

Extend

Ansible Automation Platform 2.6

Contents

1 Extend	7
Deploy Ansible MCP server on Ansible Automation Platform	7
Overview	7
Benefits	7
Workflow	8
Ansible MCP server toolsets	9
Server-level and user-level permissions	11
Telemetry data collection for Ansible MCP server	12
Prerequisites	12
Overview of deploying the Ansible MCP server	13
Deploy the Ansible MCP server on a container-based installation	13
Deploy the Ansible MCP server on operator-based installation	15
Create an API token for the Ansible MCP server	17
Connect an AI agent to the Ansible MCP server	18
Troubleshoot Ansible MCP server errors	21
API output format rejected with 406 Status Code	21
User requests rejected with 400 status code	21
Ansible MCP server permissions are changed post deployment	22
Deploy the intelligent assistant	22
Create a chatbot configuration secret	22
Examples of chatbot configuration secrets	24
Update the YAML file of the Ansible Automation Platform operator	26
Change your LLM model	28
Use the Ansible Lightspeed intelligent assistant	29
Access the Ansible Lightspeed intelligent assistant	29
Use the Ansible Lightspeed intelligent assistant	30
Deploy Red Hat Ansible Lightspeed on containerized Ansible Automation Platform	31
Overview	31
Ansible Lightspeed intelligent assistant	32
Integration with MCP server	32
Ansible Lightspeed coding assistant	32

Deployment models.....	33
Ansible Automation Platform requirements	33
Large Language Model (LLM) provider requirements	34
Process to deploy Red Hat Ansible Lightspeed on a container-based installation.....	35
Configure Red Hat Ansible Lightspeed variables	36
Red Hat Ansible Lightspeed variables	38
Change your LLM model	47
Use the Ansible Lightspeed intelligent assistant	47
Access the Ansible Lightspeed intelligent assistant.....	48
Use the Ansible Lightspeed intelligent assistant	48
Configure the Ansible VS Code extension.....	49
Deploy the Ansible Lightspeed intelligent assistant on OpenShift Container Platform	50
Overview.....	50
Install Ansible development tools.....	54
Requirements	54
Requirements for Ansible development tools on Windows.....	55
Authenticate with the Red Hat container registry	56
Install and configure VS Code.....	57
Install the VS Code Ansible extension	57
Configure Ansible extension settings	58
Install and configure the Dev Containers extension.....	59
Set the lanaguge for the VS Code extension.....	60
Associate the Ansible language to YAML files	60
Add the persistent file association for the Ansible language to <code>settings.json</code>	60
Install Ansible development tools on a container inside VS Code	61
Install Ansible development tools from an RPM package	62
Configure Red Hat AI providers in the Ansible VS Code extension.....	64
Install Red Hat Developer Hub (Red Hat OpenShift Container Platform only).....	66
Red Hat Developer Hub	66
Ansible plug-ins for Red Hat Developer Hub.....	66
Install with a Helm chart	66
Prerequisites	67
Recommended RHDH preconfiguration	68
Choose a plug-in delivery method	68

Use OCI container delivery method	68
Download the Ansible plug-ins files	70
Create a registry for the Ansible plug-ins	71
Complete required configurations	73
Add the Ansible plug-ins configuration	73
Add a pull secret to the Red Hat Developer Hub Helm configuration	76
Add the Ansible Developer Tools container	77
Add a custom ConfigMap	79
Configure the Ansible development tools Server.....	79
Configure Ansible Automation Platform details	80
Add Ansible plug-ins software templates	81
Configure Role Based Access Control.....	82
Optional configurations	83
Enable Red Hat Developer Hub authentication	83
Configure Ansible plug-ins optional integrations	84
Configure OpenShift Dev Spaces	84
Configure the private automation hub URL	85
Example ConfigMaps	86
app-config-rhdh ConfigMap example for Ansible plug-ins	86
Full Helm chart config example for Ansible plug-ins.....	88
Install with the Ansible Automation Platform Operator on OpenShift Container Platform	89
Prerequisites	89
Recommended RHDH preconfiguration	90
Add a sidecar container for Ansible development tools	91
Choose a plug-in delivery method when using the Operator	92
Use the OCI container delivery with the Operator	92
Download the Ansible plug-ins files	94
Create a registry for the Ansible plug-ins	95
Install the dynamic plug-ins.....	96
Configure the plug-ins	99
Add a custom ConfigMap	99
Configure the Ansible development tools Server.....	100
Configure Ansible Automation Platform details	100
Add Ansible plug-ins software templates	101
Configure Role Based Access Control.....	102

Extend

Uninstall from a Helm installation on OpenShift Container Platform.....	103
Uninstall a Helm chart installation	103
Uninstall an Operator installation on OpenShift Container Platform.....	106
Remove the Ansible plug-ins from the ConfigMap	107
Remove Ansible Automation Platform and Dev Spaces from the custom Red Hat Developer Hub ConfigMap	108
Uninstall the sidecar container.....	110
Install self-service automation portal (OpenShift Container Platform only)	111
Architecture and licensing	112
Supported platforms	112
Interactive demonstration	112
Prerequisites	113
Pre-installation configuration	113
Select an organization	113
Create an OAuth application.....	114
Enable Oauth token creation for external users	115
Generate an API token for Ansible Automation Platform authentication	116
Install the self-service automation portal Helm chart.....	117
Configure the Helm chart from the OpenShift catalog	117
Verify the installation.....	119
Install in an air-gapped environment	120
Prerequisites	120
Prepare for air-gapped installation	121
Mirror container images	121
Download the helm chart package.....	123
Transfer assets to the disconnected environment	124
Install the Helm chart in the disconnected OpenShift environment.....	124
Access the disconnected OpenShift environment	125
Configure plug-in delivery for disconnected environments	125
Install the Helm chart.....	126
Configure CA certificates for private registries	130
Verify the deployment.....	131
Troubleshooting disconnected installations	132
Verify the configuration	132

Extend

View the deployment logs	133
Update the deployment URL	135
Add the deployment URL to the OAuth Application.....	135
Configure custom SSL certificates for self-service automation portal	136
Sign in to self-service automation portal	139
Set up initial RBAC rules in self-service automation portal	140
Understand the permission model	141
Configure RBAC for synchronization.....	141
Configure conditional access	145
Permissions reference for Ansible Automation Platform job template access	147
Adjust synchronization frequency between Ansible Automation Platform and self-service automation portal	148
Install automation dashboard to calculate savings (RHEL only)	148
Install automation dashboard.....	149
Configure automation dashboard	155
Synchronize data to automation dashboard.....	158
Verify cluster access tokens	159
Uninstall automation dashboard	165
Install Ansible Builder to create or edit execution environments	165
Red Hat product documentation legal notices	167
GNU GENERAL PUBLIC LICENSE	168
Apache license	179

1 Extend

Deploy Ansible MCP server on Ansible Automation Platform

As an organization administrator, you can deploy an Ansible Model Context Protocol (MCP) server on an operator-based installation or container-based installation of Ansible Automation Platform. This functionality is available as a Technology Preview release.

Overview

Model Context Protocol (MCP) is an open standard that enables AI models to use external AI tools and services via a unified interface. Using the Ansible MCP server, you can connect your Ansible Automation Platform with your preferred external AI tool (such as Claude, Cursor, or ChatGPT).

The AI tools can access key information about your Ansible Automation Platform environment and perform tasks. Ansible users can query information, execute workflows, and perform automation tasks using natural language prompts directly within their preferred AI tool.

NOTE:

Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

Benefits

The following are the benefits of the Ansible MCP server:

For external AI tools:

- Provides a standardized interface for securely querying infrastructure data and executing automation workflows within the Ansible Automation Platform.
- Enables agentic workflows to interact with the Ansible Automation Platform.

For Ansible users:

- Provides the ability to use the chatbot interface of their preferred external AI tool to get information about their Ansible Automation Platform environment, and run automation jobs directly through that tool.

For developers:

- Reduces the time and complexity of developing or integrating the Ansible Automation Platform with AI applications or agents.
- Simplifies AI integration, enabling existing automation through Ansible Automation Platform to be exposed to AI tools without writing custom API code or middleware.

Workflow

The standalone Ansible MCP server functions as a secure link between your external AI clients and the Ansible Automation Platform. The AI agent accesses underlying infrastructure only when the Ansible MCP server has appropriate permissions.

The following describes the workflow:

1. AI client (The requester): The user initiates a request through their external AI agent (for example, Cursor or Claude) using natural-language prompts.
2. The AI model (The translator): The AI agent receives the request, interprets the intent, and maps it to the appropriate exposed Ansible toolset. It then sends a structured toolset call with the necessary parameters.
3. Ansible MCP server (The gatekeeper): Upon receiving the call, the Ansible MCP server validates the request. It uses the user's API token to authenticate with the automation controller.
4. Ansible controller (The executor): The automation controller accepts the validated command from the MCP server and triggers the appropriate automation job.
5. Response loop: The automation result is returned to the Ansible MCP server, standardized into a format the AI agent can process, and displayed to the user via the AI client.

IMPORTANT:

Both the Ansible MCP server and the Ansible Automation Platform UI access the Ansible Automation Platform API. However, because the AI tool processes the API output before displaying it in its chat interface, you might observe different results when comparing the output from the AI tool with the Ansible Automation Platform UI.

Ansible MCP server toolsets

The Ansible MCP server provides a pre-configured suite of toolsets that effectively act as a bridge between your preferred AI agent and the Ansible Automation Platform. Once configured, these toolsets enable your AI agent to perform specific, authorized actions without requiring you to leave the chat interface.

The Ansible MCP server turns your AI agent from a passive assistant into an active operator that can interact with your Ansible Automation Platform infrastructure and execute workflows or automate tasks based on the permissions you define.

The following toolsets are available in this Technology Preview release:

Toolset	Description	Usage examples
Job management	Tools to list available job templates, launch automation jobs, and monitor their real-time status.	Operators can: • Launch job templates and workflows to execute automation tasks for their projects and services. • View job output and logs to troubleshoot failed automation tasks and understand what went wrong. • Relaunch failed jobs to recover from temporary failures and complete necessary automation tasks.
Inventory management	Tools to query your inventory for host details, check group membership, and verify system facts.	Operators can: • View and browse inventories across environments to understand which systems they are managing with automation. • Manage group assignments to target automation to specific sets of systems. • View hosts that are configured for automation.
System monitoring	Tools to retrieve job logs, troubleshoot failed tasks, and check the health of your automation environment.	Administrators can: • Perform platform status and health checks across all services to identify issues and ensure the automation platform is running correctly.

Toolset	Description	Usage examples
		• Monitor service health through the platform gateway to ensure all platform components are functioning correctly. • Audit user activity and generate reports to ensure compliance and identify potential security issues.
User management	Tools to allow the AI agent to administer access and organizational structure within the Ansible Automation Platform.	Administrators can: • Use natural-language prompts to provision users and enforce hierarchy, rather than manually navigating the UI. • Create, modify, and delete users and teams to manage access to the Ansible Automation Platform and support organizational changes. • Configure role-based access control to ensure users have the appropriate permissions for their responsibilities while maintaining security. • View team memberships and structure to see who else in their organization is working on automation.
Security/compliance	Tools that enable the AI agent to act as a security operator, managing sensitive credentials and verifying platform integrity without exposing raw secrets.	Operators can: • View available credentials to understand what authentication options are available for their automation jobs. Administrators can: • Manage credentials and security policies to ensure secure access to external systems while maintaining proper governance. • Manage custom credential types for seamless integration with third-party applications.
Platform configuration	Tools that enable organization administrators and developers to inspect and tune the Ansible	Administrators can: • Manage system settings across all components to configure the platform in line

Toolset	Description	Usage examples
	Automation Platform infrastructure itself.	with the organizational requirements and policies. • Manage and track licenses to ensure compliance with licensing terms and optimize license utilization. Developers can: • Tune execution environments to optimize the runtime performance of their automation content.

Server-level and user-level permissions

The Ansible MCP server employs a dual-layer security model to ensure safe integration between AI tools and your Ansible Automation Platform infrastructure. This approach combines a global administrative safeguard with the granular Role-Based Access Control (RBAC) of the Ansible Automation Platform.

You can grant the following access types to the Ansible MCP server:

- **Server-level permissions:** Organization administrators assign a global-level permission while deploying the Ansible MCP server. Administrators can choose one of the following access levels:
 - Read-only access: The default setting that enforces a strict "look but do not touch" policy. The AI agent can retrieve system data, such as logs and inventory, but the agent cannot launch jobs or modify configurations. This global safeguard overrides all individual user permissions to prevent unintended automation.
 - Read-write access: This setting authorizes the AI agent to make changes in your Ansible Automation Platform, such as executing job templates, managing resources, and applying infrastructure changes. However, these actions are subject to the specific RBAC permissions of the user-provided API token.
- **User-level permissions:** The AI agent's specific capabilities are inherited from the user account that generated the authentication API token.
 - Inherited permissions: The AI tool inherits the user's permissions and performs only the actions the user is authorized to perform. For example, if the user's token only has permissions to view the "network" inventory, the AI tool cannot access or modify the "database" inventory even if the user requests it.
 - Rejection of unauthorized actions: If the AI tool attempts an action (like launching a job) that the user's token is not authorized to perform, the Ansible Automation Platform API rejects the request.

WARNING:

Enabling read-write access for the Ansible MCP server grants the AI agent autonomy to directly make changes in your Ansible Automation Platform environment, for example, executing automation jobs. The AI agent can directly make changes in your Ansible Automation Platform environment only if the user has write permissions. Large Language Models (LLMs) can occasionally misinterpret prompts or hallucinate commands. Therefore, enabling read-write access may introduce a risk of unintended changes to your environment.

Telemetry data collection for Ansible MCP server

Red Hat collects anonymized telemetry data from the Ansible MCP server. The telemetry data includes metrics related to MCP server performance, adoption trends, and usage patterns.

Telemetry data will be automatically collected for Ansible MCP server deployments using Ansible Automation Platform patch release on 21 January 2026 and later versions. Red Hat will use this data to monitor the operational health of your MCP servers and to ensure the long-term scalability of the MCP ecosystem.

IMPORTANT:

Telemetry data collection cannot be disabled, but strict user privacy is maintained. Red Hat does not collect users' personal information, such as usernames or passwords. If any personal information is inadvertently received, the data is deleted. For more information, see the Red Hat Privacy Statement under Related Links below.

Prerequisites

- Platform version: An instance of Ansible Automation Platform 2.6 or later.
- Deployment environment:
 - OpenShift: Access to an OpenShift cluster with permissions to install operators.
 - Containerized: A supported container runtime.
- Access credentials: A valid user or service account within Ansible Automation Platform with permissions to execute the desired automation jobs. You will need to generate an API token for this account.

Overview of deploying the Ansible MCP server

Perform the following tasks to deploy and configure an Ansible MCP server and integrate it with your preferred AI tool:

Step number	Task	Description
1	Deploy and configure an Ansible MCP server on container-based installation	An organization administrator deploys and configures the Ansible MCP server on a container-based installation of Ansible Automation Platform 2.6.
2	Create an API token for the Ansible MCP server	An Ansible user creates an API token for their Ansible Automation Platform instance and uses it to connect to their preferred AI tool. The AI tools will inherit the user's permissions for authentication using the API token.
3	Connect an external AI agent to the Ansible MCP server	The Ansible user then configures an external AI tool with the Ansible MCP server's API token, enabling the AI tool to connect to the Ansible MCP server and execute workflows and automate tasks.

Related information

[Deploy and configure the Ansible MCP server on a container-based installation](#)

[Create an API token for the Ansible MCP server](#)

[Connect an AI agent to the Ansible MCP server](#)

[Technology Preview Features Support Scope](#)

[Red Hat Privacy Statement](#)

Deploy the Ansible MCP server on a container-based installation

As an organization administrator, you can deploy and configure an Ansible MCP server on a container-based installation of Ansible Automation Platform 2.6. Use the following procedure to deploy and configure the Ansible MCP server.

Before you begin

- You have a valid Ansible Automation Platform 2.6 subscription.

Procedure

1. Configure the Ansible MCP server variables in the inventory file:
 - a. Create an `[ansiblemcp]` group and add a host for the Ansible MCP server.
 - b. Add the following installation variables to your inventory file under the `[all:vars]` group:
 - `mcp_allow_write_operations` : Use to grant read-only or read-write permissions to the external AI tool.
 - `mcp_ignore_certificate_errors` : Use to bypass SSL/TLS certificate validation.
 - c. To make your system trust a self-signed custom certificate, add the following required variables to your inventory file:
 - `mcp_tls_cert` : Path to TLS certificate
 - `mcp_tls_key` : Path to TLS key

For information about using your own TLS certificates and keys, see [Configuring custom TLS certificates](#). For information about required and optional variables, see Appendix: Ansible MCP server variables.

```
# This is the list of inventory file variables required to deploy
an Ansible MCP server on a container-based installation.

# This section is for the Ansible MCP server host
# -----

[ansiblemcp]
aap.example.com

# This section is for Ansible MCP server permissions
# -----

[all:vars]

mcp_allow_write_operations=false <To enable read-write access, set
the "mcp_allow_write_operations" variable to "true">

mcp_ignore_certificate_errors=false

mcp_tls_cert= <path to tls certificate>

mcp_tls_key= <path to tls key>
```

2. Run the `install` [playbook to install containerized Ansible Automation Platform](#).

Result

Check the pods after installation is complete. You should see an `ansiblemc` pod running with the following command:

```
$ podman ps
```

Next steps

1. Obtain the location of the Ansible MCP server:
 - The service is exposed on port 8448 of the host, and HTTPS is enabled.
 - The example above deploys the MCP server on `aap.example.com`, so the service base URL will be `https://aap.example.com:8448`.
2. [Create an API token for the Ansible MCP server.](#)

Deploy the Ansible MCP server on operator-based installation

As an organization administrator, you can deploy and configure an Ansible MCP server on an operator-based installation of Ansible Automation Platform 2.6. Use the following procedure to deploy and configure the Ansible MCP server.

Before you begin

- You have a valid Ansible Automation Platform 2.6 subscription.

Procedure

1. Log in to Red Hat OpenShift Container Platform as an administrator.
2. Navigate to the namespace where you want to install the MCP server.
3. Select **Operators > Installed Operators**.
4. From the list of installed operators, select **Ansible Automation Platform**.
5. In the Ansible Automation Platform tile, click **Create instance**.
6. From the **Configure via** field, select the **Form** view, then provide the instance name. For example, `aap-mcp`.
7. Select the **YAML** view, and under the `spec:` section, add the `mcp` component:

```
spec:
  mcp:
    disabled: false
    allow_write_operations: false
```

8. Use the `allow_write_operations` variable to configure the operational access level of the Ansible MCP server:
9. Click **Create**. The Ansible MCP server is created.
10. Optional: If you changed the permissions of the Ansible MCP server after it was created and deployed, you must delete the **AnsibleMCPServer** custom resource and recreate it.

Perform the following steps:

- a. Go to the Ansible Automation Platform portal.
- b. Under **Resources**, search for the **AnsibleMCPServer** custom resource.
- c. Select the active **AnsibleMCPServer** instance. An active AnsibleMCPServer instance is identified by the `-mcp` suffix appended to the Ansible Automation Platform custom resource name.
- d. Select the **Settings menu** on the right side of the instance, and then click **Delete AnsibleMCPServer**.
- e. After the reconciliation process completes, the existing MCP server instance is deleted, and a new Ansible MCP server instance is created.

Result

1. Navigate to **Workloads > Deployments**.
2. Check that the deployment you created is listed there. For example: `aap-mcp`.
3. Check one of the pod's logs and verify there are no errors.

Next steps

1. Obtain the following information:
 - **Ansible Automation Platform login screen URL:**
 - a. Navigate to **Networking > Routes**.
 - b. For the Ansible Automation Platform deployment, click the **Copy icon** in the **Location** field. This is the URL of the Ansible Automation Platform login screen.
 - **Ansible Automation Platform administrator password:**
 - a. Navigate to **Workloads > Secrets** and click `aap-admin-password`.
 - b. Click **Reveal values** and then use the **Copy** icon to save the Ansible Automation Platform administrator password for future use.
 - **Ansible MCP server URL:**
 - a. Navigate to **Networking > Routes**.
 - b. For the deployment you recently created (`aap-mcp`), click the **Copy** icon from the **Location** field. This is the URL required to configure your AI agent to connect to the Ansible MCP server.
2. [Create an API token for the Ansible MCP server.](#)

Create an API token for the Ansible MCP server

Create an API token for your Ansible Automation Platform instance, so you can use it to connect with your preferred AI agent. The AI tool will inherit the user's permissions for API token-based authentication.

Before you begin

- Your organization administrator has deployed an Ansible MCP server.

Procedure

1. From the navigation panel, select **Access Management > Users**.
 2. Select the username for your user profile to configure OAuth 2 tokens.
 3. Select the **Tokens** tab. When no tokens are present, the **Tokens** screen prompts you to add them.
 4. Click **Create token**, and provide the following details:
 - **Application:** Enter the name of the application with which you want to associate your token. Alternatively, you can search for it by clicking **Browse**. This opens a separate window that enables you to choose from the available options. Select **Name** from the filter list to filter by name if the list is extensive.
- NOTE:**
To create a Personal Access Token (PAT) that is not linked to any application, leave the **Application** field blank.
- **Description:** (Optional) Provide a short description for your token.
 - **Scope:** (Required) Specify the level of access you want this token to have. The scope of an OAuth 2 token can be set as one of the following:
 - **Write:** Allows requests sent with this token to add, edit, and delete resources in the system.
 - **Read:** Limits actions to read only. The write scope includes the read scope.
 5. Click **Create token**. The token information is displayed.
 6. On the token information page that appears, click the **Copy icon** and save the token for future use.

IMPORTANT:

This will be the only time the token is displayed. Therefore, ensure that you save the token for future use.

Result

You can verify that the application now shows the user with the appropriate token by selecting the **Tokens** tab on the **Application Details** page:

1. From the navigation panel, select **Access Management > OAuth Applications**.
2. Select the application you want to verify from the **Applications** list view.
3. Select the **Tokens** tab.
Your token should be displayed in the list of tokens associated with the application you chose.

Next steps

- [Connect an external AI agent to the Ansible MCP server](#)

Connect an AI agent to the Ansible MCP server

Use the API token of the Ansible MCP server to connect it with your preferred AI agent, such as Claude, Cursor, or ChatGPT.

Before you begin

- An Ansible MCP server is deployed on your Ansible Automation Platform 2.6 environment.
- An API token is created for your Ansible MCP server.

Procedure

1. Go to the AI tool that you want to connect to the Ansible Automation Platform.
2. Follow your AI client's instructions to configure the MCP server settings.
Typically, you must specify the MCP server configurations in the `mcp.json` file.
3. When configuring the `mcp.json` file, add the Ansible MCP server URL in the following format:

```
<Ansible MCP server URL>/<toolset>/mcp
```

Key:

- **Ansible MCP server URL** = The URL of the Ansible MCP server. For example, `https://api.example.com/`.
To obtain the Ansible MCP server URL, contact your organization administrator.
- **Toolset** = The toolset that you want to connect to. For example, `job_management`, `inventory_management`, `system_monitoring`, `user_management`, `security_compliance`, and `platform_configuration`.
- **Token** = The API token of the Ansible MCP server.
Use the following format to add details about your Ansible MCP server in the `mcp.json` file:

```
{
  "mcpServers": {
    "aap-mcp-job-mgmt": {
      "type": "http",
      "url": "https://api.example.com/job_management/mcp",
      "headers": {
        "Authorization": "Bearer ${env:MY_SERVICE_TOKEN}"
      }
    },
    "aap-mcp-inventory-mgmt": {
      "type": "http",
      "url": "https://api.example.com/inventory_management/mcp",
      "headers": {
        "Authorization": "Bearer ${env:MY_SERVICE_TOKEN}"
      }
    },
    "aap-mcp-system-monitor": {
      "type": "http",
      "url": "https://api.example.com/system_monitoring/mcp",
      "headers": {
        "Authorization": "Bearer ${env:MY_SERVICE_TOKEN}"
      }
    },
    "aap-mcp-user-mgmt": {
      "type": "http",
      "url": "https://api.example.com/user_management/mcp",
      "headers": {
        "Authorization": "Bearer ${env:MY_SERVICE_TOKEN}"
      }
    }
  }
}
```

```

    }
  },
  "aap-mcp-security": {
    "type": "http",
    "url": "https://api.example.com/security_compliance/mcp",
    "headers": {
      "Authorization": "Bearer ${env:MY_SERVICE_TOKEN}"
    }
  },
  "aap-mcp-platform-config": {
    "type": "http",
    "url": "https://api.example.com/platform_configuration/mcp",
    "headers": {
      "Authorization": "Bearer ${env:MY_SERVICE_TOKEN}"
    }
  }
}
}
}

```

IMPORTANT:

Use a concise MCP server name, ideally limited to 20 characters. This is because AI agents combine the MCP server name with the tool name to create a unique identifier, and most AI agents enforce a 64-character limit on this combined identifier.

Result

- Verify that the AI tool successfully connects to the Ansible Automation Platform MCP server using the API token.
In your AI agent's chat window, enter a prompt like `What MCP tools are available for my Ansible Automation Platform?`. The AI agent should return a response with a list of tools that are enabled for the Ansible Automation Platform MCP server.

Next steps

- Open a new chat in your AI agent, and enter your prompt.
For example: `Give me a list of my Ansible Automation Platform jobs.` A list of all your Ansible Automation Platform jobs is displayed in the AI agent's chat window.

Related information

[Connect Claude Code to tools via MCP](#)

[Configure Model Context Protocol \(MCP\) in Cursor](#)

[Use MCP servers in VS Code](#)
[Configuring MCP Servers - Cline](#)

Troubleshoot Ansible MCP server errors

This section contains information to help you diagnose and resolve issues with deploying the Ansible MCP server and connecting it to an external AI agent.

API output format rejected with 406 Status Code

Issue: Ansible Automation Platform rejects an API request (for example, retrieving job stdout) with an HTTP `406` status code if the MCP server's requested output is not in `JSON` format.

Workaround: To obtain the output in a specific format, instruct your AI tool to use `JSON` format first. You can then transform the `JSON` output into your desired format.

User requests rejected with 400 status code

Issue: The Ansible MCP server may reject user requests from the external AI tool with `400 Bad Request` status code. This error is encountered when the Ansible Automation Platform uses a self-signed certificate.

Workaround: Configure the Ansible MCP server to ignore certificate errors using the following steps:

- For container-based installation: Set the value of variable `mcp_ignore_certificate_errors` to `true`.
- For operator-based installation:
Add the `IGNORE_CERTIFICATE_ERRORS` setting to the `mcp:` section of **AnsibleAutomationPlatform** custom resource in the following format:

```
spec:
  mcp:
    extra_settings:
      - setting: IGNORE_CERTIFICATE_ERRORS
        value: true
```

Ansible MCP server permissions are changed post deployment

Issue: If you changed the permissions of the Ansible MCP server after it was created and deployed, you must delete the **AnsibleMCPServer** custom resource and recreate it.

Workaround: Perform the following steps:

1. Navigate to the Ansible Automation Platform portal.
2. Under **Resources**, search for the **AnsibleMCPServer** custom resource.
3. Select the active **AnsibleMCPServer** instance. An active AnsibleMCPServer instance is identified by the `-mcp` suffix appended to the Ansible Automation Platform custom resource name.
4. Select the **Settings menu** (3-dot menu icon) on the right side of the instance, then click **Delete AnsibleMCPServer**.
5. After the reconciliation process is completed, the existing Ansible MCP server instance is deleted and a new Ansible MCP server instance is created.

Deploy the intelligent assistant

This section provides information about the procedures involved in deploying the intelligent assistant on OpenShift Container Platform.

Create a chatbot configuration secret

Create a configuration secret for the intelligent assistant, so that you can connect the intelligent assistant to the Ansible Automation Platform operator.

Before you begin

- [You have installed and configured the Ansible Automation Platform operator.](#)

Procedure

1. Log in to Red Hat OpenShift Container Platform as an administrator.
2. Navigate to **Workloads > Secrets**.
3. From the **Projects** list, select the namespace that you created when you installed the Ansible Automation Platform operator.
4. Click **Create > Key/value secret**.
5. In the **Secret name** field, enter a unique name for the secret. For example, `chatbot-configuration-secret`.

6. Add the following keys and their associated values individually:

Key	Value
Settings for all LLM setups	
<code>chatbot_model</code>	Enter the LLM model name that is configured on your LLM setup.
<code>chatbot_url</code>	Enter the inference API base URL on your LLM setup. For example, <code>https://your_inference_api/v1</code> . If you are using Microsoft Azure OpenAI, then set the base URL to <code>https://your_inference_api/openai/v1</code> .
<code>chatbot_token</code>	Enter the API token or the API key. This token is sent along with the authorization header when an inference API is called.
<code>chatbot_llm_provider_type</code>	<i>Optional</i> Enter the value as per the provider type of your LLM setup: • Red Hat Enterprise Linux AI: <code>rhelai_vllm</code> • Red Hat OpenShift AI: <code>rhoai_vllm</code> • OpenAI: <code>openai</code> • Microsoft Azure OpenAI: <code>azure_openai</code>
<code>chatbot_model_config_extras</code>	<i>Optional</i> Use this field to pass a JSON dictionary of extra parameters to pass directly to the model provider, for settings not covered by other standard fields. For example, you can specify a parameter <code>api_version</code> for Microsoft Azure OpenAI in the JSON format <code>'{"api_version": "<your API version>"}</code>.
<code>chatbot_agent_config_extras</code>	<i>Optional</i> Use this parameter to customize agent behavior, such as controlling the temperature of the LLM. For example, <code>'{"chatbot_temperature_override": 1}'</code>.
Additional settings for MCP server configuration	
• <code>aap_gateway_url</code>	Configure a Model Context Protocol (MCP) server that interfaces with the intelligent assistant.

Key	Value
<code>aap_controller_url</code>	The values <code>aap_gateway_url</code> and <code>aap_controller_url</code> are internal URLs accessible to the platform gateway and automation controller services on the OpenShift cluster. For example, if the name of your Ansible Automation Platform custom resource is <code>myaap</code>, these URLs will be: <code>aap_gateway_url</code>: <code>http://myaap</code> <code>aap_controller_url</code>: <code>http://myaap-controller-service</code> For MCP server configuration: - If none of these values are configured, no MCP server is provisioned or registered with the underlying LLM's tool at runtime. - If you configure the <code>aap_gateway_url</code> value only, the Ansible Lightspeed Service MCP server is provisioned. Authentication attempts to use the JSON Web Token (JWT) token associated with the user's authenticated context. - If you configure both values <code>aap_gateway_url</code> and <code>aap_controller_url</code>, the Ansible Lightspeed Service MCP server and Ansible Automation Platform Controller Service MCP server are both configured. Authentication attempts to use the JWT token associated with the user's authenticated context.

7. Click **Create**. The chatbot authorization secret is successfully created.

Examples of chatbot configuration secrets

The following snippet shows a few examples of secrets configuration for different LLM models.

```
# Example of a secret configuration for Red Hat OpenShift AI
apiVersion: v1
kind: Secret
metadata:
  name: chatbot-configuration-secret
  namespace: aap
stringData:
  chatbot_llm_provider_type: rhoai_vllm
  chatbot_url: https://llm-dev-wisdom-model-staging.apps.stage2-
west.v2dz.pl.openshiftapps.com/v1
  chatbot_model: granite-3.3-8b-instruct
  chatbot_token: <token number>
```

```
# Example of a secret configuration for OpenAI
apiVersion: v1
kind: Secret
metadata:
  name: chatbot-configuration-secret
  namespace: aap
stringData:
  chatbot_llm_provider_type: openai
  chatbot_url: https://api.openai.com/v1
  chatbot_model: gpt-4o-mini
  chatbot_token: <token number>
```

```
# Example of a secret configuration for Microsoft Azure OpenAI
apiVersion: v1
kind: Secret
metadata:
  name: chatbot-configuration-secret
  namespace: aap
stringData:
  chatbot_llm_provider_type: azure_openai
  chatbot_url: https://ols-test.openai.azure.com
  chatbot_model: gpt-4o-mini
  chatbot_token: <token number>
  chatbot_model_config_extras: '{"api_version": "2025-01-01-preview"}'
```

Update the YAML file of the Ansible Automation Platform operator

After you create the chatbot authorization secret, you must update the YAML file of the Ansible Automation Platform operator to use the secret.

Procedure

1. Log in to Red Hat OpenShift Container Platform as an administrator.
2. Navigate to **Operators > Installed Operators**.
3. From the list of installed operators, select the **Ansible Automation Platform** operator.
4. Locate and select the **Ansible Automation Platform** custom resource, and then click the required app.
5. Select the **YAML** tab.
6. Scroll the text to find the `spec:` section, and add the following details under the `spec:` section:

```
spec:
  lightspeed:
    disabled: false
    chatbot_config_secret_name: <name of your chatbot configuration secret>
```

7. Click **Save**. The intelligent assistant service takes a few minutes to set up.

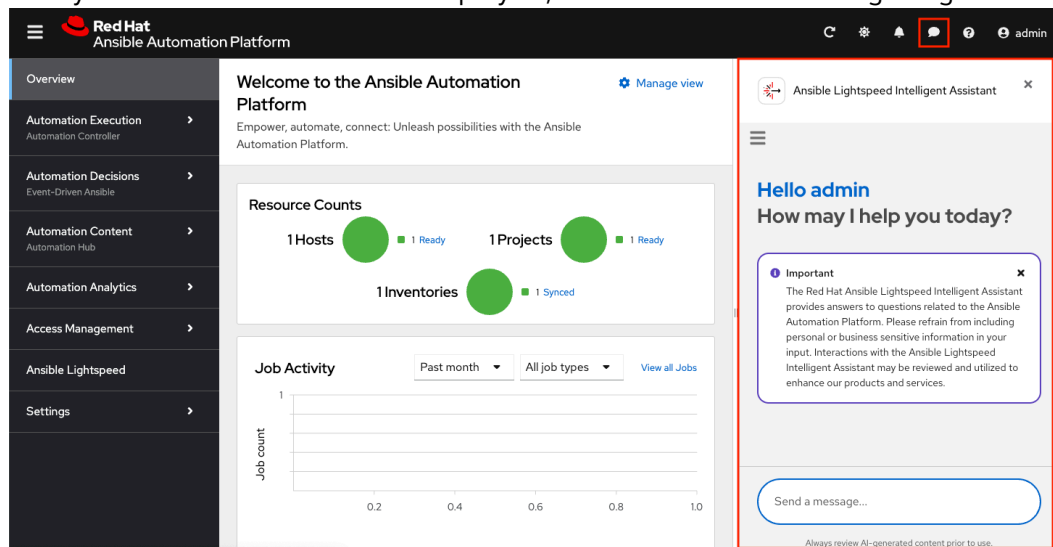
NOTE:

Upgrading from Ansible Automation Platform 2.5 to 2.6.1 enables HTTPS and enables TLS by default for internal communication between the Ansible Lightspeed API and the intelligent assistant pod. Following the upgrade to Ansible Automation Platform 2.6.1, the intelligent assistant will be unavailable for approximately 60 seconds while its pod restarts.

Result

1. Verify that the chat interface service is running successfully:
 - a. Navigate to **Workloads > Pods**.
 - b. Filter with the term **api** and ensure that the following APIs are displayed in **Running** status:
 - `myaap-lightspeed-api-<version number>`
 - `myaap-lightspeed-chatbot-api-<version number>`
2. Verify the MCP server configuration if you specified either `aap_gateway_url` or `aap_controller_url` parameter:
 - Open the `lightspeed-chatbot-api` pod and click the **Containers** section.
 - If the `ansible-mcp-lightspeed` container is displayed, the Ansible Lightspeed MCP server is running.
 - If the `ansible-mcp-controller` container is displayed, the Ansible Automation Platform Controller Service MCP server is running.
3. Verify that the chat interface is displayed on the Ansible Automation Platform:
 - a. Access the Ansible Automation Platform:
 - i. Navigate to **Operators > Installed Operators**.
 - ii. From the list of installed operators, click **Ansible Automation Platform**.
 - iii. Locate and select the **Ansible Automation Platform** custom resource, and then click the app that you created.
 - iv. From the **Details** tab, record the information available in the following fields:
 - **URL**: This is the URL of your Ansible Automation Platform instance.
 - **Gateway Admin User**: This is the username to log into your Ansible Automation Platform instance.
 - **Gateway Admin password**: This is the password to log into your Ansible Automation Platform instance.
 - v. Log in to the Ansible Automation Platform using the URL, username, and password that you recorded earlier.
 - b. Access the intelligent assistant:

- i. Click the intelligent assistant icon  that is displayed at the top right corner of the taskbar.
- ii. Verify that the chat interface is displayed, as shown in the following image:



Change your LLM model

If you have already deployed Ansible Lightspeed intelligent assistant but want to change your LLM model, you can create a new chatbot configuration secret for the new LLM model.

Alternatively, if you want to use the same chatbot configuration secret, you must delete and redeploy the Ansible Lightspeed intelligent assistant.

- To create and use a new chatbot configuration secret:
 - a. [Create a new chatbot configuration secret](#) with a different name for the new LLM model.
 - b. [Update the YAML file of the Ansible Automation Platform operator](#) with the new chatbot configuration secret name.
 The Ansible Automation Platform operator detects the new configuration and redeploys the Ansible Lightspeed intelligent assistant.
 - c. Verify that the chat interface service is running successfully. See the verification steps mentioned in the topic [Update the YAML file of the Ansible Automation Platform operator](#).

IMPORTANT:

Do not update the existing chatbot configuration secret with the new LLM model, as the reconciliation logic does not check the updates made to the secret.

- To use the same chatbot secret by deleting and redeploying the Ansible Lightspeed intelligent assistant:
 - a. Disable the Ansible Lightspeed operator instance:
 - a. Navigate to **Operators > Installed Operators**.
 - b. From the list of installed operators, select **Ansible Automation Platform**.
 - c. Locate and select the Ansible Automation Platform custom resource.
 - d. Select the **YAML** tab and under the `spec:` section for `lightspeed` category, specify `disabled:true`.
 - e. Click **Save**.
 - b. Delete the Ansible Lightspeed operator instance:
 - a. Navigate to **Operators > Installed Operators**.
 - b. From the list of installed operators, select **Ansible Lightspeed** and delete the operator.
 - c. Re-enable the Ansible Automation Platform instance:
 - a. Navigate to **Operators > Installed Operators**.
 - b. From the list of installed operators, select **Ansible Automation Platform**.
 - c. Locate and select the Ansible Automation Platform custom resource.
 - d. Select the **YAML** tab and under the `spec:` section for `lightspeed` category, specify `disabled:false`.
 - e. Click **Save**.

Use the Ansible Lightspeed intelligent assistant

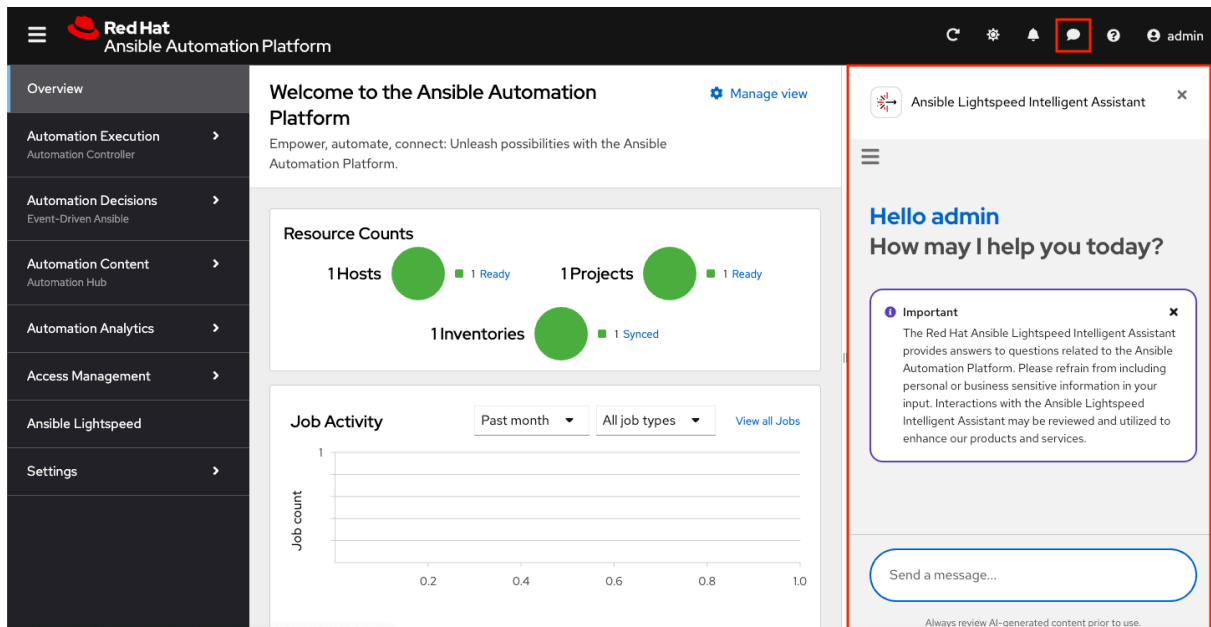
After you deploy the Ansible Lightspeed intelligent assistant, all Ansible users within the organization can access and use the chat interface to ask questions and receive information about the Ansible Automation Platform.

Access the Ansible Lightspeed intelligent assistant

1. Log in to the Ansible Automation Platform.

- Click the Ansible Lightspeed intelligent assistant icon  that is displayed at the top right corner of the taskbar.

The Ansible Lightspeed intelligent assistant window opens with a welcome message, as shown in the following image:



Use the Ansible Lightspeed intelligent assistant

You can perform the following tasks:

- Ask questions in the prompt field and get answers about the Ansible Automation Platform

NOTE:

If you are using an IBM Granite 3.3 series AI model, you might experience a delay of about one minute when waiting for a chat response. To resolve this error, restart the chat session.

- View the chat history of all conversations in a chat session.
- Search the chat history using a user prompt or answer. The chat history is deleted when you close an existing chat session or log out from the Ansible Automation Platform.
- Restore an earlier chat by clicking the relevant entry from the chat history.
- Give feedback on the quality of the chat answers, by clicking the **Thumbs up** or **Thumbs down** icon.

- Copy and record the answers by clicking the **Copy** icon.
- Change the mode of the virtual assistant to dark or light mode, by clicking the **Sun** icon



- from the top right corner of the toolbar.
- Clear the context of an existing chat by using the **New chat** button in the chat history.
- Close the chat interface while working on the Ansible Automation Platform.

Deploy Red Hat Ansible Lightspeed on containerized Ansible Automation Platform

As an organization administrator, you can deploy and use Red Hat Ansible Lightspeed when you perform a new container-based installation of Ansible Automation Platform 2.6.

Overview

You can deploy and use Red Hat Ansible Lightspeed when you perform a new container-based installation of Ansible Automation Platform 2.6, or upgrade from containerized Ansible Automation Platform 2.5 to 2.6.

Red Hat Ansible Lightspeed includes two main components that enhance your automation experience with generative artificial intelligence (AI):

- Ansible Lightspeed intelligent assistant: An AI-powered chat interface embedded within the Ansible Automation Platform.
- Ansible Lightspeed coding assistant: A generative AI service that helps developers create Ansible content more efficiently and accurately.

IMPORTANT:

Red Hat does not collect any telemetry data from your interactions with Red Hat Ansible Lightspeed.

Ansible Lightspeed intelligent assistant

Ansible Lightspeed intelligent assistant is an intuitive chat interface embedded in the Ansible Automation Platform, and uses generative artificial intelligence (AI) to answer questions about the platform.

The Ansible Lightspeed intelligent assistant interacts with users in English, and uses Large Language Models (LLMs) to generate quick, accurate, and personalized responses. These responses empower Ansible users to work more efficiently, thereby improving productivity and the overall quality of their work.

To use the Ansible Lightspeed intelligent assistant, you need:

- A valid subscription to Ansible Automation Platform.
- Deployment of an LLM service that is hosted on one of these platforms: Red Hat Enterprise Linux AI, Red Hat OpenShift AI, or Red Hat AI Inference Server.

Integration with MCP server

Ansible Lightspeed intelligent assistant integration with the Model Context Protocol (MCP) server is available as a Technology Preview release. MCP is an open protocol that enables applications to give real-time context to LLMs.

This integration enables the Ansible Lightspeed intelligent assistant to request and receive the latest information from external resources, and give more relevant, dynamically-sourced answers when responding to your questions. To set up this integration, you need to specify the MCP server variables when configuring the Red Hat Ansible Lightspeed variables in the inventory file.

NOTE:

Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production. These features give early access to upcoming product features, enabling customers to test functionality and leave feedback during the development process.

Ansible Lightspeed coding assistant

The Ansible Lightspeed coding assistant is a generative AI service that works with IBM watsonx Code Assistant to help developers create and maintain Ansible content more efficiently. It can generate code recommendations for:

- Single-task or multi-task recommendations
- Playbooks with explanations
- Roles with explanations

Ansible Lightspeed coding assistant generates code recommendations that adhere to Ansible best practices, while IBM watsonx Code Assistant fine-tunes models to improve the accuracy of suggested recommendations by using your organization's existing Ansible content. This integration produces more accurate, reliable, and workflow-integrated automation code. It also shortens the onboarding time for new Ansible developers and improves team productivity.

To use the Ansible Lightspeed coding assistant, you need:

- A valid subscription to Red Hat Ansible Automation Platform.
- A valid subscription to IBM watsonx Code Assistant.

Deployment models

The Ansible Lightspeed coding assistant supports two deployment models. No telemetry data is collected in either configuration.

- **On-premise deployment**

Both Red Hat Ansible Lightspeed and the IBM watsonx Code Assistant model (IBM Cloud Pak for Data) are on-premise deployments.

- **Hybrid deployment**

Red Hat Ansible Lightspeed is an on-premise deployment, while IBM watsonx Code Assistant model is a cloud deployment.

A hybrid deployment model provides the following benefits:

- Flexibility to choose an environment that best suits your organizational needs.
- Integrated authentication by using the Ansible Automation Platform for user authentication and removing the need for a separate Red Hat cloud login.
- Regional choice for organizations to deploy Red Hat Ansible Lightspeed in their preferred geographical region.

Ansible Automation Platform requirements

- Licensing requirements:
 - A valid Ansible Automation Platform subscription.
 - Administrator privileges for the Ansible Automation Platform.
- Additional requirements for Ansible Lightspeed coding assistant:
 - A valid subscription to IBM watsonx Code Assistant (for on-premise deployment), or IBM watsonx Code Assistant for Red Hat Ansible Lightspeed on Cloud Pak for Data (for hybrid deployment).
 - An API key and a model ID from IBM watsonx Code Assistant.
 - VS Code version 1.70.1 or later.

- Additional requirements for Ansible Lightspeed intelligent assistant:
 - Deployment of an LLM service that is hosted on one of these platforms: Red Hat Enterprise Linux AI, Red Hat OpenShift AI, or Red Hat AI Inference Server.

Large Language Model (LLM) provider requirements

You must have configured an LLM provider that you will use before deploying the Ansible Lightspeed intelligent assistant. An LLM is a type of machine learning model that can interpret and generate human-like language. When an LLM is used with the Ansible Lightspeed intelligent assistant, the LLM can interpret questions accurately and provide helpful answers in a conversational manner.

Your LLM must have tool calling enabled to handle tool-related requests. Tool calling allows the assistant to interact with platform services and execute complex workflows.

Ansible Lightspeed intelligent assistant can rely on the following LLM providers:

- **Red Hat LLM providers:**

- **Red Hat Enterprise Linux AI**

You can configure Red Hat Enterprise Linux AI as the LLM provider. As the Red Hat Enterprise Linux is in a different environment than the Ansible Lightspeed deployment, the model deployment must allow access using a secure connection.

Ansible Lightspeed intelligent assistant supports vLLM Server. When self-hosting an LLM with Red Hat Enterprise Linux AI, you can use vLLM Server as the inference engine.

- **Red Hat OpenShift AI**

You must deploy an LLM on the Red Hat OpenShift AI single-model serving platform that uses the Virtual Large Language Model (vLLM) runtime. If the model deployment lives in a different OpenShift environment than the Ansible Lightspeed deployment, include a route to expose the model deployment outside the cluster.

Ansible Lightspeed intelligent assistant supports vLLM Server. When self-hosting an LLM with Red Hat OpenShift AI, you can use vLLM Server as the inference engine.

NOTE:

For configurations with Red Hat Enterprise Linux AI or Red Hat OpenShift AI, you must host your own LLM provider instead of using a SaaS LLM provider.

- **Red Hat AI Inference Server**

You can deploy an LLM by using Red Hat AI Inference Server as your inference runtime. Red Hat AI Inference Server supports vLLM runtimes for efficient model serving and can be configured to work with Ansible Lightspeed intelligent assistant.

If the Red Hat AI Inference Server deployment is in a different environment than the Ansible Lightspeed deployment, ensure the model deployment allows access by using a secure connection and configure appropriate network routing.

Ansible Lightspeed intelligent assistant supports vLLM Server when self-hosting an LLM with Red Hat AI Inference Server as the inference engine.

- **Third-party LLM providers:**

- **OpenAI**

To use OpenAI with the Ansible Lightspeed intelligent assistant, you need access to the OpenAI API platform.

- **Microsoft Azure OpenAI**

To use Microsoft Azure with the Ansible Lightspeed intelligent assistant, you need access to Microsoft Azure OpenAI.

Process to deploy Red Hat Ansible Lightspeed on a container-based installation

Task	Description
Deploy Red Hat Ansible Lightspeed during a container-based installation of Ansible Automation Platform	An Ansible Automation Platform administrator who wants to deploy Red Hat Ansible Lightspeed for all Ansible users in the organization. Perform the following tasks: 1. Configure the Red Hat Ansible Lightspeed variables in the inventory file. 2. Install or upgrade to containerized Ansible Automation Platform 2.6. 3. If you want to install the Ansible Lightspeed coding assistant, configure the Ansible VS Code extension. 4. Optional: Change your LLM model if you want to use a different LLM provider after deploying Red Hat Ansible Lightspeed.
Access and use the Ansible Lightspeed intelligent assistant	All Ansible users within the organization who want to use the Ansible Lightspeed intelligent assistant to get answers to their questions about the Ansible Automation Platform.
Access and use the Ansible Lightspeed coding assistant	All Ansible users within the organization who want to use the coding assistant to develop Ansible content: • Single task or multitask recommendations • Create playbooks and view playbook explanations • Create roles and view role explanations

Related information

[Configure Red Hat Ansible Lightspeed variables](#)[Install containerized Ansible Automation Platform](#)[Upgrade your containerized deployment of Ansible Automation Platform](#)[Configure the Ansible VS Code extension](#)[Change your LLM model](#)[Deploy and use the Ansible Lightspeed intelligent assistant](#)[Develop Ansible content with Red Hat Ansible Lightspeed](#)[Single task or multitask recommendations](#)[Create playbooks and view playbook explanations](#)[Create roles and view role explanations](#)[Technology Preview Features Support Scope](#)[Optional: Allowing access to a model from a secure endpoint](#)[About the single-model serving platform](#)[Red Hat AI Inference Server documentation](#)[OpenAI API platform](#)[Microsoft Azure OpenAI](#)

Configure Red Hat Ansible Lightspeed variables

To deploy Red Hat Ansible Lightspeed, configure the required installation variables in your inventory file.

Procedure

1. Add the required installation variables to your inventory file under the `[all: vars]` group.
2. You will also need to add specific variables to enable the Ansible Lightspeed coding assistant, the Ansible Lightspeed intelligent assistant, and the MCP server integration. Refer to the [Appendix: Red Hat Ansible Lightspeed variables](#) for information about required and optional variables.

```
# This is the list of inventory file variables required to deploy Red Hat
Ansible Lightspeed on a containerized installation.
```

```
# Consult the docs if you are unsure what to add.
```

```
# For information about required and optional variables, refer to the
Appendix: Red Hat Ansible Lightspeed variables
```

```
# https://docs.redhat.com/en/documentation/
red_hat_automation_platform/2.6/html/containerized_installation/
appendix-inventory-files-vars#lightspeed-variables
```

```
# This section is for your Red Hat Ansible Lightspeed host
```

```
# -----
```

```
[ansiblelightspeed]
```

```
aap.example.com
```

```

# This section is for Red Hat Ansible Lightspeed deployment
# -----

lightspeed_admin_user: <set your own>
lightspeed_admin_password: <set your own>
lightspeed_admin_email: <set your own>
lightspeed_pg_host: <set your own>
lightspeed_pg_password: <set your own>


# This section is to configure Ansible Lightspeed intelligent assistant
# -----

lightspeed_chatbot_model_url: <set your own>
lightspeed_chatbot_model_api_key: <set your own>
lightspeed_chatbot_model_id: : <set your own>
lightspeed_chatbot_default_provider: 'rhoai'
lightspeed_chatbot_model_extra_settings: {}
lightspeed_chatbot_agent_extra_settings: {}

# If you want to use Microsoft Azure OpenAI as the LLM provider, specify the
lightspeed_chatbot_model_extra_settings value as '{"api_type": "openai"}', and the
lightspeed_chatbot_model_url value to 'https://your_inference_api/openai/v1'.


# This section is to configure Ansible Lightspeed intelligent assistant with
MCP server integration
#
-----

lightspeed_mcp_controller_enabled: false
lightspeed_mcp_lightspeed_enabled: false


# This section is to configure Ansible Lightspeed coding assistant
# -----

lightspeed_wca_model_type: 'wca'
lightspeed_wca_model_url: 'https://api.dataplatform.cloud.ibm.com'
lightspeed_wca_model_verify_ssl: true
lightspeed_wca_model_enable_anonymization: true
lightspeed_wca_health_check: true

```

Next steps

- [Installing containerized Ansible Automation Platform](#)

Red Hat Ansible Lightspeed variables

Configure Red Hat Ansible Lightspeed by setting inventory file variables during installation. Use this reference to determine which variables to set for your deployment requirements.

Red Hat Ansible Lightspeed variables

Inventory file variables for Red Hat Ansible Lightspeed.

RPM variable name	Container variable name	Description	Required or optional	Default
N/A	<code>lightspeed_admin_password</code>	Red Hat Ansible Lightspeed 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	
N/A	<code>lightspeed_admin_user</code>	Username used to identify and create the Red Hat Ansible Lightspeed admin user.	Optional	<code>admin</code>
N/A	<code>lightspeed_chat_rate_throttle</code>	Chat rate throttle.	Optional	<code>10/minute</code>
N/A	<code>lightspeed_nginx_client_max_body_size</code>	Maximum allowed size for data sent to Red Hat Ansible Lightspeed through NGINX.	Optional	<code>5m</code>
N/A	<code>lightspeed_nginx_disable_hsts</code>	Controls whether HTTP Strict Transport Security (HSTS) is enabled or disabled for Red Hat Ansible Lightspeed. Set	Optional	<code>false</code>

RPM variable name	Container variable name	Description	Required or optional	Default
		this variable to <code>true</code> to disable HSTS.		
N/A	<code>lightspeed_nginx_disable_https</code>	Controls whether HTTPS is enabled or disabled for Red Hat Ansible Lightspeed. Set this variable to <code>true</code> to disable HTTPS.	Optional	<code>false</code>
N/A	<code>lightspeed_nginx_hsts_max_age</code>	Maximum duration (in seconds) that HTTP Strict Transport Security (HSTS) is enforced for Red Hat Ansible Lightspeed.	Optional	<code>63072000</code>
N/A	<code>lightspeed_nginx_http_port</code>	Port number that Red Hat Ansible Lightspeed listens on for HTTP requests.	Optional	<code>8084</code>
N/A	<code>lightspeed_nginx_https_port</code>	Port number that Red Hat Ansible Lightspeed listens on for HTTPS requests.	Optional	<code>8447</code>
N/A	<code>lightspeed_nginx_https_protocols</code>	Protocols that Red Hat Ansible Lightspeed will support when handling HTTPS traffic.	Optional	<code>[TLSv1.2, TLSv1.3]</code>
N/A	<code>lightspeed_nginx_user_headers</code>	Custom Nginx headers. List of additional NGINX headers to add to Red Hat Ansible Lightspeed's NGINX configuration.	Optional	<code>[]</code>
N/A	<code>lightspeed_nginx_read_timeout</code>	Sets the HTTP timeout for end-user requests. The minimum value is <code>10</code> seconds.	Optional	<code>3600</code>

RPM variable name	Container variable name	Description	Required or optional	Default
N/A	<code>lightspeed_pg_certificate_auth</code>	Controls whether client certificate authentication is enabled or disabled on the Red Hat Ansible Lightspeed PostgreSQL database. Set this variable to <code>true</code> to enable client certificate authentication.	Optional	<code>false</code>
N/A	<code>lightspeed_pg_database</code>	Name of the PostgreSQL database used by Red Hat Ansible Lightspeed.	Optional	<code>lightspeed</code>
N/A	<code>lightspeed_pg_host</code>	Hostname of the PostgreSQL database used by Red Hat Ansible Lightspeed.	Required	
N/A	<code>lightspeed_pg_password</code>	Password for the Red Hat Ansible Lightspeed 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	
N/A	<code>lightspeed_pg_port</code>	Port number for the PostgreSQL database used by Red Hat Ansible Lightspeed.	Optional	<code>5432</code>
N/A	<code>lightspeed_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-</code>	Optional	<code>prefer</code>

RPM variable name	Container variable name	Description	Required or optional	Default
		<code>ca</code> , <code>require</code> , <code>prefer</code> , <code>allow</code> , <code>disable</code> .		
N/A	<code>lightspeed_pg_tls_cert</code>	Path to the PostgreSQL SSL/TLS certificate file for Red Hat Ansible Lightspeed.	Optional	
N/A	<code>lightspeed_pg_tls_key</code>	Path to the PostgreSQL SSL/TLS key file for Red Hat Ansible Lightspeed.	Optional	
N/A	<code>lightspeed_pg_username</code>	Username for the Red Hat Ansible Lightspeed PostgreSQL database user.	Optional	<code>lightspeed</code>
N/A	<code>lightspeed_secret_key</code>	Secret key value used by Red Hat Ansible Lightspeed to sign and encrypt data.	Optional	
N/A	<code>lightspeed_tls_cert</code>	Path to the SSL/TLS certificate file for Red Hat Ansible Lightspeed.	Optional	
N/A	<code>lightspeed_tls_key</code>	Path to the SSL/TLS key file for Red Hat Ansible Lightspeed.	Optional	
N/A	<code>lightspeed_tls_remote</code>	Denote whether the Red Hat Ansible Lightspeed 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>
N/A	<code>lightspeed_use_archive_compression</code>	Controls whether archive compression is enabled or disabled for Red Hat Ansible Lightspeed. You can control this functionality globally by using	Optional	<code>true</code>

RPM variable name	Container variable name	Description	Required or optional	Default
		<code>use_archive_compression</code> .		
N/A	<code>lightspeed_use_db_compression</code>	Controls whether database compression is enabled or disabled for Red Hat Ansible Lightspeed. You can control this functionality globally by using <code>use_db_compression</code> .	Optional	<code>false</code>

Ansible Lightspeed coding assistant variables

Inventory file variables for Ansible Lightspeed coding assistant.

RPM variable name	Container variable name	Description	Required or optional	Default
N/A	<code>lightspeed_wca_model_type</code>	IBM watsonx Code Assistant model deployment mode, cloud (<code>wca</code>) or on-premise (<code>wca-onprem</code>).	Optional	<code>wca</code>
N/A	<code>lightspeed_wca_model_url</code>	URL of the IBM watsonx Code Assistant model. For cloud deployment, the URL could be <code>https://api.dataplatform.test.cloud.ibm.com</code> .	Optional	
N/A	<code>lightspeed_wca_model_api_key</code>	API key of the IBM watsonx Code Assistant model that was generated during the model installation.	Required	
N/A	<code>lightspeed_wca_model_id</code>	ID of the IBM watsonx Code Assistant model.	Optional	

RPM variable name	Container variable name	Description	Required or optional	Default
N/A	<code>lightspeed_wca_model_verify_ssl</code>	Denotes whether or not to verify IBM watsonx Code Assistant's web certificates when making calls from Red Hat Ansible Lightspeed to itself during installation. Set to <code>false</code> to disable web certificate verification.	Optional	<code>true</code>
N/A	<code>lightspeed_wca_model_enable_anonymization</code>	Controls whether the anonymization of Personally Identifiable Information (PII) is enabled. PII information includes passwords, IP addresses, email addresses, and other sensitive data. When PII anonymization is enabled, users' personal information is modified to some generic values to protect their data and reduce the risk of data leaks. You can turn off the anonymization by specifying the value as <code>false</code> if you want to retain all original information as entered by users and improve the quality of the answers. If you set the value to <code>false</code> and the Ansible administrator is using Red Hat Ansible Lightspeed in hybrid mode (where the model	Optional	<code>true</code>

RPM variable name	Container variable name	Description	Required or optional	Default
		is in IBM watsonx Code Assistant in IBM Cloud) then their users' PII is sent to IBM Cloud.		
N/A	lightspeed_wca_model_username	For on-premise deployment only. The username you use to connect to an IBM Cloud Pak for Data deployment.	Optional	
N/A	lightspeed_wca_health_check	Enables or disables IBM watsonx Code Assistant health check.	Optional	true
N/A	lightspeed_wca_idp_url	For cloud deployment only. The IBM watsonx Code Assistant Identity Provider (IdP) URL.	Optional	
N/A	lightspeed_wca_idp_login	For cloud deployment only. The IBM watsonx Code Assistant Identity Provider (IdP) username.	Optional	
N/A	lightspeed_wca_idp_password	For cloud deployment only. The IBM watsonx Code Assistant Identity Provider (IdP) password.	Optional	

Ansible Lightspeed intelligent assistant variables

Inventory file variables for Ansible Lightspeed intelligent assistant.

RPM variable name	Container variable name	Description	Required or optional	Default
N/A	lightspeed_chatbot_model_url	The inference API base URL on your LLM setup. For example, <code>https://</code>	Optional	

RPM variable name	Container variable name	Description	Required or optional	Default
		<code>your_inference_api/v1</code> .If you are using Microsoft Azure OpenAI, then set the base URL to <code>https://your_inference_api/openai/v1</code> .		
N/A	<code>lightspeed_chatbot_model_verify_ssl</code>	Controls whether SSL/TLS certificate verification is enabled or disabled when making HTTPS requests.	Optional	<code>true</code>
N/A	<code>lightspeed_chatbot_default_provider</code>	The provider type of your LLM setup by using one of the following values: • Red Hat Enterprise Linux AI: <code>rhelai</code> • Red Hat OpenShift AI: <code>rhoai</code> • OpenAI: <code>openai</code> • Microsoft Azure OpenAI: <code>azure</code>	Optional	<code>rhoai</code>
N/A	<code>lightspeed_chatbot_model_extra_settings</code>	Use this parameter to pass a JSON dictionary of extra parameters to pass directly to the model provider, for settings not covered by other standard fields. If you want to use Microsoft Azure OpenAI as the LLM provider, specify the value as <code>'{"api_type": "azure"}'</code> .	Optional	<code>{}</code>

RPM variable name	Container variable name	Description	Required or optional	Default
N/A	<code>lightspeed_chatbot_agent_extra_settings</code>	Use this parameter to customize agent behavior, such as controlling the temperature of the LLM. For example, <code>'{"chatbot_temperature_override": 1}'</code> .	Optional	<code>{}</code>
N/A	<code>lightspeed_chatbot_chatbot_max_tokens</code>	Maximum number of tokens to generate a chat response.	Optional	4096
N/A	<code>lightspeed_chatbot_http_port</code>	Port number that Ansible Lightspeed intelligent assistant listens on for HTTP requests.	Optional	8085
N/A	<code>lightspeed_chatbot_model_id</code>	The ID of the LLM model that is configured on your LLM setup.	Optional	
N/A	<code>lightspeed_chatbot_model_api_key</code>	The API token or the API key of your LLM setup. This token is sent along with the authorization header when an inference API is called.	Optional	

Ansible Lightspeed intelligent assistant integration with MCP server variables

Inventory file variables for Ansible Lightspeed intelligent assistant integration with Model Context Protocol (MCP) server.

RPM variable name	Container variable name	Description	Required or optional	Default
N/A	<code>lightspeed_mcp_controller_enabled</code>	Controls whether the Ansible Lightspeed MCP controller is enabled or disabled.	Optional	<code>false</code>

RPM variable name	Container variable name	Description	Required or optional	Default
N/A	<code>lightspeed_mcp_controller_port</code>	Ansible Lightspeed MCP controller port.	Optional	8004
N/A	<code>lightspeed_mcp_lightspeed_enabled</code>	Ansible Lightspeed MCP lightspeed enabled.	Optional	false
N/A	<code>lightspeed_mcp_lightspeed_port</code>	Ansible Lightspeed MCP lightspeed port.	Optional	8005

Change your LLM model

To change the LLM model for your containerized Ansible Automation Platform deployment of Ansible Lightspeed intelligent assistant, you must edit the inventory file with the specific details of your new LLM provider and then rerun the install playbook.

Procedure

1. Edit the inventory file to update the following Ansible Lightspeed intelligent assistant variables with the specific details of your required LLM provider:
 - `lightspeed_chatbot_model_url`
 - `lightspeed_chatbot_model_api_key`
 - `lightspeed_chatbot_model_id`
 - `lightspeed_chatbot_default_provider`
2. Rerun the `install` playbook to [install the containerized Ansible Automation Platform](#).

Related information

[Appendix: Red Hat Ansible Lightspeed Inventory file variables](#)

Use the Ansible Lightspeed intelligent assistant

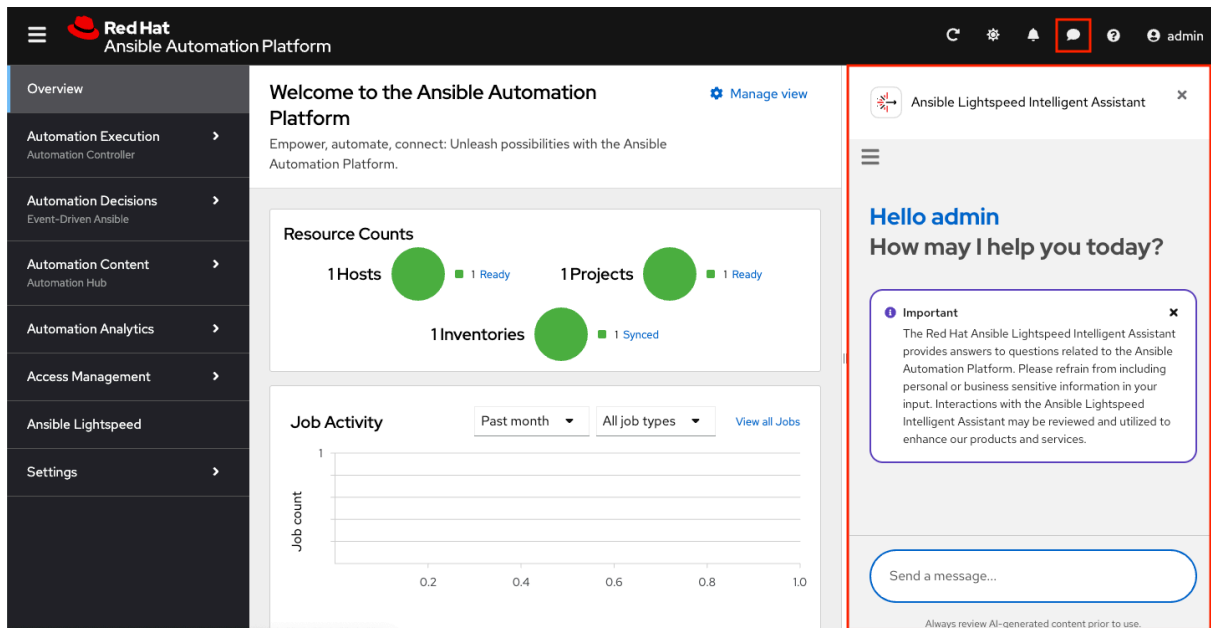
After you deploy the Ansible Lightspeed intelligent assistant, all Ansible users within the organization can access and use the chat interface to ask questions and receive information about the Ansible Automation Platform.

Access the Ansible Lightspeed intelligent assistant

1. Log in to the Ansible Automation Platform.

2. Click the Ansible Lightspeed intelligent assistant icon  that is displayed at the top right corner of the taskbar.

The Ansible Lightspeed intelligent assistant window opens with a welcome message, as shown in the following image:



Use the Ansible Lightspeed intelligent assistant

You can perform the following tasks:

- Ask questions in the prompt field and get answers about the Ansible Automation Platform

NOTE:

If you are using an IBM Granite 3.3 series AI model, you might experience a delay of about one minute when waiting for a chat response. To resolve this error, restart the chat session.

- View the chat history of all conversations in a chat session.
- Search the chat history using a user prompt or answer. The chat history is deleted when you close an existing chat session or log out from the Ansible Automation Platform.
- Restore an earlier chat by clicking the relevant entry from the chat history.
- Give feedback on the quality of the chat answers, by clicking the **Thumbs up** or **Thumbs down** icon.

- Copy and record the answers by clicking the **Copy** icon.
- Change the mode of the virtual assistant to dark or light mode, by clicking the **Sun** icon



- from the top right corner of the toolbar.
- Clear the context of an existing chat by using the **New chat** button in the chat history.
- Close the chat interface while working on the Ansible Automation Platform.

Configure the Ansible VS Code extension

If you deployed the Ansible Lightspeed coding assistant, you must also configure the Ansible VS Code extension with the generated Ansible Lightspeed URL. This configuration enables the Ansible users in your organization to use the Ansible Lightspeed coding assistant to create Ansible content.

Before you begin

- You have installed VS Code version 1.70.1 or later.
- Your organization administrator has configured an IBM watsonx Code Assistant model for your organization.
- Your network or firewall configuration permits ingress traffic on port 8447. This port is required for containerized installations to connect IBM watsonx Code Assistant with the Ansible VS Code extension, which then provides access to the Ansible Lightspeed coding assistant.

Procedure

1. Open the VS Code application.
2. From the Activity bar, click the **Extensions** icon .
3. From the Installed Extensions list, select **Ansible**.
4. From the **Ansible** extension page, click the **Settings** icon and select **Extension Settings**.
5. Select **Ansible Lightspeed** settings, and specify the following information:
 - a. Ensure that the **Enable Ansible Lightspeed with watsonx Code Assistant inline suggestions** checkbox is selected.
 - b. In the **URL for Ansible Lightspeed** field, verify that you have the following URL:
`https://<node.ansible.com>:8447`.
 - c. Select the **Enable Ansible Lightspeed with watsonx Code Assistant inline suggestions** checkbox.
6. Optional: If you want to use the custom model instead of the default model, in the **Model ID Override** field, enter the custom model ID. The **model-override** setting enables you to

override the default model and use the custom model, after your organization administrator has created a custom model and has shared the model ID with you separately.

Your settings are automatically saved in VS Code.

NOTE:

If your organization recently subscribed to the Red Hat Ansible Automation Platform, it might take a few hours for Red Hat Ansible Lightspeed to detect the new subscription. In VS Code, use the **Refresh** button in the Ansible extension from the Activity bar to check again.

Next steps

- After you configure the Ansible VS Code extension with the generated Ansible Lightspeed URL, you can use the Ansible VS Code extension and start developing Ansible content.
 - a. [Log in to Red Hat Ansible Lightspeed through the Ansible VS Code extension.](#)
 - b. [Develop Ansible content.](#)

Deploy the Ansible Lightspeed intelligent assistant on OpenShift Container Platform

As a system administrator, you can deploy Ansible Lightspeed intelligent assistant on Ansible Automation Platform 2.6 on OpenShift Container Platform.

Overview

Install and use Ansible Lightspeed intelligent assistant on Ansible Automation Platform 2.6 on OpenShift Container Platform. An intuitive chat interface, embedded in Ansible Automation Platform, it uses generative artificial intelligence (AI) to answer questions about Ansible Automation Platform.

The Ansible Lightspeed intelligent assistant interacts with users in their natural language prompts in English, and uses Large Language Models (LLMs) to generate quick, accurate, and personalized responses. These responses empower Ansible users to work more efficiently, thereby improving productivity and the overall quality of their work.

Ansible Lightspeed intelligent assistant requires the following configurations:

- Installation of Ansible Automation Platform 2.6 on Red Hat OpenShift Container Platform
- Deployment of an LLM provider served by either a Red Hat AI platform or a third-party AI platform. To know the LLM providers that you can use, see LLM Providers below.

IMPORTANT:

Red Hat does not collect any telemetry data from your interactions with the Ansible Lightspeed intelligent assistant.

Upgrading from Ansible Automation Platform 2.5 to 2.6.1 or 2.6 to 2.6.1 enables HTTPS and TLS by default for internal communication between the Ansible Lightspeed API and the Ansible Lightspeed intelligent assistant pod. Following the upgrade to Ansible Automation Platform 2.6.1, the intelligent assistant will be unavailable for approximately 60 seconds while its pod restarts.

Integration with MCP server

Ansible Lightspeed intelligent assistant integration with the Model Context Protocol (MCP) server is available as a Technology Preview release. This integration enhances the user experience by delivering relevant, dynamically sourced data results to your queries.

MCP is an open protocol that standardizes how applications provide context to LLMs. Using the protocol, an MCP server provides a standardized way for an LLM to increase context by requesting and receiving real-time information from external resources. The integration with an MCP server enables the Ansible Lightspeed intelligent assistant to offer an enhanced user experience by delivering relevant, dynamically sourced data results to your queries. You can configure a MCP server in the chatbot configuration secret.

NOTE:

Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

Ansible Automation Platform 2.6 requirements

- You have installed Ansible Automation Platform 2.6 on your OpenShift Container Platform environment.
- You have administrator privileges for the Ansible Automation Platform.
- You have provisioned an OpenShift cluster with Operator Lifecycle Management installed.

Large Language Model (LLM) provider requirements

You must have configured an LLM provider that you will use before deploying the Ansible Lightspeed intelligent assistant.

An LLM is a type of machine learning model that can interpret and generate human-like language. When an LLM is used with the Ansible Lightspeed intelligent assistant, the LLM can interpret questions accurately and provide helpful answers in a conversational manner. Your LLM must have tool calling enabled to handle tool-related requests. Tool calling allows the assistant to interact with platform services and execute complex workflows.

Ansible Lightspeed intelligent assistant can rely on the following LLM providers:

- **Red Hat LLM providers:**

- **Red Hat Enterprise Linux AI**

You can configure Red Hat Enterprise Linux AI as the LLM provider. As the Red Hat Enterprise Linux is in a different environment than the Ansible Lightspeed deployment, the model deployment must allow access using a secure connection.

Ansible Lightspeed intelligent assistant supports vLLM Server. When self-hosting an LLM with Red Hat Enterprise Linux AI, you can use vLLM Server as the inference engine.

- **Red Hat OpenShift AI**

You must deploy an LLM on the Red Hat OpenShift AI single-model serving platform that uses the Virtual Large Language Model (vLLM) runtime. If the model deployment resides in a different OpenShift environment than the Ansible Lightspeed deployment, include a route to expose the model deployment outside the cluster.

Ansible Lightspeed intelligent assistant supports vLLM Server. When self-hosting an LLM with Red Hat OpenShift AI, you can use vLLM Server as the inference engine.

NOTE:

For configurations with Red Hat Enterprise Linux AI or Red Hat OpenShift AI, you must host your own LLM provider instead of using a SaaS LLM provider.

- **Red Hat AI Inference Server**

You can deploy an LLM using Red Hat AI Inference Server as your inference runtime. Red Hat AI Inference Server supports vLLM runtimes for efficient model serving and can be configured to work with Ansible Lightspeed intelligent assistant.

If the Red Hat AI Inference Server deployment is in a different environment than the Ansible Lightspeed deployment, ensure the model deployment allows access using a secure connection and configure appropriate network routing.

Ansible Lightspeed intelligent assistant supports vLLM Server when self-hosting an LLM with Red Hat AI Inference Server as the inference engine.

- **Third-party LLM providers:**

- **OpenAI**

To use OpenAI with the Ansible Lightspeed intelligent assistant, you need access to the OpenAI API platform.

- **Microsoft Azure OpenAI**

To use Microsoft Azure with the Ansible Lightspeed intelligent assistant, you need access to Microsoft Azure OpenAI.

Process for configuring and using the Ansible Lightspeed intelligent assistant

Perform the following tasks to set up and use the Ansible Lightspeed intelligent assistant in your Ansible Automation Platform instance on the OpenShift Container Platform environment:

Task	Description
Deploy the Ansible Lightspeed intelligent assistant on OpenShift Container Platform	An Ansible Automation Platform administrator who wants to deploy the Ansible Lightspeed intelligent assistant for all Ansible users in the organization. Perform the following tasks: 1. Create a chatbot configuration secret. 2. Update the YAML file of the Ansible Automation Platform to use the chatbot connection secret. 3. Optional: Change your LLM model if you want to use a different LLM provider after deploying Red Hat Ansible Lightspeed.
Access and use the Ansible Lightspeed intelligent assistant	All Ansible users who want to use the intelligent assistant to get answers to their questions about the Ansible Automation Platform.

Related information

[Deploy the Ansible Lightspeed intelligent assistant on OpenShift Container Platform](#)

[Technology Preview Features Support Scope](#)

[Optional: Allowing access to a model from a secure endpoint](#)

[About the single-model serving platform](#)

[Red Hat AI Inference Server documentation](#)

[OpenAI API platform](#)

[Microsoft Azure OpenAI](#)

[Deploy and use the Ansible Lightspeed intelligent assistant](#)

[Change your LLM model](#)

Install Ansible development tools

You have two options for installing Ansible development tools, so that you can choose the installation method that best suits your preferred operating system and development environment.

Red Hat provides two options for installing Ansible development tools.

- Installation on a RHEL container running inside VS Code. You can install this option on MacOS, Windows, and Linux systems.
- Installation on your local RHEL system using an RPM (Red Hat Package Manager) package.

Requirements

To install and use Ansible development tools, you must meet the following requirements. Extra requirements for Windows installations and containerized installations are indicated in the procedures.

- Python 3.10 or later.
- VS Code (Visual Studio Code) with the Ansible extension added.
- For containerized installations, the Microsoft Dev Containers VS Code extension.
- A containerization platform, for example Podman, Podman Desktop, Docker, or Docker Desktop.

NOTE:

The installation procedure for Ansible development tools on Windows covers the use of Podman Desktop only.

- You have a Red Hat account and you can log in to the Red Hat container registry at `registry.redhat.io`.

Related information

[Install and configure VS Code](#)

[Install and configure the Dev Containers extension](#)

[Microsoft Dev Containers](#) extension in VS Code.

[Requirements for Ansible development tools on Windows](#)

[Authenticate with the Red Hat container registry](#)

Requirements for Ansible development tools on Windows

If you are installing Ansible development tools on a container in VS Code on Windows, there are extra requirements:

Before you begin

- Windows Subsystem for Linux(WSL2)
- Podman Desktop

Procedure

1. Install WSL2 without a distribution:

```
$ wsl --install --no-distribution
```

2. Use `cgroupsv2` by disabling `cgroupsv1` for WSL2:

Edit the `%USERPROFILE%/wsl.conf` file and add the following lines to force `cgroupv2` usage:

```
[wsl2]
kernelCommandLine = cgroup_no_v1="all"
```

3. Install Podman Desktop. Follow the instructions in [Installing Podman Desktop and Podman on Windows](#) in the Podman Desktop documentation.

You do not need to change the default settings in the set-up wizard.

4. Ensure the podman machine is using `cgroupsv2` :

```
$ podman info | findstr cgroup
```

5. Test Podman Desktop:

```
$ podman run hello
```

6. Configure the settings for Podman Desktop. Add a `%USERPROFILE%\bin\docker.bat` file with the following content:

```
@echo off
podman %*
```

This avoids having to install Docker as required by the VS Code `Dev Container` extension.

7. Add the `%USERPROFILE%\bin` directory to the `PATH` :
 - a. Select **Settings** and search for "Edit environment variables for your account" to display all of the user environment variables.
 - b. Highlight "Path" in the top user variables box, click **Edit** and add the path.
 - c. Click **Save** to set the path for any new console that you open.

Authenticate with the Red Hat container registry

All container images available through the Red Hat container catalog are hosted on an image registry, `registry.redhat.io`. The registry requires authentication for access to images.

Before you begin

- You have a Red Hat login. It is the same account that you use to log in to the Red Hat Customer Portal (access.redhat.com) and manage your Red Hat subscriptions.

NOTE:

If you are planning to install the Ansible development tools on a container inside VS Code, you must log in to `registry.redhat.io` before launching VS Code so that VS Code can pull the `devtools` container from `registry.redhat.io`.

If you are running Ansible development tools on a container inside VS Code and you want to pull execution environments or the `devcontainer` to use as an execution environment, you must log in from a terminal prompt within the `devcontainer` from a terminal inside VS Code.

Procedure

1. Check whether you are already logged in to the `registry.redhat.io` registry:

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

The command output displays your Red Hat login if you are logged in to `registry.redhat.io`.

2. If you are not logged in to `registry.redhat.io`, use the `podman login` command with your credentials to access content on the registry.

```
$ podman login registry.redhat.io
Username: my_redhat_username
Password: *****
```

Related information

[Red Hat Container Registry Authentication](#)

Install and configure VS Code

Install VS Code by following the instructions found on the [Download Visual Studio Code](#) page in the official documentation.

- To install VS Code, follow the instructions on the [Download Visual Studio Code page](#) in the Visual Studio Code documentation.

Install the VS Code Ansible extension

The Ansible extension adds language support for Ansible to VS Code. It incorporates Ansible development tools to facilitate creating and running automation content.

For a full description of the Ansible extension, see the [Visual Studio Code Marketplace](#).

Procedure

1. Open VS Code.
2. Click the **Extensions** (📦) icon in the Activity Bar, or click **View > Extensions**, to display the **Extensions** view.
3. In the search field in the **Extensions** view, type `Ansible Red Hat`.
4. Select the Ansible extension and click **Install**.

Result

When the language for a file is recognized as Ansible, the Ansible extension provides features such as auto-completion, hover, diagnostics, and goto. The language identified for a file is displayed in the Status bar at the bottom of the VS Code window.

The following files are assigned the Ansible language:

- YAML files in a `/playbooks` directory

- Files with the following double extension: `.ansible.yml` or `.ansible.yaml`
- Certain YAML names recognized by Ansible, for example `site.yml` or `site.yaml`
- YAML files whose filename contains "playbook": `playbook.yml` or `playbook.yaml`

If the extension does not identify the language for your playbook files as Ansible, follow the procedure in [Associating the Ansible language to YAML files](#).

Configure Ansible extension settings

The Ansible extension supports multiple configuration options.

Before you begin

- Open a workspace or folder, or create a new folder, in VS Code using the **File > Open Folder** menu. This is necessary because the file that stores settings preferences for workspaces is specific to a folder or workspace.

You can configure the settings for the extension on a user level, on a workspace level, or for a particular directory. User-based settings are applied globally for any instance of VS Code that is opened. A VS Code workspace is a collection of one or more folders that you can open in a single VS Code window. Workspace settings are stored within your workspace and only apply when the current workspace is opened.

It is useful to configure settings for your workspace for the following reasons:

- If you define and maintain configurations specific to your playbook project, you can customize your Ansible development environment for individual projects without altering your preferred setup for other work. You can have different settings for a Python project, an Ansible project, and a C++ project, each optimized for the respective stack without the need to manually reconfigure settings each time you switch projects.
- If you include workspace settings when setting up version control for a project you want to share with your team, everyone uses the same configuration for that project.

Procedure

1. Open the Ansible extension settings:
 - a. Click the  **Extensions** icon in the activity bar.
 - b. Select the Ansible extension, and click the 'gear' icon and then **Extension Settings** to display the extension settings.
 - Alternatively, click **Code > Settings > Settings** to open the **Settings** page.
 - c. Enter `Ansible` in the search bar to display the settings for the extension.
2. Select the **Workspace** tab to configure your settings for the current VS Code workspace.

- If the **Workspace** tab is not displayed, open a folder or create a new folder using the **File > Open Folder** menu.
3. The Ansible extension settings are pre-populated. Modify the settings to suit your requirements:
 - Check the **Ansible > Validation > Lint: Enabled** box to enable ansible-lint.
 - Check the `Ansible Execution Environment: Enabled` box to use an execution environment.
 - Specify the execution environment image you want to use in the **Ansible > Execution Environment: image** field.
 - To use Red Hat Ansible Lightspeed, check the **Ansible > Lightspeed: Enabled** box, and enter the URL for Lightspeed.

Related information

[Ansible VS Code Extension by Red Hat page](#)

[What is a VS Code workspace?](#)

Install and configure the Dev Containers extension

If you are installing the containerized version of Ansible development tools, you must install the [Microsoft Dev Containers](#) extension in VS Code.

Procedure

1. Open VS Code.
2. Click the **Extensions** (📦) icon in the Activity Bar, or click **View > Extensions**, to display the **Extensions** view.

3. In the search field in the **Extensions** view, type `Dev Containers`.

4. Select the Dev Containers extension from Microsoft and click **Install**.

If you are using Podman or Podman Desktop as your containerization platform, you must modify the default settings in the `Dev Containers` extension.

5. Replace docker with podman in the `Dev Containers` extension settings:

- a. In VS Code, open the settings editor.
- b. Search for `@ext:ms-vscode-remote.remote-containers`.

Alternatively, click the **Extensions** icon in the activity bar and click the gear icon for the `Dev Containers` extension.

6. Set `Dev > Containers: Docker Path` to `podman`.
7. Set `Dev > Containers: Docker Compose Path` to `podman-compose`.

Set the language for the VS Code extension

The Ansible VS Code extension works only when the language associated with a file is set to Ansible. The extension provides features that help create Ansible playbooks, such as auto-completion, hover, and diagnostics.

The Ansible VS Code extension automatically associates the Ansible language with some files. The procedures below describe how to set the language for files that are not recognized as Ansible files.

Associate the Ansible language to YAML files

The following procedure describes how to manually assign the Ansible language to a YAML file that is open in VS Code.

Procedure

1. Open or create a YAML file in VS Code.
2. Hover the cursor over the language identified in the status bar at the bottom of the VS Code window to open the **Select Language Mode** list.
3. Select **Ansible** in the list.

The language shown in the status bar at the bottom of the VS Code window for the file is changed to Ansible.

Add the persistent file association for the Ansible language to `settings.json`

Instead of manually associating the Ansible language to YAML files, you can add file association for the Ansible language in your `settings.json` file.

Procedure

1. Open the `settings.json` file:
 - a. Click **View > Command Palette** to open the command palette.
 - b. Enter `Workspace settings` in the search box and select **Open Workspace Settings (JSON)**.

2. Add the following code to `settings.json`.

```
{
  ...

  "files.associations": {
    "*plays.yml": "ansible",
    "*init.yml": "yaml",
  }
}
```

Install Ansible development tools on a container inside VS Code

The Dev Containers VS Code extension requires a `.devcontainer` file to store settings for your dev containers. You must use the Ansible extension to scaffold a config file for your dev container, and reopen your directory in a container in VS Code.

Before you begin

- You have installed a containerization platform, for example Podman, Podman Desktop, Docker, or Docker Desktop.
- You have a Red Hat login and you have logged in to the Red Hat registry at `registry.redhat.io`. For information about logging in to `registry.redhat.io`, see [Authenticating with the Red Hat container registry](#).
- You have installed VS Code.
- You have installed the Ansible extension in VS Code.
- You have installed the [Microsoft Dev Containers](#) extension in VS Code.
- If you are installing Ansible development tools on Windows, launch VS Code and connect to the WSL machine. Click the Remote (🔗) icon. In the dropdown menu that appears, select the option to connect to the WSL machine.

Procedure

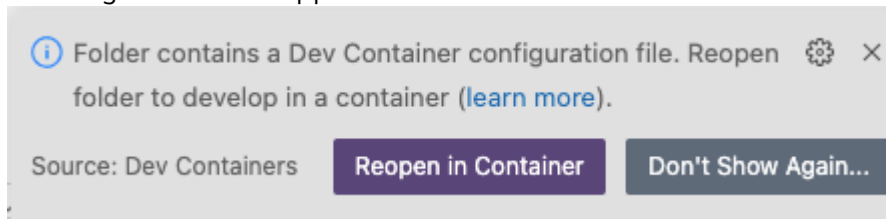
1. In VS Code, navigate to your project directory.
2. Click the Ansible icon in the VS Code activity bar to open the Ansible extension.
3. In the **Ansible Development Tools** section of the Ansible extension, scroll down to the **ADD** option and select **Devcontainer**.

4. In the **Create a devcontainer** page, select the **Downstream** container image from the **Container image** options.

This action adds `devcontainer.json` files for both Podman and Docker in a `.devcontainer` directory.

5. Reopen or reload the project directory:

- If VS Code detects that your directory contains a `devcontainer.json` file, the following notification appears:



Click **Reopen in Container**.

- If the notification does not appear, click the **Remote** (🔗) icon. In the dropdown menu that appears, select **Reopen in Container**.
6. Select the dev container for Podman or Docker according to the containerization platform you are using.

The **Remote** (🔗) status in the VS Code Status bar displays `opening Remote` and a notification indicates the progress in opening the container.

Result

When the directory reopens in a container, the **Remote** (🔗) status displays `Dev Container: ansible-dev-container`.

NOTE:

The base image for the container is a Universal Base Image Minimal (UBI Minimal) image that uses `microdnf` as a package manager. The `dnf` and `yum` package managers are not available in the container.

For information about using `microdnf` in containers based on UBI Minimal images, see [Adding software in a minimal UBI container](#) in the Red Hat Enterprise Linux *Building, running, and managing containers* guide.

Install Ansible development tools from an RPM package

Ansible development tools are bundled in the Ansible Automation Platform RPM (Red Hat Package Manager) package. Refer to the [RPM installation](#) documentation for information on installing Ansible Automation Platform.

Before you begin

- You have installed a supported version of Red Hat Enterprise Linux.
- You have registered your system with Red Hat Subscription Manager.
- You have installed a containerization platform, for example Podman or Docker.

Procedure

1. Run the following command to check whether Simple Content Access (SCA) is enabled:

```
$ sudo subscription-manager status
```

If Simple Content Access is enabled, the output contains the following message:

```
Content Access Mode is set to Simple Content Access.
```

- a. If Simple Content Access is not enabled, attach the Red Hat Ansible Automation Platform SKU:

```
$ sudo subscription-manager attach --pool=<sku-pool-id>
```

2. Install Ansible development tools with the following command:

```
$ sudo dnf install  
--enablerepo=ansible-automation-platform-aap-version-for-rhel-rhel-version-  
x86_64-rpms ansible-dev-tools
```

Replace *aap-version* with your Ansible Automation Platform version (for example, 2.6) and *rhel-version* with your Red Hat Enterprise Linux major version (for example, 9).

Result

1. Verify that the Ansible development tools have been installed:

```
$ rpm -aq | grep ansible-dev-tools
```

If the installation was successful, the output shows the `ansible-dev-tools` package and its version number, for example:

```
ansible-dev-tools-25.8.3-1.el9ap.noarch
```

2. On successful installation, you can view the help documentation for the `ansible-creator` utility:

```
$ ansible-creator --help

usage: ansible-creator [-h] [--version] command ...

The fastest way to generate all your ansible content.

Positional arguments:
  command
    add          Add resources to an existing Ansible project.
    init         Initialize a new Ansible project.

Options:
  --version      Print ansible-creator version and exit.
  -h            --help  Show this help message and exit
```

Configure Red Hat AI providers in the Ansible VS Code extension

You can configure the Ansible VS Code extension to interact with models hosted on Red Hat AI infrastructure such as the RHEL AI Inference Server and OpenShift AI.

Before you begin

- You have installed the Ansible VS Code extension version 25.12.3 or later.
- You already deployed and can access the Red Hat AI infrastructure (RHEL AI Inference Server or OpenShift AI).
- You have the API endpoint URL for the inference server.
- You have a valid API key or credential for the target model.
- You know the specific Model Name.

This integration enables you to use the "Bring Your Own LLM" (BYOLLM) capability to generate and explain Ansible playbooks and roles. You can use your own hosted models, such as Llama, Mistral, or OpenAI, without requiring external middleware.

Procedure

1. In Visual Studio Code, open the LLM provider settings:
 - a. In the Activity Bar, select the **Ansible** icon to open the Ansible Development Tools panel.
 - b. In the **GENERATIVE AI** section, select the settings icon (cog) to open the LLM provider configuration.
2. In the **API Endpoint URL** field, enter the full address of your RHEL AI or OpenShift AI instance.
3. In the **Model Name/ID** field, enter the specific identifier for the model you want to use (for example, `granite-13b-chat-v2`).
4. In the **API Key** field, enter the credentials required to authenticate with your infrastructure.

NOTE:

The extension connects directly to the provider endpoint. It does not require the `c.ai.ansible.redhat.com` middleware or Red Hat authentication to enable this feature.

5. Ensure that the Red Hat AI provider is selected as the active provider.

IMPORTANT:

Only one provider can be active at a time. If you switch between providers, ensure you select the correct provider in the LLM provider configuration to enable the corresponding features.

Result

1. Open an Ansible playbook or role file in the editor.
 2. Check the status bar to verify that the Red Hat AI provider status is displayed and indicates a successful connection.
 3. Right-click within the editor and trigger **Ansible Lightspeed > Generate Playbook**. If the configuration is correct, the extension generates content using the model hosted on your RHEL AI or OpenShift AI infrastructure.
- **Issue:** The extension fails to connect or times out.
Resolution: Verify that the **API Endpoint URL** is reachable from your network. The extension has a default timeout of 30 seconds. Check the **Output** view in VS Code under the **Ansible Lightspeed** channel for specific error logs.
 - **Issue:** Inline suggestions (ghost text) do not appear while typing.
Resolution: This is expected behavior. Inline task suggestions and content source matching are not supported for Red Hat AI infrastructure providers.

Install Red Hat Developer Hub (Red Hat OpenShift Container Platform only)

Ansible plug-ins for Red Hat Developer Hub (RHDH) offer a tailored Red Hat Developer Hub interface, curated learning paths, and software templates for creating projects, all while linking to supported development environments like OpenShift Dev Spaces and Ansible Automation Platform.

Red Hat Developer Hub

Red Hat Developer Hub (RHDH) is an open developer platform you can use to build custom developer portals for your organization.

Ansible plug-ins for Red Hat Developer Hub

Ansible plug-ins for Red Hat Developer Hub deliver an Ansible-first Red Hat Developer Hub user experience that simplifies the automation experience for Ansible users of all skill levels.

The Ansible plug-ins provide curated content and features to accelerate Ansible learner onboarding and streamline Ansible use case adoption across your organization.

The Ansible plug-ins provide:

- A customized home page and navigation tailored to Ansible users.
- Curated Ansible learning paths to help users new to Ansible.
- Software templates for creating Ansible playbook and collection projects that follow best practices.
- Links to supported development environments and tools with opinionated configurations.

Install with a Helm chart

The following procedures describe how to install Ansible plug-ins in Red Hat Developer Hub instances on Red Hat OpenShift Container Platform using a Helm chart.

The workflow is as follows:

1. Download the Ansible plug-ins files.
2. Create a plug-in registry in your OpenShift cluster to host the Ansible plug-ins.
3. Add the plug-ins to the Helm chart.
4. Create a custom ConfigMap.
5. Add your custom ConfigMap to your Helm chart.
6. Edit your custom ConfigMap and Helm chart according to the required and optional configuration procedures.

NOTE:

You can save changes to your Helm and ConfigMap after each update to your configuration. You do not have to make all the changes to these files in a single session.

Prerequisites

To proceed, you must have Red Hat Developer Hub installed on Red Hat OpenShift Container Platform (OCP) and a valid subscription to Red Hat Ansible Automation Platform.

- Red Hat Developer Hub installed on Red Hat OpenShift Container Platform.
 - For Helm installation, follow the steps in the Installing Red Hat Developer Hub on OpenShift Container Platform with the Helm chart section of *Installing Red Hat Developer Hub on OpenShift Container Platform*.
 - For Operator installation, follow the steps in the Installing Red Hat Developer Hub on OpenShift Container Platform with the Operator section of *Installing Red Hat Developer Hub on OpenShift Container Platform*.
- A valid subscription to Red Hat Ansible Automation Platform.
- An OpenShift Container Platform instance with the appropriate permissions within your project to create an application.
- The Red Hat Developer Hub instance can query the automation controller API.
- Optional: To use the integrated learning paths, you must have outbound access to `developers.redhat.com`.

Related information

[Installing Red Hat Developer Hub on OpenShift Container Platform with the Helm chart](#)
[Installing Red Hat Developer Hub on OpenShift Container Platform with the Operator](#)

Recommended RHDH preconfiguration

Red Hat recommends performing the following initial configuration tasks in Red Hat Developer Hub (RHDH). However, you can install the Ansible plug-ins for Red Hat Developer Hub before completing these tasks.

- Authentication in Red Hat Developer Hub
- Authorization in Red Hat Developer Hub

NOTE:

Red Hat provides a repository of software templates for RHDH that uses the `publish:github` action. To use these software templates, you must install the required GitHub dynamic plugins.

Related information

[Authentication in Red Hat Developer Hub](#)

[Authorization in Red Hat Developer Hub](#)

[Repository of software templates for RHDH](#)

Choose a plug-in delivery method

Ansible plug-ins for Red Hat Developer Hub support two delivery methods. Choose the method that fits your environment.

- **OCI container (recommended):** Red Hat Developer Hub pulls the Ansible plug-ins directly from `registry.redhat.io` as OCI artifacts during startup. You do not need to use any manual file downloads or plug-in registry deployment.
- **HTTP plug-in registry:** Manually download the Ansible plug-ins tarball files, deploy an HTTP plug-in registry in your OpenShift cluster, and configure Red Hat Developer Hub to pull plug-ins from that registry. Use this method if your environment cannot pull OCI artifacts from `registry.redhat.io`.

Complete one of the following procedures before configuring the Ansible plug-ins in the Required configuration section.

Use OCI container delivery method

Red Hat Developer Hub pulls the Ansible plug-ins directly from `registry.redhat.io` as OCI artifacts. This is the recommended method that requires a registry authentication secret in the same OpenShift project as your Red Hat Developer Hub deployment.

Before you begin

- You have a Red Hat account with access to `registry.redhat.io`.
- You have a registry service account token from the Red Hat Customer Portal.
- You have access to the OpenShift project. In the same project you had installed Red Hat Developer Hub.
- You have installed the OpenShift CLI (`oc`) and logged in to your cluster.

This method is the default for the Helm chart and is the recommended method for production deployments.

Procedure

1. Create an `auth.json` file on your local machine with your `registry.redhat.io` credentials:

```
{
  "auths": {
    "registry.redhat.io": {
      "auth": "<base64-encoded-username:password>"
    }
  }
}
```

To generate the base64-encoded value use the `printf '%s' '<username>:<password>' | base64` command.

2. Create the authentication secret in the OpenShift project where you installed Red Hat Developer Hub.

The secret name must follow the pattern `<deployment-name>-dynamic-plugins-registry-auth`, where `<deployment-name>` matches your Red Hat Developer Hub deployment name.

- For a default Red Hat Developer Hub Helm installation with release name `developer-hub`:

```
oc create secret generic developer-hub-dynamic-plugins-registry-auth \
  --from-file=auth.json=./auth.json
```

- If you use a different Helm release name:

```
oc create secret generic <deployment-name>-dynamic-plugins-registry-auth \
  --from-file=auth.json=./auth.json
```

- If you already have Podman credentials configured locally:

```
oc create secret generic <deployment-name>-dynamic-plugins-registry-auth \
  --from-file=auth.json=${XDG_RUNTIME_DIR}/containers/auth.json
```

Step result:

IMPORTANT:

Create this secret in the same OpenShift project as your Red Hat Developer Hub deployment, and create it before you configure the plug-ins. Use a Red Hat Registry service account token, not your personal Red Hat account credentials.

Result

Verify that the secret exists in the project:

```
oc get secret <deployment-name>-dynamic-plugins-registry-auth
```

Related information

[Registry Service Accounts](#)

Download the Ansible plug-ins files

Download the Ansible plug-ins for Red Hat Developer Hub **Setup Bundle** from the Red Hat Ansible Automation Platform Product Software downloads page.

Procedure

1. In a browser, navigate to the [Red Hat Ansible Automation Platform Product Software downloads page](#) and select the **Product Software** tab.
2. Click **Download now** next to **Ansible plug-ins for Red Hat Developer Hub Setup Bundle** to download the latest version of the plug-ins.

The format of the filename is `ansible-rhdh-plugins-x.y.z.tar.gz`. Substitute the Ansible plug-ins release version, for example `2.0.0`, for `x.y.z`.

3. Create a directory on your local machine to store the `.tar` files.

```
$ mkdir /path/to/<ansible-backstage-plugins-local-dir-changeme>
```

4. Set an environment variable (`$DYNAMIC_PLUGIN_ROOT_DIR`) to represent the directory path.

```
$ export DYNAMIC_PLUGIN_ROOT_DIR=/path/to/<ansible-backstage-plugins-local-dir-changeme>
```

5. Extract the `ansible-rhdh-plugins-<version-number>.tar.gz` contents to `$DYNAMIC_PLUGIN_ROOT_DIR`.

```
$ tar --exclude='*code*' -xzf ansible-rhdh-plugins-x.y.z.tar.gz -C $DYNAMIC_PLUGIN_ROOT_DIR
```

Substitute the Ansible plug-ins release version, for example `2.0.0`, for `x.y.z`.

Result

Run `ls` to verify that the extracted files are in the `$DYNAMIC_PLUGIN_ROOT_DIR` directory:

```
$ ls $DYNAMIC_PLUGIN_ROOT_DIR
ansible-plugin-backstage-rhaap-dynamic-x.y.z.tgz
ansible-plugin-backstage-rhaap-dynamic-x.y.z.tgz.integrity
ansible-plugin-scaffolder-backend-module-backstage-rhaap-dynamic-x.y.z.tgz
ansible-plugin-scaffolder-backend-module-backstage-rhaap-dynamic-x.y.z.tgz.integrity
```

The files with the `.integrity` file type contain the plugin SHA value. The SHA value is used during the plug-in configuration.

Create a registry for the Ansible plug-ins

Set up a registry in your OpenShift cluster to host the Ansible plug-ins and make them available for installation in Red Hat Developer Hub (RHDH).

Procedure

1. Log in to your OpenShift Container Platform instance with credentials to create a new application.
2. Open your Red Hat Developer Hub OpenShift project.

```
$ oc project <YOUR_DEVELOPER_HUB_PROJECT>
```

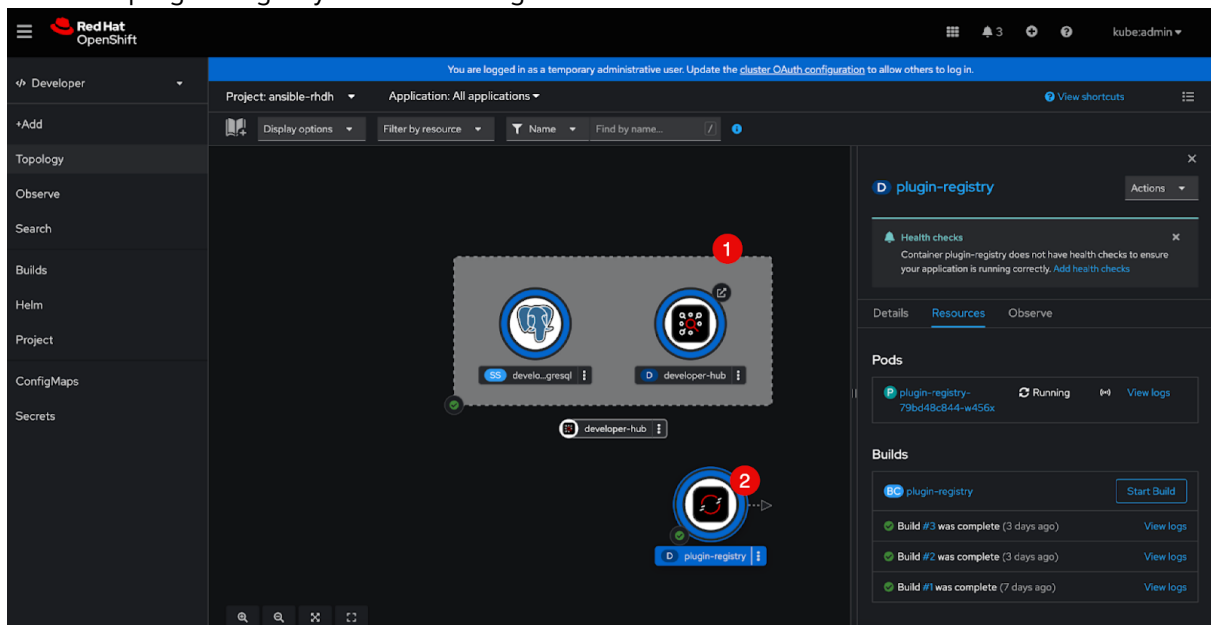
3. Run the following commands to create a plug-in registry build in the OpenShift cluster.

```
$ oc new-build httpd --name=plugin-registry --binary
$ oc start-build plugin-registry --from-dir=$DYNAMIC_PLUGIN_ROOT_DIR --wait
$ oc new-app --image-stream=plugin-registry
```

Result

To verify that the plugin-registry was deployed successfully, open the **Topology** view in the **Developer** perspective on the Red Hat Developer Hub application in the OpenShift Web console.

1. Click the plug-in registry to view the log.



(1) Developer hub instance

(2) Plug-in registry

2. Click the terminal tab and login to the container.
3. In the terminal, run `ls` to confirm that the `.tar` files are in the plugin registry.

```
ansible-plugin-backstage-rhaap-dynamic-x.y.z.tgz
ansible-plugin-scaffolder-backend-module-backstage-rhaap-dynamic-x.y.z.tgz
```

The version numbers and file names can differ.

Complete required configurations

Complete these required configuration steps to add the Ansible plug-ins to your Helm chart and link the Red Hat Developer Hub instance to the necessary Ansible Automation Platform components.

Add the Ansible plug-ins configuration

Modify the Red Hat Developer Hub Helm chart to add the Ansible plug-ins. The configuration depends on the plug-in delivery method you chose earlier.

Procedure

1. In the OpenShift Developer UI, navigate to **Helm > developer-hub > Actions > Upgrade > YAML view**.
2. Update the Helm chart configuration to add the dynamic plug-ins in the Red Hat Developer Hub instance. Under the `plugins` section in the YAML file, add the dynamic plug-ins that you want to enable. Choose the configuration that matches your delivery method.
 - **OCI container delivery (recommended):**

```

global:
  dynamic:
    includes:
      - dynamic-plugins.default.yaml
    plugins:
      - disabled: false
        package: 'oci://registry.redhat.io/ansible-automation-platform/
automation-portal:21!ansible-plugin-backstage-rhaap'
        pluginConfig:
          dynamicPlugins:
            frontend:
              ansible.plugin-backstage-rhaap:
                appIcons:
                  - importName: AnsibleLogo
                    name: AnsibleLogo
                dynamicRoutes:
                  - importName: AnsiblePage
                    menuItem:
                      icon: AnsibleLogo
                      text: Ansible
                      path: /ansible
            - disabled: false
              package: 'oci://registry.redhat.io/ansible-automation-platform/
automation-portal:2.1!ansible-plugin-scaffolder-backend-module-backstage-
rhaap'
              pluginConfig:
                dynamicPlugins:
                  backend:
                    ansible.plugin-scaffolder-backend-module-backstage-rhaap:
null

```

Replace `x.y.z` with the Ansible plug-ins version.

NOTE:

OCI delivery does not require the `integrity` hash values. The OCI registry handles integrity verification.

- **HTTP plug-in registry:**

```

global:
  dynamic:
    includes:
      - dynamic-plugins.default.yaml
    plugins:
      - disabled: false
        integrity: <SHA512 value>
        package: 'http://plugin-registry:8080/ansible-plugin-backstage-
rhaap-dynamic-x.y.z.tgz'
        pluginConfig:
          dynamicPlugins:
            frontend:
              ansible.plugin-backstage-rhaap:
                appIcons:
                  - importName: AnsibleLogo
                    name: AnsibleLogo
                dynamicRoutes:
                  - importName: AnsiblePage
                    menuItem:
                      icon: AnsibleLogo
                      text: Ansible
                      path: /ansible
            - disabled: false
              integrity: <SHA512 value>
              package: >-
                http://plugin-registry:8080/ansible-plugin-scaffolder-backend-
module-backstage-rhaap-dynamic-x.y.z.tgz
              pluginConfig:
                dynamicPlugins:
                  backend:
                    ansible.plugin-scaffolder-backend-module-backstage-rhaap:
null

```

Replace `x.y.z` with the correct plug-in version numbers and update the `integrity` values using the corresponding `.integrity` file content.

3. Click **Upgrade**.

Step result:

The Red Hat Developer Hub pods restart and the plug-ins are installed.

Result

To verify that the plug-ins have been installed, open the `install-dynamic-plugin` container logs:

1. Open the Developer perspective for the Red Hat Developer Hub application in the OpenShift Web console.
2. Select the **Topology** view.
3. Select the Red Hat Developer Hub deployment pod to open an information pane.
4. Select the **Resources** tab of the information pane.
5. In the **Pods** section, click **View logs** to open the **Pod details** page.
6. In the **Pod details** page, select the **Logs** tab.
7. Select `install-dynamic-plugins` from the drop-down list of containers to view the container log.

- **For OCI delivery**, a successful installation displays:

```
=> Successfully installed dynamic plugin oci:registry.redhat.io/ansible-automation-platform/ansible-plugin-backstage-rhaap-dynamic:x.y.z
```

- **For HTTP plug-in registry**, a successful installation displays:

```
=> Successfully installed dynamic plugin http://plugin-registry:8080/ansible-plugin-backstage-rhaap-dynamic-x.y.z.tgz
```

Add a pull secret to the Red Hat Developer Hub Helm configuration

You must add a pull secret to the Red Hat Developer Hub Helm configuration to enable the dynamic plug-ins to pull container images from authenticated registries.

Before you begin

- You have a Red Hat Customer Portal account and Red Hat Service Registry account.

Procedure

1. Create a new [Red Hat Registry Service account](#), if required.
2. Click the token name under the **Account name** column.
3. Select the **OpenShift Secret** tab and follow the instructions to add the pull secret to your Red Hat Developer Hub OpenShift project.

4. Add the new secret to the Red Hat Developer Hub Helm configuration, replacing `<your-redhat-registry-pull-secret>` with the name of the secret you generated on the Red Hat Registry Service Account website:

```
upstream:
  backstage:
    ...
  image:
    ...
  pullSecrets:
    - <your-redhat-registry-pull-secret>
    ...
```

Related information

[Red Hat Container Registry documentation](#)

Add the Ansible Developer Tools container

You must update the Helm chart configuration to add an extra container.

Procedure

1. Log in to the OpenShift UI.
2. Navigate to **Helm > developer-hub > Actions > upgrade > Yaml view** to open the Helm chart.
3. Update the `extraContainers` section in the YAML file.

Add the following code:

```
upstream:
  backstage:
    ...
    extraContainers:
      - command:
          - adt
          - server
        image: >-
            registry.redhat.io/ansible-automation-platform-26/ansible-dev-tools-
            rhel9:latest
        imagePullPolicy: IfNotPresent
        name: ansible-devtools-server
        ports:
          - containerPort: 8000
    ...
```

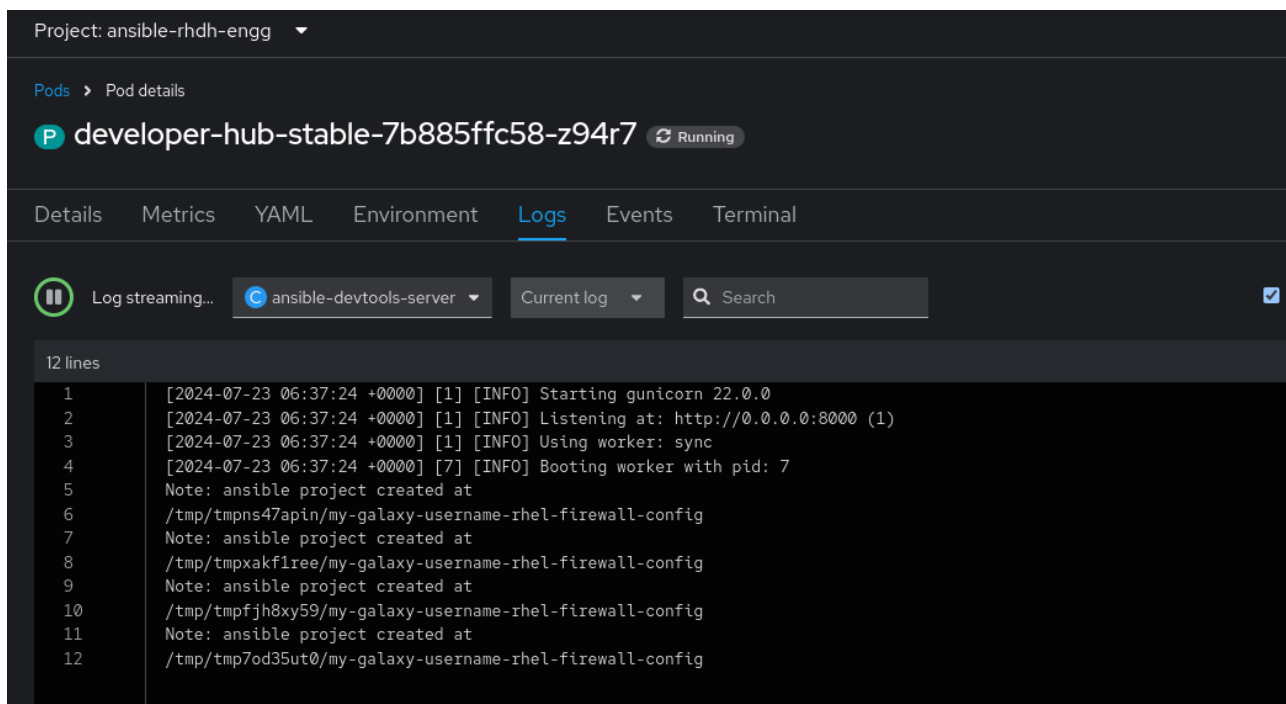
NOTE:

The image pull policy is `imagePullPolicy: IfNotPresent`. The image is pulled only if it does not already exist on the node. Update it to `imagePullPolicy: Always` if you always want to use the latest image.

4. Click **Upgrade**.

Result

To verify that the container is running, check the container log:



Add a custom ConfigMap

Create a custom Red Hat Developer Hub ConfigMap, typically named `app-config-rhdh`, to store custom application settings.

- Create a Red Hat Developer Hub ConfigMap following the procedure in the [Creating and using config maps](#) section of the OpenShift Container Platform *Nodes* guide. The following examples use a custom ConfigMap named `app-config-rhdh`.

To edit your custom ConfigMap, log in to the OpenShift UI and navigate to **Select Project (developerHubProj) > ConfigMaps > {developer-hub}-app-config > EditConfigMaps > app-config-rhdh**.

Configure the Ansible development tools Server

The `creatorService` URL is required for the Ansible plug-ins to provision new projects using the provided software templates.

Procedure

1. Edit your custom Red Hat Developer Hub config map, `app-config-rhdh`, that you created in [Adding a custom ConfigMap](#).

2. Add the following code to your Red Hat Developer Hub `app-config-rhdh.yaml` file.

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: app-config-rhdh
...
data:
  app-config-rhdh.yaml: |-
    ansible:
      creatorService:
        baseUrl: 127.0.0.1
        port: '8000'
    ...
```

Configure Ansible Automation Platform details

Connect Red Hat Developer Hub to your automation controller by configuring the Ansible Automation Platform details. This configuration uses a Personal Access Token (PAT) to authenticate the plug-ins, which allows them to interact with your automation environment.

NOTE:

The Ansible plug-ins continue to function regardless of the Ansible Automation Platform subscription status.

Procedure

1. Create a Personal Access Token (PAT) with "read and write" scope in automation controller, following the [Applications](#) section of *Access management and authentication*.
2. Edit your custom Red Hat Developer Hub config map, for example `app-config-rhdh`.
3. Add your Ansible Automation Platform details to `app-config-rhdh.yaml`.
 - a. Set the `baseUrl` key with your automation controller URL.
 - b. Set the `token` key with the generated token value that you created in Step 1.
 - c. Set the `checkSSL` key to `true` or `false`.

If `checkSSL` is set to `true`, the Ansible plug-ins verify whether the SSL certificate is valid.

```
data:
  app-config-rhdh.yaml: |
    ...
  ansible:
    ...
    rhaap:
      baseUrl: '<https://MyControllerUrl>'
      token: '<AAP Personal Access Token>'
      checkSSL: true
```

NOTE:

You are responsible for protecting your Red Hat Developer Hub installation from external and unauthorized access. Manage the backend authentication key like any other secret. Meet strong password requirements, do not expose it in any configuration files, and only inject it into configuration files as an environment variable.

Add Ansible plug-ins software templates

Ansible Automation Platform provides software templates for Red Hat Developer Hub to provision new playbooks and collection projects based on Ansible best practices.

Procedure

1. Edit your custom Red Hat Developer Hub config map, for example `app-config-rhdh`.
2. Add the following code to your Red Hat Developer Hub `app-config-rhdh.yaml` file.

```

data:
  app-config-rhdh.yaml: |
    catalog:
      ...
    locations:
      ...
      - type: url
        target: https://github.com/ansible/ansible-rhdh-templates/blob/main/
all.yaml
  rules:
    - allow: [Template]

```

Related information

[Managing templates](#)

Configure Role Based Access Control

Red Hat Developer Hub offers Role-based Access Control (RBAC) functionality. RBAC can then be applied to the Ansible plug-ins content.

- Assign the following roles:
 - Members of the `admin:superUsers` group can select templates in the **Create** tab of the Ansible plug-ins to create playbook and collection projects.
 - Members of the `admin:users` group can view templates in the **Create** tab of the Ansible plug-ins.
- The following example adds RBAC to Red Hat Developer Hub.

```

data:
  app-config-rhdh.yaml: |
    plugins:
      ...
    permission:
      enabled: true
      rbac:
        admin:
          users:
            - name: user:default/<user-scm-ida>
          superUsers:
            - name: user:default/<user-admin-idb>

```

For more information about permission policies and managing RBAC, refer to the [Authorization in Red Hat Developer Hub](#) guide for Red Hat Developer Hub.

Optional configurations

Enable Red Hat Developer Hub authentication and configure optional integrations, such as connecting to OpenShift Dev Spaces or specifying a private automation hub URL. While optional, these configurations enhance the user experience and functionality of the plug-ins.

Enable Red Hat Developer Hub authentication

Enable Red Hat Developer Hub (RHDH) authentication to integrate with Source Control Management (SCM) systems. This capability is necessary for the plug-ins to create repositories.

- To enable Red Hat Developer Hub authentication refer to the [Enabling authentication in Red Hat Developer Hub](#).

Configure Ansible plug-ins optional integrations

The Ansible plug-ins provide integrations with Ansible Automation Platform and other optional Red Hat products.

- To edit your custom ConfigMap, log in to the OpenShift UI and navigate to **Select Project (developerHubProj) > ConfigMaps > {developer-hub}-app-config-rhdh > app-config-rhdh**.

Configure OpenShift Dev Spaces

When OpenShift Dev Spaces is configured for the Ansible plug-ins, users can click a link from the catalog item view in Red Hat Developer Hub and edit their provisioned Ansible Git projects using Dev Spaces.

Before you begin

- You have a Dev Spaces installation. For more details, see [Installing Dev Spaces](#).

NOTE:

OpenShift Dev Spaces is a separate product and it is optional. The plug-ins will function without it.

It is a separate Red Hat product and is not included in the Ansible Automation Platform or Red Hat Developer Hub subscription.

If the OpenShift Dev Spaces link is not configured in the Ansible plug-ins, the **Go to OpenShift Dev Spaces dashboard** link in the **DEVELOP** section of the Ansible plug-ins landing page redirects users to the [Ansible development tools home page](#).

Procedure

1. Edit your custom Red Hat Developer Hub config map, for example `app-config-rhdh`.
2. Add the following code to your Red Hat Developer Hub `app-config-rhdh.yaml` file.

```
data:
  app-config-rhdh.yaml: |-
    ansible:
      devSpaces:
        baseUrl: >-
          https://<Your OpenShift Dev Spaces URL>
```

3. Replace `<Your OpenShift Dev Spaces URL>` with your OpenShift Dev Spaces URL.
4. In the OpenShift Developer UI, select the `Red Hat Developer Hub` pod.
5. Open **Actions**.
6. Click **Restart rollout**.

Configure the private automation hub URL

Private automation hub provides a centralized, on-premise repository for certified Ansible collections, execution environments and any additional, vetted content provided by your organization.

Before you begin

- You have a private automation hub instance. For more information, see [Manage your automation content in private automation hub](#)

If the private automation hub URL is not configured in the Ansible plug-ins, users are redirected to the [Red Hat Hybrid Cloud Console automation hub](#).

NOTE:

The private automation hub configuration is optional but recommended. The Ansible plug-ins will function without it.

Procedure

1. Edit your custom Red Hat Developer Hub config map, for example `app-config-rhdh`.
2. Add the following code to your Red Hat Developer Hub `app-config-rhdh.yaml` file.

```
data:
  app-config-rhdh.yaml: |-
    ansible:
      ...
      automationHub:
        baseUrl: '<https://MyOwnPAHUrl>'
      ...
```

3. Replace `<https://MyOwnPAHUrl/>` with your private automation hub URL.
4. In the OpenShift Developer UI, select the `Red Hat Developer Hub` pod.
5. Open **Actions**.
6. Click **Restart rollout**.

Example ConfigMaps

The following examples demonstrate how the required and optional settings for the Ansible Dev Tools Server, Ansible Automation Platform, software templates, and other integrations are correctly formatted within their respective YAML files.

app-config-rhdh ConfigMap example for Ansible plug-ins

This example details necessary settings like the creatorService URL, optional integrations for Ansible Automation Platform and OpenShift Dev Spaces, and the addition of Ansible software templates to the catalog.

```

kind: ConfigMap
...
metadata:
  name: app-config-rhdh
  ...
data:
  app-config-rhdh.yaml: |-
    ansible:
      creatorService:
        baseUrl: 127.0.0.1
        port: '8000'
      # Optional integrations
      rhaap:
        baseUrl: '<https://MyControllerUrl>'
        token: '<AAP Personal Access Token>'
        checkSSL: <true or false>
      devSpaces:
        baseUrl: '<https://MyDevSpacesURL>'
      automationHub:
        baseUrl: '<https://MyPrivateAutomationHubURL>'

    ...
  catalog:
    locations:
      - type: url
        target: https://github.com/ansible/ansible-rhdh-templates/blob/main/
all.yaml
    rules:
      - allow: [Template]

    ...

```

Full Helm chart config example for Ansible plug-ins

This example provides a full YAML configuration for the Helm chart, illustrating the precise structure needed to integrate the Ansible plug-ins into the Red Hat Developer Hub installation.

```
global:
  ...
dynamic:
  ...
plugins:
  - disabled: false
    integrity: <SHA512 Integrity key for ansible-plugin-backstage-rhaap plugin>
    package: 'http://plugin-registry:8080/ansible-plugin-backstage-rhaap-
dynamic-x.y.z.tgz'
    pluginConfig:
      dynamicPlugins:
        frontend:
          ansible.plugin-backstage-rhaap:
            appIcons:
              - importName: AnsibleLogo
                name: AnsibleLogo
            dynamicRoutes:
              - importName: AnsiblePage
                menuItem:
                  icon: AnsibleLogo
                  text: Ansible
                  path: /ansible
        - disabled: false
          integrity: <SHA512 Integrity key for ansible-plugin-scaffolder-backend-
module-backstage-rhaap plugin>
          package: >-
            http://plugin-registry:8080/ansible-plugin-scaffolder-backend-module-
backstage-rhaap-dynamic-x.y.z.tgz
          pluginConfig:
            dynamicPlugins:
              backend:
```

```

        ansible.plugin-scaffolder-backend-module-backstage-rhaap: null
    ...
    upstream:
        backstage:
            ...
            extraAppConfig:
                - configMapRef: app-config-rhdh
                  filename: app-config-rhdh.yaml
            extraContainers:
                - command:
                    - adt
                    - server
                  image: >-
                    registry.redhat.io/ansible-automation-platform-25/ansible-dev-tools-
rhel8:latest
                  imagePullPolicy: IfNotPresent
                  name: ansible-devtools-server
                  ports:
                    - containerPort: 8000
    ...

```

Install with the Ansible Automation Platform Operator on OpenShift Container Platform

The following procedures describe how to install Ansible plug-ins in Red Hat Developer Hub instances on Red Hat OpenShift Container Platform using the Operator.

Prerequisites

To proceed, you must have Red Hat Developer Hub installed on Red Hat OpenShift Container Platform (OCP) and a valid subscription to Red Hat Ansible Automation Platform.

- Red Hat Developer Hub installed on Red Hat OpenShift Container Platform.

- For Helm installation, follow the steps in the Installing Red Hat Developer Hub on OpenShift Container Platform with the Helm chart section of *Installing Red Hat Developer Hub on OpenShift Container Platform*.
- For Operator installation, follow the steps in the Installing Red Hat Developer Hub on OpenShift Container Platform with the Operator section of *Installing Red Hat Developer Hub on OpenShift Container Platform*.
- A valid subscription to Red Hat Ansible Automation Platform.
- An OpenShift Container Platform instance with the appropriate permissions within your project to create an application.
- The Red Hat Developer Hub instance can query the automation controller API.
- Optional: To use the integrated learning paths, you must have outbound access to `developers.redhat.com`.

Related information

[Installing Red Hat Developer Hub on OpenShift Container Platform with the Helm chart](#)
[Installing Red Hat Developer Hub on OpenShift Container Platform with the Operator](#)

Recommended RHDH preconfiguration

Red Hat recommends performing the following initial configuration tasks in Red Hat Developer Hub (RHDH). However, you can install the Ansible plug-ins for Red Hat Developer Hub before completing these tasks.

- Authentication in Red Hat Developer Hub
- Authorization in Red Hat Developer Hub

NOTE:

Red Hat provides a repository of software templates for RHDH that uses the `publish:github` action. To use these software templates, you must install the required GitHub dynamic plugins.

Related information

[Authentication in Red Hat Developer Hub](#)
[Authorization in Red Hat Developer Hub](#)
[Repository of software templates for RHDH](#)

Add a sidecar container for Ansible development tools

Add a sidecar container for Ansible development tools in the Developer Hub pod. To do this, you must modify the base ConfigMap for the Red Hat Developer Hub deployment.

Procedure

1. In the OpenShift console, select the **Topology** view.
2. Click **More actions**  on the developer-hub instance and select **Edit backstage** to open the **Backstage details** page.
3. Select the **YAML** tab.
4. In the editing pane, add a `containers` block in the `spec.deployment.patch.spec.template.spec` block:

```
apiVersion: rhdh.redhat.com/v1alpha4
kind: Backstage
metadata:
  name: developer-hub
spec:
  deployment:
    patch:
      spec:
        template:
          spec:
            containers:
              - command:
                  - adt
                  - server
                image: registry.redhat.io/ansible-automation-platform-26/
ansible-dev-tools-rhel8:latest
                imagePullPolicy: always
              ports:
                - containerPort: 8000
                  protocol: TCP
            terminationMessagePolicy: file
```

5. Click **Save**.

NOTE:

If you want to add extra environment variables to your deployment, you can add them in the `spec.application.extraEnvs` block:

```
spec:
  application:
    ...
    extraEnvs:
      envs:
        - name: <env_variable_name>
          value: <env_variable_value>
```

Choose a plug-in delivery method when using the Operator

Ansible plug-ins for Red Hat Developer Hub support two delivery methods. Choose the method that fits your environment.

- **OCI container (recommended):** Red Hat Developer Hub pulls the Ansible plug-ins directly from `registry.redhat.io` as OCI artifacts during startup.
- **HTTP plug-in registry:** Manually download the Ansible plug-ins tarball file and deploy an HTTP plug-in registry in your OpenShift cluster. Use this method if your environment cannot pull OCI artifacts from `registry.redhat.io`.

Complete one of the following procedures before configuring the dynamic plug-ins.

Use the OCI container delivery with the Operator

Red Hat Developer Hub pulls the Ansible plug-ins directly from `registry.redhat.io` as OCI artifacts. This method requires a registry authentication secret in the same OpenShift project as your Red Hat Developer Hub deployment.

Before you begin

- You have a Red Hat account with access to `registry.redhat.io`.

- You have a registry service account token from the Red Hat Customer Portal. For more information, see [Registry Service Accounts](#).
- You have access to the OpenShift project where you had installed Red Hat Developer Hub.
- You have installed the OpenShift CLI (`oc`) and logged in to your cluster.

Procedure

1. Create an `auth.json` file on your local machine with your `registry.redhat.io` credentials:

```
{
  "auths": {
    "registry.redhat.io": {
      "auth": "<base64-encoded-username:password>"
    }
  }
}
```

To generate the base64-encoded value, use the `printf '%s' '<username>:<password>' | base64` command.

2. Create the authentication secret in the OpenShift project where you had installed Red Hat Developer Hub.

The secret name must follow the pattern `<deployment-name>-dynamic-plugins-registry-auth`, where `<deployment-name>` matches the `metadata.name` of your Backstage custom resource.

- For a default Operator installation with name `developer-hub`:

```
oc create secret generic developer-hub-dynamic-plugins-registry-auth \
  --from-file=auth.json=./auth.json
```

- If you already have Podman credentials configured locally:

```
oc create secret generic <deployment-name>-dynamic-plugins-registry-auth \
  --from-file=auth.json=${XDG_RUNTIME_DIR}/containers/auth.json
```

Step result:

IMPORTANT:

Create this secret in the same OpenShift project as your Red Hat Developer Hub deployment, and create it before you configure the plug-ins.

Result

Verify that the secret exists in the project:

```
oc get secret <deployment-name>-dynamic-plugins-registry-auth
```

Download the Ansible plug-ins files

Download the Ansible plug-ins for Red Hat Developer Hub **Setup Bundle** from the Red Hat Ansible Automation Platform Product Software downloads page.

Procedure

1. In a browser, navigate to the [Red Hat Ansible Automation Platform Product Software downloads page](#) and select the **Product Software** tab.
2. Click **Download now** next to **Ansible plug-ins for Red Hat Developer Hub Setup Bundle** to download the latest version of the plug-ins.

The format of the filename is `ansible-rhdh-plugins-x.y.z.tar.gz`. Substitute the Ansible plug-ins release version, for example `2.0.0`, for `x.y.z`.

3. Create a directory on your local machine to store the `.tar` files.

```
$ mkdir /path/to/<ansible-backstage-plugins-local-dir-changeme>
```

4. Set an environment variable (`$DYNAMIC_PLUGIN_ROOT_DIR`) to represent the directory path.

```
$ export DYNAMIC_PLUGIN_ROOT_DIR=/path/to/<ansible-backstage-plugins-local-dir-changeme>
```

5. Extract the `ansible-rhdh-plugins-<version-number>.tar.gz` contents to `$DYNAMIC_PLUGIN_ROOT_DIR`.

```
$ tar --exclude='*code*' -xzf ansible-rhdh-plugins-x.y.z.tar.gz -C $DYNAMIC_PLUGIN_ROOT_DIR
```

Substitute the Ansible plug-ins release version, for example `2.0.0`, for `x.y.z`.

Result

Run `ls` to verify that the extracted files are in the `$DYNAMIC_PLUGIN_ROOT_DIR` directory:

```
$ ls $DYNAMIC_PLUGIN_ROOT_DIR
ansible-plugin-backstage-rhaap-dynamic-x.y.z.tgz
ansible-plugin-backstage-rhaap-dynamic-x.y.z.tgz.integrity
ansible-plugin-scaffolder-backend-module-backstage-rhaap-dynamic-x.y.z.tgz
ansible-plugin-scaffolder-backend-module-backstage-rhaap-dynamic-x.y.z.tgz.integrity
```

The files with the `.integrity` file type contain the plugin SHA value. The SHA value is used during the plug-in configuration.

Create a registry for the Ansible plug-ins

Set up a registry in your OpenShift cluster to host the Ansible plug-ins and make them available for installation in Red Hat Developer Hub (RHDH).

Procedure

1. Log in to your OpenShift Container Platform instance with credentials to create a new application.
2. Open your Red Hat Developer Hub OpenShift project.

```
$ oc project <YOUR_DEVELOPER_HUB_PROJECT>
```

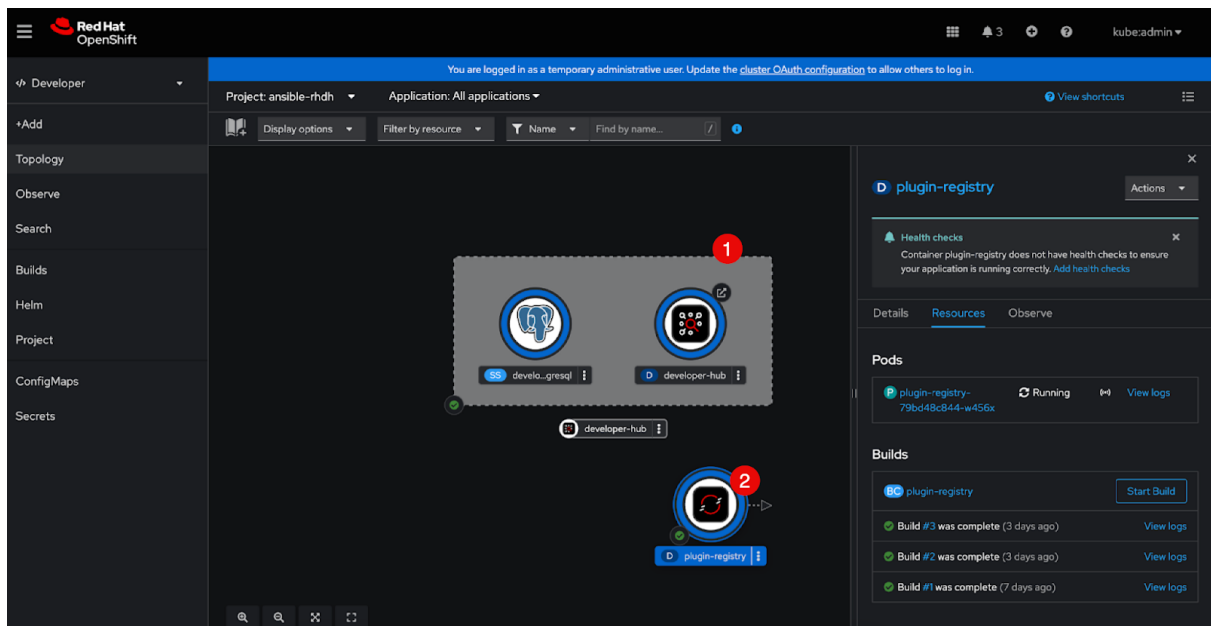
3. Run the following commands to create a plug-in registry build in the OpenShift cluster.

```
$ oc new-build httpd --name=plugin-registry --binary
$ oc start-build plugin-registry --from-dir=$DYNAMIC_PLUGIN_ROOT_DIR --wait
$ oc new-app --image-stream=plugin-registry
```

Result

To verify that the plugin-registry was deployed successfully, open the **Topology** view in the **Developer** perspective on the Red Hat Developer Hub application in the OpenShift Web console.

1. Click the plug-in registry to view the log.



(1) Developer hub instance

(2) Plug-in registry

2. Click the terminal tab and login to the container.
3. In the terminal, run `ls` to confirm that the `.tar` files are in the plugin registry.

```
ansible-plugin-backstage-rhaap-dynamic-x.y.z.tgz
ansible-plugin-scaffolder-backend-module-backstage-rhaap-dynamic-x.y.z.tgz
```

The version numbers and file names can differ.

Install the dynamic plug-ins

To install the dynamic plugins, add them to your ConfigMap for your Red Hat Developer Hub plugin settings (for example, `rhaap-dynamic-plugins-config`).

If you have not already created a ConfigMap file for your Red Hat Developer Hub plugin settings, create one by following the procedure in the Creating and using config maps section of the Red Hat OpenShift Container Platform Nodes guide.

Procedure

1. Select **ConfigMaps** in the navigation pane of the OpenShift console.
2. Select the `rhaap-dynamic-plugins-config` ConfigMap from the list.
3. Select the **YAML** tab to edit the `rhaap-dynamic-plugins-config` ConfigMap.
4. Add the Ansible plug-ins. Choose the configuration that matches your delivery method.
 - **OCI container delivery method (recommended):**

```

kind: ConfigMap
apiVersion: v1
metadata:
  name: rhaap-dynamic-plugins-config
data:
  dynamic-plugins.yaml: |
    includes:
      - dynamic-plugins.default.yaml
    plugins:
      - disabled: false
        package: 'oci://registry.redhat.io/ansible-automation-platform/
automation-portal:2.1!ansible-plugin-backstage-rhaap'
        pluginConfig:
          dynamicPlugins:
            frontend:
              ansible.plugin-backstage-rhaap:
                appIcons:
                  - importName: AnsibleLogo
                    name: AnsibleLogo
                dynamicRoutes:
                  - importName: AnsiblePage
                    menuItem:
                      icon: AnsibleLogo
                      text: Ansible
                      path: /ansible
            - disabled: false
              package: 'oci://registry.redhat.io/ansible-automation-platform/
automation-portal:2.1!ansible-plugin-scaffolder-backend-module-backstage-
rhaap'
              pluginConfig:
                dynamicPlugins:
                  backend:
                    ansible.plugin-scaffolder-backend-module-backstage-rhaap:
null

```

Replace `x.y.z` with the Ansible plug-ins version.

- **HTTP plug-in registry method:**

```

kind: ConfigMap
apiVersion: v1
metadata:
  name: rhaap-dynamic-plugins-config
data:
  dynamic-plugins.yaml: |
    includes:
      - dynamic-plugins.default.yaml
    plugins:
      - disabled: false
        package: 'http://plugin-registry:8080/ansible-plugin-backstage-
rhaap-dynamic-x.y.z.tgz'
        integrity: <SHA512 value>
        pluginConfig:
          dynamicPlugins:
            frontend:
              ansible.plugin-backstage-rhaap:
                appIcons:
                  - importName: AnsibleLogo
                    name: AnsibleLogo
                dynamicRoutes:
                  - importName: AnsiblePage
                    menuItem:
                      icon: AnsibleLogo
                      text: Ansible
                      path: /ansible
            - disabled: false
              package: >-
                http://plugin-registry:8080/ansible-plugin-scaffolder-backend-
module-backstage-rhaap-dynamic-x.y.z.tgz
                integrity: <SHA512 value>
                pluginConfig:
                  dynamicPlugins:
                    backend:
                      ansible.plugin-scaffolder-backend-module-backstage-rhaap:
null

```

5. Click **Save**.

Result

1. In the OpenShift console, select the **Topology** view.
2. Click the **Open URL** icon on the deployment pod to open your Red Hat Developer Hub instance in a browser window.

The Ansible plug-in is present in the navigation pane. If you select **Administration**, you can see the installed plug-ins in the **Plugins** tab.

Configure the plug-ins

Configure Ansible plug-ins for Red Hat Developer Hub to customize how users access and create Ansible content. Configure the plug-ins to control access to software templates and enable optional integrations that enhance the development environment.

Configuring the plug-ins helps you to:

- **Control access to templates:** Configure role-based access control to determine who can create projects from templates and who can view them.
- **Enable project provisioning:** Configure the Ansible development tools server and add software templates to enable playbook and collection project creation.
- **Enhance development workflows:** Configure optional integrations to extend functionality based on your environment needs.

Add a custom ConfigMap

Create a custom Red Hat Developer Hub ConfigMap, typically named `app-config-rhdh`, to store custom application settings.

- Create a Red Hat Developer Hub ConfigMap following the procedure in the [Creating and using config maps](#) section of the OpenShift Container Platform *Nodes* guide. The following examples use a custom ConfigMap named `app-config-rhdh`.

To edit your custom ConfigMap, log in to the OpenShift UI and navigate to **Select Project (developerHubProj) > ConfigMaps > {developer-hub}-app-config > EditConfigMaps > app-config-rhdh**.

Configure the Ansible development tools Server

The `creatorService` URL is required for the Ansible plug-ins to provision new projects using the provided software templates.

Procedure

1. Edit your custom Red Hat Developer Hub config map, `app-config-rhdh`, that you created in [Adding a custom ConfigMap](#).
2. Add the following code to your Red Hat Developer Hub `app-config-rhdh.yaml` file.

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: app-config-rhdh
...
data:
  app-config-rhdh.yaml: |-
    ansible:
      creatorService:
        baseUrl: 127.0.0.1
        port: '8000'
...
```

Configure Ansible Automation Platform details

Connect Red Hat Developer Hub to your automation controller by configuring the Ansible Automation Platform details. This configuration uses a Personal Access Token (PAT) to authenticate the plug-ins, which allows them to interact with your automation environment.

NOTE:

The Ansible plug-ins continue to function regardless of the Ansible Automation Platform subscription status.

Procedure

1. Create a Personal Access Token (PAT) with "read and write" scope in automation controller, following the [Applications](#) section of *Access management and authentication*.
2. Edit your custom Red Hat Developer Hub config map, for example `app-config-rhdh`.
3. Add your Ansible Automation Platform details to `app-config-rhdh.yaml`.
 - a. Set the `baseUrl` key with your automation controller URL.
 - b. Set the `token` key with the generated token value that you created in Step 1.
 - c. Set the `checkSSL` key to `true` or `false`.

If `checkSSL` is set to `true`, the Ansible plug-ins verify whether the SSL certificate is valid.

```
data:
  app-config-rhdh.yaml: |
    ...
  ansible:
    ...
  rhaap:
    baseUrl: '<https://MyControllerUrl>'
    token: '<AAP Personal Access Token>'
    checkSSL: true
```

NOTE:

You are responsible for protecting your Red Hat Developer Hub installation from external and unauthorized access. Manage the backend authentication key like any other secret. Meet strong password requirements, do not expose it in any configuration files, and only inject it into configuration files as an environment variable.

Add Ansible plug-ins software templates

Ansible Automation Platform provides software templates for Red Hat Developer Hub to provision new playbooks and collection projects based on Ansible best practices.

Procedure

1. Edit your custom Red Hat Developer Hub config map, for example `app-config-rhdh`.
2. Add the following code to your Red Hat Developer Hub `app-config-rhdh.yaml` file.

```
data:
  app-config-rhdh.yaml: |
    catalog:
      ...
    locations:
      ...
      - type: url
        target: https://github.com/ansible/ansible-rhdh-templates/blob/main/
all.yaml
  rules:
    - allow: [Template]
```

Related information

[Managing templates](#)

Configure Role Based Access Control

Red Hat Developer Hub offers Role-based Access Control (RBAC) functionality. RBAC can then be applied to the Ansible plug-ins content.

- Assign the following roles:
 - Members of the `admin:superUsers` group can select templates in the **Create** tab of the Ansible plug-ins to create playbook and collection projects.
 - Members of the `admin:users` group can view templates in the **Create** tab of the Ansible plug-ins.

The following example adds RBAC to Red Hat Developer Hub.

```

data:
  app-config-rhdh.yaml: |
    plugins:
      ...
    permission:
      enabled: true
      rbac:
        admin:
          users:
            - name: user:default/<user-scm-ida>
          superUsers:
            - name: user:default/<user-admin-idb>

```

For more information about permission policies and managing RBAC, refer to the [Authorization in Red Hat Developer Hub](#) guide for Red Hat Developer Hub.

Uninstall from a Helm installation on OpenShift Container Platform

To uninstall the Ansible plug-ins, you must remove any software templates that use the `ansible:content:create` action from Red Hat Developer Hub, and remove the plug-ins configuration from the Helm chart in OpenShift.

Uninstall a Helm chart installation

To uninstall Ansible plug-ins from a Helm chart, first remove templates through the `ansible:content:create` action. Then, delete the plug-in config and `extraContainers` from the YAML, clear the ConfigMap Ansible block, restart the deployment, and remove the registry app.

Procedure

1. In Red Hat Developer Hub, remove any software templates that use the `ansible:content:create` action.
2. In the OpenShift Developer UI, navigate to **Helm > developer-hub > Actions > Upgrade > Yaml view**.
3. Remove the Ansible plug-ins configuration under the `plugins` section.

```

...
global:
...
  plugins:
    - disabled: false
      integrity: <SHA512 value>
      package: 'http://plugin-registry:8080/ansible-plugin-backstage-rhaap-
dynamic-x.y.z.tgz'
      pluginConfig:
        dynamicPlugins:
          frontend:
            ansible.plugin-backstage-rhaap:
              appIcons:
                - importName: AnsibleLogo
                  name: AnsibleLogo
              dynamicRoutes:
                - importName: AnsiblePage
                  menuItem:
                    icon: AnsibleLogo
                    text: Ansible
                    path: /ansible
    - disabled: false
      integrity: <SHA512 value>
      package: >-
        http://plugin-registry:8080/ansible-plugin-scaffolder-backend-
module-backstage-rhaap-dynamic-x.y.z.tgz
      pluginConfig:
        dynamicPlugins:
          backend:
            ansible.plugin-scaffolder-backend-module-backstage-rhaap: null

```

4. Remove the `extraContainers` section.

```

upstream:
  backstage: |
    ...
    extraContainers:
      - command:
          - adt
          - server
        image: >-
            registry.redhat.io/ansible-automation-platform-25/ansible-dev-tools-
            rhel8:latest
        imagePullPolicy: IfNotPresent
        name: ansible-devtools-server
        ports:
          - containerPort: 8000
    image:
      pullPolicy: Always
      pullSecrets:
        - ...
        - rhdh-secret-registry
    ...

```

5. Click **Upgrade**.
6. Edit your custom Red Hat Developer Hub config map, for example `app-config-rhdh`.
7. Remove the `ansible` section.

```

data:
  app-config-rhdh.yaml: |
    ...
    ansible:
      analytics:
        enabled: true
      devSpaces:
        baseUrl: '<https://MyOwnDevSpacesUrl/>'
      creatorService:
        baseUrl: '127.0.0.1'
        port: '8000'
      rhaap:
        baseUrl: '<https://MyAapSubscriptionUrl>'
        token: '<TopSecretAAPTToken>'
        checkSSL: true
      automationHub:
        baseUrl: '<https://MyOwnPAHUrl/>'

```

8. Restart the Red Hat Developer Hub deployment.
9. Remove the `plugin-registry` OpenShift application.

```
oc delete all -l app=plugin-registry
```

Uninstall an Operator installation on OpenShift Container Platform

To delete the dynamic plug-ins from your installation, you must edit the ConfigMaps that reference Ansible.

The deployment auto reloads when the ConfigMaps are updated. You do not need to reload the deployment manually.

Remove the Ansible plug-ins from the ConfigMap

To remove the Ansible plug-ins from an Operator installation, you must edit the custom ConfigMap that references the plug-ins. You can either delete the entire plug-in entry block or simply disable the plug-ins by setting the `disabled` attribute to `true`.

Procedure

1. Open the custom ConfigMap where you referenced the Ansible plug-ins. For this example, the ConfigMap name is `rhaap-dynamic-plugins-config`.
2. Locate the dynamic plug-ins in the `plugins:` block.
 - To disable the plug-ins, update the `disabled` attribute to `true` for the three plug-ins.
 - To delete the plug-ins, delete the lines that reference the plug-ins from the `plugins:` block:

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: rhaap-dynamic-plugins-config
data:
  dynamic-plugins.yaml: |
    ...
    plugins: # Remove the Ansible plug-ins entries below the 'plugins'
             YAML key
      - disabled: false
        package: 'http://plugin-registry:8080/ansible-plugin-backstage-
rhaap-dynamic-x.y.z.tgz'
        integrity: <SHA512 value>
    ...
      - disabled: false
        package: >-
          http://plugin-registry:8080/ansible-plugin-scaffolder-backend-
module-backstage-rhaap-dynamic-x.y.z.tgz
        integrity: <SHA512 value>
    ...
```

3. Click **Save**.

Remove Ansible Automation Platform and Dev Spaces from the custom Red Hat Developer Hub ConfigMap

To remove Ansible Automation Platform and Dev Spaces configuration from an Operator installation, you must edit the custom Red Hat Developer Hub ConfigMap.

Procedure

1. Open the custom Red Hat Developer Hub ConfigMap where you added configuration for the templates and for connecting to Ansible Automation Platform and Dev Spaces. In this example, the Red Hat Developer Hub ConfigMap name is `app-config-rhdh`.

```

kind: ConfigMap
apiVersion: v1
metadata:
  name: rhdh-app-config
data:
  app-config-custom.yaml: |
    ...
  catalog:
    ...
    locations: # Remove the YAML entry below the 'locations' YAML key
      - type: url
        target: https://github.com/ansible/ansible-rhdh-templates/blob/main/
all.yaml
    rules:
      - allow: [Template]
    ...
  # Remove the entire 'ansible' YAML key and all sub-entries
  ansible:
    devSpaces:
      baseUrl: '<https://YOUR_DEV_SPACES_URL>'
    creatorService:
      baseUrl: '127.0.0.1'
      port: '8000'
    rhaap:
      baseUrl: '<https://YOUR_AAP_URL>'
      token: <REDACTED>
      checkSSL: false

```

2. Remove the `url` in the `locations:` block to delete the templates from the RHDH instance.
3. Remove the `ansible:` block to delete the Ansible-specific configuration.
4. Click **Save**.

Uninstall the sidecar container

To remove the sidecar container for Ansible development tools from the developer-hub pod, you must modify the base ConfigMap for the Red Hat Developer Hub deployment.

Procedure

1. In the OpenShift console, select the **Topology** view.
2. Click **More actions** ⋮ on the developer-hub instance and select **Edit backstage** to edit the base ConfigMap.
3. Select the **YAML** tab.
4. In the editing pane, remove the `containers` block for the sidecar container from the `spec.deployment.patch.spec.template.spec` block:

```
...
spec:
  deployment:
    patch:
      spec:
        template:
          spec:
            containers:
              - command:
                  - adt
                  - server
                image: ghcr.io/ansible/community-ansible-dev-tools:latest
                imagePullPolicy: always
                ports:
                  - containerPort: 8000
                    protocol: TCP
                terminationMessagePolicy: file
```

5. Click **Save**.

Install self-service automation portal (OpenShift Container Platform only)

Self-service automation portal makes enterprise automation accessible to users of varying roles and skill levels through a simplified web interface designed for business users, not automation experts.

Instead of requiring users to understand Ansible playbooks or complex automation workflows, self-service automation portal provides a streamlined "point-and-click" experience.

Self-service automation portal uses your existing Ansible Automation Platform setup: it uses the same logins, same security controls, and the same automation logic.

Job templates in Ansible Automation Platform are synced to self-service automation portal, where they appear as auto-generated self-service templates. Users can launch the auto-generated self-service templates in self-service automation portal to run the corresponding job templates in Ansible Automation Platform.

When launching a job, users follow step-by-step guided forms in self-service automation portal. These forms are identical to the forms for the equivalent job templates in Ansible Automation Platform.

You can also import custom self-service templates in self-service automation portal. These templates are pulled from a git repository, and their configuration file associates them with a job template in Ansible Automation Platform. The configuration file contains the forms for the template.

You can associate more than one custom self-service template with one Ansible Automation Platform job template. You set the RBAC for the custom self-service templates in self-service automation portal. Therefore you can set up different types of forms for users with different levels of automation experience.

For example, for a job template that configures network settings, you could associate a custom template with minimum scope for adjusting settings to a group with less automation experience, and use a more detailed custom template with deeper scope for configuration for automation experts.

Self-service automation portal connects with Red Hat Ansible Automation Platform using an OAuth application for authentication.

The following restrictions apply:

- You can only use one Ansible Automation Platform instance.
- You can only use one Ansible Automation Platform organization.

Architecture and licensing

Self-service automation portal is built on Red Hat Developer Hub (RHDH), which provides the web interface and plugin framework. Your Ansible Automation Platform subscription includes the use of RHDH for the self-service portal without requiring a separate RHDH subscription.

RHDH provides the foundational features for the self-service portal, including:

- User authentication and authorization
- Plugin architecture for custom integrations
- Web interface components and navigation
- Template and catalog management

The self-service portal extends RHDH with Ansible Automation Platform-specific plugins that enable automation job execution, template synchronization, and integration with Ansible Automation Platform organizations and job templates.

For information about support policies and the lifecycle for self-service automation portal, see *Supported platforms*.

Supported platforms

Self-service automation portal supports installation using a Helm chart on OpenShift Container Platform.

- For information about the self-service automation portal lifecycle and supported platforms, see the [Ansible Automation Platform self-service automation portal lifecycle](#) page on the Red Hat customer portal.
- For information about the self-service automation portal support policy, see the [Ansible Automation Platform self-service automation portal support policy](#) page on the Red Hat customer portal.

Interactive demonstration

You can explore the functionality of self-service automation portal in an interactive demonstration:

- Launch the [self-service automation portal interactive demo](#).

Prerequisites

Review the mandatory subscriptions, permissions, and platform access required before starting the installation. Fulfilling these needs helps ensure a successful deployment.

- You have a valid subscription to Red Hat Ansible Automation Platform.
- You have access to an instance of Red Hat Ansible Automation Platform 2.6 with the appropriate permissions to create an OAuth application.
- You have access to an OpenShift Container Platform instance with the appropriate permissions within your project to create an application.
- You have installed `oc`, the OpenShift command-line interface (CLI) tool, on your local machine. You can use `oc` commands in your terminal to interact with your OpenShift cluster.
- You have installed Helm 3.10 or newer. Helm is a package manager for applications on OpenShift Container Platform.

Related information

[Ansible Automation Platform self-service automation portal lifecycle \(for supported OpenShift Container Platform versions\)](#)

[Getting started with the OpenShift CLI](#)

[Installing Helm](#)

Pre-installation configuration

You can deploy self-service automation portal from a Helm chart on OpenShift Container Platform.

Helm is a tool that simplifies deployment of applications on Red Hat OpenShift Container Platform clusters. Helm uses a packaging format called Helm charts. A Helm chart is a package of files that define how an application is deployed and managed on OpenShift. The Helm chart for self-service automation portal is available in the OpenShift Helm catalog.

Select an organization

You must choose an organization in Ansible Automation Platform to use with self-service automation portal. If you have not set up an organization, use the following procedure to create one.

Procedure

1. Open your Ansible Automation Platform instance in a browser and log in as an administrator.
2. Navigate to **Access Management > Organizations**.

3. Click **Create organization**.
4. Enter a unique **Name** and optionally provide a **Description** for your organization.
5. Click **Next** to show the **Review** pane.
6. Click **Finish** to create your organization.

Create an OAuth application

To use the Helm chart to deploy self-service automation portal, you must have set up an OAuth application on your Ansible Automation Platform instance.

However, you cannot run automation on your Ansible Automation Platform instance until you have deployed your self-service automation portal Helm chart, because the OAuth configuration requires the URL for your deployment.

Create the OAuth Application on your Ansible Automation Platform instance, using a placeholder name for the deployment URL.

After deploying self-service automation portal, you must [replace the placeholder value with a URL derived from your deployment URL](#) in your OAuth application.

The steps below describe how to create an OAuth Application in the Ansible Automation Platform Platform console.

Procedure

1. Open your Ansible Automation Platform instance in a browser and log in.
2. Navigate to **Access Management > OAuth Applications**.
3. Click **Create OAuth Application**.
4. Complete the fields in the form.
 - **Name**: Add a name for your application.
 - **Organization**: Choose the organization.
 - **Authorization grant type**: Choose `Authorization code`.
 - **Client type**: choose `Confidential`.
 - **Redirect URIs**: Add placeholder text for the deployment URL (for example <https://example.com>).

The screenshot shows the 'Create OAuth application' page in the Red Hat Ansible Automation Platform. The left sidebar contains navigation links: Overview, Automation Execution, Automation Decisions, Automation Analytics, Automation Content, Ansible Lightspeed, and Access Management. The 'Access Management' section is expanded, showing sub-links for Authentication Methods, Organizations, Teams, Users, Roles, and OAuth Applications. The main form area is titled 'Create OAuth application' and includes a 'Configure OAuth Application' step. The form fields are: Name (test-application), Organization (Default), URL (Enter OAuth application URL), Description (Enter description), Authorization grant type (Authorization code), Client type (Confidential), and Redirect URIs (https://example.com). A 'Create OAuth application' button is at the bottom.

5. Click **Create OAuth application**.

The **Application information** popup displays the `clientId` and `clientSecret` values.

6. Copy the `clientId` and `clientSecret` values and save them.

These values are used in an OpenShift secret for Ansible Automation Platform authentication.

Enable OAuth token creation for external users

Self-service automation portal uses Ansible Automation Platform for authentication and as an OAuth provider.

You must enable OAuth token creation in Ansible Automation Platform so that users can authenticate with the platform from self-service automation portal.

NOTE:

Users who do not have permission to log in to Ansible Automation Platform cannot log in to Self-service portal, because Ansible Automation Platform provides the OAuth tokens. Therefore, a user who is removed from an external IdP (for example LDAP, SAML, Azure) can no longer log into Ansible Automation Platform or self-service automation portal. This prevents potential external token issues. For more information, refer to the [Manage OAuth2 token creation for external users](#) section of [Configure central authentication for Ansible Automation Platform](#).

Procedure

1. In a browser, log in to your Ansible Automation Platform instance as a user with admin privileges.

2. In the navigation pane, select **Settings > Platform gateway**.
3. Locate the **Allow external users to create OAuth2 tokens** setting.
4. Enable **Allow external users to create OAuth2 tokens** if it is not already enabled:
 - a. Click **Edit platform gateway settings**.
 - b. Set **Allow external users to create OAuth2 tokens** to **Enabled**.

The screenshot shows the 'Platform gateway settings' page in the Red Hat Ansible Automation Platform. The left sidebar contains a navigation menu with options like Overview, Automation Execution, Jobs, Templates, Schedules, Projects, Infrastructure, Administration, Automation Analytics, Access Management, Ansible Lightspeed, and Settings. The 'Settings' menu is expanded, showing 'Subscription', 'Platform gateway' (selected), 'User Preferences', 'System', 'Job', 'Logging', and 'Troubleshooting'. The main content area is titled 'Platform gateway settings' and includes sections for 'Password Security', 'Custom Login', and 'Other settings'. The 'Other settings' section contains several configuration fields. The 'Allow external users to create OAuth2 tokens' setting is highlighted with a red box and is currently set to 'Enabled'. Below the settings are buttons for 'Save platform gateway settings', 'Revert all to default', and 'Cancel'.

- c. Click **Save platform gateway settings** to save your updates and return to the **Platform gateway settings** page.
5. In the **Platform gateway settings** page, verify that the **Allow external users to create OAuth2 tokens** setting is enabled.

Generate an API token for Ansible Automation Platform authentication

You must create an API token in Ansible Automation Platform. The token is used in an OpenShift secret for Ansible Automation Platform authentication.

Procedure

1. Log in to your instance of Ansible Automation Platform as a user with Ansible Automation Platform administrator privileges.
2. Navigate to **Access Management > API Tokens** to display the **API Tokens** page.
3. Click **Create API Token**.
4. Add a description and select your OAuth application.
5. In the **Scope** menu, select **Write**.

6. Click **Create Token** to generate the token.

7. Save the new token.

The token is used in an OpenShift secret that is fetched by the Helm chart.

Install the self-service automation portal Helm chart

You can use the configured secrets and plugin registry to install the self-service automation portal. Deploy the application onto your OpenShift cluster using the provided Helm chart.

Configure the Helm chart from the OpenShift catalog

Deploy the Helm chart from the OpenShift catalog by configuring the base URL and organization name in the YAML view. This launches the self-service automation portal installation.

Before you begin

- You have created a project for self-service automation portal in OpenShift Container Platform.
- You have created secrets in OpenShift Container Platform for Red Hat Ansible Automation Platform authentication.
- If you configure SCM integration (for example, importing from private repositories or using templates that access SCM), you have created secrets in OpenShift for SCM authentication.
- You have completed one of the plug-in delivery methods:
 - For OCI delivery: You have created the `<release-name>-dynamic-plugins-registry-auth` secret.

Procedure

1. In the OpenShift Container Platform web console, select the **Developer** view.
2. Select your project and click the **Helm** option in the OpenShift sidebar.
3. Click **Create** and select **Helm Release**.
4. Search for `Portal` in the Helm Charts filter, and select the **Automation Portal** chart.
5. Click **Create** and select the **YAML view**.
6. Update the `clusterRouterBase` value with the base URL of your OpenShift instance.

IMPORTANT:

You must replace the default `apps.example.com` placeholder value. If the default value remains, Helm chart validation fails.

```
redhat-developer-hub:
  global:
    clusterRouterBase: apps.example.com
```

7. Configure the plug-in delivery mode by setting the `pluginMode` key:
 - For OCI delivery, set the value to `oci`.

```
redhat-developer-hub:
  global:
    pluginMode: oci
```

8. **Optional:** To use a specific plug-in version, update the `imageTagInfo` value:

```
redhat-developer-hub:
  global:
    imageTagInfo: "2.1"
```

9. Set the Red Hat Ansible Automation Platform organization to synchronize. The default value is `Default`. Update the `orgs` key to match your organization name:

```
redhat-developer-hub:
  upstream:
    backstage:
      appConfig:
        catalog:
          providers:
            rhaap:
              '{{- include "catalog.providers.env" . }}':
                orgs: "<your-aap-organization-name>"
```

10. **Optional:** Update the `CUSTOMER_SUPPORT_URL` to point to your support portal:

```
redhat-developer-hub:
  upstream:
    backstage:
      extraEnvVars:
        - name: CUSTOMER_SUPPORT_URL
          value: https://access.redhat.com/support
```

11. Click **Create**.

Related information

[Choose a plug-in delivery method](#)

Verify the installation

Verify the Helm chart installation from the OpenShift Container Platform web console.

Procedure

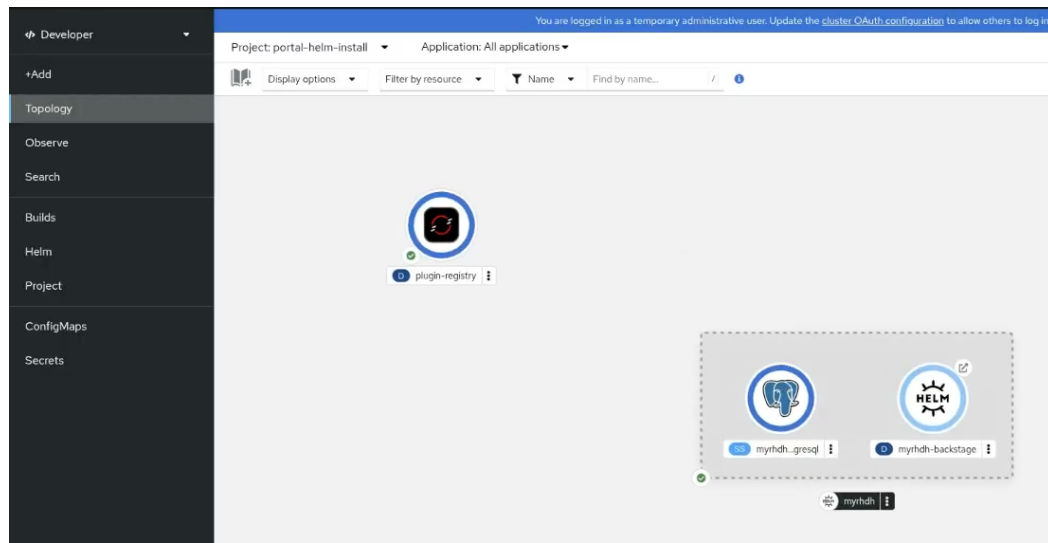
1. In a browser, log in to your OpenShift instance.
2. In the **Developer** view, navigate to the **Topology** view for the namespace where you deployed the Helm chart.

Step result:

The deployment appears with the label `D` on the icon. The name of the deployment is `<installation-name>-backstage`, for example `<my-self-service-automation-portal-backstage>`.

While it is deploying, the icon is light blue. The color changes to dark blue when deployment is complete.

OpenShift deployment topology view



Install in an air-gapped environment

You can install self-service automation portal in a disconnected OpenShift Container Platform environment.

Installing the self-service automation portal in a disconnected or air-gapped OpenShift Container Platform environment requires preparation and configuration to ensure all necessary container images and assets are available locally without internet access.

Prerequisites

Gather the required tools and access credentials necessary for air-gapped installation. This includes the OpenShift CLI, Helm, Podman, and required registry secrets.

- You have installed the OpenShift CLI (`oc`).
- You have installed Helm 3.10 or newer.
- You have installed and configured Podman for pulling and pushing container images.
- You have internet access. This is required to pull images and charts from public repositories, including `registry.redhat.io` and `charts.openshift.io`.
- A Red Hat pull secret, for example `pull-secret.json` or similar credentials file that allows you to pull images from `registry.redhat.io`.
- Sufficient disk space to store downloaded images and chart packages.

- Access to public registries: Docker Hub, `quay.io`, `registry.redhat.io`, and your disconnected OpenShift cluster's internal registry.

Related information

[Getting started with the OpenShift CLI](#)

[Installing Helm](#)

[charts.openshift.io repository](#)

Prepare for air-gapped installation

Before you can install self-service automation portal in a disconnected OpenShift Container Platform environment, you must complete some processes on a connected bastion host.

Mirror container images

Mirror the required container images from the Red Hat registry to your local disconnected registry. This step prepares the images for installing self-service automation portal in an isolated environment.

If you mirror `registry.redhat.io` content to a different registry host (or to a registry prefix such as `quay.io/<org>`), you can set `redhat-developer-hub.global.imageRegistry` so the Helm chart pulls all of its images from that mirrored location. `imageRegistry` is a single override that controls the registry for the base application image, PostgreSQL image, OCI plug-in artifacts, and Ansible Dev Tools sidecar.

The dynamic plug-in init container does not use cluster-level image mirror configuration (for example, `ImageDigestMirrorSet` or `ImageTagMirrorSet`). You must set `imageRegistry` even if your cluster redirects `registry.redhat.io` pulls.

Procedure

1. Log in to `registry.redhat.io`:

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

2. Enter your Red Hat username and password when prompted.
3. Log in to your disconnected registry:

```
$ podman login <disconnected_registry_url>
```

4. Pull the Red Hat Developer Hub image from `registry.redhat.io`:

```
$ podman pull registry.redhat.io/rhdh/rhdh-hub-rhel9:x.y.z
```

5. Tag the image for your disconnected registry:

```
$ podman tag registry.redhat.io/rhdh/rhdh-hub-rhel9:x.y.z  
<disconnected_registry_url>/<your_namespace>/rhdh/rhdh-hub-rhel9:x.y.z
```

6. Push the tagged image to your disconnected registry:

```
$ podman push <disconnected_registry_url>/<your_namespace>/rhdh/rhdh-hub-rhel9:x.y.z
```

7. If you use the built-in PostgreSQL database, mirror the PostgreSQL 15 image. An external database is the supported production architecture and does not require this step:

```
$ podman pull registry.redhat.io/rhel9/postgresql-15:<tag>  
  
$ podman tag registry.redhat.io/rhel9/postgresql-15:<tag>  
<disconnected_registry_url>/<your_namespace>/rhel9/postgresql-15:<tag>  
  
$ podman push <disconnected_registry_url>/<your_namespace>/rhel9/  
postgresql-15:<tag>
```

8. If you use OCI container delivery, mirror the Ansible plug-ins OCI artifacts image:

```
$ podman pull registry.redhat.io/ansible-automation-platform/automation-portal:<plugin-version>  
  
$ podman tag registry.redhat.io/ansible-automation-platform/automation-portal:<plugin-version> <disconnected_registry_url>/<your_namespace>/ansible-automation-platform/automation-portal:<plugin-version>  
  
$ podman push <disconnected_registry_url>/<your_namespace>/ansible-automation-platform/automation-portal:<plugin-version>
```

9. Mirror the Ansible Dev Tools sidecar image. Use the path that matches your Red Hat Ansible Automation Platform version:

Red Hat Ansible Automation Platform 2.5:

```
$ podman pull registry.redhat.io/ansible-automation-platform-25/ansible-dev-
tools-rhel9:latest

$ podman tag registry.redhat.io/ansible-automation-platform-25/ansible-dev-
tools-rhel9:latest <disconnected_registry_url>/<your_namespace>/ansible-
automation-platform-25/ansible-dev-tools-rhel9:latest

$ podman push <disconnected_registry_url>/<your_namespace>/ansible-automation-
platform-25/ansible-dev-tools-rhel9:latest
```

Red Hat Ansible Automation Platform 2.6:

```
$ podman pull registry.redhat.io/ansible-automation-platform-26/ansible-dev-
tools-rhel9:latest

$ podman tag registry.redhat.io/ansible-automation-platform-26/ansible-dev-
tools-rhel9:latest <disconnected_registry_url>/<your_namespace>/ansible-
automation-platform-26/ansible-dev-tools-rhel9:latest

$ podman push <disconnected_registry_url>/<your_namespace>/ansible-automation-
platform-26/ansible-dev-tools-rhel9:latest
```

Download the helm chart package

Download the Helm chart package and modify the internal image references to point to your disconnected registry. This prepares the installation package for the air-gapped environment.

Procedure

1. Add the OpenShift Helm charts repository:

```
$ helm repo add openshift-helm-charts https://charts.openshift.io/
```

2. Update your Helm repositories to fetch the latest chart information:

```
$ helm repo update
```

3. Pull the chart:

```
$ helm pull openshift-helm-charts/redhat-rhaap-portal --version x.y.z
```

This command downloads the chart as a `.tgz` file, for example `redhat-rhaap-portal-1.0.1.tgz`.

4. Unpack the chart:

```
$ tar -xvf redhat-rhaap-portal-x.y.z.tgz
```

This creates a directory with a name similar to `redhat-rhaap-portal-1.0.1/`.

5. Navigate to the unpacked chart directory (for example, `cd redhat-rhaap-portal-1.0.1`) and open the `values.yaml` file in a text editor.
6. Find all the `image:` entries in `values.yaml` and replace the original image references with the full path to the image in your disconnected registry.

For example, replace `image: registry.redhat.io/rhdh/rhdh-hub-rhel9:x.y.z` with `image: <disconnected_registry_url>/<your_namespace>/rhdh-hub-rhel9:x.y.z`

7. Repack the modified chart:

```
$ helm package redhat-rhaap-portal-x.y.z
```

This creates a new `.tgz` file with your changes (for example, `redhat-rhaap-portal-1.0.1.tgz`).

Transfer assets to the disconnected environment

Transfer the modified Helm chart package from the connected bastion host to a machine inside your disconnected network. This action stages the installation assets for deployment within the isolated OpenShift environment.

- Copy the modified Helm chart `.tgz` file or files (for example, `redhat-rhaap-portal-1.0.1.tgz`) from your connected bastion host to a machine or jump box within your disconnected OpenShift network.

Install the Helm chart in the disconnected OpenShift environment

You can install the modified Helm chart using the `helm install` command in your disconnected OpenShift environment. This deploys the self-service automation portal using the locally available assets.

After preparing the disconnected environment with mirrored images and transferred assets, install the Helm chart to deploy the self-service automation portal.

Access the disconnected OpenShift environment

Ensure your disconnected OpenShift cluster is configured to trust the private registry containing the mirrored container images. This step is crucial for successful image pulling during installation.

Before you begin

- You have the necessary kubeconfig and permissions. For example `cluster-admin`, for setting up image pull secrets or insecure registries.

Procedure

1. In a terminal, log in to your disconnected OpenShift cluster using the `oc` CLI.

```
oc login --token=<your_token> --server=<your_openshift_api_url>
```

Use the following command if you have a kubeconfig:

```
export KUBECONFIG=/path/to/your/kubeconfig
oc login
```

2. Ensure that your OpenShift cluster is configured to trust your disconnected registry:
 - a. Use `ImageContentSourcePolicy` for mirroring.
 - b. Use `additionalTrustedCA` in `image.config.openshift.io/cluster` for self-signed certificates.
 - c. Use `insecure-registries` for plain HTTP.

Configure plug-in delivery for disconnected environments

Complete one of the plug-in delivery methods for disconnected environments.

- For OCI delivery: Create the `<release-name>-dynamic-plugins-registry-auth` secret with credentials for your disconnected registry that hosts the mirrored Ansible plug-ins OCI artifacts.

Related information

[Choose a plug-in delivery method](#)

Install the Helm chart

Install self-service automation portal by using the `helm install` command. You must reference the local Helm chart file and include the required configuration using a values file (`-f`) or `--set` flags.

Procedure

1. On the machine within your disconnected environment, navigate to the directory where you placed the transferred Helm chart `.tgz` file:

```
$ cd /path/to/your/transferred/charts/
```

2. Define your namespace and cluster router base as environment variables:

```
$ export MY_NAMESPACE="<your_namespace_name>"  
$ export MY_CLUSTER_ROUTER_BASE="<your_cluster_router_base>"
```

3. If the namespace does not exist, create it:

```
$ oc new-project "${MY_NAMESPACE}"
```

4. Create a custom values file and reuse it for install and upgrade. Choose the example that matches your plug-in delivery method.

OCI delivery (*values-oci.yaml*):

```

redhat-developer-hub:
  global:
    clusterRouterBase: <your_cluster_router_base>
    pluginMode: oci
    imageTagInfo: "<plugin-version>"
    # Registry host only. Replaces registry.redhat.io for all images.
    imageRegistry: "<disconnected_registry_url>"
    # Optional. Full OCI plugin image path for mirrors with non-standard
    # repository paths. Overrides imageRegistry for OCI plugin artifacts only.
    # Use when your mirror does not preserve the default repository path
    # (ansible-automation-platform/automation-portal).
    # ociPluginImage: "<disconnected_registry_url>/custom-path/automation-portal"
  upstream:
    backstage:
      # The Helm chart pins images by SHA256 digest by default.
      # When mirroring with podman tag/push, the digest is not preserved.
      # Override with tag-based references.
      image:
        repository: rhdh/rhdh-hub-rhel9
        tag: "1.8"
      appConfig:
        ansible:
          rhaap:
            checkSSL: true # Set to false if using self-signed certificates.
      catalog:
        providers:
          rhaap:
            '{{- include "catalog.providers.env" . }}':
              orgs: "<your-aap-organization-name>"
      # If you use the built-in PostgreSQL database, uncomment the following
      # lines.
      # An external database is the supported production architecture.
      # postgresql:
      #   image:
      #     repository: rhel9/postgresql-15
      #     tag: "latest"

```

HTTP plug-in registry delivery (*values-tarball.yaml*):

```

redhat-developer-hub:
  global:
    clusterRouterBase: <your_cluster_router_base>
    pluginMode: tarball
    # Registry host only. Replaces registry.redhat.io for the images.
    imageRegistry: "<disconnected_registry_url>"
  upstream:
    backstage:
      image:
        repository: rhdh/rhdh-hub-rhel9
        tag: "1.8"
      appConfig:
        ansible:
          rhaap:
            checkSSL: true # Set to false if using self-signed certificates.
      catalog:
        providers:
          rhaap:
            '{{- include "catalog.providers.env" . }}':
              orgs: "<your-aap-organization-name>"
      # If you use the built-in PostgreSQL database, uncomment the following
      # lines.
      # An external database is the supported production architecture.
      # postgresql:
      #   image:
      #     repository: rhel9/postgresql-15
      #     tag: "latest"

```

IMPORTANT:

Set `imageRegistry` to the registry host only (for example, `mirror.example.com`). Do not include repository paths. The chart appends the default repository path `ansible-automation-platform/automation-portal` automatically. If your mirror uses a different repository structure, set `ociPluginImage` to the full image path instead.

5. Install the chart using the `helm install` command:

```
$ helm install redhat-rhaap-portal \
  redhat-rhaap-portal-x.y.z.tgz \
  --namespace "${MY_NAMESPACE}" \
  -f /path/to/values-oci.yaml
```

To apply changes after deployment, upgrade the Helm release. If you install from the OpenShift web console, use Developer → Helm → select the release → Actions → Upgrade → YAML view.

To upgrade from the command line:

```
$ helm upgrade redhat-rhaap-portal \
  redhat-rhaap-portal-x.y.z.tgz \
  --namespace "${MY_NAMESPACE}" \
  -f /path/to/values-oci.yaml
```

Alternatively, you can pass values using `--set` flags.

OCI delivery:

```
$ helm install redhat-rhaap-portal \
  redhat-rhaap-portal-x.y.z.tgz \
  --namespace "${MY_NAMESPACE}" \
  --set redhat-developer-hub.global.clusterRouterBase="${MY_CLUSTER_ROUTER_BASE}" \
  --set redhat-developer-hub.global.imageRegistry="<disconnected_registry_url>" \
  --set redhat-developer-hub.global.pluginMode=oci
```

To also set `ociPluginImage` via `--set`:

```
--set redhat-developer-hub.global.ociPluginImage="<disconnected_registry_url>/custom-path/automation-portal"
```

HTTP plug-in registry delivery:

```
$ helm install redhat-rhaap-portal \
  redhat-rhaap-portal-x.y.z.tgz \
  --namespace "${MY_NAMESPACE}" \
  --set redhat-developer-hub.global.clusterRouterBase="${MY_CLUSTER_ROUTER_BASE}" \
  --set redhat-developer-hub.global.imageRegistry="<disconnected_registry_url>" \
  --set redhat-developer-hub.global.pluginMode=tarball
```

Configure CA certificates for private registries

If your private registry uses a certificate signed by an internal or self-signed CA, the `install-dynamic-plugins` init container fails with `x509: certificate signed by unknown authority`. Mount the CA certificate into the init container so that `skopeo` trusts the registry.

Procedure

1. Obtain the CA certificate chain that signed your mirror registry's TLS certificate. If the registry uses a certificate chain, include the full chain:

```
$ cat registry.crt intermediate.crt root.crt > ca-bundle.crt
```

2. Create a ConfigMap from the CA certificate:

```
$ oc create configmap registry-ca-crt \
  --from-file=ca.crt=ca-bundle.crt -n <namespace>
```

3. Add the volume and mount to your values file:

```

redhat-developer-hub:
  upstream:
    backstage:
      extraVolumes:
        - name: registry-ca-crt
          configMap:
            name: registry-ca-crt
      initContainers:
        - name: install-dynamic-plugins
      volumeMounts:
        - name: registry-ca-crt
          mountPath: /etc/containers/certs.d/<registry-host>
          readOnly: true

```

IMPORTANT:

The `mountPath` must be the registry hostname only (for example, `/etc/containers/certs.d/mirror.example.com`). Do not include repository paths. If the registry uses a non-standard port, include it in the path (for example, `/etc/containers/certs.d/mirror.example.com:5000`).

Verify the deployment

Verify the successful installation of the Helm chart in the disconnected environment. Check the Helm release status, monitor the pods, and verify that the application routes are accessible.

Procedure

1. Check the Helm release status:

```
$ helm list -n ${MY_NAMESPACE}
```

2. Monitor the pods in your namespace to ensure they are running:

```
$ oc get pods -n ${MY_NAMESPACE}
```

3. Check for `ImagePullBackOff` or other errors in pod events:

```
$ oc describe pod <pod_name> -n ${MY_NAMESPACE}
```

4. If the chart uses routes to expose services, verify that the routes are created and accessible:

```
$ oc get route -n ${MY_NAMESPACE}
```

Troubleshooting disconnected installations

Use this reference to troubleshoot common issues that occur during disconnected self-service automation portal installations.

Symptom	Cause	Solution
authentication required or unauthorized in <code>install-dynamic-plugins</code> init container logs	Auth secret missing or malformed. The init container uses <code>skopeo</code> and does not use cluster pull secrets.	Create <code><release-name>-dynamic-plugins-registry-auth</code> secret. Use <code>base64 -w0</code> to avoid multiline values that corrupt <code>auth.json</code> .
Duplicate path in OCI URL (for example, <code>.../ansible-automation-platform/ansible-automation-platform/...</code>)	<code>imageRegistry</code> includes a repository path instead of the registry host only.	Set <code>imageRegistry</code> to the registry host only. If your mirror uses a different repository structure, use <code>ociPluginImage</code> to set the full image path.
<code>x509: certificate signed by unknown authority</code> in init container logs	Private registry uses a self-signed or internal CA certificate.	Mount the CA certificate into the init container. See See "Configure CA certificates for private registries" on page 1130 .

Verify the configuration

You can inspect the deployment logs and ConfigMap on the OpenShift console to verify that the deployment conforms with the settings in your Helm chart.

View the deployment logs

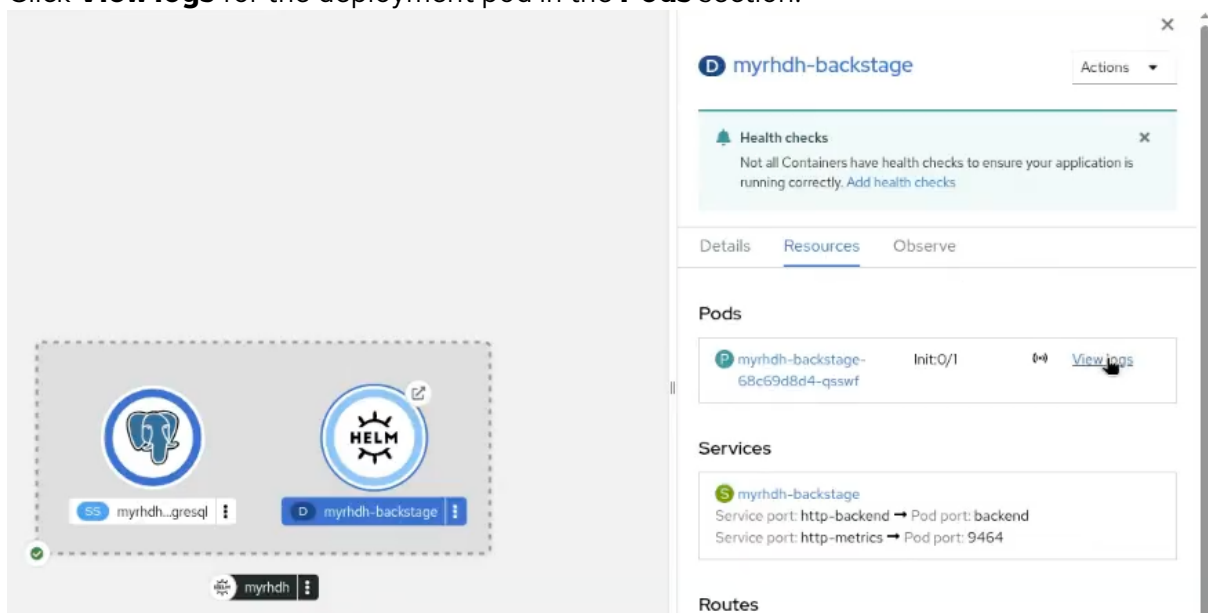
You can view the logs in the OpenShift console. Pay close attention to the `install-dynamic-plugins` container logs. This helps confirm that the required plug-ins installed successfully.

Procedure

1. In a browser, log in to your OpenShift instance.
2. In the **Developer** view, navigate to the **Topology** view for the namespace where you deployed the Helm chart.
3. The deployment appears: the label on the icon is `D`.

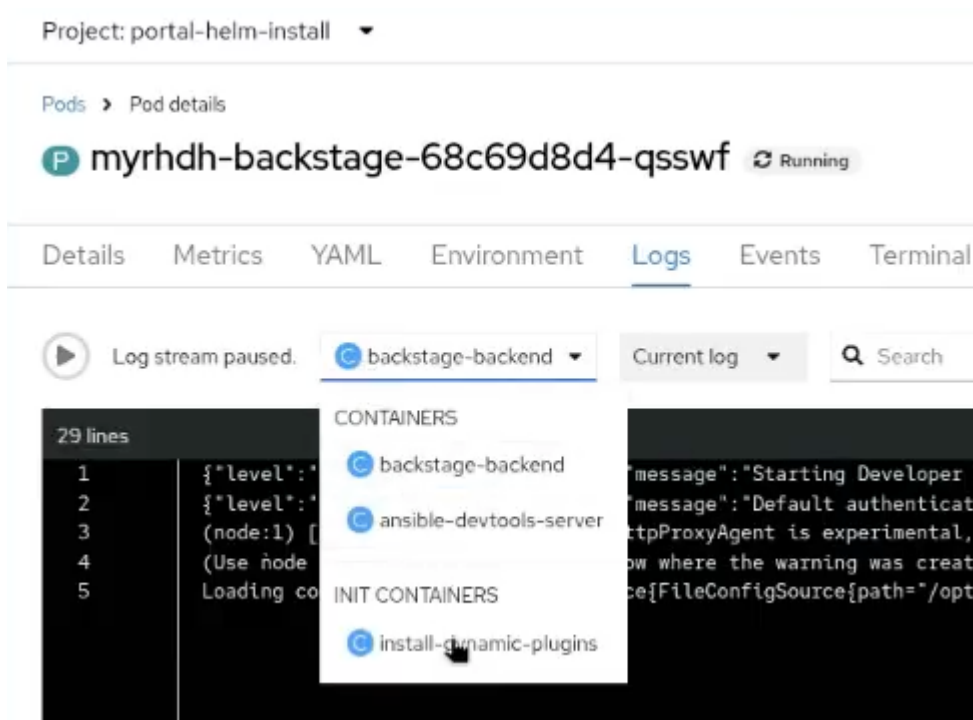
The name of the deployment is `<installation-name>-backstage`, for example `<my-self-service-automation-portal>-backstage`.

4. Click the icon representing the deployment.
5. The **Details** pane for the deployment opens.
6. Select the **Resources** tab.
7. Click **View logs** for the deployment pod in the **Pods** section:



The **Pod details** page opens for the deployment pod.

8. Select the **Logs** tab in the **Pod details** page.
9. To view the install messages, select the `install-dynamic-plugins` container from the **INIT CONTAINERS** section of the dropdown list of containers:



The log stream displays the progress of the installation of the plug-ins from the plug-in registry.

The log stream for successful installation of the plug-ins resembles the following output:

```

===== Installing dynamic plugin http://plugin-registry:8080/ansible-
backstage-plugin-catalog-backend-module-rhaap-dynamic-0.1.0.tgz

*=> Grabbing package archive through pm pack'

•=> Verifying package Integrity

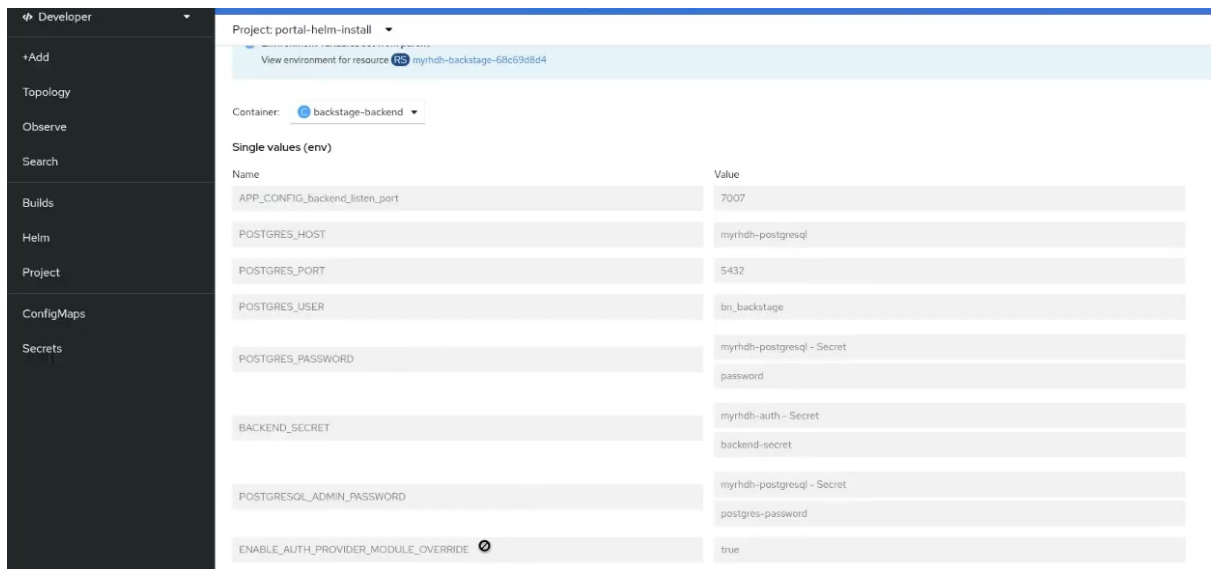
•*> Extracting package archive /dynamic-plugins-root/ansible-backstage-
plugin-catalog-backend-module-rhaap-dynamic-0.1.0.tgz

•*> Removing package archive /dynamic-plugins-root/ansible-backstage-plugin-
catalog-backend-module-rhaap-dynamic-0.1.0.tgz

-> Successfully installed dynamic plugin http://plugin-registry:8080/ansible-
backstage-plugin-catalog-backend-module-rhaap-dynamic-0.1.0.tgz

```

10. Select the **Environment** tab in the **Pod details** page to view the environment variables for the containers. If you set additional environment variables in your Helm chart, check that they are listed here.



Update the deployment URL

Complete the necessary post-installation configuration, including updating the OAuth application and setting up initial Role-Based Access Control (RBAC). You can then access and sign in to the portal.

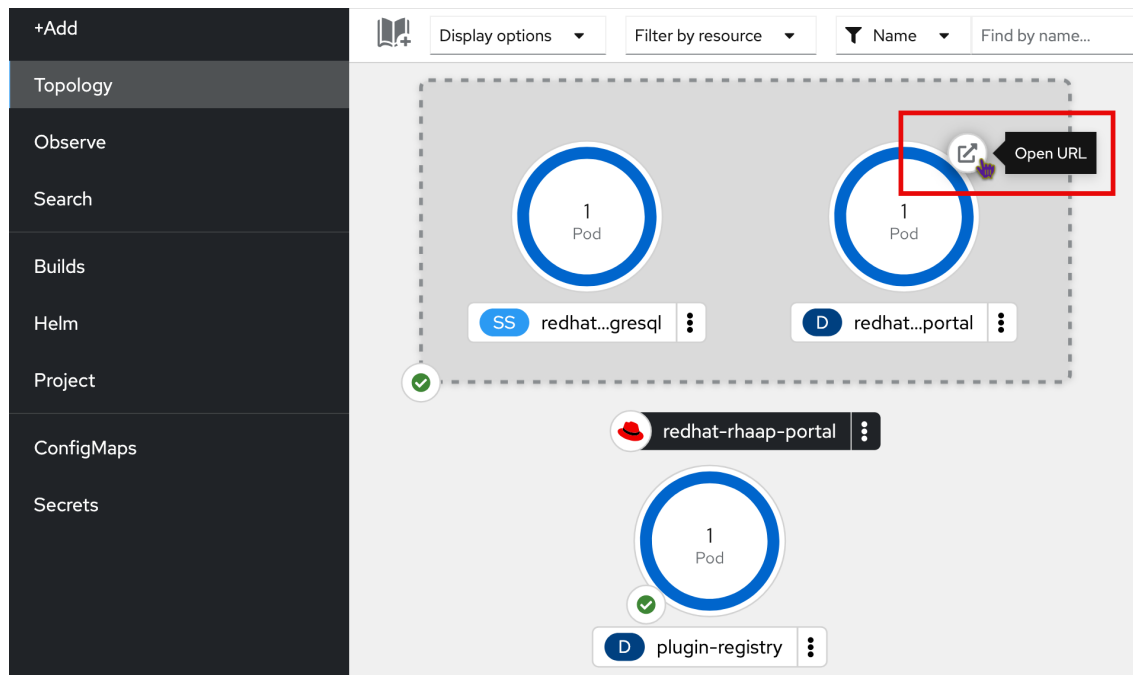
Add the deployment URL to the OAuth Application

When you set up your OAuth application in Ansible Automation Platform before deploying self-service automation portal, you added placeholder text for the `Redirect URIs` value.

You must update this value using the URL from the deployed application so that you can run automation on self-service automation portal from self-service automation portal.

Procedure

1. Determine the `Redirect URI` from your OpenShift deployment:
 - a. Open the URL for the deployment from the OpenShift console to display the sign-in page for self-service automation portal.



- b. Copy the URL for the sign-in page for self-service automation portal.
- c. To determine the `Redirect URI` value, append `/api/auth/rhaap/handler/frame` to the end of the deployment URL.

For example, if the URL for the deployment is `https://my-automation-portal-project.mycluster.com`, then the `Redirect URI` value is `https://my-automation-portal-project.mycluster.com/api/auth/rhaap/handler/frame`.

2. Update the `Redirect URIs` field in the OAuth application in Ansible Automation Platform:
 - a. In a browser, open your instance of Ansible Automation Platform.
 - b. Navigate to **Access Management > OAuth Applications**.
 - c. In the list view, click the OAuth application you created.
 - d. Replace the placeholder text in the `Redirect URIs` field with the value you determined from your OpenShift deployment.
 - e. Click `Save` to apply the changes.

Configure custom SSL certificates for self-service automation portal

If your Ansible Automation Platform instance uses custom or self-signed SSL certificates, you must configure self-service automation portal to trust those certificates to avoid SSL verification errors.

Before you begin

- You have administrator access to your OpenShift Container Platform cluster.

- You have the custom Certificate Authority (CA) certificate file used by your Ansible Automation Platform instance.
- Self-service automation portal is installed in your OpenShift Container Platform cluster.

Procedure

1. Obtain the CA certificate file from your Ansible Automation Platform instance.

If you do not have the CA certificate file, you can extract it from your Ansible Automation Platform server:

```
openssl s_client -showcerts -connect <aap-hostname>:443 </dev/null 2>/dev/null
| openssl x509 -outform PEM > aap-ca-cert.pem
```

Replace `<aap-hostname>` with your Ansible Automation Platform hostname.

2. Log in to your OpenShift Container Platform cluster with administrator privileges.
3. Create a ConfigMap containing your custom CA certificate:

```
oc create configmap custom-ca-bundle \
  --from-file=ca-bundle.crt=aap-ca-cert.pem \
  -n <namespace>
```

Replace `<namespace>` with the namespace where self-service automation portal is installed.

4. Update your self-service automation portal Helm chart values to mount the custom CA certificate:

```
upstream:
  backstage:
    extraEnvVarsSecrets:
      - custom-ca-bundle
    extraVolumes:
      - name: custom-ca
        configMap:
          name: custom-ca-bundle
    extraVolumeMounts:
      - name: custom-ca
        mountPath: /etc/pki/ca-trust/source/anchors/
        readOnly: true
```

5. Apply the updated configuration by upgrading the self-service automation portal Helm chart:

```
helm upgrade <release-name> <chart-name> \  
-f values.yaml \  
-n <namespace>
```

Replace `<release-name>` with your Helm release name and `<chart-name>` with the self-service automation portal chart name.

6. Wait for the self-service automation portal pods to restart with the new configuration.

Result

1. Verify that the self-service automation portal pods are running:

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

All self-service automation portal pods should show a status of `Running`.

2. Attempt to sign in to self-service automation portal using your Ansible Automation Platform credentials.
If the SSL certificate configuration is correct, you can authenticate successfully without SSL verification errors.
3. Check the self-service automation portal logs for SSL-related errors:

```
oc logs -n <namespace> <pod-name> | grep -i ssl
```

If you see no SSL verification errors, the custom CA certificate is trusted correctly.

If you continue to experience SSL verification errors after following this procedure:

- Verify that the CA certificate file contains the complete certificate chain.
- Ensure that the certificate file is in PEM format.
- Confirm that the Ansible Automation Platform hostname in your configuration matches the hostname in the SSL certificate.
- Check that the `checkSSL` parameter in your Helm values is set to `true` (the default). Setting it to `false` disables SSL verification entirely, which is not recommended for production environments.

Sign in to self-service automation portal

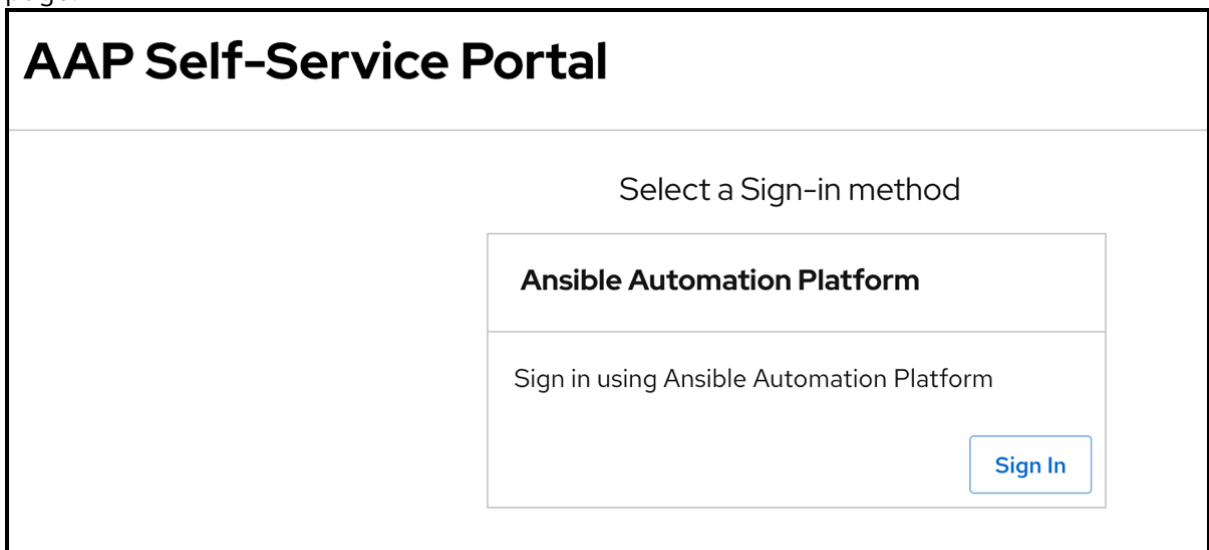
Log in to the deployed self-service automation portal using your existing Ansible Automation Platform credentials. The portal uses these credentials for authentication.

Before you begin

- You have configured an OAuth application in Ansible Automation Platform for self-service automation portal.
- You have configured a user account in Ansible Automation Platform.

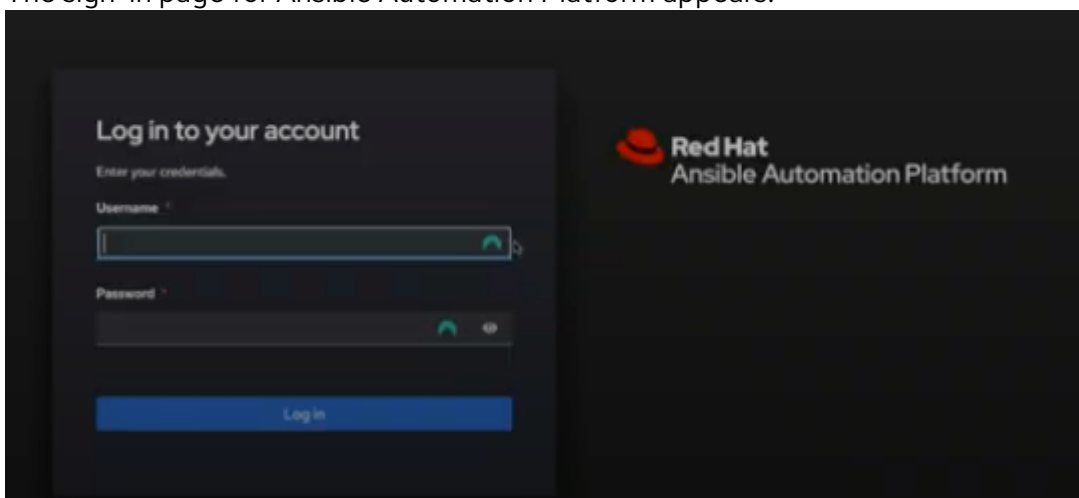
Procedure

1. In a browser, navigate to the URL for self-service automation portal to open the sign-in page.



The screenshot shows the 'AAP Self-Service Portal' interface. At the top, it says 'Select a Sign-in method'. Below this, there is a box titled 'Ansible Automation Platform' with the text 'Sign in using Ansible Automation Platform' and a 'Sign In' button.

2. Click **Sign In**.
3. The sign-in page for Ansible Automation Platform appears:



The screenshot shows the 'Log in to your account' page for the Red Hat Ansible Automation Platform. It features a dark background with a light gray login form. The form includes fields for 'Username' and 'Password', each with a green eye icon for toggling visibility. A blue 'Log in' button is at the bottom of the form. The Red Hat logo and 'Ansible Automation Platform' text are visible on the right side of the page.

4. Enter your Ansible Automation Platform credentials and click **Log in**.

5. The self-service automation portal web console opens.

If you are using custom or self-signed SSL certificates and when attempting to log in to self-service automation portal, it displays the error:

```
Login failed; caused by Error: Failed to send POST request: fetch failed
```

This error indicates that self-service automation portal cannot verify the SSL certificate from your Ansible Automation Platform instance.

To resolve this issue, configure self-service automation portal to trust your custom CA certificate.

NOTE:

While you can disable SSL validation by setting `checkSSL: false` in the Helm chart configuration, this approach is not recommended for production environments as it reduces security. Instead, configure self-service automation portal to trust your custom CA certificate.

Set up initial RBAC rules in self-service automation portal

After you install self-service automation portal and synchronize it with Ansible Automation Platform, only users with Ansible Automation Platform administrator privileges can view the auto-generated templates.

You must configure initial Role-Based Access Control (RBAC) permissions to allow non-admin users to view and execute synchronized Ansible Automation Platform job templates.

IMPORTANT:

Role-Based Access Control (RBAC) differs by template type:

- - **Auto-generated templates:** Permissions are synchronized from Ansible Automation Platform. Users must have permissions on the underlying Ansible Automation Platform job template.
 - **Custom templates:** Permissions must be explicitly configured within the Self-Service Automation Portal. Users must also have permissions to run the associated job templates in Ansible Automation Platform.

Understand the permission model

Self-service automation portal and Ansible Automation Platform use separate but related permission systems. Ansible Automation Platform RBAC is the source of truth for synchronization scope and execution permissions.

Self-service automation portal RBAC:

- Controls which users can view templates in the self-service automation portal catalog.
- Controls which users can access portal templates and submit jobs.

Ansible Automation Platform RBAC:

- **Controls synchronization scope:** Only Ansible Automation Platform job templates accessible by the configured Ansible Automation Platform token (ansible.rhaap.token) are synchronized to self-service automation portal.
- **Controls Ansible Automation Platform job template visibility and execution:** Ansible Automation Platform permissions determine whether authenticated users can view and execute Ansible Automation Platform job templates in self-service automation portal.
- **Validates execution permissions:** When a self-service automation portal user executes a template, Ansible Automation Platform checks that user's execute permissions before launching the job.

NOTE:

If a user can see a self-service automation portal template in the catalog but lacks Ansible Automation Platform execution permissions for the associated Ansible Automation Platform job template in Ansible Automation Platform, the user cannot run the Ansible Automation Platform Job.

Configure RBAC for synchronization

Synchronization uses an Ansible Automation Platform token configured in the self-service automation portal to determine which data to synchronize from Ansible Automation Platform.

Before you begin

- You have credentials for an Ansible Automation Platform administrator.
- Synchronization of Ansible Automation Platform Organization information from Ansible Automation Platform is complete.

- Users who execute Ansible Automation Platform job templates through self-service automation portal must have job template **Execute** permissions assigned in Ansible Automation Platform for the respective Ansible Automation Platform job templates.
- The **Allow external users to create OAuth2 tokens** setting is enabled in the **Settings > Platform gateway** settings in Ansible Automation Platform.

By default, self-service automation portal synchronizes Ansible Automation Platform Organization, Team, and User identity information. Self-service automation portal also synchronizes Ansible Automation Platform job template information accessible by the configured Ansible Automation Platform token.

Procedure

1. Log in to self-service automation portal with an account that has Ansible Automation Platform administrator privileges.
2. In the navigation pane of self-service automation portal, select **Administration > RBAC**.
3. Click **Create** to create a new role and enter a name, for example `portal-users`.
4. Click **Next**.

The screenshot shows the Red Hat Ansible Automation Platform self-service portal. The top navigation bar includes the Red Hat logo, a search bar, and a user profile dropdown for 'Platform Admin'. The left sidebar contains navigation links for 'Templates' and 'History', and a main menu with 'Administration' (selected) and 'RBAC'. The main content area is titled 'RBAC / Create role' and 'Create role'. The 'Create Role' form is displayed with three steps: 1. Enter name and description of role, 2. Add users and groups, and 3. Add permission policies. Step 1 is active, showing input fields for 'Name' (containing 'auto-generated-templates-role'), 'Description', and 'Owner'. A 'Next' button is visible below the 'Owner' field. A 'Cancel' button is located at the bottom right of the form.

5. In the **Users and Groups** section, select the Ansible Automation Platform teams and users to assign to this role, then click **Next**.

Create role

Enter name and description of role

Add users and groups

Select users and groups *

Name	Type	Members	Actions
cloud	Group	5	
network	Group	5	
windows	Group	5	

5 Rows 1/2

Back Next

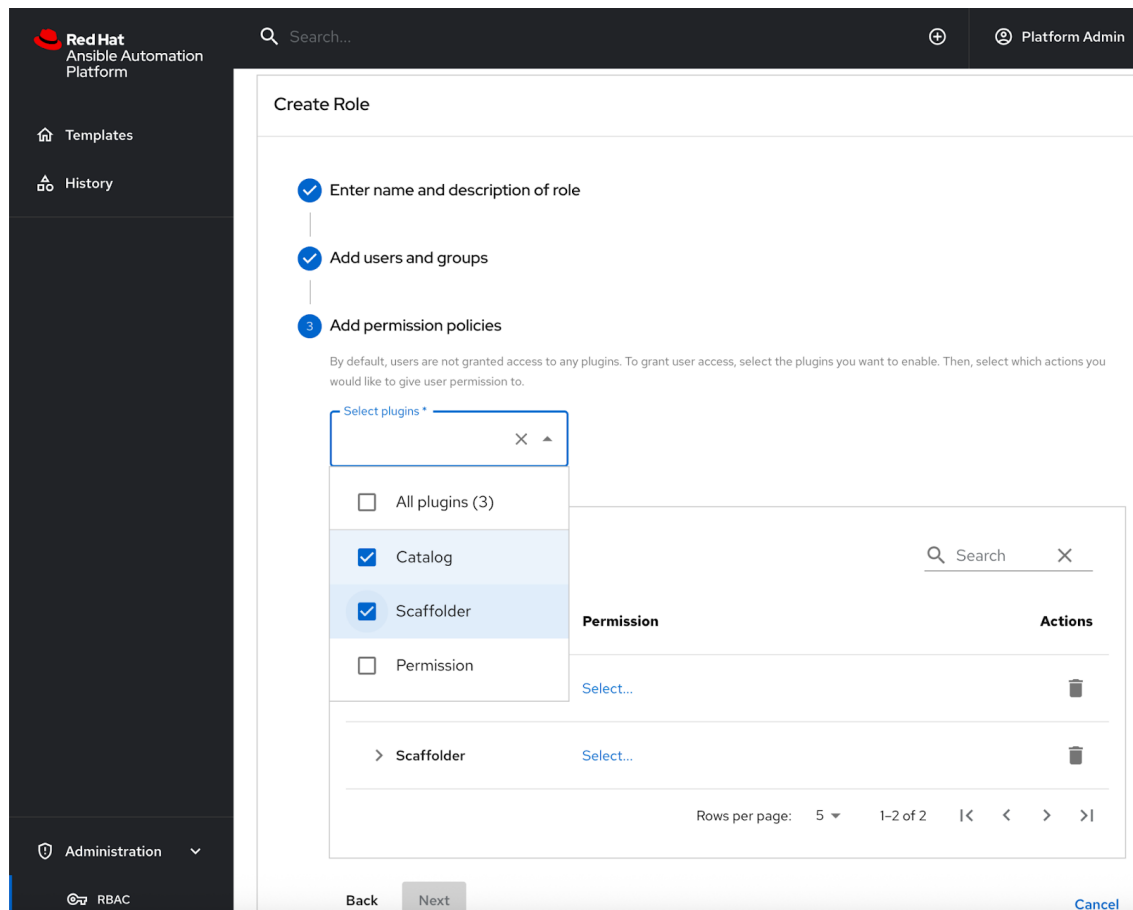
Add permission policies Cancel

NOTE:

The **Members** column displays the total count of users in each team, including both regular team members and administrators.

You can only select Ansible Automation Platform teams and users from the Ansible Automation Platform Organization that you have configured for synchronization with self-service automation portal.

6. Click **Next** to configure permissions in the **Add permission policies** section:
 - a. Select the **Catalog** plugin from the **Select plugins** dropdown menu.
 - b. Select the checkbox for `catalog.entity.read`.



7. Select the **Scaffolder** plugin and enable all scaffolder permissions:

- `scaffolder.template.parameter.read`
- `scaffolder.template.step.read`
- `scaffolder.action.execute`
- `scaffolder.task.cancel`
- `scaffolder.task.create`
- `scaffolder.task.read`

NOTE:

The `scaffolder.task.read` permission must be enabled so that users can view previous task runs in the **History** page in the self-service automation portal console.

8. Click **Next** to review your settings, then **Create** to create the new role.

On successful completion, your new role is included in the **All roles** list when you select **Administration > RBAC** in the navigation pane in self-service automation portal.

Result

On successful completion, your new role is included in the **All roles** list when you select **Administration > RBAC** in the navigation pane in self-service automation portal.

Related information

[Ansible Automation Platform token best practices](#)

Configure conditional access

Optionally, you can configure conditional self-service automation portal RBAC policies to filter role access to specific Ansible Automation Platform job templates by tag for specific Ansible Automation Platform teams or users.

Before you begin

- Ansible Automation Platform job templates must have Ansible Automation Platform labels applied and synchronized with self-service automation portal.
- Users who execute Ansible Automation Platform job templates through self-service automation portal must have Ansible Automation Platform job template **Execute** permissions assigned in Ansible Automation Platform for the respective Ansible Automation Platform job templates.

Ansible Automation Platform labels applied to Ansible Automation Platform job templates are synchronized to self-service automation portal as tags and can be used for conditional access control.

NOTE:

Ansible Automation Platform labels are converted to lowercase tags with special characters replaced by hyphens (for example, the Ansible Automation Platform label `Network-Automation` becomes the tag `network-automation`).

Procedure

1. Log in to self-service automation portal with an account that has Ansible Automation Platform administrator privileges.
2. In the navigation pane of self-service automation portal, select **Administration > RBAC**.
3. Click **Create** to create a new role and enter a name, for example `network-templates`.
4. Click **Next**.
5. In the **Users and Groups** section, select the Ansible Automation Platform teams and users to assign to this role (for example, the Ansible Automation Platform `network-team`), then click **Next**.

RBAC > Create role
Create role

Create Role

Enter name and description of role

Add users and groups

Select users and groups *

Name	Type	Members	Actions
cloud	Group	5	
network	Group	5	
windows	Group	5	

Back Next

Add permission policies

Cancel

NOTE:

The **Members** column displays the total count of users in each team, including both regular team members and administrators.

You can only select Ansible Automation Platform teams and users from the Ansible Automation Platform Organization that you are using in self-service automation portal.

6. Click **Next** to configure permissions in the **Add permission policies** section:
 - Select the **Catalog** plugin from the **Select plugins** dropdown menu.
 - Select the checkbox for `catalog.entity.read`.
 - Click **Conditional** to configure a condition-based policy.
7. In the condition builder, configure a rule to filter by tag:
 - **Rule:** Select `HAS_METADATA` from the dropdown menu
 - **Key:** Enter `tags`
 - **Value:** Enter the tag value to filter by (for example, `network-automation`)
8. Select the **Scaffolder** plugin and enable all scaffolder permissions:
 - `scaffolder.template.parameter.read`
 - `scaffolder.template.step.read`
 - `scaffolder.action.execute`
 - `scaffolder.task.cancel`
 - `scaffolder.task.create`
 - `scaffolder.task.read`
9. Click **Next** to review your settings, then click **Create** to create the new role.

Result

On successful completion, your new role is included in the **All roles** list when you select **Administration > RBAC** in the navigation pane in self-service automation portal.

1. Log in to self-service automation portal as a non-Ansible Automation Platform administrator user who is a member of a team you granted permissions to.
2. Verify that the user can see auto-generated templates in self-service automation portal.
 - If you configured conditional access by tag, the user should see only templates with the specified tags.
 - If you did not configure conditional access, the user should see all Ansible Automation Platform job templates for which they have job template **Execute** permissions in Ansible Automation Platform.
3. To verify execution permissions work correctly, attempt to execute a template:
 - a. If the user has job template **Execute** permissions in Ansible Automation Platform for the template, the user can view the template, and the job launches successfully.

Permissions reference for Ansible Automation Platform job template access

Permissions for Ansible Automation Platform job templates

Permission	Resource type	Policy	Description
<code>catalog.entity.read</code>	catalog-entity	read	Users can view synchronized Ansible Automation Platform job templates in the self-service automation portal.
<code>scaffolder.template.parameter.read</code>	scaffolder-template	read	Users can read template parameters.
<code>scaffolder.action.execute</code>	scaffolder-action	use	Users can execute actions through templates.
<code>scaffolder.task.create</code>		create	Users can trigger the execution of Ansible Automation Platform job templates.
<code>scaffolder.task.read</code>		read	Users can view task execution history and logs on the History page.
<code>scaffolder.task.cancel</code>		use	Users can cancel currently running templates.

Adjust synchronization frequency between Ansible Automation Platform and self-service automation portal

The Helm chart defines how frequently users, teams and organization configuration information is synchronized from Ansible Automation Platform to self-service automation portal.

The frequency is set by the `catalog.providers.rhaap.schedule.frequency` key. By default, the synchronization occurs hourly.

- To adjust the synchronization frequency, edit the value for the `catalog.providers.rhaap.schedule.frequency` key in the Helm chart.

```
catalog:
  ...
  providers:
    rhaap:
      '{{- include "catalog.providers.env" . }}':
        schedule:
          frequency: {minutes: 60}
          timeout: {seconds: 30}
```

NOTE:

Increasing the synchronization frequency generates extra traffic.

Bear this in mind when deciding the frequency, particularly if you have a large number of users.

Install automation dashboard to calculate savings (RHEL only)

By effectively using automation dashboard, you can gain valuable insights into your Ansible Automation Platform usage and drive continuous improvement in your automation practices.

Install automation dashboard

Install automation dashboard to collect and analyze key metrics related to job execution, efficiency, and automation savings across your Ansible Automation Platform deployments.

Before you begin

- One of the following tested configurations:
 - RHEL 9 or RHEL 10 on x86 or ARM-based physical or virtual hosts.
 - With an external database: PostgreSQL v15 database.

IMPORTANT: Do not install automation dashboard on the same host(s) as Ansible Automation Platform.

- Automation dashboard installation has been tested with the following configuration:
 - 80 GB hard drive (depending on data growth)
 - 4 vCPUs x 16 GB RAM
 - Disk IOPS - 3000
 - Handle up to 10,000 jobs/month and 47M summaries/month
 - Connects to three Ansible Automation Platform deployments of the same version
- Access to *baseos* and *Ansible Automation Platformstream* repository packages for the RHEL 9 or RHEL 10 host.
- A non-root login account to the RHEL 9 or RHEL 10 host for installation.
 - The login account requires `sudo` access to root.
 - By default, we use the `$HOMEDIR` of the user account.
- URL details for access to your Ansible Automation Platform instances.
- An Ansible Automation Platform OAuth2 token, which is used for communication between the Ansible Automation Platform instances and automation dashboard.
- Access to download the installation bundle providing installation components for the automation dashboard.
- Open firewall access to allow for bidirectional communication between Ansible Automation Platform instances and the automation dashboard.
 - This includes HTTPS/443 (or your Ansible Automation Platform configured port) from the dashboard to the Ansible Automation Platform instance(s).
 - Port 8447 is the default ingress port for the automation dashboard. This port can be configured during installation.

- RHEL firewall ports that might block 5432 to PostgreSQL.
- A supported version of `ansible-core` installed on supported RHEL versions.

Procedure

1. Download the latest installer tar file from access.redhat.com. Navigate to Downloads > Red Hat Ansible Automation Platform Product Software.
2. Copy the installation source file to your RHEL 9 or RHEL 10 host.
3. Extract the installation source. This will require ~500Mb. of disk space. Throughout this example we will use the ec2-user home directory: `/home/<username>`.

```
tar -xzf ansible-automation-dashboard-containerized-setup-bundle-0.1-
x86_64.tar.gz

cd ansible-automation-dashboard-containerized-setup/
```

4. Install the required `ansible-core` package.

```
cd ansible-automation-dashboard-containerized-setup

sudo dnf install ansible-core
```

5. Create an application `client_id/client_secret` in your Ansible Automation Platform instance:

- a. Create an OAuth2 application using the following steps :

- i. **For Ansible 2.4:**

- Navigate to https://AAP_CONTROLLER_FQDN/#/applications

- ii. **For Ansible 2.5 and 2.6:**

- Navigate to https://AAP_GATEWAY_FQDN/access/applications

- iii. Add the following information:

- **Name:** automation-dashboard-sso
- **Authorization grant type:** authorization-code
- **Organization:** Default
- **Redirect URIs:** https://AUTOMATION_DASHBOARD_FQDN/auth-callback
- **Client type:** Confidential

NOTE: The values for **Name**, **Organization**, and HTTPS port number for Ansible Automation Platform are configurable. The examples provided in this document assume use of port 443.

- b. Save the `client_id` and `client_secret` information inputs into the inventory file.
 - c. Create an Ansible Automation Platform access token:
 - i. Navigate to `https://AAP_GATEWAY_FQDN/#/users/<id>/tokens`, and create a token using the following information:
 - OAuth application: `automation-dashboard-sso`
 - Scope: `read`
 - ii. Store the access token and refresh token values. `clusters.yaml` uses these tokens.
6. Copy the example inventory and change it before running the installation program.

```
cp -i inventory.example inventory
vi inventory
```

IMPORTANT:

- This is an example tested inventory containing default values for Ansible Automation Platform 2.4, 2.5, and 2.6.
- You must change the following values to use this inventory configuration in your environment:
 - Change the RHEL 9 or RHEL 10 host occurrences from `host.example.com` to your FQDN host
 - Change the phrase `TODO` to match your passwords within all `_admin_password` or `_pg_password` values.
 - Change the `dashboard_pg_host` value to the IP address or DNS name of the database server.
- For more information, see the [Inventory variables](#) section of this document.

```
# This is our automation dashboard front-end application
[automationdashboard]
host.example.com ansible_connection=local

# These are required vars for the installation and should not be removed
[automationdashboard:vars]
# Configure AAP OAuth2 authentication.
# aap_auth_provider_name - name as shown on login page.
aap_auth_provider_name=Ansible Automation Platform
# aap_auth_provider_protocol - http or https
```

```

aap_auth_provider_protocol=https
# AAP version - 2.4, 2.5 or 2.6
aap_auth_provider_aap_version=2.6
# aap_auth_provider_host - AAP IP or DNS name, with optional port
aap_auth_provider_host=my-aap.example.com
# aap_auth_provider_check_ssl - enforce TLS check or not.
aap_auth_provider_check_ssl=true
# aap_auth_provider_client_id and aap_auth_provider_client_secret -
# they are obtained from AAP when OAuth2 application is created in AAP.
aap_auth_provider_client_id=TODO
aap_auth_provider_client_secret=TODO

# Specify amount of old data to synchronize after installation.
# The initial_sync_days=N requests N days of old data, counting from "today".
# The initial_sync_since requests data from the specified date until "today".
# If both are specified, the initial_sync_since will be used.
initial_sync_days=1
# initial_sync_since=2025-08-08

# Hide warnings when insecure https request are made.
# Use this if your AAP uses self-signed TLS certificate.
# show_urllib3_insecure_request_warning=False

# Force clean install-like
# dashboard_update_secret=true

# HTTP/HTTPS settings
# nginx_disable_https=true
# Change nginx_http_port or nginx_https_port if you want to access dashboard
on a different TCP port.
# nginx_http_port=8083
# nginx_https_port=8447
# TLS certificate configuration
# The dashboard_tls_cert needs:
# - contain server certificate, intermediate CA certificates and root CA
certificate.

```

```

# - the server certificate must be the first one in the file.
# dashboard_tls_cert=/path/to/tls/dashboard.crt
# dashboard_tls_key=/path/to/tls/dashboard.key

# Enable Django DEBUG.
# django_debug=True

[database]
host.example.com ansible_connection=local

[redis]
host.example.com ansible_connection=local

[all:vars]
redis_mode=standalone

[all:vars]
postgresql_admin_username=postgres
postgresql_admin_password=TODO

# registry_username=
# registry_password=
# AAP Dashboard - mandatory
# -----
dashboard_pg_containerized=True
dashboard_admin_password=TODO

# The value of dashboard_pg_host needs to resolve to an IP address of the
# database.
# It cannot be localhost or 127.0.0.1.
# It can be IP address of the server.

dashboard_pg_host=host.example.com
dashboard_pg_username=aapdashboard
dashboard_pg_password=TODO
dashboard_pg_database=aapdashboard

bundle_install=true

#

```

```
# Relevant if bundle_install=false

# Set container_image_update to True to fetch most recent container images
from registry.

# container_image_update=True

#

# Relevant if bundle_install=true

# <full path to the bundle directory>

bundle_dir='{{ lookup("ansible.builtin.env", "PWD") }}/bundle'

# AAP Dashboard - optional

# -----

# Set to True to expose Django admin page.

# nginx_dashboard_admin_exposed=False
```

7. Install the required Ansible collections. You must complete this step to prevent module resolution errors during the installation.

```
ansible-galaxy collection install -r requirements.yml
```

8. Run the installation program.

```
ansible-playbook -i inventory
ansible.containerized_installer.dashboard_install --ask-become-pass
```

Result

For reference, see the following example output:

```
PLAY RECAP
*****
*****

ec2-54-147-26-173.compute-1.amazonaws.com : ok=126  changed=51  unreachable=0
failed=0  skipped=42  rescued=0  ignored=0

localhost : ok=12  changed=0  unreachable=0  failed=0
skipped=9  rescued=0  ignored=0
```

Alternative configurations are possible (for example, the database for automation dashboard can be set on a different host). This requires additional changes to variables in the inventory file. Consult the [Inventory variables](#) section of this document for available variables.

Configure automation dashboard

Integrate your Ansible Automation Platform instances into the automation dashboard configuration to collect and visualise data and gain insights into your automation.

Before you begin

- You have installed automation dashboard.
- You have verified that automation dashboard is running on HTTPS port 8447 on your Red Hat Enterprise Linux host.

NOTE:

- This verification requires your Ansible Automation Platform login details.
- Port 8447 is enabled by default, but this is configurable.

Procedure

1. Configure a personal access token. For more information, see [Configuring access to external applications with token-based authentication](#).
2. Create or update the `clusters.yaml` file. You must include the `access_token`, `refresh_token`, `client_id`, and `client_secret` to enable persistent authentication and automatic token rotation.

```
clusters:
  - protocol: https
    address: <aap_controller_url>
    port: 443
    access_token: <access_token_string>
    refresh_token: <refresh_token_string>
    client_id: <client_id_string>
    client_secret: <client_secret_string>
    verify_ssl: false
    name: <unique_cluster_name>
```

Configuring TLS verification

When configuring the `verify_ssl` parameter, choose the setting that matches your Ansible Automation Platform certificate type:

- **Commercial certificates:** Set `verify_ssl: true`.

- **Self-signed certificates:** Set `verify_ssl: false`.

NOTE:

The automation dashboard cannot verify self-signed certificates against a custom Certificate Authority (CA).

3. You can add one or more Ansible Automation Platform instances (of the same Ansible Automation Platform version) into the automation dashboard configuration for pulling and combining data by using the following:

NOTE:

If you only have one Ansible Automation Platform instance, then remove the second entry.

```

---
clusters:
  - protocol: https          <--- Normally https
    address: my-aap.example.com <--- Can use IP or FQDN without http(s)://
    port: 443                <--- Normally 443
    access_token: sampleToken <--- Your preconfigured Ansible Automation
Platform read access token
    Platform read access token
    refresh_token: myRefreshToken
    client_id: myClientID
    client_secret: myClientSecret
    verify_ssl: false        <--- Can be used when using self signed certs
    sync_schedules:
      - name: Every 5 minutes sync
        rrule: DTSTART;TZID=Europe/Ljubljana:20250630T070000
FREQ=MINUTELY;INTERVAL=5
        enabled: true

    # If there is one Ansible Automation Platform instance, then remove the
second entry below.

    # Alternately. If there is more than one AAP instance to connect, copy
additional entries.
  - protocol: https
    address: aap2.example.com
    port: 443
    access_token: WRn2swiqg5spEwUndDkrJoCeg4QwuW
    verify_ssl: true
    sync_schedules:
      - name: Every 5 minutes sync
        rrule: DTSTART;TZID=Europe/Ljubljana:20250630T070000
FREQ=MINUTELY;INTERVAL=5
        enabled: true

```

NOTE:

The `access_token`, `refresh_token`, and `client_secret` are stored in the automation dashboard database. These values are encrypted for security.

4. Load and activate your automation dashboard configuration. You must copy the configuration file to the container's `/tmp/` directory. The system automatically removes the file from the container after successful processing.

```
podman cp clusters.yaml automation-dashboard-web:/tmp

podman exec -it automation-dashboard-web /venv/bin/python manage.py
setclusters /tmp/clusters.yaml
```

NOTE: If the system cannot remove the file from the /tmp/ directory, it displays an error message and continues running.

NOTE:

Token Rotation: The automation dashboard rotates tokens internally for security but does not update your local `clusters.yaml` file. If you must re-run the `setclusters` command later, your defined tokens might be expired.

To retrieve the current valid tokens, run the following command and save the output to a new file:

```
podman exec -it automation-dashboard-web /venv/bin/python manage.py
getclusters --decrypt > clusters_current.yaml
```

Synchronize data to automation dashboard

Synchronize data from your connected Ansible Automation Platform clusters to the automation dashboard to view the latest automation metrics.

Before you begin

- You have installed automation dashboard.
- You have configured the `clusters.yaml` file with your platform details.

Procedure

1. Identify the running automation dashboard container:

```
$ podman ps | grep automation-dashboard-web
```

2. Run the synchronization command using one of the following methods:

- **Interactive mode:** Run `syncdata` without arguments. You must confirm the synchronization request when prompted.

```
$ podman exec -it automation-dashboard-web /venv/bin/python manage.py syncdata
```

- **Noninteractive mode:** Use the `--since` and `--until` flags to define the date range. This method bypasses the user prompt and is required for automated scripts or `cron` jobs.

```
$ podman exec -it automation-dashboard-web /venv/bin/python manage.py syncdata --since=2024-01-01 --until=2024-06-30
```

Result

1. Verify that the terminal displays the success message: `Successfully created Sync task for Cluster <cluster_url>.`
2. Refresh automation dashboard in your browser to view the updated metrics.

Verify cluster access tokens

After you configure and load your cluster data, verify the stored access tokens for debugging purposes.

Procedure

1. Use the `getclusters` management command with the `--decrypt` option to display the stored `access_token` and `refresh_token` in plain text.
2. Run the following command inside the `automation-dashboard-web` container:

```
podman exec -it automation-dashboard-web /venv/bin/python ./manage.py getclusters --decrypt
```

3. Review the output to confirm that the stored tokens are correct and up-to-date.

```
clusters:
  - protocol: https
    address: my-aap.example.com
    port: 443
    access_token: sampleToken
    refresh_token: myRefreshToken
    client_id: myClientID
    client_secret: myClientSecret
    verify_ssl: false
    sync_schedules:
      - name: Every 5 minutes sync
        rrule: DTSTART;TZID=Europe/Ljubljana:20250630T070000
        FREQ=MINUTELY;INTERVAL=5
        enabled: true
```

NOTE:

Displaying the encrypted `access_token` and `refresh_token` in plain text for debugging requires the `--decrypt` flag. Do not use this command on unsecured systems.

You can write output produced by `./manage.py getclusters --decrypt` to a file `clusters.yaml` and use it as input for `./manage.py setclusters clusters.yaml`.

Result

If you come across error messages during installation, consult the following table:

Issue	Possible Cause	Solution
401 error	This is an unauthorized access message indicating authentication errors such as wrong credentials or tokens.	Verify that your access token is correct in <code>clusters.yaml</code>
401 error	A temporary 401 error is expected behavior when the token expires, followed immediately by trying to refresh.	If the automatic token refresh fails (for example, due to invalid <code>client_secret</code> or <code>refresh_token</code>), use the <code>getclusters --decrypt</code> command to manually verify that the credentials stored in the database match those in your source <code>clusters.yaml</code> file. If they do not match, re-run the

Issue	Possible Cause	Solution
		<code>setclusters</code> command with the correct configuration. You can only use the refresh token once. If you need to execute <code>setclusters</code> because of invalid access token, create new access and refresh tokens, and use them in your source <code>clusters.yaml</code> .
404 error	This is a “not found” message indicating that something is not configured correctly or pointing to the correct endpoint.	Verify that your Ansible Automation Platform instance URLs used in <code>clusters.yaml</code> are correct.

A successful installation should be running the following three container services:

```
podman ps --all --format "{{.Names}}"
```

```
postgresql
```

```
automation-dashboard-task
```

```
automation-dashboard-web
```

You can check your container logs by running the following:

```
journalctl CONTAINER_NAME=container (where container equals one of postgresql
automation-dashboard-task or automation-dashboard-web)
```

For example:

```
journalctl CONTAINER_NAME=automation-dashboard-task
```

```
May 22 13:02:07 automation-dashboard automation-dashboard-task[1607]: [wait-for-
migrations-dashboard.sh] Waiting for database migrations...
```

```
May 22 13:02:07 automation-dashboard automation-dashboard-task[1607]: [wait-for-
migrations-dashboard.sh] Attempt 1
```

```
May 22 13:02:10 automation-dashboard automation-dashboard-task[1607]: INFO
2025-05-22 13:02:10,636 periodic 2 140568371550016 Starting sync task.
```

```
May 22 13:02:10 automation-dashboard automation-dashboard-task[1607]: INFO
2025-05-22 13:02:10,636 periodic 2 140568371550016 Retrieving clusters inform>
```

```
May 22 13:02:10 automation-dashboard automation-dashboard-task[1607]: INFO
2025-05-22 13:02:10,747 periodic 2 140568371550016 Retrieved 1 clusters.
```

```
May 22 13:02:10 automation-dashboard automation-dashboard-task[1607]: INFO
2025-05-22 13:02:10,761 periodic 2 140568371550016 Retrieving data from clust>
```

```
May 22 13:02:10 automation-dashboard automation-dashboard-task[1607]: INFO
2025-05-22 13:02:10,761 connector 2 140568371550016 Checking Ansible Automation
Platform version at h>
```

```
May 22 13:02:10 automation-dashboard automation-dashboard-task[1607]: INFO
2025-05-22 13:02:10,761 connector 2 140568371550016 Checking if is Ansible
Automation Platform 2.5 at>
```

```
May 22 13:02:10 automation-dashboard automation-dashboard-task[1607]: INFO
2025-05-22 13:02:10,761 connector 2 140568371550016 Pinging api https://ec2-3>
```

```
May 22 13:02:10 automation-dashboard automation-dashboard-task[1607]: INFO
2025-05-22 13:02:10,761 connector 2 140568371550016 Executing GET request to >
```

```
May 22 13:02:13 automation-dashboard automation-dashboard-task[1607]: ERROR
2025-05-22 13:02:13,820 connector 2 140568371550016 GET request failed with >
```

```
May 22 13:02:13 automation-dashboard automation-dashboard-task[1607]: INFO
2025-05-22 13:02:13,821 connector 2 140568371550016 Checking if is Ansible
Automation Platform 2.4 at>
```

```
May 22 13:02:13 automation-dashboard automation-dashboard-task[1607]: INFO
2025-05-22 13:02:13,821 connector 2 140568371550016 Pinging api https://ec2-3>
```

```
May 22 13:02:13 automation-dashboard automation-dashboard-task[1607]: INFO
2025-05-22 13:02:13,821 connector 2 140568371550016 Executing GET request to >
```

```
May 22 13:02:16 automation-dashboard automation-dashboard-task[1607]: ERROR
2025-05-22 13:02:16,892 connector 2 140568371550016 GET request failed with ...
```

The following log snippet shows a successful token refresh:

NOTE:

This log snippet omits timestamps and hostname for brevity.

Example

```
journalctl CONTAINER_NAME=automation-dashboard-task
INFO Checking if is AAP 2.5 ... 2.6 at https://app.example.com:443
INFO Pinging api https://app.example.com:443/api/gateway/v1/ping/
INFO Detected AAP version AAP 2.6 at https://app.example.com:443
INFO Executing GET request to https://app.example.com:443/api/controller/v2/
organizations/?page_size=100&page=1
ERROR GET request failed with status 401
INFO Token refresh POST request succeeded with status 201
ERROR GET after reauth response.status_code=200
INFO Executing GET request to https://app.example.com:443/api/controller/v2/
job_templates/?page_size=200&page=1
Executing GET request to https://app.example.com:443/api/controller/v2/jobs/?
page_size=100&page=1&order_by=finished&finished__gt=2025-10-23T13:01:09.768681Z
```

Check how the services are running by using `systemd` :

```
systemctl status --user

● automation-dashboard
   State: running
   Units: 76 loaded (incl. loaded aliases)
   Jobs: 0 queued
   Failed: 0 units
   Since: Thu 2025-05-22 13:02:07 UTC; 22min ago
systemd: 252-51.el9
   CGroup: /user.slice/user-1000.slice/user@1000.service
           └─app.slice
               │ └─automation-dashboard-task.service
                   │ │ └─1607 /usr/bin/conmon --api-version 1 -c
                   │ │ 84e46532e8ca31b0cadb037479289d030103aa01b7a1591e62b83b17f031e47d -u
                   │ │ 84e46532e8ca31b0cadb037479>
                   │ │ └─automation-dashboard-web.service
                   │ │ │ └─1608 /usr/bin/conmon --api-version 1 -c
                   │ │ │ d060f3e3fb2b4c4c5c588149253beed83c78ccc9c9a8c1bf4c96157142a210dc -u
                   │ │ │ d060f3e3fb2b4c4c5c58814925>
                   │ │ └─dbus-broker.service
                   │ │ │ └─1621 /usr/bin/dbus-broker-launch --scope user
                   │ │ │ │ └─1624 dbus-broker --log 4 --controller 9 --machine-id
                   │ │ │ │ 612db98503014199bfd8c788c8d3da58 --max-bytes 1000000000000000 --max-fds 2500000000>
                   │ │ └─postgresql.service
```

```

    |   └─1614 /usr/bin/conmon --api-version 1 -c
eec61745cb6fc3a89a4f7475d7ef63b5899699157d943c2f16a3243311927bef -u
eec61745cb6fc3a89a4f7475d7>

└─init.scope
| └─1093 /usr/lib/systemd/systemd --user
| └─1128 "(sd-pam)"
└─user.slice

└─libpod-84e46532e8ca31b0cadb037479289d030103aa01b7a1591e62b83b17f031e47d.scope
| └─container
| | └─1619 /usr/bin/dumb-init -- /usr/bin/launch_dashboard_task.sh
| | └─1681 /venv/bin/python periodic.py
└─libpod-
d060f3e3fb2b4c4c5c588149253beed83c78ccc9c9a8c1bf4c96157142a210dc.scope
| └─container
| | └─1617 /usr/bin/dumb-init -- /usr/bin/launch_dashboard_web.sh
| | └─1646 /usr/bin/python3.9 /usr/local/bin/supervisord -c /etc/
supervisord_dashboard_web.conf
| | └─1877 /bin/bash /usr/local/bin/stop-supervisor
| | └─1878 "nginx: master process nginx -g daemon off;"
| | └─1879 /venv/bin/uwsgi /etc/tower/uwsgi.ini
| | └─1880 "nginx: worker process"
| | └─1881 "nginx: worker process"
| | └─1882 "nginx: worker process"
| | └─1883 "nginx: worker process"
| | └─1884 /venv/bin/uwsgi /etc/tower/uwsgi.ini
| | └─1885 /venv/bin/uwsgi /etc/tower/uwsgi.ini
| | └─1886 /venv/bin/uwsgi /etc/tower/uwsgi.ini
| | └─1887 /venv/bin/uwsgi /etc/tower/uwsgi.ini
| | └─1888 /venv/bin/uwsgi /etc/tower/uwsgi.ini
└─libpod-
eec61745cb6fc3a89a4f7475d7ef63b5899699157d943c2f16a3243311927bef.scope
| └─container
| | └─1623 postgres
| | └─1869 "postgres: logger "
| | └─1871 "postgres: checkpointer "
| | └─1872 "postgres: background writer "
| | └─1873 "postgres: walwriter "
| | └─1874 "postgres: autovacuum launcher "
```

```

|   |─1875 "postgres: stats collector "
|   |─1876 "postgres: logical replication launcher "
|   |─1889 "postgres: aapdashboard aapdashboard 172.31.28.99(39338)
idle"
    |─podman-pause-b6c4e853.scope
        |─1359 catatonit -P

```

Uninstall automation dashboard

Uninstall automation dashboard and its dependencies by using a single command, ensuring a clean removal from your host.

- Run the following command to uninstall automation dashboard and its dependencies, including the PostgreSQL database container:

```

ansible-playbook -i inventory
ansible.containerized_installer.dashboard_uninstall

```

NOTE: Uninstalling automation dashboard permanently deletes all collected metrics data, saved reports, and the PostgreSQL database.

Install Ansible Builder to create or edit execution environments

Install Ansible Builder to create custom execution environments that contain the dependencies and collections required for your automation content. You need a valid Red Hat subscription and Podman installed on your RHEL system.

Before you begin

- You have installed the Podman container runtime.
- You have valid subscriptions attached on the host. With a valid subscription you can access the subscription-only resources needed to install `ansible-builder`, and ensures that the necessary repository for `ansible-builder` is automatically enabled. See [Attaching your Red Hat Ansible Automation Platform subscription](#) for more information.

NOTE:

To install the developer tools without consuming a managed node subscription, you can use MCT4589-Red Hat Ansible Developer, Standard (10 Managed Nodes), which is available at no cost. This subscription requires the approval of the Ansible Business Unit.

- Run the following command to install Ansible Builder and activate your Ansible Automation Platform repo:

```
# dnf install --enablerepo=ansible-automation-platform-aap-version-for-rhel-rhel-version-x86_64-rpms  
ansible-builder
```

Replace *aap-version* with your Ansible Automation Platform version (for example, `2.6`) and *rhel-version* with your Red Hat Enterprise Linux major version (for example, `9`).

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