Red Hat Ansible Automation Platform 2.4

Red Hat Ansible Automation Platform planning guide

Plan for installation of Ansible Automation Platform

Last Updated: 2025-01-15

Red Hat Ansible Automation Platform 2.4 Red Hat Ansible Automation Platform planning guide

Plan for installation of Ansible Automation Platform

Legal Notice

Abstract

This guide provides requirements, options, and recommendations for installing Red Hat Ansible Automation Platform.

Table of Contents

PREFACE

Thank you for your interest in Red Hat Ansible Automation Platform. Ansible Automation Platform is a commercial offering that helps teams manage complex multitiered deployments by adding control, knowledge, and delegation to Ansible-powered environments.

Use the information in this guide to plan your Red Hat Ansible Automation Platform installation.

PROVIDING FEEDBACK ON RED HAT DOCUMENTATION

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

CHAPTER 1. PLANNING YOUR RED HAT ANSIBLE AUTOMATION PLATFORM INSTALLATION

Red Hat Ansible Automation Platform is supported on both Red Hat Enterprise Linux and Red Hat OpenShift. Use this guide to plan your Red Hat Ansible Automation Platform installation on Red Hat Enterprise Linux.

To install Red Hat Ansible Automation Platform on your Red Hat OpenShift Container Platform environment, see Deploying the Red Hat Ansible Automation Platform operator on OpenShift Container Platform.

CHAPTER 2. RED HAT ANSIBLE AUTOMATION PLATFORM ARCHITECTURE

As a modular platform, Ansible Automation Platform provides the flexibility to easily integrate components and customize your deployment to best meet your automation requirements. The following section provides a comprehensive architectural example of an Ansible Automation Platform deployment.

2.1. EXAMPLE ANSIBLE AUTOMATION PLATFORM ARCHITECTURE

The Red Hat Ansible Automation Platform 2.4 reference architecture provides an example setup of a standard deployment of Ansible Automation Platform using automation mesh on Red Hat Enterprise Linux. The deployment shown takes advantage of the following components to provide a simple, secure and flexible method of handling your automation workloads, a central location for content collections, and automated resolution of IT requests.

Automation controller

Provides the control plane for automation through its UI, Restful API, RBAC workflows and CI/CD integrations.

Automation mesh

Is an overlay network that provides the ability to ease the distribution of work across a large and dispersed collection of workers through nodes that establish peer-to-peer connections with each other using existing networks.

Private automation hub

Provides automation developers the ability to collaborate and publish their own automation content and streamline delivery of Ansible code within their organization.

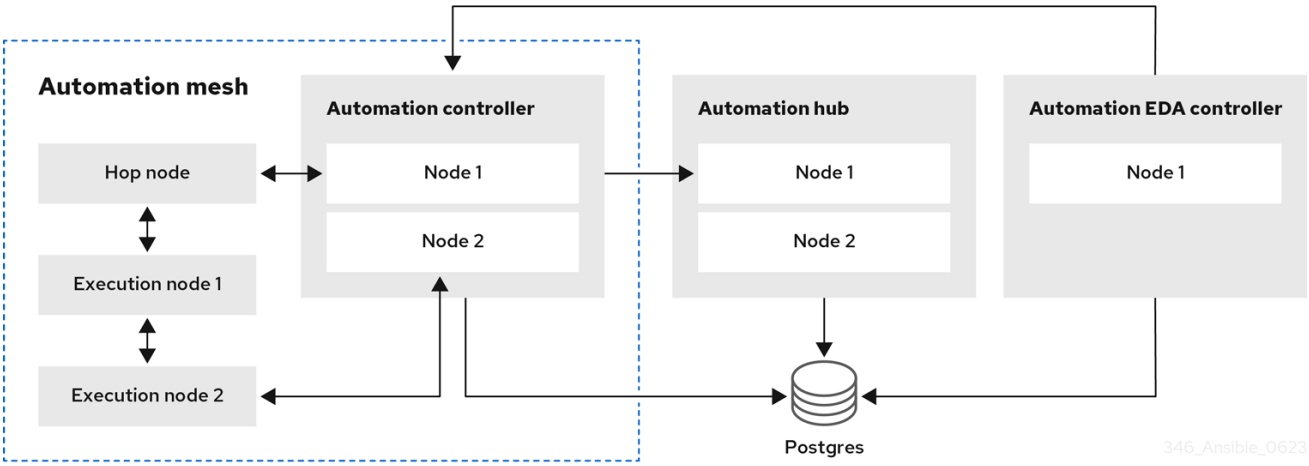
Event-Driven Ansible

Provides the event-handling capability needed to automate time-consuming tasks and respond to changing conditions in any IT domain.

The architecture for this example consists of the following:

- A two node automation controller cluster
- An optional hop node to connect automation controller to execution nodes
- A two node automation hub cluster
- A single node Event-Driven Ansible controller cluster
- A single PostgreSQL database connected to the automation controller, automation hub, and Event-Driven Ansible controller clusters
- Two execution nodes per automation controller cluster

Figure 2.1. Example Ansible Automation Platform 2.4 architecture



CHAPTER 3. RED HAT ANSIBLE AUTOMATION PLATFORM COMPONENTS

Ansible Automation Platform is a modular platform composed of separate components that can be connected together to meet your deployment needs. Ansible Automation Platform deployments start with automation controller which is the enterprise framework for controlling, securing, and managing Ansible automation with a user interface (UI) and RESTful application programming interface (API). Then, you can add to your deployment any combination of the following automation platform components:

3.1. ANSIBLE AUTOMATION HUB

Ansible automation hub is a repository for certified content of Ansible Content Collections. It is the centralized repository for Red Hat and its partners to publish content, and for customers to discover certified, supported Ansible Content Collections. Red Hat Ansible Certified Content provides users with content that has been tested and is supported by Red Hat.

3.2. PRIVATE AUTOMATION HUB

Private automation hub provides both disconnected and on-premise solutions for synchronizing content. You can synchronize collections and execution environment images from Red Hat cloud automation hub, storing and serving your own custom automation collections and execution images. You can also use other sources such as Ansible Galaxy or other container registries to provide content to your private automation hub. Private automation hub can integrate into your enterprise directory and your CI/CD pipelines.

3.3. HIGH AVAILABILITY AUTOMATION HUB

A high availability (HA) configuration increases reliability and scalablility for automation hub deployments.

HA deployments of automation hub have multiple nodes that concurrently run the same service with a load balancer distributing workload (an "active-active" configuration). This configuration eliminates single points of failure to minimize service downtime and allows you to easily add or remove nodes to meet workload demands.

3.4. EVENT-DRIVEN ANSIBLE CONTROLLER

The Event-Driven Ansible controller is the interface for event-driven automation and introduces automated resolution of IT requests. Event-Driven Ansible controller helps you connect to sources of events and act on those events by using rulebooks. This technology improves IT speed and agility, and enables consistency and resilience. With Event-Driven Ansible, you can:

- Automate decision making
- Use many event sources
- Implement event-driven automation within and across many IT use cases

Additional resources

- Getting Started with Event-Driven Ansible Guide .

3.5. AUTOMATION MESH

Automation mesh is an overlay network intended to ease the distribution of work across a large and dispersed collection of workers through nodes that establish peer-to-peer connections with each other using existing networks.

Automation mesh provides:

- Dynamic cluster capacity that scales independently, allowing you to create, register, group, ungroup and deregister nodes with minimal downtime.
- Control and execution plane separation that enables you to scale playbook execution capacity independently from control plane capacity.
- Deployment choices that are resilient to latency, reconfigurable without outage, and that dynamically re-reroute to choose a different path when outages exist.
- Mesh routing changes.
- Connectivity that includes bi-directional, multi-hopped mesh communication possibilities which are Federal Information Processing Standards (FIPS) compliant.

3.6. AUTOMATION EXECUTION ENVIRONMENTS

Automation execution environments are container images on which all automation in Red Hat Ansible Automation Platform is run. They provide a solution that includes the Ansible execution engine and hundreds of modules that help users automate all aspects of IT environments and processes. Automation execution environments automate commonly used operating systems, infrastructure platforms, network devices, and clouds.

3.7. ANSIBLE GALAXY

Ansible Galaxy is a hub for finding, reusing, and sharing Ansible content. Community-provided Galaxy content, in the form of prepackaged roles, can help start automation projects. Roles for provisioning infrastructure, deploying applications, and completing other tasks can be dropped into Ansible Playbooks and be applied immediately to customer environments.

3.8. AUTOMATION CONTENT NAVIGATOR

Automation content navigator is a *textual user interface* (TUI) that becomes the primary command line interface into the automation platform, covering use cases from content building, running automation locally in an execution environment, running automation in Ansible Automation Platform, and providing the foundation for future *integrated development environments* (IDEs).

CHAPTER 4. SYSTEM REQUIREMENTS

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

Prerequisites

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

4.1. RED HAT ANSIBLE AUTOMATION PLATFORM SYSTEM REQUIREMENTS

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

Table 4.1. Base system

Requirement	Required	Notes
Subscription	Valid Red Hat Ansible Automation Platform	
OS	Red Hat Enterprise Linux 8.8 or later 64-bit (x86, ppc64le, s390x, aarch64), or Red Hat Enterprise Linux 9.0 or later 64-bit (x86, ppc64le, s390x, aarch64)	Red Hat Ansible Automation Platform is also supported on OpenShift, see Deploying the Red Hat Ansible Automation Platform operator on OpenShift Container Platform for more information.
Ansible-core	Ansible-core version 2.14 or later	Ansible Automation Platform includes execution environments that contain ansible-core 2.15.
Python	3.9 or later	
Browser	A currently supported version of Mozilla FireFox or Google Chrome	
Database	PostgreSQL version 13	

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

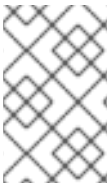
- Ensure that the network ports and protocols listed in *Table 5.3. Automation Hub* are available for successful connection and download of collections from automation hub or Ansible Galaxy server.
- Disable SSL inspection either when using self-signed certificates or for the Red Hat domains.

NOTE

The requirements for systems managed by Ansible Automation Platform are the same as for Ansible. See Installing Ansible in the Ansible Community Documentation.

Additional notes for Red Hat Ansible Automation Platform requirements

- Red Hat Ansible Automation Platform depends on Ansible Playbooks and requires the installation of the latest stable version of ansible-core. You can download ansible-core manually or download it automatically as part of your installation of Red Hat Ansible Automation Platform.
- For new installations, automation controller installs the latest release package of ansible-core.
- If performing a bundled Ansible Automation Platform installation, the installation setup.sh script attempts to install ansible-core (and its dependencies) from the bundle for you.
- If you have installed Ansible manually, the Ansible Automation Platform installation setup.sh script detects that Ansible has been installed and does not attempt to reinstall it.



NOTE

You must install Ansible using a package manager such as **dnf**, and the latest stable version of the package manager must be installed for Red Hat Ansible Automation Platform to work properly. Ansible version 2.14 is required for versions 2.4 and later.

4.2. AUTOMATION CONTROLLER SYSTEM REQUIREMENTS

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

Use the following recommendations for node sizing:

NOTE

On control and hybrid nodes, allocate a minimum of 20 GB to **/var/lib/awx** for execution environment storage.

Execution nodes

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

NOTE

- The RAM and CPU resources stated might not be required for packages installed on an execution node but are the minimum recommended to handle the job load for a node to run an average number of jobs simultaneously.
- Recommended RAM and CPU node sizes are not supplied. The required RAM or CPU depends directly on the number of jobs you are running in that environment.

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

Table 4.2. Execution nodes

Requirement	Minimum required
RAM	16 GB
CPUs	4
Local disk	40GB minimum

Control nodes

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

Table 4.3. Control nodes

Requirement	Minimum required
RAM	16 GB
CPUs	4
Local disk	• 40GB minimum with at least 20GB available under /var/lib/awx • Storage volume must be rated for a minimum baseline of 1500 IOPS • Projects are stored on control and hybrid nodes, and for the duration of jobs, are also stored on execution nodes. If the cluster has many large projects, consider doubling the GB in /var/lib/awx/projects, to avoid disk space errors.

Hop nodes

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

Table 4.4. Hop nodes

Requirement	Minimum required
RAM	16 GB
CPUs	4
Local disk	40 GB

- Actual RAM requirements vary based on how many hosts automation controller will manage simultaneously (which is controlled by the **forks** parameter in the job template or the system **ansible.cfg** file). To avoid possible resource conflicts, Ansible recommends 1 GB of memory per 10 forks and 2 GB reservation for automation controller. For more information, see Automation controller capacity determination and job impact. If **forks** is set to 400, 42 GB of memory is recommended.
- Automation controller hosts check if **umask** is set to 0022. If not, the setup fails. Set **umask=0022** to avoid this error.
- A larger number of hosts can be addressed, but if the fork number is less than the total host count, more passes across the hosts are required. You can avoid these RAM limitations by using any of the following approaches:
  - Use rolling updates.
  - Use the provisioning callback system built into automation controller, where each system requesting configuration enters a queue and is processed as quickly as possible.
  - In cases where automation controller is producing or deploying images such as AMIs.

Additional resources

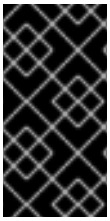
- For more information about obtaining an automation controller subscription, see Importing a subscription.
- For questions, contact Ansible support through the Red Hat Customer Portal .

4.3. AUTOMATION HUB SYSTEM REQUIREMENTS

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

Automation hub has the following system requirements:

Requirement	Required	Notes
RAM	8 GB minimum	• 8 GB RAM (minimum and recommended for Vagrant trial installations) • 8 GB RAM (minimum for external standalone PostgreSQL databases) • For capacity based on forks in your configuration, see Automation controller capacity determination and job impact.
CPUs	2 minimum	For capacity based on forks in your configuration, see Automation controller capacity determination and job impact.
Local disk	60 GB disk	Dedicate a minimum of 40GB to **/var** for collection storage.



IMPORTANT

Ansible automation execution nodes and automation hub system requirements are different and might not meet your network's needs. The general formula for determining how much memory you need is: Total control capacity = Total Memory in MB / Fork size in MB.

NOTE

Private automation hub

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

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

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

For information about inventory file variables, see Inventory file variables in the *Red Hat Ansible Automation Platform Installation Guide*.

4.3.1. High availability automation hub requirements

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

4.3.1.1. Required shared filesystem

A high availability automation hub requires you to have a shared file system, such as NFS, already installed in your environment. Before you run the Red Hat Ansible Automation Platform installer, verify that you installed the **/var/lib/pulp** directory across your cluster as part of the shared file system installation. The Red Hat Ansible Automation Platform installer returns an error if **/var/lib/pulp** is not detected in one of your nodes, causing your high availability automation hub setup to fail.

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

4.3.1.2. Installing firewalld for network storage

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

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

1. Install the **firewalld** daemon:

```
$ dnf install firewalld
```

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

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

NOTE

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

3. Reload to apply the configuration:

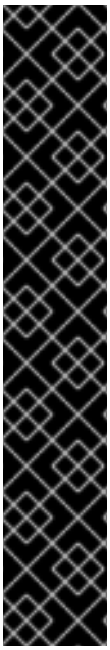
```
$ firewall-cmd --reload
```

4.4. EVENT-DRIVEN ANSIBLE CONTROLLER SYSTEM REQUIREMENTS

The Event-Driven Ansible controller is a single-node system capable of handling a variable number of long-running processes (such as rulebook activations) on-demand, depending on the number of CPU cores. Use the following minimum requirements to run, by default, a maximum of 12 simultaneous activations:

Requirement	Required
RAM	16 GB
CPUs	4

Requirement	Required
Local disk	40 GB minimum



IMPORTANT

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

4.5. POSTGRESQL REQUIREMENTS

Red Hat Ansible Automation Platform uses PostgreSQL 13. PostgreSQL user passwords are hashed with SCRAM-SHA-256 secure hashing algorithm before storing in the database.

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

Table 4.5. Database

Service	Required	Notes

Service	Required	Notes
Database	• 20 GB dedicated hard disk space • 4 CPUs • 16 GB RAM	• 150 GB+ recommended • Storage volume must be rated for a high baseline IOPS (1500 or more). • All automation controller data is stored in the database. Database storage increases with the number of hosts managed, number of jobs run, number of facts stored in the fact cache, and number of tasks in any individual job. For example, a playbook run every hour (24 times a day) across 250 hosts, with 20 tasks, will store over 800000 events in the database every week. • If not enough space is reserved in the database, the old job runs and facts must be cleaned on a regular basis. For more information, see Management Jobs in the *Automation Controller Administration Guide*.

NOTE

PostgreSQL versions older than v10 might not have ICU support. You must build your PostgreSQL server with ICU support or you may encounter unexpected errors.

PostgreSQL configurations

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

Additional resources

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

4.5.1. Setting up an external (customer supported) database

IMPORTANT

When using an external database with Ansible Automation Platform, you must create and maintain that database. Ensure that you clear your external database when uninstalling Ansible Automation Platform.

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

Procedure

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

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

Where:

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

Specifies the host name of the machine on which the server is running. If the value begins with a slash, it is used as the directory for the Unix-domain socket.

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

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

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

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

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

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

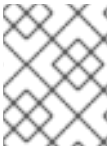
```
[database]
pg_host='db.example.com'
pg_port=5432
pg_database='awx'
pg_username='awx'
pg_password='redhat'
```

4. Run the installer.

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

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

6. Check the permission of the user, the user should have the **createDB** or administrator role.



NOTE

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

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

From Ansible Automation Platform 2.4, the database migration script uses **hstore** fields to store information, therefore the **hstore** extension to the automation hub PostgreSQL database must be enabled.

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

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

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

Procedure

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

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

Where the default value for **<automation hub database>** is **automationhub**.

Example output with hstore available:

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

Example output with hstore not available:

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

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

```
-
```

```
dnf install postgresql-contrib
```

3. Create the **hstore** PostgreSQL extension on the automation hub database with the following command:

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

The output of which is:

```
CREATE EXTENSION
```

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

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

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

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

Prerequisites

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

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

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

Procedure

1. Run a random write test:

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

2. Run a random read test:

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

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

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

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

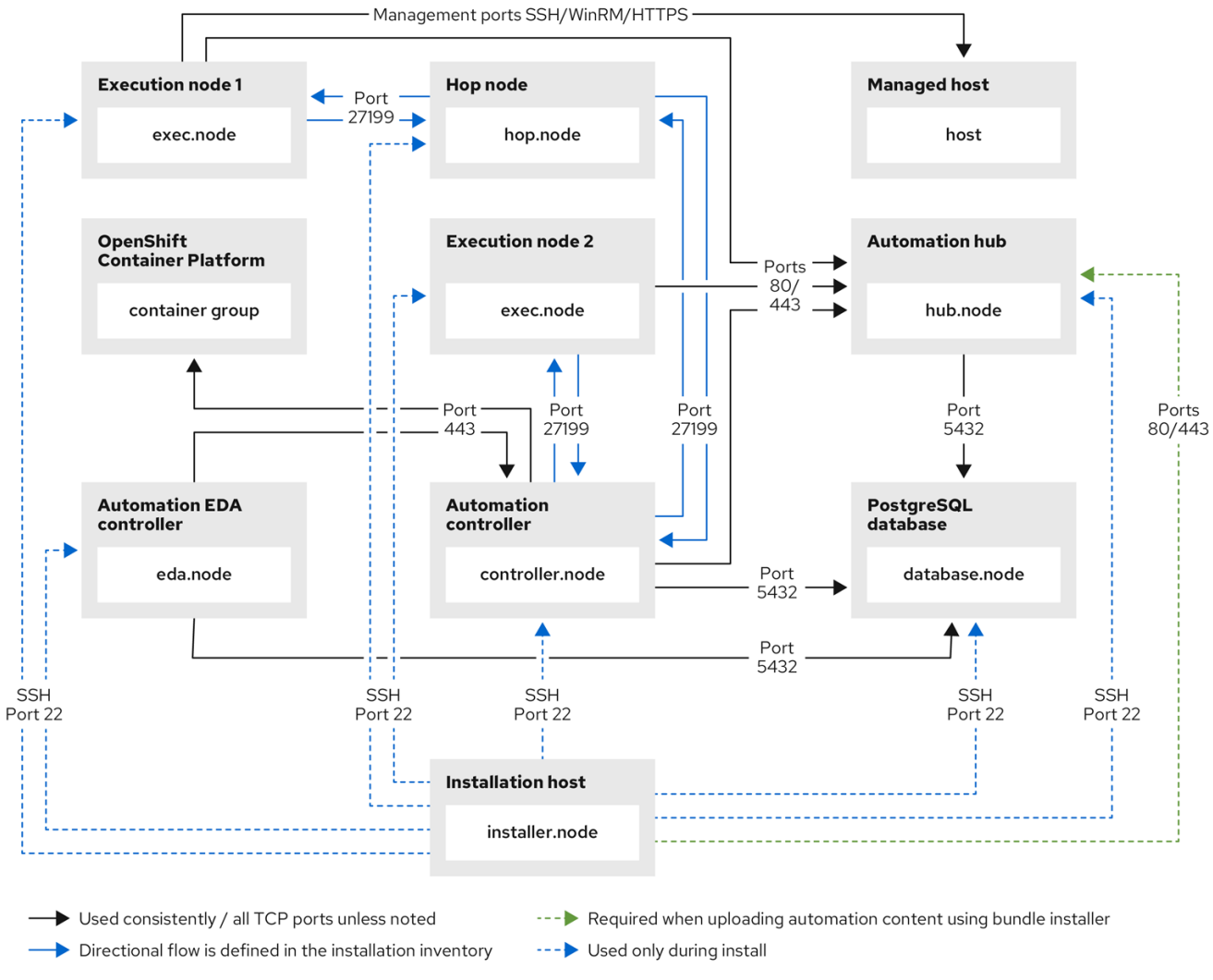
   You must review, monitor, and revisit the log files according to your own business requirements, application workloads, and new demands.

CHAPTER 5. NETWORK PORTS AND PROTOCOLS

Red Hat Ansible Automation Platform uses several ports to communicate with its services. These ports must be open and available for incoming connections to the Red Hat Ansible Automation Platform server in order for it to work. Ensure that these ports are available and are not blocked by the server firewall.

The following architectural diagram is an example of a fully deployed Ansible Automation Platform with all possible components.

Figure 5.1. Ansible Automation Platform Network ports and protocols



The following table indicates the destination port and the direction of network traffic:

NOTE

The following default destination ports and installer inventory listed are configurable. If you choose to configure them to suit your environment, you might experience a change in behavior.

Table 5.1. Network ports and protocols

Port	Protocol	Service	Source	Destination	Required for	Installer Inventory Variable
22	TCP	SSH	Installer node	Automation hub	Installation (temporary)	**ansible_port**
22	TCP	SSH	Installer node	Controller node	Installation (temporary)	**ansible_port**
22	TCP	SSH	Installer node	EDA node	Installation (temporary)	**ansible_port**
22	TCP	SSH	Installer node	Execution node	Installation (temporary)	**ansible_port**
22	TCP	SSH	Installer node	Hop node	Installation (temporary)	**ansible_port**
22	TCP	SSH	Installer node	Hybrid node	Installation (temporary)	**ansible_port**
22	TCP	SSH	Installer node	PostgreSQL database	Remote access during installation (temporary)	**pg_port**
80/443	TCP	HTTP/HTTPS	Installer node	Automation hub	Allows installer node to push the execution environment image to automation hub when using the bundle installer.	Fixed value
80/443	TCP	HTTP/HTTPS	Execution node	Automation hub	Allows execution nodes to pull the execution environment image from automation hub.	Fixed value
80/443	TCP	HTTP/HTTPS	Automation controller	Automation hub	For pulling collections and/or container images execution environments.	Fixed value
443	TCP	HTTPS	Controller node	Client	Web UI/API	**nginx_https_port**

Port	Protocol	Service	Source	Destination	Required for	Installer Inventory Variable
443	TCP	HTTPS	Controller node	OpenShift Container Platform	Only required when using container groups to run jobs.	Host name of OpenShift API server
5432	TCP	PostgreSQL	Controller node	PostgreSQL database	Open only if the internal database is used along with another component. Otherwise, this port should not be open.	**automationcontroller_pg_port**
5432	TCP	PostgreSQL	EDA node	PostgreSQL database	Open only if the internal database is used along with another component. Otherwise, this port should not be open.	**automationedacontroller_pg_port**
5432	TCP	PostgreSQL	Automation hub	PostgreSQL database	Open only if the internal database is used along with another component. Otherwise, this port should not be open.	**automationhub_pg_port**
27199	TCP	Receptor	Controller node	Execution node	Configurable Mesh nodes directly peered to controllers. Direct nodes involved. 27199 communication can be both ways (depending on installation inventory) for execution nodes	**receptor_listener_port** **peers**

Port	Protocol	Service	Source	Destination	Required for	Installer Inventory Variable
27199	TCP	Receptor	Controller node	Hop node	Configurable ENABLE connections from hop nodes to Receptor port if relayed through hop nodes.	**receptor_listener_port** **peers**
27199	TCP	Receptor	Controller node	Hybrid node	Configurable ENABLE connections from controllers to Receptor port if relayed through non-hop connected nodes.	**receptor_listener_port** **peers**
27199	TCP	Receptor	Execution node	Hop node	Configurable Mesh 27199 communication can be both ways (depending on installation inventory) for execution nodes ALLOW connection from controller(s) to Receptor port	**receptor_listener_port** **peers**
27199	TCP	Receptor	Execution node	Controller node	Configurable Mesh 27199 communication can be both ways (depending on installation inventory) for execution nodes ALLOW connection from controller(s) to Receptor port	**receptor_listener_port** **peers**

NOTE

- Hybrid nodes act as a combination of control and execution nodes, and therefore Hybrid nodes share the connections of both.
- If **receptor_listener_port** is defined, the machine also requires an available open port on which to establish inbound TCP connections, for example, 27199.
- It might be the case that some servers do not listen on receptor port (the default is 27199)
  Suppose you have a Control plane with nodes A, B, C, D

  The RPM installer creates a strongly connected peering between the control plane nodes with a least privileged approach and opens the tcp listener only on those nodes where it is required. All the receptor connections are bidirectional, so once the connection is created, the receptor can communicate in both directions.

  The following is an example peering set up for three controller nodes:

  Controller node A -→ Controller node B

  Controller node A -→ Controller node C

  Controller node B -→ Controller node C

  You can force the listener by setting

  receptor_listener=True

  However, a connection Controller B -→ A is likely to be rejected as that connection already exists.

  This means that nothing connects to Controller A as Controller A is creating the connections to the other nodes, and the following command does not return anything on Controller A:

  [root@controller1 ~]# ss -ntlp | grep 27199 [root@controller1 ~]#

Table 5.2. Red Hat Insights for Red Hat Ansible Automation Platform

URL	Required for
https://api.access.redhat.com:443	General account services, subscriptions
https://cert-api.access.redhat.com:443	Insights data upload
https://cert.console.redhat.com:443	Inventory upload and Cloud Connector connection
https://console.redhat.com:443	Access to Insights dashboard

Table 5.3. Automation Hub

URL	Required for
https://console.redhat.com:443	General account services, subscriptions
https://catalog.redhat.com:443	Indexing execution environments
https://sso.redhat.com:443	TCP
https://automation-hub-prd.s3.amazonaws.com:443https://automation-hub-prd.s3.us-east-2.amazonaws.com:443	Firewall access
https://galaxy.ansible.com:443	Ansible Community curated Ansible content
https://ansible-galaxy-ng.s3.dualstack.us-east-1.amazonaws.com:443	Dual Stack IPv6 endpoint for Community curated Ansible content repository
https://registry.redhat.io:443	Access to container images provided by Red Hat and partners
https://cert.console.redhat.com:443	Red Hat and partner curated Ansible Collections

Table 5.4. Execution Environments (EE)

URL	Required for
https://registry.redhat.io:443	Access to container images provided by Red Hat and partners
cdn.quay.io:443	Access to container images provided by Red Hat and partners
cdn01.quay.io:443	Access to container images provided by Red Hat and partners
cdn02.quay.io:443	Access to container images provided by Red Hat and partners
cdn03.quay.io:443	Access to container images provided by Red Hat and partners

IMPORTANT

Image manifests and filesystem blobs are served directly from **registry.redhat.io**. However, from 1 May 2023, filesystem blobs are served from **quay.io** instead. To avoid problems pulling container images, you must enable outbound connections to the listed **quay.io** hostnames.

This change should be made to any firewall configuration that specifically enables outbound connections to **registry.redhat.io**.

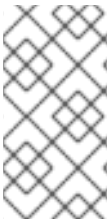
Use the hostnames instead of IP addresses when configuring firewall rules.

After making this change, you can continue to pull images from **registry.redhat.io**. You do not require a **quay.io** login, or need to interact with the **quay.io** registry directly in any way to continue pulling Red Hat container images.

For more information, see Firewall changes for container image pulls.

CHAPTER 6. ATTACHING YOUR RED HAT ANSIBLE AUTOMATION PLATFORM SUBSCRIPTION

You **must** have valid subscriptions attached on all nodes before installing Red Hat Ansible Automation Platform. Attaching your Ansible Automation Platform subscription provides access to subscription-only resources necessary to proceed with the installation.



NOTE

Attaching a subscription is unnecessary if you have enabled Simple Content Access Mode on your Red Hat account. Once enabled, you will need to register your systems to either Red Hat Subscription Management (RHSM) or Satellite before installing the Ansible Automation Platform. For more information, see Simple Content Access .

Procedure

1. Obtain the **pool_id** for your Red Hat Ansible Automation Platform subscription:

```
# subscription-manager list --available --all | grep "Ansible Automation Platform" -B 3 -A 6
```

NOTE

Do not use MCT4022 as a **pool_id** for your subscription because it can cause Ansible Automation Platform subscription attachment to fail.

Example

An example output of the **subsciption-manager list** command. Obtain the **pool_id** as seen in the **Pool ID:** section:

```
Subscription Name: Red Hat Ansible Automation, Premium (5000 Managed Nodes)
  Provides: Red Hat Ansible Engine
  Red Hat Ansible Automation Platform
  SKU: MCT3695
  Contract: ````
  Pool ID: <pool_id>
  Provides Management: No
  Available: 4999
  Suggested: 1
```

2. Attach the subscription:

```
# subscription-manager attach --pool=<pool_id>
```

You have now attached your Red Hat Ansible Automation Platform subscriptions to all nodes.

Verification

- Verify the subscription was successfully attached:

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

Troubleshooting

- If you are unable to locate certain packages that came bundled with the Ansible Automation Platform installer, or if you are seeing a ***Repositories disabled by configuration*** message, try enabling the repository by using the command:
  Red Hat Ansible Automation Platform 2.4 for RHEL 8

  ```
  subscription-manager repos --enable ansible-automation-platform-2.4-for-rhel-8-x86_64-rpms
  ```

  Red Hat Ansible Automation Platform 2.4 for RHEL 9

  ```
  subscription-manager repos --enable ansible-automation-platform-2.4-for-rhel-9-x86_64-rpms
  ```

CHAPTER 7. CHOOSING AND OBTAINING A RED HAT ANSIBLE AUTOMATION PLATFORM INSTALLER

Choose the Red Hat Ansible Automation Platform installer you need based on your Red Hat Enterprise Linux environment internet connectivity. Review the following scenarios to decide which Red Hat Ansible Automation Platform installer meets your needs.

7.1. INSTALLING WITH INTERNET ACCESS

Choose the Red Hat Ansible Automation Platform installer if your Red Hat Enterprise Linux environment is connected to the internet. Installing with internet access retrieves the latest required repositories, packages, and dependencies. Choose one of the following ways to set up your Ansible Automation Platform installer.

Tarball install

1. Navigate to the Red Hat Ansible Automation Platform download page.
2. Click **Download Now** for the **Ansible Automation Platform <latest-version> Setup**.
3. Extract the files:

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

RPM install

1. Install Ansible Automation Platform Installer Package
   v.2.4 for RHEL 8 for x86_64

```
$ sudo dnf install --enablerepo=ansible-automation-platform-2.4-for-rhel-8-x86_64-rpms
ansible-automation-platform-installer
```

   v.2.4 for RHEL 9 for x86-64

```
$ sudo dnf install --enablerepo=ansible-automation-platform-2.4-for-rhel-9-x86_64-rpms
ansible-automation-platform-installer
```

NOTE

dnf install enables the repo as the repo is disabled by default.

When you use the RPM installer, the files are placed under the **/opt/ansible-automation-platform/installer** directory.

7.2. INSTALLING WITHOUT INTERNET ACCESS

Use the Red Hat Ansible Automation Platform **Bundle** installer if you are unable to access the internet, or would prefer not to install separate components and dependencies from online repositories. Access to Red Hat Enterprise Linux repositories is still needed. All other dependencies are included in the tar archive.

Procedure

Procedure

1. Go to the Red Hat Ansible Automation Platform download page.
2. Click **Download Now** for the **Ansible Automation Platform <latest-version> Setup Bundle**
3. Extract the files:

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

CHAPTER 8. ABOUT THE INSTALLER INVENTORY FILE

Red Hat Ansible Automation Platform works against a list of managed nodes or hosts in your infrastructure that are logically organized, using an inventory file. You can use the Red Hat Ansible Automation Platform installer inventory file to specify your installation scenario and describe host deployments to Ansible. By using an inventory file, Ansible can manage a large number of hosts with a single command. Inventories also help you use Ansible more efficiently by reducing the number of command line options you have to specify.

The inventory file can be in one of many formats, depending on the inventory plugins that you have. The most common formats are **INI** and **YAML**. Inventory files listed in this document are shown in INI format.

The location of the inventory file depends on the installer you used. The following table shows possible locations:

Installer	Location
Bundle tar	**/ansible-automation-platform-setup-bundle-<latest-version>**
Non-bundle tar	**/ansible-automation-platform-setup-<latest-version>**
RPM	**/opt/ansible-automation-platform/installer**

You can verify the hosts in your inventory using the command:

```
ansible all -i <path-to-inventory-file. --list-hosts
```

Example inventory file

```
[automationcontroller]
host1.example.com
host2.example.com
Host4.example.com

[automationhub]
host3.example.com

[database]
Host5.example.com

[all:vars]
admin_password='<password>'

pg_host=''
pg_port=''

pg_database='awx'
pg_username='awx'
pg_password='<password>'
```

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

The first part of the inventory file specifies the hosts or groups that Ansible can work with.

8.1. GUIDELINES FOR HOSTS AND GROUPS

Databases

- When using an external database, ensure the **[database]** sections of your inventory file are properly set up.
- To improve performance, do not colocate the database and the automation controller on the same server.

Automation hub

- If there is an **[automationhub]** group, you must include the variables **automationhub_pg_host** and **automationhub_pg_port**.
- Add Ansible automation hub information in the **[automationhub]** group.
- Do not install Ansible automation hub and automation controller on the same node.
- Provide a reachable IP address or fully qualified domain name (FQDN) for the **[automationhub]** and **[automationcontroller]** hosts to ensure that users can synchronize and install content from Ansible automation hub and automation controller from a different node. The FQDN must not contain the _ symbol, as it will not be processed correctly in Skopeo. You may use the **-** symbol, as long as it is not at the start or the end of the host name.

  Do not use **localhost**.

Private automation hub

- Do not install private automation hub and automation controller on the same node.
- You can use the same PostgreSQL (database) instance, but they must use a different (database) name.
- If you install private automation hub from an internal address, and have a certificate which only encompasses the external address, it can result in an installation you cannot use as a container registry without certificate issues.

IMPORTANT

You must separate the installation of automation controller and Ansible automation hub because the **[database]** group does not distinguish between the two if both are installed at the same time.

If you use one value in **[database]** and both automation controller and Ansible automation hub define it, they would use the same database.

Automation controller

- Automation controller does not configure replication or failover for the database that it uses.
- automation controller works with any replication that you have.

Event-Driven Ansible controller

- Event-Driven Ansible controller must be installed on a separate server and cannot be installed on the same host as automation hub and automation controller.

Clustered installations

- When upgrading an existing cluster, you can also reconfigure your cluster to omit existing instances or instance groups. Omitting the instance or the instance group from the inventory file is not enough to remove them from the cluster. In addition to omitting instances or instance groups from the inventory file, you must also deprovision instances or instance groups before starting the upgrade. For more information, see Deprovisioning nodes or groups. Otherwise, omitted instances or instance groups continue to communicate with the cluster, which can cause issues with automation controller services during the upgrade.
- If you are creating a clustered installation setup, you must replace **[localhost]** with the hostname or IP address of all instances. Installers for automation controller and automation hub do not accept **[localhost]** All nodes and instances must be able to reach any others by using this hostname or address. You cannot use the localhost **ansible_connection=local** on one of the nodes. Use the same format for the host names of all the nodes.
  Therefore, this does not work:

```
[automationhub]
localhost ansible_connection=local
hostA
hostB.example.com
172.27.0.4
```

  Instead, use these formats:

```
[automationhub]
hostA
hostB
hostC
```

  or

```
[automationhub]
hostA.example.com
hostB.example.com
hostC.example.com
```

8.2. DEPROVISIONING NODES OR GROUPS

You can deprovision nodes and instance groups using the Ansible Automation Platform installer. Running the installer will remove all configuration files and logs attached to the nodes in the group.

NOTE

You can deprovision any hosts in your inventory except for the first host specified in the **[automationcontroller]** group.

To deprovision nodes, append **node_state=deprovision** to the node or group within the inventory file.

For example:

To remove a single node from a deployment:

```
[automationcontroller]
host1.example.com
host2.example.com
host4.example.com    node_state=deprovision
```

or

To remove an entire instance group from a deployment:

```
[instance_group_restrictedzone]
host4.example.com
host5.example.com

[instance_group_restrictedzone:vars]
node_state=deprovision
```

8.3. INVENTORY VARIABLES

The second part of the example inventory file, following **[all:vars]**, is a list of variables used by the installer. Using **all** means the variables apply to all hosts.

To apply variables to a particular host, use **[hostname:vars]**. For example, **[automationhub:vars]**.

8.4. RULES FOR DECLARING VARIABLES IN INVENTORY FILES

The values of string variables are declared in quotes. For example:

```
pg_database='awx'
pg_username='awx'
pg_password='<password>'
```

When declared in a **:vars** section, INI values are interpreted as strings. For example, **var=FALSE** creates a string equal to **FALSE**. Unlike host lines, **:vars** sections accept only a single entry per line, so everything after the **=** must be the value for the entry. Host lines accept multiple **key=value** parameters per line. Therefore they need a way to indicate that a space is part of a value rather than a separator. Values that contain whitespace can be quoted (single or double). For more information, see Python shlex parsing rules.

If a variable value set in an INI inventory must be a certain type (for example, a string or a boolean value), always specify the type with a filter in your task. Do not rely on types set in INI inventories when consuming variables.

NOTE

Consider using YAML format for inventory sources to avoid confusion on the actual type of a variable. The YAML inventory plugin processes variable values consistently and correctly.

If a parameter value in the Ansible inventory file contains special characters, such as #, { or }, you must double-escape the value (that is enclose the value in both single and double quotation marks).

For example, to use **mypasswordwith#hashsigns** as a value for the variable **pg_password**, declare it as **pg_password='"mypasswordwith#hashsigns"'** in the Ansible host inventory file.

8.5. SECURING SECRETS IN THE INVENTORY FILE

You can encrypt sensitive or secret variables with Ansible Vault. However, encrypting the variable names and the variable values makes it hard to find the source of the values. To circumvent this, you can encrypt the variables individually by using **ansible-vault encrypt_string**, or encrypt a file containing the variables.

Procedure

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

```
$ cat credentials.yml

admin_password: my_long_admin_pw
pg_password: my_long_pg_pw
registry_password: my_long_registry_pw
```

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

```
$ ansible-vault encrypt credentials.yml
New Vault password:
Confirm New Vault password:
Encryption successful
```

IMPORTANT

Store your encrypted vault password in a safe place.

3. Verify that the **credentials.yml** file is encrypted.

```
$ cat credentials.yml
$ANSIBLE_VAULT;1.1;
AES256363836396535623865343163333339613833363064653364656138313534353135303
764646165393765393063303065323466663330646232363065316666310a37306230313337
633963383130303334313534383962613632303761636632623932653062343839613639653
635643365616233313365363661663931386430 0a353239373433313339613465326339313 0
356335653534643565386536316334643438353464323766386235336136663261363433323
131633436393939646132656164333634306335343039356462646330343839663362323033
65383763
```

4. Run **setup.sh** for installation of Ansible Automation Platform 2.4 and pass both **credentials.yml** and the **--ask-vault-pass option**.

```
$ ANSIBLE_BECOME_METHOD='sudo' ANSIBLE_BECOME=True
ANSIBLE_HOST_KEY_CHECKING=False ./setup.sh -e @credentials.yml -- --ask-vault-pass
```

8.6. ADDITIONAL INVENTORY FILE VARIABLES

You can further configure your Red Hat Ansible Automation Platform installation by including additional variables in the inventory file. These configurations add optional features for managing your Red Hat Ansible Automation Platform. Add these variables by editing the inventory file using a text editor.

A table of predefined values for inventory file variables can be found in Inventory file variables in the *Red Hat Ansible Automation Platform Installation Guide* .

CHAPTER 9. SUPPORTED INSTALLATION SCENARIOS

Red Hat supports the following installations scenarios for Red Hat Ansible Automation Platform:

Additional resources

To edit inventory file parameters to specify a supported installation scenario, see Inventory file examples based on installation scenarios in the *Red Hat Ansible Automation Platform Installation Guide* .

9.1. STANDALONE AUTOMATION CONTROLLER WITH A DATABASE ON THE SAME NODE, OR A NON-INSTALLER MANAGED DATABASE

This scenario includes installation of automation controller, including the web front end, REST API backend, and database on a single machine. It installs PostgreSQL, and configures the automation controller to use that as its database. This is considered the standard automation controller installation scenario.

9.2. STANDALONE AUTOMATION CONTROLLER WITH AN EXTERNAL MANAGED DATABASE

This scenario includes installation of the automation controller server on a single machine and configures communication with a remote PostgreSQL instance as its database. This remote PostgreSQL can be a server you manage, or can be provided by a cloud service such as Amazon RDS.

9.3. SINGLE EVENT-DRIVEN ANSIBLE CONTROLLER NODE WITH INTERNAL DATABASE

This scenario includes installation of Event-Driven Ansible controller on a single machine with an internal database. It installs an installer managed PostgreSQL that is similar to the automation controller installation scenario.

IMPORTANT

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

Additional resources

- Single automation controller, single automation hub, and single Event-Driven Ansible controller node with external (installer managed) database
- Appendix A.5. Event-Driven Ansible controller variables

9.4. STANDALONE AUTOMATION HUB WITH A DATABASE ON THE SAME NODE, OR A NON-INSTALLER MANAGED DATABASE

This scenario includes installation of automation hub, including the web frontend, REST API backend, and database on a single machine. It installs PostgreSQL, and configures the automation hub to use that as its database.

9.5. STANDALONE AUTOMATION HUB WITH AN EXTERNAL MANAGED DATABASE

This scenario includes installation of the automation hub server on a single machine, and installs a remote PostgreSQL database, managed by the Red Hat Ansible Automation Platform installer.

9.6. PLATFORM INSTALLATION WITH A DATABASE ON THE AUTOMATION CONTROLLER NODE, OR NON-INSTALLER MANAGED DATABASE

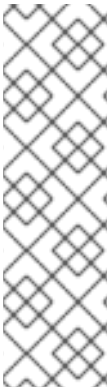
This scenario includes installation of automation controller and automation hub with a database on the automation controller node, or a non-installer managed database.

9.7. PLATFORM INSTALLATION WITH AN EXTERNAL MANAGED DATABASE

This scenario includes installation of automation controller and automation hub and configures communication with a remote PostgreSQL instance as its database. This remote PostgreSQL can be a server you manage, or can be provided by a cloud service such as Amazon RDS.

9.8. MULTI-MACHINE CLUSTER INSTALLATION WITH AN EXTERNAL MANAGED DATABASE

This scenario includes installation of multiple automation controller nodes and an automation hub instance and configures communication with a remote PostgreSQL instance as its database. This remote PostgreSQL can be a server you manage, or can be provided by a cloud service such as Amazon RDS. In this scenario, all automation controller are active and can execute jobs, and any node can receive HTTP requests.



NOTE

- Running in a cluster setup requires any database that automation controller uses to be external—PostgreSQL must be installed on a machine that is not one of the primary or secondary tower nodes. When in a redundant setup, the remote PostgreSQL version requirements is **PostgreSQL 13**.
    - See Clustering for more information on configuring a clustered setup.
- Provide a reachable IP address for the **[automationhub]** host to ensure users can sync content from Private Automation Hub from a different node.