Red Hat Ansible Automation Platform 2.5

RPM installation

Install the RPM version of Ansible Automation Platform

Last Updated: 2024-10-16

Red Hat Ansible Automation Platform 2.5 RPM installation

Install the RPM version of Ansible Automation Platform

Legal Notice

Abstract

This guide shows you how to install Red Hat Ansible Automation Platform based on supported installation scenarios.

Table of Contents

PREFACE

Thank you for your interest in Red Hat Ansible Automation Platform. 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 installing Ansible Automation Platform. This document has been updated to include information for the latest release of Ansible Automation Platform.

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. RED HAT ANSIBLE AUTOMATION PLATFORM INSTALLATION OVERVIEW

The Red Hat Ansible Automation Platform installation program offers you flexibility, allowing you to install Ansible Automation Platform by using several supported installation scenarios.

Regardless of the installation scenario you choose, installing Ansible Automation Platform involves the following steps:

Editing the Red Hat Ansible Automation Platform installer inventory file

The Ansible Automation Platform installer inventory file allows you to specify your installation scenario and describe host deployments to Ansible. The examples provided in this document show the parameter specifications needed to install that scenario for your deployment.

Running the Red Hat Ansible Automation Platform installer setup script

The setup script installs Ansible Automation Platform by using the required parameters defined in the inventory file.

Verifying your Ansible Automation Platform installation

After installing Ansible Automation Platform, you can verify that the installation has been successful by logging in to the platform UI and seeing the relevant functionality.

Additional resources

1. For more information about the supported installation scenarios, see the Planning your installation.
2. For more information on available topologies, see Tested deployment models .

1.1. PREREQUISITES

- You chose and obtained a platform installer from the Red Hat Ansible Automation Platform Product Software.
- You are installing on a machine that meets base system requirements.
- You have updated all of the packages to the recent version of your RHEL nodes.

WARNING

To prevent errors, upgrade your RHEL nodes fully before installing Ansible Automation Platform.

- You have created a Red Hat Registry Service Account, by using the instructions in Creating Registry Service Accounts.

Additional resources

For more information about obtaining a platform installer or system requirements, see the System requirements in the *Planning your installation*.

CHAPTER 2. SYSTEM REQUIREMENTS

Use this information when planning your Red Hat Ansible Automation Platform installations and designing automation mesh topologies that fit your use case.

Prerequisites

- You can obtain root access either through the **sudo** command, or through privilege escalation. For more on privilege escalation, see Understanding privilege escalation .
- You can de-escalate privileges from root to users such as: AWX, PostgreSQL, Event-Driven Ansible, or Pulp.
- You have configured an NTP client on all nodes.

2.1. RED HAT ANSIBLE AUTOMATION PLATFORM SYSTEM REQUIREMENTS

Your system must meet the following minimum system requirements to install and run Red Hat Ansible Automation Platform. A resilient deployment requires 10 virtual machines with a minimum of 16 gigabytes(GB) of ram and 4 virtual cpus(vCPU). See, Tested deployment models for more information on topology options.

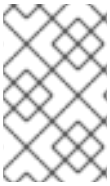
Table 2.1. Base system

Requirement	Required	Notes
Subscription	Valid Red Hat Ansible Automation Platform	
OS	Red Hat Enterprise Linux 8.8 or later (x86_64, aarch64), or Red Hat Enterprise Linux 9.2 or later (x86_64, aarch64)	Red Hat Ansible Automation Platform are also supported on OpenShift, see Installing on OpenShift Container Platform for more information.
Ansible-core	Ansible-core version 2.16 or later	Ansible Automation Platform uses the system-wide ansible-core package to install the platform, but uses ansible-core 2.16 for both its control plane and built-in execution environments.
Database	PostgreSQL version 15	

Table 2.2. Virtual machine requirements

Component	RAM	VCPU	Storage
Platform gateway	16GB	4	20GB minimum

Component	RAM	VCPU	Storage
Control nodes	16GB	4	80GB minimum with at least 20GB available under **/var/lib/awx**
Execution nodes	16GB	4	40GB minimum
Hop nodes	16GB	4	40GB minimum
Automation hub	16GB	4	40GB minimum allocated to **/var/lib/pulp**
Database	16GB	4	100GB minimum allocated to **/var/lib/pgsql**
Event-Driven Ansible controller	16GB	4	40GB minimum



NOTE

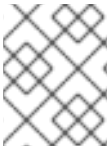
These are minimum requirements and can be increased for larger workloads in increments of 2x (for example 16GB becomes 32GB and 4 vCPU becomes 8vCPU). See the horizontal scaling guide for more information.

The following are necessary for you to work with project updates and collections:

- Ensure that the Network ports and protocols listed in *Table 6.3. Automation Hub* are available for successful connection and download of collections from automation hub or Ansible Galaxy server.

Additional notes for Red Hat Ansible Automation Platform requirements

- If performing a bundled Ansible Automation Platform installation, the installation setup.sh script attempts to install ansible-core (and its dependencies) from the bundle for you.
- If you have installed Ansible-core manually, the Ansible Automation Platform installation setup.sh script detects that Ansible has been installed and does not attempt to reinstall it.



NOTE

You must use Ansible-core, which is installed via dnf. Ansible-core version 2.16 is required for versions 2.5 and later.

2.2. PLATFORM GATEWAY SYSTEM REQUIREMENTS

The platform gateway is the service that handles authentication and authorization for Ansible Automation Platform. It provides a single entry into the platform and serves the platform's user interface.

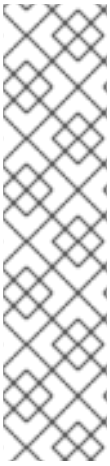
2.3. AUTOMATION CONTROLLER SYSTEM REQUIREMENTS

Automation controller is a distributed system, where different software components can be co-located or deployed across multiple compute nodes. In the installer, four node types are provided as abstractions to help you design the topology appropriate for your use case: control, hybrid, execution, and hop nodes.

Use the following recommendations for node sizing:

Execution nodes

Execution nodes run automation. Increase memory and CPU to increase capacity for running more forks.



NOTE

- The RAM and CPU resources stated are minimum recommendations to handle the job load for a node to run an average number of jobs simultaneously.
- Recommended RAM and CPU node sizes are not supplied. The required RAM or CPU depends directly on the number of jobs you are running in that environment.
- For capacity based on forks in your configuration, see Automation controller capacity determination and job impact.

For further information about required RAM and CPU levels, see Performance tuning for automation controller.

Control nodes

Control nodes process events and run cluster jobs including project updates and cleanup jobs. Increasing CPU and memory can help with job event processing.

- 40GB minimum with at least 20GB available under /var/lib/awx
- Storage volume must be rated for a minimum baseline of 1500 IOPS
- Projects are stored on control and hybrid nodes, and for the duration of jobs, are also stored on execution nodes. If the cluster has many large projects, consider doubling the GB in /var/lib/awx/projects, to avoid disk space errors.

Hop nodes

Hop nodes serve to route traffic from one part of the automation mesh to another (for example, a hop node could be a bastion host into another network). RAM can affect throughput, CPU activity is low. Network bandwidth and latency are generally a more important factor than either RAM or CPU.

- Actual RAM requirements vary based on how many hosts automation controller manages simultaneously (which is controlled by the **forks** parameter in the job template or the system **ansible.cfg** file). To avoid possible resource conflicts, Ansible recommends 1 GB of memory per 10 forks and 2 GB reservation for automation controller. See Automation controller capacity determination and job impact. If **forks** is set to 400, 42 GB of memory is recommended.

- Automation controller hosts check if **umask** is set to 0022. If not, the setup fails. Set **umask=0022** to avoid this error.
- A larger number of hosts can be addressed, but if the fork number is less than the total host count, more passes across the hosts are required. You can avoid these RAM limitations by using any of the following approaches:
  - Use rolling updates.
  - Use the provisioning callback system built into automation controller, where each system requesting configuration enters a queue and is processed as quickly as possible.
  - In cases where automation controller is producing or deploying images such as AMIs.

Additional resources

- For more information about obtaining an automation controller subscription, see Attaching your Red Hat Ansible Automation Platform subscription.
- For questions, contact Ansible support through the Red Hat Customer Portal .

2.4. AUTOMATION HUB SYSTEM REQUIREMENTS

Automation hub allows you to discover and use new certified automation content from Red Hat Ansible and Certified Partners. On Ansible automation hub, you can discover and manage Ansible Collections, which are supported automation content developed by Red Hat and its partners for use cases such as cloud automation, network automation, and security automation.

NOTE

Private automation hub

If you install private automation hub from an internal address, and have a certificate which only encompasses the external address, this can result in an installation which cannot be used as container registry without certificate issues.

To avoid this, use the **automationhub_main_url** inventory variable with a value such as https://pah.example.com linking to the private automation hub node in the installation inventory file.

This adds the external address to **/etc/pulp/settings.py**. This implies that you only want to use the external address.

For information about inventory file variables, see Inventory file variables.

2.4.1. High availability automation hub requirements

Before deploying a high availability (HA) automation hub, ensure that you have a shared filesystem installed in your environment and that you have configured your network storage system, if applicable.

2.4.1.1. Required shared filesystem

A high availability automation hub requires you to have a shared file system, such as NFS, already installed in your environment. Before you run the Red Hat Ansible Automation Platform installer, verify that you installed the **/var/lib/pulp** directory across your cluster as part of the shared file system

installation. The Red Hat Ansible Automation Platform installer returns an error if **/var/lib/pulp** is not detected in one of your nodes, causing your high availability automation hub setup to fail.

If you receive an error stating **/var/lib/pulp** is not detected in one of your nodes, ensure **/var/lib/pulp** is properly mounted in all servers and re-run the installer.

2.4.1.2. Installing firewalld for HA hub deployment

If you intend to install a HA automation hub using a network storage on the automation hub nodes itself, you must first install and use **firewalld** to open the necessary ports as required by your shared storage system before running the Ansible Automation Platform installer.

Install and configure **firewalld** by executing the following commands:

1. Install the **firewalld** daemon:

```
$ dnf install firewalld
```

2. Add your network storage under <service> using the following command:

```
$ firewall-cmd --permanent --add-service=<service>
```

NOTE

For a list of supported services, use the **$ firewall-cmd --get-services** command

3. Reload to apply the configuration:

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

2.5. EVENT-DRIVEN ANSIBLE CONTROLLER SYSTEM REQUIREMENTS

The Event-Driven Ansible controller is a single-node system capable of handling a variable number of long-running processes (such as rulebook activations) on-demand, depending on the number of CPU cores.

[NOTE] If you want to use Event-Driven Ansible 2.5 with a 2.4 automation controller version, see Using Event-Driven Ansible 2.5 with Ansible Automation Platform 2.4.

Use the following minimum requirements to run, by default, a maximum of 12 simultaneous activations:

Requirement	Required
RAM	16 GB
CPUs	4

Requirement	Required
Local disk	• Hard drive must be 40 GB minimum with at least 20 GB available under /var. • Storage volume must be rated for a minimum baseline of 1500 IOPS. • If the cluster has many large projects or decision environment images, consider doubling the GB in /var to avoid disk space errors.

IMPORTANT

- If you are running Red Hat Enterprise Linux 8 and want to set your memory limits, you must have cgroup v2 enabled before you install Event-Driven Ansible. For specific instructions, see the Knowledge-Centered Support (KCS) article, Ansible Automation Platform Event-Driven Ansible controller for Red Hat Enterprise Linux 8 requires cgroupv2.
- When you activate an Event-Driven Ansible rulebook under standard conditions, it uses about 250 MB of memory. However, the actual memory consumption can vary significantly based on the complexity of your rules and the volume and size of the events processed. In scenarios where a large number of events are anticipated or the rulebook complexity is high, conduct a preliminary assessment of resource usage in a staging environment. This ensures that your maximum number of activations is based on the capacity of your resources.

For an example of setting Event-Driven Ansible controller maximumrunning activations, see Single automation controller, single automation hub, and single Event-Driven Ansible controller node with external (installer managed) database.

2.6. POSTGRESQL REQUIREMENTS

Red Hat Ansible Automation Platform uses PostgreSQL 15. PostgreSQL user passwords are hashed with SCRAM-SHA-256 secure hashing algorithm before storing in the database.

To determine if your automation controller instance has access to the database, you can do so with the command, **awx-manage check_db** command.

NOTE

- Automation controller data is stored in the database. Database storage increases with the number of hosts managed, number of jobs run, number of facts stored in the fact cache, and number of tasks in any individual job. For example, a playbook runs every hour (24 times a day) across 250 hosts, with 20 tasks, stores over 800000 events in the database every week.
- If not enough space is reserved in the database, the old job runs and facts must be cleaned on a regular basis. For more information, see Management Jobs in the *Configuring automation execution*.

PostgreSQL Configurations

Optionally, you can configure the PostgreSQL database as separate nodes that are not managed by the Red Hat Ansible Automation Platform installer. When the Ansible Automation Platform installer manages the database server, it configures the server with defaults that are generally recommended for most workloads. For more information about the settings you can use to improve database performance, see Database Settings.

Additional resources

For more information about tuning your PostgreSQL server, see the PostgreSQL documentation.

2.6.1. Setting up an external (customer supported) database

IMPORTANT

Red Hat does not support the use of external (customer supported) databases, however they are used by customers. The following guidance on inital configuration, from a product installation perspective only, is provided to avoid related support requests.

To create a database, user and password on an external PostgreSQL compliant database for use with automation controller, use the following procedure.

Procedure

1. Install and then connect to a PostgreSQL compliant database server with superuser privileges.

```
# psql -h <db.example.com> -U superuser -p 5432 -d postgres <Password for user superuser>:
```

2. Where the default value for <hostname> is **hostname**:

```
-h hostname
--host=hostname
```

3. Specify the hostname of the machine on which the server is running. If the value begins with a slash, it is used as the directory for the UNIX-domain socket.

```
-d dbname
--dbname=dbname
```

4. Specify the name of the database to connect to. This is equal to specifying **dbname** as the first non-option argument on the command line. The **dbname** can be a connection string. If so, connection string parameters override any conflicting command line options.

```
-U username
--username=username
```

5. Connect to the database as the user **username** instead of the default (you must have permission to do so).

6. Create the user, database, and password with the **createDB** or administrator role assigned to the user. For further information, see Database Roles.

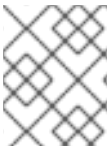
7. Add the database credentials and host details to the automation controller inventory file as an external database. The default values are used in the following example:

```
[database]
 pg_host='db.example.com'
 pg_port=5432
 pg_database='awx'
 pg_username='awx'
 pg_password='redhat'
```

8. Run the installer. If you are using a PostgreSQL database with automation controller, the database is owned by the connecting user and must have a **createDB** or administrator role assigned to it.

9. Check that you are able to connect to the created database with the user, password and database name.

10. Check the permission of the user. The user should have the **createDB** or administrator role.



NOTE

During this procedure, you must check the External Database coverage. For further information, see https://access.redhat.com/articles/4010491

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

Added in Ansible Automation Platform 2.5, the database migration script uses **hstore** fields to store information, therefore the **hstore** extension to the automation hub PostgreSQL database must be enabled.

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 to the automation hub PostreSQL database manually before automation hub installation.

If the **hstore** extension is not enabled before automation hub installation, a failure is raised 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. Create the **hstore** PostgreSQL extension on the automation hub database with the following command:

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

   The output of which is:

```
CREATE EXTENSION
```

   In the following output, the **installed_version** field contains 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)
```

2.6.3. Benchmarking storage performance for the Ansible Automation Platform PostgreSQL database

Check whether the minimum Ansible Automation Platform PostgreSQL database requirements are met by using the Flexible I/O Tester (FIO) tool. FIO is a tool used to benchmark read and write IOPS performance of the storage system.

Prerequisites

- You have installed the Flexible I/O Tester (**fio**) storage performance benchmarking tool.
  To install **fio**, run the following command as the root user:

```
# yum -y install fio
```

- You have adequate disk space to store the **fio** test data log files.
  The examples shown in the procedure require at least 60GB disk space in the **/tmp** directory:
  - **numjobs** sets the number of jobs run by the command.
  - **size=10G** sets the file size generated by each job.

- You have adjusted the value of the **size** parameter. Adjusting this value reduces the amount of test data.

Procedure

1. Run a random write test:

```
$ fio --name=write_iops --directory=/tmp --numjobs=3 --size=10G \
--time_based --runtime=60s --ramp_time=2s --ioengine=libaio --direct=1 \
--verify=0 --bs=4K --iodepth=64 --rw=randwrite \
--group_reporting=1 > /tmp/fio_benchmark_write_iops.log \
2>> /tmp/fio_write_iops_error.log
```

2. Run a random read test:

```
$ fio --name=read_iops --directory=/tmp \
--numjobs=3 --size=10G --time_based --runtime=60s --ramp_time=2s \
--ioengine=libaio --direct=1 --verify=0 --bs=4K --iodepth=64 --rw=randread \
--group_reporting=1 > /tmp/fio_benchmark_read_iops.log \
2>> /tmp/fio_read_iops_error.log
```

3. Review the results:
   In the log files written by the benchmark commands, search for the line beginning with **iops**. This line shows the minimum, maximum, and average values for the test.

   The following example shows the line in the log file for the random read test:

```
$ cat /tmp/fio_benchmark_read_iops.log
read_iops: (g=0): rw=randread, bs=(R) 4096B-4096B, (W) 4096B-4096B, (T) 4096B-4096B,
ioengine=libaio, iodepth=64
[…]
  iops        : min=50879, max=61603, avg=56221.33, stdev=679.97, samples=360
[…]
```

NOTE

The above is a baseline to help evaluate the best case performance on your systems. Systems can and will change and performance may vary depending on what else is happening on your systems, storage or network at the time of testing. You must review, monitor, and revisit the log files according to your own business requirements, application workloads, and new demands.

CHAPTER 3. INSTALLING RED HAT ANSIBLE AUTOMATION PLATFORM

Ansible Automation Platform is a modular platform. The platform gateway deploys automation platform components, such as automation controller, automation hub, and Event-Driven Ansible controller.

For more information about the components provided with Ansible Automation Platform, see Red Hat Ansible Automation Platform components in Planning your installation.

There are several supported installation scenarios for Red Hat Ansible Automation Platform. To install Red Hat Ansible Automation Platform, you must edit the inventory file parameters to specify your installation scenario. You can use the enterprise installer as a basis for your own inventory file.

Additional resources

For a comprehensive list of pre-defined variables used in Ansible installation inventory files, see Ansible variables.

3.1. EDITING THE RED HAT ANSIBLE AUTOMATION PLATFORM INSTALLER INVENTORY FILE

You can use the Red Hat Ansible Automation Platform installer inventory file to specify your installation scenario.

Procedure

1. Navigate to the installer:
   a. [RPM installed package]

   ```
   $ cd /opt/ansible-automation-platform/installer/
   ```

   b. [bundled installer]

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

   c. [online installer]

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

2. Open the **inventory** file with a text editor.
3. Edit **inventory** file parameters to specify your installation scenario. You can use one of the supported Installation scenario examples as the basis for your **inventory** file.

Additional resources

- For a comprehensive list of pre-defined variables used in Ansible installation inventory files, see Inventory file variables.

3.2. INVENTORY FILE EXAMPLES BASED ON INSTALLATION SCENARIOS

Red Hat supports several installation scenarios for Ansible Automation Platform. You can develop your own inventory files using the example files as a basis, or you can use the example closest to your preferred installation scenario.

3.2.1. Inventory file recommendations based on installation scenarios

Before selecting your installation method for Ansible Automation Platform, review the following recommendations. Familiarity with these recommendations will streamline the installation process.

- Provide a reachable IP address or fully qualified domain name (FQDN) for hosts to ensure users can sync and install content from automation hub from a different node.
  The FQDN must not contain either the **-** or the _ symbols, as it will not be processed correctly.

  Do not use **localhost**.

- **admin** is the default user ID for the initial log in to Ansible Automation Platform and cannot be changed in the inventory file.

- Use of special characters for **pg_password** is limited. The **!**, **#**, **0** and **@** characters are supported. Use of other special characters can cause the setup to fail.

- Enter your Red Hat Registry Service Account credentials in **registry_username** and **registry_password** to link to the Red Hat container registry.

- The inventory file variables **registry_username** and **registry_password** are only required if a non-bundle installer is used.

3.2.2. Setting registry_username and registry_password

If you intend to use the **registry_username** and **registry_password** variables in an inventory file you are recommended to use the following method to create a Registry Service Account to set a token with an expiration in the plaintext **inventory/vars.yml** file instead of using a plaintext username and password, for reasons of security.

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

Procedure

1. Navigate 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 accepted characters.
4. Optionally enter a description for the account.
5. Click **Create account**.
6. Find the created account in the list. The list of accounts is long so you might have to click **Next** multiple times before finding the account you created. 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>
7. Click the name of the account that you created.

8. A **token** page opens, displaying a generated Username (different to the account name) and a token.
   If no Username and token are displayed, click **Regenerate token**. You can also click this to generate a new Username and token.

9. Copy the service account name and use it to set **registry_username**.

10. Copy the token and use it to set **registry_password**.

3.2.2.1. Single platform gateway and automation controller with an external (installer managed) database

Use this example to see what is minimally needed within the inventory file to deploy single instances of platform gateway and automation controller with an external (installer managed) database.

```
[automationcontroller]
controller.example.com

[automationgateway]
gateway.example.com

[database]
data.example.com

[all:vars]
admin_password='<password>'
pg_host='data.example.com'
pg_port=5432
pg_database='awx'
pg_username='awx'
pg_password='<password>'
pg_sslmode='prefer' # set to 'verify-full' for client-side enforced SSL

registry_url='registry.redhat.io'
registry_username='<registry username>'
registry_password='<registry password>'

# Automation Gateway configuration
automationgateway_admin_password=''

automationgateway_pg_host='data.example.com'
automationgateway_pg_port=5432

automationgateway_pg_database='automationgateway'
automationgateway_pg_username='automationgateway'
automationgateway_pg_password=''
automationgateway_pg_sslmode='prefer'

# The main automation gateway URL that clients will connect to (e.g. https://<load balancer host>).
# If not specified, the first node in the [automationgateway] group will be used when needed.
# automationgateway_main_url = ''

# Certificate and key to install in Automation Gateway
# automationgateway_ssl_cert=/path/to/automationgateway.cert
# automationgateway_ssl_key=/path/to/automationgateway.key
```

```
# SSL-related variables
# If set, this will install a custom CA certificate to the system trust store.
# custom_ca_cert=/path/to/ca.crt
# Certificate and key to install in nginx for the web UI and API
# web_server_ssl_cert=/path/to/tower.cert
# web_server_ssl_key=/path/to/tower.key
# Server-side SSL settings for PostgreSQL (when we are installing it).
# postgres_use_ssl=False
# postgres_ssl_cert=/path/to/pgsql.crt
# postgres_ssl_key=/path/to/pgsql.key
```

3.2.2.2. Single platform gateway, automation controller, and automation hub with an external (installer managed) database

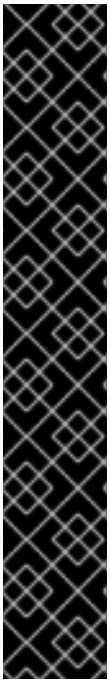
Use this example to populate the inventory file to deploy single instances of platform gateway, automation controller, and automation hub with an external (installer managed) database.

```
[automationcontroller]
controller.example.com

[automationhub]
automationhub.example.com

[automationgateway]
gateway.example.com

[database]
data.example.com

[all:vars]
admin_password='<password>'
pg_host='data.example.com'
pg_port='5432'
pg_database='awx'
pg_username='awx'
pg_password='<password>'
pg_sslmode='prefer'  # set to 'verify-full' for client-side enforced SSL

registry_url='registry.redhat.io'
registry_username='<registry username>'
registry_password='<registry password>'

automationhub_admin_password= <PASSWORD>

automationhub_pg_host='data.example.com'
automationhub_pg_port=5432

automationhub_pg_database='automationhub'
automationhub_pg_username='automationhub'
automationhub_pg_password=<PASSWORD>
automationhub_pg_sslmode='prefer'

# The default install will deploy a TLS enabled Automation Hub.
# If for some reason this is not the behavior wanted one can
# disable TLS enabled deployment.
```

```
#
# automationhub_disable_https = False
# The default install will generate self-signed certificates for the Automation
# Hub service. If you are providing valid certificate via automationhub_ssl_cert
# and automationhub_ssl_key, one should toggle that value to True.
#
# automationhub_ssl_validate_certs = False
# SSL-related variables
# If set, this will install a custom CA certificate to the system trust store.
# custom_ca_cert=/path/to/ca.crt
# Certificate and key to install in Automation Hub node
# automationhub_ssl_cert=/path/to/automationhub.cert
# automationhub_ssl_key=/path/to/automationhub.key

# Automation Gateway configuration
automationgateway_admin_password=''

automationgateway_pg_host=''
automationgateway_pg_port=5432

automationgateway_pg_database='automationgateway'
automationgateway_pg_username='automationgateway'
automationgateway_pg_password=''
automationgateway_pg_sslmode='prefer'

# The main automation gateway URL that clients will connect to (e.g. https://<load balancer host>).
# If not specified, the first node in the [automationgateway] group will be used when needed.
# automationgateway_main_url = ''

# Certificate and key to install in Automation Gateway
# automationgateway_ssl_cert=/path/to/automationgateway.cert
# automationgateway_ssl_key=/path/to/automationgateway.key

# Certificate and key to install in nginx for the web UI and API
# web_server_ssl_cert=/path/to/tower.cert
# web_server_ssl_key=/path/to/tower.key
# Server-side SSL settings for PostgreSQL (when we are installing it).
# postgres_use_ssl=False
# postgres_ssl_cert=/path/to/pgsql.crt
# postgres_ssl_key=/path/to/pgsql.key
```

3.2.2.3. Single platform gateway, automation controller, automation hub, and Event-Driven Ansible controller with an external (installer managed) database

Use this example to populate the inventory file to deploy single instances of platform gateway, automation controller, automation hub, and Event-Driven Ansible controller with an external (installer managed) database.



IMPORTANT

- This scenario requires a minimum of automation controller 2.4 for successful deployment of Event-Driven Ansible controller.
- Event-Driven Ansible controller must be installed on a separate server and cannot be installed on the same host as automation hub and automation controller.
- When you activate an Event-Driven Ansible rulebook under standard conditions, it uses approximately 250 MB of memory. However, the actual memory consumption can vary significantly based on the complexity of your rules and the volume and size of the events processed. In scenarios where a large number of events are anticipated or the rulebook complexity is high, conduct a preliminary assessment of resource usage in a staging environment. This ensures that your maximum number of activations is based on the capacity of your resources. In the following example, the default **automationedacontroller_max_running_activations** setting is 12, but you can adjust according to your capacity.

```
[automationcontroller]
controller.example.com

[automationhub]
automationhub.example.com

[automationedacontroller]
automationedacontroller.example.com

[automationgateway]
gateway.example.com

[database]
data.example.com

[all:vars]
admin_password='<password>'
pg_host='data.example.com'
pg_port='5432'
pg_database='awx'
pg_username='awx'
pg_password='<password>'
pg_sslmode='prefer'  # set to 'verify-full' for client-side enforced SSL

registry_url='registry.redhat.io'
registry_username='<registry username>'
registry_password='<registry password>'

# Automation hub configuration

automationhub_admin_password= <PASSWORD>

automationhub_pg_host='data.example.com'
automationhub_pg_port=5432

automationhub_pg_database='automationhub'
```

```
automationhub_pg_username='automationhub'
automationhub_pg_password=<PASSWORD>
automationhub_pg_sslmode='prefer'

# Automation Event-Driven Ansible controller configuration

automationedacontroller_admin_password='<eda-password>'

automationedacontroller_pg_host='data.example.com'
automationedacontroller_pg_port=5432

automationedacontroller_pg_database='automationedacontroller'
automationedacontroller_pg_username='automationedacontroller'
automationedacontroller_pg_password='<password>'

# Keystore file to install in SSO node
# sso_custom_keystore_file='/path/to/sso.jks'

# This install will deploy SSO with sso_use_https=True
# Keystore password is required for https enabled SSO
sso_keystore_password=''

# This install will deploy a TLS enabled Automation Hub.
# If for some reason this is not the behavior wanted one can
# disable TLS enabled deployment.
#
# automationhub_disable_https = False
# The default install will generate self-signed certificates for the Automation
# Hub service. If you are providing valid certificate via automationhub_ssl_cert
# and automationhub_ssl_key, one should toggle that value to True.
#
# automationhub_ssl_validate_certs = False
# SSL-related variables
# If set, this will install a custom CA certificate to the system trust store.
# custom_ca_cert=/path/to/ca.crt
# Certificate and key to install in Automation Hub node
# automationhub_ssl_cert=/path/to/automationhub.cert
# automationhub_ssl_key=/path/to/automationhub.key

# Automation Gateway configuration
automationgateway_admin_password=''

automationgateway_pg_host=''
automationgateway_pg_port=5432

automationgateway_pg_database='automationgateway'
automationgateway_pg_username='automationgateway'
automationgateway_pg_password=''
automationgateway_pg_sslmode='prefer'

# The main automation gateway URL that clients will connect to (e.g. https://<load balancer host>).
# If not specified, the first node in the [automationgateway] group will be used when needed.
# automationgateway_main_url = ''

# Certificate and key to install in Automation Gateway
# automationgateway_ssl_cert=/path/to/automationgateway.cert
```

```
# automationgateway_ssl_key=/path/to/automationgateway.key

# Certificate and key to install in nginx for the web UI and API
# web_server_ssl_cert=/path/to/tower.cert
# web_server_ssl_key=/path/to/tower.key
# Server-side SSL settings for PostgreSQL (when we are installing it).
# postgres_use_ssl=False
# postgres_ssl_cert=/path/to/pgsql.crt
# postgres_ssl_key=/path/to/pgsql.key

# Boolean flag used to verify Automation Controller's
# web certificates when making calls from Automation Event-Driven Ansible controller.
# automationedacontroller_controller_verify_ssl = true
#
# Certificate and key to install in Automation Event-Driven Ansible controller node
# automationedacontroller_ssl_cert=/path/to/automationeda.crt
# automationedacontroller_ssl_key=/path/to/automationeda.key
```

Additional resources

For more information about these inventory variables, refer to the Ansible automation hub variables .

3.2.2.4. High availability automation hub

Use the following examples to populate the inventory file to install a highly available automation hub. This inventory file includes a highly available automation hub with a clustered setup.

You can configure your HA deployment further to enable a high availability deployment of automation hub on SELinux.

Specify database host IP

- Specify the IP address for your database host, using the **automation_pg_host** and **automation_pg_port** inventory variables. For example:

```
automationhub_pg_host='192.0.2.10'
automationhub_pg_port=5432
```

- Also specify the IP address for your database host in the [database] section, using the value in the **automationhub_pg_host** inventory variable:

```
[database]
192.0.2.10
```

List all instances in a clustered setup

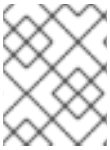
- If installing a clustered setup, replace **localhost ansible_connection=local** in the [automationhub] section with the hostname or IP of all instances. For example:

```
[automationhub]
automationhub1.testing.ansible.com ansible_user=cloud-user
automationhub2.testing.ansible.com ansible_user=cloud-user
automationhub3.testing.ansible.com ansible_user=cloud-user
```

Next steps

Check that the following directives are present in **/etc/pulp/settings.py** in each of the private automation hub servers:

```
USE_X_FORWARDED_PORT = True
USE_X_FORWARDED_HOST = True
```

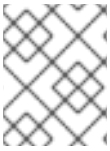


NOTE

If **automationhub_main_url** is not specified, the first node in the [automationhub] group will be used as default.

3.2.2.5. Enabling a high availability (HA) deployment of automation hub on SELinux

You can configure the inventory file to enable high availability deployment of automation hub on SELinux. You must create two mount points for **/var/lib/pulp** and **/var/lib/pulp/pulpcore_static**, and then assign the appropriate SELinux contexts to each.



NOTE

You must add the context for **/var/lib/pulp** pulpcore_static and run the Ansible Automation Platform installer before adding the context for **/var/lib/pulp**.

Prerequisites

- You have already configured a NFS export on your server.

NOTE

The NFS share is hosted on an external server and is not a part of high availability automation hub deployment.

Procedure

1. Create a mount point at **/var/lib/pulp**:

```
$ mkdir /var/lib/pulp/
```

2. Open **/etc/fstab** using a text editor, then add the following values:

```
srv_rhel8:/data /var/lib/pulp nfs
defaults,_netdev,nosharecache,context="system_u:object_r:var_lib_t:s0" 0 0
srv_rhel8:/data/pulpcore_static /var/lib/pulp/pulpcore_static nfs
defaults,_netdev,nosharecache,context="system_u:object_r:httpd_sys_content_rw_t:s0" 0 0
```

3. Run the reload systemd manager configuration command:

```
$ systemctl daemon-reload
```

4. Run the mount command for **/var/lib/pulp**:

```
$ mount /var/lib/pulp
```

5. Create a mount point at **/var/lib/pulp/pulpcore_static**:

```
$ mkdir /var/lib/pulp/pulpcore_static
```

6. Run the mount command:

```
$ mount -a
```

7. With the mount points set up, run the Ansible Automation Platform installer:

```
$ setup.sh -- -b --become-user root
```

8. After the installation is complete, unmount the **/var/lib/pulp/** mount point.

Next steps

1. Apply the appropriate SELinux context .
2. Configure the pulpcore.serivce.

Additional Resources

- See the SELinux Requirements on the Pulp Project documentation for a list of SELinux contexts.
- See the Filesystem Layout for a full description of Pulp folders.

3.2.2.5.1. Configuring pulpcore.service

After you have configured the inventory file, and applied the SELinux context, you now need to configure the pulp service.

Procedure

1. With the two mount points set up, shut down the Pulp service to configure **pulpcore.service**:

```
$ systemctl stop pulpcore.service
```

2. Edit **pulpcore.service** using **systemctl**:

```
$ systemctl edit pulpcore.service
```

3. Add the following entry to **pulpcore.service** to ensure that automation hub services starts only after starting the network and mounting the remote mount points:

```
[Unit]
After=network.target var-lib-pulp.mount
```

4. Enable **remote-fs.target**:

```
$ systemctl enable remote-fs.target
```

5. Reboot the system:

```
$ systemctl reboot
```

Troubleshooting

A bug in the pulpcore SELinux policies can cause the token authentication public/private keys in **etc**/**pulp**/**certs**/ to not have the proper SELinux labels, causing the pulp process to fail. When this occurs, run the following command to temporarily attach the proper labels:

```
$ chcon system_u:object_r:pulpcore_etc_t:s0 /etc/pulp/certs/token_{private,public}_key.pem
```

Repeat this command to reattach the proper SELinux labels whenever you relabel your system.

3.2.2.5.2. Applying the SELinux context

After you have configured the inventory file, you must now apply the context to enable the high availability (HA) deployment of automation hub on SELinux.

Procedure

1. Shut down the Pulp service:

```
$ systemctl stop pulpcore.service
```

2. Unmount /**var**/**lib**/**pulp**/**pulpcore_static**:

```
$ umount /var/lib/pulp/pulpcore_static
```

3. Unmount /**var**/**lib**/**pulp**/:

```
$ umount /var/lib/pulp/
```

4. Open /**etc**/**fstab** using a text editor, then replace the existing value for /**var**/**lib**/**pulp** with the following:

```
srv_rhel8:/data /var/lib/pulp nfs
defaults,_netdev,nosharecache,context="system_u:object_r:pulpcore_var_lib_t:s0" 0 0
```

5. Run the mount command:

```
$ mount -a
```

3.2.2.6. Configuring content signing on private automation hub

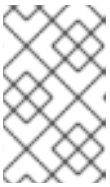
To successfully sign and publish Ansible Certified Content Collections, you must configure private automation hub for signing.

Prerequisites

- Your GnuPG key pairs have been securely set up and managed by your organization.
- Your public-private key pair has proper access for configuring content signing on private automation hub.

Procedure

1. Create a signing script that accepts only a filename.



NOTE

This script acts as the signing service and must generate an ascii-armored detached **gpg** signature for that file using the key specified through the **PULP_SIGNING_KEY_FINGERPRINT** environment variable.

The script prints out a JSON structure with the following format.

```
{"file": "filename", "signature": "filename.asc"}
```

All the file names are relative paths inside the current working directory. The file name must remain the same for the detached signature.

Example:

The following script produces signatures for content:

```
#!/usr/bin/env bash

FILE_PATH=$1
SIGNATURE_PATH="$1.asc"

ADMIN_ID="$PULP_SIGNING_KEY_FINGERPRINT"
PASSWORD="password"

# Create a detached signature
gpg --quiet --batch --pinentry-mode loopback --yes --passphrase \
  $PASSWORD --homedir ~/.gnupg/ --detach-sign --default-key $ADMIN_ID \
  --armor --output $SIGNATURE_PATH $FILE_PATH

# Check the exit status
STATUS=$?
if [ $STATUS -eq 0 ]; then
  echo {\"file\": \"$FILE_PATH\", \"signature\": \"$SIGNATURE_PATH\"}
else
  exit $STATUS
fi
```

After you deploy a private automation hub with signing enabled to your Ansible Automation Platform cluster, new UI additions are displayed in collections.

2. Review the Ansible Automation Platform installer inventory file for options that begin with **automationhub_***.

```
[all:vars]
```

```
.
.
.
automationhub_create_default_collection_signing_service = True
automationhub_auto_sign_collections = True
automationhub_require_content_approval = True
automationhub_collection_signing_service_key = /abs/path/to/galaxy_signing_service.gpg
automationhub_collection_signing_service_script = /abs/path/to/collection_signing.sh
```

The two new keys (**automationhub_auto_sign_collections** and **automationhub_require_content_approval**) indicate that the collections must be signed and approved after they are uploaded to private automation hub.

3.2.2.7. 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/automationedacontroller**
2. Create a file within that directory for your new setting (for example, **touch ./group_vars/automationedacontroller/custom.yml**)
3. Add the variable **automationedacontroller_additional_settings** to extend the default **settings.yaml** template for Event-Driven Ansible controller and add the **SAFE_PLUGINS** field with a list of plugins to enable. For example:

```
automationedacontroller_additional_settings:
  SAFE_PLUGINS:
    - ansible.eda.webhook
    - ansible.eda.alertmanager
```

NOTE

You can also extend the **automationedacontroller_additional_settings** variable beyond SAFE_PLUGINS in the Django configuration file, /etc/ansible-automation-platform/eda/settings.yaml

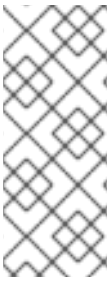
3.3. RUNNING THE RED HAT ANSIBLE AUTOMATION PLATFORM INSTALLER SETUP SCRIPT

After you update the inventory file with required parameters, run the installer setup script.

Procedure

- Run the **setup.sh** script

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



NOTE

If you are running the setup as a non-root user with **sudo** privileges, you can use the following command:

```
$ ANSIBLE_BECOME_METHOD='sudo'
ANSIBLE_BECOME=True ./setup.sh
```

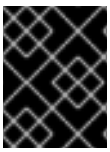
Installation of Red Hat Ansible Automation Platform will begin.

Additional resources

See Understanding privilege escalation for additional **setup.sh** script examples.

3.4. VERIFYING INSTALLATION OF ANSIBLE AUTOMATION PLATFORM

Upon a successful login, your installation of Red Hat Ansible Automation Platform is complete.



IMPORTANT

If the installation fails and you are a customer who has purchased a valid license for Red Hat Ansible Automation Platform, contact Ansible through the Red Hat Customer portal.

Additional resources

See Getting started with Ansible Automation Platform for post installation instructions.

3.5. ADDING A SUBSCRIPTION MANIFEST TO ANSIBLE AUTOMATION PLATFORM

Before you first log in, you must add your subscription information to the platform. To add a subscription to Ansible Automation Platform, see Obtaining a manifest file in the Access management and authentication.

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 would set up an inventory file for horizontal scaling of Event-Driven Ansible controller on Red Hat Enterprise Linux VMs using the host group name **[automationedacontroller]** and the node type variable **eda_node_type**:

```
[automationedacontroller]

3.88.116.111
routable_hostname=automationedacontroller-api.example.com eda_node_type=api

# worker node
3.88.116.112 routable_hostname=automationedacontroller-api.example.com eda_node_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 desired number of users of the application and the number of worker nodes correlates to the desired 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.

To set up a multi-node deployment, follow the procedure in Setting up horizontal scaling for Event-Driven Ansible controller.

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 to add or remove nodes and rerun the installer.

Procedure

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

```
[automationedacontroller]

3.88.116.111 routable_hostname=automationedacontroller-api.example.com
eda_node_type=api

3.88.116.112 routable_hostname=automationedacontroller-api.example.com
eda_node_type=worker

# two more worker nodes
3.88.116.113 routable_hostname=automationedacontroller-api.example.com
eda_node_type=worker

3.88.116.114 routable_hostname=automationedacontroller-api.example.com
eda_node_type=worker
```

2. Re-run the installer.

CHAPTER 5. DISCONNECTED INSTALLATION

If you are not connected to the internet or do not have access to online repositories, you can install Red Hat Ansible Automation Platform without an active internet connection.

5.1. PREREQUISITES

Before installing Ansible Automation Platform on a disconnected network, you must meet the following prerequisites:

- A subscription manifest that you can upload to the platform.
  For more information, see Obtaining a manifest file .
- The Ansible Automation Platform setup bundle at Customer Portal is downloaded.
- The DNS records for the automation controller and private automation hub servers are created.

5.2. ANSIBLE AUTOMATION PLATFORM INSTALLATION ON DISCONNECTED RHEL

You can install Ansible Automation Platform without an internet connection by using the installer-managed database located on the automation controller. The setup bundle is recommended for disconnected installation because it includes additional components that make installing Ansible Automation Platform easier in a disconnected environment. These include the Ansible Automation Platform Red Hat package managers (RPMs) and the default execution environment (EE) images.

Additional Resources

For a comprehensive list of pre-defined variables used in Ansible installation inventory files, see Ansible variables.

5.2.1. System requirements for disconnected installation

Ensure that your system has all the hardware requirements before performing a disconnected installation of Ansible Automation Platform. You can find these in system requirements.

5.2.2. RPM Source

RPM dependencies for Ansible Automation Platform that come from the BaseOS and AppStream repositories are not included in the setup bundle. To add these dependencies, you must first obtain access to BaseOS and AppStream repositories. Use Satellite to sync repositories and add dependencies. If you prefer an alternative tool, you can choose between the following options:

- Reposync
- The RHEL Binary DVD

NOTE

The RHEL Binary DVD method requires the DVD for supported versions of RHEL. See Red Hat Enterprise Linux Life Cycle for information on which versions of RHEL are currently supported.

Additional resources

- Satellite

5.3. SYNCHRONIZING RPM REPOSITORIES USING REPOSYNC

To perform a reposync you need a RHEL host that has access to the internet. After the repositories are synced, you can move the repositories to the disconnected network hosted from a web server.

When downloading RPM, ensure you use the applicable distro.

Procedure

1. Attach the BaseOS and AppStream required repositories:

```
# subscription-manager repos \
    --enable rhel-9-for-x86_64-baseos-rpms \
    --enable rhel-9-for-x86_64-appstream-rpms
```

2. Perform the reposync:

```
# dnf install yum-utils
# reposync -m --download-metadata --gpgcheck \
    -p /path/to/download
```

   i. Use reposync with **--download-metadata** and without **--newest-only**. See RHEL 8 Reposync.

   - If you are not using **--newest-only,** the repos downloaded may take an extended amount of time to sync due to the large number of GB.
   - If you are using **--newest-only,** the repos downloaded may take an extended amount of time to sync due to the large number of GB.

   After the reposync is completed, your repositories are ready to use with a web server.

3. Move the repositories to your disconnected network.

5.4. CREATING A NEW WEB SERVER TO HOST REPOSITORIES

If you do not have an existing web server to host your repositories, you can create one with your synced repositories.

Procedure

1. Install prerequisites:

```
$ sudo dnf install httpd
```

2. Configure httpd to serve the repo directory:

```
/etc/httpd/conf.d/repository.conf

DocumentRoot '/path/to/repos'
```

```
<LocationMatch "^/+$">
    Options -Indexes
    ErrorDocument 403 /.noindex.html
</LocationMatch>

<Directory '/path/to/repos'>
    Options All Indexes FollowSymLinks
    AllowOverride None
    Require all granted
</Directory>
```

3. Ensure that the directory is readable by an apache user:

```
$ sudo chown -R apache /path/to/repos
```

4. Configure SELinux:

```
$ sudo semanage fcontext -a -t httpd_sys_content_t "/path/to/repos(/.*)?"
$ sudo restorecon -ir /path/to/repos
```

5. Enable httpd:

```
$ sudo systemctl enable --now httpd.service
```

6. Open firewall:

```
$ sudo firewall-cmd --zone=public --add-service=http –add-service=https --permanent
$ sudo firewall-cmd --reload
```

7. On automation services, add a repo file at */etc/yum.repos.d/local.repo*, and add the optional repos if needed:

```
[Local-BaseOS]
name=Local BaseOS
baseurl=http://<webserver_fqdn>/rhel-8-for-x86_64-baseos-rpms
enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release

[Local-AppStream]
name=Local AppStream
baseurl=http://<webserver_fqdn>/rhel-8-for-x86_64-appstream-rpms
enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release
```

5.5. ACCESSING RPM REPOSITORIES FROM A LOCALLY MOUNTED DVD

If you plan to access the repositories from the RHEL binary DVD, you must first set up a local repository.

Procedure

1. Mount DVD or ISO:
   a. DVD

```
# mkdir /media/rheldvd && mount /dev/sr0 /media/rheldvd
```

   b. ISO

```
# mkdir /media/rheldvd && mount -o loop rhrhel-8.6-x86_64-dvd.iso /media/rheldvd
```

2. Create yum repo file at **/etc/yum.repos.d/dvd.repo**

```
[dvd-BaseOS]
name=DVD for RHEL - BaseOS
baseurl=file:///media/rheldvd/BaseOS
enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release

[dvd-AppStream]
name=DVD for RHEL - AppStream
baseurl=file:///media/rheldvd/AppStream
enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release
```

3. Import the gpg key:

```
# rpm --import /media/rheldvd/RPM-GPG-KEY-redhat-release
```

NOTE

If the key is not imported you will see an error similar to

```
# Curl error (6): Couldn't resolve host name for
https://www.redhat.com/security/data/fd431d51.txt [Could not resolve host:
www.redhat.com]
```

Additional Resources

For further detail on setting up a repository see Need to set up yum repository for locally-mounted DVD on Red Hat Enterprise Linux 8.

5.6. DOWNLOADING AND INSTALLING THE ANSIBLE AUTOMATION PLATFORM SETUP BUNDLE

Choose the setup bundle to download Ansible Automation Platform for disconnected installations. This bundle includes the RPM content for Ansible Automation Platform and the default execution environment images that will be uploaded to your private automation hub during the installation process.

Procedure

1. Download the Ansible Automation Platform setup bundle package by navigating to the Red Hat Ansible Automation Platform download page and clicking **Download Now** for the Ansible Automation Platform 2.5 Setup Bundle.

2. On control node, untar the bundle:

```
$ tar xvf \
  ansible-automation-platform-setup-bundle-2.5-1.tar.gz
$ cd ansible-automation-platform-setup-bundle-2.5-1
```

3. Edit the inventory file to include variables based on your host names and desired password values.

NOTE

See section 3.2 Inventory file examples base on installation scenarios for a list of examples that best fits your scenario.

5.7. COMPLETING POST INSTALLATION TASKS

After you have completed the installation of Ansible Automation Platform, ensure that automation hub and automation controller deploy properly.

Before your first login, you must add your subscription information to the platform. To obtain your subscription information in uploadable form, see Obtaining a manifest file in *Access management and authentication*.

Once you have obtained your subscription manifest, see Getting started with Ansible Automation Platform for instructions on how to upload your subscription information.

Now that you have successfully installed Ansible Automation Platform, to begin using its features, see the following guides for your next steps:

Getting started with Ansible Automation Platform.

Managing automation content.

Creating and using execution environments.

APPENDIX A. 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.

A.1. GENERAL VARIABLES

RPM variable name	Container variable name	Description
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. Default = **true**
nginx_tls_protocols		Defines support for **ssl_protocols** in NGINX. Values available **TLSv1**, **TLSv1.1**, **TLSv1.2**, **TLSv1.3**. The TLSv1.1 and TLSv1.2 parameters only work when OpenSSL 1.0.1 or higher is used. The TLSv1.3 parameter only works when OpenSSL 1.1.1 or higher is used. If **nginx_tls-protocols = ['TLSv1.3']** only TLSv1.3 is enabled. To set more than one protocol use **nginx_tls_protocols = ['TLSv1.2', 'TLSv.1.3']**. Default = **TLSv1.2**
nginx_user_http_config		List of NGINX configurations for **/etc/nginx/nginx.conf** under the http section. Each element in the list is provided into **http nginx config** as a separate line. Default = empty list

RPM variable name	Container variable name	Description
registry_password	**registry_password**	This variable is only required if a non-bundle installer is used. Password credential for access to **registry_url**. Enter your Red Hat Registry Service Account credentials in **registry_username** and **registry_password** to link to the Red Hat container registry. When **registry_url** is **registry.redhat.io**, username and password are required if not using a bundle installer. For more information, see Setting registry_username and registry_password.
registry_url	**registry_url**	URL for the registry source. Default = **registry.redhat.io**
registry_username	**registry_username**	This variable is only required if a non-bundle installer is used. User credential for access to **registry_url**. Enter your Red Hat Registry Service Account credentials in **registry_username** and **registry_password** to link to the Red Hat container registry. For more information, see Setting registry_username and registry_password.

RPM variable name	Container variable name	Description
routable_hostname	**routable_hostname**	This variable is used if the machine running the installer 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 installer can only resolve that host by using a FQDN. If **routable_hostname** is not set, it should default 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.
	bundle_dir	The path to the bundle directory. Default = **false**
	bundle_install	Use offline installation. Set to **true** to enable offline installation. Default = **false**
	ca_tls_cert	TLS CA certificate.
	ca_tls_key	TLS CA key.
	ca_tls_remote	TLS CA remote files. Default = **false**
	container_compress	Container compression software. Default = **gzip**
	container_keep_images	Keep container images. Default = **false**

RPM variable name	Container variable name	Description
	container_pull_images	Pull newer container images. Default = **true**
	custom_ca_cert	Custom TLS CA certificate.
	registry_auth	Use registry authentication. Default = **true**
	registry_ns_aap	Ansible Automation Platform registry namespace. Default = **ansible-automation-platform-25**
	registry_ns_rhel	RHEL registry namespace. Default = **rhel8**
	registry_tls_verify	Verify registry TLS. Default = **true**

A.2. AUTOMATION HUB VARIABLES

RPM variable name	Container variable name	Description
automationhub_admin_password	**hub_admin_password**	*Required* Required passwords must be enclosed in quotes when they are provided in plain text in the inventory file. Use of special characters for this variable is limited. The password can include any printable ASCII character except /, ", or @.

RPM variable name	Container variable name	Description
automationhub_api_token		This variable can be used to provide the installer with an existing token. For example, a regenerated token in Hub UI will invalidate an existing token. Use **automationhub_api_token** to use that token in the installer the next time you run the installer.
automationhub_auto_sign_collections	**hub_collection_auto_sign**	If a collection signing service is enabled, collections are not signed automatically by default. Setting this parameter to **true** signs them by default. Default = **false**
automationhub_backup_collections		*Optional* Ansible automation hub provides artifacts in **/var/lib/pulp**. Automation controller automatically backs up the artifacts by default. You can also set **automationhub_backup_collections** to **false** and the backup and restore process will not backup or restore **/var/lib/pulp**. Default = **true**
automationhub_collection_download_count		*Optional* Determines whether download count is displayed on the UI. Default = **false**

RPM variable name	Container variable name	Description
automationhub_collection_seed_repository		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, both certified and validated content are uploaded. Possible values of this variable are **certified** or **validated**. If you do not want to install content, set **automationhub_seed_collections** to **false** to disable the seeding. If you only want one type of content, set **automationhub_seed_collections** to **true** and **automationhub_collection_seed_repository** to the type of content you do want to include.
automationhub_collection_signing_service_key	**hub_collection_signing_key**	If a collection signing service is enabled, you must provide this variable to ensure that collections can be properly signed. **/absolute/path/to/key/to/sign**
automationhub_collection_signing_service_script		If a collection signing service is enabled, you must provide this variable to ensure that collections can be properly signed. **/absolute/path/to/script/that/signs**
automationhub_container_signing_service_key	**hub_container_signing_key**	If a container signing service is enabled, you must provide this variable to ensure that containers can be properly signed. **/absolute/path/to/key/to/sign**

RPM variable name	Container variable name	Description
automationhub_container_signing_service_script		If a container signing service is enabled, you must provide this variable to ensure that containers can be properly signed. **/absolute/path/to/script/that/signs**
automationhub_create_default_collection_signing_service	**hub_collection_signing_service**	Set this variable to **true** to create a collection signing service. Default = **false**
automationhub_create_default_container_signing_service	**hub_container_signing_service**	Set this variable to **true** to create a container signing service. Default = **false**
automationhub_disable_hsts	**hub_nginx_disable_hsts**	The default installation deploys a TLS enabled automation hub. Use this variable if you deploy automation hub with *HTTP Strict Transport Security* (HSTS) web-security policy enabled. This variable disables the HSTS web-security policy mechanism. Default = **false**
automationhub_disable_https	**hub_nginx_disable_https**	*Optional* If automation hub is deployed with HTTPS enabled. Default = **false**
automationhub_enable_analytics		A Boolean indicating whether to enable pulp analytics for the version of **pulpcore** used in automation hub in Ansible Automation Platform 2.5. To enable pulp analytics, set **automationhub_enable_analytics** to **true**. Default = **false**

RPM variable name	Container variable name	Description
automationhub_enable_api_access_log		When set to **true**, this variable creates a log file at **/var/log/galaxy_api_access.log** that logs all user actions made to the platform, including their username and IP address. Default = **false**
automationhub_enable_unauthenticated_collection_access		Set this variable to **true** to enable unauthorized users to view collections. Default = **false**
automationhub_enable_unauthenticated_collection_download		Set this variable to **true** to enable unauthorized users to download collections. Default = **false**
automationhub_importer_settings	**hub_galaxy_importer**	*Optional* Dictionary of setting to pass to galaxy-importer. At import time, collections can go through a series of checks. Behavior is driven by **galaxy-importer.cfg** configuration. Examples are **ansible-doc**, **ansible-lint**, and **flake8**. This parameter enables you to drive this configuration.
automationhub_pg_database	**hub_pg_database**	The PostgreSQL database name. RPM default = **automationhub** Container default = **pulp**
automationhub_pg_host	**hub_pg_host**	Required if not using an internal database. The hostname of the remote PostgreSQL database used by automation hub. Default = **127.0.0.1**

RPM variable name	Container variable name	Description
automationhub_pg_password	**hub_pg_password**	Required if not using an internal database. The password for the automation hub PostgreSQL database. 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.
automationhub_pg_port	**hub_pg_port**	Required if not using an internal database. Default = **5432**
automationhub_pg_sslmode		*Required* Default = **prefer**
automationhub_pg_username	**hub_pg_username**	The username for your automation hub PostgreSQL database RPM default = **automationhub** Container default = **pulp**
automationhub_require_content_approval		*Optional* Value is **true** if automation hub enforces the approval mechanism before collections are made available. By default when you upload collections to automation hub, an administrator must approve it before they are made available to the users. If you want to disable the content approval flow, set the variable to **false**. Default = **true**

RPM variable name	Container variable name	Description
automationhub_seed_collections		A Boolean that defines whether or not pre-loading 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, both certified and validated content are uploaded. If you do not want to install content, set **automationhub_seed_collections** to **false** to disable the seeding. If you only want one type of content, set **automationhub_seed_collections** to **true** and **automationhub_collection_seed_repository** to the type of content you do want to include. Default = **true**
automationhub_ssl_cert	**hub_tls_cert**	*Optional* **/path/to/automationhub.cert** Same as **web_server_ssl_cert** but for automation hub UI and API.
automationhub_ssl_key	**hub_tls_key**	*Optional* **/path/to/automationhub.key**. Same as **web_server_ssl_key** but for automation hub UI and API.

RPM variable name	Container variable name	Description
automationhub_user_headers		List of nginx headers for Ansible automation hub's web server. Each element in the list is provided to the web server's nginx configuration as a separate line. Default = empty list
ee_from_hub_only		When deployed with automation hub, the installer pushes execution environment images to automation hub and configures automation controller to pull images from the automation hub registry. To make automation hub the only registry to pull execution environment images from, set this variable to **true**. If set to **false**, execution environment images are also taken directly from Red Hat. Default = **true** when the bundle installer is used.
generate_automationhub_token		When performing a fresh installation, a new token will automatically be generated by default. If you want the installer to regenerate a new token, set **generate_automationhub_token=true** and the installer will use it in the installation process.
nginx_hsts_max_age	**hub_nginx_hsts_max_age**	This variable specifies how long, in seconds, the system should be considered as an *HTTP Strict Transport Security* (HSTS) host. That is, how long HTTPS is used exclusively for communication. Default = **63072000** seconds, or two years.

RPM variable name	Container variable name	Description
pulp_db_fields_key		Relative or absolute path to the Fernet symmetric encryption key that you want to import. The path is on the Ansible management node. It is used to encrypt certain fields in the database, such as credentials. If not specified, a new key will be generated.
	hub_tls_remote	Automation hub TLS remote files. Default = **false**
	hub_main_url	Automation hub main URL.
	hub_nginx_client_max_body_size	NGINX maximum body size. Default = **20m**
	hub_nginx_http_port	NGINX HTTP port. Default = **8081**
	hub_nginx_https_port	NGINX HTTPS port. Default = **8444**
	hub_nginx_https_protocols	NGINX HTTPS protocols. Default = **[TLSv1.2, TLSv1.3]**
	hub_pg_socket	PostgreSQL Automation hub UNIX socket.
	hub_secret_key	Automation hub secret key.
	hub_storage_backend	Automation hub storage backend.
	hub_workers	Automation hub workers count.
	hub_collection_signing	Enable Automation hub collection signing. Default = **false**
	hub_container_signing	Enable Automation hub container signing. Default = **false**

RPM variable name	Container variable name	Description
	hub_container_signing_pass	Automation hub container signing passphrase.
	hub_collection_signing_pass	Automation hub collection signing passphrase.
	hub_postinstall	Enable Automation hub postinstall. Default = **false**
	hub_postinstall_async_delay	Postinstall delay between retries. Default = **1**
	hub_postinstall_async_retries	Postinstall number of retries to perform. Default = **30**
	hub_postinstall_dir	Automation hub postinstall directory.
	hub_postinstall_ignore_files	Automation hub ignore files.
	hub_postinstall_repo_ref	Automation hub repository branch or tag. Default = **main**
	hub_postinstall_repo_url	Automation hub repository URL.

A.3. AUTOMATION CONTROLLER VARIABLES

RPM variable name	Container variable name	Description
admin_email		The email address used for the admin user for automation controller.

RPM variable name	Container variable name	Description
admin_password	**controller_admin_password**	*Required* Automation controller admin password. Passwords must be enclosed in quotes when they are provided in plain text in the inventory file. Use of special characters for this variable is limited. The password can include any printable ASCII character except /, ", or @.
admin_username	**controller_admin_user**	Automation controller admin user. Default = **admin**
automation_controller_main_url		Automation controller main URL.
controller_tls_files_remote	**controller_tls_remote**	Automation controller TLS remote files. Default = **false**
nginx_disable_hsts	**controller_nginx_disable_hsts**	Disable NGINX HTTP Strict Transport Security (HSTS). Default = **false**
nginx_disable_https	**controller_nginx_disable_https**	Disable NGINX HTTPS. Default = **false**
nginx_hsts_max_age	**controller_nginx_hsts_max_age**	This variable specifies how long, in seconds, the system must be considered as an *HTTP Strict Transport Security* (HSTS) host. That is, how long HTTPS is used only for communication. Default = **63072000** seconds, or two years.
nginx_http_port	**controller_nginx_http_port**	The NGINX HTTP server listens for inbound connections. RPM default = **80** Container default = **8080**

RPM variable name	Container variable name	Description
nginx_https_port	**controller_nginx_https_port**	The NGINX HTTPS server listens for secure connections. RPM Default = **443** Container default = **8443**
nginx_user_headers	**controller_nginx_user_headers**	List of NGINX headers for the automation controller web server. Each element in the list is provided to the web server's NGINX configuration as a separate line. Default = empty list
node_state		*Optional* The status of a node or group of nodes. Valid options are **active**, **deprovision** to remove a node from a cluster, or **iso_migrate** to migrate a legacy isolated node to an execution node. Default = **active**

RPM variable name	Container variable name	Description
node_type		For **[automationcontroller]** group. Two valid **node_types** can be assigned for this group. A **node_type=control** means that the node only runs project and inventory updates, but not regular jobs. A **node_type=hybrid** can run everything. Default for this group = **hybrid**. For **[execution_nodes]** group: Two valid **node_types** can be assigned for this group. A **node_type=hop** implies that the node forwards jobs to an execution node. A **node_type=execution** implies that the node can run jobs. Default for this group = **execution**.
peers		*Optional* The **peers** variable is 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 is used to add **tcp-peer** entries in the **receptor.conf** file used for establishing network connections with other nodes. The peers 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.

RPM variable name	Container variable name	Description
pg_database	**controller_pg_database**	The name of the PostgreSQL database. Default = **awx**
pg_host	**controller_pg_host**	*Required* The PostgreSQL host, which can be an externally managed database.
pg_password	**controller_pg_password**	*Required* The password for the PostgreSQL database. 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. NOTE You no longer have to provide a **pg_hashed_password** in your inventory file at the time of installation, because PostgreSQL 13 can now store user passwords more securely. When you supply **pg_password** in the inventory file for the installer, PostgreSQL uses the SCRAM-SHA-256 hash to secure that password as part of the installation process.
pg_port	**controller_pg_port**	The PostgreSQL port to use. Default = **5432**
pg_username	**controller_pg_username**	Your PostgreSQL database username. Default = **awx**.

RPM variable name	Container variable name	Description
supervisor_start_retry_count		When specified, it adds **startretries = <value specified>** to the supervisor config file (/**etc/supervisord.d/tower.ini**). See program:x Section Values for more information about **startretries**. No default value exists.
web_server_ssl_cert	**controller_tls_cert**	*Optional* **/path/to/webserver.cert** Same as **automationhub_ssl_cert** but for web server UI and API.
web_server_ssl_key	**controller_tls_key**	*Optional* **/path/to/webserver.key** Same as **automationhub_server_ssl_key** but for web server UI and API.
	controller_event_workers	Automation controller event workers. Default = **4**
	controller_license_file	The location of your automation controller license file. For example: **controller_license_file=/path/to/license.zip** If you are defining this variable as part of the postinstall process (**controller_postinstall = true**), then you need to also set the **controller_postinstall_dir** variable.
	controller_nginx_client_max_body_size	NGINX maximum body size. Default = **5m**

RPM variable name	Container variable name	Description
	controller_nginx_https_protocols	NGINX HTTPS protocols. Default = **[TLSv1.2, TLSv1.3]**
	controller_pg_socket	PostgreSQL Controller UNIX socket.
	controller_secret_key	Automation controller secret key.
	controller_uwsgi_listen_queue_size	Automation controller uWSGI listen queue size. Default = **2048**
	controller_postinstall	Enable or disable the postinstall feature of the containerized installer. If set to **true**, then you also need to set **controller_license_file** and **controller_postinstall_dir**. Default = **false**
	controller_postinstall_dir	The location of your automation controller postinstall directory.
	controller_postinstall_async_delay	Postinstall delay between retries. Default = **1**
	controller_postinstall_async_retries	Postinstall number of tries to attempt. Default = **30**
	controller_postinstall_ignore_files	Automation controller ignore files.
	controller_postinstall_repo_ref	Automation controller repository branch or tag. Default = **main**
	controller_postinstall_repo_url	Automation controller repository URL.

A.4. EVENT-DRIVEN ANSIBLE CONTROLLER VARIABLES

RPM variable name	Container variable name	Description
automationedacontroller_activation_workers	**eda_activation_workers**	*Optional* Number of workers for ansible-rulebook activation pods in Event-Driven Ansible. Default = (# of cores or threads) * 2 + 1
automationedacontroller_admin_email	**eda_admin_email**	*Optional* Email address used by Django for the admin user for Event-Driven Ansible controller. Default = **admin@example.com**
automationedacontroller_admin_password	**eda_admin_password**	*Required* The admin password used by the Event-Driven Ansible controller instance. Passwords must be enclosed in quotes when they are provided in plain text in the **inventory** file. Use of special characters for this variable is limited. The password can include any printable ASCII character except /, **"**, or **@**.
automationedacontroller_admin_username	**eda_admin_user**	Username used by Django to identify and create the admin superuser in Event-Driven Ansible controller. Default = **admin**
automationedacontroller_allowed_hostnames		List of additional addresses to enable for user access to Event-Driven Ansible controller. Default = empty list

RPM variable name	Container variable name	Description
automationedacontroller_controller_verify_ssl		Boolean flag used to verify automation controller's web certificates when making calls from Event-Driven Ansible controller. Verified is **true** and not verified is **false**. Default = **false**
automationedacontroller_disable_hsts	**eda_nginx_disable_hsts**	*Optional* Boolean flag to disable HSTS for Event-Driven Ansible controller. Default = **false**
automationedacontroller_disable_https	**eda_nginx_disable_https**	*Optional* Boolean flag to disable HTTPS for Event-Driven Ansible controller. Default = **false**
automationedacontroller_event_stream_path	**eda_event_stream_prefix_path**	API prefix path used for Event-Driven Ansible event-stream through platform gateway. Default = **/eda-event-streams**
automationedacontroller_gunicorn_workers	**eda_gunicorn_workers**	Number of workers for the API served through Gunicorn. Default = (# of cores or threads) * 2 + 1
automationedacontroller_max_running_activations	**eda_max_running_activations**	*Optional* The number of maximum activations running concurrently per node. This is an integer that must be greater than 0. Default = **12**
automationedacontroller_nginx_tls_files_remote	**eda_tls_remote**	Boolean flag to specify whether cert sources are on the remote host (true) or local (false). Default = **false**

RPM variable name	Container variable name	Description
automationedacontroller_pg_database	**eda_pg_database**	*Optional* The PostgreSQL database used by Event-Driven Ansible controller. RPM default = **automationedacontroller** Container default = **eda**
automationedacontroller_pg_password	**eda_pg_password**	*Required* The password for the PostgreSQL database used by Event-Driven Ansible controller. 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.
automationedacontroller_pg_port	**eda_pg_port**	*Optional* The port number of the PostgreSQL database used by Event-Driven Ansible controller. Default = **5432**
automationedacontroller_pg_username	**eda_pg_username**	*Optional* The username for your Event-Driven Ansible controller PostgreSQL database. RPM default = **automationedacontroller** Container default = **eda**
automationedacontroller_redis_host	**eda_redis_host**	The Redis hostname used by Event-Driven Ansible controller.
automationedacontroller_redis_port	**eda_redis_port**	The port used for the Redis host defined by **automationedacontroller_redis_host** for Event-Driven Ansible controller.

RPM variable name	Container variable name	Description
automationedacontroller_rq_workers		Number of Redis Queue (RQ) workers used by Event-Driven Ansible controller. RQ workers are Python processes that run in the background. Default = (# of cores or threads) * 2 + 1
automationedacontroller_ssl_cert	**eda_tls_cert**	*Optional* **/root/ssl_certs/eda.<example>.com.crt** Same as **automationhub_ssl_cert** but for Event-Driven Ansible controller UI and API.
automationedacontroller_ssl_key	**eda_tls_key**	*Optional* **/root/ssl_certs/eda.<example>.com.key** Same as **automationhub_server_ssl_key** but for Event-Driven Ansible controller UI and API.
automationedacontroller_user_headers	**eda_nginx_user_headers**	List of additional NGINX headers to add to Event-Driven Ansible controller's NGINX configuration. Default = empty list
automationnedacontroller_pg_host	**eda_pg_host**	*Required* The hostname of the PostgreSQL database used by Event-Driven Ansible controller, which can be an externally managed database.
eda_node_type	**eda_type**	*Optional* Event-Driven Ansible controller node type. Default = **hybrid**

RPM variable name	Container variable name	Description
	eda_debug	Event-Driven Ansible controller debug. Default = **false**
	eda_event_stream_url	Event-Driven Ansible controller event stream URL.
	eda_main_url	Event-Driven Ansible controller main URL.
	eda_nginx_client_max_body_size	NGINX maximum body size. Default = **1m**
	eda_nginx_hsts_max_age	NGINX HSTS maximum age. Default = **63072000**
	eda_nginx_http_port	NGINX HTTP port. Default = **8082**
	eda_nginx_https_port	NGINX HTTPS port. Default = **8445**
	eda_nginx_https_protocols	NGINX HTTPS protocols. Default = **[TLSv1.2, TLSv1.3]**
	eda_pg_socket	PostgreSQL Event-Driven Ansible UNIX socket.
	eda_redis_disable_tls	Disable TLS Redis (for many nodes). Default = **false**
	eda_redis_password	Redis Event-Driven Ansible controller password (for many nodes).
	eda_redis_tls_cert	Event-Driven Ansible controller Redis TLS certificate.
	eda_redis_tls_key	Event-Driven Ansible controller Redis TLS key.

RPM variable name	Container variable name	Description
	eda_redis_username	Redis Event-Driven Ansible controller username (for many nodes).
	eda_safe_plugins	Event-Driven Ansible controller safe plugins.
	eda_secret_key	Event-Driven Ansible controller secret key.
	eda_workers	Event-Driven Ansible controller workers count. Default = **2**

A.5. PLATFORM GATEWAY VARIABLES

RPM variable name	Container variable name	Description
automationgateway_admin_email	**gateway_admin_email**	The email address used for the admin user for platform gateway.
automationgateway_admin_password	**gateway_admin_password**	*Required* The admin password used to connect to the platform gateway instance. Passwords must be enclosed in quotes when they are provided in plain text in the **inventory** file. Use of special characters for this variable is limited. The password can include any printable ASCII character except /, ", or @.
automationgateway_admin_username	**gateway_admin_user**	*Optional* The username used to identify and create the admin superuser in platform gateway. Default = **admin**

RPM variable name	Container variable name	Description
automationgateway_disable_hsts	**gateway_nginx_disable_hsts**	*Optional* Disable NGINX HSTS. Default = **false**
automationgateway_disable_https	**gateway_nginx_disable_https**	*Optional* Disable NGINX HTTPS. Default = **false**
automationgateway_grpc_auth_service_timeout	**gateway_grpc_auth_service_timeout**	Platform gateway auth server timeout. Default = **30s**
automationgateway_grpc_server_max_threads_per_process	**gateway_grpc_server_max_threads_per_process**	Platform gateway auth server threads per process. Default = **10**
automationgateway_grpc_server_processes	**gateway_grpc_server_processes**	Platform gateway auth server processes Default = **5**
automationgateway_main_url	**gateway_main_url**	*Optional* The main platform gateway URL that clients will connect to (e.g. **https://<gateway_node>**). If not specified, the first the first node in the **[automationgateway]** group will be used when needed.
automationgateway_pg_database	**gateway_pg_database**	*Optional* The PostgreSQL database used by platform gateway. RPM default = **automationgateway** Container default = **gateway**

RPM variable name	Container variable name	Description
automationgateway_pg_host	**gateway_pg_host**	*Required* The hostname of the PostgreSQL database used by platform gateway, which can be an externally managed database.
automationgateway_pg_password	**gateway_pg_password**	*Required* The password for the PostgreSQL database used by platform gateway. Use of special characters for **automationgateway_pg_password** is limited. The **!**, **#**, **0** and **@** characters are supported. Use of other special characters can cause the setup to fail.
automationgateway_pg_port	**gateway_pg_port**	*Optional* The port number of the PostgreSQL database used by platform gateway. Default = **5432**
automationgateway_pg_sslmode	**gateway_pg_sslmode**	Choose one of the two available modes: **prefer** and **verify-full**. Set to **verify-full** for client-side enforced SSL. Default = **prefer**
automationgateway_pg_username	**gateway_pg_username**	*Optional* The username for your platform gateway PostgreSQL database. RPM default = **automationgateway** Container default = **gateway**
automationgateway_redis_host	**gateway_redis_host**	The Redis hostname used by platform gateway.

RPM variable name	Container variable name	Description
automationgateway_redis_port	**gateway_redis_port**	The Redis platform gateway port. Default = **6379**
automationgateway_ssl_cert	**gateway_tls_cert**	*Optional* **/path/to/automationgateway.cert** Same as **automationhub_ssl_cert** but for platform gateway UI and API.
automationgateway_ssl_key	**gateway_tls_key**	*Optional* **/path/to/automationgateway.key** Same as **automationhub_server_ssl_key** but for platform gateway UI and API.
	gateway_nginx_client_max_body_size	NGINX maximum body size. Default = **5m**
	gateway_nginx_hsts_max_age	NGINX HSTS maximum age. Default = **63072000**
	gateway_nginx_http_port	NGINX HTTP port.
	gateway_nginx_https_port	NGINX HTTPS port.
	gateway_nginx_https_protocols	NGINX HTTPS protocols. Default = **[TLSv1.2, TLSv1.3]**
	gateway_nginx_user_headers	Custom NGINX headers.
	gateway_redis_disable_tls	Disable TLS Redis. Default = **false**
	gateway_redis_password	Redis platform gateway password.

RPM variable name	Container variable name	Description
	gateway_redis_tls_cert	Platform gateway Redis TLS certificate.
	gateway_redis_tls_key	Platform gateway Redis TLS key.
	gateway_redis_username	Redis platform gateway username. Default = **gateway**
	gateway_secret_key	Platform gateway secret key.
	gateway_tls_remote	Platform gateway TLS remote files. Default = **false**
	gateway_uwsgi_listen_queue_size	Platform gateway uWSGI listen queue size. Default = **4096**

A.6. DATABASE VARIABLES

RPM variable name	Container variable name	Description
pg_ssl_mode		Choose one of the two available modes: **prefer** and **verify-full**. Set to **verify-full** for client-side enforced SSL. Default = **prefer**
postgres_ssl_cert	**postgresql_tls_cert**	Location of the PostgreSQL SSL/TLS certificate. **/path/to/pgsql_ssl.cert**
postgres_ssl_key	**postgresql_tls_key**	Location of the PostgreSQL SSL/TLS key. **/path/to/pgsql_ssl.key**
postgres_use_cert		Location of the PostgreSQL user certificate. **/path/to/pgsql.crt**

RPM variable name	Container variable name	Description
postgres_use_ssl	**postgresql_disable_tls**	Determines if the connection between Ansible Automation Platform and the PostgreSQL database should use SSL/TLS. The default for this variable is **false** which means SSL/TLS is not used for PostgreSQL connections. When set to **true**, the platform connects to PostgreSQL by using SSL/TLS.
postgres_max_connections	**postgresql_max_connections**	Maximum database connections setting to apply if you are using installer-managed PostgreSQL. See PostgreSQL database configuration and maintenance for automation controller for help selecting a value. Default = **1024**
	postgresql_admin_database	PostgreSQL admin database. Default = **postgres**
	postgresql_admin_username	PostgreSQL admin user. Default = **postgres**
	postgresql_admin_password	*Required* PostgreSQL admin password.
	postgresql_effective_cache_size	PostgreSQL effective cache size.
	postgresql_keep_databases	Keep databases during uninstall. Default = **false**
	postgresql_log_destination	PostgreSQL log file location. Default = **/dev/stderr**
	postgresql_password_encryption	PostgreSQL password encryption. Default = **scram-sha-256**

RPM variable name	Container variable name	Description
	postgresql_shared_buffers	PostgreSQL shared buffers.
	postgresql_tls_remote	PostgreSQL TLS remote files. Default = **false**
	postgresql_port	PostgreSQL port number. Default = **5432**

A.7. IMAGE VARIABLES

RPM variable name	Container variable name	Description
	controller_image	Automation controller image. Default = **controller-rhel8:latest**
	de_extra_images	Decision environment extra images.
	de_supported_image	Decision environment supported image. Default = **de-supported-rhel8:latest**
	eda_image	Event-Driven Ansible image. Default = **eda-controller-rhel8:latest**
	eda_web_image	Event-Driven Ansible web image. Default = **eda-controller-ui-rhel8:latest**
	ee_29_enabled	Enable execution environment 29. Default = **false**
	ee_29_image	Execution environment 29 image. Default = **ee-29-rhel8:latest**

RPM variable name	Container variable name	Description
	ee_extra_images	Execution environment extra images.
	ee_minimal_image	Execution environment minimal image. Default = **ee-minimal-rhel8:latest**
	ee_supported_image	Execution environment supported image. Default = **ee-supported-rhel8:latest**
	hub_image	Automation hub image. Default = **hub-rhel8:latest**
	hub_web_image	Automation hub web image. Default = **hub-web-rhel8:latest**
	postgresql_image	PostgreSQL image. Default = **postgresql-15:latest**
	receptor_image	Receptor image. Default = **receptor-rhel8:latest**
	redis_image	Redis image. Default = **redis-6:latest**
	pcp_image	Performance Co-Pilot image. Default = **rhel8-pcp:latest**

A.8. RECEPTOR VARIABLES

RPM variable name	Container variable name	Description
	receptor_disable_signing	Disable receptor signing. Default = **false**

RPM variable name	Container variable name	Description
	receptor_disable_tls	Disable receptor TLS. Default = **false**
	receptor_log_level	Receptor logging level. Default = **info**
	receptor_mintls13	Receptor TLS 1.3 minimal. Default = **false**
	receptor_peers	Receptor peers list.
	receptor_port	Receptor port. Default = **27199**
	receptor_protocol	Receptor protocol. Default = **tcp**
	receptor_signing_private_key	Receptor signing private key.
	receptor_signing_public_key	Receptor signing public key.
	receptor_signing_remote	Receptor signing remote files. Default = **false**
	receptor_tls_cert	Receptor TLS certificate.
	receptor_tls_key	Receptor TLS key.
	receptor_tls_remote	Receptor TLS remote files. Default = **false**
	receptor_type	Receptor node type. Default = **execution**

A.9. ANSIBLE VARIABLES

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

For more information about variables specific to certain plugins, see the documentation for Ansible.Builtin.

For a list of global configuration options, see Ansible Configuration Settings.

Variable	Description
ansible_connection	The connection plugin used for the task on the target host. This can be the name of any of Ansible connection plugins. SSH protocol types are **smart**, **ssh** or **paramiko**. Default = **smart**
ansible_host	The IP or name of the target host to use instead of **inventory_hostname**.
ansible_port	The connection port number. Default: 22 for ssh
ansible_user	The user name to use when connecting to the host.
ansible_password	The password to authenticate to the host. Never store this variable in plain text. Always use a vault.
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. Useful if using multiple keys and you do not want to use an SSH agent.
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_sftp_extra_args	This setting is always appended to the default **sftp** command line.
ansible_scp_extra_args	This setting is always appended to the default **scp** command line.
ansible_ssh_extra_args	This setting is always appended to the default **ssh** command line.

Variable	Description
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_executable	Added in version 2.2. This setting overrides the default behavior to use the system SSH. This can override the ssh_executable setting in **ansible.cfg**.
ansible_ssh_executable	Added in version 2.2. This setting overrides the default behavior to use the system SSH. This can override the ssh_executable setting in 'ansible.cfg'.
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_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**. 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.