Red Hat Ansible Automation Platform 2.5

Planning your installation

Plan for installation of Ansible Automation Platform

Last Updated: 2024-12-02

Red Hat Ansible Automation Platform 2.5 Planning your installation

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 Installing on OpenShift Container Platform .

CHAPTER 2. RED HAT ANSIBLE AUTOMATION PLATFORM ARCHITECTURE

Deploy all components of Ansible Automation Platform so that all features and capabilities are available for use without the need to take further action.

Red Hat tests the installation of Ansible Automation Platform 2.5 based on a defined set of infrastructure topologies or reference architectures. Enterprise organizations can use one of the enterprise topologies for production deployments to ensure the highest level of uptime, performance, and continued scalability. Organizations or deployments that are resource constrained can use a "growth" topology.

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 2.5 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.

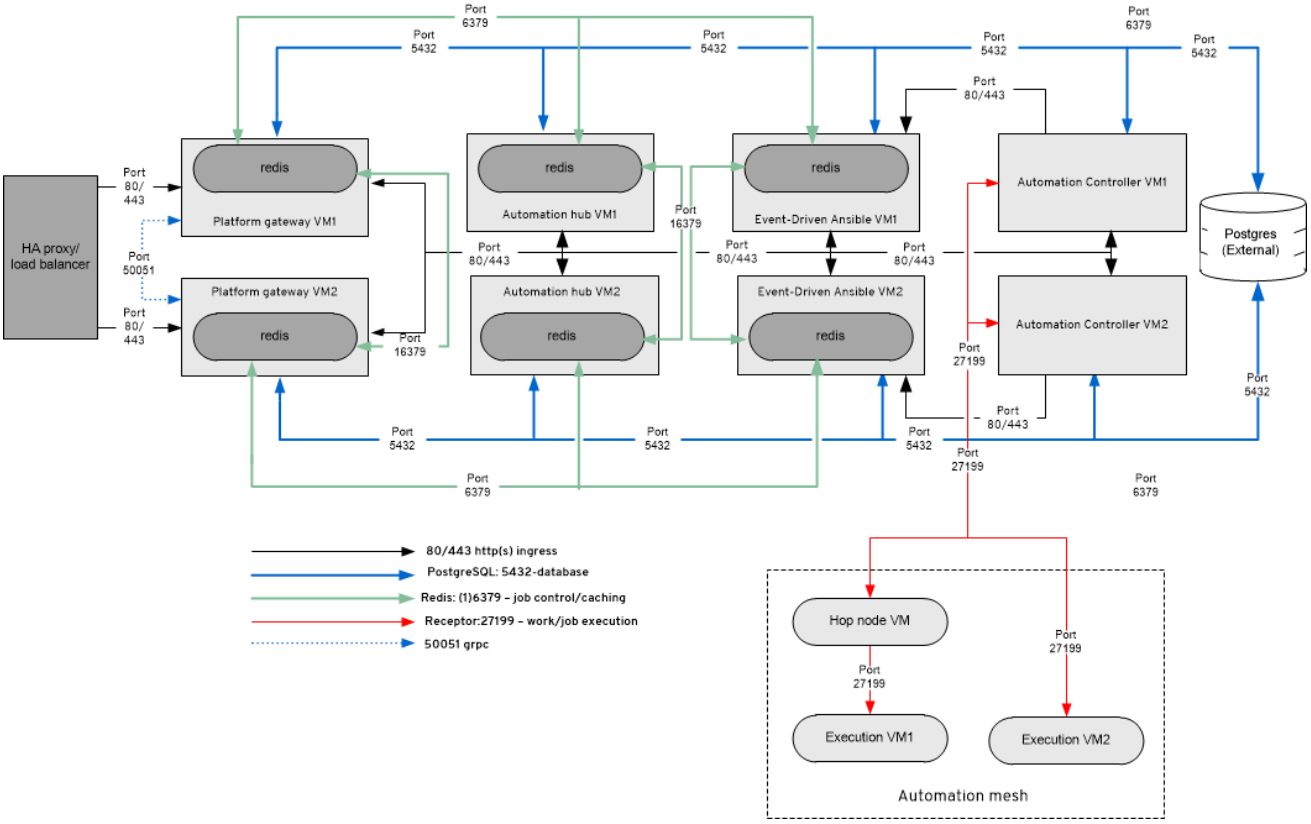
Event-Driven Ansible

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

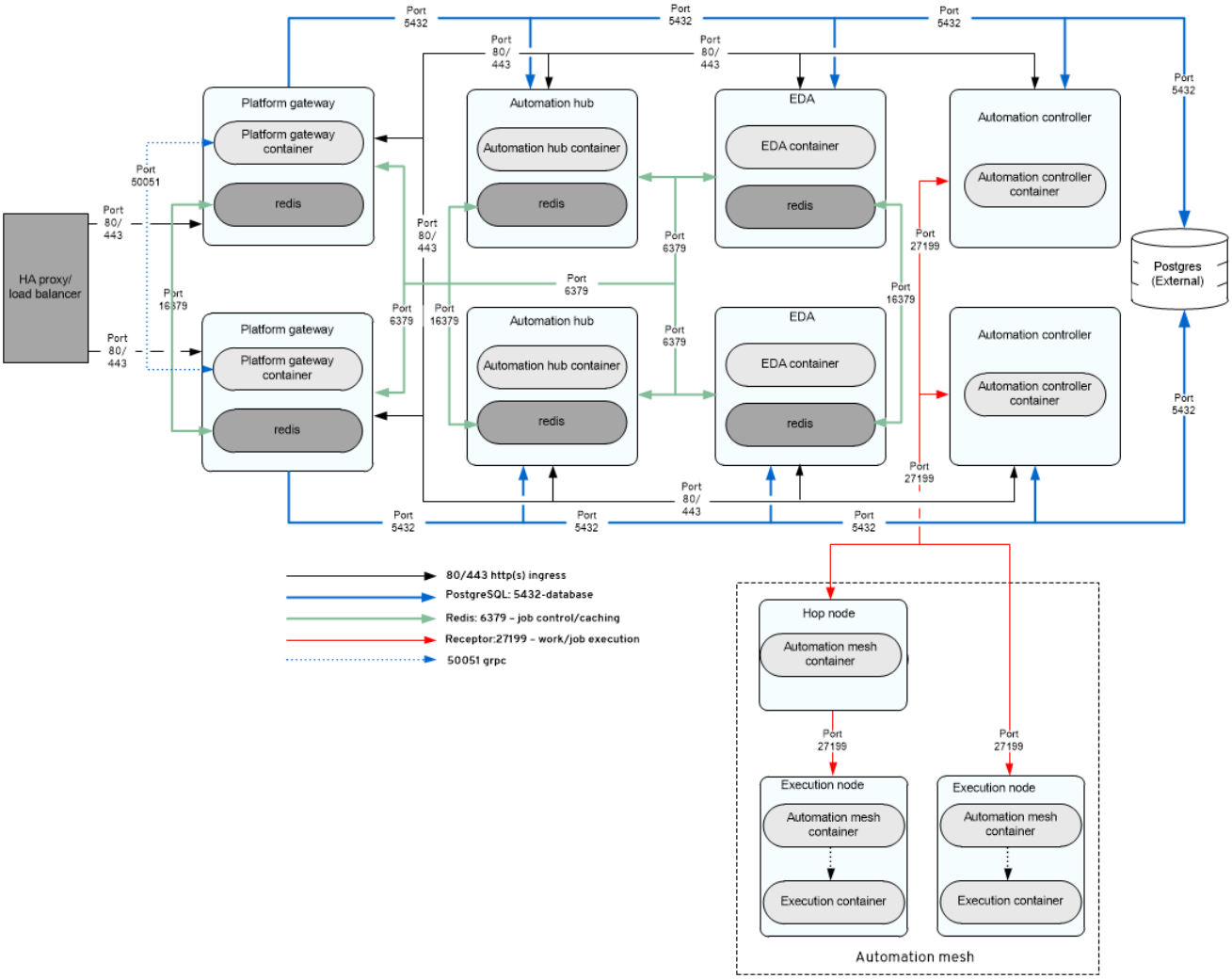
Figure 2.1. Example Ansible Automation Platform 2.5 architecture



2.2. EXAMPLE CONTAINERIZED DEPLOYMENT ARCHITECTURE

The following reference architecture provides an example setup of an enterprise deployment of containerized Ansible Automation Platform.

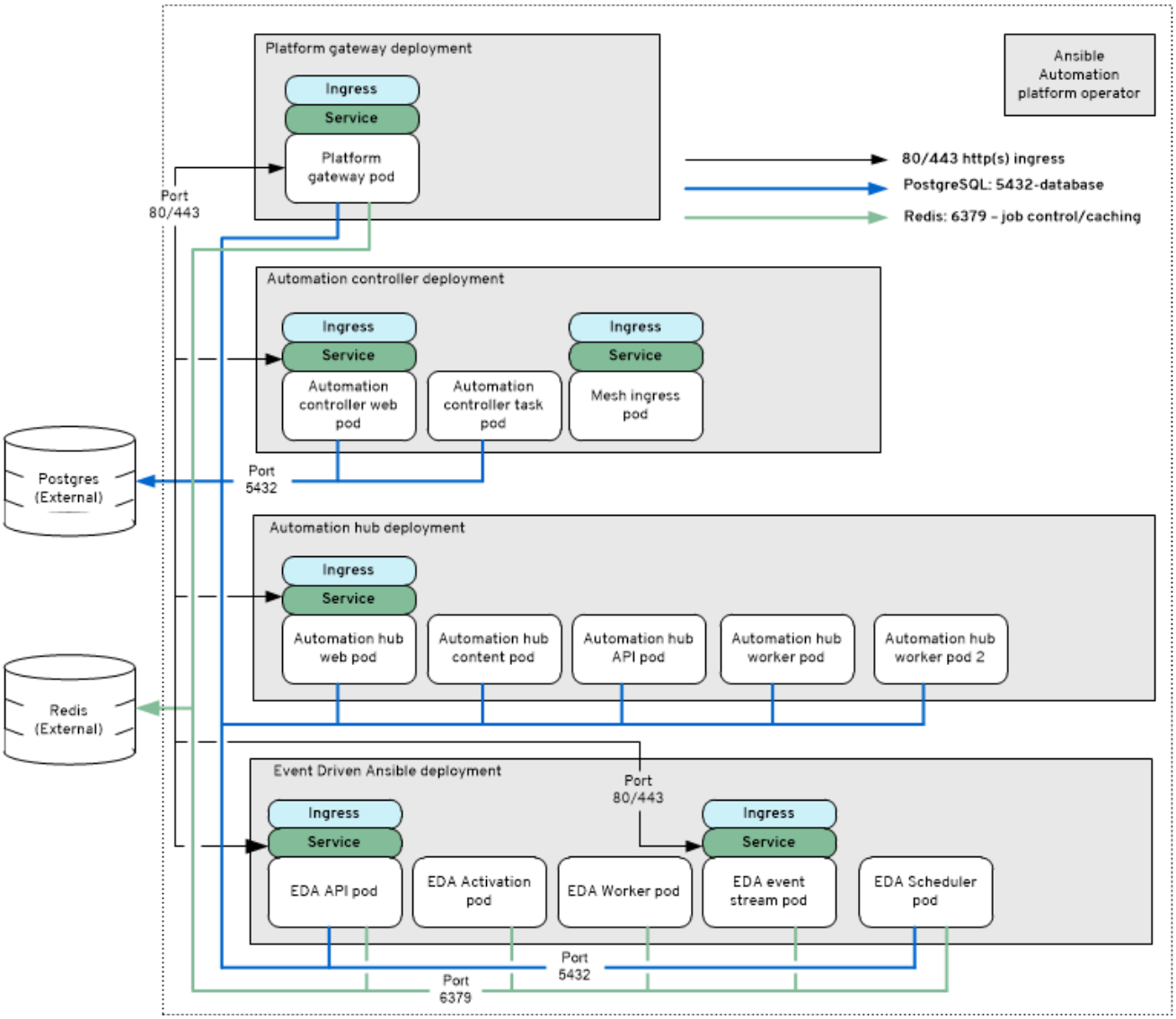
Figure 2.2. Example enterprise containerized deployment architecture



2.3. EXAMPLE OPERATOR-BASED DEPLOYMENT ARCHITECTURE

The following reference architecture provides an example setup of an enterprise deployment of Ansible Automation Platform on OpenShift Container Platform.

Figure 2.3. Example enterprise Operator-based deployment 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. PLATFORM GATEWAY

Platform gateway is the service that handles authentication and authorization for the Ansible Automation Platform. It provides a single entry into the Ansible Automation Platform and serves the platform user interface so you can authenticate and access all of the Ansible Automation Platform services from a single location. For more information about the services available in the Ansible Automation Platform, refer to Key functionality and concepts in *Getting started with Ansible Automation Platform*.

The platform gateway includes an activity stream that captures changes to gateway resources, such as the creation or modification of organizations, users, and service clusters, among others. For each change, the activity stream collects information about the time of the change, the user that initiated the change, the action performed, and the actual changes made to the object, when possible. The information gathered varies depending on the type of change.

You can access the details captured by the activity stream from the API:

```
/api/gateway/v1/activitystream/
```

3.2. 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.

3.3. 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.

3.4. 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.

3.5. 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

- Using automation decisions

3.6. 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.

3.7. 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.

3.8. 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.

3.9. 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. CACHING AND QUEUEING SYSTEM

In Ansible Automation Platform 2.5, Redis (REmote DIctionary Server) is used as the caching and queueing system. Redis is an open source, in-memory, NoSQL key/value store that is used primarily as an application cache, quick-response database and lightweight message broker.

Centralized Redis is provided for the platform gateway and Event-Driven Ansible and shared between those components. Automation controller and automation hub have their own instances of Redis.

This cache and queue system stores data in memory, rather than on a disk or solid-state drive (SSD), which helps deliver speed, reliability, and performance. In Ansible Automation Platform, the system caches the following types of data for the various services in Ansible Automation Platform:

Table 4.1. Data types cached by Centralized Redis

Automation controller	Event-Driven Ansible server	Automation hub	Platform gateway
N/A automation controller does not use shared Redis in Ansible Automation Platform 2.5	Event queues	N/A automation hub does not use shared Redis in Ansible Automation Platform 2.5	Settings, Session Information, JSON Web Tokens

This data can contain sensitive Personal Identifiable Information (PII). Your data is protected through secure communication with the cache and queue system through both Transport Layer Security (TLS) encryption and authentication.

NOTE

The data in Redis from both the platform gateway and Event-Driven Ansible are partitioned; therefore, neither service can access the other's data.

4.1. CENTRALIZED REDIS

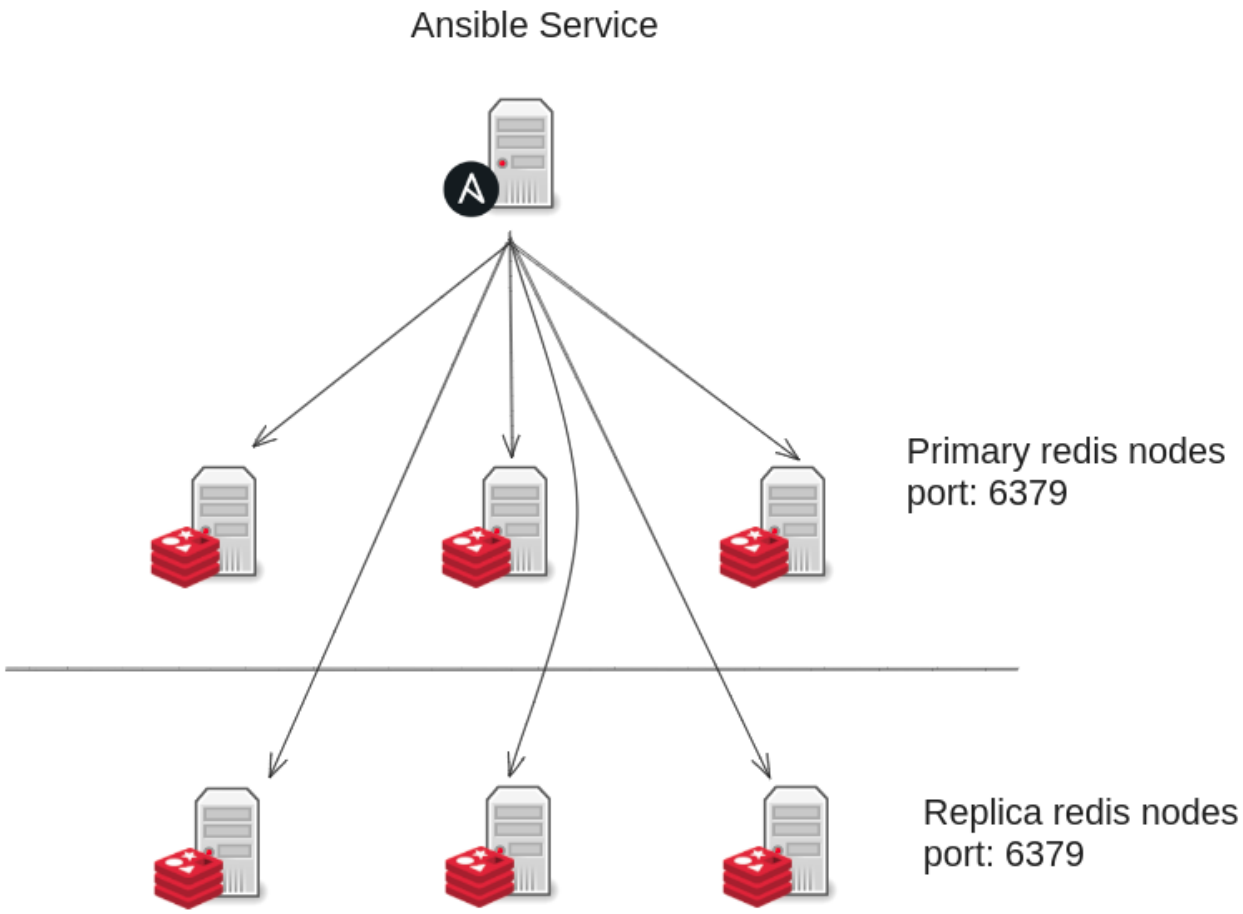
Ansible Automation Platform offers a centralized Redis instance in both standalone and clustered topologies. This enables resiliency by providing consistent performance and reliability.

4.2. CLUSTERED REDIS

With clustered Redis, data is automatically partitioned over multiple nodes to provide performance stability and nodes are assigned as replicas to provide reliability. Clustered Redis shared between the platform gateway and Event-Driven Ansible is provided by default when installing Ansible Automation Platform in containerized and operator-based deployments.

A cluster contains three primary nodes and each primary node contains a replica node.

If a primary instance becomes unavailable due to failures, the other primary nodes will initiate a failover state to promote a replica node to a primary node.



The benefits of deploying clustered Redis over standalone Redis include the following:

- Data is automatically split across multiple nodes.
- Data can be dynamically adjusted.
- Automatic failover of the primary nodes is initiated during system failures.

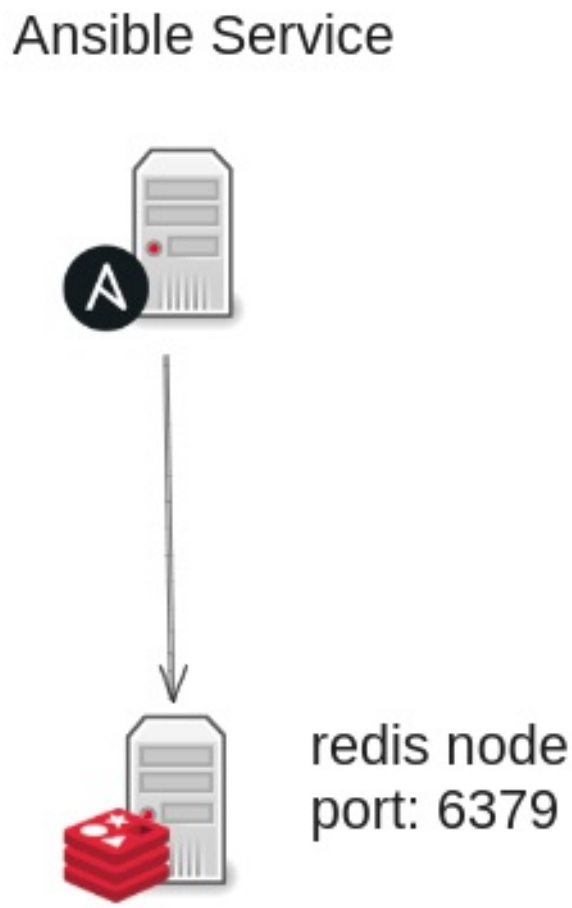
Therefore, if you need data scalability and automatic failover, deploy Ansible Automation Platform with a clustered Redis. For more information about scalability with Redis, refer to Scale with Redis Cluster in the Redis product documentation.

For information on deploying Ansible Automation Platform with clustered Redis, refer to the RPM installation, Containerized installation, and Installing on OpenShift Container Platform guides.

Disclaimer: Links contained in this information to external website(s) are provided for convenience only. Red Hat has not reviewed the links and is not responsible for the content or its availability. The inclusion of any link to an external website does not imply endorsement by Red Hat of the website or their entities, products or services. You agree that Red Hat is not responsible or liable for any loss or expenses that may result due to your use of (or reliance on) the external site or content.

4.3. STANDALONE REDIS

Standalone Redis consists of a simple architecture that is easy to deploy and configure.



If a resilient solution is not a requirement, deploy Ansible Automation Platform with a standalone Redis.

CHAPTER 5. 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.

5.1. SYSTEM REQUIREMENTS FOR RPM INSTALLATION

For system requirements for the RPM installation method of Ansible Automation Platform, see the System requirements section of *RPM installation*.

5.2. SYSTEM REQUIREMENTS FOR CONTAINERIZED INSTALLATION

For system requirements for the containerized installation method of Ansible Automation Platform, see the System requirements section of *Containerized installation*.

5.3. SYSTEM REQUIREMENTS FOR INSTALLING ON OPENSHIFT CONTAINER PLATFORM

For system requirements for installing Ansible Automation Platform on OpenShift Container Platform, see the Tested system configurations section of *Tested deployment models*.

CHAPTER 6. NETWORK PORTS AND PROTOCOLS

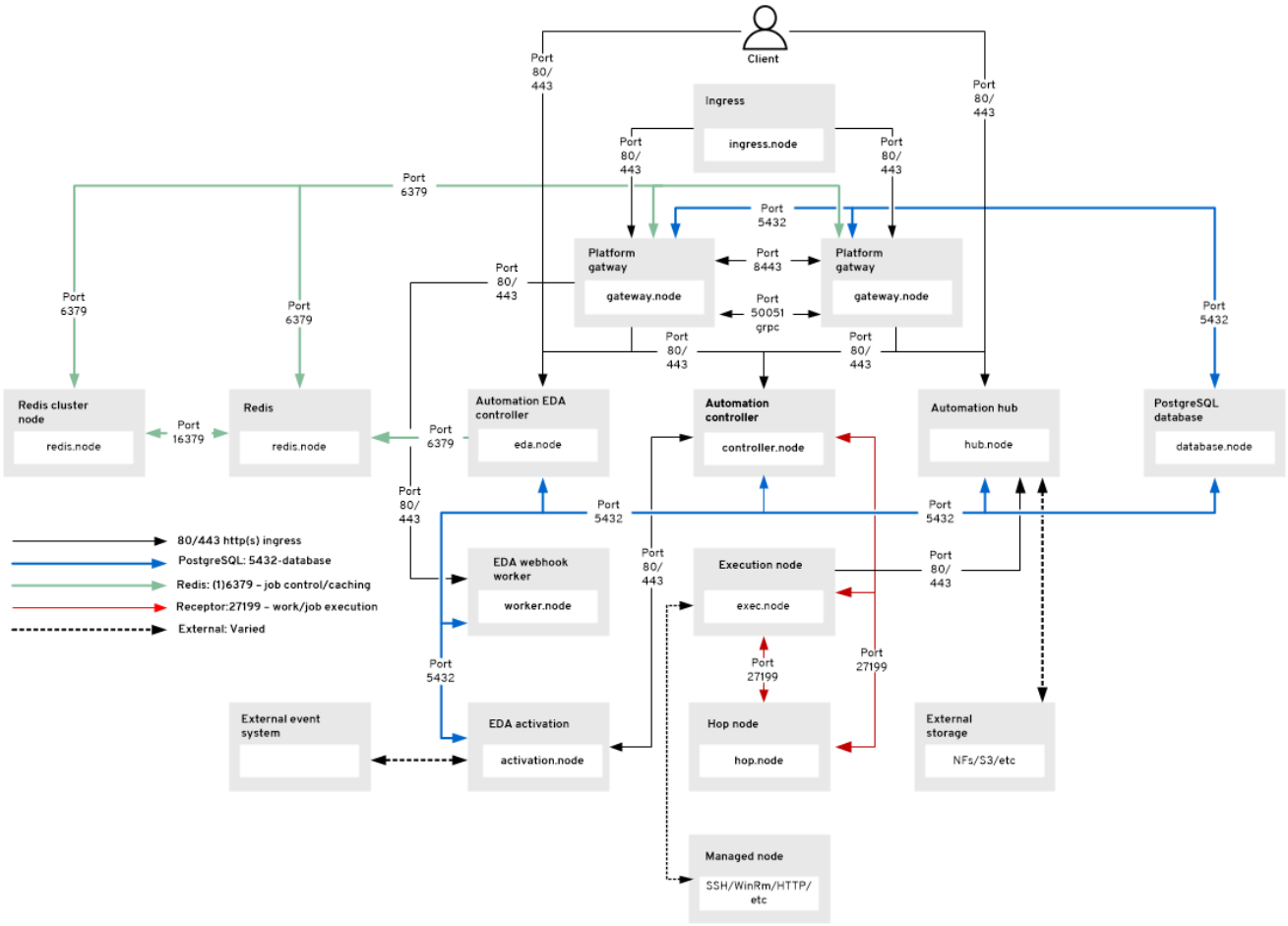
Red Hat Ansible Automation Platform uses several ports to communicate with its services. These ports must be open and available for incoming connections to the Red Hat Ansible Automation Platform server in order for it to work. Ensure that these ports are available and are not blocked by the server firewall.

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

NOTE

In some of the following use cases, hop nodes are used instead of a direct link from an execution node. Hop nodes are an option for connecting control and execution nodes. Hop nodes use minimal CPU and memory, so vertically scaling hop nodes does not impact system capacity.

Figure 6.1. 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 6.1. Network ports and protocols

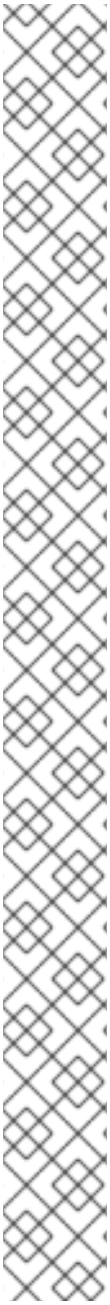
Port	Protocol	Service	Source	Destination	Required for	Installer Inventory Variable
22	TCP	SSH	Installer node	Automation hub	Installation (temporary)	**ansible_port**
22	TCP	SSH	Installer node	Controller node	Installation (temporary)	**ansible_port**
22	TCP	SSH	Installer node	Event-Driven Ansible node	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	Event-Driven Ansible node	Automation hub		Fixed value
80/443	TCP	HTTP/HTTPS	Event-Driven Ansible node	Automation controller		Fixed value
80/443	TCP	HTTP/HTTPS	Automation controller	Automation hub		Fixed value

Port	Protocol	Service	Source	Destination	Required for	Installer Inventory Variable
80/443	TCP	HTTP/HTTPS	Execution node	Automation hub	Allows execution nodes to pull the execution environment image from automation hub.	Fixed value
443	TCP	HTTPS	Controller node	Client	Web UI/API This exposes the mesh ingress receptor entry point for inbound connections.	**nginx_https_port**
443	TCP	HTTPS	Controller node	OpenShift Container Platform	Only required when using container groups to run jobs.	Host name of OpenShift API server
443	TCP	HTTPS	HA Proxy load balancer	Platform gateway	This is the ingress above the gateway that is customer controlled and can load balance requests to multiple gateways.	This port is customer managed outside of Ansible Automation Platform.
443	TCP	HTTPS	Platform gateway	Automation controller		
443	TCP	HTTPS	Platform gateway	Automation hub		
443	TCP	HTTPS	Platform gateway	Event-Driven Ansible		
443	HTTPS	Receptor	Execution node	OCP Mesh ingress		
443	HTTPS	Receptor	Hop node	OCP Mesh ingress		

Port	Protocol	Service	Source	Destination	Required for	Installer Inventory Variable
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	Event-Driven Ansible 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**
5432	TCP	PostgreSQL	Platform gateway	External database	Open only if the internal database is used along with another component. Otherwise, this port should not be open.	**automationgateway_pg_port**
6379	TCP	PostgreSQL	Event-Driven Ansible	Redis node		
6379	TCP	PostgreSQL	Platform gateway	Redis node		
8443	TCP	HTTPS	Platform gateway	Platform gateway	nginx	

Port	Protocol	Service	Source	Destination	Required for	Installer Inventory Variable
16379	TCP	Redis	Redis nodes	Redis nodes	Redis cluster bus port for a resilient Redis configuration	
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**
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**

Port	Protocol	Service	Source	Destination	Required for	Installer Inventory Variable
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	Hop node	Execution node		**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**
27199	TCP	Receptor	OCP cluster	Execution node		
50051	TCP	GRPC	Platform gateway	Platform gateway		



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 6.2. Red Hat Insights for Red Hat Ansible Automation Platform

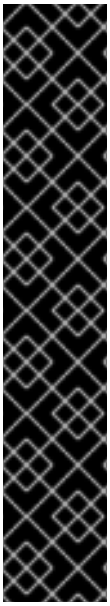
URL	Required for
https://api.access.redhat.com:443	General account services, subscriptions
https://cert-api.access.redhat.com:443	Insights data upload
https://cert.console.redhat.com:443	Inventory upload and Cloud Connector connection
https://console.redhat.com:443	Access to Insights dashboard

Table 6.3. 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 https://automation-hub-prd.s3.us-east-2.amazonaws.com	Firewall access
https://galaxy.ansible.com:443	Ansible Community curated Ansible content
https://ansible-galaxy-ng.s3.dualstack.us-east-1.amazonaws.com	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 6.4. 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 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.2.5 for RHEL 8 for x86_64

```
$ sudo dnf install --enablerepo=ansible-automation-platform-2.5-for-rhel-8-x86_64-rpms
ansible-automation-platform-installer
```

   v.2.5 for RHEL 9 for x86_64

```
$ sudo dnf install --enablerepo=ansible-automation-platform-2.5-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
RPM	**/opt/ansible-automation-platform/installer**
RPM bundle tar	**/ansible-automation-platform-setup-bundle-<latest-version>**
RPM non-bundle tar	**/ansible-automation-platform-setup-<latest-version>**
Container bundle tar	**/ansible-automation-platform-containerized-setup-bundle-<latest-version>**
Container non-bundle tar	**/ansible-automation-platform-containerized-setup-<latest-version>**

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

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

Example inventory file

```
[automationcontroller]
controller.example.com


[automationhub]
automationhub.example.com

[automationedacontroller]
automationedacontroller.example.com

[automationgateway]
gateway.example.com

[database]
data.example.com
```

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

pg_host=''
pg_port=''

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

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

automationgateway_admin_password=""
automationgateway_pg_host=""
automationgateway_pg_database=""
automationgateway_pg_username=""
automationgateway_pg_password=""
```

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

For more information on **registry_username** and **registry_password**, see Setting registry_username and registry_password.

Platform gateway is the service that handles authentication and authorization for the Ansible Automation Platform. It provides a single entry into the Ansible Automation Platform and serves the platform user interface so you can authenticate and access all of the Ansible Automation Platform services from a single location. For more information about the services available in the Ansible Automation Platform, refer to Key functionality and concepts in *Getting started with Ansible Automation Platform*.

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

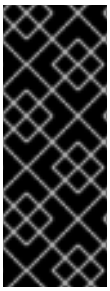
- If there is an **[automationhub]** group, you must include the variables **automationhub_pg_host** and **automationhub_pg_port**.
- Add Ansible automation hub information in the **[automationhub]** group.
- Do not install Ansible automation hub and automation controller on the same node.
- Provide 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.

Platform gateway

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

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.

Disclaimer: Links contained in this information to external website(s) are provided for convenience only. Red Hat has not reviewed the links and is not responsible for the content or its availability. The inclusion of any link to an external website does not imply endorsement by Red Hat of the website or their entities, products or services. You agree that Red Hat is not responsible or liable for any loss or expenses that may result due to your use of (or reliance on) the external site or content.

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;
AES256363836965356238653431633333396138333630646533646561383135343531353037646461653937653930633030653234666633306462323630653166663 10a37306230313337633963383130303334313534383962613632303761636632623932653062343839613639653635643365616233313365363661663931386430 0a353239373433313339613465326339313035633565353464356538653631633464343835346432376638623533613666326136343332313163343639393964613265616433363430633534303935646264633034383966336232303365383763
```

4. Run **setup.sh** for installation of Ansible Automation Platform 2.5 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. OVERVIEW OF TESTED DEPLOYMENT MODELS

Red Hat tests Ansible Automation Platform 2.5 with a defined set of topologies to give you opinionated deployment options. Deploy all components of Ansible Automation Platform so that all features and capabilities are available for use without the need to take further action.

Red Hat tests the installation of Ansible Automation Platform 2.5 based on a defined set of infrastructure topologies or reference architectures. Enterprise organizations can use one of the enterprise topologies for production deployments to ensure the highest level of uptime, performance, and continued scalability. Organizations or deployments that are resource constrained can use a "growth" topology.

It is possible to install the Ansible Automation Platform on different infrastructure topologies and with different environment configurations. Red Hat does not fully test topologies outside of published reference architectures. Use a tested topology for all new deployments.

9.1. INSTALLATION AND DEPLOYMENT MODELS

The following table outlines the different ways to install or deploy Ansible Automation Platform:

Table 9.1. Ansible Automation Platform installation and deployment models

Mode	Infrastructure	Description	Tested topologies
RPM	Virtual machines and bare metal	The RPM installer deploys Ansible Automation Platform on Red Hat Enterprise Linux by using RPMs to install the platform on host machines. Customers manage the product and infrastructure lifecycle.	• RPM growth topology • RPM mixed growth topology • RPM enterprise topology • RPM mixed enterprise topology
Containers	Virtual machines and bare metal	The containerized installer deploys Ansible Automation Platform on Red Hat Enterprise Linux by using Podman which runs the platform in containers on host machines. Customers manage the product and infrastructure lifecycle.	• Container growth topology • Container enterprise topology

Mode	Infrastructure	Description	Tested topologies
Operator	Red Hat OpenShift	The Operator uses Red Hat OpenShift Operators to deploy Ansible Automation Platform within Red Hat OpenShift. Customers manage the product and infrastructure lifecycle.	• Operator growth topology • Operator enterprise topology