/container-signing-key.priv
# This variable is required if the key is protected by a passphrase
hub_container_signing_pass=<password>
```

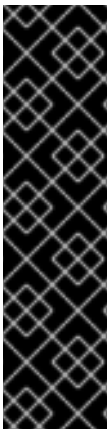
Additional resources

- Working with signed containers

2.7.6. Setting up a customer provided (external) database

There are two possible scenarios for setting up an external database:

1. An external database with PostgreSQL admin credentials
2. An external database without PostgreSQL admin credentials



IMPORTANT

- When using an external database with Ansible Automation Platform, you must create and maintain that database. Ensure that you clear your external database when uninstalling Ansible Automation Platform.
- Red Hat Ansible Automation Platform requires customer provided (external) database to have ICU support.
- During configuration of an external database, you must check the external database coverage. For more information, see Red Hat Ansible Automation Platform Database Scope of Coverage.

2.7.6.1. Setting up an external database with PostgreSQL admin credentials

If you have PostgreSQL admin credentials, you can supply them in the inventory file and the installation program creates the PostgreSQL users and databases for each component for you. The PostgreSQL admin account must have **SUPERUSER** privileges.

Procedure

- To configure the PostgreSQL admin credentials, add the following variables to the inventory file under the **[all:vars]** group:

```
postgresql_admin_username=<set your own>
postgresql_admin_password=<set your own>
```

2.7.6.2. Setting up an external database without PostgreSQL admin credentials

If you do not have PostgreSQL admin credentials, then PostgreSQL users and databases need to be created for each component (platform gateway, automation controller, automation hub, and Event-Driven Ansible) before running the installation program.

Procedure

1. Connect to a PostgreSQL compliant database server with a user that has **SUPERUSER** privileges.

```
# psql -h <hostname> -U <username> -p <port_number>
```

   For example:

```
# psql -h db.example.com -U superuser -p 5432
```

2. Create the user with a password and ensure the **CREATEDB** role is assigned to the user. For more information, see Database Roles.

```
CREATE USER <username> WITH PASSWORD <password> CREATEDB;
```

3. Create the database and add the user you created as the owner.

```
CREATE DATABASE <database_name> OWNER <username>;
```

4. When you have created the PostgreSQL users and databases for each component, you can supply them in the inventory file under the **[all:vars]** group.

```
# Platform gateway
gateway_pg_host=aap.example.org
gateway_pg_database=<set your own>
gateway_pg_username=<set your own>
gateway_pg_password=<set your own>

# Automation controller
controller_pg_host=aap.example.org
controller_pg_database=<set your own>
controller_pg_username=<set your own>
```

```
controller_pg_password=<set your own>

# Automation hub
hub_pg_host=aap.example.org
hub_pg_database=<set your own>
hub_pg_username=<set your own>
hub_pg_password=<set your own>

# Event-Driven Ansible
eda_pg_host=aap.example.org
eda_pg_database=<set your own>
eda_pg_username=<set your own>
eda_pg_password=<set your own>
```

2.7.6.3. Enabling the hstore extension for the automation hub PostgreSQL database

The database migration script uses **hstore** fields to store information, therefore the **hstore** extension must be enabled in the automation hub PostgreSQL database.

This process is automatic when using the Ansible Automation Platform installer and a managed PostgreSQL server.

If the PostgreSQL database is external, you must enable the **hstore** extension in the automation hub PostgreSQL database manually before installation.

If the **hstore** extension is not enabled before installation, a failure raises during database migration.

Procedure

1. Check if the extension is available on the PostgreSQL server (automation hub database).

```
$ psql -d <automation hub database> -c "SELECT * FROM pg_available_extensions WHERE
name='hstore'"
```

2. Where the default value for **<automation hub database>** is **automationhub**.
   Example output with hstore available:

```
name  | default_version | installed_version |comment
------+-----------------+-------------------+----------------------------------------------------
 hstore | 1.7           |                   | data type for storing sets of (key, value) pairs
(1 row)
```

   Example output with hstore not available:

```
 name | default_version | installed_version | comment
------+-----------------+-------------------+---------
(0 rows)
```

3. On a RHEL based server, the **hstore** extension is included in the **postgresql-contrib** RPM package, which is not installed automatically when installing the PostgreSQL server RPM package.
   To install the RPM package, use the following command:

```
dnf install postgresql-contrib
```

4. Load the **hstore** PostgreSQL extension into the automation hub database with the following command:

```
$ psql -d <automation hub database> -c "CREATE EXTENSION hstore;"
```

In the following output, the **installed_version** field lists the **hstore** extension used, indicating that **hstore** is enabled.

```
name | default_version | installed_version | comment
-----+-----------------+-------------------+-----------------------------------------------------
hstore |     1.7      |       1.7         | data type for storing sets of (key, value) pairs
(1 row)
```

2.7.6.4. Optional: configuring mutual TLS (mTLS) authentication for an external database

mTLS authentication is disabled by default. To configure each component's database with mTLS authentication, add the following variables to your inventory file under the **[all:vars]** group and ensure each component has a different TLS certificate and key:

Procedure

- Add the following variables to your inventory file under the **[all:vars]** group:

```
# Platform gateway
gateway_pg_cert_auth=true
gateway_pg_tls_cert=/path/to/gateway.cert
gateway_pg_tls_key=/path/to/gateway.key
gateway_pg_sslmode=verify-full

# Automation controller
controller_pg_cert_auth=true
controller_pg_tls_cert=/path/to/awx.cert
controller_pg_tls_key=/path/to/awx.key
controller_pg_sslmode=verify-full

# Automation hub
hub_pg_cert_auth=true
hub_pg_tls_cert=/path/to/pulp.cert
hub_pg_tls_key=/path/to/pulp.key
hub_pg_sslmode=verify-full

# Event-Driven Ansible
eda_pg_cert_auth=true
eda_pg_tls_cert=/path/to/eda.cert
eda_pg_tls_key=/path/to/eda.key
eda_pg_sslmode=verify-full
```

2.7.7. Using custom TLS certificates

Red Hat Ansible Automation Platform uses X.509 certificate and key pairs to secure traffic both internally between Ansible Automation Platform components and externally for public UI and API connections.

There are two primary ways to manage TLS certificates for your Ansible Automation Platform deployment:

1. Ansible Automation Platform generated certificates (this is the default)
2. User-provided certificates

2.7.7.1. Ansible Automation Platform generated certificates

By default, the installation program creates a self-signed Certificate Authority (CA) and uses it to generate self-signed TLS certificates for all Ansible Automation Platform services. The self-signed CA certificate and key are generated on one node under the **~/aap/tls/** directory and copied to the same location on all other nodes. This CA is valid for 10 years after the initial creation date.

Self-signed certificates are not part of any public chain of trust. The installation program creates a certificate truststore that includes the self-signed CA certificate under **~/aap/tls/extracted/** and bind-mounts that directory to each Ansible Automation Platform service container under **/etc/pki/ca-trust/extracted/**. This allows each Ansible Automation Platform component to validate the self-signed certificates of the other Ansible Automation Platform services. The CA certificate can also be added to the truststore of other systems or browsers as needed.

2.7.7.2. User-provided certificates

To use your own TLS certificates and keys to replace some or all of the self-signed certificates generated during installation, you can set specific variables in your inventory file. These certificates and keys must be generated by a public or organizational CA in advance so that they are available during the installation process.

2.7.7.2.1. Using a custom CA to generate all TLS certificates

Use this method when you want Ansible Automation Platform to generate all of the certificates, but you want them signed by a custom CA rather than the default self-signed certificates.

Procedure

- To use a custom Certificate Authority (CA) to generate TLS certificates for all Ansible Automation Platform services, set the following variables in your inventory file:

```
ca_tls_cert=<path_to_ca_tls_certificate>
ca_tls_key=<path_to_ca_tls_key>
```

2.7.7.2.2. Providing custom TLS certificates for each service

Use this method if your organization manages TLS certificates outside of Ansible Automation Platform and requires manual provisioning.

Procedure

- To manually provide TLS certificates for each individual service (for example, automation controller, automation hub, and Event-Driven Ansible), set the following variables in your inventory file:

```
# Platform gateway
gateway_tls_cert=<path_to_tls_certificate>
gateway_tls_key=<path_to_tls_key>
gateway_pg_tls_cert=<path_to_tls_certificate>
gateway_pg_tls_key=<path_to_tls_key>
gateway_redis_tls_cert=<path_to_tls_certificate>
gateway_redis_tls_key=<path_to_tls_key>

# Automation controller
controller_tls_cert=<path_to_tls_certificate>
controller_tls_key=<path_to_tls_key>
controller_pg_tls_cert=<path_to_tls_certificate>
controller_pg_tls_key=<path_to_tls_key>

# Automation hub
hub_tls_cert=<path_to_tls_certificate>
hub_tls_key=<path_to_tls_key>
hub_pg_tls_cert=<path_to_tls_certificate>
hub_pg_tls_key=<path_to_tls_key>

# Event-Driven Ansible
eda_tls_cert=<path_to_tls_certificate>
eda_tls_key=<path_to_tls_key>
eda_pg_tls_cert=<path_to_tls_certificate>
eda_pg_tls_key=<path_to_tls_key>
eda_redis_tls_cert=<path_to_tls_certificate>
eda_redis_tls_key=<path_to_tls_key>

# PostgreSQL
postgresql_tls_cert=<path_to_tls_certificate>
postgresql_tls_key=<path_to_tls_key>

# Receptor
receptor_tls_cert=<path_to_tls_certificate>
receptor_tls_key=<path_to_tls_key>

# Redis
redis_tls_cert=<path_to_tls_certificate>
redis_tls_key=<path_to_tls_key>
```

2.7.7.2.3. Considerations for certificates provided per service

When providing custom TLS certificates for each individual service, consider the following:

- It is possible to provide unique certificates per host. This requires defining the specific **_tls_cert** and **_tls_key** variables in your inventory file as shown in the earlier inventory file example.

- For services deployed across many nodes (for example, when following the enterprise topology), the provided certificate for that service must include the FQDN of all associated nodes in its Subject Alternative Name (SAN) field.

- If an external-facing service (such as automation controller or platform gateway) is deployed

behind a load balancer that performs SSL/TLS offloading, the service's certificate must include the load balancer's FQDN in its SAN field, in addition to the FQDNs of the individual service nodes.

Additional resources

- Securing networks

2.7.7.2.4. Providing a custom CA certificate

When you manually provide TLS certificates, those certificates might be signed by a custom CA. Provide a custom CA certificate to ensure proper authentication and secure communication within your environment. If you have multiple custom CA certificates, you must merge them into a single file.

Procedure

- If any of the TLS certificates you manually provided are signed by a custom CA, you must specify the CA certificate by using the following variable in your inventory file:

```
custom_ca_cert=<path_to_custom_ca_certificate>
```

  If you have more than one CA certificate, combine them into a single file and reference the combined certificate with the **custom_ca_cert** variable.

2.7.7.3. Receptor certificate considerations

When using a custom certificate for Receptor nodes, the certificate requires the **otherName** field specified in the Subject Alternative Name (SAN) of the certificate with the value **1.3.6.1.4.1.2312.19.1**. For more information, see Above the mesh TLS.

Receptor does not support the usage of wildcard certificates. Additionally, each Receptor certificate must have the host FQDN specified in its SAN for TLS hostname validation to be correctly performed.

2.7.7.4. Redis certificate considerations

When using custom TLS certificates for Redis-related services, consider the following for mutual TLS (mTLS) communication if specifying Extended Key Usage (EKU):

- The Redis server certificate (**redis_tls_cert**) should include the **serverAuth** (web server authentication) and **clientAuth** (client authentication) EKU.
- The Redis client certificates (**gateway_redis_tls_cert**, **eda_redis_tls_cert**) should include the **clientAuth** (client authentication) EKU.

2.7.8. Using custom Receptor signing keys

Receptor signing is enabled by default unless **receptor_disable_signing=true** is set, and an RSA key pair (public and private) is generated by the installation program. However, you can set custom RSA public and private keys by using the following variables:

```
receptor_signing_private_key=<full_path_to_private_key>
receptor_signing_public_key=<full_path_to_public_key>
```

2.8. INSTALLING CONTAINERIZED ANSIBLE AUTOMATION PLATFORM

After you prepare the Red Hat Enterprise Linux host, download Ansible Automation Platform, and configure the inventory file, run the **install** playbook to install containerized Ansible Automation Platform.

Prerequisites

You have done the following:

- Prepared the Red Hat Enterprise Linux host
- Prepared the managed nodes
- Downloaded Ansible Automation Platform
- Configured the inventory file
- Logged in to the Red Hat Enterprise Linux host as your non-root user

Procedure

1. Go to the installation directory on your Red Hat Enterprise Linux host.
2. Run the **install** playbook:

```
ansible-playbook -i <inventory_file_name> ansible.containerized_installer.install
```

For example:

```
ansible-playbook -i inventory ansible.containerized_installer.install
```

You can add additional parameters to the installation command as needed:

```
ansible-playbook -i <inventory_file_name> -e @<vault_file_name> --ask-vault-pass -K -v
ansible.containerized_installer.install
```

For example:

```
ansible-playbook -i inventory -e @vault.yml --ask-vault-pass -K -v
ansible.containerized_installer.install
```

- **-i <inventory_file_name>** - The inventory file to use for the installation.
- **-e @<vault_file_name> --ask-vault-pass** - (Optional) If you are using a vault to store sensitive variables, add this to the installation command.
- **-K** - (Optional) If your privilege escalation requires you to enter a password, add this to the installation command. You are then prompted for the BECOME password.
- **-v** - (Optional) You can use increasing verbosity, up to 4 v's (**-vvvv**) to see the details of the installation process. However, it is important to note that this can significantly increase installation time, so use it only as needed or requested by Red Hat support.

3. The installation of containerized Ansible Automation Platform begins.

Verification

- After the installation completes, verify that you can access the platform UI which is available by default at the following URL:

```
https://<gateway_node>:443
```

- Log in as the admin user with the credentials you created for **gateway_admin_username** and **gateway_admin_password**.
- The default ports and protocols used for Ansible Automation Platform are 80 (HTTP) and 443 (HTTPS). You can customize the ports with the following variables:

```
envoy_http_port=80
envoy_https_port=443
```

- If you want to disable HTTPS, set **envoy_disable_https** to **true**:

```
envoy_disable_https: true
```

Additional resources

- Understanding privilege escalation: become
- Sensitive variables in the installation inventory
- Getting started with Ansible Automation Platform
- Troubleshooting containerized Ansible Automation Platform

2.9. UPDATING CONTAINERIZED ANSIBLE AUTOMATION PLATFORM

Perform a patch update for a container-based installation of Ansible Automation Platform from 2.5 to 2.5.x.

Upgrades from 2.4 Containerized Ansible Automation Platform Tech Preview to 2.5 Containerized Ansible Automation Platform are not supported.

Prerequisites

- You have reviewed the release notes for the associated patch release.
- You have created a backup of your Ansible Automation Platform deployment.

Procedure

1. Log in to the Red Hat Enterprise Linux host as your dedicated non-root user.
2. Follow the steps in Downloading Ansible Automation Platform to download the latest version of containerized Ansible Automation Platform.
3. Copy the downloaded installation program to your Red Hat Enterprise Linux Host.

4. Edit the **inventory** file to match your required configuration. You can keep the same parameters from your existing Ansible Automation Platform deployment or you can change the parameters to match any modifications to your environment.

5. Run the **install** playbook:

```
$ ansible-playbook -i inventory ansible.containerized_installer.install
```

   - If your privilege escalation requires a password to be entered, append **-K** to the command. You will then be prompted for the **BECOME** password.

   - You can use increasing verbosity, up to 4 v's (**-vvvv**) to see the details of the installation process. However it is important to note that this can significantly increase installation time, so it is recommended that you use it only as needed or requested by Red Hat support.

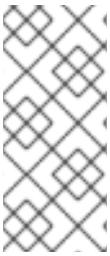
6. The update begins.

Additional resources

- Ansible Automation Platform Release notes
- Backing up container-based Ansible Automation Platform

2.10. BACKING UP CONTAINERIZED ANSIBLE AUTOMATION PLATFORM

Perform a backup of your container-based installation of Ansible Automation Platform.



NOTE

When backing up Ansible Automation Platform, use the installation program that matches your currently installed version of Ansible Automation Platform.

Backup functionality only works with the PostgreSQL versions supported by your current Ansible Automation Platform version. For more information, see System requirements.

Prerequisites

- You have logged in to the Red Hat Enterprise Linux host as your dedicated non-root user.

Procedure

1. Go to the Red Hat Ansible Automation Platform installation directory on your Red Hat Enterprise Linux host.

2. To control compression of the backup artifacts before they are sent to the host running the backup operation, you can use the following variables in your inventory file:

   a. For control of compression for filesystem related backup files:

```
# For global control of compression for filesystem related backup files
use_archive_compression=true

# For component-level control of compression for filesystem related backup files
```

```
#controller_use_archive_compression=true
#eda_use_archive_compression=true
#gateway_use_archive_compression=true
#hub_use_archive_compression=true
#pcp_use_archive_compression=true
#postgresql_use_archive_compression=true
#receptor_use_archive_compression=true
#redis_use_archive_compression=true
```

b. For control of compression for database related backup files:

```
# For global control of compression for database related backup files
use_db_compression=true

# For component-level control of compression for database related backup files
#controller_use_db_compression=true
#eda_use_db_compression=true
#hub_use_db_compression=true
#gateway_use_db_compression=true
```

3. Run the **backup** playbook:

```
$ ansible-playbook -i <path_to_inventory> ansible.containerized_installer.backup
```

This backs up the important data deployed by the containerized installer such as:

- PostgreSQL databases
- Configuration files
- Data files

4. By default, the backup directory is set to **./backups**. You can change this by using the **backup_dir** variable in your **inventory** file.

Next steps

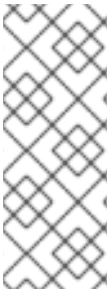
To customize the backup, use the following variables in your **inventory** file.

- Change the backup destination directory from the default **./backups** by using the **backup_dir** variable.
- Exclude paths that contain duplicated data, such as snapshot subdirectories, by using the **hub_data_path_exclude** variable. For instance, to exclude a .snapshots subdirectory, specify hub_data_path_exclude=['**/.snapshots/**'] in your inventory file.
  - Alternatively, you can use the command-line interface with the **-e** flag to pass this variable at runtime:

```
$ ansible-playbook -i inventory ansible.containerized_installer.backup -e
hub_data_path_exclude="['*/.snapshots/*']"
```

2.11. RESTORING CONTAINERIZED ANSIBLE AUTOMATION PLATFORM

Restore your container-based installation of Ansible Automation Platform from a backup, or to a different environment.



NOTE

When restoring Ansible Automation Platform, use the latest installation program available at the time of the restore. For example, if you are restoring a backup taken from version **2.5-1**, use the latest **2.5-x** installation program available at the time of the restore.

Restore functionality only works with the PostgreSQL versions supported by your current Ansible Automation Platform version. For more information, see System requirements.

Prerequisites

- You have logged in to the Red Hat Enterprise Linux host as your dedicated non-root user.
- You have a backup of your Ansible Automation Platform deployment. For more information, see Backing up containerized Ansible Automation Platform.
- If restoring to a different environment with the same hostnames, you have performed a fresh installation on the target environment with the same topology as the original (source) environment.
- You have ensured that the administrator credentials on the target environment match the administrator credentials from the source environment.

Procedure

1. Go to the installation directory on your Red Hat Enterprise Linux host.
2. Perform the relevant restoration steps:
   - If you are restoring to the same environment with the same hostnames, run the **restore** playbook:

     ```
     $ ansible-playbook -i <path_to_inventory> ansible.containerized_installer.restore
     ```

     This restores the important data deployed by the containerized installer such as:

     - PostgreSQL databases
     - Configuration files
     - Data files
       By default, the backup directory is set to **./backups**. You can change this by using the **backup_dir** variable in your **inventory** file.
   - If you are restoring to a different environment with different hostnames, perform the following additional steps before running the **restore** playbook:

     IMPORTANT

     Restoring to a different environment with different hostnames is not recommended and is intended only as a workaround.

i. For each component, identify the backup file from the source environment that contains the PostgreSQL dump file.
For example:

```
$ cd ansible-automation-platform-containerized-setup-<version_number>/backups
```

```
$ tar tvf gateway_env1-gateway-node1.tar.gz | grep db

-rw-r--r-- ansible/ansible 4850774 2025-06-30 11:05 aap/backups/awx.db
```

ii. Copy the backup files from the source environment to the target environment.

iii. Rename the backup files on the target environment to reflect the new node names.
For example:

```
$ cd ansible-automation-platform-containerized-setup-<version_number>/backups
```

```
$ mv gateway_env1-gateway-node1.tar.gz gateway_env2-gateway-node1.tar.gz
```

iv. For enterprise topologies, ensure that the component backup file containing the **component.db** file is listed first in its group within the inventory file.
For example:

```
$ cd ansible-automation-platform-containerized-setup-<version_number>
```

```
$ ls backups/gateway*

gateway_env2-gateway-node1.tar.gz
gateway_env2-gateway-node2.tar.gz
```

```
$ tar tvf backups/gateway_env2-gateway-node1.tar.gz | grep db

-rw-r--r-- ansible/ansible 416687 2025-06-30 11:05 aap/backups/gateway.db
```

```
$ tar tvf backups/gateway_env2-gateway-node2.tar.gz | grep db
```

```
$ vi inventory

[automationgateway]
env2-gateway-node1
env2-gateway-node2
```

2.12. UNINSTALLING CONTAINERIZED ANSIBLE AUTOMATION PLATFORM

Uninstall your container-based installation of Ansible Automation Platform.

Prerequisites

- You have logged in to the Red Hat Enterprise Linux host as your dedicated non-root user.

Procedure

1. If you intend to reinstall Ansible Automation Platform and want to use the preserved databases, you must collect the existing secret keys:

   a. First, list the available secrets:

   ```
   $ podman secret list
   ```

   b. Next, collect the secret keys by running the following command:

   ```
   $ podman secret inspect --showsecret <secret_key_variable> | jq -r .[].SecretData
   ```

   For example:

   ```
   $ podman secret inspect --showsecret controller_secret_key | jq -r .[].SecretData
   ```

2. Run the **uninstall** playbook:

   ```
   $ ansible-playbook -i inventory ansible.containerized_installer.uninstall
   ```

   - This stops all systemd units and containers and then deletes all resources used by the containerized installer such as:
     - configuration and data directories and files
     - systemd unit files
     - Podman containers and images
     - RPM packages
   - To keep container images, set the **container_keep_images** parameter to **true**.

     ```
     $ ansible-playbook -i inventory ansible.containerized_installer.uninstall -e
     container_keep_images=true
     ```

   - To keep PostgreSQL databases, set the **postgresql_keep_databases** parameter to **true**.

     ```
     $ ansible-playbook -i inventory ansible.containerized_installer.uninstall -e
     postgresql_keep_databases=true
     ```

Additional resources

- Inventory file variables

2.13. REINSTALLING CONTAINERIZED ANSIBLE AUTOMATION PLATFORM

To reinstall a containerized deployment after uninstalling and preserving the database, follow the steps in Installing containerized Ansible Automation Platform and include the existing secret key value in the playbook command:

```
$ ansible-playbook -i inventory ansible.containerized_installer.install -e controller_secret_key=
<secret_key_value>
```

Additional resources

- Inventory file variables

CHAPTER 3. DISCONNECTED INSTALLATION

You can install containerized Ansible Automation Platform in an environment that does not have an active internet connection. To do this you need to obtain and configure the RPM source dependencies before performing the disconnected installation.

3.1. OBTAINING AND CONFIGURING RPM SOURCE DEPENDENCIES

The Ansible Automation Platform containerized setup bundle installation program does not include RPM source dependencies from the BaseOS and AppStream repositories. It relies on the host system's package manager to resolve these dependencies.

To access these dependencies in a disconnected environment, you can use one of the following methods:

- Use Red Hat Satellite to synchronize repositories in your disconnected environment.
- Use a local repository that you create with the **reposync** command on a Red Hat Enterprise Linux host that has an active internet connection.
- Use a local repository that you create from a mounted Red Hat Enterprise Linux Binary DVD ISO image.

3.1.1. Configuring a local repository using reposync

With the **reposync** command you can to synchronize the BaseOS and AppStream repositories to a local directory on a Red Hat Enterprise Linux host with an active internet connection. You can then transfer the repositories to your disconnected environment.

Prerequisites

- A Red Hat Enterprise Linux host with an active internet connection.

Procedure

1. Attach the BaseOS and AppStream repositories using **subscription-manager**:

```
$ sudo subscription-manager repos \
    --enable rhel-9-baseos-rhui-rpms \
    --enable rhel-9-appstream-rhui-rpms
```

2. Install the **yum-utils** package:

```
$ sudo dnf install yum-utils
```

3. Synchronize the repositories with the **reposync** command. Replace **<path_to_download>** with a suitable value.

```
$ sudo reposync -m --download-metadata --gpgcheck \
    -p <path_to_download>
```

   For example:

```
$ sudo reposync -m --download-metadata --gpgcheck \
    -p rhel-repos
```

- Use reposync with the **--download-metadata** option and without the **--newest-only** option for optimal download time.

4. After the **reposync** operation is complete, compress the directory:

```
$ tar czvf rhel-repos.tar.gz rhel-repos
```

5. Move the compressed archive to your disconnected environment.

6. On the disconnected environment, create a directory to store the repository files:

```
$ sudo mkdir /opt/rhel-repos
```

7. Extract the archive into the **/opt/rhel-repos** directory. The following command assumes the archive file is in your home directory:

```
$ sudo tar xzvf ~/rhel-repos.tar.gz -C /opt
```

8. Create a Yum repository file at **/etc/yum.repos.d/rhel.repo** with the following content:

```
[RHEL-BaseOS]
name=Red Hat Enterprise Linux BaseOS
baseurl=file:///opt/rhel-repos/rhel-9-baseos-rhui-rpms
enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release

[RHEL-AppStream]
name=Red Hat Enterprise Linux AppStream
baseurl=file:///opt/rhel-repos/rhel-9-appstream-rhui-rpms
enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release
```

9. Import the gpg key to allow the system to verify the packages:

```
$ sudo rpm --import /opt/rhel-repos/rhel-9-baseos-rhui-rpms/RPM-GPG-KEY-redhat-release
```

10. Verify the repository configuration:

```
$ sudo yum repolist
```

3.1.2. Configuring a local repository from a mounted ISO

You can use a Red Hat Enterprise Linux Binary DVD image to access the necessary RPM source dependencies in a disconnected environment.

Prerequisites

- You have downloaded the Red Hat Enterprise Linux Binary DVD image from the Red Hat Enterprise Linux downloads page and moved it to your disconnected environment.

Procedure

1. In your disconnected environment, create a mount point directory to serve as the location for the ISO file:

```
$ sudo mkdir /media/rhel
```

2. Mount the ISO image to the mount point. Replace **<version_number>** and **<arch_name>** with suitable values:

```
$ sudo mount -o loop rhel-<version_number>-<arch_name>-dvd.iso /media/rhel
```

   - Note: The ISO is mounted in a read-only state.

3. Create a Yum repository file at **/etc/yum.repos.d/rhel.repo** with the following content:

```
[RHEL-BaseOS]
name=Red Hat Enterprise Linux BaseOS
baseurl=file:///media/rhel/BaseOS
enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release

[RHEL-AppStream]
name=Red Hat Enterprise Linux AppStream
baseurl=file:///media/rhel/AppStream
enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release
```

4. Import the gpg key to allow the system to verify the packages:

```
$ sudo rpm --import /media/rhel/RPM-GPG-KEY-redhat-release
```

5. Verify the repository configuration:

```
$ sudo yum repolist
```

3.2. PERFORMING A DISCONNECTED INSTALLATION

Use the following steps to perform a disconnected installation of containerized Ansible Automation Platform.

Prerequisites

You have done the following:

- Prepared the Red Hat Enterprise Linux host

- Obtained and configured the RPM source dependencies. The installation program uses your host system's **dnf** package manager to resolve these dependencies.
- Prepared the managed nodes
- Downloaded the containerized Ansible Automation Platform setup bundle from the Ansible Automation Platform download page.

Procedure

1. Log in to the Red Hat Enterprise Linux host as your non-root user.
2. Update the inventory file by following the steps in Configuring the inventory file.
3. Ensure the following variables are included in your inventory file under the **[all:vars]** group:

```
bundle_install=true
# The bundle directory must include /bundle in the path
bundle_dir='{{ lookup("ansible.builtin.env", "PWD") }}/bundle'
```

4. Follow the steps in Installing containerized Ansible Automation Platform to install containerized Ansible Automation Platform and verify your installation.

CHAPTER 4. HORIZONTAL SCALING IN RED HAT ANSIBLE AUTOMATION PLATFORM

You can set up multi-node deployments for components across Ansible Automation Platform. Whether you require horizontal scaling for Automation Execution, Automation Decisions, or automation mesh, you can scale your deployments based on your organization's needs.

4.1. HORIZONTAL SCALING IN EVENT-DRIVEN ANSIBLE CONTROLLER

With Event-Driven Ansible controller, you can set up horizontal scaling for your events automation. This multi-node deployment enables you to define as many nodes as you prefer during the installation process. You can also increase or decrease the number of nodes at any time according to your organizational needs.

The following node types are used in this deployment:

API node type

Responds to the HTTP REST API of Event-Driven Ansible controller.

Worker node type

Runs an Event-Driven Ansible worker, which is the component of Event-Driven Ansible that not only manages projects and activations, but also executes the activations themselves.

Hybrid node type

Is a combination of the API node and the worker node.

The following example shows how you can set up an inventory file for horizontal scaling of Event-Driven Ansible controller on Red Hat Enterprise Linux VMs using the host group name **[automationeda]** and the node type variable **eda_type**:

```
[automationeda]

3.88.116.111 routable_hostname=automationeda-api.example.com eda_type=api

# worker node
3.88.116.112 routable_hostname=automationeda-api.example.com eda_type=worker
```

4.1.1. Sizing and scaling guidelines

API nodes process user requests (interactions with the UI or API) while worker nodes process the activations and other background tasks required for Event-Driven Ansible to function properly. The number of API nodes you require correlates to the required number of users of the application and the number of worker nodes correlates to the required number of activations you want to run.

Since activations are variable and controlled by worker nodes, the supported approach for scaling is to use separate API and worker nodes instead of hybrid nodes due to the efficient allocation of hardware resources by worker nodes. By separating the nodes, you can scale each type independently based on specific needs, leading to better resource utilization and cost efficiency.

An example of an instance in which you might consider scaling up your node deployment is when you want to deploy Event-Driven Ansible for a small group of users who will run a large number of activations. In this case, one API node is adequate, but if you require more, you can scale up to three additional worker nodes.

4.1.2. Setting up horizontal scaling for Event-Driven Ansible controller

To scale up (add more nodes) or scale down (remove nodes), you must update the content of the inventory file to add or remove nodes and rerun the installation program.

Procedure

1. Update the inventory to add two more worker nodes:

```
[automationeda]

3.88.116.111 routable_hostname=automationeda-api.example.com eda_type=api

3.88.116.112 routable_hostname=automationeda-api.example.com eda_type=worker

# two more worker nodes
3.88.116.113 routable_hostname=automationeda-api.example.com eda_type=worker

3.88.116.114 routable_hostname=automationeda-api.example.com eda_type=worker
```

2. Re-run the installer.

APPENDIX A. TROUBLESHOOTING CONTAINERIZED ANSIBLE AUTOMATION PLATFORM

Use this information to troubleshoot your containerized Ansible Automation Platform installation.

A.1. GATHERING ANSIBLE AUTOMATION PLATFORM LOGS

With the **sos** utility, you can collect configuration, diagnostic, and troubleshooting data, and give those files to Red Hat Technical Support. An **sos** report is a common starting point for Red Hat technical support engineers when performing analysis of a service request for Ansible Automation Platform.

You can collect an **sos** report for each host in your containerized Ansible Automation Platform deployment by running the **log_gathering** playbook with the appropriate parameters.

Procedure

1. Go to the Ansible Automation Platform installation directory.
2. Run the **log_gathering** playbook. This playbook connects to each host in the inventory file, installs the **sos** tool, and generates the **sos** report.

```
$ ansible-playbook -i <path_to_inventory_file> ansible.containerized_installer.log_gathering
```

3. Optional: To define additional parameters, specify them with the **-e** option. For example:

```
$ ansible-playbook -i <path_to_inventory_file> ansible.containerized_installer.log_gathering -e 'target_sos_directory=<path_to_files>' -e 'case_number=0000000' -e 'clean=true' -e 'upload=true' -s
```

   a. You can use the **-s** option to step through each task in the playbook and confirm its execution. This is optional but can be helpful for debugging.

   b. The following is a list of the parameters you can use with the **log_gathering** playbook:

Table A.1. Parameter reference

Parameter name	Description	Default
target_sos_directory	Used to change the default location for the **sos** report files.	**/tmp** directory of the current server.
case_number	Specifies the support case number if relevant to the log gathering.	
clean	Obfuscates sensitive data that might be present on the **sos** report.	**false**

Parameter name	Description	Default
upload	Automatically uploads the **sos** report data to Red Hat.	**false**

4. Gather the **sos** report files described in the playbook output and share them with the support engineer or directly upload the **sos** report to Red Hat using the **upload=true** additional parameter.

Additional resources

- What is an sos report and how to create one in Red Hat Enterprise Linux?

A.2. DIAGNOSING THE PROBLEM

For general container-based troubleshooting, you can inspect the container logs for any running service to help troubleshoot underlying issues.

Identifying the running containers

To get a list of the running container names run the following command:

```
$ podman ps --all --format "{{.Names}}"
```

Table A.2. Container details

Component group	Container name	Purpose
Automation controller	**automation-controller-rsyslog**	Handles centralized logging for automation controller.
Automation controller	**automation-controller-task**	Manages and runs tasks related to automation controller, such as running playbooks and interacting with inventories.
Automation controller	**automation-controller-web**	A web server that provides a REST API for automation controller. This is accessed and routed through platform gateway for user interaction.
Event-Driven Ansible	**automation-eda-api**	Exposes the API for Event-Driven Ansible, allowing external systems to trigger and manage event-driven automations.
Event-Driven Ansible	**automation-eda-daphne**	A web server for Event-Driven Ansible, handling WebSocket connections and serving static files.
Event-Driven Ansible	**automation-eda-web**	A web server that provides a REST API for Event-Driven Ansible. This is accessed and routed through platform gateway for user interaction.

Component group	Container name	Purpose
Event-Driven Ansible	**automation-eda-worker-<number>**	These containers run the automation rules and playbooks based on incoming events.
Event-Driven Ansible	**automation-eda-activation-worker-<number>**	These containers manage the activation of automation rules, ensuring they run when specific conditions are met.
Event-Driven Ansible	**automation-eda-scheduler**	Responsible for scheduling and managing recurring tasks and rule activations.
Platform gateway	**automation-gateway-proxy**	Acts as a reverse proxy, routing incoming requests to the appropriate Ansible Automation Platform services.
Platform gateway	**automation-gateway**	Responsible for authentication, authorization, and overall request handling for the platform, all of which is exposed through a REST API and served by a web server.
Automation hub	**automation-hub-api**	Provides the API for automation hub, enabling interaction with collection content, user management, and other automation hub functionality.
Automation hub	**automation-hub-content**	Manages and serves Ansible Content Collections, roles, and modules stored in automation hub.
Automation hub	**automation-hub-web**	A web server that provides a REST API for automation hub. This is accessed and routed through platform gateway for user interaction.
Automation hub	**automation-hub-worker-<number>**	These containers handle background tasks for automation hub, such as content synchronization, indexing, and validation.
Performance Co-Pilot	**pcp**	If Performance Co-Pilot Monitoring is enabled, this container is used for system performance monitoring and data collection.
PostgreSQL	**postgresql**	Hosts the PostgreSQL database for Ansible Automation Platform.
Receptor	**receptor**	Facilitates secure and reliable communication within Ansible Automation Platform.
Redis	**redis-<suffix>**	Responsible for caching, real-time analytics and fast data retrieval.

Inspecting the logs

Containerized Ansible Automation Platform uses **journald** for Podman logging. To inspect any running container logs, run the **journalctl** command:

```
$ journalctl CONTAINER_NAME=<container_name>
```

Example command with output:

```
$ journalctl CONTAINER_NAME=automation-gateway-proxy

Oct 08 01:40:12 aap.example.org automation-gateway-proxy[1919]: [2024-10-08 00:40:12.885][2]
[info][upstream] [external/envoy/source/common/upstream/cds_ap>
Oct 08 01:40:12 aap.example.org automation-gateway-proxy[1919]: [2024-10-08 00:40:12.885][2]
[info][upstream] [external/envoy/source/common/upstream/cds_ap>
Oct 08 01:40:19 aap.example.org automation-gateway-proxy[1919]: [2024-10-08T00:40:16.753Z]
"GET /up HTTP/1.1" 200 - 0 1138 10 0 "192.0.2.1" "python->
```

To view the logs of a running container in real-time, run the **podman logs -f** command:

```
$ podman logs -f <container_name>
```

Controlling container operations

You can control operations for a container by running the **systemctl** command:

```
$ systemctl --user status <container_name>
```

Example command with output:

```
$ systemctl --user status automation-gateway-proxy
● automation-gateway-proxy.service - Podman automation-gateway-proxy.service
    Loaded: loaded (/home/user/.config/systemd/user/automation-gateway-proxy.service; enabled;
preset: disabled)
    Active: active (running) since Mon 2024-10-07 12:39:23 BST; 23h ago
      Docs: man:podman-generate-systemd(1)
   Process: 780 ExecStart=/usr/bin/podman start automation-gateway-proxy (code=exited,
status=0/SUCCESS)
  Main PID: 1919 (conmon)
     Tasks: 1 (limit: 48430)
    Memory: 852.0K
       CPU: 2.996s
    CGroup: /user.slice/user-1000.slice/user@1000.service/app.slice/automation-gateway-
proxy.service
            └─1919 /usr/bin/conmon --api-version 1 -c
2dc3c7b2cecd73010bad1e0aaa806015065f92556ed3591c9d2084d7ee209c7a -u
2dc3c7b2cecd73010bad1e0aaa80>
Oct 08 11:44:10 aap.example.org automation-gateway-proxy[1919]: [2024-10-08T10:44:02.926Z]
"GET /api/galaxy/_ui/v1/settings/ HTTP/1.1" 200 - 0 654 58 47 ">
Oct 08 11:44:10 aap.example.org automation-gateway-proxy[1919]: [2024-10-08T10:44:03.387Z]
"GET /api/controller/v2/config/ HTTP/1.1" 200 - 0 4018 58 44 "1>
Oct 08 11:44:10 aap.example.org automation-gateway-proxy[1919]: [2024-10-08T10:44:03.370Z]
"GET /api/galaxy/v3/plugin/ansible/search/collection-versions/?>
Oct 08 11:44:10 aap.example.org automation-gateway-proxy[1919]: [2024-10-08T10:44:03.405Z]
```

```
"GET /api/controller/v2/organizations/?role_level=notification_>
Oct 08 11:44:10 aap.example.org automation-gateway-proxy[1919]: [2024-10-08T10:44:04.366Z]
"GET /api/galaxy/_ui/v1/me/ HTTP/1.1" 200 - 0 1368 79 40 "192.1>
Oct 08 11:44:10 aap.example.org automation-gateway-proxy[1919]: [2024-10-08T10:44:04.360Z]
"GET /api/controller/v2/workflow_approvals/?page_size=200&statu>
Oct 08 11:44:10 aap.example.org automation-gateway-proxy[1919]: [2024-10-08T10:44:04.379Z]
"GET /api/controller/v2/job_templates/7/ HTTP/1.1" 200 - 0 1356>
Oct 08 11:44:10 aap.example.org automation-gateway-proxy[1919]: [2024-10-08T10:44:04.378Z]
"GET /api/galaxy/_ui/v1/feature-flags/ HTTP/1.1" 200 - 0 207 81>
Oct 08 11:44:13 aap.example.org automation-gateway-proxy[1919]: [2024-10-08 10:44:13.856][2]
[info][upstream] [external/envoy/source/common/upstream/cds_ap>
Oct 08 11:44:13 aap.example.org automation-gateway-proxy[1919]: [2024-10-08 10:44:13.856][2]
[info][upstream] [external/envoy/source/common/upstream/cds_ap
```

Getting container information about the execution plane

To get container information about automation controller, Event-Driven Ansible, and **execution_nodes** nodes, prefix any Podman commands with either:

```
CONTAINER_HOST=unix://run/user/<user_id>/podman/podman.sock
```

or

```
CONTAINERS_STORAGE_CONF=<user_home_directory>/aap/containers/storage.conf
```

Example with output:

```
$ CONTAINER_HOST=unix://run/user/1000/podman/podman.sock podman images

REPOSITORY                                                  TAG         IMAGE ID      CREATED     SIZE
registry.redhat.io/ansible-automation-platform-25/ee-supported-rhel8  latest      59d1bc680a7c  6 days
ago  2.24 GB
registry.redhat.io/ansible-automation-platform-25/ee-minimal-rhel8    latest      a64b9fc48094  6 days
ago  338 MB
```

A.3. TROUBLESHOOTING CONTAINERIZED ANSIBLE AUTOMATION PLATFORM INSTALLATION

Use this information to troubleshoot your containerized installation of Ansible Automation Platform.

The installation takes a long time, or has errors, what should I check?

1. Ensure your system meets the minimum requirements as outlined in System requirements. Factors such as improper storage choices and high latency when distributing across many hosts will all have an impact on installation time.

2. Review the installation log file which is located by default at **./aap_install.log**. You can change the log file location within the **ansible.cfg** file in the installation directory.

3. Enable task profiling callbacks on an ad hoc basis to give an overview of where the installation program spends the most time. To do this, use the local **ansible.cfg** file. Add a callback line under the **[defaults]** section, for example:

```
$ cat ansible.cfg
```

```
[defaults]
callbacks_enabled = ansible.posix.profile_tasks
```

Automation controller returns an error of 413

This error occurs when **manifest.zip** license files that are larger than the **controller_nginx_client_max_body_size** setting. If this error occurs, update the inventory file to include the following variable:

```
controller_nginx_client_max_body_size=5m
```

The default setting of **5m** should prevent this issue, but you can increase the value as needed.

When attempting to install containerized Ansible Automation Platform in Amazon Web Services you receive output that there is no space left on device

```
TASK [ansible.containerized_installer.automationcontroller : Create the receptor container]
**************************************************
fatal: [ec2-13-48-25-168.eu-north-1.compute.amazonaws.com]: FAILED! => {"changed": false, "msg":
"Can't create container receptor", "stderr": "Error: creating container storage: creating an ID-mapped
copy of layer \"98955f43cc908bd50ff43585fec2c7dd9445eaf05eecd1e3144f93ffc00ed4ba\": error
during chown: storage-chown-by-maps: lchown usr/local/lib/python3.9/site-
packages/azure/mgmt/network/v2019_11_01/operations/__pycache__/_available_service_aliases_oper
ations.cpython-39.pyc: no space left on device: exit status 1\n", "stderr_lines": ["Error: creating
container storage: creating an ID-mapped copy of layer
\"98955f43cc908bd50ff43585fec2c7dd9445eaf05eecd1e3144f93ffc00ed4ba\": error during chown:
storage-chown-by-maps: lchown usr/local/lib/python3.9/site-
packages/azure/mgmt/network/v2019_11_01/operations/__pycache__/_available_service_aliases_oper
ations.cpython-39.pyc: no space left on device: exit status 1"], "stdout": "", "stdout_lines": []}
```

If you are installing a **/home** filesystem into a default Amazon Web Services marketplace RHEL instance, it might be too small since **/home** is part of the root / filesystem. To resolve this issue you must make more space available. For more information about the system requirements, see System requirements.

"Install container tools" task fails due to unavailable packages

This error can be seen in the installation process output as the following:

```
TASK [ansible.containerized_installer.common : Install container tools]
**************************************************************************************************
fatal: [192.0.2.1]: FAILED! => {"changed": false, "failures": ["No package crun available.", "No
package podman available.", "No package slirp4netns available.", "No package fuse-overlayfs
available."], "msg": "Failed to install some of the specified packages", "rc": 1, "results": []}
fatal: [192.0.2.2]: FAILED! => {"changed": false, "failures": ["No package crun available.", "No
package podman available.", "No package slirp4netns available.", "No package fuse-overlayfs
available."], "msg": "Failed to install some of the specified packages", "rc": 1, "results": []}
fatal: [192.0.2.3]: FAILED! => {"changed": false, "failures": ["No package crun available.", "No
package podman available.", "No package slirp4netns available.", "No package fuse-overlayfs
available."], "msg": "Failed to install some of the specified packages", "rc": 1, "results": []}
fatal: [192.0.2.4]: FAILED! => {"changed": false, "failures": ["No package crun available.", "No
package podman available.", "No package slirp4netns available.", "No package fuse-overlayfs
available."], "msg": "Failed to install some of the specified packages", "rc": 1, "results": []}
fatal: [192.0.2.5]: FAILED! => {"changed": false, "failures": ["No package crun available.", "No
package podman available.", "No package slirp4netns available.", "No package fuse-overlayfs
available."], "msg": "Failed to install some of the specified packages", "rc": 1, "results": []}
```

-

To fix this error, run the following command on the target hosts:

```
sudo subscription-manager register
```

A.4. TROUBLESHOOTING CONTAINERIZED ANSIBLE AUTOMATION PLATFORM CONFIGURATION

Use this information to troubleshoot your containerized Ansible Automation Platform configuration.

Sometimes the post install for seeding my Ansible Automation Platform content errors out

This could manifest itself as output similar to this:

```
TASK [infra.controller_configuration.projects : Configure Controller Projects | Wait for finish the
projects creation] ************************************
Friday 29 September 2023  11:02:32 +0100 (0:00:00.443)       0:00:53.521 ******
FAILED - RETRYING: [daap1.lan]: Configure Controller Projects | Wait for finish the projects creation
(1 retries left).
failed: [daap1.lan] (item={'failed': 0, 'started': 1, 'finished': 0, 'ansible_job_id': '536962174348.33944',
'results_file': '/home/aap/.ansible_async/536962174348.33944', 'changed': False,
'__controller_project_item': {'name': 'AAP Config-As-Code Examples', 'organization': 'Default',
'scm_branch': 'main', 'scm_clean': 'no', 'scm_delete_on_update': 'no', 'scm_type': 'git',
'scm_update_on_launch': 'no', 'scm_url': 'https://github.com/user/repo.git'}, 'ansible_loop_var':
'__controller_project_item'}) => {"__projects_job_async_results_item": {"__controller_project_item":
{"name": "AAP Config-As-Code Examples", "organization": "Default", "scm_branch": "main",
"scm_clean": "no", "scm_delete_on_update": "no", "scm_type": "git", "scm_update_on_launch": "no",
"scm_url": "https://github.com/user/repo.git"}, "ansible_job_id": "536962174348.33944",
"ansible_loop_var": "__controller_project_item", "changed": false, "failed": 0, "finished": 0,
"results_file": "/home/aap/.ansible_async/536962174348.33944", "started": 1}, "ansible_job_id":
"536962174348.33944", "ansible_loop_var": "__projects_job_async_results_item", "attempts": 30,
"changed": false, "finished": 0, "results_file": "/home/aap/.ansible_async/536962174348.33944",
"started": 1, "stderr": "", "stderr_lines": [], "stdout": "", "stdout_lines": []}
```

The **infra.controller_configuration.dispatch** role uses an asynchronous loop with 30 retries to apply each configuration type, and the default delay between retries is 1 second. If the configuration is large, this might not be enough time to apply everything before the last retry occurs.

Increase the retry delay by setting the **controller_configuration_async_delay** variable to 2 seconds for example. You can set this variable in the **[all:vars]** section of the installation program inventory file.

Re-run the installation program to ensure everything works as expected.

A.5. CONTAINERIZED ANSIBLE AUTOMATION PLATFORM REFERENCE

Use this information to understand the architecture for your containerized Ansible Automation Platform deployment.

Can you give details of the architecture for the Ansible Automation Platform containerized design?

We use as much of the underlying native Red Hat Enterprise Linux technology as possible. Podman is used for the container runtime and management of services.

Use **podman ps** to list the running containers on the system:

```
$ podman ps

CONTAINER ID  IMAGE                                                                COMMAND               CREATED
STATUS         PORTS       NAMES
88ed40495117  registry.redhat.io/rhel8/postgresql-13:latest                        run-postgresql        48
minutes ago  Up 47 minutes              postgresql
8f55ba612f04  registry.redhat.io/rhel8/redis-6:latest                              run-redis             47
minutes ago  Up 47 minutes              redis
56c40445c590  registry.redhat.io/ansible-automation-platform-24/ee-supported-rhel8:latest
/usr/bin/receptor...  47 minutes ago  Up 47 minutes              receptor
f346f05d56ee  registry.redhat.io/ansible-automation-platform-24/controller-rhel8:latest
/usr/bin/launch_a...  47 minutes ago  Up 45 minutes              automation-controller-rsyslog
26e3221963e3  registry.redhat.io/ansible-automation-platform-24/controller-rhel8:latest
/usr/bin/launch_a...  46 minutes ago  Up 45 minutes              automation-controller-task
c7ac92a1e8a1  registry.redhat.io/ansible-automation-platform-24/controller-rhel8:latest
/usr/bin/launch_a...  46 minutes ago  Up 28 minutes              automation-controller-web
```

Use **podman images** to display information about locally stored images:

```
$ podman images

REPOSITORY                                                TAG         IMAGE ID      CREATED      SIZE
registry.redhat.io/ansible-automation-platform-24/ee-supported-rhel8  latest      b497bdbee59e  10
days ago  3.16 GB
registry.redhat.io/ansible-automation-platform-24/controller-rhel8    latest      ed8ebb1c1baa  10 days
ago  1.48 GB
registry.redhat.io/rhel8/redis-6                          latest      78905519bb05  2 weeks ago  357 MB
registry.redhat.io/rhel8/postgresql-13                      latest      9b65bc3d0413  2 weeks ago  765
MB
```

Containerized Ansible Automation Platform runs as rootless containers for enhanced security by default. This means you can install containerized Ansible Automation Platform by using any local unprivileged user account. Privilege escalation is only needed for certain root level tasks, and by default is not needed to use root directly.

The installation program adds the following files to the filesystem where you run the installation program on the underlying Red Hat Enterprise Linux host:

```
$ tree -L 1
.
├── aap_install.log
├── ansible.cfg
├── collections
├── galaxy.yml
├── inventory
├── LICENSE
├── meta
├── playbooks
├── plugins
├── README.md
├── requirements.yml
├── roles
```

The installation root directory includes other containerized services that make use of Podman volumes.

Here are some examples for further reference:

The **containers** directory includes some of the Podman specifics used and installed for the execution plane:

```
containers/
├── podman
├── storage
│   ├── defaultNetworkBackend
│   ├── libpod
│   ├── networks
│   ├── overlay
│   ├── overlay-containers
│   ├── overlay-images
│   ├── overlay-layers
│   ├── storage.lock
│   └── userns.lock
└── storage.conf
```

The **controller** directory has some of the installed configuration and runtime data points:

```
controller/
├── data
│   ├── job_execution
│   ├── projects
│   └── rsyslog
├── etc
│   ├── conf.d
│   ├── launch_awx_task.sh
│   ├── settings.py
│   ├── tower.cert
│   └── tower.key
├── nginx
│   └── etc
├── rsyslog
│   └── run
└── supervisor
    └── run
```

The **receptor** directory has the automation mesh configuration:

```
receptor/
├── etc
│   └── receptor.conf
└── run
    ├── receptor.sock
    └── receptor.sock.lock
```

After installation, you will also find other files in the local user's /**home** directory such as the **.cache** directory:

```
.cache/
├── containers
```

```
|   └── short-name-aliases.conf.lock
└── rhsm
    └── rhsm.log
```

As services are run using rootless Podman by default, you can use other services such as running **systemd** as non-privileged users. Under **systemd** you can see some of the component service controls available:

The **.config** directory:

```
.config/
├── cni
|   └── net.d
|       └── cni.lock
├── containers
|   ├── auth.json
|   └── containers.conf
└── systemd
    └── user
        ├── automation-controller-rsyslog.service
        ├── automation-controller-task.service
        ├── automation-controller-web.service
        ├── default.target.wants
        ├── podman.service.d
        ├── postgresql.service
        ├── receptor.service
        ├── redis.service
        └── sockets.target.wants
```

This is specific to Podman and conforms to the Open Container Initiative (OCI) specifications. When you run Podman as the root user **/var/lib/containers** is used by default. For standard users the hierarchy under **$HOME/.local** is used.

The **.local** directory:

```
.local/
└── share
    └── containers
        ├── cache
        ├── podman
        └── storage
```

As an example **.local/storage/volumes** contains what the output from **podman volume ls** provides:

```
$ podman volume ls

DRIVER      VOLUME NAME
local       d73d3fe63a957bee04b4853fd38c39bf37c321d14fdab9ee3c9df03645135788
local       postgresql
local       redis_data
local       redis_etc
local       redis_run
```

The execution plane is isolated from the control plane main services to ensure it does not affect the main services.

Control plane services run with the standard Podman configuration and can be found in: **~/.local/share/containers/storage**.

Execution plane services (automation controller, Event-Driven Ansible and execution nodes) use a dedicated configuration found in **~/aap/containers/storage.conf**. This separation prevents execution plane containers from affecting the control plane services.

You can view the execution plane configuration with one of the following commands:

```
CONTAINERS_STORAGE_CONF=~/aap/containers/storage.conf podman <subcommand>
```

```
CONTAINER_HOST=unix://run/user/<user uid>/podman/podman.sock podman <subcommand>
```

How can I see host resource utilization statistics?

Run the following command to display host resource utilization statistics:

```
$ podman container stats -a
```

Example output based on a Dell sold and offered containerized Ansible Automation Platform solution (DAAP) install that utilizes ~1.8 GB RAM:

```
ID           NAME                        CPU %       MEM USAGE / LIMIT  MEM %       NET IO      BLOCK
IO    PIDS        CPU TIME    AVG CPU %
0d5d8eb93c18  automation-controller-web      0.23%       959.1MB / 3.761GB  25.50%      0B / 0B
0B / 0B     16          20.885142s  1.19%
3429d559836d  automation-controller-rsyslog  0.07%       144.5MB / 3.761GB  3.84%       0B / 0B
0B / 0B     6           4.099565s   0.23%
448d0bae0942  automation-controller-task     1.51%       633.1MB / 3.761GB  16.83%      0B / 0B     0B
/ 0B     33          34.285272s  1.93%
7f140e65b57e  receptor                    0.01%       5.923MB / 3.761GB  0.16%       0B / 0B     0B / 0B
7           1.010613s   0.06%
c1458367ca9c  redis                       0.48%       10.52MB / 3.761GB  0.28%       0B / 0B     0B / 0B     5
9.074042s   0.47%
ef712cc2dc89  postgresql                  0.09%       21.88MB / 3.761GB  0.58%       0B / 0B     0B / 0B
21          15.571059s  0.80%
```

How much storage is used and where?

The container volume storage is under the local user at **$HOME/.local/share/containers/storage/volumes**.

1. To view the details of each volume, run the following command:

   ```
   $ podman volume ls
   ```

2. Run the following command to display detailed information about a specific volume:

   ```
   $ podman volume inspect <volume_name>
   ```

For example:

```
$ podman volume inspect postgresql
```

Example output:

```
[
    {
        "Name": "postgresql",
        "Driver": "local",
        "Mountpoint": "/home/aap/.local/share/containers/storage/volumes/postgresql/_data",
        "CreatedAt": "2024-01-08T23:39:24.983964686Z",
        "Labels": {},
        "Scope": "local",
        "Options": {},
        "MountCount": 0,
        "NeedsCopyUp": true
    }
]
```

Several files created by the installation program are located in **$HOME/aap/** and bind-mounted into various running containers.

1. To view the mounts associated with a container run the following command:

   ```
   $ podman ps --format "{{.ID}}\t{{.Command}}\t{{.Names}}"
   ```

   Example output:

   ```
   89e779b81b83 run-postgresql postgresql
   4c33cc77ef7d run-redis redis
   3d8a028d892d /usr/bin/receptor... receptor
   09821701645c /usr/bin/launch_a... automation-controller-rsyslog
   a2ddb5cac71b /usr/bin/launch_a... automation-controller-task
   fa0029a3b003 /usr/bin/launch_a... automation-controller-web
   20f192534691 gunicorn --bind 1... automation-eda-api
   f49804c7e6cb daphne -b 127.0.0... automation-eda-daphne
   d340b9c1cb74 /bin/sh -c nginx ... automation-eda-web
   111f47de5205 aap-eda-manage rq... automation-eda-worker-1
   171fcb1785af aap-eda-manage rq... automation-eda-worker-2
   049d10555b51 aap-eda-manage rq... automation-eda-activation-worker-1
   7a78a41a8425 aap-eda-manage rq... automation-eda-activation-worker-2
   da9afa8ef5e2 aap-eda-manage sc... automation-eda-scheduler
   8a2958be9baf gunicorn --name p... automation-hub-api
   0a8b57581749 gunicorn --name p... automation-hub-content
   68005b987498 nginx -g daemon o... automation-hub-web
   cb07af77f89f pulpcore-worker automation-hub-worker-1
   a3ba05136446 pulpcore-worker automation-hub-worker-2
   ```

2. Run the following command:

   ```
   $ podman inspect <container_name> | jq -r .[].Mounts[].Source
   ```

   Example output:

   ```
   /home/aap/.local/share/containers/storage/volumes/receptor_run/_data
   ```

```
/home/aap/.local/share/containers/storage/volumes/redis_run/_data
/home/aap/aap/controller/data/rsyslog
/home/aap/aap/controller/etc/tower.key
/home/aap/aap/controller/etc/conf.d/callback_receiver_workers.py
/home/aap/aap/controller/data/job_execution
/home/aap/aap/controller/nginx/etc/controller.conf
/home/aap/aap/controller/etc/conf.d/subscription_usage_model.py
/home/aap/aap/controller/etc/conf.d/cluster_host_id.py
/home/aap/aap/controller/etc/conf.d/insights.py
/home/aap/aap/controller/rsyslog/run
/home/aap/aap/controller/data/projects
/home/aap/aap/controller/etc/settings.py
/home/aap/aap/receptor/etc/receptor.conf
/home/aap/aap/controller/etc/conf.d/execution_environments.py
/home/aap/aap/tls/extracted
/home/aap/aap/controller/supervisor/run
/home/aap/aap/controller/etc/uwsgi.ini
/home/aap/aap/controller/etc/conf.d/container_groups.py
/home/aap/aap/controller/etc/launch_awx_task.sh
/home/aap/aap/controller/etc/tower.cert
```

3. If the **jq** RPM is not installed, install it by running the following command:

```
$ sudo dnf -y install jq
```

APPENDIX B. INVENTORY FILE VARIABLES

The following tables contain information about the variables used in Ansible Automation Platform's installation **inventory** files. The tables include the variables that you can use for RPM-based installation and container-based installation.

B.1. ANSIBLE VARIABLES

The following variables control how Ansible Automation Platform interacts with remote hosts.

Table B.1. Ansible variables

Variable	Description
ansible_connection	The connection plugin used for the task on the target host. This can be the name of any Ansible connection plugin. SSH protocol types are **smart**, **ssh**, or **paramiko**. You can also use **local** to run tasks on the control node itself. Default = **smart**
ansible_host	The IP address or name of the target host to use instead of **inventory_hostname**.
ansible_password	The password to authenticate to the host. Do not store this variable in plain text. Always use a vault. For more information, see Keep vaulted variables safely visible.
ansible_port	The connection port number. The default for SSH is **22**.
ansible_scp_extra_args	This setting is always appended to the default **scp** command line.
ansible_sftp_extra_args	This setting is always appended to the default **sftp** command line.
ansible_shell_executable	This sets the shell that the Ansible controller uses on the target machine and overrides the executable in **ansible.cfg** which defaults to **/bin/sh**.

Variable	Description
ansible_shell_type	The shell type of the target system. Do not use this setting unless you have set the **ansible_shell_executable** to a non-Bourne (sh) compatible shell. By default commands are formatted using sh-style syntax. Setting this to **csh** or **fish** causes commands executed on target systems to follow the syntax of those shells instead.
ansible_ssh_common_args	This setting is always appended to the default command line for **sftp**, **scp**, and **ssh**. Useful to configure a **ProxyCommand** for a certain host or group.
ansible_ssh_executable	This setting overrides the default behavior to use the system **ssh**. This can override the**ssh_executable** setting in **ansible.cfg**.
ansible_ssh_extra_args	This setting is always appended to the default **ssh** command line.
ansible_ssh_pipelining	Determines if SSH **pipelining** is used. This can override the **pipelining** setting in **ansible.cfg**. If using SSH key-based authentication, the key must be managed by an SSH agent.
ansible_ssh_private_key_file	Private key file used by SSH. Useful if using multiple keys and you do not want to use an SSH agent.
ansible_user	The user name to use when connecting to the host. Do not change this variable unless **/bin/sh** is not installed on the target machine or cannot be run from sudo.
inventory_hostname	This variable takes the hostname of the machine from the inventory script or the Ansible configuration file. You cannot set the value of this variable. Because the value is taken from the configuration file, the actual runtime hostname value can vary from what is returned by this variable.

Additional resources

- Ansible.Builtin

- Ansible Configuration Settings

B.2. AUTOMATION HUB VARIABLES

Inventory file variables for automation hub.

RPM variable name	Container variable name	Description	Required or optional	Default
automationhub_admin_password	**hub_admin_password**	Automation hub administrator password. Use of special characters for this variable is limited. The password can include any printable ASCII character except /, ", or @.	Required	
automationhub_api_token		Set the existing token for the installation program. For example, a regenerated token in the automation hub UI will invalidate an existing token. Use this variable to set that token in the installation program the next time you run the installation program.	Optional	
automationhub_auto_sign_collections	**hub_collection_auto_sign**	If a collection signing service is enabled, collections are not signed automatically by default. Set this variable to **true** to sign collections by default.	Optional	**false**
automationhub_backup_collections		Ansible automation hub provides artifacts in **/var/lib/pulp**. These artifacts are automatically backed up by default. Set this variable to **false** to prevent backup or restore of **/var/lib/pulp**.	Optional	**true**
automationhub_client_max_body_size	**hub_nginx_client_max_body_size**	Maximum allowed size for data sent to automation hub through NGINX.	Optional	**20m**
automationhub_collection_download_count		Denote whether or not the collection download count should be displayed in the UI.	Optional	**false**

RPM variable name	Container variable name	Description	Required or optional	Default
automationhub_collection_seed_repository		Controls the type of content to upload when **hub_seed_collections** is set to **true**. Valid options include: **certified**, **validated**	Optional	Both certified and validated are enabled by default.
automationhub_collection_signing_service_key	**hub_collection_signing_key**	Path to the collection signing key file.	Required if a collection signing service is enabled.	
automationhub_container_repair_media_type		Denote whether or not to run the command **pulpcore-manager container-repair-media-type**. Valid options include: **true**, **false**, **auto**	Optional	**auto**
automationhub_container_signing_service_key	**hub_container_signing_key**	Path to the container signing key file.	Required if a container signing service is enabled.	
automationhub_create_default_collection_signing_service	**hub_collection_signing**	Set this variable to **true** to enable a collection signing service.	Optional	**false**
automationhub_create_default_container_signing_service	**hub_container_signing**	Set this variable to **true** to enable a container signing service.	Optional	**false**

RPM variable name	Container variable name	Description	Required or optional	Default
	hub_data_path_exclude	automation hub backup path to exclude.	Optional	**[]**
automationhub_disable_hsts	**hub_nginx_disable_hsts**	Controls whether HTTP Strict Transport Security (HSTS) is enabled or disabled for automation hub. Set this variable to **true** to disable HSTS.	Optional	**false**
automationhub_disable_https	**hub_nginx_disable_https**	Controls whether HTTPS is enabled or disabled for automation hub. Set this variable to **true** to disable HTTPS.	Optional	**false**
automationhub_enable_api_access_log		Controls whether logging is enabled or disabled at **/var/log/galaxy_api_access.log**. The file logs all user actions made to the platform, including username and IP address. Set this variable to **true** to enable this logging.	Optional	**false**
automationhub_enable_unauthenticated_collection_access		Controls whether read-only access is enabled or disabled for unauthorized users viewing collections or namespaces for automation hub. Set this variable to **true** to enable read-only access.	Optional	**false**
automationhub_enable_unauthenticated_collection_download		Controls whether or not unauthorized users can download read-only collections from automation hub. Set this variable to **true** to enable download of read-only collections.	Optional	**false**

RPM variable name	Container variable name	Description	Required or optional	Default
automationhub_firewalld_zone	**hub_firewall_zone**	The firewall zone where automation hub related firewall rules are applied. This controls which networks can access automation hub based on the zone's trust level.	Optional	RPM = no default set. Container = **public**.
automationhub_force_change_admin_password		Denote whether or not to require the change of the default administrator password for automation hub during installation. Set to **true** to require the user to change the default administrator password during installation.	Optional	**false**
automationhub_importer_settings	**hub_galaxy_importer**	Dictionary of settings to pass to the **galaxy-importer.cfg** configuration file. These settings control how the **galaxy-importer** service processes and validates Ansible content. Example values include: **ansible-doc**, **ansible-lint**, and **flake8**.	Optional	
automationhub_nginx_tls_files_remote		Denote whether the web certificate sources are local to the installation program (**false**) or on the remote component server (**true**).	Optional	The value defined in **automationhub_tls_files_remote**.
automationhub_pg_cert_auth	**hub_pg_cert_auth**	Controls whether client certificate authentication is enabled or disabled on the automation hub PostgreSQL database. Set this variable to **true** to enable client certificate authentication.	Optional	**false**

RPM variable name	Container variable name	Description	Required or optional	Default
automationhub_pg_database	**hub_pg_database**	Name of the PostgreSQL database used by automation hub.	Optional	RPM = **automationhub**. Container = **pulp**
automationhub_pg_host	**hub_pg_host**	Hostname of the PostgreSQL database used by automation hub.	Required	RPM = **127.0.0.1**. Container = no default.
automationhub_pg_password	**hub_pg_password**	Password for the automation hub PostgreSQL database user. Use of special characters for this variable is limited. The **!**, **#**, **0** and **@** characters are supported. Use of other special characters can cause the setup to fail.	Optional	
automationhub_pg_port	**hub_pg_port**	Port number for the PostgreSQL database used by automation hub.	Optional	**5432**
automationhub_pg_sslmode	**hub_pg_sslmode**	Controls the SSL/TLS mode to use when automation hub connects to the PostgreSQL database. Valid options include **verify-full**, **verify-ca**, **require**, **prefer**, **allow**, **disable**.	Optional	**prefer**

RPM variable name	Container variable name	Description	Required or optional	Default
automationhub_pg_username	**hub_pg_username**	Username for the automation hub PostgreSQL database user.	Optional	RPM = **automationhub**. Container = **pulp**.
automationhub_pgclient_sslcert	**hub_pg_tls_cert**	Path to the PostgreSQL SSL/TLS certificate file for automation hub.	Required if using client certificate authentication.	
automationhub_pgclient_sslkey	**hub_pg_tls_key**	Path to the PostgreSQL SSL/TLS key file for automation hub.	Required if using client certificate authentication.	
automationhub_pgclient_tls_files_remote		Denote whether the PostgreSQL client certificate sources are local to the installation program (**false**) or on the remote component server (**true**).	Optional	The value defined in **automationhub_tls_files_remote**.

RPM variable name	Container variable name	Description	Required or optional	Default
automationhub_require_content_approval		Controls whether content signing is enabled or disabled for automation hub. By default when you upload collections to automation hub, an administrator must approve it before they are made available to users. To disable the content approval flow, set the variable to **false**.	Optional	**true**
automationhub_restore_signing_keys		Controls whether or not existing signing keys should be restored from a backup. Set to **false** to disable restoration of existing signing keys.	Optional	**true**
automationhub_seed_collections	**hub_seed_collections**	Controls whether or not pre-loading of collections is enabled. When you run the bundle installer, validated content is uploaded to the **validated** repository, and certified content is uploaded to the **rh-certified** repository. By default, certified content and validated content are both uploaded. If you do not want to pre-load content, set this variable to **false**. For the RPM-based installer, if you only want one type of content, set this variable to **true** and set the **automationhub_collection_seed_repository** variable to the type of content you want to include.	Optional	**true**
automationhub_ssl_cert	**hub_tls_cert**	Path to the SSL/TLS certificate file for automation hub.	Optional	
automationhub_ssl_key	**hub_tls_key**	Path to the SSL/TLS key file for automation hub.	Optional	

RPM variable name	Container variable name	Description	Required or optional	Default
automationhub_tls_files_remote	**hub_tls_remote**	Denote whether the automation hub provided certificate files are local to the installation program (**false**) or on the remote component server (**true**).	Optional	**false**
automationhub_use_archive_compression	**hub_use_archive_compression**	Controls whether archive compression is enabled or disabled for automation hub. You can control this functionality globally by using **use_archive_compression**.	Optional	**true**
automationhub_use_db_compression	**hub_use_db_compression**	Controls whether database compression is enabled or disabled for automation hub. You can control this functionality globally by using **use_db_compression**.	Optional	**true**
automationhub_user_headers	**hub_nginx_user_headers**	List of additional NGINX headers to add to automation hub's NGINX configuration.	Optional	**[]**
ee_from_hub_only		Controls whether automation hub is the only registry for execution environment images. If set to **true**, automation hub is the exclusive registry. If set to **false**, images are also pulled directly from Red Hat.	Optional	**true** when using the bundle installer, otherwise **false**.
generate_automationhub_token		Controls whether or not a token is generated for automation hub during installation. By default, a token is automatically generated during a fresh installation. If set to **true**, a token is regenerated during installation.	Optional	**false**

RPM variable name	Container variable name	Description	Required or optional	Default

	hub_extra_settings	Defines additional settings for use by automation hub during installation.
		
		For example:
		
		`hub_extra_settings:`
		`  - setting: REDIRECT_IS_HTTPS`
		`  value: True`
			Optional	**[]**

nginx_hsts_max_age	**hub_nginx_hsts_max_age**	Maximum duration (in seconds) that HTTP Strict Transport Security (HSTS) is enforced for automation hub.	Optional	**63072000**
pulp_secret	**hub_secret_key**	Secret key value used by automation hub to sign and encrypt data.	Optional	
	hub_azure_account_key	Azure blob storage account key.	Required if using an Azure blob storage backend.	
	hub_azure_account_name	Account name associated with the Azure blob storage.	Required when using an Azure blob storage backend.	
	hub_azure_container	Name of the Azure blob storage container.	Optional	**pulp**

RPM variable name	Container variable name	Description	Required or optional	Default
	hub_azure_extra_settings	Defines extra parameters for the Azure blob storage backend. For more information about the list of parameters, see django-storages documentation - Azure Storage.	Optional	**{}**
	hub_collection_signing_pass	Password for the automation content collection signing service.	Required if the collection signing service is protected by a passphrase.	
	hub_collection_signing_service	Service for signing collections.	Optional	**ansible-default**
	hub_container_signing_pass	Password for the automation content container signing service.	Required if the container signing service is protected by a passphrase.	
	hub_container_signing_service	Service for signing containers.	Optional	**container-default**
	hub_nginx_http_port	Port number that automation hub listens on for HTTP requests.	Optional	**8081**

RPM variable name	Container variable name	Description	Required or optional	Default
	hub_nginx_https_port	Port number that automation hub listens on for HTTPS requests.	Optional	**8444**
nginx_tls_protocols	**hub_nginx_https_protocols**	Protocols that automation hub will support when handling HTTPS traffic.	Optional	RPM = **[TLSv1.2]**. Container = **[TLSv1.2, TLSv1.3]**.
	hub_pg_socket	UNIX socket used by automation hub to connect to the PostgreSQL database.	Optional	
	hub_s3_access_key	AWS S3 access key.	Required if using an AWS S3 storage backend.	
	hub_s3_bucket_name	Name of the AWS S3 storage bucket.	Optional	**pulp**
	hub_s3_extra_settings	Used to define extra parameters for the AWS S3 storage backend. For more information about the list of parameters, see django-storages documentation - Amazon S3.	Optional	**{}**

RPM variable name	Container variable name	Description	Required or optional	Default
	hub_s3_secret_key	AWS S3 secret key.	Required if using an AWS S3 storage backend.	
	hub_shared_data_mount_opts	Mount options for the Network File System (NFS) share.	Optional	**rw,sync,hard**
	hub_shared_data_path	Path to the Network File System (NFS) share with read, write, and execute (RWX) access. The value must match the format **host:dir**, for example **nfs-server.example.com:/exports/hub**.	Required if installing more than one instance of automation hub with a **file** storage backend. When installing a single instance of automation hub, it is optional.	
	hub_storage_backend	Automation hub storage backend type. Possible values include: **azure**, **file**, **s3**.	Optional	**file**

RPM variable name	Container variable name	Description	Required or optional	Default
	hub_workers	Number of automation hub workers.	Optional	**2**

B.3. AUTOMATION CONTROLLER VARIABLES

Inventory file variables for automation controller.

RPM variable name	Container variable name	Description	Required or optional	Default
admin_email	**controller_admin_email**	Email address used by Django for the admin user for automation controller.	Optional	**admin@example.com**
admin_password	**controller_admin_password**	Automation controller administrator password. Use of special characters for this variable is limited. The password can include any printable ASCII character except /, ", or @.	Required	
admin_username	**controller_admin_user**	Username used to identify and create the administrator user in automation controller.	Optional	**admin**
automationcontroller_client_max_body_size	**controller_nginx_client_max_body_size**	Maximum allowed size for data sent to automation controller through NGINX.	Optional	**5m**
automationcontroller_use_archive_compression	**controller_use_archive_compression**	Controls whether archive compression is enabled or disabled for automation controller. You can control this functionality globally by using **use_archive_compression**.	Optional	**true**

RPM variable name	Container variable name	Description	Required or optional	Default
automationcontroller_use_db_compression	**controller_use_db_compression**	Controls whether database compression is enabled or disabled for automation controller. You can control this functionality globally by using **use_db_compression**.	Optional	**true**
awx_pg_cert_auth	**controller_pg_cert_auth**	Controls whether client certificate authentication is enabled or disabled on the automation controller PostgreSQL database. Set this variable to **true** to enable client certificate authentication.	Optional	**false**
controller_firewalld_zone	**controller_firewall_zone**	The firewall zone where automation controller related firewall rules are applied. This controls which networks can access automation controller based on the zone's trust level.	Optional	**public**
controller_nginx_tls_files_remote		Denote whether the web certificate sources are local to the installation program (**false**) or on the remote component server (**true**).	Optional	The value defined in **controller_tls_files_remote**.
controller_pgclient_tls_files_remote		Denote whether the PostgreSQL client certificate sources are local to the installation program (**false**) or on the remote component server (**true**).	Optional	The value defined in **controller_tls_files_remote**.

RPM variable name	Container variable name	Description	Required or optional	Default
controller_tls_files_remote	**controller_tls_remote**	Denote whether the automation controller provided certificate files are local to the installation program (**false**) or on the remote component server (**true**).	Optional	**false**
nginx_disable_hsts	**controller_nginx_disable_hsts**	Controls whether HTTP Strict Transport Security (HSTS) is enabled or disabled for automation controller. Set this variable to **true** to disable HSTS.	Optional	**false**
nginx_disable_https	**controller_nginx_disable_https**	Controls whether HTTPS is enabled or disabled for automation controller. Set this variable to **true** to disable HTTPS.	Optional	**false**
nginx_hsts_max_age	**controller_nginx_hsts_max_age**	Maximum duration (in seconds) that HTTP Strict Transport Security (HSTS) is enforced for automation controller.	Optional	**63072000**
nginx_http_port	**controller_nginx_http_port**	Port number that automation controller listens on for HTTP requests.	Optional	RPM = **80**. Container = **8080**
nginx_https_port	**controller_nginx_https_port**	Port number that automation controller listens on for HTTPS requests.	Optional	RPM = **443**. Container = **8443**

RPM variable name	Container variable name	Description	Required or optional	Default
nginx_tls_protocols	**controller_nginx_https_protocols**	Protocols that automation controller supports when handling HTTPS traffic.	Optional	RPM = **[TLSv1.2]**. Container = **[TLSv1.2, TLSv1.3]**
nginx_user_headers	**controller_nginx_user_headers**	List of additional NGINX headers to add to automation controller's NGINX configuration.	Optional	**[]**
	controller_create_preload_data	Controls whether or not to create preloaded content during installation.	Optional	**true**
node_state		The status of a node or group of nodes. Valid options include **active**, **deprovision** to remove a node from a cluster, or **iso_migrate** to migrate a legacy isolated node to an execution node.	Optional	**active**

RPM variable name	Container variable name	Description	Required or optional	Default
node_type	See **receptor_type** for the container equivalent variable.	For the **[automationcontroller]** group the two options are: • **node_type=control** - The node only runs project and inventory updates, but not regular jobs. • **node_type=hybrid** - The node runs everything. For the **[execution_nodes]** group the two options are: • **node_type=hop** - The node forwards jobs to an execution node. • **node_type=execution** - The node can run jobs.	Optional	For **[automationcontroller]** = **hybrid**, for **[execution_nodes]** = **execution**
peers	See **receptor_peers** for the container equivalent variable.	Used to indicate which nodes a specific host or group connects to. Wherever this variable is defined, an outbound connection to the specific host or group is established. This variable can be a comma-separated list of hosts and groups from the inventory. This is resolved into a set of hosts that is used to construct the **receptor.conf** file.	Optional	
pg_database	**controller_pg_database**	Name of the PostgreSQL database used by automation controller.	Optional	**awx**
pg_host	**controller_pg_host**	Hostname of the PostgreSQL database used by automation controller.	Required	

RPM variable name	Container variable name	Description	Required or optional	Default
pg_password	**controller_pg_password**	Password for the automation controller PostgreSQL database user. Use of special characters for this variable is limited. The **!**, **#**, **0** and **@** characters are supported. Use of other special characters can cause the setup to fail.	Required if not using client certificate authentication.	
pg_port	**controller_pg_port**	Port number for the PostgreSQL database used by automation controller.	Optional	**5432**
pg_sslmode	**controller_pg_sslmode**	Controls the SSL/TLS mode to use when automation controller connects to the PostgreSQL database. Valid options include **verify-full**, **verify-ca**, **require**, **prefer**, **allow**, **disable**.	Optional	**prefer**
pg_username	**controller_pg_username**	Username for the automation controller PostgreSQL database user.	Optional	**awx**
pgclient_sslcert	**controller_pg_tls_cert**	Path to the PostgreSQL SSL/TLS certificate file for automation controller.	Required if using client certificate authentication.	

RPM variable name	Container variable name	Description	Required or optional	Default
pgclient_sslkey	**controller_pg_tls_key**	Path to the PostgreSQL SSL/TLS key file for automation controller.	Required if using client certificate authentication.	
precreate_partition_hours		Number of hours worth of events table partitions to pre-create before starting a backup to avoid **pg_dump** locks.	Optional	3
uwsgi_listen_queue_size	**controller_uwsgi_listen_queue_size**	Number of requests **uwsgi** allows in the queue on automation controller until **uwsgi_processes** can serve them.	Optional	**2048**
web_server_ssl_cert	**controller_tls_cert**	Path to the SSL/TLS certificate file for automation controller.	Optional	
web_server_ssl_key	**controller_tls_key**	Path to the SSL/TLS key file for automation controller.	Optional	
	controller_event_workers	Number of event workers that handle job-related events inside automation controller.	Optional	**4**

	controller_extra_settings	Defines additional settings for use by automation controller during installation.
		
		For example:
		
		`controller_extra_settings:`
		
		`- setting: USE_X_FORWARDED_HOST`
		`value: true`
			Optional	**[]**

RPM variable name	Container variable name	Description	Required or optional	Default
	controller_license_file	Path to the automation controller license file.		
	controller_percent_memory_capacity	Memory allocation for automation controller.	Optional	**1.0** (allocates 100% of the total system memory to automation controller)
	controller_pg_socket	UNIX socket used by automation controller to connect to the PostgreSQL database.	Optional	
	controller_secret_key	Secret key value used by automation controller to sign and encrypt data.	Optional	

B.4. DATABASE VARIABLES

Inventory file variables for the database used with Ansible Automation Platform.

RPM variable name	Container variable name	Description	Required or optional	Default
install_pg_port	**postgresql_port**	Port number for the PostgreSQL database.	Optional	**5432**

RPM variable name	Container variable name	Description	Required or optional	Default

postgres_extra_settings	**postgresql_extra_settings**	Defines additional settings for use by PostgreSQL.
		
		Example usage for RPM:
		
		`postgresql_extra_settings:`
		`  ssl_ciphers: 'HIGH:!aNULL:!MD5'`
		
		Example usage for containerized:
		
		`postgresql_extra_settings:`
		`  - setting: ssl_ciphers`
		`  value: 'HIGH:!aNULL:!MD5'`
			Optional	

postgres_firewalld_zone	**postgresql_firewall_zone**	The firewall zone where PostgreSQL related firewall rules are applied. This controls which networks can access PostgreSQL based on the zone's trust level.	Optional	RPM = no default set. Container = **public**.
postgres_max_connections	**postgresql_max_connections**	Maximum number of concurrent connections to the database if you are using an installer-managed database. For more information see PostgreSQL database configuration and maintenance for automation controller.	Optional	**1024**
postgres_ssl_cert	**postgresql_tls_cert**	Path to the PostgreSQL SSL/TLS certificate file.	Optional	
postgres_ssl_key	**postgresql_tls_key**	Path to the PostgreSQL SSL/TLS key file.	Optional	

RPM variable name	Container variable name	Description	Required or optional	Default
postgres_use_ssl	**postgresql_disable_tls**	Controls whether SSL/TLS is enabled or disabled for the PostgreSQL database.	Optional	**false**
	postgresql_admin_database	Database name used for connections to the PostgreSQL database server.	Optional	**postgres**
	postgresql_admin_password	Password for the PostgreSQL admin user. When used, the installation program creates each component's database and credentials.	Required if using **postgresql_admin_username**.	
	postgresql_admin_username	Username for the PostgreSQL admin user. When used, the installation program creates each component's database and credentials.	Optional	**postgres**
	postgresql_effective_cache_size	Memory allocation available (in MB) for caching data.	Optional	
	postgresql_keep_databases	Controls whether or not to keep databases during uninstall. This variable applies to databases managed by the installation program only, and not external (customer-managed) databases. Set to **true** to keep databases during uninstall.	Optional	**false**
	postgresql_log_destination	Destination for server log output.	Optional	**/dev/stderr**
	postgresql_password_encryption	The algorithm for encrypting passwords.	Optional	**scram-sha-256**
	postgresql_shared_buffers	Memory allocation (in MB) for shared memory buffers.	Optional	

RPM variable name	Container variable name	Description	Required or optional	Default
	postgresql_tls_remote	Denote whether the PostgreSQL provided certificate files are local to the installation program (**false**) or on the remote component server (**true**).	Optional	**false**
	postgresql_use_archive_compression	Controls whether archive compression is enabled or disabled for PostgreSQL. You can control this functionality globally by using **use_archive_compression**.	Optional	**true**

B.5. EVENT-DRIVEN ANSIBLE CONTROLLER VARIABLES

Inventory file variables for Event-Driven Ansible controller.

RPM variable name	Container variable name	Description	Required or optional	Default
automationedacontroller_activation_workers	**eda_activation_workers**	Number of workers used for ansible-rulebook activation pods in Event-Driven Ansible.	Optional	RPM = (# of cores or threads) * 2 + 1. Container = **2**
automationedacontroller_admin_email	**eda_admin_email**	Email address used by Django for the admin user for Event-Driven Ansible.	Optional	**admin@example.com**

RPM variable name	Container variable name	Description	Required or optional	Default
automationedacontroller_admin_password	**eda_admin_password**	Event-Driven Ansible administrator password. Use of special characters for this variable is limited. The password can include any printable ASCII character except /, ", or @.	Required	
automationedacontroller_admin_username	**eda_admin_user**	Username used to identify and create the administrator user in Event-Driven Ansible.	Optional	**admin**
automationedacontroller_backend_gunicorn_workers		Number of workers for handling the API served through Gunicorn on worker nodes.	Optional	**2**
automationedacontroller_cache_tls_files_remote		Denote whether the cache cert sources are local to the installation program (**false**) or on the remote component server (**true**).	Optional	**false**
automationedacontroller_client_regen_cert		Controls whether or not to regenerate Event-Driven Ansible client certificates for the platform cache. Set to **true** to regenerate Event-Driven Ansible client certificates.	Optional	**false**
automationedacontroller_default_workers	**eda_workers**	Number of workers used in Event-Driven Ansible for application work.	Optional	Number of cores or threads
automationedacontroller_disable_hsts	**eda_nginx_disable_hsts**	Controls whether HTTP Strict Transport Security (HSTS) is enabled or disabled for Event-Driven Ansible. Set this variable to **true** to disable HSTS.	Optional	**false**

RPM variable name	Container variable name	Description	Required or optional	Default
automationedacontroller_disable_https	**eda_nginx_disable_https**	Controls whether HTTPS is enabled or disabled for Event-Driven Ansible. Set this variable to **true** to disable HTTPS.	Optional	**false**
automationedacontroller_event_stream_path	**eda_event_stream_prefix_path**	API prefix path used for Event-Driven Ansible event-stream through platform gateway.	Optional	**/eda-event-streams**
automationedacontroller_firewalld_zone	**eda_firewall_zone**	The firewall zone where Event-Driven Ansible related firewall rules are applied. This controls which networks can access Event-Driven Ansible based on the zone's trust level.	Optional	RPM = no default set. Container = **public**.
automationedacontroller_gunicorn_event_stream_workers		Number of workers for handling event streaming for Event-Driven Ansible.	Optional	**2**
automationedacontroller_gunicorn_workers	**eda_gunicorn_workers**	Number of workers for handling the API served through Gunicorn.	Optional	(Number of cores or threads) * 2 + 1
automationedacontroller_http_port	**eda_nginx_http_port**	Port number that Event-Driven Ansible listens on for HTTP requests.	Optional	RPM = **80**. Container = **8082**.
automationedacontroller_https_port	**eda_nginx_https_port**	Port number that Event-Driven Ansible listens on for HTTPS requests.	Optional	RPM = **443**. Container = **8445**.

RPM variable name	Container variable name	Description	Required or optional	Default
automationedacontroller_max_running_activations	**eda_max_running_activations**	Number of maximum activations running concurrently per node. This is an integer that must be greater than 0.	Optional	**12**
automationedacontroller_nginx_tls_files_remote		Denote whether the web cert sources are local to the installation program (**false**) or on the remote component server (**true**).	Optional	**false**
automationedacontroller_pg_cert_auth	**eda_pg_cert_auth**	Controls whether client certificate authentication is enabled or disabled on the Event-Driven Ansible PostgreSQL database. Set this variable to **true** to enable client certificate authentication.	Optional	**false**
automationedacontroller_pg_database	**eda_pg_database**	Name of the PostgreSQL database used by Event-Driven Ansible.	Optional	RPM = **automationedacontroller**. Container = **eda**.
automationedacontroller_pg_host	**eda_pg_host**	Hostname of the PostgreSQL database used by Event-Driven Ansible.	Required	
automationedacontroller_pg_password	**eda_pg_password**	Password for the Event-Driven Ansible PostgreSQL database user. Use of special characters for this variable is limited. The **!**, **#**, **0** and **@** characters are supported. Use of other special characters can cause the setup to fail.	Required if not using client certificate authentication.	

RPM variable name	Container variable name	Description	Required or optional	Default
automationedacontroller_pg_port	**eda_pg_port**	Port number for the PostgreSQL database used by Event-Driven Ansible.	Optional	**5432**
automationedacontroller_pg_sslmode	**eda_pg_sslmode**	Determines the level of encryption and authentication for client server connections. Valid options include **verify-full**, **verify-ca**, **require**, **prefer**, **allow**, **disable**.	Optional	**prefer**
automationedacontroller_pg_username	**eda_pg_username**	Username for the Event-Driven Ansible PostgreSQL database user.	Optional	RPM = **automationedacontroller**. Container = **eda**.
automationedacontroller_pgclient_sslcert	**eda_pg_tls_cert**	Path to the PostgreSQL SSL/TLS certificate file for Event-Driven Ansible.	Required if using client certificate authentication.	
automationedacontroller_pgclient_sslkey	**eda_pg_tls_key**	Path to the PostgreSQL SSL/TLS key file for Event-Driven Ansible.	Required if using client certificate authentication.	

RPM variable name	Container variable name	Description	Required or optional	Default
automationedacontroller_pgclient_tls_files_remote		Denote whether the PostgreSQL client cert sources are local to the installation program (**false**) or on the remote component server (**true**).	Optional	**false**
automationedacontroller_public_event_stream_url	**eda_event_stream_url**	URL for connecting to the event stream. The URL must start with the **http://** or **https://** prefix	Optional	
automationedacontroller_redis_host	**eda_redis_host**	Hostname of the Redis host used by Event-Driven Ansible.	Optional	First node in the **[automationgateway]** inventory group
automationedacontroller_redis_password	**eda_redis_password**	Password for Event-Driven Ansible Redis.	Optional	Randomly generated string

RPM variable name	Container variable name	Description	Required or optional	Default
automationedacontroller_redis_port	**eda_redis_port**	Port number for the Redis host for Event-Driven Ansible.	Optional	RPM = The value defined in platform gateway's implementation (**automationgateway_redis_port**). Container = **6379**
automationedacontroller_redis_username	**eda_redis_username**	Username for Event-Driven Ansible Redis.	Optional	**eda**
automationedacontroller_secret_key	**eda_secret_key**	Secret key value used by Event-Driven Ansible to sign and encrypt data.	Optional	
automationedacontroller_ssl_cert	**eda_tls_cert**	Path to the SSL/TLS certificate file for Event-Driven Ansible.	Optional	
automationedacontroller_ssl_key	**eda_tls_key**	Path to the SSL/TLS key file for Event-Driven Ansible.	Optional	
automationedacontroller_tls_files_remote	**eda_tls_remote**	Denote whether the Event-Driven Ansible provided certificate files are local to the installation program (**false**) or on the remote component server (**true**).	Optional	**false**

RPM variable name	Container variable name	Description	Required or optional	Default
automationedacontroller_trusted_origins		List of host addresses in the form: **<scheme>//:<address>:<port>** for trusted Cross-Site Request Forgery (CSRF) origins.	Optional	**[]**
automationedacontroller_use_archive_compression	**eda_use_archive_compression**	Controls whether archive compression is enabled or disabled for Event-Driven Ansible. You can control this functionality globally by using **use_archive_compression**.	Optional	**true**
automationedacontroller_use_db_compression	**eda_use_db_compression**	Controls whether database compression is enabled or disabled for Event-Driven Ansible. You can control this functionality globally by using **use_db_compression**.	Optional	**true**
automationedacontroller_user_headers	**eda_nginx_user_headers**	List of additional NGINX headers to add to Event-Driven Ansible's NGINX configuration.	Optional	**[]**
automationedacontroller_websocket_ssl_verify		Controls whether or not to perform SSL verification for the Daphne WebSocket used by Podman to communicate from the pod to the host. Set to **false** to disable SSL verification.	Optional	**true**
eda_node_type	**eda_type**	Event-Driven Ansible node type. Valid options include **api**, **event-stream**, **hybrid**, **worker**.	Optional	**hybrid**
	eda_debug	Controls whether debug mode is enabled or disabled for Event-Driven Ansible. Set to **true** to enable debug mode for Event-Driven Ansible.	Optional	**false**

RPM variable name	Container variable name	Description	Required or optional	Default

	eda_extra_settings	Defines additional settings for use by Event-Driven Ansible during installation.
		
		For example:
		
		`eda_extra_settings:`
		`  - setting: RULEBOOK_READINESS_TIMEOUT_SECONDS`
		`  value: 120`
			Optional	**[]**

	eda_nginx_client_max_body_size	Maximum allowed size for data sent to Event-Driven Ansible through NGINX.	Optional	**1m**
	eda_nginx_hsts_max_age	Maximum duration (in seconds) that HTTP Strict Transport Security (HSTS) is enforced for Event-Driven Ansible.	Optional	**63072000**
nginx_tls_protocols	**eda_nginx_https_protocols**	Protocols that Event-Driven Ansible supports when handling HTTPS traffic.	Optional	RPM = **[TLSv1.2]**. Container = **[TLSv1.2, TLSv1.3]**.
	eda_pg_socket	UNIX socket used by Event-Driven Ansible to connect to the PostgreSQL database.	Optional	
redis_disable_tls	**eda_redis_disable_tls**	Controls whether TLS is enabled or disabled for Event-Driven Ansible Redis. Set this variable to true to disable TLS.	Optional	**false**
	eda_redis_tls_cert	Path to the Event-Driven Ansible Redis certificate file.	Optional	

RPM variable name	Container variable name	Description	Required or optional	Default
	eda_redis_tls_key	Path to the Event-Driven Ansible Redis key file.	Optional	
	eda_safe_plugins	List of plugins that are allowed to run within Event-Driven Ansible. For more information, see Adding a safe plugin variable to Event-Driven Ansible controller.	Optional	**[]**

B.6. GENERAL VARIABLES

General inventory file variables for Ansible Automation Platform.

RPM variable name	Container variable name	Description	Required or optional	Default
aap_ca_cert_file	**ca_tls_cert**	Path to the user provided CA certificate file used to generate SSL/TLS certificates for all Ansible Automation Platform services. For more information, see Using custom TLS certificates.	Optional	
aap_ca_cert_files_remote	**ca_tls_remote**	Denote whether the CA certificate files are local to the installation program (**false**) or on the remote component server (**true**).	Optional	**false**
aap_ca_cert_size		Bit size of the internally managed CA certificate private key.	Optional	**4096**

RPM variable name	Container variable name	Description	Required or optional	Default
aap_ca_key_file	**ca_tls_key**	Path to the key file for the CA certificate provided in **aap_ca_cert_file** (RPM) and **ca_tls_cert** (Container). For more information, see Using custom TLS certificates.	Optional	
aap_ca_passphrase_cipher		Cipher used for signing the internally managed CA certificate private key.	Optional	**aes256**
aap_ca_regenerate		Denotes whether or not to regenerate the internally managed CA certificate key pair.	Optional	**false**
aap_service_cert_size		Bit size of the component key pair managed by the internal CA.	Optional	**4096**
aap_service_regen_cert		Denotes whether or not to regenerate the component key pair managed by the internal CA.	Optional	**false**
aap_service_san_records		A list of additional SAN records for signing a service. Assign these to components in the inventory file as host variables rather than group or all variables. All strings must also contain their corresponding SAN option prefix such as **DNS:** or **IP:**.	Optional	**[]**
backup_dest		Directory local to **setup.sh** for the final backup file.	Optional	The value defined in **setup_dir**.

RPM variable name	Container variable name	Description	Required or optional	Default
backup_dir	**backup_dir**	Directory used to store backup files.	Optional	RPM = **/var/backups/automation-platform/**. Container = **~/backups**
backup_file_prefix		Prefix used for the file backup name for the final backup file.	Optional	**automation-platform-backup**
bundle_install	**bundle_install**	Controls whether or not to perform an offline or bundled installation. Set this variable to **true** to enable an offline or bundled installation.	Optional	**false** if using the setup installation program. **true** if using the setup bundle installation program.

RPM variable name	Container variable name	Description	Required or optional	Default
bundle_install_folder	**bundle_dir**	Path to the bundle directory used when performing a bundle install.	Required if **bundle_install=true**	RPM = **/var/lib/ansible-automation-platform-bundle**. Container = **<current_dir>/bundle**.
custom_ca_cert	**custom_ca_cert**	Path to the custom CA certificate file. This is required if any of the TLS certificates you manually provided are signed by a custom CA. For more information, see Using custom TLS certificates.	Optional	
enable_insights_collection		The default install registers the node to the Red Hat Insights for Red Hat Ansible Automation Platform for the Red Hat Ansible Automation Platform Service if the node is registered with Subscription Manager. Set to **false** to disable this functionality.	Optional	**true**

RPM variable name	Container variable name	Description	Required or optional	Default
registry_password	**registry_password**	Password credential for access to the registry source defined in **registry_url**. For more information, see Setting registry_username and registry_password.	RPM = Required if you need a password to access **registry_url**. Container = Required if **registry_auth=true**.	
registry_url	**registry_url**	URL of the registry source from which to pull execution environment images.	Optional	**registry.redhat.io**
registry_username	**registry_username**	Username credential for access to the registry source defined in **registry_url**. For more information, see Setting registry_username and registry_password.	RPM = Required if you need a password to access **registry_url**. Container = Required if **registry_auth=true**.	
registry_verify_ssl	**registry_tls_verify**	Controls whether SSL/TLS certificate verification is enabled or disabled when making HTTPS requests.	Optional	**true**

RPM variable name	Container variable name	Description	Required or optional	Default
restore_backup_file		Path to the tar file used for the platform restore.	Optional	**{{ setup_dir }}/automation-platform-backup-latest.tar.gz**
restore_file_prefix		Path prefix for the staged restore components.	Optional	**automation-platform-restore**
routable_hostname	**routable_hostname**	Used if the machine running the installation program can only route to the target host through a specific URL. For example, if you use short names in your inventory, but the node running the installation program can only resolve that host by using a FQDN. If **routable_hostname** is not set, it defaults to **ansible_host**. If you do not set **ansible_host**, **inventory_hostname** is used as a last resort. This variable is used as a host variable for particular hosts and not under the **[all:vars]** section. For further information, see Assigning a variable to one machine: host variables.	Optional	

RPM variable name	Container variable name	Description	Required or optional	Default
use_archive_compression	**use_archive_compression**	Controls at a global level whether the filesystem-related backup files are compressed before being sent to the host to run the backup operation. If set to **true**, a **tar.gz** file is generated on each Ansible Automation Platform host and then gzip compression is used. If set to **false**, a simple tar file is generated. You can control this functionality at a component level by using the **<component_name>_use_archive_compression** variables.	Optional	**true**
use_db_compression	**use_db_compression**	Controls at a global level whether the database-related backup files are compressed before being sent to the host to run the backup operation. You can control this functionality at a component level by using the **<component_name>_use_db_compression** variables.	Optional	**true**
	ca_tls_key_passphrase	Passphrase used to decrypt the key provided in **ca_tls_key**.	Optional	
	client_request_timeout	Sets the HTTP timeout for end-user requests. The minimum value is **10** seconds.	Optional	**30**
	container_compress	Compression software to use for compressing container images.	Optional	**gzip**

RPM variable name	Container variable name	Description	Required or optional	Default
	container_keep_images	Controls whether or not to keep container images when uninstalling Ansible Automation Platform. Set to **true** to keep container images when uninstalling Ansible Automation Platform.	Optional	**false**
	container_pull_images	Controls whether or not to pull newer container images during installation. Set to **false** to prevent pulling newer container images during installation.	Optional	**true**
	images_tmp_dir	The directory where the installation program temporarily stores container images during installation.	Optional	The system's temporary directory.
	pcp_firewall_zone	The firewall zone where Performance Co-Pilot related firewall rules are applied. This controls which networks can access Performance Co-Pilot based on the zone's trust level.	Optional	public
	pcp_use_archive_compression	Controls whether archive compression is enabled or disabled for Performance Co-Pilot. You can control this functionality globally by using **use_archive_compression**.	Optional	**true**
	registry_auth	Set whether or not to use registry authentication. When this variable is set to true, **registry_username** and **registry_password** are required.	Optional	**true**

RPM variable name	Container variable name	Description	Required or optional	Default
	registry_ns_aap	Ansible Automation Platform registry namespace.	Optional	**ansible-automation-platform-26**
	registry_ns_rhel	RHEL registry namespace.	Optional	**rhel8**

B.7. IMAGE VARIABLES

Inventory file variables for images.

RPM variable name	Container variable name	Description	Required or optional	Default
extra_images		Additional container images to pull from the configured container registry during deployment.	Optional	**ansible-builder-rhel8**
	controller_image	Container image for automation controller.	Optional	**controller-rhel8:latest**
	de_extra_images	Additional decision environment container images to pull from the configured container registry during deployment.	Optional	**[]**
	de_supported_image	Supported decision environment container image.	Optional	**de-supported-rhel8:latest**

RPM variable name	Container variable name	Description	Required or optional	Default
	eda_image	Backend container image for Event-Driven Ansible.	Optional	**eda-controller-rhel8:latest**
	eda_web_image	Front-end container image for Event-Driven Ansible.	Optional	**eda-controller-ui-rhel8:latest**
	ee_extra_images	Additional execution environment container images to pull from the configured container registry during deployment.	Optional	**[]**
	ee_minimal_image	Minimal execution environment container image.	Optional	**ee-minimal-rhel8:latest**
	ee_supported_image	Supported execution environment container image.	Optional	**ee-supported-rhel8:latest**
	gateway_image	Container image for platform gateway.	Optional	**gateway-rhel8:latest**
	gateway_proxy_image	Container image for platform gateway proxy.	Optional	**gateway-proxy-rhel8:latest**

RPM variable name	Container variable name	Description	Required or optional	Default
	hub_image	Backend container image for automation hub.	Optional	**hub-rhel8:latest**
	hub_web_image	Front-end container image for automation hub.	Optional	**hub-web-rhel8:latest**
	pcp_image	Container image for Performance Co-Pilot.	Optional	**pcp:latest**
	postgresql_image	Container image for PostgreSQL.	Optional	**postgresql-15:latest**
	receptor_image	Container image for receptor.	Optional	**receptor-rhel8:latest**
	redis_image	Container image for Redis.	Optional	**redis-6:latest**

B.8. PLATFORM GATEWAY VARIABLES

Inventory file variables for platform gateway.

RPM variable name	Container variable name	Description	Required or optional	Default
automationgateway_admin_email	**gateway_admin_email**	Email address used by Django for the admin user for platform gateway.	Optional	**admin@example.com**

RPM variable name	Container variable name	Description	Required or optional	Default
automationgateway_admin_password	**gateway_admin_password**	Platform gateway administrator password. Use of special characters for this variable is limited. The password can include any printable ASCII character except /, ", or @.	Required	
automationgateway_admin_username	**gateway_admin_user**	Username used to identify and create the administrator user in platform gateway.	Optional	**admin**
automationgateway_cache_cert	**gateway_redis_tls_cert**	Path to the platform gateway Redis certificate file.	Optional	
automationgateway_cache_key	**gateway_redis_tls_key**	Path to the platform gateway Redis key file.	Optional	
automationgateway_cache_tls_files_remote		Denote whether the cache client certificate files are local to the installation program (**false**) or on the remote component server (**true**).	Optional	The value defined in **automationgateway_tls_files_remote** which defaults to **false**.
automationgateway_client_regen_cert		Controls whether or not to regenerate platform gateway client certificates for the platform cache. Set to **true** to regenerate platform gateway client certificates.	Optional	**false**
automationgateway_control_plane_port	**gateway_control_plane_port**	Port number for the platform gateway control plane.	Optional	**50051**

RPM variable name	Container variable name	Description	Required or optional	Default
automationgateway_disable_hsts	**gateway_nginx_disable_hsts**	Controls whether HTTP Strict Transport Security (HSTS) is enabled or disabled for platform gateway. Set this variable to **true** to disable HSTS.	Optional	**false**
automationgateway_disable_https	**gateway_nginx_disable_https**	Controls whether HTTPS is enabled or disabled for platform gateway. Set this variable to **true** to disable HTTPS.	Optional	RPM = The value defined in **disable_https** which defaults to **false**. Container = **false**.
automationgateway_firewalld_zone	**gateway_proxy_firewall_zone**	The firewall zone where platform gateway related firewall rules are applied. This controls which networks can access platform gateway based on the zone's trust level.	Optional	RPM = no default set. Container = 'public'.
automationgateway_grpc_auth_service_timeout	**gateway_grpc_auth_service_timeout**	Timeout duration (in seconds) for requests made to the gRPC service on platform gateway.	Optional	**30s**
automationgateway_grpc_server_max_threads_per_process	**gateway_grpc_server_max_threads_per_process**	Maximum number of threads that each gRPC server process can create to handle requests on platform gateway.	Optional	**10**
automationgateway_grpc_server_processes	**gateway_grpc_server_processes**	Number of processes for handling gRPC requests on platform gateway.	Optional	**5**

RPM variable name	Container variable name	Description	Required or optional	Default
automationgateway_http_port	**gateway_nginx_http_port**	Port number that platform gateway listens on for HTTP requests.	Optional	RPM = **8080**. Container = **8083**.
automationgateway_https_port	**gateway_nginx_https_port**	Port number that platform gateway listens on for HTTPS requests.	Optional	RPM = **8443**. Container = **8446**.
automationgateway_main_url	**gateway_main_url**	URL of the main instance of platform gateway that clients connect to. Use if you are performing a clustered deployment and you need to use the URL of the load balancer instead of the component's server. The URL must start with **http://** or **https://** prefix.	Optional	
automationgateway_nginx_tls_files_remote		Denote whether the web cert sources are local to the installation program (**false**) or on the remote component server (**true**).	Optional	The value defined in **automationgateway_tls_files_remote** which defaults to **false**.

RPM variable name	Container variable name	Description	Required or optional	Default
automationgateway_pg_cert_auth	**gateway_pg_cert_auth**	Controls whether client certificate authentication is enabled or disabled on the platform gateway PostgreSQL database. Set this variable to **true** to enable client certificate authentication.	Optional	**false**
automationgateway_pg_database	**gateway_pg_database**	Name of the PostgreSQL database used by platform gateway.	Optional	RPM = **automationgateway**. Container = **gateway**.
automationgateway_pg_host	**gateway_pg_host**	Hostname of the PostgreSQL database used by platform gateway.	Required	
automationgateway_pg_password	**gateway_pg_password**	Password for the platform gateway PostgreSQL database user. Use of special characters for this variable is limited. The **!**, **#**, **0** and **@** characters are supported. Use of other special characters can cause the setup to fail.	Optional	
automationgateway_pg_port	**gateway_pg_port**	Port number for the PostgreSQL database used by platform gateway.	Optional	**5432**
automationgateway_pg_sslmode	**gateway_pg_sslmode**	Controls the SSL mode to use when platform gateway connects to the PostgreSQL database. Valid options include **verify-full**, **verify-ca**, **require**, **prefer**, **allow**, **disable**.	Optional	**prefer**

RPM variable name	Container variable name	Description	Required or optional	Default
automationgateway_pg_username	**gateway_pg_username**	Username for the platform gateway PostgreSQL database user.	Optional	RPM = **automationgateway**. Container = **gateway**
automationgateway_pgclient_sslcert	**gateway_pg_tls_cert**	Path to the PostgreSQL SSL/TLS certificate file for platform gateway.	Required if using client certificate authentication.	
automationgateway_pgclient_sslkey	**gateway_pg_tls_key**	Path to the PostgreSQL SSL/TLS key file for platform gateway.	Required if using client certificate authentication.	
automationgateway_pgclient_tls_files_remote		Denote whether the PostgreSQL client cert sources are local to the installation program (**false**) or on the remote component server (**true**).	Optional	The value defined in **automationgateway_tls_files_remote** which defaults to **false**.

RPM variable name	Container variable name	Description	Required or optional	Default
automationgateway_redis_host	**gateway_redis_host**	Hostname of the Redis host used by platform gateway.	Optional	First node in the **[automationgateway]** inventory group.
automationgateway_redis_password	**gateway_redis_password**	Password for platform gateway Redis.	Optional	Randomly generated string.
automationgateway_redis_username	**gateway_redis_username**	Username for platform gateway Redis.	Optional	**gateway**
automationgateway_secret_key	**gateway_secret_key**	Secret key value used by platform gateway to sign and encrypt data.	Optional	
automationgateway_ssl_cert	**gateway_tls_cert**	Path to the SSL/TLS certificate file for platform gateway.	Optional	
automationgateway_ssl_key	**gateway_tls_key**	Path to the SSL/TLS key file for platform gateway.	Optional	
automationgateway_tls_files_remote	**gateway_tls_remote**	Denote whether the platform gateway provided certificate files are local to the installation program (**false**) or on the remote component server (**true**).	Optional	**false**

RPM variable name	Container variable name	Description	Required or optional	Default
automationgateway_uwsgi_processes	**gateway_uwsgi_processes**	The number of **uwsgi** processes for the platform gateway container. The value is calculated based on the number of available vCPUs (virtual CPUs).	Optional	The number of vCPUs multiplied by two, plus one.
automationgateway_use_archive_compression	**gateway_use_archive_compression**	Controls whether archive compression is enabled or disabled for platform gateway. You can control this functionality globally by using **use_archive_compression**.	Optional	**true**
automationgateway_use_db_compression	**gateway_use_db_compression**	Controls whether database compression is enabled or disabled for platform gateway. You can control this functionality globally by using **use_db_compression**.	Optional	**true**
automationgateway_user_headers	**gateway_nginx_user_headers**	List of additional NGINX headers to add to platform gateway's NGINX configuration.	Optional	**[]**
automationgateway_verify_ssl		Denotes whether or not to verify platform gateway's web certificates when making calls from platform gateway to itself during installation. Set to **false** to disable web certificate verification.	Optional	**true**

RPM variable name	Container variable name	Description	Required or optional	Default
automationgatewayproxy_disable_https	**envoy_disable_https**	Controls whether or not HTTPS is disabled when accessing the platform UI. Set to **true** to disable HTTPS (HTTP is used instead).	Optional	RPM = The value defined in **disable_https** which defaults to **false**. Container = **false**.
automationgatewayproxy_http_port	**envoy_http_port**	Port number on which the Envoy proxy listens for incoming HTTP connections.	Optional	**80**
automationgatewayproxy_https_port	**envoy_https_port**	Port number on which the Envoy proxy listens for incoming HTTPS connections.	Optional	**443**
nginx_tls_protocols	**gateway_nginx_https_protocols**	Protocols that platform gateway will support when handling HTTPS traffic.	Optional	RPM = **[TLSv1.2]**. Container = **[TLSv1.2, TLSv1.3]**.
redis_disable_tls	**gateway_redis_disable_tls**	Controls whether TLS is enabled or disabled for platform gateway Redis. Set this variable to **true** to disable TLS.	Optional	**false**
redis_port	**gateway_redis_port**	Port number for the Redis host for platform gateway.	Optional	**6379**

RPM variable name	Container variable name	Description	Required or optional	Default

	gateway_extra_settings	Defines additional settings for use by platform gateway during installation.
		
		For example:
		
		`gateway_extra_settings:`
		`  - setting: OAUTH2_PROVIDER['ACCESS_TOKEN_EXPIRE_SECONDS']`
		`  value: 600`
			Optional	**[]**

	gateway_nginx_client_max_body_size	Maximum allowed size for data sent to platform gateway through NGINX.	Optional	**5m**
	gateway_nginx_hsts_max_age	Maximum duration (in seconds) that HTTP Strict Transport Security (HSTS) is enforced for platform gateway.	Optional	**63072000**
	gateway_uwsgi_listen_queue_size	Number of requests **uwsgi** will allow in the queue on platform gateway until **uwsgi_processes** can serve them.	Optional	**4096**

B.9. RECEPTOR VARIABLES

Inventory file variables for Receptor.

RPM variable name	Container variable name	Description	Required or optional	Default

RPM variable name	Container variable name	Description	Required or optional	Default
receptor_datadir		The directory where receptor stores its runtime data and local artifacts. The target directory must be accessible to **awx** users. If the target directory is a temporary file system **tmpfs**, ensure it is remounted correctly after a reboot. Failure to do so results in the receptor no longer having a working directory.	Optional	**/tmp/receptor**
receptor_listener_port	**receptor_port**	Port number that receptor listens on for incoming connections from other receptor nodes.	Optional	**27199**
receptor_listener_protocol	**receptor_protocol**	Protocol that receptor will support when handling traffic.	Optional	**tcp**
receptor_log_level	**receptor_log_level**	Controls the verbosity of logging for receptor. Valid options include: **error**, **warning**, **info**, or **debug**.	Optional	**info**
receptor_tls		Controls whether TLS is enabled or disabled for receptor. Set this variable to **false** to disable TLS.	Optional	**true**

RPM variable name	Container variable name	Description	Required or optional	Default
See **node_type** for the RPM equivalent variable.	**receptor_type**	For the **[automationcontroller]** group the two options are: • **receptor_type=control** - The node only runs project and inventory updates, but not regular jobs. • **receptor_type=hybrid** - The node runs everything. For the **[execution_nodes]** group the two options are: • **receptor_type=hop** - The node forwards jobs to an execution node. • **receptor_type=execution** - The node can run jobs.	Optional	For the **[automationcontroller]** group: **hybrid**. For the **[execution_nodes]** group: **execution**.
See **peers** for the RPM equivalent variable	**receptor_peers**	Used to indicate which nodes a specific host connects to. Wherever this variable is defined, an outbound connection to the specific host is established. The value must be a comma-separated list of hostnames. Do not use inventory group names. This is resolved into a set of hosts that is used to construct the **receptor.conf** file. For more information, see Adding execution nodes.	Optional	**[]**
	receptor_disable_signing	Controls whether signing of communications between receptor nodes is enabled or disabled. Set this variable to **true** to disable communication signing.	Optional	**false**

RPM variable name	Container variable name	Description	Required or optional	Default
	receptor_disable_tls	Controls whether TLS is enabled or disabled for receptor. Set this variable to **true** to disable TLS.	Optional	**false**
	receptor_firewall_zone	The firewall zone where receptor related firewall rules are applied. This controls which networks can access receptor based on the zone's trust level.	Optional	**public**
	receptor_mintls13	Controls whether or not receptor only accepts connections that use TLS 1.3 or higher. Set to **true** to only accept connections that use TLS 1.3 or higher.	Optional	**false**
	receptor_signing_private_key	Path to the private key used by receptor to sign communications with other receptor nodes in the network.	Optional	
	receptor_signing_public_key	Path to the public key used by receptor to sign communications with other receptor nodes in the network.	Optional	
	receptor_signing_remote	Denote whether the receptor signing files are local to the installation program (**false**) or on the remote component server (**true**).	Optional	**false**
	receptor_tls_cert	Path to the TLS certificate file for receptor.	Optional	
	receptor_tls_key	Path to the TLS key file for receptor.	Optional	

RPM variable name	Container variable name	Description	Required or optional	Default
	receptor_tls_remote	Denote whether the receptor provided certificate files are local to the installation program (**false**) or on the remote component server (**true**).	Optional	**false**
	receptor_use_archive_compression	Controls whether archive compression is enabled or disabled for receptor. You can control this functionality globally by using **use_archive_compression**.	Optional	**true**

B.10. REDIS VARIABLES

Inventory file variables for Redis.

RPM variable name	Container variable name	Description	Required or optional	Default

RPM variable name	Container variable name	Description	Required or optional	Default
redis_cluster_ip	**redis_cluster_ip**	The IPv4 address used by the Redis cluster to identify each host in the cluster. When defining hosts in the **[redis]** group, use this variable to identify the IPv4 address if the default is not what you want. Specific to container: Redis clusters cannot use hostnames or IPv6 addresses.	Optional	RPM = Discovered IPv4 address from Ansible facts. If IPv4 address is not available, IPv6 address is used. Container = Discovered IPv4 address from Ansible facts.
redis_disable_mtls		Controls whether mTLS is enabled or disabled for Redis. Set this variable to **true** to disable mTLS.	Optional	**false**
redis_firewalld_zone	**redis_firewall_zone**	The firewall zone where Redis related firewall rules are applied. This controls which networks can access Redis based on the zone's trust level.	Optional	RPM = no default set. Container = **public**.

RPM variable name	Container variable name	Description	Required or optional	Default
redis_hostname		Hostname used by the Redis cluster when identifying and routing the host. By default **routable_hostname** is used.	Optional	The value defined in **routable_hostname**
redis_mode	**redis_mode**	The Redis mode to use for your Ansible Automation Platform installation. Valid options include: **standalone** and **cluster**. For more information about Redis, see Caching and queueing system in *Planning your installation*.	Optional	**cluster**
redis_server_regen_cert		Denotes whether or not to regenerate the Ansible Automation Platform managed TLS key pair for Redis.	Optional	**false**
redis_tls_cert	**redis_tls_cert**	Path to the Redis server TLS certificate.	Optional	
redis_tls_files_remote	**redis_tls_remote**	Denote whether the Redis provided certificate files are local to the installation program (**false**) or on the remote component server (**true**).	Optional	**false**
redis_tls_key	**redis_tls_key**	Path to the Redis server TLS certificate key.	Optional	
	redis_use_archive_compression	Controls whether archive compression is enabled or disabled for Redis. You can control this functionality globally by using **use_archive_compression**.	Optional	**true**