Red Hat Ansible Automation Platform

2.4

Red Hat Ansible Automation Platform Installation Guide

Install Ansible Automation Platform
Install Ansible Automation Platform
Abstract

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

**PREFACE**  ................................................................. 4

**MAKING OPEN SOURCE MORE INCLUSIVE**  ......................................... 5

**PROVIDING FEEDBACK ON RED HAT DOCUMENTATION**  ................................. 6

**CHAPTER 1. RED HAT ANSIBLE AUTOMATION PLATFORM INSTALLATION OVERVIEW**  .................... 7

  1.1. PREREQUISITES  ......................................................... 7

**CHAPTER 2. INSTALLING RED HAT ANSIBLE AUTOMATION PLATFORM**  ................. 9

  2.1. EDITING THE RED HAT ANSIBLE AUTOMATION PLATFORM INSTALLER INVENTORY FILE  9

    2.1.1. Inventory file examples based on installation scenarios .......................... 10

      2.1.1.1. Standalone automation controller with internal database ................. 10

      2.1.1.2. Single automation controller with external (installer managed) database 11

      2.1.1.3. Single automation controller with external (customer provided) database 12

      2.1.1.4. Ansible Automation Platform with an external (installer managed) database 12

      2.1.1.5. Ansible Automation Platform with an external (customer provided) database 15

      2.1.1.6. Single Event-Driven Ansible controller node with internal database .......... 18

      2.1.1.7. Standalone automation hub with internal database ........................... 18

      2.1.1.8. Single automation hub with external (installer managed) database .......... 19

      2.1.1.9. Single automation hub with external (customer provided) database .......... 20

      2.1.1.10. LDAP configuration on private automation hub ............................. 21

      2.1.1.10.1. Setting up your inventory file variables ................................. 21

      2.1.1.10.2. Configuring extra LDAP parameters ......................................... 22

  2.2. RUNNING THE RED HAT ANSIBLE AUTOMATION PLATFORM INSTALLER SETUP SCRIPT 24

  2.3. VERIFYING INSTALLATION OF AUTOMATION CONTROLLER ........................... 24

    2.3.1. Additional automation controller configuration and resources ............... 25

  2.4. VERIFYING INSTALLATION OF AUTOMATION HUB .................................. 25

    2.4.1. Additional automation hub configuration and resources ....................... 25

  2.5. VERIFYING EVENT-DRIVEN ANSIBLE CONTROLLER INSTALLATION ................... 26

  2.6. POST-INSTALLATION STEPS ................................................................ 26

    2.6.1. Migrating data to Ansible Automation Platform 2.4 ......................... 26

    2.6.1.1. Migrating from legacy virtual environments (venvs) to automation execution environments 26

    2.6.1.2. Migrating to Ansible Engine 2.9 images using Ansible Builder ........... 27

    2.6.1.3. Migrating to Ansible Core 2.13 .................................................... 27

    2.6.2. Updating execution environment image locations ................................ 27

    2.6.3. Scale up your automation using automation mesh ............................ 28

**CHAPTER 3. DISCONNECTED INSTALLATION**  ........................................ 29

  3.1. ANSIBLE AUTOMATION PLATFORM INSTALLATION ON DISCONNECTED RHEL .... 29

    3.1.1. Prerequisites .............................................................................. 29

    3.1.2. System Requirements ............................................................... 29

    3.1.3. RPM Source ............................................................................. 29

  3.2. SYNCHRONIZING RPM REPOSITORIES BY USING REPOSYNC ..................... 29

  3.3. CREATING A NEW WEB SERVER TO HOST REPOSITORIES ....................... 30

  3.4. ACCESSING RPM REPOSITORIES FOR LOCALLY MOUNTED DVD ................... 31

  3.5. ADDING A SUBSCRIPTION MANIFEST TO AAP WITHOUT AN INTERNET CONNECTION .... 32

  3.6. INSTALLING THE AAP SETUP BUNDLE ............................................ 33

    3.6.1. Downloading the Setup Bundle ................................................... 33

    3.6.1.1. Installing the Setup Bundle ..................................................... 33

  3.7. COMPLETING POST INSTALLATION TASKS ......................................... 34

    3.7.1. Adding Controller Subscription .................................................. 35
3.7.2. Updating the CA trust store
3.7.2.1. Self-Signed Certificates
3.7.2.2. Copying the root certificate on the private automation hub to the automation controller using secure copy (SCP)
3.7.2.3. Copying and Pasting

3.8. IMPORTING COLLECTIONS INTO PRIVATE AUTOMATION HUB
3.8.1. Downloading collection from Red Hat Automation Hub

3.9. CREATING COLLECTION NAMESPACE
3.9.1. Importing the collection tarball with GUI
3.9.1.1. Importing the collection tarball using ansible-galaxy via CLI

3.10. APPROVING THE IMPORTED COLLECTION
3.10.1. Custom Execution Environments
3.10.1.1. Transferring a Custom EE Images Across a Disconnected Boundary

3.11. BUILDING AN EXECUTION ENVIRONMENT IN A DISCONNECTED ENVIRONMENT
3.12. INSTALLING THE ANSIBLE-BUILDER RPM
3.12.1. Workflow for upgrading between minor AAP releases

APPENDIX A. INVENTORY FILE VARIABLES
A.1. GENERAL VARIABLES
A.2. ANSIBLE AUTOMATION HUB VARIABLES
A.3. AUTOMATION CONTROLLER VARIABLES
A.4. ANSIBLE VARIABLES
A.5. EVENT-DRIVEN ANSIBLE CONTROLLER VARIABLES
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.
MAKING OPEN SOURCE MORE INCLUSIVE

Red Hat is committed to replacing problematic language in our code, documentation, and web properties. We are beginning with these four terms: master, slave, blacklist, and whitelist. Because of the enormity of this endeavor, these changes will be implemented gradually over several upcoming releases. For more details, see our CTO Chris Wright’s message.
PROVIDING FEEDBACK ON RED HAT DOCUMENTATION

We appreciate your feedback on our technical content and encourage you to tell us what you think. If you’d like to add comments, provide insights, correct a typo, or even ask a question, you can do so directly in the documentation.

**NOTE**

You must have a Red Hat account and be logged in to the customer portal.

To submit documentation feedback from the customer portal, do the following:

1. Select the **Multi-page HTML** format.

2. Click the **Feedback** button at the top-right of the document.

3. Highlight the section of text where you want to provide feedback.

4. Click the **Add Feedback** dialog next to your highlighted text.

5. Enter your feedback in the text box on the right of the page and then click **Submit**.

We automatically create a tracking issue each time you submit feedback. Open the link that is displayed after you click **Submit** and start watching the issue or add more comments.
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 using a number of supported installation scenarios. Starting with Ansible Automation Platform 2.4, the installation scenarios include the optional deployment of Event-Driven Ansible controller, which introduces the automated resolution of IT requests.

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 your Private Automation Hub using the required parameters defined in the inventory file.

Verifying automation controller installation

After installing Ansible Automation Platform, you can verify that the installation has been successful by logging in to the automation controller.

Verifying automation hub installation

After installing Ansible Automation Platform, you can verify that the installation has been successful by logging in to the automation hub.

Verifying Event-Driven Ansible controller installation

After installing Ansible Automation Platform, you can verify that the installation has been successful by logging in to the Event-Driven Ansible controller.

Post-installation steps

After successful installation, you can begin using the features of Ansible Automation Platform.

Additional resources

For more information about the supported installation scenarios, see the Red Hat Ansible Automation Platform Planning Guide.

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

You may experience errors if you do not fully upgrade your RHEL nodes prior to your Ansible Automation Platform installation.

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

Additional resources

For more information about obtaining a platform installer or system requirements, refer to the Red Hat Ansible Automation Platform system requirements in the Red Hat Ansible Automation Platform Planning Guide.
CHAPTER 2. INSTALLING RED HAT ANSIBLE AUTOMATION PLATFORM

Ansible Automation Platform is a modular platform and you can deploy automation controller with other automation platform components, such as 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 the Red Hat Ansible Automation Platform Planning Guide.

IMPORTANT

With the Ansible Automation Platform installer, you can deploy only one automation hub per inventory. You can use the Ansible Automation Platform installer for a standalone instance of automation hub and run the installer any number of times with any number of different inventories to deploy multiple automation hub nodes.

There are a number of 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 using one of the following examples:

- Standalone automation controller with internal database
- Single automation controller with external (installer managed) database
- Single automation controller with external (customer provided) database
- Ansible Automation Platform with an external (installer managed) database
- Ansible Automation Platform with an external (customer provided) database
- Single Event-Driven Ansible controller node with internal database
- Standalone automation hub with internal database
- Single automation controller with external (installer managed) database
- Single automation controller with external (customer provided) database
- LDAP configuration on private automation hub

2.1. 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>

2. Open the **inventory** file with a text editor.

3. Edit **inventory** file parameters to specify your installation scenario. Use one of the supported **Installation scenario examples** to update your **inventory** file.

**Additional resources**

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

### 2.1.1. Inventory file examples based on installation scenarios

Red Hat supports a number of installation scenarios for Ansible Automation Platform. Review the following examples and choose those suitable for your preferred installation scenario.

**IMPORTANT**

- For Red Hat Ansible Automation Platform or automation hub: Add an automation hub host in the [automationhub] group.

- For internal databases: [database] cannot be used to point to another host in the Ansible Automation Platform cluster. The database host set to be installed by the installer needs to be a unique host.

- Do not install automation controller and automation hub on the same node for versions of Ansible Automation Platform in a production or customer environment. This can cause contention issues and heavy resource use.

- Provide a reachable IP address or fully qualified domain name (FDQN) for the [automationhub] and [automationcontroller] hosts to ensure users can sync and install content from automation hub from a different node. Do not use 'localhost'.

- Do not use special characters for `pg_password`. It 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.

#### 2.1.1.1. Standalone automation controller with internal database

Use this example to populate the inventory file to install Red Hat Ansible Automation Platform. This installation inventory file includes a single automation controller node with an internal database.

```
[automationcontroller]
controller.example.com
```
2.1.1.2. Single automation controller with external (installer managed) database

Use this example to populate the inventory file to install Red Hat Ansible Automation Platform. This installation inventory file includes a single automation controller node with an external database on a separate node.

```
[automationcontroller]
controller.example.com

[data]
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>'

# 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
```
2.1.1.3. Single automation controller with external (customer provided) database

Use this example to populate the inventory file to install Red Hat Ansible Automation Platform. This installation inventory file includes a single automation controller node with an external database on a separate node that is not managed by the platform installer.

IMPORTANT

This example does not have a host under the database group. This indicates to the installer that the database already exists, and is being managed elsewhere.

```
[automationcontroller]
controller.example.com

[database]

[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>'

# 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
```

2.1.1.4. Ansible Automation Platform with an external (installer managed) database

Use this example to populate the inventory file to install Ansible Automation Platform. This installation inventory file includes two automation controller nodes, two execution nodes, automation hub with an external managed database, and Event-Driven Ansible controller.
# Automation Controller Nodes
# There are two valid node_types that can be assigned for this group.
# A node_type=control implies that the node will only be able to run
# project and inventory updates, but not regular jobs.
# A node_type=hybrid will have the ability to run everything.
# If you do not define the node_type, it defaults to hybrid.
#
# control.example node_type=control
# hybrid.example node_type=hybrid
# hybrid2.example <- this will default to hybrid

[automationcontroller]
controller1.example.com node_type=control
ccontroller2.example.com node_type=control

# Execution Nodes
# There are two valid node_types that can be assigned for this group.
# A node_type=hop implies that the node will forward jobs to an execution node.
# A node_type=execution implies that the node will be able to run jobs.
# If you do not define the node_type, it defaults to execution.
#
# hop.example node_type=hop
# execution.example node_type=execution
# execution2.example <- this will default to execution

[execution_nodes]
execution1.example.com node_type=execution
execution2.example.com node_type=execution

[automationhub]
automationhub.example.com
[automationedacontroller]
eda.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>'
pq_sslmode='prefer' # set to 'verify-full' for client-side enforced SSL

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

# Receptor Configuration
#
receptor_listener_port=27199

# 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'

# 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 nginx for the web UI and API
# web_server_ssl_cert=/path/to/tower.cert
# web_server_ssl_key=/path/to/tower.key
# Certificate and key to install in Automation Hub node
# automationhub_ssl_cert=/path/to/automationhub.cert
# automationhub_ssl_key=/path/to/automationhub.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

# Automation EDA Controller Configuration
#
automatedacontroller_admin_password='<eda-password>'

automatedacontroller_pg_host='data.example.com'
automatedacontroller_pg_port=5432

automatedacontroller_pg_database='automatedacontroller'
automatedacontroller_pg_username='automatedacontroller'
automatedacontroller_pg_password='<password>'

# The full routable URL used by EDA to connect to a controller host.
# This URL is required if there is no Automation Controller configured
# in inventory.
#
# automation_controller_main_url="

# Boolean flag used to verify Automation Controller's
# web certificates when making calls from Automation EDA Controller.
#
# automatedacontroller_controller_verify_ssl = true

# 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

# Certificate and key to install in Automation Hub node
# automationhub_ssl_cert=/path/to/automationhub.cert
# automationhub_ssl_key=/path/to/automationhub.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

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

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

2.1.1.5. Ansible Automation Platform with an external (customer provided) database

Use this example to populate the inventory file to install Red Hat Ansible Automation Platform. This installation inventory file includes one of each node type (control, hybrid, hop, and execution), automation hub with an external managed database that is not managed by the platform installer, and Event-Driven Ansible controller.

**IMPORTANT**

This example does not have a host under the database group. This indicates to the installer that the database already exists, and is being managed elsewhere.

# Automation Controller Nodes
# There are two valid node_types that can be assigned for this group.
# A node_type=control implies that the node will only be able to run
# project and inventory updates, but not regular jobs.
# A node_type=hybrid will have the ability to run everything.
# If you do not define the node_type, it defaults to hybrid.
#
# control.example node_type=control
# hybrid.example node_type=hybrid
# hybrid2.example <- this will default to hybrid

[automationcontroller]
hybrid1.example.com node_type=hybrid
controller1.example.com node_type=control

# Execution Nodes
# There are two valid node_types that can be assigned for this group.
# A node_type=hop implies that the node will forward jobs to an execution node.
# A node_type=execution implies that the node will be able to run jobs.


# If you do not define the node_type, it defaults to execution.
#
# hop.example node_type=hop
# execution.example node_type=execution
# execution2.example <- this will default to execution

[execution_nodes]
hop1.example.com node_type=hop
execution1.example.com node_type=execution

[automationhub]
automationhub.example.com
[automationedacontroller]
eda.example.com
[database]

[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>\'

# Receptor Configuration
#
# receptor_listener_port=27199

# 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\'

# 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 nginx for the web UI and API
# web_server_ssl_cert=/path/to/tower.cert
# web_server_ssl_key=/path/to/tower.key
# Certificate and key to install in Automation Hub node
# automationhub_ssl_cert=/path/to/automationhub.cert
# automationhub_ssl_key=/path/to/automationhub.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

# Automation EDA 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>'

# The full routable URL used by EDA to connect to a controller host.
# This URL is required if there is no Automation Controller configured
# in inventory.
#
# automation_controller_main_url=""

# Boolean flag used to verify Automation Controller's
# web certificates when making calls from Automation EDA Controller.
#
# automationedacontroller_controller_verify_ssl = true

# SSL-related variables

# If set, this installs 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

# Certificate and key to install in Automation Hub node
# automationhub_ssl_cert=/path/to/automationhub.cert
# automationhub_ssl_key=/path/to/automationhub.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

# Keystore file to install in SSO node
# sso_custom_keystore_file=’/path/to/sso.jks’
# The default install will deploy SSO with sso_use_https=True
# Keystore password is required for https enabled SSO
# sso_keystore_password=

2.1.1.6. Single Event-Driven Ansible controller node with internal database

Use this example to populate the inventory file to install Event-Driven Ansible controller. This installation inventory file includes a single Event-Driven Ansible controller node with an internal database.

**IMPORTANT**

Automation controller must be installed before you populate the inventory file with the following Event-Driven Ansible variables.

```
# Automation EDA Controller Configuration
#
automationedacontroller_admin_password='<eda-password>'
automationedacontroller_pg_host="
automationedacontroller_pg_port=5432
automationedacontroller_pg_database='automationedacontroller'
automationedacontroller_pg_username='automationedacontroller'
automationedacontroller_pg_password='<password>'

# The full routable URL used by EDA to connect to a controller host.
# This URL is required if there is no Automation Controller configured
# in inventory.
# automation_controller_main_url = 'https://controller.example.com/

# Boolean flag used to verify Automation Controller's
# web certificates when making calls from Automation EDA Controller.
# automationedacontroller_controller_verify_ssl = true
```

Additional resources

- For more details on Event-Driven Ansible, see the Getting Started with Event-Driven Ansible Guide.

2.1.1.7. Standalone automation hub with internal database

Use this example to populate the inventory file to deploy a standalone instance of automation hub with an internal database.

```
[automationcontroller]

[automationhub]
automationhub.example.com ansible_connection=local
```
[all:vars]
registry_url='registry.redhat.io'
registry_username='<registry username>'
registry_password='<registry password>'

automationhub_admin_password= <PASSWORD>

automationhub_pg_host="'
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

2.1.1.8. Single automation hub with external (installer managed) database

Use this example to populate the inventory file to deploy a single instance of automation hub with an external (installer managed) database.

[automationcontroller]
[automationhub]
automationhub.example.com

[database]
data.example.com

[all:vars]
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

2.1.1.9. Single automation hub with external (customer provided) database

Use this example to populate the inventory file to deploy a single instance of automation hub with an
external database that is not managed by the platform installer.

IMPORTANT

This example does not have a host under the database group. This indicates to the
installer that the database already exists, and is being managed elsewhere.

[automationcontroller]
[automationhub]
automationhub.example.com

[database]

[all:vars]
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

2.1.10. LDAP configuration on private automation hub

You must set the following six variables in your Red Hat Ansible Automation Platform installer inventory file to configure your private automation hub for LDAP authentication:

- `automationhub_authentication_backend`
- `automationhub_ldap_server_uri`
- `automationhub_ldap_bind_dn`
- `automationhub_ldap_bind_password`
- `automationhub_ldap_user_search_base_dn`
- `automationhub_ldap_group_search_base_dn`

If any of these variables are missing, the Ansible Automation installer will not complete the installation.

2.1.10.1. Setting up your inventory file variables

When you configure your private automation hub with LDAP authentication, you must set the proper variables in your inventory files during the installation process.

Procedure

1. Access your inventory file according to the procedure in Editing the Red Hat Ansible Automation Platform installer inventory file.

2. Use the following example as a guide to set up your Ansible Automation Platform inventory file:

   ```
   automationhub_authentication_backend = "ldap"

   automationhub_ldap_server_uri = "ldap://ldap:389"  # (for LDAPs use
   automationhub_ldap_server_uri = "ldaps://ldap-server-fqdn")
   automationhub_ldap_bind_dn = "cn=admin,dc=ansible,dc=com"
   automationhub_ldap_bind_password = "GoodNewsEveryone"
   automationhub_ldap_user_search_base_dn = "ou=people,dc=ansible,dc=com"
   automationhub_ldap_group_search_base_dn = "ou=people,dc=ansible,dc=com"
   ```
The following variables will be set with default values, unless you set them with other options.

```yaml
auth_ldap_user_search_scope= 'SUBTREE'
auth_ldap_user_search_filter= '(uid=%(user)s)'
auth_ldap_group_search_scope= 'SUBTREE'
auth_ldap_group_search_filter= '(objectClass=Group)'
auth_ldap_group_type_class= 'django_auth_ldap.config:GroupOfNamesType'
```

3. If you plan to set up extra parameters in your private automation hub (such as user groups, super user access, mirroring, or others), proceed to the next section.

### 2.1.10.2. Configuring extra LDAP parameters

If you plan to set up super user access, user groups, mirroring or other extra parameters, you can create a YAML file that comprises them in your `ldap_extra_settings` dictionary.

**Procedure**

1. Create a YAML file that will contain `ldap_extra_settings`, such as the following:

   ```yaml
   # ldapextras.yml
   ---
   ldap_extra_settings:
     AUTH_LDAP_USER_ATTR_MAP: '{"first_name": "givenName", "last_name": "sn", "email": "mail"}
   ...
   
   Then run `setup.sh -e @ldapextras.yml` during private automation hub installation.

2. Use this example to set up a superuser flag based on membership in an LDAP group.

   ```yaml
   # ldapextras.yml
   ---
   ldap_extra_settings:
     AUTH_LDAP_USER_FLAGS_BY_GROUP: "{"is_superuser": "cn=pah-admins,ou=groups,dc=example,dc=com"}"
   ...
   
   Then run `setup.sh -e @ldapextras.yml` during private automation hub installation.

3. Use this example to set up superuser access.

   ```yaml
   # ldapextras.yml
   ---
   ldap_extra_settings:
     AUTH_LDAP_USER_FLAGS_BY_GROUP: "{"is_superuser": "cn=pah-admins,ou=groups,dc=example,dc=com"}"
   ...
   
   Then run `setup.sh -e @ldapextras.yml` during private automation hub installation.

4. Use this example to mirror all LDAP groups you belong to.
Then run `setup.sh -e @ldapextras.yml` during private automation hub installation.

5. Use this example to map LDAP user attributes (such as first name, last name, and email address of the user).

```yaml
#ldapextras.yml
---
ldap_extra_settings:
  AUTH_LDAP_USER_ATTR_MAP: {
    "first_name": "givenName",
    "last_name": "sn",
    "email": "mail",
  }
...
```

Then run `setup.sh -e @ldapextras.yml` during private automation hub installation.

6. Use the following examples to grant or deny access based on LDAP group membership.

   a. To grant private automation hub access (for example, members of the `cn=pah-nosoupforyou,ou=groups,dc=example,dc=com` group):

   ```yaml
   #ldapextras.yml
   ---
   ldap_extra_settings:
     AUTH_LDAP_DENY_GROUP: 'cn=pah-nosoupforyou,ou=groups,dc=example,dc=com'
   ...
   
   Then run `setup.sh -e @ldapextras.yml` during private automation hub installation.
   ```

   b. To deny private automation hub access (for example, members of the `cn=pah-nosoupforyou,ou=groups,dc=example,dc=com` group):

   ```yaml
   #ldapextras.yml
   ---
   ldap_extra_settings:
     AUTH_LDAP_DENY_GROUP: 'cn=pah-nosoupforyou,ou=groups,dc=example,dc=com'
   ...
   
   Then run `setup.sh -e @ldapextras.yml` during private automation hub installation.

7. Use this example to enable LDAP debug logging.

```yaml
#ldapextras.yml
---
ldap_extra_settings:
  GALAXY_LDAP_LOGGING: True
...
```

Then run `setup.sh -e @ldapextras.yml` during private automation hub installation.
NOTE

If it is not practical to re-run setup.sh or if debug logging is enabled for a short time, you can add a line containing GALAXY_LDAP_LOGGING: True manually to the /etc/pulp/settings.py file on private automation hub. Restart both pulpcore-api.service and nginx.service for the changes to take effect. To avoid failures due to human error, use this method only when necessary.

8. Use this example to configure LDAP caching by setting the variable AUTH_LDAP_CACHE_TIMEOUT.

```yaml
#ldapextras.yml
---
ldap_extra_settings:
  AUTH_LDAP_CACHE_TIMEOUT: 3600
...
```

Then run setup.sh -e @ldapextras.yml during private automation hub installation.

You can view all of your settings in the /etc/pulp/settings.py file on your private automation hub.

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

After you update the inventory file with required parameters for installing your private automation hub, run the installer setup script.

Procedure

- Run the setup.sh script
  
  $ sudo ./setup.sh

Installation of Red Hat Ansible Automation Platform will begin.

2.3. VERIFYING INSTALLATION OF AUTOMATION CONTROLLER

Verify that you installed automation controller successfully by logging in with the admin credentials you inserted in the inventory file.

Procedure

1. Navigate to the IP address specified for the automation controller node in the inventory file.

2. Log in with the Admin credentials you set in the inventory file.

NOTE

The automation controller server is accessible from port 80 (https://<CONTROLLER_SERVER_NAME>/) but redirects to port 443, so port 443 must also be available.
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.

Upon a successful login to automation controller, your installation of Red Hat Ansible Automation Platform 2.4 is complete.

2.3.1. Additional automation controller configuration and resources

See the following resources to explore additional automation controller configurations.

Table 2.1. Resources to configure automation controller

<table>
<thead>
<tr>
<th>Resource link</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automation Controller Quick Setup Guide</td>
<td>Set up automation controller and run your first playbook</td>
</tr>
<tr>
<td>Automation Controller Administration Guide</td>
<td>Configure automation controller administration through customer scripts, management jobs, etc.</td>
</tr>
<tr>
<td>Configuring proxy support for Red Hat Ansible Automation Platform</td>
<td>Set up automation controller with a proxy server</td>
</tr>
<tr>
<td>Managing usability analytics and data collection from automation controller</td>
<td>Manage what automation controller information you share with Red Hat</td>
</tr>
<tr>
<td>Automation Controller User Guide</td>
<td>Review automation controller functionality in more detail</td>
</tr>
</tbody>
</table>

2.4. VERIFYING INSTALLATION OF AUTOMATION HUB

Verify that you installed your automation hub successfully by logging in with the admin credentials you inserted into the inventory file.

Procedure

1. Navigate to the IP address specified for the automation hub node in the inventory file.

2. Log in with the Admin credentials you set in the inventory file.

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.

Upon a successful login to automation hub, your installation of Red Hat Ansible Automation Platform 2.4 is complete.

2.4.1. Additional automation hub configuration and resources
See the following resources to explore additional automation hub configurations.

**Table 2.2. Resources to configure automation controller**

<table>
<thead>
<tr>
<th>Resource link</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Managing user access in private automation hub</td>
<td>Configure user access for automation hub</td>
</tr>
<tr>
<td>Managing Red Hat Certified, validated, and Ansible Galaxy content in automation hub</td>
<td>Add content to your automation hub</td>
</tr>
<tr>
<td>Publishing proprietary content collections in automation hub</td>
<td>Publish internally developed collections on your automation hub</td>
</tr>
</tbody>
</table>

**2.5. VERIFYING EVENT-DRIVEN ANSIBLE CONTROLLER INSTALLATION**

Verify that you installed Event-Driven Ansible controller successfully by logging in with the admin credentials you inserted in the inventory file.

**Procedure**

1. Navigate to the IP address specified for the Event-Driven Ansible controller node in the inventory file.

2. Log in with the Admin credentials you set in the inventory file.

**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.

Upon a successful login to Event-Driven Ansible controller, your installation of Red Hat Ansible Automation Platform 2.4 is complete.

**2.6. POST-INSTALLATION STEPS**

Whether you are a new Ansible Automation Platform user looking to start automating, or an existing administrator looking to migrate old Ansible content to your latest installed version of Red Hat Ansible Automation Platform, explore the next steps to begin leveraging the new features of Ansible Automation Platform 2.4:

**2.6.1. Migrating data to Ansible Automation Platform 2.4**

For platform administrators looking to complete an upgrade to the Ansible Automation Platform 2.4, there may be additional steps needed to migrate data to a new instance:

**2.6.1.1. Migrating from legacy virtual environments (venvs) to automation execution environments**

Ansible Automation Platform 2.4 moves you away from custom Python virtual environments (venvs) in
favor of automation execution environments - containerized images that packages the necessary components needed to execute and scale your Ansible automation. This includes Ansible Core, Ansible Content Collections, Python dependencies, Red Hat Enterprise Linux UBI 8, and any additional package dependencies.

If you are looking to migrate your venvs to execution environments, you will (1) need to use the `awx-manage` command to list and export a list of venvs from your original instance, then (2) use `ansible-builder` to create execution environments.

Additional resources

- Upgrading to Automation Execution Environments guide
- Creating and Consuming Execution Environments.

2.6.1.2. Migrating to Ansible Engine 2.9 images using Ansible Builder

To migrate Ansible Engine 2.9 images for use with Ansible Automation Platform 2.4, the `ansible-builder` tool automates the process of rebuilding images (including its custom plugins and dependencies) for use with automation execution environments.

Additional resources

For more information on using Ansible Builder to build execution environments, see the Creating and Consuming Execution Environments.

2.6.1.3. Migrating to Ansible Core 2.13

When upgrading to Ansible Core 2.13, you need to update your playbooks, plugins, or other parts of your Ansible infrastructure in order to be supported by the latest version of Ansible Core. For instructions on updating your Ansible content for Ansible Core 2.13 compatibility, see the Ansible-core 2.13 Porting Guide.

2.6.2. Updating execution environment image locations

If your private automation hub was installed separately, you can update your execution environment image locations to point to your private automation hub. Use this procedure to update your execution environment image locations.

Procedure

1. Navigate to the directory containing `setup.sh`
2. Create `.group_vars/automationcontroller` by running the following command:

   ```bash
   touch ./group_vars/automationcontroller
   ```
3. Paste the following content into `.group_vars/automationcontroller`, being sure to adjust the settings to fit your environment:

   ```yaml
   # Automation Hub Registry
   registry_username: 'your-automation-hub-user'
   registry_password: 'your-automation-hub-password'
   registry_url: 'automationhub.example.org'
   ```
registry_verify_ssl: False

## Execution Environments
control_plane_execution_environment: 'automationhub.example.org/ee-supported-rhel8:latest'

global_job_execution_environments:
- name: "Default execution environment"
  image: "automationhub.example.org/ee-supported-rhel8:latest"
- name: "Ansible Engine 2.9 execution environment"
  image: "automationhub.example.org/ee-29-rhel8:latest"
- name: "Minimal execution environment"
  image: "automationhub.example.org/ee-minimal-rhel8:latest"

4. Run the `.setup.sh` script

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

Verification

1. Log into Ansible Automation Platform as a user with system administrator access.

2. Navigate to Administration → Execution Environments.

3. In the Image column, confirm that the execution environment image location has changed from the default value of `<registry url>/ansible-automation-platform-<version>/<image name>:<tag>` to `<automation hub url>/<image name>:<tag>`.

2.6.3. Scale up your automation using automation mesh

The automation mesh component of the Red Hat Ansible Automation Platform simplifies the process of distributing automation across multi-site deployments. For enterprises with multiple isolated IT environments, automation mesh provides a consistent and reliable way to deploy and scale up automation across your execution nodes using a peer-to-peer mesh communication network.

When upgrading from version 1.x to the latest version of the Ansible Automation Platform, you will need to migrate the data from your legacy isolated nodes into execution nodes necessary for automation mesh. You can implement automation mesh by planning out a network of hybrid and control nodes, then editing the inventory file found in the Ansible Automation Platform installer to assign mesh-related values to each of your execution nodes.

For instructions on how to migrate from isolated nodes to execution nodes, see the Red Hat Ansible Automation Platform Upgrade and Migration Guide.

For information about automation mesh and the various ways to design your automation mesh for your environment, see the Red Hat Ansible Automation Platform automation mesh guide.
CHAPTER 3. DISCONNECTED INSTALLATION

3.1. ANSIBLE AUTOMATION PLATFORM INSTALLATION ON DISCONNECTED RHEL

Install Ansible Automation Platform (AAP) automation controller and a private automation hub, with an installer-managed database located on the automation controller without an Internet connection.

3.1.1. Prerequisites

To install AAP on a disconnected network, complete the following prerequisites:

1. Create a subscription manifest.
2. Download the AAP setup bundle.
3. Create DNS records for the automation controller and private automation hub servers.

NOTE
The setup bundle includes additional components that make installing AAP easier in a disconnected environment. These include the AAP RPMs and the default execution environment (EE) images.

3.1.2. System Requirements

Hardware requirements are documented in the Automation Platform Installation Guide. Reference the “Red Hat Ansible Automation Platform Installation Guide” in the AAP Product Documentation for your version of AAP.

3.1.3. RPM Source

RPM dependencies for AAP that come from the BaseOS and AppStream repositories are not included in the setup bundle. To add these dependencies, you must obtain access to BaseOS and AppStream repositories.

- Satellite is the recommended method from Red Hat to synchronize repositories
- reposync - Makes full copies of the required RPM repositories and hosts them on the disconnected network
- RHEL Binary DVD - Use the RPMs available on the RHEL 8 Binary DVD

NOTE
The RHEL Binary DVD method requires the DVD for supported versions of RHEL, including version 8.6 or higher. See Red Hat Enterprise Linux Life Cycle for information on which versions of RHEL are currently supported.

3.2. SYNCHRONIZING RPM REPOSITORIES BY 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.
Procedure

1. Attach the BaseOS and AppStream required repositories:

```
# subscription-manager repos \
 --enable rhel-8-for-x86_64-baseos-rpms \
 --enable rhel-8-for-x86_64-appstream-rpms
```

2. Perform the reposync:

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

   a. Make certain that you use reposync with `--download-metadata` and without `--newest-only`.
   See [RHEL 8] Reposync.

   b. If not using `--newest-only` the repos downloaded will be ~90GB.

   c. If using `--newest-only` the repos downloaded will be ~14GB.

3. If you plan to use Red Hat Single Sign-On (RHSSO) sync these repositories.

   a. jb-eap-7.3-for-rhel-8-x86_64-rpms

   b. rh-sso-7.4-for-rhel-8-x86_64-rpms

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

4. Move the repositories to your disconnected network.

3.3. CREATING A NEW WEB SERVER TO HOST REPOSITORIES

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

Procedure

Use the following steps if creating a new web server.

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 controller and automation hub, add a repo file at /etc/yum.repos.d/local.repo, 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

3.4. ACCESSING RPM REPOSITORIES FOR LOCALLY MOUNTED DVD

If you are going to access the repositories from the DVD, it is necessary to set up a local repository. This section shows how to do that.

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]

In order to set up a repository see Need to set up yum repository for locally-mounted DVD on Red Hat Enterprise Linux 8.

3.5. ADDING A SUBSCRIPTION MANIFEST TO AAP WITHOUT AN INTERNET CONNECTION

In order to add a subscription to AAP without an Internet connection, create and import a subscription manifest.

Procedure

1. Login to access.redhat.com.

2. Navigate to Subscriptions → Subscriptions.

3. Click Subscription Allocations.

4. Click Create New subscription allocation.
5. Name the new subscription allocation.

6. Select **Satellite 6.8 → Satellite 6.8** as the type.

7. Click **Create**. The Details tab will open for your subscription allocation.

8. Click **Subscriptions** tab.

9. Click **Add Subscription**.

10. Find your AAP subscription, in the Entitlements box **add** the number of entitlements you want to assign to your environment. A single entitlement is needed for each node that will be managed by AAP: server, network device, etc.

11. Click **Submit**.

12. Click **Export Manifest**.

13. This downloads a file **manifest_<allocation name>_<date>.zip** that be imported with Automation Controller after installation.

### 3.6. INSTALLING THE AAP SETUP BUNDLE

The “bundle” version is strongly recommended for disconnected installations as it comes with the RPM content for AAP as well as the default execution environment images that will be uploaded to your Private Automation Hub during the installation process.

#### 3.6.1. Downloading the Setup Bundle

**Procedure**

1. Download the AAP setup bundle package by navigating to the Red Hat Ansible Automation Platform download page and clicking **Download Now** for the Ansible Automation Platform 2.4 Setup Bundle.

#### 3.6.1.1. Installing the Setup Bundle

The download and installation of the setup bundle needs to be located on the automation controller. From the controller, untar the bundle, edit the inventory file, and run the setup.

1. Untar the bundle

   ```bash
   $ tar xvf \
   ansible-automation-platform-setup-bundle-2.4-1.tar.gz
   $ cd ansible-automation-platform-setup-bundle-2.4-1
   ```

2. Edit the inventory file to include the required options
   
   a. automationcontroller group
   b. automationhub group
   c. admin_password
   d. pg_password
e. automationhub_admin_password
f. automationhub_pg_host, automationhub_pg_port
g. automationhub_pg_password

Example Inventory

```
[automationcontroller]
automationcontroller.example.org ansible_connection=local

[automationcontroller:vars]
peers=execution_nodes

[automationhub]
automationhub.example.org

[all:vars]
admin_password='password123'
pg_database='awx'
pg_username='awx'
pg_password='dbpassword123'
receptor_listener_port=27199

automationhub_admin_password='hubpassword123'
automationhub_pg_host='automationcontroller.example.org'
automationhub_pg_port='5432'
automationhub_pg_database='automationhub'
automationhub_pg_username='automationhub'
automationhub_pg_password='dbpassword123'
automationhub_pg_sslmode='prefer'
```

**NOTE**

The inventory should be kept intact after installation since it is used for backup, restore, and upgrade functions. Consider keeping a backup copy in a secure location, given that the inventory file contains passwords.

3. Run the AAP setup bundle executable as the root user

```bash
$ sudo -i
# cd /path/to/ansible-automation-platform-setup-bundle-2.4-1
# ./setup.sh
```

4. Once installation is complete, navigate to the Fully Qualified Domain Name (FQDN) for the Automation controller node that was specified in the installation inventory file.

5. Log in with the administrator credentials specified in the installation inventory file.

### 3.7. COMPLETING POST INSTALLATION TASKS
3.7.1. Adding Controller Subscription

Procedure

1. Navigate to the FQDN of the Automation controller. Login with admin and the password you specified as `admin_password` in your inventory file.

2. Click Browse and select the `manifest.zip` you created earlier.

3. Click Next.

4. Uncheck User analytics and Automation analytics. These rely on an Internet connection and should be turned off.

5. Click Next.

6. Read the End User License Agreement and click Submit if you agree.

3.7.2. Updating the CA trust store

3.7.2.1. Self-Signed Certificates

By default, AAP hub and controller are installed using self signed certificates. This creates an issue where the controller does not trust the hub’s certificate and will not download the execution environments from the hub. The solution is to import the hub’s CA cert as a trusted cert on the controller. You can use SCP or directly copy and paste from one file into another to perform this action. The steps below are copied from a KB article found at https://access.redhat.com/solutions/6707451.

3.7.2.2. Copying the root certificate on the private automation hub to the automation controller using secure copy (SCP)

If SSH is available as the root user between the controller and the private automation hub, use SCP to copy the root certificate on the private automation hub to the controller and run `update-ca-trust` on the controller to update the CA trust store.

On the Automation controller

```bash
$ sudo -i
# scp <hub_fqdn>/etc/pulp/certs/root.crt
/etc/pki/ca-trust/source/anchors/automationhub-root.crt
# update-ca-trust
```

3.7.2.3. Copying and Pasting

If SSH is unavailable as root between the private automation hub and the controller, copy the contents of the file `/etc/pulp/certs/root.crt` on the private automation hub and paste it into a new file on the controller called `/etc/pki/ca-trust/source/anchors/automationhub-root.crt`. After the new file is created, run the command `update-ca-trust` in order to update the CA trust store with the new certificate.

On the Private automation hub

```bash
$ sudo -i
# cat /etc/pulp/certs/root.crt
(copy the contents of the file, including the lines with 'BEGIN CERTIFICATE' and..."
`END CERTIFICATE`

On the Automation controller

```bash
$ sudo -i
# vi /etc/pki/ca-trust/source/anchors/automationhub-root.crt
(paste the contents of the root.crt file from the private automation hub into the new file and write to disk)
# update-ca-trust
```

3.8. IMPORTING COLLECTIONS INTO PRIVATE AUTOMATION HUB

You can download collection tarball files from the following sources:

- Red Hat certified collections are found on [Red Hat Automation Hub](https://console.redhat.com/ansible/automation-hub/).
- Community collections are found on [Ansible Galaxy](https://galaxy.ansible.com).

3.8.1. Downloading collection from Red Hat Automation Hub

This section gives instructions on how to download a collection from Red Hat Automation Hub. If the collection has dependencies, they will also need to be downloaded and installed.

**Procedure**


2. Click on the **collection** you wish to download.

3. Click **Download tarball**

4. To verify if a collection has dependencies, click the **Dependencies** tab.

5. Download any dependencies needed for this collection.

3.9. CREATING COLLECTION NAMESPACE

The namespace of the collection must exist for the import to be successful. You can find the namespace name by looking at the first part of the collection tarball filename. For example the namespace of the collection `ansible-netcommon-3.0.0.tar.gz` is `ansible`.

**Procedure**

1. Login to private automation hub web console.

2. Navigate to **Collections** → **Namespaces**.

3. Click **Create**.

4. Provide the namespace name.

5. Click **Create**.
3.9.1. Importing the collection tarball with GUI

1. Login to private automation hub web console.

2. Navigate to Collections → Namespaces.

3. Click on View collections of the namespace you will be importing the collection into.

4. Click Upload collection.

5. Click the folder icon and select the tarball of the collection.

6. Click Upload.

This opens the 'My Imports' page. You can see the status of the import and various details of the files and modules that have been imported.

3.9.1.1. Importing the collection tarball using ansible-galaxy via CLI

You can import collections into the private automation hub by using the command-line interface rather than the GUI.

1. Copy the collection tarballs to the private automation hub.

2. Log in to the private automation hub server via SSH.

3. Add the self-signed root CA cert to the trust store on the automation hub.

   ```
   # cp /etc/pulp/certs/root.crt /etc/pki/ca-trust/source/anchors/automationhub-root.crt
   # update-ca-trust
   ```

4. Update the `/etc/ansible/ansible.cfg` file with your hub configuration. Use either a token or a username and password for authentication.

   ```
   [galaxy]
   server_list = private_hub
   
   [galaxy_server.private_hub]
   url=https://<hub_fqdn>/api/galaxy/
   token=<token_from_private_hub>
   ```

5. Import the collection using the ansible-galaxy command.

   ```
   $ ansible-galaxy collection publish <collection_tarball>
   ```

**NOTE**

Create the namespace that the collection belongs to in advance or publishing the collection will fail.

3.10. APPROVING THE IMPORTED COLLECTION
After you have imported collections with either the GUI or the CLI method, you must approve them by using the GUI. After they are approved, they are available for use.

Procedure

1. Login to private automation hub web console.
2. Go to Collections → Approval.
3. Click Approve for the collection you wish to approve.
4. The collection is now available for use in your private automation hub.

**NOTE**

The collection is added to the "Published" repository regardless of its source.

1. Import any dependency for the collection using these same steps.

Recommended collections depend on your use case. Ansible and Red Hat provide the following collections:


### 3.10.1. Custom Execution Environments

Use the ansible-builder program to create custom execution environment images. For disconnected environments, custom EE images can be built in the following ways:

- Build an EE image on an internet-facing system and import it to the disconnected environment
- Build an EE image entirely on the disconnected environment with some modifications to the normal process of using ansible-builder
- Create a minimal base container image that includes all of the necessary modifications for a disconnected environment, then build custom EE images from the base container image

#### 3.10.1.1. Transferring a Custom EE Images Across a Disconnected Boundary

A custom execution environment image can be built on an internet-facing machine using the existing documentation. Once an execution environment has been created it is available in the local podman image cache. You can then transfer the custom EE image across a disconnected boundary. To transfer the custom EE image across a disconnected boundary, first save the image:

```
$ podman image save localhost/custom-ee:latest | gzip -c custom-ee-latest.tar.gz
```

Transfer the file across the disconnected boundary by using an existing mechanism such as sneakernet, one-way diode, etc. After the image is available on the disconnected side, import it into the local podman cache, tag it, and push it to the disconnected hub:

```
$ podman image load -i custom-ee-latest.tar.gz
```
3.11. BUILDING AN EXECUTION ENVIRONMENT IN A DISCONNECTED ENVIRONMENT

When building a custom execution environment, the ansible-builder tool defaults to downloading the following requirements from the internet:

- Ansible Galaxy (galaxy.ansible.com) or Automation Hub (cloud.redhat.com) for any collections added to the EE image.
- PyPI (pypi.org) for any python packages required as collection dependencies.
- The UBI repositories (cdn.redhat.com) for updating any UBI-based EE images.
  - The RHEL repositories might also be needed to meet certain collection requirements.
- registry.redhat.io for access to the ansible-builder-rhel8 container image.

Building an EE image in a disconnected environment requires a subset of all of these mirrored, or otherwise made available on the disconnected network. See Importing Collections into Private Automation Hub for information on importing collections from Galaxy or Automation Hub into a private automation hub.

Mirrored PyPI content once transferred into the high-side network can be made available using a web server or an artifact repository like Nexus.

The UBI repositories can be mirrored on the low-side using a tool like reposync, imported to the disconnected environment, and made available from Satellite or a simple web server (since the content is freely redistributable).

The ansible-builder-rhel8 container image can be imported into a private automation hub in the same way a custom EE can be imported. See Transferring a Custom EE Images Across a Disconnected Boundary for details substituting localhost/custom-ee for registry.redhat.io/ansible-automation-platform-21/ansible-builder-rhel8. This will make the ansible-builder-rhel8 image available in the private automation hub registry along with the default EE images.

Once all of the prerequisites are available on the high-side network, ansible-builder and podman can be used to create a custom execution environment image.

3.12. INSTALLING THE ANSIBLE-BUILDER RPM

Procedure

1. On a RHEL system, install the ansible-builder RPM. This can be done in one of several ways:
   a. Subscribe the RHEL box to a Satellite on the disconnected network.
   b. Attach the Ansible Automation Platform subscription and enable the AAP repo.
   c. Install the ansible-builder RPM.
2. Unarchive the AAP setup bundle.

3. Install the ansible-builder RPM and its dependencies from the included content:

   ```
   $ tar -xzvf ansible-automation-platform-setup-bundle-2.4-1.tar.gz
   $ cd ansible-automation-platform-setup-bundle-2.4/bundle/el8/repos/
   $ sudo yum install ansible-builder-1.0.1-2.el8ap.noarch.rpm
   python38-requirements-parser-0.2.0-3.el8ap.noarch.rpm
   ```

4. Create a directory for your custom EE build artifacts.

   ```
   $ mkdir custom-ee
   $ cd custom-ee/
   ```

5. Create an `execution-environment.yml` file that defines the requirements for your custom EE following the documentation at https://ansible-builder.readthedocs.io/en/stable/definition/. Override the `EE_BASE_IMAGE` and `EE_BUILDER_IMAGE` variables to point to the EEs available in your private automation hub.

   ```
   $ cat execution-environment.yml
   ---
   version: 1
   build_arg_defaults:
     EE_BASE_IMAGE: '<<hub_fqdn>>/ee-supported-rhel8:latest'
     EE_BUILDER_IMAGE: '<<hub_fqdn>>/ansible-builder-rhel8:latest'
   dependencies:
     python: requirements.txt
     galaxy: requirements.yml
   ```

6. Create an `ansible.cfg` file that points to your private automation hub and contains credentials that allow uploading, such as an admin user token.

   ```
   $ cat ansible.cfg
   [galaxy]
   server_list = private_hub

   [galaxy_server.private_hub]
   url=https://<<hub_fqdn>>/api/galaxy/
   token=<admin_token>
   ```

7. Create a `ubi.repo` file that points to your disconnected UBI repo mirror (this could be your Satellite if the UBI content is hosted there). This is an example output where `reposync` was used to mirror the UBI repos.

   ```
   $ cat ubi.repo
   [ubi-8-baseos]
   name = Red Hat Universal Base Image 8 (RPMs) - BaseOS
   baseurl = http://<<ubi_mirror_fqdn>>/repos/ubi-8-baseos
   ```
8. Add the CA certificate used to sign the private automation hub web server certificate.
   a. For self-signed certificates (the installer default), make a copy of the file
      `/etc/pulp/certs/root.crt` from your private automation hub and name it `hub-root.crt`.
   b. If an internal certificate authority was used to request and sign the private automation hub
      web server certificate, make a copy of that CA certificate called `hub-root.crt`.

9. Create your python requirements.txt and ansible collection requirements.yml with the content
   needed for your custom EE image. Note that any collections you require should already be
   uploaded into your private automation hub.

10. Use ansible-builder to create the context directory used to build the EE image.

    $ ansible-builder create
    Complete! The build context can be found at: /home/cloud-user/custom-ee/context
    $ ls -1F
    ansible.cfg
    context/
    execution-environment.yml
    hub-root.crt
    pip.conf
    requirements.txt
    requirements.yml
    ubi.repo

11. Copy the files used to override the internet-facing defaults into the context directory.

    $ cp ansible.cfg hub-root.crt pip.conf ubi.repo context/

12. Edit the file `context/Containerfile` and add the following modifications.

    a. In the first `EE_BASE_IMAGE` build section, add the `ansible.cfg` and `hub-root.crt` files and
       run the `update-ca-trust` command.
    b. In the `EE_BUILDER_IMAGE` build section, add the `ubi.repo` and `pip.conf` files.
    c. In the final `EE_BASE_IMAGE` build section, add the `ubi.repo` and `pip.conf` files.

    $ cat context/Containerfile
    ARG EE_BASE_IMAGE=<hub_fqdn>/ee-supported-rhel8:latest
    ARG EE_BUILDER_IMAGE=<hub_fqdn>/ansible-builder-rhel8:latest
    FROM $EE_BASE_IMAGE as galaxy
    ARG ANSIBLE_GALAXY_CLI_COLLECTION_OPTS=
USER root
ADD _build /build
WORKDIR /build

# this section added
ADD ansible.cfg /etc/ansible/ansible.cfg
ADD hub-root.crt /etc/pki/ca-trust/source/anchors/hub-root.crt
RUN update-ca-trust
# end additions
RUN ansible-galaxy role install -r requirements.yml \
    --roles-path /usr/share/ansible/roles
RUN ansible-galaxy collection install \ 
    $ANSIBLE_GALAXY_CLI_COLLECTION_OPTS -r requirements.yml \ 
    --collections-path /usr/share/ansible/collections

FROM $EE_BUILDER_IMAGE as builder
COPY --from=galaxy /usr/share/ansible /usr/share/ansible
ADD _build/requirements.txt requirements.txt
RUN ansible-builder introspect --sanitize \
    --user-pip=requirements.txt \
    --write-bindep=/tmp/src/bindep.txt \
    --write-pip=/tmp/src/requirements.txt
# this section added
ADD ubi.repo /etc/yum.repos.d/ubi.repo
ADD pip.conf /etc/pip.conf
# end additions
RUN assemble

FROM $EE_BASE_IMAGE
USER root
COPY --from=galaxy /usr/share/ansible /usr/share/ansible
# this section added
ADD ubi.repo /etc/yum.repos.d/ubi.repo
ADD pip.conf /etc/pip.conf
# end additions

COPY --from=builder /output/ /output/
RUN /output/install-from-bindep && rm -rf /output/wheels

13. Create the EE image in the local podman cache using the podman command.

$ podman build -f context/Containerfile \
    -t <hub_fqdn>/custom-ee:latest

14. Once the custom EE image builds successfully, push it to the private automation hub.

$ podman push <hub_fqdn>/custom-ee:latest

3.12.1. Workflow for upgrading between minor AAP releases

To upgrade between minor releases of AAP 2, use this general workflow.
Procedure

1. Download and unarchive the latest AAP 2 setup bundle.

2. Take a backup of the existing installation.

3. Copy the existing installation inventory file into the new setup bundle directory.

4. Run `/setup.sh` to upgrade the installation.

For example, to upgrade from version 2.2.0-7 to 2.3-1.2, make sure that both setup bundles are on the initial controller node where the installation occurred:

```bash
$ ls -1F
ansible-automation-platform-setup-bundle-2.2.0-7/
ansible-automation-platform-setup-bundle-2.2.0-7.tar.gz
ansible-automation-platform-setup-bundle-2.3-1.2/
ansible-automation-platform-setup-bundle-2.3-1.2.tar.gz
```

Back up the 2.2.0-7 installation:

```bash
$ cd ansible-automation-platform-setup-bundle-2.2.0-7
$ sudo ./setup.sh -b
$ cd ..
```

Copy the 2.2.0-7 inventory file into the 2.3-1.2 bundle directory:

```bash
$ cd ansible-automation-platform-setup-bundle-2.2.0-7
$ cp inventory ../ansible-automation-platform-setup-bundle-2.3-1.2/
$ cd ..
```

Upgrade from 2.2.0-7 to 2.3-1.2 with the setup.sh script:

```bash
$ cd ansible-automation-platform-setup-bundle-2.3-1.2
$ sudo ./setup.sh
```
APPENDIX A. INVENTORY FILE VARIABLES

The following tables contain information about the pre-defined variables used in Ansible installation inventory files. Not all of these variables are required.

### A.1. GENERAL VARIABLES

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>enable_insights_collection</td>
<td>The default install registers the node to the Red Hat Insights for Red Hat Ansible Automation Platform Service if the node is registered with Subscription Manager. Set to <code>False</code> to disable. Default = <code>true</code></td>
</tr>
<tr>
<td>registry_password</td>
<td><code>registry_password</code> is only required if a non-bundle installer is used. Password credential for access to <code>registry_url</code>. Used for both <code>[automationcontroller]</code> and <code>[automationhub]</code> groups. Enter your Red Hat Registry Service Account credentials in <code>registry_username</code> and <code>registry_password</code> to link to the Red Hat container registry. When <code>registry_url</code> is <code>registry.redhat.io</code>, username and password are required if not using bundle installer.</td>
</tr>
<tr>
<td>registry_url</td>
<td>Used for both <code>[automationcontroller]</code> and <code>[automationhub]</code> groups. Default = <code>registry.redhat.io</code>.</td>
</tr>
<tr>
<td>registry_username</td>
<td><code>registry_username</code> is only required if a non-bundle installer is used. User credential for access to <code>registry_url</code>. Used for both <code>[automationcontroller]</code> and <code>[automationhub]</code> groups, but only if the value of <code>registry_url</code> is <code>registry.redhat.io</code>. Enter your Red Hat Registry Service Account credentials in <code>registry_username</code> and <code>registry_password</code> to link to the Red Hat container registry.</td>
</tr>
</tbody>
</table>
routable_hostname is used if the machine running the installer can only route to the target host through a specific URL, for example, if you use shortnames in your inventory, but the node running the installer can only resolve that host using FQDN.

If routable_hostname is not set, it should default to ansible_host. Then if, and only if ansible_host is not set, inventory_hostname is used as a last resort.

Note that 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

### A.2. ANSIBLE AUTOMATION HUB VARIABLES

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>automationhub_admin_password</td>
<td>Required</td>
</tr>
<tr>
<td>automationhub_api_token</td>
<td>If upgrading from Ansible Automation Platform 2.0 or earlier, you must either:</td>
</tr>
<tr>
<td></td>
<td>- provide an existing Ansible automation hub token as automationhub_api_token, or</td>
</tr>
<tr>
<td></td>
<td>- set generate_automationhub_token to true to generate a new token</td>
</tr>
<tr>
<td></td>
<td>Generating a new token invalidates the existing token.</td>
</tr>
<tr>
<td>automationhub_authentication_backend</td>
<td>This variable is not set by default. Set it to ldap to use LDAP authentication.</td>
</tr>
<tr>
<td></td>
<td>When this is set to ldap, you must also set the following variables:</td>
</tr>
<tr>
<td></td>
<td>- automationhub_ldap_server_uri</td>
</tr>
<tr>
<td></td>
<td>- automationhub_ldap_bind_dn</td>
</tr>
<tr>
<td></td>
<td>- automationhub_ldap_bind_password</td>
</tr>
<tr>
<td></td>
<td>- automationhub_ldap_user_search_base_dn</td>
</tr>
<tr>
<td></td>
<td>- automationhub_ldap_group_search_base_dn</td>
</tr>
<tr>
<td>Variable</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>automationhub_auto_sign_collections</code></td>
<td>If a collection signing service is enabled, collections are not signed automatically by default. Setting this parameter to <code>true</code> signs them by default. Default = <code>false</code>.</td>
</tr>
<tr>
<td><code>automationhub_backup_collections</code></td>
<td>Optional Ansible automation hub provides artifacts in <code>/var/lib/pulp</code>. Automation controller automatically backs up the artifacts by default. You can also set <code>automationhub_backup_collections = false</code> and the backup/restore process does not then backup or restore <code>/var/lib/pulp</code>. Default = <code>true</code></td>
</tr>
<tr>
<td><code>automationhub_collection_seed_repository</code></td>
<td>When the bundle installer is run, validated content is uploaded to the <code>validated</code> repository, and certified content is uploaded to the <code>rh-certified</code> 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 <code>automationhub_seed_collections</code> to <code>false</code> to disable the seeding. If you only want one type of content, set <code>automationhub_seed_collections</code> to <code>true</code> and <code>automationhub_collection_seed_repository</code> to the type of content you do want to include.</td>
</tr>
<tr>
<td><code>automationhub_collection_signing_service_key</code></td>
<td>If a collection signing service is enabled, you must provide this variable to ensure that collections can be properly signed. <code>/absolute/path/to/key/to/sign</code></td>
</tr>
<tr>
<td><code>automationhub_collection_signing_service_script</code></td>
<td>If a collection signing service is enabled, you must provide this variable to ensure that collections can be properly signed. <code>/absolute/path/to/script/that/signs</code></td>
</tr>
<tr>
<td>Variable</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>automationhub_create_default_collection_signing_service</td>
<td>The default install does not create a collection signing service. If set to <strong>true</strong> a signing service is created.</td>
</tr>
<tr>
<td></td>
<td>Default = <strong>false</strong></td>
</tr>
<tr>
<td>automationhub_container_signing_service_key</td>
<td>If a container signing service is enabled, you must provide this variable to ensure that containers can be properly signed.</td>
</tr>
<tr>
<td></td>
<td>/absolute/path/to/key/to/sign</td>
</tr>
<tr>
<td>automationhub_container_signing_service_script</td>
<td>If a container signing service is enabled, you must provide this variable to ensure that containers can be properly signed.</td>
</tr>
<tr>
<td></td>
<td>/absolute/path/to/script/that/signs</td>
</tr>
<tr>
<td>automationhub_create_default_container_signing_service</td>
<td>The default install does not create a container signing service. If set to <strong>true</strong> a signing service is created.</td>
</tr>
<tr>
<td></td>
<td>Default = <strong>false</strong></td>
</tr>
<tr>
<td>automationhub_disable_hsts</td>
<td>The default install deploys a TLS enabled Ansible automation hub. Use if automation hub is deployed with <strong>HTTP Strict Transport Security (HSTS)</strong> web-security policy enabled. Unless specified otherwise, the HSTS web-security policy mechanism is enabled. This setting allows you to disable it if required.</td>
</tr>
<tr>
<td></td>
<td>Default = <strong>false</strong></td>
</tr>
<tr>
<td>automationhub_disable_https</td>
<td>Optional</td>
</tr>
<tr>
<td></td>
<td>If Ansible automation hub is deployed with HTTPS enabled.</td>
</tr>
<tr>
<td></td>
<td>Default = <strong>false</strong>.</td>
</tr>
<tr>
<td>automationhub_enable_api_access_log</td>
<td>When set to <strong>true</strong>, 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.</td>
</tr>
<tr>
<td></td>
<td>Default = <strong>false</strong>.</td>
</tr>
<tr>
<td>Variable</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>automationhub_enable_analytics</code></td>
<td>A Boolean indicating whether to enable pulp analytics for the version of pulpcore used in automation hub in Ansible Automation Platform 2.4. To enable pulp analytics, set <code>automationhub_enable_analytics = true</code>. Default = <code>false</code>.</td>
</tr>
<tr>
<td><code>automationhub_enable_unauthenticated_collection_access</code></td>
<td>Enables unauthorized users to view collections. To enable unauthorized users to view collections, set <code>automationhub_enable_unauthenticated_collection_access = true</code>. Default = <code>false</code>.</td>
</tr>
<tr>
<td><code>automationhub_enable_unauthenticated_collection_download</code></td>
<td>Enables unauthorized users to download collections. To enable unauthorized users to download collections, set <code>automationhub_enable_unauthenticated_collection_download = true</code>. Default = <code>false</code>.</td>
</tr>
<tr>
<td><code>automationhub_importer_settings</code></td>
<td>Optional Dictionary of setting to pass to galaxy-importer. At import time collections can go through a series of checks. Behavior is driven by <code>galaxy-importer.cfg</code> configuration. Examples are <code>ansible-doc</code>, <code>ansible-lint</code>, and <code>flake8</code>. This parameter enables you to drive this configuration.</td>
</tr>
</tbody>
</table>
## APPENDIX A. INVENTORY FILE VARIABLES

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>automationhub_main_url</code></td>
<td>The main automation hub URL that clients connect to.</td>
</tr>
<tr>
<td></td>
<td>For example, https://&lt;load balancer host&gt;.</td>
</tr>
<tr>
<td></td>
<td>If not specified, the first node in the [automationhub] group is used.</td>
</tr>
<tr>
<td></td>
<td>Use <code>automationhub_main_url</code> to specify the main automation hub URL that clients connect to if you are implementing Red Hat Single Sign-On on your automation hub environment.</td>
</tr>
<tr>
<td><code>automationhub_pg_database</code></td>
<td>Required</td>
</tr>
<tr>
<td></td>
<td>The database name.</td>
</tr>
<tr>
<td></td>
<td>Default = <code>automationhub</code></td>
</tr>
<tr>
<td><code>automationhub_pg_host</code></td>
<td>Required if not using internal database.</td>
</tr>
<tr>
<td></td>
<td>The hostname of the remote postgres database used by automation hub</td>
</tr>
<tr>
<td></td>
<td>Default = <code>127.0.0.1</code></td>
</tr>
<tr>
<td><code>automationhub_pg_password</code></td>
<td>The password for the automation hub PostgreSQL database.</td>
</tr>
<tr>
<td></td>
<td>Do not use special characters for <code>automationhub_pg_password</code>. They can cause the password to fail.</td>
</tr>
<tr>
<td><code>automationhub_pg_port</code></td>
<td>Required if not using internal database.</td>
</tr>
<tr>
<td></td>
<td>Default = <code>5432</code></td>
</tr>
<tr>
<td><code>automationhub_pg_sslmode</code></td>
<td>Required</td>
</tr>
<tr>
<td></td>
<td>Default = <code>prefer</code></td>
</tr>
<tr>
<td><code>automationhub_pg_username</code></td>
<td>Required</td>
</tr>
<tr>
<td></td>
<td>Default = <code>automationhub</code></td>
</tr>
<tr>
<td>Variable</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>&lt;automationhub_require_content_approval&gt;</code></td>
<td>Optional</td>
</tr>
<tr>
<td></td>
<td>Value is <code>true</code> if automation hub enforces the approval mechanism before collections are made available.</td>
</tr>
<tr>
<td></td>
<td>By default when you upload collections to automation hub an administrator must approve it before it is made available to the users.</td>
</tr>
<tr>
<td></td>
<td>If you want to disable the content approval flow, set the variable to <code>false</code>.</td>
</tr>
<tr>
<td></td>
<td>Default = <code>true</code></td>
</tr>
<tr>
<td><code>&lt;automationhub_seed_collections&gt;</code></td>
<td>A boolean that defines whether or not preloading is enabled.</td>
</tr>
<tr>
<td></td>
<td>When the bundle installer is run, validated content is uploaded to the <code>validated</code> repository, and certified content is uploaded to the <code>rh-certified</code> repository.</td>
</tr>
<tr>
<td></td>
<td>By default, both certified and validated content are uploaded.</td>
</tr>
<tr>
<td></td>
<td>If you do not want to install content, set <code>&lt;automationhub_seed_collections&gt;</code> to <code>false</code> to disable the seeding.</td>
</tr>
<tr>
<td></td>
<td>If you only want one type of content, set <code>&lt;automationhub_seed_collections&gt;</code> to <code>true</code> and <code>&lt;automationhub_collection_seed_repository&gt;</code> to the type of content you do want to include.</td>
</tr>
<tr>
<td></td>
<td>Default = <code>true</code>.</td>
</tr>
<tr>
<td><code>&lt;automationhub_ssl_cert&gt;</code></td>
<td>Optional</td>
</tr>
<tr>
<td></td>
<td><code>/path/to/automationhub.cert</code> Same as <code>&lt;web_server_ssl_cert&gt;</code> but for automation hub UI and API</td>
</tr>
<tr>
<td><code>&lt;automationhub_ssl_key&gt;</code></td>
<td>Optional</td>
</tr>
<tr>
<td></td>
<td><code>/path/to/automationhub.key</code></td>
</tr>
<tr>
<td></td>
<td>Same as <code>&lt;web_server_ssl_key&gt;</code> but for automation hub UI and API</td>
</tr>
<tr>
<td>Variable</td>
<td>Description</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td><strong>automationhub_ssl_validate_certs</strong></td>
<td>For Red Hat Ansible Automation Platform 2.2 and later, this value is no longer used. Value is <strong>true</strong> if automation hub should validate certificate when requesting itself because by default, Ansible Automation Platform deploys with self-signed certificates. Default = <strong>false</strong>.</td>
</tr>
<tr>
<td><strong>automationhub_upgrade</strong></td>
<td>Deprecated For Ansible Automation Platform 2.2.1 and later, the value of this has been fixed at true. Automation hub always updates with the latest packages.</td>
</tr>
<tr>
<td>ee_from_hub_only</td>
<td>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 <code>'ee_from_hub_only'</code> to <strong>true</strong>. If set to <strong>false</strong>, execution environment images are also taken directly from Red Hat. Default = <strong>true</strong> when the bundle installer is used.</td>
</tr>
</tbody>
</table>
| generate_automationhub_token | If upgrading from Red Hat Ansible Automation Platform 2.0 or earlier, you must either:  
  - provide an existing Ansible automation hub token as **automationhub_api_token** or  
  - set **generate_automationhub_token** to **true** to generate a new token. Generating a new token will invalidate the existing token. |
<p>| nginx_hsts_max_age | This variable specifies how long, in seconds, the system should be considered as a <strong>HTTP Strict Transport Security</strong> (HSTS) host. That is, how long HTTPS is used exclusively for communication. Default = 63072000 seconds, or two years. |
| nginx_tls_protocols | Defines support for <strong>ssl_protocols</strong> in Nginx. Default = <strong>TLSv1.2</strong>. |</p>
<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>pulp_db_fields_key</strong></td>
<td>Relative or absolute path to the Fernet symmetric encryption key 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.</td>
</tr>
<tr>
<td><strong>sso_automation_platform_login_theme</strong></td>
<td>Optional</td>
</tr>
<tr>
<td></td>
<td>Used for Ansible Automation Platform managed and externally managed Red Hat Single Sign-On.</td>
</tr>
<tr>
<td></td>
<td>Path to the directory where theme files are located. If changing this variable, you must provide your own theme files.</td>
</tr>
<tr>
<td></td>
<td>Default = <strong>ansible-automation-platform</strong></td>
</tr>
<tr>
<td><strong>sso_automation_platform_realm</strong></td>
<td>Optional</td>
</tr>
<tr>
<td></td>
<td>Used for Ansible Automation Platform managed and externally managed Red Hat Single Sign-On.</td>
</tr>
<tr>
<td></td>
<td>The name of the realm in SSO.</td>
</tr>
<tr>
<td></td>
<td>Default = <strong>ansible-automation-platform</strong></td>
</tr>
<tr>
<td><strong>sso_automation_platform_realm_displayname</strong></td>
<td>Optional</td>
</tr>
<tr>
<td></td>
<td>Used for Ansible Automation Platform managed and externally managed Red Hat Single Sign-On.</td>
</tr>
<tr>
<td></td>
<td>Display name for the realm.</td>
</tr>
<tr>
<td></td>
<td>Default = <strong>Ansible Automation Platform</strong></td>
</tr>
<tr>
<td><strong>sso_console_admin_username</strong></td>
<td>Optional</td>
</tr>
<tr>
<td></td>
<td>Used for Ansible Automation Platform managed and externally managed Red Hat Single Sign-On.</td>
</tr>
<tr>
<td></td>
<td>SSO administration username.</td>
</tr>
<tr>
<td></td>
<td>Default = <strong>admin</strong></td>
</tr>
<tr>
<td><strong>sso_console_admin_password</strong></td>
<td>Required</td>
</tr>
<tr>
<td></td>
<td>Used for Ansible Automation Platform managed and externally managed Red Hat Single Sign-On.</td>
</tr>
<tr>
<td></td>
<td>SSO administration password.</td>
</tr>
<tr>
<td>Variable</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>sso_custom_keystore_file</td>
<td>Optional</td>
</tr>
</tbody>
</table>
|                                  | Used for Ansible Automation Platform managed Red Hat Single Sign-On only.  
|                                  | Customer-provided keystore for SSO.                                                                                                       |
| sso_host                         | Required                                                                                                                                   |
|                                  | Used for Ansible Automation Platform externally managed Red Hat Single Sign-On only.  
|                                  | Automation hub requires SSO and SSO administration credentials for authentication.  
|                                  | If SSO is not provided in the inventory for configuration, then you must use this variable to define the SSO host.                           |
| sso_keystore_file_remote         | Optional                                                                                                                                   |
|                                  | Used for Ansible Automation Platform managed Red Hat Single Sign-On only.  
|                                  | Set to **true** if the customer-provided keystore is on a remote node.  
|                                  | Default = **false**                                                                                                                       |
| sso_keystore_name                | Optional                                                                                                                                   |
|                                  | Used for Ansible Automation Platform managed Red Hat Single Sign-On only.  
|                                  | Name of keystore for SSO.  
<p>|                                  | Default = <strong>ansible-automation-platform</strong>                                                                                                 |
| sso_keystore_password            | Password for keystore for HTTPS enabled SSO.                                                                                               |
|                                  | Required when using Ansible Automation Platform managed SSO and when HTTPS is enabled. The default install deploys SSO with <strong>sso_use_https=true</strong>. |</p>
<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>sso_redirect_host</strong></td>
<td>Optional</td>
</tr>
<tr>
<td></td>
<td>Used for Ansible Automation Platform managed and externally managed Red Hat Single Sign-On.</td>
</tr>
<tr>
<td></td>
<td>If <strong>sso_redirect_host</strong> is set, it is used by the application to connect to SSO for authentication.</td>
</tr>
<tr>
<td></td>
<td>This must be reachable from client machines.</td>
</tr>
<tr>
<td><strong>sso_ssl_validate_certs</strong></td>
<td>Optional</td>
</tr>
<tr>
<td></td>
<td>Used for Ansible Automation Platform managed and externally managed Red Hat Single Sign-On.</td>
</tr>
<tr>
<td></td>
<td>Set to <strong>true</strong> if the certificate is to be validated during connection.</td>
</tr>
<tr>
<td></td>
<td>Default = <strong>true</strong></td>
</tr>
<tr>
<td><strong>sso_use_https</strong></td>
<td>Optional</td>
</tr>
<tr>
<td></td>
<td>Used for Ansible Automation Platform managed and externally managed Red Hat Single Sign-On.</td>
</tr>
<tr>
<td></td>
<td>If Single Sign On uses https.</td>
</tr>
<tr>
<td></td>
<td>Default = <strong>true</strong></td>
</tr>
</tbody>
</table>

For Ansible automation hub to connect to LDAP directly; the following variables must be configured. A list of other LDAP related variables (not covered by the `automationhub_ldap_` variables below) that can be passed using the `ldap_extra_settings` variable can be found here: [https://django-auth-ldap.readthedocs.io/en/latest/reference.html#settings](https://django-auth-ldap.readthedocs.io/en/latest/reference.html#settings)

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>automationhub_ldap_bind_dn</strong></td>
<td>The name to use when binding to the LDAP server with <code>automationhub_ldap_bind_password</code>.</td>
</tr>
<tr>
<td><strong>automationhub_ldap_bind_password</strong></td>
<td>Required</td>
</tr>
<tr>
<td></td>
<td>The password to use with <code>automationhub_ldap_bind_dn</code>.</td>
</tr>
<tr>
<td><strong>automationhub_ldap_group_search_base_dn</strong></td>
<td>An LDAPSearch object that finds all LDAP groups that users might belong to. If your configuration makes any references to LDAP groups, this and <code>automationhub_ldap_group_type</code> must be set.</td>
</tr>
<tr>
<td></td>
<td>Default = <strong>None</strong></td>
</tr>
<tr>
<td>Variable</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>automationhub_ldap_group_search_filter</td>
<td>Optional</td>
</tr>
<tr>
<td></td>
<td>Search filter for finding group membership.</td>
</tr>
<tr>
<td></td>
<td>Variable identifies what objectClass type to use for mapping groups with automation hub and LDAP. Used for installing automation hub with LDAP.</td>
</tr>
<tr>
<td></td>
<td>Default = (objectClass=Group)</td>
</tr>
<tr>
<td>automationhub_ldap_group_search_scope</td>
<td>Optional</td>
</tr>
<tr>
<td></td>
<td>Scope to search for groups in an LDAP tree using the django framework for LDAP authentication. Used for installing automation hub with LDAP.</td>
</tr>
<tr>
<td></td>
<td>Default = SUBTREE</td>
</tr>
<tr>
<td>automationhub_ldap_group_type_class</td>
<td>Optional</td>
</tr>
<tr>
<td></td>
<td>Variable identifies the group type used during group searches within the django framework for LDAP authentication. Used for installing automation hub with LDAP.</td>
</tr>
<tr>
<td></td>
<td>Default = django_auth_ldap.config:GroupOfNamesType</td>
</tr>
<tr>
<td>automationhub_ldap_server_uri</td>
<td>The URI of the LDAP server. This can be any URI that is supported by your underlying LDAP libraries.</td>
</tr>
<tr>
<td>automationhub_ldap_user_search_base_dn</td>
<td>An LDAPSearch object that locates a user in the directory. The filter parameter should contain the placeholder %(user)s for the username. It must return exactly one result for authentication to succeed.</td>
</tr>
<tr>
<td>automationhub_ldap_user-search_scope</td>
<td>Optional</td>
</tr>
<tr>
<td></td>
<td>Scope to search for users in an LDAP tree using django framework for LDAP authentication. Used for installing automation hub with LDAP.</td>
</tr>
<tr>
<td></td>
<td>Default = SUBTREE</td>
</tr>
</tbody>
</table>

A.3. AUTOMATION CONTROLLER VARIABLES
<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>admin_password</td>
<td>The password for an administration user to access the UI upon install completion.</td>
</tr>
<tr>
<td>automation_controller_main_url</td>
<td>For an alternative front end URL needed for SSO configuration, provide the URL.</td>
</tr>
<tr>
<td>automationcontroller_password</td>
<td>Password for your automation controller instance.</td>
</tr>
<tr>
<td>automationcontroller_username</td>
<td>Username for your automation controller instance.</td>
</tr>
<tr>
<td>nginx_http_port</td>
<td>The nginx HTTP server listens for inbound connections. Default = 80</td>
</tr>
<tr>
<td>nginx_https_port</td>
<td>The nginx HTTPS server listens for secure connections. Default = 443</td>
</tr>
<tr>
<td>nginx_hsts_max_age</td>
<td>This variable specifies how long, in seconds, the system should be considered as a HTTP Strict Transport Security (HSTS) host. That is, how long HTTPS is used exclusively for communication. Default = 63072000 seconds, or two years.</td>
</tr>
<tr>
<td>nginx_tls_protocols</td>
<td>Defines support for ssl_protocols in Nginx. Default = TLSv1.2</td>
</tr>
<tr>
<td>node_state</td>
<td>Optional</td>
</tr>
</tbody>
</table>

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.
<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>node_type</td>
<td>For [automationcontroller] group. Two valid node_types can be assigned for this group. A node_type=control implies that the node only runs project and inventory updates, but not regular jobs. A node_type=hybrid has the ability to 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.</td>
</tr>
<tr>
<td>peers</td>
<td>Optional The peers variable is used to indicate which nodes a specific host or group connects to. Wherever the peers variable is defined, an outbound connection will be established to the specific host or group. This variable is used to add tcp-peer entries in the receptor.conf file used for establishing network connections with other nodes. See Peering. The peers variable can be a comma-separated list of hosts and/or groups from the inventory. This is resolved into a set of hosts that is used to construct the receptor.conf file.</td>
</tr>
<tr>
<td>pg_database</td>
<td>The name of the postgres database. Default = awx.</td>
</tr>
<tr>
<td>pg_host</td>
<td>The postgresQL host, which can be an externally managed database.</td>
</tr>
<tr>
<td>Variable</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>pg_password</strong></td>
<td>The password for the postgreSQL database.</td>
</tr>
<tr>
<td></td>
<td>Do not use special characters for <strong>pg_password</strong>. They can cause the password to fail.</td>
</tr>
<tr>
<td></td>
<td><strong>NOTE</strong></td>
</tr>
<tr>
<td></td>
<td>You no longer have to provide a <strong>pg_hashed_password</strong> in your inventory file at the time of installation because PostgreSQL 13 can now store user passwords more securely.</td>
</tr>
<tr>
<td></td>
<td>When you supply <strong>pg_password</strong> in the inventory file for the installer, PostgreSQL uses the SCRAM-SHA-256 hash to secure that password as part of the installation process.</td>
</tr>
<tr>
<td><strong>pg_port</strong></td>
<td>The postgreSQL port to use.</td>
</tr>
<tr>
<td></td>
<td>Default = 5432</td>
</tr>
<tr>
<td><strong>pg_ssl_mode</strong></td>
<td>One of <strong>prefer</strong> or <strong>verify-full</strong>.</td>
</tr>
<tr>
<td></td>
<td>Set to <strong>verify-full</strong> for client-side enforced SSL.</td>
</tr>
<tr>
<td></td>
<td>Default = <strong>prefer</strong>.</td>
</tr>
<tr>
<td><strong>pg_username</strong></td>
<td>Your postgres database username.</td>
</tr>
<tr>
<td></td>
<td>Default = <strong>awx</strong>.</td>
</tr>
<tr>
<td><strong>postgres_ssl_cert</strong></td>
<td>location of postgres ssl certificate.</td>
</tr>
<tr>
<td></td>
<td>/path/to/pgsql_ssl.cert</td>
</tr>
<tr>
<td><strong>postgres_ssl_key</strong></td>
<td>location of postgres ssl key.</td>
</tr>
<tr>
<td></td>
<td>/path/to/pgsql_ssl.key</td>
</tr>
<tr>
<td><strong>postgres_use_cert</strong></td>
<td>Location of postgres user certificate.</td>
</tr>
<tr>
<td></td>
<td>/path/to/pgsql.crt</td>
</tr>
<tr>
<td><strong>postgres_use_key</strong></td>
<td>Location of postgres user key.</td>
</tr>
<tr>
<td></td>
<td>/path/to/pgsql.key</td>
</tr>
<tr>
<td><strong>postgres_use_ssl</strong></td>
<td>If postgres is to use SSL.</td>
</tr>
</tbody>
</table>
### A.4. ANSIBLE VARIABLES

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

Additional information on variables specific to certain plugins can be found at [https://docs.ansible.com/ansible-core/devel/collections/ansible/builtin/index.html](https://docs.ansible.com/ansible-core/devel/collections/ansible/builtin/index.html)

A list of global configuration options can be found at [https://docs.ansible.com/ansible-core/devel/reference_appendices/config.html](https://docs.ansible.com/ansible-core/devel/reference_appendices/config.html)

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>receptor_listener_port</strong></td>
<td>Port to use for receptor connection.</td>
</tr>
<tr>
<td></td>
<td>Default = 27199.</td>
</tr>
<tr>
<td><strong>supervisor_start_retry_count</strong></td>
<td>When specified (no default value exists), adds startretry = &lt;value specified&gt; to the supervisor config file (/etc/supervisord.d/tower.ini). See program:x Section Values for further explanation about startretry.</td>
</tr>
<tr>
<td><strong>web_server_ssl_cert</strong></td>
<td>Optional /path/to/webserver.cert</td>
</tr>
<tr>
<td></td>
<td>Same as automationhub_ssl_cert but for web server UI and API.</td>
</tr>
<tr>
<td><strong>web_server_ssl_key</strong></td>
<td>Optional /path/to/webserver.key</td>
</tr>
<tr>
<td></td>
<td>Same as automationhub_server_ssl_key but for web server UI and API.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ansible_connection</strong></td>
<td>The connection plugin used for the task on the target host.</td>
</tr>
<tr>
<td></td>
<td>This can be the name of any of ansible connection plugin. SSH protocol types are smart, ssh or paramiko.</td>
</tr>
<tr>
<td></td>
<td>Default = smart</td>
</tr>
<tr>
<td><strong>ansible_host</strong></td>
<td>The ip or name of the target host to use instead of inventory_hostname.</td>
</tr>
<tr>
<td>Variable</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ansible_port</td>
<td>The connection port number, if not, the default (22 for ssh).</td>
</tr>
<tr>
<td>ansible_user</td>
<td>The user name to use when connecting to the host.</td>
</tr>
<tr>
<td>ansible_password</td>
<td>The password to use to authenticate to the host.</td>
</tr>
<tr>
<td></td>
<td>Never store this variable in plain text.</td>
</tr>
<tr>
<td></td>
<td>Always use a vault.</td>
</tr>
<tr>
<td>ansible_ssh_private_key_file</td>
<td>Private key file used by ssh. Useful if using multiple keys and you do not want to use an SSH agent.</td>
</tr>
<tr>
<td>ansible_ssh_common_args</td>
<td>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).</td>
</tr>
<tr>
<td>ansible_sftp_extra_args</td>
<td>This setting is always appended to the default sftp command line.</td>
</tr>
<tr>
<td>ansible_scp_extra_args</td>
<td>This setting is always appended to the default scp command line.</td>
</tr>
<tr>
<td>ansible_ssh_extra_args</td>
<td>This setting is always appended to the default ssh command line.</td>
</tr>
<tr>
<td>ansible_ssh_pipelining</td>
<td>Determines if SSH pipelining is used. This can override the pipelining setting in ansible.cfg. If using SSH key-based authentication, then the key must be managed by an SSH agent.</td>
</tr>
<tr>
<td>ansible_ssh_executable</td>
<td>(added in version 2.2)</td>
</tr>
<tr>
<td></td>
<td>This setting overrides the default behavior to use the system ssh. This can override the ssh_executable setting in ansible.cfg.</td>
</tr>
<tr>
<td>ansible_shell_type</td>
<td>The shell type of the target system. You should 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.</td>
</tr>
</tbody>
</table>
### 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`.

You should only change if it is not possible to use `/bin/sh`, that is, if `/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.

### A.5. EVENT-DRIVEN ANSIBLE CONTROLLER VARIABLES

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>automationedacontroller_admin_password</code></td>
<td>The admin password used by the Event-Driven Ansible controller instance.</td>
</tr>
<tr>
<td><code>automationedacontroller_admin_username</code></td>
<td>Username used by django to identify and create the admin superuser in Event-Driven Ansible controller. Default = <code>admin</code></td>
</tr>
<tr>
<td><code>automationedacontroller_admin_email</code></td>
<td>Email address used by django for the admin user for Event-Driven Ansible controller. Default = <code>admin@example.com</code></td>
</tr>
<tr>
<td><code>automationedacontroller_disable_https</code></td>
<td>Boolean flag to disable HTTPS Event-Driven Ansible controller. Default = <code>false</code></td>
</tr>
<tr>
<td><code>automationedacontroller_disable_hsts</code></td>
<td>Boolean flag to disable HSTS Event-Driven Ansible controller. Default = <code>false</code></td>
</tr>
<tr>
<td><code>automationedacontroller_user_headers</code></td>
<td>List of additional nginx headers to add to Event-Driven Ansible controller’s nginx configuration. Default = empty list</td>
</tr>
<tr>
<td>Variable</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td>automationedacontroller_nginx_tls_files_remote</td>
<td>Boolean flag to specify whether cert sources are on the remote host (true) or local (false). Default = false</td>
</tr>
<tr>
<td>automationedacontroller_allowed_hostnames</td>
<td>List of additional addresses to enable for user access to Event-Driven Ansible controller. Default = empty list</td>
</tr>
<tr>
<td>automationedacontroller_controller_verify_ssl</td>
<td>Boolean flag used to verify Automation Controller’s web certificates when making calls from Event-Driven Ansible controller. Verified is true; not verified is false. Default = false</td>
</tr>
<tr>
<td>automationedacontroller_gunicorn_workers</td>
<td>Number of workers for the API served through gunicorn. Default = (# of cores or threads) * 2 + 1</td>
</tr>
<tr>
<td>automationedacontroller_pg_database</td>
<td>The postgres database used by Event-Driven Ansible controller. Default = automationedacontroller.</td>
</tr>
<tr>
<td>automationedacontroller_pg_host</td>
<td>The hostname of the postgres database used by Event-Driven Ansible controller, which can be an externally managed database.</td>
</tr>
<tr>
<td>automationedacontroller_pg_password</td>
<td>The password for the postgres database used by Event-Driven Ansible controller. Do not use special characters for edacontroller_pg_password. They can cause the password to fail.</td>
</tr>
<tr>
<td>automationedacontroller_pg_port</td>
<td>The port number of the postgres database used by Event-Driven Ansible controller. Default = 5432</td>
</tr>
<tr>
<td>automationedacontroller_pg_username</td>
<td>The username for your Event-Driven Ansible controller postgres database. Default = automationedacontroller.</td>
</tr>
<tr>
<td>Variable</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>automationedacontroller_rq_workers</code></td>
<td>Number of rq workers (Python processes that run in the background) used by Event-Driven Ansible controller. Default = 2 * (# of cores or threads) + 1</td>
</tr>
</tbody>
</table>