



# Red Hat Ansible Automation Platform 2.4

## Managing content in automation hub

Create and manage collections, content and repositories in automation hub



## Red Hat Ansible Automation Platform 2.4 Managing content in automation hub

---

Create and manage collections, content and repositories in automation hub

## Legal Notice

Copyright © 2024 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, 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.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux<sup>®</sup> is the registered trademark of Linus Torvalds in the United States and other countries.

Java<sup>®</sup> is a registered trademark of Oracle and/or its affiliates.

XFS<sup>®</sup> is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL<sup>®</sup> is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js<sup>®</sup> is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack<sup>®</sup> Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

## Abstract

This guide shows you how to create, edit, delete, and move content in automation hub.

## Table of Contents

<b>MAKING OPEN SOURCE MORE INCLUSIVE</b> .....	<b>4</b>
<b>PROVIDING FEEDBACK ON RED HAT DOCUMENTATION</b> .....	<b>5</b>
<b>CHAPTER 1. RED HAT CERTIFIED, VALIDATED, AND ANSIBLE GALAXY CONTENT IN AUTOMATION HUB</b>	<b>6</b>
Why certify Ansible collections?	6
How do I get a collection certified?	6
How does the joint support agreement on Certified Collections work?	6
Can I create and certify a collection containing only Ansible Roles?	6
<b>1.1. SYNCHRONIZING ANSIBLE CONTENT COLLECTIONS IN AUTOMATION HUB</b>	<b>7</b>
1.1.1. Explanation of Red Hat Ansible Certified Content Collections synclists	7
1.1.2. Creating a synclist of Red Hat Ansible Certified Content Collections	7
<b>1.2. CONFIGURING ANSIBLE AUTOMATION HUB REMOTE REPOSITORIES TO SYNCHRONIZE CONTENT</b>	<b>8</b>
What's the difference between Ansible Galaxy and Ansible automation hub?	8
How do I request a namespace on Ansible Galaxy?	9
Are there any restrictions for Ansible Galaxy namespace naming?	9
1.2.1. Reasons to create remote configurations	9
1.2.2. Retrieving the Sync URL and API token for your Red Hat Certified Collection	9
1.2.3. Configuring the rh-certified remote repository and synchronizing Red Hat Ansible Certified Content Collection	9
1.2.4. Configuring the community remote repository and syncing Ansible Galaxy collections	10
1.2.5. Configuring proxy settings	11
<b>1.3. COLLECTIONS AND CONTENT SIGNING IN PRIVATE AUTOMATION HUB</b>	<b>12</b>
1.3.1. Configuring content signing on private automation hub	12
1.3.2. Using content signing services in private automation hub	13
1.3.3. Downloading signature public keys	14
1.3.4. Configuring Ansible-Galaxy CLI to verify collections	14
Are there any recommendations for collection naming?	15
How do I get a namespace on Ansible automation hub?	15
<b>1.4. ANSIBLE VALIDATED CONTENT</b>	<b>16</b>
1.4.1. Configuring validated collections with the installer	16
1.4.2. Installing validated content using the tarball	16
<b>CHAPTER 2. MANAGING COLLECTIONS IN AUTOMATION HUB</b> .....	<b>18</b>
<b>2.1. USING NAMESPACES TO MANAGE COLLECTIONS IN AUTOMATION HUB</b>	<b>18</b>
2.1.1. Creating a new group for content curators	18
2.1.2. Creating a namespace	19
2.1.3. Adding additional information and resources to a namespace	19
2.1.4. Uploading collections to your namespaces	20
2.1.5. Reviewing your namespace import logs	20
2.1.6. Deleting a namespace	21
<b>2.2. MANAGING THE PUBLICATION PROCESS OF INTERNAL COLLECTIONS IN AUTOMATION HUB</b>	<b>22</b>
2.2.1. About Approval	22
2.2.2. Approving collections for internal publication	22
2.2.3. Rejecting collections uploaded for review	23
<b>2.3. REPOSITORY MANAGEMENT WITH AUTOMATION HUB</b>	<b>23</b>
2.3.1. Types of repositories in automation hub	23
2.3.2. Approval pipeline for custom repositories in automation hub	23
2.3.3. Role based access control to restrict access to custom repositories	24
2.3.4. Creating a custom repository in automation hub	24
2.3.5. Providing access to a custom automation hub repository	25
2.3.6. Adding collections to an automation hub repository	26

2.3.7. Revert to a different automation hub repository version	26
2.3.8. Managing remote configurations in automation hub	26
2.3.8.1. Creating a remote configuration in automation hub	26
2.3.8.2. Providing access to a remote configuration	28
2.3.9. Synchronizing repositories in automation hub	28
2.3.10. Exporting and importing collections in automation hub	28
2.3.10.1. Exporting an automation content collection in automation hub	29
2.3.10.2. Importing an automation content collection in automation hub	29
<b>CHAPTER 3. MANAGE CONTAINERS IN PRIVATE AUTOMATION HUB</b> .....	<b>30</b>
3.1. MANAGE YOUR PRIVATE AUTOMATION HUB CONTAINER REGISTRY	30
3.1.1. Container registries	30
3.2. CONFIGURING USER ACCESS FOR CONTAINER REPOSITORIES IN PRIVATE AUTOMATION HUB	30
3.2.1. Container registry group permissions	30
3.2.2. Creating a new group in private automation hub	31
3.2.3. Assigning permissions to groups	31
3.2.4. Adding users to existing groups	31
3.3. POPULATING YOUR PRIVATE AUTOMATION HUB CONTAINER REGISTRY	31
3.3.1. Pulling images for use in automation hub	32
3.3.2. Tagging images for use in automation hub	33
3.3.3. Pushing a container image to private automation hub	33
3.4. SETTING UP YOUR CONTAINER REPOSITORY	34
3.4.1. Prerequisites to setting up your container registry	34
3.4.2. Adding a README to your container repository	34
3.4.3. Providing access to your container repository	35
3.4.4. Tagging container images	35
3.4.5. Creating a credential in automation controller	36
3.5. PULLING IMAGES FROM A CONTAINER REPOSITORY	36
3.5.1. Pulling an image	36
3.5.2. Syncing images from a container repository	37
3.6. WORKING WITH SIGNED CONTAINERS	38
3.6.1. Deploying your system for container signing	38
3.6.2. Adding containers remotely to automation hub	39
3.6.3. Adding an execution environment	39
3.6.4. Pushing container images from your local environment	40
3.6.5. Policies with signed images	41
3.6.6. Using podman to ensure an image is signed by a specific signature	41
3.6.7. Configuring the client to verify signatures	41
3.7. DELETING A CONTAINER REPOSITORY	43



## MAKING OPEN SOURCE MORE INCLUSIVE

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



## PROVIDING FEEDBACK ON RED HAT DOCUMENTATION

If you have a suggestion to improve this documentation, or find an error, please contact technical support at <https://access.redhat.com> to create an issue on the Ansible Automation Platform Jira project using the **docs-product** component.

# CHAPTER 1. RED HAT CERTIFIED, VALIDATED, AND ANSIBLE GALAXY CONTENT IN AUTOMATION HUB

Ansible Certified Content Collections are included in your subscription to Red Hat Ansible Automation Platform. Red Hat Ansible content includes two types of content: Ansible Certified Content Collections and Ansible validated content. Using Ansible automation hub, you can access and curate a unique set of collections from all forms of Ansible content.

Red Hat Ansible content contains two types of content:

- Ansible Certified Content Collections
- Ansible validated content collections

Ansible validated collections are available in your private automation hub through the Platform Installer. When you download Red Hat Ansible Automation Platform with the bundled installer, validated content is pre-populated into the private automation hub by default, but only if you enable the private automation hub as part of the inventory.

If you are not using the bundle installer, you can use a Red Hat supplied Ansible playbook to install validated content. For further information, see [Ansible validated content](#).

You can update these collections manually by downloading their packages.

## Why certify Ansible collections?

The Ansible certification program enables a shared statement of support for Red Hat Ansible Certified Content between Red Hat and the ecosystem partner. An end customer, experiencing trouble with Ansible and certified partner content, can open a support ticket, for example, a request for information, or a problem with Red Hat, and expect the ticket to be resolved by Red Hat and the ecosystem partner.

Red Hat offers go-to-market benefits for Certified Partners to grow market awareness, generate demand, and sell collaboratively.

Red Hat Ansible Certified Content Collections are distributed through Ansible automation hub (subscription required), a centralized repository for jointly supported Ansible Content. As a certified partner, publishing collections to Ansible automation hub provides end customers the power to manage how trusted automation content is used in their production environment with a well-known support life cycle.

For more information about getting started with certifying a solution, see [Red Hat Partner Connect](#).

## How do I get a collection certified?

For instructions on certifying your collection, see the Ansible certification policy guide on [Red Hat Partner Connect](#).

## How does the joint support agreement on Certified Collections work?

If a customer raises an issue with the Red Hat support team about a certified collection, Red Hat support assesses the issue and checks whether the problem exists within Ansible or Ansible usage. They also check whether the issue is with a certified collection. If there is a problem with the certified collection, support teams transfer the issue to the vendor owner of the certified collection through an agreed upon tool such as TSANet.

## Can I create and certify a collection containing only Ansible Roles?

You can create and certify collections that contain only roles. Current testing requirements are focused on collections containing modules, and additional resources are currently in progress for testing collections only containing roles. Contact [ansiblepartners@redhat.com](mailto:ansiblepartners@redhat.com) for more information.

## 1.1. SYNCHRONIZING ANSIBLE CONTENT COLLECTIONS IN AUTOMATION HUB



### IMPORTANT

As of the 2.4 release you can still synchronize content, but synclists are deprecated, and will be removed in a future version.

To synchronize content, you can now upload a manually-created requirements file from the rh-certified remote.

Remotes are configurations that enable you to synchronize content to your custom repositories from an external collection source.

You can use Ansible automation hub to distribute the relevant Red Hat Ansible Certified Content Collections to your users by creating synclists or a requirements file. For more information about using requirements files, see [Install multiple collections with a requirements file](#) in the *Using Ansible collections* guide.

### 1.1.1. Explanation of Red Hat Ansible Certified Content Collections synclists

A synclist is a curated group of Red Hat Certified Collections that is assembled by your organization administrator. It synchronizes with your local Ansible automation hub. Use synclists to manage only the content that you want and exclude unnecessary collections. Design and manage your synclist from the content available as part of Red Hat content on console.redhat.com

Each synclist has its own unique repository URL that you can use to designate as a remote source for content in automation hub. You securely access each synclist by using an API token.

### 1.1.2. Creating a synclist of Red Hat Ansible Certified Content Collections

You can create a synclist of curated Red Hat Ansible Certified Content in Ansible automation hub on console.redhat.com. Your synclist repository is located on the automation hub navigation panel under **Collection** → **Repositories**, which is updated whenever you manage content within Ansible Certified Content Collections.

All Ansible Certified Content Collections are included by default in your initial organization synclist.

#### Prerequisites

- You have a valid Ansible Automation Platform subscription.
- You have Organization Administrator permissions for console.redhat.com.
- The following domain names are part of either the firewall or the proxy's allowlist. They are required for successful connection and download of collections from automation hub or Galaxy server:
  - **galaxy.ansible.com**
  - **cloud.redhat.com**
  - **console.redhat.com**
  - **sso.redhat.com**

- Ansible automation hub resources are stored in Amazon Simple Storage. The following domain names must be in the allow list:
  - **automation-hub-prd.s3.us-east-2.amazonaws.com**
  - **ansible-galaxy.s3.amazonaws.com**
- SSL inspection is disabled either when using self signed certificates or for the Red Hat domains.

## Procedure

1. Log in to **console.redhat.com**.
2. Navigate to **Automation Hub → Collections**.
3. Set the toggle switch on each collection to exclude or include it on your synclist.
4. To initiate the remote repository synchronization, navigate to automation hub and select **Collection → Repositories**.
5. Click the **More Actions** icon **⋮** and select **Sync** to initiate the remote repository synchronization to your private automation hub.
6. Optional: If your remote repository is already configured, update the collections content that you made available to local users by manually synchronizing Red Hat Ansible Certified Content Collections to your private automation hub.

## 1.2. CONFIGURING ANSIBLE AUTOMATION HUB REMOTE REPOSITORIES TO SYNCHRONIZE CONTENT

Use remote configurations to configure your private automation hub to synchronize with Ansible Certified Content Collections hosted on **console.redhat.com** or with your collections in Ansible Galaxy.



### IMPORTANT

As of the 2.4 release you can still synchronize content, but synclists are deprecated, and will be removed in a future version.

To synchronize content, you can now upload a manually-created requirements file from the rh-certified remote.

Remotes are configurations that allow you to synchronize content to your custom repositories from an external collection source.

### What's the difference between Ansible Galaxy and Ansible automation hub?

Collections published to Ansible Galaxy are the latest content published by the Ansible community and have no joint support claims associated with them. Ansible Galaxy is the recommended frontend directory for the Ansible community accessing content.

Collections published to Ansible automation hub are targeted for joint customers of Red Hat and selected partners. Customers need an Ansible subscription to access and download collections on Ansible automation hub. A certified collection means that Red Hat and partners have a strategic relationship in place and are ready to support joint customers, and may have had additional testing and validation done against them.

## How do I request a namespace on Ansible Galaxy?

To request a namespace through an Ansible Galaxy GitHub issue, follow these steps:

- Send an email to [ansiblepartners@redhat.com](mailto:ansiblepartners@redhat.com)
- Include the GitHub username used to sign up on Ansible Galaxy.

You must have logged in at least once for the system to validate.

After users are added as administrators of the namespace, you can use the self-serve process to add more administrators.

## Are there any restrictions for Ansible Galaxy namespace naming?

Collection namespaces must follow python module name convention. This means collections should have short, all lowercase names. You can use underscores in the collection name if it improves readability.

### 1.2.1. Reasons to create remote configurations

Each remote configuration located in **Collections** → **Remotes** provides information for both the **community** and **rh-certified** repository about when the repository was **last updated**. You can add new content to Ansible automation hub at any time using the **Edit** and **Sync** features included on the **Collection** → **Repositories** page.

### 1.2.2. Retrieving the Sync URL and API token for your Red Hat Certified Collection

You can synchronize Ansible Certified Content Collections curated by your organization from **console.redhat.com** to your private automation hub. The API token is a secret token used to protect your content.

#### Prerequisites

- You have organization administrator permissions to create the synclist on console.redhat.com.

#### Procedure

1. Log in to **console.redhat.com** as an organization administrator.
2. Navigate to **Automation Hub** → **Connect to Hub**.
3. Under **Offline token**, click **Load token**.
4. Click **Copy to clipboard** to copy the API token.
5. Paste the API token into a file and store in a secure location.

### 1.2.3. Configuring the rh-certified remote repository and synchronizing Red Hat Ansible Certified Content Collection

You can edit the **rh-certified** remote repository to synchronize collections from automation hub hosted on console.redhat.com to your private automation hub. By default, your private automation hub **rh-certified** repository includes the URL for the entire group of Ansible Certified Content Collections.

To use only those collections specified by your organization, a private automation hub administrator can upload manually-created requirements files from the **rh-certified** remote.



For more information about using requirements files, see [Install multiple collections with a requirements file](#) in the *Using Ansible collections* guide.

If you have collections **A**, **B**, and **C** in your requirements file, and a new collection **X** is added to console.redhat.com that you want to use, you must add **X** to your requirements file for private automation hub to synchronize it.

### Prerequisites

- You have valid **Modify Ansible repo content** permissions. For more information on permissions, see [Configuring user access for your private automation hub](#).
- You have retrieved the Sync URL and API Token from the automation hub hosted service on console.redhat.com.
- You have configured access to port 443. This is required for synchronizing certified collections. For more information, see the automation hub table in the [Network ports and protocols](#) chapter of the Red Hat Ansible Automation Platform Planning Guide.

### Procedure

1. Log in to your private automation hub.
2. From the navigation panel, select **Collections** → **Remotes**.
3. In the **rh-certified** remote repository, click the **More Actions** icon  and click **Edit**.
4. In the **URL** field, paste the **Sync URL**.
5. In the **Token** field, paste the token you acquired from console.redhat.com.
6. Click **Save**.  
You can now synchronize collections between your organization synclist on console.redhat.com and your private automation hub.
7. Click the **More Actions** icon  and select **Sync**.

### Verification

The **Sync status** notification updates to notify you that the Red Hat Certified Content Collections synchronization is complete.

- Select **Red Hat Certified** from the collections content drop-down list to confirm that your collections content has synchronized successfully.

## 1.2.4. Configuring the community remote repository and syncing Ansible Galaxy collections

You can edit the **community** remote repository to synchronize chosen collections from Ansible Galaxy to your private automation hub. By default, your private automation hub community repository directs to **galaxy.ansible.com/api/**.



### Prerequisites

- You have **Modify Ansible repo content** permissions. For more information on permissions, see [Configuring user access for your private automation hub](#) .
- You have a **requirements.yml** file that identifies those collections to synchronize from Ansible Galaxy as in the following example:

#### Requirements.yml example

```
collections:
  # Install a collection from Ansible Galaxy.
  - name: community.aws
    version: 5.2.0
    source: https://galaxy.ansible.com
```

#### Procedure

1. Log in to automation hub.
2. From the navigation panel, select **Collections** → **Remotes**.
3. In the **Community** remote, click the **More Actions** icon  and select **Edit**.
4. In the **YAML requirements** field, click **Browse** and locate the **requirements.yml** file on your local machine.
5. Click **Save**.  
You can now synchronize collections identified in your **requirements.yml** file from Ansible Galaxy to your private automation hub.
6. Click the **More Actions** icon  and select **Sync** to sync collections from Ansible Galaxy and Ansible automation hub.

#### Verification

The **Sync status** notification updates to notify you of completion or failure of Ansible Galaxy collections synchronization to your Ansible automation hub.

- Select **Community** from the collections content drop-down list to confirm successful synchronization.

### 1.2.5. Configuring proxy settings


If your private automation hub is behind a network proxy, you can configure proxy settings on the remote to sync content located outside of your local network.

#### Prerequisites

- You have valid **Modify Ansible repo content** permissions. For more information on permissions, see [Configuring user access for your private automation hub](#) .
- You have a proxy URL and credentials from your local network administrator.

#### Procedure

1. Log in to private automation hub.

2. From the navigation panel, select **Collections** → **Remotes**.
3. In either the **rh-certified** or **Community** remote, click the **More Actions** icon  and select **Edit**.
4. Expand the **Show advanced options** drop-down menu.
5. Enter your proxy URL, proxy username, and proxy password in the appropriate fields.
6. Click **Save**.

## 1.3. COLLECTIONS AND CONTENT SIGNING IN PRIVATE AUTOMATION HUB

As an automation administrator for your organization, you can configure private automation hub for signing and publishing Ansible content collections from different groups within your organization.

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.

### 1.3.1. Configuring content signing on private automation hub

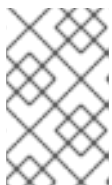
To successfully sign and publish Ansible Certified Content Collections, you must configure private automation hub for signing.

#### Prerequisites

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

#### Procedure

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



#### NOTE

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

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

```
["file": "filename", "signature": "filename.asc"]
```

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

#### Example:

The following script produces signatures for content:



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

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

ADMIN_ID="$PULP_SIGNING_KEY_FINGERPRINT"
PASSWORD="password"

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

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

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

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

```
[all:vars]
.
.
.
automationhub_create_default_collection_signing_service = True
automationhub_auto_sign_collections = True
automationhub_require_content_approval = True
automationhub_collection_signing_service_key = /abs/path/to/galaxy_signing_service.gpg
automationhub_collection_signing_service_script = /abs/path/to/collection_signing.sh
```

The two new keys (**automationhub\_auto\_sign\_collections** and **automationhub\_require\_content\_approval**) indicate that the collections must be signed and approved after they are uploaded to private automation hub.

### 1.3.2. Using content signing services in private automation hub

After you have configured content signing on your private automation hub, you can manually sign a new collection or replace an existing signature with a new one. When users download a specific collection, this signature indicates that the collection is for them and has not been modified after certification.

You can use content signing on private automation hub in the following scenarios:

- Your system does not have automatic signing configured and you must use a manual signing process to sign collections.
- The current signatures on the automatically configured collections are corrupted and need new signatures.

- You need additional signatures for previously signed content.
- You want to rotate signatures on your collections.

### Procedure

1. Log in to your Ansible Automation Platform.
2. From the navigation panel, select **Collections** → **Approval**. The Approval dashboard opens and displays a list of collections.
3. Click **Sign and approve** for each collection that you want to sign.

### Verification

- Verify that the collections you signed and manually approved are displayed in the **Collections** tab.

## 1.3.3. Downloading signature public keys

After you sign and approve collections, download the signature public keys from the automation hub UI. You must download the public key before you add it to the local system keyring.

### Procedure

1. Log in to your automation hub.
2. From the navigation panel, select **Signature Keys**. The Signature Keys dashboard displays a list of multiple keys: collections and container images.
  - To verify collections, download the key prefixed with **collections-**.
  - To verify container images, download the key prefixed with **container-**.
3. Choose one of the following methods to download your public key:
  - Select the menu icon and click **Download Key** to download the public key.
  - Select the public key from the list and click the *Copy to clipboard* icon.
  - Click the drop-down menu under the **Public Key** tab and copy the entire public key block.

Use the public key that you copied to verify the content collection that you are installing.

## 1.3.4. Configuring Ansible-Galaxy CLI to verify collections

You can configure Ansible-Galaxy CLI to verify collections. This ensures that downloaded collections are approved by your organization and have not been changed after they were uploaded to automation hub.

If a collection has been signed by automation hub, the server provides ASCII armored, GPG-detached signatures to verify the authenticity of **MANIFEST.json** before using it to verify the collection's contents. You must opt into signature verification by [configuring a keyring](#) for **ansible-galaxy** or providing the path with the **--keyring** option.

## Prerequisites

- Signed collections are available in automation hub to verify signature.
- Certified collections can be signed by approved roles within your organization.
- Public key for verification has been added to the local system keyring.

## Procedure

1. To import a public key into a non-default keyring for use with **ansible-galaxy**, run the following command.

```
gpg --import --no-default-keyring --keyring ~/.ansible/pubring.kbx my-public-key.asc
```



### NOTE

In addition to any signatures provided by the automation hub, signature sources can also be provided in the requirements file and on the command line. Signature sources should be URIs.

2. To verify the collection name provided on the CLI with an additional signature, run the following command:

```
ansible-galaxy collection install namespace.collection
--signature https://examplehost.com/detached_signature.asc
--signature file:///path/to/local/detached_signature.asc --keyring ~/.ansible/pubring.kbx
```

You can use this option multiple times to provide multiple signatures.

3. Confirm that the collections in a requirements file list any additional signature sources following the collection's signatures key, as in the following example.

```
# requirements.yml
collections:
  - name: ns.coll
    version: 1.0.0
  signatures:
    - https://examplehost.com/detached_signature.asc
    - file:///path/to/local/detached_signature.asc

ansible-galaxy collection verify -r requirements.yml --keyring ~/.ansible/pubring.kbx
```

When you install a collection from automation hub, the signatures provided by the server are saved along with the installed collections to verify the collection's authenticity.

4. (Optional) If you need to verify the internal consistency of your collection again without querying the Ansible Galaxy server, run the same command you used previously using the **--offline** option.

### Are there any recommendations for collection naming?

Create a collection with **company\_name.product** format. This format means that multiple products can have different collections under the company namespace.

### How do I get a namespace on Ansible automation hub?

By default namespaces used on Ansible Galaxy are also used on Ansible automation hub by the Ansible partner team. For any queries and clarifications contact [ansiblepartners@redhat.com](mailto:ansiblepartners@redhat.com).

## 1.4. ANSIBLE VALIDATED CONTENT

Red Hat Ansible Automation Platform includes Ansible validated content, which complements existing Red Hat Ansible Certified Content.

Ansible validated content provides an expert-led path for performing operational tasks on a variety of platforms from both Red Hat and our trusted partners.

### 1.4.1. Configuring validated collections with the installer

When you download and run the bundle installer, certified and validated collections are automatically uploaded. Certified collections are uploaded into the **rh-certified** repository. Validated collections are uploaded into the **validated** repository.

You can change to default configuration by using two variables:

- **automationhub\_seed\_collections** is a boolean that defines whether or not preloading is enabled.
- **automationhub\_collection\_seed\_repository**. A variable that enables you to specify the type of content to upload when it is set to **true**. Possible values are **certified** or **validated**. If missing both content sets will be uploaded.

### 1.4.2. Installing validated content using the tarball

If you are not using the bundle installer, you can use a standalone tarball, **ansible-validated-content-bundle-1.tar.gz**. You can also use this standalone tarball later to update validated contents in any environment, when a newer tarball becomes available, without having to re-run the bundle installer.

#### Prerequisites

You require the following variables to run the playbook.

Name	Description
<b>automationhub_admin_password</b>	Your administration password.
<b>automationhub_api_token</b>	The API token generated for your automation hub.
<b>automationhub_main_url</b>	For example, <b>https://automationhub.example.com</b>
<b>automationhub_require_content_approval</b>	Boolean ( <b>true</b> or <b>false</b> )  This must match the value used during automation hub deployment.  This variable is set to <b>true</b> by the installer.

## Procedure

1. To obtain the tarball, navigate to the [Red Hat Ansible Automation Platform download](#) page and select **Ansible Validated Content**.
2. Upload the content and define the variables (this example uses **automationhub\_api\_token**):

```
ansible-playbook collection_seed.yml
-e automationhub_api_token=<api_token>
-e automationhub_main_url=https://automationhub.example.com
-e automationhub_require_content_approval=true
```



### NOTE

Use either **automationhub\_admin\_password** or **automationhub\_api\_token**, not both.

When complete, the collections are visible in the validated collection section of private automation hub. Users can now view and download collections from your private automation hub.

## Additional Resources

For more information on running ansible playbooks, see [ansible-playbook](#).

## CHAPTER 2. MANAGING COLLECTIONS IN AUTOMATION HUB

As a content creator, you can use namespaces in automation hub to curate and manage collections for the following purposes:

- Create groups with permissions to curate namespaces and upload collections to private automation hub
- Add information and resources to the namespace to help end users of the collection in their automation tasks
- Upload collections to the namespace
- Review the namespace import logs to determine the success or failure of uploading the collection and its current approval status

For information on creating content, see the [Red Hat Ansible Automation Platform Creator Guide](#) .

### 2.1. USING NAMESPACES TO MANAGE COLLECTIONS IN AUTOMATION HUB

Namespaces are unique locations in automation hub to which you can upload and publish content collections. Access to namespaces in automation hub is governed by groups with permission to manage the content and related information that appears there.

You can use namespaces in automation hub to organize collections developed within your organization for internal distribution and use.

If you are working with namespaces, you must have a group that has permissions to create, edit and upload collections to namespaces. Collections uploaded to a namespace require administrative approval before you can publish them and make them available for use.

#### 2.1.1. Creating a new group for content curators

You can create a new group in private automation hub designed to support content curation in your organization. This group can contribute internally developed collections for publication in private automation hub.

To help content developers create a namespace and upload their internally developed collections to private automation hub, you must first create and edit a group and assign the required permissions.

##### Prerequisites

- You have administrative permissions in private automation hub and can create groups.

##### Procedure

1. Log in to your private automation hub.
2. From the navigation panel, select **User Access** → **Groups** and click **Create**.
3. Enter **Content Engineering** as a **Name** for the group in the modal and click **Create**. You have created the new group and the **Groups** page opens.

4. On the **Permissions** tab, click **Edit**.
5. Under **Namespaces**, add permissions for **Add Namespace**, **Upload to Namespace**, and **Change Namespace**.
6. Click **Save**.  
The new group is created with the permissions that you assigned. You can then add users to the group.
7. Click the **Users** tab on the **Groups** page.
8. Click **Add**.
9. Select users and click **Add**.

### 2.1.2. Creating a namespace

You can create a namespace to organize collections that your content developers upload to automation hub. When creating a namespace, you can assign a group in automation hub as owners of that namespace.

#### Prerequisites

- You have **Add Namespaces** and **Upload to Namespaces** permissions.

#### Procedure

1. Log in to your private automation hub.
2. From the navigation panel, select **Collections** → **Namespaces**.
3. Click **Create** and enter a **namespace name**.
4. Assign a group of **Namespace owners**.
5. Click **Create**.

Your content developers can now upload collections to your new namespace and allow users in groups assigned as owners to upload collections.

### 2.1.3. Adding additional information and resources to a namespace


You can add information and provide resources for your users to accompany collections included in the namespace. Add a logo and a description, and link users to your GitHub repository, issue tracker, or other online assets. You can also enter markdown text in the **Edit resources** tab to include more information. This is helpful to users who use your collection in their automation tasks.

#### Prerequisites

- You have **Change Namespaces** permissions.

#### Procedure

1. Log in to your private automation hub.

2. From the navigation panel, select **Collections** → **Namespaces**.
3. Click the **More Actions** icon  and select **Edit namespace**.
4. In the **Edit details** tab, enter information in the fields.
5. Click the **Edit resources** tab to enter markdown in the text field.
6. Click **Save**.

Your content developers can now upload collections to your new namespace, or allow users in groups assigned as owners to upload collections.

When you create a namespace, groups with permissions to upload to it can start adding their collections for approval. Collections in the namespace appear in the **Published** repository after approval.

### 2.1.4. Uploading collections to your namespaces

You can upload internally developed collections in **tar.gz** file format to your private automation hub namespace for review and approval by an automation hub administrator. When approved, the collection moves to the **Published** content repository where automation hub users can view and download it.



#### NOTE

Format your collection file name as follows: `<my_namespace-my_collection-1.0.0.tar.gz>`

#### Prerequisites

- You have a namespace to which you can upload the collection.

#### Procedure

1. Log in to your private automation hub.
2. From the navigation panel, select **Collections** → **Namespaces** and select a namespace.
3. Click **Upload collection**.
4. Click **Select file** from the **New collection** dialog.
5. Select the collection to upload.
6. Click **Upload**.

The **My Imports** screen shows a summary of tests and notifies you if the collection uploaded successfully or if it failed.

### 2.1.5. Reviewing your namespace import logs

You can review the status of collections uploaded to your namespaces to evaluate success or failure of the process.

Imported collections information includes:

#### Status



completed or failed

### Approval status

waiting for approval or approved

### Version

the version of the uploaded collection


### Import log

activities executed during the collection import

### Prerequisites

- You have access to a namespace to which you can upload collections.

### Procedure

1. Log in to your private automation hub.
2. From the navigation panel, select **Collections** → **Namespaces**.
3. Select a namespace.
4. Click the **More Actions** icon  and select **My imports**.
5. Use the search field or locate an imported collection from the list.
6. Click the imported collection.
7. Review collection import details to determine the status of the collection in your namespace.


## 2.1.6. Deleting a namespace

You can delete unwanted namespaces to manage storage on your automation hub server. You must first ensure that the namespace does not contain a collection with dependencies.

### Prerequisites

- The namespace you are deleting does not have a collection with dependencies.
- You have **Delete namespace** permissions.

### Procedure

1. Log in to your private automation hub.
2. From the navigation panel, select **Collections** → **Namespaces**.
3. Click the namespace to be deleted.
4. Click the **More Actions** icon , then click **Delete namespace**.



## NOTE

If the **Delete namespace** button is disabled, the namespace contains a collection with dependencies. Review the collections in this namespace, and delete any dependencies. See [Deleting a collection on automation hub](#) for information.

The namespace that you deleted, as well as its associated collections, is now deleted and removed from the namespace list view.

## 2.2. MANAGING THE PUBLICATION PROCESS OF INTERNAL COLLECTIONS IN AUTOMATION HUB

Use automation hub to manage and publish content collections developed within your organization. You can upload and group collections in namespaces. They need administrative approval to appear in the **Published** content repository. After you publish a collection, your users can access and download it for use.

You can reject submitted collections that do not meet organizational certification criteria.

### 2.2.1. About Approval

You can manage uploaded collections in automation hub by using the **Approval** feature located in the navigation panel.

#### Approval Dashboard

By default, the **Approval** dashboard lists all collections with **Needs Review** status. You can check these for inclusion in your **Published** repository.

#### Viewing collection details

You can view more information about the collection by clicking the **Version** number.

#### Filtering collections

Filter collections by **Namespace**, **Collection Name** or **Repository**, to locate content and update its status.

### 2.2.2. Approving collections for internal publication

You can approve collections uploaded to individual namespaces for internal publication and use. All collections awaiting review are located under the **Approval** tab in the **Staging** repository.

#### Prerequisites

- You have **Modify Ansible repo content** permissions.

#### Procedure

1. From the navigation panel, select **Collections** → **Approval**.  
Collections requiring approval have the status **Needs review**.
2. Select a collection to review.
3. Click the **Version** to view the contents of the collection.
4. Click **Certify** to approve the collection.

Approved collections are moved to the **Published** repository where users can view and download them for use.

### 2.2.3. Rejecting collections uploaded for review

You can reject collections uploaded to individual namespaces. All collections awaiting review are located under the **Approval** tab in the **Staging** repository.

Collections requiring approval have the status **Needs review**. Click the **Version** to view the contents of the collection.

#### Prerequisites

- You have **Modify Ansible repo content** permissions.

#### Procedure

1. From the navigation panel, select **Collections** → **Approval**.
2. Locate the collection to review.
3. Click **Reject** to decline the collection.

Collections you decline for publication are moved to the **Rejected** repository.

## 2.3. REPOSITORY MANAGEMENT WITH AUTOMATION HUB

As an automation hub administrator, you can create, edit, delete, and move automation content collections between repositories.

### 2.3.1. Types of repositories in automation hub

In automation hub you can publish collections to two types of repositories, depending on whether you want your collection to be verified:

#### Staging repositories

Any user with permission to upload to a namespace can publish collections into these repositories. Collections in these repositories are not available in the search page. Instead, they are displayed on the approval dashboard for an administrator to verify. Staging repositories are marked with the **pipeline=staging** label.

#### Custom repositories

Any user with write permissions on the repository can publish collections to these repositories. Custom repositories can be public where all users can see them, or private where only users with view permissions can see them. These repositories are not displayed on the approval dashboard. If the repository owner enables search, the collection can appear in search results.

By default, automation hub ships with one staging repository that is automatically used when a repository is not specified for uploading collections. Users can create new staging repositories during [repository creation](#).

### 2.3.2. Approval pipeline for custom repositories in automation hub

In automation hub you can approve collections into any repository marked with the **pipeline=approved**

label. By default, automation hub ships with one repository for approved content, but you have the option to add more from the repository creation screen. You cannot directly publish into a repository marked with the **pipeline=approved** label. A collection must first go through a staging repository and be approved before being published into a 'pipeline=approved' repository.

### Auto approval

When auto approve is enabled, any collection you upload to a staging repository is automatically promoted to all of the repositories marked as **pipeline=approved**.

### Approval required

When auto approve is disabled, the administrator can view the approval dashboard and see collections that have been uploaded into any of the staging repositories. Clicking **Approve** displays a list of approved repositories. From this list, the administrator can select one or more repositories to which the content should be promoted.

If only one approved repository exists, the collection is automatically promoted into it and the administrator is not prompted to select a repository.

### Rejection

Rejected collections are automatically placed into the rejected repository, which is pre-installed.

## 2.3.3. Role based access control to restrict access to custom repositories

Use Role Based Access Control (RBAC) to restrict user access to custom repositories by defining access permissions based on user roles. By default, users can view all public repositories in their automation hub, but they cannot modify a repository unless their role allows them access to do so. The same logic applies to other operations on the repository. For example, you can remove a user's ability to download content from a custom repository by changing their role permissions. See [Configuring user access for your private automation hub](#) for information about managing user access in automation hub.

## 2.3.4. Creating a custom repository in automation hub

When you use Red Hat Ansible Automation Platform to create a repository, you can configure the repository to be private or hide it from search results.

### Procedure

1. Log in to automation hub.
2. From the navigation panel, select **Collection** → **Repositories**.
3. Click **Add repository**.
4. Enter a **Repository name**.
5. In the **Description** field, describe the purpose of the repository.
6. To retain previous versions of your repository each time you make a change, select **Retained number of versions**. The number of retained versions can range anywhere between 0 and unlimited. To save all versions, leave this set to null.



### NOTE

If you have a problem with a change to your custom repository, you can [revert to a different repository version](#) that you have retained.

- In the **Pipeline** field, select a pipeline for the repository. This option defines who can publish a collection into the repository.

#### Staging

Anyone is allowed to publish automation content into the repository.

#### Approved

Collections added to this repository are required to go through the approval process by way of the staging repository. When auto approve is enabled, any collection uploaded to a staging repository is automatically promoted to all of the approved repositories.

#### None

Any user with permissions on the repository can publish to the repository directly, and the repository is not part of the approval pipeline.

- Optional: To hide the repository from search results, select **Hide from search**. This option is selected by default.
- Optional: To make the repository private, select **Make private**. This hides the repository from anyone who does not have permissions to view the repository.
- To sync the content from a remote repository into this repository, select **Remote** and select the remote that contains the collections you want included in your custom repository. For more information, see [Repository sync](#).
- Click **Save**.


#### Next steps

- After the repository is created, the details page is displayed. From here, you can provide access to your repository, review or add collections, and work with the saved versions of your custom repository.

### 2.3.5. Providing access to a custom automation hub repository

By default, private repositories and the automation content collections are hidden from all users in the system. Public repositories can be viewed by all users, but cannot be modified. Use this procedure to provide access to your custom repository.


#### Procedure

- Log in to private automation hub.
- From the navigation panel, select **Collection → Repositories**.
- Locate your repository in the list and click the **More Actions** icon , then select **Edit**.
- Select the **Access** tab.
- Select a group for **Repository owners**.  
See [Configuring user access for your private automation hub](#) for information about implementing user access.
- Select the roles you want assigned to the selected group.
- Click **Save**.

### 2.3.6. Adding collections to an automation hub repository

After you create your repository, you can begin adding automation content collections to it.



#### Procedure

1. From the navigation panel, select **Collection** → **Repositories**.
2. Locate your repository in the list and click the **More Actions** icon , then select **Edit**.
3. Select the **Collections version** tab.
4. Click **Add Collection** and select the collections that you want to add to your repository.
5. Click **Select**.

### 2.3.7. Revert to a different automation hub repository version

When automation content collections are added or removed from a repository, a new version is created. If a change to your repository causes a problem, you can revert to a previous version. Reverting is a safe operation and does not delete collections from the system, but rather, changes the content associated with the repository. The number of versions saved is defined in the **Retained number of versions** setting when a [repository is created](#).

#### Procedure

1. Log in to private automation hub.
2. From the navigation panel, select **Collection** → **Repositories**.
3. Locate your repository in the list and click the **More Actions** icon , then select **Edit**.
4. Locate the version you want to revert to and click the **More Actions** icon , and select **Revert to this version**.
5. Click **Revert**.

### 2.3.8. Managing remote configurations in automation hub

You can set up remote configurations to any server that is running automation hub. Remote configurations allow you to sync content to your custom repositories from an external collection source.

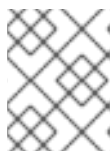
#### 2.3.8.1. Creating a remote configuration in automation hub

You can use Red Hat Ansible Automation Platform to create a remote configuration to an external collection source. Then, you can sync the content from those collections to your custom repositories.

#### Procedure

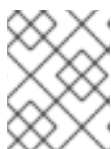
1. Log in to automation hub.
2. From the navigation panel, select **Collections** → **Remotes**.
3. Click **Add Remote**.

4. Enter a **Name** for the remote configuration.
5. Enter the **URL** for the remote server, including the path for the specific repository.

**NOTE**

To find the remote server URL and repository path, navigate to **Collection** → **Repositories**, select your repository, and click **Copy CLI configuration**.

6. Configure the credentials to the remote server by entering a **Token** or **Username** and **Password** required to access the external collection.

**NOTE**

To generate a token from the navigation panel, select **Collections** → **API token**, click **Load token** and copy the token that is loaded.

7. To access collections from console.redhat.com, enter the **SSO URL** to sign in to the identity provider (IdP).
8. Select or create a **YAML requirements** file to identify the collections and version ranges to synchronize with your custom repository. For example, to download only the kubernetes and AWS collection versions 5.0.0 or later the requirements file would look like this:

```
Collections:
- name: community.kubernetes
- name: community.aws
version:">=5.0.0"
```

**NOTE**

All collection dependencies are downloaded during the Sync process.

9. Optional: To configure your remote further, use the options available under **Advanced configuration**:
  - a. If there is a corporate proxy in place for your organization, enter a **Proxy URL**, **Proxy Username** and **Proxy Password**.
  - b. Enable or disable transport layer security using the **TLS validation** checkbox.
  - c. If digital certificates are required for authentication, enter a **Client key** and **Client certificate**.
  - d. If you are using a self-signed SSL certificate for your server, enter the PEM encoded client certificate used for authentication in the **CA certificate** field.
  - e. To accelerate the speed at which collections in this remote can be downloaded, specify the number of collections that can be downloaded in tandem in the **Download concurrency** field.
  - f. To limit the number of queries per second on this remote, specify a **Rate Limit**.


**NOTE**

Some servers can have a specific rate limit set, and if exceeded, synchronization fails.

### 2.3.8.2. Providing access to a remote configuration

After you create a remote configuration, you must provide access to it before anyone can use it.

**Procedure**

1. Log in to private automation hub.
2. From the navigation panel, select **Collections** → **Remotes**.
3. Locate your repository in the list, click the **More Actions** icon , and select **Edit**.
4. Select the **Access** tab.
5. Select a group for **Repository owners**. See [Configuring user access for your private automation hub](#) for information about implementing user access.
6. Select the appropriate roles for the selected group.
7. Click **Save**.

### 2.3.9. Synchronizing repositories in automation hub

You can distribute relevant automation content collections to your users by synchronizing repositories from one automation hub to another. To ensure you have the latest collection updates, synchronize your custom repository with the remote regularly.

**Procedure**

1. Log in to automation hub.
2. From the navigation panel, select **Collection** → **Repositories**.
3. Locate your repository in the list and click **Sync**.  
All collections in the configured remote are downloaded to your custom repository. To check the status of the collection sync, select **Task Management** from the navigation panel.

**NOTE**

To limit repository synchronization to specific collections within a remote, you can identify specific collections to be pulled by using a requirements.yml file. See [Create a remote](#) for more information.

**Additional resources**

For more information about using requirements files, see [Install multiple collections with a requirements file](#) in the *Using Ansible collections* guide.

### 2.3.10. Exporting and importing collections in automation hub



Ansible automation hub stores automation content collections within repositories. These collections are versioned by the automation content creator. Many versions of the same collection can exist in the same or different repositories at the same time.

Collections are stored as .tar files that can be imported and exported. This storage format ensures that the collection you are importing to a new repository is the same one that was originally created and exported.

### 2.3.10.1. Exporting an automation content collection in automation hub

After collections are finalized, you can import them to a location where they can be distributed to others across your organization.

#### Procedure

1. Log in to private automation hub.
2. From the navigation panel, select **Collections** → **Collections**. The **Collections** page displays all collections across all repositories. You can search for a specific collection.
3. Select the collection that you want to export. The collection details page opens.
4. From the **Install** tab, select **Download tarball**. The .tar file is downloaded to your default browser downloads folder. You can now import it to the location of your choosing.

### 2.3.10.2. Importing an automation content collection in automation hub

As an automation content creator, you can import a collection to use in a custom repository. To use a collection in your custom repository, you must first import the collection into your namespace so the automation hub administrator can approve it.

#### Procedure

1. Log in to automation hub.
2. From the navigation panel, select **Collections** → **Namespaces**. The **Namespaces** page displays all of the namespaces available.
3. Click **View Collections**.
4. Click **Upload Collection**.
5. Navigate to the collection tarball file, select the file and click **Open**.
6. Click **Upload**.  
The **My Imports** screen displays a summary of tests and notifies you if the collection upload is successful or has failed.



#### NOTE

If the collection is not approved, it is not displayed in the published repository.

#### Additional resources

- See [Approval pipeline](#) for more information about collection and repository approvals.

## CHAPTER 3. MANAGE CONTAINERS IN PRIVATE AUTOMATION HUB

Learn the administrator workflows and processes for configuring private automation hub container registry and repositories.

### 3.1. MANAGE YOUR PRIVATE AUTOMATION HUB CONTAINER REGISTRY

Manage container image repositories in your Ansible Automation Platform infrastructure by using the automation hub container registry. You can perform the following tasks with Automation hub:

- Control who can access individual container repositories
- Change tags on images
- View activity and image layers
- Provide additional information related to each container repository

#### 3.1.1. Container registries

The automation hub container registry is used for storing and managing container images. When you have built or sourced a container image, you can push that container image to the registry portion of private automation hub to create a container repository.

##### Next steps

- Push a container image to the automation hub container registry.
- Create a group with access to the container repository in the registry.
- Add the new group to the container repository.
- Add a README to the container repository to provide users with information and relevant links.

### 3.2. CONFIGURING USER ACCESS FOR CONTAINER REPOSITORIES IN PRIVATE AUTOMATION HUB

To determine who can access and manage images in your Ansible Automation Platform, you must configure user access for container repositories in your private automation hub.

#### 3.2.1. Container registry group permissions

You can control how users can interact with containers managed in private automation hub. Use the following list of permissions to create groups with the right privileges for your container registries.

**Table 3.1. List of group permissions used to manage containers in private automation hub**

Permission name	Description
Create new containers	Users can create new containers

Permission name	Description
Change container namespace permissions	Users can change permissions on the container repository
Change container	Users can change information on a container
Change image tags	Users can modify image tags
Pull private containers	Users can pull images from a private container
Push to existing container	Users can push an image to an existing container
View private containers	Users can view containers marked as private

### 3.2.2. Creating a new group in private automation hub

You can create and assign permissions to a group in private automation hub that enables users to access specified features in the system. By default, the **Admin** group in the automation hub has all permissions assigned and is available on initial login. Use the credentials created when installing private automation hub.

For more information, see [Creating a new group in private automation hub](#) in the Getting started with automation hub guide.

### 3.2.3. Assigning permissions to groups

By default, new groups do not have any assigned permissions. You can assign permissions to groups in private automation hub that enable users to access specific features in the system.

You can add permissions when first creating a group or edit an existing group to add or remove permissions

For more information, see [Assigning permissions to groups](#) in the Getting started with automation hub guide.

#### Additional resources

- See [Container registry group permissions](#) to learn more about specific permissions.

### 3.2.4. Adding users to existing groups

You can add users to groups when you create a group. But, you can also manually add users to existing groups.

For more information, see [Adding users to existing groups](#) in the Getting started with automation hub guide.

## 3.3. POPULATING YOUR PRIVATE AUTOMATION HUB CONTAINER REGISTRY

By default, private automation hub does not include container images. To populate your container registry, you must push a container image to it.

You must follow a specific workflow to populate your private automation hub container registry:

- Pull images from the Red Hat Ecosystem Catalog (registry.redhat.io)
- Tag them
- Push them to your private automation hub container registry

### IMPORTANT

Image manifests and filesystem blobs were both originally served directly from **registry.redhat.io** and **registry.access.redhat.com**. As of 1 May 2023, filesystem blobs are served from **quay.io** instead. To avoid problems pulling container images, enable outbound connections to the following hostnames:

- **cdn.quay.io**
- **cdn01.quay.io**
- **cdn02.quay.io**
- **cdn03.quay.io**

Make this change to any firewall configuration that specifically enables outbound connections to **registry.redhat.io** or **registry.access.redhat.com**.

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

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

### 3.3.1. Pulling images for use in automation hub

Before you can push container images to your private automation hub, you must first pull them from an existing registry and tag them for use. The following example details how to pull an image from the Red Hat Ecosystem Catalog (registry.redhat.io).

#### Prerequisites

You have permissions to pull images from registry.redhat.io.

#### Procedure

1. Log in to Podman by using your registry.redhat.io credentials:

```
$ podman login registry.redhat.io
```

2. Enter your username and password.
3. Pull a container image:

```
$ podman pull registry.redhat.io/<container_image_name>:<tag>
```

## Verification

To verify that the image you recently pulled is contained in the list, take these steps:

1. List the images in local storage:

```
$ podman images
```

2. Check the image name, and verify that the tag is correct.

## Additional resources

- See [Red Hat Ecosystem Catalog Help](#) for information on registering and getting images.

### 3.3.2. Tagging images for use in automation hub

After you pull images from a registry, tag them for use in your private automation hub container registry.

#### Prerequisites

- You have pulled a container image from an external registry.
- You have the FQDN or IP address of the automation hub instance.

#### Procedure

- Tag a local image with the automation hub container repository:

```
$ podman tag registry.redhat.io/<container_image_name>:<tag>  
<automation_hub_hostname>/<container_image_name>
```

#### Verification

1. List the images in local storage:

```
$ podman images
```

2. Verify that the image you recently tagged with your automation hub information is contained in the list.

### 3.3.3. Pushing a container image to private automation hub

You can push tagged container images to private automation hub to create new containers and populate the container registry.

#### Prerequisites

- You have permissions to create new containers.
- You have the FQDN or IP address of the automation hub instance.

## Procedure

1. Log in to Podman using your automation hub location and credentials:

```
$ podman login -u=<username> -p=<password> <automation_hub_url>
```

2. Push your container image to your automation hub container registry:

```
$ podman push <automation_hub_url>/<container_image_name>
```

## Troubleshooting

The **push** operation re-compresses image layers during the upload, which is not guaranteed to be reproducible and is client-implementation dependent. This may lead to image-layer digest changes and a failed push operation, resulting in **Error: Copying this image requires changing layer representation, which is not possible (image is signed or the destination specifies a digest)**.

## Verification

1. Log in to your automation hub.
2. Navigate to **Container Registry**.
3. Locate the container in the container repository list.

## 3.4. SETTING UP YOUR CONTAINER REPOSITORY

When you set up your container repository, you must add a description, include a README, add groups that can access the repository, and tag images.

### 3.4.1. Prerequisites to setting up your container registry

- You are logged in to a private automation hub.
- You have permissions to change the repository.

### 3.4.2. Adding a README to your container repository

Add a README to your container repository to provide instructions to your users on how to work with the container. Automation hub container repositories support Markdown for creating a README. By default, the README is empty.

## Prerequisites

- You have permissions to change containers.

## Procedure

1. Log in to automation hub.
2. From the navigation panel, select **Execution Environments** → **Execution Environments**.
3. Select your container repository.

4. On the **Detail** tab, click **Add**.
5. In the **Raw Markdown** text field, enter your README text in Markdown.
6. Click **Save** when you are finished.

After you add a README, you can edit it at any time by clicking **Edit** and repeating steps 4 and 5.

### 3.4.3. Providing access to your container repository

Provide access to your container repository for users who need to work with the images. Adding a group allows you to modify the permissions the group can have to the container repository. You can use this option to extend or restrict permissions based on what the group is assigned.

#### Prerequisites

- You have **change container namespace** permissions.

#### Procedure

1. Log in to automation hub.
2. From the navigation panel, select **Execution Environments** → **Execution Environments**.
3. Select your container repository.
4. From the **Access** tab, click **Select a group**.
5. Select the group or groups to which you want to grant access and click **Next**.
6. Select the roles that you want to add to this execution environment and click **Next**.
7. Click **Add**.

### 3.4.4. Tagging container images

Tag images to add an additional name to images stored in your automation hub container repository. If no tag is added to an image, automation hub defaults to **latest** for the name.

#### Prerequisites

- You have **change image tags** permissions.

#### Procedure

1. From the navigation panel, select **Execution Environments** → **Execution Environments**.
2. Select your container repository.
3. Click the **Images** tab.
4. Click the **More Actions** icon **⋮**, and click **Manage tags**.
5. Add a new tag in the text field and click **Add**.
6. Optional: Remove **current tags** by clicking **x** on any of the tags for that image.

7. Click **Save**.

### Verification

- Click the **Activity** tab and review the latest changes.

### 3.4.5. Creating a credential in automation controller

To pull container images from a password or token-protected registry, you must create a credential in automation controller.

In earlier versions of Ansible Automation Platform, you were required to deploy a registry to store execution environment images. On Ansible Automation Platform 2.0 and later, the system operates as if you already have a container registry up and running. To store execution environment images, add the credentials of only your selected container registries.

### Procedure

1. Navigate to automation controller.
2. From the navigation panel, select **Resources** → **Credentials**.
3. Click **Add** to create a new credential.
4. Enter an authorization **Name**, **Description**, and **Organization**.
5. Select the **Credential Type**.
6. Enter the **Authentication URL**. This is the container registry address.
7. Enter the **Username** and **Password or Token** required to log in to the container registry.
8. Optional: To enable SSL verification, select **Verify SSL**.
9. Click **Save**.

## 3.5. PULLING IMAGES FROM A CONTAINER REPOSITORY

Pull images from the automation hub container registry to make a copy to your local machine. Automation hub provides the **podman pull** command for each **latest** image in the container repository. You can copy and paste this command into your terminal, or use **podman pull** to copy an image based on an image tag.

### 3.5.1. Pulling an image

You can pull images from the automation hub container registry to make a copy to your local machine.

### Prerequisites

- You must have permission to view and pull from a private container repository.

### Procedure

1. If you are pulling container images from a password or token-protected registry, [create a credential in automation controller](#) before pulling the image.



2. From the navigation panel, select **Execution Environments** → **Execution Environments**.
3. Select your container repository.
4. In the **Pull this image** entry, click **Copy to clipboard**.
5. Paste and run the command in your terminal.

### Verification

- Run **podman images** to view images on your local machine.

## 3.5.2. Syncing images from a container repository

You can pull images from the automation hub container registry to sync an image to your local machine. To sync an image from a remote container registry, you must first configure a remote registry.

### Prerequisites

You must have permission to view and pull from a private container repository.

### Procedure

1. From the navigation panel, select **Execution Environments** → **Execution Environments**.
2. Add <https://registry.redhat.io> to the registry.
3. Add any required credentials to authenticate.



#### NOTE

Some container registries are aggressive with rate limiting. Set a rate limit under **Advanced Options**.

4. From the navigation panel, select **Execution Environments** → **Execution Environments**.
5. Click **Add execution environment** in the page header.
6. Select the registry you want to pull from. The **Name** field displays the name of the image displayed on your local registry.



#### NOTE

The **Upstream name** field is the name of the image on the remote server. For example, if the upstream name is set to "alpine" and the **Name** field is "local/alpine", the alpine image is downloaded from the remote and renamed to "local/alpine".

7. Set a list of tags to include or exclude. Syncing images with a large number of tags is time consuming and uses a lot of disk space.

### Additional resources

- See [Red Hat Container Registry Authentication](#) for a list of registries.

- See the [What is Podman?](#) documentation for options to use when pulling images.

## 3.6. WORKING WITH SIGNED CONTAINERS

Automation execution environments are container images used by Ansible automation controller to run jobs. You can download this content to private automation hub, and publish it within your organization.

### 3.6.1. Deploying your system for container signing

Automation hub implements image signing to offer better security for the execution environment container images.

To deploy your system so that it is ready for container signing, create a signing script.



#### NOTE

Installer looks for the script and key on the same server where installer is located.

#### Procedure

1. From a terminal, create a signing script, and pass the script path as an installer parameter.

##### Example:

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

# pulp_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. Review the Ansible Automation Platform installer inventory file for options for container signing that begin with **automationhub\_\***.

```
[all:vars]
```

```
.
```

```

.
.
automationhub_create_default_container_signing_service = True
automationhub_container_signing_service_key = /absolute/path/to/key/to/sign
automationhub_container_signing_service_script = /absolute/path/to/script/that/signs

```

3. Once installation is complete, navigate to your automation hub.
4. From the navigation panel, select **Signature Keys**.
5. Ensure that you have a key titled **container-default**, or **container-anyname**.



#### NOTE

The **container-default** service is created by the Ansible Automation Platform installer.

### 3.6.2. Adding containers remotely to automation hub

You can add containers remotely to automation hub in one of the following two ways:

- Create Remotes
- Execution Environment

#### Procedure

1. Log in to automation hub.
2. From the navigation panel, select **Execution Environments** → **Remote Registries**.
3. Click **Add remote registry**.
  - In the **Name** field, enter the name of the registry where the container resides.
  - In the **URL** field, enter the URL of the registry where the container resides.
  - In the **Username** field, enter the username if necessary.
  - In the **Password** field, enter the password if necessary.
  - Click **Save**.

### 3.6.3. Adding an execution environment

Automation execution environments are container images that make it possible to incorporate system-level dependencies and collection-based content. Each execution environment allows you to have a customized image to run jobs, and each of them contain only what you need when running the job.

#### Procedure

1. From the navigation panel, select **Execution Environments** → **Execution Environments**.
2. Click **Add execution environment**.

3. Enter the name of the execution environment.
4. Optional: Enter the upstream name.
5. Under **Registry**, select the name of the registry from the drop-down menu.
6. Enter tags in the **Add tag(s) to include** field. If the field is blank, all the tags are passed. You must specify which repository specific tags to pass.
7. The remaining fields are optional:
  - **Currently included tags**
  - **Add tag(s) to exclude**
  - **Currently excluded tag(s)**
  - **Description**
8. Click **Save**.
9. Synchronize the image.

### 3.6.4. Pushing container images from your local environment

Use the following procedure to sign images on a local system and push those signed images to the automation hub registry.

#### Procedure

1. From a terminal, log into podman, or any container client currently in use:

```
> podman pull <container-name>
```

2. After the image is pulled, add tags (for example: latest, rc, beta, or version numbers, such as 1.0; 2.3, and so on):

```
> podman tag <container-name> <server-address>/<container-name>:<tag name>
```

3. Sign the image after changes have been made, and push it back up to the automation hub registry:

```
> podman push <server-address>/<container-name>:<tag name> --tls-verify=false --sign-by  
<reference to the gpg key on your local>
```

If the image is not signed, it can only be pushed with any current signature embedded. Alternatively, you can use the following script to push the image without signing it:


```
> podman push <server-address>/<container-name>:<tag name> --tls-verify=false
```

4. Once the image has been pushed, navigate to your automation hub.
5. From the navigation panel, select **Execution Environments** → **Execution Environments**.
6. To display the new execution environment, click the **Refresh** icon.

7. Click the name of the image to view your pushed image.

### Troubleshooting

The details page in automation hub indicates whether or not an image has been signed. If the details page indicates that an image is **Unsigned**, you can sign the image from automation hub using the following steps:

1. Click the image name to navigate to the details page.
2. Click the **More Actions** icon . Three options are available:
  - **Use in Controller**
  - **Delete**
  - **Sign**
3. Click **Sign** from the drop-down menu.

The signing service signs the image. After the image is signed, the status changes to "signed".


### 3.6.5. Policies with signed images

Policies can be used by podman or other image clients to ensure the validity of the image by assigning specific policies to that signature.

### 3.6.6. Using podman to ensure an image is signed by a specific signature

When ensuring a signature is signed by specific signatures, the signature must be on your local environment.

#### Procedure

1. From the navigation panel, select **Signature Keys**.
2. Click the **More Actions** icon  next to the signature that you are using.
3. Select **Download key** from the drop-down menu. A new window opens.
4. In the **Name** field, enter the name of the key.
5. Click **Save**.

### 3.6.7. Configuring the client to verify signatures

To ensure a container image pulled from the remote registry is properly signed, you must first configure the image with the proper public key in a policy file.

#### Prerequisites

- The client must have sudo privileges configured to verify signatures.

#### Procedure

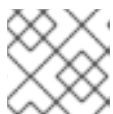
1. Open your terminal and use the following command:

```
> sudo <name of editor> /etc/containers/policy.json
```

The file that is displayed is similar to this:

```
{
  "default": [{"type": "reject"}],
  "transports": {
    "docker": {
      "quay.io": [{"type": "insecureAcceptAnything"}],
      "docker.io": [{"type": "insecureAcceptAnything"}],
      "<server-address>": [
        {
          "type": "signedBy",
          "keyType": "GPGKeys",
          "keyPath": "/tmp/containersig.txt"
        }
      ]
    }
  }
}
```

This file shows that neither **quay.io**, or **docker.io** will perform the verification, because the type is **insecureAcceptAnything** which overrides the default type of **reject**. However, **<server-address>** will perform the verification, because the parameter **type** is set to **"signedBy"**.



#### NOTE

The only **keyType** currently supported is GPG keys.

2. Under the **<server-address>** entry, modify the **keyPath** <1> to include the name of your key file.

```
{
  "default": [{"type": "reject"}],
  "transports": {
    "docker": {
      "quay.io": [{"type": "insecureAcceptAnything"}],
      "docker.io": [{"type": "insecureAcceptAnything"}],
      "<server-address>": [{
        "type": "signedBy",
        "keyType": "GPGKeys",
        "keyPath": "/tmp/<key file name>",
        "signedIdentity": {
          "type": "matchExact"
        }
      }
    ]
  }
}
```

3. Save and close the file.

## Verification

- Pull the file using Podman, or your client of choice:

```
> podman pull <server-address>/<container-name>:<tag name> --tls-verify=false
```

This response verifies the image has been signed with no errors. If the image is not signed, the command fails.

#### Additional resources

- For more information about policy.json, see [documentation for containers-policy.json](#).


## 3.7. DELETING A CONTAINER REPOSITORY

Delete a container repository from your private automation hub to manage your disk space. You can delete repositories from the Red Hat Ansible Automation Platform interface in the **Container Repository** list view.

#### Prerequisites

- You have permissions to manage repositories.

#### Procedure

1. Navigate to automation hub.
2. From the navigation panel, select **Execution Environments** → **Execution Environments**.
3. On the container repository that you want to delete, click the **More Actions** icon , and click **Delete**.
4. When the confirmation message is displayed, click the checkbox and click **Delete**.

#### Verification

- Return to the **Execution Environments** list view. If the container repository has been successfully deleted, the container repository is no longer on the list.