9.2

容器中的 *IBM MQ*

注

在使用本资料及其支持的产品之前，请阅读第 143 页的『声明』中的信息。

本版本适用于 IBM® MQ V 9 发行版 2 以及所有后续发行版和修订版，直到在新版本中另有声明为止。

当您向 IBM 发送信息时，授予 IBM 以它认为适当的任何方式使用或分发信息的非独占权利，而无须对您承担任何责任。

© Copyright International Business Machines Corporation 2007, 2025.

内容

容器和 IBM Cloud Pak for Integration 中的 IBM MQ.....5
规划容器中的 IBM MQ.....5
选择要在容器中使用 IBM MQ 的方式.....5
对 IBM MQ Operator 的支持.....6
IBM MQ Operator 的依赖关系.....9
IBM MQ Operator 所需的集群范围的许可权.....10
IBM MQ Operator 的存储注意事项.....10
支持构建您自己的 IBM MQ 队列管理器容器映像.....12
容器中 IBM MQ 的高可用性.....14
容器中 IBM MQ 的灾难恢复.....16
容器中 IBM MQ 的用户认证和授权.....16
规划容器中 IBM MQ 的可伸缩性和性能.....17
在 IBM Cloud Pak for Integration 和 Red Hat OpenShift 中使用 IBM MQ.....17
IBM MQ Operator 的发布历史记录.....18
将 IBM MQ 迁移到 IBM Cloud Pak for Integration.....33
在 Red Hat OpenShift 上安装和卸载 IBM MQ Operator.....55
升级 IBM MQ Operator 和队列管理器.....66
使用 IBM MQ Operator 部署和配置队列管理器.....73
使用 IBM MQ Operator 操作 IBM MQ.....104
对 IBM MQ Operator 的问题进行故障诊断.....113
IBM MQ Operator 的 API 参考.....114
构建您自己的 IBM MQ 容器和部署代码.....132
使用容器规划您自己的 IBM MQ 队列管理器映像.....132
构建样本 IBM MQ 队列管理器容器映像.....133
在单独的容器中运行本地绑定应用程序.....135
创建本机 HA 组 (如果创建您自己的容器).....137

声明.....143
编程接口信息.....144
商标.....144

Multi 容器和 IBM Cloud Pak for Integration 中的 IBM MQ

容器允许您将 IBM MQ 队列管理器或 IBM MQ 客户机应用程序及其所有依赖项打包到标准化单元中以进行软件开发。

您可以使用 Red Hat® OpenShift®上的 IBM MQ Operator 运行 IBM MQ 。 可以使用 IBM Cloud Pak for Integration， IBM MQ Advanced 或 IBM MQ Advanced for Developers 来完成此操作。

您还可以在自己构建的容器中运行 IBM MQ 。

CD MQ Adv. 有关 IBM MQ Operator 的更多信息，请参阅以下链接。

Multi 规划容器中的 IBM MQ

在容器中规划 IBM MQ 时，请考虑 IBM MQ 为各种体系结构选项提供的支持，例如如何管理高可用性以及如何保护队列管理器。

关于此任务

在规划容器体系结构中的 IBM MQ 之前，您应该熟悉基本 IBM MQ 概念 (请参阅 IBM MQ 技术概述) 以及基本 Kubernetes/Red Hat OpenShift 概念 (请参阅 Red Hat OpenShift Container Platform 体系结构)。

过程

- 第 5 页的『选择要在容器中使用 IBM MQ 的方式』.
- 第 6 页的『对 IBM MQ Operator 的支持』.
- 第 12 页的『支持构建您自己的 IBM MQ 队列管理器容器映像』.
- 第 10 页的『IBM MQ Operator 的存储注意事项』.
- 第 14 页的『容器中 IBM MQ 的高可用性』.
- 第 16 页的『容器中 IBM MQ 的灾难恢复』.
- 第 16 页的『容器中 IBM MQ 的用户认证和授权』.

选择要在容器中使用 IBM MQ 的方式

在容器中使用 IBM MQ 有多个选项: 您可以选择使用 IBM MQ Operator(使用预先打包的容器映像) ， 也可以构建自己的映像和部署代码。

使用 IBM MQ Operator

OpenShift CP4I

如果计划在 Red Hat OpenShift Container Platform 上部署，那么可能要使用 IBM MQ Operator。

IBM MQ Operator 向 Red Hat OpenShift Container Platform 添加新的 `QueueManager` 定制资源。 操作程序会监视新的队列管理器定义，然后将它们转换为必需的低级别资源，例如 `StatefulSet` 和 `Service` 资源。 对于本机 HA ，操作程序还可以执行队列管理器实例的复杂滚动更新。 请参阅 第 138 页的『执行本机 HA 队列管理器的滚动更新的注意事项』

使用 IBM MQ Operator 时，不支持某些 IBM MQ 功能部件。 如果要执行以下任何操作，您将需要构建自己的图像和图表:

- 使用 REST API 进行管理或消息传递
- 使用以下任何 MQ 组件:
  - Managed File Transfer 代理程序及其资源。 但是，您可以使用 IBM MQ Operator 来提供一个或多个协调队列管理器，命令队列管理器或代理队列管理器。

  - AMQP
  - IBM MQ Bridge to Salesforce
  - IBM MQ Bridge to blockchain (在容器中不受支持)
  - IBM MQ Telemetry Transport (MQTT).
- 定制用于 **crtmqm**， **strmqm** 和 **endmqm** 的选项，例如配置日志文件页面。 可以使用 INI 文件配置大多数选项。

请注意， IBM MQ Operator 和容器正在快速演变，因此在 Long Term Support 发行版下不受支持。

IBM MQ Operator 包含预先构建的容器映像以及用于在 Red Hat OpenShift Container Platform 上运行的部署代码。 IBM MQ Operator 可用于部署所提供的 IBM MQ 容器映像或在其基础上分层的容器映像，但不能用于部署定制构建的 MQ 容器映像。

构建您自己的映像和部署代码

Multi

这是最灵活的容器解决方案，但这需要您具备配置容器的强大技能，并 “拥有” 生成的容器。 如果您不打算使用 Red Hat OpenShift Container Platform，那么将需要构建自己的映像和部署代码。

提供了用于构建您自己的映像的样本。 请参阅 第 132 页的『构建您自己的 IBM MQ 容器和部署代码』。

相关概念

第 6 页的『对 IBM MQ Operator 的支持』
仅当在 Red Hat OpenShift Container Platform 上部署时，才支持 IBM MQ Operator 。

第 12 页的『支持构建您自己的 IBM MQ 队列管理器容器映像』
IBM MQ 提供了用于在 GitHub 上构建 IBM MQ 队列管理器容器的代码。 这是基于 IBM 用于构建其自己的受支持容器的过程，您可以使用此 GitHub 存储库来简化和加速构建自己的容器映像。

OpenShift CD CP4I EUS 对 IBM MQ Operator 的支持

仅当在 Red Hat OpenShift Container Platform 上部署时，才支持 IBM MQ Operator 。

IBM MQ Operator 使用基于 IBM MQ Continuous Delivery (CD) 发行版的映像，但 IBM Cloud Pak for Integration 提供了扩展更新支持 (EUS) 发行版。 支持 CD 发行版最多一年，或者支持两个 CD 发行版，以更长的时间为准。 无法通过 IBM MQ Operator 获取 IBM MQ 的 Long Term Support 发行版。 IBM Cloud Pak for Integration 2020.4.1 是扩展更新支持 (EUS) 发行版，如果使用标记为 -eus 的 IBM MQ 版本，那么支持 18 个月。 否则， IBM MQ 9.2 将被视为具有 IBM MQ Operator 的 Continuous Delivery 发行版。

IBM MQ Operator 使用在 Red Hat Universal Base Image (UBI) 上提供 IBM MQ 安装的容器映像，其中包括 IBM MQ 使用的关键 Linux® 库和实用程序。 在 Red Hat OpenShift 上运行时， Red Hat 支持 UBI。

IBM MQ Operator 在 amd64 和 s390x (z/Linux) 体系结构上受支持。

相关概念

第 12 页的『支持构建您自己的 IBM MQ 队列管理器容器映像』
IBM MQ 提供了用于在 GitHub 上构建 IBM MQ 队列管理器容器的代码。 这是基于 IBM 用于构建其自己的受支持容器的过程，您可以使用此 GitHub 存储库来简化和加速构建自己的容器映像。

OpenShift CD CP4I EUS IBM MQ Operator 的版本支持

IBM MQ， Red Hat OpenShift Container Platform 和 IBM Cloud Pak for Integration 的受支持版本之间的映射。

- 第 7 页的『可用的 IBM MQ 版本』
- 第 7 页的『兼容的 Red Hat OpenShift Container Platform 版本』
- 第 7 页的『IBM Cloud Pak for Integration 版本』
- 第 8 页的『较旧的操作程序中的可用 IBM MQ 版本』
- 第 8 页的『适用于较旧的操作程序的兼容 Red Hat OpenShift Container Platform 版本』

可用的 IBM MQ 版本

操作员通道	操作程序版本	IBM MQ 版本							
		9.1.5	9.2.0 CD	9.2.0 EUS	9.2.1	9.2.2	9.2.3	9.2.4	9.2.5
v1.6	1.6	⚠	⚠	→	⚠	●	●		
v1.7	1.7	⚠	⚠	→	⚠	●	●	●	
v1.8	1.8	⚠	⚠	→	⚠	⚠	●	●	●

Key:

●
Continuous Delivery 支持可用

▯
Extended Update Support 可用

→
仅在从 Extended Update Support 操作数迁移到 Continuous Delivery 操作数期间可用。

⚠
不推荐。 由于 IBM MQ 发行版不支持，因此它们可能仍在操作程序中可配置，但不再适合支持，并且可能会在将来的发行版中除去。

请参阅 第 18 页的『IBM MQ Operator 的发布历史记录』 以获取每个版本的完整详细信息，包括每个版本中的详细功能，更改和修订。

兼容的 Red Hat OpenShift Container Platform 版本

操作员通道	操作程序版本	Red Hat OpenShift Container Platform 版本 [1]				
		4.6	4.7[2]	4.8	4.9	4.10
v1.6	1.6	●	●	●	●	●
v1.7	1.7	●	●	●	●	●
v1.8	1.8	●	●	●	●	●

Key:

●
Continuous Delivery 支持可用

▯
Extended Update Support 可用

IBM Cloud Pak for Integration 版本

支持将 IBM MQ Operator 1.8.x 用作 IBM Cloud Pak for Integration V 2021.4.1 的一部分或单独使用。

支持将 IBM MQ Operator 1.7.x 用作 IBM Cloud Pak for Integration V 2021.4.1 的一部分或单独使用。

支持将 IBM MQ Operator 1.6.x 用作 IBM Cloud Pak for Integration V 2021.2.1， 2021.3.1 的一部分或单独使用。

不再支持 IBM MQ Operator 1.5.x 。

---

[1] Red Hat OpenShift Container Platform 版本受其自己的支持日期限制。 请参阅 Red Hat OpenShift Container Platform 生命周期策略 以获取更多信息。

[2] IBM MQ Operator 依赖于 IBM Cloud Pak foundational services。 如果要使用 Red Hat OpenShift Container Platform 4.7，应首先升级 IBM Cloud Pak foundational services 的版本。

不再支持 IBM MQ Operator 1.4.x 。

不再支持 IBM MQ Operator 1.3.x 。

不再支持 IBM MQ Operator 1.2.x 。

不再支持 IBM MQ Operators 1.1.x 和 1.0.x 。

较旧的操作程序中的可用 IBM MQ 版本

下表适用于现在已达到 "生命周期结束" 的 IBM MQ Operator 版本。

操作员通道	操作程序版本	IBM MQ 版本							
		9.1.5	9.2.0 CD	9.2.0 EUS	9.2.1	9.2.2	9.2.3	9.2.4	9.2.5
v1.0	1.0	⚠							
v1.1	1.1	⚠	⚠						
v1.2	1.2	⚠	⚠						
v1.3-eus	1.3	⚠	⚠	⚠					
v1.4	1.4	⚠	⚠	→	⚠				
v1.5	1.5	⚠	⚠	→	⚠	⚠			

Key:

→
仅在从 Extended Update Support 操作数迁移到 Continuous Delivery 操作数期间可用。

⚠
不推荐。 由于 IBM MQ 发行版不支持，因此它们可能仍可在 IBM MQ Operator 中进行配置，但不再符合支持条件。

请参阅 第 18 页的『IBM MQ Operator 的发布历史记录』 以获取每个版本的完整详细信息，包括每个版本中的详细功能，更改和修订。

适用于较旧的操作程序的兼容 Red Hat OpenShift Container Platform 版本

下表适用于现在已达到 "生命周期结束" 的 IBM MQ Operator 版本。

操作员通道	操作程序版本	Red Hat OpenShift Container Platform 版本 [3]						
		4.4[4]	4.5[5]	4.6	4.7[6]	4.8	4.9	4.10
v1.0	1.0	⚠	⚠	⚠	⚠			
v1.1	1.1	⚠	⚠	⚠	⚠	⚠		
v1.2	1.2	⚠	⚠	⚠	⚠	⚠		

[3] Red Hat OpenShift Container Platform 版本受其自己的支持日期限制。 请参阅 Red Hat OpenShift Container Platform 生命周期策略 以获取更多信息。

[4] Red Hat OpenShift Container Platform 4.4 已达到 "生命周期结束"。 请参阅 Red Hat OpenShift Container Platform 生命周期策略 以获取更多信息。

[5] Red Hat OpenShift Container Platform 4.5 已达到 "生命周期结束"。 请参阅 Red Hat OpenShift Container Platform 生命周期策略 以获取更多信息。

[6] IBM MQ Operator 依赖于 IBM Cloud Pak foundational services。 如果要使用 Red Hat OpenShift Container Platform 4.7，应首先升级 IBM Cloud Pak foundational services 的版本。

操作员通道	操作程序版本	Red Hat OpenShift Container Platform 版本 [3]						
		4.4[4]	4.5[5]	4.6	4.7[6]	4.8	4.9	4.10
v1.3-eus	1.3			⚠	→	→	→	→
v1.4	1.4			⚠	⚠	⚠	⚠	
v1.5	1.5			⚠	⚠	⚠	⚠	⚠

Key:

→
仅在从 Extended Update Support 操作数迁移到 Continuous Delivery 操作数期间可用。

⚠
IBM MQ Operator 版本已达到 "生命周期结束"，但先前在此版本的 Red Hat OpenShift Container Platform 上可用

OpenShift CP4I IBM MQ Operator 的依赖关系

IBM MQ Operator 依赖于 IBM Cloud Pak foundational services 操作程序，该操作程序还会安装 IBM Operand Deployment Lifecycle Manager (ODLM) 操作程序。 安装 IBM MQ Operator 时，将自动安装这些操作程序。 这些从属操作程序具有较小的 CPU 和内存占用量，并且用于在某些情况下部署其他资源。

创建 `QueueManager` 时， IBM MQ Operator 将为其需要的其他服务创建 `OperandRequest` 。 `OperandRequest` 由 ODLM 操作程序实现，并将在必要时安装和实例化所需服务。 根据部署队列管理器时接受的许可协议以及请求的队列管理器组件来确定需要哪些服务。

- 如果选择 IBM MQ Advanced 或 IBM MQ Advanced for Developers 许可证，那么不会请求其他服务。 例如，在以下情况下，不使用 IBM Cloud Pak foundational services :

```
spec:
  license:
    accept: true
    license: L-APIG-BZDDDY
    use: "Production"
```

- 如果选择 IBM Cloud Pak for Integration 许可证并选择启用 Web 服务器，那么 IBM MQ Operator 还将实例化 IBM Identity and Access Management (IAM) 操作程序以启用单点登录。 如果已安装 IBM Cloud Pak for Integration 操作程序，那么 IAM 操作程序将已可用。 例如：

```
spec:
  license:
    accept: true
    license: L-RJON-BUVMQX
    use: "Production"
```

  但是，如果禁用 Web 服务器，那么不会请求任何 IBM Cloud Pak foundational services 。 例如：

```
spec:
  license:
    accept: true
    license: L-RJON-BUVMQX
    use: "Production"
```

[3] Red Hat OpenShift Container Platform 版本受其自己的支持日期限制。 请参阅 Red Hat OpenShift Container Platform 生命周期策略 以获取更多信息。
[4] Red Hat OpenShift Container Platform 4.4 已达到 "生命周期结束"。 请参阅 Red Hat OpenShift Container Platform 生命周期策略 以获取更多信息。
[5] Red Hat OpenShift Container Platform 4.5 已达到 "生命周期结束"。 请参阅 Red Hat OpenShift Container Platform 生命周期策略 以获取更多信息。
[6] IBM MQ Operator 依赖于 IBM Cloud Pak foundational services。 如果要使用 Red Hat OpenShift Container Platform 4.7，应首先升级 IBM Cloud Pak foundational services 的版本。

```
  web:
    enabled: false
```

IBM MQ Operator 的较低版本始终请求安装 IBM Licensing Operator (及其依赖项) 以跟踪许可证使用情况。从 IBM MQ Operator 1.5 开始，不会请求许可服务，您需要单独请求该服务。

IBM MQ Operator 需要 1 个 CPU 核心和 1 GB 内存。 有关从属操作员的硬件和软件需求的详细细分，请参阅 基础服务的硬件需求和建议。

您可以选择队列管理器所使用的 CPU 和内存量。 请参阅 第 122 页的『.spec.queueManager.resources』 以获取更多信息。

相关参考

第 114 页的『mq.ibm.com/v1beta1 的许可参考』

OpenShift CP4I IBM MQ Operator 所需的集群范围的许可权

IBM MQ Operator 需要具有集群作用域的许可权来管理许可 Webhook 和样本，以及读取存储类和集群版本信息。

IBM MQ Operator 需要以下集群范围的许可权:

- 管理许可 Webhook 的许可权。 这允许创建，检索和更新在创建和管理操作程序提供的容器的过程中使用的特定 Webhook。
  - API 组: **admissionregistration.k8s.io**
  - 资源： **validatingwebhookconfigurations**
  - verbs: **create, get, update**
- 用于创建和管理 Red Hat OpenShift 控制台中用于在创建定制资源时提供样本和片段的资源的许可权。
  - API 组: **console.openshift.io**
  - 资源： **consoleyamlsamples**
  - verbs: **create, get, update, delete**
- 用于读取集群版本的许可权。 这允许操作员反馈集群环境的任何问题。
  - API 组: **config.openshift.io**
  - 资源： **clusterversions**
  - verbs: **get, list, watch**
- 有权读取集群上的存储类。 这允许操作员反馈容器中所选存储类的任何问题。
  - API 组: **storage.k8s.io**
  - 资源： **storageclasses**
  - verbs: **get, list**

OpenShift CP4I Kubernetes IBM MQ Operator 的存储注意事项

IBM MQ Operator 以两种存储方式运行:

- 当容器重新启动时可以废弃容器的所有状态信息时，将使用 **临时存储器** 。 在创建环境以进行演示时，或者在使用独立队列管理器进行开发时，通常会使用此参数。
- **持久存储器** 是 IBM MQ 的公共配置，并确保在重新启动容器时，现有配置，日志和持久消息在重新启动的容器中可用。

IBM MQ Operator 提供了定制存储特征的功能，根据环境和期望的存储方式，这些存储特征可能有很大差异。

短暂存储量

IBM MQ 是一个有状态的应用程序，在重新启动时将此状态持久存储到存储器以进行恢复。 如果使用临时存储器，那么重新启动时会丢失队列管理器的所有状态信息。 这包括：

- 全部消息
- 所有队列管理器到队列管理器的通信状态 (通道消息序号)
- 队列管理器的 MQ 集群标识
- 所有事务状态
- 所有队列管理器配置
- 所有本地诊断数据

因此，您需要考虑临时存储器是否是适合生产，测试或开发方案的方法。 例如，已知所有消息都是非持久消息，并且队列管理器不是 MQ 集群的成员。 除了在重新启动时处理所有消息传递状态外，还会废弃队列管理器的配置。 要启用完全临时容器，必须将 IBM MQ 配置添加到容器映像本身 (有关更多信息，请参阅 第 101 页的『使用 Red Hat OpenShift CLI 构建具有定制 MQSC 和 INI 文件的映像』)。 如果未完成此操作，那么每次容器重新启动时都需要配置 IBM MQ 。

OpenShift CP4I 例如，要使用临时存储器配置 IBM MQ， `QueueManager` 的存储类型应包括以下内容：

```
queueManager:
  storage:
    queueManager:
      type: ephemeral
```

持久存储器

OpenShift CP4I

IBM MQ 通常与持久性存储器一起运行，以确保队列管理器在重新启动后保留其持久消息和配置。 因此，这是缺省行为。 由于各个存储提供程序和每个支持的不同功能，这通常意味着需要定制配置。 以下示例概述了用于在 v1beta1 API 中定制 MQ 存储配置的公共字段：

- spec.queueManager.availability 控制可用性方式。 如果您使用的是 `SingleInstance`，那么仅需要 `ReadWriteOnce` 存储器，而 `multiInstance` 需要具有正确文件锁定特征的支持 `ReadWriteMany` 的存储类。 IBM MQ 提供了 支持声明 和 测试声明。 可用性方式还会影响持久卷布局。 有关更多信息，请参阅 第 14 页的『容器中 IBM MQ 的高可用性』
- spec.queueManager.storage 控制各个存储器设置。 可以将队列管理器配置为在 1 到 4 个持久卷之间使用

以下示例显示了使用单实例队列管理器的简单配置片段：

```
spec:
  queueManager:
    storage:
      queueManager:
        enabled: true
```

以下示例显示了具有非缺省存储类以及需要补充组的文件存储器的多实例队列管理器配置片段：

```
spec:
  queueManager:
    availability:
      type: MultiInstance
    storage:
      queueManager:
        class: ibmc-file-gold-gid
      persistedData:
        enabled: true
        class: ibmc-file-gold-gid
      recoveryLogs:
        enabled: true
        class: ibmc-file-gold-gid
  securityContext:
     supplementalGroups: [99]
```

注: 您还可以使用单实例队列管理器配置补充组。

注: 如果使用本机 HA，那么不需要共享文件系统 (请参阅 第 14 页的『容器中 IBM MQ 的高可用性』)。 特别是，您不应使用 NFSv3。

Linux 支持构建您自己的 IBM MQ 队列管理器容器映像

IBM MQ 提供了用于在 GitHub 上构建 IBM MQ 队列管理器容器的代码。 这是基于 IBM 用于构建其自己的受支持容器的过程，您可以使用此 GitHub 存储库来简化和加速构建自己的容器映像。

此处的 mq-container GitHub 存储库中提供了代码: https://github.com/ibm-messaging/mq-container。 这是在 Apache 2.0 许可证下提供的，社区提供了支持。

存储库不使用标准 Linux rpm 软件包; 它将压缩软件包用于容器部署。 此方法的优点是您可以在更安全的容器环境中运行，而无需升级的许可权。 但是，这会影响可用的安全性选项，因为 IBM MQ 传统上使用升级后的许可权进行基于操作系统的认证。 对于容器部署，使用基于操作系统的认证通常不是好的做法; 而是可以使用相互 TLS 或 LDAP 认证。 通过 IBM MQ Advanced for Developers，您还可以使用基于文件的认证，从而使用户能够快速入门。

在容器环境中不支持复制的数据队列管理器 (RDQM)。 您可以使用 第 84 页的『本机 HA』获取与 RDQM 类似的功能。

相关概念

第 6 页的『对 IBM MQ Operator 的支持』
仅当在 Red Hat OpenShift Container Platform 上部署时，才支持 IBM MQ Operator 。

IBM MQ 非安装映像

Linux 构建您自己的 IBM MQ 容器映像时的许可证注释

通过许可证注释，可以根据容器上定义的限制而不是底层机器来跟踪使用情况。 配置客户机以部署具有特定注释的容器，然后 IBM License Service 使用这些注释来跟踪使用情况。

部署自构建的 IBM MQ 容器映像时，有两种常见的许可方法:

- 许可运行容器的整个机器。
- 根据相关限制对容器进行许可。

这两个选项都可供客户使用，可以在 Passport Advantage 上的 IBM Container 许可证页面 上找到更多详细信息。

如果要根据容器限制对 IBM MQ 容器进行许可，那么需要安装 IBM License Service 以跟踪使用情况。 可在 GitHub 上的 ibm-licensing-operator 页面上找到有关受支持环境和安装指示信息的更多信息。

IBM License Service 安装在部署了 IBM MQ 容器的 Kubernetes 集群上，并且 pod 注释用于跟踪使用情况。 因此，客户机需要使用 IBM License Service 随后使用的特定注释来部署 pod。 根据您在容器内部署的权利和能力，使用以下一个或多个注释:

- 第 13 页的『IBM MQ 高级容器』
- 第 13 页的『IBM MQ 高级高可用性副本容器』
- 第 13 页的『IBM MQ 基本容器』
- 第 13 页的『IBM MQ 基本高可用性副本容器』
- 第 13 页的『IBM MQ Advanced for Developers 容器』
- 第 13 页的『IBM MQ 具有 CP4I 权利的高级容器 (生产)』
- 第 13 页的『IBM MQ 具有 CP4I 权利的高级高可用性副本容器 (生产)』
- 第 14 页的『IBM MQ 具有 CP4I 权利的高级容器 (非生产)』
- 第 14 页的『IBM MQ 具有 CP4I 权利的高级高可用性副本容器 (非生产)』
- 第 14 页的『IBM MQ 具有 CP4I 权利的基本权利 (生产)』
- 第 14 页的『IBM MQ 具有 CP4I 权利的基本高可用性副本 (生产)』

- 第 14 页的『IBM MQ 具有 CP4I 权利的基本权利 (非生产)』
- 第 14 页的『IBM MQ 具有 CP4I 权利的基本高可用性副本 (非生产)』

IBM MQ 高级容器

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

IBM MQ 高级高可用性副本容器

```
productName: "IBM MQ Advanced High Availability Replica"
productID: "546cb719714942c18748137ddd8d5659"
productChargedContainers: "All" | "NAME_OF_CONTAINER"
productMetric: "PROCESSOR_VALUE_UNIT" |  "VIRTUAL_PROCESSOR_CORE"
```

IBM MQ 基本容器

```
productName: "IBM MQ"
productID: "c661609261d5471fb4ff8970a36bccea"
productChargedContainers: "All" | "NAME_OF_CONTAINER"
productMetric: "PROCESSOR_VALUE_UNIT" |  "VIRTUAL_PROCESSOR_CORE"
```

IBM MQ 基本高可用性副本容器

```
productName: "IBM MQ High Availability Replica"
productID: "2a2a8e0511c849969d2f286670ea125e"
productChargedContainers: "All" | "NAME_OF_CONTAINER"
productMetric: "PROCESSOR_VALUE_UNIT" |  "VIRTUAL_PROCESSOR_CORE"
```

IBM MQ Advanced for Developers 容器

```
productName: "IBM MQ Advanced for Developers"
productID: "2f886a3eefbe4ccb89b2adb97c78b9cb"
productChargedContainers: "All" | "NAME_OF_CONTAINER"
productMetric: "FREE"
```

IBM MQ 具有 CP4I 权利的高级容器 (生产)

```
productName: "IBM MQ Advanced with CP4I License"
productID: "208423bb063c43288328b1d788745b0c"
productChargedContainers: "All" | "NAME_OF_CONTAINER"
productMetric: "VIRTUAL_PROCESSOR_CORE"
productCloudpakRatio: "2:1"
cloudpakName: "IBM Cloud Pak for Integration"
cloudpakId: "c8b82d189e7545f0892db9ef2731b90d"
```

IBM MQ 具有 CP4I 权利的高级高可用性副本容器 (生产)

```
productName: "IBM MQ Advanced High Availability Replica with CP4I License"
productID: "546cb719714942c18748137ddd8d5659"
productChargedContainers: "All" | "NAME_OF_CONTAINER"
productMetric: "VIRTUAL_PROCESSOR_CORE"
productCloudpakRatio: "10:1"
cloudpakName: "IBM Cloud Pak for Integration"
cloudpakId: "c8b82d189e7545f0892db9ef2731b90d"
```

IBM MQ 具有 CP4I 权利的高级容器 (非生产)

```
productName: "IBM MQ Advanced for Non-Production with CP4I License"
productID: "21dfe9a0f00f444f888756d835334909"
productChargedContainers: "All" | "NAME_OF_CONTAINER"
productMetric: "VIRTUAL_PROCESSOR_CORE"
productCloudpakRatio: "4:1"
cloudpakName: "IBM Cloud Pak for Integration"
cloudpakId: "c8b82d189e7545f0892db9ef2731b90d"
```

IBM MQ 具有 CP4I 权利的高级高可用性副本容器 (非生产)

```
productName: "IBM MQ Advanced High Availability Replica for Non-Production with CP4I License"
productID: "b3f8f984007d47fb981221589cc50081"
productChargedContainers: "All" | "NAME_OF_CONTAINER"
productMetric: "VIRTUAL_PROCESSOR_CORE"
productCloudpakRatio: "20:1"
cloudpakName: "IBM Cloud Pak for Integration"
cloudpakId: "c8b82d189e7545f0892db9ef2731b90d"
```

IBM MQ 具有 CP4I 权利的基本权利 (生产)

```
productName: "IBM MQ with CP4I License"
productID: "c661609261d5471fb4ff8970a36bccea"
productChargedContainers: "All" | "NAME_OF_CONTAINER"
productMetric: "VIRTUAL_PROCESSOR_CORE"
productCloudpakRatio: "4:1"
cloudpakName: "IBM Cloud Pak for Integration"
cloudpakId: "c8b82d189e7545f0892db9ef2731b90d"
```

IBM MQ 具有 CP4I 权利的基本高可用性副本 (生产)

```
productName: "IBM MQ High Availability Replica with CP4I License"
productID: "2a2a8e0511c849969d2f286670ea125e"
productChargedContainers: "All" | "NAME_OF_CONTAINER"
productMetric: "VIRTUAL_PROCESSOR_CORE"
productCloudpakRatio: "20:1"
cloudpakName: "IBM Cloud Pak for Integration"
cloudpakId: "c8b82d189e7545f0892db9ef2731b90d"
```

IBM MQ 具有 CP4I 权利的基本权利 (非生产)

```
productName: "IBM MQ with CP4I License Non-Production"
productID: "151bec68564a4a47a14e6fa99266deff"
productChargedContainers:  "All" | "NAME_OF_CONTAINER"
productMetric: "VIRTUAL_PROCESSOR_CORE"
productCloudpakRatio: "8:1"
cloudpakName: "IBM Cloud Pak for Integration"
cloudpakId: "c8b82d189e7545f0892db9ef2731b90d"
```

IBM MQ 具有 CP4I 权利的基本高可用性副本 (非生产)

```
productName: "IBM MQ High Availability Replica with CP4I License Non-Production"
productID: "f5d0e21c013c4d4b8b9b2ce701f31928"
productChargedContainers: "All" | "NAME_OF_CONTAINER"
productMetric: "VIRTUAL_PROCESSOR_CORE"
productCloudpakRatio: "40:1"
cloudpakName: "IBM Cloud Pak for Integration"
cloudpakId: "c8b82d189e7545f0892db9ef2731b90d"
```

OpenShift CP4I Kubernetes 容器中 IBM MQ 的高可用性

对于具有 IBM MQ Operator 的高可用性，有三个选项: **本机 HA 队列管理器** (具有一个活动副本和两个备用副本)， **多实例队列管理器** (使用共享的网络文件系统的活动/备用对) 或 **单个弹性队列管理器** (使用网络存储器为 HA 提供简单方法)。 后两者依赖于文件系统来确保可恢复数据的可用性，但是本机 HA 不会。 因此，如

果不使用本机 HA，那么文件系统的可用性对于队列管理器可用性至关重要。 如果数据恢复很重要，那么文件系统应确保通过复制实现冗余。

您应该单独考虑 **消息** 和 **服务** 可用性。 使用 IBM MQ for Multiplatforms 时，消息仅存储在一个队列管理器上。 因此，如果该队列管理器变得不可用，那么您将暂时失去对其保存的消息的访问权。 要实现高 **消息** 可用性，您需要能够尽快恢复队列管理器。 您可以通过具有多个队列实例供客户机应用程序使用 (例如，通过使用 IBM MQ 统一集群) 来实现 **服务** 可用性。

队列管理器可以分为两部分: 存储在磁盘上的数据以及允许访问数据的正在运行的进程。 只要保留相同的数据 (由 Kubernetes 持久卷提供) 并且仍可由客户机应用程序跨网络寻址，任何队列管理器都可以移动到不同的 Kubernetes 节点。 在 Kubernetes 中，服务用于提供一致的网络身份。

IBM MQ 依赖于持久卷上数据的可用性。 因此，提供持久卷的存储器的可用性对于队列管理器可用性至关重要，因为 IBM MQ 的可用性不能超过它所使用的存储器。 如果要容许整个可用性区域的中断，那么需要使用将磁盘写入复制到另一个区域的卷提供程序。

本机 HA 队列管理器

CP4I V 9.2.3

本机 HA 队列管理器可从 IBM Cloud Pak for Integration 2021.2.1 中使用 IBM MQ Operator 1.6 或更高版本以及 IBM MQ 9.2.3 或更高版本。

本机 HA 队列管理器涉及一个 **活动** 和两个 **副本** Kubernetes Pod，它们作为 Kubernetes StatefulSet 的一部分运行，每个副本正好有三个副本具有自己的一组 Kubernetes 持久卷。 使用本机 HA 队列管理器 (基于租赁的锁定除外) 时，共享文件系统的 IBM MQ 需求也适用，但您不需要使用共享文件系统。 您可以使用块存储器，顶部有合适的文件系统。 例如， *xfs* 或 *ext4*。 本机 HA 队列管理器的恢复时间由以下因素控制:

1. 副本实例检测活动实例是否失败所需要的时间。 这是可配置的。
2. Kubernetes Pod 就绪性探测器检测就绪容器是否已更改并重定向网络流量所需的时间。 这是可配置的。
3. IBM MQ 客户机重新连接所需的时间。

有关更多信息，请参阅 第 84 页的『本机 HA』

多实例队列管理器

Multi

多实例队列管理器涉及一个 **活动** 和一个 **备用** Kubernetes Pod，它们作为具有正好两个副本和一组 Kubernetes 持久卷的 Kubernetes 有状态集的一部分运行。 队列管理器事务日志和数据使用共享文件系统保存在两个持久卷上。

多实例队列管理器需要 **active** 和 **standby** Pod 都具有对持久卷的并发访问权。 要对此进行配置，请使用设置为 `ReadWrite` **多项**的 Kubernetes 持久卷 **`access mode`**。 这些卷还必须满足 IBM MQ 共享文件系统的需求，因为 IBM MQ 依赖于自动释放文件锁定来启动队列管理器故障转移。 IBM MQ 生成 已测试文件系统列表。

多实例队列管理器的恢复时间由以下因素控制:

1. 发生故障后，共享文件系统释放活动实例最初获取的锁定所需的时间。
2. 备用实例获取锁定然后启动所需的时间。
3. Kubernetes Pod 就绪性探测器检测就绪容器是否已更改并重定向网络流量所需的时间。 这是可配置的。
4. IBM MQ 客户机重新连接所需的时间。

单个弹性队列管理器

Multi

单个弹性队列管理器是在单个 Kubernetes Pod 中运行的队列管理器的单个实例，其中 Kubernetes 监视队列管理器并根据需要替换 Pod。

在使用单个弹性队列管理器 (基于租赁的锁定除外) 时，共享文件系统的 IBM MQ 需求 也适用，但您不需要使用共享文件系统。 您可以使用块存储器，顶部有合适的文件系统。 例如， *xfs* 或 *ext4*。

单个弹性队列管理器的恢复时间由以下因素控制:

1. 活动性探测器运行所需的时间，以及它容忍的失败次数。 这是可配置的。
2. Kubernetes 调度程序将失败的 Pod 重新调度到新节点所需的时间。
3. 将容器映像下载到新节点所需的时间。 如果使用 **imagePullPolicy** 值 `IfNotPresent`，那么该映像可能已在该节点上可用。
4. 启动新队列管理器实例所需的时间。
5. Kubernetes Pod 就绪性探测器检测容器是否就绪所需的时间。 这是可配置的。
6. IBM MQ 客户机重新连接所需的时间。

要点:

虽然单一弹性队列管理器模式提供了一些优势，但您需要了解是否可以通过针对 Node 故障的限制来实现可用性目标。

在 Kubernetes 中，发生故障的 Pod 通常会快速恢复; 但整个 Node 的故障会以不同方式进行处理。 将有状态工作负载 (例如， IBM MQ) 与 Kubernetes StatefulSet 配合使用时，如果 Kubernetes 主节点与工作程序节点失去联系，那么无法确定该节点是否已发生故障，也无法确定该节点是否已完全失去网络连接。 因此，在此情况下， Kubernetes 将 **不执行任何操作** ，直到发生下列其中一个事件为止:

1. 节点恢复到 Kubernetes 主节点可与其通信的状态。
2. 将执行管理操作以显式删除 Kubernetes 主节点上的 Pod。 这不一定会阻止 Pod 运行，而只是将其从 Kubernetes 商店中删除。 因此，必须非常谨慎地采取这一行政行动。

相关任务

第 84 页的『使用 IBM MQ Operator 为队列管理器配置高可用性』

相关参考

高可用性配置

OpenShift CP4I Kubernetes 容器中 IBM MQ 的灾难恢复

你需要考虑你在为什么样的灾难做准备。 在云环境中，可用性区域的使用为灾难提供了一定级别的容忍度，并且更易于使用。 如果您有奇数个数的数据中心 (针对定额) 和低延迟网络链路，那么可能运行具有多个可用性区域的单个 Red Hat OpenShift Container Platform 或 Kubernetes 集群，每个都位于单独的物理位置。 本主题讨论在无法满足这些条件的情况下进行灾难恢复的注意事项: 即，数据中心数量均匀，或者存在高延迟网络链路。

对于灾难恢复，您需要考虑以下事项:

- 将 IBM MQ 数据 (保存在一个或多个 `PersistentVolume` 资源中) 复制到灾难恢复位置
- 使用复制的数据重新创建队列管理器
- 对 IBM MQ 客户机应用程序和其他队列管理器可见的队列管理器网络标识。 例如，此标识可以是 DNS 条目。

需要以同步或异步方式将持久数据复制到灾难恢复站点。 这通常特定于存储器提供程序，但也可以使用 `VolumeSnapshot` 来完成。 有关卷快照的更多信息，请参阅 CSI 卷快照 。

从灾难恢复时，您将需要使用复制的数据在新的 Kubernetes 集群上重新创建队列管理器实例。 如果您正在使用 IBM MQ Operator，那么将需要 `QueueManager` YAML 以及其他支持资源 (例如 `ConfigMap` 或 `Secret`) 的 YAML。

相关信息

ha_for_ctr.dita

容器中 IBM MQ 的用户认证和授权

可以将 IBM MQ 配置为使用 LDAP 用户和组。 或者，可以使用容器映像中的本地操作系统用户和组。 由于安全问题， IBM MQ Operator 不允许操作系统用户和组的用户。

在多租户容器化环境中，通常会实施安全约束以防止潜在的安全问题，例如:

- **防止在容器中使用 "root" 用户**
- **强制使用随机 UID**。 例如，在 Red Hat OpenShift Container Platform 中，缺省 `SecurityContextConstraints` (称为 `restricted`) 对每个容器使用随机用户标识。
- **阻止使用特权升级**。 IBM MQ on Linux 使用特权升级来检查用户的密码-它使用 "setuid" 程序作为 "root" 用户来执行此操作。

OpenShift CP4I 为了确保符合这些安全措施， IBM MQ Operator 不允许使用在容器内的操作系统库上定义的标识。 容器中未定义 `mqm` 用户标识或组。 在 IBM Cloud Pak for Integration 和 Red Hat OpenShift 中使用 IBM MQ 时，需要配置队列管理器以使用 LDAP 进行用户认证和授权。 有关配置 IBM MQ 以执行此操作的信息，请参阅 连接认证: 用户存储库 和 LDAP 授权

Multi 规划容器中 IBM MQ 的可伸缩性和性能

在大多数情况下，容器中 IBM MQ 的缩放和性能与 IBM MQ for Multiplatforms 相同。 但是，容器平台可以施加一些额外的限制。

关于此任务

在容器中规划 IBM MQ 的可伸缩性和性能时，请考虑以下选项:

过程

- **限制线程数和进程数**。

  IBM MQ 使用线程来管理并行性。 在 Linux 中，线程实现为进程，因此您可能会迁到容器平台或操作系统对最大进程数施加的限制。 在 Red Hat OpenShift Container Platform 中，每个容器的缺省限制为 4096 个进程 (1024 个进程，直到 OpenShift 4.11)。 虽然这对于绝大多数方案都是足够的，但在某些情况下，这可能会影响队列管理器的客户机连接数。

  Kubernetes 中的进程限制可以由集群管理员使用 kubelet 配置设置 **`podPidsLimit`** 进行配置。 请参阅 Kubernetes 文档中的 进程标识限制和预留 。 在 Red Hat OpenShift Container Platform 中，还可以 创建 **`ContainerRuntimeConfig`** 定制资源以编辑 CRI-O 参数。

  在 IBM MQ 配置中，您还可以设置队列管理器的最大客户机连接数。 请参阅 服务器连接通道限制 以将限制应用于单个服务器连接通道，并参阅 MAXCHANNELS INI 属性 以将限制应用于整个队列管理器。
- **限制卷数**。

  在云和容器系统中，通常使用网络连接的存储卷。 可以连接到 Linux 节点的卷数有限制。 例如， AWS EC2 限制为每个 VM 不超过 30 个卷。 Red Hat OpenShift Container Platform 具有类似的限制，如 Microsoft Azure 和 Google Cloud Platform。

  本机 HA 队列管理器需要三个实例中的每个实例一个卷，并强制在节点之间分布实例。 但是，您可以将队列管理器配置为每个实例使用三个卷 (队列管理器数据，恢复日志和持久数据)。
- **使用 IBM MQ 缩放技术**。

  使用 IBM MQ 缩放技术 (例如 IBM MQ 统一集群) 来运行具有相同配置的多个队列管理器可能是有益的，而不是使用少量大型队列管理器。 这具有额外的好处，即单个容器重新启动 (例如，作为容器平台维护的一部分) 的影响会降低。

OpenShift CD CP4I EUS 在 IBM Cloud Pak for Integration 和 Red Hat OpenShift 中使用 IBM MQ

IBM MQ Operator 将 IBM MQ 作为 IBM Cloud Pak for Integration 的一部分进行部署和管理，或者在 Red Hat OpenShift Container Platform 上独立进行部署和管理

过程

- 第 18 页的『IBM MQ Operator 的发布历史记录』.
- 第 33 页的『将 IBM MQ 迁移到 IBM Cloud Pak for Integration』.

- 第 55 页的『在 Red Hat OpenShift 上安装和卸载 IBM MQ Operator』.
- 第 66 页的『升级 IBM MQ Operator 和队列管理器』.
- 第 73 页的『使用 IBM MQ Operator 部署和配置队列管理器』.
- 第 104 页的『使用 IBM MQ Operator 操作 IBM MQ』.
- 第 114 页的『IBM MQ Operator 的 API 参考』.

OpenShift CD CP4I EUS IBM MQ Operator 的发布历史记录

IBM MQ Operator

IBM MQ Operator 1.8.2

CD

IBM Cloud Pak for Integration 版本
IBM Cloud Pak for Integration 2021.4.1

操作员通道
v1.8

`.spec.version` 的允许值
9.1.5.0-r2, 9.2.0.0-r1, 9.2.0.0-r2, 9.2.0.0-r3, 9.2.0.1-r1-eus, 9.2.0.2-r1-eus, 9.2.0.2-r2-eus, 9.2.0.5-r1-eus, 9.2.0.5-r2-eus, 9.2.0.5-r3-eus, 9.2.1.0-r1, 9.2.1.0-r2, 9.2.2.0-r1, 9.2.3.0-r1, 9.2.4.0-r1, 9.2.5.0-r1, 9.2.5.0-r2, 9.2.5.0-r3

Red Hat OpenShift Container Platform 版本
Red Hat OpenShift Container Platform 4.6 及更高版本

IBM Cloud Pak foundational services 版本
IBM Cloud Pak foundational services 3.8 和更高版本 (v3 通道)

新增内容

- 基于 IBM MQ 操作程序 1.8.0 构建的仅安全性更新。
- 此 安全公告中详细描述了已解决的漏洞。

IBM MQ Operator 1.8.1

CD

IBM Cloud Pak for Integration 版本
IBM Cloud Pak for Integration 2021.4.1

操作员通道
v1.8

`.spec.version` 的允许值
9.1.5.0-r2, 9.2.0.0-r1, 9.2.0.0-r2, 9.2.0.0-r3, 9.2.0.1-r1-eus, 9.2.0.2-r1-eus, 9.2.0.2-r2-eus, 9.2.1.0-r1, 9.2.1.0-r2, 9.2.2.0-r1, 9.2.3.0-r1, 9.2.4.0-r1, 9.2.5.0-r1, 9.2.5.0-r2

Red Hat OpenShift Container Platform 版本
Red Hat OpenShift Container Platform 4.6 及更高版本

IBM Cloud Pak foundational services 版本
IBM Cloud Pak foundational services 3.8 和更高版本 (v3 通道)

新增内容

- 基于 IBM MQ 操作程序 1.8.0 构建的仅安全性更新。
- 此 安全公告中详细描述了已解决的漏洞。

IBM MQ Operator 1.8.0

CD

IBM Cloud Pak for Integration 版本
IBM Cloud Pak for Integration 2021.4.1

操作员通道
`v1.8`

`.spec.version` 的允许值
9.1.5.0-r2, 9.2.0.0-r1, 9.2.0.0-r2, 9.2.0.0-r3, 9.2.0.1-r1-eus, 9.2.0.2-r1-eus, 9.2.0.2-r2-eus, 9.2.1.0-r1, 9.2.1.0-r2, 9.2.2.0-r1, 9.2.3.0-r1, 9.2.4.0-r1, 9.2.5.0-r1

Red Hat OpenShift Container Platform 版本
Red Hat OpenShift Container Platform 4.6 及更高版本

IBM Cloud Pak foundational services 版本
IBM Cloud Pak foundational services 3.8 和更高版本 (v3 通道)

新增内容

- 为不推荐的 IBM MQ 版本添加状态条件。

已更改的内容

- 映像已从 Docker Hub 移动到 IBM Container Registry。
  - 具有防火墙规则的客户可能需要对其进行调整以访问 IBM Container Registry 上的映像。
  - Airgap 客户在升级到 IBM MQ Operator 1.8.0 时迁到节点重新启动的问题。
- 不推荐使用的版本: IBM MQ 9.1.5， 9.2.0 CD， 9.2.1 和 9.2.2。 这些版本可能无法由 IBM MQ Operator 的未来版本协调。
- 对许可证逻辑的更改: 升级到 IBM MQ 9.2.5 的客户只能使用指定的许可证来使用 IBM MQ 9.2.5。 请参阅第 114 页的『mq.ibm.com/v1beta1 的许可参考』。
- 此 安全公告中详细描述了已解决的漏洞。

IBM MQ Operator 1.7.0

CD

IBM Cloud Pak for Integration 版本
IBM Cloud Pak for Integration 2021.4.1

操作员通道
`v1.7`

`.spec.version` 的允许值
9.1.5.0-r2, 9.2.0.0-r1, 9.2.0.0-r2, 9.2.0.0-r3, 9.2.0.1-r1-eus, 9.2.0.2-r1-eus, 9.2.0.2-r2-eus, 9.2.1.0-r1, 9.2.1.0-r2, 9.2.2.0-r1, 9.2.3.0-r1, 9.2.4.0-r1

Red Hat OpenShift Container Platform 版本
Red Hat OpenShift Container Platform 4.6 及更高版本

IBM Cloud Pak foundational services 版本
IBM Cloud Pak foundational services 3.8 和更高版本 (v3 通道)

新增内容

- 添加 IBM MQ 9.2.4 作为持续交付发行版

IBM MQ Operator 1.6.0

CD

IBM Cloud Pak for Integration 版本
IBM Cloud Pak for Integration 2021.2.1

操作员通道
v1.6

.spec.version 的允许值
9.1.5.0-r2, 9.2.0.0-r1, 9.2.0.0-r2, 9.2.0.0-r3, 9.2.0.1-r1-eus, 9.2.0.2-r1-eus, 9.2.0.2-r2-eus, 9.2.1.0-r1, 9.2.1.0-r2, 9.2.2.0-r1, 9.2.3.0-r1

Red Hat OpenShift Container Platform 版本
Red Hat OpenShift Container Platform 4.6 及更高版本

IBM Cloud Pak foundational services 版本
IBM Cloud Pak foundational services 3.7 及更高版本 (v3 通道)

新增内容

- 添加 IBM MQ 9.2.3 作为持续交付发行版 (amd64 仅适用于 IBM Cloud Pak for Integration 2021.2.1; amd64 或 s390x 使用 IBM MQ 许可证时)
- 队列管理器的新可用性类型: 本机 HA。 可供生产使用，作为 IBM Cloud Pak for Integration 2021.2.1 的一部分。

已更改的内容

- IBM MQ Operator 1.6 和更高版本使用 IBM Container Registry 而不是 Docker Hub。 这意味着您需要使用 icr.io 中的 CatalogSource 。 请参阅 第 55 页的『在 Red Hat OpenShift 上安装和卸载 IBM MQ Operator』。
- 本机 HA 滚动更新不再等待副本同步，然后再移至下一个副本。
- 修复 OCP 4.7 及更高版本上的本机 HA 亲缘关系问题。
- 将 CA 签名的证书与本机 HA 配合使用时的修订问题。

IBM MQ Operator 1.5.0

CD

IBM Cloud Pak for Integration 版本
IBM Cloud Pak for Integration 2021.1.1

操作员通道
v1.5

.spec.version 的允许值
9.1.5.0-r2, 9.2.0.0-r1, 9.2.0.0-r2, 9.2.0.0-r3, 9.2.0.1-r1-eus, 9.2.1.0-r1, 9.2.1.0-r2, 9.2.2.0-r1

Red Hat OpenShift Container Platform 版本
Red Hat OpenShift Container Platform 4.6 及更高版本

IBM Cloud Pak foundational services 版本
IBM Cloud Pak foundational services 3.7 及更高版本 (v3 通道)

新增内容

- 添加 IBM MQ 9.2.2 作为持续交付发行版 (amd64 仅适用于 IBM Cloud Pak for Integration 2021.1.1; amd64 或 s390x 使用 IBM MQ 许可证时)
- 队列管理器的新可用性类型: 本机 HA。 仅可用于评估，作为 IBM Cloud Pak for Integration 2021.1.1 的一部分。
- 通过提供 ServiceMonitor 资源，与 Red Hat OpenShift Container Platform Cluster Monitoring for Prometheus 度量集成

已更改的内容

- 缺省情况下，当您创建队列管理器时，将不再创建 IBM Licensing Operator
- 现在按滚动顺序处理对多实例队列管理器的更新。 作为此更改的一部分，引入了 Kubernetes 启动探测器，这将影响配置活动性探测器时使用的值。 启动探测器立即启动，然后等待队列管理器成功启动。 如果启动探测器在此等待时间段内的任何时间通过，那么将启动活动性和就绪性探测器。 先前，如果您有一个启动缓慢的队列管理器，那么可能已增加了活动性探测器上的 initialDelaySeconds 设置。 如果执行了此操作，那么现在应该将 initialDelaySeconds 还原为先前的设置。

- `CustomResourceDefinition` 已从 `apiextensions.k8s.io/v1beta1` 升级到 `apiextensions.k8s.io/v1`

已知问题和限制

- 需要 IBM Cloud Pak foundational services 3.7，它在 Identity and Access Management (IAM) 组件中包含不兼容的更改。 如果您有任何使用 IBM Cloud Pak for Integration 许可证的队列管理器，那么在此升级之后，将需要重新启动队列管理器才能访问 Web 控制台，并且您还会看到登录到 Web 控制台的 其他错误 。 在操作程序升级完成后，您可以通过升级到所选 IBM MQ 版本的最新值 `.spec.version` 来修正这些错误。
- 如果要升级 MQ 版本，那么不会自动启动滚动更新。 您需要手动删除 pod。

IBM MQ Operator 1.4.0

CD

IBM Cloud Pak for Integration 版本
IBM Cloud Pak for Integration 2020.4.1 (IBM MQ Operator 1.4.0 是 CD 发行版，不适合 "扩展更新支持")

操作员通道
`v1.4`

`.spec.version` 的允许值
9.1.5.0-r2, 9.2.0.0-r1, 9.2.0.0-r2, 9.2.0.1-r1-eus, 9.2.1.0-r1

Red Hat OpenShift Container Platform 版本
Red Hat OpenShift Container Platform 4.6 及更高版本

新增内容

- 添加 IBM MQ 9.2.1 作为持续交付发行版
- 现在，可以通过将 `.spec.queueManager.route.enabled` 设置为 `false` 来阻止创建缺省队列管理器路由

已知问题和限制

- 更新可用性类型为 `MultiInstance` 的 `QueueManager` 时，将立即删除这两个 Pod。 它们都应该由 Red Hat OpenShift Container Platform 快速重新启动。

IBM MQ Operator 1.3.8 (EUS)

EUS

IBM Cloud Pak for Integration 版本
IBM Cloud Pak for Integration 2020.4.1

操作员通道
`v1.3-eus`

`.spec.version` 的允许值
9.1.5.0-r2, 9.2.0.0-r1, 9.2.0.0-r2, 9.2.0.1-r1-eus, 9.2.0.2-r1-eus, 9.2.0.4-r1-eus, 9.2.0.5-r1-eus, 9.2.0.5-r2-eus, 9.2.0.5-r3-eus, 9.2.0.6-r1-eus, 9.2.0.6-r2-eus, 9.2.0.6-r3-eus

Red Hat OpenShift Container Platform 版本
仅 Red Hat OpenShift Container Platform 4.6

IBM Cloud Pak foundational services 版本
IBM Cloud Pak foundational services 3.6 (`stable-v1` 通道)

新增内容

- 添加新的操作数版本 9.2.0.6-r3-eus。
- 此 安全公告中详细描述了已解决的漏洞。

IBM MQ Operator 1.3.7 (EUS)

EUS

IBM Cloud Pak for Integration 版本
IBM Cloud Pak for Integration 2020.4.1

操作员通道
`v1.3-eus`

`.spec.version` 的允许值
9.1.5.0-r2, 9.2.0.0-r1, 9.2.0.0-r2, 9.2.0.1-r1-eus, 9.2.0.2-r1-eus, 9.2.0.4-r1-eus, 9.2.0.5-r1-eus, 9.2.0.5-r2-eus, 9.2.0.5-r3-eus, 9.2.0.6-r1-eus, 9.2.0.6-r2-eus

Red Hat OpenShift Container Platform 版本
仅 Red Hat OpenShift Container Platform 4.6

IBM Cloud Pak foundational services 版本
IBM Cloud Pak foundational services 3.6 (`stable-v1` 通道)

新增内容

- 添加新的操作数版本 9.2.0.6-r2-eus。
- 此 安全公告中详细描述了已解决的漏洞。

IBM MQ Operator 1.3.6 (EUS)

EUS

IBM Cloud Pak for Integration 版本
IBM Cloud Pak for Integration 2020.4.1

操作员通道
`v1.3-eus`

`.spec.version` 的允许值
9.1.5.0-r2, 9.2.0.0-r1, 9.2.0.0-r2, 9.2.0.1-r1-eus, 9.2.0.2-r1-eus, 9.2.0.4-r1-eus, 9.2.0.5-r1-eus, 9.2.0.5-r2-eus, 9.2.0.5-r3-eus, 9.2.0.6-r1-eus

Red Hat OpenShift Container Platform 版本
仅 Red Hat OpenShift Container Platform 4.6

IBM Cloud Pak foundational services 版本
IBM Cloud Pak foundational services 3.6 (`stable-v1` 通道)

新增内容

- 添加新的操作数版本 9.2.0.6-r1-eus。
- 此 安全公告中详细描述了已解决的漏洞。

IBM MQ Operator 1.3.5 (EUS)

EUS

IBM Cloud Pak for Integration 版本
IBM Cloud Pak for Integration 2020.4.1

操作员通道
`v1.3-eus`

`.spec.version` 的允许值
9.1.5.0-r2, 9.2.0.0-r1, 9.2.0.0-r2, 9.2.0.1-r1-eus, 9.2.0.2-r1-eus, 9.2.0.4-r1-eus, 9.2.0.5-r1-eus, 9.2.0.5-r2-eus, 9.2.0.5-r3-eus

Red Hat OpenShift Container Platform 版本
仅 Red Hat OpenShift Container Platform 4.6

IBM Cloud Pak foundational services 版本
IBM Cloud Pak foundational services 3.6 (`stable-v1` 通道)

新增内容

- 添加新的操作数版本 9.2.0.5-r3-eus。
- 此 安全公告中详细描述了已解决的漏洞。

IBM MQ Operator 1.3.4 (EUS)

EUS

IBM Cloud Pak for Integration 版本
IBM Cloud Pak for Integration 2020.4.1

操作员通道
`v1.3-eus`

`.spec.version` 的允许值
9.1.5.0-r2, 9.2.0.0-r1, 9.2.0.0-r2, 9.2.0.1-r1-eus, 9.2.0.2-r1-eus, 9.2.0.4-r1-eus, 9.2.0.5-r1-eus, 9.2.0.5-r2-eus

Red Hat OpenShift Container Platform 版本
仅 Red Hat OpenShift Container Platform 4.6

IBM Cloud Pak foundational services 版本
IBM Cloud Pak foundational services 3.6 (`stable-v1` 通道)

新增内容

- 添加新的操作数版本 9.2.0.5-r2-eus
- 此 安全公告中详细描述了已解决的漏洞。

IBM MQ Operator 1.3.3 (EUS)

EUS

IBM Cloud Pak for Integration 版本
IBM Cloud Pak for Integration 2020.4.1

操作员通道
`v1.3-eus`

`.spec.version` 的允许值
9.1.5.0-r2, 9.2.0.0-r1, 9.2.0.0-r2, 9.2.0.1-r1-eus, 9.2.0.2-r1-eus, 9.2.0.4-r1-eus, 9.2.0.5-r1-eus

Red Hat OpenShift Container Platform 版本
仅 Red Hat OpenShift Container Platform 4.6

IBM Cloud Pak foundational services 版本
IBM Cloud Pak foundational services 3.6 (`stable-v1` 通道)

新增内容

- 添加新的操作数版本 9.2.0.5-r1-eus
- 此 安全公告中详细描述了已解决的漏洞。

IBM MQ Operator 1.3.2 (EUS)

EUS

IBM Cloud Pak for Integration 版本
IBM Cloud Pak for Integration 2020.4.1

操作员通道
`v1.3-eus`

`.spec.version` 的允许值
9.1.5.0-r2, 9.2.0.0-r1, 9.2.0.0-r2, 9.2.0.1-r1-eus, 9.2.0.2-r1-eus, 9.2.0.4-r1-eus

Red Hat OpenShift Container Platform 版本
仅 Red Hat OpenShift Container Platform 4.6

IBM Cloud Pak foundational services 版本
IBM Cloud Pak foundational services 3.6 (`stable-v1` 通道)

新增内容

- 添加新的操作数版本 9.2.0.4-r1-eus

IBM MQ Operator 1.3.1 (EUS)

EUS

IBM Cloud Pak for Integration 版本
IBM Cloud Pak for Integration 2020.4.1

操作员通道
`v1.3-eus`

`.spec.version` 的允许值
9.1.5.0-r2, 9.2.0.0-r1, 9.2.0.0-r2, 9.2.0.1-r1-eus, 9.2.0.2-r1-eus

Red Hat OpenShift Container Platform 版本
仅 Red Hat OpenShift Container Platform 4.6

IBM Cloud Pak foundational services 版本
IBM Cloud Pak foundational services 3.6 (`stable-v1` 通道)

新增内容

- 添加新的操作数版本 9.2.0.2-r1-eus

IBM MQ Operator 1.3.0 (EUS)

EUS

IBM Cloud Pak for Integration 版本
IBM Cloud Pak for Integration 2020.4.1

操作员通道
`v1.3-eus`

`.spec.version` 的允许值
9.1.5.0-r2, 9.2.0.0-r1, 9.2.0.0-r2, 9.2.0.1-r1-eus

Red Hat OpenShift Container Platform 版本
仅 Red Hat OpenShift Container Platform 4.6

IBM Cloud Pak foundational services 版本
IBM Cloud Pak foundational services 3.6 (`stable-v1` 通道)

新增内容

- 使用 IBM Cloud Pak for Integration 许可证时，针对以 `-eus` 结尾的 `.spec.version` 字段提供扩展更新支持 (EUS)
- 添加使用 `.spec.labels` 和 `.spec.annotations` 在 `QueueManager` 资源上设置标签和注释的新方法

已更改的内容

- 改进尝试从单实例更改为多实例时的错误处理
- 改进了 `QueueManager` 属性在 IBM Cloud Pak for Integration Platform Navigator 和 Red Hat OpenShift Container Platform Web 控制台的 “表单视图” 中的呈现方式
- 将使用 IBM Cloud Pak for Integration 许可证时的缺省许可证度量修订为 `VirtualProcessorCore`
- 修复了 Red Hat OpenShift Container Platform Web 控制台中 `QueueManager` 的 **资源** 选项卡，该选项卡现在正确显示该队列管理器的 IBM MQ Operator 所管理的资源

已知问题和限制

- 更新可用性类型为 `MultiInstance` 的 `QueueManager` 时，将立即删除这两个 Pod。 它们都应该由 Red Hat OpenShift Container Platform 快速重新启动。

IBM MQ Operator 1.2.0

CD

IBM Cloud Pak for Integration 版本
IBM Cloud Pak for Integration 2020.3.1

操作员通道
`v1.2`

`.spec.version` 的允许值
9.1.5.0-r2, 9.2.0.0-r1, 9.2.0.0-r2

Red Hat OpenShift Container Platform 版本
Red Hat OpenShift Container Platform 4.4 及更高版本

新增内容

- 添加对 z/Linux 的支持
- 向 `QueueManager` 资源添加更详细的状态条件。 有关更多信息，请参阅 第 130 页的『QueueManager 的状态条件 (mq.ibm.com/v1beta1)』
- 添加其他运行时检查以防止使用无效存储类。 有关更多信息，请参阅 第 103 页的『禁用运行时 Webhook 检查』
- 简化多实例队列管理器的体验: 现在只需在 `QueueManager` 资源中使用一个属性 (`.spec.queueManager.availability.type`) 即可选择此属性
- 通过在 `QueueManager` 资源中引入 `.spec.queueManager.storage.defaultClass` 属性，简化非缺省存储类的选择

已更改的内容

- 改进了 `QueueManager` 属性在 IBM Cloud Pak for Integration Platform Navigator 和 Red Hat OpenShift Container Platform Web 控制台的 “表单视图” 中的呈现方式
- 如果升级后的队列管理器版本可用，那么现在将在 IBM Cloud Pak for Integration Platform Navigator 中对其进行标记

IBM MQ Operator 1.1.0

CD

IBM Cloud Pak for Integration 版本
IBM Cloud Pak for Integration 2020.2.1

操作员通道
`v1.1`

`.spec.version` 的允许值
9.1.5.0-r2, 9.2.0.0-r1

Red Hat OpenShift Container Platform 版本
Red Hat OpenShift Container Platform 4.4 及更高版本

新增内容

- 添加 IBM MQ Advanced 9.2.0 作为持续交付发行版
- 添加用于在 ConfigMap 或 Secret 中指定 INI 和 MQSC 信息的功能部件
- 使用 Red Hat OpenShift Container Platform Web 控制台时启用模式导航器

已更改的内容

- 修复网络策略问题，影响 IBM Cloud 上的 Red Hat OpenShift

- 对验证 Web 挂钩的改进，以防止 `QueueManager` 资源中的设置组合无效

IBM MQ Operator 1.0.0

CD

IBM Cloud Pak for Integration 版本
IBM Cloud Pak for Integration 2020.2.1

操作员通道
`v1.0`

`.spec.version` 的允许值
9.1.5.0-r2

Red Hat OpenShift Container Platform 版本
Red Hat OpenShift Container Platform 4.4 及更高版本

新增内容

- 操作程序的初始版本，引入 `mq.ibm.com/v1beta1` API

用于 *IBM MQ Operator* 的队列管理器容器映像

9.2.5.0-r3

CD

必需的操作程序版本
1.8.2 或更高版本

受支持的体系结构
amd64, s390x

图像

- `cp.icr.io/cp/ibm-mqadvanced-server-integration:9.2.5.0-r3`
- `cp.icr.io/cp/ibm-mqadvanced-server:9.2.5.0-r3`
- `icr.io/ibm-messaging/mq:9.2.5.0-r3`

新增内容

- IBM MQ 9.2.5 中的新增功能

已更改的内容

- IBM MQ 9.2.5 中更改的内容
- 基于 Red Hat 通用基本映像 8.6-751

9.2.5.0-r2

CD

必需的操作程序版本
1.8.1 或更高版本

受支持的体系结构
amd64, s390x

图像

- `cp.icr.io/cp/ibm-mqadvanced-server-integration:9.2.5.0-r2`
- `cp.icr.io/cp/ibm-mqadvanced-server:9.2.5.0-r2`
- `icr.io/ibm-messaging/mq:9.2.5.0-r2`

新增内容

- IBM MQ 9.2.5 中的新增功能

已更改的内容

- IBM MQ 9.2.5 中更改的内容
- 基于 Red Hat Universal Base Image 8.5-240.1648458092

9.2.5.0-r1

CD

必需的操作程序版本
1.8.0 或更高版本

受支持的体系结构
amd64, s390x

图像

- `cp.icr.io/cp/ibm-mqadvanced-server-integration:9.2.5.0-r1`
- `cp.icr.io/cp/ibm-mqadvanced-server:9.2.5.0-r1`
- `icr.io/ibm-messaging/mq:9.2.5.0-r1`

新增内容

- IBM MQ 9.2.5 中的新增功能

已更改的内容

- IBM MQ 9.2.5 中更改的内容
- 现在从 IBM MQ Console 中除去了无效的 **远程队列管理器** 选项
- 基于 Red Hat Universal Base Image 8.5-240

9.2.4.0-r1

CD

必需的操作程序版本
1.7.0 或更高版本

受支持的体系结构
amd64, s390x

图像

- `cp.icr.io/cp/ibm-mqadvanced-server-integration:9.2.4.0-r1`
- `cp.icr.io/cp/ibm-mqadvanced-server:9.2.4.0-r1`
- `docker.io/ibmcom/mq:9.2.4.0-r1`

新增内容

- IBM MQ 9.2.4 中的新增功能

已更改的内容

- IBM MQ 9.2.4 中更改的内容
- 基于 Red Hat 通用基本映像 8.5-204

9.2.3.0-r1

CD

必需的操作程序版本
1.6.0 或更高版本

受支持的体系结构
amd64, s390x

图像

- cp.icr.io/cp/ibm-mqadvanced-server-integration:9.2.3.0-r1 (仅限 amd64)
- cp.icr.io/cp/ibm-mqadvanced-server:9.2.3.0-r1
- docker.io/ibmcom/mq:9.2.3.0-r1

新增内容

- IBM MQ 9.2.3 中的新增功能
- 与 IBM Cloud Pak for Integration 许可证一起使用时，支持 MQ 本机 HA 用于生产。 请注意，在具有 IBM MQ 9.2.2 的评估许可证下使用本机 HA 的队列管理器无法升级到 9.2.3。 评估期已结束。

已更改的内容

- IBM MQ 9.2.3 中更改的内容
- 基于 Red Hat 通用基本映像 8.4-205

9.2.2.0-r1

CD

必需的操作程序版本
1.5.0 或更高版本

受支持的体系结构
amd64, s390x

图像

- cp.icr.io/cp/ibm-mqadvanced-server-integration:9.2.2.0-r1 (仅限 amd64)
- cp.icr.io/cp/ibm-mqadvanced-server:9.2.2.0-r1
- docker.io/ibmcom/mq:9.2.2.0-r1

新增内容

- IBM MQ 9.2.2 中的新增功能
- 支持 MQ 本机 HA 以用于评估目的 (与 IBM Cloud Pak for Integration 许可证一起使用时)

已更改的内容

- IBM MQ 9.2.2 中更改的内容
- 修复了在关闭 IBM MQ Advanced for Developers 队列管理器时导致 FDC 的问题
- 基于 Red Hat Universal Base Image 8.3-291

9.2.1.0-r2

CD

必需的操作程序版本
1.5.0 或更高版本

受支持的体系结构
amd64, s390x

图像

- cp.icr.io/cp/ibm-mqadvanced-server-integration:9.2.1.0-r2
- cp.icr.io/cp/ibm-mqadvanced-server:9.2.1.0-r2
- docker.io/ibmcom/mq:9.2.1.0-r2

已更改的内容

- 修复了 IBM Cloud Pak foundational services 3.7 及更高版本的单点登录问题。
- 基于 Red Hat Universal Base Image 8.3-291

9.2.1.0-r1

CD

必需的操作程序版本
1.4.0 或更高版本

受支持的体系结构
amd64, s390x

图像

- `cp.icr.io/cp/ibm-mqadvanced-server-integration:9.2.1.0-r1`
- `cp.icr.io/cp/ibm-mqadvanced-server:9.2.1.0-r1`
- `docker.io/ibmcom/mq:9.2.1.0-r1`

新增内容

- IBM MQ 9.2.1 中的新增功能
- MQ Web 控制台中提供了缺省路由的连接信息

已更改的内容

- IBM MQ 9.2.1 中更改的内容
- 基于 Red Hat 通用基本映像 8.3-230

9.2.0.6-r3-eus

EUS

必需的操作程序版本
1.3.8 和未来修订包

受支持的体系结构
amd64, s390x

图像

- `cp.icr.io/cp/ibm-mqadvanced-server-integration:9.2.0.6-r3-eus`

已更改的内容

- 包含 IBM MQ 9.2.0 Fix Pack 6。 有关更多信息，请参阅 IBM MQ V 9.2 LTS 的修订列表。
- 基于 Red Hat Universal Base Image 8.6-941。

9.2.0.6-r2-eus

EUS

必需的操作程序版本
1.3.7 和未来修订包

受支持的体系结构
amd64, s390x

图像

- `cp.icr.io/cp/ibm-mqadvanced-server-integration:9.2.0.6-r2-eus`

已更改的内容

- 包含 IBM MQ 9.2.0 Fix Pack 6。 有关更多信息，请参阅 IBM MQ V 9.2 LTS 的修订列表。

- 基于 Red Hat Universal Base Image 8.6-902。

9.2.0.6-r1-eus

EUS

必需的操作程序版本
1.3.6 和未来修订包

受支持的体系结构
amd64, s390x

图像

- `cp.icr.io/cp/ibm-mqadvanced-server-integration:9.2.0.6-r1-eus`

已更改的内容

- 包含 IBM MQ 9.2.0 Fix Pack 6。 有关更多信息，请参阅 IBM MQ V 9.2 LTS 的修订列表。
- 基于 Red Hat Universal Base Image 8.6-854。

9.2.0.5-r3-eus

EUS

必需的操作程序版本
1.3.5 和未来修订包

受支持的体系结构
amd64, s390x

图像

- `cp.icr.io/cp/ibm-mqadvanced-server-integration:9.2.0.5-r3-eus`

已更改的内容

- 包含 IBM MQ 9.2.0 Fix Pack 5。 有关更多信息，请参阅 IBM MQ 9.2.0 Fix Pack 5 和 IBM MQ V 9.2 LTS 的修订列表中的更改内容。
- 基于 Red Hat Universal Base Image 8.6-751.1655117800。

9.2.0.5-r2-eus

EUS

必需的操作程序版本
1.3.4 和未来修订包

受支持的体系结构
amd64, s390x

图像

- `cp.icr.io/cp/ibm-mqadvanced-server-integration:9.2.0.5-r2-eus`

已更改的内容

- 包含 IBM MQ 9.2.0 Fix Pack 5。 有关更多信息，请参阅 IBM MQ 9.2.0 Fix Pack 5 中的更改内容和 IBM MQ V 9.2 LTS 的修订列表
- 基于 Red Hat 通用基本映像 8.6-751

9.2.0.5-r1-eus

EUS

必需的操作程序版本
1.3.3 和未来修订包

受支持的体系结构
amd64, s390x

图像

- `cp.icr.io/cp/ibm-mqadvanced-server-integration:9.2.0.5-r1-eus`

已更改的内容

- 包含 IBM MQ 9.2.0 Fix Pack 5。 有关更多信息，请参阅 IBM MQ 9.2.0 Fix Pack 5 中的更改内容和 IBM MQ V 9.2 LTS 的修订列表
- 基于 Red Hat Universal Base Image 8.5-240.1648458092

9.2.0.4-r1-eus

EUS

必需的操作程序版本
1.3.2 和未来修订包

受支持的体系结构
amd64, s390x

图像

- `cp.icr.io/cp/ibm-mqadvanced-server-integration:9.2.0.4-r1-eus`

已更改的内容

- 包含 IBM MQ 9.2.0 Fix Pack 4。 有关更多信息，请参阅 IBM MQ 9.2.0 Fix Pack 4 中的更改内容和 IBM MQ V 9.2 LTS 的修订列表
- 基于 Red Hat Universal Base Image 8.5-204

9.2.0.2-r2-eus

EUS

必需的操作程序版本
1.6.0 或更高版本

受支持的体系结构
amd64, s390x

图像

- `cp.icr.io/cp/ibm-mqadvanced-server-integration:9.2.0.2-r2-eus`

已更改的内容

- 修复了 IBM Cloud Pak foundational services 3.7 和更高版本的单点登录问题，仅当从 EUS 发行版迁移到 CD 发行版时才需要此问题。
- 基于 Red Hat Universal Base Image 8.4-200.1622548483

9.2.0.2-r1-eus

EUS

必需的操作程序版本
1.3.1 和未来修订包 ; 1.6.0 或更高版本

受支持的体系结构
amd64, s390x

图像

- `cp.icr.io/cp/ibm-mqadvanced-server-integration:9.2.0.2-r1-eus`

已更改的内容

- Operations Dashboard 集成使用跟踪代理程序和收集器 V 1.0.8
- 包含 IBM MQ 9.2.0 Fix Pack 2。 有关更多信息，请参阅 IBM MQ 9.2.0 Fix Pack 2 中的更改内容和 IBM MQ V 9.2 LTS 的修订列表
- 基于 Red Hat Universal Base Image 8.4-200.1622548483

9.2.0.1-r1-eus

EUS

必需的操作程序版本
1.3.0 或更高版本

受支持的体系结构
amd64, s390x

图像

- `cp.icr.io/cp/ibm-mqadvanced-server-integration:9.2.0.1-r1-eus`

新增内容

- 仅当使用 IBM Cloud Pak for Integration 许可证时可用
- 在 Red Hat OpenShift Container Platform 4.6 上使用 IBM MQ Operator 1.3.x 和 IBM Common Services 3.6 时，提供了扩展更新支持 (EUS)

已更改的内容

- 包含 IBM MQ 9.2.0 Fix Pack 1。 有关更多信息，请参阅 IBM MQ 9.2.0 Fix Pack 1 中的更改内容和 IBM MQ V 9.2 LTS 的修订列表
- 基于 Red Hat Universal Base Image 8.3-201
- 修复了在允许特权升级的 `SecurityContextContraints` 下运行时，活动性探测器 (`chkmqhealthy`) 和就绪性探测器 (`chkmqready`) 的问题。

9.2.0.0-r3

CD

必需的操作程序版本
1.5.0 或更高版本

受支持的体系结构
amd64, s390x

图像

- `cp.icr.io/cp/ibm-mqadvanced-server-integration:9.2.0.0-r3`
- `cp.icr.io/cp/ibm-mqadvanced-server:9.2.0.0-r3`
- `docker.io/ibmcom/mq:9.2.0.0-r3`

已更改的内容

- 基于 Red Hat Universal Base Image 8.3-291

9.2.0.0-r2

CD

必需的操作程序版本
1.2.0 或更高版本

受支持的体系结构
amd64, s390x

图像

- `cp.icr.io/cp/ibm-mqadvanced-server-integration:9.2.0.0-r2`
- `cp.icr.io/cp/ibm-mqadvanced-server:9.2.0.0-r2`
- `docker.io/ibmcom/mq:9.2.0.0-r2`

新增内容

- 现在可在 z/Linux 上使用

已更改的内容

- 基于 Red Hat Universal Base Image 8.2-349

9.2.0.0-r1

CD

必需的操作程序版本
1.1.0 或更高版本

受支持的体系结构
amd64

图像

- `cp.icr.io/cp/ibm-mqadvanced-server-integration:9.2.0.0-r1-amd64`
- `cp.icr.io/cp/ibm-mqadvanced-server:9.2.0.0-r1-amd64`
- `docker.io/ibmcom/mq:9.2.0.0-r1`

新增内容

- IBM MQ 9.2.0 中的新增功能

已更改的内容

- IBM MQ 9.2.0 中更改的内容
- 将 `-ic` 自变量用于 `crtmqm` 以自动应用 MQSC 文件。 替换先前使用的 `runmqsc` 命令
- 基于 Red Hat 通用基本映像 8.2-301.1593113563

9.1.5.0-r2

CD

必需的操作程序版本
1.0.0 或更高版本

受支持的体系结构
amd64

图像

- `cp.icr.io/cp/ibm-mqadvanced-server-integration:9.1.5.0-r2-amd64`
- `cp.icr.io/cp/ibm-mqadvanced-server:9.1.5.0-r2-amd64`
- `docker.io/ibmcom/mq:9.1.5.0-r2`

已更改的内容

- 基于 Red Hat 通用基本映像 8.2-267

V 9.2.1 OpenShift CD EUS 将 IBM MQ 迁移到 IBM Cloud Pak for Integration

此组主题描述了使用 IBM Cloud Pak for Integration 中的 IBM MQ Operator 将现有 IBM MQ 队列管理器迁移到容器环境的关键步骤。

关于此任务

可以将在 Red Hat OpenShift 上部署 IBM MQ 的客户机分为以下方案:

1. 在 Red Hat OpenShift 中为新应用程序创建新的 IBM MQ 部署。
2. 针对 Red Hat OpenShift 中的新应用程序将 IBM MQ 网络扩展至 Red Hat OpenShift 。
3. 将 IBM MQ 部署移至 Red Hat OpenShift 以继续支持现有应用程序。

仅适用于需要迁移 IBM MQ 配置的方案 3。 其他方案被视为新部署。

这组主题重点介绍了场景 3 ，并描述了使用 IBM MQ Operator 将现有 IBM MQ 队列管理器迁移到容器环境中的关键步骤。 由于 IBM MQ 的灵活性和广泛使用，因此有几个可选步骤。 其中每个都包含一个 "我是否需要执行此操作" 部分。 在迁移期间，验证您的需求应可节省您的时间。

您还需要考虑要迁移哪些数据:

1. 迁移具有相同配置但没有任何现有排队消息的 IBM MQ 。
2. 迁移具有相同配置和现有消息的 IBM MQ 。

典型版本到版本迁移可以使用任一方法。 在迁移点的典型 IBM MQ 队列管理器中，如果队列上存储了任何消息，那么几乎没有任何消息，这使选项 1 适合于许多情况。 在迁移到容器平台的情况下，更常见的方法是使用选项 1 ，以降低迁移的复杂性并允许进行蓝色绿色部署。 因此，指示信息将重点放在此场景上。

此方案的目标是在容器环境中创建与现有队列管理器的定义相匹配的队列管理器。 这允许仅将现有网络连接的应用程序重新配置为指向新的队列管理器，而不更改任何其他配置或应用程序逻辑。

在整个迁移过程中，您将生成多个要应用于新队列管理器的配置文件。 要简化这些文件的管理，您应该创建一个目录并将它们生成到该目录中。

过程

1. 第 34 页的『正在检查必需的功能是否可用』
2. 第 35 页的『抽取队列管理器配置』
3. 可选： 第 35 页的『可选: 抽取和获取队列管理器密钥和证书』
4. 可选： 第 37 页的『可选: 配置 LDAP』
5. 可选： 第 44 页的『可选: 更改 IBM MQ 配置中的 IP 地址和主机名』
6. 第 45 页的『更新容器环境的队列管理器配置』
7. 第 48 页的『为在容器中运行的 IBM MQ 选择目标 HA 体系结构』
8. 第 49 页的『为队列管理器创建资源』
9. 第 50 页的『在 Red Hat OpenShift 上创建新的队列管理器』
10. 第 53 页的『验证新的容器部署』

V 9.2.1 OpenShift CD EUS 正在检查必需的功能是否可用

IBM MQ Operator 不包含 IBM MQ Advanced 中提供的所有功能部件，您必须验证这些功能部件是否是必需的。 其他功能部件部分受支持，可以进行重新配置以与容器中可用的功能部件相匹配。

开始之前

这是 第 33 页的『将 IBM MQ 迁移到 IBM Cloud Pak for Integration』中的第一步。

过程

1. 验证目标容器映像是否包含所需的所有功能。

   有关最新信息，请参阅 第 5 页的『选择要在容器中使用 IBM MQ 的方式』。
2. IBM MQ Operator 具有单个 IBM MQ 流量端口，称为侦听器。 如果您有多个侦听器，请将其简化为在容器中使用单个侦听器。 由于这不是常见场景，因此未详细记录此修改。

3. 如果使用 IBM MQ 出口，请通过在 IBM MQ 出口二进制文件中分层将其迁移到容器中。 这是高级迁移方案，因此此处不包含此方案。 有关步骤的大纲，请参阅 第 101 页的『使用 Red Hat OpenShift CLI 构建具有定制 MQSC 和 INI 文件的映像』。
4. 如果 IBM MQ 系统包含高可用性，请查看可用选项。

   请参阅 第 14 页的『容器中 IBM MQ 的高可用性』。

下一步做什么

现在，您已准备好 抽取队列管理器配置。

V 9.2.1 OpenShift CD EUS 抽取队列管理器配置

大多数配置在队列管理器之间可移植。 例如，应用程序与之交互的内容，例如队列，主题和通道的定义。 使用此任务从现有 IBM MQ 队列管理器中抽取配置。

开始之前

此任务假定您已 检查必需的功能是否可用。

过程

1. 使用现有 IBM MQ 安装登录到机器。
2. 备份配置。

   运行以下命令：

   ```
   dmpmqcfg -m QMGR_NAME > /tmp/backup.mqsc
   ```

   此命令的用法说明：

   - 此命令将备份存储在 `tmp` 目录中。 您可以将备份存储在另一位置，但此方案假定 `tmp` 目录用于后续命令。
   - 将 *QMGR_NAME* 替换为环境中的队列管理器名称。 如果不确定该值，请运行 **dspmq** 命令以查看机器上的可用队列管理器。 以下是名为 `qm1` 的队列管理器的样本 **dspmq** 命令输出：

     ```
     QMNAME(qm1)                   STATUS(Running)
     ```

     dspmq 命令要求启动 IBM MQ 队列管理器，否则您将收到以下错误：

     ```
     AMQ8146E: IBM MQ queue manager not available.
     ```

     如果需要，通过运行以下命令来启动队列管理器：

     ```
     strmqm QMGR_NAME
     ```

下一步做什么

现在，您已准备好 抽取和获取队列管理器密钥和证书。

V 9.2.1 OpenShift CD EUS 可选: 抽取和获取队列管理器密钥和证书

可以使用 TLS 配置 IBM MQ ， 以加密进入队列管理器的流量。 使用此任务来验证队列管理器是否正在使用 TLS ， 抽取密钥和证书以及在已迁移的队列管理器上配置 TLS。

开始之前

此任务假定您已 抽取队列管理器配置。

关于此任务

我需要执行此操作吗?

可以将 IBM MQ 配置为对进入队列管理器的流量进行加密。 此加密是使用队列管理器上配置的密钥存储库完成的。 然后， IBM MQ 通道将启用 TLS 通信。 如果您不确定是否在环境中配置了此参数，请运行以下命令以进行验证:

```
grep 'SECCOMM(ALL\|SECCOMM(ANON\|SSLCIPH' backup.mqsc
```

如果找不到任何结果，那么不会使用 TLS。 但是，这并不意味着不应在迁移的队列管理器中配置 TLS。 您可能希望更改此行为的原因有以下几个:

- 与先前环境相比，应该增强 Red Hat OpenShift 环境上的安全方法。
- 如果需要从 Red Hat OpenShift 环境外部访问迁移的队列管理器，那么需要 TLS 才能通过 Red Hat OpenShift 路由。

过程

1. 从现有存储库中抽取任何可信证书。

   如果队列管理器上当前正在使用 TLS ，那么队列管理器可能存储了大量可信证书。 需要将这些内容抽取并复制到新的队列管理器。 完成下列其中一个可选步骤:

   - 要简化证书的抽取，请在本地系统上运行以下脚本:

     ```
     #!/bin/bash

     keyr=$(grep SSLKEYR $1)
     if [ -n "${keyr}" ]; then
         keyrlocation=$(sed -n "s/^.*'\(.*\)'.*$/\1/ p" <<< ${keyr})
         mapfile -t runmqckmResult < <(runmqckm -cert -list -db ${keyrlocation}.kdb -stashed)
             cert=1
         for i in "${runmqckmResult[@]:1}"
             do
                certlabel=$(echo ${i} | xargs)
                echo Extracting certificate $certlabel to $cert.cert
            runmqckm -cert -extract -db ${keyrlocation}.kdb -label "$certlabel" -target $
     {cert}.cert -stashed
            cert=$[$cert+1]
             done
     fi
     ```

     运行脚本时，将 IBM MQ 备份的位置指定为自变量，并抽取证书。 例如，如果脚本名为 `extractCert.sh` ，并且 IBM MQ 备份位于 `/tmp/backup.mqsc` ，请运行以下命令:

     ```
     extractCert.sh /tmp/backup.mqsc
     ```

   - 或者，按显示的顺序运行以下命令:

     a. 标识 TLS 商店的位置:

     ```
     grep SSLKEYR /tmp/backup.mqsc
     ```

     样本输出:

     ```
     SSLKEYR('/run/runmqserver/tls/key') +
     ```

     其中密钥库位于 `/run/runmqserver/tls/key.kdb`

     b. 根据此位置信息，查询密钥库以确定存储的证书:

     ```
     runmqckm -cert -list -db /run/runmqserver/tls/key.kdb -stashed
     ```

     样本输出:

     ```
     Certificates in database /run/runmqserver/tls/key.kdb:
               default
               CN=cs-ca-certificate,O=cert-manager
     ```

     c. 抽取列出的每个证书。 通过运行以下命令来执行此操作:

```
runmqckm -cert -extract -db KEYSTORE_LOCATION -label "LABEL_NAME" -target OUTPUT_FILE
-stashed
```

在先前显示的样本中，这等同于以下内容:

```
runmqckm -cert -extract -db /run/runmqserver/tls/key.kdb -label "CN=cs-ca-
certificate,O=cert-manager" -target /tmp/cert-manager.crt -stashed
runmqckm -cert -extract -db /run/runmqserver/tls/key.kdb -label "default" -target /tmp/
default.crt -stashed
```

2. 获取队列管理器的新密钥和证书

   要在迁移的队列管理器上配置 TLS ， 请生成新的密钥和证书。 然后在部署期间使用此参数。 在许多组织中，这意味着联系您的安全团队以请求密钥和证书。 在某些组织中，此选项不可用，并且将使用自签名证书。

   以下示例生成到期设置为 10 年的自签名证书:

```
openssl req \
       -newkey rsa:2048 -nodes -keyout qmgr.key \
       -subj "/CN=mq queuemanager/OU=ibm mq" \
       -x509 -days 3650 -out qmgr.crt
```

   将创建两个新文件:

   - `qmgr.key` 是队列管理器的专用密钥
   - `qmgr.crt` 是公用证书

下一步做什么

现在，您已准备好 配置 LDAP。

V 9.2.1 OpenShift CD EUS 可选: 配置 LDAP

可以将 IBM MQ Operator 配置为使用多种不同的安全方法。 通常， LDAP 对于企业部署最有效，而 LDAP 用于此迁移方案。

开始之前

此任务假定您已 抽取并获取队列管理器密钥和证书。

关于此任务

我需要执行此操作吗?

如果您已在使用 LDAP 进行认证和授权，那么不需要进行任何更改。

如果不确定是否正在使用 LDAP ， 请运行以下命令:

```
connauthname="$(grep CONNAUTH backup.mqsc | cut -d "(" -f2 | cut -d ")" -f1)"; grep -A 20
AUTHINFO\($connauthname\) backup.mqsc
```

样本输出:

```
DEFINE AUTHINFO('USE.LDAP') +
   AUTHTYPE(IDPWLDAP) +
   ADOPTCTX(YES) +
   CONNAME('ldap-service.ldap(389)') +
   CHCKCLNT(REQUIRED) +
   CLASSGRP('groupOfUniqueNames') +
   FINDGRP('uniqueMember') +
   BASEDNG('ou=groups,dc=ibm,dc=com') +
   BASEDNU('ou=people,dc=ibm,dc=com') +
   LDAPUSER('cn=admin,dc=ibm,dc=com') +
*  LDAPPWD('********************************') +
   SHORTUSR('uid') +
   GRPFIELD('cn') +
   USRFIELD('uid') +
```

```
    AUTHORMD(SEARCHGRP) +
 *  ALTDATE(2020-11-26) +
 *  ALTTIME(15.44.38) +
    REPLACE
```

输出中有两个特别重要的属性:

AUTHTYPE
如果此值为 IDPWLDAP，那么您将使用 LDAP 进行认证。

如果该值为空或其他值，那么不会配置 LDAP。 在这种情况下，请检查 AUTHORMD 属性以查看是否正在使用 LDAP 用户进行授权。

AUTHORMD
如果此值为 OS，那么表示您未使用 LDAP 进行授权。

要修改授权和认证以使用 LDAP ，请完成以下任务:

过程

1. 更新 LDAP 服务器的 IBM MQ 备份。
2. 更新 IBM MQ 备份以获取 LDAP 授权信息。

V 9.2.1 OpenShift CD EUS *LDAP 部件 1:* 更新 *LDAP* 服务器的 *IBM MQ* 备份

有关如何设置 LDAP 的全面描述不在此方案的范围内。 本主题提供了过程的摘要，样本以及对进一步信息的引用。

开始之前

此任务假定您已 抽取并获取队列管理器密钥和证书。

关于此任务

我需要执行此操作吗?

如果您已在使用 LDAP 进行认证和授权，那么不需要进行任何更改。 如果不确定是否正在使用 LDAP ，请参阅 第 37 页的『可选: 配置 LDAP』。

设置 LDAP 服务器有两个部分:

1. 定义 LDAP 配置。
2. 将 LDAP 配置与队列管理器定义相关联。

有关帮助您执行此配置的更多信息:

- 用户存储库概述
- AUTHINFO 命令参考指南

过程

1. 定义 LDAP 配置。

   编辑 backup.mqsc 文件以定义 LDAP 系统的新 **AUTHINFO** 对象。 例如:

```
DEFINE AUTHINFO(USE.LDAP) +
   AUTHTYPE(IDPWLDAP) +
   CONNAME('ldap-service.ldap(389)') +
   LDAPUSER('cn=admin,dc=ibm,dc=com') +
   LDAPPWD('admin') +
   SECCOMM(NO) +
   USRFIELD('uid') +
   SHORTUSR('uid') +
   BASEDNU('ou=people,dc=ibm,dc=com') +
   AUTHORMD(SEARCHGRP) +
   BASEDNG('ou=groups,dc=ibm,dc=com') +
   GRPFIELD('cn') +
```

```
    CLASSGRP('groupOfUniqueNames') +
    FINDGRP('uniqueMember')
    REPLACE
```

其中:

- **CONNAME** 是对应于 LDAP 服务器的主机名和端口。 如果存在多个弹性地址，那么可以使用逗号分隔列表来配置这些地址。
- **LDAPUSER** 是与 IBM MQ 在连接到 LDAP 以查询用户记录时使用的用户对应的专有名称。
- **LDAPPWD** 是对应于 **LDAPUSER** 用户的密码。
- **SECCOM** 指定与 LDAP 服务器的通信是否应使用 TLS。 可能的值为:
  - YES: 使用 TLS 并由 IBM MQ 服务器提供证书。
  - ANON: 使用 TLS 时， IBM MQ 服务器不会提供证书。
  - NO: 在连接期间不使用 TLS。
- **USRFIELD** 指定 LDAP 记录中要与提供的用户名匹配的字段。
- **SHORTUSR** 是 LDAP 记录中长度不超过 12 个字符的字段。 如果认证成功，那么此字段中的值为已断言的身份。
- **BASEDNU** 是应该用于搜索 LDAP 的基本 DN。
- **BASEDNG** 是 LDAP 中组的基本 DN。
- **AUTHORMD** 定义用于解析用户组成员资格的机制。 有四个选项:
  - OS: 查询操作系统以查找与短名称关联的组。
  - SEARCHGRP: 在 LDAP 中搜索已认证用户的组条目。
  - SEARCHUSR: 搜索已认证的用户记录以获取组成员资格信息。
  - SRCHGRPSN: 在 LDAP 中搜索组条目以查找已认证的用户短用户名 (由 SHORTUSR 字段定义)。
- **GRPFIELD** 是 LDAP 组记录中对应于简单名称的属性。 如果指定了此项，那么可以用于定义授权记录。
- **CLASSUSR** 是对应于用户的 LDAP 对象类。
- **CLASSGRP** 是对应于组的 LDAP 对象类。
- **FINDGRP** 是 LDAP 记录中对应于组成员资格的属性。

新条目可以放在文件中的任何位置，但是您可能会发现在文件开头有任何新条目很有用:

```
Open ▾
backup.mqsc
************************************************************
* Script generated on 2020-10-21   at 11.48.32
* Script generated by user ' CallumJackso' on host 'LAPTOP-VLC
* Queue manager name: qm1
* Queue manager platform: Windows
* Queue manager command level: (920/920)
* Command issued: dmpmqcfg -m qm1
************************************************************
DEFINE AUTHINFO(USE.LDAP) +
   AUTHTYPE(IDPWLDAP) +
   CONNAME('ldap-service.ldap(389)') +
   LDAPUSER('cn=admin,dc=ibm,dc=com') +
   LDAPPWD('admin') +
   SECCOMM(NO) +
   USRFIELD('uid') +
   SHORTUSR('uid') +
   BASEDNU('ou=people,dc=ibm,dc=com') +
   AUTHORMD(SEARCHGRP) +
   BASEDNG('ou=groups,dc=ibm,dc=com') +
   GRPFIELD('cn') +
   CLASSGRP('groupOfUniqueNames') +
   FINDGRP('uniqueMember') +
   REPLACE
ALTER QMGR +
*  ALTDATE(2020-10-26) +
*  ALTTIME(11.43.11) +
```

2. 将 LDAP 配置与队列管理器定义相关联。

   您需要将 LDAP 配置与队列管理器定义相关联。 紧跟在 DEFINE AUTHINFO 条目下方的是 ALTER QMGR 条目。 修改 CONNAUTH 条目以对应于新创建的 AUTHINFO 名称。 例如，在先前的示例中定义了 AUTHINFO(USE.LDAP)，这意味着名称为 USE.LDAP。 因此，将 CONNAUTH('SYSTEM.DEFAULT.AUTHINFO.IDPWOS') 更改为 CONNAUTH('USE.LDAP'):

```
Open ▾
backup.mqsc
*******************************************************
* Script generated on 2020-10-21   at 11.48.32
* Script generated by user ' CallumJackso' on host 'L
* Queue manager name: qm1
* Queue manager platform: Windows
* Queue manager command level: (920/920)
* Command issued: dmpmqcfg -m qm1
*******************************************************
DEFINE AUTHINFO(USE.LDAP) +
   AUTHTYPE(IDPWLDAP) +
   CONNAME('ldap-service.ldap(389)') +
   LDAPUSER('cn=admin,dc=ibm,dc=com') +
   LDAPPWD('admin') +
   SECCOMM(NO) +
   USRFIELD('uid') +
   SHORTUSR('uid') +
   BASEDNU('ou=people,dc=ibm,dc=com') +
   AUTHORMD(SEARCHGRP) +
   BASEDNG('ou=groups,dc=ibm,dc=com') +
   GRPFIELD('cn') +
   CLASSGRP('groupOfUniqueNames') +
   FINDGRP('uniqueMember') +
   REPLACE
ALTER QMGR +
*  ALTDATE(2020-10-26) +
*  ALTTIME(11.43.11) +
   CCSID(850) +
   CERTLABL('default') +
   CLWLUSEQ(LOCAL) +
*  COMMANDQ(SYSTEM.ADMIN.COMMAND.QUEUE) +
   CONNAUTH('USE.LDAP') +
```

要立即切换到 LDAP，请在 ALTER QMGR 命令之后立即添加一行来调用 REFRESH SECURITY 命令:

```
*backup.mqsc
************************************************************************
* Script generated on 2020-10-21   at 11.48.32
* Script generated by user ' CallumJackso' on host 'LAPTOP-VLQKJ5UH'
* Queue manager name: qm1
* Queue manager platform: Windows
* Queue manager command level: (920/920)
* Command issued: dmpmqcfg -m qm1
************************************************************************
DEFINE AUTHINFO(USE.LDAP) +
   AUTHTYPE(IDPWLDAP) +
   CONNAME('ldap-service.ldap(389)') +
   LDAPUSER('cn=admin,dc=ibm,dc=com') +
   LDAPPWD('admin') +
   SECCOMM(NO) +
   USRFIELD('uid') +
   SHORTUSR('uid') +
   BASEDNU('ou=people,dc=ibm,dc=com') +
   AUTHORMD(SEARCHGRP) +
   BASEDNG('ou=groups,dc=ibm,dc=com') +
   GRPFIELD('cn') +
   CLASSGRP('groupOfUniqueNames') +
   FINDGRP('uniqueMember') +
   REPLACE
ALTER QMGR +
*  ALTDATE(2020-10-26) +
*  ALTTIME(11.43.11) +
   CCSID(850) +
   CERTLABL('default') +
   CLWLUSEQ(LOCAL) +
*  COMMANDQ(SYSTEM.ADMIN.COMMAND.QUEUE) +
   CONNAUTH('USE.LDAP') +
*  CRDATE(2020-10-26) +
*  CRTIME(11.43.11) +
*  QMID(qm1_2020-10-26_11.43.11) +
   SSLCRYP(' ') +
   SSLKEYR('/run/runmqserver/tls/key') +
   SUITEB(NONE) +
*  VERSION(09020000) +
   FORCE
REFRESH SECURITY
```

下一步做什么

现在，您已准备好 更新 IBM MQ 备份以获取 LDAP 授权信息。

V 9.2.1 OpenShift CD EUS

LDAP 部件 2: 更新 IBM MQ 备份以获取 LDAP 授权信息

IBM MQ 提供了用于控制对 IBM MQ 对象的访问的细颗粒度授权规则。 如果更改了对 LDAP 的认证和授权，那么授权规则可能无效并且需要更新。

开始之前

此任务假定您已 更新 LDAP 服务器的备份。

关于此任务

我需要执行此操作吗?

如果您已在使用 LDAP 进行认证和授权，那么不需要进行任何更改。 如果不确定是否正在使用 LDAP，请参阅 第 37 页的『可选: 配置 LDAP』。

更新 LDAP 授权信息有两个部分:

1. 从文件中除去所有现有授权。
2. 定义 LDAP 的新授权信息。

过程

1. 从文件中除去所有现有权限。

   在靠近文件末尾的备份文件中，您应该会看到几个以 SET AUTHREC 开头的条目:

```
Open ▾                                                    *backup.mqsc
                                                          /tmp
   OBJTYPE(PROCESS) +
   AUTHADD(CRT)
SET AUTHREC +
   PROFILE('@CLASS') +
   PRINCIPAL('CallumJackson@AzureAD') +
   OBJTYPE(QMGR) +
   AUTHADD(CRT)
SET AUTHREC +
   PROFILE('@CLASS') +
   GROUP('mqm@LAPTOP-VLQKJ5UH') +
   OBJTYPE(QMGR) +
   AUTHADD(CRT)
SET AUTHREC +
   PROFILE('SYSTEM.ADMIN.CHANNEL.EVENT') +
   PRINCIPAL('CallumJackson@AzureAD') +
   OBJTYPE(QUEUE) +
   AUTHADD(BROWSE,CHG,CLR,DLT,DSP,GET,INQ,PUT,PASSALL,PASSID,SET,SETALL,SETID)
SET AUTHREC +
   PROFILE('SYSTEM.ADMIN.CHANNEL.EVENT') +
   GROUP('mqm@LAPTOP-VLQKJ5UH') +
   OBJTYPE(QUEUE) +
   AUTHADD(BROWSE,CHG,CLR,DLT,DSP,GET,INQ,PUT,PASSALL,PASSID,SET,SETALL,SETID)
********************************************************************************
* Script ended on 2020-10-26   at 11.48.32
* Number of Inquiry commands issued: 14
* Number of Inquiry commands completed: 14
* Number of Inquiry responses processed: 295
* QueueManager count: 1
* Queue count: 57
* NameList count: 3
* Process count: 1
* Channel count: 11
* AuthInfo count: 4
* Listener count: 4
* Service count: 2
* CommInfo count: 1
* Topic count: 6
* Subscription count: 1
* ChlAuthRec count: 3
* AuthRec count: 199
* Number of objects/records: 293
********************************************************************************
```

   查找现有条目并将其删除。 最简单的方法是除去所有现有 SET AUTHREC 规则，然后根据 LDAP 条目创建新条目。

2. 定义 LDAP 的新授权信息

根据您的队列管理器配置以及资源和组的数量，这可能是一个耗时或简单的活动。 以下示例假定队列管理器只有一个名为 Q1 的队列，并且您希望允许 LDAP 组 apps 具有访问权。

```
SET AUTHREC GROUP('apps') OBJTYPE(QMGR) AUTHADD(ALL)
SET AUTHREC PROFILE('Q1') GROUP('apps') OBJTYPE(QUEUE) AUTHADD(ALL)
```

第一个 AUTHREC 命令添加访问队列管理器的许可权，第二个命令提供对队列的访问权。 如果需要访问第二个队列，那么需要第三个 AUTHREC 命令，除非您决定使用通配符来提供更通用的访问权。

以下是另一个示例。 如果管理员组 (称为 admins) 需要对队列管理器的完全访问权，请添加以下命令:

```
SET AUTHREC PROFILE('*') OBJTYPE(QUEUE) GROUP('admins') AUTHADD(ALL)
SET AUTHREC PROFILE('*') OBJTYPE(TOPIC) GROUP('admins') AUTHADD(ALL)
SET AUTHREC PROFILE('*') OBJTYPE(CHANNEL) GROUP('admins') AUTHADD(ALL)
SET AUTHREC PROFILE('*') OBJTYPE(CLNTCONN) GROUP('admins') AUTHADD(ALL)
SET AUTHREC PROFILE('*') OBJTYPE(AUTHINFO) GROUP('admins') AUTHADD(ALL)
SET AUTHREC PROFILE('*') OBJTYPE(LISTENER) GROUP('admins') AUTHADD(ALL)
SET AUTHREC PROFILE('*') OBJTYPE(NAMELIST) GROUP('admins') AUTHADD(ALL)
SET AUTHREC PROFILE('*') OBJTYPE(PROCESS) GROUP('admins') AUTHADD(ALL)
SET AUTHREC PROFILE('*') OBJTYPE(SERVICE) GROUP('admins') AUTHADD(ALL)
SET AUTHREC PROFILE('*') OBJTYPE(QMGR) GROUP('admins') AUTHADD(ALL)
```

下一步做什么

现在，您已准备好 更改 IBM MQ 配置中的 IP 地址和主机名。

V 9.2.1 OpenShift CD EUS 可选: 更改 IBM MQ 配置中的 IP 地址和主机名

IBM MQ 配置可能指定了 IP 地址和主机名。 在某些情况下，可以保留这些内容，而在其他情况下，需要更新这些内容。

开始之前

此任务假定您已 配置 LDAP。

关于此任务

我需要执行此操作吗?

首先，确定是否指定了除上一节中定义的 LDAP 配置以外的任何 IP 地址或主机名。 要执行此操作，请运行以下命令:

```
grep 'CONNAME\|LOCLADDR\|IPADDRV' -B 3  backup.mqsc
```

样本输出:

```
*****************************************************
DEFINE AUTHINFO(USE.LDAP) +
   AUTHTYPE(IDPWLDAP) +
   CONNAME('ldap-service.ldap(389)') +
--
DEFINE AUTHINFO('SYSTEM.DEFAULT.AUTHINFO.IDPWLDAP') +
   AUTHTYPE(IDPWLDAP) +
   ADOPTCTX(YES) +
   CONNAME(' ') +
--
   REPLACE
DEFINE AUTHINFO('SYSTEM.DEFAULT.AUTHINFO.CRLLDAP') +
   AUTHTYPE(CRLLDAP) +
   CONNAME(' ') +
```

在此示例中，搜索将返回三个结果。 一个结果对应于先前定义的 LDAP 配置。 这可以忽略，因为 LDAP 服务器的主机名保持不变。 其他两个结果是空的连接条目，因此也可以忽略这些条目。 如果您没有任何其他条目，那么可以跳过本主题的其余部分。

过程

1. 了解返回的条目。

   IBM MQ 可以在配置的许多方面中包含 IP 地址，主机名和端口。 我们可以将这些分类为两类:

   a. **此队列管理器的位置**: 此队列管理器使用或发布的位置信息， IBM MQ 网络中的其他队列管理器或应用程序可用于连接。

   b. **队列管理器依赖关系的位置**: 此队列管理器需要识别的其他队列管理器或系统的位置。

   由于此场景仅关注对此队列管理器配置的更改，因此我们仅处理类别 (a) 的配置更新。 但是，如果此队列管理器位置由其他队列管理器或应用程序引用，那么它们的配置可能需要更新以与此队列管理器的新位置匹配。

   有两个关键对象可能包含需要更新的信息:

   - 侦听器: 这些表示 IBM MQ 正在侦听的网络地址。
   - 集群接收方通道: 如果队列管理器是 IBM MQ 集群的一部分，那么此对象存在。 它指定其他队列管理器可连接到的网络地址。

2. 在 `grep 'CONNAME\|LOCLADDR\|IPADDRV' -B 3 backup.mqsc` 命令的原始输出中，确定是否定义了任何 CLUSTER RECEIVER 通道。 如果是这样，请更新 IP 地址。

   要确定是否定义了任何 CLUSTER RECEIVER 通道，请在原始输出中查找具有 CHLTYPE (CLUSRCVR) 的任何条目:

   ```
   DEFINE CHANNEL(ANY_NAME) +
      CHLTYPE(CLUSRCVR) +
   ```

   如果存在条目，请使用 IBM MQ Red Hat OpenShift 路由更新 `CONNAME` 。 此值基于 Red Hat OpenShift 环境并使用可预测的语法:

   ```
   queue_manager_resource_name-ibm-mq-qm-openshift_project_name.openshift_app_route_hostname
   ```

   例如，如果队列管理器部署在 `cp4i` 名称空间中名为 `qm1` ，并且 *openshift_app_route_hostname* 为 `apps.callumj.icp4i.com`，那么路径 URL 为:

   ```
   qm1-ibm-mq-qm-cp4i.apps.callumj.icp4i.com
   ```

   路由的端口号通常为 443。 除非 Red Hat OpenShift 管理员以不同方式告诉您，否则这通常是正确的值。 使用此信息，更新 `CONNAME` 字段。 例如:

   ```
   CONNAME('qm1-ibm-mq-qm-cp4i.apps.callumj.icp4i.com(443)')
   ```

   在 `grep 'CONNAME\|LOCLADDR\|IPADDRV' -B 3 backup.mqsc` 命令的原始输出中，验证 `LOCLADDR` 或 `IPADDRV` 是否存在任何条目。 如果有，请将其删除。 它们在容器环境中不相关。

下一步做什么

现在，您已准备好 更新容器环境的队列管理器配置。

V 9.2.1 OpenShift CD EUS 更新容器环境的队列管理器配置

在容器中运行时，某些配置方面由容器定义，并且可能与导出的配置冲突。

开始之前

此任务假定您 已更改 IP 地址和主机名的 IBM MQ 配置。

关于此任务

容器定义了以下配置方面:

- 侦听器定义 (对应于公开的端口)。

- 任何潜在 TLS 商店的位置。

因此，您需要更新导出的配置:

1. 除去任何侦听器定义。
2. 定义 TLS 密钥存储库的位置。

过程

1. 除去任何侦听器定义。

   在备份配置中，搜索 DEFINE LISTENER。 这应该介于 AUTHINFO 和 SERVICE 定义之间。 突出显示该区域，然后将其删除。

```
*backup.mqsc
*   ALTDATE(2020-11-26) +
*   ALTTIME(11.43.28) +
    REPLACE
DEFINE AUTHINFO('SYSTEM.DEFAULT.AUTHINFO.CRLLDAP') +
    AUTHTYPE(CRLLDAP) +
    CONNAME(' ') +
*   ALTDATE(2020-10-26) +
*   ALTTIME(11.43.28) +
    REPLACE
DEFINE LISTENER('SYSTEM.DEFAULT.LISTENER.LU62') +
    TRPTYPE(LU62) +
    CONTROL(MANUAL) +
*   ALTDATE(2020-10-26) +
*   ALTTIME(11.43.28) +
    REPLACE
DEFINE LISTENER('SYSTEM.DEFAULT.LISTENER.NETBIOS') +
    TRPTYPE(NETBIOS) +
    CONTROL(MANUAL) +
    LOCLNAME(' ') +
*   ALTDATE(2020-10-26) +
*   ALTTIME(11.43.28) +
    REPLACE
DEFINE LISTENER('SYSTEM.DEFAULT.LISTENER.SPX') +
    TRPTYPE(SPX) +
    CONTROL(MANUAL) +
*   ALTDATE(2020-10-26) +
*   ALTTIME(11.43.28) +
    REPLACE
DEFINE LISTENER('SYSTEM.DEFAULT.LISTENER.TCP') +
    TRPTYPE(TCP) +
    CONTROL(MANUAL) +
*   ALTDATE(2020-10-26) +
*   ALTTIME(11.43.28) +
    REPLACE
DEFINE SERVICE('SYSTEM.AMQP.SERVICE') +
    CONTROL(QMGR) +
    SERVTYPE(SERVER) +
    STARTCMD('+MQ_INSTALL_PATH+\bin\amqp.bat') +
    STARTARG('start -m +QMNAME+ -d "+MQ_Q_MGR_DATA_PATH+\."
    STOPCMD('+MQ_INSTALL_PATH+\bin64\endmqsde.exe') +
```

2. 定义 TLS 密钥存储库的位置。

   队列管理器备份包含原始环境的 TLS 配置。 这与容器环境不同，因此需要进行一些更新:

   - 将 **CERTLABL** 条目更改为 `default`
   - 将 TLS 密钥存储库 (**SSLKEYR**) 的位置更改为: `/run/runmqserver/tls/key`

   要在文件中查找 **SSLKEYR** 属性的位置，请搜索 **SSLKEYR**。 通常只找到一个条目。 如果找到多个条目，请检查您是否正在编辑 **QMGR** 对象，如下图所示:

```
*backup.mqsc
************************************************************************
* Script generated on 2020-10-21   at 11.48.32
* Script generated by user ' CallumJackso' on host 'LAPTOP-VLQKJ5UH'
* Queue manager name: qm1
* Queue manager platform: Windows
* Queue manager command level: (920/920)
* Command issued: dmpmqcfg -m qm1
************************************************************************
DEFINE AUTHINFO(USE.LDAP) +
   AUTHTYPE(IDPWLDAP) +
   CONNAME('ldap-service.ldap(389)') +
   LDAPUSER('cn=admin,dc=ibm,dc=com') +
   LDAPPWD('admin') +
   SECCOMM(NO) +
   USRFIELD('uid') +
   SHORTUSR('uid') +
   BASEDNU('ou=people,dc=ibm,dc=com') +
   AUTHORMD(SEARCHGRP) +
   BASEDNG('ou=groups,dc=ibm,dc=com') +
   GRPFIELD('cn') +
   CLASSGRP('groupOfUniqueNames') +
   FINDGRP('uniqueMember') +
   REPLACE
ALTER QMGR +
*  ALTDATE(2020-10-26) +
*  ALTTIME(11.43.11) +
   CCSID(850) +
   CERTLABL('default') +
   CLWLUSEQ(LOCAL) +
*  COMMANDQ(SYSTEM.ADMIN.COMMAND.QUEUE) +
   CONNAUTH('USE.LDAP') +
*  CRDATE(2020-10-26) +
*  CRTIME(11.43.11) +
*  QMID(qm1_2020-10-26_11.43.11) +
   SSLCRYP(' ') +
   SSLKEYR('/run/runmqserver/tls/key') +
   SUITEB(NONE) +
*  VERSION(09020000) +
   FORCE
REFRESH SECURITY
```

下一步做什么

现在，您已准备好 选择在容器中运行的 IBM MQ 的目标体系结构。

V 9.2.1 OpenShift CD EUS 为在容器中运行的 IBM MQ 选择目标 HA 体系结构

在单个实例 (单个 Kubernetes Pod) 和多个实例 (两个 Pod) 之间进行选择，以满足您的高可用性需求。

开始之前

此任务假定您已 更新容器环境的队列管理器配置。

关于此任务

IBM MQ Operator 提供了两个高可用性选项:

- **单个实例**: 将启动单个容器 (Pod) ，并且 Red Hat OpenShift 负责在发生故障时重新启动。 由于 Kubernetes 中有状态集的特征，在某些情况下，此故障转移可能需要较长时间，或者需要完成管理操作。
- **多实例**: 启动两个容器 (每个容器位于单独的 Pod 中) ，一个处于活动方式，另一个处于备用状态。 此拓扑支持更快的故障转移。 它需要符合 IBM MQ 要求的 "多读" 文件系统。

在此任务中，您仅选择目标 HA 体系结构。 在此场景中的后续任务 (第 50 页的『在 Red Hat OpenShift 上创建新的队列管理器』) 中描述了用于配置所选体系结构的步骤。

过程

1. 请查看这两个选项。

   有关这两个选项的全面描述，请参阅 第 14 页的『容器中 IBM MQ 的高可用性』。
2. 选择目标 HA 体系结构。

   如果不确定要选择哪个选项，请从 **单个实例** 选项开始，并验证这是否满足高可用性需求。

下一步做什么

现在，您已准备好 创建队列管理器资源。

V 9.2.1 OpenShift CD EUS 为队列管理器创建资源

将 IBM MQ 配置以及 TLS 证书和密钥导入到 Red Hat OpenShift 环境中。

开始之前

此任务假定您已 选择在容器中运行的 IBM MQ 的目标体系结构。

关于此任务

在前面的部分中，您已抽取，更新和定义了两个资源:

- IBM MQ 配置
- TLS 证书和密钥

在部署队列管理器之前，需要将这些资源导入到 Red Hat OpenShift 环境中。

过程

1. 将 IBM MQ 配置导入到 Red Hat OpenShift 中。

   以下指示信息假定您在名为 `backup.mqsc` 的文件中的当前目录中具有 IBM MQ 配置。 否则，需要根据您的环境定制文件名。

   a) 使用 `oc login` 登录到集群。

   b) 将 IBM MQ 配置装入到 `configmap` 中。

   运行以下命令:

   ```
   oc create configmap my-mqsc-migrated --from-file=backup.mqsc
   ```

   c) 验证是否已成功装入该文件。

   运行以下命令:

   ```
   oc describe configmap my-mqsc-migrated
   ```
2. 导入 IBM MQ TLS 资源

如 第 35 页的『可选: 抽取和获取队列管理器密钥和证书』中所述，队列管理器部署可能需要 TLS。 如果是这样，您应该已经有许多以 `.crt` 和 `.key` 结尾的文件。 您需要将这些内容添加到 Kubernetes 私钥中，以供队列管理器在部署时引用。

例如，如果您具有队列管理器的密钥和证书，那么可以调用这些密钥和证书:

- `qmgr.crt`
- `qmgr.key`

要导入这些文件，请运行以下命令:

```
oc create secret tls my-tls-migration --cert=qmgr.crt --key=qmgr.key
```

在导入匹配的公用密钥和专用密钥时， Kubernetes 提供了此有用的实用程序。 如果要添加其他证书，例如要添加到队列管理器信任库中，请运行以下命令:

```
oc create secret generic my-extra-tls-migration --from-file=comma_separated_list_of_files
```

例如，如果要导入的文件是 `trust1.crt`， `trust2.crt` 和 `trust3.crt`，那么命令如下所示:

```
oc create secret generic my-extra-tls-migration --from-file=trust1.crt,trust2.crt,trust3.crt
```

下一步做什么

现在，您已准备好 在 Red Hat OpenShift 上创建新的队列管理器。

V 9.2.1 OpenShift CD EUS 在 Red Hat OpenShift 上创建新的队列管理器

在 Red Hat OpenShift 上部署单个实例或多实例队列管理器。

开始之前

此任务假定您已 创建队列管理器资源，并且 已将 IBM MQ Operator 安装到 Red Hat OpenShift 中。

关于此任务

如 第 48 页的『为在容器中运行的 IBM MQ 选择目标 HA 体系结构』中所述，有两种可能的部署拓扑。 因此，本主题提供了两个不同的模板:

- 部署单个实例队列管理器。
- 部署多实例队列管理器。

要点: 根据您的首选拓扑，仅完成两个模板中的一个。

过程

- 部署单个实例队列管理器。

  已迁移的队列管理器将使用 YAML 文件部署到 Red Hat OpenShift 。 以下是基于先前主题中使用的名称的样本:

```
apiVersion: mq.ibm.com/v1beta1
kind: QueueManager
metadata:
  name: qm1
spec:
  version: 9.2.5.0-r3
  license:
    accept: true
    license: L-RJON-C7QG3S
    use: "Production"
  pki:
    keys:
```

```
    - name: default
      secret:
        secretName: my-tls-migration
        items:
          - tls.key
          - tls.crt
  web:
    enabled: true
  queueManager:
    name: QM1
    mqsc:
      - configMap:
          name: my-mqsc-migrated
          items:
            - backup.mqsc
```

根据您执行的步骤，可能需要定制先前的 YAML。 为了帮助您实现此目的，以下是对此 YAML 的说明:

```
apiVersion: mq.ibm.com/v1beta1
kind: QueueManager
metadata:
  name: qm1
```

这将定义 Kubernetes 对象，类型和名称。 唯一需要定制的字段是 `name` 字段。

```
 spec:
  version: 9.2.5.0-r3
  license:
    accept: true
    license: L-RJON-C7QG3S
    use: "Production"
```

这对应于部署的版本和许可证信息。 如果需要对此进行定制，请使用 第 114 页的『mq.ibm.com/v1beta1 的许可参考』中提供的信息。

```
  pki:
    keys:
    - name: default
      secret:
        secretName: my-tls-migration
        items:
          - tls.key
          - tls.crt
```

要将队列管理器配置为使用 TLS ，它必须引用相关证书和密钥。 `secretName` 字段引用在 导入 IBM MQ TLS 资源 部分中创建的 Kubernetes 私钥，并且项列表 (`tls.key` 和 `tls.crt`) 是 Kubernetes 使用 `oc create secret tls` 语法时指定的标准名称。 如果要将其他证书添加到信任库中，那么可以通过类似方式添加这些证书，但这些项是导入期间使用的相应文件名。 例如，可以使用以下代码来创建信任库证书:

```
oc create secret generic my-extra-tls-migration --from-file=trust1.crt,trust2.crt,trust3.crt
```

```
  pki:
    trust:
    - name: default
      secret:
        secretName: my-extra-tls-migration
        items:
          - trust1.crt
          - trust2.crt
          - trust3.crt
```

要点: 如果不需要 TLS ，请删除 YAML 的 TLS 部分。

```
  web:
    enabled: true
```

这将为部署启用 Web 控制台

```
  queueManager:
    name: QM1
```

这将队列管理器的名称定义为 QM1。 队列管理器根据您的需求 (例如，原始队列管理器名称) 进行定制。

```
    mqsc:
      - configMap:
          name: my-mqsc-migrated
          items:
            - backup.mqsc
```

先前的代码会拉入在 导入 IBM MQ 配置 部分中导入的队列管理器配置。 如果使用了不同的名称，那么需要修改 `my-mqsc-migrated` 和 `backup.mqsc`。

请注意，样本 YAML 假定 Red Hat OpenShift 环境的缺省存储类定义为 RWX 或 RWO 存储类。 如果未在环境中定义缺省值，那么需要指定要使用的存储类。 您可以通过扩展 YAML 来执行此操作，如下所示:

```
  queueManager:
    name: QM1
    storage:
      defaultClass: my_storage_class
      queueManager:
        type: persistent-claim
```

添加突出显示的文本，并定制类属性以与您的环境匹配。 要发现环境中的存储类名，请运行以下命令:

```
oc get storageclass
```

以下是此命令返回的样本输出:

```
NAME          PROVISIONER             RECLAIMPOLICY
aws-efs       openshift.org/aws-efs   Delete
gp2 (default) kubernetes.io/aws-ebs   Delete
```

以下代码显示了如何引用在 导入 IBM MQ 配置 部分中导入的 IBM MQ 配置。 如果使用了不同的名称，那么需要修改 `my-mqsc-migrated` 和 `backup.mqsc`。

```
    mqsc:
      - configMap:
          name: my-mqsc-migrated
          items:
            - backup.mqsc
```

您已部署单实例队列管理器。 这将完成模板。 现在，您已准备好 验证新的容器部署。

- 部署多实例队列管理器。

  已迁移的队列管理器将使用 YAML 文件部署到 Red Hat OpenShift 。 以下样本基于先前部分中使用的名称。

```
apiVersion: mq.ibm.com/v1beta1
kind: QueueManager
metadata:
  name: qm1mi
spec:
  version: 9.2.5.0-r3
  license:
    accept: true
    license: L-RJON-C7QG3S
    use: "Production"
  pki:
    keys:
    - name: default
      secret:
        secretName: my-tls-migration
        items:
          - tls.key
          - tls.crt
  web:
    enabled: true
```

```
  queueManager:
    name: QM1
    availability: MultiInstance
    storage:
      defaultClass: aws-efs
      persistedData:
        enabled: true
      queueManager:
        enabled: true
      recoveryLogs:
        enabled: true
    mqsc:
      - configMap:
          name: my-mqsc-migrated
          items:
            - backup.mqsc
```

以下是对此 YAML 的说明。 大多数配置遵循与 部署单个实例队列管理器相同的方法，因此此处仅说明队列管理器可用性和存储方面。

```
queueManager:
    name: QM1
    availability: MultiInstance
```

这将队列管理器名称指定为 `QM1`，并将部署设置为 `MultiInstance` 而不是缺省单个实例。

```
    storage:
      defaultClass: aws-efs
      persistedData:
        enabled: true
      queueManager:
        enabled: true
      recoveryLogs:
        enabled: true
```

IBM MQ 多实例队列管理器依赖于 RWX 存储器。 缺省情况下，队列管理器以单实例方式部署，因此在更改为多实例方式时需要其他存储选项。 在先前的 YAML 样本中，定义了三个存储持久卷和一个持久卷类。 此持久卷类需要是 RWX 存储类。 如果您不确定环境中的存储类名，那么可以运行以下命令来发现这些存储类名:

```
oc get storageclass
```

以下是此命令返回的样本输出:

```
NAME          PROVISIONER             RECLAIMPOLICY
aws-efs       openshift.org/aws-efs   Delete
gp2 (default) kubernetes.io/aws-ebs   Delete
```

以下代码显示了如何引用在 导入 IBM MQ 配置 部分中导入的 IBM MQ 配置。 如果使用了不同的名称，那么需要修改 `my-mqsc-migrated` 和 `backup.mqsc`。

```
    mqsc:
      - configMap:
          name: my-mqsc-migrated
          items:
            - backup.mqsc
```

您已部署多实例队列管理器。 这将完成模板。 现在，您已准备好 验证新的容器部署。

V 9.2.1 OpenShift CD EUS 验证新的容器部署

现在， IBM MQ 已部署在 Red Hat OpenShift 上，您可以使用 IBM MQ 样本来验证环境。

开始之前

此任务假定您已 在 Red Hat OpenShift 上创建新的队列管理器。

要点: 此任务假定未在队列管理器中启用 TLS。

关于此任务

在此任务中，您从已迁移队列管理器的容器中运行 IBM MQ 样本。 但是，您可能更愿意使用自己从其他环境运行的应用程序。

您需要以下信息：

- LDAP 用户名
- LDAP 密码
- IBM MQ 通道名称
- 队列名称

此示例代码使用以下设置。 请注意，您的设置将有所不同。

- LDAP 用户名 :mqapp
- LDAP 密码 :mqapp
- IBM MQ 通道名称: DEV.APP.SVRCONN
- 队列名称: Q1

过程

1. 执行到正在运行的 IBM MQ 容器中。

   使用以下命令：

   ```
   oc exec -it qm1-ibm-mq-0 /bin/bash
   ```

   其中， `qm1-ibm-mq-0` 是我们在 第 50 页的『在 Red Hat OpenShift 上创建新的队列管理器』中部署的 Pod。 如果您调用了不同的部署，请定制此值。
2. 发送消息。

   运行以下命令：

   ```
   cd /opt/mqm/samp/bin
   export IBM MQSAMP_USER_ID=mqapp
   export IBM MQSERVER=DEV.APP.SVRCONN/TCP/'localhost(1414)'
   ./amqsputc Q1 QM1
   ```

   系统会提示您输入密码，然后可以发送消息。
3. 验证是否已成功接收消息。

   运行 GET 样本：

   ```
   ./amqsgetc Q1 QM1
   ```

结果

您已完成 第 33 页的『将 IBM MQ 迁移到 IBM Cloud Pak for Integration』。

下一步做什么

使用以下信息来帮助您处理更复杂的迁移方案：

迁移排队的消息

要迁移现有已排队的消息，请遵循以下主题中的指导，在新队列管理器就绪后导出和导入消息: 在两个系统之间使用 dmpmqmsg 实用程序。

从 Red Hat OpenShift 环境外部连接到 IBM MQ

可以向 Red Hat OpenShift 环境外部的 IBM MQ 客户机和队列管理器公开已部署的队列管理器。 此过程取决于连接到 Red Hat OpenShift 环境的 IBM MQ 版本。 请参阅第 98 页的『配置路由以从 Red Hat OpenShift 集群外部连接到队列管理器』。

OpenShift CP4I 在 Red Hat OpenShift 上安装和卸载 IBM MQ Operator

可以使用 Operator Hub 将 IBM MQ Operator 安装到 Red Hat OpenShift 上。

过程

- 第 9 页的『IBM MQ Operator 的依赖关系』.
- 第 10 页的『IBM MQ Operator 所需的集群范围的许可权』.
- 第 55 页的『使用 Red Hat OpenShift Web 控制台安装 IBM MQ Operator』.
- 第 57 页的『使用 Red Hat OpenShift CLI 安装 IBM MQ Operator』.
- 第 60 页的『在气郄环境中安装 IBM MQ Operator』.

相关任务

第 56 页的『使用 Red Hat OpenShift Web 控制台卸载 IBM MQ Operator』
您可以使用 Red Hat OpenShift Web 控制台从 Red Hat OpenShift 卸载 IBM MQ Operator 。

第 59 页的『使用 Red Hat OpenShift CLI 卸载 IBM MQ Operator』
您可以使用 Red Hat OpenShift CLI 从 Red Hat OpenShift 卸载 IBM MQ Operator 。 卸载过程存在差异，具体取决于 IBM MQ Operator 是安装在单个名称空间中，还是安装在集群上的所有名称空间中并可供其使用。

OpenShift CP4I 使用 Red Hat OpenShift Web 控制台安装 IBM MQ Operator

可以使用 Operator Hub 将 IBM MQ Operator 安装到 Red Hat OpenShift 上。

开始之前

登录到 Red Hat OpenShift 集群 Web 控制台。

过程

1. EUS
   可选： 将 IBM Common Services 操作程序添加到可安装操作程序列表中。

   注:

   此步骤适用于 IBM MQ Operator 1.5 和更低版本的发行版。 此步骤将添加单独的 Common Services 目录。 对于操作程序的更高发行版， Common Services 包含在 IBM 目录中。

   a) 单击屏幕右上角的加号图标。 您会看到**导入 YAML** 对话框。

   b) 将以下资源定义粘贴到对话框中。

   ```
   apiVersion: operators.coreos.com/v1alpha1
   kind: CatalogSource
   metadata:
     name: opencloud-operators
     namespace: openshift-marketplace
   spec:
     displayName: IBMCS Operators
     publisher: IBM
     sourceType: grpc
     image: icr.io/cpopen/ibm-common-service-catalog:latest
     updateStrategy:
       registryPoll:
         interval: 45m
   ```

   c) 单击**创建**。

2. 将 IBM 操作程序添加到可安装操作程序列表

   a) 单击屏幕右上角的加号图标。 您会看到**导入 YAML** 对话框。

   b) 将以下资源定义粘贴到对话框中。

   ```
   apiVersion: operators.coreos.com/v1alpha1
   kind: CatalogSource
   metadata:
   ```

```
  name: ibm-operator-catalog
  namespace: openshift-marketplace
spec:
  displayName: IBM Operator Catalog
  image: icr.io/cpopen/ibm-operator-catalog:latest
  publisher: IBM
  sourceType: grpc
  updateStrategy:
    registryPoll:
      interval: 45m
```

c) 单击**创建**。

3. 创建要用于 IBM MQ Operator 的名称空间

   可以将 IBM MQ Operator 安装到单个名称空间或所有名称空间。 仅当要安装到尚不存在的特定名称空间时，才需要执行此步骤。

   a) 在导航窗格中，单击 **主页** > **项目**。

      此时将显示 "项目" 页面。

   b) 单击**创建项目**。 将显示 "创建项目" 区域。

   c) 输入正在创建的名称空间的详细信息。 例如，可以指定 "ibm-mq" 作为名称。

   d) 单击**创建**。 将创建 IBM MQ Operator 的名称空间。

4. 安装 IBM MQ Operator。

   a) 在导航窗格中，单击 **操作程序** > **OperatorHub**。

      此时将显示 " OperatorHub " 页面。

   b) 在 **所有项** 字段中，输入 "IBM MQ"。

      这将显示 IBM MQ 目录条目。

   c) 选择 **IBM MQ**。

      此时将显示 " IBM MQ " 窗口。

   d) 单击**安装**。

      您将看到 "创建操作员预订" 页面。

   e) 查看 第 6 页的『IBM MQ Operator 的版本支持』 以确定要选择的操作员通道。

   f) 将安装方式设置为您创建的特定名称空间或集群范围的作用域。

      建议选择集群范围的作用域，因为在不同名称空间中安装不同版本的操作程序可能会导致问题。 操作员被设计为控制平面的扩展。

   g) 单击**预订**。

      您将在 "已安装的操作程序" 页面上看到 IBM MQ 。

   h) 在 "已安装的操作程序" 页面上检查操作程序的状态，当安装完成时，状态将更改为 "已成功"。

下一步做什么

第 73 页的『使用 Red Hat OpenShift Web 控制台为 IBM MQ 准备 Red Hat OpenShift 项目』

OpenShift CP4I 使用 *Red Hat OpenShift Web* 控制台卸载 *IBM MQ Operator*

您可以使用 Red Hat OpenShift Web 控制台从 Red Hat OpenShift 卸载 IBM MQ Operator 。

开始之前

登录到 Red Hat OpenShift 集群的 Web 控制台。

如果 IBM MQ Operator 安装在集群上的所有项目/名称空间中，请针对要在其中删除队列管理器的每个项目重复以下过程中的步骤 1-5。

过程

1. 选择 **操作程序** > **已安装的操作程序**。

2. 从 **项目** 下拉列表中，选择项目。
3. 单击 **IBM MQ** 操作程序。
4. 单击 **队列管理器** 选项卡以查看由此 IBM MQ Operator 管理的队列管理器。
5. 删除一个或多个队列管理器。

   请注意，尽管这些队列管理器继续运行，但如果没有 IBM MQ Operator，它们可能无法按预期运行。
6. 可选： 如果适用，请对要在其中删除队列管理器的每个项目重复步骤 1-5。
7. 返回到 **操作程序** > **已安装的操作程序**。
8. 在 **IBM MQ** 操作程序旁边，单击三个点菜单，然后选择 **卸载操作程序**。
9. 如果您正在使用 Red Hat OpenShift Container Platform 4.7，那么可能需要从命令行手动删除验证 Web 挂钩:

```
oc delete validatingwebhookconfiguration namespace.validator.queuemanagers.mq.ibm.com
```

OpenShift CP4I 使用 Red Hat OpenShift CLI 安装 IBM MQ Operator

可以使用 Operator Hub 将 IBM MQ Operator 安装到 Red Hat OpenShift 上。

开始之前

使用 **oc login** 登录到 Red Hat OpenShift 命令行界面 (CLI)。 对于这些步骤，您将需要是集群管理员。

过程

1. EUS

   可选： 为 IBM Common Services 操作员创建 **CatalogSource** 。

   注:

   此步骤适用于 IBM MQ Operator 1.5 和更低版本的发行版。 此步骤将添加单独的 Common Services 目录。 对于操作程序的更高发行版， Common Services 包含在 IBM 目录中。

   a) 创建用于定义 **CatalogSource** 资源的 YAML 文件。

   创建名为 "operator-source-cs.yaml" 的文件，其中包含以下内容:

```
apiVersion: operators.coreos.com/v1alpha1
kind: CatalogSource
metadata:
  name: opencloud-operators
  namespace: openshift-marketplace
spec:
  displayName: IBMCS Operators
  publisher: IBM
  sourceType: grpc
  image: icr.io/cpopen/ibm-common-service-catalog:latest
  updateStrategy:
    registryPoll:
      interval: 45m
```

   b) 将 **CatalogSource** 应用于服务器。

```
oc apply -f operator-source-cs.yaml -n openshift-marketplace
```

2. 为 IBM 操作员创建 **CatalogSource**

   a) 创建用于定义 **CatalogSource** 资源的 YAML 文件

   使用以下内容创建名为 "operator-source-ibm.yaml" 的文件:

```
apiVersion: operators.coreos.com/v1alpha1
kind: CatalogSource
metadata:
  name: ibm-operator-catalog
  namespace: openshift-marketplace
spec:
```

```
  displayName: IBM Operator Catalog
  image: icr.io/cpopen/ibm-operator-catalog:latest
  publisher: IBM
  sourceType: grpc
  updateStrategy:
    registryPoll:
      interval: 45m
```

b) 将 **CatalogSource** 应用于服务器。

```
oc apply -f operator-source-ibm.yaml -n openshift-marketplace
```

3. 创建要用于 IBM MQ Operator 的名称空间

   可以将 IBM MQ Operator 安装到单个名称空间或所有名称空间。 仅当要安装到尚不存在的特定名称空间时，才需要执行此步骤。

```
oc new-project ibm-mq
```

4. 从 OperatorHub 查看可用于集群的操作程序列表

```
oc get packagemanifests -n openshift-marketplace
```

5. 检查 IBM MQ Operator 以验证其受支持的 InstallModes 和可用通道

```
oc describe packagemanifests ibm-mq -n openshift-marketplace
```

6. 创建 **OperatorGroup** 对象 YAML 文件

   OperatorGroup 是 OLM 资源，用于选择目标名称空间，在这些名称空间中，将为与 **OperatorGroup** 相同的名称空间中的所有操作程序生成必需的 RBAC 访问权。

   向其预订操作程序的名称空间必须具有与操作程序的 **InstallMode**(`AllNamespaces` 或 `SingleNamespace` 方式) 匹配的 **OperatorGroup** 。 如果您打算安装的操作程序使用 `AllNamespaces`，那么 `openshift-operators` 名称空间已具有适当的 **OperatorGroup** 。

   但是，如果操作程序使用 `SingleNamespace` 方式，并且您还没有相应的 **OperatorGroup** ，那么必须创建一个。

   a) 使用以下内容创建名为 "mq-operator-group.yaml" 的文件:

```
apiVersion: operators.coreos.com/v1
kind: OperatorGroup
metadata:
  name: <operatorgroup_name>
  namespace: <namespace_name>
spec:
  targetNamespaces:
  - <namespace_name>
```

   b) 创建 **OperatorGroup** 对象

```
oc apply -f mq-operator-group.yaml
```

7. 创建 **Subscription** 对象 YAML 文件以将名称空间预订到 IBM MQ Operator

   a) 查看 第 6 页的『IBM MQ Operator 的版本支持』 以确定要选择的操作员通道。

   b) 使用以下内容创建名为 "mq-sub.yaml" 的文件，但更改 **channel** 以匹配要安装的 IBM MQ Operator 版本的通道。

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: ibm-mq
  namespace: openshift-operators
spec:
  channel: <ibm-mq-operator-channel>
  name: ibm-mq
  source: ibm-operator-catalog
  sourceNamespace: openshift-marketplace
```

对于 `AllNamespaces` **InstallMode** 用法，请在名称空间中指定 **openshift-operators** 。 否则，请为 `SingleNamespace` **InstallMode** 用法指定相关的单个名称空间。 请注意，您只应该更改 **namespace** 字段，使 **sourceNamespace** 字段保持不变。

c) 创建 **Subscription** 对象

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

8. 检查操作员的状态

   安装操作程序成功后， pod 状态显示为 *Running*。 对于 `AllNamespaces` **InstallMode** 用法，请指定 **openshift-operators** 作为名称空间。 否则，请为 `SingleNamespace` **InstallMode** 用法指定相关的单个名称空间。

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

下一步做什么

第 74 页的『使用 Red Hat OpenShift CLI 为 IBM MQ 准备 Red Hat OpenShift 项目』

OpenShift CP4I 使用 *Red Hat OpenShift CLI* 卸载 *IBM MQ Operator*

您可以使用 Red Hat OpenShift CLI 从 Red Hat OpenShift 卸载 IBM MQ Operator 。 卸载过程存在差异，具体取决于 IBM MQ Operator 是安装在单个名称空间中，还是安装在集群上的所有名称空间中并可供其使用。

开始之前

使用 `oc login` 登录到 Red Hat OpenShift 集群。

过程

- 如果 IBM MQ Operator 安装在单个名称空间中，请完成以下子步骤:

  a) 确保您在正确的项目中:

```
oc project <project_name>
```

  b) 查看项目中安装的队列管理器:

```
oc get qmgr
```

  c) 删除一个或多个队列管理器:

```
oc delete qmgr <qmgr_name>
```

  请注意，尽管这些队列管理器继续运行，但如果没有 IBM MQ Operator，它们可能无法按预期运行。

  d) 查看 **ClusterServiceVersion** 实例:

```
oc get csv
```

  e) 删除 IBM MQ **ClusterServiceVersion**:

```
oc delete csv <ibm_mq_csv_name>
```

  f) 查看预订:

```
oc get subscription
```

  g) 删除所有预订:

```
oc delete subscription <ibm_mq_subscription_name>
```

  h) 可选： 如果没有任何其他服务在使用公共服务，那么您可能想要卸载公共服务操作程序，并删除操作程序组:

a. 通过遵循 IBM Cloud Pak foundational services 产品文档中的 卸载公共服务 中的指示信息来卸载公共服务操作程序。

b. 查看操作员组:

```
oc get operatorgroup
```

c. 删除操作程序组:

```
oc delete OperatorGroup <operator_group_name>
```

- 如果 IBM MQ Operator 已安装并且可供集群上的所有名称空间使用，请完成以下子步骤:

a) 查看所有已安装的队列管理器:

```
oc get qmgr -A
```

b) 删除一个或多个队列管理器:

```
oc delete qmgr <qmgr_name> -n <namespace_name>
```

请注意，尽管这些队列管理器继续运行，但如果没有 IBM MQ Operator，它们可能无法按预期运行。

c) 查看 **ClusterServiceVersion** 实例:

```
oc get csv -A
```

d) 从集群中删除 IBM MQ **ClusterServiceVersion** :

```
oc delete csv <ibm_mq_csv_name> -n openshift-operators
```

e) 查看预订:

```
oc get subscription -n openshift-operators
```

f) 删除预订:

```
oc delete subscription <ibm_mq_subscription_name> -n openshift-operators
```

g) 如果您正在使用 Red Hat OpenShift Container Platform 4.7，那么可能需要手动删除验证 Web 挂钩:

```
oc delete validatingwebhookconfiguration namespace.validator.queuemanagers.mq.ibm.com
```

h) 可选： 如果没有任何其他服务在使用公共服务，那么您可能想要卸载公共服务操作程序:

遵循 IBM Cloud Pak foundational services 产品文档中的 卸载公共服务 中的指示信息。

OpenShift CP4I Linux 在气郄环境中安装 IBM MQ Operator

本教程指导您将 IBM MQ Operator 安装到没有因特网连接的 Red Hat OpenShift 集群中。 您可以使用便携式存储设备或使用防御机器在气郄环境中安装 IBM MQ Operator 。

使用便携式存储设备在气郄环境中安装 IBM MQ Operator

有关完成安装的步骤，请参阅 IBM Cloud Pak for Integration 文档中的 使用可移植存储设备的镜像映像 。 如果仅安装 IBM MQ，请将出现的所有以下环境变量替换为此处给出的值:

```
export CASE_NAME=ibm-mq
export CASE_ARCHIVE_VERSION=version_number
export CASE_INVENTORY_SETUP=ibmMQOperator
```

其中 *version_number* 是要用于执行气郄安装的案例版本。 有关可用案例版本的列表，请参阅 https://github.com/IBM/cloud-pak/tree/master/repo/case/ibm-mq。 查看 第 6 页的『IBM MQ Operator 的版本支持』 以确定要选择的操作员通道。

使用防御机器在气郄环境中安装 IBM MQ Operator

1. 第 61 页的『先决条件』
2. 第 61 页的『准备 Docker 注册表』
3. 第 62 页的『准备防御主机』
4. 第 62 页的『创建对应于安装程序和映像清单的环境变量』
5. 第 63 页的『下载 IBM MQ 安装程序和映像清单』
6. 第 63 页的『以集群管理员身份登录 Red Hat OpenShift Container Platform 集群』
7. 第 63 页的『为 IBM MQ Operator 创建 Kubernetes 名称空间』
8. 第 63 页的『建立映像的镜像并配置集群』
9. 第 65 页的『安装 IBM MQ Operator。』
10. 第 65 页的『部署 IBM MQ 队列管理器』

先决条件

1. 必须安装 Red Hat OpenShift Container Platform 集群。 有关受支持的 Red Hat OpenShift Container Platform 版本，请参阅 第 6 页的『IBM MQ Operator 的版本支持』。
2. Docker 注册表必须可用。 有关更多信息，请参阅 第 61 页的『准备 Docker 注册表』。
3. 必须配置防御服务器。 有关更多信息，请参阅第 62 页的『准备防御主机』。

准备 Docker 注册表

本地 Docker 注册表用于存储本地环境中的所有映像。 您必须创建这样的注册表，而且必须确保它符合下列要求：

- 支持 Docker 清单 V2，模式 2。
- 支持多体系结构映像。
- 可以从防御服务器和 Red Hat OpenShift Container Platform 集群节点进行访问。
- 具有可以从防御主机写入目标注册表的用户名和密码。
- 具有可以读取 Red Hat OpenShift 集群节点上目标注册表的用户的用户名和密码。
- 允许在映像名称中使用路径分隔符。

创建 Docker 注册表后，必须配置注册表:

1. 创建注册表名称空间
   - `ibmcom` - 此名称空间用于存储所有来自 `dockerhub.io/ibmcom` 名称空间的映像。

     `ibmcom` 名称空间适用于所有公开可用的 IBM 映像，并且不需要凭证即可拉取。
   - `cp` -用于存储 `cp.icr.io/cp` 存储库中的 IBM 映像的名称空间。

     `cp` 名称空间用于 IBM 授权注册表中需要产品权利密钥和凭证来拉取的映像。 要获取权利密钥，请使用与授权软件关联的 IBM 标识和密码登录到 MyIBM Container Software Library 。 在 **权利密钥** 部分中，选择 **复制密钥** 以将权利密钥复制到剪贴板，然后将其保存以在以下步骤中使用。
   - `opencloudio` - 此名称空间用于存储来自 `quay.io/opencloudio` 的映像。

     `opencloudio` 名称空间用于选择 IBM 在 quay.io 上可用的开放式源代码组件映像。 IBM Cloud Pak foundational services 映像托管在 `opencloudio` 上。
2. 请确认每个名称空间都满足下列要求。
   - 支持自动创建存储库。
   - 具有可以写入和创建存储库的用户的凭证。 防御主机会使用这些凭证。
   - 具有可以读取所有存储库的用户的凭证。 Red Hat OpenShift Container Platform 集群会使用这些凭证。

准备防御主机

准备可访问 Red Hat OpenShift Container Platform 集群，本地 Docker 注册表和因特网的防御主机。 防御主机必须位于具有 IBM Cloud Pak CLI 和 Red Hat OpenShift Container Platform CLI 支持的任何操作系统的 Linux for x86-64 平台上。

请在防御节点上完成下列步骤:

1. 安装 OpenSSL V 1.11.1 或更高版本。
2. 在防御节点上安装 Docker 或 Podman 。
   - 要安装 Docker，请运行以下命令:

     ```
     yum check-update
     yum install docker
     ```

   - 要安装 Podman，请参阅 Podman 安装指示信息
3. 在防御节点上安装 skopeo V 1.x.x 。 要安装 skopeo ，请运行以下命令:

   ```
   yum check-update
   yum install skopeo
   ```

4. 安装 IBM Cloud Pak CLI。 安装对应于您所用平台的最新版本二进制文件。 有关更多信息，请参阅 cloud-pak-cli。
   a. 下载二进制文件。

      ```
      wget https://github.com/IBM/cloud-pak-cli/releases/download/vversion-number/binary-file-name
      ```

      例如:

      ```
      wget https://github.com/IBM/cloud-pak-cli/releases/latest/download/cloudctl-linux-amd64.tar.gz
      ```

   b. 解压缩该二进制文件。

      ```
      tar -xf binary-file-name
      ```

   c. 运行以下命令以修改和移动文件

      ```
      chmod 755 file-name
      mv file-name /usr/local/bin/cloudctl
      ```

   d. 确认 `cloudctl` 已安装:

      ```
      cloudctl --help
      ```

5. 安装 oc Red Hat OpenShift Container Platform CLI 工具。

   有关更多信息，请参阅 Red Hat OpenShift Container Platform CLI 工具
6. 创建一个充当离线存储库的目录。

   以下是示例目录。 此示例会在后续的步骤中使用。

   ```
   mkdir $HOME/offline
   ```

   注: 此脱机存储必须是持久存储，以避免多次传输数据。 持久性也有助于多次运行或按时间表运行镜像过程。

创建对应于安装程序和映像清单的环境变量

使用安装程序映像名称和映像清单创建以下环境变量:

```
export CASE_ARCHIVE_VERSION=version_number
export CASE_ARCHIVE=ibm-mq-$CASE_ARCHIVE_VERSION.tgz
export CASE_INVENTORY=ibmMQOperator
```

其中 *version_number* 是要用于执行气郄安装的案例版本。 有关可用案例版本的列表，请参阅 https://github.com/IBM/cloud-pak/tree/master/repo/case/ibm-mq。 查看 IBM MQ Operator 的版本支持 以确定要选择的操作员通道。

下载 IBM MQ 安装程序和映像清单

将 `ibm-mq` 安装程序和映像清单下载到防御主机:

```
cloudctl case save \
    --case https://github.com/IBM/cloud-pak/raw/master/repo/case/ibm-mq/$CASE_ARCHIVE_VERSION/
$CASE_ARCHIVE \
    --outputdir $HOME/offline/
```

以集群管理员身份登录 Red Hat OpenShift Container Platform 集群

以下是用于登录到 Red Hat OpenShift Container Platform 集群的示例命令:

```
oc login cluster_host:port --username=cluster_admin_user --password=cluster_admin_password
```

为 IBM MQ Operator 创建 Kubernetes 名称空间

使用名称空间创建环境变量以安装 IBM MQ Operator，然后创建名称空间:

```
export NAMESPACE=ibm-mq-test
oc create namespace ${NAMESPACE}
```

建立映像的镜像并配置集群

请完成下列步骤，以建立映像的镜像并配置集群:

注: 请勿在任何命令中的双引号内使用波浪号。 例如，请不要使用 `args "--registry registry --user registry_userid --pass registry_password --inputDir ~/offline"`。 波浪号不会展开，您的命令可能会失败。

1. 存储所有源 Docker 注册表的认证凭证。

   所有 IBM Cloud Platform Common Services， IBM MQ Operator 映像和 IBM MQ Advanced Developer 映像都存储在不需要认证的公共注册表中。 但是， IBM MQ Advanced Server (非开发者) ，其他产品和第三方组件需要一个或多个已认证的注册表。 下列注册表需要认证:

   - `cp.icr.io`
   - `registry.redhat.io`
   - `registry.access.redhat.com`

   有关这些注册表的更多信息，请参阅创建注册表名称空间。

   您必须运行以下命令，以便为所有需要认证的注册表配置凭证。 请针对每个这样的注册表分别运行此命令:

   ```
   cloudctl case launch \
   --case $HOME/offline/${CASE_ARCHIVE} \
   --inventory ${CASE_INVENTORY} \
   --action configure-creds-airgap \
   --namespace ${NAMESPACE} \
   --args "--registry registry --user registry_userid --pass registry_password --inputDir $HOME/
   offline"
   ```

   以上命令会将注册表凭证存储并缓存在文件系统上 `$HOME/.airgap/secrets` 位置的文件中。

2. 使用本地 Docker 注册表连接信息创建环境变量。

```
export LOCAL_DOCKER_REGISTRY=IP_or_FQDN_of_local_docker_registry
export LOCAL_DOCKER_USER=username
export LOCAL_DOCKER_PASSWORD=password
```

注: Docker 注册表使用标准端口，例如 80 或 443。 如果 Docker 注册表使用非标准端口，请使用语法 *host:port* 指定端口。 例如:

```
export LOCAL_DOCKER_REGISTRY=myregistry.local:5000
```

3. 为本地 Docker 注册表配置认证密钥。

注: 此步骤只需执行一次。

```
cloudctl case launch \
--case $HOME/offline/${CASE_ARCHIVE} \
--inventory ${CASE_INVENTORY} \
--action configure-creds-airgap \
--namespace ${NAMESPACE} \
--args "--registry ${LOCAL_DOCKER_REGISTRY} --user ${LOCAL_DOCKER_USER} --pass $
{LOCAL_DOCKER_PASSWORD}"
```

以上命令会将注册表凭证存储并缓存在文件系统上 `$HOME/.airgap/secrets` 位置的文件中。

4. 配置全局映像拉取私钥和 **ImageContentSourcePolicy**。

   a. 检查是否需要重新启动节点。

   - 在 Red Hat OpenShift Container Platform V 4.4 及更高版本中，以及在使用气郄的 IBM MQ Operator 的新安装上，此步骤将重新启动所有集群节点。 可能要到应用新的提取密钥之后，集群资源才可供使用。
   - 在 IBM MQ Operator 1.8 中， CASE 已更新为包含映像的其他镜像源。 因此，当您从先前版本的 IBM MQ Operator 升级到 V 1.8 或更高版本时，将触发节点重新启动。
   - 要检查此步骤是否需要节点重新启动，请将 `--dry-run` 选项添加到此步骤的代码中。 这将生成最新的 **ImageContentSourcePolicy** ，并将其显示在控制台窗口 (**stdout**) 中。 如果此 **ImageContentSourcePolicy** 与集群配置的 **ImageContentSourcePolicy** 不同，那么将重新启动。

```
cloudctl case launch \
--case $HOME/offline/${CASE_ARCHIVE} \
--inventory ${CASE_INVENTORY} \
--action configure-cluster-airgap \
--namespace ${NAMESPACE} \
--args "--registry ${LOCAL_DOCKER_REGISTRY} --user ${LOCAL_DOCKER_USER} --pass $
{LOCAL_DOCKER_PASSWORD} --inputDir $HOME/offline --dryRun"
```

   b. 要配置全局映像拉取私钥和 **ImageContentSourcePolicy**，请在不使用 `--dry-run` 选项的情况下运行此步骤的代码:

```
cloudctl case launch \
--case $HOME/offline/${CASE_ARCHIVE} \
--inventory ${CASE_INVENTORY} \
--action configure-cluster-airgap \
--namespace ${NAMESPACE} \
--args "--registry ${LOCAL_DOCKER_REGISTRY} --user ${LOCAL_DOCKER_USER} --pass $
{LOCAL_DOCKER_PASSWORD} --inputDir $HOME/offline"
```

5. 确认已创建 **ImageContentSourcePolicy** 资源。

```
oc get imageContentSourcePolicy
```

6. 可选: 如果您正在使用不安全的注册表，那么必须将本地注册表添加到集群 **insecureRegistries** 列表。

```
oc patch image.config.openshift.io/cluster --type=merge -p '{"spec":{"registrySources":
{"insecureRegistries":["'${LOCAL_DOCKER_REGISTRY}'"]}}}'
```

7. 验证集群节点状态。

```
oc get nodes
```

在应用 **imageContentsourcePolicy** 和全局映像提取私钥之后，您可能会看到节点状态为 **Ready**、**Scheduling** 或 **Disabled**。 请等待所有节点都显示 **Ready** 状态。

8. 将映像镜像到本地注册表。

```
cloudctl case launch \
--case $HOME/offline/${CASE_ARCHIVE} \
--inventory ${CASE_INVENTORY} \
--action mirror-images \
--namespace ${NAMESPACE} \
--args "--registry ${LOCAL_DOCKER_REGISTRY} --user ${LOCAL_DOCKER_USER} --pass $
{LOCAL_DOCKER_PASSWORD} --inputDir $HOME/offline"
```

安装 IBM MQ Operator。

1. 登录到 Red Hat OpenShift 集群 Web 控制台。
2. 创建目录源。 使用执行先前步骤的同一终端。

```
cloudctl case launch \
--case $HOME/offline/${CASE_ARCHIVE} \
--inventory ${CASE_INVENTORY} \
--action install-catalog \
--namespace ${NAMESPACE} \
--args "--registry ${LOCAL_DOCKER_REGISTRY} --recursive"
```

3. 验证是否为 Common Services 安装程序操作程序创建了 **CatalogSource** 。

```
oc get pods -n openshift-marketplace
oc get catalogsource -n openshift-marketplace
```

4. 使用 OLM 安装 IBM MQ Operator 。
   a. 在导航窗格中，单击 **操作程序** > **OperatorHub**。

      此时将显示 **OperatorHub** 页面。
   b. 在 **所有项** 字段中，输入 IBM MQ。

      这将显示 IBM MQ 商品。
   c. 选择 **IBM MQ**。

      此时将显示 " **IBM MQ** " 窗口。
   d. 单击**安装**。

      这将显示 " **创建操作员预订** " 页面。
   e. 查看 第 6 页的『IBM MQ Operator 的版本支持』 以确定要选择的操作员通道。
   f. 将 **安装方式** 设置为您创建的特定名称空间或集群范围内的名称空间。
   g. 单击**预订**。

      IBM MQ 将添加到 " **已安装的操作程序** " 页面。
   h. 在 " **已安装的操作程序** " 页面上检查操作程序的状态。 安装完成时，状态将更改为 **Succeeded** 。

部署 IBM MQ 队列管理器

要在已安装的操作程序下创建新的队列管理器，请参阅 第 73 页的『使用 IBM MQ Operator 部署和配置队列管理器』。

相关任务

第 66 页的『准备在气郄环境中升级 IBM MQ Operator 或队列管理器』
在没有因特网连接的 Red Hat OpenShift 集群中，在升级 IBM MQ Operator 之前需要执行一些准备步骤。

OpenShift CP4I 升级 IBM MQ Operator 和队列管理器

升级 IBM MQ Operator 将允许您升级队列管理器。

过程

- 第 69 页的『使用 Red Hat OpenShift Web 控制台升级 IBM MQ Operator』.
- 第 70 页的『使用 Red Hat OpenShift CLI 升级 IBM MQ Operator』.
- 第 71 页的『使用 Red Hat OpenShift Web 控制台升级 IBM MQ 队列管理器』.
- 第 72 页的『使用 Red Hat OpenShift CLI 升级 IBM MQ 队列管理器』.

OpenShift CP4I Linux 准备在气郄环境中升级 IBM MQ Operator 或队列管理器

在没有因特网连接的 Red Hat OpenShift 集群中，在升级 IBM MQ Operator 之前需要执行一些准备步骤。

开始之前

本主题假定您已配置本地映像注册表，在该注册表中镜像了先前发布的 IBM Cloud Pak for Integration 映像。

关于此任务

必须先镜像最新的 IBM Cloud Pak for Integration 映像，然后才能在气郄环境中升级 IBM MQ Operator 或队列管理器。

请注意，此任务中的前四个步骤与您在 第 60 页的『在气郄环境中安装 IBM MQ Operator』时执行的步骤相同。

过程

1. 创建安装程序和映像清单的环境变量。

   使用安装程序映像名称和映像清单创建以下环境变量:

   ```
   export CASE_ARCHIVE_VERSION=version_number
   export CASE_ARCHIVE=ibm-mq-$CASE_ARCHIVE_VERSION.tgz
   export CASE_INVENTORY=ibmMQOperator
   ```

   其中 *version_number* 是要用于执行气郄安装的案例版本。 有关可用案例版本的列表，请参阅 https://github.com/IBM/cloud-pak/tree/master/repo/case/ibm-mq。 查看 IBM MQ Operator 的版本支持 以确定要选择的操作员通道。

2. 下载 IBM MQ 安装程序和映像清单。

   将 `ibm-mq` 安装程序和映像清单下载到防御主机:

   ```
   cloudctl case save \
       --case https://github.com/IBM/cloud-pak/raw/master/repo/case/ibm-mq/
   $CASE_ARCHIVE_VERSION/$CASE_ARCHIVE \
       --outputdir $HOME/offline/
   ```

3. 以集群管理员身份登录到 Red Hat OpenShift Container Platform 集群。

   以下是用于登录到 Red Hat OpenShift Container Platform 集群的示例命令:

   ```
   oc login cluster_host:port --username=cluster_admin_user --password=cluster_admin_password
   ```

4. 镜像映像并配置集群。

   请完成下列步骤，以建立映像的镜像并配置集群:

注: 请勿在任何命令中的双引号内使用波浪号。 例如，请不要使用 `args "--registry registry --user registry_userid --pass registry_password --inputDir ~/offline"`。 波浪号不会展开，您的命令可能会失败。

a. 存储所有源 Docker 注册表的认证凭证。

所有 IBM Cloud Platform Common Services， IBM MQ Operator 映像和 IBM MQ Advanced Developer 映像都存储在不需要认证的公共注册表中。 但是， IBM MQ Advanced Server (非开发者)，其他产品和第三方组件需要一个或多个已认证的注册表。 下列注册表需要认证:

- `cp.icr.io`
- `registry.redhat.io`
- `registry.access.redhat.com`

有关这些注册表的更多信息，请参阅创建注册表名称空间。

您必须运行以下命令，以便为所有需要认证的注册表配置凭证。 请针对每个这样的注册表分别运行此命令:

```
cloudctl case launch \
--case $HOME/offline/${CASE_ARCHIVE} \
--inventory ${CASE_INVENTORY} \
--action configure-creds-airgap \
--namespace ${NAMESPACE} \
--args "--registry registry --user registry_userid --pass registry_password --inputDir
$HOME/offline"
```

以上命令会将注册表凭证存储并缓存在文件系统上 `$HOME/.airgap/secrets` 位置的文件中。

b. 使用本地 Docker 注册表连接信息创建环境变量。

```
export LOCAL_DOCKER_REGISTRY=IP_or_FQDN_of_local_docker_registry
export LOCAL_DOCKER_USER=username
export LOCAL_DOCKER_PASSWORD=password
```

注: Docker 注册表使用标准端口，例如 80 或 443。 如果 Docker 注册表使用非标准端口，请使用语法 *host:port* 指定端口。 例如:

```
export LOCAL_DOCKER_REGISTRY=myregistry.local:5000
```

c. 为本地 Docker 注册表配置认证密钥。

注: 此步骤只需执行一次。

```
cloudctl case launch \
--case $HOME/offline/${CASE_ARCHIVE} \
--inventory ${CASE_INVENTORY} \
--action configure-creds-airgap \
--namespace ${NAMESPACE} \
--args "--registry ${LOCAL_DOCKER_REGISTRY} --user ${LOCAL_DOCKER_USER} --pass $
{LOCAL_DOCKER_PASSWORD}"
```

以上命令会将注册表凭证存储并缓存在文件系统上 `$HOME/.airgap/secrets` 位置的文件中。

d. 配置全局映像拉取私钥和 **ImageContentSourcePolicy**。

i) 检查是否需要重新启动节点。

- 在 Red Hat OpenShift Container Platform V 4.4 及更高版本中，以及在使用气郄的 IBM MQ Operator 的新安装上，此步骤将重新启动所有集群节点。 可能要到应用新的提取密钥之后，集群资源才可供使用。
- 在 IBM MQ Operator 1.8 中， CASE 已更新为包含映像的其他镜像源。 因此，当您从先前版本的 IBM MQ Operator 升级到 V 1.8 或更高版本时，将触发节点重新启动。
- 要检查此步骤是否需要节点重新启动，请将 `--dry-run` 选项添加到此步骤的代码中。 这将生成最新的 **ImageContentSourcePolicy**，并将其显示在控制台窗口 (**stdout**) 中。 如果此

ImageContentSourcePolicy 与集群配置的 **ImageContentSourcePolicy** 不同，那么将重新启动。

```
cloudctl case launch \
--case $HOME/offline/${CASE_ARCHIVE} \
--inventory ${CASE_INVENTORY} \
--action configure-cluster-airgap \
--namespace ${NAMESPACE} \
--args "--registry ${LOCAL_DOCKER_REGISTRY} --user ${LOCAL_DOCKER_USER} --pass $
{LOCAL_DOCKER_PASSWORD} --inputDir $HOME/offline --dryRun"
```

ii) 要配置全局映像拉取私钥和 **ImageContentSourcePolicy**，请在不使用 `--dry-run` 选项的情况下运行此步骤的代码:

```
cloudctl case launch \
--case $HOME/offline/${CASE_ARCHIVE} \
--inventory ${CASE_INVENTORY} \
--action configure-cluster-airgap \
--namespace ${NAMESPACE} \
--args "--registry ${LOCAL_DOCKER_REGISTRY} --user ${LOCAL_DOCKER_USER} --pass $
{LOCAL_DOCKER_PASSWORD} --inputDir $HOME/offline"
```

e. 确认已创建 **ImageContentSourcePolicy** 资源。

```
oc get imageContentSourcePolicy
```

f. 可选: 如果您正在使用不安全的注册表，那么必须将本地注册表添加到集群 **insecureRegistries** 列表。

```
oc patch image.config.openshift.io/cluster --type=merge -p '{"spec":{"registrySources":
{"insecureRegistries":["'${LOCAL_DOCKER_REGISTRY}'"]}}}'
```

g. 验证集群节点状态。

```
oc get nodes
```

在应用 **imageContentsourcePolicy** 和全局映像提取私钥之后，您可能会看到节点状态为 **Ready**、**Scheduling** 或 **Disabled**。 请等待所有节点都显示 **Ready** 状态。

h. 将映像镜像到本地注册表。

```
cloudctl case launch \
--case $HOME/offline/${CASE_ARCHIVE} \
--inventory ${CASE_INVENTORY} \
--action mirror-images \
--namespace ${NAMESPACE} \
--args "--registry ${LOCAL_DOCKER_REGISTRY} --user ${LOCAL_DOCKER_USER} --pass $
{LOCAL_DOCKER_PASSWORD} --inputDir $HOME/offline"
```

5. 升级目录源。

使用执行先前步骤的同一终端。

```
cloudctl case launch \
--case $HOME/offline/${CASE_ARCHIVE} \
--inventory ${CASE_INVENTORY} \
--action install-catalog \
--namespace ${NAMESPACE} \
--args "--registry ${LOCAL_DOCKER_REGISTRY} --recursive"
```

下一步做什么

现在，您已准备好通过完成下列其中一项任务来升级 IBM MQ Operator 和队列管理器:

- 第 69 页的『使用 Red Hat OpenShift Web 控制台升级 IBM MQ Operator』
- 第 70 页的『使用 Red Hat OpenShift CLI 升级 IBM MQ Operator』
- 第 71 页的『使用 Red Hat OpenShift Web 控制台升级 IBM MQ 队列管理器』
- 第 72 页的『使用 Red Hat OpenShift CLI 升级 IBM MQ 队列管理器』

- 第 72 页的『使用 "平台 Navigator " 在 Red Hat OpenShift 中升级 IBM MQ 队列管理器』

OpenShift CP4I 使用 Red Hat OpenShift Web 控制台升级 IBM MQ Operator

可以使用 Operator Hub 升级 IBM MQ Operator 。

开始之前

登录到 Red Hat OpenShift 集群 Web 控制台。

必须先镜像最新的 IBM Cloud Pak for Integration 映像，然后才能在气郄环境中升级 IBM MQ Operator 。 请参阅 准备在气郄环境中升级 IBM MQ Operator 或队列管理器。

过程

1. 查看 第 6 页的『IBM MQ Operator 的版本支持』 以确定要升级到的操作员通道。
2. 可选： 如果要从低于 1.5 的 IBM MQ Operator 版本升级到 IBM MQ Operator 1.5 或更高版本，那么必须首先升级 IBM Cloud Pak foundational services 版本。

   有关更多信息，请参阅第 69 页的『使用 Red Hat OpenShift Web 控制台升级 IBM Cloud Pak foundational services』。
3. 升级 IBM MQ Operator。 新的主要或次要 IBM MQ Operator 版本通过新的预订通道交付。 要将操作程序升级到新的主版本或次版本，您将需要更新 IBM MQ Operator 预订中的所选通道。
   a) 在导航窗格中，单击 **操作程序 > 已安装的操作程序**。
      将显示指定项目中的所有已安装的操作程序。
   b) 选择 **IBM MQ 操作程序**
   c) 浏览至 **预订** 选项卡
   d) 单击 **通道**
      此时将显示 " **更改预订更新通道** " 窗口。
   e) 选择所需通道，然后单击 **保存**。
      操作员将升级到可用于新通道的最新版本。 请参阅 第 6 页的『IBM MQ Operator 的版本支持』。

下一步做什么

如果已升级到 IBM Cloud Pak foundational services 3.7，那么将需要升级或重新启动使用 IBM Cloud Pak for Integration 许可证的任何队列管理器。 有关如何执行此操作的更多信息，请参阅 第 71 页的『使用 Red Hat OpenShift Web 控制台升级 IBM MQ 队列管理器』。

OpenShift CP4I 使用 *Red Hat OpenShift Web* 控制台升级 *IBM Cloud Pak foundational services*

如果要从低于 1.5 的 IBM MQ Operator 版本升级到 IBM MQ Operator 1.5 或更高版本，那么必须首先升级 IBM Cloud Pak foundational services 版本。

开始之前

注: 仅当要从低于 1.5 的 IBM MQ Operator 版本升级到 IBM MQ Operator 1.5 或更高版本时，才需要完成此任务。

CP4I 如果您有任何使用 IBM Cloud Pak for Integration 许可证的队列管理器，那么在此升级之后，将需要重新启动队列管理器才能访问 Web 控制台，并且您还会看到登录到 Web 控制台的 其他错误 。 在操作程序升级完成后，可以通过升级到所选 IBM MQ 版本的最新值 `.spec.version` 来修正这些错误。

CP4I 如果您具有现有队列管理器并使用 IBM Cloud Pak for Integration 操作仪表板，请在升级之前参阅 第 101 页的『在 IBM Cloud Pak for Integration 2021.4 中使用 "操作仪表板" 集成来部署或升级 IBM MQ 9.2.2 或 9.2.3』 。

过程

1. 登录到 Red Hat OpenShift 集群 Web 控制台。
2. 在导航窗格中，单击 **操作程序 > 已安装的操作程序**。
   将显示指定项目中的所有已安装的操作程序。
3. 选择 **IBM Cloud Pak foundational services 运算符**。 请注意，在 V 3.7 之前，这称为 **IBM Common Services 操作程序**
4. 浏览至 **预订** 选项卡。
5. 单击 **通道**。
   此时将显示 " **更改预订更新通道** " 窗口。
6. 选择 **v3** 通道，然后单击 **保存**。
   IBM Cloud Pak foundational services 操作程序将升级到可用于新通道的最新版本。 请参阅 第 6 页的『IBM MQ Operator 的版本支持』。

下一步做什么

现在，您已准备好 升级 IBM MQ Operator。

OpenShift CP4I 使用 Red Hat OpenShift CLI 升级 IBM MQ Operator

可以从命令行升级 IBM MQ Operator 。

开始之前

使用 **cloudctl login** (对于 IBM Cloud Pak for Integration) 或 **oc login** 登录到集群。

必须先镜像最新的 IBM Cloud Pak for Integration 映像，然后才能在气郄环境中升级 IBM MQ Operator 。 请参阅 准备在气郄环境中升级 IBM MQ Operator 或队列管理器。

过程

1. 查看 第 6 页的『IBM MQ Operator 的版本支持』 以确定要升级到的操作员通道。
2. 可选： 如果要从低于 1.5 的 IBM MQ Operator 版本升级到 IBM MQ Operator 1.5 或更高版本，那么必须首先升级 IBM Cloud Pak foundational services 版本。

   有关更多信息，请参阅第 70 页的『使用 Red Hat OpenShift CLI 升级 IBM Cloud Pak foundational services』。
3. 升级 IBM MQ Operator。 新的主/次 IBM MQ Operator 版本通过新的预订通道交付。 要将操作程序升级到新的主/次版本，您将需要更新 IBM MQ Operator 预订中的所选通道。
   a) 确保所需的 IBM MQ Operator 升级通道可用。

   ```
   oc get packagemanifest ibm-mq -o=jsonpath='{.status.channels[*].name}'
   ```

   b) 修补 `Subscription` 以移至所需的更新通道 (其中 *vX*)。*Y* 是上一步中标识的所需更新通道。

   ```
   oc patch subscription ibm-mq --patch '{"spec":{"channel":"vX.Y"}}' --type=merge
   ```

下一步做什么

如果已升级到 IBM Cloud Pak foundational services 3.7，那么将需要升级或重新启动使用 IBM Cloud Pak for Integration 许可证的任何队列管理器。 有关如何执行此操作的更多信息，请参阅 第 72 页的『使用 Red Hat OpenShift CLI 升级 IBM MQ 队列管理器』。

OpenShift CP4I 使用 *Red Hat OpenShift CLI 升级 IBM Cloud Pak foundational services*

如果要从低于 1.5 的 IBM MQ Operator 版本升级到 IBM MQ Operator 1.5 或更高版本，那么必须首先升级 IBM Cloud Pak foundational services 版本。

开始之前

注: 仅当要从低于 1.5 的 IBM MQ Operator 版本升级到 IBM MQ Operator 1.5 或更高版本时，才需要完成此任务。

CP4I 如果您有任何使用 IBM Cloud Pak for Integration 许可证的队列管理器，那么在此升级之后，将需要重新启动队列管理器才能访问 Web 控制台，并且您还会看到登录到 Web 控制台的 其他错误 。 在操作程序升级完成后，可以通过升级到所选 IBM MQ 版本的最新值 `.spec.version` 来修正这些错误。

CP4I 如果您具有现有队列管理器并使用 IBM Cloud Pak for Integration 操作仪表板，请在升级之前参阅 第 101 页的『在 IBM Cloud Pak for Integration 2021.4 中使用 "操作仪表板" 集成来部署或升级 IBM MQ 9.2.2 或 9.2.3』 。

过程

1. 使用 **`cloudctl login`** (对于 IBM Cloud Pak for Integration) 或 **`oc login`** 登录到集群。
2. 确保 v3 IBM Cloud Pak foundational services 升级通道可用。

```
oc get packagemanifest -n ibm-common-services ibm-common-service-operator
-o=jsonpath='{.status.channels[*].name}'
```

3. 修补 `Subscription` 以移至所需更新通道: `v3`

```
oc patch subscription ibm-common-service-operator --patch '{"spec":{"channel":"v3"}}' --
type=merge
```

下一步做什么

现在，您已准备好 升级 IBM MQ Operator。

OpenShift CP4I 使用 Red Hat OpenShift Web 控制台升级 IBM MQ 队列管理器

可以使用 Operator Hub 在 Red Hat OpenShift 中升级使用 IBM MQ Operator 部署的 IBM MQ 队列管理器。

开始之前

- 登录到 Red Hat OpenShift 集群 Web 控制台。
- 确保 IBM MQ Operator 正在使用所需的更新通道。 请参阅 第 66 页的『升级 IBM MQ Operator 和队列管理器』。

必须先镜像最新的 IBM Cloud Pak for Integration 映像，然后才能在气郄环境中升级队列管理器。 请参阅 准备在气郄环境中升级 IBM MQ Operator 或队列管理器。

过程

1. 在导航窗格中，单击 **操作程序 > 已安装的操作程序**。
   将显示指定项目中的所有已安装的操作程序。
2. 选择 **IBM MQ 操作程序**。
   这样会显示 " **IBM MQ 操作程序** " 窗口。
3. 浏览至 **队列管理器** 选项卡。
   此时将显示 " **队列管理器详细信息** " 窗口。
4. 选择要升级的队列管理器。
5. 浏览至 YAML 选项卡。
6. 必要时更新以下字段以与期望的 IBM MQ 队列管理器版本升级相匹配。
   - spec.version
   - spec.license.licence

请参阅 第 6 页的『IBM MQ Operator 的版本支持』 以获取通道到 IBM MQ Operator 版本和 IBM MQ 队列管理器版本的映射。

7. 保存更新后的队列管理器 YAML。

OpenShift CP4I 使用 Red Hat OpenShift CLI 升级 IBM MQ 队列管理器

可以使用命令行在 Red Hat OpenShift 中升级使用 IBM MQ Operator 部署的 IBM MQ 队列管理器。

开始之前

您需要是集群管理员才能完成这些步骤。

- 使用 `oc login` 登录到 Red Hat OpenShift 命令行界面 (CLI)。
- 确保 IBM MQ Operator 正在使用所需的更新通道。 请参阅 第 66 页的『升级 IBM MQ Operator 和队列管理器』。

必须先镜像最新的 IBM Cloud Pak for Integration 映像，然后才能在气郄环境中升级队列管理器。 请参阅 准备在气郄环境中升级 IBM MQ Operator 或队列管理器。

过程

必要时，编辑 **QueueManager** 资源以更新以下字段，从而与期望的 IBM MQ 队列管理器版本升级相匹配。

- spec.version
- spec.license.licence

请参阅 第 6 页的『IBM MQ Operator 的版本支持』 以获取通道到 IBM MQ Operator 版本和 IBM MQ 队列管理器版本的映射。

使用以下命令：

```
oc edit queuemanager my_qmgr
```

其中 *my_qmgr* 是要升级的 QueueManager 资源的名称。

CP4I 使用 "平台 Navigator " 在 Red Hat OpenShift 中升级 IBM MQ 队列管理器

可以使用 IBM Cloud Pak for Integration Platform Navigator 在 Red Hat OpenShift 中升级使用 IBM MQ Operator 部署的 IBM MQ 队列管理器。

开始之前

- 登录到包含要升级的队列管理器的名称空间中的 IBM Cloud Pak for Integration Platform Navigator 。
- 确保 IBM MQ Operator 正在使用所需的更新通道。 请参阅 第 66 页的『升级 IBM MQ Operator 和队列管理器』。

必须先镜像最新的 IBM Cloud Pak for Integration 映像，然后才能在气郄环境中升级队列管理器。 请参阅 准备在气郄环境中升级 IBM MQ Operator 或队列管理器。

过程

1. 从 IBM Cloud Pak for Integration Platform Navigator 主页，单击 **运行时** 选项卡。
2. 具有可用升级的队列管理器在 **版本**旁边具有蓝色 **i** 。 单击 **i** 以显示 **新的可用版本**。
3. 单击要升级的队列管理器最右边的三个点，然后单击 **更改版本**。
4. 在 **选择新通道或版本**下，选择所需的升级版本。
5. 单击 **更改版本**。

结果

队列管理器已升级。

OpenShift CP4I 使用 IBM MQ Operator 部署和配置队列管理器

IBM MQ 9.1.5 和更高版本将使用 IBM MQ Operator 部署到 Red Hat OpenShift 。

关于此任务

过程

- 第 73 页的『为 IBM MQ 准备 Red Hat OpenShift 项目』.
- 第 74 页的『将队列管理器部署到 Red Hat OpenShift Container Platform 集群』.

OpenShift CP4I 为 IBM MQ 准备 Red Hat OpenShift 项目

准备 Red Hat OpenShift Container Platform 集群，以便它可以部署队列管理器。

过程

- 第 73 页的『使用 Red Hat OpenShift Web 控制台为 IBM MQ 准备 Red Hat OpenShift 项目』.
- 第 74 页的『使用 Red Hat OpenShift CLI 为 IBM MQ 准备 Red Hat OpenShift 项目』.

相关任务

第 74 页的『将队列管理器部署到 Red Hat OpenShift Container Platform 集群』
使用 QueueManager 定制资源将队列管理器部署到 Red Hat OpenShift Container Platform 集群上。

OpenShift CP4I 使用 *Red Hat OpenShift Web* 控制台为 *IBM MQ* 准备 *Red Hat OpenShift* 项目

准备 Red Hat OpenShift Container Platform 集群，以便它可以使用 IBM MQ Operator 部署队列管理器。 此任务应由项目管理员完成。

开始之前

注: 如果计划在已安装其他 IBM Cloud Pak for Integration 组件的项目中使用 IBM MQ ，那么可能不需要遵循这些指示信息。

登录到 Red Hat OpenShift 集群 Web 控制台。

关于此任务

将从执行许可证权利检查的容器注册表中拉取 IBM MQ Operator 映像。 此检查需要存储在 `docker-registry` 拉取私钥中的权利密钥。 如果您还没有权利密钥，请遵循以下指示信息以获取权利密钥并创建拉取私钥。

过程

1. 获取分配给您的标识的权利密钥。
   a) 使用与授权软件关联的 IBM 标识和密码登录到 MyIBM Container Software Library 。
   b) 在**权利密钥**部分中，选择**复制密钥**以将权利密钥复制到剪贴板。
2. 在要部署队列管理器的项目中创建包含权利密钥的私钥。
   a) 在导航窗格中，单击 **工作负载** > **密钥**。
      将显示 "私钥" 页面。
   b) 在 **项目** 下拉列表中，选择要安装 IBM MQ 的项目
   c) 单击 **创建** 按钮，然后选择 **映像拉取私钥**

d) 在 **名称** 字段中，输入 `ibm-entitlement-key`

e) 在 **注册表服务器地址** 字段中，输入 `cp.icr.io`

f) 在 **用户名** 字段中，输入 `cp`

g) 在 **密码** 字段中，输入您在上一步中复制的权利密钥

h) 在 **电子邮件** 字段中，输入与授权软件关联的 IBM 标识

下一步做什么

第 76 页的『使用 Red Hat OpenShift Web 控制台部署队列管理器』

OpenShift CP4I 使用 *Red Hat OpenShift CLI* 为 *IBM MQ* 准备 *Red Hat OpenShift* 项目

准备 Red Hat OpenShift Container Platform 集群，以便它可以使用 IBM MQ Operator 部署队列管理器。 此任务应由项目管理员完成。

开始之前

注: 如果计划在已安装其他 IBM Cloud Pak for Integration 组件的项目中使用 IBM MQ ，那么可能不需要遵循这些指示信息。

使用 **`cloudctl login`** (对于 IBM Cloud Pak for Integration) 或 **`oc login`** 登录到集群。

关于此任务

将从执行许可证权利检查的容器注册表中拉取 IBM MQ Operator 映像。 此检查需要存储在 `docker-registry` 拉取私钥中的权利密钥。 如果您还没有权利密钥，请遵循以下指示信息以获取权利密钥并创建拉取私钥。

过程

1. 获取分配给您的标识的权利密钥。

   a) 使用与授权软件关联的 IBM 标识和密码登录到 MyIBM Container Software Library 。

   b) 在**权利密钥**部分中，选择**复制密钥**以将权利密钥复制到剪贴板。

2. 在要部署队列管理器的项目中创建包含权利密钥的私钥。

   运行以下命令，其中 *<entitlement-key>* 是在步骤 1 中检索的密钥， *<user-email>* 是与授权软件关联的 IBM 标识。

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

下一步做什么

第 76 页的『使用 Red Hat OpenShift CLI 部署队列管理器』

OpenShift CP4I 将队列管理器部署到 Red Hat OpenShift Container Platform 集群

使用 QueueManager 定制资源将队列管理器部署到 Red Hat OpenShift Container Platform 集群上。

过程

- CP4I

  第 75 页的『使用 IBM Cloud Pak for Integration Platform Navigator 部署队列管理器』.

- OpenShift
  第 76 页的『使用 Red Hat OpenShift Web 控制台部署队列管理器』.
- OpenShift
  第 76 页的『使用 Red Hat OpenShift CLI 部署队列管理器』.

相关任务

第 78 页的『配置队列管理器的示例』
可以通过调整 QueueManager 定制资源的内容来配置队列管理器。

CP4I 使用 *IBM Cloud Pak for Integration Platform Navigator* 部署队列管理器

使用 QueueManager 定制资源将队列管理器部署到使用 IBM Cloud Pak for Integration Platform Navigator 的 Red Hat OpenShift Container Platform 集群上。 此任务应由项目管理员完成

开始之前

在浏览器中，启动 IBM Cloud Pak for Integration Platform Navigator。

如果这是首次将队列管理器部署到此 Red Hat OpenShift 项目中，请遵循 第 73 页的『为 IBM MQ 准备 Red Hat OpenShift 项目』的步骤。

过程

1. 部署队列管理器。

   以下示例部署 "快速启动" 队列管理器，该队列管理器使用临时 (非持久) 存储器，并关闭 MQ 安全性。 消息不会在队列管理器重新启动时持久存储。 您可以调整配置以更改许多队列管理器设置。

   a) 在 IBM Cloud Pak for Integration Platform Navigator 中，单击 **管理** ，然后单击 **集成运行时**。 在较低版本的 IBM Cloud Pak for Integration Platform Navigator 中，单击 **运行时和实例**。

   b) 单击**创建实例**。

   c) 选择 **消息传递**，然后单击 **下一步**。 在 IBM Cloud Pak for Integration Platform Navigator 的较旧版本中，单击 **队列管理器**，然后单击 **下一步**。

   将显示用于创建 `QueueManager` 实例的表单。

   注: 您还可以单击 **代码** 以查看或更改 `QueueManager` 配置 YAML。

   d) 在 **详细信息** 部分中，检查或更新 **名称** 字段，并指定要在其中创建队列管理器实例的 **名称空间** 。

   e) 如果您接受 IBM Cloud Pak for Integration 许可协议，请将 **许可接受** 更改为 **开启**。

   您必须接受部署队列管理器的许可证。

   f) 在 **队列管理器** 部分中，检查或更新底层队列管理器的 **名称** 。 在较低版本的 IBM Cloud Pak for Integration Platform Navigator 中，使用 **队列管理器配置** 部分。

   缺省情况下， IBM MQ 客户机应用程序使用的队列管理器的名称将与 `QueueManager` 的名称相同，但除去了任何无效字符 (例如连字符)。

   g) 单击**创建**

   现在将显示当前项目 (名称空间) 中的队列管理器列表。 新 `QueueManager` 的状态应该为 `Pending`

2. 检查队列管理器是否正在运行

   当 `QueueManager` 状态为 `Running` 时，将完成创建。

相关任务

第 98 页的『配置路由以从 Red Hat OpenShift 集群外部连接到队列管理器』
您需要 Red Hat OpenShift 路由以将应用程序从 Red Hat OpenShift 集群外部连接到 IBM MQ 队列管理器。 必须在 IBM MQ 队列管理器和客户机应用程序上启用 TLS ，因为仅当使用 TLS 1.2 或更高版本的协议时，SNI 才在 TLS 协议中可用。 Red Hat OpenShift Container Platform Router 使用 SNI 将请求路由到 IBM MQ 队列管理器。

第 104 页的『连接到 Red Hat OpenShift 集群中部署的 IBM MQ Console』
如何连接到已部署到 Red Hat OpenShift Container Platform 集群的队列管理器的 IBM MQ Console 。

OpenShift CP4I 使用 *Red Hat OpenShift Web* 控制台部署队列管理器

使用 QueueManager 定制资源，通过 Red Hat OpenShift Web 控制台将队列管理器部署到 Red Hat OpenShift Container Platform 集群上。 此任务应由项目管理员完成

开始之前

登录到 Red Hat OpenShift 集群 Web 控制台。 您将需要选择要使用的现有项目 (名称空间)，或者创建新的项目 (名称空间)。

如果这是首次将队列管理器部署到此 Red Hat OpenShift 项目中，请遵循 第 73 页的『为 IBM MQ 准备 Red Hat OpenShift 项目』的步骤。

过程

1. 部署队列管理器。

   以下示例部署 "快速启动" 队列管理器，该队列管理器使用临时 (非持久) 存储器，并关闭 MQ 安全性。 消息不会在队列管理器重新启动时持久存储。 您可以调整配置以更改许多队列管理器设置。

   a) 在 Red Hat OpenShift Web 控制台中，从导航窗格单击 **操作程序** > **已安装的操作程序**

   b) 单击 **IBM MQ**。

   c) 单击 **队列管理器** 选项卡。

   d) 单击 **创建 QueueManager** 按钮。

   将显示 YAML 编辑器，其中包含 `QueueManager` 资源的示例 YAML。

   注: 您还可以单击 **编辑表单** 以查看或更改 `QueueManager` 配置。

   e) 如果您接受许可协议，请将 **许可接受** 更改为 **开启**。

   IBM MQ 在多个不同的许可证下可用。 有关有效许可证的更多信息，请参阅 第 114 页的『mq.ibm.com/v1beta1 的许可参考』。 您必须接受部署队列管理器的许可证。

   f) 单击**创建**

   现在将显示当前项目 (名称空间) 中的队列管理器列表。 新的 `QueueManager` 应处于 `Pending` 状态。

2. 检查队列管理器是否正在运行

   当 `QueueManager` 状态为 `Running` 时，将完成创建。

相关任务

第 98 页的『配置路由以从 Red Hat OpenShift 集群外部连接到队列管理器』
您需要 Red Hat OpenShift 路由以将应用程序从 Red Hat OpenShift 集群外部连接到 IBM MQ 队列管理器。 必须在 IBM MQ 队列管理器和客户机应用程序上启用 TLS，因为仅当使用 TLS 1.2 或更高版本的协议时，SNI 才在 TLS 协议中可用。 Red Hat OpenShift Container Platform Router 使用 SNI 将请求路由到 IBM MQ 队列管理器。

第 104 页的『连接到 Red Hat OpenShift 集群中部署的 IBM MQ Console』
如何连接到已部署到 Red Hat OpenShift Container Platform 集群的队列管理器的 IBM MQ Console 。

OpenShift CP4I 使用 *Red Hat OpenShift CLI* 部署队列管理器

使用 QueueManager 定制资源，通过命令行界面 (CLI) 将队列管理器部署到 Red Hat OpenShift Container Platform 集群上。 此任务应由项目管理员完成

开始之前

您需要安装 Red Hat OpenShift Container Platform 命令行界面。

使用 **`cloudctl login`** (对于 IBM Cloud Pak for Integration) 或 **`oc login`** 登录到集群。

如果这是首次将队列管理器部署到此 Red Hat OpenShift 项目中，请遵循 第 73 页的『为 IBM MQ 准备 Red Hat OpenShift 项目』的步骤。

过程

1. 部署队列管理器。

   以下示例部署 "快速启动" 队列管理器，该队列管理器使用临时 (非持久) 存储器，并关闭 MQ 安全性。 消息不会在队列管理器重新启动时持久存储。 您可以调整 YAML 的内容以更改许多队列管理器设置。

   a) 创建 `QueueManager` YAML 文件

   例如，要在 IBM Cloud Pak for Integration 中安装基本队列管理器，请创建具有以下内容的文件 "mq-quickstart.yaml":

```
apiVersion: mq.ibm.com/v1beta1
kind: QueueManager
metadata:
  name: quickstart-cp4i
spec:
  version: 9.2.5.0-r3
  license:
    accept: false
    license: L-RJON-C7QG3S
    use: NonProduction
  web:
    enabled: true
  queueManager:
    name: "QUICKSTART"
    storage:
      queueManager:
        type: ephemeral
  template:
    pod:
      containers:
       - name: qmgr
         env:
         - name: MQSNOAUT
           value: "yes"
```

   要点: 如果您接受 IBM Cloud Pak for Integration 许可协议，请将 `accept: false` 更改为 `accept: true`。 请参阅 第 114 页的『mq.ibm.com/v1beta1 的许可参考』 以获取有关许可证的详细信息。

   此示例还包含随队列管理器一起部署的 Web 服务器，以及随 IBM Cloud Pak Identity and Access Manager 一起启用单点登录的 Web 控制台。

   要独立于 IBM Cloud Pak for Integration 安装基本队列管理器，请创建具有以下内容的文件 "mq-quickstart.yaml":

```
apiVersion: mq.ibm.com/v1beta1
kind: QueueManager
metadata:
  name: quickstart
spec:
  version: 9.2.5.0-r3
  license:
    accept: false
    license: L-APIG-BZDDDY
  web:
    enabled: true
  queueManager:
    name: "QUICKSTART"
    storage:
      queueManager:
        type: ephemeral
  template:
    pod:
      containers:
       - name: qmgr
         env:
         - name: MQSNOAUT
           value: "yes"
```

   重要信息:如果您接受 MQ 许可协议，请将 `accept: false` 更改为 `accept: true`。 请参阅 第 114 页的『mq.ibm.com/v1beta1 的许可参考』 以获取有关许可证的详细信息。

   b) 创建 `QueueManager` 对象

```
oc apply -f mq-quickstart.yaml
```

2. 检查队列管理器是否正在运行
   您可以通过运行以下命令来验证部署:

```
oc describe queuemanager <QueueManagerResourceName>
```

   ，然后检查状态。
   例如，运行

```
oc describe queuemanager quickstart
```

   ，并检查 `status.Phase` 字段是否指示 `Running`

相关任务

第 98 页的『配置路由以从 Red Hat OpenShift 集群外部连接到队列管理器』
您需要 Red Hat OpenShift 路由以将应用程序从 Red Hat OpenShift 集群外部连接到 IBM MQ 队列管理器。必须在 IBM MQ 队列管理器和客户机应用程序上启用 TLS ，因为仅当使用 TLS 1.2 或更高版本的协议时，SNI 才在 TLS 协议中可用。 Red Hat OpenShift Container Platform Router 使用 SNI 将请求路由到 IBM MQ 队列管理器。

第 104 页的『连接到 Red Hat OpenShift 集群中部署的 IBM MQ Console』
如何连接到已部署到 Red Hat OpenShift Container Platform 集群的队列管理器的 IBM MQ Console 。

OpenShift CP4I 配置队列管理器的示例

可以通过调整 QueueManager 定制资源的内容来配置队列管理器。

关于此任务

使用以下示例来帮助您使用 QueueManager YAML 文件配置队列管理器。

过程

- 第 78 页的『示例: 提供 MQSC 和 INI 文件』
- 第 79 页的『示例: 配置 TLS』

OpenShift CP4I 示例: 提供 *MQSC* 和 *INI* 文件

此示例创建包含两个 MQSC 文件和一个 INI 文件的 Kubernetes ConfigMap 。 然后部署队列管理器以处理这些 MQSC 和 INI 文件。

关于此任务

部署队列管理器时，可以提供 MQSC 和 INI 文件。 MQSC 和 INI 数据必须在一个或多个 Kubernetes ConfigMaps 和 Secrets 中定义。 必须在将部署队列管理器的名称空间 (项目) 中创建这些名称空间。

注: 当 MQSC 或 INI 文件包含敏感数据时，应使用 Kubernetes 私钥。

以此方式提供 MQSC 和 INI 需要 IBM MQ Operator 1.1 或更高版本。

示例

以下示例创建包含两个 MQSC 文件和一个 INI 文件的 Kubernetes ConfigMap 。 然后部署队列管理器以处理这些 MQSC 和 INI 文件。

示例 ConfigMap -在集群中应用以下 YAML:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: mqsc-ini-example
data:
  example1.mqsc: |
```

```
    DEFINE QLOCAL('DEV.QUEUE.1') REPLACE
    DEFINE QLOCAL('DEV.QUEUE.2') REPLACE
  example2.mqsc: |
    DEFINE QLOCAL('DEV.DEAD.LETTER.QUEUE') REPLACE
  example.ini: |
    Channels:
       MQIBindType=FASTPATH
```

示例 QueueManager -使用命令行或 IBM Cloud Pak for Integration Platform Navigator 通过以下配置部署队列管理器:

```
apiVersion: mq.ibm.com/v1beta1
kind: QueueManager
metadata:
  name: mqsc-ini-cp4i
spec:
  version: 9.2.5.0-r3
  license:
    accept: false
    license: L-RJON-C7QG3S
    use: NonProduction
  web:
    enabled: true
  queueManager:
    name: "MQSCINI"
    mqsc:
    - configMap:
        name: mqsc-ini-example
        items:
        - example1.mqsc
        - example2.mqsc
    ini:
    - configMap:
        name: mqsc-ini-example
        items:
        - example.ini
    storage:
      queueManager:
        type: ephemeral
```

要点: 如果您接受 IBM Cloud Pak for Integration 许可协议，请将 `accept: false` 更改为 `accept: true`。 有关许可证的详细信息，请参阅 mq.ibm.com/v1beta1 的许可证发放参考 。

其他信息:

- 可以将队列管理器配置为使用单个 Kubernetes ConfigMap 或 Secret (如本示例中所示) 或多个 Kubernetes ConfigMaps 和 Secret。
- 您可以选择使用 Kubernetes ConfigMap 或 Secret 中的所有 MQSC 和 INI 数据 (如本示例中所示) ，或者将每个队列管理器配置为仅使用一部分可用文件。
- MQSC 和 INI 文件根据其密钥按字母顺序进行处理。 因此， `example1.mqsc` 将始终在 `example2.mqsc` 之前进行处理，而不考虑它们在队列管理器配置中的显示顺序。
- 如果多个 MQSC 或 INI 文件具有相同的密钥 (跨多个 Kubernetes ConfigMaps 或密钥) ，那么将根据在队列管理器配置中定义文件的顺序来处理这组文件。

OpenShift CP4I Linux 示例: 配置 *TLS*

此示例使用 IBM MQ Operator 将队列管理器部署到 Red Hat OpenShift Container Platform 中。 在样本客户机与队列管理器之间配置单向 TLS 通信。 此示例通过放置和获取消息来演示成功配置。

开始之前

要完成此示例，必须首先完成以下先决条件:

- 安装 IBM MQ client，并将 `samp/bin` 和 `bin` 添加到 *PATH*。 您需要 **runmqakm**， **amqsputc** 和 **amqsgetc** 应用程序，这些应用程序可以作为 IBM MQ client 的一部分进行安装，如下所示:
  - Windows Linux 对于 Windows 和 Linux: 从 https://ibm.biz/mq92redistclients 安装适用于您操作系统的 IBM MQ 可再分发客户机

– macOS 对于 Mac: 下载并设置 IBM MQ MacOS Toolkit: https://developer.ibm.com/tutorials/mq-macos-dev/

- 安装适用于您操作系统的 OpenSSL 工具。
- 为此示例创建 Red Hat OpenShift Container Platform (OCP) 项目/名称空间。
- 在命令行上，登录到 OCP 集群，然后切换到以上名称空间。
- 确保 IBM MQ Operator 已安装并在以上名称空间中可用。

关于此任务

此示例提供了定制资源 YAML，用于定义要部署到 Red Hat OpenShift Container Platform 中的队列管理器。 它还详细说明了在启用 TLS 的情况下部署队列管理器所需的其他步骤。 完成后，放入和获取消息将验证队列管理器是否已配置为使用 TLS。

为 IBM MQ 服务器创建 TLS 专用密钥和证书

以下代码示例显示如何为队列管理器创建自签名证书，以及如何将证书添加到密钥数据库以充当客户机的信任库。 如果您已有专用密钥和证书，那么可以改为使用这些密钥和证书。

请注意，自签名证书仅应用于开发目的。

在当前目录中创建自签名专用密钥和公用证书

运行以下命令：

```
openssl req -newkey rsa:2048 -nodes -keyout tls.key -subj "/CN=localhost" -x509 -days 3650
-out tls.crt
```

将服务器公用密钥添加到客户机密钥数据库

密钥数据库用作客户机应用程序的信任库。

创建客户机密钥数据库：

```
runmqakm -keydb -create -db clientkey.kdb -pw password -type cms -stash
```

将先前生成的公用密钥添加到客户机密钥数据库：

```
runmqakm -cert -add -db clientkey.kdb -label mqservercert -file tls.crt -format ascii
-stashed
```

为队列管理器部署配置 TLS 证书

因此，队列管理器可以引用和应用密钥和证书，创建 Kubernetes TLS 私钥，并引用上面创建的文件。 执行此操作时，请确保您位于开始此任务之前创建的名称空间中。

```
oc create secret tls example-tls-secret --key="tls.key" --cert="tls.crt"
```

创建包含 MQSC 命令的配置映射

创建包含 MQSC 命令的 Kubernetes 配置映射，以创建新队列和 SVRCONN 通道，并添加通道认证记录，通过仅阻止那些称为 没人的用户来访问通道。

请注意，此方法应仅用于开发目的。

确保您位于先前创建的名称空间中 (请参阅 开始之前)，然后在 OCP UI 中或使用命令行输入以下 YAML。

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: example-tls-configmap
data:
  tls.mqsc: |
    DEFINE QLOCAL('EXAMPLE.QUEUE') REPLACE
    DEFINE CHANNEL(SECUREQMCHL) CHLTYPE(SVRCONN) TRPTYPE(TCP) SSLCAUTH(OPTIONAL)
SSLCIPH('ANY_TLS12_OR_HIGHER')
    SET CHLAUTH(SECUREQMCHL) TYPE(BLOCKUSER) USERLIST('nobody') ACTION(ADD)
```

创建所需的 OCP 路由

请确保您位于开始此任务之前创建的名称空间中，然后在 OCP UI 中输入以下 YAML 或使用命令行。

```
apiVersion: route.openshift.io/v1
kind: Route
metadata:
  name: example-tls-route
spec:
  host: secureqmchl.chl.mq.ibm.com
  to:
    kind: Service
    name: secureqm-ibm-mq
  port:
    targetPort: 1414
  tls:
    termination: passthrough
```

请注意， Red Hat OpenShift Container Platform Router 使用 SNI 将请求路由到 IBM MQ 队列管理器。 如果在先前创建的配置映射中更改 MQSC 中指定的通道名称，那么还需要更改此处的主机字段以及稍后创建的 CCDT 文件中的主机字段。 有关更多信息，请参阅第 98 页的『配置路由以从 Red Hat OpenShift 集群外部连接到队列管理器』。

部署队列管理器

重要信息: 在此示例中，我们使用 *MQSNOAUT* 变量来禁用队列管理器上的授权，这使我们能够关注使用 TLS 连接客户机所需的步骤。 在 IBM MQ 的生产部署中建议不要这样做，因为这会导致任何连接的应用程序都具有完整的管理权限，而没有任何机制来降低个别应用程序的许可权。

使用以下定制资源 YAML 创建新的队列管理器。 请注意，它引用先前创建的配置映射和私钥以及 *MQSNOAUT* 变量。

确保您位于开始此任务之前创建的名称空间中，然后使用命令行或使用 IBM Cloud Pak for Integration Platform Navigator 在 OCP UI 中输入以下 YAML。 检查是否指定了正确的许可证，并通过将 `false` 更改为 `true` 来接受该许可证。

```
apiVersion: mq.ibm.com/v1beta1
kind: QueueManager
metadata:
  name: secureqm
spec:
  license:
    accept: false
    license: L-RJON-C7QG3S
    use: Production
  queueManager:
    name: SECUREQM
    mqsc:
    - configMap:
        name: example-tls-configmap
        items:
        - tls.mqsc
    storage:
      queueManager:
        type: ephemeral
  template:
    pod:
      containers:
        - env:
            - name: MQSNOAUT
              value: 'yes'
          name: qmgr
  version: 9.2.5.0-r3
  web:
    enabled: true
  pki:
    keys:
      - name: example
        secret:
          secretName: example-tls-secret
          items:
          - tls.key
          - tls.crt
```

确认队列管理器正在运行

现在正在部署队列管理器。 请先确认其处于 Running 状态，然后再继续。 例如：

```
oc get qmgr secureqm
```

测试与队列管理器的连接

要确认为单向 TLS 通信配置了队列管理器，请使用 **amqsputc** 和 **amqsgetc** 样本应用程序：

查找队列管理器主机名

使用以下命令来查找路由 secureqm-ibm-mq-qm 的队列管理器标准主机名：

```
oc get routes secureqm-ibm-mq-qm
```

指定队列管理器详细信息

创建用于指定队列管理器详细信息的文件 CCDT.JSON 。 将主机值替换为上一步中的主机名。

```
{
    "channel":
    [
        {
            "name": "SECUREQMCHL",
            "clientConnection":
            {
                "connection":
                [
                    {
                        "host": "<hostname from previous step>",
                        "port": 443
                    }
                ],
                "queueManager": "SECUREQM"
            },
            "transmissionSecurity":
            {
              "cipherSpecification": "ECDHE_RSA_AES_128_CBC_SHA256"
            },
            "type": "clientConnection"
        }
   ]
}
```

导出环境变量

以适合您的操作系统的方式导出以下环境变量。 **amqsputc** 和 **amqsgetc** 将读取这些变量。

更新系统上文件的路径：

```
export MQCCDTURL='<full path to file>/CCDT.JSON'
export MQSSLKEYR='<full path to file>/clientkey'
```

将消息放入队列

运行以下命令：

```
amqsputc EXAMPLE.QUEUE SECUREQM
```

如果成功连接到队列管理器，那么将输出以下响应：

```
target queue is EXAMPLE.QUEUE
```

通过输入一些文本，然后每次按 **Enter** 键，将多条消息放入队列。

要完成此操作，请按两次 **Enter** 键。

从队列检索消息

运行以下命令：

```
amqsgetc EXAMPLE.QUEUE SECUREQM
```

您在上一步中添加的消息已被使用，并且已输出。

几秒钟后，命令退出。

恭喜您成功部署了启用了 TLS 的队列管理器，并显示您可以从客户机安全地将消息放入队列管理器。

OpenShift CP4I 示例: 定制许可证服务注释

IBM MQ Operator 会自动向已部署的资源添加 IBM License Service 注释。 这些受 IBM License Service 监视，并生成对应于所需权利的报告。

关于此任务

IBM MQ Operator 添加的注释是标准情境中期望的注释，并且基于在队列管理器部署期间选择的许可证值。

示例

如果 **License** 设置为 `L-RJON-BZFQU2 (IBM Cloud Pak for Integration 2021.2.1)`，并且 **Use** 设置为 `NonProduction`，那么将应用以下注释:

- cloudpakId: c8b82d189e7545f0892db9ef2731b90d
- cloudpakName: IBM Cloud Pak for Integration
- productCharged 容器 :qmgr
- productCloudpak 比率: "4:1"
- productID: 21dfe9a0f00f444f888756d835334909
- productName: IBM MQ Advanced for Non-Production
- productMetric:VIRTUAL_PROCESSOR_CORE
- productVersion: 9.2.3.0

在 IBM Cloud Pak for Integration 中， IBM App Connect Enterprise 的部署包含 IBM MQ 的受限权利。 在这些情况下，需要覆盖这些注释以确保 IBM License Service 捕获正确的用法。 要执行此操作，请使用 第 103 页的『向队列管理器资源添加定制注释和标签』中描述的方法。

例如，如果 IBM MQ 部署在 IBM App Connect Enterprise 权利下，请使用以下代码片段中显示的方法:

```
apiVersion: mq.ibm.com/v1beta1
kind: QueueManager
metadata:
  name: mq4ace
  namespace: cp4i
spec:
  annotations:
    productMetric: FREE
```

许可注释可能需要修改的其他两个常见原因:

1. IBM MQ Advanced 包含在另一个 IBM 产品的权利中。
   - 在此情况下，请使用先前为 IBM App Connect Enterprise 描述的方法。
2. IBM MQ 是在 IBM Cloud Pak for Integration 许可证下部署的。
   - 如果您具有 IBM Cloud Pak for Integration 许可证，那么可以决定以 IBM MQ 或 IBM MQ Advanced 比率部署队列管理器。 如果在 IBM MQ 比率下进行部署，那么必须确保不使用任何高级功能，例如本机 HA 或 Advanced Message Security。
   - 在此情况下，请将以下注释用于生产用途:

     ```
     apiVersion: mq.ibm.com/v1beta1
     kind: QueueManager
     metadata:
       name: mq4ace
       namespace: cp4i
     spec:
       annotations:
         productID: c661609261d5471fb4ff8970a36bccea
         productCloudpakRatio: '4:1'
         productName: IBM MQ for Production
         productMetric: VIRTUAL_PROCESSOR_CORE
     ```

   - 将以下注释用于非生产用途:

```
apiVersion: mq.ibm.com/v1beta1
kind: QueueManager
metadata:
  name: mq4ace
  namespace: cp4i
spec:
  annotations:
    productID: 151bec68564a4a47a14e6fa99266deff
    productCloudpakRatio: '8:1'
    productName: IBM MQ for Non-Production
    productMetric: VIRTUAL_PROCESSOR_CORE
```

OpenShift CP4I 使用 IBM MQ Operator 为队列管理器配置高可用性

关于此任务

过程

- V 9.2.3
  第 84 页的『本机 HA』.
- V 9.2.3
  第 85 页的『示例: 配置本机 HA 队列管理器』.
- 第 93 页的『示例: 配置多实例队列管理器』.

CD CP4I V 9.2.3 本机 *HA*

本机 HA 是适用于 IBM MQ 的本机 (内置) 高可用性解决方案，适用于云块存储器。

本机 HA 配置提供了一个高可用性队列管理器，其中可恢复的 MQ 数据 (例如，消息) 在多个存储器集中进行复制，从而防止因存储器故障而丢失。 队列管理器由多个正在运行的实例组成，其中一个实例是引导者，其他实例准备好在发生故障时快速接管，从而最大化对队列管理器及其消息的访问权。

本机 HA 配置由三个 Kubernetes pod 组成，每个 pod 都具有队列管理器的实例。 一个实例是活动队列管理器，用于处理消息并写入其恢复日志。 每当写入恢复日志时，活动队列管理器都会将数据发送到其他两个实例 (称为副本)。 每个副本写入自己的恢复日志，确认数据，然后从复制的恢复日志更新自己的队列数据。 如果运行活动队列管理器的 pod 失败，那么队列管理器的其中一个副本实例将接管活动角色并具有要使用的当前数据。

日志类型称为 "复制日志"。 复制日志本质上是线性日志，启用了自动日志管理和自动介质映像。 请参阅 日志记录类型。 您可以使用用于管理线性日志的相同方法来管理复制的日志。

Kubernetes Service 用于将 TCP/IP 客户机连接路由到当前活动实例，该实例被标识为可供网络流量使用的唯一 pod。 发生这种情况时，不需要客户机应用程序了解不同的实例。

三个 pod 用于大大降低出现裂脑情况的可能性。 在双 pod 高可用性系统中，当两个 pod 之间的连接中断时，可能会发生裂脑。 在没有连接的情况下，两个 pod 都可以同时运行队列管理器，从而累积不同的数据。 在恢复连接时，将有两个不同版本的数据 ("分割-大脑") ，需要手动干预来决定要保留的数据集以及要废弃的数据集。

本机 HA 使用具有定额的三个 pod 系统来避免裂脑情况。 可以与至少一个其他 pod 进行通信的 pod 构成定额。 队列管理器只能成为具有定额的 pod 上的活动实例。 队列管理器无法在未连接到至少一个其他 pod 的 pod 上变为活动状态，因此绝不会同时存在两个活动实例:

- 如果单个 pod 发生故障，那么其他两个 pod 中的一个 pod 上的队列管理器可以接管。 如果两个 pod 发生故障，那么队列管理器无法成为其余 pod 上的活动实例，因为该 pod 没有定额 (其余 pod 无法判断其他两个 pod 是否已发生故障，或者它们仍在运行并且已失去连接)。
- 如果单个 pod 失去连接，那么队列管理器无法在此 pod 上变为活动状态，因为该 pod 没有定额。 其余两个 pod 中的一个 pod 上的队列管理器可以接管，这些 pod 具有定额。 如果所有 pod 都失去连接，那么队列管理器无法在任何 pod 上变为活动状态，因为没有任何 pod 具有定额。

如果活动 pod 发生故障并随后恢复，那么它可以以副本角色重新加入组。

下图显示了在三个容器中部署了队列管理器的三个实例的典型部署。

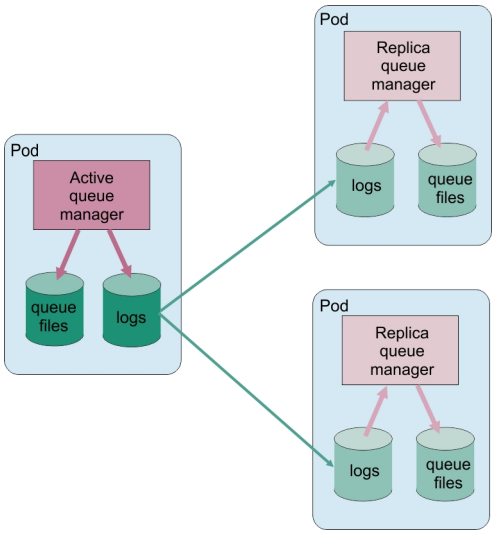

图 *1:* 本机 *HA* 配置示例

CD CP4I V 9.2.3 使用 *IBM MQ Operator* 配置本机 *HA*

本机 HA 是使用 `QueueManager` API 配置的，高级选项是使用 INI 文件提供的。

本机 HA 是使用 `QueueManager` API 的 `.spec.queueManager.availability` 配置的，例如:

```
apiVersion: mq.ibm.com/v1beta1
kind: QueueManager
metadata:
name: nativeha-example
spec:
  license:
    accept: false
    license: L-RJON-C7QG3S
    use: Production
  queueManager:
    availability:
      type: NativeHA
  version: 9.2.5.0-r3
```

`.spec.queueManager.availability.type` 字段必须设置为 `NativeHA`。

本机 HA 在 IBM MQ 9.2.3 或更高版本中可用。

在 `.spec.queueManager.availability` 下，您还可以配置要在复制时在队列管理器实例之间使用的 TLS 密钥和密码。 强烈建议您这样做， 第 85 页的『示例: 配置本机 HA 队列管理器』中提供了逐步指南。

相关参考

第 85 页的『示例: 配置本机 HA 队列管理器』
此示例显示如何使用本机高可用性功能将队列管理器部署到使用 IBM MQ Operator 的 Red Hat OpenShift Container Platform (OCP) 中。

OpenShift CD CP4I Linux V 9.2.3 示例*:* 配置本机 *HA* 队列管理器

此示例显示如何使用本机高可用性功能将队列管理器部署到使用 IBM MQ Operator 的 Red Hat OpenShift Container Platform (OCP) 中。

开始之前

要完成此示例，必须首先完成以下先决条件:

- 安装 IBM MQ client，并将已安装的 `samp/bin` 和 `bin` 目录添加到 *PATH*。 客户机提供此示例所需的 **`runmqakm`**， **`amqsputc`** 和 **`amqsgetc`** 应用程序。 按如下所示安装 IBM MQ client :

– Windows Linux 对于 Windows 和 Linux: 从 https://ibm.biz/mq92redistclients 安装适用于您操作系统的 IBM MQ 可再分发客户机

– macOS 对于 Mac: 下载并设置 IBM MQ MacOS Toolkit。 请参阅 https://ibm.biz/mqdevmacclient。

- 安装适用于您操作系统的 OpenSSL 工具。 如果您还没有专用密钥和证书，那么需要这样才能为队列管理器生成自签名证书。
- 为此示例创建 Red Hat OpenShift Container Platform (OCP) 项目/名称空间，并执行任务 第 73 页的『为 IBM MQ 准备 Red Hat OpenShift 项目』 中的步骤
- 在命令行上，登录到 OCP 集群，然后切换到刚刚创建的名称空间。
- 确保 IBM MQ Operator 已安装并且在名称空间中可用。
- 在 OCP 中配置要由队列管理器使用的缺省存储类。 如果要在不设置缺省存储类的情况下完成本教程，请参阅 注 2: 使用非缺省存储类。

关于本任务

本机 HA 队列管理器涉及一个活动和两个副本 Kubernetes Pod。 它们作为 Kubernetes 有状态集的一部分运行，其中正好有三个副本和一组 Kubernetes 持久卷。 有关本机 HA 队列管理器的更多信息，请参阅 第 14 页的『容器中 IBM MQ 的高可用性』。

此示例提供了定制资源 YAML ，用于定义使用持久存储器并使用 TLS 配置的本机 HA 队列管理器。 将队列管理器部署到 OCP 后，将模拟活动队列管理器 pod 的故障。 您会看到自动恢复发生，并通过在失败后放入和获取消息来证明它已成功。

示例

为 MQ 服务器创建 TLS 专用密钥和证书

您可以为队列管理器创建自签名证书，并将该证书添加到密钥数据库以充当客户机的信任库。 如果您已有专用密钥和证书，那么可以改为使用这些密钥和证书。 请注意，您只应该将自签名证书用于开发目的。

要在当前目录中创建自签名专用密钥和公用证书，请运行以下命令:

```
openssl req -newkey rsa:2048 -nodes -keyout tls.key -subj "/CN=localhost" -x509 -days 3650
-out tls.crt
```

创建 TLS 专用密钥和证书以供本机 HA 内部使用

本机 HA 队列管理器中的三个 pod 通过网络复制数据。 您可以创建自签名证书以在内部复制时使用。 请注意，您只应该将自签名证书用于开发目的。

要在当前目录中创建自签名专用密钥和公用证书，请运行以下命令:

```
openssl req -newkey rsa:2048 -nodes -keyout nativeha.key -subj "/CN=localhost" -x509 -days
3650 -out nativeha.crt
```

将队列管理器公用密钥添加到客户机密钥数据库

客户机密钥数据库用作客户机应用程序的信任库。

创建客户机密钥数据库:

```
runmqakm -keydb -create -db clientkey.kdb -pw password -type cms -stash
```

将先前生成的公用密钥添加到客户机密钥数据库:

```
runmqakm -cert -add -db clientkey.kdb -label mqservercert -file tls.crt -format ascii
-stashed
```

为队列管理器部署创建包含 TLS 证书的私钥

因此，您的队列管理器可以引用并应用密钥和证书，创建 Kubernetes TLS 私钥，并引用上面创建的文件。 执行此操作时，请确保您位于开始此任务之前创建的名称空间中。

```
oc create secret tls example-ha-secret --key="tls.key" --cert="tls.crt"
```

创建包含内部本机 HA TLS 证书和密钥的私钥

因此，您的队列管理器可以引用并应用密钥和证书，创建 Kubernetes TLS 私钥，并引用上面创建的文件。 执行此操作时，请确保您位于开始此任务之前创建的名称空间中。

```
oc create secret tls example-ha-secret-internal --key="nativeha.key" --cert="nativeha.crt"
```

创建包含 MQSC 命令的配置映射

创建包含 MQSC 命令的 Kubernetes 配置映射，以创建新队列和 SVRCONN 通道，并添加通道认证记录，通过仅阻止那些称为 没人的用户来允许访问通道。

请注意，此方法应仅用于开发目的。

确保您位于先前创建的名称空间中 (请参阅 第 85 页的『开始之前』)，然后在 OCP UI 中输入以下 YAML，或者使用命令行:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: example-mi-configmap
data:
  tls.mqsc: |
    DEFINE QLOCAL('EXAMPLE.QUEUE') DEFPSIST(YES) REPLACE
    DEFINE CHANNEL(HAQMCHL) CHLTYPE(SVRCONN) TRPTYPE(TCP) SSLCAUTH(OPTIONAL)
SSLCIPH('ANY_TLS12_OR_HIGHER')
    SET CHLAUTH(HAQMCHL) TYPE(BLOCKUSER) USERLIST('nobody') ACTION(ADD)
```

配置路由

如果您在 IBM MQ 9.2.1 或更高版本上使用 IBM MQ client 或 Toolkit，那么可以使用队列管理器配置文件 (INI 文件) 来配置到队列管理器的路由。 在该文件中，将 *OutboundSNI* 变量设置为基于主机名而不是通道名称进行路由。

在运行命令的目录中创建一个名为 `mqclient.ini` 的文件，其中正好包含以下文本:

```
SSL:
   OutboundSNI=HOSTNAME
```

请勿更改此 INI 文件中的任何值。 例如，不得更改字符串 `HOSTNAME`。

有关更多详细信息，请参阅 客户机配置文件的 SSL 节。

如果您使用的是早于 IBM MQ 9.2.1 的 IBM MQ client 或 Toolkit，那么需要创建 OCP 路径而不是先前的配置文件。 遵循 注 1 中的步骤: 创建路径。

部署队列管理器

重要信息: 在此示例中，我们使用 *MQSNOAUT* 变量来禁用队列管理器上的授权，这使我们能够关注使用 TLS 连接客户机所需的步骤。 在 IBM MQ 的生产部署中建议不要这样做，因为这会导致任何连接的应用程序都具有完整的管理权限，而没有任何机制来降低个别应用程序的许可权。

复制并更新以下 YAML。

- 请确保指定了正确的许可证。 请参阅 mq.ibm.com/v1beta1。 在 IBM Cloud Pak for Integration 2021.1.1 中，许可证必须是评估许可证 `L-RJON-BYRMYW`
- 通过将 `false` 更改为 `true` 来接受许可证。

队列管理器定制资源 YAML:

```
apiVersion: mq.ibm.com/v1beta1
kind: QueueManager
metadata:
  name: nativeha-example
spec:
```

```
  license:
    accept: false
    license: L-RJON-C7QG3S
    use: Production
  queueManager:
    name: HAEXAMPLE
    availability:
      type: NativeHA
      tls:
        secretName: example-ha-secret-internal
    mqsc:
    - configMap:
        name: example-mi-configmap
        items:
        - tls.mqsc
  template:
    pod:
      containers:
        - env:
            - name: MQSNOAUT
              value: 'yes'
          name: qmgr
  version: 9.2.5.0-r3
  pki:
    keys:
      - name: example
        secret:
          secretName: example-ha-secret
          items:
          - tls.key
          - tls.crt
```

确保您位于先前创建的名称空间中，使用 Red Hat OpenShift Container Platform Web 控制台， 命令行或使用 IBM Cloud Pak for Integration Platform Navigator 部署更新的 YAML。

系统配置本机 HA 队列管理器时存在短暂延迟，之后队列管理器应该可供使用。

验证

在本部分中，我们将验证队列管理器的行为是否与预期相同。

确认队列管理器正在运行

现在正在部署队列管理器。 请先确认其处于 `Running` 状态，然后再继续。 例如：

```
oc get qmgr nativeha-example
```

测试与队列管理器的连接

要确认为单向 TLS 通信配置了队列管理器，请使用 **amqsputc** 和 **amqsgetc** 样本应用程序：

查找队列管理器主机名

要查找路由 `nativeha-example-ibm-mq-qm` 的队列管理器主机名，请运行以下命令。 将在 `HOST` 字段中返回主机名。

```
oc get routes nativeha-example-ibm-mq-qm
```

指定队列管理器详细信息

创建用于指定队列管理器详细信息的文件 `CCDT.JSON` 。 将主机值替换为上一步返回的主机名。

```
{
    "channel":
    [
        {
            "name": "HAQMCHL",
            "clientConnection":
            {
                "connection":
                [
                    {
                        "host": "<host from previous step>",
                        "port": 443
                    }
                ],
                "queueManager": "HAEXAMPLE"
            },
```

```
            "transmissionSecurity":
            {
              "cipherSpecification": "ECDHE_RSA_AES_128_CBC_SHA256"
            },
            "type": "clientConnection"
        }
    ]
}
```

导出环境变量

以适合您的操作系统的方式导出以下环境变量。 **amqsputc** 和 **amqsgetc** 将读取这些变量。

更新系统上文件的路径:

```
export MQCCDTURL='<full_path_to_file>/CCDT.JSON'
export MQSSLKEYR='<full_path_to_file>/clientkey'
```

将消息放入队列

运行以下命令:

```
amqsputc EXAMPLE.QUEUE HAEXAMPLE
```

如果成功连接到队列管理器，那么将输出以下响应:

```
target queue is EXAMPLE.QUEUE
```

通过输入一些文本，然后每次按 **Enter** 键，将多条消息放入队列。

要完成此操作，请按两次 **Enter** 键。

从队列检索消息

运行以下命令:

```
amqsgetc EXAMPLE.QUEUE HAEXAMPLE
```

您在上一步中添加的消息已被使用，并且已输出。

几秒钟后，命令退出。

强制活动 pod 失败

要验证队列管理器的自动恢复，请模拟 pod 故障:

查看活动和备用 pod

运行以下命令:

```
oc get pods --selector app.kubernetes.io/instance=nativeha-example
```

请注意，在 **READY** 字段中，活动 pod 返回值 `1/1`，而副本 pod 返回值 `0/1`。

删除活动 pod

运行以下命令，并指定活动 pod 的全名:

```
oc delete pod nativeha-example-ibm-mq-<value>
```

再次查看 pod 状态

运行以下命令:

```
oc get pods --selector app.kubernetes.io/instance=nativeha-example
```

查看队列管理器状态

运行以下命令，并指定其他某个 pod 的全名:

```
oc exec -t Pod -- dspmq -o nativeha -x -m HAEXAMPLE
```

您应该会看到状态显示活动实例已更改，例如:

```
QMNAME(HAEXAMPLE) ROLE(Active) INSTANCE(inst1) INSYNC(Yes) QUORUM(3/3)
INSTANCE(inst1) ROLE(Active) REPLADDR(9.20.123.45) CONNACTV(Yes) INSYNC(Yes) BACKLOG(0)
CONNINST(Yes) ALTDATE(2022-01-12) ALTTIME(12.03.44)
INSTANCE(inst2) ROLE(Replica) REPLADDR(9.20.123.46) CONNACTV(Yes) INSYNC(Yes) BACKLOG(0)
CONNINST(Yes) ALTDATE(2022-01-12) ALTTIME(12.03.44)
```

```
INSTANCE(inst3) ROLE(Replica) REPLADDR(9.20.123.47) CONNACTV(Yes) INSYNC(Yes) BACKLOG(0)
CONNINST(Yes) ALTDATE(2022-01-12) ALTTIME(12.03.44)
```

再次放入并获取消息

在备用 pod 变为活动 pod (即，在 READY 字段值变为 1/1 之后) 之后，再次使用以下命令 (如前所述) 将消息放入队列管理器，然后从队列管理器中检索消息:

```
amqsputc EXAMPLE.QUEUE HAEXAMPLE
```

```
amqsgetc EXAMPLE.QUEUE HAEXAMPLE
```

恭喜您成功部署了本机 HA 队列管理器，并显示它可以从 pod 故障中自动恢复。

其他信息

注 1: 创建路由

如果您使用的是早于 IBM MQ 9.2.1 的 IBM MQ client 或 Toolkit ，那么需要创建路径。

要创建路由，请确保您位于先前创建的名称空间中 (请参阅 第 85 页的『开始之前』)，然后在 Red Hat OpenShift Container Platform Web 控制台中或使用命令行输入以下 YAML:

```
apiVersion: route.openshift.io/v1
kind: Route
metadata:
  name: example-mi-route
spec:
  host: hamqchl.chl.mq.ibm.com
  to:
    kind: Service
    name: nativeha-example-ibm-mq
  port:
    targetPort: 1414
  tls:
    termination: passthrough
```

请注意， Red Hat OpenShift Container Platform Router 使用 SNI 将请求路由到 IBM MQ 队列管理器。 如果您更改 包含 MQSC 命令的配置映射中指定的通道名称，那么还必须更改此处的主机字段以及指定队列管理器详细信息的 `CCDT.JSON` 文件中的主机字段。 有关更多信息，请参阅第 98 页的『配置路由以从 Red Hat OpenShift 集群外部连接到队列管理器』。

注 2: 使用非缺省存储类

此示例期望已在 Red Hat OpenShift Container Platform 中配置缺省存储类，因此 队列管理器定制资源 YAML 中不需要任何存储信息。 如果未将存储类配置为缺省值，或者要使用其他存储类，请在 `spec.queueManager.storage` 下添加 `defaultClass: <storage_class_name>`。

存储类名必须与已存在的存储类的名称完全匹配。 即，它必须与命令 `oc get storageclass` 返回的名称匹配。 它还必须支持 `ReadWriteMany`。 有关更多信息，请参阅第 10 页的『IBM MQ Operator 的存储注意事项』。

相关任务

第 90 页的『查看 IBM MQ 认证容器的本机 HA 队列管理器的状态』
对于 IBM MQ 认证的容器，您可以通过在其中一个正在运行的 Pod 中运行 **dspmq** 命令来查看本机 HA 实例的状态。

CD CP4I V9.2.2 查看 *IBM MQ* 认证容器的本机 *HA* 队列管理器的状态

对于 IBM MQ 认证的容器，您可以通过在其中一个正在运行的 Pod 中运行 **dspmq** 命令来查看本机 HA 实例的状态。

关于此任务

要点:

您可以在其中一个正在运行的 Pod 中使用 **dspmq** 命令来查看队列管理器实例的操作状态。 返回的信息取决于实例是活动实例还是副本实例。 活动实例提供的信息是明确的，来自副本节点的信息可能已过时。

您可以执行以下操作:

- 查看当前节点上的队列管理器实例是处于活动状态还是处于副本状态。
- 查看当前节点上实例的本机 HA 操作状态。
- 查看本机 HA 配置中所有三个实例的操作状态。

以下状态字段用于报告本机 HA 配置状态:

角色
指定实例的当前角色，该角色是 `Active`， `Replica` 或 `Unknown` 之一。

INSTANCE
使用 **crtmqm** 命令的 **-lr** 选项创建队列管理器时为此实例提供的名称。

INSYNC
指示实例是否能够作为活动实例进行接管 (如果需要)。

QUORUM
以 *number_of_instances_in-sync*/*number_of_instances_configured* 格式报告定额状态。

REPLADDR
队列管理器实例的复制地址。

连接 ACTV
指示节点是否已连接到活动实例。

BACKLOG
指示实例延迟的 KB 数。

连接
指示指定的实例是否已连接到此实例。

ALTDATE
指示上次更新此信息的日期 (如果从未更新此信息，那么为空白)。

ALTTIME
指示上次更新此信息的时间 (如果从未更新此信息，那么为空白)。

过程

- 查找属于队列管理器的 pod。

```
oc get pod --selector app.kubernetes.io/instance=nativeha-qm
```

- 在其中一个 pod 中运行 `dspmq`

```
oc exec -t Pod dspmq
```

```
oc rsh Pod
```

用于交互式 shell ，您可以在其中直接运行 `dspmq` 。

- 要确定队列管理器实例是作为活动实例运行还是作为副本运行:

```
oc exec -t Pod dspmq -o status -m QMgrName
```

名为 BOB 的队列管理器的活动实例将报告以下状态:

```
QMNAME(BOB)                STATUS(Running)
```

名为 BOB 的队列管理器的副本实例将报告以下状态:

```
QMNAME(BOB)                STATUS(Replica)
```

不活动的实例将报告以下状态:

```
QMNAME(BOB)                STATUS(Ended Immediately)
```

- 要确定指定 pod 中实例的本机 HA 操作状态，请执行以下操作:

```
oc exec -t Pod dspmq -o nativeha -m QMgrName
```

名为 BOB 的队列管理器的活动实例可能会报告以下状态:

```
QMNAME(BOB)               ROLE(Active) INSTANCE(inst1) INSYNC(Yes) QUORUM(3/3)
```

名为 BOB 的队列管理器的副本实例可能会报告以下状态:

```
QMNAME(BOB)               ROLE(Replica) INSTANCE(inst2) INSYNC(Yes) QUORUM(2/3)
```

名为 BOB 的队列管理器的不活动实例可能会报告以下状态:

```
QMNAME(BOB)               ROLE(Unknown) INSTANCE(inst3) INSYNC(no) QUORUM(0/3)
```

- 要确定本机 HA 配置中所有实例的本机 HA 操作状态:

```
oc exec -t Pod dspmq -o nativeha -x -m QMgrName
```

如果在运行队列管理器 BOB 的活动实例的节点上发出此命令，那么可能会收到以下状态:

```
QMNAME(BOB)            ROLE(Active) INSTANCE(inst1) INSYNC(Yes) QUORUM(3/3)
 INSTANCE(inst1) ROLE(Active)  REPLADDR(9.20.123.45) CONNACTV(Yes) INSYNC(Yes) BACKLOG(0)
CONNINST(Yes) ALTDATE(2022-01-12) ALTTIME(12.03.44)
 INSTANCE(inst2) ROLE(Replica) REPLADDR(9.20.123.46) CONNACTV(Yes) INSYNC(Yes) BACKLOG(0)
CONNINST(Yes) ALTDATE(2022-01-12) ALTTIME(12.03.44)
 INSTANCE(inst3) ROLE(Replica) REPLADDR(9.20.123.47) CONNACTV(Yes) INSYNC(Yes) BACKLOG(0)
CONNINST(Yes) ALTDATE(2022-01-12) ALTTIME(12.03.44)
```

如果在运行队列管理器 BOB 的副本实例的节点上发出此命令，那么可能会收到以下状态，这指示其中一个副本落后:

```
QMNAME(BOB)            ROLE(Replica) INSTANCE(inst2) INSYNC(Yes) QUORUM(2/3)
 INSTANCE(inst2) ROLE(Replica) REPLADDR(9.20.123.46) CONNACTV(Yes) INSYNC(Yes) BACKLOG(0)
CONNINST(Yes) ALTDATE(2022-01-12) ALTTIME(12.03.44)
 INSTANCE(inst1) ROLE(Active)  REPLADDR(9.20.123.45) CONNACTV(Yes) INSYNC(Yes) BACKLOG(0)
CONNINST(Yes) ALTDATE(2022-01-12) ALTTIME(12.03.44)
 INSTANCE(inst3) ROLE(Replica) REPLADDR(9.20.123.47) CONNACTV(Yes) INSYNC(No)  BACKLOG(435)
CONNINST(Yes) ALTDATE(2022-01-12) ALTTIME(12.03.44)
```

如果在运行队列管理器 BOB 的不活动实例的节点上发出此命令，那么可能会收到以下状态:

```
QMNAME(BOB)            ROLE(Unknown) INSTANCE(inst3) INSYNC(no) QUORUM(0/3)
 INSTANCE(inst1) ROLE(Unknown) REPLADDR(9.20.123.45) CONNACTV(Unknown) INSYNC(Unknown)
BACKLOG(Unknown) CONNINST(No) ALTDATE() ALTTIME()
 INSTANCE(inst2) ROLE(Unknown) REPLADDR(9.20.123.46) CONNACTV(Unknown) INSYNC(Unknown)
BACKLOG(Unknown) CONNINST(No) ALTDATE() ALTTIME()
 INSTANCE(inst3) ROLE(Unknown) REPLADDR(9.20.123.47) CONNACTV(No) INSYNC(Unknown)
BACKLOG(Unknown) CONNINST(No) ALTDATE() ALTTIME()
```

如果在实例仍在协商哪些是活动的副本时发出该命令，那么您将收到以下状态:

```
QMNAME(BOB)              STATUS(Negotiating)
```

相关参考

dspmq (显示队列管理器) 命令

第 85 页的『示例: 配置本机 HA 队列管理器』

此示例显示如何使用本机高可用性功能将队列管理器部署到使用 IBM MQ Operator 的 Red Hat OpenShift Container Platform (OCP) 中。

CD CP4I V 9.2.3 本机 *HA* 的高级调整

用于调整计时和时间间隔的高级设置。 除非已知缺省值与系统的需求不匹配，否则应该不需要使用这些设置。

用于配置本机 HA 的基本选项使用 `QueueManager` API 进行处理， IBM MQ Operator 使用此 API 为您配置底层队列管理器 INI 文件。 在 NativeHALocal 实例节下，有一些更高级的选项只能使用 INI 文件进行配置。 另请参阅 第 78 页的『示例: 提供 MQSC 和 INI 文件』，以获取有关如何配置 INI 文件的更多信息。

HeartbeatInterval
脉动信号间隔定义本机 HA 队列管理器的活动实例发送网络脉动信号的频率 (以毫秒计)。 脉动信号间隔值的有效范围是 500 (0.5 秒) 到 60000 (1 分钟) ，超出此范围的值将导致队列管理器无法启动。 如果省略此属性，那么将使用缺省值 5000 (5 秒)。 每个实例必须使用相同的脉动信号间隔。

HeartbeatTimeout
脉动信号超时定义本机 HA 队列管理器的副本实例在确定活动实例无响应之前等待的时间长度。 脉动信号间隔超时值的有效范围为 500 (0.5 秒) 到 120000 (2 分钟)。 脉动信号超时的值必须大于或等于脉动信号间隔。

无效值导致队列管理器无法启动。 如果省略此属性，那么副本将在启动进程以选择新的活动实例之前等待 2 x `HeartbeatInterval` 。 每个实例必须使用相同的脉动信号超时。

RetryInterval
重试时间间隔定义本机 HA 队列管理器应重试失败复制链接的频率 (以毫秒计)。 重试时间间隔的有效范围为 500 (0.5 秒) 到 120000 (2 分钟)。 如果省略此属性，那么副本将在重试失败的复制链接之前等待 2 x `HeartbeatInterval` 。

CP4I 结束本机 *HA* 队列管理器

您可以使用 **endmqm** 命令来结束属于本机 HA 组的活动队列管理器或副本队列管理器。

过程

- 要结束队列管理器的活动实例，请参阅本文档的 "配置" 部分中的 结束本机 HA 队列管理器 。

CD CP4I V 9.2.2 在 *IBM Cloud Pak for Integration 2021.1.1* 中评估本机 *HA* 功能

IBM Cloud Pak for Integration 2021.1.1 本机 HA 评估周期已结束。 请使用 IBM Cloud Pak for Integration 2021.2.1 中提供的已更新本机 HA 功能，将 IBM MQ Operator 1.6 或更高版本与 IBM MQ 9.2.3 或更高版本配合使用。

相关任务

第 90 页的『查看 IBM MQ 认证容器的本机 HA 队列管理器的状态』
对于 IBM MQ 认证的容器，您可以通过在其中一个正在运行的 Pod 中运行 **dspmq** 命令来查看本机 HA 实例的状态。

相关参考

第 85 页的『示例: 配置本机 HA 队列管理器』
此示例显示如何使用本机高可用性功能将队列管理器部署到使用 IBM MQ Operator 的 Red Hat OpenShift Container Platform (OCP) 中。

OpenShift CP4I 示例: 配置多实例队列管理器

此示例显示如何使用 IBM MQ Operator 将多实例队列管理器部署到 Red Hat OpenShift Container Platform (OCP) 中。 在此示例中，您还配置样本客户机与队列管理器之间的单向 TLS 通信。 此示例通过在模拟 pod 发生故障之前和之后放置和获取消息来演示成功配置。

开始之前

要完成此示例，必须首先完成以下先决条件:

- 安装 IBM MQ client，并将已安装的 `samp/bin` 和 `bin` 目录添加到 *PATH*。 客户机提供此示例所需的 **runmqakm**， **amqsputc** 和 **amqsgetc** 应用程序。 按如下所示安装 IBM MQ client :
  - Windows Linux 对于 Windows 和 Linux: 从 https://ibm.biz/mq92redistclients 安装适用于您操作系统的 IBM MQ 可再分发客户机
  - macOS 对于 Mac: 下载并设置 IBM MQ MacOS Toolkit。 请参阅 https://developer.ibm.com/tutorials/mq-macos-dev/。
- 安装适用于您操作系统的 OpenSSL 工具。 如果您还没有专用密钥和证书，那么需要这样才能为队列管理器生成自签名证书。
- 为此示例创建 Red Hat OpenShift Container Platform (OCP) 项目/名称空间。

- 在命令行上，登录到 OCP 集群，然后切换到以上名称空间。
- 确保 IBM MQ Operator 已安装并在以上名称空间中可用。
- 在 OCP 中配置要由队列管理器使用的缺省存储类。 如果要在不设置缺省存储类的情况下完成本教程，请参阅 注 2: 使用非缺省存储类。

关于本任务

多实例队列管理器涉及活动和备用 Kubernetes Pod。 它们作为具有正好两个副本和一组 Kubernetes 持久卷的 Kubernetes 有状态集的一部分运行。 有关多实例队列管理器的更多信息，请参阅 第 14 页的『容器中 IBM MQ 的高可用性』。

此示例提供了定制资源 YAML ，用于定义具有持久存储器的多实例队列管理器，并使用 TLS 进行配置。 将队列管理器部署到 OCP 后，将模拟活动队列管理器 pod 的故障。 您会看到自动恢复发生，并通过在失败后放入和获取消息来证明它已成功。

示例

为 MQ 服务器创建 TLS 专用密钥和证书

本部分记录了如何为队列管理器创建自签名证书，以及如何将证书添加到密钥数据库以充当客户机的信任库。 如果您已有专用密钥和证书，那么可以改为使用这些密钥和证书。 请注意，您只应该将自签名证书用于开发目的。

要在当前目录中创建自签名专用密钥和公用证书，请运行以下命令:

```
openssl req -newkey rsa:2048 -nodes -keyout tls.key -subj "/CN=localhost" -x509 -days 3650
-out tls.crt
```

将队列管理器公用密钥添加到客户机密钥数据库

客户机密钥数据库用作客户机应用程序的信任库。

创建客户机密钥数据库:

```
runmqakm -keydb -create -db clientkey.kdb -pw password -type cms -stash
```

将先前生成的公用密钥添加到客户机密钥数据库:

```
runmqakm -cert -add -db clientkey.kdb -label mqservercert -file tls.crt -format ascii
-stashed
```

为队列管理器部署创建包含 TLS 证书的私钥

因此，您的队列管理器可以引用并应用密钥和证书，创建 Kubernetes TLS 私钥，并引用上面创建的文件。 执行此操作时，请确保您位于开始此任务之前创建的名称空间中。

```
oc create secret tls example-mi-secret --key="tls.key" --cert="tls.crt"
```

创建包含 MQSC 命令的配置映射

创建包含 MQSC 命令的 Kubernetes 配置映射，以创建新队列和 SVRCONN 通道，并添加通道认证记录，通过仅阻止那些称为 没人的用户来允许访问通道。

请注意，此方法应仅用于开发目的。

确保您位于先前创建的名称空间中 (请参阅 第 93 页的『开始之前』) ，然后在 OCP UI 中输入以下 YAML ，或者使用命令行:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: example-mi-configmap
data:
  tls.mqsc: |
    DEFINE QLOCAL('EXAMPLE.QUEUE') DEFPSIST(YES) REPLACE
    DEFINE CHANNEL(MIQMCHL) CHLTYPE(SVRCONN) TRPTYPE(TCP) SSLCAUTH(OPTIONAL)
SSLCIPH('ANY_TLS12_OR_HIGHER')
    SET CHLAUTH(MIQMCHL) TYPE(BLOCKUSER) USERLIST('nobody') ACTION(ADD)
```

配置路由

如果您在 IBM MQ 9.2.1 或更高版本上使用 IBM MQ client 或 Toolkit ，那么可以使用队列管理器配置文件 (INI 文件) 来配置到队列管理器的路由。 在该文件中，将 *OutboundSNI* 变量设置为基于主机名而不是通道名称进行路由。

在运行命令的目录中创建一个名为 `mqclient.ini` 的文件，其中包含以下文本:

```
#* Module Name: mqclient.ini                                          *#
#* Type       : IBM MQ MQI client configuration file                  *#
#  Function   : Define the configuration of a client                  *#
#*                                                                    *#
#**********************************************************************#
#* Notes      :                                                       *#
#* 1) This file defines the configuration of a client                 *#
#*                                                                    *#
#**********************************************************************#
SSL:
   OutboundSNI=HOSTNAME
```

注: 请勿更改此页面中的任何值。 例如，应该保留字符串 `HOSTNAME` 。

有关更多详细信息，请参阅 客户机配置文件的 SSL 节。

如果您使用的是早于 IBM MQ 9.2.1 的 IBM MQ client 或 Toolkit ，那么需要创建 OCP 路径而不是先前的配置文件。 遵循 注 1 中的步骤: 创建路径。

部署队列管理器

重要信息: 在此示例中，我们使用 *MQSNOAUT* 变量来禁用队列管理器上的授权，这使我们能够关注使用 TLS 连接客户机所需的步骤。 在 IBM MQ 的生产部署中建议不要这样做，因为这会导致任何连接的应用程序都具有完整的管理权限，而没有任何机制来降低个别应用程序的许可权。

复制并更新以下 YAML。

- 请确保指定了正确的许可证。 请参阅 mq.ibm.com/v1beta1。
- 通过将 `false` 更改为 `true` 来接受许可证。
- 如果使用的是 IBM Cloud File Storage，请参阅 注 3: 使用 IBM Cloud File Storage

队列管理器定制资源 YAML:

```
apiVersion: mq.ibm.com/v1beta1
kind: QueueManager
metadata:
  name: miexample
spec:
  license:
    accept: false
    license: L-RJON-C7QG3S
    use: NonProduction
  queueManager:
    name: MIEXAMPLE
    availability:
      type: MultiInstance
    mqsc:
    - configMap:
        name: example-mi-configmap
        items:
        - tls.mqsc
  template:
    pod:
      containers:
        - env:
            - name: MQSNOAUT
              value: 'yes'
          name: qmgr
  version: 9.2.5.0-r3
  web:
    enabled: true
  pki:
    keys:
      - name: example
        secret:
          secretName: example-mi-secret
          items:
```

```
          - tls.key
          - tls.crt
```

确保您位于先前创建的名称空间中，使用命令行或使用 IBM Cloud Pak for Integration Platform Navigator 在 OCP UI 中部署更新后的 YAML。

验证

经过短暂延迟后，应配置多实例队列管理器并可供使用。 在本部分中，我们将验证队列管理器的行为是否与预期相同。

确认队列管理器正在运行

现在正在部署队列管理器。 请先确认其处于 `Running` 状态，然后再继续。 例如：

```
oc get qmgr miexample
```

测试与队列管理器的连接

要确认为单向 TLS 通信配置了队列管理器，请使用 **amqsputc** 和 **amqsgetc** 样本应用程序：

查找队列管理器主机名

要查找路由 `miexample-ibm-mq-qm` 的队列管理器主机名，请运行以下命令。 将在 `HOST` 字段中返回主机名。

```
oc get routes miexample-ibm-mq-qm
```

指定队列管理器详细信息

创建用于指定队列管理器详细信息的文件 `CCDT.JSON` 。 将主机值替换为上一步返回的主机名。

```
{
    "channel":
    [
        {
            "name": "MIQMCHL",
            "clientConnection":
            {
                "connection":
                [
                    {
                        "host": "<host from previous step>",
                        "port": 443
                    }
                ],
                "queueManager": "MIEXAMPLE"
            },
            "transmissionSecurity":
            {
              "cipherSpecification": "ECDHE_RSA_AES_128_CBC_SHA256"
            },
            "type": "clientConnection"
        }
   ]
}
```

导出环境变量

以适合您的操作系统的方式导出以下环境变量。 **amqsputc** 和 **amqsgetc** 将读取这些变量。

更新系统上文件的路径：

```
export MQCCDTURL='<full_path_to_file>/CCDT.JSON'
export MQSSLKEYR='<full_path_to_file>/clientkey'
```

将消息放入队列

运行以下命令：

```
amqsputc EXAMPLE.QUEUE MIEXAMPLE
```

如果成功连接到队列管理器，那么将输出以下响应：

```
target queue is EXAMPLE.QUEUE
```

通过输入一些文本，然后每次按 **Enter** 键，将多条消息放入队列。

要完成此操作，请按两次 **Enter** 键。

从队列检索消息

运行以下命令：

```
amqsgetc EXAMPLE.QUEUE MIEXAMPLE
```

您在上一步中添加的消息已被使用，并且已输出。

几秒钟后，命令退出。

强制活动 pod 失败

要验证多实例队列管理器的自动恢复，请模拟 pod 故障：

查看活动和备用 pod

运行以下命令：

```
oc get pods
```

请注意，在 **READY** 字段中，活动 pod 返回值 `1/1`，而备用 pod 返回值 `0/1`。

删除活动 pod

运行以下命令，并指定活动 pod 的全名：

```
oc delete pod miexample-ibm-mq-<value>
```

再次查看 pod 状态

运行以下命令：

```
oc get pods
```

查看备用 pod 日志

运行以下命令并指定备用 pod 的全名：

```
oc logs miexample-ibm-mq-<value>
```

您应该会看到以下消息：

```
IBM MQ queue manager 'MIEXAMPLE' becoming the active instance.
```

再次放入并获取消息

在备用 pod 变为活动 pod (即，在 `READY` 字段值变为 `1/1` 之后) 之后，再次使用以下命令 (如前所述) 将消息放入队列管理器，然后从队列管理器中检索消息：

```
amqsputc EXAMPLE.QUEUE MIEXAMPLE
```

```
amqsgetc EXAMPLE.QUEUE MIEXAMPLE
```

恭喜您成功部署了多实例队列管理器，并显示它可以从 pod 故障中自动恢复。

其他信息

注 1: 创建路由

如果使用的是早于 IBM MQ 9.2.1 的 IBM MQ client 或 Toolkit ，那么需要创建 OCP 路由。

要创建路由，请确保您位于先前创建的名称空间中 (请参阅 第 93 页的『开始之前』) ，然后在 OCP UI 中或使用命令行输入以下 YAML：

```
apiVersion: route.openshift.io/v1
kind: Route
metadata:
  name: example-mi-route
spec:
  host: miqmchl.chl.mq.ibm.com
  to:
    kind: Service
    name: miexample-ibm-mq
  port:
```

```
    targetPort: 1414
  tls:
    termination: passthrough
```

请注意， Red Hat OpenShift Container Platform Router 使用 SNI 将请求路由到 IBM MQ 队列管理器。 如果您更改 包含 MQSC 命令的配置映射中指定的通道名称，那么还必须更改此处的主机字段以及指定队列管理器详细信息的 CCDT.JSON 文件中的主机字段。 有关更多信息，请参阅第 98 页的『配置路由以从 Red Hat OpenShift 集群外部连接到队列管理器』。

注 2: 使用非缺省存储类

此示例期望在 OCP 中配置缺省存储类，因此 队列管理器定制资源 YAML 中不需要存储信息。 如果未将存储类配置为缺省值，或者要使用其他存储类，请在 spec.queueManager.storage 下添加 defaultClass: *<storage_class_name>*。

存储类名必须与 OCP 系统上存在的存储类的名称完全匹配。 即，它必须与命令 oc get storageclass 返回的名称匹配。 它还必须支持 ReadWriteMany。 有关更多信息，请参阅第 10 页的『IBM MQ Operator 的存储注意事项』。

注 3: 正在使用 IBM Cloud File Storage

在某些情况下，例如使用 IBM Cloud File Storage 时，还需要在 队列管理器定制资源 YAML 中指定 **securityGroups** 字段。 例如，通过将以下子字段直接添加到 spec: 下

```
securityContext:
  supplementalGroups: [99]
```

有关更多信息，请参阅第 10 页的『IBM MQ Operator 的存储注意事项』。

OpenShift CD CP4I 配置路由以从 Red Hat OpenShift 集群外部连接到队列管理器

您需要 Red Hat OpenShift 路由以将应用程序从 Red Hat OpenShift 集群外部连接到 IBM MQ 队列管理器。 必须在 IBM MQ 队列管理器和客户机应用程序上启用 TLS， 因为仅当使用 TLS 1.2 或更高版本的协议时，SNI 才在 TLS 协议中可用。 Red Hat OpenShift Container Platform Router 使用 SNI 将请求路由到 IBM MQ 队列管理器。

关于此任务

注意: 本文档适用于 IBM MQ 客户机的 V 9.2.1 Continuous Delivery 和更高版本。 如果客户机正在使用 V 9.2.0 Long Term Support 或更低版本，请参阅 IBM MQ 9.1 文档页面 连接到 Red Hat OpenShift 集群中部署的队列管理器。

V 9.2.1 Red Hat OpenShift 路由 的必需配置取决于客户机应用程序的 服务器名称指示 (SNI) 行为。 IBM MQ 支持两种不同的 SNI 头设置，具体取决于配置和客户机类型。 SNI 头设置为客户机目标的主机名，或者设置为 IBM MQ 通道名称。 有关 IBM MQ 如何将通道名称映射到主机名的信息，请参阅 IBM MQ 如何提供多个证书功能。

V 9.2.1 是将 SNI 头设置为 IBM MQ 通道名称，还是使用 **OutboundSNI** 属性控制主机名。 可能的值为 OutboundSNI=CHANNEL (缺省值) 或 OutboundSNI=HOSTNAME。 有关更多信息，请参阅 客户机配置文件的 SSL 节。 请注意， CHANNEL 和 HOSTNAME 是您使用的精确值; 它们不是您替换为实际通道名称或主机名的变量名称。

V 9.2.1

具有不同 OutboundSNI 设置的客户机行为

如果 **OutboundSNI** 设置为 HOSTNAME，那么只要在连接名称中提供了主机名，以下客户机就会设置主机名 SNI:

- C 客户
- 非受管方式下的 .NET 客户机
- Java/JMS 个客户机

如果 **OutboundSNI** 设置为 HOSTNAME ，并且在连接名称中使用了 IP 地址，那么以下客户机将发送空白 SNI 头:

- C 客户
- 非受管方式下的 .NET 客户机
- Java/JMS 客户机 (无法对主机名执行逆向 DNS 查找)

如果 **OutboundSNI** 设置为 CHANNEL 或未设置，那么将改为使用 IBM MQ 通道名称，并且无论是否使用主机名或 IP 地址连接名称，都将始终发送该名称。

以下客户机类型不支持将 SNI 头设置为 IBM MQ 通道名称，因此始终尝试将 SNI 头设置为主机名，而不考虑 **OutboundSNI** 设置:

- AMQP 客户机
- XR 客户机
- 受管方式下的 .NET 客户机 (在 Long Term Support 的 IBM MQ 9.2.0 Fix Pack 4 之前) 以及在 Continuous Delivery 的 IBM MQ 9.2.3 之前。)

V 9.2.0.4 V 9.2.3

从 IBM MQ 9.2.0 Fix Pack 4 for Long Term Support 和 IBM MQ 9.2.3 for Continuous Delivery 开始，如果 **OutboundSNI** 属性设置为 HOSTNAME，那么 IBM MQ 受管 .NET 客户机已更新为将 SERVERNAME 设置为相应的主机名，这将允许 IBM MQ 受管 .NET 客户机使用 Red Hat OpenShift 路由连接到队列管理器。 请注意，在 IBM MQ 9.2.0 Fix Pack 4 中，仅从 `mqclient.ini` 文件添加并支持 **OutboundSNI** 属性; 不能从 .NET 应用程序设置该属性。

V 9.2.5

如果客户机应用程序通过 IBM MQ Internet Pass-Thru (MQIPT) 连接到 Red Hat OpenShift 集群中部署的队列管理器，那么可以将 MQIPT 配置为使用路由定义中的 SSLClientOutboundSNI 属性将 SNI 设置为主机名。

OutboundSNI，多个证书和 Red Hat OpenShift 路径

IBM MQ 使用 SNI 头来提供多个证书功能。 如果应用程序正在通过 CERTLABL 字段连接到配置为使用其他证书的 IBM MQ 通道，那么该应用程序必须使用 **OutboundSNI** 设置 CHANNEL 进行连接。

如果 Red Hat OpenShift 路由配置需要 HOSTNAME SNI ，那么您无法使用 IBM MQ 的多个证书功能，并且无法在任何 IBM MQ 通道对象上设置 CERTLABL 设置。

If an application with an **OutboundSNI** setting of anything other than CHANNEL connects to a channel with a certificate label configured, the application is rejected with an MQRC_SSL_INITIALIZATION_ERROR, and an AMQ9673 message is printed in the queue manager error logs.

有关 IBM MQ 如何提供多个证书功能的更多信息，请参阅 IBM MQ 如何提供多个证书功能 。

示例

将 SNI 设置为 MQ 通道的客户机应用程序需要为要连接到的每个通道创建新的 Red Hat OpenShift 路由。 您还必须在 Red Hat OpenShift Container Platform 集群中使用唯一通道名称，以允许路由到正确的队列管理器。

重要的是，由于 IBM MQ 将通道名称映射到 SNI 头的方式， MQ 通道名称不会以小写字母结尾。

要确定每个新 Red Hat OpenShift 路由所需的主机名，需要将每个通道名称映射到 SNI 地址。 请参阅 IBM MQ 如何提供多个证书功能 以获取更多信息。

然后，必须通过在集群中应用以下 `yaml` ，为每个通道创建新的 Red Hat OpenShift 路由:

```
apiVersion: route.openshift.io/v1
kind: Route
metadata:
  name: <provide a unique name for the Route>
  namespace: <the namespace of your MQ deployment>
```

```
  spec:
    host: <SNI address mapping for the channel>
    to:
      kind: Service
      name: <the name of the Kubernetes Service for your MQ deployment (for example "<Queue
Manager Name>-ibm-mq")>
    port:
      targetPort: 1414
    tls:
      termination: passthrough
```

配置客户机应用程序连接详细信息

您可以通过运行以下命令来确定要用于客户机连接的主机名:

```
oc get route <Name of hostname based Route (for example "<Queue Manager Name>-ibm-mq-qm")>
-n <namespace of your MQ deployment> -o jsonpath="{.spec.host}"
```

客户机连接的端口应该设置为 Red Hat OpenShift Container Platform 路由器使用的端口-通常为 443。

相关任务

第 104 页的『连接到 Red Hat OpenShift 集群中部署的 IBM MQ Console』
如何连接到已部署到 Red Hat OpenShift Container Platform 集群的队列管理器的 IBM MQ Console 。

CP4I 与 IBM Cloud Pak for Integration 操作仪表板集成

通过 IBM Cloud Pak for Integration 跟踪事务的能力由操作仪表板提供。

关于此任务

启用与 "操作仪表板" 的集成会将 MQ API 出口安装到队列管理器。 API 出口将向 "操作仪表板" 数据存储器发送有关流经队列管理器的消息的跟踪数据。

请注意，仅跟踪使用 MQ 客户机绑定发送的消息。

另请注意，对于 IBM MQ Operator 低于 1.5 的版本，在启用跟踪时，与队列管理器一起部署的跟踪代理程序和收集器映像始终是可用的最新版本，如果您未使用最新版本的 IBM Cloud Pak for Integration，那么这可能会引入不兼容性。

过程

1. 在启用跟踪的情况下部署队列管理器

   缺省情况下，禁用跟踪功能。

   如果要使用 IBM Cloud Pak for Integration Platform Navigator 进行部署，那么可以在部署时启用跟踪，方法是将 **启用跟踪** 设置为 **开启**，并将 **跟踪名称空间** 设置为安装了操作仪表板的名称空间。 有关部署队列管理器的更多信息，请参阅 第 75 页的『使用 IBM Cloud Pak for Integration Platform Navigator 部署队列管理器』

   如果要使用 Red Hat OpenShift CLI 或 Red Hat OpenShift Web 控制台进行部署，那么可以使用以下 YAML 片段来启用跟踪:

   ```
   spec:
     tracing:
       enabled: true
       namespace: <Operations_Dashboard_Namespace
   ```

   重要信息: 在向 "操作仪表板" 注册 MQ 之前，队列管理器将不会启动 (请参阅下一步)。

   请注意，启用此功能后，除队列管理器容器外，它还将运行两个侧柜容器 ("代理程序" 和 "收集器")。 这些侧柜容器的映像将在与主 MQ 映像相同的注册表中可用，并且将使用相同的拉取策略和拉取私钥。 有其他设置可用于配置 CPU 和内存限制。

2. 如果这是首次在此名称空间中部署具有 "操作仪表板" 集成的队列管理器，那么需要向 "操作仪表板" 注册 。

   注册将创建一个密钥对象，队列管理器 Pod 需要该对象才能成功启动。

CD CP4I 在 *IBM Cloud Pak for Integration 2021.4* 中使用 "操作仪表板" 集成来部署或升级 *IBM MQ 9.2.2* 或 *9.2.3*

每个 IBM MQ 版本都与特定版本的操作仪表板代理程序和收集器组件相关联，这些组件与队列管理器一起部署。 IBM Cloud Pak for Integration 2021.4.1 引入了导致较旧的代理程序和收集器组件无法使用 "操作仪表板" 的更改。 要解决此问题，当您使用 IBM MQ 9.2.2 或 9.2.3 时，必须覆盖您使用的 Operations Dashboard 代理程序和收集器映像的版本。

部署新的 IBM MQ 9.2.2 或 9.2.3 队列管理器

将 IBM Cloud Pak for Integration 2021.4.1 与 IBM MQ 9.2.2 或 9.2.3 配合使用时，必须将 Operations Dashboard 代理程序和收集器映像覆盖到 `QueueManager` YAML 中的 2.4 版本。 例如：

```
spec:
  tracing:
    agent:
      image: cp.icr.io/cp/icp4i/od/icp4i-od-
agent@sha256:27a211f0f78eff765d1f9520e0f9841f902600bb556827477b206e209cb44d20
    collector:
      image: cp.icr.io/cp/icp4i/od/icp4i-od-
collector@sha256:dc70b1341b23dc72642ce68809811f9db0e8a0c46bda2508e8eb3d4035e04f4b
```

如果不执行此操作，那么 `QueueManager` Pod 将陷入 `Pending` 状态。 升级到 IBM MQ 9.2.4 时，可以除去这些覆盖。

升级到 IBM Cloud Pak for Integration 2021.4.1

注: 如果要保留 IBM MQ 9.2.2 或 9.2.3 队列管理器，请不要完成步骤 3。

1. 更新 `QueueManager` 以覆盖代理程序和收集器映像，如前所述。
2. 升级 IBM Cloud Pak for Integration 操作程序，包括操作仪表板和 IBM MQ 操作程序，如 第 66 页的『升级 IBM MQ Operator 和队列管理器』中所述。
3. (可选) 要升级到 IBM MQ 9.2.4 或更高版本，请更新 `QueueManager` 以将 `.spec.version` 用于您的 IBM MQ 版本，然后除去代理程序和收集器映像的覆盖。

OpenShift CP4I 使用 Red Hat OpenShift CLI 构建具有定制 MQSC 和 INI 文件的映像

使用 Red Hat OpenShift Container Platform 管道来创建新的 IBM MQ 容器映像，其中包含要应用于使用此映像的队列管理器的 MQSC 和 INI 文件。 此任务应由项目管理员完成

开始之前

您需要安装 Red Hat OpenShift Container Platform 命令行界面。

使用 **`cloudctl login`** (对于 IBM Cloud Pak for Integration) 或 **`oc login`** 登录到集群。

如果您在 Red Hat OpenShift 项目中没有 IBM Entitled Registry 的 Red Hat OpenShift 私钥，请遵循 第 73 页的『为 IBM MQ 准备 Red Hat OpenShift 项目』的步骤。

过程

1. 创建 `ImageStream`

   映像流及其关联标记提供了用于从 Red Hat OpenShift Container Platform 中引用容器映像的抽象。 映像流及其标记允许您查看可用的映像，并确保您正在使用所需的特定映像，即使存储库中的映像发生更改也是如此。

   ```
   oc create imagestream mymq
   ```

2. 为新映像创建 `BuildConfig`

BuildConfig 将允许构建新映像，该映像将基于 IBM 官方映像，但将添加要在容器启动时运行的任何 MQSC 或 INI 文件。

a) 创建用于定义 BuildConfig 资源的 YAML 文件

例如，使用以下内容创建名为 "mq-build-config.yaml" 的文件:

```
apiVersion: build.openshift.io/v1
kind: BuildConfig
metadata:
  name: mymq
spec:
  source:
    dockerfile: |-
      FROM cp.icr.io/cp/ibm-mqadvanced-server-integration:9.2.5.0-r3
      RUN printf "DEFINE QLOCAL(foo) REPLACE\n" > /etc/mqm/my.mqsc \
        && printf "Channels:\n\tMQIBindType=FASTPATH\n" > /etc/mqm/my.ini
      LABEL summary "My custom MQ image"
  strategy:
    type: Docker
    dockerStrategy:
      from:
        kind: "DockerImage"
        name: "cp.icr.io/cp/ibm-mqadvanced-server-integration:9.2.5.0-r3"
      pullSecret:
        name: ibm-entitlement-key
  output:
    to:
      kind: ImageStreamTag
      name: 'mymq:latest-amd64'
```

您将需要替换提及基本 IBM MQ 的两个位置，以指向要使用的版本和修订的正确基本映像 (请参阅 第 18 页的『IBM MQ Operator 的发布历史记录』 以获取详细信息)。 应用修订后，您将需要重复这些步骤以重新构建映像。

此示例基于 IBM 官方映像创建新映像，并将名为 "my.mqsc" 和 "my.ini" 的文件添加到 /etc/mqm 目录中。 在此目录中找到的任何 MQSC 或 INI 文件都将由容器在启动时应用。 INI 文件使用 **crtmqm -ii** 选项进行应用，并与现有 INI 文件合并。 MQSC 文件按字母顺序应用。

MQSC 命令可重复很重要，因为 每次 队列管理器启动时都将运行这些命令。 这通常意味着在任何 DEFINE 命令上添加 REPLACE 参数，并将 IGNSTATE(YES) 参数添加到任何 START 或 STOP 命令。

b) 将 BuildConfig 应用于服务器。

```
oc apply -f mq-build-config.yaml
```

3. 运行构建以创建映像

a) 启动构建

```
oc start-build mymq
```

您应该会看到类似于以下内容的输出:

```
build.build.openshift.io/mymq-1 started
```

b) 检查构建的状态
例如，您可以使用上一步中返回的构建标识来运行以下命令:

```
oc describe build mymq-1
```

4. 使用新映像部署队列管理器

遵循 第 74 页的『将队列管理器部署到 Red Hat OpenShift Container Platform 集群』中描述的步骤，将新的定制映像添加到 YAML 中。

您可以将 YAML 的以下片段添加到常规 QueueManager YAML 中，其中 *my-namespace* 是您正在使用的 Red Hat OpenShift 项目/名称空间， *image* 是您先前创建的映像的名称 (例如， "mymq:latest-amd64"):

```
spec:
  queueManager:
    image: image-registry.openshift-image-registry.svc:5000/my-namespace/my-image
```

相关任务

第 74 页的『将队列管理器部署到 Red Hat OpenShift Container Platform 集群』
使用 QueueManager 定制资源将队列管理器部署到 Red Hat OpenShift Container Platform 集群上。

OpenShift CP4I 向队列管理器资源添加定制注释和标签

将定制注释和标签添加到 QueueManager 元数据。

关于此任务

定制注释和标签将添加到除 PVC 以外的所有资源。 如果定制注释或标签与现有键匹配，那么将使用 IBM MQ Operator 设置的值。

过程

- 添加定制注释。

  要向队列管理器资源 (包括 pod) 添加定制注释，请在 `metadata` 下添加注释。 例如：

```
apiVersion: mq.ibm.com/v1beta1
kind: QueueManager
metadata:
  name: quickstart-cp4i
  annotations:
    annotationKey: "value"
```

- 添加定制标签。

  要向队列管理器资源 (包括 pod) 添加定制标签，请在 `metadata` 下添加标签。 例如：

```
apiVersion: mq.ibm.com/v1beta1
kind: QueueManager
metadata:
  name: quickstart-cp4i
  labels:
    labelKey: "value"
```

OpenShift CP4I 禁用运行时 Webhook 检查

运行时 Webhook 检查可确保存储类适用于您的队列管理器。 禁用它们以提高性能，或者因为它们对您的环境无效。

关于此任务

对队列管理器配置执行运行时 Webhook 检查。 它们检查存储类是否适合所选队列管理器类型。

您可以选择禁用这些检查以减少创建队列管理器所花费的时间，或者因为这些检查对于特定环境无效。

注: 禁用运行时 Webhook 检查后，将允许任何存储类值。 这可能导致队列管理器中断。

IBM MQ Operator 1.2 中引入了对运行时检查的支持。

过程

- 禁用运行时 Webhook 检查。

  在 `metadata` 下添加以下注释。 例如：

```
apiVersion: mq.ibm.com/v1beta1
kind: QueueManager
metadata:
  name: quickstart-cp4i
  annotations:
    "com.ibm.cp4i/disable-webhook-runtime-checks" : "true"
```

OpenShift CP4I 使用 IBM MQ Operator 操作 IBM MQ

关于此任务

过程

- 第 73 页的『为 IBM MQ 准备 Red Hat OpenShift 项目』.
- 第 74 页的『将队列管理器部署到 Red Hat OpenShift Container Platform 集群』.

OpenShift CP4I 连接到 Red Hat OpenShift 集群中部署的 IBM MQ Console

如何连接到已部署到 Red Hat OpenShift Container Platform 集群的队列管理器的 IBM MQ Console 。

关于此任务

可以在 Red Hat OpenShift Web 控制台中的 `QueueManager` 详细信息页面或 IBM Cloud Pak for Integration Platform Navigator 中找到 IBM MQ Console URL。 或者，可以通过运行以下命令从 Red Hat OpenShift CLI 中找到此命令:

```
oc get queuemanager <QueueManager Name> -n <namespace of your MQ deployment> --output
jsonpath='{.status.adminUiUrl}'
```

如果您正在使用 IBM Cloud Pak for Integration 许可证，那么 IBM MQ Web 控制台将配置为使用 IBM Cloud Pak Identity and Access Manager (IAM)。 IAM 组件可能已由集群管理员设置。 但是，如果这是首次在 Red Hat OpenShift 集群上使用 IAM ，那么您将需要检索初始管理密码。 有关更多信息，请参阅 获取初始管理密码。

如果您正在使用 IBM MQ 许可证，那么不会预先配置 MQ Web 控制台，您需要自己进行配置。 有关更多信息，请参阅 配置用户和角色。

相关任务

第 98 页的『配置路由以从 Red Hat OpenShift 集群外部连接到队列管理器』
您需要 Red Hat OpenShift 路由以将应用程序从 Red Hat OpenShift 集群外部连接到 IBM MQ 队列管理器。 必须在 IBM MQ 队列管理器和客户机应用程序上启用 TLS ，因为仅当使用 TLS 1.2 或更高版本的协议时，SNI 才在 TLS 协议中可用。 Red Hat OpenShift Container Platform Router 使用 SNI 将请求路由到 IBM MQ 队列管理器。

OpenShift CP4I 使用 *IBM Cloud Pak IAM* 为 *IBM MQ Console* 授予许可权

IBM MQ Console 的许可权通过 IBM Cloud Pak Administration Hub 而不是 IBM Cloud Pak for Integration Platform Navigator 进行管理。 IBM MQ 不使用 IBM Cloud Pak for Integration 提供的 "自动化" 许可权，而是使用 IBM Cloud Pak Identity and Access Manager (IAM) 启用的基本许可权。

过程

1. 打开 IBM Cloud Pak 管理控制台。

   从 IBM Cloud Pak for Integration Platform UI 中，单击工具栏右上角的 Cloud Pak 切换器 (9 点图标) ，然后单击 **IBM Cloud Pak Administration** 面板。
2. 在左上角的导航菜单中，选择 **身份和访问权**，然后选择 **团队和服务标识**。
3. 创建团队，然后向其添加用户。
   a) 选择 **创建团队**。
   b) 输入团队名称，然后选择要管理的用户的安全域。
   c) 搜索用户。
      这些用户必须已存在于身份提供者中。
   d) 当您找到每个用户时，请为其提供角色。 这必须是 "管理员" 或 "集群管理员" ，才能使用 IBM MQ Console 来管理 IBM MQ 。

4. 将每个用户添加到名称空间。
   a) 选择要编辑的团队。
   b) 选择 **资源** > **管理资源**。
   c) 选择您希望此团队管理的名称空间。 这些名称空间可以是具有队列管理器的任何名称空间。

OpenShift CP4I 使用 IBM MQ Operator 时进行监视

由 IBM MQ Operator 管理的队列管理器可以生成与 Prometheus 兼容的度量。

您可以使用 Red Hat OpenShift Container Platform (OCP) 监视堆栈来查看这些度量值。 打开 OCP 中的 **度量** 选项卡，然后单击 **观察** > **度量**。 缺省情况下已启用队列管理器度量值，但可以通过将 **.spec.metrics.enabled** 设置为 `false` 来禁用这些度量值。

Prometheus 是时间序列数据库和度量值的规则评估引擎。 IBM MQ 容器公开可由 Prometheus 查询的度量值端点。 这些度量是从 MQ 系统主题生成的，用于监视和活动跟踪。

Red Hat OpenShift Container Platform 包含使用 Prometheus 服务器的预配置，预安装和自更新监视堆栈。 需要配置 Red Hat OpenShift Container Platform 监视堆栈以监视用户定义的项目。 有关更多信息，请参阅对用户定义的项目启用监视。 当您创建启用了度量的 `QueueManager` 时， IBM MQ Operator 将创建 `ServiceMonitor`，然后 Prometheus 操作程序可以发现这些度量。

在较低版本的 IBM Cloud Pak for Integration 中，您还可以使用 IBM Cloud Platform Monitoring 服务来提供 Prometheus 服务器。

OpenShift CP4I 使用 *IBM MQ Operator* 时发布的度量

队列管理器容器可以发布与 Red Hat OpenShift Monitoring 兼容的度量。

度量	类型	描述
`ibmmq_qmgr_commit_total`	`counter`	落实计数
`ibmmq_qmgr_cpu_load_fifteen_minute_average_percentage`	`gauge`	CPU 负载 - 15 分钟平均值
`ibmmq_qmgr_cpu_load_five_minute_average_percentage`	`gauge`	CPU 负载 - 5 分钟平均值
`ibmmq_qmgr_cpu_load_one_minute_average_percentage`	`gauge`	CPU 负载 - 1 分钟平均值
`ibmmq_qmgr_destructive_get_bytes_total`	`counter`	时间间隔总破坏性获取 - 字节计数
`ibmmq_qmgr_destructive_get_total`	`counter`	时间间隔总破坏性获取 - 计数
`ibmmq_qmgr_durable_subscription_alter_total`	`counter`	更改持久预订计数
`ibmmq_qmgr_durable_subscription_create_total`	`counter`	创建持久预订计数
`ibmmq_qmgr_durable_subscription_delete_total`	`counter`	删除持久预订计数

度量	类型	描述
ibmmq_qmgr_durable_subscription_resume_total	counter	恢复持久预订计数
ibmmq_qmgr_errors_file_system_free_space_percentage	gauge	MQ 错误文件系统 - 可用空间
ibmmq_qmgr_errors_file_system_in_use_bytes	gauge	MQ 错误文件系统 - 使用的字节数
ibmmq_qmgr_expired_message_total	counter	到期消息计数
ibmmq_qmgr_failed_browse_total	counter	失败的浏览计数
ibmmq_qmgr_failed_mqcb_total	counter	失败的 MQCB 计数
ibmmq_qmgr_failed_mqclose_total	counter	失败的 MQCLOSE 计数
ibmmq_qmgr_failed_mqconn_mqconnx_total	counter	失败的 MQCONN/MQCONNX 计数
ibmmq_qmgr_failed_mqget_total	counter	失败的 MQGET - 计数
ibmmq_qmgr_failed_mqinq_total	counter	失败的 MQINQ 计数
ibmmq_qmgr_failed_mqopen_total	counter	失败的 MQOPEN 计数
ibmmq_qmgr_failed_mqput1_total	counter	失败的 MQPUT1 计数
ibmmq_qmgr_failed_mqput_total	counter	失败的 MQPUT 计数
ibmmq_qmgr_failed_mqset_total	counter	失败的 MQSET 计数
ibmmq_qmgr_failed_mqsubrq_total	counter	失败的 MQSUBRQ 计数
ibmmq_qmgr_failed_subscription_create_alter_resume_total	counter	失败的创建/更改/恢复预订计数
ibmmq_qmgr_failed_subscription_delete_total	counter	预订删除失败计数
ibmmq_qmgr_failed_topic_mqput_mqput1_total	counter	失败的主题 MQPUT/MQPUT1 计数

度量	类型	描述
ibmmq_qmgr_fdc_files	gauge	MQ FDC 文件数
ibmmq_qmgr_log_file_system_in_use_bytes	gauge	日志文件系统 - 使用的字节数
ibmmq_qmgr_log_file_system_max_bytes	gauge	日志文件系统 - 最大字节数
ibmmq_qmgr_log_in_use_bytes	gauge	日志 - 使用的字节数
ibmmq_qmgr_log_logical_written_bytes_total	counter	日志 - 写入的逻辑字节数
ibmmq_qmgr_log_max_bytes	gauge	日志 - 最大字节数
ibmmq_qmgr_log_physical_written_bytes_total	counter	日志 - 写入的物理字节数
ibmmq_qmgr_log_primary_space_in_use_percentage	gauge	日志 - 当前使用的主空间
ibmmq_qmgr_log_workload_primary_space_utilization_percentage	gauge	日志 - 工作负载主空间利用率
ibmmq_qmgr_log_write_latency_seconds	gauge	日志 - 写等待时间
ibmmq_qmgr_log_write_size_bytes	gauge	日志 - 写大小
ibmmq_qmgr_mqcb_total	counter	MQCB 计数
ibmmq_qmgr_mqclose_total	counter	MQCLOSE 计数
ibmmq_qmgr_mqconn_mqconnx_total	counter	MQCONN/MQCONNX 计数
ibmmq_qmgr_mqctl_total	counter	MQCTL 计数
ibmmq_qmgr_mqdisc_total	counter	MQDISC 计数
ibmmq_qmgr_mqinq_total	counter	MQINQ 计数
ibmmq_qmgr_mqopen_total	counter	MQOPEN 计数
ibmmq_qmgr_mqput_mqput1_bytes_total	counter	时间间隔总 MQPUT/MQPUT1 字节计数

度量	类型	描述
ibmmq_qmgr_mqput_mqput1_total	counter	时间间隔总 MQPUT/MQPUT1 计数
ibmmq_qmgr_mqset_total	counter	MQSET 计数
ibmmq_qmgr_mqstat_total	counter	MQSTAT 计数
ibmmq_qmgr_mqsubrq_total	counter	MQSUBRQ 计数
ibmmq_qmgr_non_durable_subscription_create_total	counter	创建非持久预订计数
ibmmq_qmgr_non_durable_subscription_delete_total	counter	删除非持久预订计数
ibmmq_qmgr_non_persistent_message_browse_bytes_total	counter	非持久消息浏览 - 字节计数
ibmmq_qmgr_non_persistent_message_browse_total	counter	非持久消息浏览 - 计数
ibmmq_qmgr_non_persistent_message_destructive_get_total	counter	非持久消息破坏性获取 - 计数
ibmmq_qmgr_non_persistent_message_get_bytes_total	counter	获取非持久消息 - 字节计数
ibmmq_qmgr_non_persistent_message_mqput1_total	counter	非持久消息 MQPUT1 计数
ibmmq_qmgr_non_persistent_message_mqput_total	counter	非持久消息 MQPUT 计数
ibmmq_qmgr_non_persistent_message_put_bytes_total	counter	放入非持久消息 - 字节计数
ibmmq_qmgr_non_persistent_topic_mqput_mqput1_total	counter	非持久 - 主题 MQPUT/MQPUT1 计数
ibmmq_qmgr_persistent_message_browse_bytes_total	counter	持久消息浏览 - 字节计数
ibmmq_qmgr_persistent_message_browse_total	counter	持久消息浏览 - 计数

度量	类型	描述
ibmmq_qmgr_persistent_message_destructive_get_total	counter	持久消息破坏性获取 - 计数
ibmmq_qmgr_persistent_message_get_bytes_total	counter	获取持久消息 - 字节计数
ibmmq_qmgr_persistent_message_mqput1_total	counter	持久消息 MQPUT1 计数
ibmmq_qmgr_persistent_message_mqput_total	counter	持久消息 MQPUT 计数
ibmmq_qmgr_persistent_message_put_bytes_total	counter	放入持久消息 - 字节计数
ibmmq_qmgr_persistent_topic_mqput_mqput1_total	counter	持久 - 主题 MQPUT/MQPUT1 计数
ibmmq_qmgr_published_to_subscribers_bytes_total	counter	发布到订户 - 字节计数
ibmmq_qmgr_published_to_subscribers_message_total	counter	发布到订户 - 消息计数
ibmmq_qmgr_purged_queue_total	counter	队列清除计数
ibmmq_qmgr_queue_manager_file_system_free_space_percentage	gauge	队列管理器文件系统 - 可用空间
ibmmq_qmgr_queue_manager_file_system_in_use_bytes	gauge	队列管理器文件系统 - 使用的字节数
ibmmq_qmgr_ram_free_percentage	gauge	可用 RAM 百分比
ibmmq_qmgr_ram_usage_estimate_for_queue_manager_bytes	gauge	RAM 总字节数 - 队列管理器的估算值
ibmmq_qmgr_rollback_total	counter	回滚计数
ibmmq_qmgr_system_cpu_time_estimate_for_queue_manager_percentage	gauge	系统 CPU 时间 - 队列管理器的百分比估算值
ibmmq_qmgr_system_cpu_time_percentage	gauge	系统 CPU 时间百分比

度量	类型	描述
ibmmq_qmgr_topic_mqput_mqput1_total	counter	主题 MQPUT/MQPUT1 总时间间隔
ibmmq_qmgr_topic_put_bytes_total	counter	时间间隔总放入主题字节数
ibmmq_qmgr_trace_file_system_free_space_percentage	gauge	MQ 跟踪文件系统 - 可用空间
ibmmq_qmgr_trace_file_system_in_use_bytes	gauge	MQ 跟踪文件系统 - 使用的字节数
ibmmq_qmgr_user_cpu_time_estimate_for_queue_manager_percentage	gauge	用户 CPU 时间 - 队列管理器的百分比估算值
ibmmq_qmgr_user_cpu_time_percentage	gauge	用户 CPU 时间百分比

CD CP4I V 9.2.2 查看 IBM MQ 认证容器的本机 HA 队列管理器的状态

对于 IBM MQ 认证的容器，您可以通过在其中一个正在运行的 Pod 中运行 **dspmq** 命令来查看本机 HA 实例的状态。

关于此任务

要点:

您可以在其中一个正在运行的 Pod 中使用 **dspmq** 命令来查看队列管理器实例的操作状态。 返回的信息取决于实例是活动实例还是副本实例。 活动实例提供的信息是明确的，来自副本节点的信息可能已过时。

您可以执行以下操作:

- 查看当前节点上的队列管理器实例是处于活动状态还是处于副本状态。
- 查看当前节点上实例的本机 HA 操作状态。
- 查看本机 HA 配置中所有三个实例的操作状态。

以下状态字段用于报告本机 HA 配置状态:

角色
指定实例的当前角色，该角色是 `Active`， `Replica` 或 `Unknown` 之一。

INSTANCE
使用 **crtmqm** 命令的 **-lr** 选项创建队列管理器时为此实例提供的名称。

INSYNC
指示实例是否能够作为活动实例进行接管 (如果需要)。

QUORUM
以 *number_of_instances_in-sync/number_of_instances_configured* 格式报告定额状态。

REPLADDR
队列管理器实例的复制地址。

连接 ACTV
指示节点是否已连接到活动实例。

BACKLOG
指示实例延迟的 KB 数。

连接
指示指定的实例是否已连接到此实例。

ALTDATE
指示上次更新此信息的日期 (如果从未更新此信息，那么为空白)。

ALTTIME
指示上次更新此信息的时间 (如果从未更新此信息，那么为空白)。

过程

- 查找属于队列管理器的 pod。

```
oc get pod --selector app.kubernetes.io/instance=nativeha-qm
```

- 在其中一个 pod 中运行 dspmq

```
oc exec -t Pod dspmq
```

```
oc rsh Pod
```

用于交互式 shell ，您可以在其中直接运行 dspmq 。

- 要确定队列管理器实例是作为活动实例运行还是作为副本运行:

```
oc exec -t Pod dspmq -o status -m QMgrName
```

名为 BOB 的队列管理器的活动实例将报告以下状态:

```
QMNAME(BOB)                STATUS(Running)
```

名为 BOB 的队列管理器的副本实例将报告以下状态:

```
QMNAME(BOB)                STATUS(Replica)
```

不活动的实例将报告以下状态:

```
QMNAME(BOB)                STATUS(Ended Immediately)
```

- 要确定指定 pod 中实例的本机 HA 操作状态，请执行以下操作:

```
oc exec -t Pod dspmq -o nativeha -m QMgrName
```

名为 BOB 的队列管理器的活动实例可能会报告以下状态:

```
QMNAME(BOB)              ROLE(Active) INSTANCE(inst1) INSYNC(Yes) QUORUM(3/3)
```

名为 BOB 的队列管理器的副本实例可能会报告以下状态:

```
QMNAME(BOB)              ROLE(Replica) INSTANCE(inst2) INSYNC(Yes) QUORUM(2/3)
```

名为 BOB 的队列管理器的不活动实例可能会报告以下状态:

```
QMNAME(BOB)              ROLE(Unknown) INSTANCE(inst3) INSYNC(no) QUORUM(0/3)
```

- 要确定本机 HA 配置中所有实例的本机 HA 操作状态:

```
oc exec -t Pod dspmq -o nativeha -x -m QMgrName
```

如果在运行队列管理器 BOB 的活动实例的节点上发出此命令，那么可能会收到以下状态:

```
QMNAME(BOB)            ROLE(Active) INSTANCE(inst1) INSYNC(Yes) QUORUM(3/3)
 INSTANCE(inst1) ROLE(Active)  REPLADDR(9.20.123.45) CONNACTV(Yes) INSYNC(Yes) BACKLOG(0)
CONNINST(Yes) ALTDATE(2022-01-12) ALTTIME(12.03.44)
 INSTANCE(inst2) ROLE(Replica) REPLADDR(9.20.123.46) CONNACTV(Yes) INSYNC(Yes) BACKLOG(0)
CONNINST(Yes) ALTDATE(2022-01-12) ALTTIME(12.03.44)
```

```
 INSTANCE(inst3) ROLE(Replica) REPLADDR(9.20.123.47) CONNACTV(Yes) INSYNC(Yes) BACKLOG(0)
CONNINST(Yes) ALTDATE(2022-01-12) ALTTIME(12.03.44)
```

如果在运行队列管理器 BOB 的副本实例的节点上发出此命令，那么可能会收到以下状态，这指示其中一个副本落后:

```
QMNAME(BOB)            ROLE(Replica) INSTANCE(inst2) INSYNC(Yes) QUORUM(2/3)
 INSTANCE(inst2) ROLE(Replica) REPLADDR(9.20.123.46) CONNACTV(Yes) INSYNC(Yes) BACKLOG(0)
CONNINST(Yes) ALTDATE(2022-01-12) ALTTIME(12.03.44)
 INSTANCE(inst1) ROLE(Active)  REPLADDR(9.20.123.45) CONNACTV(Yes) INSYNC(Yes) BACKLOG(0)
CONNINST(Yes) ALTDATE(2022-01-12) ALTTIME(12.03.44)
 INSTANCE(inst3) ROLE(Replica) REPLADDR(9.20.123.47) CONNACTV(Yes) INSYNC(No)  BACKLOG(435)
CONNINST(Yes) ALTDATE(2022-01-12) ALTTIME(12.03.44)
```

如果在运行队列管理器 BOB 的不活动实例的节点上发出此命令，那么可能会收到以下状态:

```
QMNAME(BOB)            ROLE(Unknown) INSTANCE(inst3) INSYNC(no) QUORUM(0/3)
 INSTANCE(inst1) ROLE(Unknown) REPLADDR(9.20.123.45) CONNACTV(Unknown) INSYNC(Unknown)
BACKLOG(Unknown) CONNINST(No) ALTDATE() ALTTIME()
 INSTANCE(inst2) ROLE(Unknown) REPLADDR(9.20.123.46) CONNACTV(Unknown) INSYNC(Unknown)
BACKLOG(Unknown) CONNINST(No) ALTDATE() ALTTIME()
 INSTANCE(inst3) ROLE(Unknown) REPLADDR(9.20.123.47) CONNACTV(No) INSYNC(Unknown)
BACKLOG(Unknown) CONNINST(No) ALTDATE() ALTTIME()
```

如果在实例仍在协商哪些是活动的副本时发出该命令，那么您将收到以下状态:

```
QMNAME(BOB)              STATUS(Negotiating)
```

相关参考

dspmq (显示队列管理器) 命令

第 85 页的『示例: 配置本机 HA 队列管理器』

此示例显示如何使用本机高可用性功能将队列管理器部署到使用 IBM MQ Operator 的 Red Hat OpenShift Container Platform (OCP) 中。

OpenShift CP4I 使用 Red Hat OpenShift CLI 备份和复原队列管理器配置

如果队列管理器配置丢失，那么备份队列管理器配置可帮助您根据其定义重建队列管理器。 此过程不会备份队列管理器日志数据。 由于消息的瞬态性质，在复原时，历史日志数据很可能不相关。

开始之前

使用 **cloudctl login** (对于 IBM Cloud Pak for Integration) 或 **oc login** 登录到集群。

过程

- 备份队列管理器配置。

  您可以使用 **dmpmqcfg** 命令来转储 IBM MQ 队列管理器的配置。

  a) 获取队列管理器的 pod 的名称。
  例如，可以运行以下命令，其中 *queue_manager_name* 是 `QueueManager` 资源的名称:

  ```
  oc get pods --selector app.kubernetes.io/name=ibm-mq,app.kubernetes.io/
  instance=queue_manager_name
  ```

  b) 在 pod 上运行 **dmpmqcfg** 命令，将输出定向到本地机器上的文件中。

  dmpmqcfg 输出队列管理器的 MQSC 配置。

  ```
  oc exec -it pod_name -- dmpmqcfg > backup.mqsc
  ```

- 复原队列管理器配置。

  遵循上一步中概述的备份过程后，您应该有一个包含队列管理器配置的 `backup.mqsc` 文件。 您可以通过将此文件应用于新的队列管理器来复原配置。

a) 获取队列管理器的 pod 的名称。
例如，可以运行以下命令，其中 *queue_manager_name* 是 `QueueManager` 资源的名称:

```
oc get pods --selector app.kubernetes.io/name=ibm-mq,app.kubernetes.io/
instance=queue_manager_name
```

b) 在 pod 上运行 **runmqsc** 命令，指示 `backup.mqsc` 文件的内容。

```
oc exec -i pod_name -- runmqsc < backup.mqsc
```

OpenShift CP4I 对 IBM MQ Operator 的问题进行故障诊断

如果您在使用 IBM MQ Operator 时遇到问题，请使用所述方法来帮助您诊断和解决问题。

过程

- 第 113 页的『故障诊断: 获取对队列管理器数据的访问权』

OpenShift CP4I 故障诊断: 获取对队列管理器数据的访问权

使用 PVC 检验员工具来获取对队列管理器 PVC 上的文件的访问权，在这些文件中无法对队列管理器 pod 建立远程 shell。 这可能是因为 pod 处于 **Error** 或 **CrashLoopBackOff** 状态。 此工具旨在与 IBM MQ Operator 部署的队列管理器配合使用。

开始之前

使用 PVC Inspector 工具。 您必须有权访问队列管理器名称空间。

关于此任务

为了帮助进行故障诊断，您可以访问与给定队列管理器关联的持久卷声明 (PVC) 上存储的数据。 要执行此操作，请使用工具将 PVC 安装到一组检验器 pod。 然后，可以将远程 shell 放入任何检验器 pod 中以读取文件。

根据部署类型，将创建 1 到 3 个 Inspector pod。 特定于本机 HA 或多实例队列管理器的给定 pod 的卷在关联的 PVC 检验器 pod 上可用。 共享卷在所有检验员上都可用。 检验员 pod 的名称包含关联队列管理器 pod 的名称。

过程

1. 下载 MQ PVC Inspector 工具。

   该工具在以下位置提供: https://github.com/ibm-messaging/mq-pvc-tool。
2. 确保您已登录到集群。
3. 找出队列管理器的名称以及队列管理器正在其中运行的名称空间。
4. 对队列管理器运行检验员工具。

   a) 运行以下命令，指定队列管理器名称及其名称空间名称。

   ```
   ./pvc-tool.sh queue_manager_name queue_manager_namespace_name
   ```

   b) 在工具完成后，运行以下命令以查看要创建的检验器 pod。

   ```
   oc get pods
   ```
5. 查看安装到检验员 pod 的文件。

   a) 每个 PVC 检验器 pod 都与一个队列管理器 pod 相关联，因此可能有多个检验器 pod。 通过运行以下命令，访问其中一个 pod:

   ```
   oc rsh pvc-inspector-pod-name
   ```

您将放置在包含已安装 PVC 目录的目录中。

b) 通过运行以下命令，将远程 shell 打开到 pod 中:

```
ls
```

c) 您可以看到与已安装的 PVC 同名的目录。 通过浏览这些目录来访问队列管理器 PVC 上的文件。 要查看 PVC 列表，请在远程 shell 会话外部运行以下命令:

```
oc get pvc
```

d) 通过运行以下命令来清除该工具创建的 Pod:

```
'oc delete pods -l tool=mq-pvc-inspector
```

OpenShift CP4I IBM MQ Operator 的 API 参考

IBM MQ 提供 Kubernetes 操作程序，该操作程序提供与 Red Hat OpenShift Container Platform 的本机集成。

OpenShift CP4I mq.ibm.com/v1beta1 的 API 参考

v1beta1 API 可用于创建和管理 QueueManager 资源。

OpenShift CD CP4I EUS *mq.ibm.com/v1beta1* 的许可参考

当前许可证版本

`spec.license.license` 字段必须包含要接受的许可证的许可证标识。 有效值如下所示:

spec.license.license 的值	**spec.license.use 的值**	**许可证信息**	**适用的 IBM MQ 版本**
L-RJON-C7QG3S	Production 或 NonProduction	IBM Cloud Pak for Integration 2021.4.1	9.2.4 或 9.2.5
L-RJON-C7QFZX	Production 或 NonProduction	IBM Cloud Pak for Integration Limited Edition 2021.4.1	9.2.4 或 9.2.5
L-RJON-C5CSNH	Production 或 NonProduction	IBM Cloud Pak for Integration 2021.3.1	9.2.3 或 9.2.4
L-RJON-C5CSM2	Production 或 NonProduction	IBM Cloud Pak for Integration Limited Edition 2021.3.1	9.2.3 或 9.2.4
L-RJON-BZFQU2	Production 或 NonProduction	IBM Cloud Pak for Integration 2021.2.1	9.2.3
L-RJON-BZFQSB	Production 或 NonProduction	IBM Cloud Pak for Integration Limited Edition 2021.2.1	9.2.3
L-RJON-BUVMQX	Production 或 NonProduction	IBM Cloud Pak for Integration 2020.4.1	9.2.0 EUS 或 9.2.1
L-RJON-BUVMYB	Production 或 NonProduction	IBM Cloud Pak for Integration Limited Edition 2020.4.1	9.2.0 EUS 或 9.2.1
L-APIG-BZDDDY	Production	IBM MQ Advanced 和 IBM MQ Advanced for Non-Production Environment 9.2 -07/2021	9.2.3， 9.2.4 或 9.2.5
L-APIG-BYHCL7	Development	IBM MQ Advanced for Developers (非 Warranted) V9.2 -07/2021	9.2.3， 9.2.4 或 9.2.5

spec.license.license 的值	spec.license.use 的值	许可证信息	适用的 IBM MQ 版本
L-APIG-BVJJB3	Production	IBM MQ Advanced 和 IBM MQ Advanced (对于非生产环境) 9.2 -03/2021	9.2.2
L-APIG-BMJJBM	Production	IBM MQ Advanced V9.2	9.2.0 CD 或 9.2.1
L-APIG-BMKG5H	Development	IBM MQ Advanced for Developers (非保修) V9.2	9.2.0 CD， 9.2.1 或 9.2.2

请注意，已指定许可证 版本，这并非始终与 IBM MQ 的版本相同。

较旧的许可证版本

spec.license.license 字段必须包含要接受的许可证的许可证标识。 有效值如下所示：

spec.license.license 的值	spec.license.use 的值	许可证信息	适用的 IBM MQ 版本
L-RJON-BXUPZ2	Production 或 NonProduction	IBM Cloud Pak for Integration 2021.1.1	9.2.2
L-RJON-BXUQ34	Production 或 NonProduction	IBM Cloud Pak for Integration Limited Edition 2021.1.1	9.2.2
L-RJON-BYRMYW	NonProduction	IBM Cloud Pak for Integration Eval-Demo 2021.1.1。 仅用于 IBM MQ Operator 1.5 的 本机 HA 的早期发行版。	9.2.2
L-RJON-BQPGWD	Production 或 NonProduction	IBM Cloud Pak for Integration 2020.3.1	9.2.0 CD
L-RJON-BN7PN3	Production 或 NonProduction	IBM Cloud Pak for Integration 2020.2.1	9.1.5 或 9.2.0 CD
L-RJON-BPHL2Y	Production 或 NonProduction	IBM Cloud Pak for Integration Limited Edition 2020.2.1	9.1.5
L-APIG-BJAKBF	Production	IBM MQ Advanced V9.1 -04/2020	9.1.5
L-APIG-BM7GDH	Development	IBM MQ Advanced for Developers (未登录) V9.1 -04/2020	9.1.5

请注意，已指定许可证 版本，这并非始终与 IBM MQ 的版本相同。

OpenShift CP4I *QueueManager (mq.ibm.com/v1beta1) 的 API 参考*

QueueManager

QueueManager 是 IBM MQ 服务器，用于向应用程序提供排队和发布/预订服务。

字段	描述
apiVersion 字符串	APIVersion 定义对象的此表示的版本化模式。 服务器应将识别的模式转换为最新的内部值，并可能拒绝无法识别的值。 更多信息: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#resources。

字段	描述
kind 字符串	种类是表示此对象所表示的 REST 资源的字符串值。 服务器可以从客户机向其提交请求的端点推断这一点。 Cannot be updated. In CamelCase. 更多信息: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-趋。
metadata	
spec QueueManager 规范	QueueManager 的期望状态。
status QueueManager 状态	QueueManager 的观察状态。

.spec

QueueManager 的期望状态。

显示在:

- 第 115 页的『QueueManager』

字段	描述
affinity	标准 Kubernetes 亲缘关系规则。 有关更多信息，请参阅 https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.17/#affinity-v1-core。
annotations 注释	注释字段充当 Pod 注释的传递。 用户可以向此字段添加任何注释，并将其应用于 Pod。 此处的注释将覆盖缺省注释 (如果提供)。 需要 MQ Operator 1.3.0 或更高版本。
imagePullSecrets LocalObject 参考 数组	对同一名称空间中私钥的引用的可选列表，用于拉取此 QueueManager 所使用的任何映像。 如果指定了这些私钥，那么这些私钥将传递到各个拉取器实现以供其使用。 例如，对于 docker ，仅接受 DockerConfig 类型的私钥。 有关更多信息，请参阅 https://kubernetes.io/docs/concepts/containers/images#specifying-imagepullsecret-on-a-pod。
labels 标签	“标签”字段充当 Pod 标签的传递方式。 用户可以向此字段添加任何标签，并将其应用于 Pod。 此处的标签将覆盖缺省标签 (如果提供)。 需要 MQ Operator 1.3.0 或更高版本。
license 许可证	用于控制您接受许可证以及要使用的许可证度量的设置。
pki PKI	公用密钥基础结构设置，用于定义用于传输层安全性 (TLS) 或 MQ Advanced Message Security (AMS) 的密钥和证书。
queueManager QueueManager 配置	队列管理器容器和底层队列管理器的设置。
securityContext SecurityContext	要添加到队列管理器 Pod 的 securityContext 的安全设置。
template 模板	Kubernetes 资源的高级模板。 该模板允许用户覆盖 IBM MQ 如何生成底层 Kubernetes 资源，例如 StatefulSet， Pod 和服务。 这仅适用于高级用户，因为如果使用不正确，可能会中断 MQ 的正常操作。 在 QueueManager 资源中的任何其他位置指定的任何值都将被模板中的设置覆盖。
terminationGracePeriodSeconds 整数	Pod 需要正常终止的可选持续时间 (以秒计)。 值必须是非负整数。 值 0 指示立即删除。 尝试结束队列管理器的目标时间，将应用程序断开连接的阶段升级。 必要时，将中断基本队列管理器维护任务。 缺省为 30 秒。
tracing TracingConfig	用于跟踪与 Cloud Pak for Integration 操作仪表板的集成的设置。
version 字符串	用于控制将使用 (必需) 的 MQ 版本的设置。 例如: `9.1.5.0-r2` 将使用容器映像的第二个修订版指定 MQ V 9.1.5.0。 特定于容器的修订通常在修订中应用，例如基本映像的修订。

字段	描述
web WebServer 配置	MQ Web 服务器的设置。

.spec.annotations

注释字段充当 Pod 注释的传递。 用户可以向此字段添加任何注释，并将其应用于 Pod。 此处的注释将覆盖缺省注释 (如果提供)。 需要 MQ Operator 1.3.0 或更高版本。

显示在:

- 第 116 页的『.spec』

.spec.imagePullSecrets

LocalObjectReference 包含足够的信息，使您能够在同一名称空间中找到引用的对象。

显示在:

- 第 116 页的『.spec』

字段	描述
name 字符串	引用的名称。 更多信息: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names TODO: 添加其他有用的字段。 apiVersion， kind， uid?。

.spec.labels

“标签”字段充当 Pod 标签的传递方式。 用户可以向此字段添加任何标签，并将其应用于 Pod。 此处的标签将覆盖缺省标签 (如果提供)。 需要 MQ Operator 1.3.0 或更高版本。

显示在:

- 第 116 页的『.spec』

.spec.license

用于控制您接受许可证以及要使用的许可证度量的设置。

显示在:

- 第 116 页的『.spec』

字段	描述
accept 布尔值	是否接受与此软件关联的许可证 (必需)。
license 字符串	您正在接受的许可证的标识。 这必须是您正在使用的 MQ 版本的正确许可证标识。 请参阅 http://ibm.biz/BdqvCF 以获取有效值。
metric 字符串	用于指定要使用的许可证度量的设置。 例如， ProcessorValueUnit， VirtualProcessorCore 或 ManagedVirtualServer。 使用 MQ 许可证时缺省为 ProcessorValueUnit， 使用 Cloud Pak for Integration 许可证时缺省为 VirtualProcessorCore 。
use 字符串	用于控制软件使用方式的设置，其中许可证支持多次使用。 请参阅 http://ibm.biz/BdqvCF 以获取有效值。

.spec.pki

公用密钥基础结构设置，用于定义用于传输层安全性 (TLS) 或 MQ Advanced Message Security (AMS) 的密钥和证书。

显示在:

- 第 116 页的『.spec』

字段	描述
keys PKISource 数组	要添加到队列管理器密钥存储库的专用密钥。
trust PKISource 数组	要添加到队列管理器密钥存储库的证书。

.spec.pki.keys

PKISSource 定义公用密钥基础结构信息 (例如密钥或证书) 的源。

显示在:

- 第 117 页的『.spec.pki』

字段	描述
name 字符串	名称用作密钥或证书的标签。 必须是小写字母数字字符串。
secret 私钥	使用 Kubernetes 密钥提供密钥。

.spec.pki.keys.secret

使用 Kubernetes 密钥提供密钥。

显示在:

- 第 118 页的『.spec.pki.keys』

字段	描述
items 阵列	应该添加到队列管理器容器的 Kubernetes 私钥内的密钥。
secretName 字符串	Kubernetes 私钥的名称。

.spec.pki.trust

PKISSource 定义公用密钥基础结构信息 (例如密钥或证书) 的源。

显示在:

- 第 117 页的『.spec.pki』

字段	描述
name 字符串	名称用作密钥或证书的标签。 必须是小写字母数字字符串。
secret 私钥	使用 Kubernetes 密钥提供密钥。

.spec.pki.trust.secret

使用 Kubernetes 密钥提供密钥。

显示在:

- 第 118 页的『.spec.pki.trust』

字段	描述
items 阵列	应该添加到队列管理器容器的 Kubernetes 私钥内的密钥。
secretName 字符串	Kubernetes 私钥的名称。

.spec.queueManager

队列管理器容器和底层队列管理器的设置。

显示在:

- 第 116 页的『.spec』

字段	描述
`availability` 可用性	队列管理器的可用性设置，例如是否使用活动/备用对或本机高可用性。
`debug` 布尔值	是否将调试消息从特定于容器的代码记录到容器日志。 缺省值为 false。
`image` 字符串	将使用的容器映像。
`imagePullPolicy` 字符串	用于控制 kubelet 何时尝试拉取指定映像的设置。 缺省值为 `IfNotPresent`。
`ini` INISource 数组	用于为队列管理器提供 INI 的设置。 需要 MQ Operator 1.1.0 或更高版本。
`livenessProbe` QueueManagerLivenessProbe	用于控制活动性探测器的设置。
`logFormat` 字符串	要用于此容器的日志格式。 将 `JSON` 用于来自容器的 JSON 格式的日志。 将 `Basic` 用于文本格式的消息。 缺省值为 `Basic`。
`metrics` QueueManager 度量	Prometheus 样式度量的设置。
`mqsc` MQSCSource 数组	用于为队列管理器提供 MQSC 的设置。 需要 MQ Operator 1.1.0 或更高版本。
`name` 字符串	底层 MQ 队列管理器的名称 (如果与 metadata.name 不同)。 如果要使用不符合 Kubernetes 规则的队列管理器名称 (例如，包含字母的名称)，请使用此字段。
`readinessProbe` QueueManagerReadinessProbe	用于控制就绪性探测器的设置。
`resources` 资源	用于控制资源需求的设置。
`route` 路由	队列管理器路由的设置。 需要 MQ 操作程序 1.4.0 或更高版本。
`startupProbe` StartupProbe	用于控制启动探测器的设置。 仅适用于 MultiInstance 和 NativeHA 部署。 需要 MQ Operator 1.5.0 或更高版本。
`storage` QueueManager 存储器	用于控制队列管理器对持久卷和存储类的使用的存储设置。

.spec.queueManager.availability

队列管理器的可用性设置，例如是否使用活动/备用对或本机高可用性。

显示在:

- 第 119 页的『.spec.queueManager』

字段	描述
`tls` Tls	用于配置 NativeHA 副本之间的安全通信的可选 TLS 设置。 需要 MQ Operator 1.5.0 或更高版本。
`type` 字符串	要使用的可用性类型。 将 `SingleInstance` 用于单个 Pod，这将由 Kubernetes 自动重新启动 (在某些情况下)。 将 `MultiInstance` 用于一对 Pod，其中一个是 `active` 队列管理器，另一个是备用 Pod。 将 `NativeHA` 用于本机高可用性复制 (需要 MQ 操作程序 1.5.0 或更高版本)。 缺省值为 `SingleInstance`。 有关更多详细信息，请参阅 http://ibm.biz/BdqAQa 。

字段	描述
updateStrategy 字符串	要用于 MultiInstance 和 NativeHA 队列管理器的更新策略。 使用 RollingUpdate 可在队列管理器配置更改时启用自动滚动更新。 使用 OnDelete 来禁用自动滚动更新，仅当删除 Pod (包括由外部因素触发的 Pod 删除) 时，才会应用队列管理器更改。 缺省值为 RollingUpdate。 需要 MQ 操作程序 1.6.0 或更高版本。

.spec.queueManager.availability.tls

用于配置 NativeHA 副本之间的安全通信的可选 TLS 设置。 需要 MQ Operator 1.5.0 或更高版本。

显示在:

- 第 119 页的『.spec.queueManager.availability』

字段	描述
cipherSpec 字符串	NativeHA TLS 的 CipherSpec 的名称。
secretName 字符串	Kubernetes 私钥的名称。

.spec.queueManager.ini

INI 配置文件的源。

显示在:

- 第 119 页的『.spec.queueManager』

字段	描述
configMap ConfigMapINISource	ConfigMap 表示包含 INI 信息的 Kubernetes ConfigMap 。
secret SecretINISource	私钥表示包含 INI 信息的 Kubernetes 私钥。

.spec.queueManager.ini.configMap

ConfigMap 表示包含 INI 信息的 Kubernetes ConfigMap 。

显示在:

- 第 120 页的『.spec.queueManager.ini』

字段	描述
items 阵列	应该应用的 Kubernetes 源中的密钥。
name 字符串	Kubernetes 源的名称。

.spec.queueManager.ini.secret

私钥表示包含 INI 信息的 Kubernetes 私钥。

显示在:

- 第 120 页的『.spec.queueManager.ini』

字段	描述
items 阵列	应该应用的 Kubernetes 源中的密钥。
name 字符串	Kubernetes 源的名称。

.spec.queueManager.livenessProbe

用于控制活动性探测器的设置。

显示在:

- 第 119 页的『.spec.queueManager』

字段	描述
`failureThreshold` 整数	探测器在成功后被视为失败的最小连续失败次数。 缺省值为 1。
`initialDelaySeconds` 整数	在启动探测器之前容器已启动的秒数。 对于 SingleInstance，缺省为 90 秒。 对于 MultiInstance 和 NativeHA 部署，缺省为 0 秒。 更多信息: https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle#container-探针。
`periodSeconds` 整数	执行探测的频率（以秒计）。 缺省为 10 秒。
`successThreshold` 整数	探测器在失败后被视为成功的最小连续成功数。 缺省值为 1。
`timeoutSeconds` 整数	秒数，经过此秒数之后探测就会超时。 缺省为 5 秒。 更多信息: https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle#container-探针。

.spec.queueManager.metrics

Prometheus 样式度量的设置。

显示在:

- 第 119 页的『.spec.queueManager』

字段	描述
`enabled` 布尔值	是否为兼容 Prometheus 的度量值启用端点。 缺省值为 true。

.spec.queueManager.mqsc

MQSC 配置文件的源。

显示在:

- 第 119 页的『.spec.queueManager』

字段	描述
`configMap` ConfigMapMQSCSource	ConfigMap 表示包含 MQSC 信息的 Kubernetes ConfigMap 。
`secret` SecretMQSCSource	私钥表示包含 MQSC 信息的 Kubernetes 私钥。

.spec.queueManager.mqsc.configMap

ConfigMap 表示包含 MQSC 信息的 Kubernetes ConfigMap 。

显示在:

- 第 121 页的『.spec.queueManager.mqsc』

字段	描述
`items` 阵列	应该应用的 Kubernetes 源中的密钥。
`name` 字符串	Kubernetes 源的名称。

.spec.queueManager.mqsc.secret

私钥表示包含 MQSC 信息的 Kubernetes 私钥。

显示在:

- 第 121 页的『.spec.queueManager.mqsc』

字段	描述
`items` 阵列	应该应用的 Kubernetes 源中的密钥。
`name` 字符串	Kubernetes 源的名称。

.spec.queueManager.readinessProbe

用于控制就绪性探测器的设置。

显示在:

- 第 119 页的『.spec.queueManager』

字段	描述
`failureThreshold` 整数	探测器在成功后被视为失败的最小连续失败次数。 缺省值为 1。
`initialDelaySeconds` 整数	在启动探测器之前容器已启动的秒数。 对于 SingleInstance，缺省为 10 秒。 对于 MultiInstance 和 NativeHA 部署，缺省为 0。 更多信息: https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle#container-探针。
`periodSeconds` 整数	执行探测的频率（以秒计）。 缺省为 5 秒。
`successThreshold` 整数	探测器在失败后被视为成功的最小连续成功数。 缺省值为 1。
`timeoutSeconds` 整数	秒数，经过此秒数之后探测就会超时。 缺省为 3 秒。 更多信息: https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle#container-探针。

.spec.queueManager.resources

用于控制资源需求的设置。

显示在:

- 第 119 页的『.spec.queueManager』

字段	描述
`limits` 限制	CPU 和内存设置。
`requests` 请求	CPU 和内存设置。

.spec.queueManager.resources.limits

CPU 和内存设置。

显示在:

- 第 122 页的『.spec.queueManager.resources』

字段	描述
`cpu`	
`memory`	

.spec.queueManager.resources.requests

CPU 和内存设置。

显示在:

- 第 122 页的『.spec.queueManager.resources』

字段	描述
cpu	
memory	

.spec.queueManager.route

队列管理器路由的设置。 需要 MQ 操作程序 1.4.0 或更高版本。

显示在:

- 第 119 页的『.spec.queueManager』

字段	描述
enabled 布尔值	是否启用路由。 缺省值为 true。

.spec.queueManager.startupProbe

用于控制启动探测器的设置。 仅适用于 MultiInstance 和 NativeHA 部署。 需要 MQ Operator 1.5.0 或更高版本。

显示在:

- 第 119 页的『.spec.queueManager』

字段	描述
failureThreshold 整数	要视为失败的探测器的最小连续失败次数。 缺省值为 60。
initialDelaySeconds 整数	在启动探测器之前容器已启动的秒数。 缺省为 0 秒。 更多信息: https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle#container-探针。
periodSeconds 整数	执行探测的频率（以秒计）。 缺省为 5 秒。
successThreshold 整数	要视为成功的探测器的最小连续成功次数。 缺省值为 1。
timeoutSeconds 整数	秒数，经过此秒数之后探测就会超时。 缺省为 5 秒。 更多信息: https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle#container-探针。

.spec.queueManager.storage

用于控制队列管理器对持久卷和存储类的使用的存储设置。

显示在:

- 第 119 页的『.spec.queueManager』

字段	描述
defaultClass 字符串	缺省情况下要应用于此队列管理器的所有持久卷的存储类。 特定持久卷可以定义其自己的存储类，这将覆盖此缺省存储类设置。 如果 type of availability 为 SingleInstance 或 NativeHA，那么存储类的类型可以是 ReadWrite 一次或 ReadWrite 多次。 如果 type of availability 为 MultiInstance，那么存储类的类型必须为 ReadWrite。

字段	描述
defaultDeleteClaim 布尔值	删除队列管理器时是否应删除所有卷。 特定持久卷可以为 deleteClaim 定义自己的值，这将覆盖此 defaultDeleteClaim 设置。 缺省值为 false。
persistedData QueueManagerOptionalVolume	MQ 持久数据的 PersistentVolume 详细信息，包括配置，队列和消息。 使用多实例队列管理器时必需。
queueManager QueueManager 卷	通常在 /var/mqm 下的任何数据的缺省 PersistentVolume 。 将包含所有持久数据和恢复日志 (如果未指定其他卷)。
recoveryLogs QueueManagerOptionalVolume	MQ 恢复日志的持久卷详细信息。 使用多实例队列管理器时必需。

.spec.queueManager.storage.persistedData

MQ 持久数据的 PersistentVolume 详细信息，包括配置，队列和消息。 使用多实例队列管理器时必需。

显示在:

- 第 123 页的『.spec.queueManager.storage』

字段	描述
class 字符串	要用于此卷的存储类。 仅当 type 为 persistent-claim 时才有效。 如果 type of availability 为 SingleInstance 或 NativeHA，那么存储类的类型可以是 ReadWrite 一次或 ReadWrite 多次。 如果 type of availability 为 MultiInstance，那么存储类的类型必须为 ReadWrite。
deleteClaim 布尔值	删除队列管理器时是否应删除此卷。
enabled 布尔值	是应将此卷作为单独卷启用，还是将其放在缺省 queueManager 卷上。 缺省值为 false。
size 字符串	要传递到 Kubernetes 的 PersistentVolume 的大小，包括 SI 单元。 仅当 type 为 persistent-claim 时才有效。 例如，2Gi。 缺省值为 2Gi。
sizeLimit 字符串	使用 ephemeral 卷时的大小限制。 文件仍会写入临时目录，因此您可以使用此选项来限制大小。 仅当 type 为 ephemeral 时才有效。
type 字符串	要使用的卷的类型。 选择 ephemeral 以使用非持久存储器，或选择 persistent-claim 以使用持久卷。 缺省值为 persistent-claim。

.spec.queueManager.storage.queueManager

通常在 /var/mqm 下的任何数据的缺省 PersistentVolume 。 将包含所有持久数据和恢复日志 (如果未指定其他卷)。

显示在:

- 第 123 页的『.spec.queueManager.storage』

字段	描述
class 字符串	要用于此卷的存储类。 仅当 type 为 persistent-claim 时才有效。 如果 type of availability 为 SingleInstance 或 NativeHA，那么存储类的类型可以是 ReadWrite 一次或 ReadWrite 多次。 如果 type of availability 为 MultiInstance，那么存储类的类型必须为 ReadWrite。
deleteClaim 布尔值	删除队列管理器时是否应删除此卷。

字段	描述
size 字符串	要传递到 Kubernetes 的 PersistentVolume 的大小，包括 SI 单元。 仅当 type 为 persistent-claim 时才有效。 例如，2Gi。 缺省值为 2Gi。
sizeLimit 字符串	使用 ephemeral 卷时的大小限制。 文件仍会写入临时目录，因此您可以使用此选项来限制大小。 仅当 type 为 ephemeral 时才有效。
type 字符串	要使用的卷的类型。 选择 ephemeral 以使用非持久存储器，或选择 persistent-claim 以使用持久卷。 缺省值为 persistent-claim。

.spec.queueManager.storage.recoveryLogs

MQ 恢复日志的持久卷详细信息。 使用多实例队列管理器时必需。

显示在:

- 第 123 页的『.spec.queueManager.storage』

字段	描述
class 字符串	要用于此卷的存储类。 仅当 type 为 persistent-claim 时才有效。 如果 type of availability 为 SingleInstance 或 NativeHA，那么存储类的类型可以是 ReadWrite 一次或 ReadWrite 多次。 如果 type of availability 为 MultiInstance，那么存储类的类型必须为 ReadWrite。
deleteClaim 布尔值	删除队列管理器时是否应删除此卷。
enabled 布尔值	是应将此卷作为单独卷启用，还是将其放在缺省 queueManager 卷上。 缺省值为 false。
size 字符串	要传递到 Kubernetes 的 PersistentVolume 的大小，包括 SI 单元。 仅当 type 为 persistent-claim 时才有效。 例如，2Gi。 缺省值为 2Gi。
sizeLimit 字符串	使用 ephemeral 卷时的大小限制。 文件仍会写入临时目录，因此您可以使用此选项来限制大小。 仅当 type 为 ephemeral 时才有效。
type 字符串	要使用的卷的类型。 选择 ephemeral 以使用非持久存储器，或选择 persistent-claim 以使用持久卷。 缺省值为 persistent-claim。

.spec.securityContext

要添加到队列管理器 Pod 的 securityContext 的安全设置。

显示在:

- 第 116 页的『.spec』

字段	描述
fsGroup 整数	适用于 pod 中所有容器的特殊补充组。 某些卷类型允许 Kubelet 更改要由 pod 拥有的该卷的所有权: 1。 拥有的 GID 将是 FSGroup 2。 设置了 setgid 位 (在卷中创建的新文件将由 FSGroup 拥有) 3。 许可权位为 OR ' d with rw-rw ---- 如果未设置，那么 Kubelet 将不会修改任何卷的所有权和许可权。
initVolumeAsRoot 布尔值	这会影响初始化 PersistentVolume 的容器所使用的 securityContext 。 如果您使用的是要求您作为 root 用户访问新供应卷的存储器提供程序，请将此值设置为 true 。 将此项设置为 true 会影响您可以使用的安全上下文约束 (SCC) 对象，如果您无权使用允许 root 用户的 SCC ，那么队列管理器可能无法启动。 缺省值为 false。 有关更多信息，请参阅 https://docs.openshift.com/container-platform/latest/authentication/managing-security-context-constraints.html。
supplementalGroups 阵列	应用于每个容器中运行的第一个进程的组的列表，以及容器的主 GID。 如果未指定，那么不会向任何容器添加任何组。

.spec.template

Kubernetes 资源的高级模板。 该模板允许用户覆盖 IBM MQ 如何生成底层 Kubernetes 资源，例如 StatefulSet， Pod 和服务。 这仅适用于高级用户，因为如果使用不正确，可能会中断 MQ 的正常操作。 在 QueueManager 资源中的任何其他位置指定的任何值都将被模板中的设置覆盖。

显示在:

- 第 116 页的『.spec』

字段	描述
pod	用于 Pod 的模板的覆盖。 请参阅 https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.17/#podspec-v1-core。

.spec.tracing

用于跟踪与 Cloud Pak for Integration 操作仪表板的集成的设置。

显示在:

- 第 116 页的『.spec』

字段	描述
`agent` TracingAgent	仅在 Cloud Pak for Integration 中，可以配置可选跟踪代理程序的设置。
`collector` TracingCollector	仅在 Cloud Pak for Integration 中，您可以配置可选跟踪收集器的设置。
`enabled` 布尔值	是否通过跟踪启用与 Cloud Pak for Integration 操作仪表板的集成。 缺省值为 false。
`namespace` 字符串	安装了 Cloud Pak for Integration 操作仪表板的名称空间。

.spec.tracing.agent

仅在 Cloud Pak for Integration 中，可以配置可选跟踪代理程序的设置。

显示在:

- 第 126 页的『.spec.tracing』

字段	描述
`image` 字符串	将使用的容器映像。
`imagePullPolicy` 字符串	用于控制 kubelet 何时尝试拉取指定映像的设置。 缺省值为 `IfNotPresent`。
`livenessProbe` TracingProbe	用于控制活动性探测器的设置。
`readinessProbe` TracingProbe	用于控制就绪性探测器的设置。

.spec.tracing.agent.livenessProbe

用于控制活动性探测器的设置。

显示在:

- 第 126 页的『.spec.tracing.agent』

字段	描述
`failureThreshold` 整数	探测器在成功后被视为失败的最小连续失败次数。 缺省值为 1。

字段	描述
initialDelaySeconds 整数	在容器启动之后，启动活动性探测器之前的秒数。 缺省为 10 秒。 更多信息: https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle#container-探针。
periodSeconds 整数	执行探测的频率（以秒计）。 缺省为 10 秒。
successThreshold 整数	探测器在失败后被视为成功的最小连续成功数。 缺省值为 1。
timeoutSeconds 整数	秒数，经过此秒数之后探测就会超时。 缺省为 2 秒。 更多信息: https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle#container-探针。

.spec.tracing.agent.readinessProbe

用于控制就绪性探测器的设置。

显示在:

- 第 126 页的『.spec.tracing.agent』

字段	描述
failureThreshold 整数	探测器在成功后被视为失败的最小连续失败次数。 缺省值为 1。
initialDelaySeconds 整数	在容器启动之后，启动活动性探测器之前的秒数。 缺省为 10 秒。 更多信息: https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle#container-探针。
periodSeconds 整数	执行探测的频率（以秒计）。 缺省为 10 秒。
successThreshold 整数	探测器在失败后被视为成功的最小连续成功数。 缺省值为 1。
timeoutSeconds 整数	秒数，经过此秒数之后探测就会超时。 缺省为 2 秒。 更多信息: https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle#container-探针。

.spec.tracing.collector

仅在 Cloud Pak for Integration 中，您可以配置可选跟踪收集器的设置。

显示在:

- 第 126 页的『.spec.tracing』

字段	描述
image 字符串	将使用的容器映像。
imagePullPolicy 字符串	用于控制 kubelet 何时尝试拉取指定映像的设置。 缺省值为 IfNotPresent。
livenessProbe TracingProbe	用于控制活动性探测器的设置。
readinessProbe TracingProbe	用于控制就绪性探测器的设置。

.spec.tracing.collector.livenessProbe

用于控制活动性探测器的设置。

显示在:

- 第 127 页的『.spec.tracing.collector』

字段	描述
`failureThreshold` 整数	探测器在成功后被视为失败的最小连续失败次数。 缺省值为 1。
`initialDelaySeconds` 整数	在容器启动之后，启动活动性探测器之前的秒数。 缺省为 10 秒。 更多信息: https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle#container-探针。
`periodSeconds` 整数	执行探测的频率（以秒计）。 缺省为 10 秒。
`successThreshold` 整数	探测器在失败后被视为成功的最小连续成功数。 缺省值为 1。
`timeoutSeconds` 整数	秒数，经过此秒数之后探测就会超时。 缺省为 2 秒。 更多信息: https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle#container-探针。

.spec.tracing.collector.readinessProbe

用于控制就绪性探测器的设置。

显示在:

- 第 127 页的『.spec.tracing.collector』

字段	描述
`failureThreshold` 整数	探测器在成功后被视为失败的最小连续失败次数。 缺省值为 1。
`initialDelaySeconds` 整数	在容器启动之后，启动活动性探测器之前的秒数。 缺省为 10 秒。 更多信息: https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle#container-探针。
`periodSeconds` 整数	执行探测的频率（以秒计）。 缺省为 10 秒。
`successThreshold` 整数	探测器在失败后被视为成功的最小连续成功数。 缺省值为 1。
`timeoutSeconds` 整数	秒数，经过此秒数之后探测就会超时。 缺省为 2 秒。 更多信息: https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle#container-探针。

.spec.web

MQ Web 服务器的设置。

显示在:

- 第 116 页的『.spec』

字段	描述
`enabled` 布尔值	是否启用 Web 服务器。 缺省值为 false。

.status

QueueManager 的观察状态。

显示在:

- 第 115 页的『QueueManager』

字段	描述
`adminUiUrl` 字符串	管理 UI 的 URL。
`availability` 可用性	队列管理器的可用性状态。

字段	描述
conditions QueueManagerStatusCondition 数组	条件表示队列管理器状态的最新可用观测值。
endpoints QueueManagerStatusEndpoint 数组	有关此队列管理器正在公开的端点 (例如 API 或 UI 端点) 的信息。
name 字符串	队列管理器的名称。
phase 字符串	队列管理器状态的阶段。
versions QueueManagerStatusVersion	正在使用的 MQ 版本以及 IBM 授权注册表中提供的其他版本。

.status.availability

队列管理器的可用性状态。

显示在:

- 第 128 页的『.status』

字段	描述
initialQuorumEstablished 布尔值	是否已为 NativeHA 建立初始定额。

.status.conditions

QueueManagerStatusCondition 定义队列管理器的条件。

显示在:

- 第 128 页的『.status』

字段	描述
lastTransitionTime 字符串	上次将条件从一个状态转换为另一个状态的时间。
message 字符串	指示有关上次转换的详细信息的人类可读消息。
reason 字符串	最近一次转换此状态的原因。
status 字符串	条件的状态。
type 字符串	条件的类型。

.status.endpoints

QueueManagerStatusEndpoint 定义 QueueManager 的端点。

显示在:

- 第 128 页的『.status』

字段	描述
name 字符串	端点的名称。
type 字符串	端点的类型，例如 UI 端点的 "UI"， API 端点的 "API"， API 文档的 "OpenAPI"。

字段	描述
uri 字符串	端点的 URI。

.status.versions

正在使用的 MQ 版本以及 IBM 授权注册表中提供的其他版本。

显示在:

- 第 128 页的『.status』

字段	描述
available QueueManagerStatusVersion 可用	其他版本的 MQ 可从 IBM Entitled Registry 获取。
reconciled 字符串	正在使用的 IBM MQ 的特定版本。 如果指定了定制映像，那么这可能与实际使用的 MQ 版本不匹配。

.status.versions.available

其他版本的 MQ 可从 IBM Entitled Registry 获取。

显示在:

- 第 130 页的『.status.versions』

字段	描述
channels 阵列	可用于自动更新 MQ 版本的通道。
versions 版本 数组	可用的特定 MQ 版本。

.status.versions.available.versions

QueueManagerStatusVersion 定义 MQ 的版本。

显示在:

- 第 130 页的『.status.versions.available』

字段	描述
name 字符串	此版本的 QueueManager 的版本 name 。 这些是 spec.version 字段的有效值。

OpenShift CP4I *QueueManager* 的状态条件 *(mq.ibm.com/v1beta1)*

将更新 **status.conditions** 字段以反映 QueueManager 资源的条件。 通常，条件描述异常情况。 处于正常就绪状态的队列管理器没有 **Error** 或 **Pending** 条件。 它可能具有一些咨询 **Warning** 条件。

IBM MQ Operator 1.2 中引入了对条件的支持。

为 QueueManager 资源定义了以下条件:

表 *1:* 队列管理器状态条件			
组件	**条件类型**	**原因码**	**消息警告**
QueueManager[7]	暂挂	正在创建	正在部署 MQ 队列管理器
	暂挂	OidcPending	MQ 队列管理器正在等待 OIDC 客户机注册
	Error	失败	MQ 队列管理器部署失败
	警告	UnsupportedVersion	[8] 操作数已由 OCP 版本 *<ocp_version>*上不支持的操作程序安装。 不支持此操作数。
	警告	EUSSupport	[9] 已安装 EUS 操作数 *<mq_version>* ，但该操作数正由不符合扩展支持持续时间的操作程序管理。 此操作数不符合扩展支持持续时间的要求。
	警告	EUSSupport	[10] 已安装 EUS 操作数 *<mq_version>* ，但 OCP 版本 4*<ocp_version>* 不符合扩展支持持续时间的条件。 此操作数不符合扩展支持持续时间的要求。
	警告	EUSSupport	[11] 已安装 EUS 操作数 *<mq_version>* ，但 OCP 版本 *<ocp_version>* 不符合扩展支持持续时间的要求。 根据常规 CD 发行版支持此操作数。
展舱 [12]	暂挂	PodPending	正在部署 MQ 队列管理器的 Pod
	Error	PodFailed	正在部署 MQ 队列管理器的 Pod

[7] 条件 `Creating` 和 `Failed` 监视队列管理器部署的整体进度。 如果您正在使用 IBM Cloud Pak for Integration 许可证，并且已启用 MQ Web 控制台，那么 `OidcPending` 条件会在等待 OIDC 客户机注册完成 IAM 时记录队列管理器的状态。

[8] 操作程序 1.4.0 和更高版本

[9] 操作程序 1.4.0 和更高版本

[10] 操作程序 1.4.0 和更高版本

[11] 仅限操作程序 1.3.0

[12] 部署队列管理器期间， pod 条件会监视 pod 的状态。 如果您看到任何 `PodFailed` 条件，那么整体队列管理器条件也将设置为 `Failed`。

表 1: 队列管理器状态条件 (继续)			
组件	**条件类型**	**原因码**	**消息警告**
存储器 [13]	暂挂	StoragePending	正在供应 MQ 队列管理器的存储器
	警告	StorageEphemeral	将临时存储器用于生产 MQ 队列管理器
	Error	StorageFailed	Storage for MQ 队列管理器无法供应

Multi 构建您自己的 IBM MQ 容器和部署代码

开发自建容器。 这是最灵活的容器解决方案，但这需要您具备配置容器的强大技能，并 “拥有” 生成的容器。

开始之前

在开发自己的容器之前，请考虑是否可以改为使用 IBM 提供的其中一个预先打包的容器。 请参阅容器中的 IBM MQ

关于此任务

将 IBM MQ 打包为容器映像时，可以快速轻松地部署对应用程序的更改以测试和登台系统。 这可能是企业持续交付的主要优势。

过程

- 第 132 页的『使用容器规划您自己的 IBM MQ 队列管理器映像』
- 第 133 页的『构建样本 IBM MQ 队列管理器容器映像』
- 第 135 页的『在单独的容器中运行本地绑定应用程序』

相关概念

容器中的 IBM MQ

Multi 使用容器规划您自己的 IBM MQ 队列管理器映像

在容器中运行 IBM MQ 队列管理器时需要考虑多个需求。 样本容器映像提供了处理这些需求的方法，但是如果要使用您自己的映像，需要考虑如何处理这些需求。

过程监管

运行容器时，本质上是运行单个进程 (容器内的 PID 1)，该进程稍后会衍生子进程。

如果主进程结束，那么容器运行时将停止该容器。 IBM MQ 队列管理器需要多个进程在后台运行。

因此，只要队列管理器正在运行，您就需要确保主进程保持活动状态。 最好通过执行管理查询等方法来检查此进程中的队列管理器是否处于活动状态。

填充 /var/mqm

容器必须以 /var/mqm 作为卷进行配置。

[13] 存储条件监视为持久存储创建卷的请求的进度 (StoragePending 条件)，并报告回绑定错误和其他故障。 如果在存储供应期间发生任何错误，那么 StorageFailed 条件将添加到条件列表中，并且整体队列管理器条件也将设置为 Failed。

执行此操作时，当容器首次启动时，卷的目录为空。 通常在安装时填充此目录，但在使用容器时，安装和运行时是不同的环境。

要解决此问题，当容器启动时，可以在首次运行时使用 **crtmqdir** 命令来填充 /var/mqm 。

容器安全性

为了最大限度降低运行时安全性需求，将使用 IBM MQ unzippable 安装来安装样本容器映像。 这将确保未设置任何 setuid 位，并且容器不需要使用特权升级。 某些容器系统定义了您能够使用的用户标识，并且不可压缩的安装不会对可用的操作系统用户进行任何假定。

Multi 构建样本 IBM MQ 队列管理器容器映像

使用此信息来构建用于在容器中运行 IBM MQ 队列管理器的样本容器映像。

关于此任务

首先，构建包含 Red Hat 通用基本映像文件系统和 IBM MQ 的全新安装的基本映像。

其次，在基础上构建另一个容器映像层，这将添加一些 IBM MQ 配置以允许基本用户标识和密码安全性。

最后，使用此映像作为其文件系统运行容器，主机文件系统上特定于容器的卷提供 /var/mqm 的内容。

过程

- 有关如何构建样本容器映像以在容器中运行 IBM MQ 队列管理器的信息，请参阅以下子主题:
  - 第 133 页的『构建样本基本 IBM MQ 队列管理器映像』
  - 第 133 页的『构建样本配置的 IBM MQ 队列管理器映像』

Multi 构建样本基本 IBM MQ 队列管理器映像

为了在您自己的容器映像中使用 IBM MQ ，最初需要使用干净的 IBM MQ 安装来构建基本映像。 以下步骤显示如何使用 GitHub 上托管的样本代码来构建样本基本映像。

过程

- 使用 mq-container GitHub 存储库 中提供的 make 文件来构建生产容器映像。

  遵循 GitHub 上的 构建容器映像 中的指示信息。 如果计划使用 Red Hat OpenShift Container Platform “受限” 安全上下文约束 (SCC) 配置安全访问，那么必须使用 "不安装" IBM MQ 软件包。

结果

现在，您已安装了 IBM MQ 的基本容器映像。

现在，您已准备好 构建样本配置的 IBM MQ 队列管理器映像。

Multi 构建样本配置的 IBM MQ 队列管理器映像

构建通用基本 IBM MQ 容器映像后，需要应用自己的配置以允许安全访问。 为此，您可以使用通用映像作为父映像来创建自己的容器映像层。

开始之前

V 9.2.0 此任务假定当您 构建样本基本 IBM MQ 队列管理器映像时，已使用 “No-Install” IBM MQ 软件包。 否则，无法使用 Red Hat OpenShift Container Platform “受限” 安全上下文约束 (SCC) 来配置安全访问。 缺省情况下使用的 “restricted” SCC 使用随机用户标识，并通过更改为其他用户来阻止特权升级。 基于 IBM MQ 传统 RPM 的安装程序依赖于 mqm 用户和组，并且还在可执行程序上使用 setuid 位。 在 IBM MQ 9.2 中，使用 “No-Install” IBM MQ 软件包时，不再有 mqm 用户，也没有 mqm 组。

过程

1. 创建新目录，并添加名为 `config.mqsc` 的文件，其中包含以下内容:

```
DEFINE QLOCAL(EXAMPLE.QUEUE.1) REPLACE
```

请注意，上述示例使用简单用户标识和密码认证。 但是，您可以应用企业所需的任何安全配置。

2. 创建名为 `Dockerfile` 的文件，其中包含以下内容:

```
FROM mq
COPY config.mqsc /etc/mqm/
```

3. 使用以下命令构建定制容器映像:

```
docker build -t mymq .
```

其中 `"."` 是包含您刚刚创建的两个文件的目录。

然后， Docker 将使用该映像创建临时容器，并运行其余命令。

注: 在 Red Hat Enterprise Linux (RHEL) 上，使用命令 **`docker`** (RHEL V7) 或 **podman** (RHEL V7 或 RHEL V8)。 在 Linux 上，您需要在命令开头运行带有 **sudo** 的 **`docker`** 命令，以获取额外特权。

4. 使用刚刚创建的磁盘映像运行新的定制映像以创建新的容器。

新映像层未指定要运行的任何特定命令，因此已从父映像继承该命令。 父代的入口点 (代码在 GitHub 上可用):

- 创建队列管理器
- 启动该队列管理器
- 创建缺省侦听器
- 然后从 `/etc/mqm/config.mqsc.` 运行任何 MQSC 命令

发出以下命令以运行新的定制映像:

```
docker run \
  --env LICENSE=accept \
  --env MQ_QMGR_NAME=QM1 \
  --volume /var/example:/var/mqm \
  --publish 1414:1414 \
  --detach \
  mymq
```

其中:

前 env 个参数
将环境变量传递到容器中，这将确认您接受 IBM IBM WebSphere MQ 的许可证。 您还可以设置 LICENSE 变量以查看许可证。

请参阅 IBM MQ 许可证信息 ， 以获取有关 IBM MQ 许可证的更多详细信息。

第二个 env 参数
设置您正在使用的队列管理器名称。

卷参数
告知容器，实际上应该将 MQ 写入 `/var/mqm` 的任何内容写入主机上的 `/var/example` 。

此选项意味着您可以在以后轻松删除容器，并且仍然保留任何持久数据。 此选项还使查看日志文件更容易。

发布参数
将主机系统上的端口映射到容器中的端口。 缺省情况下，容器使用其自己的内部 IP 地址运行，这意味着您需要专门映射要公开的任何端口。

在此示例中，这意味着将主机上的端口 1414 映射到容器中的端口 1414。

Detach 参数
在后台运行容器。

结果

您已构建已配置的容器映像，并且可以使用 **docker ps** 命令来查看正在运行的容器。 您可以使用 **docker top** 命令来查看在容器中运行的 IBM MQ 进程。

注意:

您可以使用 **docker logs ${CONTAINER_ID}** 命令来查看容器的日志。

下一步做什么

- 如果在使用 **docker ps** 命令时未显示容器，那么容器可能已失败。 您可以使用 **docker ps -a** 命令来查看失败的容器。
- 使用 **docker ps -a** 命令时，将显示容器标识。 发出 **docker run** 命令时，也打印了此标识。
- 您可以使用 **docker logs ${CONTAINER_ID}** 命令来查看容器的日志。

Multi 在单独的容器中运行本地绑定应用程序

通过在 Docker 中的容器之间共享进程名称空间，您可以在与 IBM MQ 队列管理器不同的容器中运行需要到 IBM MQ 的本地绑定连接的应用程序。

关于此任务

此功能在 IBM MQ 9.0.3 和更高版本的队列管理器中受支持。

您必须遵守以下限制:

- 必须使用 `--pid` 参数共享容器 PID 名称空间。
- 必须使用 `--ipc` 参数共享容器 IPC 名称空间。
- 您必须:
  1. 使用 `--uts` 参数与主机共享容器 UTS 名称空间，或者
  2. 使用 `-h` 或 `--hostname` 参数确保容器具有相同的主机名。
- 必须将 IBM MQ 数据目录安装在可供 `/var/mqm` 目录下的所有容器使用的卷中。

您可以通过在已安装 Docker 的 Linux 系统上完成以下步骤来尝试此功能。

以下示例使用样本 IBM MQ 容器映像。 您可以在 Github 上找到此图像的详细信息。

过程

1. 通过发出以下命令，创建临时目录以充当卷:

```
mkdir /tmp/dockerVolume
```

2. 通过发出以下命令，在名为 `sharedNamespace` 的容器中创建队列管理器 (`QM1`):

```
docker run -d -e LICENSE=accept -e MQ_QMGR_NAME=QM1 --volume /tmp/dockerVol:/mnt/mqm
--uts host --name sharedNamespace ibmcom/mq
```

3. 通过发出以下命令，启动另一个名为 `secondaryContainer`，基于 `ibmcom/mq` 的容器，但不创建队列管理器:

```
docker run --entrypoint /bin/bash --volumes-from sharedNamespace --pid
container:sharedNamespace --ipc container:sharedNamespace --uts host --name
secondaryContainer -it --detach ibmcom/mq
```

4. 通过发出以下命令，在第二个容器上运行 **dspmq** 命令，以查看两个队列管理器的状态:

```
docker exec secondaryContainer dspmq
```

5. 运行以下命令以针对在另一个容器上运行的队列管理器处理 MQSC 命令:

```
docker exec -it secondaryContainer runmqsc QM1
```

结果

现在，本地应用程序在单独的容器中运行，现在可以成功运行命令 (例如 **dspmq**， **amqsput**， **amqsget** 和 **runmqsc**) 作为从辅助容器到 QM1 队列管理器的本地绑定。

如果未看到期望的结果，请参阅 第 136 页的『对名称空间应用程序进行故障诊断』 以获取更多信息。

Multi 对名称空间应用程序进行故障诊断

使用共享名称空间时，必须确保共享所有名称空间 (IPC， PID 和 UTS/hostname) 和已安装的卷，否则应用程序将无法工作。

请参阅 第 135 页的『在单独的容器中运行本地绑定应用程序』，以获取必须遵循的限制列表。

如果您的应用程序未满足列出的所有限制，那么可能会迁到容器启动的问题，但您期望的功能不起作用。

以下列表概述了一些常见原因，以及您可能看到的行为是否已忘记满足其中一个限制。

- 如果忘记共享名称空间 (UTS/PID/IPC) 或容器的主机名，并安装卷，那么容器将能够看到队列管理器，但无法与队列管理器交互。
  - 对于 **dspmq** 命令，您将看到以下内容:

```
docker exec container dspmq

QMNAME(QM1)                        STATUS(Status not available)
```

  - 对于 **runmqsc** 命令或尝试连接到队列管理器的其他命令，您可能会收到 AMQ8146 错误消息:

```
docker exec -it container runmqsc QM1
5724-H72 (C) Copyright IBM Corp. 1994, 2025.
Starting MQSC for queue manager QM1.
AMQ8146: IBM MQ queue manager not available
```

- 如果共享所有必需的名称空间，但未将共享卷安装到 /var/mqm 目录，并且您具有有效的 IBM MQ 数据路径，那么您的命令还会接收到 AMQ8146 错误消息。

  但是， **dspmq** 根本无法看到您的队列管理器，而是返回空白响应:

```
docker exec container dspmq
```

- 如果共享所有必需的名称空间，但未将共享卷安装到 /var/mqm 目录，并且您没有有效的 IBM MQ 数据路径 (或没有 IBM MQ 数据路径)，那么您会看到各种错误，因为数据路径是 IBM MQ 安装的关键组件。 如果没有数据路径，那么 IBM MQ 无法运行。

  如果运行以下任何命令，并且看到与这些示例中显示的响应类似的响应，那么应验证是否已安装该目录或创建 IBM MQ 数据目录:

```
docker exec container dspmq
'No such file or directory' from /var/mqm/mqs.ini
AMQ6090: IBM MQ was unable to display an error message FFFFFFFF.
AMQffff

docker exec container dspmqver
AMQ7047: An unexpected error was encountered by a command. Reason code is 0.

docker exec container mqrc
<file path>/mqrc.c[1152]
lpiObtainQMDetails --> 545261715

docker exec container crtmqm QM1
AMQ8101: IBM MQ error (893) has occurred.

docker exec container strmqm QM1
AMQ6239: Permission denied attempting to access filesystem location '/var/mqm'.
```

```
AMQ7002: An error occurred manipulating a file.

docker exec container endmqm QM1
AMQ8101: IBM MQ error (893) has occurred.

docker exec container dltmqm QM1
AMQ7002: An error occurred manipulating a file.

docker exec container strmqweb
<file path>/mqrc.c[1152]
lpiObtainQMDetails --> 545261715
```

CP4I 创建本机 HA 组 (如果创建您自己的容器)

您必须创建，配置和启动三个队列管理器以创建本机 HA 组。

关于此任务

创建本机 HA 解决方案的建议方法是使用 IBM MQ 操作程序 (请参阅 本机 HA)。 或者，如果您创建自己的容器，那么可以遵循以下指示信息。

要创建本机 HA 组，请在日志类型设置为 `log replication` 的三个节点上创建三个队列管理器。 然后，编辑每个队列管理器的 `qm.ini` 文件，以添加三个节点中每个节点的连接详细信息，以便它们可以相互复制日志数据。

然后，必须启动所有三个队列管理器，以便它们可以检查所有三个实例是否可以相互通信，并确定其中哪些将是活动实例，哪些将是副本。

过程

1. 在三个节点中的每个节点上，创建一个队列管理器，指定日志类型的日志副本，并为每个日志实例提供唯一的名称。 每个队列管理器具有相同的名称:

   ```
   crtmqm -lr instance_name qmname
   ```

   例如:

   ```
   node 1> crtmqm -lr qm1_inst1 qm1

   node 2> crtmqm -lr qm1_inst2 qm1

   node 3> crtmqm -lr qm1_inst3 qm1
   ```

2. 成功创建每个队列管理器时，会将名为 `NativeHALocalInstance` 的附加节添加到队列管理器配置文件 `qm.ini`。 `Name` 属性将添加到指定所提供实例名称的节中。

   您可以选择性地将以下属性添加到 `qm.ini` 文件中的 `NativeHALocalInstance` 节:

   KeyRepository
   保存要用于保护日志复制流量的数字证书的密钥存储库的位置。 该位置以词干格式提供，即，它包含不带扩展名的完整路径和文件名。 如果省略了 `KeyRepository` 节属性，那么将以纯文本在实例之间交换日志复制数据。

   CertificateLabel
   用于标识用于保护日志复制流量的数字证书的证书标签。 如果提供了 `KeyRepository` 但省略了 `CertificateLabel` ，那么将使用缺省值 `ibmwebspheremq`*`queue_manager`* 。

   CipherSpec
   用于保护日志复制流量的 MQ CipherSpec 。 如果提供了此节属性，那么还必须提供 `KeyRepository` 。 如果提供了 `KeyRepository` 但省略了 `CipherSpec` ，那么将使用缺省值 `ANY` 。

   LocalAddress
   接受日志复制流量的本地网络接口地址。 如果提供了此节属性，那么它将使用格式 "[addr] [(port)]" 来标识本地网络接口和/或端口。 可以将网络地址指定为主机名， IPv4 点分十进制或 IPv6 十六进制格式。 如果省略此属性，那么队列管理器将尝试绑定到所有网络接口，它将使用 `NativeHAInstances` 节中与本地实例名称匹配的 `ReplicationAddress` 中指定的端口。

HeartbeatInterval
脉动信号间隔定义本机 HA 队列管理器的活动实例发送网络脉动信号的频率 (以毫秒计)。 脉动信号间隔值的有效范围是 500 (0.5 秒) 到 60000 (1 分钟) ，超出此范围的值将导致队列管理器无法启动。 如果省略此属性，那么将使用缺省值 5000 (5 秒)。 每个实例必须使用相同的脉动信号间隔。

HeartbeatTimeout
脉动信号超时定义本机 HA 队列管理器的副本实例在确定活动实例无响应之前等待的时间长度。 脉动信号间隔超时值的有效范围为 500 (0.5 秒) 到 120000 (2 分钟)。 脉动信号超时的值必须大于或等于脉动信号间隔。

无效值导致队列管理器无法启动。 如果省略此属性，那么副本将在启动进程以选择新的活动实例之前等待 2 x `HeartbeatInterval` 。 每个实例必须使用相同的脉动信号超时。

RetryInterval
重试时间间隔定义本机 HA 队列管理器应重试失败复制链接的频率 (以毫秒计)。 重试时间间隔的有效范围为 500 (0.5 秒) 到 120000 (2 分钟)。 如果省略此属性，那么副本将在重试失败的复制链接之前等待 2 x `HeartbeatInterval` 。

3. 编辑每个队列管理器的 `qm.ini` 文件并添加连接详细信息。 添加三个 `NativeHAInstance` 节，一个用于本机 HA 组中的每个队列管理器实例 (包括本地实例)。 添加以下属性:

名称
指定创建队列管理器实例时使用的实例名称。

ReplicationAddress
指定实例的主机名， IPv4 点分十进制或 IPv6 十六进制格式地址。 可以将地址指定为主机名， IPv4 点分十进制或 IPv6 十六进制格式地址。 复制地址必须可解析且可从组中的每个实例路由。 必须在方括号中指定用于日志复制的端口号，例如:

```
ReplicationAddress=host1.example.com(4444)
```

注: `NativeHAInstance` 节在每个实例上都是相同的，可以使用自动配置 (**crtmqm -ii**) 来提供。

4. 启动三个实例中的每个实例:

```
strmqm QMgrName
```

启动实例时，它们通信以检查所有三个实例是否都在运行，然后确定这三个实例中的哪个是活动实例，而其他两个实例继续作为副本运行。

示例

以下示例显示了 `qm.ini` 文件中指定三个实例之一的必需本机 HA 详细信息的部分:

```
NativeHALocalInstance:
  LocalName=node-1

NativeHAInstance:
  Name=node-1
  ReplicationAddress=host1.example.com(4444)
NativeHAInstance:
  Name=node-2
  ReplicationAddress=host2.example.com(4444)
NativeHAInstance:
  Name=node-3
  ReplicationAddress=host3.example.com(4444)
```

Kubernetes 执行本机 HA 队列管理器的滚动更新的注意事项

对本机 HA 队列管理器的 IBM MQ 版本或 Pod 规范的任何更新都将要求您对队列管理器实例执行滚动更新。 IBM MQ Operator 会自动处理此问题，但如果您正在构建自己的部署代码，那么存在一些重要注意事项。

注: 样本 Helm Chart 包含用于执行滚动更新的 shell 脚本，但该脚本 **不** 适合生产用途，因为它未解决本主题中的注意事项。

在 Kubernetes 中， `StatefulSet` 资源用于管理有序启动和滚动更新。 启动过程的一部分是单独启动每个 Pod ，等待它准备就绪，然后移至下一个 Pod。 这对本机 HA 不起作用，因为所有 Pod 都需要启动，才能进

行领导者选举。 因此， `StatefulSet` 上的 `.spec.podManagementPolicy` 字段需要设置为 `Parallel`。 这也意味着所有 Pod 也会并行更新，这是特别不可取的。 因此， `StatefulSet` 还应该使用 `OnDelete` 更新策略。

无法使用 `StatefulSet` 滚动更新代码会导致需要定制滚动更新代码，这应考虑以下内容:

- 常规滚动更新过程
- 通过以最佳顺序更新 Pod ， 最大限度缩短停机时间
- 处理集群状态中的更改
- 处理错误
- 处理计时问题

常规滚动更新过程

滚动更新代码应等待每个实例显示来自 `dspmq` 的状态 `REPLICA` 。 这意味着实例已执行某种级别的启动 (例如，容器已启动， MQ 进程正在运行) ，但它还不一定能够与其他实例进行对话。 例如: Pod A 将重新启动，一旦处于 `REPLICA` 状态， Pod B 将重新启动。 一旦 Pod B 从新配置开始，它应该能够与 Pod A 对话，并且可以构成定额，并且 A 或 B 将成为新的活动实例。

作为此过程的一部分，在每个 Pod 达到 `REPLICA` 状态后有一个延迟很有用，以允许它连接到其同级并建立定额。

通过以最佳顺序更新 Pod ，最大限度缩短停机时间

滚动更新代码应该一次删除一个 Pod ，从处于已知错误状态的 Pod 开始，然后是未成功启动的任何 Pod。 通常应该最后更新活动队列管理器 Pod。

如果上次更新导致 Pod 进入已知错误状态，那么暂停删除 Pod 也很重要。 这将阻止在所有 Pod 中推出中断的更新。 例如，如果 Pod 更新为使用不可访问 (或包含 typo) 的新容器映像，那么可能会发生此情况。

处理集群状态中的更改

滚动更新代码需要对集群状态的实时更改作出相应的反应。 例如，其中一个队列管理器的 Pod 可能由于 Node 重新引导或 Node 压力而被逐出。 如果集群繁忙，那么可能不会立即重新调度已逐出的 Pod。 在这种情况下，滚动更新代码需要在重新启动任何其他 Pod 之前进行相应的等待。

处理错误

在调用 Kubernetes API 和其他意外集群行为时，滚动更新代码需要稳健以避免失败。

此外，滚动更新代码本身需要容忍被重新启动。 滚动更新可以长时间运行，并且可能需要重新启动代码。

处理计时问题

滚动更新代码需要检查 Pod 的更新修订版，这样可以确保 Pod 已重新启动。 这可避免 Pod 可能指示其 "已启动" 但实际上尚未终止的计时问题。

相关概念

第 5 页的『选择要在容器中使用 IBM MQ 的方式』
在容器中使用 IBM MQ 有多个选项: 您可以选择使用 IBM MQ Operator(使用预先打包的容器映像) ， 也可以构建自己的映像和部署代码。

CP4I 查看定制构建的容器的本机 HA 队列管理器的状态

对于定制构建的容器，您可以使用 **dspmq** 命令来查看本机 HA 实例的状态。

关于此任务

您可以使用 **dspmq** 命令来查看节点上队列管理器实例的操作状态。 返回的信息取决于实例是活动实例还是副本实例。 活动实例提供的信息是明确的，来自副本节点的信息可能已过时。

您可以执行以下操作：

- 查看当前节点上的队列管理器实例是处于活动状态还是处于副本状态。
- 查看当前节点上实例的本机 HA 操作状态。
- 查看本机 HA 配置中所有三个实例的操作状态。

以下状态字段用于报告本机 HA 配置状态:

角色
指定实例的当前角色，该角色是 `Active`， `Replica` 或 `Unknown` 之一。

INSTANCE
使用 **crtmqm** 命令的 **-lr** 选项创建队列管理器时为此实例提供的名称。

INSYNC
指示实例是否能够作为活动实例进行接管 (如果需要)。

QUORUM
以 *number_of_instances_in-sync*/*number_of_instances_configured* 格式报告定额状态。

REPLADDR
队列管理器实例的复制地址。

连接 ACTV
指示节点是否已连接到活动实例。

BACKLOG
指示实例延迟的 KB 数。

连接
指示指定的实例是否已连接到此实例。

ALTDATE
指示上次更新此信息的日期 (如果从未更新此信息，那么为空白)。

ALTTIME
指示上次更新此信息的时间 (如果从未更新此信息，那么为空白)。

过程

- 要确定队列管理器实例是作为活动实例运行还是作为副本运行:

  ```
  dspmq -o status -m QMgrName
  ```

  名为 BOB 的队列管理器的活动实例将报告以下状态:

  ```
  QMNAME(BOB)                 STATUS(Running)
  ```

  名为 BOB 的队列管理器的副本实例将报告以下状态:

  ```
  QMNAME(BOB)                 STATUS(Replica)
  ```

  不活动的实例将报告以下状态:

  ```
  QMNAME(BOB)                 STATUS(Ended Immediately)
  ```

- 要确定当前节点上实例的本机 HA 操作状态，请执行以下操作:

  ```
  dspmq -o nativeha -m QMgrName
  ```

  名为 BOB 的队列管理器的活动实例可能会报告以下状态:

```
QMNAME(BOB)               ROLE(Active) INSTANCE(inst1) INSYNC(Yes) QUORUM(3/3)
```

名为 BOB 的队列管理器的副本实例可能会报告以下状态:

```
QMNAME(BOB)               ROLE(Replica) INSTANCE(inst2) INSYNC(Yes) QUORUM(2/3)
```

名为 BOB 的队列管理器的不活动实例可能会报告以下状态:

```
QMNAME(BOB)               ROLE(Unknown) INSTANCE(inst3) INSYNC(no) QUORUM(0/3)
```

- 要确定本机 HA 配置中所有实例的本机 HA 操作状态:

```
dspmq -o nativeha -x -m QMgrName
```

如果在运行队列管理器 BOB 的活动实例的节点上发出此命令，那么可能会收到以下状态:

```
QMNAME(BOB)            ROLE(Active) INSTANCE(inst1) INSYNC(Yes) QUORUM(3/3)
 INSTANCE(inst1) ROLE(Active)  REPLADDR(9.20.123.45) CONNACTV(Yes) INSYNC(Yes) BACKLOG(0)
CONNINST(Yes) ALTDATE(2022-01-12) ALTTIME(12.03.44)
 INSTANCE(inst2) ROLE(Replica) REPLADDR(9.20.123.46) CONNACTV(Yes) INSYNC(Yes) BACKLOG(0)
CONNINST(Yes) ALTDATE(2022-01-12) ALTTIME(12.03.44)
 INSTANCE(inst3) ROLE(Replica) REPLADDR(9.20.123.47) CONNACTV(Yes) INSYNC(Yes) BACKLOG(0)
CONNINST(Yes) ALTDATE(2022-01-12) ALTTIME(12.03.44)
```

如果在运行队列管理器 BOB 的副本实例的节点上发出此命令，那么可能会收到以下状态，这指示其中一个副本落后:

```
QMNAME(BOB)            ROLE(Replica) INSTANCE(inst2) INSYNC(Yes) QUORUM(2/3)
 INSTANCE(inst2) ROLE(Replica) REPLADDR(9.20.123.46) CONNACTV(Yes) INSYNC(Yes) BACKLOG(0)
CONNINST(Yes) ALTDATE(2022-01-12) ALTTIME(12.03.44)
 INSTANCE(inst1) ROLE(Active)  REPLADDR(9.20.123.45) CONNACTV(Yes) INSYNC(Yes) BACKLOG(0)
CONNINST(Yes) ALTDATE(2022-01-12) ALTTIME(12.03.44)
 INSTANCE(inst3) ROLE(Replica) REPLADDR(9.20.123.47) CONNACTV(Yes) INSYNC(No)  BACKLOG(435)
CONNINST(Yes) ALTDATE(2022-01-12) ALTTIME(12.03.44)
```

如果在运行队列管理器 BOB 的不活动实例的节点上发出此命令，那么可能会收到以下状态:

```
QMNAME(BOB)            ROLE(Unknown) INSTANCE(inst3) INSYNC(no) QUORUM(0/3)
 INSTANCE(inst1) ROLE(Unknown) REPLADDR(9.20.123.45) CONNACTV(Unknown) INSYNC(Unknown)
BACKLOG(Unknown) CONNINST(No) ALTDATE() ALTTIME()
 INSTANCE(inst2) ROLE(Unknown) REPLADDR(9.20.123.46) CONNACTV(Unknown) INSYNC(Unknown)
BACKLOG(Unknown) CONNINST(No) ALTDATE() ALTTIME()
 INSTANCE(inst3) ROLE(Unknown) REPLADDR(9.20.123.47) CONNACTV(No) INSYNC(Unknown)
BACKLOG(Unknown) CONNINST(No) ALTDATE() ALTTIME()
```

如果在实例仍在协商哪些是活动的副本时发出该命令，那么您将收到以下状态:

```
QMNAME(BOB)              STATUS(Negotiating)
```

相关参考

dspmq

CP4I 结束本机 HA 队列管理器

您可以使用 **endmqm** 命令来结束属于本机 HA 组的活动队列管理器或副本队列管理器。

过程

- 要结束队列管理器的活动实例，请参阅本文档的 "配置" 部分中的 结束本机 HA 队列管理器 。

声明

本信息是为在美国提供的产品和服务编写的。

IBM 可能在其他国家或地区不提供本文档中讨论的产品、服务或功能。 有关您当前所在区域的产品和服务的信息，请向您当地的 IBM 代表咨询。 任何对 IBM 产品、程序或服务的引用并非意在明示或默示只能使用 IBM 的产品、程序或服务。 只要不侵犯 IBM 的知识产权，任何同等功能的产品、程序或服务都可以代替 IBM 产品、程序或服务。 但是， 评估和验证任何非 IBM 产品、程序或服务的操作，由用户自行负责。

IBM 公司可能已拥有或正在申请与本文档内容有关的各项专利。 提供本文档并未授予用户使用这些专利的任何许可。 您可以以书面形式将许可查询寄往：

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

有关双字节（DBCS）信息的许可查询，请与您所在国家或地区的 IBM 知识产权部门联系，或用书面方式将查询寄往：

知识产权许可
Legal and Intellectual Property Law
IBM Japan, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 063-8506 Japan

本条款不适用英国或任何这样的条款与当地法律不一致的国家或地区: International Business Machines Corporation “按现状” 提供本出版物，不附有任何种类的 (无论是明示的还是暗含的) 保证，包括但不限于暗含的有关非侵权，适销和适用于某种特定用途的保证。 某些国家或地区在某些交易中不允许免除明示或暗含的保证。 因此本条款可能不适用于您。

本信息中可能包含技术方面不够准确的地方或印刷错误。 此处的信息将定期更改；这些更改将编入本资料的新版本中。 IBM 可以随时对本出版物中描述的产品和/或程序进行改进和/或更改，而不另行通知。

本信息中对非 IBM Web 站点的任何引用都只是为了方便起见才提供的，不以任何方式 充当对那些 Web 站点的保证。 那些 Web 站点中的资料不是 IBM 产品资料的一部分，使用那些 Web 站点带来的风险将由您自行承担。

IBM 可以按它认为适当的任何方式使用或分发您所提供的任何信息而无须对您承担任何责任。

本程序的被许可方如果要了解有关程序的信息以达到如下目的：(i) 允许在独立创建的程序和其他程序（包括本程序）之间进行信息交换，以及 (ii) 允许对已经交换的信息进行相互使用，请与下列地址联系：

IBM Corporation
软件互操作性协调员，部门 49XA
北纬 3605 号公路
罗切斯特，明尼苏达州 55901
U.S.A.

只要遵守适当的条件和条款，包括某些情形下的一定数量的付费，都可获得这方面的信息。

本资料中描述的许可程序及其所有可用的许可资料均由 IBM 依据 IBM 客户协议、IBM 国际软件许可协议或任何同等协议中的条款提供。

此处包含的任何性能数据都是在受控环境中测得的。 因此，在其他操作环境中获得的数据可能会有明显的不同。 有些测量可能是在开发级的系统上进行的，因此不保证与一般可用系统上进行的测量结果相同。 此外，有些测量是通过推算而估计的， 实际结果可能会有差异。 本文档的用户应当验证其特定环境的适用数据。

© Copyright IBM Corp. 2007, 2025

涉及非 IBM 产品的信息可从这些产品的供应商、其出版说明或其他可公开获得的资料中 获取。 IBM 没有对这些产品进行测试，也无法确认其性能的精确性、兼容性或任何其他关于非 IBM 产品的声明。 有关非 IBM 产品性能的问题应当向这些产品的供应商提出。

所有关于 IBM 未来方向或意向的声明都可随时更改或收回，而不另行通知，它们仅仅表示了目标和意愿而已。

本信息包含日常商业运作所使用的数据和报表的示例。 为了尽可能全面地说明这些数据和报表，这些示例包括个人、公司、品牌和产品的名称。 所有这些名称都是虚构的，如与实际商业企业所使用的名称和地址有任何雷同，纯属巧合。

版权许可：

本信息包含源语言形式的样本应用程序，用以阐明在不同操作平台上的编程技术。 如果是为按照在编写样本程序的操作平台上的应用程序编程接口（API）进行应用程序的开发、使用、经销或分发为目的，您可以任何形式对这些样本程序进行复制、修改、分发，而无须向 IBM 付费。 这些示例并未在所有条件下作全面测试。 因此，IBM 不能担保或默示这些程序的可靠性、可维护性或功能。

如果您正在查看本信息的软拷贝，图片和彩色图例可能无法显示。

编程接口信息

编程接口信息 (如果提供) 旨在帮助您创建用于此程序的应用软件。

本书包含有关允许客户编写程序以获取 WebSphere MQ 服务的预期编程接口的信息。

但是，该信息还可能包含诊断、修改和调优信息。 提供诊断、修改和调优信息是为了帮助您调试您的应用程序软件。

要点: 请勿将此诊断，修改和调整信息用作编程接口，因为它可能会发生更改。

商标

IBM IBM 徽标 ibm.com 是 IBM Corporation 在全球许多管辖区域的商标。 当前的 IBM 商标列表可从 Web 上的 “Copyright and trademark information”www.ibm.com/legal/copytrade.shtml 获取。 其他产品和服务名称可能是 IBM 或其他公司的商标。

Microsoft 和 Windows 是 Microsoft Corporation 在美国和/或其他国家或地区的商标。

UNIX 是 Open Group 在美国和其他国家或地区的注册商标。

Linux 是 Linus Torvalds 在美国和/或其他国家或地区的商标。

此产品包含由 Eclipse 项目 (https://www.eclipse.org/) 开发的软件。

Java 和所有基于 Java 的商标和徽标是 Oracle 和/或其附属公司的商标或注册商标。

部件号:

(1P) P/N: