

10.0

IBM MQ Agent

IBM

Note

Before using this information and the product it supports, read the information in [“Notices” on page 51.](#)

This edition applies to version 9 release 4 of IBM® MQ and to all subsequent releases and modifications until otherwise indicated in new editions.

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

© **Copyright International Business Machines Corporation 2025, 2026.**

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

IBM MQ Agent.....	5
What's new and changed, and known issues for the IBM MQ Agent.....	8
Support and entitlement for the IBM MQ Agent.....	9
Installing and configuring the IBM MQ Agent.....	11
Preparing an AI runtime for the IBM MQ Agent.....	11
Granting the IBM MQ Agent access to queue manager resources.....	14
Deploying the IBM MQ Agent into your cluster.....	15
Configuring the IBM MQ Console to connect to the IBM MQ Agent.....	30
Upgrading the IBM MQ Agent.....	33
Uninstalling the IBM MQ Agent.....	36
Interacting with the IBM MQ Agent.....	39
Troubleshooting the IBM MQ Agent.....	43
Collecting IBM MQ Agent troubleshooting information with IBM MQ container inspector.....	45
Foundation models for the IBM MQ Agent.....	47
Security for the IBM MQ Agent.....	48
 Notices.....	 51
Programming interface information.....	52
Trademarks.....	52

IBM MQ Agent

The IBM MQ Agent is a separately downloadable IBM MQ component that provides an AI agent to help you explore your configuration, check the status of objects, and diagnose problems with your IBM MQ system.

Overview

The IBM MQ Agent is designed to help you understand how your IBM MQ queue managers are configured, lead real-time diagnostic investigation, and provide general information about IBM MQ concepts and terms. For example, in a system that is running well, messages flow freely from producers to consumers and should not spend a long time queued in the system. However, some issues can cause messages to build up on a queue. If queues fill up, applications can no longer put messages to the affected queues. Where messages are not flowing correctly, applications might not be able to get messages for processing. The IBM MQ Agent can help you identify and understand the cause of the message build-up (and other problems) so that you can get your messages moving again.

Note: The IBM MQ Agent can query and reason about your system based on queue manager state. The IBM MQ Agent cannot process or run commands such as IBM MQ script (MQSC) commands.

The IBM MQ Agent is a container-based offering that includes the following items:

- Configuration for the agent to connect to (and interact with) up to twenty queue managers, which can be running anywhere in your estate.
- A Model Context Protocol (MCP) server that hosts a suite of tools for gathering information from your IBM MQ queue managers.
- An agent runtime that you interact with, which uses AI models, and tools that are hosted by the MCP server, to answer your questions.

You manage your entitlement to use the IBM MQ Agent as an extension of the entitlement that you purchase for IBM MQ Advanced or IBM Cloud Pak® for Integration. For more information, see [“Support and entitlement for the IBM MQ Agent” on page 9](#).

After you install the IBM MQ Agent and complete the configuration, you can use the chat assistant to ask the IBM MQ Agent questions about your IBM MQ installation and environment. To answer your questions, the IBM MQ Agent has capabilities that are designed around core queue manager concepts. For example:

- Queue manager: configuration and status
- Channels: client, server, sender, receiver, status, and related diagnostics
- Queues: local, remote, transmission, model queues, status, and related diagnostics
- Listeners: listener status and related diagnostics
- Clustering: clustering objects, status, and related diagnostics
- Applications: application connectivity status and related diagnostics
- Message build-up: analysis of queue managers to identify message build-up and recommend remedial actions
- Documentation: context enhanced search and next-step planning by using IBM MQ documentation

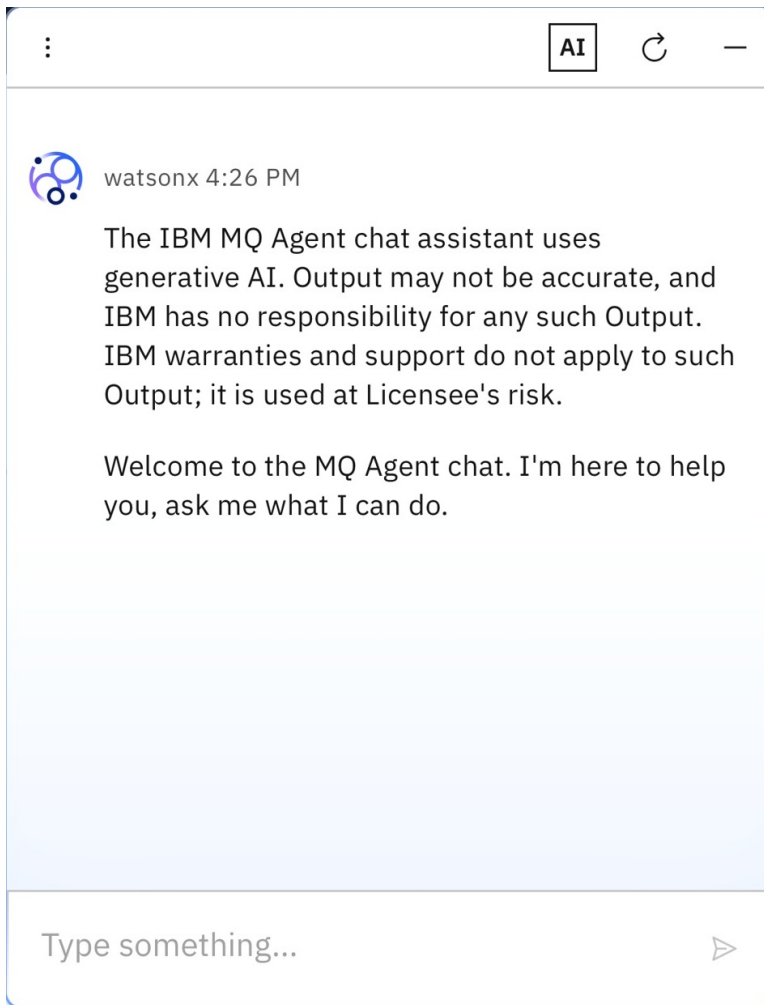


Figure 1. Chat assistant welcome page

Use cases

The agent uses conversation context and conversation history to select which of its capabilities can help you analyze your IBM MQ configuration and diagnose problems. Here are some examples of the type of question and problem determination that the IBM MQ Agent can help you with:

Questions about IBM MQ objects and status

- Report cluster memberships of a queue manager
- Report IBM MQ versions and platforms in your queue manager estate
- Show the queues that are available on a queue manager or set of queue managers
- Identify queues that have a current depth of more than 100 messages (for example)
- Identify queues on queue managers where messages haven't been processed for more than 30 minutes (for example)
- Identify queues with no connected consuming applications
- Report queues with a specified attribute, for example list shared queues on a queue manager
- Show channels that are in a retry state
- Show all running listeners on a queue manager or set of queue managers
- Identify where an application is connected

Problem determination

- Diagnose why a channel is in a stopped state

- Understand why messages were delivered to a dead letter queue
- Search error logs to help diagnose problems that are related to channels
- Understand which ports listeners are running on
- Identify and diagnose general message build-up problems
- Diagnose slow message flow over a network
- Understand why messages are building up

Intelligent insights and next step planning

- Explain an IBM MQ concept
- Reason about configuration issues
- Predict future outcomes based on real-time status
- Help you plan your next steps to resolve a problem

Examples

Here are some examples of general questions and instructions that you might ask the IBM MQ Agent:

- What queue managers can you see?
- What version are my queue managers?
- Do I have message build up?
- What is the problem with *<queue name>* on *<queue manager name>*?
- Diagnose *<queue name>* on *<queue manager name>*
- Why are messages building up on *<queue name>* on *<queue manager name>*?
- Is there a problem with the dead letter queue on *<queue manager name>*?

Here is an example of a conversation that you might have with the IBM MQ Agent:

User: What queue managers are available?

Agent: The available queue managers are:

- QM1
- QM2
- QM3

User: What MQ versions are those queue managers on?

Agent: Here's the MQ version information for the queue managers you asked about:

- QM1: 9.4.5.0
- QM2: 9.4.5.0
- QM3: 9.4.3.0

User: What's the deepest queue on QM1?

Agent: The deepest queue on the QM1 queue manager is queue4, with a current depth of 5,000 messages. Other queues have lower depths, with the next highest being important at 4,950 messages. Some queues (like model, alias, or remote queues) don't have depth values.

User: Why is this queue built up?

Agent: The queue queue4 on queue manager QM1 is built up because there are no applications consuming or producing messages from it. Here are the key details:

- The queue is enabled for getting messages, but there are no active consumers (IPPROCS = 0).
- There are also no active producers (OPPROCS = 0), so no new messages are being added.
- The queue is currently at 100% capacity, with 5000 messages (the maximum allowed).

- No messages have been consumed from the queue, and the last time a message was added was 29 days ago.

In short, the queue is full because messages are not being processed or removed, and no new messages are being added either.

User: How should I fix this problem?

Agent: To fix the problem with queue qm4 on queue manager QM1, you'll need to address the lack of message consumers. Here's what I found:

The queue is completely full (5000/5000 messages) because:

- There are no applications consuming messages from it (IPPROCS = 0)
- Messages have been accumulating for 29 days without being processed
- While the queue is enabled for getting messages, nothing is actually retrieving them

To resolve this:

1. You'll need to either:
 - Start an application that should be consuming messages from this queue, or
 - Clear out the existing messages if they're no longer needed
2. For immediate relief, you could:
 - Increase the queue's maximum depth temporarily
 - Manually clear some messages to prevent further buildup

The root cause appears to be that whatever application should be processing these messages isn't running or isn't properly connected to this queue.

User: How do I increase the queue depth?

Agent: To increase the queue depth for queue queue4 on queue manager QM1, you'll need to modify the MAXDEPTH parameter. Here's how to do it:

1. First, determine the new value for MAXDEPTH based on your expected message volume and available storage.
2. Use the ALTER QUEUE command to update the queue depth. For your specific queue, the command would be:

```
ALTER QLOCAL ('queue4')
MAXDEPTH(new_value)
```

Replace new_value with your desired maximum depth.

What's new and changed, and known issues for the IBM MQ Agent

Note: For a list of fixes for each release, see [Fix list for the IBM MQ Agent Continuous Delivery releases](#) on the IBM Support website.

IBM MQ Agent 2.0.0

What's new

- Support for Amazon Elastic Kubernetes Service (Amazon EKS). You can now deploy and run the IBM MQ Agent in the same Kubernetes environment where IBM MQ runs, without using Red Hat OpenShift. For information about supported versions of Amazon EKS, see the system requirements pages, which you can access by using the information in [Where to find product requirements and support information](#).
- Support for the vLLM open source AI inference runtime. You can now use this runtime as well as the default, IBM watsonx.ai™ Runtime. With vLLM support, you can use the IBM MQ Agent in an on-premises environment instead of a cloud environment. To use vLLM, set the relevant properties

in the overrides file for the Helm chart before you deploy or upgrade the IBM MQ Agent. For more information, see [“IBM MQ Agent Helm chart values”](#) on page 22.

- Increased flexibility of AI foundation models. Previously, the IBM MQ Agent required access to three models (listed in [“Foundation models for the IBM MQ Agent”](#) on page 47). These models are now labelled Primary, Secondary, and Embedding. The Primary model is still required. The Secondary model (a more lightweight model that is used for simple tasks) is now optional. Use of this model can reduce cost and latency when you use IBM watsonx.ai Runtime. However, you might want to disable it when you use an on-premises runtime, so that you don't need another model. The Embedding model (which is used to query IBM MQ documentation) is now provided as part of the IBM MQ Agent.

IBM MQ Agent 1.0.1

What's new

This release contains improvements rather than new features.

Known issues

There are no known issues in this release.

IBM MQ Agent 1.0.0

What's new

This is the initial release of the IBM MQ Agent.

Known issues

If you upgrade the IBM MQ Agent by using the `helm upgrade` command, you see the following error in the pod logs:

```
qm1 pod log error - DATA - (@static) responseCode 401
agent pod - "GET /create_new_thread HTTP/1.1" 401 Unauthorized
```

The `agentAPIKey` that is used to connect the IBM MQ Console to the IBM MQ Agent is reissued; you must manually update the `agentAPIKey` property of your IBM MQ Console's `mqwebuser.xml` file with the new value. If you have an IBM MQ container deployment, delete any pods that contain the IBM MQ Console configuration. These pods continue to use the old API key so will fail to connect. For more information, see [“Configuring the IBM MQ Console to connect to the IBM MQ Agent”](#) on page 30.

Support and entitlement for the IBM MQ Agent

The IBM MQ Agent requires an IBM MQ Advanced or IBM Cloud Pak for Integration license and runs on Red Hat OpenShift Container Platform or Amazon Elastic Kubernetes Service (Amazon EKS), with IBM MQ Console 9.4.5 or later as a prerequisite. The IBM MQ Agent follows the continuous delivery support model with one-year support duration, uses container licensing, and requires separately charged services for AI foundation model access.

Supported platforms

You deploy the IBM MQ Agent into a supported version of Red Hat OpenShift Container Platform or Amazon EKS, as listed in the **Containers** tab of the system requirements pages. For information about accessing these pages, see [Where to find product requirements and support information](#). Red Hat also offers some free trials; for details, see the [Try Red Hat OpenShift](#) page.

Your cluster should have 4 CPU cores and 5 GB of memory available. The agent can be configured to run with more or fewer resources but such scenarios have not been tested.

Notes:

- IBM provides defect and usage support for the IBM-supplied container enablement code but does not help users configure and debug problems with their container environment or deployment pipeline.

- Red Hat provides support for the Red Hat packages only for containers running on Red Hat Enterprise Linux® or Red Hat Enterprise Linux CoreOS hosts. See [Red Hat Container Support Policy](#) for more information.
- For information about the support provided by AWS for Amazon EKS, see [Understand the Kubernetes version lifecycle on EKS](#) in the Amazon EKS documentation.

Supported CPU architectures

The IBM MQ Agent is supported on the amd64 architecture only.

Required software

To use the IBM MQ Agent, you must have a configured and running instance of the IBM MQ Console, at version 9.4.5 or later. The IBM MQ Console can be either a built-in installation (installed as a component of IBM MQ) or a standalone IBM MQ Web Server installation.

For more information about the IBM MQ Console, and links to instructions for installing and configuring it, see [The IBM MQ Console and REST API](#).

Licensing

- The IBM MQ Agent is eligible for container licensing. Container licensing allows you to license only the available capacity of your individual IBM MQ Agent containers, rather than requiring you to license the entire server where your containers are running. To take advantage of container licensing, you use the IBM License Service to track license usage and determine your required entitlement. For more information, see the [Container Licenses](#) page of the Passport Advantage® website.
- The IBM MQ Agent requires either an IBM MQ Advanced license (VPC or PVU container licensing metric) or an IBM Cloud Pak for Integration license (IBM MQ Advanced ratio).
- If you have an IBM Cloud Pak for Integration license, you may deploy the IBM MQ Agent on the Cloud Pak for Integration licensed Red Hat OpenShift Container Platform, provided that you also deployed the [Cloud Pak for Integration Agent](#) and accepted the associated license terms.
- You can use the IBM MQ Agent to query queue manager resources in a licensed installation of any supported version of IBM MQ. For more information about IBM MQ licenses, see [IBM MQ license information](#).

Entitlement

You manage your entitlement to use the IBM MQ Agent as an extension of the entitlement that you purchase for IBM MQ Advanced or IBM Cloud Pak for Integration. The IBM MQ Agent is not entitled with an IBM MQ Advanced for Non-Production Environment license.

The IBM MQ Agent requires IBM MQ Advanced entitlement or IBM Cloud Pak for Integration entitlement at the IBM MQ Advanced ratio in all deployment scenarios, regardless of whether you use the IBM MQ Agent with Production, Non-Production, Development, or Trial queue managers. Deployment under the IBM Cloud Pak for Integration IBM MQ Non-Production entitlement ratio is not permitted.

Each IBM MQ Agent instance consumes the equivalent of 4 VPCs of IBM MQ Advanced entitlement (or 4 cores of IBM MQ Advanced entitlement, depending on the licensing metric). In IBM Cloud Pak for Integration environments, this consumption corresponds to 2 VPCs of IBM Cloud Pak for Integration entitlement, based on the IBM MQ Advanced 2:1 entitlement ratio.

The IBM MQ Agent requires AI models, the use of which is a separately chargeable service. The supported AI runtimes are as follows:

- IBM watsonx.ai runtime. You add this runtime to an IBM Cloud® account. If you already have IBM watsonx.ai with foundation models deployed, you can share them for use with the IBM MQ Agent. Alternatively, you can deploy the models on-demand or configure them to be consumed by using the IBM watsonx.ai pricing (the Essentials plan is suggested). For more information about pricing, see the [IBM watsonx.ai pricing web page](#).

- vLLM. vLLM is an open source runtime that is popular for on-premises deployments. If you already have vLLM with foundation models deployed, you can share them for use with the IBM MQ Agent.

For a list of required foundation models, see [“Foundation models for the IBM MQ Agent”](#) on page 47.

Support duration and fix availability

IBM provides defect and usage support for the IBM MQ Agent on a continuous delivery basis. The IBM MQ Agent is maintained at the latest modification level only. Periodic and interim fixes are available from IBM Cloud Container Registry.

Installing and configuring the IBM MQ Agent

The IBM MQ Agent can help you find and fix problems with messages building up on message queues.

Before you begin

Check that your system meets the requirements described in [“Support and entitlement for the IBM MQ Agent”](#) on page 9.

Procedure

1. Set up access to large language models (LLMs) by following the instructions in [“Preparing an AI runtime for the IBM MQ Agent”](#) on page 11.

The IBM MQ Agent requires access to the LLMs in an AI runtime (IBM Cloud’s IBM watsonx.ai Runtime or vLLM) to understand your questions. To get access to IBM watsonx.ai Runtime, you create an IBM Cloud account, add an IBM watsonx.ai runtime, then create a project in the runtime.

2. Grant the necessary access to your queue managers by following the instructions in [“Granting the IBM MQ Agent access to queue manager resources”](#) on page 14.
3. Deploy the IBM MQ Agent by following the instructions in [“Deploying the IBM MQ Agent into your cluster”](#) on page 15.

This task includes creating objects within your cluster that contain information about your IBM MQ queue managers, then downloading and installing the IBM MQ Agent Helm chart to deploy the IBM MQ Agent into the cluster. This process creates the resources that are required for the IBM MQ Agent to connect to your IBM MQ installation and the AI runtime.

4. Configure your IBM MQ Console to connect to the IBM MQ Agent, by following the instructions in [“Configuring the IBM MQ Console to connect to the IBM MQ Agent”](#) on page 30.

Results

You can use the IBM MQ Console to talk to the IBM MQ Agent. The agent uses the LLMs to understand your questions, and the information you gave it during deployment to query your IBM MQ installation to provide answers.

Preparing an AI runtime for the IBM MQ Agent

To understand your questions, the IBM MQ Agent uses large language models (LLMs) that are hosted in an AI runtime. If you are using IBM watsonx.ai Runtime, create a project and an API key, which the IBM MQ Agent can use to access the LLMs. If you are using vLLM, you need only note any existing API key.

Before you begin

Currently, the supported AI runtimes are IBM watsonx.ai Runtime (which you add to an IBM Cloud account) and vLLM. For more information about entitlement for IBM watsonx.ai Runtime, see [“Entitlement”](#) on page 10.

If you are using vLLM, you do not need to complete this task; you need only record your [API key](#), if your vLLM runtime requires one for access. You specify this key when you later deploy the IBM MQ Agent, so

that the IBM MQ Agent can access the vLLM runtime. After recording the key, proceed to [“Granting the IBM MQ Agent access to queue manager resources”](#) on page 14.

If you are using IBM watsonx.ai Runtime, complete the rest of this task to create a project with an API key.

1. If you don't have an IBM Cloud account already, create one by following the instructions in [Setting up your IBM Cloud account](#) in the IBM Cloud documentation.
2. Install the IBM Cloud command line interface tool then use it to log in to IBM Cloud. For more information, see [Installing the stand-alone IBM Cloud CLI](#) in the IBM Cloud documentation.

About this task

For more information about the set of LLMs that is used by the IBM MQ Agent, see [“Foundation models for the IBM MQ Agent”](#) on page 47.

Procedure

1. Create IBM Cloud Object Storage.

Follow the instructions in [Create an instance of IBM Cloud Object Storage](#) in the IBM Cloud documentation (you don't need to create any buckets).

2. Create an IBM watsonx.ai runtime.

You can use either the [IBM Cloud catalog](#) or the CLI as described in [IBM Cloud CLI](#) in the IBM watsonx™ documentation.

For example, use the IBM Cloud CLI tool to run the following command:

```
ibmcloud resource service-instance-create <service_instance_name> pm-20 <plan_name> <region>
--service-endpoints <endpoint_network>
```

where:

- <service_instance_name>: a custom name for your [service instance](#).
- pm-20: the service ID for the IBM watsonx.ai Runtime.
- <plan_name>: an IBM watsonx.ai Runtime pricing plan, such as essentials or standard. For more information, see [watsonx.ai Runtime service plans](#) in the IBM watsonx documentation.
- <region>: the IBM watsonx regional data center, such as us-south (Dallas).

Note: Not all data centers support the foundation models that are required by the agent. For a list of required foundation models, see [“Foundation models for the IBM MQ Agent”](#) on page 47. For a list of data centers and the models that are available in each, see [Regional availability of services and features](#) in the IBM watsonx documentation.

- <endpoint_network>: defines network traffic. Options are public, private, or public-and-private. For more information, see [Managing the watsonx.ai Runtime service endpoint](#) in the IBM watsonx documentation.

3. Navigate to the Manage page for the IBM watsonx Runtime instance that you created.
4. Click the **Options** (down arrow) icon next to **Launch in** then click **IBM watsonx** to open an IBM watsonx browser tab.

If the "Welcome to watsonx" dialog opens, close it.

5. Create a new project.

- a) In the IBM watsonx browser tab, scroll to the Projects section at the end of the page then click the **Create a new project** (+) icon.

- b) Associate cloud object storage with the project.

Use the **Select storage service** drop-down list to associate the cloud object storage that you created earlier with this new project.

- c) Click **Create**.

You might need to create an access key before the **Create** button is enabled.

- d) Note the project ID for future use.
- In the **Project** navigation, click **General** to open the "General" page, which shows the **Project ID**.
6. Add collaborators to the project.
- a) In the project, click the **Manage** tab, then in the **Project** navigation, click **Access Control** to open the "Access control" page.
 - b) In the "Access control" page, click **Add collaborators**.
7. Associate the IBM watsonx.ai Runtime with the project.
- a) In the project, click the **Manage** tab, then in the **Project** navigation, click **Services & Integrations** to open the "Services & integrations" page.
 - b) In the "Services & integrations" page, click **Associate service**.
- Without this association, you cannot run or deploy assets that you create within a IBM watsonx.ai project.
8. Create an IBM Cloud API key then retain it for future use.
- For example, use the IBM Cloud CLI tool to run the following command, which creates a key called MyAPIKey in a file called my-api-key.json in the directory in which you run the command:

```
ibmcloud iam api-key-create MyAPIKey -d "API key for use with IBM MQ Agent" --file my-api-key.json
```

9. Optional: As an account administrator, you might want to grant access to other users.
- For example, use the IBM Cloud CLI tool to run one or more of the following commands. For more information about each command, see the following topics in the IBM Cloud CLI documentation: [Managing IAM access, API keys, trusted profiles, service IDs, and access groups \(ibmcloud iam\)](#) and [Managing accounts and users \(ibmcloud account\)](#)

Create a group:

```
ibmcloud iam access-group-create [<group_name>]
```

Check if a group exists:

```
ibmcloud iam access-groups
```

Add a policy to the group:

```
ibmcloud iam access-group-policy-create [<group_name>] --roles [<role>] --service-name [<service_name>] --service-instance [<service_instance>] --region [<region>]
```

Invite users:

```
ibmcloud account user-invite [<email>]
```

Add a user to the group:

```
ibmcloud iam access-group-user-add [<group_name>] [<email>]
```

Remove a user from a group:

```
ibmcloud iam access-group-user-remove [<group_name>] [<email>] -f
```

Remove a user from an account:

```
ibmcloud account user-remove [<email>] -f
```

Results

You created a project that is associated with an IBM watsonx.ai Runtime service. The IBM MQ Agent will use the LLMs in the IBM watsonx.ai service to understand your questions.

You also recorded the following information, which you will use in the next task:

- The ID of the project that you created.
- The IBM Cloud API key that you created.

What to do next

Proceed to [“Granting the IBM MQ Agent access to queue manager resources”](#) on page 14.

Granting the IBM MQ Agent access to queue manager resources

The IBM MQ Agent MCP server needs access to your IBM MQ queue manager resources to fetch information. You grant this access by using standard IBM MQ APIs.

About this task

Because you use standard APIs, setting an authentication and authorization policy for the MCP server user is the same as for any other IBM MQ user. The IBM MQ Agent requires read-only access for querying configuration details, observing real-time object state, and diagnostic fault-finding. To do further work with a queue manager, the IBM MQ Agent requires additional access. For example, the agent needs permission to connect to a queue manager to send PCF commands to it and receive responses.

The example commands in this topic use the **setmqaut** command, which you can use on AIX®, Linux, or Windows. On these operating systems, you can also use the **SETAUTHREC** command. For IBM i, use the **GRTMQMAUT** command. On z/OS®, use RACF® profiles. For examples on all operating systems, see [Granting read-only access to all resources on a queue manager](#).

Note: Record the MCP server user name that you use; you will specify it when you deploy the IBM MQ Agent.

Procedure

1. Decide on the queue managers that you want the IBM MQ Agent to work with, noting the following restrictions.
 - The number of queue managers that the IBM MQ Agent can work with is limited to 20 because higher numbers impact performance. If you exceed this limit, the IBM MQ Agent MCP server does not start and generates an error. If you want the IBM MQ Agent to work with more than 20 queue managers, you must deploy another instance of the IBM MQ Agent.
 - The IBM MQ Agent does not support the use of [queue manager groups](#) or duplicate queue manager names. The MCP server ignores these names.
 - The IBM MQ Agent does not support the use of asterisks in queue manager names. The IBM MQ Agent finds such names ambiguous when you use them in the chat assistant, and asks you to clarify.
2. Grant access for the MCP server user to work with each queue manager.

To work with a queue manager, the IBM MQ Agent needs to connect to the queue manager to put PCF commands and receive responses.

- a) Grant access for the MCP server to connect to a queue manager to query configuration.

The optional `+ctrl` parameter enables the IBM MQ Agent to access the queue manager error logs. For example:

```
setmqaut -m <QMGrName> -t qmgr -g <MCPServerUserName> +connect +ctrl
```

- b) Grant put access for the MCP server user to put PCF messages to a queue manager's command queue to run read-only PCF commands.

For example:

```
setmqaut -m <QMGrName> -n SYSTEM.ADMIN.COMMAND.QUEUE -t queue -g <MCPServerUserName> +dsp +inq +put
```

- c) Grant get access for the MCP server user to receive PCF responses from the queue manager. For example:

```
setmqaut -m <QMGrName> -n SYSTEM.DEFAULT.MODEL.QUEUE -t queue -g <MCPServerUserName> +dsp +get
```

3. Grant access for the MCP server user to inspect dead-letter queues and transmission queues to enable the IBM MQ Agent to read messages to get header information.

- a) Grant access for the MCP server user to inspect the dead-letter queue for the queue manager so that the IBM MQ Agent can read a message to identify a dead-letter header.

For example:

```
setmqaut -m <QMGrName> -n <dead_letter_queue_name> -t queue -g <MCPServerUserName> +dsp +browse
```

- b) Grant access for the MCP server to inspect transmission queues so that the IBM MQ Agent can read a message to identify a transmission queue header. (Alternatively, you could grant read-only access for all queue manager resources, as described in the next step.)

For example:

```
setmqaut -m <QMGrName> -n <transmission_queue_name> -t queue -g <MCPServerUserName> +dsp +browse
```

4. Grant read-only access for the MCP server user to other queue manager resources.

If you want to grant read-only access to all queue manager resources, follow the instructions in [Granting read-only access to all resources on a queue manager](#), using the MCP server user for the GroupName. However, you need to grant access only to objects that you want the IBM MQ Agent to work with, which might be specific object types or individual objects.

For example, run the following command to grant read-only access to an individual queue:

```
setmqaut -m <QMGrName> -n <ApplicationQueueName> -t queue -g <MCPServerUserName> +dsp +browse
```

Grant read-only access to an individual channel:

```
setmqaut -m <QMGrName> -n <ApplicationChannelName> -t channel -g <MCPServerUserName> +dsp
```

Grant read-only access to all queues:

```
setmqaut -m QMGrName -n '*' -t queue -g <MCPServerUserName> +dsp +browse
```

What to do next

Proceed to [“Deploying the IBM MQ Agent into your cluster”](#) on page 15.

Deploying the IBM MQ Agent into your cluster

Deploy the IBM MQ Agent into your cluster and configure it to be able to connect to your IBM MQ queue managers. Deploying the IBM MQ Agent creates a pod with three containers: one for the IBM MQ Agent, one for the embedding LLM, and one for a Model Context Protocol (MCP) server that connects to your queue managers

Before you begin

- Ensure that you meet the requirements and that you completed the previous steps listed in [“Installing and configuring the IBM MQ Agent”](#) on page 11.
- Install the following command-line utilities in your cluster (the instructions in this task use commands, though you can use your container environment’s web interface if you prefer):
 - **oc** (if you are using Red Hat OpenShift Container Platform). For more information, see the documentation for your version of Red Hat OpenShift Container Platform. For example, [Getting started with the OpenShift CLI](#).
 - **kubect1** (if you are using Amazon Elastic Kubernetes Service). For more information, see [Set up kubect1 and eksct1](#) in the Amazon EKS documentation.

- **helm** version 3 or above. For more information, see for example [Working with Helm charts](#) in the Red Hat OpenShift documentation or [Deploy applications with Helm on Amazon EKS](#) in the Amazon EKS documentation.
- IBM Cloud Container Registry CLI. For more information, see [Setting up the Container Registry CLI and namespace](#) in the IBM Cloud documentation.
- Deploy the IBM License Service into your cluster. This software is required for monitoring license usage of queue managers. For instructions, see [Installing License Service from public container registry](#) in the IBM Cloud Pak foundational services documentation.

About this task

In this task, you create the objects in your cluster that the IBM MQ Agent requires to connect to your IBM MQ installation and instance of IBM MQ Console. On Red Hat OpenShift, you can create objects by using either the console or the CLI. The following steps use the CLI.

You start by creating the following objects:

- A Kubernetes ConfigMap object that contains information about your IBM MQ queue managers, in the form of a client channel definition table (CCDT).
- Kubernetes Secret objects that contain credentials and keys for secure connections.
- An override file, to override some of the default values for the IBM MQ Agent configuration properties.

You then deploy the IBM MQ Agent, which creates more objects in your cluster, including a pod with containers for the following components:

- The IBM MQ Agent.
- An MCP server that uses an IBM MQ client with your CCDT ConfigMap to connect to your queue managers. An MCP server is a program that connects AI agents to external data, tools, and systems. For more information, see [Understanding MCP servers](#) in the Model Context Protocol documentation.
- The embedding foundation model that the IBM MQ Agent uses to answer IBM MQ documentation questions. For more information, see [“Foundation models for the IBM MQ Agent” on page 47](#).

For more information about config maps and secrets, see the documentation for your platform. For example, [Creating and using config maps](#) and [Providing sensitive data to pods by using secrets](#) in the Red Hat OpenShift documentation..

Procedure

1. Log in to your cluster by using the **oc** or the **kubectl** command. For more information, see the documentation for your version of Red Hat OpenShift or Amazon EKS.

For example, [Logging in to the OpenShift CLI](#).

2. Create a namespace (project on Red Hat OpenShift) for the IBM MQ Agent.
For example, run the following command.

On Red Hat OpenShift:

```
oc new-project mq-agent
```

On Amazon EKS:

```
kubectl create namespace <your-namespace>
```

3. Switch to the created namespace (project) where you want to deploy the IBM MQ Agent.
For example, run the following command.

On Red Hat OpenShift:

```
oc project mq-agent
```

On Amazon EKS:

```
kubectl config set-context --current --namespace=<namespace-name>
```

4. In your cluster, create a Secret object to contain your key for pulling software from the IBM Cloud Container Registry.

The key must be an image pull secret called `ibm-entitlement-key`. The key will be used during deployment to pull the IBM MQ Agent and IBM MQ Agent images from the IBM Cloud Container Registry.

- a) Get the entitlement key that is assigned to your IBM ID. Log in to [My IBM Container Software Library](#) with the IBM ID and password that are associated with the entitled software.
- b) In the **Entitlement keys** section, click **Copy** to copy the entitlement key to the clipboard.
- c) Create an image pull secret called `ibm-entitlement-key`, by running the following command.

On Red Hat OpenShift:

```
oc create secret docker-registry ibm-entitlement-key \
--docker-server=cp.icr.io \
--docker-username=cp \
--docker-password=<entitlement_key> \
--docker-email=<user_email> \
--namespace=<namespace>
```

On Amazon EKS:

```
kubectl create secret docker-registry ibm-entitlement-key \
--docker-server=cp.icr.io \
--docker-username=cp \
--docker-password=<entitlement_key> \
--docker-email=<user_email> \
--namespace=<namespace>
```

In these commands, the `<entitlement_key>` is the entitlement key that you copied earlier, `<user_email>` is the IBMID that is associated with the entitled software, and `<namespace>` is the namespace where you are deploying the IBM MQ Agent.

5. Create a ConfigMap object to contain information about the IBM MQ queue managers that you want the IBM MQ Agent to work with.

This information is used by the IBM MQ Agent MCP server to connect to your queue managers.

- a) Create an IBM MQ client channel definition table (CCDT), in JSON format.
For example, copy the following JSON code and save it as a local file called `my-ccd.json`:

```
{
  "channel": [
    {
      "name": "QM1.SVRCONN",
      "clientConnection": {
        "connection": [
          {
            "host": "qm1.yourdomain.com",
            "port": 1414
          }
        ],
        "queueManager": "qm1"
      },
      "transmissionSecurity": {
        "cipherSpecification": "ANY_TLS13_OR_HIGHER",
        "certificateLabel": "CN=agent"
      },
      "type": "clientConnection"
    }
  ]
}
```

- b) Edit the file to replace the contents with your own information.

Add an entry for each of the queue managers that you granted access for in [“Granting the IBM MQ Agent access to queue manager resources”](#) on page 14.

Restrictions:

- Do not add more than 20 queue managers in the CCDT; higher numbers impact performance. If you exceed this limit, the IBM MQ Agent MCP server does not start and an error is generated in the MCP server container log.
- Do not use queue manager groups or duplicate queue manager names in the CCDT; such names are ignored by the MCP server.
- Do not use asterisks in queue manager names in the CCDT. The IBM MQ Agent finds such names ambiguous; the IBM MQ Agent chat assistant will lead to clarification questions.

- c) Create the ConfigMap from the JSON file
For example, run the following command.

On Red Hat OpenShift:

```
oc create configmap my-ccdt --from-file=ccdt.json=my-ccdt.json
```

On Amazon EKS:

```
kubectl create configmap my-ccdt --from-file=ccdt.json=my-ccdt.json
```

6. If queue manager connections in your CCDT require a password to connect, create a Secret object to contain the necessary credentials.

The IBM MQ Agent MCP server uses these credentials when it connects to the queue managers. The secret must contain a key called `my-mcp-configuration`, with a value that lists the username and password for each connection.

- a) Create a `.ini` file.

Copy the following `.ini` file template and save it as a local file called `my-mcpconfig.ini`:

```
[mqenv]
# Default username and password for client connectivity to IBM MQ queue managers
# (you can override this for individual queue managers, as shown in the following
section)
username = mq-username
password = mq-password
# To override connection information for an individual queue manager
# Provide one or more stanzas in the form [QM:<name>]
[QM:QM1]
username=<QM1-username>
password=<QM1-password>
[routes]
# Enable /health custom route for generic health check of running MCP server
health = enabled
```

- b) Edit the file to replace the username and stanza details with your own information (leave the `[routes]` and following lines).

- c) Create the Secret object from the `.ini` file.

For example, run the following command.

On Red Hat OpenShift:

```
oc create secret generic my-mcp-configuration --from-file=mcpconfig.ini=my-mcpconfig.ini
```

On Amazon EKS:

```
kubectl create secret generic my-mcp-configuration --from-file=mcpconfig.ini=my-
mcpconfig.ini
```

7. If queue manager connections in your CCDT require a TLS connection, create a Secret object to contain a keystore with the necessary keys.

The IBM MQ Agent MCP server uses this keystore when it connects to the queue managers.

- a) Create a PKCS#12 keystore called `mcpserver.p12`, and record the password that you set for the keystore.

The keystore must be a valid PKCS#12 keystore for use with IBM MQ.

b) Create the following keys then import them into the keystore:

- Public keys for the queue managers that you want the IBM MQ Agent to interact with
- A public and private key for the IBM MQ Agent MCP server (the IBM MQ client application)

c) Create the Secret object from the `mcpserver.p12` file.

For example, run the following command to create a secret called `agent-qm-keystore`.

On Red Hat OpenShift:

```
oc create secret generic agent-qm-keystore --from-file=keystore.p12=mcpserver.p12
```

On Amazon EKS:

```
kubectl create secret generic agent-qm-keystore --from-file=keystore.p12=mcpserver.p12
```

8. Create a Secret object to contain a TLS key for the IBM MQ Console to use to connect to the IBM MQ Agent.

The IBM MQ Console communicates with the agent by using an HTTPS API endpoint, which needs a key and a certificate. You must create a TLS Kubernetes Secret with the keys `tls.key`, `tls.crt`, and `ca.crt`. You also need the `ca.crt` file for configuring the IBM MQ Console.

a) Optional: For testing purposes, you could create a self-signed certificate with OpenSSL.

Create a certificate authority:

```
openssl genpkey -algorithm rsa -pkeyopt rsa_keygen_bits:4096 -out "ca.key"
openssl req -x509 -new -nodes -key "ca.key" -sha512 -days 30 -subj "/CN=selfsigned-ca" -out "ca.crt"
```

Generate a certificate, replacing `<cluster_name>` with the name of your cluster.:

```
openssl req -new -nodes -out "cert.csr" -newkey rsa:4096 -keyout "cert.key" -subj "/"
CN=*.apps.<cluster_name>
openssl x509 -req -in "cert.csr" -CA "ca.crt" -CAkey "ca.key" -CAcreateserial -out
"cert.crt" -days 30 -sha512
```

b) Create a TLS Kubernetes Secret object called `mq-agent-tls` from the key and certificate, by running the following command.

On Red Hat OpenShift:

```
oc create secret generic mq-agent-tls --type="kubernetes.io/tls" --from-
file=tls.key=cert.key --from-file=tls.crt=cert.crt --from-file=ca.crt
```

On Amazon EKS:

```
kubectl create secret generic mq-agent-tls --type="kubernetes.io/tls" --from-
file=tls.key=cert.key --from-file=tls.crt=cert.crt --from-file=ca.crt
```

9. Provide access to the IBM MQ Agent Helm chart in the IBM Helm chart repository.

- If you already have the IBM Helm chart repository, run the following command to update it:

```
helm repo update ibm-helm
```

- If you do not have the IBM Helm chart repository, run the following command to add it:

```
helm repo add ibm-helm https://raw.githubusercontent.com/IBM/charts/master/repo/ibm-helm
```

10. Create an overrides file to override some of the default values for the IBM MQ Agent configuration properties.

The default values are set in the `values.yaml` file in the Helm chart package. You must override some of the values, for example `license.license` and `license.accept`, which specify the license that you are entitled to and whether you accept it. You can also specify that the IBM MQ Agent connects to a vLLM AI runtime instead of the default, IBM watsonx.ai Runtime.

To override the default values, create a new YAML file called `overrides.yaml`, then add to that file the properties that you want to override, along with their new values. You will specify the `overrides.yaml` file when you install the IBM MQ Agent. You might also need the `overrides.yaml` when you upgrade the agent.

For a list of values that you can set, including which values are required, see [“IBM MQ Agent Helm chart values”](#) on page 22. If you want to see the default values, you can download the `values.yaml` file by using Helm; for more information, see [helm get values](#) in the Helm documentation.

An example `overrides.yaml` file:

```
license:
  accept: true
  license: L-CXDB-XTZ9J8
metadata:
  annotations:
    productID: "208423bb063c43288328b1d788745b0c"
    productName: "IBM MQ Advanced"
    productMetric: "PROCESSOR_VALUE_UNIT"
    productChargedContainers: "All"
agent:
  model:
    providers:
      myWatsonxRuntime:
        watsonxService:
          projectID: my-watsonx-project
          key: my-watsonx-project-key
mcpServer:
  ccdtConfigMapName: my-ccdt
  keyStoreSecretName: agent-qm-keystore
  keyStorePassword: <mcpserver.p12_keystore_password>
  configurationSecretName: my-mcp-configuration
route:
  enabled: true
```



Attention: Do not set the `route.enabled` property to `True` on Amazon EKS. You must manually create an AWS Network Load Balancer for external access. See the [Step 2 in Configuring the IBM MQ Console to connect to the IBM MQ Agent](#). For more information about this parameter, see [IBM MQ Agent Helm chart values](#).

11. Deploy the IBM MQ Agent by using the **helm install** command.

```
helm install <release_name> ibm-helm/ibm-mq-agent --values overrides.yaml
```

where `<release_name>` is a name that you choose for the release, for example `ibm-mq-agent`.

Results

The Helm deployment creates the following objects in your cluster:

- A pod with three containers: one for the IBM MQ Agent itself, one for the IBM MQ Agent MCP server, and one for the embedding foundation model that the IBM MQ Agent uses for IBM MQ documentation questions.
- A service for connecting to the IBM MQ Agent from within the cluster
- **Optional:** For Red Hat OpenShift only, a Route for connecting to the IBM MQ Agent from outside the cluster
- A secret that contains your IBM watsonx.ai or vLLM credentials
- A secret that contains an IBM MQ client INI file, with the keystore password
- A secret with a generated API key for the IBM MQ Console to use to access the IBM MQ Agent
- A ServiceAccount for the pod to use. No access to the Kubernetes API is required by the pod

What to do next

Check that your installation was successful.

1. Ensure that you are connected to your project. For example, run the following command.

On Red Hat OpenShift:

```
oc project <your_project_name>
```

On Amazon EKS:

```
kubectl config set-context --current --namespace=<your_namespace>
```

2. Watch the pod status to ensure that the new pod starts successfully. For example, run the following command.

On Red Hat OpenShift:

```
oc get pods -w
```

On Amazon EKS:

```
kubectl get pods -w
```

The STATUS should be Running and the pod name should be followed by 3/3, which indicates that all the deployed containers are running. For example:

NAME	READY	STATUS	RESTARTS	AGE
ibm-mq-agent-mq-agent-79565f867c-p2d5j	3/3	Running	0	1h

Press Ctrl+C to stop watching.

3. Optionally, you can run the following command to get the pod details, including the containers that were deployed (agent, embedding-service, and mcp-server).

On Red Hat OpenShift:

```
oc describe pod <your_pod_name>
```

On Amazon EKS:

```
kubectl describe pod <your_pod_name>
```

4. Examine the container logs to check that the running applications started correctly. Note that the containers might take 1 to 2 minutes to start, complete their initial setup tasks, then establish connectivity. During this time, you might observe error messages in the logs because service dependencies are not yet available.

- For example, run the following command to examine logs for the agent container.

On Red Hat OpenShift:

```
oc logs <your_pod_name> -c agent | grep -A 2 -B 2 "Application startup complete"
```

On Amazon EKS:

```
kubectl logs <your_pod_name> -c agent | grep -A 2 -B 2 "Application startup complete"
```

You should see output similar to the following:

```
INFO    : Initializing API server
INFO    : Agent Build Info: {'commit_id': '5e9b78a', 'build_version': 'v1.0.0'}
INFO:    Application startup complete.
INFO:    Uvicorn running on https://0.0.0.0:8001 (Press CTRL+C to quit)
INFO    : 10.254.24.2:52616 - New thread ID created: TD_2ad15dd3-18b3-4139-b761-081aaf40dc89
```

- Run the following command to examine logs for the mcp-server container.

On Red Hat OpenShift:

```
oc logs ibm-mq-agent-ibm-mq-agent-6f6f48d85-qz4kn -c mcp-server | grep -A 2 -B 2 "Application startup complete"
```

On Amazon EKS:

```
kubectl logs ibm-mq-agent-ibm-mq-agent-6f6f48d85-qz4kn -c mcp-server | grep -A 2 -B 2 "Application startup complete"
```

You should see output similar to the following:

```
INFO:      Started server process [5]
INFO:      Waiting for application startup.
INFO:      Application startup complete.
INFO:      Uvicorn running on http://0.0.0.0:8000 (Press CTRL+C to quit)
INFO:      10.254.20.2:42122 - "GET /health HTTP/1.1" 200 OK
```

- Run the following command to examine logs for the `embedding-service` container.

On Red Hat OpenShift:

```
oc logs ibm-mq-agent-ibm-mq-agent-6f6f48d85-qz4kn -c embedding-service | grep -A 2 -B 2 "Application startup complete"
```

On Amazon EKS:

```
kubectl logs ibm-mq-agent-ibm-mq-agent-6f6f48d85-qz4kn -c embedding-service | grep -A 2 -B 2 "Application startup complete"
```

You should see output similar to the following:

```
182026-06-24 12:45:51,418 - __main__ - INFO - MQ Embedding & Reranking Service ready to accept requests
19INFO: Application startup complete.
20INFO: Uvicorn running on http://0.0.0.0:8002 (Press CTRL+C to quit)
```

If the logs contain errors, fix the problem, and then retry. For example, the following error message in the MCP server log indicates that you have too many queue managers in your CCDT:

```
CCDT contains 22 queue managers, which exceeds the recommended maximum of 20.
```

Correct the CCDT, refresh the ConfigMap object, and then restart the pod.

For more information about error logs, see [“Troubleshooting the IBM MQ Agent” on page 43](#)

When the pod is running as expected, proceed to [“Configuring the IBM MQ Console to connect to the IBM MQ Agent” on page 30](#).

IBM MQ Agent Helm chart values

You deploy the IBM MQ Agent by using a Helm chart. The Helm chart package contains default values for various configuration properties, some of which you must override when you deploy the IBM MQ Agent.

You override the values by setting new values in an override file, as described in [“Deploying the IBM MQ Agent into your cluster” on page 15](#). The following table lists the properties that you can set, their default values, and whether the property is required.

Licensing configuration properties

These properties set your license version, whether you accept the license, and annotations for tracking usage.

Table 1. IBM MQ Agent licensing configuration properties			
Property	Default value	Required	Description
license.license	L - CXDB - XTZ9J8	Yes	The IBM MQ Agent license. Change this value to the license that you are accepting. For a list of valid values, see the following table.
license.accept	false	Yes	Indicates whether you accept the terms of the IBM license agreement. If you accept the terms, change this value to true.
metadata.annotations	See the information later in this section	Yes	The annotations to add to the IBM MQ Agent pod so that the IBM License Service can track usage, as described in the Container Licenses IBM web page. Change the annotations, if required, to match your IBM MQ entitlement. For a list of annotations, see the information later in this section.

Valid values for the license.license property:

Table 2. Valid values for the license.license property		
Value of license.license	License information	Applicable IBM MQ versions
L - CXDB - XTZ9J8	IBM MQ Advanced with AI Agents 10.0	10.0.0
L - VEMB - RABZSN	IBM Cloud Pak for Integration 16.2.0	10.0.0
L - UHEB - WGGTYF	IBM Cloud Pak for Integration Limited Edition 16.2.0	10.0.0

Valid values for the metadata.annotations property:

Annotations for IBM MQ Advanced entitlement with a VPC Metric

This is the default set of annotations.

```
productID: "208423bb063c43288328b1d788745b0c"
productName: "IBM MQ Advanced"
productMetric: "VIRTUAL_PROCESSOR_CORE"
productChargedContainers: "All"
```

Annotations for IBM MQ Advanced entitlement with a PVU metric

```
productID: "208423bb063c43288328b1d788745b0c"
productName: "IBM MQ Advanced"
productMetric: "PROCESSOR_VALUE_UNIT"
productChargedContainers: "All"
```

Annotations for IBM Cloud Pak for Integration entitlement with an IBM MQ Advanced ratio and a VPC metric

```
cloudpakId: "c8b82d189e7545f0892db9ef2731b90d"
cloudpakName: "IBM Cloud Pak for Integration"
productID: "208423bb063c43288328b1d788745b0c"
productName: "IBM MQ Advanced"
productMetric: "VIRTUAL_PROCESSOR_CORE"
productChargedContainers: "All"
productCloudpakRatio: "2:1"
```

Example:

```

license:
  accept: false
  license: L-CXDB-XTZ9J8
metadata:
  annotations:
    productID: "208423bb063c43288328b1d788745b0c"
    productName: "IBM MQ Advanced"
    productMetric: "VIRTUAL_PROCESSOR_CORE"
    productChargedContainers: "All"

```

Diagnostics configuration properties

These properties set values for the IBM MQ Agent self-diagnostic capabilities.

Table 3. IBM MQ Agent diagnostic configuration properties			
Property	Default value	Required	Description
agent.diagnostics.enabled	true	No	Whether diagnostics are enabled. You can collect dump and trace information about the IBM MQ Agent to provide to IBM Support. For more information, see “Collecting diagnostic and trace information for IBM support” on page 44.
agent.diagnostics.authentication.enabled	true	No	Whether to use a key to access diagnostics. The IBM MQ Agent automatically generates a key, which is stored as a Secret object called <code>ibm-mq-agent-<release_name>-diag</code> , where <code><release_name></code> is the name that you chose when you ran the helm command to deploy the IBM MQ Agent. If you prefer, you can provide your own key (see <code>agent.diagnostics.authentication.adminKey</code>). Set the value of this property to <code>false</code> to disable authentication, which means that any user can generate dump and trace information.
agent.diagnostics.authentication.adminKey	Empty	No	Your key for accessing diagnostics, if you choose to use one. The key must be an alphanumeric sequence (mixed case is permitted) not exceeding 32 characters in length.

Example:

```

agent:
  diagnostics:
    enabled: true
  authentication:
    enabled: true
    adminKey: myKEY86

```

AI runtime configuration properties

These properties set values for the AI inference runtime that runs the foundation models that the IBM MQ Agent uses. For lists of supported runtimes and foundation models, see [“Foundation models for the IBM MQ Agent”](#) on page 47.

In the next property table, the variables <runtime> and <type> have the following values, depending on your chosen runtime.

Variable	IBM watsonx.ai Runtime	vLLM
<runtime>	myWatsonxRuntime (default)	myVLLMRuntime
<type>	watsonxService (default)	vllsmService

Table 4. IBM MQ Agent model provider configuration properties

Property	Default value	Required	Description
agent.model.providers.defaultProvider	myWatsonxRuntime	Yes	The name of the section of the .yaml override file that defines the provider that you want to use as the default for each foundation model. You can override this value in the values for the foundation models. Valid values are myWatsonxRuntime (for IBM watsonx.ai Runtime) and myVLLMRuntime (for vLLM).
agent.model.providers.<runtime>.uri	https://us-south.ml.cloud.ibm.com	Yes	The URI of your chosen AI runtime. If you are using IBM watsonx.ai Runtime, this is the URI of the IBM watsonx.ai data center. This data center must provide the foundation models that the IBM MQ Agent requires.
agent.model.providers.<runtime>.type	watsonxService	Yes	The format that the IBM MQ Agent uses to communicate with the LLM endpoint. Set to either watsonxService or vllmService, depending on your chosen runtime.
agent.model.providers.<runtime>.<type>.projectId	Empty	Yes (if you are using IBM watsonx.ai Runtime)	The alphanumeric project identifier for the IBM watsonx.ai project, which you create when you follow the instructions in “Preparing an AI runtime for the IBM MQ Agent” on page 11 . This property does not apply to vLLM runtimes.
agent.model.providers.<runtime>.<type>.key	Empty	Yes (if you are using IBM watsonx.ai Runtime)	Your API key for accessing your chosen runtime. For vLLM, this property is optional. For IBM watsonx.ai Runtime, this is the IBM Cloud API key that you create when you follow the instructions in “Preparing an AI runtime for the IBM MQ Agent” on page 11 .
agent.model.roles.primaryLLM.provider	myWatsonxRuntime	Yes	The provider of the primary foundation model. The IBM MQ Agent uses this model for complex tasks. If not set, defaults to the value of agent.model.providers.defaultProvider.
agent.model.roles.primaryLLM.modelId	openai/gpt-oss-120b	Yes	The identifier of the primary foundation model

Table 4. IBM MQ Agent model provider configuration properties (continued)

Property	Default value	Required	Description
agent.model.roles.secondaryLLM.enabled	true	Yes	Whether use of the secondary model is enabled. The IBM MQ Agent uses this model for simple tasks. Disable the secondary model if you want to use only the primary model. For more information, see “Foundation models for the IBM MQ Agent” on page 47.
agent.model.roles.secondaryLM.provider	myWatsonxRuntime	No	The provider of the secondary foundation model. If not set, defaults to the value of agent.model.providers.defaultProvider.
agent.model.roles.secondaryLLM.modelId	ibm/granite-4-h-small	No	The identifier of the secondary foundation model

Example for IBM watsonx.ai Runtime:

```
agent:
model:
  providers:
    myWatsonxRuntime:
      type: watsonxService
      uri: https://us-south.ml.cloud.ibm.com
      watsonxService:
        projectID: my-watsonx-project
        key: my-watsonx-project-key
      defaultProvider: myWatsonxRuntime
  roles:
    primaryLLM:
      modelId: "openai/gpt-oss-120b"
    secondaryLLM:
      enabled: true
      modelId: "ibm/granite-4-h-small"
```

Example for vLLM:

```
agent:
model:
  providers:
    myVLLMRuntime:
      type: vllmService
      uri: http://vllm-service:8000/v1
      vllmService:
        key: your-vllm-api-key
      defaultProvider: myVLLMRuntime
  roles:
    primaryLLM:
      modelId: "openai/gpt-oss-120b"
    secondaryLLM:
      enabled: false
```

IBM MQ Console configuration properties

This property sets a TLS key for communication with the IBM MQ Console.

Table 5. IBM MQ Console configuration properties

Property	Default value	Required	Description
agent.tlsSecretName	mq-agent-tls	Yes	The name of a Secret that contains a TLS key for IBM MQ Console to use to communicate with the agent's HTTPS API endpoint. You create the Secret when you follow the instructions in “Deploying the IBM MQ Agent into your cluster” on page 15.

Example:

```
agent:
  tlsSecretName: "mq-agent-tls"
```

Container resource configuration properties

These properties set resource values for the IBM MQ Agent container. See the following section for similar values for the IBM MQ Agent MCP server container.

Table 6. IBM MQ Agent resource configuration properties

Property	Default value	Required	Description
agent.deployment.resources.requests.memory agent.deployment.resources.limits.memory	4Gi	Yes	The amount of memory for the IBM MQ Agent container. The IBM MQ Agent is optimized to run with the default values; reducing them is not recommended.

MCP server configuration properties

These properties set values for the IBM MQ Agent MCP server.

Table 7. IBM MQ Agent MCP server configuration properties

Property	Default value	Required	Description
mcpServer.ccdtConfigMapName	Empty	Yes	The name of your CCDT ConfigMap, which you create when you follow the instructions in “Deploying the IBM MQ Agent into your cluster” on page 15.
mcpServer.keyStoreSecretName	Empty	No	The name of a Secret that contains the PKCS#12 keystore for the IBM MQ Agent to use when it connects to your queue managers, if required. You create the Secret when you follow the instructions in “Deploying the IBM MQ Agent into your cluster” on page 15.
mcpServer.keyStorePassword	Empty	No	The password for the mcpServer.keyStoreSecretName keystore.

Table 7. IBM MQ Agent MCP server configuration properties (continued)

Property	Default value	Required	Description
mcpServer.configurationSecretName	Empty	No	The name of a Secret that contains a configuration INI file, which you create when you follow the instructions in “Deploying the IBM MQ Agent into your cluster” on page 15.
mcpServer.deployment.resources.requests.memory mcpServer.deployment.resources.limits.memory	1Gi	Yes	The amount of memory for the IBM MQ Agent MCP server container. The IBM MQ Agent is optimized to run with the default memory value; reducing it is not recommended.

Example:

```
mcpServer:
  ccdtConfigMapName: my-ccdt
  keyStoreSecretName: agent-qm-keystore
  keyStorePassword: <mcpserver.p12_keystore_password>
  configurationSecretName: my-mcp-configuration
```

Registry configuration properties

These properties set values for getting the IBM MQ Agent packages from a registry.

Table 8. IBM MQ Agent registry configuration properties

Property	Default value	Required	Description
imagePullSecrets	ibm-entitlement-key	Yes	A list that contains an image pull secret that contains your IBM Cloud credentials. The credentials are used to pull the IBM MQ Agent and IBM MQ Agent MCP server images from the IBM Cloud registry. You create the image pull secret when you follow the instructions in “Deploying the IBM MQ Agent into your cluster” on page 15. If you choose to use a private registry that requires credentials, create an image pull secret for that registry then specify it in this value.

Table 8. IBM MQ Agent registry configuration properties (continued)

Property	Default value	Required	Description
privateRegistry	Empty	No	A private registry that contains the IBM MQ Agent and IBM MQ Agent MCP server images. For example, my-private-registry.com:5000. You might choose to use a private registry if you are deploying the IBM MQ Agent in an air-gap (offline) environment. Note: If you change this value (for example, if you install with one value then run the <code>helm upgrade</code> command with a new value), the agents and MCP servers that you installed with the previous value are restarted. For some guidance on using private registries, see Installing the IBM MQ Operator on Amazon EKS in an air-gap environment in the IBM MQ in containers and IBM Cloud Pak for Integration documentation.

Example:

```
imagePullSecrets:
- ibm-entitlement-key
privateRegistry: my-private-registry.com:5000
```

Storage configuration properties

This property sets the amount of storage that is used by the IBM MQ Agent containers.

Table 9. IBM MQ Agent storage configuration properties

Property	Default value	Required	Description
ephemeralStorage	500Mi	Yes	The amount of ephemeral storage. This storage is shared by both IBM MQ Agent containers. The storage is of Kubernetes type <code>emptyDir</code> so uses the host's disk rather than networked file storage. The IBM MQ Agent is optimized to run with the default value; reducing it is not recommended.

Communication configuration properties

These properties set values for communication with the IBM MQ Agent in the cluster.

Table 10. IBM MQ Agent communication configuration properties

Property	Default value	Required	Description
route.enabled	false	No	Whether to create a Red Hat OpenShift Route to enable you to contact the IBM MQ Agent from an IBM MQ Console outside the cluster. If you want to use something other than a Route (for example a Kubernetes Ingress, or a cloud-specific load balancer), leave this value as <code>false</code> but note that later instructions refer to the Route. This property does not apply on Amazon EKS. If you are using Amazon EKS and want to enable similar access, create an AWS Network Load Balancer, as described in Configuring the IBM MQ Console to connect to the IBM MQ Agent.

Example:

```
route:
  enabled: true
```

Configuring the IBM MQ Console to connect to the IBM MQ Agent

The final task in installing and configuring the IBM MQ Agent is to configure your IBM MQ Console to connect to the agent. The IBM MQ Console provides the chat interface that you use to ask questions of the agent.

Before you begin

Ensure that you meet the requirements and complete the earlier steps in [“Installing and configuring the IBM MQ Agent”](#) on page 11 to get access to an AI runtime and to deploy the IBM MQ Agent into your cluster.

Procedure

1. Add the IBM MQ Agent certificate authority to the IBM MQ Console truststore, to secure the connection between the IBM MQ Agent and the IBM MQ Console with TLS.

The IBM MQ Agent certificate authority is the `ca.crt` file that you used in step 8 of [“Deploying the IBM MQ Agent into your cluster”](#) on page 15 (the step to create a secret object to contain a TLS key for communication between the IBM MQ Console and the IBM MQ Agent).

- a) Ensure that the IBM MQ Console ran at least once after being installed and configured; starting the IBM MQ Console creates the `mqweb` server keystore.
- b) Copy the `ca.crt` file to the location where you are hosting the IBM MQ Console.
- c) Run the following command, replacing `<mq_console_dir>` with the hosting location of your IBM MQ Console and `<password>` with the password of your `trust.jks` truststore, to import the certificate authority into the truststore:

```
runmqktool -importcert -keystore <mq_console_dir>/web/installations/installationName/
servers/mqweb/resources/security/trust.jks -storepass <password> -alias label -file ~/
ca.crt
```

If you don't have the [runmqktool](#) utility installed, you can use the Java **keytool** command instead.

2. Get the IBM MQ Agent hostname.

You specify this hostname when you configure the IBM MQ Console in a later step. If you want to access the IBM MQ Agent from an IBM MQ Console outside your cluster, find the hostname for the Red Hat OpenShift Route, or on Amazon EKS, create an AWS Network Load Balancer for the IBM MQ Agent then find its hostname.

- On Red Hat OpenShift, the IBM MQ Agent is typically accessible from outside your cluster by using a Route. If you set the option to create a Route when you installed the IBM MQ Agent, find the hostname by logging into your cluster then running the following command (remove `&& echo` on Windows; this part of the command makes the output easier to read but does not work on Windows):

```
$ oc get route ibm-mq-agent-<release_name> -o jsonpath="{.status.ingress[0].host}" && echo
```

where `<release_name>` is the name that you chose when you ran the **helm** command to deploy the IBM MQ Agent.

- On Amazon EKS, create an AWS Network Load Balancer service to enable access from outside the cluster.

a. Create the following YAML file and save it in your current directory, for example as `nlb.yaml`:

```
apiVersion: v1
kind: Service
metadata:
  name: "ibm-mq-agent-mq-agent-nlb"
  annotations:
    service.beta.kubernetes.io/aws-load-balancer-type: "external"
    service.beta.kubernetes.io/aws-load-balancer-scheme: "internet-facing"
    service.beta.kubernetes.io/aws-load-balancer-nlb-target-type: "ip"
spec:
  type: LoadBalancer
  externalTrafficPolicy: Local # Preserves the client IP
  selector:
    app.kubernetes.io/name: "ibm-mq-agent"
    app.kubernetes.io/instance: "mq-agent"
    app.kubernetes.io/managed-by: "Helm"
    app.kubernetes.io/component: integration
  ports:
    - name: tls-passthrough
      protocol: TCP
      port: 443 # The external port used for internet facing traffic
      targetPort: 8001
```

- b. Run the following command to apply the `nlb.yaml` file, replacing `<namespace>` with the namespace in which the IBM MQ Agent is installed:

```
kubectl apply -f nlb.yaml -n <namespace>
```

This command creates a service, which provisions a Network Load Balancer. This process can take a few minutes.

- c. Retrieve the external hostname of the Network Load Balancer by running the following command:

```
kubectl get svc ibm-mq-agent-mq-agent-nlb -o
jsonpath="{.status.loadBalancer.ingress[0].hostname}"
```

`ibm-mq-agent-mq-agent-nlb` is name of the Network Load Balancer service. The `-o` parameter extracts only the hostname from the service status.

If your IBM MQ Console is deployed in the same Amazon EKS cluster as the IBM MQ Agent, you can use the following internal Kubernetes DNS name instead of the external hostname:

```
ibm-mq-agent-mq-agent-internal.<namespace>.svc.cluster.local
```

where `<namespace>` is the namespace in which the IBM MQ Agent is installed.

3. Get the IBM MQ Agent API key.

You specify this key when you configure the IBM MQ Console in a later step. The IBM MQ Agent API key was generated automatically when you installed the IBM MQ Agent. Retrieve the key by logging into your cluster then running the following command (assuming you are using a Linux or macOS system).

- On Red Hat OpenShift:

```
oc get secret ibm-mq-agent-mq-agent-api-key-file -o jsonpath="{.data['api\.key']}" | base64 --decode
```

- On Amazon EKS:

```
kubectl get secret ibm-mq-agent-mq-agent-api-key-file -o jsonpath="{.data['api\.key']}" | base64 --decode
```

Note: On Windows, the **base64** tool that is used in the previous commands does not exist. Instead, run the command without the `| base64 --decode` part, to get the key, then use a Windows tool such as **certutil** or PowerShell to decode the key from Base64 format.

On Red Hat OpenShift, you can also navigate to the secret in the Red Hat OpenShift console then click **Reveal**.

4. Configure the IBM MQ Console to use the IBM MQ Agent.

Update the `mqwebuser.xml` file that contains the configuration for your instance of the IBM MQ Console. For information about the location of this file, see [File structure of the IBM MQ Console and REST API installation component](#). For general information about editing the file, see [Basic configuration for the mqweb server](#).

Notes for container deployments:

- For IBM MQ container deployments, the `mqwebuser.xml` file is a property of the container image. Although you can make changes in a running instance, such changes are likely to be replaced with the image copy when the container is restarted.
 - If you are managing your deployment with the IBM MQ Operator, you can configure the IBM MQ Console by using a ConfigMap object, for example as described in [Configuring the IBM MQ Console with a basic registry using the IBM MQ Operator](#) in the IBM MQ in containers and IBM Cloud Pak for Integration documentation.
- a) Add the following variables, with values that contain the information that you retrieved in the previous steps
- `agentHostname`: a required variable whose value is the hostname that you retrieved in step “2” on page 31. The IBM MQ Console uses this variable to connect to the running instance of the IBM MQ Agent.
 - `agentAPIKey`: an optional variable whose value is the API key that you retrieved in step “3” on page 31. If you add the key to the `mqwebuser.xml` file, the key enables access to the IBM MQ Agent for all users of this instance of the IBM MQ Console. If you want to restrict access to selected users, share the API key directly with those users instead. Each user then adds the key in the **Agents** tab of the **Settings** page when they are logged in to the IBM MQ Console in their browser.

In most cases, you can set the variables as IBM MQ Console properties by using the `setmqweb` command, as follows:

```
setmqweb properties -k agentHostname -v https://YOUR_AGENT_INGRESS_HOSTNAME
setmqweb properties -k agentAPIKey -v YOUR_AGENT_API_KEY
```

Alternatively, you can append the variables to the `server` tags of the `mqwebuser.xml` file. For example:

```
<?xml version="1.0" encoding="UTF-8"?>
<server>
  <variable name="agentHostname" value="https://YOUR_AGENT_INGRESS_HOSTNAME"/>
```

```
<variable name="agentAPIKey" value="YOUR_AGENT_API_KEY"/>
</server>
```

b) Configure the mqweb server to use client certificate authentication.

i) Turn off the default security configuration by commenting out the following line in the `mqwebuser.xml` file.

```
<sslDefault sslRef="mqDefaultSSLConfig"/>
```

ii) Uncomment the section in the `mqwebuser.xml` file that enables client certificate authentication. The section contains text similar to the following:

```
<keyStore id="defaultKeyStore" location="<mq_console_dir>/web/installations/
installationName/servers/mqweb/resources/security/key.jks" type="JKS"
password="<password>"/>
<keyStore id="defaultTrustStore" location="<mq_console_dir>/web/installations/
installationName/servers/mqweb/resources/security/trust.jks" type="JKS"
password="<password>"/>
<ssl id="thisSSLConfig" clientAuthentication="true"
clientAuthenticationSupported="true" keyStoreRef="defaultKeyStore"
trustStoreRef="defaultTrustStore" sslProtocol="TLSv1.2" serverKeyAlias="default"/>
<sslDefault sslRef="thisSSLConfig"/>
```

5. Stop the mqweb server by using the **endmqweb** command.

6. Start the mqweb server by using the **strmqweb** command.

Results

The IBM MQ Console is updated to show an AI (**Open the chat window**) icon in the corner of the screen (you might need to refresh your browser tab to see it).

What to do next

See [“Interacting with the IBM MQ Agent” on page 39](#) for information about how to have a conversation with the IBM MQ Agent.

Upgrading the IBM MQ Agent

Upgrade your existing IBM MQ Agent deployment to a newer version to access new features, improvements, and bug fixes.

Before you begin

- Review the information about the new version in [“What's new and changed, and known issues for the IBM MQ Agent” on page 8](#) to understand any breaking changes or new requirements.
- Verify that your AI runtime contains the foundation models that are required by the new version. For information about required foundation models, see [“Foundation models for the IBM MQ Agent” on page 47](#).
- Ensure that you have access to the cluster where the IBM MQ Agent is deployed.

About this task

To upgrade the IBM MQ Agent, you use the Helm **upgrade** command to update the deployment with a newer version of the Helm chart. The upgrade process preserves your existing configuration and updates the IBM MQ Agent containers to the new version.

During the upgrade, the IBM MQ Agent pod is recreated with the new version. This action causes a brief service interruption while the new containers start. Any active chat sessions are disconnected and users must restart their conversations after the upgrade completes.

Important: The upgrade process does not preserve conversation history. Users should save any important conversations before you begin the upgrade.

Procedure

1. Log in to your cluster by using the **oc** command for Red Hat OpenShift or the **kubectl** command for Amazon EKS to communicate with the cluster.

For more information, see the documentation for your version of Red Hat OpenShift or Amazon EKS. For example, [Logging in to the OpenShift CLI](#) or [Organize and monitor cluster resources](#).

2. Switch to the namespace (project) where the IBM MQ Agent is deployed.

For example:

- For Red Hat OpenShift, use the following command.

```
oc project mq-agent
```

- For Amazon EKS, use the following command.

```
kubectl project namespace=<mq-agent>
```

3. Optional: If you need to update your IBM entitlement key, refresh the `ibm-entitlement-key` secret in the cluster.

If your entitlement key has changed or expired, delete the existing secret then create a new one.

In Red Hat OpenShift, for example:

```
# Delete existing secret
oc delete secret ibm-entitlement-key

# Create new docker-registry secret
oc create secret docker-registry ibm-entitlement-key \
--docker-server=cp.icr.io \
--docker-username=cp \
--docker-password=<entitlement_key> \
--docker-email=<user_email> \
--namespace=<namespace>
```

In Amazon EKS, for example:

```
# Delete existing secret
kubectl delete secret ibm-entitlement-key -n <namespace>

# Create new docker-registry secret
kubectl create secret docker-registry ibm-entitlement-key \
--docker-server=cp.icr.io \
--docker-username=cp \
--docker-password=<entitlement_key> \
--docker-email=<user_email> \
--namespace=<namespace>
```

4. Identify your current IBM MQ Agent Helm release.

List all Helm releases in the current namespace:

```
helm list
```

Note the release name and chart version of your IBM MQ Agent deployment.

5. Update the IBM Helm chart repository to access the latest version of the IBM MQ Agent chart.

Run the following command:

```
helm repo update ibm-helm
```

6. Review your IBM MQ Agent configuration values.

- a) Get your current configuration values.

When you deployed the IBM MQ Agent, you used a file called `overrides.yaml` to override the default configuration values in the Helm chart (see [“Deploying the IBM MQ Agent into your cluster”](#) on page 15). You use this file to upgrade the IBM MQ Agent.

If you no longer have the `overrides.yaml` file, you can recreate it by running the following command and saving the output:

```
helm get values <release_name>
```

where `<release_name>` is the name of your IBM MQ Agent Helm release.

- b) View the default configuration values in the new IBM MQ Agent version by using Helm to download the `values.yaml` file.

```
helm get values <release_name>
```

- c) Compare your `overrides.yaml` file with the `values.yaml` file for the new IBM MQ Agent version, to identify any new configuration values or deprecated settings.

- d) If required, update your `overrides.yaml` file.

For a list of values that you can set, including which values are required, see [“IBM MQ Agent Helm chart values” on page 22](#).

7. Upgrade the IBM MQ Agent by using the **helm upgrade** command.

```
helm upgrade <release_name> ibm-helm/ibm-mq-agent --values overrides.yaml
```

where `<release_name>` is the name of your existing IBM MQ Agent Helm release.

The upgrade process completes the following steps:

- a. Pulls the new container images from the IBM Cloud Container Registry
- b. Stops the existing IBM MQ Agent pod
- c. Creates a new pod with the updated containers

8. Verify that the upgrade was successful.

- a) Check the Helm release status and verify that your IBM MQ Agent release is updated compared to the release that you noted in step [“4” on page 34](#):

```
helm list
```

- b) Watch the pod status to ensure that the new pod starts successfully:

- For Red Hat OpenShift, use the following command.

```
oc get pods -w
```

- For Amazon EKS, use the following command.

```
kubectl get pods -W
```

The STATUS should be Running and the pod name should be followed by 3/3, which indicates that all the deployed containers are running. For example:

NAME	READY	STATUS	RESTARTS	AGE
ibm-mq-agent-mq-agent-79565f867c-p2d5j	3/3	Running	0	1h

Press **Ctrl+C** to stop watching.

9. Optional: If you refreshed your `ibm-entitlement-key` secret (step [“3” on page 34](#)), the upgrade process generates a new API key for the IBM MQ Agent. Update your IBM MQ system as follows:

- a) Update the IBM MQ Console with the new API key, as described in [“Configuring the IBM MQ Console to connect to the IBM MQ Agent” on page 30](#) (either update your IBM MQ Console configuration or share the API key with specific users).

- b) If you have an IBM MQ container deployment, delete any pods that contain the IBM MQ Console configuration. These pods will continue using the old API key so will fail to connect.

For example:

- For Red Hat OpenShift, use the following command.

```
oc delete pod <pod_name>
```

- For Amazon EKS, use the following command.

```
kubectl delete pod <pod_name>
```

What to do next

Test the IBM MQ Agent. Open the IBM MQ Console then access the chat assistant. Ask a simple question such as "What queue managers can you see?" to verify that the IBM MQ Agent responds correctly.

If you encounter problems, try the following steps:

- Review the container logs for error messages. For more information, see [“Troubleshooting the IBM MQ Agent”](#) on page 43.
- If the upgrade fails, you can roll back to the previous version by using the **helm rollback** command:

```
helm rollback <release_name>
```

Uninstalling the IBM MQ Agent

Remove the IBM MQ Agent deployment from your Red Hat OpenShift Container Platform cluster or your Amazon Elastic Kubernetes Service cluster when you no longer need it.

Before you begin

- Ensure that you have access to the Red Hat OpenShift Container Platform cluster or the Amazon EKS cluster where the IBM MQ Agent is deployed.
- Back up any important configuration files.
- Notify users that the IBM MQ Agent will be unavailable and to save any important conversation histories from the chat assistant.

About this task

Uninstalling the IBM MQ Agent removes the Helm release and all associated Kubernetes resources that were created automatically during deployment. You then manually remove objects that you created during deployment, such as ConfigMaps, Secrets, and custom configurations.

After uninstalling the IBM MQ Agent, users can no longer access the chat assistant through the IBM MQ Console.

The uninstall process does not affect your queue managers or their configurations.

Procedure

1. Log in to your cluster by using the **oc** or the **kubectl** command.
For more information, see the documentation for your version of Red Hat OpenShift or Amazon EKS.
2. Switch to the namespace where the IBM MQ Agent is deployed.
For example, run the following command.

On Red Hat OpenShift:

```
oc project mq-agent
```

On Amazon EKS

```
kubectl get namespaces
```

3. Identify your IBM MQ Agent Helm release.

List all Helm releases in the current namespace:

```
helm list
```

Note the release name of your IBM MQ Agent deployment. The release name is the name that you specified when you ran the **helm install** command during deployment.

4. Optional: Review the resources that will be removed.

You can view the Kubernetes resources that are associated with the Helm release by running the following command:

```
helm get manifest <release_name>
```

where **<release_name>** is the name of your IBM MQ Agent Helm release. This command displays the YAML definitions of all resources that Helm will remove.

5. Uninstall the IBM MQ Agent by using the **helm uninstall** command.

Run the following command:

```
helm uninstall <release_name>
```

where **<release_name>** is the name of your IBM MQ Agent Helm release.

The uninstall command removes the following resources that were created by the Helm deployment:

- The IBM MQ Agent pod (including the IBM MQ Agent, embedding model, and MCP server containers)
- The Service for connecting to the IBM MQ Agent
- The Route (if one was created during deployment)
- The Secret that contains your IBM watsonx.ai or vLLM credentials
- The Secret that contains an IBM MQ client INI file, with the keystore password
- The Secret that contains the API key for the IBM MQ Console
- The ServiceAccount for the pod

Helm displays a message confirming that the release has been uninstalled.

6. Verify that the IBM MQ Agent pod was removed.

List the pods in the namespace by running the following command.

On Red Hat OpenShift:

```
oc get pods
```

On Amazon EKS

```
kubectl get pods
```

Verify that the IBM MQ Agent pod (with a name starting with **ibm-mq-agent-**) is no longer listed.

7. Optional: Remove the ConfigMaps and Secrets that you created manually during deployment.

The Helm uninstall command does not remove these objects. If you no longer need these resources, you can remove them manually.

a) Remove the CCDT ConfigMap.

For example, if you created a ConfigMap called **my-ccd**t, run the following command.

On Red Hat OpenShift:

```
oc delete configmap my-ccd
```

On Amazon EKS

```
kubectl delete configmap my-ccd
```

- b) Remove the MCP server configuration Secret.

For example, if you created a Secret called `my-mcp-configuration`, run the following command.

On Red Hat OpenShift:

```
oc delete secret my-mcp-configuration
```

On Amazon EKS

```
kubectl delete secret my-mcp-configuration
```

- c) Remove the keystore Secret.

For example, if you created a Secret called `agent-qm-keystore`, run the following command.

On Red Hat OpenShift:

```
oc delete secret agent-qm-keystore
```

On Amazon EKS

```
kubectl delete secret agent-qm-keystore
```

- d) Remove the TLS Secret.

For example, if you created a Secret called `mq-agent-tls`, run the following command.

On Red Hat OpenShift:

```
oc delete secret mq-agent-tls
```

On Amazon EKS

```
kubectl delete secret mq-agent-tls
```

- e) Remove the IBM entitlement key Secret.

If you created the `ibm-entitlement-key` secret specifically for the IBM MQ Agent and you do not need it for other deployments, you can remove it. Only remove this secret if you are sure that no other deployments in the namespace require it. Run the following command.

On Red Hat OpenShift:

```
oc delete secret ibm-entitlement-key
```

On Amazon EKS

```
kubectl delete secret ibm-entitlement-key
```

8. Optional: Remove the IBM MQ Agent configuration from your IBM MQ Console.

If you configured your IBM MQ Console to connect to the IBM MQ Agent, remove or comment out the agent configuration from the `mqwebuser.xml` file. For more information about the configuration, see [“Configuring the IBM MQ Console to connect to the IBM MQ Agent” on page 30](#).

9. Optional: Remove the IBM MQ Agent namespace (project).

If you created a dedicated namespace for the IBM MQ Agent and you no longer need it, you can delete the entire namespace. Deleting a namespace removes all resources within it; ensure that the namespace does not contain any other important deployments before you delete it. To delete the namespace, run the following command, where `<namespace_name>` is the name of the namespace.

On Red Hat OpenShift:

```
oc delete project <namespace_name>
```

On Amazon EKS

```
kubectl delete namespace <namespace_name>
```

Results

The IBM MQ Agent is removed from your cluster. The following resources are deleted:

- The IBM MQ Agent Helm release and all associated Kubernetes resources
- The IBM MQ Agent pod and containers
- ConfigMaps and Secrets that you deleted manually

Your queue managers and their configurations remain unchanged (the IBM MQ Agent does not modify any queue manager settings).

What to do next

- If you created an IBM watsonx.ai project and no longer need the project or API keys, you can manage them through your IBM Cloud account.
- If you want to revoke the access that you granted for the MCP server to work with your queue manager resources, run the appropriate IBM MQ security commands. For more information, see [“Granting the IBM MQ Agent access to queue manager resources”](#) on page 14.

Interacting with the IBM MQ Agent

After you install and configure the IBM MQ Agent, you can open the chat assistant and start asking questions.

Before you begin

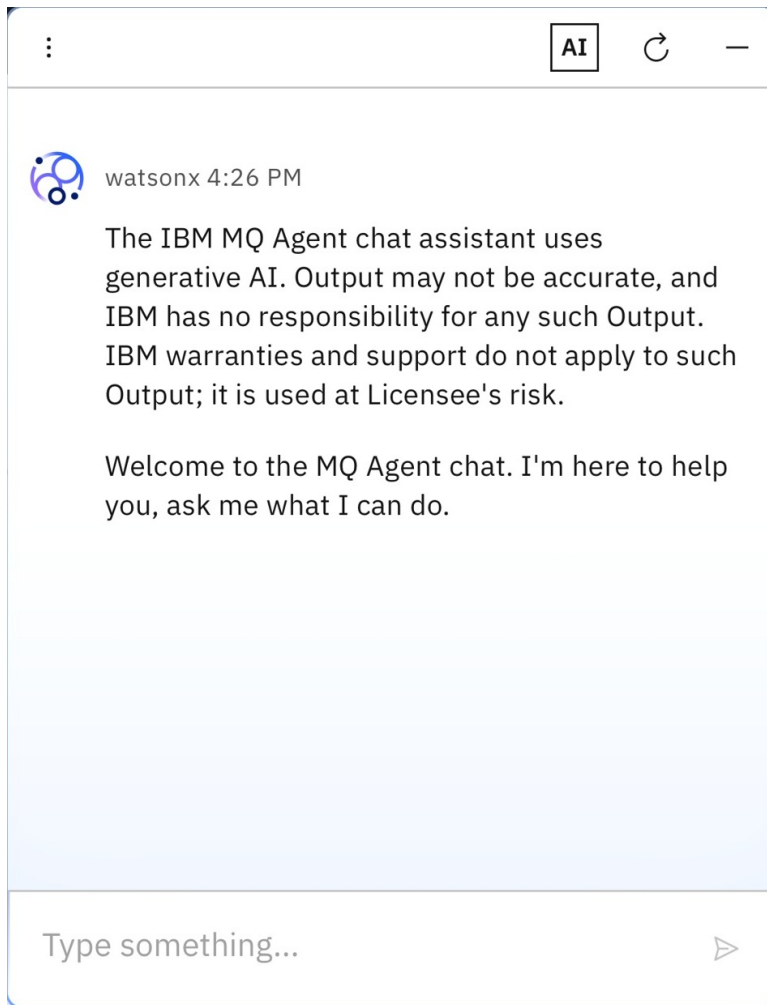
- Complete the steps to install and configure the IBM MQ Agent, as described in [“Installing and configuring the IBM MQ Agent”](#) on page 11.
- Ensure that the status of the IBM MQ Agent pod is Running. For example, run the following command:

```
oc get pods
```

Check that the IBM MQ Agent pod has a status of Running. The pod name is `ibm-mq-agent-<release_name>`, where *<release_name>* is the name that you chose when you ran the **helm** command to deploy the IBM MQ Agent.

Procedure

1. In your IBM MQ Console, open the chat assistant by clicking the **AI (Open the chat window)** icon in the bottom right-hand corner of the console page.



2. Ask questions in the chat assistant.

You can ask the agent about its capabilities at any time.

You 4:31 PM

Hi, tell me about your capabilities



watsonx 4:31 PM

I can help you with several IBM MQ tasks:

- Query queue managers - Check which queue managers are available and their status
- Diagnose message build-up - Investigate why messages are accumulating on queues
- Check queue parameters - Verify settings like MQGET/MQPUT enablement, depth limits, etc.
- Investigate channels and listeners - Check status and diagnose connection issues
- Explain MQ concepts - Answer questions about error codes, configuration, and best practices

For example, you could ask: 'Is queue manager QM1 running?' or 'Why are messages building up on queue Q1?'

For other types of question and problem determination that the IBM MQ Agent can help you with, see the use cases and examples in “IBM MQ Agent” on page 5. You might want to start with “Which queue managers are you connected to?”, which you can use to check that your client channel definition table is correct and that the IBM MQ Agent can connect to your queue managers.

Follow these guidelines when you ask questions:

- If you ask about an object, provide the object type as well.
- If you want the output to include system queues, ask for them explicitly; the IBM MQ Agent omits them by default. For example: "List all my queues on queue manager qm1 including system queues". For a list of IBM MQ system queues, see [System and default objects](#).
- Focus on one conversation at a time, otherwise the IBM MQ Agent might ask you clarification questions.
- If a response to your question results in a large volume of data, the large language model (LLM) might summarize the information. If you suspect that a response is missing information, repeat your question but ask for a full answer.

Note: The IBM MQ Agent cannot run commands and is not intended as a general purpose AI chat interface.

The IBM MQ Agent provides regular status updates as it processes your questions by working through its planning, information gathering, and reasoning steps. For example, you might see status messages similar to the following:

- Analyzing query
- Planning

- Working [step 1 of 1] -> Checking queue status

One IBM MQ Agent can support multiple chat assistants. The maximum tested is 50. For example, 35 users can each have a chat assistant that interacts with the same IBM MQ Agent at the same time. Each chat assistant uses the same IBM MQ Agent credentials.

3. Optional: If you want to restart a conversation, click the **Restart Conversation** button.

You might want to restart a conversation when you start analyzing a new problem or if the output starts to deviate from your expectations. Deviation can occur with long conversations because context can leak across questions, particularly when you are working across different queue managers and objects with duplicate names. Similarly, in longer running conversations the chat assistant might shift to focus on more recent context.

4. Optional: If you want to save your conversation, click the three dot menu icon at the top of the chat window then click **Save conversation**.

You might want to save conversations for use in internal support tickets, for example, or for deeper analysis of problems.

The conversation history is saved as a JSON file to your local machine, according to your web browser preferences. Here is an example of a saved conversation:

```
message: "how should I fix this problem?"
24:
sender: "Agent"
message: "To fix the problem with queue qm4 on queue manager
QM1, you'll need to address the lack of message
consumers. Here's what I found:\n\nThe queue is completely
full (5000/5000 messages) because:\n- There are no
applications consuming messages from it (IPPROCS = 0)\n-
Messages have been accumulating for 29 days without being
processed\n- While the queue is enabled for getting messages,
nothing is actually retrieving them\n\nTo resolve this:\n1.
You'll need to either:\n - Start an application that should
be consuming messages from this queue, or\n - Clear out the
existing messages if they're no longer needed\n\n2. For
immediate relief, you could:\n - Increase the queue's
maximum depth temporarily\n - Manually clear some messages
to prevent further buildup\n\nThe root cause appears to be
that whatever application should be processing these messages
isn't running or isn't properly connected to this queue."
```

Example

Example questions to ask the IBM MQ Agent:

General and documentation lookup

- What can you do?
- Tell me about your capabilities
- What is an MQ listener?
- What is a transmission queue?
- How do I authorise an application to get messages from queue <queue name>?
- Show me the MQSC to create a queue called myQueue
- How should I contact IBM support for MQ problems

Queue manager estate and environment questions

- List all of my queue managers
- What is the version of <queue manager name>?
- What platforms are my queue managers running on?
- Is a DLQ set for <queue manager name>?

Message build-up investigations

- Do I have any messages building up*

- List the top <n> deepest queues on <queue manager name>
- List the queues on <queue manager name> which have messages from over <unit of time> ago

Channel related questions

- List my channels on <queue manager name>
- Is channel <channel name> running?*
- Do I have any channel problems on <queue manager name>?
- Do I have any channels in retry on <queue manager name>?
- Why is channel <channel name> in state <channel state>?

Troubleshooting the IBM MQ Agent

If you encounter problems when using the IBM MQ Agent chat assistant, you can retry your query or restart the conversation. You can also review log files to help debug issues.

Finding the details of your installation

You installed the IBM MQ Agent by using Helm. In Helm, an installation of a chart that is running in a Kubernetes cluster is called a *release*. You can list the installed releases by logging onto your cluster then running the following command:

```
helm list
```

After you identify the name of the release for the IBM MQ Agent, you can identify the name of the Kubernetes Deployment, which takes the form `ibm-mq-agent-RELEASENAME`. For example, `ibm-mq-agent-mymqagent` for a Helm release called `mymqagent`.

Viewing log files

The components of the IBM MQ Agent have their own log files.

IBM MQ Agent MCP server logs

The IBM MQ Agent contains an Model Context Protocol (MCP) server that uses an IBM MQ client with a client channel definition table (CCDT) to connect to your queue managers. If you experience issues with connectivity to your queue managers, check the MCP server logs. The MCP server runs in a separate container. To access the logs, first obtain the pod name by running the following command.

- On Red Hat OpenShift:

```
oc get pods
```

- On Amazon EKS:

```
kubectl get pods
```

You should see output similar to the following:

NAME	READY	STATUS	RESTARTS	AGE
ibm-mq-agent-mq-agent-79565f867c-p2d5j	2/2	Running	0	1h

The pod name starts with `ibm-mq-agent-<release_name>`, where `<release_name>` is the name that you chose when you ran the **helm** command to deploy the IBM MQ Agent. Use the pod name in the following command to show the logs.

- On Red Hat OpenShift:

```
oc logs <pod_name> --container mcp-server
```

- On Amazon EKS:

```
kubectl logs <pod_name> --container mcp-server
```

IBM MQ Agent logs

The IBM MQ Agent invokes tools in its MCP server and uses the large language model (LLM) to respond to queries. The agent runs in its own container. View the logs in the same way as for the MCP server; the container name is `agent`.

Embedding model container logs

The IBM MQ Agent contains an embedding LLM model, which runs in its own container. View the logs in the same way as for the MCP server; the container name is `embedding-service`. For information about the LLMs that the IBM MQ Agent uses, see [“Foundation models for the IBM MQ Agent”](#) on page 47.

IBM MQ Console logs

The IBM MQ Agent communicates with the IBM MQ Console, through which you ask your questions. The IBM MQ Console log files contain messages that were issued by the `mqweb` server, capturing traffic on both sides of the server (between the IBM MQ Console user interface and the `mqweb` server, and between the `mqweb` server and the IBM MQ Agent).

The default location of the log files is listed in [Troubleshooting IBM MQ Console and REST API problems](#).

Collecting diagnostic and trace information for IBM support

For general information about troubleshooting your IBM MQ installation, see [IBM MQ troubleshooting and support](#).

If you need to raise a case with IBM support regarding the IBM MQ Agent, provide troubleshooting information to help diagnose the issue. If possible, use the [IBM MQ container inspector](#).

The IBM MQ Agent includes some additional diagnostic features to assist the IBM support team. These diagnostic features are enabled by default when you deploy the IBM MQ Agent (see [“Deploying the IBM MQ Agent into your cluster”](#) on page 15).

Typically, the first sign of a problem is unusual or error responses in the chat assistant. If the IBM MQ Console can still connect to the IBM MQ Agent and if diagnostic features are enabled, you can enter the following commands in the chat assistant to generate diagnostic traces and dump files to help with troubleshooting:

/diag [<KEY>]

Start a diagnostic session. The optional KEY parameter is a key called `ibm-mq-agent-<release_name>-diag`, where `<release_name>` is the name that you chose when you ran the **helm** command to deploy the IBM MQ Agent. The key was created (either by you or by the IBM MQ Agent) during the deployment process. For more information, see [“Deploying the IBM MQ Agent into your cluster”](#) on page 15.

/dump

Write a `.dmp` file that contains the internal IBM MQ Agent state.

/trace_start

Start writing a runtime trace (`.trc`) file that records your interactions and LLM model calls.

/trace_stop

Stop trace collection and close the trace file.

/end

Close the diagnostic session.

Dump and trace files are written to the `/opt/run` directory in the IBM MQ Agent container. Copy the files from the IBM MQ Agent container as follows:

1. Open a remote shell into the IBM MQ Agent container to inspect the dump and trace files. For example:

```
oc rsh --container agent <your_pod_id>
```

2. Change the directory to /opt/run:

```
cd /opt/run
```

3. List the files. The most recent file is displayed at the end of the list.

List trace files:

```
ls -alrt *.trc
```

List dump files:

```
ls -alrt *.dmp
```

4. Exit the container:

```
exit
```

5. Output the contents of a file into another file in a suitable directory.

For example, for trace files:

```
oc exec <your_pod_id> --container agent -- cat /opt/run/<trace_file_name>.trc > <trace_file_name>.trc
```

For dump files:

```
oc exec <your_pod_id> --container agent -- cat /opt/run/<dump_file_name>.dmp > <dump_file_name>.dmp
```

Collecting IBM MQ Agent troubleshooting information with IBM MQ container inspector

The IBM MQ container inspector tool collects troubleshooting information about your IBM MQ Agent deployment. You can provide this information to IBM if you open a support case.

Before you begin

Check that you meet the following prerequisites:

1. You are logged into the cluster that hosts your IBM MQ Agent deployment.
2. You have permission to access the target namespace.

The commands in this topic use the following variables:

- *<agent_release_name>*: the name of the IBM MQ Agent Helm release. You can list your installed releases by running the following command:

```
helm list
```

- *<agent_namespace>*: the namespace where the IBM MQ Agent is deployed
- *<output_directory>*: the directory to which the IBM MQ container inspector tool writes the collected information

About this task

You can view the IBM MQ container inspector source code and license in the [mq-container-inspector](#) GitHub repository.

You can run the IBM MQ container inspector in the following ways, which are described in this topic:

- As a Go binary file that you run locally
- As an image that you run in Podman or Docker
- As an image that you run in a pod on Red Hat OpenShift Container Platform

Procedure

Running the IBM MQ container inspector Go binary file

1. Run the IBM MQ container inspector binary file directly from your machine to collect troubleshooting information.
 - a) Download or build the IBM MQ container inspector tool by following the instructions in the [mq-container-inspector](#) GitHub repository.
 - b) Run the tool by entering the following command, substituting your own IBM MQ Agent release name, namespace, and the output directory.

```
mq-container-inspector mq-agent --agent-release-name <agent_release_name> --namespace <agent_namespace> --output-dir <output_directory>
```

The tool collects diagnostic data including logs, configuration, and deployment information. Provide the generated MQ-Agent_MustGather_<timestamp>.tar.gz file to IBM support when you raise your case.

Running IBM MQ container inspector as an image in Podman or Docker

2. IBM MQ container inspector is included in the IBM Cloud Pak foundational services MustGather image ([icr.io/cpopen/cpfs/must-gather](#)). If you have Podman or Docker running on your machine, you can use this image to collect IBM MQ Agent troubleshooting information.
 - a) Create a directory to send the troubleshooting information to.

```
mkdir -p <output_directory>
chmod 775 <output_directory>
```

- b) Run the following command (if you are using Docker, replace podman with docker), substituting your own IBM MQ Agent release name and namespace, and the output directory that you created in the previous step.

```
podman run -v ~/.kube/config:/.kube/config -v $(pwd)/<output_directory>:/tmp/must-gather \
icr.io/cpopen/cpfs/must-gather:latest gather -m mq -- mq-agent --agent-release-name <agent_release_name> --namespace <agent_namespace>
```

This command runs the IBM MQ container inspector **mq-agent** tool from your Podman or Docker agent by using the `~/.kube/config` file to connect to your cluster. The output from the tool is written to the output directory that you created.

Running IBM MQ container inspector in a pod in Red Hat OpenShift Container Platform

3. IBM MQ container inspector is included in the IBM Cloud Pak foundational services MustGather image ([icr.io/cpopen/cpfs/must-gather](#)). You can run this image by using the OpenShift **oc adm must-gather** command, as described in [Gathering data about your cluster](#) in the Red Hat documentation. This command is available only on OpenShift and requires admin permissions for the OpenShift cluster.
 - a) Run the **oc adm must-gather** command, substituting the output directory and your own IBM MQ Agent release name and namespace.

```
oc adm must-gather --dest-dir=<output_directory> --image=icr.io/cpopen/cpfs/must-gather:latest -- gather -m mq -- mq-agent --agent-release-name <agent_release_name> --namespace <agent_namespace>
```

- b) Verify the output.

The IBM MQ container inspector tool sends output to `<output_directory>/icr-io-cpopen-cpfs-must-gather-<image_sha>/cloudpak-must-gather-timestamp.tar.gz`. Extract the package then verify the contents of the MQ-Agent_MustGather_<timestamp>

directory before you submit the MQ-Agent_Must_Gather_<timestamp>.tar.gz file to IBM Support. If this file is missing, check the gather.log file (created by the oc adm must-gather command) for an error message that explains why the data gather operation was unsuccessful.

Foundation models for the IBM MQ Agent

The IBM MQ Agent feature uses *large language models* (LLMs) to understand your questions. LLMs are a type of *AI foundation model* that is trained on text. An *inference runtime* runs the foundation models to produce results. Each version of the IBM MQ Agent feature uses a specific set of foundation models. Only some inference runtimes are supported.

IBM MQ Advanced does not include an inference runtime; you must separately obtain access to, and an entitlement for, a runtime.

The following lists might change over time as new foundation models and runtimes are developed.

Supported inference runtimes

The following inference runtimes are supported for the deployment of the AI foundation models that are required by the IBM MQ Agent feature.

- [IBM watsonx.ai Runtime as a Service](#)
- [vLLM](#)

For set up instructions, see [“Preparing an AI runtime for the IBM MQ Agent”](#) on page 11.

Foundation models used by the IBM MQ Agent feature

The IBM MQ Agent uses multiple AI foundation models, categorized as follows:

Primary

The main reasoning model. This model is required.

Secondary

A more lightweight model that the IBM MQ Agent uses for simple tasks. Using a secondary model can reduce cost and latency. You can disable the secondary model if you want to use only the primary model. For example, if you are running on premises with a single GPU and you don't want to purchase another GPU to run a secondary model.

The IBM MQ Agent also uses a vector embedding model for documentation retrieval and semantic search. This model is included as part of the IBM MQ Agent.

The following table lists the foundation models that are required by each release of the IBM MQ Agent feature. The availability of models might vary, for example between IBM watsonx.ai data centers or because models were deprecated. If models are deprecated, you must update the IBM MQ Agent so that it can continue to function with the latest available models. Before you update to a newer release of the IBM MQ Agent feature, ensure that the required models are available in your inference runtime. For information about which models are deprecated in IBM watsonx.ai, see [Supported foundation models in watsonx.ai](#) in the IBM watsonx documentation.

Table 11. The foundation models that are required by each version of the IBM MQ Agent feature	
IBM MQ Agent feature version	Required foundation models
2.0.0	Primary openai/gpt-oss-120b Secondary ibm/granite-4-h-small

Security for the IBM MQ Agent

You configure the IBM MQ Agent with read-only access to IBM MQ queue managers so that you can use the agent to view configuration details, observe real-time object state, and complete diagnostics fault-finding tasks. You can encrypt the various communication channels involved by using TLS.

IBM MQ Agent container communication

The IBM MQ Agent is delivered as multiple container processes:

- An MCP server, which communicates with IBM MQ queue managers by using standard IBM MQ administrative APIs
- An IBM MQ Agent container, which communicates with the following components: the MCP server container instance, the IBM MQ Console (which you use to ask questions), and an AI inference runtime that runs large language models (LLMs) to understand your questions
- An embedding model container, which communicates with the IBM MQ Agent container

These processes communicate with each other over a private network within the same pod.

The IBM MQ Agent maintains the following outbound connections:

- Connections to queue managers by using a CCDT, which can be secured using a TLS network connection
- Connections to an inference runtime, which can be secured using a TLS network connection
- Responding to user queries originating from IBM MQ Console chat sessions connected to the IBM MQ Agent runtime.

Other network considerations depend on your configuration, and might include the following items:

- DNS: if you have channels or other configuration that uses DNS names
- Online Certificate Status Protocol (OCSP) and Certificate Revocation Lists (CRLs): determined by your certificate provider

For more information about restricting network traffic in a containerized environment, see [Considerations for restricting network traffic to IBM MQ in containers](#) in the *IBM MQ in containers and IBM Cloud Pak for Integration* documentation

IBM MQ Agent access to queue manager data

You control the scope of the IBM MQ Agent's access to a queue manager configuration and state by using IBM MQ standard authentication and authorization mechanisms. For more information about how to set up access for the IBM MQ Agent, see [“Granting the IBM MQ Agent access to queue manager resources”](#) on page 14.

After you deploy and configure the IBM MQ Agent according to the instructions, it has read-only access to the following types of information.

IBM MQ queue manager information

General information about queue managers that you configured the IBM MQ Agent to work with. For example:

- The version and fix pack of the queue manager's IBM MQ installation
- The operating system on which the queue manager is running. For example, UNIX
- General queue manager configuration details
- Cluster configuration details

IBM MQ object information

Information about IBM MQ objects that belong to queue managers that you configured the IBM MQ Agent to work with. For example:

- Queues
- Channels

- Listeners

Application names and connection information

Names and connection information of applications that exchange messages with the queue managers that you configured the IBM MQ Agent to work with.

Queue manager error logs

Error logs for the queue managers that you configured the IBM MQ Agent to work with.

Messages

Message header information from messages on queues to which you granted the MCP server appropriate browse authority. When the IBM MQ Agent detects a message header, it can use the following header fields to support your investigations:

- Dead letter header: to determine reasons why a message was sent to the dead letter queue
- Transmit queue header: to determine routing information related to a message on transmission queues
- Messages header information: to support the diagnosis of message build-up. For example:
 - Timestamps
 - Backout count
 - Message size
 - Routing and destination information

IBM MQ Agent communication with AI runtimes

During normal operation, the IBM MQ Agent processes your requests to help examine and interpret the real-time state of a queue manager. The IBM MQ Agent combines your prompts with queue manager state and internal prompts, then forwards the information to an AI inference runtime to apply reasoning. Inference runtime providers have their own data privacy and GDPR processes. For example, for IBM watsonx.ai, see [Security policies and responsibilities in IBM Cloud](#). For vLLM, see [Security](#) in the vLLM documentation.

IBM MQ Agent logs

At runtime, the IBM MQ Agent writes high-level log information to the container stdout stream. The log files record HTTP request and response codes that relate to calls routed through the MCP server to queue manager administrative interfaces. The logs also record token usage and errors that relate to the IBM MQ Agent's internal processing.

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Software Interoperability Coordinator, Department 49XA
3605 Highway 52 N
Rochester, MN 55901
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Programming interface information

Programming interface information, if provided, is intended to help you create application software for use with this program.

This book contains information on intended programming interfaces that allow the customer to write programs to obtain the services of IBM MQ.

However, this information may also contain diagnosis, modification, and tuning information. Diagnosis, modification and tuning information is provided to help you debug your application software.

Important: Do not use this diagnosis, modification, and tuning information as a programming interface because it is subject to change.

Trademarks

IBM, the IBM logo, ibm.com[®], are trademarks of IBM Corporation, registered in many jurisdictions worldwide. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" www.ibm.com/legal/copytrade.shtml. Other product and service names might be trademarks of IBM or other companies.

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

This product includes software developed by the Eclipse Project (<https://www.eclipse.org/>).

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.



Part Number:

(1P) P/N: