

Install

Ansible Automation Platform 2.6

Contents

1 Install	10
Install containerized Ansible Automation Platform	10
Choose an installation type	10
Online installation.....	10
Disconnected (bundled) installation	10
Set up your subscription	11
Trial and evaluation.....	11
Node counting in subscriptions.....	11
Subscription Types	12
Attach your Ansible Automation Platform subscription	12
Activate your subscription	14
Activate with credentials.....	14
Activate with a manifest file	16
Obtain a manifest file	17
System requirements	18
Prerequisites	18
Ansible Automation Platform system requirements	19
Database requirements.....	22
Prepare the Red Hat Enterprise Linux host.....	22
Prepare the managed nodes.....	24
Download Ansible Automation Platform	25
Configure the inventory file.....	26
Inventory file for online installation for containerized growth topology (all-in-one)	27
Inventory file for online installation for containerized enterprise topology.....	29
Tips for building inventories	32
Create a registry service account.....	33
Advanced configuration options	33
Add a safe plugin variable to Event-Driven Ansible controller	34
Add execution nodes.....	34
Configure storage for automation hub.....	35

Configure Amazon S3 storage for automation hub	35
Configure Azure Blob Storage for automation hub	36
Configure Network File System (NFS) storage for automation hub.....	37
Configure a HAProxy load balancer.....	37
Enable automation content collection and container signing.....	38
Configure an external (customer provided) PostgreSQL database	41
Set up an external database with PostgreSQL admin credentials.....	42
Set up an external database without PostgreSQL admin credentials.....	42
Enable the hstore extension.....	44
Optional: configure mutual TLS (mTLS) authentication for an external database	46
Configure custom TLS certificates	47
Ansible Automation Platform generated certificates	48
User-provided certificates	48
Use a custom CA to generate all TLS certificates.....	48
Provide custom TLS certificates for each service	49
Considerations for certificates provided per service	51
Provide a custom CA certificate.....	51
Receptor certificate considerations.....	52
Redis certificate considerations	52
Use custom Receptor signing keys	52
Install containerized Ansible Automation Platform	53
Install in a disconnected environment	54
Obtain and configure RPM source dependencies	55
Configure a local repository using reposync.....	55
Configure a local repository from a mounted ISO	57
Perform a disconnected installation	59
Maintain containerized Ansible Automation Platform	60
Uninstall containerized Ansible Automation Platform	60
Reinstall containerized Ansible Automation Platform.....	61
Inventory file variables	61
Ansible variables.....	62
Automation hub variables	63
Automation controller variables.....	78
Database variables.....	85
Event-Driven Ansible controller variables.....	88
General variables.....	98

Install

Image variables.....	109
Platform gateway variables	112
Receptor variables	122
Redis variables	126
Install on OpenShift Container Platform.....	128
Plan your installation of Ansible Automation Platform on Red Hat OpenShift Container Platform ...	129
About Ansible Automation Platform Operator	129
OpenShift Container Platform version compatibility.....	130
Supported installation scenarios for Red Hat OpenShift Container Platform.....	130
Ansible Automation Platform Operator CSRF management	131
Additional resources	132
Install the Ansible Automation Platform Operator through OperatorHub	132
Install the Red Hat Ansible Automation Platform Operator on Red Hat OpenShift Container Platform.....	132
Install Ansible Automation Platform Operator from the CLI	134
Install the Ansible Automation Platform Operator in a namespace.....	134
Deploy Ansible Automation Platform	139
Deploy a new Ansible Automation Platform instance with components	139
Add components to an existing deployment.....	141
Access Ansible Automation Platform through the UI.....	143
Access Ansible Automation Platform through the CLI	144
Fetch the platform gateway web address.....	144
Fetch the platform gateway password	145
Decode the platform gateway password	146
Manage Ansible Automation Platform subscriptions, updates, and support	146
Trial and evaluation.....	146
Node counting in subscriptions.....	146
Subscription Types	147
Obtain a manifest file	147
Create a subscription allocation	148
Add subscriptions to a subscription allocation	148
Download a manifest file.....	148
Activate your subscription	149
Activate with credentials.....	149

Activate with a manifest file	150
Customize your Red Hat Ansible Automation Platform Operator	152
View configuration parameters in the UI	152
View configuration parameters by using the CLI	152
Configure component parameters	153
Customize your resource requirements	154
Configure your Ansible Automation Platform deployment	161
Enable HTTPS redirect for single sign-on	162
Configure your CSRF settings	162
Find and delete PVCs	165
Review platform gateway FAQs	166
Configure automation controller	167
Prerequisites	167
Configure your automation controller image pull policy	167
Configure your LDAP for automation controller	170
Configure automation controller route options	171
Configure ingress options for automation controller	172
Configure automation hub	174
Prerequisites	175
Storage options for automation hub	175
Provision OCP storage with <code>ReadWriteMany</code> access mode	175
Configure object storage on Amazon S3	176
Configure object storage on Azure Blob	177
Configure automation hub route options	179
Configure ingress options for automation hub	180
Finding the automation hub route	181
Add a collection download count to automation hub	182
Add allowed container registries	182
Configure content signing for automation hub	183
Configure custom PostgreSQL settings	186
Configure an external database for Ansible Automation Platform	187
Configure an external database for platform gateway on Red Hat Ansible Automation Platform Operator	188
Configure an external database for automation controller on Red Hat Ansible Automation Platform Operator	190

Configure an external database for automation hub on Red Hat Ansible Automation Platform	
Operator	193
Upgrading an external database for platform gateway on Ansible Automation Platform	196
Enable the hstore extension	198
Troubleshoot an external database	200
Configure Redis	201
Scale down your Ansible Automation Platform Operator deployment	202
Ansible Automation Platform Resource Operator	202
Resource Operator overview	202
Use Resource Operator	203
Connect Resource Operator to Ansible Automation Platform	204
Create an Ansible Automation Platform connection secret for Resource Operator	204
Create a custom resource for Resource Operator	205
Create an AnsibleJob custom resource	205
Create a JobTemplate custom resource	207
Create an automation controller project custom resource	207
Create an automation controller schedule custom resource	208
Create an automation controller workflow custom resource	209
Create an automation controller workflow template custom resource	210
Create an automation controller inventory custom resource	211
Create an automation controller credential custom resource	212
Red Hat Ansible Automation Platform custom resources	214
Custom resources	214
aap-existing-controller-and-hub-new-eda.yml	215
aap-all-defaults.yml	216
aap-existing-controller-only.yml	218
aap-existing-hub-and-controller.yml	220
aap-existing-hub-controller-eda.yml	221
aap-existing-hub-controller-eda.yml	222
aap-fresh-controller-eda.yml	223
aap-fresh-external-db.yml	225
aap-configuring-external-db-all-default-components.yml	227
aap-configuring-existing-external-db-all-default-components.yml	227
aap-configuring-external-db-with-lightspeed-enabled.yml	228
aap-fresh-install-with-settings.yml	228
aap-fresh-install.yml	230

aap-fresh-only-controller.yml	232
aap-fresh-only-hub.yml	234
aap-lightspeed-enabled.yml	235
gateway-only.yml	237
Custom resources	238
aap-existing-controller-and-hub-new-eda.yml	238
aap-all-defaults.yml	239
aap-existing-controller-only.yml	241
aap-existing-hub-and-controller.yml	243
aap-existing-hub-controller-eda.yml	244
aap-existing-hub-controller-eda.yml	245
aap-fresh-controller-eda.yml	246
aap-fresh-external-db.yml	248
aap-configuring-external-db-all-default-components.yml	250
aap-configuring-existing-external-db-all-default-components.yml	250
aap-configuring-external-db-with-lightspeed-enabled.yml	251
aap-fresh-install-with-settings.yml	251
aap-fresh-install.yml	253
aap-fresh-only-controller.yml	255
aap-fresh-only-hub.yml	257
aap-lightspeed-enabled.yml	258
gateway-only.yml	260
Install RPM-based Ansible Automation Platform	260
Prerequisites	261
System requirements	261
Prerequisites	261
Red Hat Ansible Automation Platform system requirements	262
Repository requirements	263
Additional notes for Red Hat Ansible Automation Platform requirements	264
Platform gateway system requirements	264
Automation controller system requirements	264
Automation hub system requirements	266
High availability automation hub requirements	266
Event-Driven Ansible system requirements	267
PostgreSQL requirements	268
PostgreSQL Configurations	269
Set up an external database	269

Enable mutual TLS (mTLS) authentication	272
Use custom TLS certificates	273
Receptor certificate considerations.....	274
Enable the hstore extension.....	274
Benchmark storage performance for the Ansible Automation Platform PostgreSQL database .	276
Install Ansible Automation Platform	278
Edit the inventory file	278
For an RPM installation	278
For online installations.....	279
Tips for building inventories	279
Guidelines for configuring your inventory file	279
Single platform gateway and automation controller with an external (installer managed)	
database	280
Single platform gateway, automation controller, and automation hub with an external (installer	
managed) database.....	282
Single platform gateway, automation controller, automation hub, and Event-Driven Ansible	
controller with an external (installer managed) database	284
High availability automation hub	288
Enable a high availability (HA) deployment of automation hub on SELinux	288
Configure pulpcore.service	290
Apply the SELinux context	291
Configure content signing on private automation hub	292
Add a safe plugin variable to Event-Driven Ansible controller	294
Create a registry service account.....	294
Configure Redis.....	295
Run the setup script.....	296
Verify your installation of Ansible Automation Platform	297
Install in a disconnected environment	297
Ansible Automation Platform installation on disconnected RHEL.....	297
System requirements for disconnected installation	297
Obtain and configure RPM source dependencies	298
Synchronize the RPM repositories by using reposync	298
Create a new web server to host repositories	299
Access RPM repositories from a locally mounted DVD	301
Download and install the Ansible Automation Platform setup bundle	302
Complete post installation tasks	303
Log in after installation	303

Install

Import your inventory file from source control	304
Inventory file fields for source control	304
Supported file syntax	305
Activate your subscription	307
Activate with credentials.....	307
Activate with a manifest file	308
Keep subscriptions for managed hosts in compliance	310
View hosts automated in the user interface	311
View hosts automated in the CLI.....	311
awx-manage utility.....	312
Deleting Hosts automated using API endpoint.....	312
Red Hat product documentation legal notices	314
GNU GENERAL PUBLIC LICENSE	315
Apache license	326

1 Install

Install containerized Ansible Automation Platform

Containerized Ansible Automation Platform uses Podman to run the platform in containers on Red Hat Enterprise Linux host machines. With this installation method, you manage both the product and infrastructure lifecycle while taking advantage of containerized architecture.

Containerized Ansible Automation Platform runs as rootless containers for enhanced security by default. You can install and operate Ansible Automation Platform with a non-root user account.

Choose an installation type

Containerized Ansible Automation Platform supports two installation types: online and disconnected. Review the requirements for each to decide which is appropriate for your environment.

Online installation

An online installation pulls container images directly from Red Hat registries during the installation process.

Requirements:

- An active internet connection on all Ansible Automation Platform nodes
- A Red Hat registry service account with credentials (`registry_username` and `registry_password`)
- Network access to Red Hat registries (registry.redhat.io)

For online installation instructions, see *Prepare the installation* in the related information section.

Disconnected (bundled) installation

A disconnected installation uses a pre-packaged bundle that includes all container images and dependencies. This installation type is designed for air-gapped or restricted network environments.

Requirements:

- Local RPM repository configured with required dependencies
- No internet connection required during installation
- Red Hat registry credentials are not required

For disconnected installation instructions, see *Install in a disconnected environment* in the related information section.

Related information

[Install in a disconnected environment](#)

Set up your subscription

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

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

Trial and evaluation

You need a subscription to run Ansible Automation Platform. You can start by signing up for a free trial subscription.

- Trial subscriptions for Ansible Automation Platform are available at the [Red Hat product trial center](#).
- Support is not included in a trial subscription or during an evaluation of the Ansible Automation Platform.

Node counting in subscriptions

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

For more information about managed node requirements for subscriptions, see [How are "managed nodes" defined as part of the Red Hat Ansible Automation Platform offering](#).

NOTE:

Ansible does not recycle node counts or reset automated hosts.

Subscription Types

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

- **Standard:**

- Manage any size environment
- Enterprise 8x5 support and SLA
- Maintenance and upgrades included
- Review the SLA at [Product Support Terms of Service](#)
- Review the [Red Hat Support Severity Level Definitions](#)

- **Premium:**

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

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

For more information, contact Ansible through the [Red Hat Customer Portal](#) or at the [Ansible site](#).

Attach your Ansible Automation Platform subscription

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

NOTE:

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

Procedure

1. Register your system with Red Hat Subscription Management:

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

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

NOTE:

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

Result

1. Refresh the subscription information on your system:

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

2. Verify your registration:

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

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

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

```
$ sudo subscription-manager list --available --all | grep -A 30 "Ansible Automation Platform"
```

This command displays the subscription details including the Pool ID. Look for the `Pool ID:` line in the output.

Once you have identified the correct Pool ID, attach the subscription:

```
$ sudo subscription-manager attach --pool=<pool_id>
```

NOTE:

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

- For legacy accounts not using SCA, 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, use the following steps to identify and enable the required repository:

- a. List available repositories:

```
$ sudo subscription-manager repos --list | grep -i ansible-automation-platform
```

- b. Identify the repository name that matches your RHEL version, Ansible Automation Platform version, and architecture (for example, `ansible-automation-platform-2.6-for-rhel-9-x86_64-rpms`).

- c. Enable the repository:

```
$ sudo subscription-manager repos --enable <repository_name>
```

Activate your subscription

Red Hat Ansible Automation Platform uses available subscriptions or a subscription manifest to allow the use of Ansible Automation Platform.

To obtain a subscription, you can do either of the following:

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

Related information

[Activate with credentials](#)

[Activate with a manifest file](#)

Activate with credentials

Activate your Ansible Automation Platform subscription at first launch using Red Hat service account credentials or a personal login. This retrieves and imports the license, granting access to Red Hat content and entitlement services.

NOTE:

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

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

Procedure

1. Log in to Red Hat Ansible Automation Platform.
2. Select the **Service Account** tab in the subscription wizard.
3. Enter your **Client ID** and **Client secret**.
4. Select your subscription from the **Subscription** list.

NOTE:

You can also enter your Satellite username and password in the **Satellite** tab if your cluster nodes are registered to Satellite through Subscription Manager.

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

Result

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

Hosts automated

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

Hosts imported

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

Hosts remaining

Total host count minus hosts automated

Activate with a manifest file

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

Before you begin

You must have a Red Hat subscription manifest file exported from the Red Hat Customer Portal. For more information, see [Obtaining a manifest file](#).

NOTE:

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

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

Procedure

1. Log in to Red Hat Ansible Automation Platform.
 - a. If you are not immediately taken to the subscription wizard, go to **Settings > Subscription**.
2. Select the **Subscription manifest** tab.
3. Click **Browse** and select your manifest file.
4. Review the End User License Agreement and select **I agree to the End User License Agreement**.
5. Click **Finish**.

NOTE:

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

Result

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

Hosts automated

Host count automated by the job, which uses the subscription count

Hosts imported

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

Hosts remaining

Total host count minus hosts automated

Next steps

- You can return to the subscription wizard by selecting **Settings > Subscription** from the navigation panel and clicking **Edit subscription**.

Obtain a manifest file

You can obtain a subscription manifest in the **Subscription Allocations** section of Red Hat Subscription Management.

After you obtain a subscription allocation, you can download its manifest file and upload it to activate Ansible Automation Platform.

To begin, log in to the **Red Hat Customer Portal** by using your administrator user account and follow the procedures listed.

Related information

[Red Hat Customer Portal](#)

[Subscription Allocations](#)

Create a subscription allocation

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

Procedure

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

Add subscriptions to a subscription allocation

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

Procedure

1. From the [Subscription Allocations](#) page, click the name of the **Subscription Allocation** to which you want to add a subscription.
2. Click the **Subscriptions** tab.
3. Click **Add Subscriptions**.
4. Enter the number of Ansible Automation Platform Entitlements you plan to add.
5. Click **Submit**.

Download a manifest file

After you create an allocation with the appropriate subscriptions on it, you can download the manifest file from Red Hat Subscription Management.

Procedure

1. From the [Subscription Allocations](#) page, click the name of the **Subscription Allocation** to which you want to generate a manifest.
2. Click the **Subscriptions** tab.
3. Click **Export Manifest** to download the manifest file.

Result

This downloads a file `manifest_<allocation name>_<date>.zip` to your default downloads folder.

System requirements

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

Prerequisites

- Configure a dedicated non-root user on the Red Hat Enterprise Linux host.

- This user requires `sudo` or other Ansible supported privilege escalation (`sudo` is recommended) to perform administrative tasks during the installation.
- This user is responsible for the installation of containerized Ansible Automation Platform.
- This user is also the service account for the containers running Ansible Automation Platform.
- For managed nodes, configure a dedicated user on each node. Ansible Automation Platform connects as this user to run tasks on the node. For more information about configuring a dedicated user on each node, see [See "Prepare the managed nodes" on page 124](#).
- For remote host installations, configure SSH public key authentication for the non-root user. For guidelines on setting up SSH public key authentication for the non-root user, see [How to configure SSH public key authentication for passwordless login](#).
- Ensure the Red Hat Enterprise Linux host has internet access if you are using the default online installation method.
- Open the appropriate network ports if you have a firewall in place. For more information about the ports to open, see [Container enterprise topology](#) in *Tested deployment models*.

IMPORTANT:

Podman does not support storing container images on an NFS share. To use an NFS share for the user home directory, set up the Podman storage backend path outside of the NFS share. For more information, see [Rootless Podman and NFS](#).

Ansible Automation Platform system requirements

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

System configuration

Type	Description	Notes
Subscription	• Valid Red Hat Ansible Automation Platform subscription • Valid Red Hat Enterprise Linux subscription (to consume the BaseOS	

Type	Description	Notes
	and AppStream repositories)	
Operating system	- Red Hat Enterprise Linux 9.4 or later minor versions of Red Hat Enterprise Linux 9. - Red Hat Enterprise Linux 10 or later minor versions of Red Hat Enterprise Linux 10.	
CPU architecture	x86_64, AArch64, s390x (IBM Z), ppc64le (IBM Power)	
<code>ansible-core</code>	- RHEL 9: installation program uses <code>ansible-core</code> 2.14, Ansible Automation Platform operation uses <code>ansible-core</code> 2.16. - RHEL 10: installation program uses <code>ansible-core</code> 2.16, Ansible Automation Platform operation uses <code>ansible-core</code> 2.16.	- The installation program uses the <code>ansible-core</code> package from the RHEL AppStream repository. - Ansible Automation Platform bundles <code>ansible-core</code> 2.16 for operation, so you do not need to install it manually.
Browser	A currently supported version of Mozilla Firefox or Google Chrome.	
Database	- For Ansible Automation Platform managed databases: PostgreSQL 15. - For customer provided (external) databases:	External (customer supported) databases require International Components for Unicode (ICU) support.

Type	Description	Notes
	PostgreSQL 15, 16, or 17.	External databases using PostgreSQL 16 or 17 must rely on external backup and restore processes. Backup and restore functionality is dependent on utilities provided with PostgreSQL 15.
IP version	IPv4, IPv6 (single-stack and dual-stack)	

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

Virtual machine requirements

Requirement	Minimum requirement
RAM	16 GB 32 GB required for growth topology bundled installations with <code>hub_seed_collections=true</code>. Seeding the collections can take 45 or more minutes.
CPUs	4
Local disk	- Total available disk space: 60 GB - Installation directory: 15 GB (if on a dedicated partition) <code>/var/tmp</code> for online installations: 1 GB <code>/var/tmp</code> for offline or bundled installations: 3 GB - Temporary directory (defaults to <code>/tmp</code>) for offline or bundled installations: 10GB
Disk IOPS	3000

Database requirements

Ansible Automation Platform can work with two varieties of database:

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

Ansible Automation Platform requires a customer provided (external) database to have International Components for Unicode (ICU) support.

Related information

[Red Hat Ansible Automation Platform Database Scope of Coverage](#)

[Configure an external \(customer provided\) PostgreSQL database](#)

[Collation Support](#)

Prepare the Red Hat Enterprise Linux host

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

Procedure

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

```
hostname -f
```

Example output:

```
aap.example.org
```

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

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

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

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

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

```
$ sudo dnf repolist
```

Example output for RHEL 9:

```
Updating Subscription Management repositories.

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

Example output for RHEL 10:

```
Updating Subscription Management repositories.

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

- For disconnected installations, follow the steps in [Obtain and configure RPM source dependencies](#) to access these repositories.

5. Ensure the host can resolve host names and IP addresses using DNS. This is essential to ensure services can talk to one another.
6. Install `ansible-core`:

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

7. Optional: Install additional utilities that are useful for troubleshooting purposes, for example `wget`, `git-core`, `rsync`, and `vim`:

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

- Optional: To have the installation program automatically pick up and apply your Ansible Automation Platform subscription manifest license, follow the steps in [See "Obtain a manifest file" on page 1147](#).

Related information

[Setting up an unbound DNS server](#)

[Setting up and configuring a BIND DNS server](#)

Prepare the managed nodes

Managed nodes, or hosts, are the devices managed by Ansible Automation Platform. To ensure a secure containerized setup, create a dedicated user on each node for Ansible Automation Platform to use when connecting and running tasks.

Procedure

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

```
$ sudo adduser <username>
```

- Set a password for the new user. Replace `<username>` with the username you created.

```
$ sudo passwd <username>
```

- Configure the user to run `sudo` commands.

For a secure and maintainable installation, configure `sudo` privileges for the installation user in a dedicated file within the `/etc/sudoers.d/` directory.

- Create a dedicated `sudoers` file for the user:

```
$ sudo visudo -f /etc/sudoers.d/<username>
```

- Add the following line to the file, replacing `<username>` with the username you created:

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

- Save and exit the file.

Download Ansible Automation Platform

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

Before you begin

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

Procedure

1. Download the latest version of containerized Ansible Automation Platform from the [Ansible Automation Platform download page](#).
 - a. For online installations: **Ansible Automation Platform 2.6 Containerized Setup**
 - b. For offline or bundled installations: **Ansible Automation Platform 2.6 Containerized Setup Bundle**

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

Use the `scp` command to securely copy the files. The basic syntax for `scp` is:

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

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

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

3. Decide where you want the installation program to reside on the file system. This is your installation directory.
 - a. The installation creates installation-related files under this location and requires at least 15 GB for the initial installation.
4. Unpack the installation program `.tar.gz` file into your installation directory, and go to the unpacked directory.
 - a. To unpack the online installer:

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

- b. To unpack the offline or bundled installer:

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

Related information

[scp\(1\) - Linux manual page](#)

Configure the inventory file

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

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

IMPORTANT:

The inventory file requirements differ based on your installation type:

- **Online installation:** Requires the `registry_username` and `registry_password` variables to authenticate and pull container images from Red Hat registries during installation.
- **Disconnected (bundled) installation:** Does not require `registry_username` or `registry_password` because all container images are pre-packaged in the bundle. Instead, requires the `bundle_install=true` and `bundle_dir` variables.

The following inventory file examples are for online installations. For disconnected installation inventory requirements, see [See "Perform a disconnected installation" on page 159](#).

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

- In the downloaded installation program package:
 - The default inventory file, named `inventory`, is for the enterprise topology pattern.
 - To deploy the growth topology (all-in-one) pattern, use the `inventory-growth` file instead.
- In [Container growth topology](#) and [Container enterprise topology](#) in *Tested deployment models*.

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

Refer to the `README.md` file in the installation directory or [See "Inventory file variables" on page 161](#) for more information about optional and required variables.

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

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

```
# This is the Ansible Automation Platform installer inventory file intended for the
# container growth deployment topology.

# This inventory file expects to be run from the host where Ansible Automation
# Platform will be installed.

# Consult the Ansible Automation Platform product documentation about this
# topology's tested hardware configuration.

# https://docs.redhat.com/en/documentation/red_hat_ansible_automation_platform/2.6/
# plan-ref_installation_deployment_models

#

# Consult the docs if you are unsure what to add
# For all optional variables consult the included README.md
# or the Ansible Automation Platform documentation:

# https://docs.redhat.com/en/documentation/red_hat_ansible_automation_platform/2.6/
# install-con_aap_containerized_installation_intro


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


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


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


# This section is for your Event-Driven Ansible controller hosts
# -----
```

```

[automationeda]
aap.example.org

# This section is for the Ansible Automation Platform database
# -----

[database]
aap.example.org

[all:vars]
# Ansible
ansible_connection=local

# Common variables
# https://docs.redhat.com/en/documentation/red_hat_automation_platform/2.6/
# install-ref_general_inventory_variables
# -----
postgresql_admin_username=postgres
postgresql_admin_password=<set your own>

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

redis_mode=standalone

# Platform gateway
# https://docs.redhat.com/en/documentation/red_hat_automation_platform/2.6/
# install-ref_gateway_variables
# -----
gateway_admin_password=<set your own>
gateway_pg_host=aap.example.org
gateway_pg_password=<set your own>

# Automation controller
# https://docs.redhat.com/en/documentation/red_hat_automation_platform/2.6/
# install-ref_controller_variables
# -----
controller_admin_password=<set your own>
controller_pg_host=aap.example.org

```



```

controller_pg_password=<set your own>
controller_percent_memory_capacity=0.5

# Automation hub
# https://docs.redhat.com/en/documentation/red_hat_ansible_automation_platform/2.6/
install-ref_hub_variables
# -----
hub_admin_password=<set your own>
hub_pg_host=aap.example.org
hub_pg_password=<set your own>
hub_seed_collections=false

# Event-Driven Ansible controller
# https://docs.redhat.com/en/documentation/red_hat_ansible_automation_platform/2.6/
install-ref_eda_controller_variables
# -----
eda_admin_password=<set your own>
eda_pg_host=aap.example.org
eda_pg_password=<set your own>

```

- `ansible_connection=local` - Used for all-in-one installations where the installation program is run on the same node that hosts Ansible Automation Platform.
 - If the installation program is run from a separate node, do not include `ansible_connection=local`. In this case, use an SSH connection instead.
- `[database]` - This group in the inventory file defines the Ansible Automation Platform managed database.

Inventory file for online installation for containerized enterprise topology

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

```

# This is the Ansible Automation Platform enterprise installer inventory file
# Consult the docs if you are unsure what to add
# For all optional variables consult the included README.md
# or the Red Hat documentation:

```

```

# https://docs.redhat.com/en/documentation/red_hat_ansible_automation_platform/2.6/
install-con_aap_containerized_installation_intro

# This section is for your platform gateway hosts
# -----

[automationgateway]
gateway1.example.org
gateway2.example.org

# This section is for your automation controller hosts
# -----

[automationcontroller]
controller1.example.org
controller2.example.org

# This section is for your Ansible Automation Platform execution hosts
# -----

[execution_nodes]
hop1.example.org receptor_type='hop'
exec1.example.org
exec2.example.org

# This section is for your automation hub hosts
# -----

[automationhub]
hub1.example.org
hub2.example.org

# This section is for your Event-Driven Ansible controller hosts
# -----

[automationeda]
eda1.example.org
eda2.example.org

[redis]
gateway1.example.org
gateway2.example.org

```

```

hub1.example.org
hub2.example.org
eda1.example.org
eda2.example.org

[all:vars]

# Common variables
# https://docs.redhat.com/en/documentation/red_hat_automation_platform/2.6/
install-ref_general_variables
# -----

postgresql_admin_username=<set your own>
postgresql_admin_password=<set your own>
registry_username=<your RHN username>
registry_password=<your RHN password>

# Platform gateway
# https://docs.redhat.com/en/documentation/red_hat_automation_platform/2.6/
install-ref_gateway_variables
# -----

gateway_admin_password=<set your own>
gateway_pg_host=externaldb.example.org
gateway_pg_database=<set your own>
gateway_pg_username=<set your own>
gateway_pg_password=<set your own>

# Automation controller
# https://docs.redhat.com/en/documentation/red_hat_automation_platform/2.6/
install-ref_controller_variables
# -----

controller_admin_password=<set your own>
controller_pg_host=externaldb.example.org
controller_pg_database=<set your own>
controller_pg_username=<set your own>
controller_pg_password=<set your own>

# Automation hub
# https://docs.redhat.com/en/documentation/red_hat_automation_platform/2.6/

```

```
install-ref_hub_variables

# -----

hub_admin_password=<set your own>
hub_pg_host=externaldb.example.org
hub_pg_database=<set your own>
hub_pg_username=<set your own>
hub_pg_password=<set your own>

# Event-Driven Ansible controller

# https://docs.redhat.com/en/documentation/red_hat_ansible_automation_platform/2.6/
install-ref_eda_controller_variables

# -----

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

Related information

[Container growth topology](#)

[Container enterprise topology](#)

Tips for building inventories

When building inventories for Ansible automation, consider the following best practices to ensure efficient and effective management of your hosts.

- Ensure that group names are meaningful and unique.
- Group names are also case sensitive.
- Do not use spaces, hyphens, or preceding numbers (use floor_19, not 19th_floor) in group names.
- Group hosts in your inventory logically according to their What, Where, and When:
 - What: Group hosts according to the topology, for example: db, web, leaf, spine.
 - Where: Group hosts by geographic location, for example: data center, region, floor, building.
 - When: Group hosts by stage, for example: development, test, staging, production.

Create a registry service account

Create a registry service account on the Red Hat Customer Portal to generate a token and username. Use these to set the `registry_username` and `registry_password` variables required for online non-bundled installations.

Procedure

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

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

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

Advanced configuration options

Configure external databases, custom TLS certificates, execution nodes, HAProxy load balancers, and hub storage for complex containerized Ansible Automation Platform deployments.

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

Related information

[Install containerized Ansible Automation Platform](#)

Add a safe plugin variable to Event-Driven Ansible controller

When using the `redhat.insights_eda` plugin for rulebook activations, add a safe plugin variable to a platform directory to ensure a secure connection and correct port mapping displays.

Procedure

1. Create a directory for the safe plugin variable:

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

2. Create a file within that directory for your new setting (for example, `touch ./group_vars/automationeda/custom.yml`)
3. Add the variable `eda_safe_plugins` with a list of plugins to enable. For example:

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

Add execution nodes

Containerized Ansible Automation Platform can deploy remote execution nodes.

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

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

By default, an execution node uses the following settings that you can update as needed:

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

- `receptor_port` - The port number that receptor listens on for incoming connections from other receptor nodes.
- `receptor_type` - The role of the node. Valid options include `execution` or `hop`.

- `receptor_protocol` - The protocol used for communication. Valid options include `tcp` or `udp`.

By default, execution nodes automatically peer with all automation controller nodes. To configure an execution node to peer with specific automation controller nodes instead, use the `receptor_peers` variable.

NOTE:

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

Example:

```
[execution_nodes]
# Uses default peering (peers with all controller nodes)
exec1.example.com

# Only peers with specific controller nodes
exec2.example.com
receptor_peers='["controller1.example.com", "controller2.example.com"]'

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

Configure storage for automation hub

Configure storage backends for automation hub to store automation content by using Amazon S3, Azure Blob Storage, or Network File System (NFS).

Configure Amazon S3 storage for automation hub

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

Procedure

1. Ensure your AWS S3 bucket exists before proceeding with the installation.
2. Add the following variables to your inventory file under the `[all:vars]` group to configure S3 storage:

```
[all:vars]
hub_storage_backend=s3
hub_s3_access_key=<access_key>
hub_s3_secret_key=<secret_key>
hub_s3_bucket_name=<bucket_name>
```

- Optional: You can pass extra parameters to the AWS S3 storage backend by using the `hub_s3_extra_settings` variable. For example:

```
hub_s3_extra_settings={'AWS_S3_REGION_NAME': 'eu-south-1',
'AWS_S3_ENDPOINT_URL': 'https://endpoint'}
```

Related information

[django-storages documentation - Amazon S3](#)

Configure Azure Blob Storage for automation hub

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

Procedure

- Ensure your Azure container exists before proceeding with the installation.
- Add the following variables to your inventory file under the `[all:vars]` group to configure Azure Blob storage:

```
[all:vars]
hub_storage_backend=azure
hub_azure_account_key=<account_key>
hub_azure_account_name=<account_name>
hub_azure_container=<container_name>
```

- Optional: You can pass extra parameters to the Azure Blob storage backend by using the `hub_azure_extra_settings` variable. For example:

```
hub_azure_extra_settings={'AZURE_LOCATION': 'foo', 'AZURE_SSL': True,
'Azure_URL_EXPIRATION_SECS': 60}
```


Related information

[django-storages documentation - Azure Storage](#)

Configure Network File System (NFS) storage for automation hub

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

Procedure

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

```
hub_shared_data_path=<path_to_nfs_share>
```

The value must match the format `host:dir`, for example `nfs-server.example.com:/exports/hub`.

2. (Optional) To change the mount options for your NFS share, use the `hub_shared_data_mount_opts` variable. The default value is `rw, sync, hard`.

Configure a HAProxy load balancer

Configure a HAProxy load balancer in front of platform gateway with a custom CA cert.

Procedure

1. Set the following inventory file variables under the `[all:vars]` group:

```
custom_ca_cert=<path_to_cert.crt>
gateway_main_url=<https://load_balancer_url>
```

IMPORTANT:

- Ensure your load balancer is configured to use HTTP/1.1 when communicating with platform gateway. HTTP/2 is not supported.
- HAProxy SSL passthrough mode is not supported with platform gateway.

Enable automation content collection and container signing

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

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

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

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

```
# Collection signing
hub_collection_signing_pass=<gpg_key_passphrase>

# Container signing
hub_container_signing_pass=<gpg_key_passphrase>
```

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

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

NOTE:

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

Procedure

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

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

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

Example output:

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

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

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

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

Change (N)ame, (E)mail, or (O)kay/(Q)uit? 0
```

- If this fails, your environment does not have the necessary prerequisite packages installed for GPG. Install the necessary packages to proceed.
- A dialog box will appear and ask you for a passphrase. This is optional but recommended.
- The keys are then generated, and produce output similar to the following:

```
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.

gpg: key 022E4FBFB650F1C4 marked as ultimately trusted
gpg: revocation certificate stored as '/home/aapuser/.gnupg/openpgp-
revocs.d/F001B037976969DD3E17A829022E4FBFB650F1C4.rev'
public and secret key created and signed.

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

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

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

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

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

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

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

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

- Enter the passphrase if prompted.

6. To view the private key file contents, run the following command:

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

Example output:

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

lQWFBGcbN14BDADTg5BsZGbSGMHypUJMuzmIffzzz4LULrZA8L/I616lzpBHJvEs
sSN6KuKY1TcIwIDCCa/U50bm46kurpP2Y+vNA1YSEtMJoSeHeamWMDd99f49ItBp

<snippet>

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

7. Repeat steps 1 to 7 to create a key pair for container signing.
8. Add the following variables to the inventory file and run the installation to create the signing services:

```
# Collection signing
hub_collection_signing=true

hub_collection_signing_key=/home/aapuser/aap/ansible-automation-platform-
containerized-setup-<version_number>/collection-signing-key.priv

# This variable is required if the key is protected by a passphrase
hub_collection_signing_pass=<password>


# Container signing
hub_container_signing=true

hub_container_signing_key=/home/aapuser/aap/ansible-automation-platform-
containerized-setup-<version_number>/container-signing-key.priv

# This variable is required if the key is protected by a passphrase
hub_container_signing_pass=<password>
```

Configure an external (customer provided) PostgreSQL database

Set up an external (customer provided) PostgreSQL database for containerized Ansible Automation Platform to use your own database infrastructure.

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

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

IMPORTANT:

- When using an external database with Ansible Automation Platform, you must create and support that database. Ensure that you clear your external database when uninstalling Ansible Automation Platform.
- Red Hat Ansible Automation Platform requires customer provided (external) database to have International Components for Unicode (ICU) support.
- During configuration of an external database, you must check the external database coverage. For more information, see *Red Hat Ansible Automation Platform Database Scope of Coverage* in the related information section.
- The `[database]` group in your inventory file defines the Ansible Automation Platform managed database. When using an externally managed database, do not include the `[database]` group in your inventory file.

Related information

[Red Hat Ansible Automation Platform Database Scope of Coverage](#)

Set up an external database with PostgreSQL admin credentials

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

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

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

Set up an external database without PostgreSQL admin credentials

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

Procedure

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

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

For example:

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

2. Create the user with a password and ensure the `CREATEDB` role is assigned to the user. For more information, see [Database Roles](#).

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

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

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

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

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

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

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

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

Enable the hstore extension

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

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

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

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

Procedure

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

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

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

Example output with hstore available:

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

Example output with hstore not available:

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

3. On a RHEL based server, the hstore extension is included in the postgresql-contrib RPM package, which is not installed automatically when installing the PostgreSQL server RPM package.

To install the RPM package, use the following command:

```
dnf install postgresql-contrib
```

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

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

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

```

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

```

Optional: configure mutual TLS (mTLS) authentication for an external database

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

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

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

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

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

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

Configure custom TLS certificates

Red Hat Ansible Automation Platform uses X.509 certificate and key pairs to secure traffic. These certificates secure internal traffic between Ansible Automation Platform components and external traffic for public UI and API connections.

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

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

Ansible Automation Platform generated certificates

By default, the installation program creates a self-signed Certificate Authority (CA) and uses it to generate self-signed TLS certificates for all Ansible Automation Platform services.

The self-signed CA certificate and key are generated on one node under the `~/aap/tls/` directory and copied to the same location on all other nodes. This CA is valid for 10 years after the initial creation date.

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

User-provided certificates

Set specific variables in your inventory file to replace self-signed certificates with your own TLS certificates and keys. These must be generated by a public or organizational CA before beginning the installation.

Use a custom CA to generate all TLS certificates

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

When you use `ca_tls_cert` and `ca_tls_key`, the installation program automatically creates TLS certificates for each Ansible Automation Platform service using your provided CA certificate. You do not need to define individual service certificate variables (such as `gateway_tls_cert`, `controller_tls_cert`, or `hub_tls_cert`) because the installation program generates these certificates for you.

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

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

Where:

ca_tls_cert

is the path to your custom CA certificate file.

ca_tls_key

is the path to the key file for your custom CA certificate.

Provide custom TLS certificates for each service

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

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

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

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

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

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

```

eda_redis_tls_key=<path_to_tls_key>

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

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

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

```

- If all components share the same fully qualified domain name (FQDN), use the same certificate and key for each service:

```

gateway_tls_cert=/home/user/certs/myhost.example.com.crt
gateway_tls_key=/home/user/certs/myhost.example.com.key
controller_tls_cert=/home/user/certs/myhost.example.com.crt
controller_tls_key=/home/user/certs/myhost.example.com.key
hub_tls_cert=/home/user/certs/myhost.example.com.crt
hub_tls_key=/home/user/certs/myhost.example.com.key
eda_tls_cert=/home/user/certs/myhost.example.com.crt
eda_tls_key=/home/user/certs/myhost.example.com.key
postgresql_tls_cert=/home/user/certs/myhost.example.com.crt
postgresql_tls_key=/home/user/certs/myhost.example.com.key

```

- If components are deployed on separate hosts with different FQDNs, provide a unique certificate for each service:

```

gateway_tls_cert=/home/user/certs/gateway.example.com.crt
gateway_tls_key=/home/user/certs/gateway.example.com.key
controller_tls_cert=/home/user/certs/controller.example.com.crt
controller_tls_key=/home/user/certs/controller.example.com.key
hub_tls_cert=/home/user/certs/hub.example.com.crt
hub_tls_key=/home/user/certs/hub.example.com.key
eda_tls_cert=/home/user/certs/eda.example.com.crt
eda_tls_key=/home/user/certs/eda.example.com.key
postgresql_tls_cert=/home/user/certs/postgresql.example.com.crt
postgresql_tls_key=/home/user/certs/postgresql.example.com.key

```

Considerations for certificates provided per service

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

- Each service has its own `_tls_cert` and `_tls_key` variables. You can provide unique certificates for each service, or use the same certificate across multiple services if they share a fully qualified domain name (FQDN). If you do not define a certificate for a service, the installation program generates a self-signed certificate for that service.
- For services deployed across many nodes (for example, when following the enterprise topology), the provided certificate for that service must include the FQDN of all associated nodes in its Subject Alternative Name (SAN) field.
- If an external-facing service (such as automation controller or platform gateway) is deployed behind a load balancer that performs SSL/TLS offloading, the service's certificate must include the load balancer's FQDN in its SAN field, in addition to the FQDNs of the individual service nodes.

Related information

[Securing networks](#)

Provide a custom CA certificate

When you manually provide TLS certificates for Ansible Automation Platform services (such as `gateway_tls_cert`, `controller_tls_cert`, or `hub_tls_cert`), those certificates might be signed by a custom CA.

Use the `custom_ca_cert` variable to add your CA certificate to the environment for proper authentication and trust of the manually provided certificates.

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

```
custom_ca_cert=<path_to_custom_ca_certificate>
```

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

Receptor certificate considerations

When using a custom certificate for Receptor nodes, the certificate requires the `otherName` field specified in the Subject Alternative Name (SAN) of the certificate with the value `1.3.6.1.4.1.2312.19.1`.

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

Related information

[Above the mesh TLS](#)

Redis certificate considerations

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

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

Use custom Receptor signing keys

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

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


Install containerized Ansible Automation Platform

Run the `install` playbook to install containerized Ansible Automation Platform after preparing the Red Hat Enterprise Linux host, downloading the installation program, and configuring the inventory file.

Before you begin

- You have prepared the Red Hat Enterprise Linux host
- You have prepared the managed nodes
- You have downloaded Ansible Automation Platform
- You have configured the inventory file
- You are logged in to the Red Hat Enterprise Linux host as your non-root user

Procedure

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

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

For example:

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

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

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

For example:

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

- `-i <inventory_file_name>` - The inventory file to use for the installation.

- `-e @<vault_file_name> --ask-vault-pass` - (Optional) If you are using a vault to store sensitive variables, add this to the installation command.
- `-K` - (Optional) If your privilege escalation (becoming root) requires you to enter a password, add this to the installation command. You are then prompted for the BECOME password.
- `-v` - (Optional) You can use increasing verbosity, up to 4 (`-vvvv`) to see installation process details. This can significantly increase installation time. Use it only as needed or when requested by Red Hat support.

Result

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

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

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

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

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

```
envoy_disable_https: true
```

Install in a disconnected environment

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

Obtain and configure RPM source dependencies

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

- Use one of the following methods to access these dependencies in a disconnected environment:
 - Use [Red Hat Satellite](#) to synchronize repositories in your disconnected environment.
 - Use a local repository that you create with the `reposync` command on a Red Hat Enterprise Linux host that has an active internet connection.
 - Use a local repository that you create from a mounted Red Hat Enterprise Linux Binary DVD ISO image.

Configure a local repository using reposync

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

Before you begin

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

Procedure

1. Attach the BaseOS and AppStream repositories using `subscription-manager`, replacing `<RHEL_VERSION>` with your RHEL version number:

```
$ sudo subscription-manager repos \
    --enable rhel-<RHEL_VERSION>-baseos-rhui-rpms \
    --enable rhel-<RHEL_VERSION>-appstream-rhui-rpms
```

2. Install the `yum-utils` package:

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

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

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

For example:

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

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

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

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

5. Move the compressed archive to your disconnected environment.
6. On the disconnected environment, create a directory to store the repository files:

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

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

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

8. Create a Yum repository file at `/etc/yum.repos.d/rhel.repo` with the following content, replacing `<RHEL_VERSION>` with your RHEL version number:

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

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

9. Import the gpg key to allow the system to verify the packages, replacing `<RHEL_VERSION>` with your RHEL version number:

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

10. Verify the repository configuration:

```
$ sudo yum repolist
```

Configure a local repository from a mounted ISO

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

Before you begin

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

Procedure

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

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

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

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

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

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

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

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

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

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

5. Verify the repository configuration:

```
$ sudo yum repolist
```

Perform a disconnected installation

A disconnected installation installs containerized Ansible Automation Platform without requiring network access to external registries.

Before you begin

- You have prepared the Red Hat Enterprise Linux host
- You have obtained and configured the RPM source dependencies. The installation program uses your host system's `dnf` package manager to resolve these dependencies.
- You have prepared the managed nodes
- You have downloaded the containerized Ansible Automation Platform setup bundle from the [Ansible Automation Platform download page](#).

Procedure

1. Log in to the Red Hat Enterprise Linux host as your non-root user.
2. Update the inventory file by following the steps in [Configure the inventory file](#).

NOTE:

Do not include `registry_username` or `registry_password` in your inventory file for disconnected installations. These variables are only required for online installations. All container images are pre-packaged in the setup bundle.

3. Ensure you include the following variables in your inventory file under the `[all:vars]` group:

```
bundle_install=true

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

4. Follow the steps in [Install containerized Ansible Automation Platform](#) to install containerized Ansible Automation Platform and verify your installation.

Maintain containerized Ansible Automation Platform

Update, backup, restore, uninstall, or reinstall containerized Ansible Automation Platform deployments to support your automation infrastructure.

Uninstall containerized Ansible Automation Platform

Uninstall your container-based installation of Ansible Automation Platform.

Before you begin

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

Procedure

1. If you intend to reinstall Ansible Automation Platform and want to use the preserved databases, you must collect the existing secret keys:
 - a. First, list the available secrets:

```
$ podman secret list
```

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

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

For example:

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

2. Run the `uninstall` playbook:

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


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

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

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

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

Reinstall containerized Ansible Automation Platform

Reinstall a containerized deployment after uninstalling and preserving the database.

- Follow the steps in [See "Install containerized Ansible Automation Platform" on page 153](#) and include the existing secret key value in the playbook command:

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

Inventory file variables

The following tables contain information about the variables used in Ansible Automation Platform's installation `inventory` files.

Ansible variables

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

Ansible variables

Variable	Description
<code>ansible_connection</code>	The connection plugin used for the task on the target host. This can be the name of any Ansible connection plugin. SSH protocol types are <code>smart</code>, <code>ssh</code>, or <code>paramiko</code>. You can also use <code>local</code> to run tasks on the control node itself. Default = <code>smart</code>
<code>ansible_host</code>	The IP address or name of the target host to use instead of <code>inventory_hostname</code> .
<code>ansible_password</code>	The password to authenticate to the host. Do not store this variable in plain text. Always use a vault.
<code>ansible_port</code>	The connection port number. The default for SSH is <code>22</code>.
<code>ansible_scp_extra_args</code>	This setting is always appended to the default <code>scp</code> command line.
<code>ansible_sftp_extra_args</code>	This setting is always appended to the default <code>sftp</code> command line.
<code>ansible_shell_executable</code>	This sets the shell that the Ansible controller uses on the target machine and overrides the executable in <code>ansible.cfg</code> which defaults to <code>/bin/sh</code> .
<code>ansible_shell_type</code>	The shell type of the target system. Do not use this setting unless you have set the <code>ansible_shell_executable</code> to a non-Bourne (sh) compatible shell. By default commands are formatted using sh-style syntax. Setting this to <code>csh</code> or <code>fish</code> causes

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

Automation hub variables

Reference information for the inventory file variables used to configure automation hub.

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

RPM variable name	Container variable name	Description	Required or optional	Default
<code>automationhub_collection_download_count</code>		Denote whether or not the collection download count should be displayed in the UI.	Optional	<code>false</code>
<code>automationhub_collection_seed_repository</code>		Controls the type of content to upload when <code>hub_seed_collections</code> is set to <code>true</code> . Valid options include: <code>certified</code> , <code>validated</code>	Optional	Both <code>certified</code> and <code>validated</code> are enabled by default.
<code>automationhub_collection_signing_service_key</code>	<code>hub_collection_signing_key</code>	Path to the collection signing key file.	Required if a collection signing service is enabled.	
<code>automationhub_container_repair_media_type</code>		Denote whether or not to run the command <code>pulpcore-manager container-repair-media-type</code> . Valid options include: <code>true</code> , <code>false</code> , <code>auto</code>	Optional	<code>auto</code>
<code>automationhub_container_signing_service_key</code>	<code>hub_container_signing_key</code>	Path to the container signing key file.	Required if a container signing service is	

RPM variable name	Container variable name	Description	Required or optional	Default
			enabled.	
<code>automationhub_create_default_collection_signing_service</code>	<code>hub_collection_signing</code>	Set this variable to <code>true</code> to enable a collection signing service.	Optional	<code>false</code>
<code>automationhub_create_default_container_signing_service</code>	<code>hub_container_signing</code>	Set this variable to <code>true</code> to enable a container signing service.	Optional	<code>false</code>
	<code>hub_data_path_exclude</code>	automation hub backup path to exclude.	Optional	<code>[]</code>
<code>automationhub_disable_hsts</code>	<code>hub_nginx_disable_hsts</code>	Controls whether HTTP Strict Transport Security (HSTS) is enabled or disabled for automation hub. Set this variable to <code>true</code> to disable HSTS.	Optional	<code>false</code>
<code>automationhub_disable_https</code>	<code>hub_nginx_disable_https</code>	Controls whether HTTPS is enabled or disabled for automation hub. Set this variable to <code>true</code> to disable HTTPS.	Optional	<code>false</code>
<code>automationhub_enable_api_access_log</code>		Controls whether logging is enabled or disabled at <code>/var/log/galaxy_api_access.log</code> . The file logs all user actions made to the platform, including username and IP address. Set this variable to <code>true</code> to enable this logging.	Optional	<code>false</code>
<code>automationhub_enable_unauthenticated_collection_access</code>		Controls whether read-only access is enabled or disabled for unauthorized users viewing collections or	Optional	<code>false</code>

RPM variable name	Container variable name	Description	Required or optional	Default
		namespaces for automation hub. Set this variable to <code>true</code> to enable read-only access.		
<code>automationhub_enable_unauthenticated_collection_download</code>		Controls whether or not unauthorized users can download read-only collections from automation hub. Set this variable to <code>true</code> to enable download of read-only collections.	Optional	<code>false</code>
<code>automationhub_firewalld_zone</code>	<code>hub_firewall_zone</code>	The firewall zone where automation hub related firewall rules are applied. This controls which networks can access automation hub based on the zone's trust level.	Optional	RPM = no default set. Container = <code>public</code> .
<code>automationhub_force_change_admin_password</code>		Denote whether or not to require the change of the default administrator password for automation hub during installation. Set to <code>true</code> to require the user to change the default administrator password during installation.	Optional	<code>false</code>
<code>automationhub_importer_settings</code>	<code>hub_galaxy_importer</code>	Dictionary of settings to pass to the <code>galaxy-importer.cfg</code> configuration file. These settings control how the <code>galaxy-importer</code> service processes and validates Ansible content. Example values include: <code>ansible-doc</code> ,	Optional	

RPM variable name	Container variable name	Description	Required or optional	Default
		<code>ansible-lint</code> , and <code>flake8</code> .		
<code>automationhub_nginx_tls_files_remote</code>		Denote whether the web certificate sources are local to the installation program (<code>false</code>) or on the remote component server (<code>true</code>).	Optional	The value defined in <code>automationhub_tls_files_remote</code> .
<code>automationhub_pg_cert_auth</code>	<code>hub_pg_cert_auth</code>	Controls whether client certificate authentication is enabled or disabled on the automation hub PostgreSQL database. Set this variable to <code>true</code> to enable client certificate authentication.	Optional	<code>false</code>
<code>automationhub_pg_database</code>	<code>hub_pg_database</code>	Name of the PostgreSQL database used by automation hub.	Optional	RPM = <code>automationhub</code> . Container = <code>pulp</code>
<code>automationhub_pg_host</code>	<code>hub_pg_host</code>	Hostname of the PostgreSQL database used by automation hub.	Required	RPM = <code>127.0.0.1</code> . Container

RPM variable name	Container variable name	Description	Required or optional	Default
				= no default.
<code>automationhub_pg_password</code>	<code>hub_pg_password</code>	Password for the automation hub PostgreSQL database user. Use of special characters for this variable is limited. The <code>!</code> , <code>#</code> , <code>0</code> and <code>@</code> characters are supported. Use of other special characters can cause the setup to fail.	Optional	
<code>automationhub_pg_port</code>	<code>hub_pg_port</code>	Port number for the PostgreSQL database used by automation hub.	Optional	5432
<code>automationhub_pg_sslmode</code>	<code>hub_pg_sslmode</code>	Controls the SSL/TLS mode to use when automation hub connects to the PostgreSQL database. Valid options include <code>verify-full</code> , <code>verify-ca</code> , <code>require</code> , <code>prefer</code> , <code>allow</code> , <code>disable</code> .	Optional	<code>prefer</code>
<code>automationhub_pg_username</code>	<code>hub_pg_username</code>	Username for the automation hub PostgreSQL database user.	Optional	RPM = <code>automationhub</code> . Container = <code>pulp</code> .
<code>automationhub_pg_client_sslcert</code>	<code>hub_pg_tls_cert</code>	Path to the PostgreSQL SSL/TLS certificate file for automation hub.	Required if using	

RPM variable name	Container variable name	Description	Required or optional	Default
			client certificate authentication.	
<code>automationhub_pg_client_sslkey</code>	<code>hub_pg_tls_key</code>	Path to the PostgreSQL SSL/TLS key file for automation hub.	Required if using client certificate authentication.	
<code>automationhub_pg_client_tls_files_remote</code>		Denote whether the PostgreSQL client certificate sources are local to the installation program (<code>false</code>) or on the remote component server (<code>true</code>).	Optional	The value defined in <code>automationhub_tls_files_remote</code> .
<code>automationhub_require_content_approval</code>		Controls whether content signing is enabled or disabled for automation hub. By default when you upload collections to automation hub, an administrator must approve it before they are made available to users. To disable the content approval flow, set the variable to <code>false</code> .	Optional	<code>true</code>

RPM variable name	Container variable name	Description	Required or optional	Default
<code>automationhub_restore_signing_keys</code>		Controls whether or not existing signing keys should be restored from a backup. Set to <code>false</code> to disable restoration of existing signing keys.	Optional	<code>true</code>
<code>automationhub_seed_collections</code>	<code>hub_seed_collections</code>	Controls whether or not pre-loading of collections is enabled. When you run the bundle installer, validated content is uploaded to the <code>validated</code> repository, and certified content is uploaded to the <code>rh-certified</code> repository. By default, certified content and validated content are both uploaded. If you do not want to pre-load content, set this variable to <code>false</code> . For the RPM-based installer, if you only want one type of content, set this variable to <code>true</code> and set the <code>automationhub_collection_seed_repository</code> variable to the type of content you want to include.	Optional	<code>true</code>
<code>automationhub_ssl_cert</code>	<code>hub_tls_cert</code>	Path to the SSL/TLS certificate file for automation hub.	Optional	
<code>automationhub_ssl_key</code>	<code>hub_tls_key</code>	Path to the SSL/TLS key file for automation hub.	Optional	
<code>automationhub_tls_files_remote</code>	<code>hub_tls_remote</code>	Denote whether the automation hub provided certificate files are local to the	Optional	<code>false</code>

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

RPM variable name	Container variable name	Description	Required or optional	Default
		installation. By default, a token is automatically generated during a fresh installation. If set to <code>true</code> , a token is regenerated during installation.		
	<code>hub_extra_settings</code>	Defines additional settings for use by automation hub during installation. For example: <pre>hub_extra_settings=[{"setting": "REDIRECT_IS_HTTPS", "value": True}]</pre>	Optional	<code>[]</code>
<code>nginx_hsts_max_age</code>	<code>hub_nginx_hsts_max_age</code>	Maximum duration (in seconds) that HTTP Strict Transport Security (HSTS) is enforced for automation hub.	Optional	<code>63072000</code>
<code>pulp_secret</code>	<code>hub_secret_key</code>	Secret key value used by automation hub to sign and encrypt data.	Optional	
	<code>hub_azure_account_key</code>	Azure blob storage account key.	Required if using an Azure blob storage backend.	

RPM variable name	Container variable name	Description	Required or optional	Default
	<code>hub_azure_account_name</code>	Account name associated with the Azure blob storage.	Required when using an Azure blob storage backend.	
	<code>hub_azure_container</code>	Name of the Azure blob storage container.	Optional	<code>pulp</code>
	<code>hub_azure_extra_settings</code>	Defines extra parameters for the Azure blob storage backend. For more information about the list of parameters, see django-storages documentation - Azure Storage .	Optional	<code>{}</code>
	<code>hub_collection_signing_pass</code>	Password for the automation content collection signing service.	Required if the collection signing service is protected by a passphrase.	
	<code>hub_collection_signing_service</code>	Service for signing collections.	Optional	<code>ansible</code>

RPM variable name	Container variable name	Description	Required or optional	Default
				default
	hub_container_signing_pass	Password for the automation content container signing service.	Required if the container signing service is protected by a passphrase.	
	hub_container_signing_service	Service for signing containers.	Optional	container-default
	hub_nginx_http_port	Port number that automation hub listens on for HTTP requests.	Optional	8081
	hub_nginx_https_port	Port number that automation hub listens on for HTTPS requests.	Optional	8444
nginx_tls_protocols	hub_nginx_https_protocols	Protocols that automation hub will support when handling HTTPS traffic.	Optional	[TLS v1.2, TLSv1.3]
	hub_pg_socket	UNIX socket used by automation hub to connect to the PostgreSQL database.	Optional	
	hub_s3_access_key	AWS S3 access key.	Required	

RPM variable name	Container variable name	Description	Required or optional	Default
			if using an AWS S3 storage backend.	
	<code>hub_s3_bucket_name</code>	Name of the AWS S3 storage bucket.	Optional	<code>pullp</code>
	<code>hub_s3_extra_settings</code>	Used to define extra parameters for the AWS S3 storage backend. For more information about the list of parameters, see django-storages documentation - Amazon S3 .	Optional	<code>{}</code>
	<code>hub_s3_secret_key</code>	AWS S3 secret key.	Required if using an AWS S3 storage backend.	
	<code>hub_shared_data_mount_opts</code>	Mount options for the Network File System (NFS) share.	Optional	<code>rw, sync, hard</code>
	<code>hub_shared_data_path</code>	Path to the Network File System (NFS) share with read, write, and execute (RWX) access. The value must match the format <code>host:dir</code> , for example <code>nfs-</code>	Required if installing more than one	

RPM variable name	Container variable name	Description	Required or optional	Default
		<code>server.example.com:/exports/hub.</code>	instance of automation hub with a <code>file</code> storage backend. When installing a single instance of automation hub, it is optional.	
	<code>hub_storage_backend</code>	Automation hub storage backend type. Possible values include: <code>azure</code> , <code>file</code> , <code>s3</code> .	Optional	<code>file</code>
	<code>hub_workers</code>	Number of automation hub workers.	Optional	<code>2</code>

Automation controller variables

Inventory file variables for automation controller.

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

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

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

RPM variable name	Container variable name	Description	Required or optional	Default
		listens on for HTTPS requests.		Container = 8443
<code>nginx_tls_protocols</code>	<code>controller_nginx_https_protocols</code>	Protocols that automation controller supports when handling HTTPS traffic.	Optional	[TLS v1.2, TLSv1.3]
<code>nginx_user_headers</code>	<code>controller_nginx_user_headers</code>	List of additional NGINX headers to add to automation controller's NGINX configuration.	Optional	[]
	<code>controller_create_preload_data</code>	Controls whether or not to create preloaded content during installation.	Optional	true
<code>node_state</code>		The status of a node or group of nodes. Valid options include <code>active</code> , <code>deprovision</code> to remove a node from a cluster, or <code>iso_migrate</code> to migrate a legacy isolated node to an execution node.	Optional	active
<code>node_type</code>	See <code>receptor_type</code> for the container equivalent variable.	For the <code>[automationcontroller]</code> group the two options are: <code>node_type=controller</code> - The node only runs project and inventory updates, but not regular jobs.	Optional	For <code>[automationcontroller]</code> = <code>hybrid</code> , for <code>[execution_node]</code>

RPM variable name	Container variable name	Description	Required or optional	Default
		<code>node_type=hybrid</code> - The node runs everything. For the <code>[execution_nodes]</code> group the two options are: <code>node_type=hop</code> - The node forwards jobs to an execution node. <code>node_type=execution</code> - The node can run jobs.		<code>odes]</code> = exec ution
<code>peers</code>	See <code>receptor_peers</code> for the container equivalent variable.	Used to indicate which nodes a specific host or group connects to. Wherever this variable is defined, an outbound connection to the specific host or group is established. This variable can be a comma-separated list of hosts and groups from the inventory. This is resolved into a set of hosts that is used to construct the <code>receptor.conf</code> file.	Optional	
<code>pg_database</code>	<code>controller_pg_database</code>	Name of the PostgreSQL database used by automation controller.	Optional	<code>awx</code>
<code>pg_host</code>	<code>controller_pg_host</code>	Hostname of the PostgreSQL database used by automation controller.	Required	

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

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

RPM variable name	Container variable name	Description	Required or optional	Default
		<pre>controller_extra_settings=[{"setting": "USE_X_FORWARDED_HOST", "value": True}]</pre>		
	<code>controller_license_file</code>	Path to the automation controller license file.		
	<code>controller_percent_memory_capacity</code>	Memory allocation for automation controller.	Optional	1.0 (allocates 100% of the total system memory to automation controller)
	<code>controller_pg_socket</code>	UNIX socket used by automation controller to connect to the PostgreSQL database.	Optional	
	<code>controller_secret_key</code>	Secret key value used by automation controller to sign and encrypt data.	Optional	

Database variables

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

RPM variable name	Container variable name	Description	Required or optional	Default
<code>install_pg_port</code>	<code>postgresql_port</code>	Port number for the PostgreSQL database.	Optional	5432
<code>postgres_extra_settings</code>	<code>postgresql_extra_settings</code>	Defines additional settings for use by PostgreSQL. Example usage for RPM: <pre>postgresql_extra_settings={ 'ssl_ciphers': 'HIGH:!aNULL:!MD5' }</pre> Example usage for containerized: <pre>postgresql_extra_settings=[{"setting": "ssl_ciphers", "value": "HIGH:!aNULL:!MD5"}]</pre>	Optional	
<code>postgres_firewall_zone</code>	<code>postgresql_firewall_zone</code>	The firewall zone where PostgreSQL related firewall rules are applied. This controls which networks can access PostgreSQL based on the zone's trust level.	Optional	RPM = no default set. Container = <code>public</code> .
<code>postgres_max_connections</code>	<code>postgresql_max_connections</code>	Maximum number of concurrent connections to the database if you are using an installer-managed database.	Optional	1024
<code>postgres_ssl_certificate</code>	<code>postgresql_tls_certificate</code>	Path to the PostgreSQL SSL/TLS certificate file.	Optional	

RPM variable name	Container variable name	Description	Required or optional	Default
<code>postgres_ssl_key</code>	<code>postgresql_tls_key</code>	Path to the PostgreSQL SSL/TLS key file.	Optional	
<code>postgres_use_ssl</code>	<code>postgresql_disable_tls</code>	Controls whether SSL/TLS is enabled or disabled for the PostgreSQL database.	Optional	<code>false</code>
	<code>postgresql_admin_database</code>	Database name used for connections to the PostgreSQL database server.	Optional	<code>postgres</code>
	<code>postgresql_admin_password</code>	Password for the PostgreSQL admin user. When used, the installation program creates each component's database and credentials.	Required if using <code>postgresql_admin_username</code> .	
	<code>postgresql_admin_username</code>	Username for the PostgreSQL admin user. When used, the installation program creates each component's database and credentials.	Optional	<code>postgres</code>
	<code>postgresql_effective_cache_size</code>	Memory allocation available (in MB) for caching data.	Optional	
	<code>postgresql_keep_databases</code>	Controls whether or not to keep databases during uninstall. This variable applies to databases managed by the installation program only, and not external (customer-managed) databases. Set to <code>true</code>	Optional	<code>false</code>

RPM variable name	Container variable name	Description	Required or optional	Default
		to keep databases during uninstall.		
	<code>postgresql_log_destination</code>	Destination for server log output.	Optional	<code>/dev/stderr</code>
	<code>postgresql_password_encryption</code>	The algorithm for encrypting passwords.	Optional	<code>scram-sha-256</code>
	<code>postgresql_shared_buffers</code>	Memory allocation (in MB) for shared memory buffers.	Optional	
	<code>postgresql_tls_remote</code>	Denote whether the PostgreSQL provided certificate files are local to the installation program (<code>false</code>) or on the remote component server (<code>true</code>).	Optional	<code>false</code>
	<code>postgresql_use_archive_compression</code>	Controls whether archive compression is enabled or disabled for PostgreSQL. You can control this functionality globally by using <code>use_archive_compression</code> .	Optional	<code>true</code>

Event-Driven Ansible controller variables

Inventory file variables for Event-Driven Ansible controller.

RPM variable name	Container variable name	Description	Required or optional	Default
<code>automationedacontroller_activation_workers</code>	<code>eda_activation_workers</code>	Number of workers used for ansible-rulebook activation pods in Event-Driven Ansible.	Optional	RPM = (# of cores or threads) * 2 + 1. Container = 2
<code>automationedacontroller_admin_email</code>	<code>eda_admin_email</code>	Email address used by Django for the admin user for Event-Driven Ansible.	Optional	<code>admin@example.com</code>
<code>automationedacontroller_admin_password</code>	<code>eda_admin_password</code>	Event-Driven Ansible administrator password. Use of special characters for this variable is limited. The password can include any printable ASCII character except <code>/</code> , <code>"</code> , or <code>@</code> .	Required	
<code>automationedacontroller_admin_username</code>	<code>eda_admin_user</code>	Username used to identify and create the administrator user in Event-Driven Ansible.	Optional	<code>admin</code>
<code>automationedacontroller_backend_unicorn_workers</code>		Number of workers for handling the API served through Unicorn on worker nodes.	Optional	2
<code>automationedacontroller_cache_tls_files_remote</code>		Denote whether the cache cert sources are local to the installation program (<code>false</code>) or on the remote component server (<code>true</code>).	Optional	<code>false</code>

RPM variable name	Container variable name	Description	Required or optional	Default
<code>automationedacontroller_client_regen_cert</code>		Controls whether or not to regenerate Event-Driven Ansible client certificates for the platform cache. Set to <code>true</code> to regenerate Event-Driven Ansible client certificates.	Optional	<code>false</code>
<code>automationedacontroller_default_workers</code>	<code>eda_workers</code>	Number of workers used in Event-Driven Ansible for application work.	Optional	Number of cores or threads
<code>automationedacontroller_disable_hsts</code>	<code>eda_nginx_disable_hsts</code>	Controls whether HTTP Strict Transport Security (HSTS) is enabled or disabled for Event-Driven Ansible. Set this variable to <code>true</code> to disable HSTS.	Optional	<code>false</code>
<code>automationedacontroller_disable_https</code>	<code>eda_nginx_disable_https</code>	Controls whether HTTPS is enabled or disabled for Event-Driven Ansible. Set this variable to <code>true</code> to disable HTTPS.	Optional	<code>false</code>
<code>automationedacontroller_event_stream_mtls</code>	<code>eda_event_stream_mtls</code>	Controls whether event stream mutual TLS (mTLS) authentication is enabled or disabled for Event-Driven Ansible. Set this variable to <code>false</code> to disable mTLS authentication.	Optional	<code>true</code>
<code>automationedacontroller_event_stream_mtls_path</code>	<code>eda_event_stream_mtls_prefix_path</code>	The prefix path for the event stream mTLS URLs.	Optional	<code>/mtls/eda-even</code>

RPM variable name	Container variable name	Description	Required or optional	Default
				t-streams
automationedacontroller_event_stream_path	eda_event_stream_prefix_path	API prefix path used for Event-Driven Ansible event-stream through platform gateway.	Optional	/eda-event-streams
automationedacontroller_firewall_zone	eda_firewall_zone	The firewall zone where Event-Driven Ansible related firewall rules are applied. This controls which networks can access Event-Driven Ansible based on the zone's trust level.	Optional	RPM = no default set. Container = public.
automationedacontroller_unicorn_event_stream_workers		Number of workers for handling event streaming for Event-Driven Ansible.	Optional	2
automationedacontroller_unicorn_workers	eda_unicorn_workers	Number of workers for handling the API served through Unicorn.	Optional	(Number of cores or threads) * 2 + 1
automationedacontroller_http_port	eda_nginx_http_port	Port number that Event-Driven Ansible listens on for HTTP requests.	Optional	RPM = 80. Container = 8082.

RPM variable name	Container variable name	Description	Required or optional	Default
<code>automationedacontroller_https_port</code>	<code>eda_nginx_https_port</code>	Port number that Event-Driven Ansible listens on for HTTPS requests.	Optional	RPM = 443 . Container = 8445 .
<code>automationedacontroller_nginx_tls_files_remote</code>		Denote whether the web cert sources are local to the installation program (<code>false</code>) or on the remote component server (<code>true</code>).	Optional	<code>false</code>
<code>automationedacontroller_pg_cert_auth</code>	<code>eda_pg_cert_auth</code>	Controls whether client certificate authentication is enabled or disabled on the Event-Driven Ansible PostgreSQL database. Set this variable to <code>true</code> to enable client certificate authentication.	Optional	<code>false</code>
<code>automationedacontroller_pg_database</code>	<code>eda_pg_database</code>	Name of the PostgreSQL database used by Event-Driven Ansible.	Optional	RPM = <code>automationedacontroller</code> . Container = <code>eda</code> .
<code>automationedacontroller_pg_host</code>	<code>eda_pg_host</code>	Hostname of the PostgreSQL database used by Event-Driven Ansible.	Required	

RPM variable name	Container variable name	Description	Required or optional	Default
<code>automationedacontroller_pg_password</code>	<code>eda_pg_password</code>	Password for the Event-Driven Ansible PostgreSQL database user. Use of special characters for this variable is limited. The <code>!</code> , <code>#</code> , <code>0</code> and <code>@</code> characters are supported. Use of other special characters can cause the setup to fail.	Required if not using client certificate authentication.	
<code>automationedacontroller_pg_port</code>	<code>eda_pg_port</code>	Port number for the PostgreSQL database used by Event-Driven Ansible.	Optional	5432
<code>automationedacontroller_pg_sslmode</code>	<code>eda_pg_sslmode</code>	Determines the level of encryption and authentication for client server connections. Valid options include <code>verify-full</code> , <code>verify-ca</code> , <code>require</code> , <code>prefer</code> , <code>allow</code> , <code>disable</code> .	Optional	<code>prefer</code>
<code>automationedacontroller_pg_username</code>	<code>eda_pg_username</code>	Username for the Event-Driven Ansible PostgreSQL database user.	Optional	RPM = <code>automationedacontroller</code> . Container = <code>eda</code> .
<code>automationedacontroller_pgclient_sslcert</code>	<code>eda_pg_tls_cert</code>	Path to the PostgreSQL SSL/TLS certificate file for Event-Driven Ansible.	Required if using client certificate	

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

RPM variable name	Container variable name	Description	Required or optional	Default
				string
<code>automationedacontroller_redis_port</code>	<code>eda_redis_port</code>	Port number for the Redis host for Event-Driven Ansible.	Optional	RPM = The value defined in platform gateway's implementation (<code>automationgateway_redis_port</code>). Container = 6379
<code>automationedacontroller_redis_username</code>	<code>eda_redis_username</code>	Username for Event-Driven Ansible Redis.	Optional	<code>eda</code>
<code>automationedacontroller_secret_key</code>	<code>eda_secret_key</code>	Secret key value used by Event-Driven Ansible to sign and encrypt data.	Optional	
<code>automationedacontroller_ssl_cert</code>	<code>eda_tls_cert</code>	Path to the SSL/TLS certificate file for Event-Driven Ansible.	Optional	
<code>automationedacontroller_ssl_key</code>	<code>eda_tls_key</code>	Path to the SSL/TLS key file for Event-Driven Ansible.	Optional	

RPM variable name	Container variable name	Description	Required or optional	Default
<code>automationedacontroller_tls_files_remote</code>	<code>eda_tls_remote</code>	Denote whether the Event-Driven Ansible provided certificate files are local to the installation program (<code>false</code>) or on the remote component server (<code>true</code>).	Optional	<code>false</code>
<code>automationedacontroller_trusted_origins</code>		List of host addresses in the form: <code><scheme>://<address>:<port></code> for trusted Cross-Site Request Forgery (CSRF) origins.	Optional	<code>[]</code>
<code>automationedacontroller_use_archive_compression</code>	<code>eda_use_archive_compression</code>	Controls whether archive compression is enabled or disabled for Event-Driven Ansible. You can control this functionality globally by using <code>use_archive_compression</code> .	Optional	<code>true</code>
<code>automationedacontroller_use_db_compression</code>	<code>eda_use_db_compression</code>	Controls whether database compression is enabled or disabled for Event-Driven Ansible. You can control this functionality globally by using <code>use_db_compression</code> .	Optional	<code>true</code>
<code>automationedacontroller_user_headers</code>	<code>eda_nginx_user_headers</code>	List of additional NGINX headers to add to Event-Driven Ansible's NGINX configuration.	Optional	<code>[]</code>
<code>automationedacontroller_websocket_ssl_verify</code>		Controls whether or not to perform SSL verification for the Daphne WebSocket used by Podman to communicate from the	Optional	<code>true</code>

RPM variable name	Container variable name	Description	Required or optional	Default
		pod to the host. Set to <code>false</code> to disable SSL verification.		
<code>eda_node_type</code>	<code>eda_type</code>	Event-Driven Ansible node type. Valid options include <code>api</code> , <code>event-stream</code> , <code>hybrid</code> , <code>worker</code> .	Optional	<code>hybrid</code>
	<code>eda_debug</code>	Controls whether debug mode is enabled or disabled for Event-Driven Ansible. Set to <code>true</code> to enable debug mode for Event-Driven Ansible.	Optional	<code>false</code>
	<code>eda_extra_settings</code>	Defines additional settings for use by Event-Driven Ansible during installation. For example: <pre>eda_extra_settings=[{"setting": "RULEBOOK_READINESS_TIMEOUT_SECONDS", "value": 120}]</pre>	Optional	<code>[]</code>
	<code>eda_nginx_client_max_body_size</code>	Maximum allowed size for data sent to Event-Driven Ansible through NGINX.	Optional	<code>1m</code>
	<code>eda_nginx_hsts_max_age</code>	Maximum duration (in seconds) that HTTP Strict Transport Security (HSTS) is enforced for Event-Driven Ansible.	Optional	<code>63072000</code>
<code>nginx_tls_protocols</code>	<code>eda_nginx_https_protocols</code>	Protocols that Event-Driven Ansible supports	Optional	<code>[TLS v1.2, ...]</code>

RPM variable name	Container variable name	Description	Required or optional	Default
		when handling HTTPS traffic.		TLSv1.3]
	eda_pg_socket	UNIX socket used by Event-Driven Ansible to connect to the PostgreSQL database.	Optional	
redis_disable_tls	eda_redis_disable_tls	Controls whether TLS is enabled or disabled for Event-Driven Ansible Redis. Set this variable to true to disable TLS.	Optional	false
	eda_redis_tls_cert	Path to the Event-Driven Ansible Redis certificate file.	Optional	
	eda_redis_tls_key	Path to the Event-Driven Ansible Redis key file.	Optional	
	eda_safe_plugins	List of plugins that are allowed to run within Event-Driven Ansible. For more information, see See "Add a safe plugin variable to Event-Driven Ansible controller" on page 134.	Optional	[]

General variables

General inventory file variables for Ansible Automation Platform.

RPM variable name	Container variable name	Description	Required or optional	Default
aap_ca_cert_file	ca_tls_cert	Path to the user-provided CA certificate	Optional	

RPM variable name	Container variable name	Description	Required or optional	Default
		file. When you specify this variable, the installation program automatically generates TLS certificates for each Ansible Automation Platform service signed by this CA. You do not need to define individual service certificate variables.		
<code>aap_ca_certificate_remote</code>	<code>ca_tls_remote</code>	Denote whether the CA certificate files are local to the installation program (<code>false</code>) or on the remote component server (<code>true</code>).	Optional	<code>false</code>
<code>aap_ca_certificate_size</code>		Bit size of the internally managed CA certificate private key.	Optional	<code>4096</code>
<code>aap_ca_key_file</code>	<code>ca_tls_key</code>	Path to the key file for the CA certificate provided in <code>aap_ca_certificate_file</code> (RPM) and <code>ca_tls_certificate</code> (Container). The installation program uses this key to sign the automatically generated TLS certificates for each Ansible Automation Platform service.	Optional	
<code>aap_ca_certificate_private_cipher</code>		Cipher used for signing the internally managed CA certificate private key.	Optional	<code>aes256</code>
<code>aap_ca_certificate_regenerate</code>		Denotes whether or not to regenerate the internally managed CA certificate key pair.	Optional	<code>false</code>

RPM variable name	Container variable name	Description	Required or optional	Default
<code>aap_service_cert_size</code>		Bit size of the component key pair managed by the internal CA.	Optional	4096
<code>aap_service_regen_cert</code>		Denotes whether or not to regenerate the component key pair managed by the internal CA.	Optional	false
<code>aap_service_san_records</code>		A list of additional SAN records for signing a service. Assign these to components in the inventory file as host variables rather than group or all variables. All strings must also contain their corresponding SAN option prefix such as <code>DNS:</code> or <code>IP:</code> .	Optional	[]
<code>backup_dest</code>		Directory local to <code>setup.sh</code> for the final backup file.	Optional	The value defined in <code>setup_dir</code> .
<code>backup_dir</code>	<code>backup_dir</code>	Directory used to store backup files.	Optional	RPM = <code>/var/backups/automation-platform/</code> . Container = <code>~/</code>

RPM variable name	Container variable name	Description	Required or optional	Default
				backups
backup_file_prefix		Prefix used for the file backup name for the final backup file.	Optional	automation-platform-backup
bundle_install	bundle_install	Controls whether or not to perform an offline or bundled installation. Set this variable to <code>true</code> to enable an offline or bundled installation.	Optional	false if using the setup installation program. true if using the setup bundle installation program.
bundle_install_folder	bundle_dir	Path to the bundle directory used when performing a bundle install.	Required if <code>bundle_install=true</code>	RPM = <code>/var/lib/ansible-automation-platform-bund</code>

RPM variable name	Container variable name	Description	Required or optional	Default
				le . Container = <current_dir >/ bundle .
custom_ca_cert	custom_ca_cert	Path to the custom CA certificate file. Use this variable when you have manually provided TLS certificates for Ansible Automation Platform services (such as <code>gateway_tls_cert</code>, <code>controller_tls_cert</code>, or <code>hub_tls_cert</code>) that are signed by a custom CA. This variable adds the CA certificate to the environment to ensure proper authentication and trust of the manually provided certificates. This variable is not needed when using <code>ca_tls_cert</code> and <code>ca_tls_key</code>, which automatically generate TLS certificates.	Optional	
enable_insights_collection		The default install registers the node to the Red Hat Insights for Red Hat Ansible Automation Platform for the Red Hat Ansible Automation Platform Service if the node is registered with Subscription Manager.	Optional	true

RPM variable name	Container variable name	Description	Required or optional	Default
		Set to <code>false</code> to disable this functionality.		
<code>registry_password</code>	<code>registry_password</code>	Password credential for access to the registry source defined in <code>registry_url</code>. For more information, see See "Create a registry service account" on page 133. Not required for disconnected (bundled) installations where <code>bundle_install=true</code>.	RPM = Required if you need a password to access <code>registry_url</code>. Container = Required for online installations if <code>registry_auth=true</code>. Not required for disconnected installations.	

RPM variable name	Container variable name	Description	Required or optional	Default
<code>registry_url</code>	<code>registry_url</code>	URL of the registry source from which to pull execution environment images.	Optional	<code>registry.redhat.io</code>
<code>registry_username</code>	<code>registry_username</code>	Username credential for access to the registry source defined in <code>registry_url</code>. For more information, see See "Create a registry service account" on page 133. Not required for disconnected (bundled) installations where <code>bundle_install=true</code>.	RPM = Required if you need a password to access <code>registry_url</code>. Container = Required for online installations if <code>registry_auth=true</code>. Not required for disconnected	

RPM variable name	Container variable name	Description	Required or optional	Default
			installations.	
<code>registry_verify_ssl</code>	<code>registry_tls_verify</code>	Controls whether SSL/TLS certificate verification is enabled or disabled when making HTTPS requests.	Optional	<code>true</code>
<code>restore_backup_file</code>		Path to the tar file used for the platform restore.	Optional	<code>{{ setup_dir }}/automation-platform-backup-latest.tar.gz</code>
<code>restore_file_prefix</code>		Path prefix for the staged restore components.	Optional	<code>automation-platform-restore</code>
<code>routable_hostname</code>	<code>routable_hostname</code>	Used if the machine running the installation program can only route to the target host through a specific URL. For example, if you use short names in your inventory, but the node running the installation program can only	Optional	

RPM variable name	Container variable name	Description	Required or optional	Default
		resolve that host by using a FQDN. If <code>routable_hostname</code> is not set, it defaults to <code>ansible_host</code>. If you do not set <code>ansible_host</code>, <code>inventory_hostname</code> is used as a last resort. This variable is used as a host variable for particular hosts and not under the <code>[all:vars]</code> section.		
<code>use_archive_compression</code>	<code>use_archive_compression</code>	Controls at a global level whether the filesystem-related backup files are compressed before being sent to the host to run the backup operation. If set to <code>true</code>, a <code>tar.gz</code> file is generated on each Ansible Automation Platform host and then gzip compression is used. If set to <code>false</code>, a simple tar file is generated. You can control this functionality at a component level by using the <code><component_name>_use_archive_compression</code> variables.	Optional	<code>true</code>
<code>use_db_compression</code>	<code>use_db_compression</code>	Controls at a global level whether the database-related backup files are compressed before being sent to the host to	Optional	<code>true</code>

RPM variable name	Container variable name	Description	Required or optional	Default
		run the backup operation. You can control this functionality at a component level by using the <code><component_name>_use_db_compression</code> variables.		
	<code>ca_tls_key_passphrase</code>	Passphrase used to decrypt the key provided in <code>ca_tls_key</code> .	Optional	
	<code>client_request_timeout</code>	Sets the HTTP timeout for end-user requests. The minimum value is <code>10</code> seconds.	Optional	<code>30</code>
	<code>container_compression</code>	Compression software to use for compressing container images.	Optional	<code>gzip</code>
	<code>container_keep_images</code>	Controls whether or not to keep container images when uninstalling Ansible Automation Platform. Set to <code>true</code> to keep container images when uninstalling Ansible Automation Platform.	Optional	<code>false</code>
	<code>container_pull_images</code>	Controls whether or not to pull newer container images during installation. Set to <code>false</code> to prevent pulling newer container images during installation.	Optional	<code>true</code>
	<code>images_tmp_dir</code>	The directory where the installation program temporarily stores	Optional	The system's

RPM variable name	Container variable name	Description	Required or optional	Default
		container images during installation.		temporary directory.
	<code>pcp_firewall_zone</code>	The firewall zone where Performance Co-Pilot related firewall rules are applied. This controls which networks can access Performance Co-Pilot based on the zone's trust level.	Optional	public
	<code>pcp_use_archive_compression</code>	Controls whether archive compression is enabled or disabled for Performance Co-Pilot. You can control this functionality globally by using <code>use_archive_compression</code> .	Optional	true
	<code>registry_auth</code>	Controls whether to use registry authentication. When set to <code>true</code> , <code>registry_username</code> and <code>registry_password</code> are required. Not applicable for disconnected (bundled) installations.	Optional	true
	<code>registry_ns_aap</code>	Ansible Automation Platform registry namespace.	Optional	ansible-automation-platform-26
	<code>registry_ns_rhel</code>	RHEL registry namespace.	Optional	rhel8

RPM variable name	Container variable name	Description	Required or optional	Default
	<code>setup_monitoring</code>	Set to <code>true</code> to enable Performance Co-Pilot for system performance monitoring and data collection on Ansible Automation Platform control plane nodes.	Optional	<code>false</code>

Image variables

Reference information for the inventory file variables used to configure images.

RPM variable name	Container variable name	Description	Required or optional	Default
<code>extra_images</code>		Additional container images to pull from the configured container registry during deployment.	Optional	<code>ansible-builder-rhel8</code>
	<code>controller_image</code>	Container image for automation controller.	Optional	<code>controller-rhel9:latest</code>
	<code>de_extra_images</code>	Additional decision environment container images to pull from the configured container registry during deployment.	Optional	<code>[]</code>
	<code>de_supported_image</code>	Supported decision environment container image.	Optional	<code>de-supported-</code>

RPM variable name	Container variable name	Description	Required or optional	Default
				rhel9:latest
	eda_image	Backend container image for Event-Driven Ansible.	Optional	eda-controller-rhel9:latest
	eda_web_image	Front-end container image for Event-Driven Ansible.	Optional	eda-controller-ui-rhel9:latest
	ee_extra_images	Additional execution environment container images to pull from the configured container registry during deployment.	Optional	[]
	ee_minimal_image	Minimal execution environment container image.	Optional	ee-minimal-rhel9:latest
	ee_supported_image	Supported execution environment container image.	Optional	ee-supported-rhel9:latest
	gateway_image	Container image for platform gateway.	Optional	gateway-rhel

RPM variable name	Container variable name	Description	Required or optional	Default
				9:latest
	gateway_proxy_image	Container image for platform gateway proxy.	Optional	gateway-proxy-rhel9:latest
	hub_image	Backend container image for automation hub.	Optional	hub-rhel9:latest
	hub_web_image	Front-end container image for automation hub.	Optional	hub-web-rhel9:latest
	pcp_image	Container image for Performance Co-Pilot.	Optional	pcp:latest
	postgresql_image	Container image for PostgreSQL.	Optional	postgresql-15:latest
	receptor_image	Container image for receptor.	Optional	receptor-rhel9:latest
	redis_image	Container image for Redis.	Optional	redis-6:latest

Platform gateway variables

Reference information for the inventory file variables used to configure platform gateway.

RPM variable name	Container variable name	Description	Required or optional	Default
<code>automationgateway_admin_email</code>	<code>gateway_admin_email</code>	Email address used by Django for the admin user for platform gateway.	Optional	<code>admin@example.com</code>
<code>automationgateway_admin_password</code>	<code>gateway_admin_password</code>	Platform gateway administrator password. Use of special characters for this variable is limited. The password can include any printable ASCII character except <code>/</code> , <code>"</code> , or <code>@</code> .	Required	
<code>automationgateway_admin_username</code>	<code>gateway_admin_user</code>	Username used to identify and create the administrator user in platform gateway. The installation program uses this account to register services with platform gateway. If you have deleted the default <code>`admin`</code> user, set this variable to an existing system administrator account to avoid installation or upgrade failures.	Optional	<code>admin</code>
<code>automationgateway_cache_cert</code>	<code>gateway_redis_tls_cert</code>	Path to the platform gateway Redis certificate file.	Optional	
<code>automationgateway_cache_key</code>	<code>gateway_redis_tls_key</code>	Path to the platform gateway Redis key file.	Optional	

RPM variable name	Container variable name	Description	Required or optional	Default
<code>automationgateway_cache_tls_files_remote</code>		Denote whether the cache client certificate files are local to the installation program (<code>false</code>) or on the remote component server (<code>true</code>).	Optional	The value defined in <code>automationgateway_cache_tls_files_remote</code> which defaults to <code>false</code> .
<code>automationgateway_client_regen_certificate</code>		Controls whether or not to regenerate platform gateway client certificates for the platform cache. Set to <code>true</code> to regenerate platform gateway client certificates.	Optional	<code>false</code>
<code>automationgateway_control_plane_port</code>	<code>gateway_control_plane_port</code>	Port number for the platform gateway control plane.	Optional	<code>50051</code>
<code>automationgateway_disable_hsts</code>	<code>gateway_nginx_disable_hsts</code>	Controls whether HTTP Strict Transport Security (HSTS) is enabled or disabled for platform gateway. Set this variable to <code>true</code> to disable HSTS.	Optional	<code>false</code>
<code>automationgateway_disable_https</code>	<code>gateway_nginx_disable_https</code>	Controls whether HTTPS is enabled or disabled for platform gateway. Set this	Optional	RPM = The value

RPM variable name	Container variable name	Description	Required or optional	Default
		variable to <code>true</code> to disable HTTPS.		defined in <code>disable_https</code> which defaults to <code>false</code> . Container = <code>false</code> .
<code>automationgateway_firewalld_zone</code>	<code>gateway_proxy_firewall_zone</code>	The firewall zone where platform gateway related firewall rules are applied. This controls which networks can access platform gateway based on the zone's trust level.	Optional	RPM = no default set. Container = 'public'.
<code>automationgateway_grpc_auth_service_timeout</code>	<code>gateway_grpc_auth_service_timeout</code>	Timeout duration (in seconds) for requests made to the gRPC service on platform gateway.	Optional	<code>30s</code>
<code>automationgateway_grpc_server_max_threads_per_process</code>	<code>gateway_grpc_server_max_threads_per_process</code>	Maximum number of threads that each gRPC server process can create to handle requests on platform gateway.	Optional	<code>10</code>
<code>automationgateway_grpc_server_processes</code>	<code>gateway_grpc_server_processes</code>	Number of processes for handling gRPC requests on platform gateway.	Optional	<code>5</code>

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

RPM variable name	Container variable name	Description	Required or optional	Default
				remote which defaults to false.
automationgateway_pg_cert_auth	gateway_pg_cert_auth	Controls whether client certificate authentication is enabled or disabled on the platform gateway PostgreSQL database. Set this variable to true to enable client certificate authentication.	Optional	false
automationgateway_pg_database	gateway_pg_database	Name of the PostgreSQL database used by platform gateway.	Optional	RPM = automationgateway. Container = gateway.
automationgateway_pg_host	gateway_pg_host	Hostname of the PostgreSQL database used by platform gateway.	Required	
automationgateway_pg_password	gateway_pg_password	Password for the platform gateway PostgreSQL database user. Use of special characters for this variable is limited. The ! , # , 0 and @	Optional	

RPM variable name	Container variable name	Description	Required or optional	Default
		characters are supported. Use of other special characters can cause the setup to fail.		
<code>automationgateway_pg_port</code>	<code>gateway_pg_port</code>	Port number for the PostgreSQL database used by platform gateway.	Optional	5432
<code>automationgateway_pg_sslmode</code>	<code>gateway_pg_sslmode</code>	Controls the SSL mode to use when platform gateway connects to the PostgreSQL database. Valid options include <code>verify-full</code> , <code>verify-ca</code> , <code>require</code> , <code>prefer</code> , <code>allow</code> , <code>disable</code> .	Optional	prefer
<code>automationgateway_pg_username</code>	<code>gateway_pg_username</code>	Username for the platform gateway PostgreSQL database user.	Optional	RPM = <code>automationgateway</code> . Container = <code>gateway</code>
<code>automationgateway_pgclient_sslcert</code>	<code>gateway_pg_tls_cert</code>	Path to the PostgreSQL SSL/TLS certificate file for platform gateway.	Required if using client certificate authentication.	
<code>automationgateway_pgclient_sslkey</code>	<code>gateway_pg_tls_key</code>	Path to the PostgreSQL SSL/TLS key file for platform gateway.	Required if	

RPM variable name	Container variable name	Description	Required or optional	Default
			using client certificate authentication.	
<code>automationgateway_pgclient_tls_files_remote</code>		Denote whether the PostgreSQL client certificates are local to the installation program (<code>false</code>) or on the remote component server (<code>true</code>).	Optional	The value defined in <code>automationgateway_tls_files_remote</code> which defaults to <code>false</code> .
<code>automationgateway_redis_host</code>	<code>gateway_redis_host</code>	Hostname of the Redis host used by platform gateway.	Optional	First node in the <code>[automationgateway]</code> inventory group.
<code>automationgateway_redis_password</code>	<code>gateway_redis_password</code>	Password for platform gateway Redis.	Optional	Randomly generated

RPM variable name	Container variable name	Description	Required or optional	Default
				string.
<code>automationgateway_redis_username</code>	<code>gateway_redis_username</code>	Username for platform gateway Redis.	Optional	<code>gateway</code>
<code>automationgateway_secret_key</code>	<code>gateway_secret_key</code>	Secret key value used by platform gateway to sign and encrypt data.	Optional	
<code>automationgateway_ssl_cert</code>	<code>gateway_tls_cert</code>	Path to the SSL/TLS certificate file for platform gateway.	Optional	
<code>automationgateway_ssl_key</code>	<code>gateway_tls_key</code>	Path to the SSL/TLS key file for platform gateway.	Optional	
<code>automationgateway_tls_files_remote</code>	<code>gateway_tls_remote</code>	Denote whether the platform gateway provided certificate files are local to the installation program (<code>false</code>) or on the remote component server (<code>true</code>).	Optional	<code>false</code>
<code>automationgateway_uwsgi_processes</code>	<code>gateway_uwsgi_processes</code>	The number of <code>uwsgi</code> processes for the platform gateway container. The value is calculated based on the number of available vCPUs (virtual CPUs).	Optional	The number of vCPUs multiplied by two, plus one.
<code>automationgateway_use_archive_compression</code>	<code>gateway_use_archive_compression</code>	Controls whether archive compression is enabled or disabled for platform gateway. You can control this functionality globally by	Optional	<code>true</code>

RPM variable name	Container variable name	Description	Required or optional	Default
		using <code>use_archive_compression</code> .		
<code>automationgateway_use_db_compression</code>	<code>gateway_use_db_compression</code>	Controls whether database compression is enabled or disabled for platform gateway. You can control this functionality globally by using <code>use_db_compression</code> .	Optional	<code>true</code>
<code>automationgateway_user_headers</code>	<code>gateway_nginx_user_headers</code>	List of additional NGINX headers to add to platform gateway's NGINX configuration.	Optional	<code>[]</code>
<code>automationgateway_verify_ssl</code>		Denotes whether or not to verify platform gateway's web certificates when making calls from platform gateway to itself during installation. Set to <code>false</code> to disable web certificate verification.	Optional	<code>true</code>
<code>automationgatewayproxy_disable_https</code>	<code>envoy_disable_https</code>	Controls whether or not HTTPS is disabled when accessing the platform UI. Set to <code>true</code> to disable HTTPS (HTTP is used instead).	Optional	RPM = The value defined in <code>disable_https</code> which defaults to <code>false</code> . Cont

RPM variable name	Container variable name	Description	Required or optional	Default
				ainer = false.
automationgatewayproxy_http_port	envoy_http_port	Port number on which the Envoy proxy listens for incoming HTTP connections.	Optional	80
automationgatewayproxy_https_port	envoy_https_port	Port number on which the Envoy proxy listens for incoming HTTPS connections.	Optional	443
nginx_tls_protocols	gateway_nginx_https_protocols	Protocols that platform gateway will support when handling HTTPS traffic.	Optional	[TLSv1.2, TLSv1.3]
redis_disable_tls	gateway_redis_disable_tls	Controls whether TLS is enabled or disabled for platform gateway Redis. Set this variable to true to disable TLS.	Optional	false
redis_port	gateway_redis_port	Port number for the Redis host for platform gateway.	Optional	6379
	gateway_extra_settings	Defines additional settings for use by platform gateway during installation. For example: <pre>gateway_extra_settings=[{"setting": "OAUTH2_PROVIDER['ACCESS_TOKEN_EXPIRE_SECONDS']", "value": 600}]</pre>	Optional	[]

RPM variable name	Container variable name	Description	Required or optional	Default
	<code>gateway_nginx_client_max_body_size</code>	Maximum allowed size for data sent to platform gateway through NGINX.	Optional	5m
	<code>gateway_nginx_hsts_max_age</code>	Maximum duration (in seconds) that HTTP Strict Transport Security (HSTS) is enforced for platform gateway.	Optional	63072000
	<code>gateway_uwsgi_listen_queue_size</code>	Number of requests <code>uwsgi</code> will allow in the queue on platform gateway until <code>uwsgi_processes</code> can serve them.	Optional	4096

Receptor variables

Reference information for the inventory file variables used to configure Receptor.

RPM variable name	Container variable name	Description	Required or optional	Default
<code>receptor_datadir</code>		The directory where receptor stores its runtime data and local artifacts. The target directory must be accessible to awx users. If the target directory is a temporary file system tmpfs , ensure it is remounted correctly after a reboot. Failure to do so results in the receptor no longer having a working directory.	Optional	<code>/tmp/receptor</code>

RPM variable name	Container variable name	Description	Required or optional	Default
<code>receptor_listen_port</code>	<code>receptor_port</code>	Port number that receptor listens on for incoming connections from other receptor nodes.	Optional	2719
<code>receptor_listen_protocol</code>	<code>receptor_protocol</code>	Protocol that receptor will support when handling traffic.	Optional	tcp
<code>receptor_log_level</code>	<code>receptor_log_level</code>	Controls the verbosity of logging for receptor. Valid options include: <code>error</code> , <code>warning</code> , <code>info</code> , or <code>debug</code> .	Optional	info
<code>receptor_tls</code>		Controls whether TLS is enabled or disabled for receptor. Set this variable to <code>false</code> to disable TLS.	Optional	true
See <code>node_type</code> for the RPM equivalent variable.	<code>receptor_type</code>	For the <code>[automationcontroller]</code> group the two options are: <code>receptor_type=control</code> - The node only runs project and inventory updates, but not regular jobs. <code>receptor_type=hybrid</code> - The node runs everything. For the <code>[execution_nodes]</code> group the two options are:	Optional	For the <code>[automationcontroller]</code> group: <code>hybrid</code> . For the <code>[execution_nodes]</code> group: <code>exec</code>

RPM variable name	Container variable name	Description	Required or optional	Default
		<code>receptor_type=hop</code> - The node forwards jobs to an execution node. <code>receptor_type=execution</code> - The node can run jobs.		<code>hop</code> .
See <code>peers</code> for the RPM equivalent variable	<code>receptor_peers</code>	Used to indicate which nodes a specific host connects to. Wherever this variable is defined, an outbound connection to the specific host is established. The value must be a comma-separated list of hostnames. Do not use inventory group names. This is resolved into a set of hosts that is used to construct the <code>receptor.conf</code> file. For more information, see "Add execution nodes" on page 134.	Optional	<code>[]</code>
	<code>receptor_disable_signing</code>	Controls whether signing of communications between receptor nodes is enabled or disabled. Set this variable to <code>true</code> to disable communication signing.	Optional	<code>false</code>
	<code>receptor_disable_tls</code>	Controls whether TLS is enabled or disabled for receptor. Set this variable to <code>true</code> to disable TLS.	Optional	<code>false</code>

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

RPM variable name	Container variable name	Description	Required or optional	Default
		program (<code>false</code>) or on the remote component server (<code>true</code>).		
	<code>receptor_use_archive_compression</code>	Controls whether archive compression is enabled or disabled for receptor. You can control this functionality globally by using <code>use_archive_compression</code> .	Optional	<code>true</code>

Redis variables

Reference information for the inventory file variables used to configure Redis.

RPM variable name	Container variable name	Description	Required or optional	Default
<code>redis_cluster_ip</code>	<code>redis_cluster_ip</code>	The IPv4 address used by the Redis cluster to identify each host in the cluster. When defining hosts in the <code>[redis]</code> group, use this variable to identify the IPv4 address if the default is not what you want. Specific to container: Redis clusters cannot use hostnames or IPv6 addresses.	Optional	RPM = Discovered IPv4 address from Ansible facts. If IPv4 address is not available, IPv6

RPM variable name	Container variable name	Description	Required or optional	Default
				address is used. Container = Discover IPv4 address from Ansible facts.
<code>redis_disable_mtls</code>		Controls whether mTLS is enabled or disabled for Redis. Set this variable to <code>true</code> to disable mTLS.	Optional	<code>false</code>
<code>redis_firewalld_zone</code>	<code>redis_firewall_zone</code>	The firewall zone where Redis related firewall rules are applied. This controls which networks can access Redis based on the zone's trust level.	Optional	RPM = no default set. Container = <code>public</code> .
<code>redis_hostname</code>		Hostname used by the Redis cluster when identifying and routing the host. By default <code>routable_hostname</code> is used.	Optional	The value defined in <code>routable_hostname</code>
<code>redis_mode</code>	<code>redis_mode</code>	The Redis mode to use for your Ansible	Optional	<code>cluster</code>

RPM variable name	Container variable name	Description	Required or optional	Default
		Automation Platform installation. Valid options include: <code>standalone</code> and <code>cluster</code> .		
<code>redis_server_regen_cert</code>		Denotes whether or not to regenerate the Ansible Automation Platform managed TLS key pair for Redis.	Optional	<code>false</code>
<code>redis_tls_cert</code>	<code>redis_tls_cert</code>	Path to the Redis server TLS certificate.	Optional	
<code>redis_tls_files_remote</code>	<code>redis_tls_remote</code>	Denote whether the Redis provided certificate files are local to the installation program (<code>false</code>) or on the remote component server (<code>true</code>).	Optional	<code>false</code>
<code>redis_tls_key</code>	<code>redis_tls_key</code>	Path to the Redis server TLS certificate key.	Optional	
	<code>redis_use_archive_compression</code>	Controls whether archive compression is enabled or disabled for Redis. You can control this functionality globally by using <code>use_archive_compression</code> .	Optional	<code>true</code>

Install on OpenShift Container Platform

As a system administrator, you can use Ansible Automation Platform Operator to deploy new Ansible Automation Platform instances in your OpenShift environment.

Plan your installation of Ansible Automation Platform on Red Hat OpenShift Container Platform

Red Hat Ansible Automation Platform is supported on both Red Hat Enterprise Linux and Red Hat OpenShift.

OpenShift operators help install and automate day-2 operations of complex, distributed software on Red Hat OpenShift Container Platform. The Ansible Automation Platform Operator enables you to deploy and manage Ansible Automation Platform components on Red Hat OpenShift Container Platform.

You can use this section to help plan your Red Hat Ansible Automation Platform installation on your Red Hat OpenShift Container Platform environment. Before installing, review the supported installation scenarios to determine which meets your requirements.

About Ansible Automation Platform Operator

The Ansible Automation Platform Operator provides cloud-native, push-button deployment of new Ansible Automation Platform instances in your OpenShift environment.

The Ansible Automation Platform Operator includes resource types to deploy and manage instances of automation controller and private automation hub.

It also includes automation controller job resources for defining and launching jobs inside your automation controller deployments.

Deploying Ansible Automation Platform instances with a Kubernetes native operator offers several advantages over launching instances from a playbook deployed on Red Hat OpenShift Container Platform, including upgrades and full lifecycle support for your Red Hat Ansible Automation Platform deployments.

You can install the Ansible Automation Platform Operator from the Red Hat Operators catalog in OperatorHub.

Related information
[Operator topologies](#)

OpenShift Container Platform version compatibility

For supported OpenShift Container Platform versions, see the Red Hat Ansible Automation Platform Life Cycle page.

Related information

[Red Hat Ansible Automation Platform Life Cycle](#)

Supported installation scenarios for Red Hat OpenShift Container Platform

You can use the OperatorHub on the Red Hat OpenShift Container Platform web console to install Ansible Automation Platform Operator.

Alternatively, you can install Ansible Automation Platform Operator from the OpenShift Container Platform command-line interface (CLI), `oc`.

After you have installed Ansible Automation Platform Operator you must create an **Ansible Automation Platform** custom resource (CR). This enables you to manage Ansible Automation Platform components from a single unified interface known as the platform gateway. In version 2.6, you must create an Ansible Automation Platform CR, even if you have an existing automation controller, automation hub, or Event-Driven Ansible, components.

If existing components have already been deployed, you must specify these components on the Ansible Automation Platform CR. You must create the custom resource in the same namespace as the existing components.

Supported scenarios	Supported scenarios with existing components
• Ansible Automation Platform CR for blank slate install with automation controller, automation hub, and Event-Driven Ansible enabled • Ansible Automation Platform CR with just automation controller enabled • Ansible Automation Platform CR with just automation controller, automation hub enabled	• Ansible Automation Platform CR created in the same namespace as an existing automation controller CR with the automation controller name specified on the Ansible Automation Platform CR spec • Same with automation controller and automation hub

Supported scenarios	Supported scenarios with existing components
Ansible Automation Platform CR with just automation controller, Event-Driven Ansible enabled	- Same with automation controller, automation hub, and Event-Driven Ansible - Same with automation controller and Event-Driven Ansible

Related information

[Access Ansible Automation Platform through the CLI](#)

Ansible Automation Platform Operator CSRF management

The Ansible Automation Platform Operator on OpenShift Container Platform creates OpenShift Routes and configures your Cross-site request forgery (CSRF) settings automatically.

When using external ingress, you must configure your CSRF on the ingress.

IMPORTANT:

CSRF settings in the automation controller user interface have no impact on CSRF settings for the platform gateway. The platform gateway manages its own CSRF configuration.

The following table helps to clarify which settings are applicable for which component.

UI setting	Applicable for
Subscription	automation controller
platform gateway	platform gateway
User Preferences	User interface
System	Automation controller
Job	Automation controller
Logging	Automation controller
Troubleshooting	Automation controller

Related information

[Configure your CSRF settings](#)

Additional resources

To learn more about OpenShift Container Platform OperatorHub you can review OpenShift Container Platform documentation.

Related information

[Understanding OperatorHub](#)

Install the Ansible Automation Platform Operator through OperatorHub

Use this procedure to guide you through deploying the Red Hat Ansible Automation Platform Operator through the **Operators** section on Red Hat OpenShift Container Platform, selecting the appropriate update channel and installation mode, and then verifying the successful deployment.

Install the Red Hat Ansible Automation Platform Operator on Red Hat OpenShift Container Platform

When installing your Ansible Automation Platform Operator you have a choice of a namespace-scoped operator or a cluster-scoped operator. This depends on the update channel you choose, `stable-2.x` or `cluster-scoped-2.x`.

Before you begin

- You have installed the Red Hat Ansible Automation Platform catalog in OperatorHub.
- You have created a `StorageClass` object for your platform and a persistent volume claim (PVC) with `ReadWriteMany` access mode. See [Dynamic provisioning](#) for details. See [Configure static storage for Ansible Automation Platform](#) for manual setup.
- To run Red Hat OpenShift Container Platform clusters on Amazon Web Services (AWS) with `ReadWriteMany` access mode, you must add NFS or other storage.
 - For information about the AWS Elastic Block Store (EBS) or to use the `aws-ebs` storage class, see [Persistent storage using AWS Elastic Block Store](#).
 - To use multi-attach `ReadWriteMany` access mode for AWS EBS, see [Attaching a volume to multiple instances with Amazon EBS Multi-Attach](#).

A namespace-scoped operator is confined to one namespace, offering tighter security. A cluster-scoped operator spans multiple namespaces, which grants broader permissions.

If you are managing multiple Ansible Automation Platform instances with the same Ansible Automation Platform Operator version, use the cluster-scoped operator, which uses a single operator to manage all Ansible Automation Platform custom resources in your cluster.

If you need multiple operator versions in the same cluster, you must use the namespace-scoped operator. The operator and the deployment share the same namespace. This can also be helpful when debugging because the operator logs pertain to custom resources in that namespace only.

NOTE:

For information about the Ansible Automation Platform Operator system requirements and infrastructure topology see [Operator topologies](#) in *Tested deployment models*.

For help with installing a namespace or cluster-scoped operator see the following procedure.

IMPORTANT:

You cannot deploy Ansible Automation Platform in the default namespace on your OpenShift Cluster. The `aap` namespace is recommended. You can use a custom namespace, but it should run only Ansible Automation Platform.

Procedure

1. Log in to Red Hat OpenShift Container Platform.
2. Navigate to **Operators > OperatorHub**.
3. Search for Ansible Automation Platform and click **Install**.
4. Select an **Update Channel**:
 - **stable-2.x**: installs a namespace-scoped operator, which limits deployments of automation hub and automation controller instances to the namespace the operator is installed in, this is suitable for most cases. The stable-2.x channel does not require administrator privileges and utilizes fewer resources because it only monitors a single namespace.
 - **stable-2.x-cluster-scoped**: installs the Ansible Automation Platform Operator in a single namespace that manages Ansible Automation Platform custom resources and deployments in all namespaces. The Ansible Automation Platform Operator requires administrator privileges for all namespaces in the cluster.
5. Select **Installation Mode**, **Installed Namespace**, and **Approval Strategy**.
6. Click **Install**.

Result

The installation process begins. When installation finishes, a modal appears notifying you that the Ansible Automation Platform Operator is installed in the specified namespace.

- Click **View Operator** to view your newly installed Ansible Automation Platform Operator and verify the following operator custom resources are present:

Automation controller	Automation hub	Event-Driven Ansible (EDA)	Red Hat Ansible Lightspeed
• Automation Controller • Automation Controller Backup • Automation Controller Restore • Automation Controller Mesh Ingress	• Automation Hub • Automation Hub Backup • Automation Hub Restore	• EDA • EDA Backup • EDA Restore	• Ansible Lightspeed

- Verify that the Ansible Automation Platform operator displays a **Succeeded** status.

Install Ansible Automation Platform Operator from the CLI

Use these instructions to install the Ansible Automation Platform Operator on Red Hat OpenShift Container Platform from the OpenShift Container Platform command-line interface (CLI) using the `oc` command.

Install the Ansible Automation Platform Operator in a namespace

Use this procedure to subscribe a namespace to an operator.

Before you begin

- Access to Red Hat OpenShift Container Platform using an account with operator installation permissions.

- The OpenShift Container Platform CLI `oc` command is installed on your local system. Refer to [Installing the OpenShift CLI](#) in the Red Hat OpenShift Container Platform product documentation for further information.

IMPORTANT:

You cannot deploy Ansible Automation Platform in the default namespace on your OpenShift Cluster. The 'ansible-automation-platform' namespace is recommended. You can use a custom namespace, but it should run only Ansible Automation Platform.

Procedure

1. Create a project for the operator.

```
oc new-project ansible-automation-platform
```

2. Create a file called *sub.yaml*.
3. Add the following YAML code to the *sub.yaml* file.

```

---
apiVersion: operators.coreos.com/v1
kind: OperatorGroup
metadata:
  name: ansible-automation-platform-operator
  namespace: ansible-automation-platform
spec:
  targetNamespaces:
    - ansible-automation-platform
---
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: ansible-automation-platform
  namespace: ansible-automation-platform
spec:
  channel: 'stable-2.6'
  installPlanApproval: Automatic
  name: ansible-automation-platform-operator
  source: redhat-operators
  sourceNamespace: openshift-marketplace
---

```

This file creates a `Subscription` object called `ansible-automation-platform` that subscribes the `ansible-automation-platform` namespace to the `ansible-automation-platform-operator` operator.

4. Run the **oc apply** command to create the objects specified in the `sub.yaml` file:

```
oc apply -f sub.yaml
```

5. Verify the CSV PHASE reports "Succeeded" before proceeding using the **oc get csv -n ansible-automation-platform** command:

```
oc get csv -n ansible-automation-platform
```

NAME	REPLACES	DISPLAY PHASE	VERSION
aap-operator.v2.6.0-0.1728520175		Ansible Automation Platform	
2.6.0+0.1728520175	aap-operator.v2.6.0-0.1727875185	Succeeded	

6. Create an `AnsibleAutomationPlatform` object called `example` in the `ansible-automation-platform` namespace.

To change the Ansible Automation Platform and its components from `example`, edit the `name` field in the `metadata:` section and replace `example` with the name you want to use:

```

oc apply -f - <<EOF
apiVersion: aap.ansible.com/v1alpha1
kind: AnsibleAutomationPlatform
metadata:
  name: example
  namespace: ansible-automation-platform
spec:
  # Platform
  image_pull_policy: IfNotPresent
  # Components
  controller:
    disabled: false
  eda:
    disabled: false
  hub:
    disabled: false
    ## Modify to contain your RWM storage class name
    storage_type: file
    file_storage_storage_class: <your-read-write-many-storage-class>
    file_storage_size: 10Gi

    ## uncomment if using S3 storage for Content pod
    # storage_type: S3
    # object_storage_s3_secret: example-galaxy-object-storage

    ## uncomment if using Azure storage for Content pod
    # storage_type: azure
    # object_storage_azure_secret: azure-secret-name
  lightspeed:
    disabled: true
EOF

```

Related information

[Installing from OperatorHub using the CLI](#)

Deploy Ansible Automation Platform

As a namespace administrator, you can use Ansible Automation Platform gateway to manage Ansible Automation Platform components in your OpenShift environment.

The Ansible Automation Platform gateway uses the Ansible Automation Platform custom resource to manage and integrate the following Ansible Automation Platform components into a unified user interface:

- Automation controller
- Automation hub
- Event-Driven Ansible
- Red Hat Ansible Lightspeed (This feature is disabled by default, you must opt in to use it.)

Before you can deploy the platform gateway you must have Ansible Automation Platform Operator installed in a namespace. If you have not installed Ansible Automation Platform Operator see **Installing the Red Hat Ansible Automation Platform Operator on Red Hat OpenShift Container Platform**.

If you have the Ansible Automation Platform Operator and some or all of the Ansible Automation Platform components installed see **Deploying the platform gateway with existing Ansible Automation Platform components** for how to proceed.

Related information

[Install the Red Hat Ansible Automation Platform Operator on Red Hat OpenShift Container Platform](#)

[Add components to an existing deployment](#)

Deploy a new Ansible Automation Platform instance with components

After installing the Ansible Automation Platform Operator in your namespace you can set up your **Ansible Automation Platform** instance. Then link all the platform components to a single user interface.

Procedure

1. Log in to Red Hat OpenShift Container Platform.
2. Navigate to **Operators > Installed Operators**.
3. Select your Ansible Automation Platform Operator deployment.
4. Select the **Details** tab.
5. On the **Ansible Automation Platform** tile click **Create instance**.

6. From the **Create Ansible Automation Platform** page enter a name for your instance in the **Name** field.
7. Click **YAML view** and replace the `spec` section with the following:

```
spec:
  database:
    resource_requirements:
      requests:
        cpu: 200m
        memory: 512Mi
    storage_requirements:
      requests:
        storage: 100Gi

  controller:
    disabled: false

  eda:
    disabled: false

  hub:
    disabled: false
    storage_type: file
    file_storage_storage_class: <read-write-many-storage-class>
    file_storage_size: 100Gi
```

8. You must specify your desired value for the `<read-write-many-storage-class>` placeholder.
9. Click **Create**.

Result

Verify instance deployment (UI):

1. Navigate to **Operators > Installed Operators**.
2. Select your Ansible Automation Platform Operator deployment.
3. Select the **All instances** tab.
4. Verify that the **Ansible Automation Platform** instance, **Automation Controller**, **Event-Driven Ansible**, and **Automation Hub** instances are listed.

Verify pod status (UI):

1. Navigate to **Workloads > Pods**.
2. Switch to the project (namespace) where you deployed the instance.
3. Verify that all related pods display a **Running** or **Completed** status.

Verify Platform Route (CLI):

Run the following command to confirm the URL for accessing the platform gateway:

```
oc get route
```

Add components to an existing deployment

You can link any components of the Ansible Automation Platform, that you have already installed to a new **Ansible Automation Platform** instance.

The following procedure simulates a scenario where you have automation controller as an existing component and want to add automation hub and Event-Driven Ansible.

Procedure

1. Log in to Red Hat OpenShift Container Platform.
2. Navigate to **Operators > Installed Operators**.
3. Select your Ansible Automation Platform Operator deployment.
4. Click **Subscriptions** and edit your **Update channel** to the stable channel for your Ansible Automation Platform version (for example, `stable-2.6`).
5. Click **Details** and on the **Ansible Automation Platform** tile click **Create instance**.
6. From the **Create Ansible Automation Platform** page enter a name for your instance in the **Name** field.
 - When deploying an Ansible Automation Platform instance, ensure that `auto_update` is set to the default value of `false` on your existing automation controller instance in order for the integration to work.
7. Click **YAML view** and copy in the following:

```

apiVersion: aap.ansible.com/v1alpha1
kind: AnsibleAutomationPlatform
metadata:
  name: example-aap
  namespace: aap
spec:
  database:
    resource_requirements:
      requests:
        cpu: 200m
        memory: 512Mi
    storage_requirements:
      requests:
        storage: 100Gi

  # Platform
  image_pull_policy: IfNotPresent

  # Components
  controller:
    disabled: false
    name: existing-controller-name
  eda:
    disabled: false
  hub:
    disabled: false
    ## uncomment if using file storage for Content pod
    storage_type: file
    file_storage_storage_class: <your-read-write-many-storage-class>
    file_storage_size: 10Gi

    ## uncomment if using S3 storage for Content pod
    # storage_type: S3
    # object_storage_s3_secret: example-galaxy-object-storage

    ## uncomment if using Azure storage

```

- a. For new components, if you do not specify a name, a default name is generated.
8. Click **Create**.
9. To access your new instance, see [Accessing the platform gateway](#).

NOTE:

If you have an existing controller with a managed Postgres pod, after creating the **Ansible Automation Platform** resource your automation controller instance will continue to use that original Postgres pod. If you were to do a fresh install you would have a single Postgres managed pod for all instances.

Access Ansible Automation Platform through the UI

Use the **Ansible Automation Platform** instance as your default. This instance links the automation controller, automation hub, and Event-Driven Ansible deployments to a single interface.

Procedure

1. Log in to Red Hat OpenShift Container Platform.
2. Navigate to **Networking > Routes**
3. Click the link under **Location** for **Ansible Automation Platform**.
4. This redirects you to the Ansible Automation Platform login page. Enter "admin" as your username in the **Username** field.
5. For the password you must:
 - a. Go to **Workloads > Secrets**.
 - b. Click **<your instance name>-admin-password** and copy the password.
 - c. Paste the password into the **Password** field.
6. Click **Login**.
7. Apply your subscription:
 - a. Click **Subscription manifest** or **Username/password**.
 - b. Upload your manifest or enter your username and password.
 - c. Select your subscription from the **Subscription** list.
 - d. Click **Next**. This redirects you to the **Analytics** page.
8. Click **Next**.
9. Select the **I agree to the terms of the license agreement** checkbox.
10. Click **Next**.

Result

You now have access to the platform gateway user interface.

If you cannot access the Ansible Automation Platform see [Frequently asked questions on platform gateway](#) for help with troubleshooting and debugging.

Access Ansible Automation Platform through the CLI

You can use the OpenShift Container Platform CLI to fetch the web address and the password of the Automation controller that you created. To login to the platform gateway, you need the web address and the password.

Fetch the platform gateway web address

A Red Hat OpenShift Container Platform route exposes a service at a host name, so that external clients can reach it by name. When you created the platform gateway instance, a route was created for it. The route inherits the name that you assigned to the platform gateway object in the YAML file.

- Use the following command to fetch the routes:

```
oc get routes -n <platform_namespace>
```

Result

You can see in the following example, the `example` platform gateway is running in the `ansible-automation-platform` namespace.

```
$ oc get routes -n ansible-automation-platform
```

NAME	HOST/PORT	WILDCARD	PATH	SERVICES
	PORT	TERMINATION		
example service	example-ansible-automation-platform.apps-crc.testing http	edge/Redirect	None	example-

The address for the platform gateway instance is `example-ansible-automation-platform.apps-crc.testing`.

Fetch the platform gateway password

The YAML block for the platform gateway instance in the `AnsibleAutomationPlatform` object assigns values to the `name` and `admin_user` keys.

Procedure

1. Use these values in the following command to fetch the password for the platform gateway instance.

```
oc get secret/<your instance name>-<admin_user>-password -o yaml
```

2. The default value for `admin_user` is `admin`. Modify the command if you changed the admin username in the `AnsibleAutomationPlatform` object.

The following example retrieves the password for a platform gateway object called `example`:

```
oc get secret/example-admin-password -o yaml
```

The base64 encoded password for the platform gateway instance is listed in the `metadata` field in the output:

```
$ oc get secret/example-admin-password -o yaml

apiVersion: v1
data:
  password: ODzLODzLODzLODzLODzLODzLODzLODzLODzLODzL
kind: Secret
metadata:
  labels:
    app.kubernetes.io/component: aap
    app.kubernetes.io/name: example
    app.kubernetes.io/operator-version: ""
    app.kubernetes.io/part-of: example
  name: example-admin-password
  namespace: ansible-automation-platform
```

Decode the platform gateway password

After you have fetched your gateway password, you must decode it from base64.

- Run the following command to decode your password from base64:

```
oc get secret/example-admin-password -o jsonpath={.data.password} | base64 --decode
```

Related information

[OpenShift Container Platform product documentation](#)

Manage Ansible Automation Platform subscriptions, updates, and support

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

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

Related information

[Ansible Source Code](#)

Trial and evaluation

You need a subscription to run Ansible Automation Platform. You can start by signing up for a free trial subscription.

- Trial subscriptions for Ansible Automation Platform are available at the [Red Hat product trial center](#).
- Support is not included in a trial subscription or during an evaluation of the Ansible Automation Platform.

Node counting in subscriptions

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

For more information about managed node requirements for subscriptions, see [How are "managed nodes" defined as part of the Red Hat Ansible Automation Platform offering](#).

NOTE:

Ansible does not recycle node counts or reset automated hosts.

Subscription Types

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

- **Standard:**

- Manage any size environment
- Enterprise 8x5 support and SLA
- Maintenance and upgrades included
- Review the SLA at [Product Support Terms of Service](#)
- Review the [Red Hat Support Severity Level Definitions](#)

- **Premium:**

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

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

For more information, contact Ansible through the [Red Hat Customer Portal](#) or at the [Ansible site](#).

Obtain a manifest file

You can obtain a subscription manifest in the **Subscription Allocations** section of Red Hat Subscription Management.

After you obtain a subscription allocation, you can download its manifest file and upload it to activate Ansible Automation Platform.

To begin, log in to the **Red Hat Customer Portal** by using your administrator user account and follow the procedures listed.

Related information

[Red Hat Customer Portal](#)
[Subscription Allocations](#)

Create a subscription allocation

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

Procedure

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

Add subscriptions to a subscription allocation

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

Procedure

1. From the [Subscription Allocations](#) page, click the name of the **Subscription Allocation** to which you want to add a subscription.
2. Click the **Subscriptions** tab.
3. Click **Add Subscriptions**.
4. Enter the number of Ansible Automation Platform Entitlements you plan to add.
5. Click **Submit**.

Download a manifest file

After you create an allocation with the appropriate subscriptions on it, you can download the manifest file from Red Hat Subscription Management.

Procedure

1. From the [Subscription Allocations](#) page, click the name of the **Subscription Allocation** to which you want to generate a manifest.

2. Click the **Subscriptions** tab.
3. Click **Export Manifest** to download the manifest file.

Result

This downloads a file `manifest_<allocation name>_<date>.zip` to your default downloads folder.

Activate your subscription

Red Hat Ansible Automation Platform uses available subscriptions or a subscription manifest to allow the use of Ansible Automation Platform.

To obtain a subscription, you can do either of the following:

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

Related information

[Activate with credentials](#)

[Activate with a manifest file](#)

Activate with credentials

Activate your Ansible Automation Platform subscription at first launch using Red Hat service account credentials or a personal login. This retrieves and imports the license, granting access to Red Hat content and entitlement services.

NOTE:

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

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

Procedure

1. Log in to Red Hat Ansible Automation Platform.
2. Select the **Service Account** tab in the subscription wizard.
3. Enter your **Client ID** and **Client secret**.
4. Select your subscription from the **Subscription** list.

NOTE:

You can also enter your Satellite username and password in the **Satellite** tab if your cluster nodes are registered to Satellite through Subscription Manager.

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

Result

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

Hosts automated

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

Hosts imported

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

Hosts remaining

Total host count minus hosts automated

Activate with a manifest file

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

Before you begin

You must have a Red Hat subscription manifest file exported from the Red Hat Customer Portal. For more information, see [Obtaining a manifest file](#).

NOTE:

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

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

Procedure

1. Log in to Red Hat Ansible Automation Platform.
 - a. If you are not immediately taken to the subscription wizard, go to **Settings > Subscription**.
2. Select the **Subscription manifest** tab.
3. Click **Browse** and select your manifest file.
4. Review the End User License Agreement and select **I agree to the End User License Agreement**.
5. Click **Finish**.

NOTE:

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

Result

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

Hosts automated

Host count automated by the job, which uses the subscription count

Hosts imported

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

Hosts remaining

Total host count minus hosts automated

Next steps

- You can return to the subscription wizard by selecting **Settings > Subscription** from the navigation panel and clicking **Edit subscription**.

Customize your Red Hat Ansible Automation Platform Operator

Customize your deployment by setting configuration parameters in the parent **AnsibleAutomationPlatform** custom resource (CR). The operator automatically applies these settings to all nested platform components.

View configuration parameters in the UI

You can discover the configuration parameters for your Ansible Automation Platform Operator by viewing its Custom Resource (CR). The parameters are listed in the YAML schema.

Procedure

1. Log in to Red Hat OpenShift Container Platform.
2. Navigate to **Operators > Installed Operators**.
3. Select your Ansible Automation Platform Operator deployment.
4. Go to the Ansible Automation Platform tab and click the name of your CR.
5. Switch to the **YAML view** tab to view and edit the configuration. The available parameters are listed in the YAML schema.

NOTE:

If you cannot see the **Schema** panel, you might have closed or minimized the side bar. Click **view sidebar** to reopen it.

View configuration parameters by using the CLI

The Ansible Automation Platform Operator manages multiple custom resources (CRs), each with its own configuration parameters. Use the `oc explain` command to discover all available configuration options for the `AnsibleAutomationPlatform` CR and its nested components.

Procedure

1. To see all available configuration parameters for a top-level CR, run:

```
oc explain ansibleautomationplatform.spec
```

2. To view component-specific configuration options nested under the Ansible Automation Platform CR, query them through the component section:

```
oc explain  
ansibleautomationplatform.spec.controller.postgres_configuration_secret  
  
oc explain  
ansibleautomationplatform.spec.controller.route_tls_termination_mechanism  
  
oc explain ansibleautomationplatform.spec.hub.storage_type  
  
oc explain ansibleautomationplatform.spec.eda.automation_server_url
```

3. To explore all nested fields for a specific component, use the `--recursive` flag:

```
oc explain ansibleautomationplatform.spec.controller --recursive  
oc explain ansibleautomationplatform.spec.hub --recursive  
oc explain ansibleautomationplatform.spec.eda --recursive
```

NOTE: You can also query individual component CRs directly if needed:

```
oc explain automationcontroller.spec  
oc explain automationhub.spec  
oc explain eda.spec
```

However, when configuring components through the Ansible Automation Platform CR (recommended approach), use the nested paths shown above.

Configure component parameters

To define a parameter, such as the `resource_requirements` for automation controller, you add the configuration to the parent Ansible Automation Platform CR YAML. This ensures that the Ansible Automation Platform CR is the single source of truth for your deployment.

NOTE:

The `image` and `image_version`, as well as the `{component}_image` and `{component}_image_version` parameters are intended for development or hotfix purposes only.

Do not use these in production environments. These settings bypass standard version management and can lead to configuration drift, inconsistent deployments, and difficulty troubleshooting issues.

Procedure

1. Log in to OpenShift Container Platform.
2. Navigate to **Operators > Installed Operators**.
3. Select your Ansible Automation Platform Operator deployment.
4. Go to the Ansible Automation Platform tab and click the name of your CR.
5. In the **YAML view** tab, locate the **spec** section.
6. Add the `automationcontroller` parameter with the nested `resource_requirements` parameter and its value:

```
spec:
  database:
    resource_requirements:
      requests:
        cpu: 200m
        memory: 512Mi
    storage_requirements:
      requests:
        storage: 100Gi
```

7. Click **Save** to apply the changes. The operator automatically applies this configuration to the automation controller component.

Customize your resource requirements

Customize resource requirements for your Ansible Automation Platform components to optimize performance and resource allocation in your specific environment.

The following section provides a complete code block with the default resource requirements for each component. The main reasons for customizing your resource requirements include:

- Performance Tuning: Increase resource limits for components that perform heavy workloads.
- To comply with a `ResourceQuota` enforced by the cluster admin.
- Resource Constrained Environments: Decrease resource requests to conserve cluster resources in development or test environments.
- Environment Specifics: Align the resource allocation with the capacity of your OpenShift or Kubernetes cluster nodes.

You can use this reference as a starting point. Copy the full code block for your Ansible Automation Platform instance and modify the values for the components you want to change. This method helps ensure all default settings are applied correctly, reducing the risk of deployment errors.

NOTE:

When *adding* parameters, you can add it to the Ansible Automation Platform custom resource (CR) only and those parameters will work their way down to the nested CRs.

When *removing* parameters, you have to remove them both from the Ansible Automation Platform CR *and* the nested CR, for example, the **Automation Controller** CR.

```
# Example of defining custom resource requirements for all components

# This can be useful for clusters with a ResourceQuota in the AAP
namespaceapiVersion: aap.ansible.com/v1alpha1

kind: AnsibleAutomationPlatform

metadata:
  name: aap

spec:

  # Platform
  api:
    replicas: 1
    resource_requirements:
      requests:
        cpu: 100m
        memory: 256Mi
      limits:
        cpu: 500m
        memory: 1000Mi
  redis:
    replicas: 1
```

```

resource_requirements:
  requests:
    cpu: 100m
    memory: 256Mi
  limits:
    cpu: 500m
    memory: 500Mi
database:
  resource_requirements:
    requests:
      cpu: 100m
      memory: 256Mi
    limits:
      cpu: 500m
      memory: 800Mi

# Components
controller:
  disabled: false
  uwsgi_processes: 2
  task_resource_requirements:
    requests:
      cpu: 100m
      memory: 150Mi
    limits:
      cpu: 1000m
      memory: 1200Mi
  web_resource_requirements:
    requests:
      cpu: 100m
      memory: 200Mi
    limits:
      cpu: 200m
      memory: 1600Mi
  ee_resource_requirements:
    requests:

```



```

    cpu: 100m
    memory: 64Mi
  limits:
    cpu: 1000m
    memory: 500Mi
  redis_resource_requirements:
    requests:
      cpu: 50m
      memory: 64Mi
    limits:
      cpu: 100m
      memory: 200Mi
  rsyslog_resource_requirements:
    requests:
      cpu: 100m
      memory: 128Mi
    limits:
      cpu: 500m
      memory: 250Mi
  init_container_resource_requirements:
    requests:
      cpu: 100m
      memory: 128Mi
    limits:
      cpu: 500m
      memory: 200Mi
eda:
  disabled: false
  api:
    replicas: 1
    resource_requirements:
      requests:
        cpu: 50m
        memory: 350Mi
      limits:
        cpu: 500m

```

```
        memory: 400Mi
ui:
  replicas: 1
  resource_requirements:
    requests:
      cpu: 25m
      memory: 64Mi
    limits:
      cpu: 500m
      memory: 150Mi
scheduler:
  replicas: 1
  resource_requirements:
    requests:
      cpu: 50m
      memory: 200Mi
    limits:
      cpu: 500m
      memory: 250Mi
worker:
  replicas: 2
  resource_requirements:
    requests:
      cpu: 25m
      memory: 200Mi
    limits:
      cpu: 250m
      memory: 250Mi
default_worker:
  replicas: 1
  resource_requirements:
    requests:
      cpu: 25m
      memory: 200Mi
    limits:
      cpu: 500m
```

```

        memory: 400Mi
activation_worker:
  replicas: 1
  resource_requirements:
    requests:
      cpu: 25m
      memory: 150Mi
    limits:
      cpu: 500m
      memory: 400Mi
event_stream:
  replicas: 1
  resource_requirements:
    requests:
      cpu: 25m
      memory: 150Mi
    limits:
      cpu: 100m
      memory: 300Mi
hub:
  disabled: false
  ## uncomment if using file storage for Content pod
  storage_type: file
  file_storage_storage_class: nfs-local-rwx # replace with the rwx storage class
for your cluster
  file_storage_size: 50Gi

  ## uncomment if using S3 storage for Content pod
  # storage_type: S3
  # object_storage_s3_secret: example-galaxy-object-storage

  ## uncomment if using Azure storage for Content pod
  # storage_type: azure
  # object_storage_azure_secret: azure-secret-name

api:
  replicas: 1

```

```
resource_requirements:
  requests:
    cpu: 150m
    memory: 256Mi
  limits:
    cpu: 800m
    memory: 500Mi
content:
  replicas: 1
  resource_requirements:
    requests:
      cpu: 150m
      memory: 256Mi
    limits:
      cpu: 800m
      memory: 1200Mi
worker:
  replicas: 1
  resource_requirements:
    requests:
      cpu: 150m
      memory: 256Mi
    limits:
      cpu: 800m
      memory: 400Mi
web:
  replicas: 1
  resource_requirements:
    requests:
      cpu: 100m
      memory: 256Mi
    limits:
      cpu: 500m
      memory: 300Mi
redis:
  replicas: 1
```

```
resource_requirements:
  requests:
    cpu: 100m
    memory: 250Mi
  limits:
    cpu: 300m
    memory: 400Mi

# lightspeed:
#   disabled: true

# End state:
# * Controller deployed and named: myaap-controller
# * EDA deployed and named: myaap-eda
# * Hub deployed and named: myaap-hub
```

Configure your Ansible Automation Platform deployment

Configuring your Ansible Automation Platform deployment after installation customizes the platform to match your organizational requirements. Apply configuration settings to secure connections and manage storage resources.

Configuring your deployment helps you to:

- **Secure authentication flows:** Enable HTTPS redirect for single sign-on to ensure secure authentication through platform gateway.
- **Protect against security threats:** Configure CSRF protection settings to prevent cross-site request forgery attacks.
- **Manage storage resources:** Find and delete persistent volume claims to reclaim storage space and manage platform resources efficiently.

Enable HTTPS redirect for single sign-on

HTTPS redirect for SAML, allows you to log in once and access all of the platform gateway without needing to reauthenticate.

Before you begin

- You have successfully configured SAML in the gateway from the Ansible Automation Platform Operator. Refer to [Configuring SAML authentication](#) for help with this.

Procedure

1. Log in to Red Hat OpenShift Container Platform.
2. Go to **Operators > Installed Operators**.
3. Select your Ansible Automation Platform Operator deployment.
4. Select **All Instances** and go to your **AnsibleAutomationPlatform** instance.
5. Click the  icon and then select **Edit AnsibleAutomationPlatform**.
6. In the **YAML view** paste the following YAML code under the `spec:` section:

```
spec:
  extra_settings:
    - setting: REDIRECT_IS_HTTPS
      value: '"True"'
```

7. Click **Save**.

Result

After you have added the `REDIRECT_IS_HTTPS` setting, wait for the pod to redeploy automatically. You can verify this setting makes it into the pod by running:

```
oc exec -it <gateway-pod-name> -- grep REDIRECT /etc/ansible-automation-platform/gateway/settings.py
```

Configure your CSRF settings

The Red Hat Ansible Automation Platform Operator creates Openshift Routes and configures your Cross-site request forgery (CSRF) settings automatically. .

When using external ingress, you must configure your CSRF on the ingress to allow for cross-site requests. You can configure your platform gateway operator ingress under **Advanced configuration**

Procedure

1. Log in to Red Hat OpenShift Container Platform.
2. Navigate to **Operators > Installed Operators**.
3. Select your Ansible Automation Platform Operator deployment.
4. Select the **Ansible Automation Platform** tab.
5. For new instances, click **Create AnsibleAutomationPlatform**.
 - a. For existing instances, you can edit the YAML view by clicking the `:` icon and then **Edit AnsibleAutomationPlatform**.
6. Click **Advanced Configuration**.
7. Under **Ingress annotations**, enter any annotations to add to the ingress.
8. Under **Ingress TLS secret**, click the drop-down list and select a secret from the list.
9. Under **YAML view** paste in the following code:

```
spec:
  extra_settings:
    - setting: CSRF_TRUSTED_ORIGINS
      value:
        - https://my-aap-domain.com
```

10. After you have configured your platform gateway, click **Create** at the bottom of the form view (Or **Save** in the case of editing existing instances).

Result

Red Hat OpenShift Container Platform creates the pods. This may take a few minutes. You can view the progress by navigating to **Workloads > Pods** and locating the newly created instance. Verify that the following operator pods provided by the Red Hat Ansible Automation Platform Operator installation from platform gateway are running:

Operator manager controllers pods	Automation controller pods	Automation hub pods	Event-Driven Ansible pods	Platform gateway pods
The operator manager controllers for each of the four operators, include the following:	After deploying automation controller, you can see the addition of the following pods:	After deploying automation hub, you can see the addition of the following pods:	After deploying EDA, you can see the addition of the following pods: • EDA API	After deploying platform gateway, you can see the addition of the following pods: • platform gateway

Operator manager controllers pods	Automation controller pods	Automation hub pods	Event-Driven Ansible pods	Platform gateway pods
• automation-controller-operator-controller-manager • automation-hub-operator-controller-manager • resource-operator-controller-manager • aap-gateway-operator-controller-manager • ansible-lightspeed-operator-controller-manager • eda-server-operator	• Automation controller web • Automation controller task • Mesh ingress • Automation controller postgres	• Automation hub web • Automation hub task • Automation hub API • Automation hub worker	• EDA Activation • EDA worker • EDA stream • EDA Scheduler	• platform gateway redis

Operator manager controllers pods	Automation controller pods	Automation hub pods	Event-Driven Ansible pods	Platform gateway pods
-controll er- manager				

NOTE:

A missing pod can indicate the need for a pull secret. Pull secrets are required for protected or private image registries. See [Using image pull secrets](#) for more information. You can diagnose this issue further by running `oc describe pod <pod-name>` to see if there is an ImagePullBackOff error on that pod.

Find and delete PVCs

A persistent volume claim (PVC) is a storage volume used to store data that automation hub and automation controller applications use.

This persistence is a key feature of static provisioning. If you redeploy an instance using the same name, the Operator must bind to these existing PVCs, allowing for data continuity across deployments. If you are confident that you no longer need a PVC, or have backed it up elsewhere, you can manually delete them.

Procedure

1. List the existing PVCs in your deployment namespace:

```
oc get pvc -n <namespace>
```

2. Identify the PVC associated with your previous deployment by comparing the old deployment name and the PVC name.
3. Delete the old PVC:

```
oc delete pvc -n <namespace> <pvc-name>
```

Review platform gateway FAQs

Manage your Ansible Automation Platform deployment and troubleshoot common issues with these frequently asked questions. Learn about resource management, logging, and error recovery for your components.

If I delete my Ansible Automation Platform deployment will I still have access to automation controller?

No, automation controller, automation hub, and Event-Driven Ansible are nested within the deployment and are also deleted.

How must I manage parameters when adding or removing them in the Ansible Automation Platform custom resource (CR) hierarchy?

When *adding* parameters, you can add it to the Ansible Automation Platform custom resource (CR) only and those parameters will work their way down to the nested CRs.

When *removing* parameters, you have to remove them both from the Ansible Automation Platform CR *and* the nested CR, for example, the **Automation Controller** CR.

Something went wrong with my deployment but I'm not sure what, how can I find out?

You can follow along in the command line while the operator is reconciling, this can be helpful for debugging. Alternatively you can click into the deployment instance to see the status conditions being updated as the deployment goes on.

Is it still possible to view individual component logs?

When troubleshooting you should examine the **Ansible Automation Platform** instance for the main logs and then each individual component (**EDA**, **AutomationHub**, **AutomationController**) for more specific information.

Where can I view the condition of an instance?

To display status conditions click into the instance, and look under the **Details** or **Events** tab. Alternatively, to display the status conditions you can run the get command: `oc get automationcontroller <instance-name> -o jsonpath=Pipe "| jq"`

Can I track my migration in real time?

To help track the status of the migration or to understand why migration might have failed you can look at the migration logs as they are running. Use the logs command: `oc logs fresh-install-controller-migration-4.6.0-jwfm6 -f`

I have configured my SAML but authentication fails with this error: "Unable to complete social auth login"

What can I do?

You must update your Ansible Automation Platform instance to include the `REDIRECT_IS_HTTPS` extra setting. See [Enabling single sign-on \(SSO\) for platform gateway on OpenShift Container Platform](#) for help with this.

Configure automation controller

You can use these instructions to configure the automation controller operator on Red Hat OpenShift Container Platform, specify custom resources, and deploy Ansible Automation Platform with an external database.

Automation controller configuration can be done through the automation controller `extra_settings` or directly in the user interface after deployment. However, it is important to note that configurations made in `extra_settings` take precedence over settings made in the user interface.

NOTE:

When an instance of automation controller is removed, the associated PVCs are not automatically deleted. This can cause issues during migration if the new deployment has the same name as the previous one. Therefore, it is recommended that you manually remove old PVCs before deploying a new automation controller instance in the same namespace.

Prerequisites

- You have installed the Red Hat Ansible Automation Platform catalog in Operator Hub.
- For automation controller, a default StorageClass must be configured on the cluster for the operator to dynamically create needed PVCs. This is not necessary if an external PostgreSQL database is configured.
- For Hub a StorageClass that supports ReadWriteMany must be available on the cluster to dynamically create the PVC needed for the content, redis and api pods. If it is not the default StorageClass on the cluster, you can specify it when creating your AutomationHub object.

Related information

[OpenShift Container Platform product documentation](#)

[Find and delete PVCs](#)

Configure your automation controller image pull policy

Use this procedure to configure the image pull policy on your automation controller.

Procedure

1. Log in to Red Hat OpenShift Container Platform.

2. Go to **Operators > Installed Operators**.
3. Select your Ansible Automation Platform Operator deployment.
4. Select the **Ansible Automation Platform** tab.
5. Click the  icon next to your Ansible Automation Platform instance and select **Edit AnsibleAutomationPlatform**.
6. Click **YAML view** and locate the `spec.controller:` section.
7. Configure the image pull policy and resource requirements under the `controller:` section:

```
spec:
  controller:
    image_pull_policy: IfNotPresent # Options: Always, Never, IfNotPresent
    image_pull_secrets:
      - pull-secret-name
    web_resource_requirements:
      limits:
        cpu: 1000m
        memory: 2Gi
      requests:
        cpu: 500m
        memory: 1Gi
    task_resource_requirements:
      limits:
        cpu: 2000m
        memory: 4Gi
      requests:
        cpu: 1000m
        memory: 2Gi
    ee_resource_requirements:
      limits:
        cpu: 500m
        memory: 1Gi
      requests:
        cpu: 250m
        memory: 512Mi
    redis_resource_requirements:
      limits:
```

```
    cpu: 500m
    memory: 1Gi
  requests:
    cpu: 250m
    memory: 512Mi
  postgres_resource_requirements:
    limits:
      cpu: 1000m
      memory: 2Gi
    requests:
      cpu: 500m
      memory: 1Gi
  postgres_storage_requirements:
    limits:
      storage: 10Gi
    requests:
      storage: 8Gi
  replicas: 1
  garbage_collect_secrets: false
  create_preload_data: true
```

8. Click **Save**.

NOTE:

These settings apply to the automation controller component managed by this Ansible Automation Platform instance. If you specified an existing controller under `controller.name`, these settings will update that instance.

For more examples of Ansible Automation Platform custom resources, see [Red Hat Ansible Automation Platform custom resources](#)

Configure your LDAP for automation controller

You can configure your LDAP SSL configuration for automation controller through any of the following options:

- The automation controller user interface.
The platform gateway user interface. See [Configure LDAP authentication](#) for additional steps.
- The following procedure steps.

Procedure

1. Create a secret in your Ansible Automation Platform namespace for the `bundle-ca.crt` file (the filename must be `bundle-ca.crt`):

```
$ oc create secret -n aap generic bundle-ca-secret --from-file=bundle-ca.crt
```

NOTE:

The target filename for this operation must be `bundle-ca.crt` and the secret name should be `bundle-ca-secret`.

2. Add the `bundle_cacert_secret` to the Ansible Automation Platform customer resource:

```
...
spec:
  bundle_cacert_secret: bundle-ca-secret
...
```

Result

You can verify the expected certificate by running:

```
oc get deployments -l 'app.kubernetes.io/component=aap-gateway'
```

Followed by:

```
oc exec -it deployment.apps/<gateway-deployment-name-from-above> -- openssl x509
-in /etc/pki/tls/certs/ca-bundle.crt -noout -text
```

Configure automation controller route options

The Red Hat Ansible Automation Platform operator installation form allows you to further configure your automation controller operator route options under **Advanced configuration**.

IMPORTANT: You must assign a unique metadata.name to each custom resource (CR) in your namespace. If you assign an `AutomationControllerMeshIngress` the same name as your `AnsibleAutomationPlatform` installation, the operator overrides default routes and services. This conflict causes the platform installation to fail.

Procedure

1. Log in to Red Hat OpenShift Container Platform.
2. Navigate to **Operators > Installed Operators**.
3. Select your Ansible Automation Platform Operator deployment.
4. Select the **Ansible Automation Platform** tab.
5. Click the  icon next to your Ansible Automation Platform instance and select **Edit AnsibleAutomationPlatform**.
6. Click **YAML view** and locate the `spec.controller:` section..
7. Configure the route options under the `controller:` section:

```
spec:
  controller:
    ingress_type: Route
    route_host: controller.example.com # Custom hostname for the route
    route_tls_termination_mechanism: Edge # Options: Edge, Passthrough
    route_tls_secret: controller-tls-secret # Optional: TLS credential secret
    projects_persistence: false # Enable/disable persistence for /var/lib/
    projects
```

8. Click **Save**.

NOTE:

Edge termination is recommended for most instances. After configuring your route, you can customize additional route settings by adding them to the `controller:` section in the Ansible Automation Platform custom resource.

For more examples of Ansible Automation Platform custom resources, see [Red Hat Ansible Automation Platform custom resources](#).

Configure ingress options for automation controller

The Ansible Automation Platform Operator installation form allows you to further configure your automation controller operator ingress under **Advanced configuration**.

Procedure

1. Log in to Red Hat OpenShift Container Platform.
2. Navigate to **Operators > Installed Operators**.
3. Select your Ansible Automation Platform Operator deployment.
4. Select the **Ansible Automation Platform** tab.
5. Click the  icon next to your Ansible Automation Platform instance and select **Edit AnsibleAutomationPlatform**.
6. Click **YAML view** and locate the `spec.controller:` section..
7. Configure the route options under the `controller:` section:

```
spec:
  controller:
    ingress_type: Ingress
    ingress_annotations: |
      nginx.ingress.kubernetes.io/proxy-body-size: "0"
      nginx.ingress.kubernetes.io/proxy-connect-timeout: "600"
    ingress_tls_secret: controller-ingress-tls-secret
```

8. Click **Save**.

NOTE:

These settings apply to the automation controller component managed by this Ansible Automation Platform instance. The operator automatically updates the ingress configuration for the controller.

For more examples of Ansible Automation Platform custom resources, see [Red Hat Ansible Automation Platform custom resources](#)

Result

After you have configured your automation controller ingress setting, OpenShift Container Platform updates the pods. This may take a few minutes.

You can view the progress by navigating to **Workloads > Pods** and locating the newly created instance.

Verify that the following operator pods provided by the Ansible Automation Platform Operator installation from automation controller are running:

Operator manager controllers	Automation controller	Automation hub	Event-Driven Ansible (EDA)
The operator manager controllers for each of the three operators, include the following: - automation-controller-operator-controller-manager - automation-hub-operator-controller-manager - resource-operator-controller-manager - aap-gateway-operator-	After deploying automation controller, you can see the addition of the following pods: - controller - controller-postgres - controller-web - controller-task	After deploying automation hub, you can see the addition of the following pods: - hub-api - hub-content - hub-postgres - hub-redis - hub-worker	After deploying EDA, you can see the addition of the following pods: - eda-activation-worker - da-api - eda-default-worker - eda-event-stream - eda-scheduler

Operator manager controllers	Automation controller	Automation hub	Event-Driven Ansible (EDA)
controller-manager • ansible-lightspeed-operator-controller-manager • eda-server-operator-controller-manager			

NOTE:

A missing pod can indicate the need for a pull secret. Pull secrets are required for protected or private image registries. See [Using image pull secrets](#) for more information. You can diagnose this issue further by running `oc describe pod <pod-name>` to see if there is an ImagePullBackOff error on that pod.

Configure automation hub

You can use these instructions to configure the automation hub operator on Red Hat OpenShift Container Platform, specify custom resources, and deploy Ansible Automation Platform with an external database.

Automation hub configuration can be done through the automation hub `pulp_settings` or directly in the user interface after deployment. However, it is important to note that configurations made in `pulp_settings` take precedence over settings made in the user interface. Hub settings should always be set as lowercase on the Hub custom resource specification.

NOTE:

When an instance of automation hub is removed, the PVCs are not automatically deleted. This can cause issues during migration if the new deployment has the same name as the previous one. Therefore, it is recommended that you manually remove old PVCs before deploying a new automation hub instance in the same namespace.

Prerequisites

- You have installed the Ansible Automation Platform Operator in Operator Hub.

Related information

[OpenShift Container Platform product documentation](#)
[Find and delete PVCs](#)

Storage options for automation hub

Automation hub requires `ReadWriteMany` file-based storage, Azure Blob storage, or Amazon S3 storage for operation so that multiple pods can access shared content, such as collections.

The process for configuring object storage on the `AutomationHub` CR is similar for Amazon S3 and Azure Blob Storage.

If you are using file-based storage and your installation scenario includes automation hub, ensure that the storage option for Ansible Automation Platform Operator is set to `ReadWriteMany`. `ReadWriteMany` is the default storage option.

In addition, OpenShift Data Foundation provides a `ReadWriteMany` or S3 implementation. Also, you can set up NFS storage configuration to support `ReadWriteMany`. This, however, introduces the NFS server as a potential, single point of failure.

Related information

[Persistent storage using NFS](#)
[How do I create a storage class for NFS dynamic storage provisioning in an OpenShift environment?](#)

Provision OCP storage with `ReadWriteMany` access mode

To ensure successful installation of Ansible Automation Platform Operator, you must provision your storage type for automation hub initially to `ReadWriteMany` access mode.

Procedure

1. Go to **Storage > PersistentVolume**.
2. Click **Create PersistentVolume**.
3. In the first step, update the `accessModes` from the default `ReadWriteOnce` to `ReadWriteMany`.

- a. See [Provisioning](#) to update the access mode. for a detailed overview.
4. Complete the additional steps in this section to create the persistent volume claim (PVC).

Configure object storage on Amazon S3

Red Hat supports Amazon Simple Storage Service (S3) for automation hub. You can configure it when deploying the `AnsibleAutomationPlatform` custom resource (CR), or you can configure it for an existing instance.

Before you begin

- Create an Amazon S3 bucket to store the objects.
- Note the name of the S3 bucket.

Procedure

1. Create a Kubernetes secret containing the AWS credentials and connection details, and the name of your Amazon S3 bucket. The following example creates a secret called `test-s3` :

```
$ oc -n $HUB_NAMESPACE apply -f- <<EOF
apiVersion: v1
kind: Secret
metadata:
  name: 'test-s3'
stringData:
  s3-access-key-id: $S3_ACCESS_KEY_ID
  s3-secret-access-key: $S3_SECRET_ACCESS_KEY
  s3-bucket-name: $S3_BUCKET_NAME
  s3-region: $S3_REGION
EOF
```

2. Add the secret to the Ansible Automation Platform custom resource (CR) under the `hub` section in the `spec` :

```

apiVersion: aap.ansible.com/v1alpha1
kind: AnsibleAutomationPlatform
metadata:
  name: myaap
spec:
  hub:
    storage_type: S3
    object_storage_s3_secret: test-s3

```

NOTE:

If you have an existing automation hub instance, specify its name using `hub.name: existing-hub-name` to apply these settings to the existing instance.

For more examples of Ansible Automation Platform custom resources, see [Red Hat Ansible Automation Platform custom resources](#)

3. If you are applying this secret to an existing instance, restart the API pods for the change to take effect. `<hub-name>` is the name of your hub instance.

```
$ oc -n $HUB_NAMESPACE delete pod -l app.kubernetes.io/name=<hub-name>-api
```

Configure object storage on Azure Blob

Red Hat supports Azure Blob Storage for automation hub. You can configure it when deploying the `AnsibleAutomationPlatform` custom resource (CR), or you can configure it for an existing instance.

Before you begin

- Create an Azure Storage blob container to store the objects.
- Note the name of the blob container.

Procedure

1. Create a Kubernetes secret containing the credentials and connection details for your Azure account, and the name of your Azure Storage blob container. The following example creates a secret called `test-azure`:

```
$ oc -n $HUB_NAMESPACE apply -f- <<EOF
apiVersion: v1
kind: Secret
metadata:
  name: 'test-azure'
stringData:
  azure-account-name: $AZURE_ACCOUNT_NAME
  azure-account-key: $AZURE_ACCOUNT_KEY
  azure-container: $AZURE_CONTAINER
  azure-container-path: $AZURE_CONTAINER_PATH
  azure-connection-string: $AZURE_CONNECTION_STRING
EOF
```

2. Add the secret to the Ansible Automation Platform custom resource (CR) under the `hub` section in the `spec`:

```
apiVersion: aap.ansible.com/v1alpha1
kind: AnsibleAutomationPlatform
metadata:
  name: myaap
spec:
  hub:
    storage_type: azure
    object_storage_azure_secret: test-azure
```

NOTE:

If you have an existing automation hub instance, specify its name using `hub.name: existing-hub-name` to apply these settings to the existing instance.

For more examples of Ansible Automation Platform custom resources, see [Red Hat Ansible Automation Platform custom resources](#)

3. If you are applying this secret to an existing instance, restart the API pods for the change to take effect. `<hub-name>` is the name of your hub instance.

```
$ oc -n $HUB_NAMESPACE delete pod -l app.kubernetes.io/name=<hub-name>-api
```

Configure automation hub route options

The Red Hat Ansible Automation Platform operator installation form allows you to further configure your automation hub operator route options under **Advanced configuration**.

Procedure

1. Log in to Red Hat OpenShift Container Platform.
2. Navigate to **Operators > Installed Operators**.
3. Select your Ansible Automation Platform Operator deployment.
4. Select the **Ansible Automation Platform** tab.
5. Click the  icon next to your Ansible Automation Platform instance and select **Edit AnsibleAutomationPlatform**.
6. Click **YAML view** and locate the `spec.hub:` section..
7. Configure the route options under the `hub:` section:

```
spec:
  hub:
    ingress_type: Route
    route_host: hub.example.com # Custom hostname for the route
    route_tls_termination_mechanism: Edge # Options: Edge, Passthrough
    route_tls_secret: hub-tls-secret # Optional: TLS credential secret
```

8. Click **Save**.

NOTE:

Edge termination is recommended for most instances. After configuring your route, you can customize additional route settings by adding them to the `hub:` section in the Ansible Automation Platform custom resource.

For more examples of Ansible Automation Platform custom resources, see [Red Hat Ansible Automation Platform custom resources](#).

- 10.

Configure ingress options for automation hub

The Ansible Automation Platform Operator installation form allows you to further configure your automation hub operator ingress under **Advanced configuration**.

Procedure

1. Log in to Red Hat OpenShift Container Platform.
2. Navigate to **Operators > Installed Operators**.
3. Select your Ansible Automation Platform Operator deployment.
4. Select the **Ansible Automation Platform** tab.
5. Click the  icon next to your Ansible Automation Platform instance and select **Edit AnsibleAutomationPlatform**.
6. Click **YAML view** and locate the `spec.hub:` section..
7. Configure the route options under the `hub:` section:

```
spec:
  hub:
    ingress_type: Ingress
    ingress_annotations: |
      nginx.ingress.kubernetes.io/proxy-body-size: "0"
      nginx.ingress.kubernetes.io/proxy-connect-timeout: "600"
    ingress_tls_secret: hub-ingress-tls-secret
```

8. Click **Save**.

NOTE:

These settings apply to the automation hub component managed by this Ansible Automation Platform instance. The operator automatically updates the ingress configuration for the hub.

For more examples of Ansible Automation Platform custom resources, see [Red Hat Ansible Automation Platform custom resources](#)

9. Result

After you have configured your automation hub ingress settings, OpenShift Container Platform updates the pods. This may take a few minutes.

You can view the progress by navigating to **Workloads > Pods** and locating the newly created instance.

Verify that the following operator pods provided by the Ansible Automation Platform Operator installation from automation hub are running:

Operator manager controllers	Automation controller	Automation hub
The operator manager controllers for each of the 3 operators, include the following: • automation-controller-operator-controller-manager • automation-hub-operator-controller-manager • resource-operator-controller-manager	After deploying automation controller, you will see the addition of these pods: • controller • controller-postgres	After deploying automation hub, you will see the addition of these pods: • hub-api • hub-content • hub-postgres • hub-redis • hub-worker

NOTE:

A missing pod can indicate the need for a pull secret. Pull secrets are required for protected or private image registries. See [Using image pull secrets](#) for more information. You can diagnose this issue further by running `oc describe pod <pod-name>` to see if there is an ImagePullBackOff error on that pod.

Finding the automation hub route

You can access the automation hub through the platform gateway or through the following procedure.

Procedure

1. Log into Red Hat OpenShift Container Platform.
2. Navigate to **Networking > Routes**.
3. Under **Location**, click on the URL for your automation hub instance.

Result

The automation hub user interface launches where you can sign in with the administrator credentials specified during the operator configuration process.

NOTE:

If you did not specify an administrator password during configuration, one was automatically created for you. To locate this password, go to your project, select **Workloads > Secrets** and open controller-admin-password. From there you can copy the password and paste it into the Automation hub password field.

Add a collection download count to automation hub

A collection download count can help you understand collection usage. To add a collection download count to automation hub, set the following configuration:

```
spec:
  pulp_settings:
    ansible_collect_download_count: true
```

When `ansible_collect_download_count` is enabled, automation hub will display a download count by the collection.

Add allowed container registries

Before you can deploy a container image in automation hub, you must add the registry to the `allowedRegistries` in the automation controller image configuration. To do this you can copy and paste the following code into your automation controller image YAML.

Procedure

1. Log in to **Red Hat OpenShift Container Platform**.
2. Navigate to **Home > Search**.
3. Select the **Resources** drop-down list and type "Image".
4. Select **Image (config.openshift.io/v1)**.
5. Click **Cluster** under the **Name** heading.
6. Select the **YAML** tab.
7. Paste in the following under spec value:

```
spec:
  registrySources:
    allowedRegistries:
      - quay.io
      - registry.redhat.io
      - image-registry.openshift-image-registry.svc:5000
      - <OCP route for your automation hub>
```

8. Click **Save**.

Configure content signing for automation hub

As an automation administrator for your organization, you can configure Ansible Automation Platform Hub Operator for signing and publishing Ansible content collections from different groups within your organization.

Before you begin

- A GPG key pair. If you do not have one, you can generate one using the `gpg --full-generate-key` command.
- Your public-private key pair has proper access for configuring content signing on Ansible Automation Platform Hub Operator.

For additional security, automation creators can configure Ansible-Galaxy CLI to verify these collections to ensure that they have not been changed after they were uploaded to automation hub.

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

Procedure

1. Create a ConfigMap for signing scripts. The ConfigMap you create contains the scripts used by the signing service for collections and container images.

NOTE:

This script is used as part of the signing service and must generate an ascii-armored detached `gpg` signature for that file using the key specified through the `PULP_SIGNING_KEY_FINGERPRINT` environment variable.

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

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

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

Example: The following script produces signatures for content:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: signing-scripts
data:
  collection_sign.sh: |-
    #!/usr/bin/env bash

    FILE_PATH=$1
    SIGNATURE_PATH=$1.asc

    ADMIN_ID="$PULP_SIGNING_KEY_FINGERPRINT"
    PASSWORD="password"

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

    # Check the exit status
    STATUS=$?
    if [ $STATUS -eq 0 ]; then
        echo {"file\: \"$FILE_PATH\", \"signature\: \"$SIGNATURE_PATH\"}
    else
        exit $STATUS
    fi
  container_sign.sh: |-
    #!/usr/bin/env bash

    # galaxy_container SigningService will pass the next 4 variables to the
```

```

script.
    MANIFEST_PATH=$1
    FINGERPRINT="$PULP_SIGNING_KEY_FINGERPRINT"
    IMAGE_REFERENCE="$REFERENCE"
    SIGNATURE_PATH="$SIG_PATH"

    # Create container signature using skopeo
    skopeo standalone-sign \
        $MANIFEST_PATH \
        $IMAGE_REFERENCE \
        $FINGERPRINT \
        --output $SIGNATURE_PATH

    # Optionally pass the passphrase to the key if password protected.
    # --passphrase-file /path/to/key_password.txt

    # Check the exit status
    STATUS=$?
    if [ $STATUS -eq 0 ]; then
        echo {"signature_path": "$SIGNATURE_PATH"}
    else
        exit $STATUS
    fi

```

2. Create a secret for your GnuPG private key. This secret securely stores the GnuPG private key you use for signing.

```

gpg --export --armor <your-gpg-key-id> > signing_service.gpg

oc create secret generic signing-galaxy --from-file=signing_service.gpg

```

The secret must have a key named `signing_service.gpg`.

3. Configure the AnsibleAutomationPlatform CR.

```

apiVersion: aap.ansible.com/v1alpha1
kind: AnsibleAutomationPlatform
metadata:
  name: aap-hub-signing-sample
spec:
  hub:
    signing_secret: "signing-galaxy"
    signing_scripts_configmap: "signing-scripts"

```

Next steps

[Using content signing services in private automation hub](#)

Configure custom PostgreSQL settings

The `postgres_extra_settings` variable allows you to pass a list of custom *name: value* pairs directly to the PostgreSQL configuration file (`/var/lib/pgsql/data/postgresql.conf`) within the database pod.

Before you begin

- You have installed the Ansible Automation Platform Operator.

Procedure

1. Log in to Red Hat OpenShift Container Platform.
2. Go to **Operators** → **Installed Operators**.
3. Select your Ansible Automation Platform Operator deployment.
4. Select **All Instances** and go to your **Ansible Automation Platform** instance.
5. Click the `:` icon and then select **Edit Ansible Automation Platform**.
6. In the **YAML** view, locate the `spec:` section
7. Add the `database` section and the required settings under `spec:`. The following example configures SSL ciphers and the maximum connections:

```
spec:
  database:
    postgres_extra_settings:
      - name: max_connections
        value: '1000'
```

8. Click **Save**.

Result

Inspect the PostgreSQL pod logs to verify the new settings.

Alternatively, you can run the following command to check the settings. Replace `<aap postgres pod>` with the name of your PostgreSQL pod.

```
$ oc exec -it <aap postgres pod> -- psql -d gateway -c "SHOW max_connections;"
```

Configure an external database for Ansible Automation Platform

Configure an external database for Ansible Automation Platform Operator to use your own database infrastructure.

Using an external database lets you share and reuse resources and manually manage backups, upgrades, and performance optimizations. To configure an external database, create Kubernetes secrets with credentials for connecting to the database. The same external database (PostgreSQL instance) can be used for platform gateway, automation controller, automation hub, and Event-Driven Ansible controller as long as the database names are different.

NOTE: The automation hub database requires the `hstore` PostgreSQL extension.

Configure an external database for platform gateway on Red Hat Ansible Automation Platform Operator

There are two scenarios for deploying Ansible Automation Platform with an external database:

Before you begin

The external database must be a PostgreSQL database that is the version supported by the current release of Ansible Automation Platform. The external postgres instance credentials and connection information must be stored in a secret, which is then set on the platform gateway spec.

NOTE:

Ansible Automation Platform 2.6 supports PostgreSQL 15 for its managed databases and additionally supports PostgreSQL 15, 16, and 17 for external databases.

If you choose to use an externally managed database with version 16 or 17 you must also rely on external backup and restore processes.

Scenario	Action required
Fresh install	You must specify a single external database instance for the platform to use for the following: * Platform gateway * Automation controller * Automation hub * Event-Driven Ansible * Red Hat Ansible Lightspeed (If enabled) See the <i>aap-configuring-external-db-all-default-components.yml</i> example in the Custom resources section for help with this. If using Red Hat Ansible Lightspeed, use the <i>aap-configuring-external-db-with-lightspeed-enabled.yml</i> example.
Existing external database in 2.4	Your existing external database remains the same after upgrading but you must specify the <code>external-postgres-configuration-gateway</code> (spec.database.database_secret) on the Ansible Automation Platform custom resource. For detailed steps, see Upgrading an external database for platform gateway on on Ansible Automation Platform

To deploy Ansible Automation Platform with an external database, you must first create a Kubernetes secret with credentials for connecting to the database.

By default, the Ansible Automation Platform Operator automatically creates and configures a managed PostgreSQL pod in the same namespace as your Ansible Automation Platform deployment. You can deploy Ansible Automation Platform with an external database instead of the managed PostgreSQL pod that the Ansible Automation Platform Operator automatically creates.

Using an external database lets you share and reuse resources and manually manage backups, upgrades, and performance optimizations.

NOTE:

The same external database (PostgreSQL instance) can be used for both automation hub, automation controller, and platform gateway as long as the database names are different. In other words, you can have multiple databases with different names inside a single PostgreSQL instance.

The following section outlines the steps to configure an external database for your platform gateway on a Ansible Automation Platform Operator.

Procedure

1. Create a `postgres_configuration_secret` YAML file, following the template below:

```
apiVersion: v1
kind: Secret
metadata:
  name: external-postgres-configuration
  namespace: <target_namespace>
stringData:
  host: "<external_ip_or_url_resolvable_by_the_cluster>"
  port: "<external_port>"
  database: "<desired_database_name>"
  username: "<username_to_connect_as>"
  password: "<password_to_connect_with>"
  type: "unmanaged"
type: Opaque
```

Where:

<target_namespace>

Namespace to create the secret in. This should be the same namespace you want to deploy to.

<external_ip_or_url_resolvable_by_the_cluster>

The resolvable hostname for your database node.

<external_port>

External port defaults to 5432 .

<password_to_connect_with>

Value for variable `password` should not contain single or double quotes (', ") or backslashes (\) to avoid any issues during deployment, backup or restoration.

2. Apply `external-postgres-configuration-secret.yml` to your cluster using the `oc create` command.

```
$ oc create -f external-postgres-configuration-secret.yml
```

NOTE:

The following example is for a platform gateway deployment. To configure an external database for all components, use the *aap-configuring-external-db-all-default-components.yml* example in the [Custom resources](#) section.

3. When creating your `AnsibleAutomationPlatform` custom resource object, specify the secret on your spec, following the example below:

```
apiVersion: aap.ansible.com/v1alpha1
kind: AnsibleAutomationPlatform
metadata:
  name: example-aap
  Namespace: aap
spec:
  database:
    database_secret: automation-platform-postgres-configuration
```

Configure an external database for automation controller on Red Hat Ansible Automation Platform Operator

For users who prefer to deploy Ansible Automation Platform with an external database, they can do so by configuring a secret with instance credentials and connection information, then applying it to their cluster using the `oc create` command.

Before you begin

The external database must be a PostgreSQL database that is the version supported by the current release of Ansible Automation Platform. The external postgres instance credentials and connection information must be stored in a secret, which is then set on the automation controller spec.

NOTE:

Ansible Automation Platform 2.6 supports PostgreSQL 15 for its managed databases and additionally supports PostgreSQL 15, 16, and 17 for external databases.

If you choose to use an externally managed database with version 16 or 17 you must also rely on external backup and restore processes.

By default, the Ansible Automation Platform Operator automatically creates and configures a managed PostgreSQL pod in the same namespace as your Ansible Automation Platform deployment. You can deploy Ansible Automation Platform with an external database instead of the managed PostgreSQL pod that the Ansible Automation Platform Operator automatically creates.

Using an external database lets you share and reuse resources and manually manage backups, upgrades, and performance optimizations.

NOTE:

The same external database (PostgreSQL instance) can be used for both automation hub, automation controller, and platform gateway as long as the database names are different. In other words, you can have multiple databases with different names inside a single PostgreSQL instance.

The following section outlines the steps to configure an external database for your automation controller on a Ansible Automation Platform Operator.

Procedure

1. Create a `postgres_configuration_secret` YAML file, following the template below:

```

apiVersion: v1
kind: Secret
metadata:
  name: external-postgres-configuration
  namespace: <target_namespace>
stringData:
  host: "<external_ip_or_url_resolvable_by_the_cluster>"
  port: "<external_port>"
  database: "<desired_database_name>"
  username: "<username_to_connect_as>"
  password: "<password_to_connect_with>"
  sslmode: "prefer"
  type: "unmanaged"
type: Opaque

```

When configuring the secret:

- `namespace` : Specify the namespace to create the secret in. This should be the same namespace you want to deploy to.
 - `host` : Specify the resolvable hostname for your database node.
 - `port` : Specify the external port. The default is `5432` .
 - `password` : Ensure the password does not contain single or double quotes (', ") or backslashes (\) to avoid any issues during deployment, backup or restoration.
 - `sslmode` : This variable is valid for external databases only. The allowed values are: `prefer` , `disable` , `allow` , `require` , `verify-ca` , and `verify-full` .
2. Apply `external-postgres-configuration-secret.yml` to your cluster using the `oc create` command.

```
$ oc create -f external-postgres-configuration-secret.yml
```

3. When creating your `AnsibleAutomationPlatform` custom resource object, specify the secret under the `controller` section in your spec, following the example below:

```

apiVersion: aap.ansible.com/v1alpha1
kind: AnsibleAutomationPlatform
metadata:
  name: myaap
spec:
  controller:
    name: controller-dev # Optional: specify existing instance or custom name
    postgres_configuration_secret: external-postgres-configuration

```

NOTE:

If you have an existing automation controller instance, specify its name under `controller.name` to apply these settings to the existing instance. If you omit the `name` field, the operator will create a new instance with the default name pattern `<aap-instance-name>-controller`.

For more examples of Ansible Automation Platform custom resources, see [Red Hat Ansible Automation Platform custom resources](#)

Configure an external database for automation hub on Red Hat Ansible Automation Platform Operator

For users who prefer to deploy Ansible Automation Platform with an external database, they can do so by configuring a secret with instance credentials and connection information, then applying it to their cluster using the `oc create` command.

Before you begin

The external database must be a PostgreSQL database that is the version supported by the current release of Ansible Automation Platform. The external postgres instance credentials and connection information will need to be stored in a secret, which will then be set on the automation hub spec.

NOTE:

Ansible Automation Platform 2.6 supports PostgreSQL 15 for its managed databases and additionally supports PostgreSQL 15, 16, and 17 for external databases.

If you choose to use an externally managed database with version 16 or 17 you must also rely on external backup and restore processes.

By default, the Ansible Automation Platform Operator automatically creates and configures a managed PostgreSQL pod in the same namespace as your Ansible Automation Platform deployment.

You can choose to use an external database instead if you prefer to use a dedicated node to ensure dedicated resources or to manually manage backups, upgrades, or performance tweaks.

NOTE:

The same external database (PostgreSQL instance) can be used for both automation hub, automation controller, and platform gateway as long as the database names are different. In other words, you can have multiple databases with different names inside a single PostgreSQL instance.

The following section outlines the steps to configure an external database for your automation hub on a Ansible Automation Platform Operator.

Procedure

1. Create a `postgres_configuration_secret` YAML file, following the template below:

```
apiVersion: v1
kind: Secret
metadata:
  name: external-postgres-configuration
  namespace: <target_namespace> ❶
stringData:
  host: "<external_ip_or_url_resolvable_by_the_cluster>" ❷
  port: "<external_port>" ❸
  database: "<desired_database_name>"
  username: "<username_to_connect_as>"
  password: "<password_to_connect_with>" ❹
  sslmode: "prefer" ❺
  type: "unmanaged"
type: Opaque
```

- a. Namespace to create the secret in. This should be the same namespace you want to deploy to.
 - b. The resolvable hostname for your database node.
 - c. External port defaults to 5432 .
 - d. Value for variable `password` should not contain single or double quotes (', ") or backslashes (\) to avoid any issues during deployment, backup or restoration.
 - e. The variable `sslmode` is valid for `external` databases only. The allowed values are: **prefer** , **disable** , **allow** , **require** , **verify-ca** , and **verify-full** .
2. Apply `external-postgres-configuration-secret.yml` to your cluster using the `oc create` command.

```
$ oc create -f external-postgres-configuration-secret.yml
```

3. When creating your `AnsibleAutomationPlatform` custom resource object, specify the secret under the `hub` section in your spec, following the example below:

```
apiVersion: aap.ansible.com/v1alpha1
kind: AnsibleAutomationPlatform
metadata:
  name: myaap
spec:
  hub:
    name: hub-dev # Optional: specify existing instance or custom name
    postgres_configuration_secret: external-postgres-configuration
    storage_type: file
    file_storage_storage_class: <your-read-write-many-storage-class>
    file_storage_size: 10Gi
```

NOTE:

If you have an existing automation hub instance, specify its name under `hub.name` to apply these settings to the existing instance. If you omit the `name` field, the operator will create a new instance with the default name pattern `<aap-instance-name>-hub` .

For more examples of Ansible Automation Platform custom resources, see [Red Hat Ansible Automation Platform custom resources](#)

Upgrading an external database for platform gateway on Ansible Automation Platform

To upgrade from Ansible Automation Platform 2.4 to 2.6 with an external database, you must scale down your Operator deployment, upgrade your PostgreSQL, then scale your deployment back up.

Before you begin

- A 2.4 automation controller and automation hub deployment with an external PostgreSQL 13 database
- A newly provisioned PostgreSQL 15 database for the new platform gateway component

Procedure

1. Create a secret `postgres-config-gateway` with PostgreSQL 15 credentials for the platform gateway component. For example:

```
apiVersion: v1
kind: Secret
metadata:
  name: postgres-config-gateway
  namespace: aap
stringData:
  host: "<DB_HOST_OR_IP>"
  port: "<DB_PORT>"      # default is 5432
  database: "<DB_NAME>" # for example "gateway"
  username: "<DB_USER>" # for example "gateway"
  password: "<DB_PASSWORD>"
  sslmode: "prefer"
  type: "unmanaged"
type: Opaque
```

2. Add your newly created secret to your Ansible Automation Platform instance:


```
spec:
  postgres_configuration_secret: postgres-config-gateway
```

3. Scale down your deployments in their respective namespaces using:

```
oc scale deployment --replicas=0 -n <component-namespace> <component-deployment>
```

4. Automation controller:

- a. automation-controller-operator-controller-manager
- b. <controller-name>-controller-task
- c. <controller-name>-controller-web

5. Automation hub:

- a. automation-hub-operator-controller-manager
- b. <hub-name>-hub-api
- c. <hub-name>-hub-content
- d. <hub-name>-hub-redis
- e. <hub-name>-hub-worker

6. The remaining operators:

- a. ansible-lightspeed-operator-controller-manager
- b. eda-server-operator-controller-manager
- c. resource-operator-controller-manager

7. Upgrade your PostgreSQL 13 to PostgreSQL 15.

8. Scale your deployments back up using:

```
oc scale deployment --replicas=1 -n <component-namespace> <component-deployment>
```

9. Log into OpenShift Container Platform.

10. Go to **Operators > Installed Operators**.

11. Click the  icon next to your deployment and then click **Edit Subscription**

12. From the **Details** tab, select **Update Channel**.

13. Select **stable-2.6** as the channel and click **Save**.

14. Deploy Ansible Automation Platform 2.6 using the following custom resource (CR):

```
apiVersion: aap.ansible.com/v1alpha1
kind: AnsibleAutomationPlatform
metadata:
  name: aap
spec:

  database:
    database_secret: postgres-config-gateway

  controller:
    name: existing-controller

  eda:
    disabled: true

  hub:
    name: existing-hub
```

Result

To verify your upgrade was successful, go to your users, collection, job history or similar and confirm that they are on the new 2.6 instance and in the new PostgreSQL 15 databases.

Enable the hstore extension

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

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

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

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

Procedure

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

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

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

Example output with `hstore` available:

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

Example output with `hstore` not available:

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

3. On a RHEL based server, the `hstore` extension is included in the `postgresql-contrib` RPM package, which is not installed automatically when installing the PostgreSQL server RPM package.

To install the RPM package, use the following command:

```
dnf install postgresql-contrib
```

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

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

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

```

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

```

Troubleshoot an external database

When upgrading the Ansible Automation Platform Operator you may encounter an error like the following:

```

NotImplementedError: can't parse timestampz with DateStyle 'Redwood, SHOW_TIME':
'18-MAY-23 20:33:55.765755 +00:00'

```

Errors like this occur when you have an external database with an unexpected DateStyle set. You can refer to the following steps to resolve this issue.

Procedure

1. Edit the `/var/lib/pgsql/data/postgres.conf` file on the database server:

```
# vi /var/lib/pgsql/data/postgres.conf
```

2. Find and comment out the line:

```
#datestyle = 'Redwood, SHOW_TIME'
```

3. Add the following setting immediately below the newly-commented line:

```
datestyle = 'iso, mdy'
```

4. Save and close the `postgres.conf` file.
5. Reload the database configuration:

```
# systemctl reload postgresql
```

NOTE:

Running this command does not disrupt database operations.

Configure Redis

When you create an Ansible Automation Platform instance through the Ansible Automation Platform Operator, standalone Redis is assigned by default. If you would prefer to deploy clustered Redis, you can use the following procedure.

Before you begin

- You have installed an Ansible Automation Platform Operator deployment.

IMPORTANT:

Switching Redis modes on an existing instance is not supported and can lead to unexpected consequences, including data loss. To change the Redis mode, you must deploy a new instance.

Procedure

1. Log in to Red Hat OpenShift Container Platform.
2. Navigate to **Operators > Installed Operators**.
3. Select your Ansible Automation Platform Operator deployment.
4. Select the **Details** tab.
5. On the **Ansible Automation Platform** tile click **Create instance**.
 - a. For existing instances, you can edit the YAML view by clicking the `:` icon and then **Edit AnsibleAutomationPlatform**.
 - b. Change the **redis_mode** value to "cluster".
 - c. Click **Reload**, then **Save**.
6. Click to expand **Advanced configuration**.
7. For the **Redis Mode** list, select **Cluster**.
8. Configure the rest of your instance as necessary, then click **Create**.

Result

Your instance deploys with a cluster Redis with 6 Redis replicas as default.

NOTE:

You can modify your automation hub default redis cache PVC volume size, for help with this see, [Modifying the default redis cache PVC volume size automation hub](#).

Scale down your Ansible Automation Platform Operator deployment

You can scale down all Ansible Automation Platform deployments and StatefulSets by using the `idle_aap` variable. This is useful for scenarios such as upgrades, migrations, or disaster recovery.

Procedure

1. Log in to Red Hat OpenShift Container Platform.
2. Go to **Operators > Installed Operators**.
3. Select your Ansible Automation Platform Operator deployment.
4. Select **All Instances** and go to your **AnsibleAutomationPlatform** instance.
5. Click the `:` icon and then select **Edit AnsibleAutomationPlatform**.
6. In the **YAML view** paste the following YAML code under the `spec:` section:

```
idle_aap: true
```

7. Click **Save**.

Next steps

Setting the `idle_aap` value to `true` scales down all active deployments. Setting the value to `false` scales the deployments back up.

Ansible Automation Platform Resource Operator

The **Ansible Automation Platform Resource Operator** is a custom resource that allows you to manage and define automation controller objects, such as projects and job templates, through YAML files.

Resource Operator overview

Resource Operator is a custom resource (CR) that you can deploy after you have created your platform gateway deployment.

With Resource Operator you can define resources such as projects, job templates, and inventories in YAML files.

Ansible Automation Platform then uses the YAML files to create these resources. You can create the YAML through the **Form view** that prompts you for keys and values for your YAML code. Alternatively, to work with YAML directly, you can select **YAML view**.

The Resource Operator provides the following CRs:

- AnsibleJob
- JobTemplate
- Automation controller project
- Automation controller schedule
- Automation controller workflow
- Automation controller workflow template:
- Automation controller inventory
- Automation controller credential

Use Resource Operator

The Resource Operator itself does not do anything until the user creates an object. As soon as the user creates an **AutomationControllerProject** or **AnsibleJob** resource, the Resource Operator starts processing that object.

Before you begin

- Install the Kubernetes-based cluster of your choice.
- Deploy automation controller using the `automation-controller-operator`.

Procedure

1. After installing the `automation-controller-resource-operator` in your cluster, you must create a Kubernetes (k8s) secret with the connection information for your Ansible Automation Platform instance.
2. Then you can use Resource Operator to create a k8s resource to manage your Ansible Automation Platform instance.

Connect Resource Operator to Ansible Automation Platform

To connect Resource Operator with platform gateway you must create a Kubernetes secret with the connection information for your Ansible Automation Platform instance.

Use the following procedure to create an OAuth2 token for your user in the platform gateway UI.

NOTE:

You can only create OAuth 2 Tokens for your own user through the API or UI, which means you can only configure or view tokens from your own user profile.

Procedure

1. Log in to Red Hat OpenShift Container Platform.
2. In the navigation panel, select **Access Management > Users**.
3. Select the username you want to create a token for.
4. Select **Tokens > Automation Execution**
5. Click **Create Token**.
6. You can leave **Applications** empty. Add a description and select **Read** or **Write** for the **Scope**.

NOTE:

Make sure you provide a valid user when creating tokens. Otherwise, you get an error message that you tried to issue the command without either specifying a user, or supplying a username that does not exist.

Create an Ansible Automation Platform connection secret for Resource Operator

To make your connection information available to the Resource Operator, create a k8s secret with the token and host value.

Procedure

1. The following is an example of the YAML for the connection secret. Save the following example to a file, for example, `aap-connection-secret.yml`.


```
apiVersion: v1
kind: Secret
metadata:
  name: aap-access
  type: Opaque
stringData:
  token: <generated-token>
  host: https://aap-host.example.com/
```

2. Edit the file with your host and token value.
3. Apply it to your cluster by running the `kubectl create` command:

```
kubectl create -f aap-connection-secret.yml
```

Create a custom resource for Resource Operator

Use the Resource Operator to manage Ansible Automation Platform resources directly from your Kubernetes cluster. This section provides the procedures for creating custom resources like **AnsibleJob**, **JobTemplate**, **AnsibleProject**, and more.

Create an AnsibleJob custom resource

An AnsibleJob custom resource launches a job in the Ansible Automation Platform instance specified in the Kubernetes secret (automation controller host URL, token). You can launch an automation job on automation controller by creating an AnsibleJob resource.

Procedure

1. Specify the connection secret and job template you want to launch.

```

apiVersion: tower.ansible.com/v1alpha1
kind: AnsibleJob
metadata:
  generateName: demo-job-1 # generate a unique suffix per 'kubectl create'
spec:
  connection_secret: aap-access
  job_template_name: Demo Job Template

```

2. Configure features such as, inventory, extra variables, and time to live for the job.

```

spec:
  connection_secret: aap-access
  job_template_name: Demo Job Template
  inventory: Demo Inventory # Inventory prompt on launch
  needs_to_be_enabled: true
  runner_image: quay.io/ansible/controller-resource-runner
  runner_version: latest
  job_ttl: 100
  extra_vars: # Extra variables prompt on
  launch needs to be enabled
    test_var: test
  job_tags: "provision,install,configuration" # Specify tags to run
  skip_tags: "configuration,restart" # Skip tasks with a given tag

```

NOTE:

You must enable prompt on launch for inventories and extra variables if you are configuring those. To enable **Prompt on launch**, within the automation controller UI: From the **Resources > Templates** page, select your template and select the **Prompt on launch** checkbox next to **Inventory** and **Variables** sections.

3. Launch a workflow job template with an AnsibleJob object by specifying the `workflow_template_name` instead of `job_template_name`:

```

apiVersion: tower.ansible.com/v1alpha1
kind: AnsibleJob
metadata:
  generateName: demo-job-1 # generate a unique suffix per 'kubectl create'
spec:
  connection_secret: aap-access
  workflow_template_name: Demo Workflow Template

```

Create a JobTemplate custom resource

A job template is a definition and set of parameters for running an Ansible job. For more information see [Standardize and streamline automation with automation job templates](#).

- Create a job template on automation controller by creating a JobTemplate custom resource:

```

apiVersion: tower.ansible.com/v1alpha1
kind: JobTemplate
metadata:
  name: jobtemplate-4
spec:
  connection_secret: aap-access
  job_template_name: ExampleJobTemplate4
  job_template_project: Demo Project
  job_template_playbook: hello_world.yml
  job_template_inventory: Demo Inventory

```

Create an automation controller project custom resource

A Project is a logical collection of Ansible playbooks, represented in automation controller. For more information see [Logically group playbooks with projects](#).

- Create a project on automation controller by creating an automation controller project custom resource:

```
apiVersion: tower.ansible.com/v1alpha1
kind: AnsibleProject
metadata:
  name: git
spec:
  repo: https://github.com/ansible/ansible-tower-samples
  branch: main
  name: ProjectDemo-git
  scm_type: git
  organization: Default
  description: demoProject
  connection_secret: aap-access
  runner_pull_policy: IfNotPresent
```

Create an automation controller schedule custom resource

Define an `AnsibleSchedule` custom resource to create a schedule on the automation controller, ensuring you specify the necessary `apiVersion`, `kind`, and a unique `metadata.name`.

- Create a schedule on automation controller by creating an automation controller schedule custom resource:

```

apiVersion: tower.ansible.com/v1alpha1
kind: AnsibleSchedule
metadata:
  name: schedule
spec:
  connection_secret: aap-access
  runner_pull_policy: IfNotPresent
  name: "Demo Schedule"
  rrule: "DTSTART:20210101T000000Z RRULE:FREQ=DAILY;INTERVAL=1;COUNT=1"
  unified_job_template: "Demo Job Template"

```

Create an automation controller workflow custom resource

Workflows enable you to configure a sequence of disparate job templates (or workflow templates) that may or may not share inventory, playbooks, or permissions. For more information see [Workflows in automation controller](#).

- Create a workflow on automation controller by creating a workflow custom resource:

```

apiVersion: tower.ansible.com/v1alpha1
kind: AnsibleWorkflow
metadata:
  name: workflow
spec:
  inventory: Demo Inventory
  workflow_template_name: Demo Job Template
  connection_secret: aap-access
  runner_pull_policy: IfNotPresent

```

Create an automation controller workflow template custom resource

A workflow job template links together a sequence of disparate resources to track the full set of jobs that were part of the release process as a single unit.

For more information see [Workflow job templates](#).

- Create a workflow template on automation controller by creating a workflow template custom resource:

```
apiVersion: tower.ansible.com/v1alpha1
kind: WorkflowTemplate
metadata:
  name: workflowtemplate-sample
spec:
  connection_secret: aap-access
  name: ExampleTowerWorkflow
  description: Example Workflow Template
  organization: Default
  inventory: Demo Inventory
  workflow_nodes:
    - identifier: node101
      unified_job_template:
        name: Demo Job Template
        inventory:
          organization:
            name: Default
          type: job_template
    - identifier: node102
      unified_job_template:
        name: Demo Job Template
        inventory:
          organization:
            name: Default
          type: job_template
```

Create an automation controller inventory custom resource

By using an inventory file, Ansible Automation Platform can manage a large number of hosts with a single command.

Inventories also help you use Ansible Automation Platform more efficiently by reducing the number of command line options you have to specify. For more information see [Inventories](#).

- Create an inventory on automation controller by creating an inventory custom resource:

```
metadata:
  name: inventory-new
spec:
  connection_secret: aap-access
  description: my new inventory
  name: newinventory
  organization: Default
  state: present
  instance_groups:
    - default
  variables:
    string: "string_value"
    bool: true
    number: 1
    list:
      - item1: true
      - item2: "1"
    object:
      string: "string_value"
      number: 2
```

Create an automation controller credential custom resource

Credentials authenticate the automation controller user when launching jobs against machines, synchronizing with inventory sources, and importing project content from a version control system.

SSH and AWS are the most commonly used credentials. For a full list of supported credentials see [Credential types](#).

For help with defining values you can refer to the [OpenAPI \(Swagger\) file for Red Hat Ansible Automation Platform API](#) KCS article.

NOTE:

You can use `https://<aap-instance>/api/controller/v2/credential_types/` to view the list of credential types on your instance. To get the full list use the following `curl` command:

```
export AAP_TOKEN="your-oauth2-token"
export AAP_URL="https://your-aap-controller.example.com"

curl -s -H "Authorization: Bearer $AAP_TOKEN" "$AAP_URL/api/controller/v2/credential_types/" | jq -r '.results[].name'
```

- Create an AWS or SSH credential on automation controller by creating a credential custom resource:
 - SSH credential:


```
apiVersion: tower.ansible.com/v1alpha1
kind: AnsibleCredential
metadata:
  name: ssh-cred
spec:
  name: ssh-cred
  organization: Default
  connection_secret: aap-access
  description: "SSH credential"
  type: "Machine"
  ssh_username: "cat"
  ssh_secret: my-ssh-secret
  runner_pull_policy: IfNotPresent
```

- AWS credential:

```
apiVersion: tower.ansible.com/v1alpha1
kind: AnsibleCredential
metadata:
  name: aws-cred
spec:
  name: aws-access
  organization: Default
  connection_secret: aap-access
  description: "This is a test credential"
  type: "Amazon Web Services"
  username_secret: aws-secret
  password_secret: aws-secret
  runner_pull_policy: IfNotPresent
```

Red Hat Ansible Automation Platform custom resources

This appendix provides a reference for the Ansible Automation Platform custom resources for various deployment scenarios.

NOTE:

You can link in existing components by specifying the component name under the `name` variable. You can also use `name` to create a custom name for a new component.

Custom resources

Refer to the following custom resources you can use for various Ansible Automation Platform deployment scenarios.

aap-existing-controller-and-hub-new-eda.yml

```
---
apiVersion: aap.ansible.com/v1alpha1
kind: AnsibleAutomationPlatform
metadata:
  name: myaap
spec:
  # Development purposes only
  no_log: false

  controller:
    name: existing-controller
    disabled: false

  eda:
    disabled: false

  hub:
    name: existing-hub
    disabled: false
```

aap-all-defaults.yml

```

apiVersion: aap.ansible.com/v1alpha1
kind: AnsibleAutomationPlatform
metadata:
  name: myaap
spec:
  # Development purposes only
  no_log: false

  # Platform
  ## uncomment to test bundle certs
  # bundle_cacert_secret: gateway-custom-certs

  # Components

  hub:
    disabled: false
    ## uncomment if using file storage for Content pod
    storage_type: file
    file_storage_storage_class: nfs-local-rwx
    file_storage_size: 10Gi

    ## uncomment if using S3 storage for Content pod
    # storage_type: S3
    # object_storage_s3_secret: example-galaxy-object-storage

    ## uncomment if using Azure storage for Content pod
    # storage_type: azure
    # object_storage_azure_secret: azure-secret-name

  # lightspeed:
  #   disabled: true

  # End state:
  # * Automation controller deployed and named: myaap-controller
  # * * Event-Driven Ansible deployed and named: myaap-eda
  # * * Automation hub deployed and named: myaap-hub

```

aap-existing-controller-only.yml

```

---
apiVersion: aap.ansible.com/v1alpha1
kind: AnsibleAutomationPlatform
metadata:
  name: myaap
spec:
  # Development purposes only
  no_log: false

  controller:
    name: existing-controller

  eda:
    disabled: true

  hub:
    disabled: true

    ## uncomment if using file storage for Content pod
    # storage_type: file
    # file_storage_storage_class: nfs-local-rwx
    # file_storage_size: 10Gi

    ## uncomment if using S3 storage for Content pod
    # storage_type: S3
    # object_storage_s3_secret: example-galaxy-object-storage

    ## uncomment if using Azure storage for Content pod
    # storage_type: azure
    # object_storage_azure_secret: azure-secret-name

# End state:
# * Automation controller: existing-controller registered with Ansible Automation Platform UI
# * * Event-Driven Ansible deployed and named: myaap-eda
# * * Automation hub deployed and named: myaap-hub

```

aap-existing-hub-and-controller.yml

```
---
apiVersion: aap.ansible.com/v1alpha1
kind: AnsibleAutomationPlatform
metadata:
  name: myaap
spec:
  # Development purposes only
  no_log: false

  controller:
    name: existing-controller
    disabled: false

  eda:
    disabled: true

  hub:
    name: existing-hub
    disabled: false

# End state:
# * Automation controller: existing-controller registered with Ansible Automation Platform UI
# * * Event-Driven Ansible deployed and named: myaap-eda
# * * Automation hub: existing-hub registered with Ansible Automation Platform UI
```


aap-existing-hub-controller-eda.yml

```

---
apiVersion: aap.ansible.com/v1alpha1
kind: AnsibleAutomationPlatform
metadata:
  name: myaap
spec:
  # Development purposes only
  no_log: false

  controller:
    name: existing-controller # <-- this is the name of the existing
AutomationController CR
    disabled: false

  eda:
    name: existing-eda
    disabled: false

  hub:
    name: existing-hub
    disabled: false

# End state:
# * Controller: existing-controller registered with Ansible Automation Platform UI
# * * Event-Driven Ansible: existing-eda registered with Ansible Automation Platform
UI
# * * Automation hub: existing-hub registered with Ansible Automation Platform UI
#

# Note: The automation controller, Event-Driven Ansible, and automation hub names
must match the names of the existing.

# Automation controller, Event-Driven Ansible, and automation hub CRs in the same
namespace as the Ansible Automation Platform CR. If the names do not match, the
Ansible Automation Platform CR will not be able to register the existing automation
controller, Event-Driven Ansible, and automation hub with the Ansible Automation
Platform UI, and will instead deploy new automation controller, Event-Driven Ansible,
and automation hub instances.

```

aap-existing-hub-controller-eda.yml

```

---
apiVersion: aap.ansible.com/v1alpha1
kind: AnsibleAutomationPlatform
metadata:
  name: myaap
spec:
  # Development purposes only
  no_log: false

  controller:
    name: existing-controller # <-- this is the name of the existing
AutomationController CR
    disabled: false

  eda:
    name: existing-eda
    disabled: false

  hub:
    name: existing-hub
    disabled: false

# End state:

# * Automation controller: existing-controller registered with Ansible Automation
Platform UI

# * * Event-Driven Ansible: existing-eda registered with Ansible Automation Platform
UI

# * * Automation hub: existing-hub registered with Ansible Automation Platform UI

#

# Note: The automation controller, Event-Driven Ansible, and automation hub names
must match the names of the existing.

# Automation controller, Event-Driven Ansible, and automation hub CRs in the same
namespace as the Ansible Automation Platform CR. If the names do not match, the
Ansible Automation Platform CR will not be able to register the existing automation
controller, Event-Driven Ansible, and automation hub with the Ansible Automation
Platform UI, and will instead deploy new automation controller, Event-Driven Ansible,
and automation hub instances.

```

aap-fresh-controller-eda.yml

```

---
apiVersion: aap.ansible.com/v1alpha1
kind: AnsibleAutomationPlatform
metadata:
  name: myaap
spec:
  # Development purposes only
  no_log: false

  controller:
    disabled: false

  eda:
    disabled: false

  hub:
    disabled: true

    ## uncomment if using file storage for Content pod
    storage_type: file
    file_storage_storage_class: nfs-local-rwx
    file_storage_size: 10Gi

    ## uncomment if using S3 storage for Content pod
    # storage_type: S3
    # object_storage_s3_secret: example-galaxy-object-storage

    ## uncomment if using Azure storage for Content pod
    # storage_type: azure
    # object_storage_azure_secret: azure-secret-name

# End state:
# * Automation controller deployed and named: myaap-controller
# * * Event-Driven Ansible deployed and named: myaap-eda
# * * Automation hub disabled
# * Red Hat Ansible Lightspeed disabled

```

INSTALL

aap-fresh-external-db.yml

```

---
apiVersion: aap.ansible.com/v1alpha1
kind: AnsibleAutomationPlatform
metadata:
  name: myaap
spec:
  # Development purposes only
  no_log: false

  controller:
    disabled: false

  eda:
    disabled: false

  hub:
    disabled: false
    ## uncomment if using file storage for Content pod
    storage_type: file
    file_storage_storage_class: nfs-local-rwx
    file_storage_size: 10Gi

    ## uncomment if using S3 storage for Content pod
    # storage_type: S3
    # object_storage_s3_secret: example-galaxy-object-storage

    ## uncomment if using Azure storage for Content pod
    # storage_type: azure
    # object_storage_azure_secret: azure-secret-name

# End state:
# * Automation controller deployed and named: myaap-controller
# * * Event-Driven Ansible deployed and named: myaap-eda
# * * Automation hub deployed and named: myaap-hub

```

aap-configuring-external-db-all-default-components.yml

```
---
apiVersion: aap.ansible.com/v1alpha1
kind: AnsibleAutomationPlatform
metadata:
  name: myaap
spec:
  database:
    database_secret: external-postgres-configuration-gateway
  controller:
    postgres_configuration_secret: external-postgres-configuration-controller
  hub:
    postgres_configuration_secret: external-postgres-configuration-hub
  eda:
    database:
      database_secret: external-postgres-configuration-eda
```

aap-configuring-existing-external-db-all-default-components.yml

```
---
apiVersion: aap.ansible.com/v1alpha1
kind: AnsibleAutomationPlatform
metadata:
  name: myaap
spec:
  database:
    database_secret: external-postgres-configuration-gateway
```

NOTE:

The system uses the external database for platform gateway, and automation controller, automation hub, and Event-Driven Ansible continues to use the existing databases that were used in 2.4.

aap-configuring-external-db-with-lightspeed-enabled.yml

```
---
apiVersion: aap.ansible.com/v1alpha1
kind: AnsibleAutomationPlatform
metadata:
  name: myaap
spec:
  database:
    database_secret: external-postgres-configuration-gateway
  controller:
    postgres_configuration_secret: external-postgres-configuration-controller
  hub:
    postgres_configuration_secret: external-postgres-configuration-hub
  eda:
    database:
      database_secret: external-postgres-configuration-eda
  lightspeed:
    disabled: false
    database:
      database_secret: <secret-name>-postgres-configuration
      auth_config_secret_name: 'auth-configuration-secret'
      model_config_secret_name: 'model-configuration-secret'
```

NOTE:

You can follow the [Red Hat Ansible Lightspeed with IBM watsonx Code Assistant User Guide](#) for help with creating the model and auth secrets.

aap-fresh-install-with-settings.yml

```
---
apiVersion: aap.ansible.com/v1alpha1
kind: AnsibleAutomationPlatform
metadata:
  name: myaap
```



```
spec:
  # Development purposes only
  no_log: false
  image_pull_policy: Always

  # Platform
  ## uncomment to test bundle certs
  # bundle_cacert_secret: gateway-custom-certs

  # Components
  controller:
    disabled: false
    image_pull_policy: Always

    extra_settings:
      - setting: MAX_PAGE_SIZE
        value: '501'

  eda:
    disabled: false
    image_pull_policy: Always

    extra_settings:
      - setting: EDA_MAX_PAGE_SIZE
        value: '501'

  hub:
    disabled: false
    image_pull_policy: Always

    ## uncomment if using file storage for Content pod
    storage_type: file
    file_storage_storage_class: rook-cephfs
    file_storage_size: 10Gi

    ## uncomment if using S3 storage for Content pod
```

```

# storage_type: S3
# object_storage_s3_secret: example-galaxy-object-storage

## uncomment if using Azure storage for Content pod
# storage_type: azure
# object_storage_azure_secret: azure-secret-name

pulp_settings:
  MAX_PAGE_SIZE: 501
  cache_enabled: false

# lightspeed:
#   disabled: true

# End state:
# * Automation controller deployed and named: myaap-controller
# * * Event-Driven Ansible deployed and named: myaap-eda
# * * Automation hub deployed and named: myaap-hub

```

aap-fresh-install.yml

```

---
apiVersion: aap.ansible.com/v1alpha1
kind: AnsibleAutomationPlatform
metadata:
  name: myaap
spec:
  # Development purposes only
  no_log: false

  # Redis Mode
  # redis_mode: cluster

  # Platform
  ## uncomment to test bundle certs

```

```

# bundle_cacert_secret: gateway-custom-certs
# extra_settings:
#   - setting: MAX_PAGE_SIZE
#     value: '501'

# Components
controller:
  disabled: false

eda:
  disabled: false

hub:
  disabled: false
  ## uncomment if using file storage for Content pod
  storage_type: file
  file_storage_storage_class: nfs-local-rwx
  file_storage_size: 10Gi

  ## uncomment if using S3 storage for Content pod
  # storage_type: S3
  # object_storage_s3_secret: example-galaxy-object-storage

  ## uncomment if using Azure storage for Content pod
  # storage_type: azure
  # object_storage_azure_secret: azure-secret-name

# lightspeed:
#   disabled: true

# End state:
# * Automation controller deployed and named: myaap-controller
# * * Event-Driven Ansible deployed and named: myaap-eda
# * * Automation hub deployed and named: myaap-hub

```

aap-fresh-only-controller.yml

```

---
apiVersion: aap.ansible.com/v1alpha1
kind: AnsibleAutomationPlatform
metadata:
  name: myaap
spec:
  # Development purposes only
  no_log: false

  controller:
    disabled: false

  eda:
    disabled: true

  hub:
    disabled: true
    ## uncomment if using file storage for Content pod
    # storage_type: file
    # file_storage_storage_class: nfs-local-rwx
    # file_storage_size: 10Gi

    ## uncomment if using S3 storage for Content pod
    # storage_type: S3
    # object_storage_s3_secret: example-galaxy-object-storage

    ## uncomment if using Azure storage for Content pod
    # storage_type: azure
    # object_storage_azure_secret: azure-secret-name

# End state:
# * Automation controller: existing-controller registered with Ansible Automation Platform UI
# * * Event-Driven Ansible deployed and named: myaap-eda
# * * Automation hub deployed and named: myaap-hub

```

aap-fresh-only-hub.yml

```
---
apiVersion: aap.ansible.com/v1alpha1
kind: AnsibleAutomationPlatform
metadata:
  name: myaap
spec:
  # Development purposes only
  no_log: false

  controller:
    disabled: true

  eda:
    disabled: true

  hub:
    disabled: false
    ## uncomment if using file storage for Content pod
    storage_type: file
    file_storage_storage_class: nfs-local-rwx
    file_storage_size: 10Gi

    # # AaaS Hub Settings
    # pulp_settings:
    #   cache_enabled: false

    ## uncomment if using S3 storage for Content pod
    # storage_type: S3
    # object_storage_s3_secret: example-galaxy-object-storage

    ## uncomment if using Azure storage for Content pod
    # storage_type: azure
    # object_storage_azure_secret: azure-secret-name
```

```

lightspeed:
  disabled: false

# End state:
# * Automation controller disabled
# * * Event-Driven Ansible disabled
# * * Automation hub deployed and named: myaap-hub
# * Red Hat Ansible Lightspeed disabled

```

aap-lightspeed-enabled.yml

```

---
apiVersion: aap.ansible.com/v1alpha1
kind: AnsibleAutomationPlatform
metadata:
  name: myaap
spec:
  # Development purposes only
  no_log: false

  controller:
    disabled: false

  eda:
    disabled: false

  hub:
    disabled: false
    ## uncomment if using file storage for Content pod
    storage_type: file
    file_storage_storage_class: nfs-local-rwx
    file_storage_size: 10Gi

    ## uncomment if using S3 storage for Content pod
    # storage_type: S3

```

```
# object_storage_s3_secret: example-galaxy-object-storage

## uncomment if using Azure storage for Content pod
# storage_type: azure
# object_storage_azure_secret: azure-secret-name

lightspeed:
  disabled: false

# End state:
# * Automation controller deployed and named: myaap-controller
# * * Event-Driven Ansible deployed and named: myaap-eda
# * * Automation hub deployed and named: myaap-hub
# * Red Hat Ansible Lightspeed deployed and named: myaap-lightspeed
```


gateway-only.yml

```
---
apiVersion: aap.ansible.com/v1alpha1
kind: AnsibleAutomationPlatform
metadata:
  name: myaap
spec:
  # Development purposes only
  no_log: false

  controller:
    disabled: true

  eda:
    disabled: true

  hub:
    disabled: true

  lightspeed:
    disabled: true

# End state:
# * Platform gateway deployed and named: myaap-gateway
#   * UI is reachable at: https://myaap-gateway-gateway.apps.ocp4.example.com
# * Automation controller is not deployed
# * * Event-Driven Ansible is not deployed
# * * Automation hub is not deployed
# * Red Hat Ansible Lightspeed is not deployed
```

Custom resources

Refer to the following custom resources you can use for various Ansible Automation Platform deployment scenarios.

aap-existing-controller-and-hub-new-eda.yml

```
---
apiVersion: aap.ansible.com/v1alpha1
kind: AnsibleAutomationPlatform
metadata:
  name: myaap
spec:
  # Development purposes only
  no_log: false

  controller:
    name: existing-controller
    disabled: false

  eda:
    disabled: false

  hub:
    name: existing-hub
    disabled: false
```

aap-all-defaults.yml

```

apiVersion: aap.ansible.com/v1alpha1
kind: AnsibleAutomationPlatform
metadata:
  name: myaap
spec:
  # Development purposes only
  no_log: false

  # Platform
  ## uncomment to test bundle certs
  # bundle_cacert_secret: gateway-custom-certs

  # Components

  hub:
    disabled: false
    ## uncomment if using file storage for Content pod
    storage_type: file
    file_storage_storage_class: nfs-local-rwx
    file_storage_size: 10Gi

    ## uncomment if using S3 storage for Content pod
    # storage_type: S3
    # object_storage_s3_secret: example-galaxy-object-storage

    ## uncomment if using Azure storage for Content pod
    # storage_type: azure
    # object_storage_azure_secret: azure-secret-name

  # lightspeed:
  #   disabled: true

  # End state:
  # * Automation controller deployed and named: myaap-controller
  # * * Event-Driven Ansible deployed and named: myaap-eda
  # * * Automation hub deployed and named: myaap-hub

```

aap-existing-controller-only.yml

```

---
apiVersion: aap.ansible.com/v1alpha1
kind: AnsibleAutomationPlatform
metadata:
  name: myaap
spec:
  # Development purposes only
  no_log: false

  controller:
    name: existing-controller

  eda:
    disabled: true

  hub:
    disabled: true
    ## uncomment if using file storage for Content pod
    # storage_type: file
    # file_storage_storage_class: nfs-local-rwx
    # file_storage_size: 10Gi

    ## uncomment if using S3 storage for Content pod
    # storage_type: S3
    # object_storage_s3_secret: example-galaxy-object-storage

    ## uncomment if using Azure storage for Content pod
    # storage_type: azure
    # object_storage_azure_secret: azure-secret-name

# End state:
# * Automation controller: existing-controller registered with Ansible Automation
  Platform UI
# * * Event-Driven Ansible deployed and named: myaap-eda
# * * Automation hub deployed and named: myaap-hub

```

aap-existing-hub-and-controller.yml

```
---
apiVersion: aap.ansible.com/v1alpha1
kind: AnsibleAutomationPlatform
metadata:
  name: myaap
spec:
  # Development purposes only
  no_log: false

  controller:
    name: existing-controller
    disabled: false

  eda:
    disabled: true

  hub:
    name: existing-hub
    disabled: false

# End state:
# * Automation controller: existing-controller registered with Ansible Automation
#   Platform UI
# * * Event-Driven Ansible deployed and named: myaap-eda
# * * Automation hub: existing-hub registered with Ansible Automation Platform UI
```

aap-existing-hub-controller-eda.yml

```

---
apiVersion: aap.ansible.com/v1alpha1
kind: AnsibleAutomationPlatform
metadata:
  name: myaap
spec:
  # Development purposes only
  no_log: false

  controller:
    name: existing-controller # <-- this is the name of the existing
AutomationController CR
    disabled: false

  eda:
    name: existing-eda
    disabled: false

  hub:
    name: existing-hub
    disabled: false

# End state:
# * Controller: existing-controller registered with Ansible Automation Platform UI
# * * Event-Driven Ansible: existing-eda registered with Ansible Automation Platform
UI
# * * Automation hub: existing-hub registered with Ansible Automation Platform UI
#
# Note: The automation controller, Event-Driven Ansible, and automation hub names
must match the names of the existing.

# Automation controller, Event-Driven Ansible, and automation hub CRs in the same
namespace as the Ansible Automation Platform CR. If the names do not match, the
Ansible Automation Platform CR will not be able to register the existing automation
controller, Event-Driven Ansible, and automation hub with the Ansible Automation
Platform UI, and will instead deploy new automation controller, Event-Driven Ansible,
and automation hub instances.

```


aap-existing-hub-controller-eda.yml

```

---
apiVersion: aap.ansible.com/v1alpha1
kind: AnsibleAutomationPlatform
metadata:
  name: myaap
spec:
  # Development purposes only
  no_log: false

  controller:
    name: existing-controller # <-- this is the name of the existing
AutomationController CR
    disabled: false

  eda:
    name: existing-eda
    disabled: false

  hub:
    name: existing-hub
    disabled: false

# End state:
# * Automation controller: existing-controller registered with Ansible Automation
Platform UI
# * * Event-Driven Ansible: existing-eda registered with Ansible Automation Platform
UI
# * * Automation hub: existing-hub registered with Ansible Automation Platform UI
#
# Note: The automation controller, Event-Driven Ansible, and automation hub names
must match the names of the existing.

# Automation controller, Event-Driven Ansible, and automation hub CRs in the same
namespace as the Ansible Automation Platform CR. If the names do not match, the
Ansible Automation Platform CR will not be able to register the existing automation
controller, Event-Driven Ansible, and automation hub with the Ansible Automation
Platform UI, and will instead deploy new automation controller, Event-Driven Ansible,
and automation hub instances.

```

INSTALL

aap-fresh-controller-eda.yml

```

---
apiVersion: aap.ansible.com/v1alpha1
kind: AnsibleAutomationPlatform
metadata:
  name: myaap
spec:
  # Development purposes only
  no_log: false

  controller:
    disabled: false

  eda:
    disabled: false

  hub:
    disabled: true

    ## uncomment if using file storage for Content pod
    storage_type: file
    file_storage_storage_class: nfs-local-rwx
    file_storage_size: 10Gi

    ## uncomment if using S3 storage for Content pod
    # storage_type: S3
    # object_storage_s3_secret: example-galaxy-object-storage

    ## uncomment if using Azure storage for Content pod
    # storage_type: azure
    # object_storage_azure_secret: azure-secret-name

# End state:
# * Automation controller deployed and named: myaap-controller
# * * Event-Driven Ansible deployed and named: myaap-eda
# * * Automation hub disabled
# * Red Hat Ansible Lightspeed disabled

```

aap-fresh-external-db.yml

```

---
apiVersion: aap.ansible.com/v1alpha1
kind: AnsibleAutomationPlatform
metadata:
  name: myaap
spec:
  # Development purposes only
  no_log: false

  controller:
    disabled: false

  eda:
    disabled: false

  hub:
    disabled: false
    ## uncomment if using file storage for Content pod
    storage_type: file
    file_storage_storage_class: nfs-local-rwx
    file_storage_size: 10Gi

    ## uncomment if using S3 storage for Content pod
    # storage_type: S3
    # object_storage_s3_secret: example-galaxy-object-storage

    ## uncomment if using Azure storage for Content pod
    # storage_type: azure
    # object_storage_azure_secret: azure-secret-name

# End state:
# * Automation controller deployed and named: myaap-controller
# * * Event-Driven Ansible deployed and named: myaap-eda
# * * Automation hub deployed and named: myaap-hub

```

aap-configuring-external-db-all-default-components.yml

```
---
apiVersion: aap.ansible.com/v1alpha1
kind: AnsibleAutomationPlatform
metadata:
  name: myaap
spec:
  database:
    database_secret: external-postgres-configuration-gateway
  controller:
    postgres_configuration_secret: external-postgres-configuration-controller
  hub:
    postgres_configuration_secret: external-postgres-configuration-hub
  eda:
    database:
      database_secret: external-postgres-configuration-eda
```

aap-configuring-existing-external-db-all-default-components.yml

```
---
apiVersion: aap.ansible.com/v1alpha1
kind: AnsibleAutomationPlatform
metadata:
  name: myaap
spec:
  database:
    database_secret: external-postgres-configuration-gateway
```

NOTE:

The system uses the external database for platform gateway, and automation controller, automation hub, and Event-Driven Ansible continues to use the existing databases that were used in 2.4.

aap-configuring-external-db-with-lightspeed-enabled.yml

```
---
apiVersion: aap.ansible.com/v1alpha1
kind: AnsibleAutomationPlatform
metadata:
  name: myaap
spec:
  database:
    database_secret: external-postgres-configuration-gateway
  controller:
    postgres_configuration_secret: external-postgres-configuration-controller
  hub:
    postgres_configuration_secret: external-postgres-configuration-hub
  eda:
    database:
      database_secret: external-postgres-configuration-eda
  lightspeed:
    disabled: false
    database:
      database_secret: <secret-name>-postgres-configuration
      auth_config_secret_name: 'auth-configuration-secret'
      model_config_secret_name: 'model-configuration-secret'
```

NOTE:

You can follow the [Red Hat Ansible Lightspeed with IBM watsonx Code Assistant User Guide](#) for help with creating the model and auth secrets.

aap-fresh-install-with-settings.yml

```
---
apiVersion: aap.ansible.com/v1alpha1
kind: AnsibleAutomationPlatform
metadata:
  name: myaap
```

```
spec:
  # Development purposes only
  no_log: false
  image_pull_policy: Always

  # Platform
  ## uncomment to test bundle certs
  # bundle_cacert_secret: gateway-custom-certs

  # Components
  controller:
    disabled: false
    image_pull_policy: Always

    extra_settings:
      - setting: MAX_PAGE_SIZE
        value: '501'

  eda:
    disabled: false
    image_pull_policy: Always

    extra_settings:
      - setting: EDA_MAX_PAGE_SIZE
        value: '501'

  hub:
    disabled: false
    image_pull_policy: Always

    ## uncomment if using file storage for Content pod
    storage_type: file
    file_storage_storage_class: rook-cephfs
    file_storage_size: 10Gi

    ## uncomment if using S3 storage for Content pod
```



```

# storage_type: S3
# object_storage_s3_secret: example-galaxy-object-storage

## uncomment if using Azure storage for Content pod
# storage_type: azure
# object_storage_azure_secret: azure-secret-name

pulp_settings:
  MAX_PAGE_SIZE: 501
  cache_enabled: false

# lightspeed:
#   disabled: true

# End state:
# * Automation controller deployed and named: myaap-controller
# * * Event-Driven Ansible deployed and named: myaap-eda
# * * Automation hub deployed and named: myaap-hub

```

aap-fresh-install.yml

```

---
apiVersion: aap.ansible.com/v1alpha1
kind: AnsibleAutomationPlatform
metadata:
  name: myaap
spec:
  # Development purposes only
  no_log: false

  # Redis Mode
  # redis_mode: cluster

  # Platform
  ## uncomment to test bundle certs

```

```

# bundle_cacert_secret: gateway-custom-certs
# extra_settings:
#   - setting: MAX_PAGE_SIZE
#     value: '501'

# Components
controller:
  disabled: false

eda:
  disabled: false

hub:
  disabled: false
  ## uncomment if using file storage for Content pod
  storage_type: file
  file_storage_storage_class: nfs-local-rwx
  file_storage_size: 10Gi

  ## uncomment if using S3 storage for Content pod
  # storage_type: S3
  # object_storage_s3_secret: example-galaxy-object-storage

  ## uncomment if using Azure storage for Content pod
  # storage_type: azure
  # object_storage_azure_secret: azure-secret-name

# lightspeed:
#   disabled: true

# End state:
# * Automation controller deployed and named: myaap-controller
# * * Event-Driven Ansible deployed and named: myaap-eda
# * * Automation hub deployed and named: myaap-hub

```

aap-fresh-only-controller.yml

```

---
apiVersion: aap.ansible.com/v1alpha1
kind: AnsibleAutomationPlatform
metadata:
  name: myaap
spec:
  # Development purposes only
  no_log: false

  controller:
    disabled: false

  eda:
    disabled: true

  hub:
    disabled: true
    ## uncomment if using file storage for Content pod
    # storage_type: file
    # file_storage_storage_class: nfs-local-rwx
    # file_storage_size: 10Gi

    ## uncomment if using S3 storage for Content pod
    # storage_type: S3
    # object_storage_s3_secret: example-galaxy-object-storage

    ## uncomment if using Azure storage for Content pod
    # storage_type: azure
    # object_storage_azure_secret: azure-secret-name

# End state:
# * Automation controller: existing-controller registered with Ansible Automation Platform UI
# * * Event-Driven Ansible deployed and named: myaap-eda
# * * Automation hub deployed and named: myaap-hub

```

aap-fresh-only-hub.yml

```
---
apiVersion: aap.ansible.com/v1alpha1
kind: AnsibleAutomationPlatform
metadata:
  name: myaap
spec:
  # Development purposes only
  no_log: false

  controller:
    disabled: true

  eda:
    disabled: true

  hub:
    disabled: false
    ## uncomment if using file storage for Content pod
    storage_type: file
    file_storage_storage_class: nfs-local-rwx
    file_storage_size: 10Gi

    # # AaaS Hub Settings
    # pulp_settings:
    #   cache_enabled: false

    ## uncomment if using S3 storage for Content pod
    # storage_type: S3
    # object_storage_s3_secret: example-galaxy-object-storage

    ## uncomment if using Azure storage for Content pod
    # storage_type: azure
    # object_storage_azure_secret: azure-secret-name
```

```

lightspeed:
  disabled: false

# End state:
# * Automation controller disabled
# * * Event-Driven Ansible disabled
# * * Automation hub deployed and named: myaap-hub
# * Red Hat Ansible Lightspeed disabled

```

aap-lightspeed-enabled.yml

```

---
apiVersion: aap.ansible.com/v1alpha1
kind: AnsibleAutomationPlatform
metadata:
  name: myaap
spec:
  # Development purposes only
  no_log: false

  controller:
    disabled: false

  eda:
    disabled: false

  hub:
    disabled: false
    ## uncomment if using file storage for Content pod
    storage_type: file
    file_storage_storage_class: nfs-local-rwx
    file_storage_size: 10Gi

    ## uncomment if using S3 storage for Content pod
    # storage_type: S3

```

```
# object_storage_s3_secret: example-galaxy-object-storage

## uncomment if using Azure storage for Content pod
# storage_type: azure
# object_storage_azure_secret: azure-secret-name

lightspeed:
  disabled: false

# End state:
# * Automation controller deployed and named: myaap-controller
# * * Event-Driven Ansible deployed and named: myaap-eda
# * * Automation hub deployed and named: myaap-hub
# * Red Hat Ansible Lightspeed deployed and named: myaap-lightspeed
```

gateway-only.yml

```

---
apiVersion: aap.ansible.com/v1alpha1
kind: AnsibleAutomationPlatform
metadata:
  name: myaap
spec:
  # Development purposes only
  no_log: false

  controller:
    disabled: true

  eda:
    disabled: true

  hub:
    disabled: true

  lightspeed:
    disabled: true

# End state:
# * Platform gateway deployed and named: myaap-gateway
#   * UI is reachable at: https://myaap-gateway-gateway.apps.ocp4.example.com
# * Automation controller is not deployed
# * * Event-Driven Ansible is not deployed
# * * Automation hub is not deployed
# * Red Hat Ansible Lightspeed is not deployed

```

Install RPM-based Ansible Automation Platform

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

Prerequisites

Prerequisites define the software, accounts, and system configuration for a successful Ansible Automation Platform installation. Meeting these ensures compatibility, prevents errors like outdated RHEL nodes, and verifies access to required installation files.

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

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

- You have created a Red Hat Registry Service Account, by using the instructions in [Creating Registry Service Accounts](#).

Related information

[System requirements](#)

[Choose a deployment method and topology](#)

[Installation and deployment models](#)

[Download Red Hat Ansible Automation Platform](#)

[Creating Registry Service Accounts](#)

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

- Obtain root access either through the `sudo` command, or through privilege escalation.
 - De-escalate privileges from root to users such as: AWX, PostgreSQL, Event-Driven Ansible, or Pulp.
- Configured an NTP client on all nodes.

Red Hat Ansible Automation Platform system requirements

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

A resilient deployment requires 10 virtual machines with a minimum of 16 gigabytes (GB) of RAM and 4 virtual CPUs (vCPU). See [Choose a deployment method and topology](#) for more information on topology options.

Base system

Type	Description	Notes
Subscription	Valid Red Hat Ansible Automation Platform subscription	
Operating system	Red Hat Enterprise Linux 9.4 or later minor versions of Red Hat Enterprise Linux 9	Red Hat Ansible Automation Platform are also supported on OpenShift, see Installing on OpenShift Container Platform for more information.
CPU architecture	x86_64, AArch64, s390x (IBM Z), ppc64le (IBM Power)	
Ansible-core	Ansible-core version 2.16 or later	Ansible Automation Platform uses the system-wide ansible-core package to install the platform, but uses ansible-core 2.16 for both its control plane and built-in execution environments.
Browser	A currently supported version of Mozilla Firefox or Google Chrome.	
Database	- For Ansible Automation Platform managed databases: PostgreSQL 15. - For customer provided (external) databases: PostgreSQL 15, 16, or 17.	- External (customer supported) databases require International Components for Unicode (ICU) support. - External databases using PostgreSQL 16 or 17 must rely on external backup and restore processes. Backup and restore functionality is

Type	Description	Notes
		dependent on utilities provided with PostgreSQL 15.
IP version	IPv4, IPv6 (single-stack and dual-stack)	

Virtual machine requirements

Component	RAM	vCPU	Disk IOPS	Storage
Platform gateway	16GB	4	3000	60GB minimum
Control nodes	16GB	4	3000	80GB minimum with at least 20GB available under <code>/var/lib/awx</code>
Execution nodes	16GB	4	3000	60GB minimum
Hop nodes	16GB	4	3000	60GB minimum
Automation hub	16GB	4	3000	60GB minimum with at least 40GB allocated to <code>/var/lib/pulp</code>
Database	16GB	4	3000	100GB minimum allocated to <code>/var/lib/pgsql</code>
Event-Driven Ansible controller	16GB	4	3000	60GB minimum

NOTE:

These are minimum requirements and can be increased for larger workloads in increments of 2x (for example 16GB becomes 32GB and 4 vCPU becomes 8vCPU). See [Horizontally scale tested deployment models to improve performance](#) for more information.

Repository requirements

Enable the following repositories only when installing Red Hat Ansible Automation Platform:

- RHEL BaseOS
- RHEL AppStream

NOTE:

If you enable repositories besides those mentioned above, the Red Hat Ansible Automation Platform installation could fail unexpectedly.

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

- Ensure that the [Network ports and protocols table](#) are available for successful connection and download of collections from automation hub or Ansible Galaxy server.

Additional notes for Red Hat Ansible Automation Platform requirements

- The Ansible Automation Platform database backups are staged on each node at `/var/backups/automation-platform` through the variable `backup_dir`. You might need to mount a new volume to `/var/backups` or change the staging location with the variable `backup_dir` to prevent issues with disk space before running the `./setup.sh -b` script.
- If performing a bundled Ansible Automation Platform installation, the installation `setup.sh` script attempts to install `ansible-core` (and its dependencies) from the bundle for you.
- If you have installed `Ansible-core` manually, the Ansible Automation Platform installation `setup.sh` script detects that Ansible has been installed and does not attempt to reinstall it.

NOTE:

You must use `Ansible-core`, which is installed by using DNF. `Ansible-core` version 2.16 is required for versions 2.6 and later.

Platform gateway system requirements

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.

Automation controller system requirements

Automation controller is a distributed system, where different software components can be co-located or deployed across many compute nodes.

The installation program provides four node types 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:

Recommended Resource Sizing for Automation controller Node Types

Node Type	RAM (Minimum)	vCPU (Minimum)	Disk IOPS (Minimum)	Storage and Notes
Execution Node	16 GB	4 vCPU	3000	Runs automation. Increase RAM/CPU to increase capacity for concurrent job forks . Performance depends heavily on the number of jobs run simultaneously.
Control Node	16 GB	4 vCPU	3000	Processes events and runs cluster jobs (e.g., project updates). * Storage: 80GB minimum, with at least 20GB available under <code>/var/lib/awx</code> . * Storage Requirement: Volume must be rated for a minimum baseline of 3000 IOPS.
Hybrid Node	16 GB	4 vCPU	3000	A combination of Control and Execution node functions. Storage requirements generally match the Control Node.
Hop Node	16 GB	4 vCPU	3000	Routes traffic within the automation mesh (e.g., bastion host). RAM can affect throughput, but CPU activity is typically low. Network latency is a more important factor than RAM or CPU.

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

Related information

[Attaching your Red Hat Ansible Automation Platform subscription](#)
[Red Hat Customer Portal](#)

Automation hub system requirements

With Automation hub you can discover and use new certified automation content from Red Hat Ansible and Certified Partners.

On Ansible automation hub, you can discover and manage Ansible Collections, which are supported automation content developed by Red Hat and its partners for use cases such as cloud automation, network automation, and security automation.

NOTE:

Private automation hub

If you install private automation hub from an internal address with a certificate that only encompasses the external address, this can result in an installation that 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](#).

High availability automation hub requirements

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

Required shared storage

Shared storage is required when installing more than one Automation hub with a `file` storage backend. The supported shared storage type for RPM-based installations is Network File System (NFS).

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 storage 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 installation program.

Install firewalld for HA hub deployment

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

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

1. Install the `firewalld` daemon:

```
$ dnf install firewalld
```

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

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

NOTE: For a list of supported services, use the `$ firewall-cmd --get-services` command

3. Reload to apply the configuration:

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

Event-Driven Ansible 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 for Event-Driven Ansible controller:

Requirement	Required
RAM	16 GB
CPUs	4
Local disk	• Hard drive must be 40 GB minimum with at least 20 GB available under /var. • Storage volume must be rated for a minimum baseline of 3000 IOPS. • If the cluster has many large projects or decision environment images, consider doubling the GB in /var to avoid disk space errors.

IMPORTANT:

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.

PostgreSQL requirements

Red Hat Ansible Automation Platform requires external (customer supported) databases to have International Components for Unicode (ICU) support. PostgreSQL user passwords are hashed with SCRAM-SHA-256 secure hashing algorithm before storing in the database.

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

NOTE:

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

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 [Tune the PostgreSQL database for optimal performance](#).

Related information

[PostgreSQL documentation](#)

Set up an external database

Use the following procedure to configure an external PostgreSQL compliant database for use with an Ansible Automation Platform component, for example automation controller, Event-Driven Ansible, automation hub, and platform gateway.

IMPORTANT:

- Red Hat Ansible Automation Platform 2.6 requires the external (customer supported) databases to have ICU support.
- When using an external database with Ansible Automation Platform, you must create and maintain that database. Ensure that you clear your external database when uninstalling Ansible Automation Platform.
- Red Hat Ansible Automation Platform 2.6 requires the external (customer supported) databases to have International Components for Unicode (ICU) support.
- During configuration of an external database, you must check the external database coverage. For more information, see [Red Hat Ansible Automation Platform Database Scope of Coverage](#).

Procedure

1. Connect to a PostgreSQL compliant database server with superuser privileges.

```
# psql -h <hostname> -U superuser -p 5432 -d postgres
<password_for_user_superuser>
```

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

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

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

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

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

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

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

6. Create the user, database, and password with the `createDB` or `administrator` role assigned to the user. For further information, see [Database Roles](#).
7. Run the installation program. If you are using a PostgreSQL database, the database is owned by the connecting user and must have a `createDB` or `administrator` role assigned to it.
8. Check that you can connect to the created database with the credentials provided in the inventory file.
9. Check the permission of the user. The user should have the `createDB` or `administrator` role.
10. After you create the PostgreSQL users and databases for each component, add the database credentials and host details in the inventory file under the `[all:vars]` group.

```
# Automation controller
pg_host=data.example.com
pg_database=<database name>
pg_port=<port_number>
pg_username=<set your own>
pg_password=<set your own>

# Platform gateway
automationgateway_pg_host=aap.example.org
automationgateway_pg_database=<set your own>
automationgateway_pg_port=<port_number>
automationgateway_pg_username=<set your own>
automationgateway_pg_password=<set your own>

# Automation hub
automationhub_pg_host=data.example.com
automationhub_pg_database=<database_name>
automationhub_pg_port=<port_number>
automationhub_pg_username=<username>
automationhub_pg_password=<password>

# Event-Driven Ansible
automationedacontroller_pg_host=data.example.com
automationedacontroller_pg_database=<database_name>
automationedacontroller_pg_port=<port_number>
automationedacontroller_pg_username=<username>
automationedacontroller_pg_password=<password>
```

Enable mutual TLS (mTLS) authentication

Enable mutual TLS authentication to secure PostgreSQL database connections with certificate-based verification. This protects against unauthorized access and man-in-the-middle attacks while meeting enterprise security and compliance requirements.

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

```
# Automation controller
pgclient_sslcert=/path/to/awx.cert
pgclient_sslkey=/path/to/awx.key
pg_sslmode=verify-full or verify-ca

# Platform gateway
automationgateway_pgclient_sslcert=/path/to/gateway.cert
automationgateway_pgclient_sslkey=/path/to/gateway.key
automationgateway_pg_sslmode=verify-full or verify-ca

# Automation hub
automationhub_pgclient_sslcert=/path/to/pulp.cert
automationhub_pgclient_sslkey=/path/to/pulp.key
automationhub_pg_sslmode=verify-full or verify-ca

# Event-Driven Ansible
automationedacontroller_pgclient_sslcert=/path/to/eda.cert
automationedacontroller_pgclient_sslkey=/path/to/eda.key
automationedacontroller_pg_sslmode=verify-full or verify-ca
```

Use custom TLS certificates

Configure the installer to use custom TLS certificates and keys for all services instead of the default self-signed ones by setting the required inventory variables for certificates and keys.

- To replace these with your own custom certificate and key, set the following inventory file variables:

```
aap_ca_cert_file=<path_to_ca_tls_certificate>
aap_ca_key_file=<path_to_ca_tls_key>
```

- If any of your certificates are signed by a custom Certificate Authority (CA), then you must specify the Certificate Authority's certificate by using the `custom_ca_cert` inventory file variable:

```
custom_ca_cert=<path_to_custom_ca_certificate>
```

NOTE:

If you have more than one custom CA certificate, combine them into a single file, then reference the combined certificate with the `custom_ca_cert` inventory file variable.

Receptor certificate considerations

When using a custom certificate for Receptor nodes, the certificate requires the `otherName` field specified in the Subject Alternative Name (SAN) of the certificate with the value `1.3.6.1.4.1.2312.19.1`.

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

Related information

[Above the mesh TLS](#)

Enable the hstore extension

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

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

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

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

Procedure

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

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

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

Example output with hstore available:

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

Example output with hstore not available:

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

3. On a RHEL based server, the hstore extension is included in the postgresql-contrib RPM package, which is not installed automatically when installing the PostgreSQL server RPM package.

To install the RPM package, use the following command:

```
dnf install postgresql-contrib
```

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

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

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

```

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

```

Benchmark storage performance for the Ansible Automation Platform PostgreSQL database

Use the Flexible I/O Tester (FIO) tool to verify that your storage system meets minimum Ansible Automation Platform PostgreSQL database requirements. FIO benchmarks read and write IOPS performance to help you evaluate storage capabilities.

Before you begin

- 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

- Run a random write test:


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

2. Run a random read test:

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

3. Review the results:

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

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

```
$ cat /tmp/fio_benchmark_read_iops.log

read_iops: (g=0): rw=randread, bs=(R) 4096B-4096B, (W) 4096B-4096B, (T)
4096B-4096B, ioengine=libaio, iodepth=64

[...]

iops          : min=50879, max=61603, avg=56221.33, stdev=679.97, samples=360

[...]
```

NOTE:

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

Install Ansible Automation Platform

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

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

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

Related information

[RPM enterprise topology](#)

[Inventory file variables](#)

[Red Hat Ansible Automation Platform components](#)

Edit the inventory file

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

For an RPM installation

Procedure

RPM installed package

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

Bundled installer

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

Online installer

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

For online installations

1. Open the `inventory` file with a text editor.
2. Edit the `inventory` file parameters to specify your installation scenario.
For containerized installation, see [Configuring the inventory file](#)

You can use one of the supported Installation scenario examples as the basis for your `inventory` file.

Related information

[Inventory file variables](#)

Tips for building inventories

When building inventories for Ansible automation, consider the following best practices to ensure efficient and effective management of your hosts.

- Ensure that group names are meaningful and unique.
- Group names are also case sensitive.
- Do not use spaces, hyphens, or preceding numbers (use `floor_19`, not `19th_floor`) in group names.
- Group hosts in your inventory logically according to their What, Where, and When:
 - What: Group hosts according to the topology, for example: `db`, `web`, `leaf`, `spine`.
 - Where: Group hosts by geographic location, for example: `data center`, `region`, `floor`, `building`.
 - When: Group hosts by stage, for example: `development`, `test`, `staging`, `production`.

Guidelines for configuring your inventory file

Inventory file recommendations provide critical variables for AAP installation. Following these rules for host identification (IP/FQDN) and credentials ensures a streamlined setup and prevents common installation failures.

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

Do not use `localhost`.

- The `[database]` group in the inventory file defines an Ansible Automation Platform managed database. When using an externally managed database, do not include the `[database]` group in your inventory file.
- `admin` is the default user ID for the initial log in to Ansible Automation Platform and cannot be changed in the inventory file.
- Use of special characters for `pg_password` is limited. The `!`, `#`, `0` and `@` characters are supported. Use of other special characters can cause the setup to fail.
- Enter your Red Hat Registry Service Account credentials in `registry_username` and `registry_password` to link to the Red Hat container registry.
- The inventory file variables `registry_username` and `registry_password` are only required if a non-bundle installer is used.

Related information

[Set up an external database](#)

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

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

```
[automationcontroller]
controller.example.com

[automationgateway]
gateway.example.com

[database]
data.example.com

[all:vars]
admin_password='<password>'
redis_mode=standalone
pg_host='data.example.com'
pg_port=5432
pg_database='awx'
```

```

pg_username='awx'
pg_password='<password>'
pg_sslmode='prefer' # set to 'verify-full' for client-side enforced SSL

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

# Automation Gateway configuration
automationgateway_admin_password=''

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

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

# The main automation gateway URL that clients will connect to (e.g. https://<load
balancer host>).

# If not specified, the first node in the [automationgateway] group will be used
when needed.

# automationgateway_main_url = ''

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

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

```

```
# postgres_ssl_cert=/path/to/pgsql.crt
# postgres_ssl_key=/path/to/pgsql.key
```

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

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

```
[automationcontroller]
controller.example.com

[automationhub]
automationhub.example.com

[automationgateway]
gateway.example.com

[database]
data.example.com

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

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

automationhub_admin_password= <PASSWORD>
```

```

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

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

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

# Automation Gateway configuration
automationgateway_admin_password=''

automationgateway_pg_host=''
automationgateway_pg_port=5432

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

```

```

# The main automation gateway URL that clients will connect to (e.g. https://<load
balancer host>).

# If not specified, the first node in the [automationgateway] group will be used
when needed.

# automationgateway_main_url = ''

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

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

```

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

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

IMPORTANT:

- This scenario requires a minimum of automation controller 2.4 for successful deployment of Event-Driven Ansible controller.
- Event-Driven Ansible controller must be installed on a separate server and cannot be installed on the same host as automation hub and automation controller.
- When an Event-Driven Ansible rulebook is activated under standard conditions, it uses approximately 250 MB of memory. However, the actual memory consumption can vary significantly based on the complexity of the 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 the maximum number of activations is based on the resource capacity.

```
[automationcontroller]
controller.example.com
```

```
[automationhub]
automationhub.example.com
```

```
[automationedacontroller]
automationedacontroller.example.com
```

```
[automationgateway]
gateway.example.com
```

```
[database]
data.example.com
```

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

```

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

# Automation hub configuration

automationhub_admin_password= <PASSWORD>

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

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

# Automation Event-Driven Ansible controller configuration

automationedacontroller_admin_password='<eda-password>'

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

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

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

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

# This install will deploy a TLS enabled Automation Hub.
# If for some reason this is not the behavior wanted one can

```

```

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

# Automation Gateway configuration
automationgateway_admin_password=''

automationgateway_pg_host=''
automationgateway_pg_port=5432

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

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

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

# Certificate and key to install in nginx for the web UI and API

```

```
# web_server_ssl_cert=/path/to/tower.cert
# web_server_ssl_key=/path/to/tower.key
# Server-side SSL settings for PostgreSQL (when we are installing it).
# postgres_use_ssl=False
# postgres_ssl_cert=/path/to/pgsql.crt
# postgres_ssl_key=/path/to/pgsql.key

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

Related information

[Ansible automation hub variables](#)

High availability automation hub

High availability increases reliability and scalability for automation hub by distributing workload across many nodes and eliminating single points of failure.

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 enable you to easily add or remove nodes to meet workload demands.

Enable a high availability (HA) deployment of automation hub on SELinux

Enable a high availability deployment of automation hub on SELinux by mounting `/var/lib/pulp` to an external NFS export. This ensures correct security contexts are applied to mount points.

Before you begin

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

NOTE:

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

NOTE:

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

Procedure

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

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

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

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

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

3. Run the reload systemd manager configuration command:

```
$ systemctl daemon-reload
```

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

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

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

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

6. Run the mount command:

```
$ mount -a
```

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

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

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

Next steps

1. [Apply the appropriate SELinux context.](#)
2. [Configure the `pulpcore.service`.](#)

Related information

[SELinux Requirements on the Pulp Project documentation](#)

[Pulp Filesystem Layout](#)

Configure `pulpcore.service`

You can configure the `pulp` service to ensure that automation hub services start only after the network and the mounting of the remote mount points.

Procedure

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

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

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

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

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

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

4. Enable `remote-fs.target`:

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

5. Reboot the system:

```
$ systemctl reboot
```

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

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

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

Apply the SELinux context

By applying the necessary SELinux context to the Pulp directories you ensure proper file access permissions and security policy compliance. They are essential for enabling the high availability (HA) deployment of automation hub on SELinux.

Procedure

1. Shut down the Pulp service:

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

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

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

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

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

4. Modify the `/etc/fstab` file as follows:

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

5. Run the mount command:

```
$ mount -a
```

Configure content signing on private automation hub

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

Before you begin

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

Procedure

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

NOTE:

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

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

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

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

Example: The following script produces signatures for content:


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

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

ADMIN_ID="$PULP_SIGNING_KEY_FINGERPRINT"
PASSWORD="password"

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

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

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

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

```
[all:vars]

.
.
.

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

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

Add a safe plugin variable to Event-Driven Ansible controller

When using the `redhat.insights_eda` plugin for rulebook activations, add a safe plugin variable to a platform directory to ensure a secure connection and correct port mapping displays.

Procedure

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

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

NOTE:

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

Create a registry service account

Create a registry service account on the Red Hat Customer Portal to generate a token and username. Use these to set the `registry_username` and `registry_password` variables required for online non-bundled installations.

Procedure

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

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

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

Configure Redis

Ansible Automation Platform offers a centralized Redis instance in both `standalone` and `clustered` topologies.

In RPM deployments, the Redis mode is set to `cluster` by default. You can change this setting in the inventory file `[all:vars]` section as in the following example:

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

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

redis_mode=cluster
```

For more information about Redis, see [Caching and queueing system](#) in *Planning your installation*.

Run the setup script

Run the installation program setup script after updating the inventory file with all the required parameters to begin the installation and configuration of the platform.

RPM installer

- Run the `setup.sh` script

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

NOTE:

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

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

Result

Installation of Red Hat Ansible Automation Platform will begin.

Verify your installation of Ansible Automation Platform

Perform the verification step for your Red Hat Ansible Automation Platform deployment by successfully logging into the platform. A successful login confirms that the installation is complete, ensuring your environment is ready for operational use.

- Log in to Red Hat Ansible Automation Platform.

IMPORTANT:

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

Related information

[Get started as a platform administrator](#)

Install in a disconnected environment

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

Ansible Automation Platform installation on disconnected RHEL

For a disconnected installation, use the setup bundle with the automation controller's installer-managed database. This bundle simplifies the process by including all necessary Red Hat package managers (RPMs) and default execution environment (EE) images within the package.

System requirements for disconnected installation

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

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

- A subscription manifest that you can upload to the platform.

For more information, see **Manage Ansible Automation Platform subscriptions, updates, and support**.

- The Ansible Automation Platform setup bundle at the **Customer Portal** is downloaded.
- The DNS records for the automation controller and private automation hub servers are created.

Obtain and configure RPM source dependencies

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

- Reposync
- The RHEL Binary DVD

NOTE:

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

Related information

[Customer portal](#)

[Installing Satellite Server in a disconnected network environment](#)

[Manage DNS records](#)

[Manage Ansible Automation Platform subscriptions, updates, and support](#)

[Red Hat Enterprise Linux Life Cycle](#)

[System requirements](#)

[Inventory file variables](#)

Synchronize the RPM repositories by using reposync

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

When downloading RPM, ensure you use the applicable distro.

Procedure

1. Attach the BaseOS and AppStream required repositories:

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

2. Perform the reposync:

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

- a. Use reposync with `--download-metadata` and without `--newest-only`. See [RHEL 8 Reposync](#).
 - If you are not using `--newest-only`, the repos downloaded may take an extended amount of time to sync due to the large number of GB.
 - If you are using `--newest-only`, the repos downloaded may take an extended amount of time to sync due to the large number of GB.

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

3. Move the repositories to your disconnected network.

Create a new web server to host repositories

Create and configure a dedicated web server to host repositories for a disconnected environment. This procedure ensures local access to synchronized RPM packages and enables you to streamline content installation and updates on isolated systems.

Procedure

1. Install prerequisites:

```
$ sudo dnf install httpd
```

2. Configure httpd to serve the repo directory:

```

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

DocumentRoot '/path/to/repos'

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

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

```

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

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

4. Configure SELinux:

```

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

```

5. Enable httpd:

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

6. Open firewall:

```

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

```

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


```

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

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

```

Access RPM repositories from a locally mounted DVD

In disconnected or air-gapped environments, you can install Ansible Automation Platform by using packages from a locally mounted RHEL DVD or ISO image. Learn how to mount the media and configure yum repositories to access BaseOS and AppStream packages for offline installation.

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

Procedure

1. Mount DVD or ISO:

a. DVD

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

b. ISO

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

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

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

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

3. Import the gpg key:

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

NOTE:

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

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

Related information

[Need to set up yum repository for locally-mounted DVD on Red Hat Enterprise Linux 9](#)

Download and install the Ansible Automation Platform setup bundle

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

Procedure

1. Download the Ansible Automation Platform setup bundle package by navigating to the [Red Hat Ansible Automation Platform download](#) page and clicking **Download Now** for the Ansible Automation Platform 2.6 Setup Bundle.
2. On control node, untar the bundle:

```
$ tar xvf \
    ansible-automation-platform-setup-bundle-2.6-1.tar.gz
$ cd ansible-automation-platform-setup-bundle-2.6-1
```

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

Complete post installation tasks

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

Before your first login, you must add your subscription information to the platform. To obtain your subscription information in uploadable form, see [Obtaining a manifest file](#).

Once you have obtained your subscription manifest, see [Add your subscription](#) for instructions on how to upload your subscription information.

Related information

[Ansible Automation Platform overview](#)

[Red Hat Certified, validated, and Ansible Galaxy content in automation hub](#)

[Define, create, and build execution environments](#)

Log in after installation

After you install Ansible Automation Platform, you must log in. Learn how to log in to the Ansible Automation Platform using the default admin credentials and how to edit user details by using the More Actions menu.

Procedure

1. With the login information provided after your installation completed, open a web browser and log in to the Ansible Automation Platform by navigating to its server URL at: `https://<CONTROLLER_SERVER_NAME>/`
2. Use the credentials specified during the installation process to login:
 - The default username is `admin`.
 - The password for `admin` is the value specified during setup.

3. Click the **More Actions** icon next to the required user.
4. Click **Edit**.
5. Edit the required details and click **Save**.

Import your inventory file from source control

With automation controller you can select an inventory file from source control, rather than creating one from scratch.

The files are non-editable, and as inventories are updated at the source, the inventories within the projects are also updated accordingly, including the `group_vars` and `host_vars` files or directory associated with them. SCM types can consume both inventory files and scripts. Both inventory files and custom inventory types use scripts.

Imported hosts have a description of *imported* by default. This can be overridden by setting the `_awx_description` variable on a given host. For example, if importing from a sourced `.ini` file, you can add the following host variables:

```
[main]
127.0.0.1 _awx_description="my host 1"
127.0.0.2 _awx_description="my host 2"
```

Similarly, group descriptions also default to *imported*, but can also be overridden by `_awx_description`.

Related information

[Export old inventory scripts](#)

Inventory file fields for source control

You can use a source control management (SCM) system, such as Git, to manage your inventory files in automation controller. You can use an SCM system to version control your inventory files and manage them in a collaborative manner.

The source fields used are:

- `source_project` : the project to use.
- `source_path` : the relative path inside the project indicating a directory or a file. If left blank, "" is still a relative path indicating the root directory of the project.

- `source_vars` : if set on a "file" type inventory source then they are passed to the environment variables when running.

Additionally:

- An update of the project automatically triggers an inventory update where it is used.
- An update of the project is scheduled immediately after creation of the inventory source.
- Neither inventory nor project updates are blocked while a related job is running.
- In cases where you have a large project (around 10 GB), disk space on `/tmp` can be an issue.

You can specify a location manually in the automation controller UI from the **Add source** page of an inventory. Refer to [Adding a source](#) for instructions on creating an inventory source.

When you update a project, refresh the listing to use the latest source control management (SCM) information. If no inventory sources use a project as an SCM inventory source, then the inventory listing might not be refreshed on update.

For inventories with SCM sources, the job **Details** page for inventory updates displays a status indicator for the project update and the name of the project.

The status indicator links to the project update job.

The project name links to the project.

You can perform an inventory update while a related job is running.

Supported file syntax

Automation controller uses the `ansible-inventory` module from Ansible to process inventory files, and supports all valid inventory syntax that automation controller requires.

IMPORTANT:

You do not need to write inventory scripts in Python. You can enter any executable file in the source field and must run `chmod +x` for that file and check it into Git.

The following is a working example of JSON output that automation controller can read for the import:

```
{
  "_meta": {
    "hostvars": {
      "host1": {
        "fly_rod": true
      }
    }
  }
}
```

```
{,
  "all": {
    "children": [
      "groupA",
      "ungrouped"
    ]
  },
  "groupA": {
    "hosts": [
      "host1",
      "host10",
      "host11",
      "host12",
      "host13",
      "host14",
      "host15",
      "host16",
      "host17",
      "host18",
      "host19",
      "host2",
      "host20",
      "host21",
      "host22",
      "host23",
      "host24",
      "host25",
      "host3",
      "host4",
      "host5",
      "host6",
      "host7",
      "host8",
      "host9"
    ]
  }
}
```

```
}
```

Related information

[test-playbooks/inventories](#)

[inventories/changes.py](#)

[How to migrate inventory scripts from Red Hat Ansible tower to Red Hat Ansible Automation Platform?](#)

Activate your subscription

Red Hat Ansible Automation Platform uses available subscriptions or a subscription manifest to allow the use of Ansible Automation Platform.

To obtain a subscription, you can do either of the following:

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

Related information

[Activate with credentials](#)

[Activate with a manifest file](#)

Activate with credentials

Activate your Ansible Automation Platform subscription at first launch using Red Hat service account credentials or a personal login. This retrieves and imports the license, granting access to Red Hat content and entitlement services.

NOTE:

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

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

Procedure

1. Log in to Red Hat Ansible Automation Platform.
2. Select the **Service Account** tab in the subscription wizard.
3. Enter your **Client ID** and **Client secret**.
4. Select your subscription from the **Subscription** list.

NOTE:

You can also enter your Satellite username and password in the **Satellite** tab if your cluster nodes are registered to Satellite through Subscription Manager.

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

Result

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

Hosts automated

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

Hosts imported

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

Hosts remaining

Total host count minus hosts automated

Activate with a manifest file

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

Before you begin

You must have a Red Hat subscription manifest file exported from the Red Hat Customer Portal. For more information, see [Obtaining a manifest file](#).

NOTE:

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

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

Procedure

1. Log in to Red Hat Ansible Automation Platform.
 - a. If you are not immediately taken to the subscription wizard, go to **Settings > Subscription**.
2. Select the **Subscription manifest** tab.
3. Click **Browse** and select your manifest file.
4. Review the End User License Agreement and select **I agree to the End User License Agreement**.
5. Click **Finish**.

NOTE:

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

Result

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

Hosts automated

Host count automated by the job, which uses the subscription count

Hosts imported

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

Hosts remaining

Total host count minus hosts automated

Next steps

- You can return to the subscription wizard by selecting **Settings > Subscription** from the navigation panel and clicking **Edit subscription**.

Keep subscriptions for managed hosts in compliance

To ensure that your automation controller installation remains compliant with your Red Hat subscription, automation controller provides a way to monitor your subscription status and usage.

Your subscription has two possible statuses:

- **Compliant:** Indicates that your subscription is appropriate for the number of hosts that you have automated within your subscription count.
- **Out of compliance:** Indicates that you have exceeded the number of hosts in your subscription.

Compliance is computed as follows:

```
managed > manifest_limit    => non-compliant
managed =< manifest_limit    => compliant
```

Where: `managed` is the number of unique managed hosts without deletions, and `manifest_limit` is the number of managed hosts in the subscription manifest.

Other important information displayed are:

- **Hosts automated:** The number of hosts automated by the job, which consumes the license count.
- **Hosts imported:** The number of hosts considering unique host names across all inventory sources. This number does not impact hosts remaining.
- **Hosts remaining:** The number of hosts minus the number of hosts automated.
- **Hosts deleted:** The number of hosts that were deleted, freeing the license capacity.
- **Active hosts previously deleted:** The number of hosts now active that were previously deleted.

For example, if you have a subscription capacity of 10 hosts:

- Starting with 9 hosts, 2 hosts were added and 3 hosts were deleted, you now have 8 hosts (compliant).
- 3 hosts were automated again, now you have 11 hosts, which puts you over the subscription limit of 10 (noncompliant).
- If you delete hosts, refresh the subscription details to see the change in count and status.

View hosts automated in the user interface

Learn how to view and manage the hosts that have been automated in automation controller.

Procedure

1. In the navigation panel, select **Automation Analytics > Host Metrics** to view the activity associated with hosts, such as those that have been automated and deleted.

Each unique hostname is listed and sorted by the user's preference.

Host Metrics

☐	Hostname	First automated	Last automated	Automation	Deleted
☐	host-1	4/18/2023, 8:08:41 AM	4/18/2023, 8:08:41 AM	1	0
☐	host-2	4/18/2023, 8:08:41 AM	4/18/2023, 8:08:41 AM	1	0
☐	host-3	4/18/2023, 8:08:41 AM	4/18/2023, 8:08:41 AM	1	0
☐	host-4	4/18/2023, 8:08:41 AM	4/18/2023, 8:08:41 AM	1	0
☐	host-5	4/18/2023, 8:08:41 AM	4/18/2023, 8:08:41 AM	1	0

NOTE:

A scheduled task automatically updates these values on a weekly basis and deletes jobs with hosts that were last automated more than a year ago.

2. Delete unnecessary hosts directly from the Host Metrics view by selecting the required hosts and clicking **Delete**.

These are soft-deleted, meaning their records are not removed, but are not being used and thereby not counted towards your subscription.

Related information

[Keeping your subscription in compliance](#)

View hosts automated in the CLI

Automation controller provides a way to generate a CSV output of the host metric data and host metric summary through the *Command Line Interface* (CLI). You can also soft delete hosts in bulk through the API.

awx-manage utility

To collect and manage host metric data and related cluster information in Ansible Automation Platform, use the `awx-manage` utility.

Procedure

1. The `awx-manage` utility supports the following options:

```
awx-manage host_metric --csv
```

2. This command produces host metric data in CSV format, a host metrics summary file, and a cluster info file.
3. To package all the files into a single tar file for distribution and sharing, use:

```
awx-manage host_metric --tarball
```

4. To specify the number of rows (`<n>`) to output to each file:

```
awx-manage host_metric --tarball --rows_per_file <n>
```

5. Automation Analytics then receives and uses the JSON file.

Related information

[Usage reporting with metrics-utility](#)

Deleting Hosts automated using API endpoint

You can soft delete hosts that have been automated in automation controller using the host metric API endpoint.

The API lists only non-deleted records and are sortable by `last_automation` and `used_in_inventories` columns.

To soft delete hosts, use:

```
api/v2/host_metric <n> DELETE
```

INSTALL

A monthly scheduled task automatically deletes jobs that uses hosts from the Host Metric table that were last automated more than a year ago.

Red Hat product documentation legal notices

Copyright © Red Hat

Except as otherwise noted below, the text of and illustrations in this documentation are licensed by Red Hat under the [Creative Commons Attribution–Share Alike 3.0 Unported license](#). If you distribute this document or an adaptation of it, you must provide the URL for the original version. Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Linux® is the registered trademark of Linus Torvalds in the United States and other countries.

XFS is a trademark or registered trademark of Hewlett Packard Enterprise Development LP or its subsidiaries in the United States and other countries.

The OpenStack® Word Mark and OpenStack logo are trademarks or registered trademarks of the Linux Foundation, used under license.

All other trademarks are the property of their respective owners.

GNU GENERAL PUBLIC LICENSE

Version 3, 29 June 2007

Copyright © 2007 Free Software Foundation, Inc. link:<https://fsf.org/>. Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program—to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If

such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS

0. Definitions. "This License" refers to version 3 of the GNU General Public License.

"Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

A "covered work" means either the unmodified Program or a work based on the Program.

To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code. The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component,

or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions. All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law. No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies. You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions. You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

- a) The work must carry prominent notices stating that you modified it, and giving a relevant date.
- b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".
- c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.
- d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so. A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms. You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

- a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.
- b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

- c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.
- d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.
- e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A “User Product” is either (1) a “consumer product”, which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, “normally used” refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

“Installation Information” for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms. “Additional permissions” are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

- a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or
- b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or
- c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or
- d) Limiting the use for publicity purposes of names of licensors or authors of the material; or
- e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or
- f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered “further restrictions” within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination. You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies. You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients. Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An “entity transaction” is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party’s predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents. A “contributor” is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor’s “contributor version”.

A contributor’s “essential patent claims” are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, “control” includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor’s essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a “patent license” is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To “grant” such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. “Knowingly relying” means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient’s use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is “discriminatory” if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom. If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License. Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License. The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty. THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES

SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16. If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the “copyright” line and a pointer to where the full notice is found.

```
<one line to give the program's name and a brief idea of what it does.>
```

```
Copyright (C) <year> <name of author>
```

```
This program is free software: you can redistribute it and/or modify  
it under the terms of the GNU General Public License as published by  
the Free Software Foundation, either version 3 of the License, or  
(at your option) any later version.
```

```
This program is distributed in the hope that it will be useful,  
but WITHOUT ANY WARRANTY; without even the implied warranty of  
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the  
GNU General Public License for more details.
```

```
You should have received a copy of the GNU General Public License  
along with this program. If not, see <https://www.gnu.org/licenses/>.
```

Also add information on how to contact you by electronic and paper mail.

If the program does terminal interaction, make it output a short notice like this when it starts in an interactive mode:


```
<program> Copyright (C) <year> <name of author>  
  
This program comes with ABSOLUTELY NO WARRANTY; for details type `show w'.  
  
This is free software, and you are welcome to redistribute it  
under certain conditions; type `show c' for details.
```

The hypothetical commands `show w' and `show c' should show the appropriate parts of the General Public License. Of course, your program's commands might be different; for a GUI interface, you would use an "about box".

You should also get your employer (if you work as a programmer) or school, if any, to sign a "copyright disclaimer" for the program, if necessary. For more information on this, and how to apply and follow the GNU GPL, see link:<https://www.gnu.org/licenses/>.

The GNU General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License. But first, please read link:<https://www.gnu.org/licenses/why-not-lgpl.html>.

Apache license

Version 2.0, January 2004

<http://www.apache.org/licenses/>

Terms and Conditions for use, reproduction, and distribution

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, **"control"** means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or **"Your"**) shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, **"submitted"** means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the

Licensors for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as **"Not a Contribution."**

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

1. You must give any other recipients of the Work or Derivative Works a copy of this License; and
2. You must cause any modified files to carry prominent notices stating that You changed the files; and
3. You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and
4. If the Work includes a **"NOTICE"** text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications,

or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS



Copyright 2026. All rights reserved.

www.redhat.com