9.4

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 267 页的『声明』中的信息。

本版本适用于 IBM® MQ V 9 发行版 4 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，而无需对您承担任何责任。

© Copyright International Business Machines Corporation 2007, 2025.

内容

容器和 IBM Cloud Pak for Integration 中的 IBM MQ.....5
关于.....5
IBM MQ Operator 的发布历史记录.....5
队列管理器容器映像的发布历史.....17
规划.....26
选择要在容器中使用 IBM MQ 的方式.....26
支持容器中的 IBM MQ.....27
规划容器中的 IBM MQ 许可.....34
规划 IBM MQ Operator 的存储器.....35
规划容器中 IBM MQ 的高可用性.....36
容器中 IBM MQ 的灾难恢复.....41
规划容器中 IBM MQ 的安全性.....44
规划容器中 IBM MQ 的可伸缩性和性能.....49
准备，安装和升级.....49
安装和升级 IBM MQ Operator.....50
通过创建拉取秘钥，为使用 "Kubernetes 做准备.....75
准备使用 Kubernetes ，安装 IBM License Service.....76
准备通过调用预构建的容器映像来使用 "Podman 或 "Docker.....76
通过构建您自己的容器映像来准备 IBM MQ.....76
部署和配置.....84
使用 IBM MQ Operator 部署和配置队列管理器.....84
在 Kubernetes 上部署和配置队列管理器.....151
在 "Podman 上部署队列管理器 165
迁移到 IBM MQ Operator.....169
正在检查必需的功能是否可用.....170
抽取队列管理器配置.....170
可选: 抽取和获取队列管理器密钥和证书.....171
可选: 配置 LDAP.....173
可选: 更改 IBM MQ 配置中的 IP 地址和主机名.....179
更新容器环境的队列管理器配置.....180
为在容器中运行的 IBM MQ 选择目标 HA 体系结构.....183
为队列管理器创建资源.....184
在 Red Hat OpenShift 上创建新的队列管理器.....185
验证新的容器部署.....189
操作.....190
使用 IBM MQ Operator 操作 IBM MQ.....190
查看本机 HA 队列管理器的状态.....203
本地 HA CRR 切换和故障切换.....208
结束本地 HA 队列管理器.....211
故障诊断.....212
对容器中 IBM MQ 的意外重新启动进行故障诊断.....212
对 IBM MQ Operator 的问题进行故障诊断.....213
收集部署在 Kubernetes 上的队列管理器的故障排除信息.....219
参考.....219
IBM MQ Operator 的许可和 API 参考资料.....219
IBM MQ 集装箱许可证注释.....249
IBM MQ 应用的容器许可注释。 IBM MQ Operator.....254
IBM MQ Advanced 容器映像 (container image).....255
IBM MQ 许可的容器映像.....258
IBM MQ Advanced for Developers 容器映像 (container image).....258
IBM MQ MFT Agent for Developers 容器映像 (container image).....260
在 "Red Hat OpenShift 上进行空隙安装和升级的 IBM MQ CASE.....261

声明.....267
编程接口信息.....268
商标.....268

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 。

有关 IBM MQ Operator 的更多信息，请参阅以下链接。

关于容器中的 IBM MQ

用于帮助您开始使用容器中的 IBM MQ 的介绍性信息。

容器是一种允许将代码与其运行时环境打包和隔离的技术，可以通过与同一基础结构上的其他软件隔离的方式运行。 这使您能够轻松地在环境 (例如，开发，测试和生产) 之间移动队列管理器或应用程序。 现代容器编排器 (例如 Red Hat OpenShift Container Platform 和 Kubernetes) 可以在同一机器上运行多种类型的容器，每个容器在资源，安全性和故障方面相互隔离。

您可以在容器中运行 IBM MQ 队列管理器或 IBM MQ 应用程序。

相关信息
什么是容器?

##

IBM MQ Operator 的发布历史记录

注意:

- 有关较早的 IBM MQ 操作程序的信息，请参阅 IBM MQ 9.3 文档中的 Release history for IBM MQ Operator 。
- 有关未来 IBM MQ 更新的信息，请参阅整个 IBM MQ 建议的修订和计划的维护发布日期 页面。

IBM MQ Operator 3.6.0

IBM Cloud Pak for Integration 版本
IBM Cloud Pak for Integration 16.1.2

操作员通道
v3.6

`.spec.version` 的允许值
9.4.3.0-r1

迁移期间 `.spec.version` 的允许值

9.3.0.0-r1, 9.3.0.0-r2, 9.3.0.0-r3, 9.3.0.1-r1, 9.3.0.1-r2, 9.3.0.1-r3, 9.3.0.1-r4, 9.3.0.3-r1, 9.3.0.4-r1, 9.3.0.4-r2, 9.3.0.5-r1, 9.3.0.5-r2, 9.3.0.5-r3, 9.3.0.6-r1, 9.3.0.10-r1, 9.3.0.10-r2, 9.3.0.11-r1, 9.3.0.11-r2, 9.3.0.15-r1, 9.3.0.16-r1 ,9.3.0.16-r2, 9.3.0.17-r1, 9.3.0.17-r2, 9.3.0.17-r3, 9.3.0.20-r1, 9.3.0.20-r2, 9.3.0.21-r1, 9.3.0.21-r2, 9.3.0.21-r3, 9.3.0.25-r1, 9.3.1.0-r1, 9.3.1.0-r2, 9.3.1.0-r3, 9.3.1.1-r1, 9.3.2.0-r1, 9.3.2.0-r2, 9.3.2.1-r1, 9.3.2.1-r2, 9.3.3.0-r1, 9.3.3.0-r2, 9.3.3.1-r1, 9.3.3.1-r2, 9.3.3.2-r1, 9.3.3.2-r2, 9.3.3.3-r1, 9.3.3.3-r2, 9.3.4.0-r1, 9.3.4.1-r1, 9.3.5.0-r1, 9.3.5.0-r2, 9.3.5.1-r1, 9.3.5.1-r2, 9.4.0.0-r1, 9.4.0.0-r2, 9.4.0.0-r3, 9.4.0.5-r1, 9.4.0.5-r2, 9.4.0.6-r1, 9.4.0.6-r2, 9.4.0.7-r1, 9.4.0.10-r1, 9.4.0.10-r2, 9.4.0.11-r1, 9.4.0.11-r2, 9.4.0.11-r3, 9.4.1.0-r1, 9.4.1.0-r2, 9.4.1.1-r1, 9.4.2.0-r1, 9.4.2.0-r2, 9.4.2.1-r1, 9.4.2.1-r2

Red Hat OpenShift Container Platform 版本
OpenShift Container Platform 4.12 及更高版本。

© Copyright IBM Corp. 2007, 2025

IBM Cloud Pak foundational services 版本
仅适用于 IBM Cloud Pak foundational services V 4.6 。

新增内容

- 改进了网络钩子访问验证的日志记录，以促进符合访问日志审计要求。 有关更多信息，请参阅第 131 页的『在容器中为 IBM MQ 配置审计日志』。
- IBM MQ Operator 已通过 IPv6 双栈环境验证，符合最新的美国联邦合规要求。
- 现在您可以配置 IBM MQ 队列管理器服务的 `ipFamilyPolicy` 和 `ipFamilies` 设置。 可在 `spec.service` 下进行配置。 这些设置适用于所有队列管理器服务。
- 现在，您可以使用 IBM MQ 许可证注释配置通过 IBM MQ Operator 部署的队列管理器。 有关更多信息，请参阅第 140 页的『使用 IBM MQ 许可证注释配置队列管理器。 IBM MQ Operator』。 请注意，此更新适用于所有 IBM MQ 9.4 队列管理器
- 通过添加新的 IBM MQ Native HA 和 Cross-Region Replication Add-on 许可证，您现在可以在 IBM MQ 许可的队列管理器上使用 Native HA 和 Cross-Region Replication 功能。 有关更多信息，请参阅第 101 页的『在 IBM MQ 许可的队列管理器上配置本地 HA 和跨区域复制，方法是使用 IBM MQ Operator』。 请注意，此更新也适用于 IBM MQ 9.4.2 队列管理器。

IBM MQ Operator 3.5.3

CD

IBM Cloud Pak for Integration 版本
IBM Cloud Pak for Integration 16.1.1

操作员通道
v3.5

`.spec.version` 的允许值
9.4.2.1-r2

迁移期间 `.spec.version` 的允许值
9.3.0.0-r1, 9.3.0.0-r2, 9.3.0.0-r3, 9.3.0.1-r1, 9.3.0.1-r2, 9.3.0.1-r3, 9.3.0.1-r4, 9.3.0.3-r1, 9.3.0.4-r1, 9.3.0.4-r2, 9.3.0.5-r1, 9.3.0.5-r2, 9.3.0.5-r3, 9.3.0.6-r1, 9.3.0.10-r1, 9.3.0.10-r2, 9.3.0.11-r1, 9.3.0.11-r2, 9.3.0.15-r1, 9.3.0.16-r1 ,9.3.0.16-r2, 9.3.0.17-r1, 9.3.0.17-r2, 9.3.0.17-r3, 9.3.0.20-r1, 9.3.0.20-r2, 9.3.0.21-r1, 9.3.0.21-r2, 9.3.0.21-r3, 9.3.0.25-r1, 9.3.1.0-r1, 9.3.1.0-r2, 9.3.1.0-r3, 9.3.1.1-r1, 9.3.2.0-r1, 9.3.2.0-r2, 9.3.2.1-r1, 9.3.2.1-r2, 9.3.3.0-r1, 9.3.3.0-r2, 9.3.3.1-r1, 9.3.3.1-r2, 9.3.3.2-r1, 9.3.3.2-r2, 9.3.3.3-r1, 9.3.3.3-r2, 9.3.4.0-r1, 9.3.4.1-r1, 9.3.5.0-r1, 9.3.5.0-r2, 9.3.5.1-r1, 9.3.5.1-r2, 9.4.0.0-r1, 9.4.0.0-r2, 9.4.0.0-r3, 9.4.0.5-r1, 9.4.0.5-r2, 9.4.0.6-r1, 9.4.0.6-r2, 9.4.0.7-r1, 9.4.0.10-r1, 9.4.0.10-r2, 9.4.0.11-r1, 9.4.1.0-r1, 9.4.1.0-r2, 9.4.1.1-r1, 9.4.2.0-r2, 9.4.2.1-r1

Red Hat OpenShift Container Platform 版本
OpenShift Container Platform 4.12 及更高版本。

IBM Cloud Pak foundational services 版本
仅适用于 IBM Cloud Pak foundational services V 4.6 。

已更改的内容

- 本安全公告中详细介绍了已解决的漏洞：https://www.ibm.com/support/pages/node/7234178

IBM MQ Operator 3.5.2

CD

IBM Cloud Pak for Integration 版本
IBM Cloud Pak for Integration 16.1.1

操作员通道
v3.5

`.spec.version` 的允许值
9.4.2.1-r1

迁移期间 `.spec.version` 的允许值
9.3.0.0-r1, 9.3.0.0-r2, 9.3.0.0-r3, 9.3.0.1-r1, 9.3.0.1-r2, 9.3.0.1-r3, 9.3.0.1-r4, 9.3.0.3-r1, 9.3.0.4-r1, 9.3.0.4-r2, 9.3.0.5-r1, 9.3.0.5-r2, 9.3.0.5-r3, 9.3.0.6-r1, 9.3.0.10-r1, 9.3.0.10-r2, 9.3.0.11-r1, 9.3.0.11-r2, 9.3.0.15-r1, 9.3.0.16-r1 ,9.3.0.16-r2, 9.3.0.17-r1, 9.3.0.17-r2, 9.3.0.17-r3, 9.3.0.20-r1, 9.3.0.20-r2, 9.3.0.21-r1, 9.3.0.21-r2, 9.3.0.21-r3, 9.3.0.25-r1, 9.3.1.0-r1, 9.3.1.0-r2, 9.3.1.0-r3, 9.3.1.1-r1, 9.3.2.0-r1, 9.3.2.0-r2, 9.3.2.1-r1, 9.3.2.1-r2, 9.3.3.0-r1, 9.3.3.0-r2, 9.3.3.1-r1, 9.3.3.1-r2, 9.3.3.2-r1, 9.3.3.2-r2, 9.3.3.3-r1, 9.3.3.3-r2, 9.3.4.0-r1, 9.3.4.1-r1, 9.3.5.0-r1, 9.3.5.0-r2, 9.3.5.1-r1, 9.3.5.1-r2, 9.4.0.0-r1, 9.4.0.0-r2, 9.4.0.0-r3, 9.4.0.5-r1, 9.4.0.5-r2, 9.4.0.6-r1, 9.4.0.6-r2, 9.4.0.7-r1, 9.4.0.10-r1, 9.4.0.10-r2, 9.4.0.11-r1, 9.4.1.0-r1, 9.4.1.0-r2, 9.4.1.1-r1, 9.4.2.0-r2

Red Hat OpenShift Container Platform 版本
OpenShift Container Platform 4.12 及更高版本。

IBM Cloud Pak foundational services 版本
仅适用于 IBM Cloud Pak foundational services V 4.6 。

已更改的内容

- IBM MQ OpenID Connect (OIDC) 客户端 ID 和客户秘密现在存储在卷中，而不是环境变量中。
- 本安全公告中详细介绍了已解决的漏洞：https://www.ibm.com/support/pages/node/7232272

IBM MQ Operator 3.5.1

CD

IBM Cloud Pak for Integration 版本
IBM Cloud Pak for Integration 16.1.1

操作员通道
v3.5

`.spec.version` 的允许值
9.4.2.0-r2

迁移期间 `.spec.version` 的允许值
9.3.0.0-r1, 9.3.0.0-r2, 9.3.0.0-r3, 9.3.0.1-r1, 9.3.0.1-r2, 9.3.0.1-r3, 9.3.0.1-r4, 9.3.0.3-r1, 9.3.0.4-r1, 9.3.0.4-r2, 9.3.0.5-r1, 9.3.0.5-r2, 9.3.0.5-r3, 9.3.0.6-r1, 9.3.0.10-r1, 9.3.0.10-r2, 9.3.0.11-r1, 9.3.0.11-r2, 9.3.0.15-r1, 9.3.0.16-r1 ,9.3.0.16-r2, 9.3.0.17-r1, 9.3.0.17-r2, 9.3.0.17-r3, 9.3.0.20-r1, 9.3.0.20-r2, 9.3.0.21-r1, 9.3.0.21-r2, 9.3.0.21-r3, 9.3.0.25-r1, 9.3.1.0-r1, 9.3.1.0-r2, 9.3.1.0-r3, 9.3.1.1-r1, 9.3.2.0-r1, 9.3.2.0-r2, 9.3.2.1-r1, 9.3.2.1-r2, 9.3.3.0-r1, 9.3.3.0-r2, 9.3.3.1-r1, 9.3.3.1-r2, 9.3.3.2-r1, 9.3.3.2-r2, 9.3.3.3-r1, 9.3.3.3-r2, 9.3.4.0-r1, 9.3.4.1-r1, 9.3.5.0-r1, 9.3.5.0-r2, 9.3.5.1-r1, 9.3.5.1-r2, 9.4.0.0-r1, 9.4.0.0-r2, 9.4.0.0-r3, 9.4.0.5-r1, 9.4.0.5-r2, 9.4.0.6-r1, 9.4.0.6-r2, 9.4.0.7-r1, 9.4.0.10-r1, 9.4.0.10-r2, 9.4.1.0-r1, 9.4.1.0-r2, 9.4.1.1-r1, 9.4.2.0-r1

Red Hat OpenShift Container Platform 版本
OpenShift Container Platform 4.12 及更高版本。

IBM Cloud Pak foundational services 版本
仅适用于 IBM Cloud Pak foundational services V 4.6 。

已更改的内容

- 修正了一个问题，即在特定条件下， IntegrationKeycloakClient 资源会更新为不正确的客户 ID。
- 本安全公告中详细介绍了已解决的漏洞：https://www.ibm.com/support/pages/node/7228945

IBM MQ Operator 3.5.0

CD

IBM Cloud Pak for Integration 版本
IBM Cloud Pak for Integration 16.1.1

操作员通道
v3.5

`.spec.version` 的允许值
9.4.2.0-r1

迁移期间 .spec.version 的允许值

9.3.0.0-r1, 9.3.0.0-r2, 9.3.0.0-r3, 9.3.0.1-r1, 9.3.0.1-r2, 9.3.0.1-r3, 9.3.0.1-r4, 9.3.0.3-r1, 9.3.0.4-r1, 9.3.0.4-r2, 9.3.0.5-r1, 9.3.0.5-r2, 9.3.0.5-r3, 9.3.0.6-r1, 9.3.0.10-r1, 9.3.0.10-r2, 9.3.0.11-r1, 9.3.0.11-r2, 9.3.0.15-r1, 9.3.0.16-r1, 9.3.0.16-r2, 9.3.0.17-r1, 9.3.0.17-r2, 9.3.0.17-r3, 9.3.0.20-r1, 9.3.0.20-r2, 9.3.0.21-r1, 9.3.0.21-r2, 9.3.0.21-r3, 9.3.0.25-r1, 9.3.1.0-r1, 9.3.1.0-r2, 9.3.1.0-r3, 9.3.1.1-r1, 9.3.2.0-r1, 9.3.2.0-r2, 9.3.2.1-r1, 9.3.2.1-r2, 9.3.3.0-r1, 9.3.3.0-r2, 9.3.3.1-r1, 9.3.3.1-r2, 9.3.3.2-r1, 9.3.3.2-r2, 9.3.3.3-r1, 9.3.3.3-r2, 9.3.4.0-r1, 9.3.4.1-r1, 9.3.5.0-r1, 9.3.5.0-r2, 9.3.5.1-r1, 9.3.5.1-r2, 9.4.0.0-r1, 9.4.0.0-r2, 9.4.0.0-r3, 9.4.0.5-r1, 9.4.0.5-r2, 9.4.0.6-r1, 9.4.0.6-r2, 9.4.0.7-r1, 9.4.0.10-r1, 9.4.1.0-r1, 9.4.1.0-r2, 9.4.1.1-r1

Red Hat OpenShift Container Platform 版本

OpenShift Container Platform 4.12 及更高版本。

IBM Cloud Pak foundational services 版本

仅适用于 IBM Cloud Pak foundational services V 4.6 。

新增内容

- 部署队列管理器时，现在可以添加其他文件。 这些文件被安装在 IBM MQ 容器中的指定位置。 有关更多信息，请参阅 第 88 页的『示例：向队列管理器提供附加文件』。
- 现在，在部署队列管理器时，可以设置其他环境变量。 有关更多信息，请参阅第 90 页的『示例：为队列管理器提供额外的环境变量』。
- IBM MQ 来自 的队列管理员可以通过 进行配置，以便使用 创建的证书通过 提供指标。 9.4.2.0-r1 IBM MQ Operator OpenShift HTTPS 有关更多信息，请参阅第 191 页的『使用 IBM MQ Operator 时进行监视』。
- IBM MQ CASE 3.5.0 以后只包含最新的队列管理器版本。 例如， IBM MQ CASE 3.5.0 仅包含队列管理器版本 9.4.2.0-r1。 在执行空隙安装时，镜像步骤会更快，但您可能需要执行额外的镜像步骤，才能安装或升级到非最新版本的队列管理器。 有关更多信息，请参阅第 261 页的『在 "Red Hat OpenShift 上进行空隙安装和升级的 IBM MQ CASE』。

已更改的内容

本安全公告中详细介绍了已解决的漏洞：https://www.ibm.com/support/pages/node/7184453

IBM MQ Operator 3.4.1

IBM Cloud Pak for Integration 版本

IBM Cloud Pak for Integration 16.1.1

操作员通道

v3.4

.spec.version 的允许值

9.4.1.1-r1

迁移期间 .spec.version 的允许值

9.3.0.0-r1, 9.3.0.0-r2, 9.3.0.0-r3, 9.3.0.1-r1, 9.3.0.1-r2, 9.3.0.1-r3, 9.3.0.1-r4, 9.3.0.3-r1, 9.3.0.4-r1, 9.3.0.4-r2, 9.3.0.5-r1, 9.3.0.5-r2, 9.3.0.5-r3, 9.3.0.6-r1, 9.3.0.10-r1, 9.3.0.10-r2, 9.3.0.11-r1, 9.3.0.11-r2, 9.3.0.15-r1, 9.3.0.16-r1, 9.3.0.16-r2, 9.3.0.17-r1, 9.3.0.17-r2, 9.3.0.17-r3, 9.3.0.20-r1, 9.3.0.20-r2, 9.3.0.21-r1, 9.3.0.21-r2, 9.3.0.21-r3, 9.3.0.25-r1, 9.3.1.0-r1, 9.3.1.0-r2, 9.3.1.0-r3, 9.3.1.1-r1, 9.3.2.0-r1, 9.3.2.0-r2, 9.3.2.1-r1, 9.3.2.1-r2, 9.3.3.0-r1, 9.3.3.0-r2, 9.3.3.1-r1, 9.3.3.1-r2, 9.3.3.2-r1, 9.3.3.2-r2, 9.3.3.3-r1, 9.3.3.3-r2, 9.3.4.0-r1, 9.3.4.1-r1, 9.3.5.0-r1, 9.3.5.0-r2, 9.3.5.1-r1, 9.3.5.1-r2, 9.4.0.0-r1, 9.4.0.0-r2, 9.4.0.0-r3, 9.4.0.5-r1, 9.4.0.5-r2, 9.4.0.6-r1, 9.4.0.6-r2, 9.4.0.7-r1, 9.4.1.0-r1, 9.4.1.0-r2

Red Hat OpenShift Container Platform 版本

OpenShift Container Platform 4.12 及更高版本。

IBM Cloud Pak foundational services 版本

仅适用于 IBM Cloud Pak foundational services V 4.6 。

已更改的内容

本安全公告中详细介绍了已解决的漏洞：https://www.ibm.com/support/pages/node/7182196

IBM MQ Operator 3.4.0

CD

IBM Cloud Pak for Integration 版本
IBM Cloud Pak for Integration 16.1.1

操作员通道
v3.4

.spec.version 的允许值
9.4.1.0-r2

迁移期间 .spec.version 的允许值
9.3.0.0-r1, 9.3.0.0-r2, 9.3.0.0-r3, 9.3.0.1-r1, 9.3.0.1-r2, 9.3.0.1-r3, 9.3.0.1-r4, 9.3.0.3-r1, 9.3.0.4-r1, 9.3.0.4-r2, 9.3.0.5-r1, 9.3.0.5-r2, 9.3.0.5-r3, 9.3.0.6-r1, 9.3.0.10-r1, 9.3.0.10-r2, 9.3.0.11-r1, 9.3.0.11-r2, 9.3.0.15-r1, 9.3.0.16-r1, 9.3.0.16-r2, 9.3.0.17-r1, 9.3.0.17-r2, 9.3.0.17-r3, 9.3.0.20-r1, 9.3.0.20-r2, 9.3.0.21-r1, 9.3.0.21-r2, 9.3.0.21-r3, 9.3.0.25-r1, 9.3.1.0-r1, 9.3.1.0-r2, 9.3.1.0-r3, 9.3.1.1-r1, 9.3.2.0-r1, 9.3.2.0-r2, 9.3.2.1-r1, 9.3.2.1-r2, 9.3.3.0-r1, 9.3.3.0-r2, 9.3.3.1-r1, 9.3.3.1-r2, 9.3.3.2-r1, 9.3.3.2-r2, 9.3.3.3-r1, 9.3.3.3-r2, 9.3.4.0-r1, 9.3.4.1-r1, 9.3.5.0-r1, 9.3.5.0-r2, 9.3.5.1-r1, 9.3.5.1-r2, 9.4.0.0-r1, 9.4.0.0-r2, 9.4.0.0-r3, 9.4.0.5-r1, 9.4.0.5-r2, 9.4.0.6-r1, 9.4.0.6-r2, 9.4.0.7-r1, 9.4.1.0-r1

Red Hat OpenShift Container Platform 版本
OpenShift Container Platform 4.12 及更高版本。

IBM Cloud Pak foundational services 版本
仅适用于 IBM Cloud Pak foundational services V 4.6 。

已更改的内容

- 从 CASE 中删除了一些旧的、不支持的操作员。
- 本安全公告中详细介绍了已解决的漏洞：https://www.ibm.com/support/pages/node/7178065

IBM MQ Operator 3.3.0

CD

IBM Cloud Pak for Integration 版本
IBM Cloud Pak for Integration 16.1.0

操作员通道
v3.3

.spec.version 的允许值
9.4.1.0-r1

迁移期间 .spec.version 的允许值
9.3.0.0-r1, 9.3.0.0-r2, 9.3.0.0-r3, 9.3.0.1-r1, 9.3.0.1-r2, 9.3.0.1-r3, 9.3.0.1-r4, 9.3.0.3-r1, 9.3.0.4-r1, 9.3.0.4-r2, 9.3.0.5-r1, 9.3.0.5-r2, 9.3.0.5-r3, 9.3.0.6-r1, 9.3.0.10-r1, 9.3.0.10-r2, 9.3.0.11-r1, 9.3.0.11-r2, 9.3.0.15-r1, 9.3.0.16-r1, 9.3.0.16-r2, 9.3.0.17-r1, 9.3.0.17-r2, 9.3.0.17-r3, 9.3.0.20-r1, 9.3.0.20-r2, 9.3.0.21-r1, 9.3.0.21-r2, 9.3.0.21-r3, 9.3.1.0-r1, 9.3.1.0-r2, 9.3.1.0-r3, 9.3.1.1-r1, 9.3.2.0-r1, 9.3.2.0-r2, 9.3.2.1-r1, 9.3.2.1-r2, 9.3.3.0-r1, 9.3.3.0-r2, 9.3.3.1-r1, 9.3.3.1-r2, 9.3.3.2-r1, 9.3.3.2-r2, 9.3.3.3-r1, 9.3.3.3-r2, 9.3.4.0-r1, 9.3.4.1-r1, 9.3.5.0-r1, 9.3.5.0-r2, 9.3.5.1-r1, 9.3.5.1-r2, 9.4.0.0-r1, 9.4.0.0-r2, 9.4.0.0-r3, 9.4.0.5-r1, 9.4.0.5-r2, 9.4.0.6-r1

Red Hat OpenShift Container Platform 版本
OpenShift Container Platform 4.12 及更高版本。

IBM Cloud Pak foundational services 版本
仅适用于 IBM Cloud Pak foundational services V 4.6 。

已更改的内容

- IBM MQ Advanced 许可证不能再配置为**开发**用途。
- 队列管理器现在已注释了正确的许可注释。 有关更新以前队列管理器的详细信息或说明，请参见 "第 150 页的『更正已部署队列管理器的许可证注释』。

- 有关 "**spec.license.metric** 值的规则现在由 webhook 执行。 现有队列管理器升级到 9.4.1.0-r1 及更高版本时必须遵守这些规则。
- 添加对 "IBM MQ Advanced for Non-Production Environment 许可证的支持。
- 删除使用 "IBM MQ Advanced 许可证进行部署且 "**Use** 设置为 "**开发** "的功能。
- 减少大型集群中 "IBM MQ Operator 的内存占用。
- 修复在 OCP4.16 上运行时 "`ServiceAccount` 对账的问题。
- 本安全公告中详细介绍了已解决的漏洞：https://www.ibm.com/support/pages/node/7174358

IBM MQ Operator 3.2.13

CP4I-SC2

IBM Cloud Pak for Integration 版本
IBM Cloud Pak for Integration 16.1.0

操作员通道
`v3.2-sc2`

`.spec.version` 的允许值
9.4.0.11-r3

迁移期间 `.spec.version` 的允许值

9.3.0.0-r1, 9.3.0.0-r2, 9.3.0.0-r3, 9.3.0.1-r1, 9.3.0.1-r2, 9.3.0.1-r3, 9.3.0.1-r4, 9.3.0.3-r1, 9.3.0.4-r1, 9.3.0.4-r2, 9.3.0.5-r1, 9.3.0.5-r2, 9.3.0.5-r3, 9.3.0.6-r1, 9.3.0.10-r1, 9.3.0.10-r2, 9.3.0.11-r1, 9.3.0.11-r2, 9.3.0.15-r1, 9.3.0.16-r1 ,9.3.0.16-r2, 9.3.0.17-r1, 9.3.0.17-r2, 9.3.0.17-r3, 9.3.0.20-r1, 9.3.0.20-r2, 9.3.0.21-r1, 9.3.0.21-r2, 9.3.0.21-r3, 9.3.0.25-r1, 9.3.1.0-r1, 9.3.1.0-r2, 9.3.1.0-r3, 9.3.1.1-r1, 9.3.2.0-r1, 9.3.2.0-r2, 9.3.2.1-r1, 9.3.2.1-r2, 9.3.3.0-r1, 9.3.3.0-r2, 9.3.3.1-r1, 9.3.3.1-r2, 9.3.3.2-r1, 9.3.3.2-r2, 9.3.3.3-r1, 9.3.3.3-r2, 9.3.4.0-r1, 9.3.4.1-r1, 9.3.5.0-r1, 9.3.5.0-r2, 9.3.5.1-r1, 9.3.5.1-r2, 9.4.0.0-r1, 9.4.0.0-r2, 9.4.0.0-r3, 9.4.0.5-r1, 9.4.0.5-r2, 9.4.0.6-r1, 9.4.0.6-r2, 9.4.0.7-r1, 9.4.0.10-r1, 9.4.0.10-r2, 9.4.0.11-r1, 9.4.0.11-r2

Red Hat OpenShift Container Platform 版本
OpenShift Container Platform 4.12 及更高版本。

IBM Cloud Pak foundational services 版本
仅适用于 IBM Cloud Pak foundational services V 4.6 。

IBM MQ Operator 3.2.12

CP4I-SC2

IBM Cloud Pak for Integration 版本
IBM Cloud Pak for Integration 16.1.0

操作员通道
`v3.2-sc2`

`.spec.version` 的允许值
9.4.0.11-r2

迁移期间 `.spec.version` 的允许值
9.3.0.0-r1, 9.3.0.0-r2, 9.3.0.0-r3, 9.3.0.1-r1, 9.3.0.1-r2, 9.3.0.1-r3, 9.3.0.1-r4, 9.3.0.3-r1, 9.3.0.4-r1, 9.3.0.4-r2, 9.3.0.5-r1, 9.3.0.5-r2, 9.3.0.5-r3, 9.3.0.6-r1, 9.3.0.10-r1, 9.3.0.10-r2, 9.3.0.11-r1, 9.3.0.11-r2, 9.3.0.15-r1, 9.3.0.16-r1 ,9.3.0.16-r2, 9.3.0.17-r1, 9.3.0.17-r2, 9.3.0.17-r3, 9.3.0.20-r1, 9.3.0.20-r2, 9.3.0.21-r1, 9.3.0.21-r2, 9.3.0.21-r3, 9.3.0.25-r1, 9.3.1.0-r1, 9.3.1.0-r2, 9.3.1.0-r3, 9.3.1.1-r1, 9.3.2.0-r1, 9.3.2.0-r2, 9.3.2.1-r1, 9.3.2.1-r2, 9.3.3.0-r1, 9.3.3.0-r2, 9.3.3.1-r1, 9.3.3.1-r2, 9.3.3.2-r1, 9.3.3.2-r2, 9.3.3.3-r1, 9.3.3.3-r2, 9.3.4.0-r1, 9.3.4.1-r1, 9.3.5.0-r1, 9.3.5.0-r2, 9.3.5.1-r1, 9.3.5.1-r2, 9.4.0.0-r1, 9.4.0.0-r2, 9.4.0.0-r3, 9.4.0.5-r1, 9.4.0.5-r2, 9.4.0.6-r1, 9.4.0.6-r2, 9.4.0.7-r1, 9.4.0.10-r1, 9.4.0.10-r2, 9.4.0.11-r1

Red Hat OpenShift Container Platform 版本
OpenShift Container Platform 4.12 及更高版本。

IBM Cloud Pak foundational services 版本
仅适用于 IBM Cloud Pak foundational services V 4.6 。

已更改的内容

- 本安全公告中详细介绍了已解决的漏洞：https://www.ibm.com/support/pages/node/7234178

IBM MQ Operator 3.2.11

CP4I-SC2

IBM Cloud Pak for Integration 版本
IBM Cloud Pak for Integration 16.1.0

操作员通道
`v3.2-sc2`

`.spec.version` 的允许值
9.4.0.11-r1

迁移期间 `.spec.version` 的允许值
9.3.0.0-r1, 9.3.0.0-r2, 9.3.0.0-r3, 9.3.0.1-r1, 9.3.0.1-r2, 9.3.0.1-r3, 9.3.0.1-r4, 9.3.0.3-r1, 9.3.0.4-r1, 9.3.0.4-r2, 9.3.0.5-r1, 9.3.0.5-r2, 9.3.0.5-r3, 9.3.0.6-r1, 9.3.0.10-r1, 9.3.0.10-r2, 9.3.0.11-r1, 9.3.0.11-r2, 9.3.0.15-r1, 9.3.0.16-r1 ,9.3.0.16-r2, 9.3.0.17-r1, 9.3.0.17-r2, 9.3.0.17-r3, 9.3.0.20-r1, 9.3.0.20-r2, 9.3.0.21-r1, 9.3.0.21-r2, 9.3.0.21-r3, 9.3.0.25-r1, 9.3.1.0-r1, 9.3.1.0-r2, 9.3.1.0-r3, 9.3.1.1-r1, 9.3.2.0-r1, 9.3.2.0-r2, 9.3.2.1-r1, 9.3.2.1-r2, 9.3.3.0-r1, 9.3.3.0-r2, 9.3.3.1-r1, 9.3.3.1-r2, 9.3.3.2-r1, 9.3.3.2-r2, 9.3.3.3-r1, 9.3.3.3-r2, 9.3.4.0-r1, 9.3.4.1-r1, 9.3.5.0-r1, 9.3.5.0-r2, 9.3.5.1-r1, 9.3.5.1-r2, 9.4.0.0-r1, 9.4.0.0-r2, 9.4.0.0-r3, 9.4.0.5-r1, 9.4.0.5-r2, 9.4.0.6-r1, 9.4.0.6-r2, 9.4.0.7-r1, 9.4.0.10-r1, 9.4.0.10-r2

Red Hat OpenShift Container Platform 版本
OpenShift Container Platform 4.12 及更高版本。

IBM Cloud Pak foundational services 版本
仅适用于 IBM Cloud Pak foundational services V 4.6 。

已更改的内容

- IBM MQ OpenID Connect (OIDC) 客户端 ID 和客户秘密现在存储在卷中，而不是环境变量中。
- 删除了未使用的 `sa-<namespace>` 图像拉取密文，以防止 Kubernetes 警告。
- 本安全公告中详细介绍了已解决的漏洞：https://www.ibm.com/support/pages/node/7232272

IBM MQ Operator 3.2.10

CP4I-SC2

IBM Cloud Pak for Integration 版本
IBM Cloud Pak for Integration 16.1.0

操作员通道
`v3.2-sc2`

`.spec.version` 的允许值
9.4.0.10-r2

迁移期间 `.spec.version` 的允许值
9.3.0.0-r1, 9.3.0.0-r2, 9.3.0.0-r3, 9.3.0.1-r1, 9.3.0.1-r2, 9.3.0.1-r3, 9.3.0.1-r4, 9.3.0.3-r1, 9.3.0.4-r1, 9.3.0.4-r2, 9.3.0.5-r1, 9.3.0.5-r2, 9.3.0.5-r3, 9.3.0.6-r1, 9.3.0.10-r1, 9.3.0.10-r2, 9.3.0.11-r1, 9.3.0.11-r2, 9.3.0.15-r1, 9.3.0.16-r1 ,9.3.0.16-r2, 9.3.0.17-r1, 9.3.0.17-r2, 9.3.0.17-r3, 9.3.0.20-r1, 9.3.0.20-r2, 9.3.0.21-r1, 9.3.0.21-r2, 9.3.0.21-r3, 9.3.0.25-r1, 9.3.1.0-r1, 9.3.1.0-r2, 9.3.1.0-r3, 9.3.1.1-r1, 9.3.2.0-r1, 9.3.2.0-r2, 9.3.2.1-r1, 9.3.2.1-r2, 9.3.3.0-r1, 9.3.3.0-r2, 9.3.3.1-r1, 9.3.3.1-r2, 9.3.3.2-r1, 9.3.3.2-r2, 9.3.3.3-r1, 9.3.3.3-r2, 9.3.4.0-r1, 9.3.4.1-r1, 9.3.5.0-r1, 9.3.5.0-r2, 9.3.5.1-r1, 9.3.5.1-r2, 9.4.0.0-r1, 9.4.0.0-r2, 9.4.0.0-r3, 9.4.0.5-r1, 9.4.0.5-r2, 9.4.0.6-r1, 9.4.0.6-r2, 9.4.0.7-r1, 9.4.0.10-r1

Red Hat OpenShift Container Platform 版本
OpenShift Container Platform 4.12 及更高版本。

IBM Cloud Pak foundational services 版本
仅适用于 IBM Cloud Pak foundational services V 4.6 。

已更改的内容

- 修正了一个问题，即在特定条件下， IntegrationKeycloakClient 资源会更新为不正确的客户 ID。
- 本安全公告中详细介绍了已解决的漏洞：https://www.ibm.com/support/pages/node/7228945

IBM MQ Operator 3.2.9

CP4I-SC2

IBM Cloud Pak for Integration 版本
IBM Cloud Pak for Integration 16.1.0

操作员通道
`v3.2-sc2`

`.spec.version` 的允许值
9.4.0.10-r1

迁移期间 `.spec.version` 的允许值
9.3.0.0-r1, 9.3.0.0-r2, 9.3.0.0-r3, 9.3.0.1-r1, 9.3.0.1-r2, 9.3.0.1-r3, 9.3.0.1-r4, 9.3.0.3-r1, 9.3.0.4-r1, 9.3.0.4-r2, 9.3.0.5-r1, 9.3.0.5-r2, 9.3.0.5-r3, 9.3.0.6-r1, 9.3.0.10-r1, 9.3.0.10-r2, 9.3.0.11-r1, 9.3.0.11-r2, 9.3.0.15-r1, 9.3.0.16-r1, 9.3.0.16-r2, 9.3.0.17-r1, 9.3.0.17-r2, 9.3.0.17-r3, 9.3.0.20-r1, 9.3.0.20-r2, 9.3.0.21-r1, 9.3.0.21-r2, 9.3.0.21-r3, 9.3.1.0-r1, 9.3.1.0-r2, 9.3.1.0-r3, 9.3.1.1-r1, 9.3.2.0-r1, 9.3.2.0-r2, 9.3.2.1-r1, 9.3.2.1-r2, 9.3.3.0-r1, 9.3.3.0-r2, 9.3.3.1-r1, 9.3.3.1-r2, 9.3.3.2-r1, 9.3.3.2-r2, 9.3.3.3-r1, 9.3.3.3-r2, 9.3.4.0-r1, 9.3.4.1-r1, 9.3.5.0-r1, 9.3.5.0-r2, 9.3.5.1-r1, 9.3.5.1-r2, 9.4.0.0-r1, 9.4.0.0-r2, 9.4.0.0-r3, 9.4.0.5-r1, 9.4.0.5-r2, 9.4.0.6-r1, 9.4.0.6-r2, 9.4.0.7-r1

Red Hat OpenShift Container Platform 版本
OpenShift Container Platform 4.12 及更高版本。

IBM Cloud Pak foundational services 版本
仅适用于 IBM Cloud Pak foundational services V 4.6 。

已更改的内容
本安全公告中详细介绍了已解决的漏洞：https://www.ibm.com/support/pages/node/7184453

IBM MQ Operator 3.2.8

CP4I-SC2

IBM Cloud Pak for Integration 版本
IBM Cloud Pak for Integration 16.1.0

操作员通道
`v3.2-sc2`

`.spec.version` 的允许值
9.4.0.7-r1

迁移期间 `.spec.version` 的允许值
9.3.0.0-r1, 9.3.0.0-r2, 9.3.0.0-r3, 9.3.0.1-r1, 9.3.0.1-r2, 9.3.0.1-r3, 9.3.0.1-r4, 9.3.0.3-r1, 9.3.0.4-r1, 9.3.0.4-r2, 9.3.0.5-r1, 9.3.0.5-r2, 9.3.0.5-r3, 9.3.0.6-r1, 9.3.0.10-r1, 9.3.0.10-r2, 9.3.0.11-r1, 9.3.0.11-r2, 9.3.0.15-r1, 9.3.0.16-r1, 9.3.0.16-r2, 9.3.0.17-r1, 9.3.0.17-r2, 9.3.0.17-r3, 9.3.0.20-r1, 9.3.0.20-r2, 9.3.0.21-r1, 9.3.0.21-r2, 9.3.0.21-r3, 9.3.1.0-r1, 9.3.1.0-r2, 9.3.1.0-r3, 9.3.1.1-r1, 9.3.2.0-r1, 9.3.2.0-r2, 9.3.2.1-r1, 9.3.2.1-r2, 9.3.3.0-r1, 9.3.3.0-r2, 9.3.3.1-r1, 9.3.3.1-r2, 9.3.3.2-r1, 9.3.3.2-r2, 9.3.3.3-r1, 9.3.3.3-r2, 9.3.4.0-r1, 9.3.4.1-r1, 9.3.5.0-r1, 9.3.5.0-r2, 9.3.5.1-r1, 9.3.5.1-r2, 9.4.0.0-r1, 9.4.0.0-r2, 9.4.0.0-r3, 9.4.0.5-r1, 9.4.0.5-r2, 9.4.0.6-r1, 9.4.0.6-r2

Red Hat OpenShift Container Platform 版本
OpenShift Container Platform 4.12 及更高版本。

IBM Cloud Pak foundational services 版本
仅适用于 IBM Cloud Pak foundational services V 4.6 。

已更改的内容
本安全公告中详细介绍了已解决的漏洞：https://www.ibm.com/support/pages/node/7182196

IBM MQ Operator 3.2.7

CP4I-SC2

IBM Cloud Pak for Integration 版本
IBM Cloud Pak for Integration 16.1.0

操作员通道
`v3.2-sc2`

`.spec.version` 的允许值
9.4.0.6-r2

迁移期间 `.spec.version` 的允许值
9.3.0.0-r1, 9.3.0.0-r2, 9.3.0.0-r3, 9.3.0.1-r1, 9.3.0.1-r2, 9.3.0.1-r3, 9.3.0.1-r4, 9.3.0.3-r1, 9.3.0.4-r1, 9.3.0.4-r2, 9.3.0.5-r1, 9.3.0.5-r2, 9.3.0.5-r3, 9.3.0.6-r1, 9.3.0.10-r1, 9.3.0.10-r2, 9.3.0.11-r1, 9.3.0.11-r2, 9.3.0.15-r1, 9.3.0.16-r1, 9.3.0.16-r2, 9.3.0.17-r1, 9.3.0.17-r2, 9.3.0.17-r3, 9.3.0.20-r1, 9.3.0.20-r2, 9.3.0.21-r1, 9.3.0.21-r2, 9.3.0.21-r3, 9.3.1.0-r1, 9.3.1.0-r2, 9.3.1.0-r3, 9.3.1.1-r1, 9.3.2.0-r1, 9.3.2.0-r2, 9.3.2.1-r1, 9.3.2.1-r2, 9.3.3.0-r1, 9.3.3.0-r2, 9.3.3.1-r1, 9.3.3.1-r2, 9.3.3.2-r1, 9.3.3.2-r2, 9.3.3.3-r1, 9.3.3.3-r2, 9.3.4.0-r1, 9.3.4.1-r1, 9.3.5.0-r1, 9.3.5.0-r2, 9.3.5.1-r1, 9.3.5.1-r2, 9.4.0.0-r1, 9.4.0.0-r2, 9.4.0.0-r3, 9.4.0.5-r1, 9.4.0.5-r2, 9.4.0.6-r1

Red Hat OpenShift Container Platform 版本
OpenShift Container Platform 4.12 及更高版本。

IBM Cloud Pak foundational services 版本
仅适用于 IBM Cloud Pak foundational services V 4.6 。

已更改的内容
本安全公告中详细介绍了已解决的漏洞：https://www.ibm.com/support/pages/node/7178065

IBM MQ Operator 3.2.6

CP4I-SC2

IBM Cloud Pak for Integration 版本
IBM Cloud Pak for Integration 16.1.0

操作员通道
`v3.2-sc2`

`.spec.version` 的允许值
9.4.0.6-r1

迁移期间 `.spec.version` 的允许值
9.3.0.0-r1, 9.3.0.0-r2, 9.3.0.0-r3, 9.3.0.1-r1, 9.3.0.1-r2, 9.3.0.1-r3, 9.3.0.1-r4, 9.3.0.3-r1, 9.3.0.4-r1, 9.3.0.4-r2, 9.3.0.5-r1, 9.3.0.5-r2, 9.3.0.5-r3, 9.3.0.6-r1, 9.3.0.10-r1, 9.3.0.10-r2, 9.3.0.11-r1, 9.3.0.11-r2, 9.3.0.15-r1, 9.3.0.16-r1, 9.3.0.16-r2, 9.3.0.17-r1, 9.3.0.17-r2, 9.3.0.17-r3, 9.3.0.20-r1, 9.3.0.20-r2, 9.3.0.21-r1, 9.3.0.21-r2, 9.3.0.21-r3, 9.3.1.0-r1, 9.3.1.0-r2, 9.3.1.0-r3, 9.3.1.1-r1, 9.3.2.0-r1, 9.3.2.0-r2, 9.3.2.1-r1, 9.3.2.1-r2, 9.3.3.0-r1, 9.3.3.0-r2, 9.3.3.1-r1, 9.3.3.1-r2, 9.3.3.2-r1, 9.3.3.2-r2, 9.3.3.3-r1, 9.3.3.3-r2, 9.3.4.0-r1, 9.3.4.1-r1, 9.3.5.0-r1, 9.3.5.0-r2, 9.3.5.1-r1, 9.3.5.1-r2, 9.4.0.0-r1, 9.4.0.0-r2, 9.4.0.0-r3, 9.4.0.5-r1, 9.4.0.5-r2

Red Hat OpenShift Container Platform 版本
OpenShift Container Platform 4.12 及更高版本。

IBM Cloud Pak foundational services 版本
仅适用于 IBM Cloud Pak foundational services V 4.6 。

已更改的内容
本安全公告中详细介绍了已解决的漏洞：https://www.ibm.com/support/pages/node/7174358

IBM MQ Operator 3.2.5

IBM Cloud Pak for Integration 版本
IBM Cloud Pak for Integration 16.1.0

操作员通道
`v3.2-sc2`

`.spec.version` 的允许值
9.4.0.5-r2

迁移期间 `.spec.version` 的允许值
9.3.0.0-r1, 9.3.0.0-r2, 9.3.0.0-r3, 9.3.0.1-r1, 9.3.0.1-r2, 9.3.0.1-r3, 9.3.0.1-r4, 9.3.0.3-r1, 9.3.0.4-r1, 9.3.0.4-r2, 9.3.0.5-r1, 9.3.0.5-r2, 9.3.0.5-r3, 9.3.0.6-r1, 9.3.0.10-r1, 9.3.0.10-r2, 9.3.0.11-r1, 9.3.0.11-r2, 9.3.0.15-r1, 9.3.0.16-r1, 9.3.0.16-r2, 9.3.0.17-r1, 9.3.0.17-r2, 9.3.0.17-r3, 9.3.0.20-r1, 9.3.0.20-r2, 9.3.0.21-r1, 9.3.0.21-r2, 9.3.1.0-r1, 9.3.1.0-r2, 9.3.1.0-r3, 9.3.1.1-r1, 9.3.2.0-r1, 9.3.2.0-r2, 9.3.2.1-r1, 9.3.2.1-r2, 9.3.3.0-r1, 9.3.3.0-r2, 9.3.3.1-r1, 9.3.3.1-r2, 9.3.3.2-r1, 9.3.3.2-r2, 9.3.3.3-r1, 9.3.3.3-r2, 9.3.4.0-r1, 9.3.4.1-r1, 9.3.5.0-r1, 9.3.5.0-r2, 9.3.5.1-r1, 9.3.5.1-r2, 9.4.0.0-r1, 9.4.0.0-r2, 9.4.0.0-r3, 9.4.0.5-r1

Red Hat OpenShift Container Platform 版本
OpenShift Container Platform 4.12 及更高版本。

IBM Cloud Pak foundational services 版本
仅适用于 IBM Cloud Pak foundational services V 4.6 。

已更改的内容
本安全公告中详细介绍了已解决的漏洞：https://www.ibm.com/support/pages/node/7171536

IBM MQ Operator 3.2.4

IBM Cloud Pak for Integration 版本
IBM Cloud Pak for Integration 16.1.0

操作员通道
`v3.2-sc2`

`.spec.version` 的允许值
9.4.0.5-r1

迁移期间 `.spec.version` 的允许值
9.3.0.0-r1, 9.3.0.0-r2, 9.3.0.0-r3, 9.3.0.1-r1, 9.3.0.1-r2, 9.3.0.1-r3, 9.3.0.1-r4, 9.3.0.3-r1, 9.3.0.4-r1, 9.3.0.4-r2, 9.3.0.5-r1, 9.3.0.5-r2, 9.3.0.5-r3, 9.3.0.6-r1, 9.3.0.10-r1, 9.3.0.10-r2, 9.3.0.11-r1, 9.3.0.11-r2, 9.3.0.15-r1, 9.3.0.16-r1, 9.3.0.16-r2, 9.3.0.17-r1, 9.3.0.17-r2, 9.3.0.17-r3, 9.3.0.20-r1, 9.3.0.20-r2, 9.3.0.21-r1, 9.3.1.0-r1, 9.3.1.0-r2, 9.3.1.0-r3, 9.3.1.1-r1, 9.3.2.0-r1, 9.3.2.0-r2, 9.3.2.1-r1, 9.3.2.1-r2, 9.3.3.0-r1, 9.3.3.0-r2, 9.3.3.1-r1, 9.3.3.1-r2, 9.3.3.2-r1, 9.3.3.2-r2, 9.3.3.3-r1, 9.3.3.3-r2, 9.3.4.0-r1, 9.3.4.1-r1, 9.3.5.0-r1, 9.3.5.0-r2, 9.3.5.1-r1, 9.3.5.1-r2, 9.4.0.0-r1, 9.4.0.0-r2, 9.4.0.0-r3

Red Hat OpenShift Container Platform 版本
OpenShift Container Platform 4.12 及更高版本。

IBM Cloud Pak foundational services 版本
仅适用于 IBM Cloud Pak foundational services V 4.6 。

已更改的内容
本安全公告中详细介绍了已解决的漏洞：https://www.ibm.com/support/pages/node/7167732

IBM MQ Operator 3.2.3

IBM Cloud Pak for Integration 版本
IBM Cloud Pak for Integration 16.1.0

操作员通道
v3.2-sc2

.spec.version 的允许值
9.4.0.0-r3

迁移期间 .spec.version 的允许值
9.3.0.0-r1, 9.3.0.0-r2, 9.3.0.0-r3, 9.3.0.1-r1, 9.3.0.1-r2, 9.3.0.1-r3, 9.3.0.1-r4, 9.3.0.3-r1, 9.3.0.4-r1, 9.3.0.4-r2, 9.3.0.5-r1, 9.3.0.5-r2, 9.3.0.5-r3, 9.3.0.6-r1, 9.3.0.10-r1, 9.3.0.10-r2, 9.3.0.11-r1, 9.3.0.11-r2, 9.3.0.15-r1, 9.3.0.16-r1, 9.3.0.16-r2, 9.3.0.17-r1, 9.3.0.17-r2, 9.3.0.17-r3, 9.3.0.20-r1, 9.3.0.20-r2, 9.3.1.0-r1, 9.3.1.0-r2, 9.3.1.0-r3, 9.3.1.1-r1, 9.3.2.0-r1, 9.3.2.0-r2, 9.3.2.1-r1, 9.3.2.1-r2, 9.3.3.0-r1, 9.3.3.0-r2, 9.3.3.1-r1, 9.3.3.1-r2, 9.3.3.2-r1, 9.3.3.2-r2, 9.3.3.3-r1, 9.3.3.3-r2, 9.3.4.0-r1, 9.3.4.1-r1, 9.3.5.0-r1, 9.3.5.0-r2, 9.3.5.1-r1, 9.3.5.1-r2, 9.4.0.0-r1, 9.4.0.0-r2

Red Hat OpenShift Container Platform 版本
OpenShift Container Platform 4.12 及更高版本。

IBM Cloud Pak foundational services 版本
仅适用于 IBM Cloud Pak foundational services V 4.6 。

已更改的内容
这些安全公告中详细介绍了已解决的漏洞:

- https://www.ibm.com/support/pages/node/7162095
- https://www.ibm.com/support/pages/node/7162272

IBM MQ Operator 3.2.2

CD CP4I-SC2

IBM Cloud Pak for Integration 版本
IBM Cloud Pak for Integration 16.1.0

操作员通道
v3.2-sc2

.spec.version 的允许值
9.4.0.0-r2

迁移期间 .spec.version 的允许值
9.3.0.0-r1, 9.3.0.0-r2, 9.3.0.0-r3, 9.3.0.1-r1, 9.3.0.1-r2, 9.3.0.1-r3, 9.3.0.1-r4, 9.3.0.3-r1, 9.3.0.4-r1, 9.3.0.4-r2, 9.3.0.5-r1, 9.3.0.5-r2, 9.3.0.5-r3, 9.3.0.6-r1, 9.3.0.10-r1, 9.3.0.10-r2, 9.3.0.11-r1, 9.3.0.11-r2, 9.3.0.15-r1, 9.3.0.16-r1, 9.3.0.16-r2, 9.3.0.17-r1, 9.3.0.17-r2, 9.3.0.17-r3, 9.3.0.20-r1, 9.3.1.0-r1, 9.3.1.0-r2, 9.3.1.0-r3, 9.3.1.1-r1, 9.3.2.0-r1, 9.3.2.0-r2, 9.3.2.1-r1, 9.3.2.1-r2, 9.3.3.0-r1, 9.3.3.0-r2, 9.3.3.1-r1, 9.3.3.1-r2, 9.3.3.2-r1, 9.3.3.2-r2, 9.3.3.3-r1, 9.3.3.3-r2, 9.3.4.0-r1, 9.3.4.1-r1, 9.3.5.0-r1, 9.3.5.0-r2, 9.3.5.1-r1, 9.3.5.1-r2, 9.4.0.0-r1

Red Hat OpenShift Container Platform 版本
OpenShift Container Platform 4.12 及更高版本。

IBM Cloud Pak foundational services 版本
仅适用于 IBM Cloud Pak foundational services V 4.6 。

已更改的内容

- 此 安全公告中详细描述了已解决的漏洞。

IBM MQ Operator 3.2.1

CD CP4I-SC2

IBM Cloud Pak for Integration 版本
IBM Cloud Pak for Integration 16.1.0

操作员通道
v3.2-sc2

.spec.version 的允许值
9.4.0.0-r1

迁移期间 .spec.version 的允许值
9.3.0.0-r1, 9.3.0.0-r2, 9.3.0.0-r3,9.3.3.2-r3 9.3.0.1-r1, 9.3.0.1-r2, 9.3.0.1-r3, 9.3.0.1-r4, 9.3.0.3-r1, 9.3.0.4-r1, 9.3.0.4-r2, 9.3.0.5-r1, 9.3.0.5-r2, 9.3.0.5-r3, 9.3.0.6-r1, 9.3.0.10-r1, 9.3.0.10-r2, 9.3.0.11-r1, 9.3.0.11-r2, 9.3.0.15-r1, 9.3.0.16-r1, 9.3.0.16-r2, 9.3.0.17-r1, 9.3.0.17-r2, 9.3.0.17-r3, 9.3.1.0-r1, 9.3.1.0-r2, 9.3.1.0-r3, 9.3.1.1-r1, 9.3.2.0-r1, 9.3.2.0-r2, 9.3.2.1-r1, 9.3.2.1-r2, 9.3.3.0-r1, 9.3.3.0-r2, 9.3.3.1-r1, 9.3.3.1-r2, 9.3.3.2-r1, 9.3.3.2-r2, 9.3.3.2-r3, 9.3.3.3-r1, 9.3.3.3-r2, 9.3.4.0-r1, 9.3.4.1-r1, 9.3.5.0-r1, 9.3.5.0-r2, 9.3.5.1-r1, 9.3.5.1-r2

Red Hat OpenShift Container Platform 版本
OpenShift Container Platform 4.12 及更高版本。

IBM Cloud Pak foundational services 版本
仅适用于 IBM Cloud Pak foundational services V 4.6 。

已更改的内容

- 解决了 OpenShift Container Platform 4.12 上的问题，其中升级到 v3.2-sc2 通道可能会导致 IBM Cloud Pak for Integration 用户发生意外行为。 有关更多信息，请参阅 IBM Cloud Pak for Integration 文档中的 从 2023.4 。

IBM MQ Operator 3.2.0

IBM Cloud Pak for Integration 版本
IBM Cloud Pak for Integration 16.1.0

操作员通道
v3.2-sc2

.spec.version 的允许值
9.4.0.0-r1

迁移期间 .spec.version 的允许值
9.3.0.0-r1, 9.3.0.0-r2, 9.3.0.0-r3,9.3.3.2-r3 9.3.0.1-r1, 9.3.0.1-r2, 9.3.0.1-r3, 9.3.0.1-r4, 9.3.0.3-r1, 9.3.0.4-r1, 9.3.0.4-r2, 9.3.0.5-r1, 9.3.0.5-r2, 9.3.0.5-r3, 9.3.0.6-r1, 9.3.0.10-r1, 9.3.0.10-r2, 9.3.0.11-r1, 9.3.0.11-r2, 9.3.0.15-r1, 9.3.0.16-r1, 9.3.0.16-r2, 9.3.0.17-r1, 9.3.0.17-r2, 9.3.0.17-r3, 9.3.1.0-r1, 9.3.1.0-r2, 9.3.1.0-r3, 9.3.1.1-r1, 9.3.2.0-r1, 9.3.2.0-r2, 9.3.2.1-r1, 9.3.2.1-r2, 9.3.3.0-r1, 9.3.3.0-r2, 9.3.3.1-r1, 9.3.3.1-r2, 9.3.3.2-r1, 9.3.3.2-r2, 9.3.3.2-r3, 9.3.3.3-r1, 9.3.3.3-r2, 9.3.4.0-r1, 9.3.4.1-r1, 9.3.5.0-r1, 9.3.5.0-r2, 9.3.5.1-r1, 9.3.5.1-r2

Red Hat OpenShift Container Platform 版本
OpenShift Container Platform 4.12 及更高版本。

IBM Cloud Pak foundational services 版本
仅适用于 IBM Cloud Pak foundational services V 4.6 。

新增内容

- 现在支持 第 147 页的『扩展持久卷』。
- 现在，可以通过添加 `mq.ibm.com/stop` 注释并将其设置为 `true` 来停止队列管理器。 请参阅第 202 页的『停止队列管理器 (mq.ibm.com/stop)』

  注意:

  - 已停止的队列管理器在其 `StatefulSet` 中的 *.replicas* 字段设置为 0。
  - 由于 IBM MQ Operator 现在主动管理 `StatefulSet` 中的 *.replicas* 字段，因此如果修改此字段，那么操作员将立即还原此字段。
  - 如果修改 *.replicas* 字段，但仍保留修改后的值，那么较旧版本的 IBM MQ 将进入 "失败" 状态。 如果现有操作过程依赖于此行为，那么必须从 IBM MQ 9.4 使用 `mq.ibm.com/stop` 注释。

已更改的内容

- 现在支持编号为奇数的 Red Hat OpenShift Container Platform 发行版。

- IBM MQ 目录映像已从 SQLite 数据库格式移至基于文件的目录格式。
- 基于 Red Hat 通用基础图像 9.4-949.1716471857。 **注：** UBI 9 尚待 FIPS 140-3 认证。 Power 8 架构不支持 UBI 9。
- 此 安全公告中详细描述了已解决的漏洞。

OpenShift CP4I 队列管理器容器映像的发布历史

注: For information about earlier queue manager container images, see IBM MQ Operator 的发布历史 in the IBM MQ 9.3 documentation.

支持独立图像，或使用 "IBM MQ Operator

9.4.3.0-r1

CD

兼容的操作员版本
支持独立运行，或与 IBM MQ Operator 3.6.0 或更高版本。

受支持的体系结构
amd64, s390x, ppc64le

映像

- `cp.icr.io/cp/ibm-mqadvanced-server-integration:9.4.3.0-r1` (IBM Cloud Pak for Integration)
- `cp.icr.io/cp/ibm-mqadvanced-server:9.4.3.0-r1` (IBM MQ Advanced)

  注: 如果遵守相应的许可条款并对这些队列管理器进行注释，还可以使用此镜像部署 IBM MQ 许可的队列管理器。 有关更多信息，请参阅第 140 页的『使用 IBM MQ 许可证注释配置队列管理器。 IBM MQ Operator』和第 249 页的『IBM MQ 集装箱许可证注释』。
- `icr.io/ibm-messaging/mq:9.4.3.0-r1` (IBM MQ Advanced for Developers)

新增内容

- 现在，您可以启用访问 Prometheus 指标端点的审计日志。 有关更多信息，请参阅第 131 页的『在容器中为 IBM MQ 配置审计日志』。
- 现在，您可以使用 IBM MQ 许可证注释配置通过 IBM MQ Operator 部署的队列管理器。 有关更多信息，请参阅第 140 页的『使用 IBM MQ 许可证注释配置队列管理器。 IBM MQ Operator』。 请注意，此更新适用于所有 IBM MQ 9.4 队列管理器
- 通过添加新的 IBM MQ Native HA 和 Cross-Region Replication Add-on 许可证，您现在可以在 IBM MQ 许可的队列管理器上使用 Native HA 和 Cross-Region Replication 功能。 有关更多信息，请参阅第 101 页的『在 IBM MQ 许可的队列管理器上配置本地 HA 和跨区域复制，方法是使用 IBM MQ Operator』或第 163 页的『在 IBM MQ 许可的队列管理器上配置本机 HA 和跨区域复制。 Kubernetes』。 请注意，此更新也适用于 IBM MQ 9.4.2 队列管理器。

已更改的内容

- 发生了什么变化 IBM MQ 9.4.3。
- 本安全公告详细介绍了已解决的漏洞： https://www.ibm.com/support/pages/node/7236608

9.4.2.1-r2

CD

兼容的操作员版本
支持独立运行，或与 IBM MQ Operator 3.5.3 或更高版本。

受支持的体系结构
amd64, s390x, ppc64le

映像

- `cp.icr.io/cp/ibm-mqadvanced-server-integration:9.4.2.1-r2` (IBM Cloud Pak for Integration)
- `cp.icr.io/cp/ibm-mqadvanced-server:9.4.2.1-r2` (IBM MQ Advanced)
- `icr.io/ibm-messaging/mq:9.4.2.1-r2` (IBM MQ Advanced for Developers)

新增内容

- 多平台 "IBM MQ 9.4.2 的新功能--基本和高级权限.

已更改的内容

- 发生了什么变化 IBM MQ 9.4.2。
- 本安全公告详细介绍了已解决的漏洞： https://www.ibm.com/support/pages/node/7234178

9.4.2.1-r1

CD

兼容的操作员版本
支持独立运行，或与 IBM MQ Operator 3.5.2 或更高版本。

受支持的体系结构
amd64, s390x, ppc64le

映像

- `cp.icr.io/cp/ibm-mqadvanced-server-integration:9.4.2.1-r1` (IBM Cloud Pak for Integration)
- `cp.icr.io/cp/ibm-mqadvanced-server:9.4.2.1-r1` (IBM MQ Advanced)
- `icr.io/ibm-messaging/mq:9.4.2.1-r1` (IBM MQ Advanced for Developers)

新增内容

- 多平台 "IBM MQ 9.4.2 的新功能--基本和高级权限.

已更改的内容

- 发生了什么变化 IBM MQ 9.4.2。
- 本安全公告详细介绍了已解决的漏洞： https://www.ibm.com/support/pages/node/7232272

9.4.2.0-r2

CD

兼容的操作员版本
支持独立运行，或与 IBM MQ Operator 3.5.1 或更高版本。

受支持的体系结构
amd64, s390x, ppc64le

映像

- `cp.icr.io/cp/ibm-mqadvanced-server-integration:9.4.2.0-r2` (IBM Cloud Pak for Integration)
- `cp.icr.io/cp/ibm-mqadvanced-server:9.4.2.0-r2` (IBM MQ Advanced)
- `icr.io/ibm-messaging/mq:9.4.2.0-r2` (IBM MQ Advanced for Developers)

新增内容

- 多平台 "IBM MQ 9.4.2 的新功能--基本和高级权限.

已更改的内容

- 发生了什么变化 IBM MQ 9.4.2。

- 本安全公告详细介绍了已解决的漏洞： https://www.ibm.com/support/pages/node/7228945

9.4.2.0-r1

CD

兼容的操作员版本
支持独立运行，或与 IBM MQ Operator 3.5.0 或更高版本。

受支持的体系结构
amd64, s390x, ppc64le

映像

- `cp.icr.io/cp/ibm-mqadvanced-server-integration:9.4.2.0-r1` (IBM Cloud Pak for Integration)
- `cp.icr.io/cp/ibm-mqadvanced-server:9.4.2.0-r1` (IBM MQ Advanced)
- `icr.io/ibm-messaging/mq:9.4.2.0-r1` (IBM MQ Advanced for Developers)

新增内容

- 队列管理器指标可通过 HTTPS 提供，具体方法是使用 TLS 密钥填充 `/etc/mqm/metrics/pki/keys`。 有关更多信息，请参阅 第 257 页的『文件系统和挂载点』。
- 多平台 "IBM MQ 9.4.2 的新功能--基本和高级权限.

已更改的内容

- 发生了什么变化 IBM MQ 9.4.2。
- 本安全公告详细介绍了已解决的漏洞： https://www.ibm.com/support/pages/node/7184453

9.4.1.1-r1

CD

兼容的操作员版本
支持单机版，或 IBM MQ Operator 3.4.1 或更高版本

受支持的体系结构
amd64, s390x, ppc64le

映像

- `cp.icr.io/cp/ibm-mqadvanced-server-integration:9.4.1.1-r1` (IBM Cloud Pak for Integration)
- `cp.icr.io/cp/ibm-mqadvanced-server:9.4.1.1-r1` (IBM MQ Advanced)
- `icr.io/ibm-messaging/mq:9.4.1.1-r1` (IBM MQ Advanced for Developers)

新增内容

- 多平台 "IBM MQ 9.4.1 的新功能--基本和高级权限.

已更改的内容

- 发生了什么变化 IBM MQ 9.4.1。
- 本安全公告详细介绍了已解决的漏洞： https://www.ibm.com/support/pages/node/7182196

9.4.1.0-r2

CD

兼容的操作员版本
支持独立运行，或与 "IBM MQ Operator33.4.0 或更高版本一起运行

受支持的体系结构
amd64, s390x, ppc64le

映像

- `cp.icr.io/cp/ibm-mqadvanced-server-integration:9.4.1.0-r2` (IBM Cloud Pak for Integration)
- `cp.icr.io/cp/ibm-mqadvanced-server:9.4.1.0-r2` (IBM MQ Advanced)
- `icr.io/ibm-messaging/mq:9.4.1.0-r2` (IBM MQ Advanced for Developers)

新增内容

- 多平台 "IBM MQ 9.4.1 的新功能--基本和高级权限.

已更改的内容

- 发生了什么变化 IBM MQ 9.4.1。
- 本安全公告详细介绍了已解决的漏洞： https://www.ibm.com/support/pages/node/7178065

9.4.1.0-r1

兼容的操作员版本
支持独立运行，或与 "IBM MQ Operator33.3.0 或更高版本一起运行

受支持的体系结构
amd64, s390x, ppc64le

映像

- `cp.icr.io/cp/ibm-mqadvanced-server-integration:9.4.1.0-r1` (IBM Cloud Pak for Integration)
- `cp.icr.io/cp/ibm-mqadvanced-server:9.4.1.0-r1` (IBM MQ Advanced)
- `icr.io/ibm-messaging/mq:9.4.1.0-r1` (IBM MQ Advanced for Developers)

新增内容

- 现在可为 9.4.1.0-r1 及更高版本的队列管理器配置 "IBM MQ Advanced for Non-Production Environment 许可证。 有关许可证信息，请参见 "第 219 页的『IBM MQ Operator 的许可和 API 参考资料』。
- IBM MQ 9.4.1 for Multiplatforms-基本权利和高级权利中的新增内容

已更改的内容

- APARIT46430 的修复。
- 修复了处理中间证书时出现的问题。
- 修正了当 FIPS 未启用时，在 **runmqakm** 密钥存储生成时禁用 FIPS 模式的问题。
- 对文件路径进行消毒的修复。
- 更新以缩小图像大小。
- IBM MQ 队列管理器容器映像 9.4.1.0-r1 包含 "IBM MQ 跟踪用户退出的 3.1.112024.3.0) 版本。 See IBM MQ 跟踪 in the IBM Instana documentation.
- 允许从外部提供本地 HA 配置，而不是从模板生成。
  - 不再使用本地 HA 的环境变量配置（仍需使用的 "`MQ_NATIVE_HA=true` 除外）。
  - 澄清 "IBM Documentation 中现已废弃的环境变量配置的行为。
- 发生了什么变化 IBM MQ 9.4.1。
- 本安全公告详细介绍了已解决的漏洞： https://www.ibm.com/support/pages/node/7174358

仅支持与 "IBM MQ Operator 一起使用的图像

9.4.0.11-r3

CP4I-SC2

必需的操作程序版本

第 10 页的『IBM MQ Operator 3.2.13』 或更高

受支持的体系结构

amd64, s390x, ppc64le

映像

- `cp.icr.io/cp/ibm-mqadvanced-server-integration:9.4.0.11-r3` (IBM Cloud Pak for Integration)
- `cp.icr.io/cp/ibm-mqadvanced-server:9.4.0.11-r3` (IBM MQ Advanced)

  注: 如果遵守相应的许可条款并对这些队列管理器进行注释，还可以使用此镜像部署 IBM MQ 许可的队列管理器。 有关更多信息，请参阅第 140 页的『使用 IBM MQ 许可证注释配置队列管理器。 IBM MQ Operator』和第 249 页的『IBM MQ 集装箱许可证注释』。
- `icr.io/ibm-messaging/mq:9.4.0.11-r3` (IBM MQ Advanced for Developers)

新增内容

- 现在，您可以使用 IBM MQ 许可证注释配置通过 IBM MQ Operator 部署的队列管理器。 有关更多信息，请参阅第 140 页的『使用 IBM MQ 许可证注释配置队列管理器。 IBM MQ Operator』。 请注意，此更新适用于所有 IBM MQ 9.4 队列管理器

已更改的内容

- 发生了什么变化 IBM MQ 9.4.0。
- 本安全公告详细介绍了已解决的漏洞： https://www.ibm.com/support/pages/node/7236608

9.4.0.11-r2

CP4I-SC2

必需的操作程序版本

第 10 页的『IBM MQ Operator 3.2.12』 或更高

受支持的体系结构

amd64, s390x, ppc64le

映像

- `cp.icr.io/cp/ibm-mqadvanced-server-integration:9.4.0.11-r2` (IBM Cloud Pak for Integration)
- `cp.icr.io/cp/ibm-mqadvanced-server:9.4.0.11-r2` (IBM MQ Advanced)
- `icr.io/ibm-messaging/mq:9.4.0.11-r2` (IBM MQ Advanced for Developers)

新增内容

- IBM MQ 9.4.0 for Multiplatforms-基本权利和高级权利中的新增内容

已更改的内容

- 发生了什么变化 IBM MQ 9.4.0。
- 本安全公告详细介绍了已解决的漏洞： https://www.ibm.com/support/pages/node/7234178

9.4.0.11-r1

CP4I-SC2

必需的操作程序版本

3.2.11 或更高

受支持的体系结构

amd64, s390x, ppc64le

映像

- `cp.icr.io/cp/ibm-mqadvanced-server-integration:9.4.0.11-r1` (IBM Cloud Pak for Integration)
- `cp.icr.io/cp/ibm-mqadvanced-server:9.4.0.11-r1` (IBM MQ Advanced)
- `icr.io/ibm-messaging/mq:9.4.0.11-r1` (IBM MQ Advanced for Developers)

新增内容

- IBM MQ 9.4.0 for Multiplatforms-基本权利和高级权利中的新增内容

已更改的内容

- 发生了什么变化 IBM MQ 9.4.0。

9.4.0.10-r2

CP4I-SC2

必需的操作程序版本
3.2.10 或更高

受支持的体系结构
amd64, s390x, ppc64le

映像

- `cp.icr.io/cp/ibm-mqadvanced-server-integration:9.4.0.10-r2` (IBM Cloud Pak for Integration)
- `cp.icr.io/cp/ibm-mqadvanced-server:9.4.0.10-r2` (IBM MQ Advanced)
- `icr.io/ibm-messaging/mq:9.4.0.10-r2` (IBM MQ Advanced for Developers)

新增内容

- IBM MQ 9.4.0 for Multiplatforms-基本权利和高级权利中的新增内容

已更改的内容

- 发生了什么变化 IBM MQ 9.4.0。

9.4.0.10-r1

CP4I-SC2

必需的操作程序版本
3.2.9 或更高

受支持的体系结构
amd64, s390x, ppc64le

映像

- `cp.icr.io/cp/ibm-mqadvanced-server-integration:9.4.0.10-r1` (IBM Cloud Pak for Integration)
- `cp.icr.io/cp/ibm-mqadvanced-server:9.4.0.10-r1` (IBM MQ Advanced)
- `icr.io/ibm-messaging/mq:9.4.0.10-r1` (IBM MQ Advanced for Developers)

新增内容

- IBM MQ 9.4.0 for Multiplatforms-基本权利和高级权利中的新增内容

已更改的内容

- 发生了什么变化 IBM MQ 9.4.0。

9.4.0.7-r1

CP4I-SC2

必需的操作程序版本
3.2.8 或更高

受支持的体系结构
amd64, s390x, ppc64le

映像

- `cp.icr.io/cp/ibm-mqadvanced-server-integration:9.4.0.7-r1` (IBM Cloud Pak for Integration)
- `cp.icr.io/cp/ibm-mqadvanced-server:9.4.0.7-r1` (IBM MQ Advanced)
- `icr.io/ibm-messaging/mq:9.4.0.7-r1` (IBM MQ Advanced for Developers)

新增内容

- IBM MQ 9.4.0 for Multiplatforms-基本权利和高级权利中的新增内容

已更改的内容

- 发生了什么变化 IBM MQ 9.4.0。

9.4.0.6-r2

CP4I-SC2

必需的操作程序版本
3.3.2.7 或更高

受支持的体系结构
amd64, s390x, ppc64le

映像

- `cp.icr.io/cp/ibm-mqadvanced-server-integration:9.4.0.6-r2` (IBM Cloud Pak for Integration)
- `cp.icr.io/cp/ibm-mqadvanced-server:9.4.0.6-r2` (IBM MQ Advanced)
- `icr.io/ibm-messaging/mq:9.4.0.6-r2` (IBM MQ Advanced for Developers)

新增内容

- IBM MQ 9.4.0 for Multiplatforms-基本权利和高级权利中的新增内容

已更改的内容

- 发生了什么变化 IBM MQ 9.4.0。

9.4.0.6-r1

CP4I-SC2

必需的操作程序版本
3.3.2.6 或更高

受支持的体系结构
amd64, s390x, ppc64le

映像

- `cp.icr.io/cp/ibm-mqadvanced-server-integration:9.4.0.6-r1` (IBM Cloud Pak for Integration)
- `cp.icr.io/cp/ibm-mqadvanced-server:9.4.0.6-r1` (IBM MQ Advanced)
- `icr.io/ibm-messaging/mq:9.4.0.6-r1` (IBM MQ Advanced for Developers)

新增内容

- IBM MQ 9.4.0 for Multiplatforms-基本权利和高级权利中的新增内容

已更改的内容

- 发生了什么变化 IBM MQ 9.4.0。
- 已应用修复程序，以解决在使用 Rosetta 的 Mac 上运行 "**endmqm** 命令时容器挂起的问题。

9.4.0.5-r2

CD CP4I-SC2

必需的操作程序版本
3.2.5 或更高版本

受支持的体系结构
amd64, s390x, ppc64le

映像

- `cp.icr.io/cp/ibm-mqadvanced-server-integration:9.4.0.5-r2` (IBM Cloud Pak for Integration)
- `cp.icr.io/cp/ibm-mqadvanced-server:9.4.0.5-r2` (IBM MQ Advanced)
- `icr.io/ibm-messaging/mq:9.4.0.5-r2` (IBM MQ Advanced for Developers)

新增内容

- IBM MQ 9.4.0 for Multiplatforms-基本权利和高级权利中的新增内容

已更改的内容

- 发生了什么变化 IBM MQ 9.4.0。
- 基于 Red Hat 通用基础镜像 9.4-1227.1725849298。

9.4.0.5-r1

CD CP4I-SC2

必需的操作程序版本
3.2.4 或更高版本

受支持的体系结构
amd64, s390x, ppc64le

映像

- `cp.icr.io/cp/ibm-mqadvanced-server-integration:9.4.0.5-r1` (IBM Cloud Pak for Integration)
- `cp.icr.io/cp/ibm-mqadvanced-server:9.4.0.5-r1` (IBM MQ Advanced)
- `icr.io/ibm-messaging/mq:9.4.0.5-r1` (IBM MQ Advanced for Developers)

新增内容

- IBM MQ 9.4.0 for Multiplatforms-基本权利和高级权利中的新增内容

已更改的内容

- 发生了什么变化 IBM MQ 9.4.0。
- 基于 Red Hat 通用基础镜像 9.4-1194。

9.4.0.0-r3

CD CP4I-SC2

必需的操作程序版本
3.2.3 或更高版本

受支持的体系结构
amd64, s390x, ppc64le

映像

- `cp.icr.io/cp/ibm-mqadvanced-server-integration:9.4.0.0-r3` (IBM Cloud Pak for Integration)
- `cp.icr.io/cp/ibm-mqadvanced-server:9.4.0.0-r3` (IBM MQ Advanced)
- `icr.io/ibm-messaging/mq:9.4.0.0-r3` (IBM MQ Advanced for Developers)

新增内容

- IBM MQ 9.4.0 for Multiplatforms-基本权利和高级权利中的新增内容

已更改的内容

- 发生了什么变化 IBM MQ 9.4.0。
- 基于 Red Hat 通用基础镜像 9.4-1134 。
- 解决了仅在 OpenShift Container Platform (OCP) 4.16 上出现的一个问题，在该问题中，调和 `ServiceAccount` 会创建一个无限循环的图像拉取秘密。 OCP 4.16 引入了一个行为变化，即 OCP 将 `.metadata.annotation` 注入 `ServiceAccount` 中。

9.4.0.0-r2

CD CP4I-SC2

必需的操作程序版本
3.2.2 或更高

受支持的体系结构
amd64, s390x, ppc64le

映像

- `cp.icr.io/cp/ibm-mqadvanced-server-integration:9.4.0.0-r2` (IBM Cloud Pak for Integration)
- `cp.icr.io/cp/ibm-mqadvanced-server:9.4.0.0-r2` (IBM MQ Advanced)
- `icr.io/ibm-messaging/mq:9.4.0.0-r2` (IBM MQ Advanced for Developers)

新增内容

- IBM MQ 9.4.0 for Multiplatforms-基本权利和高级权利中的新增内容

已更改的内容

- 发生了什么变化 IBM MQ 9.4.0。
- 基于 Red Hat 通用基础镜像 9.4-1134 。

9.4.0.0-r1

CD CP4I-SC2

必需的操作程序版本
3.2.0 或更高版本

受支持的体系结构
amd64, s390x, ppc64le

映像

- `cp.icr.io/cp/ibm-mqadvanced-server-integration:9.4.0.0-r1` (IBM Cloud Pak for Integration)
- `cp.icr.io/cp/ibm-mqadvanced-server:9.4.0.0-r1` (IBM MQ Advanced)

- `icr.io/ibm-messaging/mq:9.4.0.0-r1` (IBM MQ Advanced for Developers)

新增内容

- IBM MQ 9.4.0 for Multiplatforms-基本权利和高级权利中的新增内容

已更改的内容

- IBM MQ 9.4.0 中更改的内容
- Deprecated 使用 IBM MQ Advanced for Developers 时，不推荐通过环境变量为 `admin` 和 `app` 用户设置密码。 请改为使用私钥。
- 为环境变量 *MQ_LOGGING_CONSOLE_SOURCE* 添加了新的可选值 `mqsc` 。 此选项可用于反映容器日志中 `autocfgmqsc.LOG` 的内容。
- 基于 Red Hat 通用基础图像 9.4-949.1716471857。 **注：** UBI 9 尚待 FIPS 140-3 认证。 Power 8 架构不支持 UBI 9。

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 OpenShift Container Platform 体系结构)。

过程

- 第 26 页的『选择要在容器中使用 IBM MQ 的方式』.
- 第 27 页的『支持容器中的 IBM MQ』.
- 第 35 页的『规划 IBM MQ Operator 的存储器』.
- 第 36 页的『规划容器中 IBM MQ 的高可用性』.
- 第 41 页的『容器中 IBM MQ 的灾难恢复』.
- 第 44 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

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 扩展 Red Hat OpenShift Container Platform API 以添加新的 `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 40 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。 有关使用 IBM MQ Operator 时支持的功能，请参阅 第 27 页的『支持 IBM MQ Operator』。

使用预置的 IBM MQ Advanced 图像

V 9.4.1

您可以使用 IBM Container Registry 中的预建图像。 请参阅第 28 页的『支持 IBM MQ Advanced 容器图像』以了解详细信息。

使用预构建的 IBM MQ 许可图像

从 IBM MQ 9.4.1 ，您可以使用 IBM Container Registry 中提供的预置图像。 有关更多信息，请参阅第 29 页的『支持 IBM MQ 许可的容器映像』。

构建您自己的映像和部署代码

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。

提供了用于构建您自己的映像的样本。 请参阅 第 76 页的『通过构建您自己的容器映像来准备 IBM MQ』。

请参阅 第 30 页的『支持构建自己的 IBM MQ 容器图像』，了解在构建自己的映像和部署代码时支持的详细信息。

相关参考

第 27 页的『支持容器中的 IBM MQ』
并非所有 IBM MQ 功能在容器中都可用且受支持的方式相同。

支持容器中的 IBM MQ

并非所有 IBM MQ 功能在容器中都可用且受支持的方式相同。

相关概念

适用于长期支持和持续交付发行版的 IBM MQ 常见问题及回答

相关参考

IBM Cloud Pak for Integration 软件支持生命周期附录

相关信息

IBM MQ 支持 SELinux 和 AppArmor

支持 IBM MQ Operator

作为 IBM MQ Advanced 和 IBM Cloud Pak for Integration 的一部分，支持 IBM MQ Operator

受支持的平台

IBM MQ Operator 仅在 Red Hat OpenShift Container Platform 上支持。 一旦 Red Hat 停止支持， IBM MQ 将不再支持 Red Hat OpenShift Container Platform 的发布。 有关详细信息，请参阅 第 32 页的『IBM MQ Operator 的版本支持』 和 Red Hat OpenShift Container Platform 生命周期日期

支持的 CPU 架构

IBM MQ Operator 支持 amd64 和 s390x z/Linux 以及 ppc64le Power Systems 版本 9 及以上。 请注意，Red Hat OpenShift Container Platform 集群中的所有节点必须使用相同的 CPU 架构。

支助期限

IBM 为 IBM MQ Operator 和相关队列管理器映像提供缺陷和使用支持。 支持期限取决于您使用的许可证：

- 如果您使用的是 **IBM Cloud Pak for Integration** 许可证，那么 IBM MQ Operator 的支持期限与相应 IBM Cloud Pak for Integration 版本的支持期限相同。 有些版本是 IBM Cloud Pak for Integration - Support Cycle 2 (formerly Long Term Support)，有些是 Continuous Delivery （CD）。 在下一个 IBM Cloud Pak for Integration CD 或 CP4I-LTS 版本发布前，将支持 CD 操作符和队列管理器 使用 IBM Cloud Pak for Integration 许可证部署的 CP4I-LTS 操作员和队列管理器将一直支持到下一个 CP4I-LTS 版本，另外还有一个宽限期允许升级。

例如，如果使用 IBM Cloud Pak for Integration 许可证 “L-JTPV-KYG8TF” 创建队列管理器，则该队列管理器与 CP4I-LTS 版本相关联。 这将部署 IBM MQ Advanced 9.4.0，并与该 IBM Cloud Pak for Integration 版本（两年）保持一致。

- 如果使用 **IBM MQ Advanced** 或 **IBM MQ** 许可证，则只支持 IBM MQ Continuous Delivery 数据流。 使用 IBM MQ Advanced 或 IBM MQ 许可证部署的每个 IBM MQ Operator 和队列管理器，自第一个支持该队列管理器版本的运营商起，支持期为一年。 在下一个 Continuous Delivery 版本发布之前，各版本会主动接受安全修复。 例如，如果您使用 IBM MQ Advanced 许可证 “L-EHXT-MQCRN9” 创建队列管理器，它将使用 IBM MQ Advanced 9.4.0 作为 Continuous Delivery 的发布版本，并将获得为期一年的支持，在 IBM MQ 9.4.1 发布之前，将发布安全修复程序。 在 9.4.1 版本发布后， IBM 将根据要求评估是否可以做出合理努力，在上一版 CD 上提供修复功能

有关详细信息，请参阅 IBM MQ 有关长期支持和持续交付版本的常见问题解答。

修复可用性

IBM Container Registry 提供定期和临时队列管理器修复。 不提供 IBM MQ Operator 的临时修复。

无法使用的功能

IBM MQ Advanced 的以下功能不适用于由 IBM MQ Operator 操作的队列管理器：

- 不提供复制数据队列管理器 (RDQM)--该功能与 Linux 内核紧密耦合，因此容器中不支持该功能
- 队列管理器的线性日志记录不可用
- 无法为 **crtmqdir**、**crtmqm**、**strmqm** 和 **endmqm** 指定自定义命令行选项 - 大多数选项可使用 INI 文件进行配置，但有些选项无法配置，如使用线性日志记录。
- 操作系统 (OS) 用户不可用
- 未安装 AMQP
- 未安装 MQTT
- 托管文件传输代理不可用。 但是，您可以使用 IBM MQ 操作员提供一个或多个协调、命令或代理队列管理器。
- 不提供使用单独密钥存储区拦截 MCA 的高级信息安全功能。 MCA 拦截仍可配置为使用主队列管理器密钥存储。

您只能使用 IBM 提供的队列管理器容器映像和 MQ 操作器，而不能使用自定义代码构建自己的映像。 例外情况是在 IBM 提供的映像中添加容器映像层，以便添加配置文件或退出。

相关概念

适用于长期支持和持续交付发行版的 IBM MQ 常见问题及回答

相关参考

IBM Cloud Pak for Integration 软件支持生命周期附录

相关信息

容器中 IBM MQ 的生命周期日期

CD MQ Adv. Linux V 9.4.1 支持 IBM MQ Advanced 容器图像

从 IBM MQ Advanced 版本 9.4.1 开始，您可以使用预构建的 IBM MQ Advanced 容器映像，而无需 IBM MQ Operator

预构建的 IBM MQ Advanced 容器图像由多个逻辑层组成：

- IBM MQ 容器启用代码（`runmqserver`, `chkmqhealthy`, `chkmqready` 和 `chkmqstarted`）
- IBM MQ Advanced 自身
- Red Hat 来自 Red Hat Universal Base Image 的软件包

受支持的平台

预构建的 IBM MQ Advanced 映像可部署到 containerd、cri-o、Podman 或 Docker 提供的容器运行时环境中。 有关详细信息，请参阅 IBM MQ 的系统要求。 IBM 为 IBM MQ 容器启用代码和 "IBM MQ Advanced 本身提供缺陷和使用支持，但不能帮助用户配置和调试容器环境或部署管道中的问题。

要点: Red Hat 仅为运行在 "Red Hat Enterprise Linux®或 "Red Hat Enterprise Linux CoreOS 主机上的容器提供对 "Red Hat 软件包的支持。 更多信息，请参阅 Red HatContainer Support Policy。

许可

如果在 Kubernetes 上使用 IBM License Service，预构建的 IBM MQ Advanced 映像有资格获得容器许可。 如果您没有使用 Kubernetes 协调（例如，使用 Podman 或 Docker），则无法利用此优化许可模式。 有关详细信息，请参阅 容器许可常见问题解答。

支持的 CPU 架构

amd64 和 s390x z/Linux 以及 ppc64le Power Systems 版本 9 及以上支持预构建的 IBM MQ Advanced 映像。 请注意，属于同一队列管理器的所有容器（例如本地 HA 复制）必须使用相同的 CPU 架构。

支助期限

仅支持 IBM MQ Advanced 持续交付流 每个队列管理器版本根据正常支持期限（一年）提供支持。

在下一个 Continuous Delivery 版本发布之前，各版本会主动接收安全修复。 有关详细信息，请参阅 IBM MQ 有关长期支持和持续交付版本的常见问题解答。

修复可用性

IBM Container Registry 可提供定期和临时修复。

无法使用的功能

预构建的容器映像不提供 IBM MQ Advanced 的以下功能：

- 不可用复制数据队列管理器 (RDQM)--该功能与 Linux 内核紧密耦合，因此在容器中不受支持
- 队列管理器的线性日志记录不可用
- 无法为 **crtmqdir**、**crtmqm**、**strmqm** 和 **endmqm** 指定自定义命令行选项 - 大多数选项可使用 INI 文件进行配置，但有些选项无法配置，如使用线性日志记录。
- 操作系统 (OS) 用户不可用
- 未安装 AMQP
- 未安装 MQTT
- 托管文件传输代理不可用。 不过，您可以使用预构建的容器映像来提供一个或多个协调、命令或代理队列管理器。

如果想使用这些功能，可以构建自己的容器镜像。

相关概念

适用于长期支持和持续交付发行版的 IBM MQ 常见问题及回答

Red Hat 容器支持政策

相关信息

容器中 IBM MQ 的生命周期日期

V 9.4.1 支持 IBM MQ 许可的容器映像

从 IBM MQ 9.4.1 开始，您可以使用预构建的 IBM MQ 授权容器映像，而无需 IBM MQ Operator。

预构建的 IBM MQ 容器映像由多个逻辑层组成：

- IBM MQ 集装箱启用代码 (,, 和)`runmqserver` `chkmqhealthy` `chkmqready` `chkmqstarted`
- IBM MQ 自身

- Red Hat 软件包 Red Hat Universal Base Image

注: IBM MQ 许可的容器映像使用 IBM MQ Advanced 映像，但如果没有相应的许可，则不能使用 IBM MQ Advanced 特有的功能。 有关详细信息，请参阅本主题后面的 第 30 页的『来自 IBM MQ Advanced』 部分。

受支持的平台

预构建的 IBM MQ 映像可部署到 containerd、cri-o、 Podman 或 Docker 提供的容器运行时环境中。 有关详细信息，请参阅 IBM MQ 的系统要求。 IBM 为 -supplied 容器启用代码和 本身提供缺陷和使用支持，但不能帮助用户配置和调试容器环境或部署管道中的问题。 IBM IBM MQ IBM MQ

要点: Red Hat 仅为运行在 或 主机上的容器提供对 软件包的支持。 Red Hat Enterprise Linux Red Hat Enterprise Linux CoreOS Red Hat 更多信息，请参阅 Red Hat 容器支持政策。

许可

预构建的 IBM MQ 映像如果与 IBM License Service 一起在 Kubernetes 上使用，则有资格获得容器许可。 如果不采用 Kubernetes 协调（例如，使用 Podman 或 Docker ），则无法利用此优化许可模式。 更多详情，请参阅 " 集装箱许可常见问题"。

支持的 CPU 架构

amd64 和 s390x z/ Linux 以及 ppc64le Power Systems 版本 9 及以上支持预编译的 IBM MQ 映像。 请注意，属于同一队列管理器（例如本地 HA 复制）的所有容器必须使用相同的 CPU 架构。

支助期限

仅支持 IBM MQ 持续交付流。 每个队列管理器版本根据正常支持期限（一年）提供支持。 在下一个 Continuous Delivery 版本发布之前，各版本会主动接受安全修复。 有关详细信息，请参阅 IBM MQ 长期支持和持续交付版本常见问题。

修复可用性

定期和临时修复程序可从 IBM Container Registry 获取。

来自 IBM MQ Advanced

IBM MQ Advanced 的以下功能不能与预构建的 IBM MQ 许可容器映像一起使用：

- 本地 HA 和跨区域复制。 要在 IBM MQ 许可的队列管理器上使用本机 HA 和跨区域复制功能，请参阅 第 101 页的『在 IBM MQ 许可的队列管理器上配置本地 HA 和跨区域复制，方法是使用 IBM MQ Operator』。
- AMS
- Managed File Transfer 支持 类型的 SFTP 服务器私钥 OpenSSH
- 为本地 HA 队列管理器自动恢复受损对象
- 媒体图像调度--复制日志
- RDQM 可在 Red Hat Enterprise Linux 9 上查阅
- Kafka 连接

如果您想使用这些功能，可以部署 IBM MQ Advanced.NET Framework。 有关详细信息，请参阅支持 IBM MQ Advanced 容器映像。

Linux 支持构建自己的 IBM MQ 容器图像

IBM MQ 容器示例为在容器中启用 "IBM MQ 提供了不支持的封装示例代码。 这是最灵活的容器解决方案，但它要求你具备强大的容器配置技能，并“拥有”由此产生的容器。

下载

这里有 "IBM MQ 容器示例： https://github.com/ibm-messaging/mq-container.

注: IBM 不支持容器示例以及您添加或修改的任何代码。

要与容器示例一起使用，您需要下载一个 "IBM MQ 非安装镜像。

- 如果您有 "IBM MQ Advanced for Developers 许可证，请使用 "Continuous Delivery（CD）版本的 "IBM MQ。 这些非安装映像可从下载 IBM MQ 9.4 光盘中获取。
- 如果您拥有生产许可证，请使用 "IBM MQ 的 "Long Term Support（LTS）或 "CD 版本。 这些非安装镜像可从 Fix Central 获取。

受支持的平台

有关支持的容器运行时环境和 Linux 发行版的详细信息，请参阅 IBM MQ 的系统要求 。 IBM MQ 可与 containerd、cri-o、Podman 或 "Docker 提供的容器运行时环境一起使用。 IBM 为核心软件 "IBM MQ 提供缺陷和使用支持，但不能帮助您配置和调试容器环境或部署管道中的问题。

容器示例使用 Red Hat Universal Base Image 作为基本图像。 Red Hat 仅为运行在 Red Hat Enterprise Linux 或 Red Hat Enterprise Linux CoreOS 主机上的容器提供 Red Hat 软件包支持。 更多信息，请参阅 Red HatContainer Support Policy。 您可以在任何与 "IBM MQ 支持的 "Linux 发行版相匹配的 "Linux 基础镜像上构建自己的容器镜像：

- Red Hat Universal Base Image
- Ubuntu
- SUSE Linux Enterprise Server 基本容器图像

支持的 CPU 架构

请参阅 " IBM MQ 的系统要求。

IBM MQ 的支持期限

支持生产版本的 "IBM MQ 非安装镜像。 对于 "LTS，支持期限为 "全面可用 "后 5 年。 对于 "CD，每个增量版本自全面可用起支持 1 年。 有关详细信息，请参阅长期支持和持续交付版本的 IBM MQFAQ。

修复 "IBM MQ 的可用性

对于生产版本的 "IBM MQ 非安装镜像，将在整个支持期间提供修复。 您还可以申请修复。 如果 "CD 更新不再符合缺陷修复条件，请更新到符合条件的版本，如果缺陷可以重现，将提供修复。 有关可用的 "LTS 和 "CD 版本及最新修复的信息，请参阅按版本分列的 IBM MQ 修复。

无法使用的功能

您不能使用 "IBM MQ 非安装映像的操作系统 (OS) 用户。 如果要使用操作系统用户，则需要使用 RPM 安装程序构建自己的镜像，目前还没有这方面的示例。

有关支持组件的完整详细信息，请参阅 " IBM MQ 的系统要求。 有一些样本，但不是每个组件都有。

相关概念

适用于长期支持和持续交付发行版的 IBM MQ 常见问题及回答

相关任务

第 76 页的『通过构建您自己的容器映像来准备 IBM MQ』

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。

MQ Adv. Linux IBM MQ Advanced for Developers 容器图像的可用性

预构建的 IBM MQ Advanced for Developers 图像可从 IBM Container Registry 获取：`icr.io/ibm-messaging/mq`. 该图片是免费下载的，不需要任何理由。 **不支持该图像**。

可用平台

预构建的 IBM MQ Advanced for Developers 映像可部署到 containerd、cri-o、Podman 或 Docker 提供的容器运行时环境中。 有关详细信息，请参阅 IBM MQ 的系统要求。

可用的 CPU 架构

预构建的 IBM MQ Advanced for Developers 映像可用于 `amd64` 和 `s390x` z/Linux 以及 `ppc64le` Power Systems 版本 9 及以上。 请注意，属于同一队列管理器的所有容器（例如本地 HA 复制）必须使用相同的 CPU 架构。

无法使用的功能

预构建的容器映像不提供 IBM MQ Advanced for Developers 的以下功能：

- 不可用复制数据队列管理器 (RDQM)--该功能与 Linux 内核紧密耦合，因此在容器中不受支持
- 队列管理器的线性日志记录不可用
- 无法为 **crtmqdir**、**crtmqm**、**strmqm** 和 **endmqm** 指定自定义命令行选项 - 大多数选项可使用 INI 文件进行配置，但有些选项无法配置，如使用线性日志记录。
- 操作系统 (OS) 用户不可用
- 未安装 AMQP
- 未安装 MQTT
- 托管文件传输代理不可用。 不过，您可以使用预构建的容器映像来提供一个或多个协调、命令或代理队列管理器。

相关概念

适用于长期支持和持续交付发行版的 IBM MQ 常见问题及回答

第 258 页的『IBM MQ Advanced for Developers 容器映像 (container image)』

预构建的容器映像可用于 IBM MQ Advanced for Developers。 此映像可从 IBM Container Registry 获取。此镜像适用于 Docker、Podman、Kubernetes 和其他容器环境。

OpenShift CD CP4I CP4I-SC2 IBM MQ Operator 的版本支持

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

- 第 32 页的『可用的 IBM MQ 版本』
- 第 33 页的『兼容的 Red Hat OpenShift Container Platform 版本』
- 第 34 页的『IBM Cloud Pak for Integration 版本』
- 第 34 页的『较旧的操作程序中的可用 IBM MQ 版本』
- 第 34 页的『适用于较旧的操作程序的兼容 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									
		9.4.3	9.4.2	9.4.1	9.4.0	9.3.5	9.3.4	9.3.3	9.3.2	9.3.1	9.3.0
v3.6	3.6	CD	环境保护部	环境保护部	MIG	环境保护部	环境保护部	环境保护部	环境保护部	环境保护部	MIG
v3.5	3.5		CD	环境保护部	MIG	环境保护部	环境保护部	环境保护部	环境保护部	环境保护部	MIG

操作员通道	操作程序版本	IBM MQ 版本									
		9.4.3	9.4.2	9.4.1	9.4.0	9.3.5	9.3.4	9.3.3	9.3.2	9.3.1	9.3.0
v3.4	3.4			CD	MIG	环境保护部	环境保护部	环境保护部	环境保护部	环境保护部	MIG
v3.3	3.3			CD	MIG	环境保护部	环境保护部	环境保护部	环境保护部	环境保护部	MIG
v3.2-sc2	3.2				CD 和 SC2	环境保护部	环境保护部	环境保护部	环境保护部	环境保护部	MIG

Key:

- CD: Continuous Delivery 支持可用。
- SC2: IBM Cloud Pak for Integration - Support Cycle 2 (formerly Long Term Support) 可用。
- 迁移: 仅在从 IBM Cloud Pak for Integration - Support Cycle 2 (formerly Long Term Support) 操作数到 Continuous Delivery 操作数的迁移期间可用。
- DEP: Deprecated不推荐使用。 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移除。

请参阅 第 5 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	OpenShift Container Platform 版本 [1]					
		4.18	4.17	4.16	4.15	4.14	4.12
v3.6	3.6.0 以后	CD	CD	CD	CD	CD	CD
v3.5	3.5.1 以后	CD	CD	CD	CD	CD	CD
	3.5.0 以后		CD	CD	CD	CD	CD
v3.4	3.3.4.0 起		CD	CD	CD	CD	CD
v3.3	3.3.3.0 起		CD	CD	CD	CD	CD
v3.2-sc2	3.2.10 以后	SC2	SC2	SC2	SC2	SC2	SC2
	3.3.2.6 起		SC2	SC2	SC2	SC2	SC2
	3.2.3 开始			SC2	SC2	SC2	SC2
	3.2.0 及更高版本				SC2	SC2	SC2

Key:

- CD: Continuous Delivery 支持可用。
- SC2: IBM Cloud Pak for Integration - Support Cycle 2 (formerly Long Term Support) 可用。
- EOS: 不再受支持。 请迁移到更高的 OpenShift Container Platform 版本。

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

IBM Cloud Pak for Integration 版本

每个 IBM MQ Operator 版本都支持作为 IBM Cloud Pak for Integration 版本的一部分（如下表所示）或单独使用。

IBM Cloud Pak for Integration 版本	IBM MQ Operator 版本
16.1.2	3.6.x
16.1.1	3.4.x, 3.5.x
16.1.0	3.2.x, 3.3.x

较旧的操作程序中的可用 IBM MQ 版本

请参阅 IBM MQ 9.3 文档中的 可用 IBM MQ 版本 。

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

请参阅 IBM MQ 9.3 文档中的 兼容 OpenShift Container Platform 版本 。

OpenShift MQ Adv. 编辑 IBM MQ Operator 创建的资源

IBM MQ Operator 通过创建和管理本机 Kubernetes 资源来协调 QueueManager 定制资源。 这些受管资源 **不得** 直接编辑。

通常可以通过查看 `ownerReferences` 来确定某个资源是否由另一个更高级别的资源拥有。 例如，从 `StatefulSet` 获取的以下元数据显示它由 `QueueManager` 资源 "qm1" 拥有:

```
metadata:
  ownerReferences:
    - apiVersion: mq.ibm.com/v1beta1
      kind: QueueManager
      name: qm1
      uid: 60fda34c-9f7c-42d2-a293-78fec4315c62
      controller: true
      blockOwnerDeletion: true
```

请注意，并非所有资源都具有此元数据。

IBM MQ Operator 负责管理底层资源，例如 `StatefulSet`， `Service` 和 `Route`。 如果您更改其中任何底层资源，那么 IBM MQ Operator 将重新更改这些资源，如果该更改需要滚动更新，那么您可能会迁到停机时间。

队列管理器的大部分重要设置在 `QueueManager` 资源上可用。 但是，如果您发现需要完全控制底层资源，那么有一些选项:

- 如果需要覆盖 IBM MQ Operator 创建的 Pod 上的设置，那么可以在 `QueueManager` YAML 的 `.spec.template` 部分中添加 Pod 覆盖模板。
- 如果需要覆盖由 IBM MQ Operator 创建的队列管理器 `Route` 上的设置，那么需要禁用将 `.spec.route.enabled` 设置完全设置为 "false" 的路由，然后创建您自己的路由。
- 可以在 `QueueManager` 资源上设置诸如标签和注释之类的设置以及诸如 `security Context` 之类的 Pod 设置。
- 在其他情况下，如果需要完全控制，那么 IBM MQ Operator 可能不适用于您的用例。

规划容器中的 IBM MQ 许可

容器许可允许您仅许可单个 IBM MQ 容器的可用容量，而不是要求您许可运行容器的整个服务器。 要利用容器许可，必须使用 IBM License Service 跟踪许可使用情况并确定所需的权利。

请注意，只有在 Red Hat OpenShift Container Platform 或 Kubernetes 上才提供基于容器的许可。 如需了解更多信息，请参阅容器许可常见问题解答。

要使用 IBM License Service：

- 如果使用 IBM MQ Operator，请参阅 IBM MQ Operator 安装过程中的部署 License Service 步骤。
- 如果在 Kubernetes 上部署，请参阅 IBM Cloud Pak foundational services 文档中的在 Kubernetes 集群上安装 License Service。

如果不使用 Red Hat OpenShift Container Platform 或 Kubernetes，则必须使用 IBM License Metric Tool。需要使用该工具来计量和报告 IBM 产品的许可证使用情况，这些产品部署在传统的虚拟化环境中，以利用分容量许可。 更多信息，请参阅 IBM License Metric Tool 文档。

相关参考

第 249 页的『IBM MQ 集装箱许可证注释』
在 Kubernetes 和 Red Hat OpenShift Container Platform 环境中， IBM License Service 使用 Pod 注释，根据容器而不是底层机器上定义的限制来跟踪使用情况。 IBM MQ Operator 会为您设置这些 Pod 注释，但使用其他方式部署时，您必须自己设置这些注释。

相关信息

IBM 容器许可证
容器许可常见问题解答
安装许可证服务
查看和跟踪许可证使用情况

OpenShift CP4I Kubernetes 规划 IBM MQ Operator 的存储器

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。 其中包括:

- 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 139 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。 这是缺省行为。 因为有各种存储提供程序，每个存储提供程序都支持不同的功能，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 IBM MQ 存储配置的公共字段:

- **spec.queueManager.availability** 控制可用性方式。 如果您正在使用 SingleInstance 或 NativeHA，那么仅需要 ReadWriteOnce 存储器。 对于 multiInstance，您需要支持具有正确文件锁定特征的 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 36 页的『规划容器中 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: [65534] # Change to 99 for clusters with RHEL7 or earlier worker nodes
```

有关本机 HA 队列管理器的存储器注意事项的信息，请参阅 第 38 页的『本机 HA』。

注: 您还可以使用单实例队列管理器配置补充组。

存储容量

OpenShift CP4I

使用 IBM MQ Operator 时，应尝试确保请求的卷足以满足持续需要。 但是，如果需要增加一个或多个卷的存储容量，那么如果存储类支持卷扩展，那么可以扩展这些卷。 可以通过联机或脱机过程来扩展卷。 脱机过程需要重新启动 QueueManager Pod，而联机过程不需要重新启动。 要确定存储类是否支持卷扩展，以及卷扩展遵循的过程，请参阅存储提供程序文档。 选择存储类时，应考虑此信息。 有关卷扩展的指南，请参阅 第 147 页的『扩展持久卷』。

加密

OpenShift CP4I

IBM MQ 不会主动加密静态数据。 因此，您应该使用被动加密存储器和/或 IBM MQ Advanced Message Security 来加密消息。 在 IBM Cloud 上，块存储器和文件存储器都可使用静态被动加密。

OpenShift Kubernetes 规划容器中 IBM MQ 的高可用性

对于 IBM MQ Operator，有三种高可用性选项: **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** (这是使用共享的网络文件系统的主动/备用对) 或 **单个弹性队列管理器** (通过网络存储器为 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

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 Kubernetes 持久卷集。 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 38 页的『本机 HA』。

本地 HA CRR（跨区域复制）

您可以实施由位于不同区域中的两个本机 HA 组组成的灾难恢复解决方案。 一个本地 HA 组具有**实时**角色，而另一个组具有**恢复**角色。 当实时组所在区域发生计划内或计划外停电时，用户操作可将远程区域组的角色从恢复状态切换为实时状态。 然后，此组将接管原始队列管理器实例的工作，接受应用程序连接等。

有关更多信息，请参阅第 42 页的『本地 HA 跨区域复制』。

多实例队列管理器

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。 要配置此功能，您可以使用 "Kubernetes 持久卷，并将 "**access mode** 设置为 `ReadWriteMany`。 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

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 `IfNotPresent` 的 "**imagePullPolicy** 值，那么图像可能已经在该节点上可用。
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。 当使用像 "IBM MQ 这样的有状态工作负载和 "Kubernetes StatefulSet,时，如果 "Kubernetes 主节点与工作节点失去联系，它就无法确定该节点是发生了故障，还是仅仅失去了网络连接。 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。

注: 通过 IBM MQ Operator 创建队列管理器时，不支持更改 IBM MQ 队列管理器的 StatefulSet 详细信息 (包括副本数)。

相关概念

高可用性配置

相关任务

第 97 页的『使用 IBM MQ Operator 为队列管理器配置高可用性』

您可以使用本地 HA 解决方案或多实例队列管理器解决方案为队列管理器配置高可用性。

CP4I MQ Adv. Kubernetes 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。

为了提高性能和可靠性，建议将 RWOReadWriteOnce）持久存储与本地 HA 配置一起使用。 如果来自任何存储器提供者的 RWO 卷满足以下条件，那么这些卷受支持:

- 从块存储器提供程序获取。
- 格式化为 ext4 或 XFS (确保 POSIX 合规性)。
- 支持动态卷供应和"volumeBindingMode: WaitForFirstConsumer"）。

明确禁止以下提供程序:

- NFS
- GlusterFS
- 其他非块提供程序。

注: V 9.4.2 如果配置原生高可用性跨区域复制，所需的磁盘空间至少是原生高可用性的两倍。 这是因为在重建索引期间需要日志备份目录（请参阅 第 42 页的『本地 HA 跨区域复制』）

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。

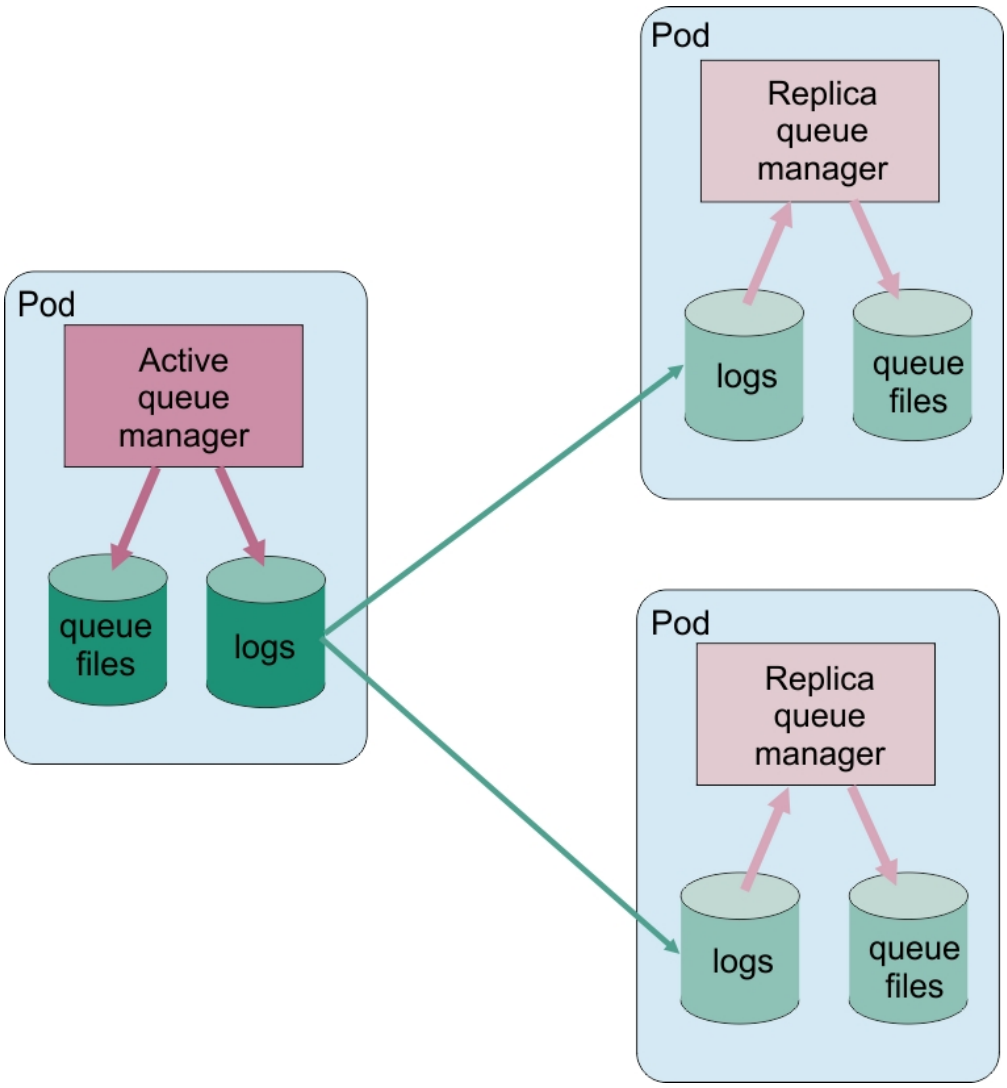


图 *1:* 本机 *HA* 配置示例

MQ Adv. 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。

Kubernetes 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 26 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。

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

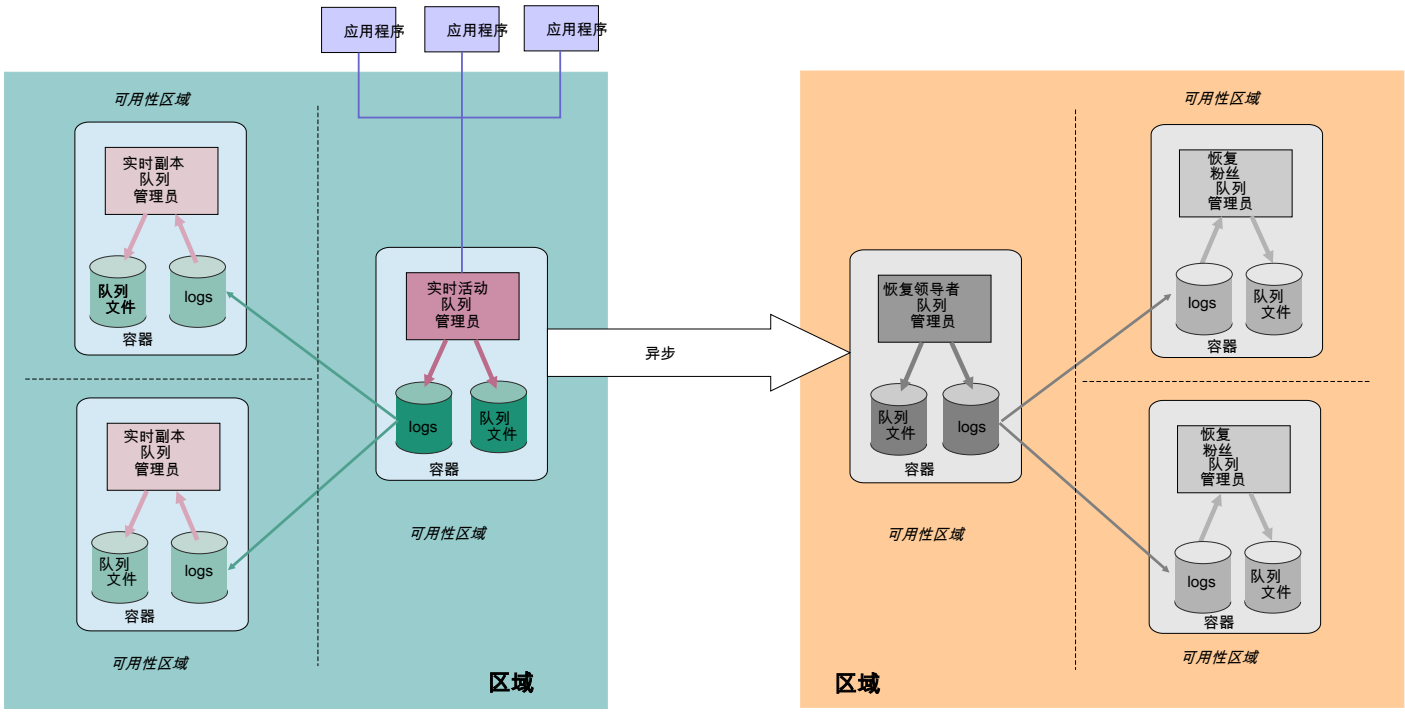
ha_for_ctr.dita

V 9.4.2 本地 HA 跨区域复制

您可以基于原生高可用性解决方案，实施原生高可用性跨区域复制（CRR）配置。 CRR 配置提供灾难恢复功能。

如需了解 Native HA 解决方案的概况，请发送电子邮件至 第 38 页的『本机 HA』 。

本机 HA 使用三个队列管理器实例来维护日志数据的复制副本。 该组通过比较和选择最佳数据副本来自动选择活动实例。 CRR 扩展了日志复制，使其包括一个可能地理位置较远但同样高度可用的组（区域）。 当主站点出现计划内或计划外停机时，恢复站点的团队可以接管工作。 您首先定义一个组扮演的角色，然后根据需要手动切换组角色，以完成计划的切换，或者在发生意外中断时进行切换。 日志数据是异步复制到恢复组的，因此在意外中断后进行切换时可能会丢失一些数据。



要求角色和实际角色

请求的角色是本地高可用性实例的 GroupRole 配置设置，该设置在其 `qm.ini` 的 `NativeHALocalInstance` 节中指定。 该设置要求一个实例与其组中的其他实例合作以完成一个角色。 当大多数实例具有相同的请求角色时，可以选举一个小组领导。 在赢得选举时，请求的角色与实例用于推导有效角色的最后一个角色进行比较。 请参阅 qm.ini 文件的 NativeHAInstance 节和 qm.ini 文件的 NativeHALocalInstance 节。

现场角色

Live 组是一个由三个实例组成的常规 Native HA 队列管理器。 一个实例被选为活动实例，该实例接受来自应用程序的新工作。 现有的原生高可用性队列管理器可以迁移到原生持续关系恢复配置。

如果 `qm.ini` 中没有明确指定所需角色，则默认角色为“在线”。 Live 群组的当选领导也被称为活动队列管理器，它可以接受应用程序和工具连接，以执行新的消息传递工作并管理队列管理器对象。

恢复角色

恢复组也由三个实例组成，其中一个实例被选为组长。 领导者仅作为代理，接受来自配对实时群组的工作。 不允许任何应用程序连接到恢复组，但允许一些控制命令，以便完成结束队列管理器实例等操作。

Live 组的 `qm.ini` 配置文件中的 `NativeHARecoveryGroup` 节确定了恢复组的网络地址。

待定角色转换

当执行计划中的切换时，当前的活动组将成为新的恢复组（当前恢复组将成为新的活动组），此时需要协调两个组。 这种协调确保了日志的一致性，并实现了 RPO（恢复点目标）为零。

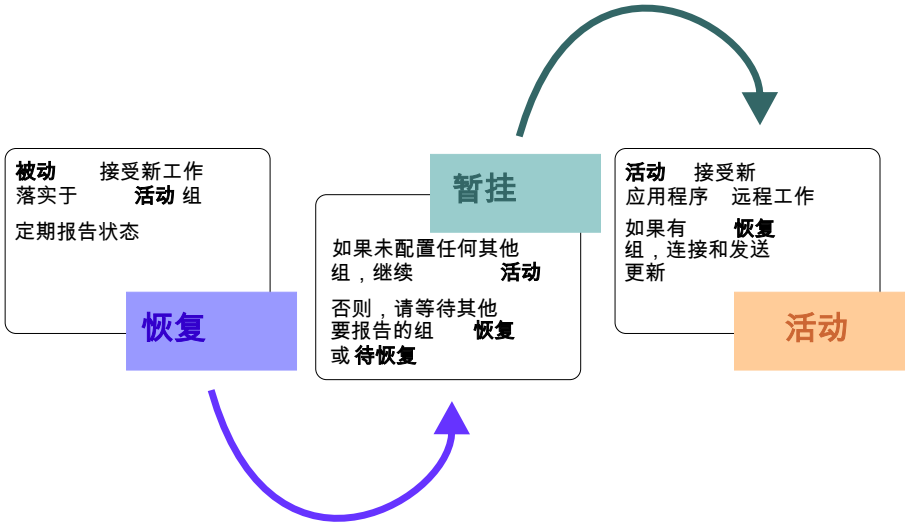
当两个小组协调恢复和实时之间的切换时，两个小组会临时采用一个有效的待定角色。

恢复角色到实时角色（待处理）

之前以恢复角色运行、现在打算改为实时角色的群组，会根据是否有其他配置的群组来决定何时使用该角色。

如果没有其他已配置的组，则立即转换为实时角色。

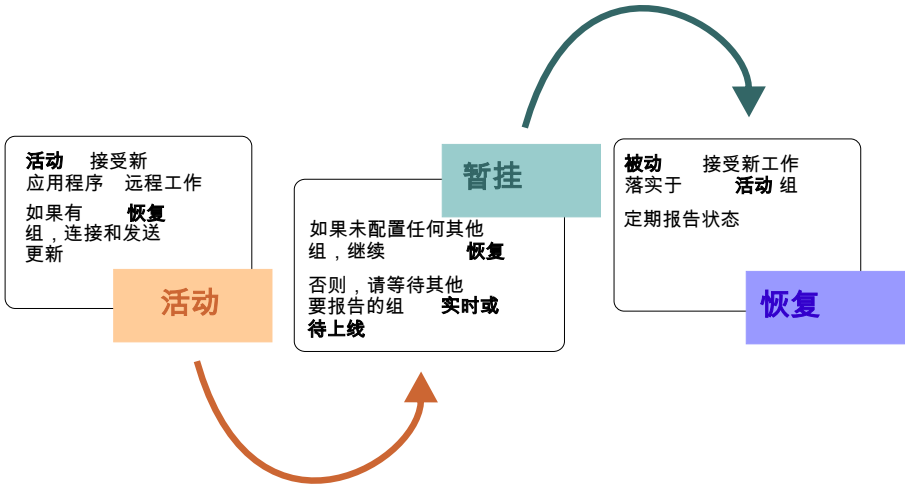
如果配置并启用了另一个组，则向实时角色的转换将被阻止，直到另一个组采用恢复或待恢复角色，并且两个组保持同步。



从“运行”角色切换到“恢复”角色（待恢复）

之前以实时角色运行、打算改为恢复角色的组将根据是否有其他组配置的情况，决定何时使用恢复角色。

如果配置并启用了另一个组，则向恢复角色的转换将被阻止，直到另一个组采用实时或待定实时角色，并且两个组保持同步。



复制和重新定位

在 Native HA CRR 配置中，当两个组连接时，日志更新将传递给恢复组的领导者，并传递给其他恢复实例。这就是所谓的**复制**。

如果两个组之间的网络连接断开，恢复组可能无法使用复制的日志进行恢复（例如，因为实时组重复使用了恢复所需的日志范围）。 在这种情况下，恢复负责人会执行**重设基线操作**。 重新定位是指丢弃接收到的日志数据，并根据实时组发送的完整日志数据集重建队列管理器的过程。 恢复负责人完成重建后，日志将复制到恢复实例。

如果在重新定位操作期间遇到问题，则队列管理器可能无法启动恢复组中的任何成员。 为了避免这种可能性，在重新定位操作开始之前，会先备份日志。 如果重排失败，可以使用备份日志数据重新启动队列管理器。 备份日志将在重建成功之后被删除。

由于需要备份日志，原生高可用性 CRR 配置所需的存储空间至少是原生高可用性配置的两倍。

OpenShift CP4I 规划容器中 IBM MQ 的安全性

在容器配置中规划 IBM MQ 时的安全注意事项。

过程

- 第 44 页的『容器中 IBM MQ 的用户认证和授权』
  - 第 44 页的『有关在容器中使用操作系统用户的安全性约束』
- 第 45 页的『限制容器中 IBM MQ 的网络流量的注意事项』

容器中 IBM MQ 的用户认证和授权

可以将容器中的 IBM MQ 配置为通过 LDAP，相互 TLS 或定制 MQ 插件来认证用户。

请注意， IBM MQ 操作程序不允许在容器映像中使用操作系统用户和组。 有关更多信息，请参阅第 44 页的『有关在容器中使用操作系统用户的安全性约束』。

LDAP

有关配置 IBM MQ 以使用 LDAP 用户存储库的信息，请参阅 连接认证: 用户存储库 和 LDAP 授权。

相互 TLS

如果将与队列管理器的入局连接配置为需要 TLS 证书 (相互 TLS)，那么可以将证书的专有名称映射到用户名。 你需要做两件事:

- 使用 SSLPEER 配置通道认证记录以创建到用户名的映射。 有关更多信息，请参阅 将 SSL 或 TLS 专有名称映射到 MCAUSER 用户标识。
- 配置队列管理器以允许您为系统未知的用户名定义权限记录。 有关更多信息，请参阅 qm.ini 文件的服务节。

JSON Web 令牌

有关配置 IBM MQ 以使用 JSON Web 令牌 (JWT) 的信息，请参阅 使用认证令牌。

定制 MQ 插件

这是一种先进的技术，需要做更多的工作。 有关更多信息，请参阅 使用定制授权服务。

相关任务

第 92 页的『示例: 配置具有相互 TLS 认证的队列管理器』
此示例使用 IBM MQ Operator 将队列管理器部署到 OpenShift Container Platform 中。 相互 TLS 用于认证，以从 TLS 证书映射到队列管理器中的身份。

有关在容器中使用操作系统用户的安全性约束

建议不要在容器中使用操作系统用户， IBM MQ 操作程序禁止使用这些用户。

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，以便成为 "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。

限制容器中 IBM MQ 的网络流量的注意事项

您可以在 OpenShift Container Platform 和 Kubernetes 中定义网络策略，以限制集群中 pod 的流量。 本主题描述了网络策略如何应用于 IBM MQ 的一些注意事项。

对于队列管理器的网络入口，需要考虑多个端口:

- 队列管理器流量的端口 1414
- 用于本机 HA 的端口 9414
- 用于度量的端口 9157
- Web 控制台和 REST API 的端口 9443

网络出口更复杂。 您可能要考虑的网络出口示例:

- DNS-如果您有使用 DNS 名称的通道或其他配置
- 其他队列管理器
- 联机证书状态协议 (OCSP) 和证书撤销列表 (CRL)-由证书提供者确定。
- 认证服务提供者:
  - LDAP
  - 为 IBM MQ Web 服务器打开 ID Connect 或其他已配置的登录提供程序。 这包括 IBM Cloud Pak Keycloak。
- 跟踪提供程序:
  - IBM Instana

  注: 对于较早的 IBM MQ 版本， IBM Cloud Pak for Integration 操作仪表板也可用作跟踪提供程序。 但是，已在 IBM MQ 9.3.3 CD 和 IBM MQ 9.4.0 LTS 上除去操作仪表板。

示例入口 NetworkPolicy

以下是一个示例网络策略，用于控制名为 "myqm" 的队列管理器的入口，以便在 Red Hat OpenShift Container Platform 上使用。

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: myqm
spec:
  podSelector:
    matchLabels:
      app.kubernetes.io/instance: myqm
      app.kubernetes.io/name: ibm-mq
  ingress:
    # Allow access to queue manager listener from anywhere
    - ports:
        - protocol: TCP
          port: 1414
    # Allow access to Native HA port from other instances of the same queue manager
    - from:
        - podSelector:
            matchLabels:
              app.kubernetes.io/instance: myqm
              app.kubernetes.io/name: ibm-mq
      ports:
        - protocol: TCP
```

```
      port: 9414
  # Allow access to metrics from monitoring project
  - from:
      - namespaceSelector:
          matchLabels:
            network.openshift.io/policy-group: monitoring
    ports:
      - protocol: TCP
        port: 9157
  # Allow access to web server via Route
  - from:
      - namespaceSelector:
          matchLabels:
            network.openshift.io/policy-group: ingress
    ports:
      - protocol: TCP
        port: 9443
```

FIPS IBM MQ

启动时，容器中的 IBM MQ 会检测启动容器的操作系统是否兼容 FIPS，如果兼容，则会自动配置 FIPS 支持。此处说明了需求和限制。

Federal Information Processing Standards

美国政府对 IT 系统和安全（包括数据加密）制定了技术建议。国家标准技术研究所 (NIST) 是一个关注 IT 系统和安全的政府机构。NIST 提出建议和标准，包括 Federal Information Processing Standards (FIPS)。

FIPS 的一个重要标准是 FIPS 140-2，它要求使用强大的加密算法。FIPS 140-2 还规定了用于防止数据包在传输过程中被修改的哈希算法的要求。

IBM MQ FIPS 140-2 支持。

注: 在 AIX, Linux, and Windows 上，IBM MQ 通过 IBM Crypto for C (ICC) 密码模块提供 FIPS 140-2 合规性。此模块的证书已移至历史状态。客户应查看 IBM Crypto for C (ICC) 证书，并了解 NIST 提供的任何建议。

V 9.4.2 这 FIPS 140-3 加密模块 IBM Semeru Runtime 于 2024 年 8 月获得 NIST 批准。IBM MQ 9.4.2 在 和 中为 添加了使用 TLS 处理 和 客户端连接的支持。Java 8 IBM Semeru Runtime 11+ FIPS 140-3 IBM MQ classes for JMS IBM MQ classes for Java 与 FIPS 140-3 模块相关的 NIST 认证可在 https://csrc.nist.gov/projects/cryptographic-module-validation-program/certificate/4755 上查看。FIPS 140-2 提供商仍是默认配置文件。IBM MQ 9.4.2 不会更改默认行为，但可以通过 配置连接。FIPS 140-3

对于 IBM MQ in Containers，IBM MQ Operator 3.2.0 和队列管理器容器映像 9.4.0.0 之后都基于 UBI 9。FIPS 140-3 IBM MQ in Containers 的合规性目前尚待。

需求

有关集群设置的要求和其他注意事项，请参见 FIPS Wall：IBM 当前的 FIPS 合规方法。

IBM MQ 可以在 合规模式下运行。FIPS 140-2 在启动过程中，容器中的 IBM MQ 会检测启动容器的主机操作系统是否兼容 FIPS。如果主机操作系统符合 FIPS 标准，且已提供私钥和证书，则 IBM MQ 容器会将队列管理器、IBM MQ Web 服务器以及本地高可用性部署中节点间的数据传输配置为在 FIPS 兼容模式下运行。

使用 IBM MQ Operator 部署队列管理器时，操作程序将创建终止类型为 **Passthrough** 的路由。这意味着将流量直接发送到目标，而不需要路由器提供 TLS 终止。在这种情况下，IBM MQ 队列管理器和 IBM MQ 网络服务器就是目的地，它们已经提供了符合 FIPS 标准的安全通信。

主要要求:

1. 在队列管理器和 Web 服务器的私钥中提供的专用密钥和证书，允许外部客户机安全地连接到队列管理器和 Web 服务器。
2. 用于在本机高可用性配置中的不同节点之间进行数据传输的专用密钥和证书。

局限性

要在容器中部署符合 FIPS 标准的 IBM MQ，请考虑以下几点：

- 容器中的 IBM MQ 提供了用于收集度量值的端点。 目前，该终端只有 HTTP。 您可以关闭度量端点，使 IBM MQ FIPS 的其他部分符合要求。
- 容器中的 IBM MQ 允许定制映像覆盖。 即，您可以使用 IBM MQ 容器映像作为基本映像来构建定制映像。 FIPS 合规可能不适用于此类定制图像。
- 对于使用 IBM Instana 进行信息跟踪的情况， IBM MQ 和 IBM Instana 之间的通信是 HTTP 或 HTTPS，不符合 FIPS 的规定。
- IBM MQ Operator 访问 身份和访问管理 (IAM)/Zen 服务不符合 标准。 IBM FIPS

如何检测 *FIPS* 合规性并自动配置 *FIPS* 支持

如果启动容器的操作系统兼容 FIPS，则会自动配置 FIPS 支持。

注: 在 AIX, Linux, and Windows 上， IBM MQ 通过 IBM Crypto for C (ICC) 密码模块提供 FIPS 140-2 合规性。 此模块的证书已移至历史状态。 客户应查看 IBM Crypto for C (ICC) 证书，并了解 NIST 提供的任何建议。

V 9.4.2 这 FIPS 140-3 加密模块 IBM Semeru Runtime 于 2024 年 8 月获得 NIST 批准。 IBM MQ 9.4.2 在 和 中为 添加了使用 TLS 处理 和 客户端连接的支持。 Java 8 IBM Semeru Runtime 11+ FIPS 140-3 IBM MQ classes for JMS IBM MQ classes for Java 与 FIPS 140-3 模块相关的 NIST 认证可在 https://csrc.nist.gov/projects/cryptographic-module-validation-program/certificate/4755 上查看。 FIPS 140-2 提供商仍是默认配置文件。 IBM MQ 9.4.2 不会更改默认行为，但可以通过 配置连接。 FIPS 140-3

对于 IBM MQ in Containers， IBM MQ Operator 3.2.0 和队列管理器容器映像 9.4.0.0 之后都基于 UBI 9。 FIPS 140-3 IBM MQ in Containers 的合规性目前尚待。

在启动过程中，容器中的 IBM MQ 会检测启动容器的操作系统是否符合 FIPS 标准。 如果是这样，那么将自动执行以下操作:

队列管理器
如果主机操作系统符合 FIPS 标准，并提供了私钥和证书，队列管理器属性 **SSLFIPS** 就会设置为 "**是**"。 否则， **SSLFIPS** 属性将设为 "NO"。

IBM MQ Web 服务器
IBM MQ 网络服务器提供 HTTP / HTTPS 界面，用于管理 IBM MQ。 如果主机操作系统兼容 FIPS，则会更新 JVM 选项，使网络服务器使用兼容 FIPS 的加密技术。 为了能够使用 FIPS，必须在容器启动期间提供专用密钥和证书。

本机 HA
节点之间复制的数据的安全性由 qm.ini 文件的 **NativeHALocalInstance** 节控制。 例如:

```
NativeHALocalInstance:
    KeyRepository=/run/runmqserver/ha/tls/key.kdb
    CertificateLabel=NHAQM
    CipherSpec=ECDHE_RSA_AES_256_GCM_SHA384
```

如果启用了 FIPS，则 **SSLFipsRequired** 属性会添加到协议中，其值设为 "**是**"：

```
NativeHALocalInstance:
    KeyRepository=/run/runmqserver/ha/tls/key.kdb
    CertificateLabel=NHAQM
    CipherSpec=ECDHE_RSA_AES_256_GCM_SHA384
    SSLFipsRequired=Yes
```

如果容器在不支持 FIPS 的 OpenShift 集群中运行，则队列管理器、 IBM MQ Web 服务器和本地 HA 组件不会自动启用 FIPS 支持。 FIPS 的 OpenShift 平台目前仅支持 x86-64 架构。 对于 Power 和 Linux on IBM Z 架构， OpenShift 不提供 FIPS 支持。 要在 IBM MQ 组件中为这些架构明确启用 FIPS 支持，请在队列管理器 YAML 中将 *MQ_ENABLE_FIPS* 环境变量设为 true。 下面的 YAML 代码段描述了 *MQ_ENABLE_FIPS* 环境变量的用法:

```
template:
    pod:
```

```
      containers:
        - env:
            - name: MQ_ENABLE_FIPS
              value: "true"
          name: qmgr
```

覆盖 *IBM MQ* 容器中的自动 *FIPS* 模式

使用环境变量 *MQ_ENABLE_FIPS* ，可为 IBM MQ 容器中的组件明确启用或禁用 FIPS 模式。

开始之前

注: 在 AIX, Linux, and Windows 上， IBM MQ 通过 IBM Crypto for C (ICC) 密码模块提供 FIPS 140-2 合规性。 此模块的证书已移至历史状态。 客户应查看 IBM Crypto for C (ICC) 证书 ，并了解 NIST 提供的任何建议。

V 9.4.2 这 FIPS 140-3 加密模块 IBM Semeru Runtime 于 2024 年 8 月获得 NIST 批准。 IBM MQ 9.4.2 在 和 中为 添加了使用 TLS 处理 和 客户端连接的支持。 Java 8 IBM Semeru Runtime 11+ FIPS 140-3 IBM MQ classes for JMS IBM MQ classes for Java 与 FIPS 140-3 模块相关的 NIST 认证可在 https://csrc.nist.gov/projects/cryptographic-module-validation-program/certificate/4755 上查看。 FIPS 140-2 提供商仍是默认配置文件。 IBM MQ 9.4.2 不会更改默认行为，但可以通过 配置连接。 FIPS 140-3

对于 IBM MQ in Containers， IBM MQ Operator 3.2.0 和队列管理器容器映像 9.4.0.0 之后都基于 UBI 9。 FIPS 140-3 IBM MQ in Containers 的合规性目前尚待。

关于此任务

MQ_ENABLE_FIPS 支持三个值:

自动
这是缺省值。

如果主机操作系统启用了 FIPS ，那么所有组件（队列管理器、 IBM MQ 网络服务器和本地 HA）都在 FIPS 模式下运行。

如果主机操作系统未启用 FIPS ，则所有组件都不会以 FIPS 模式运行。

true
此值可为容器中的选定组件打开 FIPS 。

即使容器中的 IBM MQ 运行在不兼容 FIPS 的主机操作系统上，队列管理器属性 **SSLFIPS** 也会被设置为 "**是**"。 也就是说，如果 IBM MQ 队列管理器、网络服务器和本地 HA 兼容 FIPS ，但容器的操作系统不兼容。

false
该值将关闭 FIPS 合规性。

队列管理器属性 **SSLFIPS** 设置为 NO ，即使容器中的 IBM MQ 运行在符合 FIPS 标准的主机上。 不过，如果提供私钥和证书， IBM MQ 仍可确保连接安全。

不会更新 IBM MQ Web 服务器的 JVM 选项。 不过，如果提供了私钥和证书， IBM MQ 网络服务器仍可运行 HTTPS 端点。

本地 HA 中的数据复制不使用 FIPS 加密技术。

示例

下面是一个队列管理器 YAML 示例，描述了队列管理器组件启用 TLS 和 FIPS 的情况。

注: 本示例使用 IBM MQ Advanced 许可证，但也可以使用 IBM MQ 许可证。 有关更多信息，请参阅第 140 页的『使用 IBM MQ 许可证注释配置队列管理器。 IBM MQ Operator』。

```
apiVersion: mq.ibm.com/v1beta1
kind: QueueManager
metadata:
  namespace: ibm-mq-fips
  name: exampleqm
spec:
  license:
```

```
    accept: true
    license: L-NUUP-23NH8Y
    use: Production
  queueManager:
    name: EXAMPLEQM
    storage:
      queueManager:
        type: ephemeral
  template:
    pod:
      containers:
        - env:
            - name: MQ_ENABLE_FIPS
              value: "true"
          name: qmgr
  version: 9.4.3.0-r1
  web:
    enabled: false
  pki:
    keys:
      - name: ibm-mq-tls-certs
        secret:
          secretName: ibm-mq-tls-secret
          items:
          - tls.key
          - tls.crt
```

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。 从 Red Hat OpenShift Container Platform 4.11 开始，每个容器的缺省限制为 4096 个进程。 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。

在容器中为 IBM MQ 准备，安装和升级环境

您可以执行一系列任务，为 IBM MQ 准备环境

关于此任务

如果您正在使用 IBM MQ Operator，那么通过安装操作程序来准备 Red Hat OpenShift Container Platform 集群。 请参阅 第 50 页的『安装和升级 IBM MQ Operator』

否则，您可以通过构建自己的容器映像来准备容器环境。 请参阅第 76 页的『通过构建您自己的容器映像来准备 IBM MQ』

安装和升级 IBM MQ Operator

您可以执行一系列任务来安装，卸载和升级 IBM MQ Operator。

关于此任务

要开始安装和升级 Red Hat OpenShift Container Platform 上的 IBM MQ Operator ， 请参阅以下主题。

过程

- 第 50 页的『IBM MQ Operator 的依赖关系』
- 第 50 页的『IBM MQ Operator 所需的集群范围的许可权』
- 第 51 页的『验证图像特征符』
- 第 51 页的『安装 IBM MQ Operator』
- 第 61 页的『升级 IBM MQ Operator 和队列管理器』
- 第 73 页的『卸载 IBM MQ Operator』

OpenShift CP4I IBM MQ Operator 的依赖关系

安装 IBM MQ Operator 时，不会自动安装其他操作程序。

需要单独安装 IBM Licensing Operator 以跟踪许可证使用情况。 请参阅 IBM Cloud Pak for Integration 文档中的 部署 License Service 。

使用 IBM Cloud Pak for Integration 许可证创建 `QueueManager` 时，可以选择是否要将单点登录用于 Keycloak 的 IBM Cloud Pak for Integration 实例。 缺省情况下，通过 IBM Cloud Pak for Integration 许可证启用了 Keycloak 的使用，但如果未安装此许可证，那么 `QueueManager` 将进入 "已阻止" 状态，直到安装正确的依赖关系为止。 有关依赖关系的更多详细信息，请参阅 第 51 页的『安装 IBM MQ Operator』 。

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

IBM MQ Operator 始终需要群集权限才能读取存储类和群集版本信息。

IBM MQ Operator 可以安装在两种不同的范围内：

1. **群集范围**意味着操作员将被分配使用 `ClusterRole` 和 `ClusterRoleBinding` 资源的权限。 这将使操作员能够在所有命名空间中执行操作。
2. **命名空间范围**意味着操作符将使用 `Role` 和 `RoleBinding` 分配权限，而这两种方法只能在给定的命名空间中分配权限。 此外，还将为以下只读权限提供 `ClusterRole` 和 `ClusterRoleBinding` ：

- 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。 这样，操作员就可以尝试反馈容器中选定存储类别的任何问题。
  - API 组: **`storage.k8s.io`**
  - 资源： **`storageclasses`**
  - verbs: **`get, list`**

OpenShift CP4I 验证图像特征符

IBM MQ Operator 和 IBM MQ 队列管理器容器映像已进行数字签名。

关于此任务

数字签名为内容的消费者提供了一种方法，确保他们下载的内容既真实 (源自预期来源) ，又具有完整性 (这是我们所期望的)。

过程

- 验证 IBM MQ Operator 和 IBM MQ 队列管理器容器映像的特征符:
  - 对于 3.4.0 或更高版本的 "IBM MQ Operator，或 9.4.1.0-r2 或更高版本的 "IBM MQ 队列管理器容器映像，请参阅 "IBM Cloud Pak for Integration(CP4I)16.1.1 文档中的验证映像签名。
  - For an IBM MQ Operator at 3.2.x or 3.3.x, or an IBM MQ queue manager container image between versions 9.4.0.0-r1 and 9.4.1.0-r1, see 验证图像签名 in the IBM Cloud Pak for Integration (CP4I) 16.1.0 documentation.

OpenShift CP4I 安装 IBM MQ Operator

可以使用 OpenShift 控制台或命令行界面 (CLI) 将 IBM MQ Operator 安装到 Red Hat OpenShift 上。

开始之前

要点:

- 本主题用于安装 IBM MQ Operator， **仅供独立使用**。 如果您打算将 IBM Cloud Pak for Integration 或 Keycloak SSO 用于一个或多个队列管理器，请参阅 第 57 页的『安装 IBM MQ Operator 以用于 CP4I』。
- 在安装 IBM MQ Operator 之前，请查看有关 构造部署 的指导信息。

要确保安装尽可能顺利，请确保在开始安装之前了解所有先决条件和需求。 请参阅 第 26 页的『规划容器中的 IBM MQ』。

关于此任务

以下步骤表示用于安装 IBM MQ Operator 的典型任务流:

1. 安装 Red Hat OpenShift Container Platform.
2. 配置存储。
3. 镜像映像 (仅限气郄)。
4. 添加 IBM MQ Operator 目录。
5. 安装 IBM MQ Operator。
6. 创建权利密钥密钥 (仅联机安装)。
7. 部署 License Service。
8. 部署队列管理器。

过程

1. 安装 Red Hat OpenShift Container Platform。

   有关安装 OpenShift 的详细步骤，请参阅 安装 Red Hat 软件 4.6 或更高版本。

   要点: 确保安装受支持的 OpenShift Container Platform 版本。 例如，要使用 IBM MQ Operator 3.6 或更高版本，必须安装 OpenShift Container Platform 4.12 或更高版本。 有关详细信息，请参阅 IBM Cloud Pak 和 Red Hat OpenShift Container Platform 兼容性。

对于使用 Red Hat OpenShift Container Platform CLI 的任何步骤，必须使用 `oc login` 登录到 OpenShift 集群。 要安装 CLI，请参阅 OpenShift CLI 入门。

安装 OpenShift 后，可以使用您在 创建权利密钥中创建的 IBM 权利密钥来验证并获取对容器软件的访问权。

2. 配置存储器。

   您必须在 Red Hat OpenShift Container Platform 中定义存储类，并设置存储配置以满足您的大小调整需求。

   要点: IBM MQ 单实例和本机 HA 队列管理器可以使用 RWO 访问方式，而多实例队列管理器需要 RWX，如 第 35 页的『规划 IBM MQ Operator 的存储器』中所述。 IBM MQ 多实例队列管理器需要特定的文件系统特征，可以使用 针对 IBM MQ 测试共享文件系统的指示信息来验证这些特征。

   可以在 针对 IBM MQ 文件系统的测试语句中找到已知合规和不合规文件系统的列表以及有关其他限制或限制的注释。

   可以在 CP4I 存储器注意事项 页面上找到建议的存储器提供程序。

3. 镜像 (仅限气郄)。

   如果您的集群处于受限（无连接）网络环境中，则必须镜像 IBM MQ 镜像。

   要点: 从版本 3.5.0 开始， IBM MQ CASE 发生了变化，现在只包含一个队列管理器版本。 镜像过程运行速度更快，但可能需要多次运行；每次运行一次，用于安装每个版本的队列管理器。

   有关更多信息，请参阅第 261 页的『在 "Red Hat OpenShift 上进行空隙安装和升级的 IBM MQ CASE』。

   您必须查找 IBM MQ CASE 版本中包含您需要的操作员和队列管理器版本，并分别重复气隙镜像过程。 查看 IBM MQ CASE 查找表，了解版本对应关系。

   例如，当按照以下链接的镜像说明进行操作时，您可以通过使用以下值来安装最新的操作员和单个对应的队列管理器版本：

   ```
   export OPERATOR_PACKAGE_NAME=ibm-mq
   export OPERATOR_VERSION=3.6.0
   ```

   有关用于间隙安装和升级的 IBM MQ CASE 内容的信息，请参阅 "第 261 页的『在 "Red Hat OpenShift 上进行空隙安装和升级的 IBM MQ CASE』。

   要创建镜像映像，请参阅 气郄集群的镜像映像。

4. 添加 IBM MQ Operator 目录源。

   添加使操作程序可用于集群的目录源。 请参阅 第 53 页的『添加 IBM MQ Operator 目录源』。

5. 安装 IBM MQ Operator。

   选择以下两个选项之一 (使用控制台或 CLI):

   - 选项 1: 使用 OpenShift 控制台安装 IBM MQ Operator。
   - 选项 2: 使用 OpenShift CLI 安装 IBM MQ Operator。

6. 创建权利密钥密钥 (仅联机安装)。

   IBM MQ Operator 部署从执行许可证权利检查的容器注册表中拉取的队列管理器映像。 此检查需要存储在 `docker-registry` 拉取私钥中的权利密钥。 如果在将安装队列管理器的名称空间中还没有权利密钥，请遵循以下指示信息以获取权利密钥并创建拉取私钥。

   注: 如果将仅部署 IBM MQ Advanced for Developers (非受警告) 队列管理器，那么不需要权利密钥。

   您可以使用 OpenShift 控制台或 CLI 来创建权利密钥私钥。 以下示例使用 CLI:

   a. 获取分配给 IBM 标识的权利密钥。 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。

c. 在 OpenShift CLI 中，运行以下命令以创建名为 `ibm-entitlement-key` 的映像拉取私钥。

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

其中 *entitlement lement_key* 是您在步骤 b 中复制的权利密钥， *user_email* 是与授权软件关联的 IBM 标识， *namespace* 是您将 IBM MQ Operator 安装到的名称空间。

7. 部署 License Service。

这是监视队列管理器的许可证使用情况所必需的。 遵循 部署 License Service 中的指示信息。

8. 部署队列管理器。

有关部署示例 “快速启动” 队列管理器的指示信息，请参阅 第 85 页的『使用 IBM MQ Operator 部署简单队列管理器』。

相关任务

第 73 页的『卸载 IBM MQ Operator』
您可以使用 Red Hat OpenShift 控制台或 CLI 从 Red Hat OpenShift 卸载 IBM MQ Operator 。

OpenShift 添加 *IBM MQ Operator* 目录源

将 IBM MQ Operator 目录源添加到 OpenShift 集群，以使 IBM MQ Operator 可用于安装。 如果在完成升级之前应用目录源修订包，那么此任务也是必需的。

关于此任务

操作程序目录是可用于扩展 Red Hat OpenShift Container Platform 集群的 API 以启用 IBM 软件产品的操作程序的索引。

以下目录源可用:

选项 1: IBM MQ Operator 的特定目录源。

通过使用特定的 IBM MQ Operator 目录源，您可以完全控制集群上的软件版本控制以及升级的发生时间。 更新目录源后，新的 IBM MQ Operator 版本 **仅** 在 OpenShift 集群中可用。 此过程有效地使您能够手动控制升级，因此您不需要将 **手动** 选项用于操作程序的 **Update approval** 设置。 **手动** 选项强制同时执行所有可能的升级，并且可以阻止升级，因此仅使用 **自动** 选项。 有关更多信息，请参阅 使用 Red Hat OpenShift 控制台安装操作程序的 "使用核准策略限制自动更新" 部分。

如果要完成升级并且需要添加较新版本的 IBM MQ Operator 目录源，请选择此选项。

要使用此选项，请跳至 选项 1: 为 IBM MQ Operator 添加特定目录源。

选项 2: IBM 操作程序目录。

通过此选项，新的操作程序版本将变为可用， **无需** 来自您的任何干预即可应用。 因此，对于要 **自动** 升级 IBM MQ Operator 且不需要确定性安装的联机安装，请 **仅** 使用此选项。

注: 此选项对于概念验证环境很有用，但它 **不适用于生产环境**。

要使用此选项，请跳至 选项 2: 添加 IBM 操作程序目录。

过程

- **选项 1: 为 IBM MQ Operator 添加特定目录源**。

  此任务假定您已完成 第 51 页的『安装 IBM MQ Operator』的前 3 个步骤。

  此任务必须由集群管理员执行，并且必须使用 CLI 执行。

  a) 仅升级: 如果要在升级之前应用目录源修订包，请完成以下步骤:

  – 确认操作程序是否正常运行。

– 如果有任何需要手动核准的暂挂 IBM MQ Operator 更新，请先核准这些更新，然后再启动此过程。有关更多信息，请参阅 使用 Red Hat OpenShift 控制台安装操作程序中的 "使用核准策略限制自动更新"。

b) 如果尚未安装此插件，或者需要更新此插件，请 从 GitHub 下载 IBM 目录管理插件 (V 1.6.0 或更高版本)。

此插件允许您对集群运行 **oc ibm-pak** 命令。

c) 使用 **oc login** 命令和用户凭证登录到集群:

```
oc login openshift_url -u username -p password -n namespace
```

d) 导出 IBM MQ Operator 的以下环境变量:

```
export OPERATOR_PACKAGE_NAME=ibm-mq
export OPERATOR_VERSION=CHOSEN_OPERATOR_VERSION
export ARCH=ARCHITECTURE
```

其中:

– *ARCHITECTURE* 是部署 IBM MQ Operator 的系统架构，值为 amd64、ppc64le 或 s390x。

– *CHOSEN_OPERATOR_VERSION* 是您所需的 IBM MQ Operator 版本。 The current CD version is 3.6.0. The current SC2 version is 3.2.13.

要点: 如果要从 IBM 操作程序目录移动到 IBM MQ Operator 的特定目录源，请将 OPERATOR_VERSION 设置为 IBM MQ Operator 的部署版本。

e) 下载 IBM MQ 操作程序的文件。

注: 如果要完成 **气郄** 安装，那么在完成 "安装 IBM MQ Operator" 的 "镜像映像" 步骤之后，您应该已经具有所需的文件，在这种情况下，您可以跳至步骤 第 54 页的『8』 "将 IBM MQ Operator 目录源应用到集群"。

```
oc ibm-pak get ${OPERATOR_PACKAGE_NAME} --version ${OPERATOR_VERSION}
```

f) 生成 IBM MQ Operator 所需的目录源:

```
oc ibm-pak generate mirror-manifests ${OPERATOR_PACKAGE_NAME} icr.io --version $
{OPERATOR_VERSION}
```

g) 可选： 生成目录源并将其保存在另一个目录中。

a. 获取目录源:

```
cat ~/.ibm-pak/data/mirror/${OPERATOR_PACKAGE_NAME}/${OPERATOR_VERSION}/catalog-
sources.yaml
```

b. (可选) 浏览到文件浏览器中的目录，以将这些工件复制到可以保留以供复用或用于管道的文件中。

h) 将 IBM MQ Operator 目录源应用于集群。

```
oc apply -f ~/.ibm-pak/data/mirror/${OPERATOR_PACKAGE_NAME}/${OPERATOR_VERSION}/catalog-
sources.yaml
```

i) 确认已在 openshift-marketplace 名称空间中创建 IBM MQ Operator 目录源:

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

示例输出如下所示:

```
oc get catalogsource -n openshift-marketplace

NAME                           DISPLAY                TYPE   PUBLISHER   AGE
ibmmq-operator-catalogsource   ibm-mq-3.1.3           grpc   IBM         23h
```

您现在已准备好完成 安装 IBM MQ Operator 的步骤 5。

- **选项 2: 添加 IBM 操作程序目录。**

要点: 对于要 **自动** 升级 IBM MQ Operator 且不需要确定性安装的联机安装，请使用 IBM 操作程序目录 **只** 。 此选项对于概念验证环境很有用，但它 **不适用于生产环境**。

IBM 操作程序目录是可用于扩展 Red Hat OpenShift Container Platform 集群的 API 以启用 IBM 软件产品的操作程序的索引。 将目录源添加到 OpenShift 集群会将 IBM 操作程序添加到可以安装的操作程序列表中。

此任务假定您已完成 第 51 页的『安装 IBM MQ Operator』的前 3 个步骤。

可以使用 CLI 或 OpenShift Web 控制台来执行此任务。

使用 CLI

1. 将 IBM 操作程序的以下资源定义复制到计算机上的本地文件中:

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

2. 运行以下命令。 将 *filename*.yaml 替换为您在上一步中创建的文件的名称:

```
oc apply -f filename.yaml
```

使用 OpenShift Web 控制台

1. 使用 OpenShift 集群管理员凭证登录到 OpenShift Web 控制台。
2. 在条幅中，单击加号 ("+") 图标以打开 " **导入 YAML** " 对话框。

   注: 您不需要为 **项目**选择值。 下一步中的 YAML 代码已包含 `metadata:namespace` 的正确值，这将确保目录源安装在正确的项目 (名称空间) 中。
3. 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
```

4. 单击**创建**。

您现在已准备好完成 安装 IBM MQ Operator 的步骤 5。

OpenShift CP4I 使用 *OpenShift* 控制台安装 *IBM MQ Operator*

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

开始之前

此任务假定您已完成 第 51 页的『安装 IBM MQ Operator』的步骤 1-4。

过程

1. 登录 Red Hat OpenShift 集群控制台。

2. 在导航窗格中，单击 **操作程序** > **OperatorHub**。
   此时将显示 " OperatorHub " 页面。
3. 在 **所有项** 字段中，输入 "IBM MQ"。
   这将显示 IBM MQ 目录条目。
4. 选择 **IBM MQ**。
   此时将显示 " IBM MQ " 窗口。
5. 单击**安装**。
   将显示 "安装操作程序" 页面。
6. 输入以下值：
   a) 将 **通道** 设置为所选版本。

      查看 第 32 页的『IBM MQ Operator 的版本支持』 以确定要选择的操作员通道。

   b) 将 **安装方式** 设置为 "集群上的特定名称空间" (可以在下一步中创建) 或集群范围内的作用域。

      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。

   c) 可选： 如果选择 "集群上的特定名称空间"，请将 **名称空间** 设置为要将操作程序安装到的项目 (名称空间) 值。

      注: 使用控制台安装操作程序时，可以使用现有名称空间 (由操作程序提供的缺省名称空间) 或创建新的名称空间。 如果要创建新的名称空间，可以从此表单创建该名称空间，如下所示: 在导航窗格中，单击 **主页** > **项目**，选择 **创建项目**，指定要创建的项目 (名称空间) 的 **名称**，然后单击 **创建**。

   d) 将 **核准策略** 设置为 "自动"。
7. 单击 **安装** 并等待操作程序安装。

   安装完成时，将向您提供确认信息。

   要验证安装，请浏览至 **操作程序** > **已安装的操作程序**，然后从 **项目** 下拉列表中选择项目。 安装完成时，操作程序的状态将更改为 "已成功"。

下一步做什么

现在，您已准备好 创建权利密钥 (第 51 页的『安装 IBM MQ Operator』 的步骤 6)。

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

可以使用命令行界面 (CLI) 将 IBM MQ Operator 安装到 Red Hat OpenShift 上。

开始之前

此任务假定您已完成 第 51 页的『安装 IBM MQ Operator』 的步骤 1-4。

过程

1. 使用 **oc login** 登录到 Red Hat OpenShift 命令行界面 (CLI)。
2. 可选： 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。

   要在 CLI 中创建新的名称空间，请运行以下命令:

   ```
   oc create namespace namespace_name
   ```

   其中 *namespace_name* 是要创建的名称空间的名称。
3. 从 OperatorHub: 查看群集可用的操作员列表

   ```
   oc get packagemanifests -n openshift-marketplace
   ```
4. 检查 IBM MQ Operator 以验证其受支持的 **InstallModes** 和可用 **Channels**。

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

5. 可选： 创建 **OperatorGroup**。

   OperatorGroup 是一个 OLM 资源，用于选择目标名称空间，在这些名称空间中为与 **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。

   预订操作程序的名称空间必须具有与操作程序的 **InstallMode** 相匹配的 **OperatorGroup**（`AllNamespaces` 或 `SingleNamespace` 方式）。

   如果要安装的操作程序使用 `AllNamespaces` 方式，那么 `openshift-operators` 名称空间已具有适当的 **OperatorGroup**，您可以跳过此步骤。

   如果操作程序使用 `SingleNamespace` 方式，并且您还没有适当的 **OperatorGroup**，请通过运行以下命令来创建一个:

```
cat << EOF | oc apply -f -
apiVersion: operators.coreos.com/v1
kind: OperatorGroup
metadata:
  name: operatorgroup_name
  namespace: namespace_name
spec:
  targetNamespaces:
  - namespace_name
EOF
```

6. 查看 第 32 页的『IBM MQ Operator 的版本支持』 以确定要选择的操作员通道。
7. 安装操作程序。

   使用以下命令更改以下值:

   - 更改 *ibm_mq_operator_channel* 以匹配要安装的 IBM MQ Operator 版本的通道。
   - 如果使用 "AllNamespaces" 模式，则将 *namespace_name* 更改为 **openshift-operators**；如果使用 "SingleNamespace" 模式，则将 IBM MQ Operator 更改为要部署的名称空间。
   - 将 *catalog_source_name* 改为用于安装 IBM MQ Operator 的目录源名称，如上一步安装第 53 页的『添加 IBM MQ Operator 目录源』 所添加的。

```
cat << EOF | oc apply -f -
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: ibm-mq
  namespace: namespace_name
spec:
  channel: ibm_mq_operator_channel
  installPlanApproval: Automatic
  name: ibm-mq
  source: catalog_source_name
  sourceNamespace: openshift-marketplace
EOF
```

8. 几分钟后，安装了操作程序。 运行以下命令以验证所有组件是否都处于 "已成功" 状态:

```
oc get csv -n namespace_name | grep ibm-mq
```

   Where 命名空间名称 is **openshift-operators** if you are using "AllNamespaces" mode, or the project (namespace) name if you are using "SingleNamespace" mode.

下一步做什么

现在，您已准备好 创建权利密钥（第 51 页的『安装 IBM MQ Operator』的步骤 6）。

OpenShift CP4I 安装 IBM MQ Operator 以用于 CP4I

要与 IBM Cloud Pak for Integration (CP4I) 配合使用，可以通过 OpenShift 控制台或命令行界面 (CLI) 将 IBM MQ Operator 安装到 Red Hat OpenShift 上。

开始之前

要点:

- 本主题用于安装 IBM MQ Operator 以与 CP4I 配合使用，或者如果您打算 **仅**使用 CP4I 许可证来部署至少一个队列管理器。 有关安装 IBM MQ Operator 以供独立使用的指示信息，请参阅 第 51 页的『安装 IBM MQ Operator』。
- 在安装 IBM MQ Operator 之前，请查看有关 构造部署 的指导信息。

要确保安装尽可能顺利，请确保在开始安装之前了解所有先决条件和需求。 请参阅 第 26 页的『规划容器中的 IBM MQ』。

关于此任务

以下步骤表示用于安装 IBM MQ Operator 的典型任务流:

1. 安装 Red Hat OpenShift Container Platform.
2. 配置存储。
3. 镜像映像 (仅限气郄)。
4. 添加 IBM MQ Operator 目录 并准备集群。
5. 安装 IBM MQ Operator。
6. 创建权利密钥密钥 (仅联机安装)。
7. 可选: 安装 IBM Cloud Pak for Integration (CP4I) 及其依赖关系。
8. 部署 License Service。
9. 部署队列管理器。

过程

1. 安装 Red Hat OpenShift Container Platform。

   有关安装 OpenShift 的详细步骤，请参阅 安装 Red Hat 软件 4.6 或更高版本。

   要点: 确保安装受支持的 OpenShift Container Platform 版本。 例如，要使用 IBM MQ Operator 3.6 或更高版本，必须安装 OpenShift Container Platform 4.12 或更高版本。 有关详细信息，请参阅 IBM Cloud Pak 和 Red Hat OpenShift Container Platform 兼容性。

   对于使用 Red Hat OpenShift Container Platform CLI 的任何步骤，必须使用 `oc login` 登录到 OpenShift 集群。 要安装 CLI ，请参阅 OpenShift CLI 入门。

   安装 OpenShift 后，您可以使用在 创建权利密钥中创建的 IBM 权利密钥来验证并获取对容器软件的访问权。
2. 配置存储器。

   您必须在 Red Hat OpenShift Container Platform 中定义存储类，并设置存储配置以满足大小调整需求。

   要点: IBM MQ 单实例和本机 HA 队列管理器可以使用 RWO 访问方式，而多实例队列管理器需要 RWX ，如 第 35 页的『规划 IBM MQ Operator 的存储器』中所述。 IBM MQ 多实例队列管理器需要特定的文件系统特征，可以使用 针对 IBM MQ 测试共享文件系统的指示信息来验证这些特征。

   可以在 针对 IBM MQ 文件系统的测试语句中找到已知合规和不合规文件系统的列表以及有关其他限制或限制的注释。

   可以在 CP4I 存储器注意事项 页面上找到建议的存储器提供者。
3. 镜像 (仅限气郄)。

   如果集群处于受限 (气郄) 网络环境中，那么必须对 IBM MQ 映像进行镜像。 根据您的配置，您可能还需要对一些其他组件进行镜像。 请阅读以下信息，然后根据需要对映像进行镜像。

   - 您必须对 IBM MQ 映像进行镜像。

要点: 从版本 3.5.0 开始， IBM MQ CASE 发生了变化，现在只包含一个队列管理器版本。 镜像过程运行速度更快，但可能需要多次运行；每次运行一次，用于安装每个版本的队列管理器。

有关更多信息，请参阅第 261 页的『在 "Red Hat OpenShift 上进行空隙安装和升级的 IBM MQ CASE』。

您必须查找 IBM MQ CASE 版本中包含您需要的操作员和队列管理器版本，并分别重复气隙镜像过程。 查看 IBM MQ CASE 查找表 ，了解版本对应关系。

例如，当按照以下链接的镜像说明进行操作时，您可以通过使用以下值来安装最新的操作员和单个对应的队列管理器版本:

```
export OPERATOR_PACKAGE_NAME=ibm-mq
export OPERATOR_VERSION=3.6.0
```

有关用于间隙安装和升级的 IBM MQ CASE 内容的信息，请参阅 "第 261 页的『在 "Red Hat OpenShift 上进行空隙安装和升级的 IBM MQ CASE』。

- 如果您打算部署至少一个队列管理器，其中 **所有** 以下语句均为 true ，那么还必须对一些其他必需组件进行镜像:
  - 您正在使用 CP4I 许可证。
  - IBM MQ Console 已启用。
  - 您正在将 IBM Cloud Pak for Integration Keycloak 服务用于 IBM MQ Console 单点登录 (SSO) 认证和授权 (缺省值)。

  如果先前的所有语句都为 true ，那么 SSO 由 Keycloak 提供。 因此，除了 IBM MQ Operator ，您还必须为其他所需组件重复上述步骤:
  - IBM Cloud Pak foundational services
  - IBM Cloud Pak for Integration
  - Keycloak (Red Hat OpenShift 运算符)

要创建镜像映像，请参阅 气郄集群的镜像映像。

4. 添加 IBM MQ Operator 目录源。

使用以下值添加使 IBM MQ Operator 可供集群使用的目录源:

```
export OPERATOR_PACKAGE_NAME=ibm-mq
export OPERATOR_VERSION=CHOSEN_OPERATOR_VERSION
export ARCH=ARCHITECTURE
```

其中:

- *ARCHITECTURE* 是部署 IBM MQ Operator 的系统架构，值为 amd64、ppc64le 或 s390x。
- *CHOSEN_OPERATOR_VERSION* 是您所需的 IBM MQ Operator 版本。 The current CD version is 3.6.0. The current SC2 version is 3.2.13.

在部署至少一个队列管理器时，存在一些其他必需组件，其中 **所有** 以下语句均为 true:

- 您正在使用 CP4I 许可证。
- IBM MQ Console 已启用。
- 您正在将 IBM Cloud Pak for Integration Keycloak 服务用于 IBM MQ Console 单点登录 (SSO) 认证和授权 (缺省值)。

如果先前的所有语句都为 true ，那么 SSO 由 Keycloak 提供。 因此，对于 IBM MQ Operator 目录源，您还必须对以下每个其他必需组件重复这些步骤:

- IBM Cloud Pak foundational services
- IBM Cloud Pak for Integration
- Keycloak (Red Hat OpenShift 运算符)

请遵循 将目录源添加到集群中针对所需目录源的步骤。

5. 安装 IBM MQ Operator。

   选择以下两个选项之一 (使用控制台或 CLI):

   - 选项 1: 使用 OpenShift 控制台安装 IBM MQ Operator。
   - 选项 2: 使用 OpenShift CLI 安装 IBM MQ Operator。

6. 创建权利密钥密钥 (仅联机安装)。

   IBM MQ Operator 部署从执行许可证权利检查的容器注册表中提取的队列管理器映像。 此检查需要存储在 `docker-registry` 拉取私钥中的权利密钥。 如果在将安装队列管理器的名称空间中还没有权利密钥，请遵循以下指示信息以获取权利密钥并创建拉取私钥。

   注: 如果将仅部署 IBM MQ Advanced for Developers (非 Warranted) 队列管理器，那么不需要权利密钥。

   您可以使用 OpenShift 控制台或 CLI 来创建权利密钥私钥。 以下示例使用 CLI:

   a. 获取分配给 IBM 标识的权利密钥。 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。

   c. 在 OpenShift CLI 中，运行以下命令以创建名为 `ibm-entitlement-key` 的映像拉取私钥。

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

   其中， *entitlement lement_key* 是您在步骤 b 中复制的权利密钥， *user_email>* 是与授权软件关联的 IBM 标识， *namespace* 是您将 IBM MQ Operator 安装到的名称空间。

7. 可选： 安装 CP4I 及其依赖关系。

   在部署至少一个队列管理器时，存在一些其他必需组件，其中 **所有** 以下语句均为 true:

   - 您正在使用 CP4I 许可证。
   - IBM MQ Console 已启用。
   - 您正在将 CP4I Keycloak 服务用于 IBM MQ Console 单点登录 (SSO) 认证和授权 (缺省值)。

   如果先前的所有语句都为 true ，那么 SSO 由 Keycloak 提供，您必须完成以下其他步骤:

   - 以与 CP4I 操作程序相同的安装方式安装 IBM Cloud Pak foundational services 操作程序。 有关受支持的版本，请参阅 此发行版的操作员通道版本。
   - 安装 CP4I 操作程序。
   - 可选: 部署平台 UI。

     a. 创建 `ibm-common-services` 名称空间。 通过 CLI 登录到 OpenShift 集群时，请运行以下命令:

     ```
     oc new-project ibm-common-services
     ```

     b. 部署平台 UI。

8. 部署 License Service。

   这是监视队列管理器的许可证使用情况所必需的。 遵循 部署 License Service 中的指示信息。

9. 部署队列管理器。

   有关部署示例 “快速启动” 队列管理器的指示信息，请参阅 第 85 页的『使用 IBM MQ Operator 部署简单队列管理器』。

相关任务

第 73 页的『卸载 IBM MQ Operator』
您可以使用 Red Hat OpenShift 控制台或 CLI 从 Red Hat OpenShift 卸载 IBM MQ Operator 。

OpenShift CP4I 升级 IBM MQ Operator 和队列管理器

IBM MQ Operator 的用户有不同的升级过程，具体取决于您是使用 IBM MQ 许可证还是 IBM Cloud Pak for Integration (CP4I) 许可证。 完成部署类型的升级步骤。

关于此任务

要升级 IBM MQ Operator 和队列管理器，请完成下列其中一个步骤:

过程

- 选项 1: **将部署升级到当前操作程序通道上的最新版本**。

  要将 IBM MQ Operator 的部署升级到当前操作程序通道上的最新版本，请参阅 第 61 页的『升级到 IBM MQ Operator 通道最新安全性发行版』。
- 选项 2: **升级 IBM MQ 许可证的 IBM MQ Operator** 。

  要升级使用 **只** IBM MQ 许可证的 IBM MQ Operator 的部署，请参阅 第 61 页的『升级 IBM MQ Operator』。
- 选项 3: **为 CP4I 用户升级 IBM MQ Operator** 。

  为 IBM Cloud Pak for Integration 的用户升级 IBM MQ Operator 的部署。 这包括如果您已在 CP4I 许可证下至少部署了一个队列管理器。 请参阅 第 69 页的『为 CP4I 用户升级 IBM MQ Operator』。

OpenShift 升级 *IBM MQ Operator*

升级使用 **仅** IBM MQ 许可证的 IBM MQ Operator 的部署。

开始之前

要点: 此任务适用于 IBM MQ Operator 和 **只** IBM MQ 许可证的用户。 如果您是 IBM Cloud Pak for Integration (CP4I) 用户，或者已使用 CP4I 许可证至少部署了一个队列管理器，请参阅 第 69 页的『为 CP4I 用户升级 IBM MQ Operator』。

关于此任务

完成以下步骤中与您需要的升级相匹配的任何步骤。

注:

- Version 3.6.0 of the IBM MQ Operator is the latest CD release.
- Version 3.2.13 of the IBM MQ Operator is the latest SC2 release.

过程

- 选项 1: 第 61 页的『升级到 IBM MQ Operator 通道最新安全性发行版』
- 选项 2: 第 63 页的『升级到 IBM MQ Operator 的 3.2.x SC2 频道』
- 选项 3: 第 65 页的『升级到当前 CD 的 IBM MQ Operator 频道』

OpenShift CP4I 升级到 *IBM MQ Operator* 通道最新安全性发行版

升级 IBM MQ Operator 允许您升级队列管理器。

开始之前

要点: 本主题用于将 IBM MQ Operator 的部署升级到部署的通道上的最新安全发行版。 如果这不适用于您的部署，请参阅 第 61 页的『升级 IBM MQ Operator 和队列管理器』中描述的备用升级路径。

关于此任务

根据用于部署要升级的 IBM MQ Operator 的目录源，有不同的程序。

选项 1: IBM MQ Operator 的特定目录源

更新目录源后，新的 "IBM MQ Operator 版本将在 "OpenShift 群集 "**只是**中可用。 此过程有效地使您能够手动控制升级，因此您不需要将 **手动** 选项用于操作程序的 **Update approval** 设置。 **手动** 选项强制同时执行所有可能的升级，并且可以阻止升级，因此仅使用 **自动** 选项。

要使用此选项，请跳至 使用 IBM MQ Operator 的特定目录源进行升级。

选项 2: IBM 操作程序目录

通过此选项，新的操作程序版本将变为可用， **无需** 来自您的任何干预即可应用。 因此，对于要 **自动** 升级 IBM MQ Operator 且不需要确定性安装的联机安装，请 **仅** 使用此选项。 此选项对于概念验证环境很有用，但它 **不适用于生产环境**。

要使用此选项，请跳至 使用 IBM 操作程序目录进行升级。

要从使用 IBM 操作程序目录移至使用 IBM MQ Operator 的特定目录源 (这使您能够更好地控制升级) ，请参阅 第 68 页的『移至 IBM MQ Operator 的特定目录源』。

过程

- **使用 IBM MQ Operator 的特定目录源进行升级**

  a) 镜像（仅限气隙）。

  如果您的集群处于受限（无连接）网络环境中，则必须镜像 IBM MQ 镜像。

  要点: 从版本 3.5.0 开始， IBM MQ CASE 发生了变化，现在只包含一个队列管理器版本。 镜像过程运行速度更快，但可能需要多次运行；每次运行一次，用于安装每个版本的队列管理器。

  有关更多信息，请参阅第 261 页的『在 "Red Hat OpenShift 上进行空隙安装和升级的 IBM MQ CASE』。

  您必须查找 IBM MQ CASE 版本中包含您需要的操作员和队列管理器版本，并分别重复气隙镜像过程。 查看 IBM MQ CASE 查找表 ， 了解版本对应关系。

  例如，当按照以下链接的镜像说明进行操作时，您可以通过使用以下值来安装最新的操作员和单个对应的队列管理器版本：

  ```
  export OPERATOR_PACKAGE_NAME=ibm-mq
  export OPERATOR_VERSION=3.6.0
  ```

  有关用于间隙安装和升级的 IBM MQ CASE 内容的信息，请参阅 "第 261 页的『在 "Red Hat OpenShift 上进行空隙安装和升级的 IBM MQ CASE』。

  要创建镜像，请参阅为空气屏蔽群集创建镜像。

  b) 应用最新的目录源。

  遵循 "添加 IBM MQ Operator 目录源" 中的 "为 IBM MQ Operator 添加特定目录源"下的指示信息。

  c) 升级 IBM MQ Operator。

  如果 IBM MQ Operator 的**更新审批**状态设置为 "**自动"**，操作员就会升级。 否则，请手动升级 IBM MQ Operator：

  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 **升级可用**

  e. 单击 **预览 InstallPlan**

  f. 单击 "**批准**"完成升级

  操作程序升级到新版本。

  d) 升级任何 IBM MQ 队列管理器。

请按照升级 IBM MQ 队列管理器中的说明进行操作。

- **使用 IBM 操作程序目录进行升级**

  a) 将 IBM MQ Operator 升级到更高版本。

  如果设置了自动升级，那么在发布新的安全性发布时， IBM MQ Operator 将完成升级。 否则，请手动批准 IBM MQ Operator 升级：

  – 如果有可用的升级，那么 **Upgrade Status** 可能是 "**升级可用**"。
  – 在这种情况下，可能存在可用于核准用于升级 IBM MQ Operator 的 **InstallPlan** 的可用控件。

  b) 升级任何 IBM MQ 队列管理器

  请按照升级 IBM MQ 队列管理器中的说明进行操作。

- **升级 IBM MQ 队列管理器**。

  在升级 IBM MQ Operator 之后，应该将任何 IBM MQ 队列管理器升级到更高版本。 下表描述了每个活动操作员通道的 IBM MQ 队列管理器的最新版本。 使用相关版本，按照第 72 页的『使用 Red Hat OpenShift 升级 IBM MQ 队列管理器』 中的程序操作。

操作员通道	最新的 IBM MQ 队列管理器
3.2SC2)	9.4.0.11-r3
3.6CD)	9.4.3.0-r1

OpenShift CP4I 升级到 *IBM MQ Operator* 的 *3.2.x SC2* 频道

升级**只**使用 IBM MQ Operator 许可证的 IBM MQ 部署。

开始之前

要点:

- 此任务适用于 IBM MQ Operator 且**仅** IBM MQ 许可证的用户。 如果您是 IBM Cloud Pak for IntegrationCP4I) 用户，或已使用 CP4I 许可证部署了至少一个队列管理器，请参阅第 69 页的『为 CP4I 用户升级 IBM MQ Operator』。
- This task is for upgrading deployments of the IBM MQ Operator to the Support Cycle 2 (SC2) channel of IBM MQ Operator 3.2.x **只是**. 请注意，这也是升级到最新 CD 版本 (3.6.0) 时的潜在中间步骤。 如果这不适用于您的部署，请参阅第 61 页的『升级 IBM MQ Operator 和队列管理器』 中描述的替代升级路径。

关于此任务

根据您要升级到 3.2.x SC2 频道的 IBM MQ Operator 部署情况，完成以下选项之一：

过程

- **方案 1**：升级 2.0.x Long Term SupportLTS)IBM MQ Operator 的部署。

  遵循第 63 页的『将 2.0.x LTS IBM MQ Operator 升级到 3.2.x SC2 通道』中的步骤。

- **方案 2**：升级 CD 的 IBM MQ Operator 部署。

  遵循第 64 页的『将 CD IBM MQ Operator 升级到 v3.2.x SC2 频道』中的步骤。

OpenShift V 9.4.0 CP4I-SC2 将 *2.0.x LTS IBM MQ Operator* 升级到 *3.2.x SC2* 通道

升级 IBM MQ Operator 允许您升级队列管理器。

开始之前

要点:

- 此任务适用于 IBM MQ Operator 和 **只** IBM MQ 许可证的用户。 如果您是 IBM Cloud Pak for Integration (CP4I) 用户，或者已使用 CP4I 许可证至少部署了一个队列管理器，请参阅 第 69 页的『为 CP4I 用户升级 IBM MQ Operator』。
- 本主题用于将 2.0.x Long Term Support (LTS) IBM MQ Operator 的部署升级到 IBM MQ Operator 3.2.x **仅**的 Support Cycle 2 (SC2) 通道。 如果这不适用于您的部署，请参阅 第 61 页的『升级 IBM MQ Operator 和队列管理器』中描述的备用升级路径。

要升级到 IBM MQ Operator 3.2.13 ，您必须正在运行 Red Hat OpenShift Container Platform 4.12 或更高版本。 要验证每个 IBM MQ Operator 通道的兼容版本，请参阅 第 33 页的『兼容的 Red Hat OpenShift Container Platform 版本』。 要升级平台，请参阅 升级 Red Hat OpenShift。

过程

1. **镜像映像 (仅限气郄)**。

   如果您的集群处于受限（无连接）网络环境中，则必须镜像 IBM MQ 镜像。

   要点: 从版本 3.5.0 开始， IBM MQ CASE 发生了变化，现在只包含一个队列管理器版本。 镜像过程运行速度更快，但可能需要多次运行；每次运行一次，用于安装每个版本的队列管理器。

   有关更多信息，请参阅第 261 页的『在 "Red Hat OpenShift 上进行空隙安装和升级的 IBM MQ CASE』。

   您必须查找 IBM MQ CASE 版本中包含您需要的操作员和队列管理器版本，并分别重复气隙镜像过程。 查看 IBM MQ CASE 查找表 ，了解版本对应关系。

   例如，当按照以下链接的镜像说明进行操作时，您可以通过使用以下值来安装最新的操作员和单个对应的队列管理器版本：

   ```
   export OPERATOR_PACKAGE_NAME=ibm-mq
   export OPERATOR_VERSION=3.6.0
   ```

   您应该省略 3.5 部分 "配置集群" ，因为在先前安装或升级期间应该已设置与映像注册表的连接。

   链接: 气郄集群的镜像映像。
2. **将 IBM MQ Operator 升级到 3.2.13**。

   请参阅 第 71 页的『使用 Red Hat OpenShift 升级 IBM MQ Operator』。
3. **升级实例**。

   要接收最新功能部件和安全修订，请将 IBM MQ Operand (队列管理器容器映像) 升级到最新 SC2 版本 (9.4.0.11)。 请参阅 第 72 页的『使用 Red Hat OpenShift 升级 IBM MQ 队列管理器』。

OpenShift CP4I V 9.4.0 将 *CD IBM MQ Operator* 升级到 *v3.2.x SC2* 频道

升级 IBM MQ Operator 允许您升级队列管理器。

开始之前

要点:

- 此任务适用于 IBM MQ Operator 和 **只** IBM MQ 许可证的用户。 如果您是 IBM Cloud Pak for Integration (CP4I) 用户，或者已使用 CP4I 许可证至少部署了一个队列管理器，请参阅 第 69 页的『为 CP4I 用户升级 IBM MQ Operator』。
- 此任务**仅**用于将 3.2.0 之前版本 "IBM MQ Operator 的 "Continuous Delivery（CD）部署升级到版本 "3.2.13。 请注意，在升级到最新的 "CD 版本（v3.6.0）时，这也可能是一个中间步骤。 如果这不适用于您的部署，请参阅 "第 61 页的『升级 IBM MQ Operator 和队列管理器』中描述的其他升级路径。

要升级到 IBM MQ Operator 3.2.13 ，您必须正在运行 Red Hat OpenShift Container Platform 4.12 或更高版本。 要验证每个 IBM MQ Operator 通道的兼容版本，请参阅 第 33 页的『兼容的 Red Hat OpenShift Container Platform 版本』。 要升级平台，请参阅 升级 Red Hat OpenShift。

过程

1. 可选： **升级当前处于 CD 版本 (低于 3.0.0) 的 IBM MQ Operator** 。

   如果 IBM MQ Operator 当前为 CD 版本 (低于 3.0.0)，请遵循 迁移到 IBM MQ 操作程序的当前 CD 通道 (IBM MQ 9.3 文档)中的相关步骤，然后返回此处以升级到最新的 CD 版本。 请注意，这是升级到 V 3.2.13 之前的必需先决条件步骤。
2. **升级到您当前频道的最新运营商**

   遵循第 61 页的『升级到 IBM MQ Operator 通道最新安全性发行版』中的步骤。
3. **镜像映像 (仅限气郄)**。

   如果您的集群处于受限（无连接）网络环境中，则必须镜像 IBM MQ 镜像。

   要点: 从版本 3.5.0 开始， IBM MQ CASE 发生了变化，现在只包含一个队列管理器版本。 镜像过程运行速度更快，但可能需要多次运行；每次运行一次，用于安装每个版本的队列管理器。

   有关更多信息，请参阅第 261 页的『在 "Red Hat OpenShift 上进行空隙安装和升级的 IBM MQ CASE』。

   您必须查找 IBM MQ CASE 版本中包含您需要的操作员和队列管理器版本，并分别重复气隙镜像过程。 查看 IBM MQ CASE 查找表 ， 了解版本对应关系。

   例如，当按照以下链接的镜像说明进行操作时，您可以通过使用以下值来安装最新的操作员和单个对应的队列管理器版本：

   ```
   export OPERATOR_PACKAGE_NAME=ibm-mq
   export OPERATOR_VERSION=3.6.0
   ```

   您应该省略 3.5 部分 "配置集群" ，因为在先前安装或升级期间应该已设置与映像注册表的连接。

   链接: 气郄集群的镜像映像。
4. **将 IBM MQ Operator 升级到 3.2.13**。

   请参阅 第 71 页的『使用 Red Hat OpenShift 升级 IBM MQ Operator』。
5. **升级实例**。

   要接收最新的功能部件和安全修订，请将 IBM MQ Operand (队列管理器容器映像) 升级到最新的 SC2 版本 (9.4.0.11)。 请参阅 第 72 页的『使用 Red Hat OpenShift 升级 IBM MQ 队列管理器』。

OpenShift 升级到当前 *CD* 的 *IBM MQ Operator* 频道
在使用 "**只是**"IBM MQ 许可证的情况下升级 "IBM MQ Operator 的部署。

开始之前

要点:

- 此任务适用于 IBM MQ Operator 且**仅** IBM MQ 许可证的用户。 如果您是 IBM Cloud Pak for IntegrationCP4I) 用户，或已使用 CP4I 许可证部署了至少一个队列管理器，请参阅第 69 页的『为 CP4I 用户升级 IBM MQ Operator』。
- 本主题用于升级到 IBM MQ Operator(3.6)**只是** 的 Continuous Delivery(CD) 频道。 如果这不适用于您的部署，请参阅第 61 页的『升级 IBM MQ Operator 和队列管理器』 中描述的替代升级路径。

关于此任务

Complete one of the following options, depending on the deployment of the IBM MQ Operator that you are upgrading to the 3.6 CD channel:

过程

- **方案 1：** 升级 3.2.x'SC2'IBM MQ Operator 的部署。

  遵循第 66 页的『将 3.2.x 的 "SC2"IBM MQ Operator 升级到当前的 "CD 通道』中的步骤。

- **方案 2：** 将 "CD 部署升级为 "IBM MQ Operator。

  遵循第 67 页的『将 "CD"IBM MQ Operator"升级为当前的 "CD 通道』中的步骤。

OpenShift 将 *3.2.x* 的 *"SC2"IBM MQ Operator* 升级到当前的 *"CD* 通道
将 3.2.x 的 "SC2 部署升级到**只**使用 "IBM MQ 许可证的 "IBM MQ Operator。

开始之前

要点:

- 此任务适用于 "IBM MQ Operator 和**仅**"IBM MQ 许可证的用户。 如果您是 "IBM Cloud Pak for Integration（CP4I）用户，或者您已使用 "CP4I 许可证部署了至少一个队列管理器，请参阅 "第 69 页的『为 CP4I 用户升级 IBM MQ Operator』。
- 此任务**仅**用于将 3.2.x 版本 "SC2 的 "IBM MQ Operator 升级到当前 "Continuous Delivery（CD"）通道的 "IBM MQ Operator（v "3.6）。 如果这不适用于您的部署，请参阅 "第 61 页的『升级 IBM MQ Operator 和队列管理器』中描述的替代升级路径。

要升级到 "IBM MQ Operator""3.6.0"，您必须运行 "Red Hat OpenShift Container Platform 4.12 或更高版本。 要验证每个 "IBM MQ Operator 通道的兼容版本，请参阅 "第 33 页的『兼容的 Red Hat OpenShift Container Platform 版本』。 To upgrade the platform, see 升级 "Red Hat OpenShift in the IBM Cloud Pak for Integration documentation.

过程

1. 可选： **升级当前版本为 3.2.0 之前的 "IBM MQ Operator**。

   如果您的 "IBM MQ Operator 目前是 3.2.0 之前的版本，请按照 "第 63 页的『升级到 IBM MQ Operator 的 3.2.x SC2 频道』中的相关步骤操作，然后返回此处升级到最新的 "CD 版本。 Note that if you are already on a v3.2.x version of the IBM MQ Operator you can skip this step, however upgrading to the v3.2 channel of the IBM MQ Operator is a mandatory prerequisite step before upgrading to version 3.6.0.
2. **升级到您当前频道的最新运营商**

   遵循第 61 页的『升级到 IBM MQ Operator 通道最新安全性发行版』中的步骤。
3. **镜像（仅限气隙）**。

   如果您的集群处于受限（无连接）网络环境中，则必须镜像 IBM MQ 镜像。

   要点: 从版本 3.5.0 开始， IBM MQ CASE 发生了变化，现在只包含一个队列管理器版本。 镜像过程运行速度更快，但可能需要多次运行；每次运行一次，用于安装每个版本的队列管理器。

   有关更多信息，请参阅第 261 页的『在 "Red Hat OpenShift 上进行空隙安装和升级的 IBM MQ CASE』。

   您必须查找 IBM MQ CASE 版本中包含您需要的操作员和队列管理器版本，并分别重复气隙镜像过程。 查看 IBM MQ CASE 查找表 ， 了解版本对应关系。

   例如，当按照以下链接的镜像说明进行操作时，您可以通过使用以下值来安装最新的操作员和单个对应的队列管理器版本：

   ```
   export OPERATOR_PACKAGE_NAME=ibm-mq
   export OPERATOR_VERSION=3.6.0
   ```

   应省略第 3.5 节 "配置群集"，因为与映像注册表的连接应在之前的安装或升级过程中设置好。

   有关用于间隙安装和升级的 IBM MQ CASE 内容的信息，请参阅 "第 261 页的『在 "Red Hat OpenShift 上进行空隙安装和升级的 IBM MQ CASE』。

   链接： 空气盖群集的镜像图像
4. **将 "IBM MQ Operator 升级为 "3.6.0**。

   请遵循第 71 页的『使用 Red Hat OpenShift 升级 IBM MQ Operator』中的指示信息。

5. **升级实例**。

   要获得最新功能和安全修复，请将 IBM MQ 操作符（队列管理器容器映像）升级到最新的 CD 版本 (9.4.3.0-r1)。请参见 "第 72 页的『使用 Red Hat OpenShift 升级 IBM MQ 队列管理器』。

OpenShift 将 *"CD"IBM MQ Operator"*升级为当前的 *"CD* 通道
将 "CD 部署升级为**只**使用 "IBM MQ 许可证的 "IBM MQ Operator 部署。

开始之前

要点:

- 此任务适用于 "IBM MQ Operator 和**仅**"IBM MQ 许可证的用户。 如果您是 "IBM Cloud Pak for Integration（CP4I）用户，或者您已使用 "CP4I 许可证部署了至少一个队列管理器，请参阅 "第 69 页的『为 CP4I 用户升级 IBM MQ Operator』。
- 此任务**仅**用于将 "IBM MQ Operator 的 "Continuous Delivery（CD）版本升级到 "IBM MQ Operator（v3.6）的当前 "CD 通道。 如果这不适用于您的部署，请参阅 "第 61 页的『升级 IBM MQ Operator 和队列管理器』中描述的替代升级路径。

要升级到 "IBM MQ Operator""3.6.0"，您必须运行 "Red Hat OpenShift Container Platform 4.12 或更高版本。 要验证每个 "IBM MQ Operator 通道的兼容版本，请参阅 "第 33 页的『兼容的 Red Hat OpenShift Container Platform 版本』。 To upgrade the platform, see 升级 "Red Hat OpenShift in the IBM Cloud Pak for Integration documentation.

过程

1. 可选： **升级当前版本为 3.2.0 之前的 "IBM MQ Operator**。

   如果您的 "IBM MQ Operator 目前是 3.2.0 之前的版本，请按照 "第 63 页的『升级到 IBM MQ Operator 的 3.2.x SC2 频道』中的相关步骤操作，然后返回此处按照 "第 67 页的『2』步骤升级到最新的 "CD 版本。 Note that if you are already on a 3.2.x version of the IBM MQ Operator you can skip this step, however upgrading to the v3.2 channel of the IBM MQ Operator is a mandatory prerequisite step before upgrading to version 3.6.0.
2. 可选： **升级当前版本为 "3.2.x IBM MQ Operator**。

   如果您的 "IBM MQ Operator 目前处于 "3.2.x 版本，请按照 "第 66 页的『将 3.2.x 的 "SC2"IBM MQ Operator 升级到当前的 "CD 通道』中的相关步骤操作。
3. **升级当前版本为 3.3.0 或更高版本的 IBM MQ Operator** 。

   如果您的 IBM MQ Operator 当前版本为 3.3.0 或更高版本，请完成以下子步骤升级到 IBM MQ Operator 3.6.0。

   a) **升级到您当前频道的最新运营商**

   遵循第 61 页的『升级到 IBM MQ Operator 通道最新安全性发行版』中的步骤。

   b) **升级到下一个运营商频道**。

   i) 镜像（仅气隙）。

   如果您的集群处于受限（无连接）网络环境中，则必须镜像 IBM MQ 镜像。 仅使用这些值完成以下链接中的步骤：

   ```
   export OPERATOR_PACKAGE_NAME=ibm-mq
   export OPERATOR_VERSION=[operator_version]
   ```

   其中*[operator_version]* 是您要升级到的 IBM MQ Operator 版本。 要确定您要升级到的频道中的最新 IBM MQ Operator ，请查看 第 5 页的『IBM MQ Operator 的发布历史记录』。

   应省略第 3.5 节 "配置群集"，因为与映像注册表的连接应在之前的安装或升级过程中设置好。

   有关用于间隙安装和升级的 IBM MQ CASE 内容的信息，请参阅 "第 261 页的『在 "Red Hat OpenShift 上进行空隙安装和升级的 IBM MQ CASE』。

链接： 空气盖群集的镜像图像

ii) 升级您的 IBM MQ Operator。

请遵循第 71 页的『使用 Red Hat OpenShift 升级 IBM MQ Operator』中的指示信息。

iii) 升级实例。

要获取最新功能和安全补丁，请升级 IBM MQ 操作数（队列管理器容器映像）。 请参阅 第 72 页的『使用 Red Hat OpenShift 升级 IBM MQ 队列管理器』。

c) 重复之前的子步骤（a）和（b），直到到达最新的 CD 频道（v3.6.0 ）。

OpenShift 移至 *IBM MQ Operator* 的特定目录源

如果从先前发行版安装了 IBM MQ Operator，并且正在使用 IBM 操作程序目录，那么应用特定目录源是在集群上完全控制软件版本控制的最有效方法。

开始之前

要点: 此任务必须由集群管理员执行。 请参阅 OpenShift 角色和许可权。

使用 CLI 完成以下步骤。

关于此任务

IBM 操作程序目录是可用于扩展 Red Hat OpenShift Container Platform 集群的 API 以启用 IBM 软件产品的操作程序的索引。

此过程将从 IBM 操作程序目录中移动 IBM MQ Operator 的安装，以便您可以将特定目录源用于 IBM MQ Operator。

过程

1. 添加 IBM MQ Operator 目录。

   遵循 "添加 IBM MQ Operator 目录源" 中的 "为 IBM MQ Operator 添加特定目录源"下的指示信息。
2. 确认已在 `openshift-marketplace` 名称空间中创建 IBM MQ Operator 目录源。

   运行以下命令：

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

   示例输出如下所示：

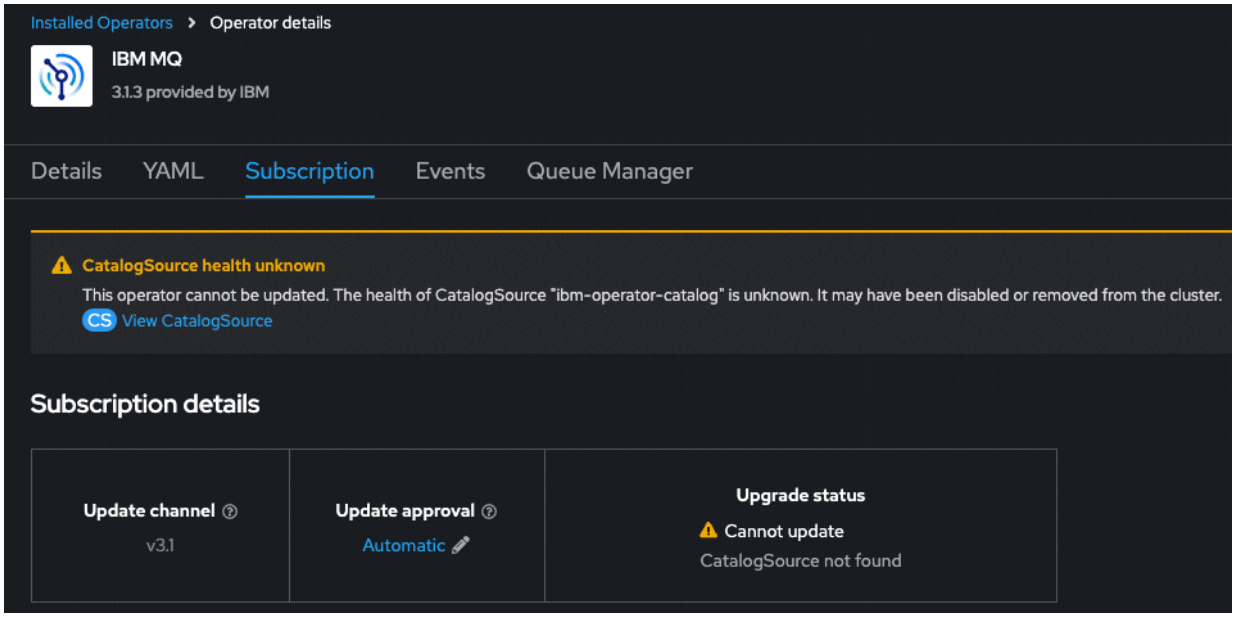
   ```
   oc get catalogsource -n openshift-marketplace

   NAME                           DISPLAY                TYPE   PUBLISHER   AGE
   ibm-operator-catalog           IBM Operator Catalog   grpc   IBM         23h
   ibmmq-operator-catalogsource   ibm-mq-3.1.3           grpc   IBM         23h
   ```
3. 可选： 删除 IBM 操作程序目录源。

   警告: 仅当您确定没有其他操作程序使用 IBM 操作程序目录时，才应完成此步骤。

   运行以下命令：

   ```
   oc delete catalogsource ibm-operator-catalog -n openshift-marketplace
   ```

   IBM MQ Operator 状态将更改为 `CatalogSource not found`。 这是期望的行为。



4. 更改 IBM MQ Operator 的预订以指向新的特定 IBM MQ Operator 目录源。
   a) 编辑预订。

      运行以下命令，将 *OPERATOR-NAMESPACE* 替换为 `openshift-operators` 以用于 IBM MQ Operator 的集群范围安装，或用于部署 IBM MQ Operator 的特定名称空间:

      ```
      oc edit subscription ibm-mq -n OPERATOR-NAMESPACE
      ```

   b) 将 `spec.source` 值从 `ibm-operator-catalog` 更改为步骤 第 68 页的『1』中创建的目录源的名称。

      例如:

      ```
      spec:
        channel: v3.1
        installPlanApproval: Automatic
        name: ibm-mq
        source: ibm-operator-catalog # CHANGE --> ibmmq-operator-catalogsource
        sourceNamespace: openshift-marketplace
      ```

   c) 保存更改。

      IBM MQ Operator 安装现在指向 IBM MQ Operator 目录源。 如果删除了 IBM 操作员目录，状态将"CatalogSource"未找到 "恢复为 "成功"。

结果

IBM MQ Operator 的安装现在指向 IBM MQ Operator 的特定目录源。 这使您能够完全控制对操作程序的升级。

OpenShift CP4I 为 *CP4I 用户升级 IBM MQ Operator*

升级使用 IBM Cloud Pak for Integration (CP4I) 许可证的 IBM MQ Operator 的部署。

开始之前

要点: 此任务适用于 CP4I 用户。 这包括如果您已在 CP4I 许可证下至少部署了一个队列管理器。 如果这不适用于您，请参阅 第 61 页的『升级 IBM MQ Operator』。

关于此任务

如果要升级到 CD 版本的 IBM MQ Operator （版本 3.6.0 ），则必须按照以下说明首先升级到 v3.2 SC2 频道。

如果您的 IBM MQ Operator 是 3.3.0 或更高版本，请跳转至 Upgrade to the CD Channel 步骤中的子步骤（c）。

过程

1. 可选： **升级至 v3.2 SC2 频道**。

   Complete one or more of the following options to upgrade your IBM MQ Operator to the Support Cycle 2 (SC2) channel of the IBM MQ Operator **只是**

   - **方案 1：** 升级 2.0.x Long Term SupportLTS)IBM MQ Operator 的部署。

     执行 通过生成升级计划从 2022.2 中的步骤。
   - **方案 2：** 升级 IBM MQ Operator 的 3.0.x 或 3.1.x 部署。

     遵循 通过生成升级计划从 2023.4 中的步骤。
   - **方案 3：** 升级 IBM MQ Operator 的其他部署。

     按照迁移到 IBM MQ Operator 的当前 CD 频道 IBM MQ 9.3 文件） 中的相关步骤操作，然后返回此处，继续执行**方案 2**。 请注意，这是必需的先决条件步骤。
2. **升级到 CD 频道 (3.6*)**。

   a) 可选： **升级到 3.2.x 部署 IBM MQ Operator**

   按照前面的主要步骤，将 "IBM MQ Operator 升级到 3.2.x 版本。

   b) 可选： **升级 "IBM MQ Operator 的 3.2.x 部署**

   请按照 "通过生成升级计划从 "16.1.0 升级中的步骤操作。

   c) 可选： **升级 3.3.x 部署的 IBM MQ Operator**

   如果您的 IBM MQ Operator 当前版本为 3.3.x ， 请完成以下步骤将操作员升级到 3.4.x ，然后再继续升级到 3.6.0。

   i) 镜像（仅气隙）。

   如果您的集群处于受限（无连接）网络环境中，则必须镜像 IBM MQ 镜像。 请仅使用以下数值，完成以下链接中的步骤：

   ```
   export OPERATOR_PACKAGE_NAME=ibm-mq
   export OPERATOR_VERSION=3.4.1
   ```

   您应该跳过 3.5 “配置集群”部分，因为与镜像注册表的连接应该在之前的安装或升级过程中已经建立。

   有关用于气隙安装和升级的 IBM MQ CASE 的内容，请参见 第 261 页的『在 "Red Hat OpenShift 上进行空隙安装和升级的 IBM MQ CASE』。

   链接： 空气间隙集群的镜像。

   ii) 升级您的 IBM MQ Operator。

   请遵循第 71 页的『使用 Red Hat OpenShift 升级 IBM MQ Operator』中的指示信息。

   iii) 升级实例。

   要获取最新功能和安全补丁，请升级 IBM MQ 操作数（队列管理器容器映像）。 请参阅 第 72 页的『使用 Red Hat OpenShift 升级 IBM MQ 队列管理器』。

   d) **升级 3.4.0 或更高版本的 IBM MQ Operator**

   i) 镜像（仅气隙）。

   如果您的集群处于受限（无连接）网络环境中，则必须镜像 IBM MQ 镜像。 请仅使用以下数值，完成以下链接中的步骤：

   ```
   export OPERATOR_PACKAGE_NAME=ibm-mq
   export OPERATOR_VERSION=[operator_version]
   ```

其中，*[operator_version]* 是您要升级到的 IBM MQ Operator 版本。 要确定您要升级到的频道中的最新 IBM MQ Operator，请查看 第 5 页的『IBM MQ Operator 的发布历史记录』。

您应该跳过 3.5 “配置集群”部分，因为与镜像注册表的连接应该在之前的安装或升级过程中已经建立。

有关用于气隙安装和升级的 IBM MQ CASE 的内容，请参见 第 261 页的『在 "Red Hat OpenShift 上进行空隙安装和升级的 IBM MQ CASE』。

链接： 空气间隙集群的镜像。

ii) 升级到当前频道的最新操作员。

遵循第 61 页的『升级到 IBM MQ Operator 通道最新安全性发行版』中的步骤。

iii) 升级 IBM MQ Operator 部署。

按照从 16.1.1 升级中的步骤生成升级计划。

OpenShift CP4I 使用 *Red Hat OpenShift* 升级 *IBM MQ Operator*

IBM MQ Operator 可通过 Red Hat OpenShift 控制台或命令行进行升级。

开始之前

注:

- CD 的最新版本 IBM MQ Operator 是 3.6.0。
- SC2 的最新版本 IBM MQ Operator 是 3.2.13。

有关 IBM MQ Operator 的最新发布说明，请参见 IBM MQ Operator 的发布历史。

过程

- **方案 1：使用 Red Hat OpenShift 控制台升级 IBM MQ Operator** 。
  a) 登录 "Red Hat OpenShift 群集控制台。
  b) 镜像（仅限气隙）。

  如果群集处于受限（空气屏蔽）网络环境中，则必须使用以下值镜像 "IBM MQ 映像:

  ```
  export OPERATOR_PACKAGE_NAME=ibm-mq
  export OPERATOR_VERSION=3.6.0
  ```

  有关用于间隙安装和升级的 IBM MQ CASE 内容的信息，请参阅 "第 261 页的『在 "Red Hat OpenShift 上进行空隙安装和升级的 IBM MQ CASE』。

  要创建镜像，请参阅为空气屏蔽群集创建镜像。

  c) 应用最新的《目录》来源。

  如果使用的是 "IBM MQ Operator 的特定目录源，而不是 "`ibm-operator-catalog`，则必须应用新版本 "IBM MQ 的目录源。

  要从使用 "IBM 操作员目录转向使用 "IBM MQ Operator 的特定目录源，并获得对升级的更大控制，请参阅 "第 68 页的『移至 IBM MQ Operator 的特定目录源』中的步骤，然后返回完成下一步。

  如果使用的是 "IBM 操作员目录（仅限某些在线安装），请继续下一步。

  否则，请按照 "第 53 页的『添加 IBM MQ Operator 目录源』中的说明进行操作，然后进入下一步。

  d) 升级 IBM MQ Operator。 新的主要/次要 "IBM MQ Operator 版本通过新的订阅渠道提供。 要将运营商升级到新的主要/次要版本，您需要更新 "IBM MQ Operator 订阅中的选定通道。

  a. 从导航窗格中单击**操作符已安装的** > **操作符**。

  显示指定项目中所有已安装的操作符。

  b. 选择 **IBM MQ 操作员**。

c. 导航至 "**订阅**"选项卡。

d. 点击**频道**。

显示 "**更改订阅更新频道**"窗口。

e. 选择所需的频道，然后点击**保存**。

运营商升级到新渠道可用的最新版本。 请参阅 第 32 页的『IBM MQ Operator 的版本支持』。

- **选项 2：使用 Red Hat OpenShift 命令行升级 IBM MQ Operator** 。

a) 使用 "**oc login** 登录群集。

b) 镜像（仅限气隙）。

如果群集处于受限（空气屏蔽）网络环境中，则必须使用以下值镜像 "IBM MQ 映像：

```
export OPERATOR_PACKAGE_NAME=ibm-mq
export OPERATOR_VERSION=3.6.0
```

有关用于间隙安装和升级的 IBM MQ CASE 内容的信息，请参阅 "第 261 页的『在 "Red Hat OpenShift 上进行空隙安装和升级的 IBM MQ CASE』。

要创建镜像，请参阅为空气屏蔽群集创建镜像。

c) 应用最新的《目录》来源。

如果使用的是 "IBM MQ Operator 的特定目录源，而不是 "`ibm-operator-catalog`，则必须应用新版本 "IBM MQ 的目录源。

要从使用 "IBM 操作员目录转向使用 "IBM MQ Operator 的特定目录源，并获得对升级的更大控制，请参阅 "第 68 页的『移至 IBM MQ Operator 的特定目录源』中的步骤，然后返回完成下一步。

如果使用的是 "IBM 操作员目录（仅限某些在线安装），请继续下一步。

否则，请按照 "第 53 页的『添加 IBM MQ Operator 目录源』中的说明进行操作，然后进入下一步。

d) 升级 IBM MQ Operator。 新的主要或次要 "IBM MQ Operator 版本通过新的订阅渠道提供。 要将操作员升级到新的主版本或次要版本，请完成以下步骤更新 "IBM MQ Operator 订阅中的选定频道。

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. 对 "`Subscription` 进行修补，移动到所需的更新通道（其中 *vX.Y* 是上一步中确定的所需更新通道）。

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

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

An IBM MQ queue manager, deployed using the IBM MQ Operator, can be upgraded in Red Hat OpenShift using any of the following interfaces: Operator Hub; Command Line; IBM Cloud Pak for Integration Platform UI.

开始之前

在升级 IBM MQ 队列管理器的过程中，您可能已从 IBM Cloud Pak for Integration 文档发送到此主题。

注:

- The latest CD version of the IBM MQ queue manager is 9.4.3.0-r1.
- The latest SC2 version of the IBM MQ queue manager is 9.4.0.11-r3.

For the latest IBM MQ queue manager release notes, see 第 17 页的『队列管理器容器映像的发布历史』.

确保您的 IBM MQ Operator 达到所需的版本。 要升级您的 IBM MQ Operator，请参阅第 61 页的『升级 IBM MQ Operator』。

在气隙环境中升级队列管理器之前，必须通过升级 IBM MQ Operator 时完成的气隙特定步骤镜像最新 IBM Cloud Pak for Integration 映像。 有关这些步骤的示例，请参见第 65 页的『升级到当前 CD 的 IBM MQ Operator 频道』。

关于此任务

此任务为升级部署在 OpenShift 上的 IBM MQ 队列管理器提供了三个选项：

1. 使用 IBM MQ 控制台升级 Red Hat OpenShift 队列管理器。
2. 使用 IBM MQCLI 升级 Red Hat OpenShift 队列管理器
3. 使用平台用户界面在 Red Hat OpenShift 中升级 IBM MQ 队列管理器

过程

- **选项 1**：使用 IBM MQ 控制台升级 Red Hat OpenShift 队列管理器。
  a) 登录 Red Hat OpenShift 群集网络控制台。
  b) 从导航窗格中单击**操作符已安装的** > **操作符**。
  显示指定项目中所有已安装的操作符。
  c) 选择 **IBM MQ 操作员**。
  显示 **IBM MQ 操作员**窗口。
  d) 导航至**队列管理器**选项卡。
  显示**队列管理器详细信息**窗口。
  e) 选择要升级的队列管理器。
  f) 导航至 YAML 选项卡。
  g) 必要时更新以下字段，以匹配所需的 IBM MQ 队列管理器版本升级。
  – spec.version
  – spec.license.licence
  有关 IBM MQ Operator 版本和 IBM MQ 队列管理器容器映像的映射，请参阅第 17 页的『队列管理器容器映像的发布历史』。
  h) 保存更新后的队列管理器 YAML。
- **选项 2**：使用 IBM MQCLI 升级 Red Hat OpenShift 队列管理器
  a) 使用 Red Hat OpenShift 登录 `oc login` 命令行界面 (CLI)。
  b) 编辑 **QueueManager** 资源，必要时更新以下字段，以匹配所需的 IBM MQ 队列管理器版本升级。
  – spec.version
  – spec.license.licence
  有关通道与 IBM MQ Operator 版本和 IBM MQ 队列管理器版本的映射，请参阅第 32 页的『IBM MQ Operator 的版本支持』。
  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。
- **方案 3**: Upgrading an IBM MQ queue manager in Red Hat OpenShift using the Platform UI.
  请按照 IBM Cloud Pak for Integration 文档中的 "使用平台用户界面管理版本和升级 "中的相关说明进行操作。

OpenShift CP4I 卸载 IBM MQ Operator

您可以使用 Red Hat OpenShift 控制台或 CLI 从 Red Hat OpenShift 卸载 IBM MQ Operator 。

过程

- 选项 1: 使用 OpenShift 控制台卸载 IBM MQ Operator 。

  注: 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 2-6。

  a) 使用 Red Hat OpenShift Container Platform 集群管理凭证登录到 Red Hat OpenShift Container Platform Web 控制台。

  b) 将 **项目** 更改为要从中卸载 IBM MQ Operator 的名称空间。 从 **项目** 下拉列表中选择名称空间。

  c) 在导航窗格中，单击 **操作程序 > 已安装的操作程序**。

  d) 单击 **IBM MQ** 操作程序。

  e) 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。

  f) 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。

  g) 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 2-6。

  h) 返回到 **操作程序 > 已安装的操作程序**。

  i) 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**。

- 选项 2: 使用 OpenShift CLI 卸载 IBM MQ Operator

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

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

  a. 确保您位于包含要卸载的 IBM MQ Operator 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. 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:

  i) 通过遵循 IBM Cloud Pak foundational services 产品文档中的 卸载基础服务 中的指示信息，卸载公共服务操作程序。

  ii) 查看操作员组:

```
oc get operatorgroup
```

iii) 删除操作程序组:

```
oc delete OperatorGroup operator_group_name
```

c) 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. 可选: 如果没有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。 要执行此操作，请遵循 IBM Cloud Pak foundational services 产品文档中的 卸载基础服务 中的指示信息。

Kubernetes V 9.4.1 通过创建拉取秘钥，为使用 "Kubernetes 做准备

预构建的 "IBM MQ Advanced 映像可以从 "IBM Container Registry 中提取，但注册表会执行许可证权利检查。 该检查需要一个权限密钥，该密钥存储在 "docker-registry 提取密钥中。

开始之前

安装 "**kubectl** 命令行工具并登录 "Kubernetes 群集。

过程

1. 获取分配给 "IBMID 的权限密钥。
   a) 使用与软件相关联的 "IBMID 和密码登录 MyIBMContainer 软件库。
   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. 创建权限密钥
   a. 在 "OpenShiftCLI 中运行以下命令，创建名为 "ibm-entitlement-key 的图像提取秘密。

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

其中，*entitlement_key* 是在步骤 b 中复制的权限密钥，*user_email* 是与权限软件相关联的 "IBMID，*namespace* 是计划用于队列管理器的命名空间。

Kubernetes 准备使用 Kubernetes，安装 IBM License Service

通过容器许可，您只需许可单个 IBM MQ 容器的可用容量，而无需许可运行容器的整个服务器。 要利用容器许可，必须使用 IBM License Service 跟踪许可使用情况并确定所需的权利。

过程

按照 IBM Cloud Pak foundational services 文档中在 Kubernetes 集群上安装 License Service 的说明安装 IBM License Service 。

Linux V 9.4.1 准备通过调用预构建的容器映像来使用 "Podman 或 "Docker

可以使用命令行工具（如 "Podman 或 "Docker）从 "IBM Container Registry 中提取预制的 "IBM MQ Advanced 图像。

开始之前

安装 "Podman 或 "Docker.

注: 以下示例命令使用的是 "Podman，但也可以使用 "Podman 或 "Docker，命令参数相同。

过程

1. 获取分配给您的 IBM ID 的权限密钥。
   a) 使用与授权软件关联的 IBM ID 和密码登录 MyIBM 容器软件库。
   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. 登录容器注册表

   例如，使用 "Podman：

   ```
   podman login --username cp --password ... cp.icr.io
   ```

   其中"....." 被替换为上一步中复制的权限密钥。
3. 拉动图像

   例如，使用 "Podman：

   ```
   podman pull cp.icr.io/cp/ibm-mqadvanced-server:9.4.3.0-r1
   ```

通过构建您自己的容器映像来准备 IBM MQ

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 MQ Operator。 请参阅第 26 页的『选择要在容器中使用 IBM MQ 的方式』

关于此任务

过程

- 第 77 页的『构建您自己的队列管理器映像时的常规注意事项』
- 第 77 页的『构建样本 IBM MQ 队列管理器容器映像』
- 第 79 页的『在单独的容器中运行本地绑定应用程序』
- 查看 IBM MQ 样本 Helm Chart。

构建您自己的队列管理器映像时的常规注意事项

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。

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。

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 77 页的『构建样本基本 IBM MQ 队列管理器映像』
  - 第 78 页的『构建样本配置的 IBM MQ 队列管理器映像』

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。
- 可选： 如果计划使用 Red Hat OpenShift Container Platform “受限” 安全上下文约束 (SCC) 来配置安全访问，请使用其中一个 IBM MQ 非安装映像。

  在 IBM MQ 下载的 "容器" 部分中提供了用于下载这些映像的链接。

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。

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

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。 在当前版本的 IBM MQ 中，当您使用 “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。

在单独的容器中运行本地绑定应用程序

通过在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

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。

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 80 页的『对名称空间应用程序进行故障诊断』 以获取更多信息。

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 79 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
```

MQ Adv. Kubernetes 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。

注: 仅当您正在运行 Kubernetes 或 Red Hat OpenShift 时，才能以此方式在自己的容器中创建本机 HA 组。

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)
```

V 9.4.2 MQ Adv. Kubernetes 如果创建自己的容器，请创建原生高可用性 CRR 配置

要创建本机高可用性组，必须创建、配置并启动三个队列管理器。

关于此任务

创建原生高可用性 CRR 解决方案的推荐方法是使用 IBM MQ 运算符（参见 第 100 页的『使用 IBM MQ Operator』）。 如果您创建自己的容器，也可以按照以下说明进行操作。

要创建原生高可用性 CRR 配置，必须为队列管理器创建两个原生高可用性组，一个在“实时”站点，一个在“恢复”站点。 然后，您可以编辑队列管理器的 qm.ini 文件，以在它们之间建立跨区域复制。

注: 如果您运行的是 Kubernetes 或 Red Hat OpenShift，则只能以这种方式在自己的容器中创建原生 HA CRR 配置。

过程

1. 在您的 Live 网站上创建一个本地 HA 组，请参阅 第 81 页的『创建本机 HA 组 (如果创建您自己的容器)』。
2. 在您的恢复站点上，创建第二个本机高可用性组，请参阅 第 81 页的『创建本机 HA 组 (如果创建您自己的容器)』。
3. 通过以下方式修改队列管理器的 qm.ini 文件，指定实时站点和恢复站点上的两个组在原生 HA CRR 配置中协同工作。

   在恢复组中，在 NativeHALocalInstance 节指定以下值：

   GroupName
   为恢复组指定一个名称，例如“NHACRR_recovery”。

   GroupRole
   请说明该小组的当前角色。 在这种情况下，设置为“恢复”。

   请在 NativeHARecoveryGroup 节中指定以下值：

   GroupName
   例如，在直播网站上，该群组的名称为“NHACRR_live”。

   ReplicationAddress

   指定主机名和端口，用于在本地高可用跨区域复制配置中处理另一个组的复制流量端口。 地址可以是单个服务地址和端口，也可以是逗号分隔的列表。

   在实时群组中，在 NativeHALocalInstance 节指定以下值：

   GroupName
   为直播群组指定一个名称，例如“NHACRR_live”。

   GroupRole
   请说明该小组的当前角色。 在这种情况下，设置为“实时”。

   请在 NativeHARecoveryGroup 节中指定以下值：

   GroupName
   例如，恢复站点上的组名称为“NHACRR_recovery”。

ReplicationAddress

指定主机名和端口，用于在本地高可用跨区域复制配置中处理另一个组的复制流量端口。 地址可以是单个服务地址和端口，也可以是逗号分隔的列表。

4. 您必须使用 TLS 确保复制连接的安全。 如果您在创建两个组时指定了内部复制的安全性，则可以使用相同的设置来保护外部复制的安全性。 或者，您可以在 NativeHALocalInstance 节中为恢复组和实时组指定以下值。

   GroupCipherSpec
   覆盖 CipherSpec 指定的密码规格。

   GroupCertificateLabel
   覆盖 CertificateLabel 指定的证书标签。

   如果您希望使用这些值进行外部复制，同时使用纯文本进行内部复制，请为实时组和恢复组设置这两个值，同时将 CipherSpec 设置为 NULL。

示例

以下示例显示了 qm.ini 文件的一部分，其中指定了实时组中三个实例之一所需的 Native HA CRR 详细信息：

```
NativeHALocalInstance:
   ...
   GroupName=NHACRR_live
   GroupRole=live

NativeHARecoveryGroup:
   GroupName=NHACRR_recovery
   ReplicationAddress=recovery-cluster.fyre.ibm.com(4445)
```

相关信息

NativeHALocalInstance qm.ini 文件的节
NativeHARecoveryGroup qm.ini 文件的节
NativeHAInstance qm.ini 文件的节

在容器中部署和配置队列管理器

您可以执行一系列任务来部署和配置 IBM MQ 队列管理器。

关于此任务

要开始部署和配置队列管理器，请参阅以下主题。

过程

- 第 84 页的『使用 IBM MQ Operator 部署和配置队列管理器』
- 第 165 页的『使用 Helm 图表示例部署和配置队列管理器』

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

配置示例; 配置 HA; 从 OpenShift 集群外部进行连接; 与 CP4i 仪表板集成; 与 Instana 跟踪集成; 使用定制 MQSC 和 INI 文件构建映像; 添加定制注释和标签。

关于此任务

过程

- 第 87 页的『配置队列管理器的示例』.
- 第 97 页的『使用 IBM MQ Operator 为队列管理器配置高可用性』.
- 第 129 页的『配置路由以从 Red Hat OpenShift 集群外部连接到队列管理器』.

- 第 132 页的『将 IBM MQ 与 IBM Instana 跟踪集成』.
- 第 139 页的『使用 Red Hat OpenShift CLI 构建具有定制 MQSC 和 INI 文件的映像』.
- 第 143 页的『向队列管理器资源添加定制注释和标签』.
- 第 143 页的『禁用运行时 Webhook 检查』.
- 第 144 页的『禁用队列管理器规范的缺省值更新』.

OpenShift CP4I 使用 IBM MQ Operator 部署简单队列管理器

此示例部署 "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 IBM 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。

关于此任务

此任务提供了用于将队列管理器部署到 OpenShift 的 3 选项:

1. 使用 OpenShift 控制台部署队列管理器。
2. 使用 OpenShift CLI 部署队列管理器。
3. 使用 IBM Cloud Pak for Integration Platform UI 部署队列管理器。

过程

- **选项 1: 使用 OpenShift 控制台部署队列管理器。**

  a) 部署队列管理器。

  a. 使用 Red Hat OpenShift Container Platform 集群管理员凭证登录到 OpenShift 控制台。

  b. 将 **项目** 更改为安装了 IBM MQ Operator 的名称空间。 从 **项目** 下拉列表中选择名称空间。

  c. 在导航窗格中，单击 **操作程序 > 已安装的操作程序**。

  d. 在 "已安装的操作程序" 面板上的列表中，查找并单击 **IBM MQ**。

  e. 单击 **队列管理器** 选项卡。

  f. 单击 **创建 QueueManager** 按钮。 将显示实例创建面板，并提供两种方法来配置资源: **表单视图** 和 **YAML 视图**。 缺省情况下会选择 **表单视图** 。

  b) 配置队列管理器。

  步骤 2 选项 1: 在 **表单视图**中进行配置。

  表单视图 将打开可用于查看或修改资源配置的表单。

  a. 在 **许可证**旁边，单击箭头以展开许可证接受部分。

  b. 如果您接受许可协议，请将 **许可证接受** 设置为 **true** 。

  c. 单击箭头以打开下拉列表，然后选择许可证。 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 219 页的『mq.ibm.com/v1beta1: 当前许可证版本』。 您必须接受部署队列管理器的许可证。

  d. 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。

  步骤 2 选项 2: 在 **YAML 视图**中配置。

  YAML 视图 将打开一个编辑器，其中包含 `QueueManager` 的示例 YAML 文件。 通过执行以下步骤来更新文件中的值。

  a. 将 `metadata.namespace` 更改为项目 (名称空间) 名称。

  b. 将 `spec.license.license` 的值更改为与您的需求匹配的许可证字符串。 请参阅 第 219 页的『mq.ibm.com/v1beta1: 当前许可证版本』 以获取许可证详细信息。

  c. 如果您接受许可协议，请将 `spec.license.accept` 更改为 `true` 。

  d. 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。

c) 验证队列管理器创建。

您可以通过完成以下步骤来验证是否已创建队列管理器:

a. 确保您位于创建 IBM MQ Operator 的名称空间中。

b. 在 " **主** " 屏幕中，单击 **操作程序 > 已安装的操作程序**，然后选择为其创建队列管理器的已安装 IBM MQ Operator 。

c. 单击 **队列管理器** 选项卡。 当 `QueueManager` 状态为 `Running` 时，将完成创建。

- **选项 2: 使用 OpenShift CLI 部署队列管理器**。

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: 9.4.3.0-r1
  license:
    accept: false
    license: L-CYPF-CRPF3H
    use: NonProduction
  web:
    enabled: true
  queueManager:
    name: "QUICKSTART"
    storage:
      queueManager:
        type: ephemeral
```

要点: 如果您接受许可协议，请将 `accept: false` 更改为 `accept: true`。 请参阅 第 219 页的『mq.ibm.com/v1beta1: 当前许可证版本』 以获取有关许可证的详细信息。

此示例还包括随队列管理器一起部署的 Web 服务器，以及在 IBM Cloud Pak for Integration 中通过单点登录启用的 Web 控制台。 要使单点登录生效，首先需要安装其他 IBM Cloud Pak for Integration 组件。 请参阅第 57 页的『安装 IBM MQ Operator 以用于 CP4I』。

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":

注: 本示例使用 IBM MQ Advanced 许可证，但也可以使用 IBM MQ 许可证。 有关更多信息，请参阅第 140 页的『使用 IBM MQ 许可证注释配置队列管理器。 IBM MQ Operator』。

```
apiVersion: mq.ibm.com/v1beta1
kind: QueueManager
metadata:
  name: quickstart
spec:
  version: 9.4.3.0-r1
  license:
    accept: false
    license: L-NUUP-23NH8Y
  web:
    enabled: true
  queueManager:
    name: "QUICKSTART"
    storage:
      queueManager:
        type: ephemeral
```

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 219 页的『mq.ibm.com/v1beta1: 当前许可证版本』 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
```

c) 验证队列管理器创建。

通过完成以下步骤验证您是否已创建队列管理器:

a. 验证部署:

```
oc describe queuemanager Queue_Manager_Resource_Name
```

b. 检查状态:

```
oc describe queuemanager quickstart
```

- **选项 3: 使用 IBM Cloud Pak for Integration Platform UI 部署队列管理器。**

  遵循 使用 Platform UI 部署实例中的指示信息。

相关任务

第 129 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 191 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 。

第 87 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。

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 87 页的『示例: 提供 MQSC 和 INI 文件』
- 第 92 页的『示例: 配置具有相互 TLS 认证的队列管理器』

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。 这些必须在部署队列管理器的命名空间（项目）中创建。

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

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: 在您的集群中应用以下 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 Red Hat OpenShift Container Platform Web 控制台，通过以下配置部署队列管理器。

注: 本示例使用 IBM MQ Advanced 许可证，但也可以使用 IBM MQ 许可证。 有关更多信息，请参阅第 140 页的『使用 IBM MQ 许可证注释配置队列管理器。 IBM MQ Operator』。

```
apiVersion: mq.ibm.com/v1beta1
kind: QueueManager
metadata:
  name: mqsc-ini-qm
spec:
  version: 9.4.3.0-r1
  license:
    accept: false
    license: L-NUUP-23NH8Y
    use: Production
  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 MQ Advanced 许可协议，请将 `accept: false` 更改为 `accept: true`。 有关许可证的详细信息，请参阅第 219 页的『mq.ibm.com/v1beta1: 当前许可证版本』。

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 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，无论它们在队列管理器配置中的顺序如何，`example1.mqsc` 总是先于 `example2.mqsc` 进行处理。
- 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。
- 当队列管理器 pod 正在运行时，不会选取对 Kubernetes ConfigMap 的任何更改，因为 IBM MQ Operator 不知道该更改。 如果对 ConfigMap, 进行更改，例如更改 MQSC 命令或 INI 文件，则必须手动重启队列管理器以接收这些更改。 对于单实例队列管理器，请删除 pod 以触发所需的重新启动。 对于本机 HA 部署，请先通过删除备用 pod 来重新启动这些 pod。 当它们再次处于运行状态时，请删除活动 pod 以将其重新启动。 此重新启动顺序可确保队列管理器的最短停机时间。

OpenShift CP4I 示例：向队列管理器提供附加文件

这个例子创建了一个包含两个文件的 Kubernetes ConfigMap。 然后部署一个队列管理器，将文件装载到 IBM MQ 容器中的指定位置。

关于此任务

部署队列管理器时，可以添加其他文件。 这些文件必须在一个或多个 Kubernetes ConfigMaps 和 Secrets 中。 这些必须在部署队列管理器的命名空间（项目）中创建。 当文件包含敏感数据时，应使用 Kubernetes Secret。

注: 此功能仅支持 IBM MQ Operator 3.5.0 或更高版本。

示例

以下示例创建一个包含两个文件的 Kubernetes ConfigMap。 然后部署一个队列管理器，将文件装载到 IBM MQ 容器中的指定位置。

示例 ConfigMap: 在您的集群中应用以下 YAML。

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: files-example
data:
  file1.txt: |
    Test File 1
  file2.txt: |
    Test File 2
```

示例 QueueManager: 使用命令行或 Red Hat OpenShift Container Platform Web 控制台，通过以下配置部署队列管理器。

注: 本示例使用 IBM MQ Advanced 许可证，但也可以使用 IBM MQ 许可证。 有关更多信息，请参阅第 140 页的『使用 IBM MQ 许可证注释配置队列管理器。 IBM MQ Operator』。

```
apiVersion: mq.ibm.com/v1beta1
kind: QueueManager
metadata:
  name: files-qm
spec:
  version: 9.4.3.0-r1
  license:
    accept: false
    license: L-NUUP-23NH8Y
    use: Production
  queueManager:
    files:
    - configMap:
        name: files-example
        defaultMountPath: "/etc/mqm/myfiles"
        items:
        - item: file1.txt
        - item: file2.txt
          mountPath: "/etc/mqm/otherfiles"
    storage:
      queueManager:
        type: ephemeral
```

要点: 如果您接受 IBM MQ Advanced 许可协议，请将 `accept: false` 更改为 `accept: true`。 如需了解许可证的详细信息，请发送电子邮件至第 219 页的『mq.ibm.com/v1beta1: 当前许可证版本』。

文件位置:

- 文件 `file1.txt` 被挂载到位置 `/etc/mqm/myfiles/file1.txt`，这是为这个 ConfigMap 指定的 **`defaultMountPath`**。
- 文件 `file2.txt` 安装在位置 `/etc/mqm/otherfiles/file2.txt` 中，这是该特定文件的 **`mountPath`** 覆盖。

其他信息:

- 当 **`defaultMountPath`** 未设置（用于 ConfigMap 或 Secret）且未指定 **`mountPath`** 覆盖时，文件默认挂载到以下位置 `/etc/mqm/customfiles`
- 如果您更改了 ConfigMap 或 Secret 中任何文件的内容，则必须手动重启队列管理器以获取这些更改。 对于单实例队列管理器，请删除容器以触发所需的重启。 对于原生高可用性部署，首先通过删除备用容器来

重启它们。 当它们再次处于运行状态时，请删除活动进程以重新启动。 这种重启顺序可确保队列管理器的停机时间最短。

OpenShift CP4I 示例：为队列管理器提供额外的环境变量

本示例部署了一个队列管理器，并添加了两个环境变量。 这些环境变量设置在 IBM MQ 容器上。

关于此任务

部署队列管理器时，可以添加其他环境变量。 这允许设置环境变量，这些变量尚未由 IBM MQ Operator 公开。

注: 此功能仅支持 IBM MQ Operator 3.5.0 或更高版本。

示例

以下示例部署了一个队列管理器，并添加了两个环境变量。 这些环境变量设置在 IBM MQ 容器上。

示例 QueueManager: 使用命令行或 Red Hat OpenShift Container Platform Web 控制台，通过以下配置部署队列管理器。

注: 本示例使用 IBM MQ Advanced 许可证，但也可以使用 IBM MQ 许可证。 有关更多信息，请参阅第 140 页的『使用 IBM MQ 许可证注释配置队列管理器。 IBM MQ Operator』。

```
apiVersion: mq.ibm.com/v1beta1
kind: QueueManager
metadata:
  name: env-qm
spec:
  version: 9.4.3.0-r1
  license:
    accept: false
    license: L-NUUP-23NH8Y
    use: Production
  queueManager:
    env:
    - name: TEST_ENV_1
      value: "true"
    - name: TEST_ENV_2
      value: "false"
    storage:
      queueManager:
        type: ephemeral
```

要点: 如果您接受 IBM MQ Advanced 许可协议，请将 `accept: false` 更改为 `accept: true`。 如需了解许可证的详细信息，请发送电子邮件至 第 219 页的『mq.ibm.com/v1beta1: 当前许可证版本』。

其他信息:

- 如果您指定了本应由 IBM MQ Operator 设置的环境变量，则可能导致不可预测的行为。 在这种情况下，您的 QueueManager 将收到一条警告状态信息。

OpenShift CP4I 使用 *OpenSSL* 创建自签名 *PKI*

IBM MQ 允许您使用相互 TLS 进行认证，其中连接两端都提供证书，证书中的详细信息用于与队列管理器建立身份。 本主题介绍了如何使用 OpenSSL 命令行工具创建示例公用密钥基础结构 (PKI)，从而创建可在其他示例中使用的两个证书。

开始之前

确保已安装 OpenSSL 命令行工具。

安装 IBM MQ client，并将 `samp/bin` 和 `bin` 添加到 *PATH*。 您需要使用 "**runmqicred** 命令，它可以作为 "IBM MQ client 的一部分安装，如下所示:

- Windows Linux 对于 Windows 和 Linux: 从 https://ibm.biz/mq94redistclients 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/

关于此任务

要点: 此处描述的示例不适合生产环境，仅用作快速执行的示例。 证书管理是高级用户的复杂主题。 对于生产，您必须考虑诸如轮换，撤销，密钥长度，灾难恢复等事项。

这些步骤已使用 OpenSSL 3.1.4 进行测试。

过程

1. 创建要用于内部认证中心的专用密钥

   ```
   openssl genpkey -algorithm rsa -pkeyopt rsa_keygen_bits:4096 -out ca.key
   ```

   内部认证中心的专用密钥是在名为 *ca.key* 的文件中创建的。 此文件应保持安全和秘密-它将用于签署内部认证中心的证书。
2. 为内部认证中心发放自签名证书

   ```
   openssl req -x509 -new -nodes -key ca.key -sha512 -days 30 -subj "/CN=example-selfsigned-ca"
   -out ca.crt
   ```

   `-days` 指定根 CA 证书将有效的天数。

   将在名为 *ca.crt* 的文件中创建证书。 此证书包含有关内部认证中心的公共信息，并且可自由共享。
3. 为队列管理器创建专用密钥和证书

   a) 为队列管理器创建专用密钥和证书签名请求

   ```
   openssl req -new -nodes -out example-qm.csr -newkey rsa:4096 -keyout example-qm.key -subj
   '/CN=example-qm'
   ```

   在名为 *example-qm.key* 的文件中创建专用密钥，并在名为 *example-qm.csr* 的文件中创建证书签名请求

   b) 使用内部认证中心对队列管理器密钥进行签名

   ```
   openssl x509 -req -in example-qm.csr -CA ca.crt -CAkey ca.key -CAcreateserial -out
   example-qm.crt -days 7 -sha512
   ```

   `-days` 指定证书将有效的天数。

   在名为 *example-qm.crt* 的文件中创建签名证书

   c) 使用队列管理器密钥和证书创建 Kubernetes 私钥

   ```
   kubectl create secret generic example-qm-tls --type="kubernetes.io/tls" --from-
   file=tls.key=example-qm.key --from-file=tls.crt=example-qm.crt --from-file=ca.crt
   ```

   将创建名为 *example-qm-tls* 的 Kubernetes 私钥。 此私钥包含队列管理器，公用证书和 CA 证书的专用密钥。
4. 为应用程序创建专用密钥和证书

   a) 为应用程序创建专用密钥和证书签名请求

   ```
   openssl req -new -nodes -out example-app1.csr -newkey rsa:4096 -keyout example-app1.key
   -subj '/CN=example-app1'
   ```

   在名为 *example-app1.key* 的文件中创建专用密钥，并在名为 *example-app1.csr* 的文件中创建证书签名请求

   b) 使用内部认证中心对队列管理器密钥进行签名

   ```
   openssl x509 -req -in example-app1.csr -CA ca.crt -CAkey ca.key -CAcreateserial -out
   example-app1.crt -days 7 -sha512
   ```

`-days` 指定证书将有效的天数。

在名为 *example-app1.crt* 的文件中创建签名证书

c) 使用应用程序的密钥和证书创建 PKCS#12 密钥库

IBM MQ 使用密钥数据库，而不是单个密钥文件。 容器化队列管理器将从私钥为队列管理器创建密钥数据库，但对于客户机应用程序，您需要手动创建密钥数据库。

```
openssl pkcs12 -export -in "example-app1.crt" -name "example-app1" -certfile "ca.crt"
-inkey "example-app1.key" -out "example-app1.p12" -passout pass:PASSWORD
```

其中 *PASSWORD* 是您自己选择的密码。

将在名为 *example-app1.p12* 的文件中创建密钥库。 应用程序的密钥和证书存储在其中，带有 "example-app1" 的 "标签" 或 "友好名称" 以及 CA 证书。

d) 如果您正在使用 arm64 Apple Mac，那么需要配置一个额外的文件来组合应用程序和 CA 证书。 例如：

```
cat example-app1.crt ca.crt > example-app1-chain.crt
```

相关任务

第 92 页的『示例: 配置具有相互 TLS 认证的队列管理器』
此示例使用 IBM MQ Operator 将队列管理器部署到 OpenShift Container Platform 中。 相互 TLS 用于认证，以从 TLS 证书映射到队列管理器中的身份。

第 98 页的『示例: 使用 IBM MQ Operator 配置本机 HA』
此示例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 OpenShift Container Platform 中。 相互 TLS 用于认证，以从 TLS 证书映射到队列管理器中的身份。

第 127 页的『使用 IBM MQ Operator 配置多实例队列管理器』
此示例使用 IBM MQ Operator 将多实例队列管理器部署到 OpenShift Container Platform 中。 相互 TLS 用于认证，以从 TLS 证书映射到队列管理器中的身份。

OpenShift CP4I Linux 示例: 配置具有相互 *TLS* 认证的队列管理器

此示例使用 IBM MQ Operator 将队列管理器部署到 OpenShift Container Platform 中。 相互 TLS 用于认证，以从 TLS 证书映射到队列管理器中的身份。

开始之前

要完成此示例，必须首先完成以下先决条件：

- 为此示例创建 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 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。

过程

1. 创建一对证书，如 第 90 页的『使用 OpenSSL 创建自签名 PKI』中所述。
2. 创建包含 MQSC 命令和 INI 文件的配置映射

   创建包含 MQSC 命令的 Kubernetes ConfigMap，以创建新队列和 SVRCONN 通道，并添加允许访问该通道的通道认证记录。

   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 Web 控制台中输入以下 YAML，或者使用命令行。

   ```
   apiVersion: v1
   kind: ConfigMap
   ```

```
metadata:
  name: example-tls-configmap
data:
  example-tls.mqsc: |
    DEFINE CHANNEL('MTLS.SVRCONN') CHLTYPE(SVRCONN) SSLCAUTH(REQUIRED)
SSLCIPH('ANY_TLS13_OR_HIGHER') REPLACE
    SET CHLAUTH('MTLS.SVRCONN') TYPE(SSLPEERMAP) SSLPEER('CN=*') USERSRC(NOACCESS)
ACTION(REPLACE)
    SET CHLAUTH('MTLS.SVRCONN') TYPE(SSLPEERMAP) SSLPEER('CN=example-app1') USERSRC(MAP)
MCAUSER('app1') ACTION(REPLACE)
    SET AUTHREC PRINCIPAL('app1') OBJTYPE(QMGR) AUTHADD(CONNECT,INQ)
    DEFINE QLOCAL('EXAMPLE.QUEUE') REPLACE
    SET AUTHREC PROFILE('EXAMPLE.QUEUE') PRINCIPAL('app1') OBJTYPE(QUEUE)
AUTHADD(BROWSE,PUT,GET,INQ)
  example-tls.ini: |
    Service:
        Name=AuthorizationService
        EntryPoints=14
        SecurityPolicy=UserExternal
```

MQSC 定义名为 *MTLS.SVRCONN* 的通道和名为 *EXAMPLE.QUEUE*。 此通道配置为仅允许访问提供 "公共名称" 为 *example-app1* 的证书的客户机。 这是在步骤 第 92 页的『1』中创建的其中一个证书中使用的公共名称。 此通道上具有此公共名称的连接将映射到用户标识 *app1*，该用户标识有权连接到队列管理器并访问示例队列。 INI 文件启用安全策略，这意味着 *app1* 用户标识不需要存在于外部用户注册表中-它仅作为此配置中的名称存在。

3. 部署队列管理器

   使用以下定制资源 YAML 创建新的队列管理器。 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 Web 控制台中或使用命令行输入以下 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。

   注: 本示例使用 IBM MQ Advanced 许可证，但也可以使用 IBM MQ 许可证。 有关更多信息，请参阅第 140 页的『使用 IBM MQ 许可证注释配置队列管理器。 IBM MQ Operator』。

```
apiVersion: mq.ibm.com/v1beta1
kind: QueueManager
metadata:
  name: exampleqm
spec:
  license:
    accept: false
    license: L-NUUP-23NH8Y
    use: Production
  queueManager:
    name: EXAMPLEQM
    mqsc:
    - configMap:
        name: example-tls-configmap
        items:
        - example-tls.mqsc
    ini:
    - configMap:
        name: example-tls-configmap
        items:
        - example-tls.ini
    storage:
      queueManager:
        type: ephemeral
  version: 9.4.3.0-r1
  pki:
    keys:
      - name: default
        secret:
          secretName: example-qm-tls
          items:
            - tls.key
            - tls.crt
            - ca.crt
```

   请注意，密钥 *example-qm-tls* 是在步骤 第 92 页的『1』中创建的，而 ConfigMap *example-tls-configmap* 是在步骤 第 92 页的『2』中创建的

4. 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 exampleqm
```

5. 测试与队列管理器的连接

   要确认为相互 TLS 通信配置了队列管理器，请遵循 第 94 页的『从笔记本电脑测试到队列管理器的相互 TLS 连接』中的步骤。

结果

恭喜您成功部署了启用了 TLS 的队列管理器，该队列管理器使用 TLS 证书中提供的详细信息向队列管理器进行认证并提供身份。

OpenShift CP4I Linux 从笔记本电脑测试到队列管理器的相互 *TLS* 连接

使用 IBM MQ Operator 创建队列管理器后，可以通过连接到该队列管理器并放置和获取消息来测试该队列管理器是否正常工作。 此任务将指导您完成如何使用 IBM MQ 样本程序进行连接，方法是在 Kubernetes 集群外部的机器 (例如笔记本电脑) 上运行这些程序。

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。 您需要 **amqsputc** 和 **amqsgetc** 命令，这些命令可以作为 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/mq94redistclients 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/
- 确保将必需的密钥和证书文件下载到机器上的某个目录，并确保您知道密钥库密码。 例如，这些文件是在第 90 页的『使用 OpenSSL 创建自签名 PKI』中创建的：
  - `example-app1.p12`
  - `example-app1-chain.crt` (仅当您正在使用 arm64 Apple Mac 时)
- 将使用 TLS 配置的队列管理器部署到 OCP 集群，例如，执行第 92 页的『示例：配置具有相互 TLS 认证的队列管理器』中的步骤

关于此任务

此示例使用在 Kubernetes 集群外部的机器 (例如笔记本电脑) 上运行的 IBM MQ 样本程序来连接到使用 TLS 配置的 QueueManager ， 并放置和获取消息。

过程

1. 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 exampleqm
   ```

2. 查找队列管理器主机名

   使用以下命令从 OCP 集群外部使用自动创建的路径来查找队列管理器的队列管理器标准主机名：`exampleqm-ibm-mq-qm`：

   ```
   oc get route exampleqm-ibm-mq-qm --template="{{.spec.host}}"
   ```

3. 创建 IBM MQ 客户机通道定义表 (CCDT)

使用以下内容创建名为 ccdt.json 的文件:

```
{
    "channel":
    [
        {
            "name": "MTLS.SVRCONN",
            "clientConnection":
            {
                "connection":
                [
                    {
                        "host": "hostname from previous step",
                        "port": 443
                    }
                ],
                "queueManager": "EXAMPLEQM"
            },
            "transmissionSecurity":
            {
              "cipherSpecification": "ANY_TLS13",
              "certificateLabel": "example-app1"
            },
            "type": "clientConnection"
        }
   ]
}
```

连接使用端口 443 ，因为这是 Red Hat OpenShift Container Platform 路由器正在侦听的端口。 流量将转发到端口 1414 上的队列管理器。

如果您使用了不同的通道名称，那么还需要对其进行调整。 相互 TLS 示例使用名为 *MTLS.SVRCONN* 的通道

有关更多详细信息，请参阅 配置 JSON 格式 CCDT

4. 创建客户机 INI 文件以配置连接详细信息

   在当前目录中创建名为 mqclient.ini 的文件。 此文件将由 **amqsputc** 和 **amqsgetc** 读取。

   ```
   Channels:
     ChannelDefinitionDirectory=.
     ChannelDefinitionFile=ccdt.json
   SSL:
     OutboundSNI=HOSTNAME
     SSLKeyRepository=example-app1.p12
     SSLKeyRepositoryPassword=password you used when creating the p12 file
   ```

   确保将 *SSLKeyRepositoryPassword* 更新为您在创建 PKCS#12 文件时选择的密码。 还有其他方法可以设置密钥库密码，包括使用加密密码。 有关更多信息，请参阅 在 AIX, Linux, and Windows 上提供 IBM MQ MQI client 的密钥存储库密码

   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。 *OutboundSNI= HOSTNAME* 属性确保 IBM MQ 客户机包含路由器使用 IBM MQ Operator 配置的缺省路由所需的信息。 有关更多信息，请参阅第 129 页的『配置路由以从 Red Hat OpenShift 集群外部连接到队列管理器』。

5. 如果您正在使用 arm64 Apple Mac，那么需要配置其他环境变量。

   ```
   export MQSSLTRUSTSTORE=example-app1-chain.crt
   ```

   此文件包含完整的证书链，包括应用程序和 CA 证书。

6. 将消息放入队列

   运行以下命令:

   ```
   /opt/mqm/samp/bin/amqsputc EXAMPLE.QUEUE EXAMPLEQM
   ```

   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。

7. 从队列中检索消息
   运行以下命令：

```
/opt/mqm/samp/bin/amqsgetc EXAMPLE.QUEUE EXAMPLEQM
```

   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。

结果

恭喜您成功测试了启用了 TLS 的队列管理器连接，并显示您可以从客户机安全地将消息放入队列管理器中。

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 143 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 MQ Adv. 使用 IBM MQ Operator 为队列管理器配置高可用性

您可以使用本地 HA 解决方案或多实例队列管理器解决方案为队列管理器配置高可用性。

关于此任务

按照所选高可用性解决方案的程序操作。

过程

- 第 38 页的『本机 HA』.
- 第 98 页的『示例: 使用 IBM MQ Operator 配置本机 HA』.
- 第 127 页的『使用 IBM MQ Operator 配置多实例队列管理器』.

OpenShift MQ Adv. 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-NUUP-23NH8Y
    use: Production
  queueManager:
    availability:
      type: NativeHA
  version: 9.4.3.0-r1
```

注: 本例使用 IBM MQ Advanced 许可证。 从 9.4.2 ， IBM MQ 许可的队列管理器也可通过向队列管理器添加额外许可配置为使用本地 HA 和跨区域复制功能。 更多信息，请参见 第 101 页的『在 IBM MQ 许可的队列管理器上配置本地 HA 和跨区域复制，方法是使用 IBM MQ Operator』。

.spec.queueManager.availability.type 字段必须设置为 NativeHA。

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 98 页的『示例: 使用 IBM MQ Operator 配置本机 HA』中提供了逐步指南。

相关任务

第 98 页的『示例: 使用 IBM MQ Operator 配置本机 HA』
此示例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 OpenShift Container Platform 中。 相互 TLS 用于认证，以从 TLS 证书映射到队列管理器中的身份。

OpenShift MQ Adv. Kubernetes 示例: 使用 *IBM MQ Operator* 配置本机 *HA*

此示例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 OpenShift Container Platform 中。 相互 TLS 用于认证，以从 TLS 证书映射到队列管理器中的身份。

开始之前

要完成此示例，必须首先完成以下先决条件:

- 为此示例创建 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 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。

过程

1. 创建一对证书，如 第 90 页的『使用 OpenSSL 创建自签名 PKI』中所述。
2. 创建包含 MQSC 命令和 INI 文件的配置映射

   创建包含 MQSC 命令的 Kubernetes ConfigMap，以创建新队列和 SVRCONN 通道，并添加允许访问该通道的通道认证记录。

   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 Web 控制台中输入以下 YAML，或者使用命令行。

   ```
   apiVersion: v1
   kind: ConfigMap
   metadata:
     name: example-nativeha-configmap
   data:
     example-tls.mqsc: |
       DEFINE CHANNEL('MTLS.SVRCONN') CHLTYPE(SVRCONN) SSLCAUTH(REQUIRED)
   SSLCIPH('ANY_TLS13_OR_HIGHER') REPLACE
       SET CHLAUTH('MTLS.SVRCONN') TYPE(SSLPEERMAP) SSLPEER('CN=*') USERSRC(NOACCESS)
   ACTION(REPLACE)
       SET CHLAUTH('MTLS.SVRCONN') TYPE(SSLPEERMAP) SSLPEER('CN=example-app1') USERSRC(MAP)
   MCAUSER('app1') ACTION(REPLACE)
       SET AUTHREC PRINCIPAL('app1') OBJTYPE(QMGR) AUTHADD(CONNECT,INQ)
       DEFINE QLOCAL('EXAMPLE.QUEUE') REPLACE
       SET AUTHREC PROFILE('EXAMPLE.QUEUE') PRINCIPAL('app1') OBJTYPE(QUEUE)
   AUTHADD(BROWSE,PUT,GET,INQ)
     example-tls.ini: |
       Service:
           Name=AuthorizationService
           EntryPoints=14
           SecurityPolicy=UserExternal
   ```

   MQSC 定义名为 *MTLS.SVRCONN* 的通道和名为 *EXAMPLE.QUEUE*。 此通道配置为仅允许访问提供 "公共名称" 为 *example-app1* 的证书的客户机。 这是在步骤 第 98 页的『1』中创建的其中一个证书中使用的公共名称。 此通道上具有此公共名称的连接将映射到用户标识 *app1*，该用户标识有权连接到队列管理器并访问示例队列。 INI 文件启用安全策略，这意味着 *app1* 用户标识不需要存在于外部用户注册表中-它仅作为此配置中的名称存在。
3. 部署队列管理器

   使用以下定制资源 YAML 创建新的队列管理器。 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 Web 控制台中或使用命令行输入以下 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。

注:

本例使用 IBM MQ Advanced 许可证。 从 IBM MQ 9.4.2 ，使用 IBM MQ 许可证的队列管理器可使用本机 HA 和跨区域复制功能。 有关更多信息，请参阅第 101 页的『在 IBM MQ 许可的队列管理器上配置本地 HA 和跨区域复制，方法是使用 IBM MQ Operator』。

```
apiVersion: mq.ibm.com/v1beta1
kind: QueueManager
metadata:
  name: exampleqm
spec:
  license:
    accept: false
    license: L-NUUP-23NH8Y
    use: Production
  queueManager:
    name: EXAMPLEQM
    availability:
      type: NativeHA
      tls:
        secretName: example-qm-tls
    mqsc:
    - configMap:
        name: example-nativeha-configmap
        items:
        - example-tls.mqsc
    ini:
    - configMap:
        name: example-nativeha-configmap
        items:
        - example-tls.ini
    storage:
      queueManager:
        type: persistent-claim
  version: 9.4.2.0-r2
  pki:
    keys:
      - name: default
        secret:
          secretName: example-qm-tls
          items:
            - tls.key
            - tls.crt
            - ca.crt
```

请注意，密钥 *example-qm-tls* 是在步骤第 98 页的『1』中创建的，而 ConfigMap *example-nativeha-configmap* 是在步骤第 98 页的『2』中创建的

可用性类型设置为 *NativeHA* ，并选择了持久存储。 将使用 Kubernetes 集群中配置的缺省存储类。 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` 。

本机 HA 队列管理器中的三个 pod 通过网络复制数据。 缺省情况下，此链路未加密，但此示例使用队列管理器的证书对流量进行加密。 您可以指定其他证书以实现其他安全性。 本机 HA TLS 私钥必须是具有特定结构 (例如，专用密钥必须称为 *tls.key*) 的 Kubernetes TLS 私钥。

4. 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 exampleqm
   ```

5. 测试与队列管理器的连接

   要确认队列管理器已配置且可用，请遵循第 94 页的『从笔记本电脑测试到队列管理器的相互 TLS 连接』中的步骤。

6. 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:

   a) 查看活动和备用 pod

   运行以下命令:

   ```
   oc get pods --selector app.kubernetes.io/instance=exampleqm
   ```

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。

b) 删除活动 pod

运行以下命令，并指定活动 pod 的全名:

```
oc delete pod exampleqm-ibm-mq-value
```

c) 再次查看 pod 状态

运行以下命令:

```
oc get pods --selector app.kubernetes.io/instance=exampleqm
```

d) 查看队列管理器状态

运行以下命令，指定 <active-pod> 的全名。活动 pod 应处于就绪状态，即步骤 c 中命令显示的就绪 1/1 的 pod:

```
oc exec -t <active-pod> -- dspmq -o nativeha -x -m EXAMPLEQM
```

您应该会看到状态显示活动实例已更改，例如:

```
QMNAME(EXAMPLEQM) 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)
```

e) 再次测试与队列管理器的连接

要确认队列管理器已恢复，请执行 第 94 页的『从笔记本电脑测试到队列管理器的相互 TLS 连接』中的步骤。

结果

恭喜您成功部署了具有本机高可用性和相互 TLS 认证的队列管理器，并验证其在活动 Pod 失败时自动恢复。

使用 *IBM MQ Operator*

本地 HA 跨区域复制（CRR）使用 QueueManager API 进行配置，高级选项可通过 INI 文件进行配置。

注: 本例使用 IBM MQ Advanced 许可证。 从 IBM MQ 9.4.2， IBM MQ 许可的队列管理器也可通过向队列管理器添加额外许可配置为使用本地 HA CRR 功能。 有关更多信息，请参阅第 101 页的『在 IBM MQ 许可的队列管理器上配置本地 HA 和跨区域复制，方法是使用 IBM MQ Operator』。

本机 HA CRR 的配置方法是使用 `.spec.queueManager.availability` 在 QueueManager API。 `.spec.queueManager.availability.type` 字段必须设置为 `NativeHA`。 `.spec.queueManager.availability.nativeHAGroups.local.role` 字段设置为 `Live` 或 `Recovery`， 以指定本地组的作用。

您可以在配置中的 `.spec.queueManager.availability.nativeHAGroups.remotes` 部分指定远程组的详细信息。 您提供远程组使用的 OpenShift 路由的主机名和端口，数据将复制到该主机。 现场小组必须至少包含 `remotes` 部分，以便复制到恢复小组。 恢复组必须包含 `remotes` 部分，以便在网站上线后能够复制到另一个网站。 （请发送邮件至 第 208 页的『本地 HA CRR 切换和故障切换』）

在 `.spec.queueManager.availability`，您还可以配置一个 TLS 密钥和密码，以便在复制时在队列管理器实例之间使用。 强烈建议这样做， 第 98 页的『示例: 使用 IBM MQ Operator 配置本机 HA』上有分步指南。

在复制时，您必须在实时组和恢复组之间配置 TLS 密钥和密码。 您可以使用为实例间复制指定的相同密钥，也可以为此指定其他密钥。

相关任务

第 98 页的『示例: 使用 IBM MQ Operator 配置本机 HA』
此示例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 OpenShift Container Platform 中。 相互 TLS 用于认证，以从 TLS 证书映射到队列管理器中的身份。

OpenShift CP4I V 9.4.2 在 *IBM MQ* 许可的队列管理器上配置本地 *HA* 和跨区域复制，方法是使用 *IBM MQ Operator*

从 IBM MQ 9.4.2 ， IBM MQ 许可的队列管理器可配置为使用本地 HA 跨区域复制 (CRR) 功能，方法是向队列管理器添加额外的许可。

开始之前

注: 这些说明仅适用于使用 IBM MQ Operator 的情况。 如果要在 Kubernetes 上部署队列管理器，请参见 第 163 页的『在 IBM MQ 许可的队列管理器上配置本机 HA 和跨区域复制。 Kubernetes』 。

在 IBM MQ 许可的队列管理器上使用本地 HA CRR 功能需要 IBM License Service 。 请参阅 IBM MQ Operator 安装过程中的部署 License Service 步骤。

要点: 这项任务包括更新队列管理器 YAML，从而触发 pod 重启。

过程

1. `IBMLicensingDefinition` CR 中添加一个或两个，具体取决于您是需要生产用途还是非生产用途。这些 CR 在队列管理器中为本地 HA CRR 功能添加了额外的许可注释。

   在要使用 Native HA CRR 功能附加许可的每个命名空间中重复此步骤。

   - **方案 1：** 使用 OpenShift 控制台添加 CR：
     - 使用 Red Hat OpenShift Container Platform 群集管理员凭据登录 OpenShift 控制台。
     - 从**项目**下拉列表中选择名称空间，将**项目**更改为部署队列管理器的名称空间。
     - 单击屏幕右上方的**快速创建** （加号）图标，打开**导入 YAML** 页面。
     - 添加相关的 `IBMLicensingDefinition` CR YAML （请参阅本步骤末尾的示例）。
     - 单击**创建**。
     - 如果需要，重复上述步骤，添加其他 `IBMLicensingDefinition` CR。
   - **方案 2：** 使用命令行添加 CR：
     - 使用 `oc login` 命令登录 Red Hat OpenShift 命令行界面 (CLI)。
     - 将 `project` （命名空间）设置为部署队列管理器的命名空间：

       ```
       oc project <QUEUE-MANAGER-NAMESPACE>
       ```

     - 创建 `IBMLicensingDefinition` YAML 文件（参见本步骤末尾的示例）。
     - 应用 `IBMLicensingDefinition` YAML 文件：

       ```
       oc apply -f <IBMLicensingDefinition-UNIQUE-NAME>.yaml
       ```

     - 如果需要，重复上述步骤，添加其他 `IBMLicensingDefinition` CR。

   在以下 `IBMLicensingDefinition` CR YAML 文件示例中，将 `<QUEUE-MANAGER-NAMESPACE>` 值改为部署队列管理器的项目。

   - 供生产使用：

     ```
     apiVersion: operator.ibm.com/v1
     kind: IBMLicensingDefinition
     metadata:
       name: ibmlicensingdefinition-mq-nha-prod
       namespace: <QUEUE-MANAGER-NAMESPACE>
     spec:
       action: cloneModify
       condition:
         metadata:
           annotations:
             additionalLicense: L-EZBK-Y86SSH
             additionalLicenseUse: Production
       scope: cluster
       set:
     ```

```
    productID: 46a57a4a66ee4b904e9b5544b53a2ba2
    productName: IBM MQ Native HA and Cross-Region Replication Add-on
```

- 供非生产使用：

```
apiVersion: operator.ibm.com/v1
kind: IBMLicensingDefinition
metadata:
  name: ibmlicensingdefinition-mq-nha-non-prod
  namespace: <QUEUE-MANAGER-NAMESPACE>
spec:
  action: cloneModify
  condition:
    metadata:
      annotations:
        additionalLicense: L-EZBK-Y86SSH
        additionalLicenseUse: NonProduction
  scope: cluster
  set:
    productID: 525b1331a74d4a209d9f6e9c908e374e
    productName: IBM MQ Native HA and Cross-Region Replication Add-on for Non-Production
Env
```

2. 将相关注释添加到队列管理器的 `spec.annotations` 中。

   要点: 更新队列管理器 YAML 会触发 pod 重启。

   按照 第 142 页的『通过使用 IBM MQ Operator』 中的说明，根据需要添加以下注释之一：

   - 用于 IBM MQ 和 IBM MQ 本地 HA 和跨区域复制附加权限：

```
spec:
  annotations:
    productID: c661609261d5471fb4ff8970a36bccea
    productName: IBM MQ
    additionalLicense: L-EZBK-Y86SSH
    additionalLicenseUse: Production
```

   - 用于 IBM MQ for Non-Production Environment 和 IBM MQ 非生产权限的本地 HA 和跨区域复制插件

```
spec:
  annotations:
    productID: 151bec68564a4a47a14e6fa99266deff
    productName: IBM MQ for Non-Production Environment
    additionalLicense: L-EZBK-Y86SSH
    additionalLicenseUse: NonProduction
```

   根据需要对其他队列管理器重复此步骤。

OpenShift MQ Adv. Kubernetes 示例：使用 *IBM MQ Operator* 部署简单的本地 *HA CRR* 配置

本示例使用简单的本地 HA 跨区域复制 (CRR) 配置，通过 IBM MQ Operator 将队列管理器部署到 OpenShift® Container Platform。

开始之前

要完成此示例，必须首先完成以下步骤：

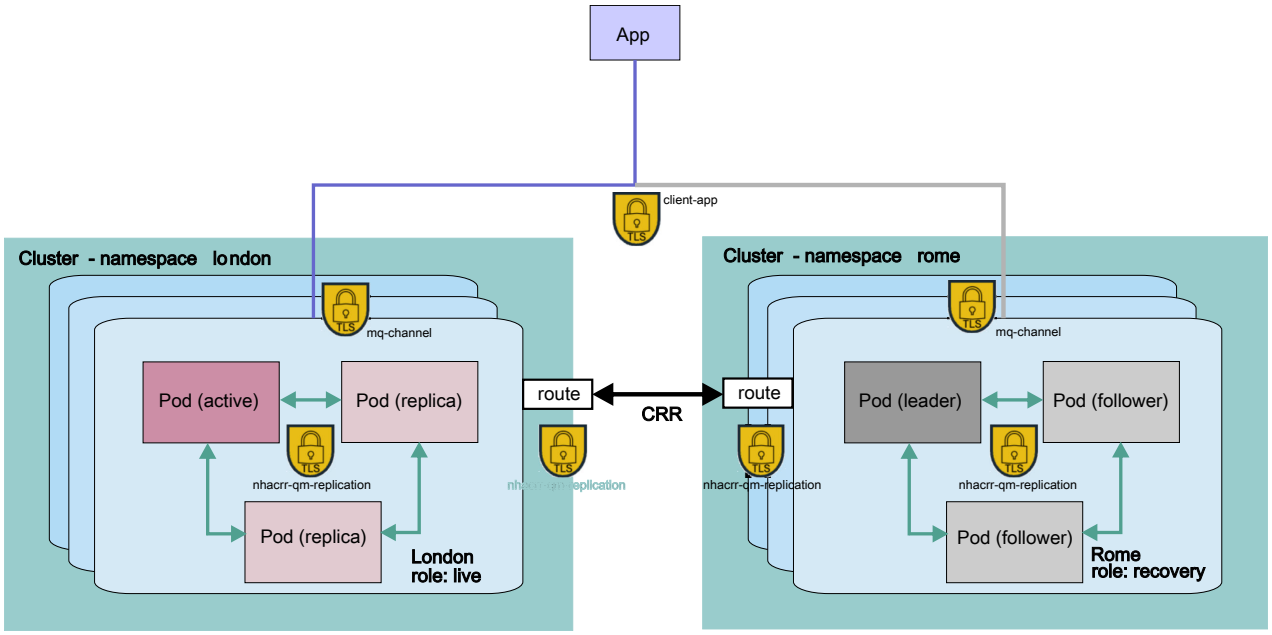
- 在命令行上登录正在使用的 OpenShift Container Platform (OCP) 群集。
- 为此示例创建两个 OCP 项目/命名空间。 示例将提及伦敦和罗马网站。 这些指的是伦敦和罗马在一个群集中的两个命名空间，或者是每个站点在每个群集中的一个命名空间。
- 确保在群集中安装了 IBM MQ Operator 3.5.0 或更高版本，并在命名空间中可用。
- 确保已安装 OpenSSL 命令行工具。
- 完成 第 75 页的『通过创建拉取秘钥，为使用 "Kubernetes 做准备』 任务。
- 请阅读 第 85 页的『使用 IBM MQ Operator 部署简单队列管理器』，以便熟悉使用 IBM MQ 操作符。

关于此任务

本地 HA CRR 配置定义了两个本地 HA 组，它们共同组成一个高可用性队列管理器。 这些组通常部署在不同地点的两个 OCP 集群中，但本示例允许您根据需要将这两个组部署到同一个集群中。 有关本地 HA CRR 的概述，请参阅本地 HA 跨区域复制。

为简单起见，本例使用单个证书来保护所有队列管理器复制流量，该证书名为 nhacrr-qm-replication。

要了解 Native HA CRR 中如何使用证书，以及如何使用不同的证书来确保组和实例流量的安全，请参阅 第 113 页的『示例：使用本地 HA CRR 的多个 TLS 证书』。



要点： 这些示例旨在帮助您入门，并不适合生产环境。 对于高级用户来说，证书管理是一个复杂的课题。 在生产过程中，你必须考虑轮换、撤销、密钥长度、灾难恢复等问题。

本例提供自定义资源 YAML，用于定义两个提供跨区域复制的本地 HA 组。

过程

1. 创建用于队列管理器复制流量的证书。

   使用以下命令创建私钥，并用它为内部证书颁发机构颁发自签名证书。 然后为队列管理器创建私钥和证书，并与内部证书颁发机构签署：

   ```
   openssl genpkey -algorithm rsa -pkeyopt rsa_keygen_bits:4096 -out nhacrr-ca.key
   openssl req -x509 -new -nodes -key "nhacrr-ca.key" -sha512 -days 30 -subj "/CN=nhacrr-
   selfsigned-ca" -out nhacrr-ca.crt
   openssl req -new -nodes -out nhacrr-qm-replication.csr -newkey rsa:4096 -keyout nhacrr-qm-
   replication.key -subj "/CN=nhacrr-qm-replication"
   openssl x509 -req -in nhacrr-qm-replication.csr -CA nhacrr-ca.crt -CAkey nhacrr-ca.key
   -CAcreateserial -out nhacrr-qm-replication.crt -days 7 -sha512
   ```

   有关这些命令的更多信息，请参阅 第 90 页的『使用 OpenSSL 创建自签名 PKI』。
2. 创建两个 Kubernetes 保密文件，其中包含队列管理器密钥和证书。 在罗马和伦敦站点上创建名为 nhacrr-qm-replication 的秘密，用于存储队列管理器实例的密钥和证书。

   使用以下命令创建秘密。 在运行该命令前，先切换到每个命名空间（先是罗马，然后是伦敦）。

   ```
   kubectl create secret generic nhacrr-qm-replication --type="kubernetes.io/tls" --from-
   file=tls.key=nhacrr-qm-replication.key --from-file=tls.crt=nhacrr-qm-replication.crt --from-
   file=nhacrr-ca.crt
   ```
3. 如 第 90 页的『使用 OpenSSL 创建自签名 PKI』 中所述，为队列管理器流量创建一对证书。 在罗马和伦敦站点上创建密文（即图中名为 mq-channel 和 client-app 的证书）。

4. 为罗马和伦敦创建包含 MQSC 命令和 INI 文件的配置地图。 创建 ConfigMap 资源，其中包含 MQSC 命令，用于创建新的持久消息队列和 SVRCONN 通道，并添加允许访问通道的通道验证记录。 在 Rome 网站上，在 OCP 网络控制台输入以下 YAML，选择 ConfigMaps ，然后创建 ConfigMap, 或使用 **oc apply** 命令。 在伦敦站点重复上述步骤：

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: nhacrr-configmap
data:
  nhacrr.mqsc: |
    DEFINE CHANNEL('MTLS.SVRCONN') CHLTYPE(SVRCONN) SSLCAUTH(REQUIRED)
SSLCIPH('ANY_TLS13_OR_HIGHER') REPLACE
    SET CHLAUTH('MTLS.SVRCONN') TYPE(SSLPEERMAP) SSLPEER('CN=*') USERSRC(NOACCESS)
ACTION(REPLACE)
    SET CHLAUTH('MTLS.SVRCONN') TYPE(SSLPEERMAP) SSLPEER('CN=example-app1') USERSRC(MAP)
MCAUSER('app1') ACTION(REPLACE)
    SET AUTHREC PRINCIPAL('app1') OBJTYPE(QMGR) AUTHADD(CONNECT,INQ)
    DEFINE QLOCAL('EXAMPLE.QUEUE') DEFPSIST(YES) REPLACE
    SET AUTHREC PROFILE('EXAMPLE.QUEUE') PRINCIPAL('app1') OBJTYPE(QUEUE)
AUTHADD(BROWSE,PUT,GET,INQ)
  nhacrr.ini: |
    Service:
        Name=AuthorizationService
        EntryPoints=14
        SecurityPolicy=UserExternal
```

MQSC 定义了一个名为 MTLS.SVRCONN 的通道和一个名为 EXAMPLE.QUEUE 的队列。 通道被配置为只允许提交 CN 为 example-app1 的证书（如步骤 3 中创建的证书）的客户端访问。

5. 部署本地 HA 恢复组。 在 Rome 网站上创建一个新的恢复组。 确保处于 rome 命名空间，然后在 OCP 网络控制台输入以下 YAML，创建 QueueManager, 或使用 **oc apply** 命令。 检查是否指定了正确的许可证，并通过将 `false` 更改为 `true` 来接受许可证。

注: 您必须在实时组和恢复组中使用相同的队列管理器名称。 默认情况下，如果不指定 `.spec.queuemanager.name` ，则默认为 `.metadata.name`。 您可以使用以下方法之一实现名称匹配：

- 在每个群集中为队列管理器资源命名相同的名称（即 `.metadata.name` 匹配），并且不要覆盖名称
- 在 `.metadata.name` 中单独命名队列管理器，并用 `.spec.queuemanager.name` 覆盖该名称，以便两个站点的名称一致。

在本例中，使用 `.metadata.name` 将队列管理器名称设置为 `exampleqm` 。

```
apiVersion: mq.ibm.com/v1beta1
kind: QueueManager
metadata:
  name: exampleqm
  labels:
    role: Recovery
spec:
  license:
    accept: false
    license: L-NUUP-23NH8Y
    use: Production
  queueManager:
    name: EXAMPLEQM
    availability:
      type: NativeHA
      tls:
        secretName: nhacrr-qm-replication
      nativeHAGroups:
        local:
          name: "rome"
          role: "Recovery"
    mqsc:
    - configMap:
        name: nhacrr-configmap
        items:
        - nhacrr.mqsc
    ini:
    - configMap:
        name: nhacrr-configmap
        items:
        - nhacrr.ini
```

```
    storage:
      queueManager:
        type: persistent-claim
  version:  9.4.2.0-r1
  pki:
    keys:
      - name: default
        secret:
          secretName: example-qm-tls
          items:
            - tls.key
            - tls.crt
            - ca.crt
```

6. 验证恢复组是否已正确部署：

   a) 发出 `oc get qmgr exampleqm` 命令显示队列管理器资源，等待队列管理器运行。 您将看到以下信息：

   ```
   NAME            PHASE
   exampleqm   Running
   ```

   b) 队列管理器运行后，使用 `oc get pods` 命令检查 pod 的状态。 一个 pod 应处于就绪状态：

   ```
   NAME                 READY   STATUS    RESTARTS   AGE
   exampleqm-ibm-mq-0   1/1     Running   0          2m45s
   exampleqm-ibm-mq-1   0/1     Running   0          2m45s
   exampleqm-ibm-mq-2   0/1     Running   0          2m45s
   ```

   c) 使用 `oc exec -t <leader-pod> -- dspmq -o nativeha -g` 查看恢复组的状态，其中 `<leader-pod>` 是就绪 1/1 pod 的名称（示例中为 exampleqm-ibm-mq-0 ）。

   ```
   QMNAME(EXAMPLEQM)                                         ROLE(Leader)
   INSTANCE(exampleqm-ibm-mq-0) INSYNC(yes) QUORUM(1/3) GRPLSN() GRPNAME(rome)
   GRPROLE(Recovery)
    GRPNAME(rome) GRPROLE(Recovery) GRPADDR(Unknown) GRPVER(9.4.2.0) GRSTATUS(Waiting for
   connection) RCOVLSN() RCOVTIME() BACKLOG(Unknown) INSYNC(Unknown) SYNCTIME()
   ALTDATE(2025-04-25) ALTTIME(13.48.15)
   ```

   这一地位的关键部分是

   - `Role(Leader)`
   - `QUORUM(1/3)` - 在实时组和恢复组连接之前，该值保持为 1/3
   - `GRSTATUS(Waiting for connection)` - 在 Live 群组连接时更新

7. 部署本地 HA Live 组。 现场小组部署在伦敦现场：

   要在伦敦站点创建新的实时群组，请确保处于正确的命名空间，然后在 OCP 网络控制台或使用命令行输入以下 YAML。 检查是否指定了正确的许可证，并通过将 false 更改为 true 来接受许可证。

```
apiVersion: mq.ibm.com/v1beta1
kind: QueueManager
metadata:
  name: exampleqm
  labels:
    role: Live
spec:
  license:
    accept: false
    license: L-NUUP-23NH8Y
    use: Production
  queueManager:
    name: EXAMPLEQM
    availability:
      type: NativeHA
      tls:
        secretName: nhacrr-qm-replication
      nativeHAGroups:
        local:
          name: "london"
          role: "Live"
        remotes:
          - name: "rome"
            addresses:
              - "<rome group hostname>:443"
```

```
    mqsc:
    - configMap:
        name: nhacrr-configmap
        items:
        - nhacrr.mqsc
    ini:
    - configMap:
        name: nhacrr-configmap
        items:
        - nhacrr.ini
    storage:
      queueManager:
        type: persistent-claim
  version: 9.4.2.0-r1
  pki:
    keys:
      - name: default
        secret:
          secretName: example-qm-tls
          items:
            - tls.key
            - tls.crt
            - ca.crt
```

更新 YAML 中指定实时组用于连接恢复组的地址（示例中为"<rome group hostname>:443 "）。 用罗马站点上 exampleqm-ibm-mq-nhacrr 路由的 host 属性替换它。 要检索该属性，请确保您在 Rome 网站上，并使用以下命令检索 nhacrr 路由主机名：

```
oc get route exampleqm-ibm-mq-nhacrr --template="{{.spec.host}}"
```

注：

部署 CRR 组时创建了两条路由，一条以 qm 结尾，另一条以 nhacrr 结尾。 qm 路由用于向队列管理器发送信息， nhacrr 路由用于组与组之间的通信。 始终将 nhacrr 路由的主机地址添加为 YAML 中组的远程地址。

8. 验证实时组和跨区复制。 重复步骤 6 中的操作，但这次是针对伦敦组。
   a) 发出 oc get qmgr exampleqm 命令显示队列管理器资源，等待队列管理器运行。 您将看到以下信息：

   ```
   NAME            PHASE
   exampleqm   Running
   ```

   b) 队列管理器运行后，使用 oc get pods 命令检查 pod 的状态。 一个 pod 应处于就绪状态：

   ```
   NAME                 READY   STATUS    RESTARTS   AGE
   exampleqm-ibm-mq-0   1/1     Running   0          2m45s
   exampleqm-ibm-mq-1   0/1     Running   0          2m45s
   exampleqm-ibm-mq-2   0/1     Running   0          2m45s
   ```

   c) 使用 oc exec -t <active-pod> -- dspmq -o nativeha -g 查看 Live 组的状态，其中 <active-pod> 是就绪 1/1 pod 的名称（示例中为 exampleqm-ibm-mq-0 ）。

   ```
   QMNAME(EXAMPLEQM)                                         ROLE(Active)
   INSTANCE(exampleqm-ibm-mq-0) INSYNC(yes) QUORUM(3/3) GRPLSN(<0:0:87:58322>)
   GRPNAME(london) GRPROLE(Live)
    GRPNAME(london) GRPROLE(Live) GRPADDR(Unknown) GRPVER(9.4.2.0) GRSTATUS(Normal)
   RCOVLSN(<0:0:87:58322>) RCOVTIME(2025-05-16T14:27:09.051799Z) INITLSN(<0:0:9:4696>)
   INITTIME(2025-05-12T16:58:13.582274Z) LIVETIME(2025-05-12T16:58:16.319844Z)
   ALTDATE(2025-05-16) ALTTIME(14.27.10)
    GRPNAME(rome) GRPROLE(Recovery) GRPADDR(exampleqm-ibm-mq-nhacrr-
   <namespace>.apps.<cluster>) GRPVER(9.4.2.0) CONNGRP(yes) GRSTATUS(Normal)
   RCOVLSN(<0:0:87:58322>) RCOVTIME(2025-05-16T14:27:09.051799Z) BACKLOG(0) INSYNC(yes)
   SYNCTIME(2025-05-16T14:28:38.084565Z) ALTDATE(2025-05-16) ALTTIME(14.28.31)
   ```

   这一地位的关键部分是

   - Role(Active)
   - QUORUM(3/3) - 直播小组的法定人数已满
   - GRSTATUS(Normal)

d) 在 Rome 网站上使用 `oc exec -t <leader-pod> -- dspmq -o nativeha -g` 命令，再次检查恢复组的状态。

```
QMNAME(EXAMPLEQM)                                         ROLE(Leader)
INSTANCE(exampleqm-ibm-mq-0) INSYNC(yes) QUORUM(3/3) GRPLSN(<0:0:87:58322>)
GRPNAME(rome) GRPROLE(Recovery)
 GRPNAME(rome) GRPROLE(Recovery) GRPADDR(exampleqm-ibm-mq-nhacrr-
<namespace>.apps.<cluster>) GRPVER(9.4.3.0) GRSTATUS(Normal) RCOVLSN(<0:0:87:58322>)
RCOVTIME(2025-05-16T14:27:09.051799Z) BACKLOG(0) INSYNC(yes)
SYNCTIME(2025-05-16T14:30:40.134627Z) ALTDATE(2025-05-16) ALTTIME(14.27.17)
 GRPNAME(london) GRPROLE(Live) GRPADDR(Unknown) GRPVER(9.4.3.0) CONNGRP(yes)
GRSTATUS(Normal) RCOVLSN(<0:0:87:58322>) RCOVTIME(2025-05-16T14:27:09.051799Z)
INITLSN(<0:0:9:4696>) INITTIME(2025-05-12T16:58:13.582274Z)
LIVETIME(2025-05-12T16:58:16.319844Z) ALTDATE(2025-05-16) ALTTIME(14.30.41)
```

请注意， `QUORUM(3/3)` 和 `GRSTATUS(Normal)` 字段已更新。

9. 测试与队列管理器的连接。 要确认队列管理器已配置且可用，请按照 第 94 页的『从笔记本电脑测试到队列管理器的相互 TLS 连接』 中的步骤操作，确保在步骤 2 中确定的队列管理器主机名是伦敦站点路由的主机名。
10. 将远程地址添加到本地 HA 恢复组。 当罗马成为实时群组时，它需要知道伦敦的远程地址，以便作为恢复群组连接到伦敦。 要添加此功能，请更新罗马站点的 `spec.queueManager.availability.nativeHAGroups` ，在本地协议下方添加以下内容：

```
          remotes:
            - name: "london"
              addresses:
              - "<london group hostname>:443"
```

将地址替换为伦敦站点 `exampleqm-ibm-mq-nhacrr` 路由的主机属性。 确保您位于伦敦站点，并使用以下命令获取 nhacrr 路由的主机名：

```
oc get route exampleqm-ibm-mq-nhacrr --template="{{.spec.host}}"
```

更新完成后，等待 pod 重新启动、运行，并等待一个 pod 成为准备就绪的 1/1 pod。

11. 将 Rome 网站主机名添加到 `ccdt.json` 文件。 切换后，罗马成为实时站点，客户需要连接到罗马的群组。 将 Rome 的主机名添加到 `ccdt.json` 文件可方便重新连接。

确保你在 Rome 网站上，并使用以下命令获取 qm 路由的主机名：

```
oc get route exampleqm-ibm-mq-qm --template="{{.spec.host}}"
```

注: 有两条路线，在这种情况下，应该使用报文流量的路线。 您应将 qm 路由的主机地址添加到 `ccdt.json` 文件中。

在 `ccdt.json` 文件中，在连接列表中添加第二个主机名和端口对，这样连接列表就会变成这样：

```
"connection":
                [
                    {
                        "host": "<hostname from London qm route>",
                        "port": 443
                    },
                    {
                        "host": "<hostname from Rome qm route>",
                        "port": 443
                    }
                ],
```

12. 向队列中添加一些持久信息。 在 ConfigMap 中指定的 `mqsc` 文件中定义的队列具有 `DEFPSIST(YES)` 属性，因此放入队列的所有报文都将是持久的，并复制到恢复组。 使用以下命令将信息放入队列：

```
/opt/mqm/samp/bin/amqsputc EXAMPLE.QUEUE EXAMPLEQM
```

如果与队列管理器的连接成功，则输出以下响应：

```
target queue is EXAMPLE.QUEUE
```

输入一些文字，然后每次都按**回车键** ，即可将多条信息放入队列。 按两次**回车键**完成。

13. 从伦敦的实时群组到罗马的恢复群组执行托管切换，以证明本地 HA - CRR 设置是正确的。 请遵循第 208 页的『在原生高可用性 CRR 配置上完成计划的切换』中的指示信息。 在更新 "实时 "组和 "恢复 "组的角色时，您可能会发现在 YAML 中更新 `metadata.labels.role` 以反映您的更改非常有用，但这对于切换来说并非必要。 完成切换后，您可以在罗马检索新实时群组中的信息。

    有关故障切换和切换的更多信息，请参阅 第 208 页的『本地 HA CRR 切换和故障切换』。
14. 运行以下命令，检索切换前发送的信息：

    ```
    /opt/mqm/samp/bin/amqsgetc EXAMPLE.QUEUE EXAMPLEQM
    ```

    您在步骤 11 中添加的信息可用，并将输出。 几秒钟后，命令退出。

下一步做什么

维护完成后，可以通过重复步骤 12 和 13 中的过程切换回原来的实时组。

您也可以尝试使用 第 209 页的『在原生高可用性 CRR 配置中完成计划外的故障切换』 上的说明进行计划外故障转移。

要进一步了解如何使用不同的证书来保护 CRR 配置的不同部分，请参阅 第 113 页的『示例：使用本地 HA CRR 的多个 TLS 证书』。

示例：使用 *IBM MQ* 操作员将恢复组添加到现有本地 *HA* 配置中

本示例使用 IBM MQ Operator 将队列管理器部署到 OpenShift® Container Platform 中，使用简单的本地 HA 配置。 该示例添加了恢复组，然后将本地 HA 队列管理器更新为实时组，将配置扩展为本地 HA 跨区域复制 (CRR)。

开始之前

要完成此示例，必须首先完成以下步骤：

- 在命令行上登录正在使用的 OpenShift Container Platform (OCP) 群集。
- 为此示例创建两个 OCP 项目/命名空间。 示例将提及伦敦和罗马网站。 这些指的是伦敦和罗马在一个群集中的两个命名空间，或者是每个群集中每个站点的一个命名空间。
- 确保在群集中安装了 IBM MQ Operator 3.5.0 或更高版本，并在命名空间中可用。
- 确保已安装 OpenSSL 命令行工具。
- 完成 第 75 页的『通过创建拉取秘钥，为使用 "Kubernetes 做准备』 任务。
- 请阅读 第 85 页的『使用 IBM MQ Operator 部署简单队列管理器』，以便熟悉使用 IBM MQ 操作符。

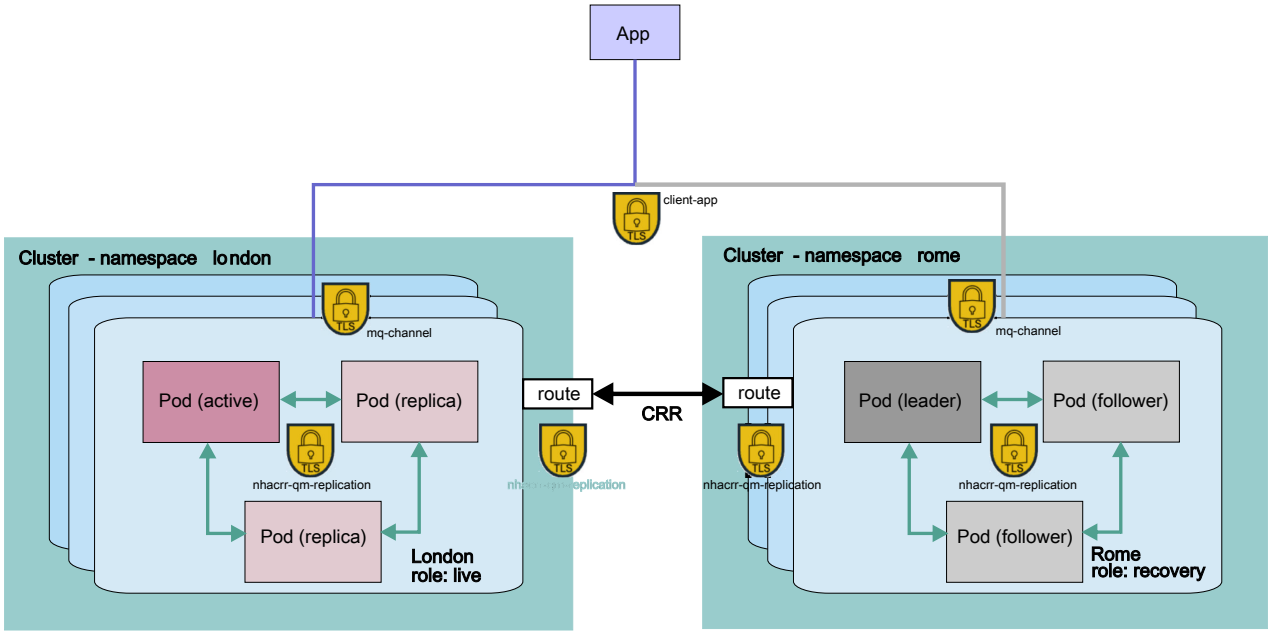
关于此任务

CRR 组通常部署在不同地点的两个 OCP 集群中，但本示例允许您根据需要将两个组部署到同一个集群中。 有关本地 HA CRR 的概述，请参阅本地 HA 跨区域复制。

为简单起见，本例使用单个证书来保护所有队列管理器复制流量，该证书名为 nhacrr-qm-replication。

要了解如何在 Native HA CRR 中使用证书，以及如何使用不同的证书来确保组和实例流量的安全，请参阅 第 113 页的『示例：使用本地 HA CRR 的多个 TLS 证书』 Deploying Native HA CRR with multiple certificates。

要点： 这些示例旨在帮助您入门，并不适合生产环境。 对于高级用户来说，证书管理是一个复杂的课题。 对于生产，你必须考虑到轮换、撤销、密钥长度、灾难恢复等问题。



本示例提供自定义资源 YAML，用于定义跨区域复制中使用的两个本地 HA 队列管理器和恢复组。

过程

1. 在伦敦站点创建本地 HA 队列管理器。 确保您处于正确群集上的伦敦命名空间，并按照 第 98 页的『示例: 使用 IBM MQ Operator 配置本机 HA』 中的说明创建本地 HA 配置。 (在完成此步骤时，您将创建图中名为 client-app 和 mq-channel 的两个证书)
2. 在 CRR 配置中创建用于队列管理器复制流量的证书。 使用以下命令创建私钥，并用它为内部证书颁发机构颁发自签名证书。 然后为队列管理器创建私钥和证书，并与内部证书颁发机构签署。

```
openssl genpkey -algorithm rsa -pkeyopt rsa_keygen_bits:4096 -out nhacrr-ca.key
openssl req -x509 -new -nodes -key "nhacrr-ca.key" -sha512 -days 30 -subj "/CN=nhacrr-
selfsigned-ca" -out nhacrr-ca.crt
openssl req -new -nodes -out nhacrr-qm-replication.csr -newkey rsa:4096 -keyout nhacrr-qm-
replication.key -subj "/CN=nhacrr-qm-replication"
openssl x509 -req -in nhacrr-qm-replication.csr -CA nhacrr-ca.crt -CAkey nhacrr-ca.key
-CAcreateserial -out nhacrr-qm-replication.crt -days 7 -sha512
```

   有关这些命令的更多信息，请参阅 第 90 页的『使用 OpenSSL 创建自签名 PKI』。
3. 创建两个 Kubernetes 保密文件，其中包含队列管理器密钥和证书。 在罗马和伦敦站点上创建名为 nhacrr-qm-replication 的秘密，用于存储队列管理器实例的密钥和证书。 使用下面的命令创建秘密，在运行该命令之前，请更改到每个命名空间（罗马，而不是伦敦）。

```
kubectl create secret generic nhacrr-qm-replication --type="kubernetes.io/tls" --from-
file=tls.key=nhacrr-qm-replication.key --from-file=tls.crt=nhacrr-qm-replication.crt --from-
file=nhacrr-ca.crt
```

4. 更新本地 HA 队列管理器，为复制流量使用新的密文。 要更新用于内部群组流量的证书，以及稍后用于群组间流量的证书，请确保您位于伦敦命名空间，然后将 OCP 网络控制台 QueueManager YAML 中的 `.spec.queuemanager.availability.tls.secretname` 属性更改为

```
      tls:
        secretName: nhacrr-qm-replication
```

5. 在 Rome 网站创建一个秘密，其中包含之前在 第 98 页的『示例: 使用 IBM MQ Operator 配置本机 HA』 第 1 步中生成的证书和密钥。 登录群集，切换到 Rome 命名空间，在创建证书文件的目录下运行以下命令来创建秘密：

```
   kubectl create secret generic example-qm-tls --type="kubernetes.io/tls" --from-
file=tls.key=example-qm.key --from-file=tls.crt=example-qm.crt --from-file=ca.crt
```

6. 在 Rome 中创建包含 MQSC 命令和 INI 文件的配置图。 在罗马复制在伦敦网站上创建的配置地图。 确保处于罗马命名空间，在 OCP 网络控制台输入以下 YAML，选择 `ConfigMaps`，然后选择 `Create ConfigMap`，或使用 `oc apply` 命令。

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: example-nativeha-configmap
data:
  example-tls.mqsc: |
    DEFINE CHANNEL('MTLS.SVRCONN') CHLTYPE(SVRCONN) SSLCAUTH(REQUIRED)
SSLCIPH('ANY_TLS13_OR_HIGHER') REPLACE
    SET CHLAUTH('MTLS.SVRCONN') TYPE(SSLPEERMAP) SSLPEER('CN=*') USERSRC(NOACCESS)
ACTION(REPLACE)
    SET CHLAUTH('MTLS.SVRCONN') TYPE(SSLPEERMAP) SSLPEER('CN=example-app1') USERSRC(MAP)
MCAUSER('app1') ACTION(REPLACE)
    SET AUTHREC PRINCIPAL('app1') OBJTYPE(QMGR) AUTHADD(CONNECT,INQ)
    DEFINE QLOCAL('EXAMPLE.QUEUE') REPLACE
    SET AUTHREC PROFILE('EXAMPLE.QUEUE') PRINCIPAL('app1') OBJTYPE(QUEUE)
AUTHADD(BROWSE,PUT,GET,INQ)
  example-tls.ini: |
    Service:
        Name=AuthorizationService
        EntryPoints=14
        SecurityPolicy=UserExternal
```

MQSC 定义了一个名为 MTLS.SVRCONN 的通道和一个名为 EXAMPLE.QUEUE 的队列。 该通道被配置为只允许提交 CN 为 example-app1 的证书的客户端访问，如步骤 4 中创建的示例-qm-tls 中存储的证书。

7. 部署本地 HA 恢复组。 使用以下自定义资源 YAML，在罗马站点上创建一个新的恢复组。 确保处于 Rome 命名空间，然后在 OCP 网络控制台或使用 `oc apply` 命令输入以下 YAML 来创建队列管理器。 检查是否指定了正确的许可证，并通过将 false 更改为 true 来接受许可证。

   注: 您必须在实时组和恢复组中使用相同的队列管理器名称。 默认情况下，如果不指定 `.spec.queuemanager.name`，则默认为 `.metadata.name`。 您可以使用以下方法之一实现名称匹配:

   - 在每个群集中为队列管理器资源取相同的名称（即 `.metadata.name` 匹配），不要覆盖该名称。
   - 在 `.metadata.name` 中单独命名队列管理器，并用 `.spec.queuemanager.name` 覆盖该名称，以便两个站点的名称一致。

   在本例中，使用 `.metadata.name` 将队列管理器名称设置为 `exampleqm`。

```
apiVersion: mq.ibm.com/v1beta1
kind: QueueManager
metadata:
  name: exampleqm
  labels:
    role: Recovery
spec:
  license:
    accept: false
    license: L-NUUP-23NH8Y
    use: Production
  queueManager:
    name: EXAMPLEQM
    availability:
      type: NativeHA
      tls:
        secretName: nhacrr-qm-replication
      nativeHAGroups:
        local:
          name: "rome"
          role: "Recovery"
    mqsc:
    - configMap:
        name: example-nativeha-configmap
        items:
        - example-tls.mqsc
    ini:
    - configMap:
        name: example-nativeha-configmap
        items:
        - example-tls.ini
```

```
  storage:
    queueManager:
      type: persistent-claim
version: 9.4.2.0-r1
pki:
  keys:
    - name: default
      secret:
        secretName: example-qm-tls
        items:
          - tls.key
          - tls.crt
          - ca.crt
```

8. 验证组已正确部署（此命令应在 Rome 网站上运行）。
   a) 发出 `oc get qmgr exampleqm` 命令显示队列管理器资源，等待队列管理器运行：

```
NAME            PHASE
exampleqm   Running
```

   b) 队列管理器运行时，可以使用 `oc get pods` 检查 pod 的状态，其中一个 pod 应处于就绪状态：

```
NAME                 READY   STATUS    RESTARTS   AGE
exampleqm-ibm-mq-0   1/1     Running   0          2m45s
exampleqm-ibm-mq-1   0/1     Running   0          2m45s
exampleqm-ibm-mq-2   0/1     Running   0          2m45s
```

   c) 发出以下命令检查恢复组的状态：

```
oc exec -t <leader-pod> -- dspmq -o nativeha -g
```

   其中 `<leader-pod>` 是就绪 1/1 pod 的名称（示例中为 `exampleqm-ibm-mq-0`）：

```
QMNAME(EXAMPLEQM)                                         ROLE(Leader)
INSTANCE(exampleqm-ibm-mq-0) INSYNC(yes) QUORUM(1/3) GRPLSN() GRPNAME(rome)
GRPROLE(Recovery)
 GRPNAME(rome) GRPROLE(Recovery) GRPADDR(Unknown) GRPVER(9.4.2.0) GRSTATUS(Waiting for
connection) RCOVLSN() RCOVTIME() BACKLOG(Unknown) INSYNC(Unknown) SYNCTIME()
ALTDATE(2025-04-25) ALTTIME(13.48.15)
```

   这一地位的关键部分是

   - `Role(Leader)`
   - `QUORUM(1/3)` - 在实时组和恢复组连接之前，该值保持为 1/3
   - `GRSTATUS(Waiting for connection)` - 在 Live 群组连接时更新

9. 更新本地 HA 队列管理器以使用实时组。
   a) 在 Rome 网站上使用以下命令读取恢复组地址：

```
oc get route exampleqm-ibm-mq-nhacrr --template="{{.spec.host}}"
```

   注： 部署 CRR 组时创建了两条路由，一条以 `qm` 结尾，另一条以 `nhacrr` 结尾。 `qm` 路由用于向队列管理器发送信息， `nhacrr` 路由用于组与组之间的通信。 确保在下面的 YAML 中添加 `nhacrr` 路由的主机地址作为组的远程地址。

   b) 将下面 YAML 中的地址替换为命令返回的主机名，该主机名指定了实时组用于连接恢复组的地址。 在 OCP 网络控制台的伦敦 QueueManager YAML 中，直接在 `type: NativeHA` 属性下添加以下 YAML：

```
      nativeHAGroups:
        local:
          name: "london"
          role: "Live"
        remotes:
          - name: "rome"
            addresses:
              - "<rome group hostname>:443"
```

   在保存 YAML 之前，在元数据章节的顶部添加 Live 角色标签：

```
metadata:
  labels:
    role: Live
```

10. 验证实时组和跨区复制。

    a) 在伦敦站点上，发出 `oc get qmgr exampleqm` 命令显示队列管理器资源，并等待队列管理器运行。 您将看到以下信息：

    ```
    NAME            PHASE
    exampleqm   Running
    ```

    b) 队列管理器运行后，使用 `oc get pods` 命令检查 pod 的状态。 一个 pod 应处于就绪状态：

    ```
    NAME                 READY   STATUS    RESTARTS   AGE
    exampleqm-ibm-mq-0   1/1     Running   0          2m45s
    exampleqm-ibm-mq-1   0/1     Running   0          2m45s
    exampleqm-ibm-mq-2   0/1     Running   0          2m45s
    ```

    c) 发出以下命令检查实时组的状态：

    ```
    oc exec -t <active-pod> -- dspmq -o nativeha -g
    ```

    其中 `<active-pod>` 是就绪 1/1 pod 的名称（示例中为 `exampleqm-ibm-mq-0`）：

    ```
    QMNAME(EXAMPLEQM)                                         ROLE(Active)
    INSTANCE(exampleqm-ibm-mq-0) INSYNC(yes) QUORUM(3/3) GRPLSN(<0:0:13:21327>)
    GRPNAME(london) GRPROLE(Live)
    GRPNAME(london) GRPROLE(Live) GRPADDR(Unknown) GRPVER(9.4.3.0) GRSTATUS(Normal)
    RCOVLSN(<0:0:13:21327>) RCOVTIME(2025-06-04T16:22:54.414645Z) INITLSN(<0:0:9:5720>)
    INITTIME(2025-06-04T16:12:09.629132Z) LIVETIME(2025-06-04T16:12:11.825065Z)
    ALTDATE(2025-06-04) ALTTIME(16.23.01)
    GRPNAME(rome) GRPROLE(Recovery) GRPADDR(exampleqm-ibm-mq-nhacrr-
    <namespace>.apps.<cluster>) GRPVER(9.4.3.0) CONNGRP(yes) GRSTATUS(Normal)
    RCOVLSN(<0:0:13:21327>) RCOVTIME(2025-06-04T16:22:54.414645Z) BACKLOG(0) INSYNC(yes)
    SYNCTIME(2025-06-04T16:29:17.206996Z) ALTDATE(2025-06-04) ALTTIME(16.29.12)
    ```

    这一地位的关键部分是

    - `Role(Active)`
    - `QUORUM(3/3)` - 直播小组的法定人数已满
    - `GRSTATUS(Normal)`

    d) 在罗马站点运行同样的命令，观察 `QUORUM(3/3)` 和 `GRSTATUS(Normal)` 是否也已更新：

    ```
    QMNAME(EXAMPLEQM)                                         ROLE(Leader)
    INSTANCE(exampleqm-ibm-mq-0) INSYNC(yes) QUORUM(3/3) GRPLSN(<0:0:87:58322>)
    GRPNAME(rome) GRPROLE(Recovery)
     GRPNAME(rome) GRPROLE(Recovery) GRPADDR(exampleqm-ibm-mq-nhacrr-
    rome.apps.destructive-1.cp.fyre.ibm.com) GRPVER(9.4.3.0) GRSTATUS(Normal)
    RCOVLSN(<0:0:87:58322>) RCOVTIME(2025-05-16T14:27:09.051799Z) BACKLOG(0) INSYNC(yes)
    SYNCTIME(2025-05-16T14:30:40.134627Z) ALTDATE(2025-05-16) ALTTIME(14.27.17)
     GRPNAME(london) GRPROLE(Live) GRPADDR(Unknown) GRPVER(9.4.3.0) CONNGRP(yes)
    GRSTATUS(Normal) RCOVLSN(<0:0:87:58322>) RCOVTIME(2025-05-16T14:27:09.051799Z)
    INITLSN(<0:0:9:4696>) INITTIME(2025-05-12T16:58:13.582274Z)
    LIVETIME(2025-05-12T16:58:16.319844Z) ALTDATE(2025-05-16) ALTTIME(14.30.41)
    ```

11. 将远程地址添加到本地 HA 恢复组。 当 Rome 成为实时群组时，它需要知道 London 的远程地址，以便作为恢复群组连接到 London。 要添加此功能，请更新罗马站点的 `spec.queueManager.availability.nativeHAGroups`，在本地协议下面添加以下内容：

    ```
        remotes:
          - name: "london"
            addresses:
              - "<london group hostname>:443"
    ```

    将 <london group hostname> 替换为伦敦站点 `exampleqm-ibm-mq-nhacrr` 路由的主机属性。 确保您位于伦敦站点，并使用以下命令获取 nhacrr 路由的主机名：

    ```
    oc get route exampleqm-ibm-mq-nhacrr --template="{{.spec.host}}"
    ```

更新完成后，等待 pod 重新启动、运行，并等待一个 pod 成为准备就绪的 1/1 pod。

12. 将 Rome 网站主机名添加到 `ccdt.json` 文件。 在切换后，罗马成为实时站点，客户端需要连接到罗马的群组，在 `ccdt.json` 文件中添加罗马的主机名就可以实现这一点。

    a) 确保你在 Rome 网站上，并使用以下命令获取 qm 路由的主机名：

    ```
    oc get route exampleqm-ibm-mq-qm --template="{{.spec.host}}"
    ```

    注: 有两条路由，在这种情况下，使用报文流量路由。 将 qm 路由的主机地址添加到 `ccdt.json`。

    b) 在 `ccdt.json` 文件中，在连接列表中添加第二个主机名和端口对，这样连接列表就会变成这样：

    ```
    "connection":
    [
        {
            "host": "<hostname from London qm route>",
            "port": 443
        },
        {
            "host": "<hostname from Rome qm route>",
            "port": 443
        }
    ],
    ```

13. 执行从伦敦到罗马的托管切换，以证明本地 HA - CRR 设置是正确的，然后执行从伦敦的实时组到罗马的恢复组之间的托管切换。 请遵循第 208 页的『在原生高可用性 CRR 配置上完成计划的切换』中的指示信息。 在更新 "实时 "组和 "恢复 "组的角色时，您可能会发现在 YAML 中更新 `metadata.labels.role` 以反映变化是有用的，但这对切换来说并非必要。 完成切换后，您可以在罗马检索新实时群组中的信息。

    有关故障切换和切换的更多信息，请参阅 第 208 页的『本地 HA CRR 切换和故障切换』。

下一步做什么

维护完成后，可以通过重复步骤 12 和 13 中的过程切换回原来的实时组。

您也可以尝试使用 第 209 页的『在原生高可用性 CRR 配置中完成计划外的故障切换』 上的说明进行计划外故障转移。

最后，要进一步了解如何使用不同的证书来保护 CRR 配置的不同部分，请参阅为 Native HA CRR 配置 TLS。

示例：使用本地 *HA CRR* 的多个 *TLS* 证书

开始之前

在开始本例之前，必须首先完成 第 102 页的『示例：使用 IBM MQ Operator 部署简单的本地 HA CRR 配置』（本例假定部署已存在，且 London 组为实时组）。

然后完成以下步骤：

- 确保可以登录 OpenShift Container Platform (OCP) 集群，并选择相关项目/命名空间。（第 102 页的『示例：使用 IBM MQ Operator 部署简单的本地 HA CRR 配置』 中描述的示例创建了两个 OpenShift Container Platform (OCP) 项目/命名空间，或者在单个群集中为伦敦和罗马创建了两个命名空间，或者在每个群集中为每个站点创建了一个命名空间。 本示例也将其称为伦敦和罗马站点）
- 确保在群集中安装了 IBM MQ Operator 3.5.0 或更高版本，并在命名空间中可用。
- 确保已安装 OpenSSL 命令行工具。
- 完成 第 75 页的『通过创建拉取秘钥，为使用 "Kubernetes 做准备』 任务。
- 请阅读 第 85 页的『使用 IBM MQ Operator 部署简单队列管理器』，以便熟悉使用 IBM MQ 操作符。

关于此任务

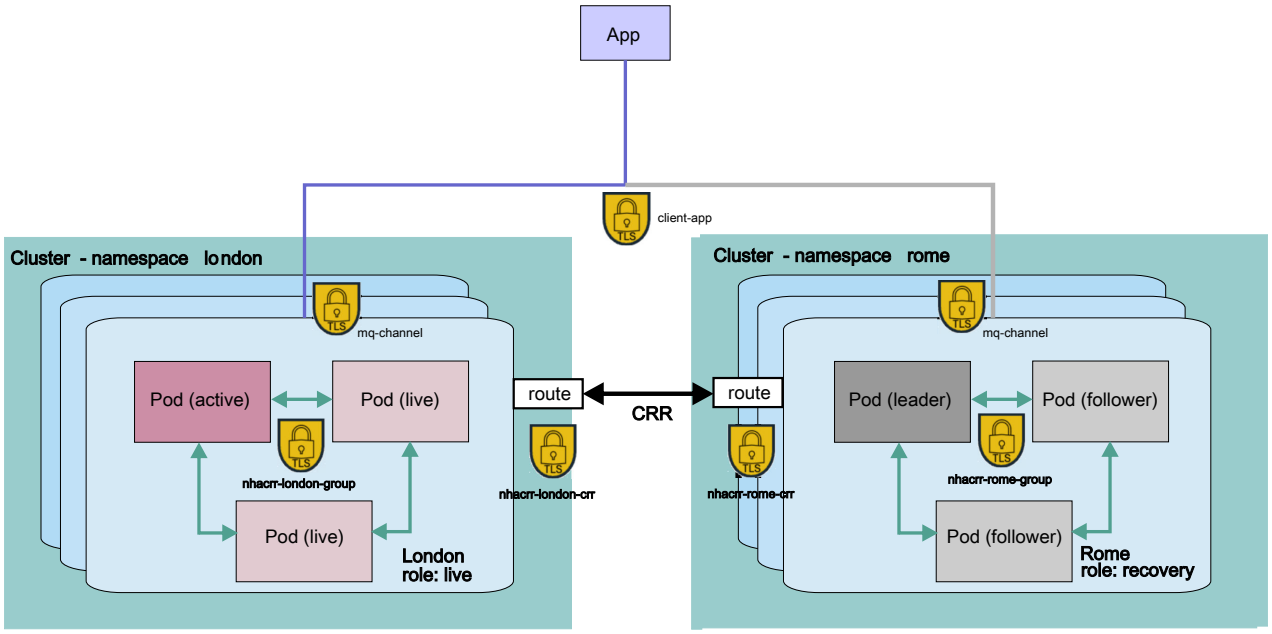
本示例首先概述了如何在本地 HA CRR 配置中使用 TLS 和证书，然后使用简单步骤演示了如何确保 `nhacrr-london-group`, `nhacrr-rome-group`, `nhacrr-london-crr` 和 `nhacrr-rome-crr` 流量的安全。

下面程序部分的步骤将指导你更改 第 102 页的『示例：使用 IBM MQ Operator 部署简单的本地 HA CRR 配置』 部署，以便为每种类型的流量使用不同的证书，并控制使用哪个 CipherSpec。

为本地 HA CRR 配置 TLS

下图显示了本地 HA CRR 配置中使用证书的位置。 证书的使用方法如下：

1. 为确保队列管理器和客户端应用程序之间的通信安全， `client-app` 和 `mq-channel`
2. 确保队列管理器复制流量安全 - `nhacrr-london-group`, `nhacrr-rome-group`, `nhacrr-london-crr` 和 `nhacrr-rome-crr`



队列管理器与应用程序之间的安全通信

图中所示的两个本地 HA 组共同组成了一个高度可用的队列管理器，客户端应用程序可通过实时组连接到该队列管理器。 切换发生时，应用程序会断开连接，必须重新连接到新的实时组。 (将这两个组的连接详情添加到应用程序连接列表可方便重新连接)

如果通过路由连接的应用程序不在其中一个或两个群组中，或与其中一个或两个群组在不同的群组中，则需要使用私钥和证书通过 TLS 进行通信。 图中，客户端的证书和密钥标为 `client-app`，队列管理器的证书和密钥标为 `mq-channel`。 `mq-channel` 证书和私钥必须作为两个机密添加，每个组一个。

第 102 页的『示例：使用 IBM MQ Operator 部署简单的本地 HA CRR 配置』 会指导你创建这些证书、密钥和秘密。 以下步骤主要针对队列管理器复制流量的证书，不更改 `mq-channel` 或 `client-app` 证书。

确保队列管理器复制流量的安全

组内和组间的流量均可复制。 两个群组之间的网络通信必须始终保持安全，但群组内的通信可以是纯文本，因为群组内的通信一般无法访问。

1. `nhacrr-london-group` 和 代表用于确保每个组内流量安全的证书和私钥。 `nhacrr-rome-group`
2. `nhacrr-london-crr` 和 表示用于确保跨组流量每一端安全的证书和私钥。 `nhacrr-rome-crr`

在示例：使用 IBM MQ Operator 部署简单的 Native HA CRR 配置中，相同的证书和私钥存储在 `nhacrr-qm-replication` 每个命名空间中的 secret 用于保护 `nhacrr-london-group`， `nhacrr-rome-group`， `nhacrr-london-crr` 和 `nhacrr-rome-crr` 交通。

要点: 这些示例旨在帮助您入门，并不适合生产环境。 对于高级用户来说，证书管理是一个复杂的课题。 在生产过程中，你必须考虑轮换、撤销、密钥长度、灾难恢复等问题。

本例向您展示了如何为每种流量使用不同的证书。

过程

1. 创建四套密钥和证书，用于队列管理器复制流量。 要为本例创建这些内容，请运行以下命令。 每组命令都会创建私钥，用它为内部证书颁发机构颁发自签证书，然后创建私钥和与内部证书颁发机构签署的证书。 重复此练习，为 nhacrr-london-group, nhacrr-rome-group, nhacrr-london-crr 和 nhacrr-rome-crr 流量创建证书。

   有关命令的详细说明，请参见 第 90 页的『使用 OpenSSL 创建自签名 PKI』。

   注: 本示例为每个证书链创建了单独的 CA，以便在需要时演示如何在组之间建立信任。 同一组织的证书通常使用共同的 CA，在这种情况下，不需要队列管理器 YAML 中的信任条款，因为 CA 已经被信任。

```
openssl genpkey -algorithm rsa -pkeyopt rsa_keygen_bits:4096 -out nhacrr-london-group-
ca.key openssl req -x509 -new -nodes -key
"nhacrr-london-group-ca.key" -sha512 -days 30 -subj "/CN=nhacrr-london-group-selfsigned-ca"
-out nhacrr-london-group-ca.crt openssl
req -new -nodes -out nhacrr-london-group.csr -newkey rsa:4096 -keyout nhacrr-london-
group.key -subj "/CN=nhacrr-london-group"
openssl x509 -req -in nhacrr-london-group.csr -CA nhacrr-london-group-ca.crt -CAkey nhacrr-
london-group-ca.key -CAcreateserial -out
nhacrr-london-group.crt -days 7 -sha512
```

```
openssl genpkey -algorithm rsa -pkeyopt rsa_keygen_bits:4096 -out nhacrr-rome-group-ca.key
openssl req -x509 -new -nodes -key
"nhacrr-rome-group-ca.key" -sha512 -days 30 -subj "/CN=nhacrr-rome-group-selfsigned-ca"
-out nhacrr-rome-group-ca.crt openssl req -new
-nodes -out nhacrr-rome-group.csr -newkey rsa:4096 -keyout nhacrr-rome-group.key -subj "/
CN=nhacrr-rome-group" openssl x509
-req -in nhacrr-rome-group.csr -CA nhacrr-rome-group-ca.crt -CAkey nhacrr-rome-group-ca.key
-CAcreateserial -out nhacrr-rome-group.crt
-days 7 -sha512
```

```
openssl genpkey -algorithm rsa -pkeyopt rsa_keygen_bits:4096 -out nhacrr-london-crr-ca.key
openssl req -x509 -new -nodes -key
"nhacrr-london-crr-ca.key" -sha512 -days 30 -subj "/CN=nhacrr-london-crr-selfsigned-ca"
-out nhacrr-london-crr-ca.crt openssl req -new -nodes
-out nhacrr-london-crr.csr -newkey rsa:4096 -keyout nhacrr-london-crr.key -subj "/CN=nhacrr-
london-crr" openssl x509 -req -in
nhacrr-london-crr.csr -CA nhacrr-london-crr-ca.crt -CAkey nhacrr-london-crr-ca.key
-CAcreateserial -out nhacrr-london-crr.crt -days 7 -sha512
```

```
openssl genpkey -algorithm rsa -pkeyopt rsa_keygen_bits:4096 -out nhacrr-rome-crr-ca.key
openssl req -x509 -new -nodes -key
"nhacrr-rome-crr-ca.key" -sha512 -days 30 -subj "/CN=nhacrr-rome-crr-selfsigned-ca" -out
nhacrr-rome-crr-ca.crt openssl req -new -nodes -out
nhacrr-rome-crr.csr -newkey rsa:4096 -keyout nhacrr-rome-crr.key -subj "/CN=nhacrr-rome-
crr" openssl x509 -req -in nhacrr-rome-crr.csr
-CA nhacrr-rome-crr-ca.crt -CAkey nhacrr-rome-crr-ca.key -CAcreateserial -out nhacrr-rome-
crr.crt -days 7 -sha512
```

2. 创建 Kubernetes 保密文件，其中包含用于 Rome 的密钥和证书。 使用以下命令在 Rome 网站上创建两个 TLS 秘密：

   a. 一个证书确保内部 Rome 组流量的安全。

   b. 一张证书，确保来自罗马组的跨组流量。

```
kubectl create secret generic nhacrr-rome-group --type="kubernetes.io/tls" --from-
file=tls.key=nhacrr-rome-group.key --from-file=tls.crt=nhacrr-rome-group.crt --from-
file=nhacrr-rome-group-ca.crt
kubectl create secret generic nhacrr-rome-crr --type="kubernetes.io/tls" --from-
file=tls.key=nhacrr-rome-crr.key --from-file=tls.crt=nhacrr-rome-crr.crt --from-file=nhacrr-
rome-crr-ca.crt
```

   现在使用以下命令创建两个包含罗马证书的机密，供伦敦组信任：

```
kubectl create secret generic nhacrr-london-group-trust --from-file=tls.crt=nhacrr-london-
group.crt --from-file=ca.crt=nhacrr-london-group-ca.crt --type=opaque
kubectl create secret generic nhacrr-london-crr-trust --from-file=tls.crt=nhacrr-london-
crr.crt  --from-file=ca.crt=nhacrr-london-crr-ca.crt --type=opaque
```

3. 创建 Kubernetes 保密文件，其中包含伦敦的密钥和证书。 使用以下命令在伦敦站点创建两个 TLS 保密协议：

   a. 一个证书确保伦敦群组内部流量的安全。

   b. 一份证书，确保来自伦敦组的跨组流量。

```
kubectl create secret generic nhacrr-london-group --type="kubernetes.io/tls" --from-
file=tls.key=nhacrr-london-group.key --from-file=tls.crt=nhacrr-london-group.crt --from-
file=nhacrr-london-group-ca.crt
kubectl create secret generic nhacrr-london-crr --type="kubernetes.io/tls" --from-
file=tls.key=nhacrr-london-crr.key --from-file=tls.crt=nhacrr-london-crr.crt --from-
file=nhacrr-london-crr-ca.crt
```

   现在使用以下命令创建两个包含伦敦证书的机密，供 Rome 组信任：

```
kubectl create secret generic nhacrr-rome-group-trust --from-file=tls.crt=nhacrr-rome-
group.crt --from-file=ca.crt=nhacrr-rome-group-ca.crt --type=opaque
kubectl create secret generic nhacrr-rome-crr-trust --from-file=tls.crt=nhacrr-rome-
crr.crt  --from-file=ca.crt=nhacrr-rome-crr-ca.crt --type=opaque
```

4. 使用不同的证书确保罗马和伦敦站点的流量安全。 在 第 102 页的『示例：使用 IBM MQ Operator 部署简单的本地 HA CRR 配置』 中创建的部署使用 nhacrr-qm-replication 秘密保护所有复制流量。 此步骤更改配置，以分别使用 nhacrr-rome-group 和 nhacrr-london-group 保密信息确保每个组内流量的安全。

   .spec.queuemanager.availability.tls.secretName 属性指定了用于确保网站组内流量安全的秘密。 在没有定义其他 TLS 属性的情况下，该秘密也用于确保组间流量的安全。

   a) 在 Rome 网站上，将 OCP 网络控制台 QueueManager YAML 中的 .spec.queuemanager.availability.tls.secretName 属性改为 nhacrr-rome-group：

```
      tls:
        secretName: nhacrr-rome-group
```

   b) 在 YAML 中添加 spec.queueManager.availability.NativeHAGroups.remotes.trust stanza，并将 nhacrr-london-group-trust secret 作为要信任的秘密：

```
...
      nativeHAGroups:
        local:
          name: "rome"
          role: "Recovery"
        remotes:
          - name: "london"
            addresses:
              - "exampleqm-ibm-mq-nhacrr-london.apps.trouble.cp.fyre.ibm.com:443"
            trust:
              - secret:
                  items:
                    - tls.crt
                    - ca.crt
                  secretName: nhacrr-london-group-trust
...
```

   nhacrr-london-group-trust secret 包含存储在伦敦站点 nhacrr-london-group secret 中的证书和 CA。 nhacrr-london-group 秘密用于确保来自伦敦群组的流量安全，因此罗马群组使用该秘密来确认所提供的证书是否可信。

   c) 完成伦敦网站上的相应步骤。 将 .spec.queuemanager.availability.tls.secretName 属性更改为

```
      tls:
        secretName: nhacrr-london-group
```

   d) 将 nhacrr-rome-group-trust 秘密添加到 spec.queueManager.availability.NativeHAGroups.remotes.trust 节中：

```
...
      nativeHAGroups:
        local:
```

```
          name: "london"
          role: "Live"
        remotes:
          - name: "rome"
            addresses:
              - "<rome group hostname>:443"
            trust:
              - secret:
                  items:
                    - tls.crt
                    - ca.crt
                  secretName: nhacrr-rome-group-trust
...
```

5. 使用新证书验证跨区域复制。 在伦敦站点，发出 `oc get qmgr exampleqm` 命令显示队列管理器资源，等待队列管理器运行。 然后使用 `oc get pods` 命令检查 pod 的状态。 其中一个 pod 应处于就绪状态，即活动 pod。

   接下来，您可以发布 `oc exec -t <active-pod> -- dspmq -o nativeha -g` 命令来检查实时组的状态。 将 `<active-pod>` 替换为 `oc get pods` 命令返回的 ready 1/1 pod 名称。

```
QMNAME(EXAMPLEQM)                                         ROLE(Active) INSTANCE(exampleqm-
ibm-mq-0) INSYNC(yes) QUORUM(3/3) GRPLSN(<0:0:87:58322>) GRPNAME(london) GRPROLE(Live)
 GRPNAME(london) GRPROLE(Live) GRPADDR(Unknown) GRPVER(9.4.3.0) GRSTATUS(Normal)
RCOVLSN(<0:0:87:58322>) RCOVTIME(2025-05-16T14:27:09.051799Z) INITLSN(<0:0:9:4696>)
INITTIME(2025-05-12T16:58:13.582274Z) LIVETIME(2025-05-12T16:58:16.319844Z)
ALTDATE(2025-05-16) ALTTIME(14.27.10)
 GRPNAME(rome) GRPROLE(Recovery) GRPADDR(exampleqm-ibm-mq-nhacrr-
<namespace>.apps.<cluster>) GRPVER(9.4.3.0) CONNGRP(yes) GRSTATUS(Normal)
RCOVLSN(<0:0:87:58322>) RCOVTIME(2025-05-16T14:27:09.051799Z) BACKLOG(0) INSYNC(yes)
SYNCTIME(2025-05-16T14:28:38.084565Z) ALTDATE(2025-05-16) ALTTIME(14.28.31)
```

   这一地位的关键部分是

   - `Role(Active)`
   - `QUORUM(3/3)` - 直播小组的法定人数已满
   - `GRSTATUS(Normal)`

6. 使用独立证书确保组之间的通信安全。 您可以为组内流量和跨组流量使用不同的证书。 在这一步中，`nhacrr-rome-crr` 和 `nhacrr-london-crr` 保密文件中的证书用于确保组间流量的安全，而 `nhacrr-rome-group` 和 `nhacrr-london-group` 保密文件则用于确保每个组内流量的安全。 为此，每个站点都需要进行两次更改，一次是指定组间流量的密文，另一次是更改信任以匹配使用的新证书。

   注: 如果 `.spec.queuemanager.availability.NativeHAGroups.local.tls.secretName` 和 `.spec.queuemanager.availability.tls.secretName` 属性使用的秘密包含相同的证书，则会发生冲突，队列管理器实例无法启动。 如果要使用单个证书，请仅设置 `.spec.queuemanager.availability.tls.secretName` 。

   a) 在 Rome 网站上，在 OCP 网络控制台的 QueueManager YAML 中添加 `spec.queuemanager.availability.NativeHAGroups.local.tls.key.secretName` 属性，并将其设置为 `nhacrr-rome-crr`。 然后将 `.spec.queueManager.availability.NativeHAGroups.remotes.trust.secret.secretName` 更改为 `nhacrr-london-crr-trust` ：

```
...
      nativeHAGroups:
        local:
          name: "rome"
          role: "Recovery"
          tls:
            key:
              secretName: nhacrr-rome-crr
        remotes:
          - name: "london"
            addresses:
              - "<london group hostname>:443"
            trust:
              - secret:
                  items:
                    - tls.crt
                    - ca.crt
```

```
                    secretName: nhacrr-london-crr-trust
...
```

b) 完成伦敦网站上的相应步骤。 在 OCP 网络控制台的 QueueManager YAML 中添加 `.spec.queuemanager.availability.NativeHAGroups.local.tls.key.secretName` 属性，并将其设置为 `nhacrr-london-crr`。 然后将 `.spec.queueManager.availability.NativeHAGroups.remotes.trust.secret.secretName` 更改为 `nhacrr-rome-crr-trust`。

```
...
      nativeHAGroups:
        local:
          name: "london"
          role: "Live"
          tls:
            key:
              secretName: nhacrr-london-crr
        remotes:
          - name: "london"
            addresses:
              - "<london group hostname>:443"
            trust:
              - secret:
                  items:
                    - tls.crt
                    - ca.crt
                  secretName: nhacrr-rome-crr-trust
...
```

7. 使用新证书验证跨区域复制。 在伦敦站点，发出 `oc get qmgr exampleqm` 命令显示队列管理器资源，并等待队列管理器运行。 然后使用 `oc get pods` 检查 pod 的状态，其中一个 pod 应处于就绪状态，即活动 pod。

   发出 `oc exec -t <active-pod> -- dspmq -o nativeha -g` 命令检查实时组的状态。 将 `<active-pod>` 替换为 `oc get pods` 命令返回的 ready 1/1 pod 名称。

```
QMNAME(EXAMPLEQM)                                         ROLE(Active) INSTANCE(exampleqm-
ibm-mq-0) INSYNC(yes) QUORUM(3/3) GRPLSN(<0:0:87:58322>) GRPNAME(london) GRPROLE(Live)
 GRPNAME(london) GRPROLE(Live) GRPADDR(Unknown) GRPVER(9.4.3.0) GRSTATUS(Normal)
RCOVLSN(<0:0:87:58322>) RCOVTIME(2025-05-16T14:27:09.051799Z) INITLSN(<0:0:9:4696>)
INITTIME(2025-05-12T16:58:13.582274Z) LIVETIME(2025-05-12T16:58:16.319844Z)
ALTDATE(2025-05-16) ALTTIME(14.27.10)
 GRPNAME(rome) GRPROLE(Recovery) GRPADDR(exampleqm-ibm-mq-nhacrr-
<namespace>.apps.<cluster>) GRPVER(9.4.3.0) CONNGRP(yes) GRSTATUS(Normal)
RCOVLSN(<0:0:87:58322>) RCOVTIME(2025-05-16T14:27:09.051799Z) BACKLOG(0) INSYNC(yes)
SYNCTIME(2025-05-16T14:28:38.084565Z) ALTDATE(2025-05-16) ALTTIME(14.28.31)
```

   这一地位的关键部分是

   - `Role(Active)`
   - `QUORUM(3/3)` - 直播小组的法定人数已满
   - `GRSTATUS(Normal)`

   最后，如果你发出 `kubectl logs <active-pod>` 命令，应该会在日志中看到以下信息，确认不同类型的流量使用了正确的证书。 对于跨组流量，您应该看到

```
AMQ3307I: The Native HA secure connection between the local group and 'rome' is using
certificate DN 'CN=nhacrr-london-crr'
```

   对于内部组流量，您应该看到以下信息，其中 `<replica-pod>` 是处于就绪 0/1 状态的 pod 的名称：

```
AMQ3306I: The Native HA secure connection between the local instance and '<replica-pod>'
accepted remote certificate DN 'CN=nhacrr-london-group'
```

8. 为复制流量添加特定的 CipherSpecs。

   如果没有为复制流量定义特定的 CipherSpecs，则使用 ANY_TLS13_OR_HIGHER，因此协议和加密是在实例之间协商的。 有关 IBM MQ 支持的 CipherSpecs 的更多信息，请阅读启用 CipherSpecs。

   要为组内和跨组流量指定 CipherSpec，可在 QueueManager YAML 中添加 `.spec.queuemanager.availability.tls.cipherSpec` 属性。

将下面的 YAML 添加到罗马和伦敦站点 OCP 网络控制台的 QueueManager YAML 中，以便对所有复制流量使用 TLS_AES_256_GCM_SHA384 CipherSpec：

```
      tls:
        secretName: nhacrr-rome-group
        cipherSpec: "TLS_AES_256_GCM_SHA384"
```

如果需要为跨组流量指定单独的 CipherSpec，可在 QueueManager YAML 中添加 `.spec.queuemanager.availability.NativeHAGroups.local.tls.cipherSpec` 属性。

将下面的 YAML 添加到 OCP 网络控制台中罗马和伦敦站点的 QueueManager YAML 中，使 TLS_AES_128_GCM_SHA256 CipherSpec 用于组间流量， TLS_AES_256_GCM_SHA384 CipherSpec 用于组内流量：

```
      nativeHAGroups:
        local:
       ...
          tls:
            key:
              secretName: nhacrr-rome-crr
            cipherSpec: "TLS_AES_128_GCM_SHA256"
```

实例重启后，在伦敦站点上发出以下命令：

```
kubectl logs
        <active-pod>
```

您应该会在日志中看到与下面示例类似的信息。 示例显示，组实例之间的通信使用 CipherSpec TLS_AES_256_GCM_SHA384：

```
[CommentInsert1(exampleqm-ibm-mq-2), CommentInsert2(exampleqm-ibm-mq-replica-2(9414)),
        CommentInsert3(TLS_AES_256_GCM_SHA384)]
```

该示例显示，与 Rome 组的连接使用 CipherSpec TLS_AES_128_GCM_SHA256：

```
AMQ3253I: A Native HA recovery group connection to 'rome' has
        been established using the 'TLS_AES_128_GCM_SHA256' CipherSpec.
```

9. 允许在组内进行纯文本通信。 组之间的流量必须始终保持安全和加密，但组内部的流量可以不加密。在伦敦站点将 `.spec.queuemanager.availability.tls.cipherSpec` 设置为空，以取消对伦敦组内部流量的加密：

   ```
         tls:
           cipherSpec: ''
           secretName: nhacrr-london-group
   ```

   实例重启后，在伦敦站点上发出以下命令：

   ```
   kubectl logs
           <active-pod>
   ```

   日志中出现以下信息

   `AMQ9722W: Plain text communication is enabled.`

10. 禁用组内流量的安全性。 删除 `.spec.queuemanager.availability.tls` stanza 可完全禁用组内部流量的安全性。

    注: 组之间的流量必须安全，因此如果 `.spec.queuemanager.availability.tls.secretName` 被删除，则必须指定 `.spec.queuemanager.availability.NativeHAGroups.local.tls.key.secretName`。

在罗马站点的 OCP 网络控制台 QueueManager YAML 中删除 `.spec.queuemanager.availability.tls` stanza，禁用罗马组的安全性：

```
    tls:
      secretName: nhacrr-rome-group
      cipherSpec: "TLS_AES_256_GCM_SHA384"
```

重新启动实例后，在 Rome 网站上发布 `kubectl logs <active-pod>` 命令，即可在日志中看到类似的信息：

```
AMQ3213I: Native HA inbound connection accepted from 'exampleqm-ibm-mq-2'.
[CommentInsert1(exampleqm-ibm-mq-2), CommentInsert2(<ip>)]
AMQ3211I: Native HA outbound connection established to 'exampleqm-ibm-mq-2'.
[CommentInsert1(exampleqm-ibm-mq-2), CommentInsert2(<replica>(9414))]
```

替代：

```
AMQ3214I: Native HA inbound secure connection accepted from 'exampleqm-ibm-mq-0'.
[CommentInsert1(exampleqm-ibm-mq-0), CommentInsert2(<ip>), CommentInsert3(<cipherspec>)]
AMQ3212I: Native HA outbound secure connection established to 'exampleqm-ibm-mq-0'.
[CommentInsert1(exampleqm-ibm-mq-0), CommentInsert2(<replica>(9414)),
CommentInsert3(<cipherspec)
```

OpenShift MQ Adv. 查看 *IBM MQ* 容器的本机 *HA* 队列管理器的状态

对于 IBM MQ 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

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确定的，副本实例提供的信息可能已经过时。

V 9.4.2 对于原生 HA 跨区域复制 (CRR) 配置，提供的信息还取决于实例是实时组还是恢复组的一部分。 实时小组的信息更为准确，但由于恢复小组仅定期发送信息，因此仍可能过时。

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。
- V 9.4.2 查看实例所属或连接的群组状态。

以下状态字段用于报告队列管理器实例的 Native HA 和 Native HA 跨区域复制 (CRR)配置状态。 这些字段始终会出现：

职能部门
指定实例的当前角色。 是 `Active`、 `Replica`、 `Unknown`、 （用于康复小组） `Leader` 或 `Not Configured` 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。

V 9.4.2 **GRPLSN**
指定在 HA 组中具有法定人数的实例之间复制的日志序列号（LSN），采用 *nnnnn:nnnnn:nnnnn:nnnnn* 格式。 如果不知道 LSN，则为空。

V 9.4.2 **GRPNAME**
指定 HA 组的名称。 如果未配置名称，则为空。

V 9.4.2 **GRPROLE**
指定 HA 组的当前角色，可以是其中之一：

- Live - 现场乐队
- Recovery - 一个康复小组
- Pending live - 一群等待成为 Live
- Pending recovery - 一群等待成为 Recovery
- Not configured - 本地 HA 尚未配置
- Unknown - 当前角色无法确定

使用 -x 参数请求所有实例的扩展信息时，这些字段就会出现：

REPLADDR
队列管理器实例的复制地址。

连接 ACTV
指示实例是否连接到活动实例。

BACKLOG
表示实例落后于活动实例的 KB 数。

连接
指示指定的实例是否已连接到此实例。

ALTDATE
显示此信息最后更新的日期。 如果从未更新，则留空。

ALTTIME
表示该信息最后一次更新的时间。 如果从未更新，则留空。

V 9.4.2 **ACKLSN**
表示实例已确认写入恢复日志的 LSN，格式为 *nnnnn:nnnnn:nnnnn:nnnnn* 。 如果不知道 LSN 值，则为空。

V 9.4.2 **HAStatus**

表示该实例的运行状态，为以下值之一：

- Normal - 实例正在正常运行。
- Checking - 正在检查该实例，以确保恢复日志的一致性。
- Synchronizing - 实例正在接收数据，使其保持同步。
- Rebasing - 新的恢复日志正在发送到实例。
- Disk full - 实例磁盘已满。
- Disconnected - 无法与实例通信。
- Unknown - 实例状态未知。

V 9.4.2 **同步时间**
以 ISO 8601 格式表示此实例与活动队列管理器保持同步的最后时间。 如果时间未知，则留空。

V 9.4.2 使用 -g 参数请求组信息时，这些字段就会出现。 当组属于原生高可用性 CRR 配置时，这些字段可用于查看状态：

V 9.4.2 **GRPNAME**
指定 HA 组的名称。 如果未配置名称，则为空。

V 9.4.2 **GRPROLE**
指定 HA 组的当前角色，可以是其中之一：

- Live - 现场乐队
- Recovery - 一个康复小组

- Pending live - 一群等待成为 Live
- Pending recovery - 一群等待成为 Recovery
- Unknown - 当前角色无法确定
- Not configured - 本地 HA 尚未配置

V 9.4.2 **GRPADDR**
用于连接到群组的 IP 地址。 如果因为尚未与群组建立连接而不知道地址，请发送邮件至 Unknown 。

V 9.4.2 **GRPVER**
以 V.R.M.F 格式表示当前组长的版本，其中 V.R.M.F 是版本号、发行号、修改号和补丁包号。 如果不知道数值，则将其设置为 ?.?.?.?

V 9.4.2 **CONNGRP**
表示该组是否与本地组相连，并且是其中之一：

- yes - 小组已连接。
- no - 该组已断开连接。
- unknown - 连接状态未知。
- suspended - 组是连接的，但在组之间的配置不兼容问题得到解决之前，组无法复制数据。

此字段仅对本地组以外的组显示。

V 9.4.2 **INSYNC**
此功能仅对具有角色 Recovery 或 Pending live 的群组显示。 它表明在发生故障切换时，该组是否能够 Live 且不丢失数据。 （请注意， Recovery 组可能不会经常与 Live 组同步，因为日志是异步复制的。 使用 RCOVLSN 和 RCOVTIME 值来指示在计划外故障切换时是否可能出现数据丢失。

V 9.4.2 **同步时间**
此功能仅对具有角色 Recovery 或 Pending live 的群组显示。 它以 ISO 8601 格式（如果不知道，则留空）表示该组与 Live 组上次同步的时间。 （请注意， Recovery 组可能不会经常与 Live 组同步，因为日志是异步复制的。 使用 RCOVLSN 和 RCOVTIME 值来指示在计划外故障切换时是否可能出现数据丢失。

V 9.4.2 **BACKLOG**
此功能仅对具有角色 Recovery 或 Pending live 的群组显示。 它表示该组落后于实时组多少 KB。

V 9.4.2 **GRSTATUS**
表示该组的运行状态，为以下值之一：

- Normal - 该小组正在正常运作。
- Checking - 正在检查该组，以确保恢复日志的一致性。
- Synchronizing - 正在向该组发送数据，使其保持同步。
- Rebasing - 新的恢复日志正在发送到实例。
- Waiting for connection - 该群正在等待一个来自 Live 或 Pending Recovery 角色的群。
- Partitioned - 有两个小组认为自己属于“现场”小组（即“大脑分裂”状态）。
- Unknown - 该群状态未知。

V 9.4.2 **RCOVLSN**
组可恢复的日志序列号 (LSN)，格式为 *nnnnn:nnnnn:nnnnn:nnnnn*。 如果发生计划外的故障切换，此后的任何数据都可能丢失。 该值可输入 **dmpmqlog** 命令的 -s 选项。

V 9.4.2 **RCOVTIME**
小组可以恢复的时间，以 ISO 8601 格式表示。 如果发生意外故障切换，在此时间之后写入恢复日志的任何数据都可能丢失。 该值可以传递给 **dmpmqlog** 命令的 -t 或 -u 选项。

V 9.4.2 **INITLSN**
此功能仅对具有角色 Live 或 Pending recovery 的群组显示。 它表示本地高可用性组最初启用时恢复的最后一个日志记录的日志序列号（LSN），格式为 *nnnnn:nnnnn:nnnnn:nnnnn*，如果该组未启用，则为空。

V 9.4.2 **initTime**
此功能仅对具有角色 Live 或 Pending recovery 的群组显示。 它以 ISO 8601 格式表示本地高可用性组最初启用时恢复的最后一个日志记录的时间。 空白，如果该组不是实时组。

V 9.4.2 **生活时间**
此功能仅对具有角色 Live 或 Pending recovery 的群组显示。 它以 ISO 8601 格式表示该组合首次成为现场组合的时间。 空白，如果该组不是实时组。

V 9.4.2 **ALTDATE**
显示此信息最后更新的日期。

V 9.4.2 **ALTTIME**
表示该信息最后一次更新的时间。

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)
```

V 9.4.2 在本地 HA CRR 配置中作为恢复组组长的 BOB 实例将报告以下状态:

```
QMNAME(BOB)                   STATUS(Recovery group leader)
```

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)
GRPLSN(<0:25:365:1000>) GRPNAME() GRPROLE(Live)
```
V 9.4.2

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) V9.4.2
GRPLSN(<0:25:365:1000>) GRPNAME() GRPROLE(Live)
```

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) V9.4.2
GRPLSN() GRPNAME() GRPROLE(Live)
```

V9.4.2 在本地 HA CRR 配置中，属于实时组的名为 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)
GRPLSN(<0:25:365:1000>) GRPNAME(alpha) GRPROLE(Live)
```

V9.4.2 在组等待成为恢复组的同一实例中，可能会报告以下状态:

```
QMNAME(BOB)         ROLE(Active) INSTANCE(inst1) INSYNC(yes) QUORUM(3/3)
GRPLSN(<0:25:365:1000>) GRPNAME(alpha) GRPROLE(Pending recovery)
```

V9.4.2 名为 CAROL 的队列管理器不是本地 HA 或本地 HA CRR 配置的实例，将报告以下状态:

```
QMNAME(CAROL)       ROLE(Not configured) INSTANCE() INSYNC() QUORUM() GRPLSN() GRPNAME()
GRPROLE(Not configured)
```

- 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
```

例如，如果您在运行队列管理器 BOB 活跃实例的节点上发出此命令，而该节点上的所有实例都是同步的，则可能会收到以下状态:

```
QMNAME(BOB)         ROLE(Active) INSTANCE(inst1) INSYNC(yes) QUORUM(2/3)
GRPLSN(<0:25:365:1000>) GRPROLE(Live) GRPNAME()
 INSTANCE(inst1) ROLE(Active)  REPLADDR(localhost) CONNACTV(yes) INSYNC(yes) BACKLOG(0)
CONNINST(yes) ACKLSN(<0:26:365:1000>)  HASTATUS(Normal)
SYNCTIME(2023-12-20T17:13:27.853412Z) ALTDATE(2023-12-20) ALTTIME(17.13.27)
 INSTANCE(inst3) ROLE(Replica) REPLADDR(localhost) CONNACTV(yes) INSYNC(yes) BACKLOG(0)
CONNINST(yes) ACKLSN (<0:24:365:1000>) HASTATUS(Normal)
SYNCTIME(2023-12-20T17:13:27.853412Z) ALTDATE(2023-12-20) ALTTIME(17.13.27)
 INSTANCE(inst2) ROLE(Replica) REPLADDR(localhost) CONNACTV(yes) INSYNC(yes) BACKLOG(0)
CONNINST(yes) ACKLSN (<0:25:365:1000>) HASTATUS(Synchronizing)
SYNCTIME(2023-12-20T17:13:22.123321Z) ALTDATE(2023-12-20) ALTTIME(17.13.27)
```

再举一个例子，如果您在运行队列管理器 BOB 副本实例的节点上发出此命令，而该副本实例滞后，则可能会收到以下状态:

```
QMNAME(BOB)         ROLE(Replica) INSTANCE(inst2) INSYNC(no) QUORUM(2/3)
GRPLSN(<0:25:365:1000>) GRPROLE(Live) GRPNAME()
 INSTANCE(inst1) ROLE(Active)  REPLADDR(localhost) CONNACTV(yes) INSYNC(yes) BACKLOG(0)
CONNINST(yes) ACKLSN(<0:26:365:1000>)  HASTATUS(Normal)
SYNCTIME(2023-12-20T17:13:27.853412Z) ALTDATE(2023-12-20) ALTTIME(17.13.27)
 INSTANCE(inst3) ROLE(Replica) REPLADDR(localhost) CONNACTV(yes) INSYNC(yes) BACKLOG(0)
CONNINST(yes) ACKLSN (<0:24:365:1000>) HASTATUS(Normal)
SYNCTIME(2023-12-20T17:13:27.853412Z) ALTDATE(2023-12-20) ALTTIME(17.13.27)
 INSTANCE(inst2) ROLE(Replica) REPLADDR(localhost) CONNACTV(yes) INSYNC(no) BACKLOG(435)
CONNINST(yes) ACKLSN (<0:25:365:1000>) HASTATUS(Synchronizing)
SYNCTIME(2023-12-20T17:13:22.123321Z) ALTDATE(2023-12-20) ALTTIME(17.13.27)
```

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) GRPLSN()
GRPROLE() GRPNAME()
 INSTANCE(inst1) ROLE(Unknown) REPLADDR(9.20.123.45) CONNACTV(Unknown) INSYNC(Unknown)
BACKLOG(Unknown) CONNINST(No) ACKLSN()  HASTATUS(Unknown) SYNCTIME() ALTDATE() ALTTIME()
ALTDATE() ALTTIME()
 INSTANCE(inst2) ROLE(Unknown) REPLADDR(9.20.123.46) CONNACTV(Unknown) INSYNC(Unknown)
```

```
BACKLOG(Unknown) CONNINST(No) ACKLSN()  HASTATUS(Unknown) SYNCTIME() ALTDATE() ALTTIME()
ALTDATE() ALTTIME()
 INSTANCE(inst3) ROLE(Unknown) REPLADDR(9.20.123.47) CONNACTV(No)      INSYNC(Unknown)
BACKLOG(Unknown) CONNINST(No) ACKLSN()  HASTATUS(Unknown) SYNCTIME() ALTDATE() ALTTIME()
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)
```

- V 9.4.2

  确定本地 HA CRR 配置中所有本地 HA 组的运行状态:

  ```
  oc exec -t Pod dspmq -o nativeha -g -m QMgrName
  ```

  如果您对实时群组中的成员 alpha 发出此命令，则 alpha 和恢复群组 beta 可能会收到以下状态:

  ```
  GRPNAME(alpha) GRPROLE(Live) GRPADDR(9.20.135.129) GRPVER(09040200)
  GRSTATUS(Normal) RCOVLSN(<0:0:104:65163>) RCOVTIME(2023-12-20T17:37:27.114787Z)
  INITLSN(<0:0:101:352>) INITTIME(2023-12-19T09:13:27.512294)
  LIVETIME(2023-12-19T06:14:27.512294) ALTDATE(2023-12-20) ALTTIME(17.13.27)
  ```

  ```
  GRPNAME(beta) GRPROLE(Recovery) GRPADDR(9.20.135.130) GRPVER(09040300)
  CONNGRP(yes) GRSTATUS(Synchronizing) RCOVLSN(<0:0:19:16723>)
  RCOVTIME(2023-12-20T17:36:23.463209Z) BACKLOG(5619) INSYNC(no) SYNCTIME()
  ALTDATE(2023-12-20) ALTTIME(17.13.27)
  ```

相关任务

第 98 页的『示例: 使用 IBM MQ Operator 配置本机 HA』

此示例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 OpenShift Container Platform 中。 相互 TLS 用于认证，以从 TLS 证书映射到队列管理器中的身份。

相关参考

dspmq (显示队列管理器) 命令

OpenShift MQ Adv. 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 87 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 。

OpenShift V 9.4.2 MQ Adv. 针对 *Native HA CRR* 的高级调校

用于调整时序和间隔的高级设置。 除非已知默认设置不符合您的系统要求，否则无需使用这些设置。

配置 Native HA CRR 的基本选项使用 QueueManager API 处理， IBM MQ Operator 使用该 API 为您配置基础队列管理器 INI 文件。 还有一些高级选项只能通过 INI 文件进行配置， 在 qm.ini 文件的 NativeHARecoveryGroup 节下。 如需了解如何配置 INI 文件的更多信息，请查看 第 87 页的『示例: 提供 MQSC 和 INI 文件』 。

ShortRetryTimer
指定在短重试阶段尝试连接恢复组之间的等待时间间隔。 以毫秒为单位，默认为 2500（ 2.5 秒）。

ShortRetryCount
指定在短重试阶段尝试连接恢复组的次数。

LongRetryTimer
指定在长时间重试阶段尝试连接恢复组之间的等待时间间隔。 在连接成功或配置更改为注释或删除 ReplicationAddress 之前，漫长的重试阶段将无限期持续。 以毫秒为单位，默认为 60000（60 秒）。

RetryReporting
请发送电子邮件至 Phase 或 All ，说明是否每次重试都进行报告，还是只报告阶段（长或短）。 如果没有指定，则默认为 Phase。

StatusInterval
指定恢复组在发送恢复日志状态之前等待的最长时间间隔。 以毫秒为单位，默认为 10000（10 秒）。

StatusCommitThreshold
指定恢复组在发送恢复日志状态之前提交的最大字节数。 以字节为单位指定，默认为 0，这意味着仅使用 StatusInterval 来确定何时发送恢复日志状态。

OpenShift MQ Adv. 结束本机 *HA* 队列管理器

对于容器中的 IBM MQ ， 可以使用 **endmqm** 命令结束作为本地 HA 组一部分的活动队列管理器或副本队列管理器。

开始之前

注: 此信息仅适用于容器环境。

关于此任务

停止作为本地 HA 组一部分的队列管理器的程序取决于它是活动实例还是副本实例。 结束任何一种类型的实例时，都会进行检查，以确保结束实例不会破坏本地 HA 组的法定人数。 如果法定人数被破坏， **endmqm** 命令将失败。

当您发出 **endmqm** 命令时，组中的其他实例会收到警告，以便在连接中断时不会报错。

如果活动实例因太多副本实例结束或断开连接而失去法定人数，则活动实例在完全结束前会等待一段可配置的时间。 这样就可以有一段时间从容关闭处理程序，而不是让应用程序的连接直接中断。 超时值可由 qm.ini 文件 NativeHALocalInstance 节中的 QuorumConnectivityTimeout 属性指定。 默认值为 0 秒。

过程

- 要结束队列管理器的活动实例，请在运行活动实例的节点上发出以下命令:

```
endmqm -s QMgrName
```

  - 指定 -r 选项可帮助客户端应用程序重新连接到另一个实例。
  - 如果该实例不是本地 HA 组中的活动实例，则命令失败。
  - 如果结束此活动实例会导致组法定人数失效，则命令失败。 (如果在运行此命令的同时其他实例结束或不可用，则法定人数检查可能无法检测到这一点，本地 HA 组就会结束，只有当有足够的实例可用时才能重新启动)

  当活动队列管理器结束时，其中一个副本实例会接替活动队列管理器的角色。 您不能指定由哪个副本接管，而是由组内协商决定，并取决于哪个副本拥有最新的事务日志。

- 要结束队列管理器的副本实例，请执行以下命令：

```
endmqm -x QMgrName
```

  - 如果该实例是活动实例，则命令失败。
  - 如果结束该副本实例会导致组法定人数失效，则命令失败。（如果在运行此命令的同时其他实例结束或不可用，则法定人数检查可能无法检测到这一点，本地 HA 组就会结束，只有当有足够的实例可用时才能重新启动）

  注： 您还可以在 Native HA 实例上使用 -c、-i、-p 或 -w 开关和 **endmqm** 命令，无论它们处于哪个角色。队列管理器实例结束，忽略对组法定人数的影响。 不过，信息仍会与组内其他实例共享。 您可以将这些开关与活动实例的 -s 一起使用。 这些开关不能与副本实例的 -x 开关一起使用。

OpenShift CP4I MQ Adv. Kubernetes 使用 *IBM MQ Operator* 配置多实例队列管理器

此示例使用 IBM MQ Operator 将多实例队列管理器部署到 OpenShift Container Platform 中。 相互 TLS 用于认证，以从 TLS 证书映射到队列管理器中的身份。

开始之前

要完成此示例，必须首先完成以下先决条件：

- 为此示例创建 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 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。

过程

1. 确定合适的存储类

   可以使用多种 持久卷访问方式来访问 Kubernetes 集群中的存储器。 多实例队列管理器创建多个持久卷：每个队列管理器一个持久卷，至少一个共享卷。 多实例队列管理器的共享卷必须使用 ReadWriteMany 存储类。 Kubernetes 集群中的缺省存储类通常用于 ReadWriteOnce 存储类 (块存储器)。 例如，如果您正在使用 Red Hat OpenShift Data Foundation，那么存储类 *ocs-storagecluster-cephfs* 将提供合适的共享文件系统。 文件系统的选择非常重要，因为并非所有共享文件系统都以相同的方式处理文件锁定。请参阅 规划 Multiplatforms 版上的文件系统支持 和 针对 IBM MQ 多实例队列管理器文件系统的测试语句。
2. 创建一对证书，如 第 90 页的『使用 OpenSSL 创建自签名 PKI』中所述。
3. 创建包含 MQSC 命令和 INI 文件的配置映射

   创建包含 MQSC 命令的 Kubernetes ConfigMap，以创建新队列和 SVRCONN 通道，并添加允许访问该通道的通道认证记录。

   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 Web 控制台中输入以下 YAML，或者使用命令行。

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: example-miqm-configmap
data:
  example-tls.mqsc: |
    DEFINE CHANNEL('MTLS.SVRCONN') CHLTYPE(SVRCONN) SSLCAUTH(REQUIRED)
SSLCIPH('ANY_TLS13_OR_HIGHER') REPLACE
    SET CHLAUTH('MTLS.SVRCONN') TYPE(SSLPEERMAP) SSLPEER('CN=*') USERSRC(NOACCESS)
ACTION(REPLACE)
    SET CHLAUTH('MTLS.SVRCONN') TYPE(SSLPEERMAP) SSLPEER('CN=example-app1') USERSRC(MAP)
MCAUSER('app1') ACTION(REPLACE)
```

```
    SET AUTHREC PRINCIPAL('app1') OBJTYPE(QMGR) AUTHADD(CONNECT,INQ)
    DEFINE QLOCAL('EXAMPLE.QUEUE') REPLACE
    SET AUTHREC PROFILE('EXAMPLE.QUEUE') PRINCIPAL('app1') OBJTYPE(QUEUE)
AUTHADD(BROWSE,PUT,GET,INQ)
  example-tls.ini: |
    Service:
        Name=AuthorizationService
        EntryPoints=14
        SecurityPolicy=UserExternal
```

MQSC 定义名为 *MTLS.SVRCONN* 的通道和名为 *EXAMPLE.QUEUE*。 此通道配置为仅允许访问提供 "公共名称" 为 *example-app1* 的证书的客户机。 这是在步骤 第 127 页的『2』中创建的其中一个证书中使用的公共名称。 此通道上具有此公共名称的连接将映射到用户标识 *app1*，该用户标识有权连接到队列管理器并访问示例队列。 INI 文件启用安全策略，这意味着 *app1* 用户标识不需要存在于外部用户注册表中-它仅作为此配置中的名称存在。

4. 部署队列管理器

   使用以下定制资源 YAML 创建新的队列管理器。 在开始这项任务前，请确保您已进入所创建的命名空间，然后在 OCP 网络控制台或使用命令行输入以下 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。

   注: 本示例使用 IBM MQ Advanced 许可证，但也可以使用 IBM MQ 许可证。 有关更多信息，请参阅第 140 页的『使用 IBM MQ 许可证注释配置队列管理器。 IBM MQ Operator』。

```
apiVersion: mq.ibm.com/v1beta1
kind: QueueManager
metadata:
  name: exampleqm
spec:
  license:
    accept: false
    license: L-NUUP-23NH8Y
    use: Production
  queueManager:
    name: EXAMPLEQM
    availability:
      type: MultiInstance
    mqsc:
    - configMap:
        name: example-miqm-configmap
        items:
        - example-tls.mqsc
    ini:
    - configMap:
        name: example-miqm-configmap
        items:
        - example-tls.ini
    storage:
      defaultClass: STORAGE_CLASS
  version: 9.4.3.0-r1
  pki:
    keys:
      - name: default
        secret:
          secretName: example-qm-tls
          items:
            - tls.key
            - tls.crt
            - ca.crt
```

   将 *STORAGE_CLASS* 更改为您在步骤 第 127 页的『1』中识别的存储类。

   请注意， Secret *example-qm-tls* 是在步骤 第 127 页的『2』中创建的，而 ConfigMap *example-miqm-configmap* 是在步骤 第 127 页的『3』 中创建的

   可用性类型设置为多实例 ，这样就会自动选择持久存储。

5. 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 exampleqm
```

6. 测试与队列管理器的连接

要确认队列管理器已配置且可用，请遵循 第 94 页的『从笔记本电脑测试到队列管理器的相互 TLS 连接』中的步骤。

7. 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:

   a) 查看活动和备用 pod

   运行以下命令:

   ```
   oc get pods --selector app.kubernetes.io/instance=exampleqm
   ```

   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`。

   b) 删除活动 pod

   运行以下命令，并指定活动 pod 的全名:

   ```
   oc delete pod exampleqm-ibm-mq-value
   ```

   c) 再次查看 pod 状态

   运行以下命令:

   ```
   oc get pods --selector app.kubernetes.io/instance=exampleqm
   ```

   d) 查看队列管理器状态

   运行以下命令并指定其他 pod 的全名:

   ```
   oc exec -t Pod -- dspmq -x
   ```

   您应该会看到状态显示活动实例已更改，例如:

   ```
   QMNAME(EXAMPLEQM)                                         STATUS(Running as standby)
       INSTANCE(exampleqm-ibm-mq-1) MODE(Active)
       INSTANCE(exampleqm-ibm-mq-0) MODE(Standby)
   ```

   e) 再次测试与队列管理器的连接

   要确认队列管理器已恢复，请执行 第 94 页的『从笔记本电脑测试到队列管理器的相互 TLS 连接』中的步骤。

结果

恭喜您成功部署了具有相互 TLS 认证的多实例队列管理器，并验证它在活动 Pod 失败时自动恢复。

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

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。

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。

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

如果 **OutboundSNI** 属性设置为 HOSTNAME，那么 IBM MQ 受管 .NET 客户机会将 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。

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: unique_name_for_the_route
  namespace: namespace_of_your_MQ_deployment
spec:
  host: SNI_address_mapping_for_the_channel
```

```
  to:
    kind: Service
    name: name_of_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 191 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 。

在容器中为 IBM MQ 配置审计日志

IBM MQ 在容器中提供了对不同事件进行审计记录的各种设施。

根据环境的审计日志要求，您可能需要配置一个或多个这些设施，以及收集和保留事件日志的适当机制。 从下列选项中进行选择。

IBM MQ Operator 验证 Webhook

从 3.6.0 版本开始， IBM MQ Operator 默认会将暴露端点的访问审计事件输出到 pod 的容器日志中。 无需配置。

IBM MQ 活动

您可以配置 IBM MQ ，以生成用于审计的事件消息。 更多信息，请参阅 " 审计 "。

V 9.4.3 MQ 容器度量

IBM MQ 容器公开了一个 Prometheus 度量端点（更多信息，请参阅 第 191 页的『使用 IBM MQ Operator 时进行监视』）。 从 IBM MQ 9.4.3 ，通过将 `MQ_LOGGING_METRICS_AUDIT_ENABLED` 环境变量设置为 `true` ，可以启用访问该端点的日志记录。 如果使用 IBM MQ Operator ，可在 `queueManager` 配置的 `env` 部分设置该变量:

```
[...]
spec:
  queueManager:
    env:
    - name: MQ_LOGGING_METRICS_AUDIT_ENABLED
      value: "true"
[...]
```

启用日志功能后，日志文件将在 `/var/mqm/errors` 目录中生成。 您可以通过将外部存储挂载到 `/mnt/mqm` 来持久化该目录中的数据（请参阅 " 容器映像参考 "）。 如果使用 IBM MQ Operator ，则可以使用 第 235 页的『.spec.queueManager.storage.persistedData』 字段来映射外部存储。 指标访问日志文件以 JSON 格式写入。 文件名的形式是 `metricaudit<nn>.json` ，其中 nn 是一个数字。 `metricaudit01.json` 文件包含最新事件。

嵌入式 IBM WebSphere Application Server Liberty 网络服务器

您可以对嵌入式 WebSphere Liberty 网络服务器进行配置，以生成审计事件。 `audit-1.0` 和 `audit-2.0` 功能已安装，可根据要求进行配置。 更多信息，请参阅 WebSphere Liberty 文档中的审计 1.0 和审计 2.0。

您可以使用配置文件启用和配置该功能。 如果您使用的是 IBM MQ Operator ，您可以通过 第 243 页的『.spec.web.manualConfig』 安装程序加载文件。 例如，以下代码可启用审计 2.0 功能的默认配置。

创建名为 mqwebuser.xml 的配置文件，以启用 audit-2.0 功能：

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: example-enable-audit
data:
  mqwebuser.xml: |
    <?xml version="1.0" encoding="UTF-8"?>
    <server>
      <featureManager>
        <feature>audit-2.0</feature>
      </featureManager>
    </server>
```

使用 spec.web.manualConfig mount 将前面的代码添加到 queueManager 配置中：

```
apiVersion: mq.ibm.com/v1beta1
kind: QueueManager
[...]
spec:
  [...]
  web:
    enabled: true
    manualConfig:
      configMap:
        name: example-enable-audit
```

注: 默认情况下，审计日志以基本（纯文本）格式镜像到容器日志。 该格式包含一套精简的审计信息。 要获取更多信息，可将 IBM MQ 容器日志更改为 JSON 格式，或禁用 web 日志镜像，直接收集 WebSphere Liberty JSON 审计日志文件。 您可以使用 MQ_LOGGING_CONSOLE_FORMAT 和 MQ_LOGGING_CONSOLE_SOURCE 环境变量来控制日志格式和日志镜像源。 有关有效值，请参见 第 255 页的『支持的环境变量』。

OpenShift CP4I 将 IBM MQ 与 IBM Instana 跟踪集成

IBM Instana 可用于跟踪 IBM Cloud Pak for Integration 中的事务。

开始之前

本文档涵盖 IBM Instana 跟踪，这是通过系统跟踪消息的过程。 它不涵盖 IBM Instana 监视，在此监视中检索有关 IBM MQ 队列管理器状态的详细信息。 有关由 IBM Instana 监视 IBM MQ 的信息，请参阅 监视 IBM MQ。 有关已认证的监视的详细指示信息，请参阅 第 134 页的『使用 TLS 配置已认证的 IBM Instana 监视』。

注:

- 该功能仅支持 IBM MQ 9.3.1.0-r2 或更高版本的操作数。
- 您可以在以前的 IBM MQ Operator 和队列管理器版本上运行 IBM Instana 跟踪，但不能在本机运行。 请参阅 IBM Instana 文档中的配置 IBM MQ 跟踪。
- 使用 IPv6 网络时不支持此功能。

必须先部署 IBM Instana 后端和 IBM Instana 代理程序，然后才能使用 IBM MQ 操作程序执行 IBM Instana 跟踪。 缺省情况下， IBM MQ 队列管理器与部署在队列管理器 pod 所在节点上的 IBM Instana 代理程序进行通信。

关于此任务

启用与 IBM Instana 的集成会导致在队列管理器中安装 IBM MQ API 出口。 API 出口向 IBM Instana 代理发送有关流经队列管理器的消息的跟踪数据。

API 出口向每条消息添加 RFH2 头。 这些头包含跟踪信息。

IBM Instana 代理程序负责将跟踪数据发送到 IBM Instana 后端。

有关部署 IBM Instana 后端和 IBM Instana 代理程序的信息，请参阅 IBM Instana 文档中的 在 Platform UI 中启用 Instana 监视链接 。

过程

标准部署

- 在启用 IBM Instana 跟踪的情况下部署队列管理器。

  缺省情况下， IBM Instana 跟踪处于禁用状态。

  如果您正在使用 IBM Cloud Pak for Integration Platform UI 或 OpenShift Web 控制台:

  1. 单击 **Operators** > **Installed Operators** > **IBM MQ** > **Details** > **Create instance**。 为您的队列管理器提供所有必要的详细信息。
  2. 在同一位置下，单击 **Telemetry** > **Tracing** > **Instana** 。
  3. 将 **启用 Instana 跟踪** 切换设置为 `true`。

  如果要通过 YAML 进行部署，请使用以下片段:

```
spec:
  telemetry:
    tracing:
      instana:
        enabled: true
```

高级部署

- 通过 https 与 IBM Instana 代理程序通信。

  缺省情况下， IBM MQ 的 IBM Instana 出口通过 HTTP 与 IBM Instana 代理程序通信。 代理程序的主机地址设置为运行队列管理器的节点的 IP 地址。 这与 IBM Instana 文档中的 启用 IBM Instana 监视 中描述的配置相匹配，其中 IBM Instana 代理程序由 IBM Instana Agent Operator 作为 daemonset 进行部署。

  目前， IBM MQ 的 IBM Instana 出口与 IBM Instana 代理程序之间的通信支持 http 或 https 协议。 要使用 https ，必须首先将 IBM Instana 代理程序配置为使用 TLS 加密。 请参阅 IBM Instana 文档中的 为代理程序端点设置 TLS 加密 。 然后，可以将协议设置为 https ，如下所示:

  如果您正在使用 OpenShift Web 控制台:

  1. 单击 **Operators** > **Installed Operators** > **IBM MQ** > **Details** > **Create instance**。 为您的队列管理器提供所有必要的详细信息。
  2. 在同一位置下，单击 **Telemetry** > **Instana** 。
  3. 展开 **高级配置** 下拉列表。
  4. 将 **Instana 代理程序通信协议** 设置为 `https`。

  如果要通过 YAML 进行部署，请使用以下片段:

```
spec:
  telemetry:
    instana:
      enabled: true
      protocol: https
```

- 设置 **agentHost**

  如果 IBM Instana 代理程序尚未部署为运行队列管理器的 Openshift 集群上的 daemonset ，那么必须将 **agentHost** 值设置为运行 IBM Instana 代理程序的主机名或 IP 地址。 **agentHost** 值不应包含协议或端口。

  如果您正在使用 OpenShift Web 控制台:

  1. 单击 **Operators** > **Installed Operators** > **IBM MQ** > **Details** > **Create instance**。 为您的队列管理器提供所有必要的详细信息。
  2. 在同一位置下，单击 **Telemetry** > **Instana** 。
  3. 展开 **高级配置** 下拉列表。
  4. 在 **Instana 代理程序主机** 文本框中输入主机名。

  如果要通过 YAML 进行部署，请使用以下片段:

```
spec:
  telemetry:
    instana:
      enabled: true
      agentHost: 9.9.9.9
```

下一步做什么

另请参阅第 85 页的『使用 IBM MQ Operator 部署简单队列管理器』。

OpenShift CP4I Operator2.2.0 使用 TLS 配置已认证的 IBM Instana 监视

要能够通过 IBM Instana 代理程序监视队列管理器，必须同时配置代理程序和队列管理器。

开始之前

IBM Instana 文档中的 "监视 IBM MQ" 的 "配置" 部分 提供有关 IBM Instana 监视配置的常规信息。 但是，它不包含有关配置队列管理器的详细信息。

必须先部署 IBM Instana 后端和 IBM Instana 代理程序，然后才能使用 IBM MQ 操作程序执行 IBM Instana 跟踪。 要执行此操作，请参阅 IBM Instana 文档中的 在 CP4I Platform UI 中启用 IBM Instana 监视 。

过程

1. 生成证书。
2. 配置 IBM Instana 代理程序。
3. 配置队列管理器。
4. 验证并调试。

相关任务

第 132 页的『将 IBM MQ 与 IBM Instana 跟踪集成』
IBM Instana 可用于跟踪 IBM Cloud Pak for Integration 中的事务。

OpenShift CP4I Operator2.2.0 为 *IBM Instana* 代理程序和队列管理器生成证书和密钥

对于 IBM Instana 代理程序与队列管理器之间的 TLS 通信，两者都必须具有证书和相应的专用密钥。

开始之前

这是 使用 TLS 配置已认证的 IBM Instana 监视的四个任务中的第一个任务。

注: 生成这些证书时使用的值用于演示目的。 在生产环境中部署时，请确保证书的主体集和到期时间都合适。

过程

IBM MQ 队列管理器

要通过 TLS 与 IBM Instana 代理进行通信，队列管理器必须具有证书和相应的专用密钥。 如果您已具有这些内容，请跳过此部分。

1. 为队列管理器生成证书和专用密钥。

   运行以下命令:

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

IBM Instana 代理程序

要使代理与 IBM MQ 队列管理器执行 TLS 通信，该代理必须具有证书和相应的专用密钥。 如果您在 JKS 密钥库中已有要使用的专用密钥和证书，请跳过此部分。

2. 为 IBM Instana 代理程序生成证书和专用密钥。

   运行以下命令：

   ```
   openssl req \
          -newkey rsa:2048 -nodes -keyout application.key \
          -subj "/CN=instana-agent/OU=app team1" \
          -x509 -days 3650 -out application.crt
   ```

3. 将证书和专用密钥存储在 PKCS12 密钥库中。

   运行以下命令，将 *your_password* 替换为要用于保护密钥库的密码。 在所有后续步骤中执行此替换。

   ```
   openssl pkcs12 -export -out application.p12 -inkey application.key -in application.crt
   -passout pass:your_password
   ```

4. 将 PKCS12 密钥库转换为 JKS 密钥库。

   运行以下命令：

   ```
   keytool -importkeystore \
           -srckeystore application.p12 \
           -srcstoretype pkcs12 \
           -destkeystore application.jks \
           -deststoretype JKS \
           -srcstorepass your_password \
           -deststorepass your_password \
           -noprompt
   ```

5. 标注证书。

   运行以下命令：

   ```
   keytool -changealias -alias "1" -destalias "instana" -keypass your_password -keystore
   application.jks -storepass your_password -noprompt
   ```

6. 将队列管理器证书导入到密钥库中。

   运行以下命令：

   ```
   keytool -importcert -file server.crt -keystore application.jks -storepass your_password
   -alias myca -noprompt
   ```

下一步做什么

现在，您已准备好 配置代理程序以进行 IBM Instana 监视。

OpenShift CP4I Operator2.2.0 **Instana 监视: 配置代理程序**

将密钥库安装到 IBM Instana 代理程序，然后为特定队列管理器配置监视。

开始之前

此任务假定您已 为 IBM Instana 代理程序和队列管理器生成证书和密钥。

过程

将密钥库安装到 IBM Instana 代理程序

1. 从 IBM Instana 代理程序名称空间中的 JKS 密钥库创建私钥。

   运行以下命令，将 *keystore_secret_name* 替换为要使用的名称。 在所有后续步骤中执行此替换。

   ```
   oc create secret generic keystore_secret_name --from-file=./application.jks -n instana-agent
   ```

2. 将秘密作为卷宗装入 Instana 特工工具包。

请在代理商 CustomResource 或 values.yaml Helm 图表中指定容积和容积安装：

```
agent:
  pod:
    volumeMounts:
    - mountPath: /opt/instana/agent/etc/jks-file-name.jks
      name: mq-key-jks-name
      subPath: jks-file-name.jks
    volumes:
    - name: mq-key-jks-name
      secret:
        secretName: keystore_secret_name
```

要对此进行编辑，请由 Instana Agent Operator 部署 Instana Agent，运行类似以下命令：

```
oc edit agent [instana-agent-name] -n [instana-agent-namespace]
```

有关在 Instana 代理程序中安装秘密的更多信息，请参阅 Instana 代理程序 Helm 图表文档中的卷和 volumeMounts。

配置特定队列管理器的监视

3. 在 instana-agent 名称空间中，使用 `oc edit configmap instana-agent` 命令来编辑 instana-agent configmap。
4. 在 `configuration.yaml: |` 下添加以下部分。 如果已定义此部分，那么只需将新队列管理器添加到列表。

```
com.instana.plugin.ibmmq:
  enabled: true
  poll_rate: 60
  queueManagers:
     QUEUE_MANAGER_NAME:
        channel: 'INSTANA.A.SVRCONN'
        keystorePassword: 'your_password'
        keystore: '/opt/instana/agent/etc/application.jks'
        cipherSuite: 'TLS_RSA_WITH_AES_256_CBC_SHA256'
```

其中：

- *your_password* 是 JKS 密钥库的密码
- *QUEUE_MANAGER_NAME* 是要部署的底层 IBM MQ 队列管理器的名称，而不是队列管理器操作的名称。

注: 如果 *QUEUE_MANAGER_NAME* 未设置为底层队列管理器名称，而是设置为 "操作"，那么监视将不起作用。 底层名称在 spec.queuemanager.name 中为队列管理器操作数定义。

5. 删除 instana-agent 名称空间中的 instana-agent pod。 这将导致它们重新启动，并开始使用新设置进行监视。

下一步做什么

现在，您已准备好 配置队列管理器以进行 IBM Instana 监视。

OpenShift CP4I Operator2.2.0

Instana 监视: 配置队列管理器

设置使用 TLS 与 IBM Instana 代理程序通信的队列管理器。此连接的认证是使用 SSLPEERMAP 完成的。

开始之前

此任务假定您已 配置代理程序以进行 IBM Instana 监视。

过程

1. 通过 MQSC 和 INI 配置队列管理器。

   MQSC 用于设置支持 TLS 的新通道，然后配置该通道以认证连接的 IBM Instana 代理程序 (如果它具有具有必需字段的证书)。 在这种情况下，我们会将任何具有包含字段 CN=instana-agent,OU=app

team1 的证书的连接客户机映射到用户 app1。 然后， MQSC 授予用户 app1 许可权以执行 IBM Instana 监视所需的操作。

INI 文件用于向外部用户 app1 授予许可权。

以下 configmap 包含必需的 MQSC 和 INI 设置。 将其部署到队列管理器名称空间中。

```
apiVersion: v1
data:
  channel.mqsc: |-
    DEFINE CHANNEL('INSTANA.A.SVRCONN') CHLTYPE(SVRCONN) SSLCAUTH(REQUIRED)
SSLCIPH('ANY_TLS12_OR_HIGHER')
    ALTER QMGR CONNAUTH(' ')
    REFRESH SECURITY
    SET CHLAUTH('INSTANA.A.SVRCONN') TYPE(SSLPEERMAP) SSLPEER('CN=*') USERSRC(NOACCESS)
ACTION(REPLACE)
    SET CHLAUTH('*') TYPE(ADDRESSMAP) ADDRESS('*') USERSRC(NOACCESS) ACTION(REPLACE)
    SET CHLAUTH('INSTANA.A.SVRCONN') TYPE(SSLPEERMAP) SSLPEER('CN=instana-agent,OU=app
team1') USERSRC(MAP) MCAUSER('app1')
    SET AUTHREC PRINCIPAL('app1') OBJTYPE(QMGR) AUTHADD(ALL)
    SET AUTHREC PROFILE('SYSTEM.ADMIN.COMMAND.QUEUE') PRINCIPAL('app1') OBJTYPE(QUEUE)
AUTHADD(PUT,INQ,DSP,CHG)
    SET AUTHREC PROFILE('SYSTEM.**') PRINCIPAL('app1') OBJTYPE(TOPIC) AUTHADD(DSP)
    SET AUTHREC PROFILE('*') PRINCIPAL('app1') OBJTYPE(TOPIC) AUTHADD(DSP)
    SET AUTHREC PROFILE('SYSTEM.**') PRINCIPAL('app1') OBJTYPE(QUEUE) AUTHADD(DSP, CHG, GET)
    SET AUTHREC PROFILE('SYSTEM.**') PRINCIPAL('app1') OBJTYPE(LISTENER) AUTHADD(DSP)
    SET AUTHREC PROFILE('AMQ.*') PRINCIPAL('app1') OBJTYPE(QUEUE) AUTHADD(DSP, CHG)
    REFRESH SECURITY TYPE(CONNAUTH)
  auth.ini: |-
    Service:
      Name=AuthorizationService
      EntryPoints=14
      SecurityPolicy=UserExternal
kind: ConfigMap
metadata:
  namespace: your-queue-manager-namespace
  name: qmgr-monitoring-config
```

其中 *your-queue-manager-namespace* 是将在其中部署队列管理器的名称空间。

注: 如果要监视用户定义的队列，那么必须向 configmap MQSC 添加其他行，向这些队列授予 DSP , CHG 和 GET 许可权。 例如：

```
SET AUTHREC PROFILE('MYQUEUE') PRINCIPAL('app1') OBJTYPE(QUEUE) AUTHADD(DSP, CHG, GET).
```

此示例将 configmap 用于 MQSC 和 INI 数据，但如果您进行的任何添加都是保密的，那么可以使用私钥。 有关使用 MQSC 和 INI 进行部署的常规信息，请参阅 第 87 页的『示例: 提供 MQSC 和 INI 文件』。

2. 要建立 TLS 连接，队列管理器必须信任 IBM Instana 代理程序的证书。 要实现此目的，请创建仅包含 IBM Instana 代理程序证书的私钥:

```
oc create secret generic instana-certificate-secret --from-file=./application.crt -n your-
queue-manager-namespace
```

3. 队列管理器必须为 TLS 握手提供自己的证书，并且需要访问关联的专用密钥。 部署包含先前创建或已拥有的密钥和证书的私钥:

```
oc create secret tls qm-tls-secret --cert server.crt --key server.key -n your-queue-manager-
namespace
```

通过创建 configmap 和私钥，您已准备好创建队列管理器本身。

4. 确保队列管理器 YAML 未在队列管理器容器中设置环境变量 **MQSNOAUT** 。

否则，在启用后，认证机制将不起作用。 在部署后除去该变量不会导致重新启用机制，并且必须重新创建队列管理器。

5. 将以下部分添加到队列管理器定义，其中 *MYQM* 是队列管理器的名称:

```
spec:
  queueManager:
    name: MYQM   #(a)
    ini:   #(b)
```

```
      - configMap:
          items:
            - auth.ini
          name: qmgr-monitoring-config
    mqsc:   #(c)
      - configMap:
          items:
            - channel.mqsc
          name: qmgr-monitoring-config
  pki:
    keys:   #(d)
        - name: default
          secret:
            items:
            - tls.key
            - tls.crt
            secretName: qm-tls-secret
    trust:   #(e)
      - name: app
        secret:
          items:
            - application.crt
          secretName: instana-certificate-secret
```

规范的标记部分描述如下:

a. 请确保为底层队列管理器提供了唯一名称。 如果底层队列管理器没有唯一名称，那么监视可能无法按预期工作。 此名称必须与先前编辑的 IBM Instana 代理程序 configmap 中的名称匹配。

b. 写入 configmap 的 INI 信息将添加到队列管理器。

c. 写入 configmap 的 MQSC 信息将添加到队列管理器。

d. 队列管理器证书和专用密钥将添加到队列管理器密钥库。

e. IBM Instana 代理程序证书将添加到队列管理器信任库。

6. 可选： 对受监视队列管理器启用 IBM Instana 跟踪。

   如果要执行此操作，请参阅 第 132 页的『将 IBM MQ 与 IBM Instana 跟踪集成』。

7. 部署队列管理器。

下一步做什么

现在，您已准备好 验证和调试 IBM Instana 监视。

OpenShift CP4I Operator2.2.0 *Instana* 监视*:* 验证和调试

要能够通过 IBM Instana 代理程序监视队列管理器，必须同时配置代理程序和队列管理器。

开始之前

此任务假定您已 配置队列管理器以进行 IBM Instana 监视。

过程

正在验证

1. 要验证您在部署中是否成功，请在 IBM Instana 仪表板中查看队列管理器。

   队列管理器应该在应用程序页面的服务部分以及 "基础结构" 视图中可视。

调试

注: 这些调试步骤假定 IBM Instana 代理程序的 Openshift 部署作为 daemonset 运行。

如果在 IBM Instana 仪表板中看不到队列管理器，那么您可能已错误配置了队列管理器。 使用以下步骤进行调查。

2. 标识正在运行活动队列管理器 pod 的节点。

   在队列管理器名称空间中运行以下命令:

```
oc get pods -o wide -n your-queue-manager-namespace
```

3. 要确定哪个 IBM Instana 代理程序 pod 与队列管理器在同一节点上运行，请在 instana-agent 名称空间中运行同一命令:

```
oc get pods -o wide -n instana-agent-namespace
```

4. 为了帮助了解 IBM Instana 代理程序端的任何问题，请获取 IBM Instana 代理程序 pod 的日志，并查找与 "mq" 或队列管理器名称相关的条目。

   运行以下命令:

```
oc logs instana-agent-pod -c instana-agent -n instana-agent
```

5. 请检查队列管理器日志。

   如果代理尝试连接到队列管理器，那么队列管理器日志应指示连接不成功的原因。 运行以下命令:

```
oc logs your-queue-manager-name -n your-queue-manager-namespace
```

结果

您已完成 使用 TLS 配置已认证的 IBM Instana 监视的所有四个任务。

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 Red Hat OpenShift IBM Entitled Registry 的私钥，请遵循 创建权利密钥私钥的步骤。

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:9.4.3.0-r1
      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:9.4.3.0-r1"
      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 5 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 85 页的『使用 IBM MQ Operator 部署简单队列管理器』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 85 页的『使用 IBM MQ Operator 部署简单队列管理器』
此示例部署 "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 IBM 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。

OpenShift CP4I 使用 IBM MQ 许可证注释配置队列管理器。 IBM MQ Operator

必须将注释应用于队列管理器，才能使用 IBM MQ 许可证对其进行配置。

开始之前

注: 这些说明仅适用于使用 IBM MQ Operator 的情况。 如果要在 Kubernetes 上部署队列管理器，请参阅 IBM MQ 容器许可证注释。

关于此任务

您可以在创建队列管理器时向队列管理器添加注释，也可以通过 spec.annotations 更新现有队列管理器。

要点: 更新队列管理器 YAML 会触发 pod 重启。

过程

按照 第 142 页的『通过使用 IBM MQ Operator』 中的说明更新现有队列管理器的许可注释，根据权限使用每个队列管理器所需的许可注释。

必要的许可证注释

IBM MQ

```
spec:
  annotations:
    productID: c661609261d5471fb4ff8970a36bccea
    productName: IBM MQ
```

IBM MQ for Non-Production Environment

```
spec:
  annotations:
    productID: 151bec68564a4a47a14e6fa99266deff
    productName: IBM MQ for Non-Production Environment
```

IBM MQ 容器多实例

```
spec:
  annotations:
    productID: 2dea73b866b648b6b4abe2a85eb76964
    productName: IBM MQ Container Multi Instance
```

IBM MQ 用于非生产环境的容器多实例：

```
spec:
  annotations:
    productID: af11b093f16a4a26806013712b860b60
    productName: IBM MQ Container Multi Instance for Non-Production Environment
```

IBM MQ CP4I

```
spec:
  annotations:
    cloudpakId: c8b82d189e7545f0892db9ef2731b90d
    cloudpakName: IBM Cloud Pak for Integration
    productID: c661609261d5471fb4ff8970a36bccea
    productName: IBM MQ
    productMetric: VIRTUAL_PROCESSOR_CORE
    productCloudpakRatio: 4:1
```

IBM MQ 用于非生产环境，具有 权限 CP4I

```
spec:
  annotations:
    cloudpakId: c8b82d189e7545f0892db9ef2731b90d
    cloudpakName: IBM Cloud Pak for Integration
    productID: 151bec68564a4a47a14e6fa99266deff
    productName: IBM MQ for Non-Production Environment
    productMetric: VIRTUAL_PROCESSOR_CORE
    productCloudpakRatio: 8:1
```

IBM MQ 具有 权限的容器多实例 CP4I

为：

```
spec:
  annotations:
    productName: IBM MQ Container Multi Instance
```

```
    productID: 2dea73b866b648b6b4abe2a85eb76964
    productMetric: VIRTUAL_PROCESSOR_CORE
    productCloudpakRatio: 10:3
    cloudpakName: IBM Cloud Pak for Integration
    cloudpakId: c8b82d189e7545f0892db9ef2731b90d
```

IBM MQ 用于非生产环境的容器多实例 权限 CP4I

```
spec:
  annotations:
    cloudpakId: c8b82d189e7545f0892db9ef2731b90d
    cloudpakName: IBM Cloud Pak for Integration
    productID: af11b093f16a4a26806013712b860b60
    productName: IBM MQ Container Multi Instance for Non-Production Environment
    productMetric: VIRTUAL_PROCESSOR_CORE
    productCloudpakRatio: 20:3
```

OpenShift CP4I 通过使用 IBM MQ Operator

您可以在创建队列管理器时添加注释，也可以通过使用 spec.annotations.NET 更新现有队列管理器来添加注释。

关于此任务

只有当以下声明均为真时，这些说明才适用：

- 您正在使用 IBM MQ Operator。 如果要在 Kubernetes 上部署队列管理器，请参见 第 249 页的『IBM MQ 集装箱许可证注释』 。
- 您正在编辑现有队列管理器。 对于新队列管理器，可在创建时添加注释。

要点: 更新队列管理器 YAML 会触发 pod 重启。

过程

- **方案 1：** 使用 OpenShift 控制台添加注释。
  a) 导航至队列管理器
    a. 使用 Red Hat OpenShift Container Platform 群集管理员凭据登录 OpenShift 控制台。
    b. 将**项目**更改为安装 IBM MQ Operator 的命名空间。 从**项目**下拉列表中选择命名空间。
    c. 在导航窗格中，单击**操作符已安装的** > **操作符**。
    d. 在已安装操作程序面板的列表中，找到并单击 **IBM MQ**.
    e. 单击**队列管理器**选项卡。
    f. 单击所需的**队列管理器名称**。
  b) 在队列管理器中添加所需的注释。
    a. 单击 **YAML** 选项卡。
    b. 找到 spec 字段。
    c. 向队列管理器添加所需的注释。 如果 annotations 字段尚未创建，则创建该字段。
    d. 单击**保存**。

       重新启动队列管理器 pod 并应用注释。
  c) 根据需要对其他队列管理器重复上述步骤。
- **方案 2：** 使用 OpenShift 命令行界面 (CLI) 添加注释。
  a) 使用 oc login 命令登录 Red Hat OpenShift CLI。
  b) 将 project （命名空间）设置为部署队列管理器的命名空间：

```
oc project <QUEUE-MANAGER-NAMESPACE>
```

  c) 确定要更新的队列管理器名称：

```
oc get queuemanager
```

d) 编辑队列管理器，将所需注释添加到队列管理器中：

```
oc edit queuemanager <QUEUE-MANAGER-NAME>
```

e) 根据需要对其他队列管理器重复上述步骤。

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。

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 Operator2.1.0 禁用队列管理器规范的缺省值更新

IBM MQ Operator 使用其缺省值更新队列管理器规范中的任何未指定值。 如果要避免对队列管理器规范进行任何修改，那么可以禁用此行为。 仍将更新队列管理器状态字段。

过程

- 禁用队列管理器缺省值更新。

  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.mq/write-defaults-spec" : "false"
```

  注: quickstart 示例在缺省情况下应用了此注释。

使用只读根文件系统运行 IBM MQ 容器

您可以将 IBM MQ 容器配置为使用只读根文件系统运行。 这可防止攻击者在容器中复制和运行恶意代码。

关于此任务

启用只读根文件系统会使容器文件不可更改。 即，在容器文件系统上，可以查看文件但不能修改文件，并且不能创建新文件。 只能在已安装的文件系统上修改或创建文件。

启用只读根文件系统时，将创建两个临时卷 Scratch 和 Tmp ，并分别安装在容器中的 /run 和 /tmp 目录中。

- **临时** 卷包含用于配置队列管理器的文件，密钥库和其他文件。
- Tmp 卷包含诊断文件，例如队列管理器 RAS 文件。

由于这些卷是临时卷，因此在 pod 重新启动时，这些卷上的文件将丢失。

为队列管理器数据创建的卷的类型取决于存储类型。 缺省情况下，将安装持久卷。 或者，如果存储类型为 ephemeral，那么将安装临时卷。 如果卷中的数据大小超过为 **sizeLimit** 属性指定的值，那么 Kubernetes 可以弹出容器并创建新的容器。

缺省情况下，未启用只读根文件系统。 要将其启用，请完成以下步骤:

过程

1. 使用 spec.securityContext API 来启用只读根文件系统。

   对于队列管理器，将 第 237 页的『.spec.securityContext』 中的 **readOnlyRootFilesystem** 属性设置为 true。

   IBM MQ Operator 创建两个临时卷 Scratch 和 Tmp。
2. 可选： 设置或更改队列管理器数据存储类型。

   缺省情况下，持久卷声明安装在 /mnt/mqm 上。 或者，如果 **type** 属性在 第 236 页的『.spec.queueManager.storage.queueManager』 中设置为 ephemeral ，那么将创建并安装临时卷。
3. 对于每个临时卷，请仔细考虑数据可能增长的程度。 相应地设置 **sizeLimit** 属性的值，包括 SI 单元。
   - 对于 Scratch 临时卷，请在 第 237 页的『.spec.queueManager.storage.scratch』 中设置 **sizeLimit** 属性。 缺省值为 "100M"。
   - 对于 Tmp 临时卷，请在 第 237 页的『.spec.queueManager.storage.tmp』 中设置 **sizeLimit** 属性。 缺省值为 "2Gi"。

- 如果队列管理器卷的 **type** 设置为 ephemeral，请在 第 236 页的『.spec.queueManager.storage.queueManager』中设置 **sizeLimit** 属性。 缺省值为 "2Gi"。

OpenShift V 9.4.0 使用 IBM MQ Operator 使用基本注册表配置 IBM MQ Console

要登录到 IBM MQ Console，您可以向队列管理器提供自己的配置。

开始之前

如果使用 IBM MQ Advanced for Developers 许可证部署队列管理器，那么会内置简单配置。 请参阅 第 259 页的『用于描述如何为 admin 和 app 用户指定密码的示例队列管理器 YAML』。 如果要部署 IBM Cloud Pak for Integration 许可证队列管理器，那么可以启用与 IBM Cloud Pak for Integration Keycloak 的集成，以使用单点登录登录来登录到 IBM MQ Console 。 请参阅第 191 页的『连接到 Red Hat OpenShift 集群中部署的 IBM MQ Console』。

过程

1. **创建密码并使用 securityUtility 对其进行加密**。

   ConfigMap 用于存储用于访问队列管理器的凭证。 为了提高安全性，请使用 securityUtility 命令对这些凭证进行编码。

   或者，您可以使用私钥来保护 Kubernetes 层中的凭证。 但是，监视或故障诊断工具可能会过早地公开底层文件。

2. 可选： **登录到 Red Hat OpenShift 命令行界面 (CLI)**。

   如果使用 OpenShift CLI ，请使用 `oc login` 登录。

   或者，您可以使用 OpenShift 控制台。

3. **使用配置创建 ConfigMap** 。

   有关创建 XML 配置的帮助，请参阅 IBM MQ Console 和 REST API 安全性。

   以下示例在组 MQWebAdminGroup 中创建用户。 将为 MQWebAdminGroup 的成员分配 MQWebAdmin 角色。 在该示例中：

   - **必须** 将 *USERNAME* 和 *PASSWORD* 替换为您自己的值。 请注意，在此示例中使用了两次 *USERNAME* 。

     必须 将 *NAMESPACE* 指定为部署了 IBM MQ Operator 的队列管理器以及将部署或已部署队列管理器的队列管理器。

   a) 使用 OpenShift 控制台或命令行来创建以下 ConfigMap:

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: mqwebuserconfigmap
  namespace: NAMESPACE
data:
  mqwebuser.xml: |
    <?xml version="1.0" encoding="UTF-8"?>
    <server>
      <featureManager>
        <feature>appSecurity-2.0</feature>
        <feature>basicAuthenticationMQ-1.0</feature>
      </featureManager>
      <enterpriseApplication id="com.ibm.mq.console">
        <application-bnd>
          <security-role name="MQWebAdmin">
            <group name="MQWebAdminGroup" realm="defaultRealm"/>
          </security-role>
        </application-bnd>
      </enterpriseApplication>
      <basicRegistry id="basic" realm="defaultRealm">
        <user name="USERNAME" password="PASSWORD"/>
        <group name="MQWebAdminGroup">
```

```
          <member name="USERNAME"/>
        </group>
      </basicRegistry>
      <sslDefault sslRef="mqDefaultSSLConfig"/>
    </server>
```

b) 可选： 如果使用命令行，请应用 ConfigMap:

```
oc apply -f mqwebuserconfigmap.yaml
```

对于其余步骤，请选择下列其中一个选项:

- 使用配置部署新的队列管理器以访问 IBM MQ Console。
- 应用用于授予 IBM MQ Console 对现有队列管理器的访问权的配置。

4. 可选： **使用配置部署新的队列管理器以访问 IBM MQ Console**。

a) 创建队列管理器。

将认证和授权提供程序设置为 manual ，并通过下列其中一个选项提供新创建的 ConfigMap mqwebuserconfigmap :

- 选项 1: 通过队列管理器 YAML

  在队列管理器 YAML 的 web 部分下添加以下代码:

```
...
web:
    enabled: true
    console:
      authentication:
        provider: manual
      authorization:
        provider: manual
    manualConfig:
      configMap:
        name: mqwebuserconfigmap
```

- 选项 2: 通过 OpenShift 控制台 "表单" 视图:

  i) 在 OpenShift 控制台上，选择 **操作程序** > **已安装的操作程序**。

  ii) 选择 IBM MQ Operator 的部署。

  iii) 选择 **队列管理器** ，然后单击 **创建 QueueManager**。

  iv) 选择队列管理器的相关选项。

  v) 选择 **Web** 并将 **启用 Web 服务器** 设置为 true。

  vi) 打开 **高级配置** 列表框。

  vii) 在 **控制台** 列表框下，将 **认证** 和 **授权** 的 **提供程序** 设置为 **手动**。

  viii) 打开 **配置** 列表框。

  ix) 打开 **ConfigMap** 列表框，然后选择在步骤 第 145 页的『3』中创建的 ConfigMap mqwebuserconfigmap 。

  x) 单击**创建**。

现在，您可以通过在步骤 第 145 页的『3』中创建的 ConfigMap 中指定的凭证来访问新队列管理器的 IBM MQ Console 。

5. 可选： **应用用于对现有队列管理器启用 IBM MQ Console 的配置**。

编辑要为其启用 IBM MQ Console 的队列管理器的 YAML:

a. 在 OpenShift 控制台上，选择 **操作程序** > **已安装的操作程序**。

b. 选择 IBM MQ Operator 的部署。

c. 选择 **队列管理器** ，然后选择队列管理器的名称。

d. 选择 **YAML**。

e. 将队列管理器 YAML 的现有 web 部分替换为以下代码:

```
...
web:
    enabled: true
    console:
      authentication:
        provider: manual
      authorization:
        provider: manual
    manualConfig:
      configMap:
        name: mqwebuserconfigmap
```

f. 单击**保存**。

现在，您可以通过在步骤 第 145 页的『3』中创建的 ConfigMap 中指定的凭证来访问现有队列管理器的 IBM MQ Console 。

OpenShift V 9.4.0 V 9.4.0 扩展持久卷

如果存储器提供者支持卷扩展，请使用此任务来扩展持久卷。 根据存储器提供程序，扩展可能在联机或脱机时进行。

开始之前

成功的卷扩展依赖于存储器提供者来实现扩展请求。 请参阅存储提供程序文档以确定是否支持联机调整大小，并获取有关脱机调整大小过程的信息。

如果存储提供程序无法满足扩展请求，那么持久卷声明可能会进入警告或错误状态。 如果扩展失败，OpenShift 管理员可以手动恢复 "持久卷声明" 状态并取消扩展。 请参阅 Red Hat OpenShift 文档中的 在扩展卷时从故障中恢复 。

关于此任务

为了帮助管理持久存储器， Kubernetes 定义了两个 API 资源:

- PersistentVolume (PV) ，这是集群中由管理员供应或使用存储类动态供应的存储器。 可以静态或动态地进行供应。
- PersistentVolume 声明 (PVC) ，这是用户对存储的请求。 它还充当对资源的声明检查。

有关更多信息，请参阅 Kubernetes 文档中的 持久卷 。

警告:

- 如果用于创建队列管理器 PVC 的存储类不支持联机调整大小，那么将进行脱机调整大小。 在脱机期间，需要调整用户干预大小以完成卷扩展，因此队列管理器会经历停机时间。
- 对于 多实例队列管理器的共享卷的脱机调整大小，在执行用户干预时，必须同时关闭活动 pod 和备用 pod。
- OpenShift 不支持减小 PVC 的大小。 尝试减小持久卷的大小将使队列管理器处于 "失败" 状态。
- 此过程不适用于临时卷。

要展开 IBM MQ 容器使用的 PV ，请完成以下步骤。

过程

1. **准备扩展卷**

   a) 决定要扩展的卷。

   b) 确定卷正在使用的一个或多个存储类。

   例如:

```
spec:
  queueManager:
```

```
    storage:
      persistedData:
        enabled: true
        type: persistent-claim
        class: ocs-storagecluster-cephfs (1)
      queueManager:
        type: persistent-claim
      recoveryLogs:
        enabled: true
        type: persistent-claim
      defaultClass: ocs-storagecluster-ceph-rbd (2)
```

备注信息：

- (1) 如果卷定义了特定存储类，那么此类型的 PVC 将使用此存储类。
- (2) 如果设置了 **defaultClass**，那么此存储类将用于没有特定存储类的所有卷。 如果未设置 **defaultClass**，并且卷类型未指定类，那么将使用集群的缺省存储类。

您还可以通过描述底层 PVC 来确认正在使用的存储类。 例如：

```
oc describe pvc pvc-name
```

c) 验证存储类是否支持卷扩展。

存储类可能定义了属性 **.allowVolumeExpansion**：

- 如果此属性设置为 `true`，那么支持卷扩展。
- 如果此属性设置为 `false`，或者未定义此属性，那么存储类不允许卷扩展。 在这种情况下，请参阅存储提供程序文档以了解是否可以启用此功能。

您还可以描述存储类以确定它是否支持卷扩展。 例如：

```
oc describe sc storage-class-name
```

d) 请参阅存储器提供者文档，以了解是使用联机还是脱机过程进行卷扩展。

脱机过程要求手动重新启动队列管理器 pod，而联机过程不需要。 请参阅存储器提供者文档以了解脱机调整大小过程。

e) 检查队列管理器是否具有状态条件以及原因 "StorageMismatch"。

如果队列管理器具有此状态条件，那么当您启用卷扩展时，将扩展条件中列出的卷。 如果不希望发生此情况，请更改与队列管理器定义中每个卷类型相关联的大小字段，以与供应的 PVC 相匹配。 对所有不匹配的卷执行此操作时，将除去状态条件。

2. **扩展卷**

警告：

- 如果先前已修改队列管理器定义中的任何卷大小字段，那么当 **.allowVolumeExpansion** 在队列管理器定义中设置为 `true` 时，卷将开始扩展。
- 由于文件系统限制或本地硬件的可用性，存储器提供者可能对卷的最大大小有限制。 为了避免发生故障，请在扩展卷之前验证存储提供程序文档中的这些限制。
- OpenShift 不支持缩减 PVC 大小。 如果扩展卷的大小，那么无法将其减小。 如果尝试执行此操作失败，那么 IBM MQ Operator 无法将 PVC 恢复到其原始状态。

说明卷扩展的示例队列管理器定义：

```
spec:
  queueManager:
    storage:
      allowVolumeExpansion: true (A)
      persistedData:
        enabled: true
        type: persistent-claim
        size: 3Gi (B)
      queueManager:
        type: persistent-claim
        size: 4Gi (B)
      recoveryLogs:
```

```
        enabled: true
        type: persistent-claim
        size: 3Gi (B)
```

a) 要允许对队列管理器进行卷扩展，请将队列管理器上的字段 **.spec.queueManager.storage.allowVolumeExpansion** (A) 设置为 `true`。

b) 现在，您可以增大任何已启用卷类型的大小字段 (B)。 应用这些更改将启动卷扩展。

3. **验证 PVC 是否已调整大小**。

注意:

- 卷扩展可能需要一些时间。 如果验证不成功，那么第一次考虑等待几分钟并再次验证。
- 仅当执行联机调整大小时，才会在不执行用户操作的情况下完成卷扩展。
- 某些存储器提供者会向上舍入您请求的存储器大小。 扩展卷的大小应该与您的请求相同或更大。

a) 请检查队列管理器以了解状态条件。 请参阅下表以了解条件，说明和建议的操作。

表 *1:* 存储状态条件

CONDITION	**消息**	**说明**
StorageMismatch	Storage sizes defined in the QueueManager resource do not match the capacity of one or more provisioned PVCs [pvc-list]. AllowVolumeExpansion is set to false in the QueueManager resource so the MQ Operator will not attempt to reconcile these differences.	由于在队列管理器定义中未将 **.allowVolumeExpansion** 设置为 `true`，因此不会发生卷扩展。
StorageExpansionPending	Volume expansion is pending for the following PVCs [pvc-list]	卷扩展仍在进行中。 如果此状态条件持续较长时间，请执行以下步骤以收集更多信息，因为可能正在进行脱机调整大小或未能调整大小。
Failed	有许多可能与存储器相关的消息可以创建 'Failed' 状态条件。 例如: 'MQ Queue Manager failed to deploy: persistentvolumeclaims "<pvc>" is forbidden: only dynamically provisioned pvc can be resized and the storageclass the provisions the pvc must support resize.'	如果队列管理器具有 'Failed' 状态条件以及引用存储器的文本，请参阅状态条件中的消息。 此处给出的示例消息是使用不支持扩展的存储类引起的。

b) 对于已扩展的每个 PVC，请检查容量是否已增加以匹配或大于队列管理器定义中指定的值。

HA 队列管理器可能具有每种类型的多个 PVC。 要获取 PVC 的容量，请运行以下命令:

```
oc get pvc pvc-name -o template --template '{{.status.capacity.storage}}'
```

c) 检查 PVC 是否没有任何状态条件或事件建议调整失败的大小:

```
oc describe pvc pvc-name
```

- PVC 可能具有状态条件 `FileSystemResizePending`，消息为 "正在等待用户 (重新) 启动 pod 以完成节点上卷的文件系统调整大小"。 对于联机和脱机大小，会提出此状态条件。 对于联机调整大小，此状态条件在联机调整大小完成后在没有用户操作的情况下消失。
- 如果 PVC 具有指示调整大小失败的事件或状态条件，请参阅 Red Hat OpenShift 文档中的 在扩展卷时从故障中恢复 。

d) 请检查队列管理器 pod 是否没有任何状态条件或事件建议调整大小失败。 对于 HA 部署，请检查每个副本。

```
oc describe pod queue-manager-pod-name
```

- 如果 pod 具有指示失败调整大小的事件或状态条件，请参阅 Red Hat OpenShift 文档中的 在扩展卷时从故障中恢复 。 错误文本可帮助您解决问题，或者如果您在恢复后尝试重新调整大小，那么可防止发生相同的问题。

4. **在脱机调整大小时重新启动 pod**

   如果存储提供程序在扩展卷时使用脱机调整大小过程，那么要完成卷扩展，您需要重新启动用于安装要调整大小的卷的队列管理器 pod。

   对于多实例队列管理器，将在活动 pod 和备用 pod 之间共享恢复日志和持久数据卷。 要完成这些卷的大小调整，请同时关闭两个 pod。

   请参阅存储器提供者文档以了解其脱机调整大小过程。

OpenShift V 9.4.1 更正已部署队列管理器的许可证注释

在 9.4.1.0-r1 版本的 "IBM MQ 队列管理器之前，一些队列管理器在部署时使用了不正确的许可证注释。 从版本 9.4.1.0-r1 开始，所有队列管理器都会自动部署正确的许可证注释。

关于此任务

对于已经部署的 "IBM MQ 队列管理器，有两个选项可以更正许可证注释：

- 将队列管理器升级到 9.4.1.0-r1 版本（需要 'IBM MQ Operator33.3.0 或更新版本）。 请参阅 第 61 页的『升级 IBM MQ Operator 和队列管理器』。 请注意，如果队列管理器 YAML 包含无效的许可证注释，则需要在升级每个队列管理器时按提示解决。
- 对于未升级且使用以下许可证使用方式部署的旧版队列管理器，可通过更新队列管理器 YAML 来更正许可证注释。 完成以下步骤，将指定注释添加到队列管理器 YAML 中。 请注意，必须为每个队列管理器单独完成此过程。

  要点: 更新队列管理器 YAML 会触发 pod 重启。

受影响的许可证使用：

- IBM MQ Advanced for Non-Production Environment 与 "CP4I 的权利
- IBM MQ Advanced for Developers
- IBM MQ Advanced Container Multi Instance
- IBM MQ Advanced Container Multi Instance 与 "CP4I 的权利
- IBM MQ Advanced Container Multi Instance for Non-Production Environment 与 "CP4I 的权利

过程

按照 第 142 页的『通过使用 IBM MQ Operator』 中的说明更新现有队列管理器的许可注释，根据权限使用每个队列管理器所需的许可注释。

必要的许可证注释

IBM MQ Advanced for Non-Production Environment 与 "CP4I 的权利

```
spec:
  annotations:
    productName: IBM MQ Advanced for Non-Production Environment
```

IBM MQ Advanced for Developers

```
spec:
  annotations:
    productName: IBM MQ Advanced for Developers (Non-Warranted)
    productChargedContainers: [container_name]
```

其中 *container_name* 位于队列管理器 pod YAML 中，指定为 "`spec.containers[0].name`。

IBM MQ Advanced Container Multi Instance

```
spec:
  annotations:
    productID: bd35bff411bb47c2a3f3a4590f33a8ef
    productName: IBM MQ Advanced Container Multi Instance
```

IBM MQ Advanced Container Multi Instance 与 "CP4I 的权利

```
spec:
  annotations:
    productID: bd35bff411bb47c2a3f3a4590f33a8ef
    productName: IBM MQ Advanced Container Multi Instance
    productCloudpakRatio: 5:3
```

IBM MQ Advanced Container Multi Instance for Non-Production Environment 与 "CP4I 的权利

```
spec:
  annotations:
    productID: 31f844f7a96b49749130cd0708fdbb17
    productName: IBM MQ Advanced Container Multi Instance for Non-Production Environments
    productCloudpakRatio: 10:3
```

Kubernetes 在 Kubernetes 上部署和配置队列管理器

如何使用本地资源在 Kubernetes 上部署队列管理器。

关于此任务

MQ Adv. Kubernetes V 9.4.1 范例：在 "Kubernetes 中配置简单的队列管理器

本例部署了一个简单的 "快速启动 "队列管理器，它使用短暂（非持久）存储并关闭了 IBM MQ 安全性。 重新启动队列管理器时，信息不会被持久保存。 您可以调整配置，更改许多队列管理器设置。

开始之前

要完成此示例，您必须首先完成以下先决条件：

- 为本示例创建 "Kubernetes 命名空间。
- 在命令行上登录 "Kubernetes 群集，并切换到上述命名空间。
- 确保您已按照上述命名空间中 "第 75 页的『通过创建拉取秘钥，为使用 "Kubernetes 做准备』的步骤进行操作。

关于此任务

本例提供了 "`Service`、"`ServiceAccount` 和 "`StatefulSet`"YAML，用于定义队列管理器，并将其部署到 "Kubernetes 中。 此示例至少需要 "IBM MQ Advanced 9.4.1。

过程

1. 创建 "Service，为 "IBM MQ 流量提供一致的 IP 地址。

   确保您处于之前创建的命名空间中（请参阅 "开始之前"），然后使用 "Kubernetes 命令行或控制台应用以下 YAML。

```
apiVersion: v1
kind: Service
metadata:
  labels:
    app.kubernetes.io/component: integration
    app.kubernetes.io/instance: quickstart
    app.kubernetes.io/name: ibm-mq
    app.kubernetes.io/version: 9.4.3.0
  name: quickstart-ibm-mq
spec:
  ports:
  - name: qmgr
    port: 1414
    protocol: TCP
    targetPort: 1414
  - name: metrics
    port: 9157
    protocol: TCP
    targetPort: 9157
  selector:
    app.kubernetes.io/component: integration
    app.kubernetes.io/instance: quickstart
    app.kubernetes.io/name: ibm-mq
  type: ClusterIP
```

2. 创建用于队列管理器的 "ServiceAccount。

   下面的示例 "ServiceAccount 指的是 "imagePullSecret，在提取图像时用于权限检查。 有关更多信息，请参阅 第 75 页的『通过创建拉取秘钥，为使用 "Kubernetes 做准备』。

```
kind: ServiceAccount
apiVersion: v1
metadata:
  name: quickstart-ibm-mq
  labels:
    app.kubernetes.io/component: integration
    app.kubernetes.io/instance: quickstart
    app.kubernetes.io/name: ibm-mq
    app.kubernetes.io/version: 9.4.3.0
imagePullSecrets:
  - name: ibm-entitlement-key
```

3. 创建队列管理器 Pod，由 "StatefulSet 管理。

   在下面的 YAML 中，"namespace 被假定为 "mq-test。 将其改为与本例中在 "Kubernetes 环境中使用的名称一致。

   此 YAML 将在指定命名空间中的 "Kubernetes 环境中创建单例队列管理器。 没有为该实例创建持久卷，因为我们的示例是用于短暂存储。

   注: 本示例使用 IBM MQ Advanced 注释。 要使用 IBM MQ 许可证，请将 spec.template.metadata 下的注释编辑为 第 249 页的『IBM MQ 集装箱许可证注释』 中 IBM MQ 或 IBM MQ for Non-Production Environment 下指定的值。

```
kind: StatefulSet
apiVersion: apps/v1
metadata:
  name: quickstart-ibm-mq
  namespace: mq-test
  labels:
    app.kubernetes.io/component: integration
    app.kubernetes.io/instance: quickstart
    app.kubernetes.io/name: ibm-mq
    app.kubernetes.io/version: 9.4.3.0
spec:
  persistentVolumeClaimRetentionPolicy:
    whenDeleted: Retain
    whenScaled: Retain
```

```
  replicas: 1
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      app.kubernetes.io/component: integration
      app.kubernetes.io/instance: quickstart
      app.kubernetes.io/name: ibm-mq
  template:
    metadata:
      labels:
        app.kubernetes.io/component: integration
        app.kubernetes.io/instance: quickstart
        app.kubernetes.io/name: ibm-mq
        app.kubernetes.io/version: 9.4.3.0
        statefulSetName: quickstart-ibm-mq
      annotations:
# These annotations are critical for the IBM License Service to charge you correctly
        productChargedContainers: qmgr
        productID: 208423bb063c43288328b1d788745b0c
        productMetric: PROCESSOR_VALUE_UNIT
        productName: IBM MQ Advanced
        productVersion: 9.4.3.0
    spec:
      restartPolicy: Always
      serviceAccountName: quickstart-ibm-mq
      terminationGracePeriodSeconds: 30
      securityContext: {}
      containers:
        - resources:
            limits:
              cpu: '1'
              memory: 1Gi
            requests:
              cpu: '1'
              memory: 1Gi
          readinessProbe:
            exec:
              command:
                - chkmqready
            initialDelaySeconds: 10
            timeoutSeconds: 3
            periodSeconds: 5
            successThreshold: 1
            failureThreshold: 1
          terminationMessagePath: /run/termination-log
          name: qmgr
          livenessProbe:
            exec:
              command:
                - chkmqhealthy
            initialDelaySeconds: 90
            timeoutSeconds: 5
            periodSeconds: 10
            successThreshold: 1
            failureThreshold: 1
          env:
            - name: LICENSE
              # if you accept the license, change the following value to "accept"
              value: view
            - name: MQ_QMGR_NAME
              value: QUICKSTART
            - name: MQ_MULTI_INSTANCE
              value: 'false'
            - name: MQ_ENABLE_METRICS
              value: 'true'
            - name: MQ_ENABLE_EMBEDDED_WEB_SERVER
              value: 'false'
            - name: MQ_LOGGING_CONSOLE_FORMAT
              value: basic
            - name: DEBUG
              value: 'false'
            - name: MQ_ENABLE_TRACE_STRMQM
              value: 'false'
            - name: MQ_EPHEMERAL_PREFIX
              value: /run/mqm
            - name: MQ_GRACE_PERIOD
              value: '29'
            - name: MQ_NATIVE_HA
              value: 'false'
          securityContext:
            allowPrivilegeEscalation: false
            capabilities:
```

```
              drop:
                - ALL
            privileged: false
            runAsNonRoot: true
            readOnlyRootFilesystem: false
          ports:
            - containerPort: 1414
              protocol: TCP
            - containerPort: 9157
              protocol: TCP
          terminationMessagePolicy: File
# The image to pull.  This example uses a prebuilt MQ Advanced image from IBM Container
Registry
          image: cp.icr.io/cp/ibm-mqadvanced-server:9.4.1.0-r1
          imagePullPolicy: IfNotPresent
      serviceAccount: quickstart-ibm-mq
      dnsPolicy: ClusterFirst
  serviceName: qm
  podManagementPolicy: OrderedReady
  updateStrategy:
    type: OnDelete
```

4. 确认队列管理器正在运行

   现在正在部署队列管理器。 例如，通过列出队列管理器 pod 来确认其处于 "Running 状态：

   ```
   kubectl get pods --selector app.kubernetes.io/instance=quickstart
   ```

   请注意，这必须返回一个 pod，而且 "READY 字段必须显示值 "1/1。

5. 查看队列管理器状态

   运行以下命令，指定队列管理器 pod 的全名：

   ```
   kubectl exec -t <qmgr_pod_name> -- dspmq -m QUICKSTART
   ```

   返回的响应应类似于下面的内容：

   ```
   QMNAME(QUICKSTART) STATUS(Running)
   ```

结果

您已在 Kubernetes 平台上成功部署了一个简单的 IBM MQ 队列管理器（单实例类型）。

MQ Adv. Kubernetes V 9.4.1 示例：在 "Kubernetes 中配置本地 HA

此示例直接使用 YAML 资源在 "Kubernetes 中部署了使用本地高可用性功能的队列管理器。 相互 TLS 用于身份验证，将 TLS 证书映射到队列管理器中的身份。

开始之前

注： 本例使用 IBM MQ Advanced 许可证。 从 IBM MQ 9.4.2 ，您还可以通过向队列管理器添加额外许可，配置 IBM MQ 许可的队列管理器，以使用本地 HA 和跨区域复制功能。 有关更多信息，请参阅第 163 页的『在 IBM MQ 许可的队列管理器上配置本机 HA 和跨区域复制。 Kubernetes』。

要完成此示例，您必须首先完成以下先决条件：

- 为本示例创建 "Kubernetes 命名空间。
- 在命令行上登录 "Kubernetes 群集，并切换到上述命名空间。
- 确保您已按照上述命名空间中 "第 75 页的『通过创建拉取秘钥，为使用 "Kubernetes 做准备』的步骤进行操作。

关于此任务

本例提供了 "Service、"ServiceAccount 和 "StatefulSet"YAML，用于定义队列管理器，并将其部署到 "Kubernetes 中。 它还详细介绍了部署启用 TLS 的队列管理器所需的额外步骤。 此示例至少需要 "IBM MQ Advanced 9.4.1。

过程

1. 如 "第 90 页的『使用 OpenSSL 创建自签名 PKI』所述，创建一对证书。
2. 创建包含 MQSC 命令和 INI 文件的配置映射。

   创建 Kubernetes `ConfigMap`，其中包含 MQSC 命令，用于创建新队列和 SVRCONN 通道，以及添加允许访问通道的通道验证记录。

   确保您处于之前创建的命名空间中（请参阅 "开始之前"），然后使用 "Kubernetes 命令行应用以下 YAML。

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: example-nativeha-configmap
data:
  example.mqsc: |
    DEFINE CHANNEL('MTLS.SVRCONN') CHLTYPE(SVRCONN) SSLCAUTH(REQUIRED)
SSLCIPH('ANY_TLS13_OR_HIGHER') REPLACE
    SET CHLAUTH('MTLS.SVRCONN') TYPE(SSLPEERMAP) SSLPEER('CN=*') USERSRC(NOACCESS)
ACTION(REPLACE)
    SET CHLAUTH('MTLS.SVRCONN') TYPE(SSLPEERMAP) SSLPEER('CN=example-app1') USERSRC(MAP)
MCAUSER('app1') ACTION(REPLACE)
    SET AUTHREC PRINCIPAL('app1') OBJTYPE(QMGR) AUTHADD(CONNECT,INQ)
    DEFINE QLOCAL('EXAMPLE.QUEUE') REPLACE
    SET AUTHREC PROFILE('EXAMPLE.QUEUE') PRINCIPAL('app1') OBJTYPE(QUEUE)
AUTHADD(BROWSE,PUT,GET,INQ)
  example.ini: |
    Service:
        Name=AuthorizationService
        EntryPoints=14
        SecurityPolicy=UserExternal
    NativeHAInstance:
      Name=exampleqm-ibm-mq-0
      ReplicationAddress=exampleqm-ibm-mq-replica-0(9414)
    NativeHAInstance:
      Name=exampleqm-ibm-mq-1
      ReplicationAddress=exampleqm-ibm-mq-replica-1(9414)
    NativeHAInstance:
      Name=exampleqm-ibm-mq-2
      ReplicationAddress=exampleqm-ibm-mq-replica-2(9414)
```

   MQSC 定义了一个名为 *MTLS.SVRCONN* 的通道和一个名为 *EXAMPLE.QUEUE* 的队列。 通道被配置为只允许出示 "通用名 "为 *example-app1* 的证书的客户端访问。 这是在步骤 "第 155 页的『1』中创建的一个证书中使用的通用名称。 该通道上使用此通用名的连接会映射到 *app1* 用户 ID，该用户 ID 有权连接到队列管理器并访问示例队列。 INI 文件启用了一项安全策略，这意味着 *app1* 用户 ID 无需存在于外部用户注册表中，它只作为名称存在于本配置中。 它还会配置复制地址，以便与本地 HA 配合使用。
3. 创建 "`Service`，为 "IBM MQ 流量提供一致的 IP 地址。

```
apiVersion: v1
kind: Service
metadata:
  labels:
    app.kubernetes.io/component: integration
    app.kubernetes.io/instance: exampleqm
    app.kubernetes.io/name: ibm-mq
    app.kubernetes.io/version: 9.4.3.0
  name: exampleqm-ibm-mq
spec:
  ports:
  - name: qmgr
    port: 1414
    protocol: TCP
    targetPort: 1414
  - name: metrics
    port: 9157
    protocol: TCP
    targetPort: 9157
  selector:
    app.kubernetes.io/component: integration
    app.kubernetes.io/instance: exampleqm
    app.kubernetes.io/name: ibm-mq
  type: ClusterIP
```

4. 为每个本地 HA 副本创建内部 "`Service`，用于相互通信。

```
---
apiVersion: v1
kind: Service
metadata:
  labels:
    app.kubernetes.io/component: integration
    app.kubernetes.io/instance: exampleqm
    app.kubernetes.io/name: ibm-mq
    app.kubernetes.io/version: 9.4.3.0
  name: exampleqm-ibm-mq-replica-0
spec:
  ports:
  - port: 9414
    protocol: TCP
    targetPort: 9414
  publishNotReadyAddresses: true
  selector:
    app.kubernetes.io/component: integration
    app.kubernetes.io/instance: exampleqm
    app.kubernetes.io/name: ibm-mq
    statefulset.kubernetes.io/pod-name: exampleqm-ibm-mq-0
  type: ClusterIP
---
apiVersion: v1
kind: Service
metadata:
  labels:
    app.kubernetes.io/component: integration
    app.kubernetes.io/instance: exampleqm
    app.kubernetes.io/name: ibm-mq
    app.kubernetes.io/version: 9.4.3.0
  name: exampleqm-ibm-mq-replica-1
spec:
  ports:
  - port: 9414
    protocol: TCP
    targetPort: 9414
  publishNotReadyAddresses: true
  selector:
    app.kubernetes.io/component: integration
    app.kubernetes.io/instance: exampleqm
    app.kubernetes.io/name: ibm-mq
    statefulset.kubernetes.io/pod-name: exampleqm-ibm-mq-1
  type: ClusterIP
---
apiVersion: v1
kind: Service
metadata:
  labels:
    app.kubernetes.io/component: integration
    app.kubernetes.io/instance: exampleqm
    app.kubernetes.io/name: ibm-mq
    app.kubernetes.io/version: 9.4.3.0
  name: exampleqm-ibm-mq-replica-2
spec:
  ports:
  - port: 9414
    protocol: TCP
    targetPort: 9414
  publishNotReadyAddresses: true
  selector:
    app.kubernetes.io/component: integration
    app.kubernetes.io/instance: exampleqm
    app.kubernetes.io/name: ibm-mq
    statefulset.kubernetes.io/pod-name: exampleqm-ibm-mq-2
  type: ClusterIP
```

5. 创建用于队列管理器的 "ServiceAccount。

   下面的示例 "ServiceAccount 指的是 "imagePullSecret，在提取图像时用于权限检查。 有关更多信息，请参阅 第 75 页的『通过创建拉取秘钥，为使用 "Kubernetes 做准备』。

```
kind: ServiceAccount
apiVersion: v1
metadata:
  name: exampleqm-ibm-mq
  labels:
    app.kubernetes.io/component: integration
    app.kubernetes.io/instance: exampleqm
```

```
    app.kubernetes.io/name: ibm-mq
    app.kubernetes.io/version: 9.4.3.0
imagePullSecrets:
  - name: ibm-entitlement-key
```

6. 创建队列管理器 Pod，由 "StatefulSet 管理。

   以下面的 YAML 为例，进行这些更改：

   - 将 namespace 的名称从 mq-test 改为与本例中 Kubernetes 环境中使用的名称一致。
   - 如果在可用节点少于 3 个的 Kubernetes 集群上试用该示例，请删除 affinity 部分。
   - 可选择更改许可证。 示例使用 IBM MQ Advanced 许可证。 从 IBM MQ 9.4.2 ，您还可以通过向队列管理器添加额外许可，配置 IBM MQ 许可的队列管理器，以使用本地 HA 和跨区域复制功能。

     a. 将 spec.template.metadata 下的注释更改为 第 249 页的『IBM MQ 集装箱许可证注释』 中 IBM MQ 或 IBM MQ for Non-Production Environment 下指定的值。

     b. 在 第 163 页的『在 IBM MQ 许可的队列管理器上配置本机 HA 和跨区域复制。 Kubernetes』 中完成 第 164 页的『3』 步骤。 此步骤将添加 IBMLicensingDefinition CR，为本地 HA 和跨区域复制功能的队列管理器添加额外的许可注释

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  labels:
    app.kubernetes.io/component: integration
    app.kubernetes.io/instance: exampleqm
    app.kubernetes.io/name: ibm-mq
    app.kubernetes.io/version: 9.4.3.0
  name: exampleqm-ibm-mq
spec:
  persistentVolumeClaimRetentionPolicy:
    whenDeleted: Retain
    whenScaled: Retain
  updateStrategy:
    type: OnDelete
  podManagementPolicy: Parallel
  replicas: 3
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      app.kubernetes.io/component: integration
      app.kubernetes.io/instance: exampleqm
      app.kubernetes.io/name: ibm-mq
  serviceName: qm
  template:
    metadata:
      # These annotations are critical for the IBM License Service to charge you correctly
      annotations:
        productChargedContainers: All
        productID: 208423bb063c43288328b1d788745b0c
        productMetric: PROCESSOR_VALUE_UNIT
        productName: IBM MQ Advanced
        productVersion: 9.4.3.0
      labels:
        app.kubernetes.io/component: integration
        app.kubernetes.io/instance: exampleqm
        app.kubernetes.io/name: ibm-mq
        app.kubernetes.io/version: 9.4.3.0
        statefulSetName: exampleqm-ibm-mq
    spec:
      # Create an anti-affinity rule, so the Native HA replicas don't end up on the same Node
      affinity:
        podAntiAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
          - labelSelector:
              matchExpressions:
              - key: statefulSetName
                operator: In
                values:
                - exampleqm-ibm-mq
            topologyKey: kubernetes.io/hostname
      containers:
      # Environment variables to supply to MQ container
      - env:
        - name: LICENSE
          # if you accept the license, change the following value to "accept"
```

```
          value: view
        - name: MQ_QMGR_NAME
          value: EXAMPLEQM
        - name: MQ_MULTI_INSTANCE
          value: "false"
        - name: MQ_ENABLE_METRICS
          value: "true"
        - name: MQ_ENABLE_EMBEDDED_WEB_SERVER
          value: "false"
        - name: MQ_LOGGING_CONSOLE_FORMAT
          value: basic
        - name: DEBUG
          value: "false"
        - name: MQ_ENABLE_TRACE_STRMQM
          value: "false"
        - name: MQ_EPHEMERAL_PREFIX
          value: /run/mqm
        - name: MQ_GRACE_PERIOD
          value: "29"
        - name: MQ_NATIVE_HA
          value: "true"
        # The image to pull.  This example uses a prebuilt MQ Advanced image from IBM
Container Registry
        image: cp.icr.io/cp/ibm-mqadvanced-server:9.4.3.0-r1
        imagePullPolicy: IfNotPresent
        livenessProbe:
          exec:
            command:
            - chkmqhealthy
          failureThreshold: 1
          periodSeconds: 10
          successThreshold: 1
          timeoutSeconds: 5
        name: qmgr
        ports:
        - containerPort: 1414
          protocol: TCP
        - containerPort: 9157
          protocol: TCP
        - containerPort: 9414
          protocol: TCP
        readinessProbe:
          exec:
            command:
            - chkmqready
          failureThreshold: 1
          periodSeconds: 5
          successThreshold: 1
          timeoutSeconds: 3
        # CPU and memory resources to allocate
        resources:
          limits:
            cpu: "1"
            memory: 1Gi
          requests:
            cpu: "1"
            memory: 1Gi
        # Container security context.  Disable all root capabilities and privileges.
        securityContext:
          allowPrivilegeEscalation: false
          capabilities:
            drop:
            - ALL
          privileged: false
          readOnlyRootFilesystem: true
          runAsNonRoot: true
        startupProbe:
          exec:
            command:
            - chkmqstarted
          failureThreshold: 24
          periodSeconds: 5
          successThreshold: 1
          timeoutSeconds: 5
        terminationMessagePath: /run/termination-log
        terminationMessagePolicy: File
        volumeMounts:
        # Mount the main queue manager persistent volume.  Sym-linked from /var/mqm
        - mountPath: /mnt/mqm
          name: data
        # Mount space for temporary files
        - mountPath: /run
```

```
          name: scratch
        - mountPath: /tmp
          name: tmp
        # Mount the "default" key for TLS
        - mountPath: /etc/mqm/pki/keys/default
          name: default
          readOnly: true
        # Mount the key to use for Native HA replication traffic
        - mountPath: /etc/mqm/ha/pki/keys/ha
          name: ha-tls
          readOnly: true
        # Mount MQSC to be applied when the queue manager starts
        - mountPath: /etc/mqm/example.mqsc
          name: cm-example-nativeha-configmap
          readOnly: true
          subPath: example.mqsc
        # Mount an INI file to be applied when the queue manager starts
        - mountPath: /etc/mqm/example.ini
          name: cm-example-nativeha-configmap
          readOnly: true
          subPath: example.ini
      restartPolicy: Always
      securityContext: {}
      serviceAccount: exampleqm-ibm-mq
      serviceAccountName: exampleqm-ibm-mq
      automountServiceAccountToken: false
      terminationGracePeriodSeconds: 30
      volumes:
      - emptyDir:
          sizeLimit: 100M
        name: scratch
      # Temporary directory is used by default to store runmqras output, so make it larger
      - emptyDir:
          sizeLimit: 2Gi
        name: tmp
      - name: default
        secret:
          defaultMode: 288
          items:
          - key: tls.key
            path: tls.key
          - key: tls.crt
            path: tls.crt
          - key: ca.crt
            path: ca.crt
          secretName: example-qm-tls
      - name: ha-tls
        secret:
          defaultMode: 288
          secretName: example-qm-tls
      - configMap:
          defaultMode: 420
          items:
          - key: example.mqsc
            path: example.mqsc
          - key: example.ini
            path: example.ini
          name: example-nativeha-configmap
        name: cm-example-nativeha-configmap
  volumeClaimTemplates:
  - apiVersion: v1
    kind: PersistentVolumeClaim
    metadata:
      labels:
        app.kubernetes.io/component: integration
        app.kubernetes.io/instance: exampleqm
        app.kubernetes.io/name: ibm-mq
      name: data
      namespace: mq-test
    spec:
      accessModes:
      - ReadWriteOnce
      resources:
        requests:
          storage: 2Gi
      volumeMode: Filesystem
```

请注意，"Secret"示例-*qm-tls* 是在步骤 "第 155 页的『1』"中创建的，而 "ConfigMap"示例-*Nativeha*-配置图"是在步骤 "第 155 页的『2』中创建的。

为每个实例创建一个持久卷。 使用在 "Kubernetes 群集中配置的默认存储类。 如果默认情况下没有配置存储类，或想使用不同的存储类，请在 `spec.volumeClaimTemplates[0].spec` 下添加 `storageClassName: storage_class_name` 。

本地 HA 队列管理器中的三个 pod 通过网络复制数据。 该链接默认不加密，但本示例使用队列管理器的证书对流量进行加密。 您可以指定不同的证书，以提高安全性。 本地 HA TLS 密钥必须是具有特定结构的 "KubernetesTLS 密钥（例如，私钥必须名为 *tls.key*）。

请注意，"`updateStrategy` 被设置为 "`OnDelete`，这会影响更新的推出方式。 有关更多信息，请参阅第 40 页的『执行本机 HA 队列管理器的滚动更新的注意事项』

7. 确认队列管理器正在运行。

   现在正在部署队列管理器。 在继续操作前，请确认其处于 "`Running` 状态，例如列出队列管理器 pod：

   ```
   kubectl get pods --selector app.kubernetes.io/instance=exampleqm
   ```

   理想情况下，您应该连接到队列管理器，以确认它正在运行。 不过，从 "Kubernetes 集群外连接到队列管理器不在本说明的范围内，因为 "Kubernetes 分布和云环境之间会有很大差异。

8. 强制活动 pod 故障

   要验证队列管理器的自动恢复功能，请模拟 pod 故障：

   a) 查看活动和备用 pod

   重新运行上一条命令：

   ```
   kubectl get pods --selector app.kubernetes.io/instance=exampleqm
   ```

   请注意，在 "**READY** 字段中，活动 pod 返回值为 "`1/1`，而副本 pod 返回值为 "`0/1`。

   b) 删除活动 pod

   运行以下命令，指定活动 pod 的全名：

   ```
   kubectl delete pod exampleqm-ibm-mq-<active_pod_value>
   ```

   c) 再次查看 pod 状态

   运行以下命令：

   ```
   kubectl get pods --selector app.kubernetes.io/instance=exampleqm
   ```

   d) 查看队列管理器状态

   运行以下命令，指定其他 pod 的全名：

   ```
   kubectl exec -t <qmgr_pod_name> -- dspmq -o nativeha -x -m EXAMPLEQM
   ```

   例如，您应该看到状态显示活动实例已更改：

   ```
   QMNAME(EXAMPLEQM) 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)
   ```

结果

您已成功部署了具有本机高可用性和相互 TLS 身份验证功能的队列管理器，并验证了当活动 Pod 出现故障时，队列管理器会自动恢复。

V 9.4.2 示例：在 Kubernetes

本示例使用 YAML 资源，将队列管理器直接部署到 Kubernetes ， 并使用本机高可用性跨区域复制（Native HA CRR）功能。

开始之前

要完成此示例，您必须首先完成以下前提条件：

- 在部署实时和恢复组的每个集群中创建一个 Kubernetes 命名空间。
- 在命令行中，能够登录到任一 Kubernetes 集群，并切换到相关的命名空间。
- 请确保您已按照上述名称空间中的步骤操作，即 第 75 页的『通过创建拉取秘钥，为使用 "Kubernetes 做准备』 。

关于此任务

这个示例使用 YAML 资源，将两个队列管理器部署到 Kubernetes ，并使用本机 HA CRR 功能。 本地 HA 队列管理器之一将处于“活动”状态，另一个将处于“恢复”状态。 此示例基于在 第 154 页的『示例：在 "Kubernetes 中配置本地 HA 』 中创建的 YAML 文件。

此示例提供了在 NativeHA: ConfigMap, 服务、 StatefulSet, 和路线定义的基础上所需的修改。

过程

1. 创建新的配置映射，或在现有 ConfigMap 中定义新的.ini 数据定义（请参阅本机 HA 示例中的第 2 步），为实时或恢复队列管理器定义 INI 文件。

   创建一个包含定义的 Kubernetes ConfigMap ，用于配置以下组：

   a) 本地高可用性组作为恢复：

   ```
   apiVersion: v1
   kind: ConfigMap
   metadata:
     name: example-nativeha-configmap
   data:
   ...
     example-crr: |
       NativeHALocalInstance:
         GroupName=beta
         GroupRole=Recovery
         GroupLocalAddress=(9415)
         GroupCipherSpec=ANY_TLS12_OR_HIGHER
   ```

   当恢复组部署后，该组的复制地址即可确定，并发送至实时组的 ConfigMap。 例如，可以通过向已部署的恢复集群发出以下命令来完成：

   ```
   kubectl get route example-ibm-mq-crr-route -n <recovery-qm-namespace> -o
   jsonpath="{.spec.host}"
   ```

   b) 一个本地 HA CRR 组作为 Live（指向恢复组）：

   ```
   apiVersion: v1
   kind: ConfigMap
   metadata:
     name: example-nativeha-configmap
   data:
   ...
     example-crr: |
       NativeHALocalInstance:
         GroupName=alpha
         GroupRole=Live
         GroupLocalAddress=(9415)
         GroupCipherSpec=ANY_TLS12_OR_HIGHER
       NativeHARecoveryGroup:
         Enabled=Yes
         GroupName=beta
         ReplicationAddress=<address.to.recovery.cluster>(443)
   ```

2. 为本地高可用性 CRR 流量开放额外的 ClusterIP 端口，使其流入 CRR 组。 这可以通过创建一个新的 YAML 定义，或通过扩展本机 HA 示例中步骤 3 中的服务定义来实现：

   ```
   apiVersion: v1
   kind: Service
   metadata:
   ```

```
...
  name: exampleqm-ibm-mq
spec:
  ports:
...
  - name: ha-crr
    port: 9415
    protocol: TCP
    targetPort: 9415
...
  selector:
...
  type: ClusterIP
```

3. 定义一条新的路由进入集群，使其他本地高可用性 CRR 组能够与本地高可用性 CRR 组通信，并启用 TLS“直通”到步骤 2 中定义的新服务 ClusterIP 端口：

```
apiVersion: route.openshift.io/v1
kind: Route
metadata:
  name: "exampleqm-ibm-mq-crr-route"
spec:
  to:
    kind: Service
    name: exampleqm-ibm-mq
  port:
    targetPort: 9415
  tls:
    termination: passthrough
```

4. 修改在原生高可用性示例第 6 步中创建的 StatefulSet ， 以确保满足以下要求：

   a) 新的 ConfigMap INI 定义用于本地 HA CRR，以便队列管理器在启动时能够识别它：

```
...
        volumeMounts:
...
        # Mount the NativeHA CRR INI file to be applied when the queue manager starts
        - mountPath: /etc/mqm/example-crr.ini
          name: cm-example-nativeha-configmap
          readOnly: true
          subPath: example-crr.ini
...
      volumes:
...
      - configMap:
          defaultMode: 420
          items:
...
          - key: example-crr.ini
            path: example-crr.ini
          name: example-nativeha-configmap
        name: cm-example-nativeha-configmap
```

   b) 两个本地 HA CRR 组之间的通信必须加密。 因此，当前 NativeHA CRR 组的 TLS 密钥声明和其他 NativeHA CRR 组的信任证书需要定义并安装在 `/etc/mqm/groupha` 目录中。

   使用组名 alpha（代表“Live”）和 beta（代表“Recovery”）， Live 组的 YAML StatefulSet 示例修改如下：

```
...
        volumeMounts:
...
          - mountPath: /etc/mqm/groupha/pki/keys/ha-group
            name: groupha-tls
            readOnly: true
          - name: groupha-tls-trust-0
            mountPath: /etc/mqm/groupha/pki/trust/0
            readOnly: true
...
      volumes:
...
      - name: groupha-tls
        secret:
          secretName: nha-crr-secret-alpha
          defaultMode: 288
      - name: groupha-tls-trust-0
        secret:
```

```
          secretName: nha-crr-secret-beta
          defaultMode: 420
          items:
            - key: tls.crt
              path: native-crr-nha-crr-secret-beta-0-tls.crt
```

5. 在实时和恢复队列管理器实例的“就绪”和“运行”容器上使用以下 **dspmq** 命令，确认本机 HA CRR 组正在运行、同步、连接且达到法定人数：

```
kubectl exec -t <qmgr_pod_name> -- dspmq -o nativeha -g -m EXAMPLEQM
```

状态应显示两个 Native HA CRR 组已连接并作为实时和恢复组运行，例如：

```
QMNAME(EXAMPLEQM) ROLE(Active) INSTANCE(testmqqm-ibm-mq-0) INSYNC(yes) QUORUM(3/3)
GRPLSN(<0:0:10:30665>) GRPNAME(alpha) GRPROLE(Live)
 GRPNAME(alpha) GRPROLE(Live) GRPADDR(Unknown) GRPVER(9.4.2.0) GRSTATUS(Normal)
RCOVLSN(<0:0:10:30665>) RCOVTIME(2025-02-17T16:25:16.527089Z) INITLSN(<0:0:9:4684>)
INITTIME(2025-02-17T16:25:01.226650Z) LIVETIME(2025-02-17T16:25:02.974677Z)
ALTDATE(2025-02-17) ALTTIME(16.25.19)
 GRPNAME(beta) GRPROLE(Recovery) GRPADDR(<address.to.recovery.cluster>) GRPVER(9.4.2.0)
CONNGRP(yes) GRSTATUS(Normal) RCOVLSN(<0:0:10:30665>) RCOVTIME(2025-02-17T16:25:16.527089Z)
BACKLOG(0) INSYNC(yes) SYNCTIME(2025-02-17T16:29:38.643916Z) ALTDATE(2025-02-17)
ALTTIME(16.29.29)
```

结果

您已成功部署了两个队列管理器组，在组内和组间均具有原生高可用性和 TLS 身份验证。 您还确认了“实时”和“恢复”组已连接并执行跨区域复制。

V 9.4.2 Kubernetes 在 IBM MQ 许可的队列管理器上配置本机 HA 和跨区域复制。Kubernetes

从 IBM MQ 9.4.2 ， IBM MQ 许可的队列管理器可配置为使用本地 HA 跨区域复制 (CRR) 功能，方法是向队列管理器添加额外的许可。

开始之前

注: 这些说明仅适用于在 Kubernetes 上部署队列管理器的情况。 如果您部署了 IBM MQ Operator ，请参阅第 101 页的『在 IBM MQ 许可的队列管理器上配置本地 HA 和跨区域复制，方法是使用 IBM MQ Operator』。

- 在 IBM MQ 许可的队列管理器上使用本地 HA CRR 功能需要 IBM License Service 。 更多信息，请参阅 IBM Cloud Pak foundational services 文档中的在 Kubernetes 集群上安装 License Service。
- 在管理队列管理器 pod 的 `StatefulSet` 上， `metadata.labels.app.kubernetes.io/instance` 标签必须设置为唯一值。

过程

1. 在命令行上登录 Kubernetes 集群，然后切换到部署队列管理器的命名空间。
2. 将相关注释添加到队列管理器的注释中。

   请注意，本步骤中的说明适用于编辑现有队列管理器。 对于新队列管理器，在创建时添加注释。

   要点: 更新队列管理器注释需要重新启动 pod。

   a) 确定要更新的队列管理器 `StatefulSet` ：

   ```
   kubectl get statefulset
   ```

   b) 打开队列管理器的 `StatefulSet` 进行编辑：

   ```
   kubectl edit statefulset <QUEUE-MANAGER-STATEFUL-SET>
   ```

   c) 在 `StatefulSet` 中添加与收费组件相关的注释。 注意 `spec.template.metadata` 下还有其他注释；只需更改以下注释：

- 对于 IBM MQ：

```
spec:
  template:
    metadata:
      productID: c661609261d5471fb4ff8970a36bccea
      productName: IBM MQ
```

- 对于 IBM MQ for Non-Production Environment：

```
spec:
  template:
    metadata:
      productID: 151bec68564a4a47a14e6fa99266deff
      productName: IBM MQ for Non-Production Environment
```

d) 保存您所做的更改。 命令行上会显示类似下面的信息：

```
statefulset.apps/<QUEUE-MANAGER-STATEFUL-SET> edited
```

e) 删除由 StatefulSet 管理的所有队列管理器 pod。

f) 根据需要对其他队列管理器重复上述步骤。

3. 添加相关的 IBMLicensingDefinition CR，为本地 HA CRR 功能的队列管理器添加额外的许可注释。

a) 创建 IBMLicensingDefinition YAML 文件。

在以下示例中，请更改这些值：

- <IBMLicensingDefinition-UNIQUE-NAME>: IBMLicensingDefinition CR 的唯一名称。好的做法是在此名称中包含您的队列管理器名称。
- <QUEUE-MANAGER-NAMESPACE>:部署队列管理器的命名空间。
- <QUEUE-MANAGER-NAME>:队列管理器 metadata.labels.app.kubernetes.io/instance 的值。
- IBMLicensingDefinition CR 用于生产的 YAML 示例：

```
apiVersion: operator.ibm.com/v1
kind: IBMLicensingDefinition
metadata:
  name: <IBMLicensingDefinition-UNIQUE-NAME>
  namespace: <QUEUE-MANAGER-NAMESPACE>
spec:
  action: cloneModify
  condition:
    metadata:
      labels:
        app.kubernetes.io/instance: <QUEUE-MANAGER-NAME>
  scope: cluster
  set:
    productID: 46a57a4a66ee4b904e9b5544b53a2ba2
    productName: IBM MQ Native HA and Cross-Region Replication Add-on
```

- IBMLicensingDefinition CR 的 YAML 示例，供非生产使用：

```
apiVersion: operator.ibm.com/v1
kind: IBMLicensingDefinition
metadata:
  name: <IBMLicensingDefinition-UNIQUE-NAME>
  namespace: <QUEUE-MANAGER-NAMESPACE>
spec:
  action: cloneModify
  condition:
    metadata:
      labels:
        app.kubernetes.io/instance: <QUEUE-MANAGER-NAME>
  scope: cluster
  set:
    productID: 525b1331a74d4a209d9f6e9c908e374e
    productName: IBM MQ Native HA and Cross-Region Replication Add-on for Non-
Production Env
```

b) 应用 IBMLicensingDefinition YAML 文件：

```
kubectl apply -f <IBMLicensingDefinition-UNIQUE-NAME>.yaml
```

c) 根据需要对其他队列管理器重复上述步骤。

Kubernetes 使用 Helm 图表示例部署和配置队列管理器

您可以使用样本 Helm Chart 在 Kubernetes 上部署和配置队列管理器。

关于此任务

如果不使用 Red Hat OpenShift Container Platform，那么不支持 IBM MQ Operator 。 您可以使用样本 Helm Chart 以部署到其他类型的 Kubernetes 集群。

过程

- 有关如何使用 "Helm 部署自己的 "IBM MQ 容器映像，请参阅 "IBM MQ"Helm"图表样本。

相关参考

第 27 页的『支持容器中的 IBM MQ』
并非所有 IBM MQ 功能在容器中都可用且受支持的方式相同。

Podman 在 "Podman 上部署队列管理器

此示例运行预构建的 "IBM MQ Advanced Container 映像，将队列管理器部署到 "Podman 中。 同样的例子也可用于 "Docker。

开始之前

要验证本示例末尾的 TLS 配置：

- 安装 IBM MQ 客户机。 您需要使用 "**amqsputc** 和 "**amqsgetc** 命令来测试发送和接收信息。 这些命令可以作为 "IBM MQ 客户端的一部分安装。
  - 对于 "Windows 和 "Linux，从 "https://ibm.biz/mq94redistclients 为操作系统安装 "IBM MQ 可再分发客户端。
  - 对于 "Mac，从 "https://developer.ibm.com/tutorials/mq-macos-dev/下载并设置 "IBM MQ MacOS Toolkit。
- 请注意，没有预制的 "ARM64"IBM MQ Advanced Container 图像。

本例使用 "Podman，但也可使用 "Docker。

关于此任务

此示例运行预构建的 "IBM MQ Advanced Container 映像，将队列管理器部署到 "Podman 中。 它详细介绍了为报文传输配置 TLS 以及配置 "IBM MQ Console 和 "REST API 所需的其他步骤。 您可以跳过那些不需要的步骤来满足您的需求，从而选择配置其中的哪些步骤。

当队列管理器运行时，本示例使用在 "Podman 以外的机器（比如你的笔记本电脑）上运行的 "IBM MQ 示例程序，在示例客户端和队列管理器之间使用相互 TLS 验证信息的收发。 它还能验证登录 "IBM MQ Console，以及拨打管理和信息 "REST API 电话。

注: 您可以运行一个预构建的 "IBM MQ Advanced for Developers 容器镜像，它有一些额外的默认开发人员配置。 请参阅 第 258 页的『IBM MQ Advanced for Developers 容器映像 (container image)』。

过程

- **为客户端和队列管理器之间的信息传送配置 TLS**

  a) 按照 "第 90 页的『使用 OpenSSL 创建自签名 PKI』中的说明创建证书，执行所有步骤，但跳过 "**kubectl** 命令。

  a. 创建一个目录，例如名为 "`qm-pki`，其中包含为队列管理器生成的密钥和证书文件：

i) example-qm.key

ii) example-qm.crt

iii) ca.crt

运行容器的用户（uid 1001）需要 "example-qm.key 的读访问权限。 默认情况下，"OpenSSL 创建的密钥可由创建密钥的用户访问。

b. 创建一个目录，例如名为 "mq-client，其中包含为客户端生成的文件：

i) example-app1.p12

ii) 在 "Mac 上，您还需要 "example-app1-chain.crt

b) 创建一个包含 MQSC 命令的文件，例如 "**example-tls.mqsc**，以创建一个新队列和一个 "SVRCONN 通道，并添加一个允许访问通道的通道验证记录。

```
DEFINE CHANNEL('MTLS.SVRCONN') CHLTYPE(SVRCONN) SSLCAUTH(REQUIRED)
SSLCIPH('ANY_TLS13_OR_HIGHER') REPLACE
SET CHLAUTH('MTLS.SVRCONN') TYPE(SSLPEERMAP) SSLPEER('CN=*') USERSRC(NOACCESS)
ACTION(REPLACE)
SET CHLAUTH('MTLS.SVRCONN') TYPE(SSLPEERMAP) SSLPEER('CN=example-app1') USERSRC(MAP)
MCAUSER('app1') ACTION(REPLACE)
SET AUTHREC PRINCIPAL('app1') OBJTYPE(QMGR) AUTHADD(CONNECT,INQ)
DEFINE QLOCAL('EXAMPLE.QUEUE') REPLACE
SET AUTHREC PROFILE('EXAMPLE.QUEUE') PRINCIPAL('app1') OBJTYPE(QUEUE)
AUTHADD(BROWSE,PUT,GET,INQ)
```

MQSC 命令定义了一个名为 "MTLS.SVRCONN 的通道和一个名为 "EXAMPLE.QUEUE 的队列。 通道配置为只允许出示 "通用名 "为 "example-app1 的证书的客户访问。 这是上一步创建的证书中使用的通用名称。 该通道上使用此通用名称的连接会映射到一个名为 "app1 的用户 ID，该用户 ID 有权连接到队列管理器并访问示例队列。

c) 创建一个 INI 文件，例如 "example-tls.ini，启用安全策略，这样 "app1 用户 ID 就不需要存在于外部用户注册表中--它只作为名称存在于该配置中。

```
Service:
    Name=AuthorizationService
    EntryPoints=14
    SecurityPolicy=UserExternal
```

- **配置 "IBM MQ Console 和 "REST API**

在本例中，您在 "mqwebuser.xml 文件中配置了一个基本注册表。 XML 文件中的用户名、密码和角色用于验证和授权 "IBM MQ Console 和 "REST API 的用户。 更多信息，请参见 "为 "IBM MQ Console 和 "REST API"配置基本注册表 。

你应该考虑使用基本注册表是否适合你的生产环境。 参见 "IBM MQ Console 和 "REST API 安全。

a) 创建名为 "mqwebuser.xml 的文件，用于配置 "IBM MQ Console。

在本例中，创建了一个基本注册表，其中包含具有网络管理员和网络用户角色的用户。

您**必须**用自己的值替换 *ADMIN_PASSWORD* 和 *APP_PASSWORD*。 您也可以根据需要更改管理员和应用程序用户名，但要注意应用程序用户名应与上一条 MQSC 命令中指定的应用程序用户名一致。 请注意，示例中每个用户名都使用了两次。

```
<?xml version="1.0" encoding="UTF-8"?>
<server>
    <featureManager>
    <feature>appSecurity-2.0</feature>
    <feature>basicAuthenticationMQ-1.0</feature>
    </featureManager>
    <enterpriseApplication id="com.ibm.mq.console">
        <application-bnd>
            <security-role name="MQWebAdmin">
                <group name="MQWebAdminGroup" realm="defaultRealm"/>
            </security-role>
        </application-bnd>
    </enterpriseApplication>
    <enterpriseApplication id="com.ibm.mq.rest">
        <application-bnd>
            <security-role name="MQWebAdmin">
                <group name="MQWebAdminGroup" realm="defaultRealm"/>
```

```
            </security-role>
            <security-role name="MQWebUser">
                <group name="MQWebMessaging" realm="defaultRealm"/>
            </security-role>
        </application-bnd>
    </enterpriseApplication>
    <basicRegistry id="basic" realm="defaultRealm">
        <user name="admin" password="ADMIN_PASSWORD"/>
        <user name="app1" password="APP_PASSWORD"/>
        <group name="MQWebAdminGroup">
            <member name="admin"/>
        </group>
        <group name="MQWebMessaging">
            <member name="app1"/>
        </group>
    </basicRegistry>
    <sslDefault sslRef="mqDefaultSSLConfig"/>
</server>
```

b) 登录容器注册表。

获取权限密钥，然后用它登录容器注册表。 请参阅 第 76 页的『准备通过调用预构建的容器映像来使用 "Podman 或 "Docker』。

c) 在 "Podman 中运行队列管理器。

完成下列其中一个选项：

– **方案 1：通过加载包含队列管理器证书的目录，在 "Podman 中运行队列管理器。**

以下命令包括 TLS 信息传输配置以及 "IBM MQ Console 和 "REST API 的配置。 如果您跳过了之前描述的任何配置，因为这些配置不符合您的需要，请删除与跳过部分相关的参数。

确保命令指向先前创建的文件和目录的正确位置。

要在 "Podman 中运行队列管理器，请运行以下命令：

```
podman run --env MQ_ENABLE_EMBEDDED_WEB_SERVER=true  \
-v "./mqwebuser.xml:/etc/mqm/web/installations/Installation1/servers/mqweb/
mqwebuser.xml" \
-v "./example-tls.ini:/etc/mqm/example-tls.ini"  \
-v "./example-tls.mqsc:/etc/mqm/example-tls.mqsc"  \
-v "./qm-pki:/etc/mqm/pki/keys/qmlabel" --env LICENSE=accept --env MQ_QMGR_NAME=QM1 \
--publish 1414:1414 --publish 9443:9443 --detach --name QM1 cp.icr.io/cp/ibm-
mqadvanced-server:9.4.1.0-r1
```

该命令调用 "IBM MQ Advanced 队列管理器镜像，并启动容器。

– **Option 2: Run the queue manager in Podman using Podman secrets**

您可以使用 "Podman 机密加载队列管理器证书，而不是直接将包含这些证书的目录加载到容器中。

注: This option does not apply when using Docker, unless you are using Docker Swarm, because Docker secrets can only be used with Docker Swarm. 请参阅 dockerdocs 中的 docker secret create 命令。

完成以下步骤：

a. 为每个队列管理器 TLS 文件创建 "Podman 机密。 例如，从创建的 "qm-pki 目录中：

```
podman secret create example-qm.key ./example-qm.key
podman secret create example-qm.crt ./example-qm.crt
podman secret create ca.crt ./ca.crt
```

以下命令包括 TLS 信息传输配置以及 "IBM MQ Console 和 "REST API 的配置。 如果您跳过了之前描述的任何配置，因为这些配置不符合您的需要，请删除与跳过部分相关的参数。

确保命令指向先前创建的文件和目录的正确位置。

要在 "Podman 中运行队列管理器，请运行以下命令：

```
podman run --env MQ_ENABLE_EMBEDDED_WEB_SERVER=true \
-v "./mqwebuser.xml:/etc/mqm/web/installations/Installation1/servers/mqweb/
mqwebuser.xml" \
```

```
-v "./example-tls.ini:/etc/mqm/example-tls.ini" \
-v "./example-tls.mqsc:/etc/mqm/example-tls.mqsc" \
--secret example-qm.key,target=/etc/mqm/pki/keys/qmlabel/example-
qm.key,uid=1001,mode=0440 \
--secret example-qm.crt,target=/etc/mqm/pki/keys/qmlabel/example-qm.crt,uid=1001 \
--secret ca.crt,target=/etc/mqm/pki/keys/qmlabel/ca.crt,uid=1001 --env
LICENSE=accept --env MQ_QMGR_NAME=QM1 \
--publish 1414:1414 --publish 9443:9443 --detach --name QM1 cp.icr.io/cp/ibm-
mqadvanced-server:9.4.1.0-r1
```

注: 如果容器运行的用户被覆盖为不同的用户，请更改 UID 以匹配。

一个队列管理器正在 "Podman 中运行。

```
podman ps
```

验证

- **使用 TLS 验证信息传送**

  a) 在上面创建的客户端目录下创建一个名为 "mqclient.ini 的客户端 INI 文件，例如 "mq-client。在 "为客户端和队列管理器之间的消息传输配置 TLS"步骤 1 中，指定用于 p12 密钥存储的密码：

```
Channels:
  ChannelDefinitionDirectory=.
  ChannelDefinitionFile=CCDT.json
SSL:
  OutboundSNI=HOSTNAME
  SSLKeyRepository=example-app1.p12
  SSLKeyRepositoryPassword=PASSWORD
```

  b) 在同一目录下创建一个 CCDT 文件，文件名应与之前定义的 "mqclient.ini 文件中的引用相匹配，例如 "CCDT.json

```
{
    "channel":
    [
        {
            "name": "MTLS.SVRCONN",
            "clientConnection":
            {
                "connection":
                [
                    {
                        "host": "localhost",
                        "port": 1414
                    }
                ],
                "queueManager": "QM1"
            },
            "transmissionSecurity":
            {
              "cipherSpecification": "ANY_TLS13",
              "certificateLabel": "example-app1"
            },
            "type": "clientConnection"
        }
   ]
}
```

  c) 在 Mac 上，运行以下命令：

```
export MQSSLTRUSTSTORE=example-app1-chain.crt
```

  d) 在同一目录下，例如 "mq-client，使用示例应用程序输入一条信息：

```
amqsputc EXAMPLE.QUEUE QM1
```

  e) 使用示例应用程序获取信息：

```
amqsgetc EXAMPLE.QUEUE QM1
```

- **验证 "IBM MQ Console**

  在浏览器中连接控制台：

a) 开放 https://localhost:9443/ibmmq/console

b) 接受自签名证书。

c) 登录：

- 用户：admin
- 密码：您之前为 *ADMIN_PASSWORD* 指定的值。

- **验证 "REST API**

  在以下示例中，请指定先前为 *ADMIN_PASSWORD* 和 *APP_PASSWORD* 配置的值。

  调用 "administrative REST API 获取队列管理器状态的示例：

```
curl -k -X GET -u admin:ADMIN_PASSWORD https://localhost:9443/ibmmq/rest/v1/admin/qmgr/QM1
```

  调用消息 "REST API 从队列中发送和接收消息的示例。

  放置消息：

```
curl -k -X POST -u app1:APP_PASSWORD \
-H 'Content-Type: application/json' -H "ibm-mq-rest-csrf-token: value" \
-d 'msg from REST' https://localhost:9443/ibmmq/rest/v2/messaging/qmgr/QM1/queue/
EXAMPLE.QUEUE/message
```

  浏览留言：

```
curl -k -X GET -u app1:APP_PASSWORD \
-H "ibm-mq-rest-csrf-token: value" https://localhost:9443/ibmmq/rest/v2/messaging/qmgr/QM1/
queue/EXAMPLE.QUEUE/message
```

  注: REST API"**GET**"操作实际上是 "IBM MQ 浏览操作--信息被留在队列中。 要执行 "IBM MQ""**GET**，请使用动词 "REST API **DELETE**。

故障诊断

- 如果 "**amqputc** 或 "**amqgetc** 示例应用程序失败，请查看客户机（例如笔记本电脑）上 "~/IBM/MQ/data/errors/AMQERR01.LOG 处的错误日志。
- 通过执行 "Podman 中的容器，在其常规位置查看队列管理器的错误日志。

  例如：

```
podman exec -it QM1 /bin/bash
```

OpenShift CD CP4I-SC2 迁移到 IBM MQ Operator

此组主题描述了使用 Red Hat OpenShift Container Platform 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 170 页的『正在检查必需的功能是否可用』
2. 第 170 页的『抽取队列管理器配置』
3. 可选： 第 171 页的『可选: 抽取和获取队列管理器密钥和证书』
4. 可选： 第 173 页的『可选: 配置 LDAP』
5. 可选： 第 179 页的『可选: 更改 IBM MQ 配置中的 IP 地址和主机名』
6. 第 180 页的『更新容器环境的队列管理器配置』
7. 第 183 页的『为在容器中运行的 IBM MQ 选择目标 HA 体系结构』
8. 第 184 页的『为队列管理器创建资源』
9. 第 185 页的『在 Red Hat OpenShift 上创建新的队列管理器』
10. 第 189 页的『验证新的容器部署』

OpenShift CD CP4I-SC2 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 169 页的『迁移到 IBM MQ Operator』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 26 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 139 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 36 页的『规划容器中 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。

OpenShift CD CP4I-SC2 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。

OpenShift CD CP4I-SC2 可选: 抽取和获取队列管理器密钥和证书

可以将 IBM MQ 配置为使用 TLS 对进入队列管理器的网络流量进行加密。 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 TLS 通信，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。

注: 不支持具有与颁发者 (CA) 证书相同的主题专有名称 (DN) 的队列管理器证书。 证书必须具有唯一的主题专有名称。 产品将检查 DN 是否不相同。

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 runmqakmResult < <(runmqakm -cert -list -db ${keyrlocation}.kdb -stashed)
        cert=1
        for i in "${runmqakmResult[@]:2}"
        do
           certlabel=$(echo ${i:2} | xargs)
           echo Extracting certificate $certlabel to $cert.cert
           runmqakm -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. 标识队列管理器的 TLS 密钥存储库的位置:

     ```
     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:

     ```
     runmqakm -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:

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

     在先前显示的样本中，这等同于以下命令:

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

OpenShift CD CP4I-SC2 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。

OpenShift CD CP4I-SC2 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 173 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:

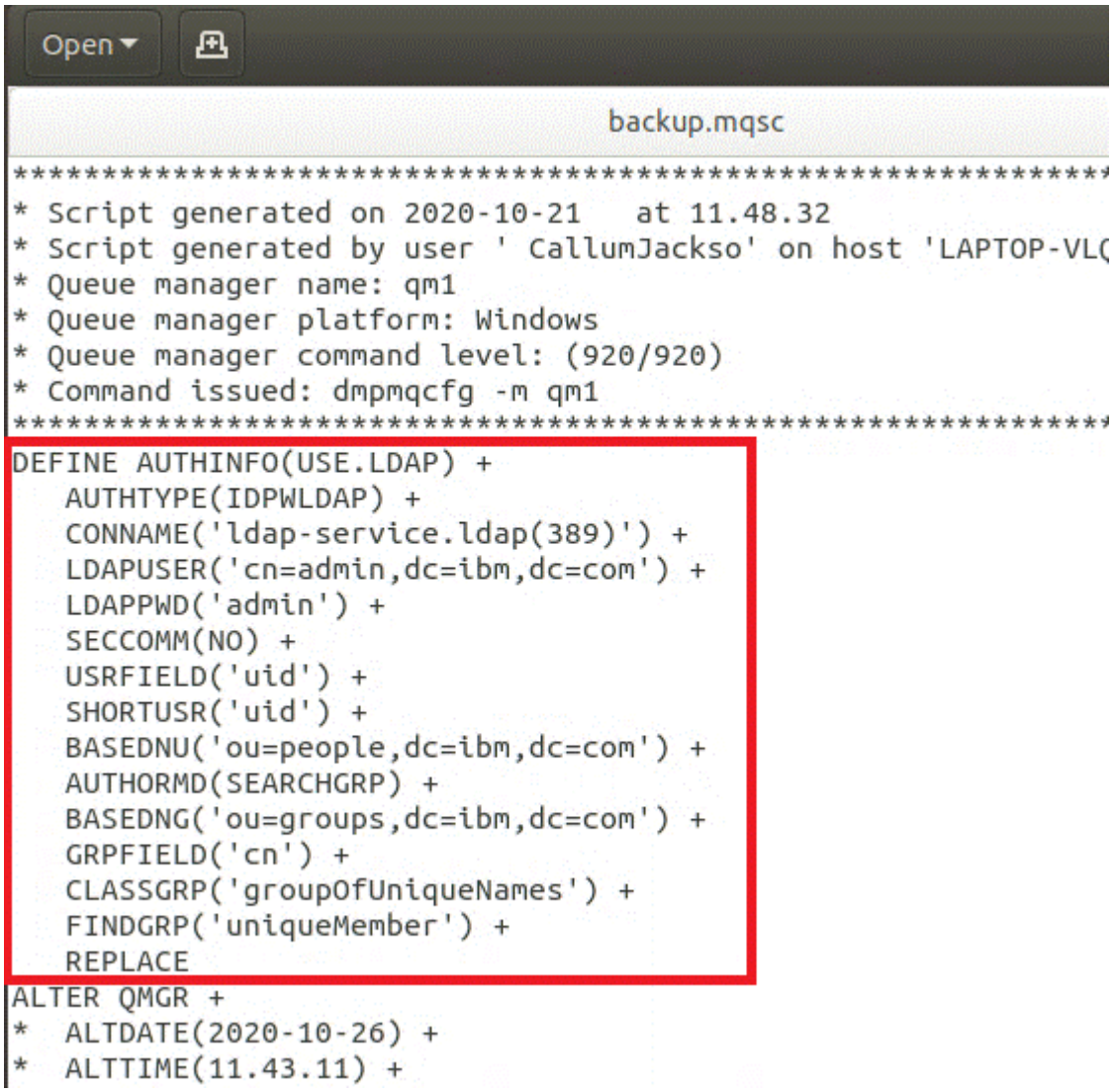
   ```
   DEFINE AUTHINFO(USE.LDAP) +
      AUTHTYPE(IDPWLDAP) +
      CONNAME('ldap-service.ldap(389)') +
      LDAPUSER('cn=admin,dc=ibm,dc=com') +
      LDAPPWD('admin') +
      SECCOMM(NO) +
      USRFIELD('uid') +
      SHORTUSR('uid') +
      BASEDNU('ou=people,dc=ibm,dc=com') +
      AUTHORMD(SEARCHGRP)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

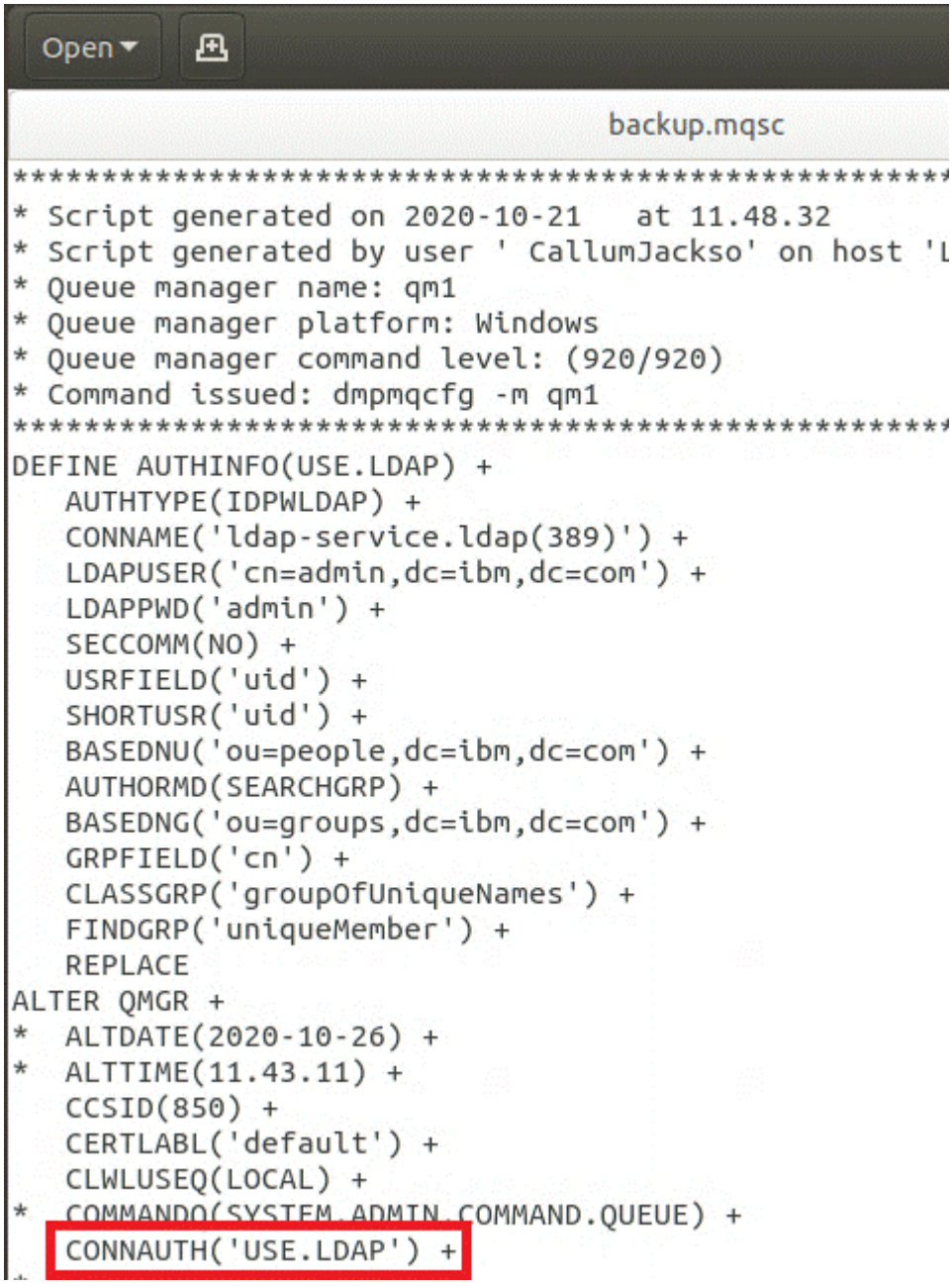
- **USRFIELD** 指定 LDAP 记录中要与提供的用户名匹配的字段。
- **SHORTUSR** 是 LDAP 记录中长度不超过 12 个字符的字段。 如果认证成功，那么此字段中的值为已断言的身份。
- **BASEDNU** 是应该用于搜索 LDAP 的基本 DN。
- **BASEDNG** 是 LDAP 中组的基本 DN。
- **AUTHORMD** 定义用于解析用户组成员资格的机制。 有四个选项:
  - OS: 查询操作系统以查找与短名称关联的组。
  - SEARCHGRP: 在 LDAP 中搜索已认证用户的组条目。
  - SEARCHUSR: 搜索已认证的用户记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:



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'):



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。

OpenShift CD CP4I-SC2

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 173 页的『可选: 配置 LDAP』。

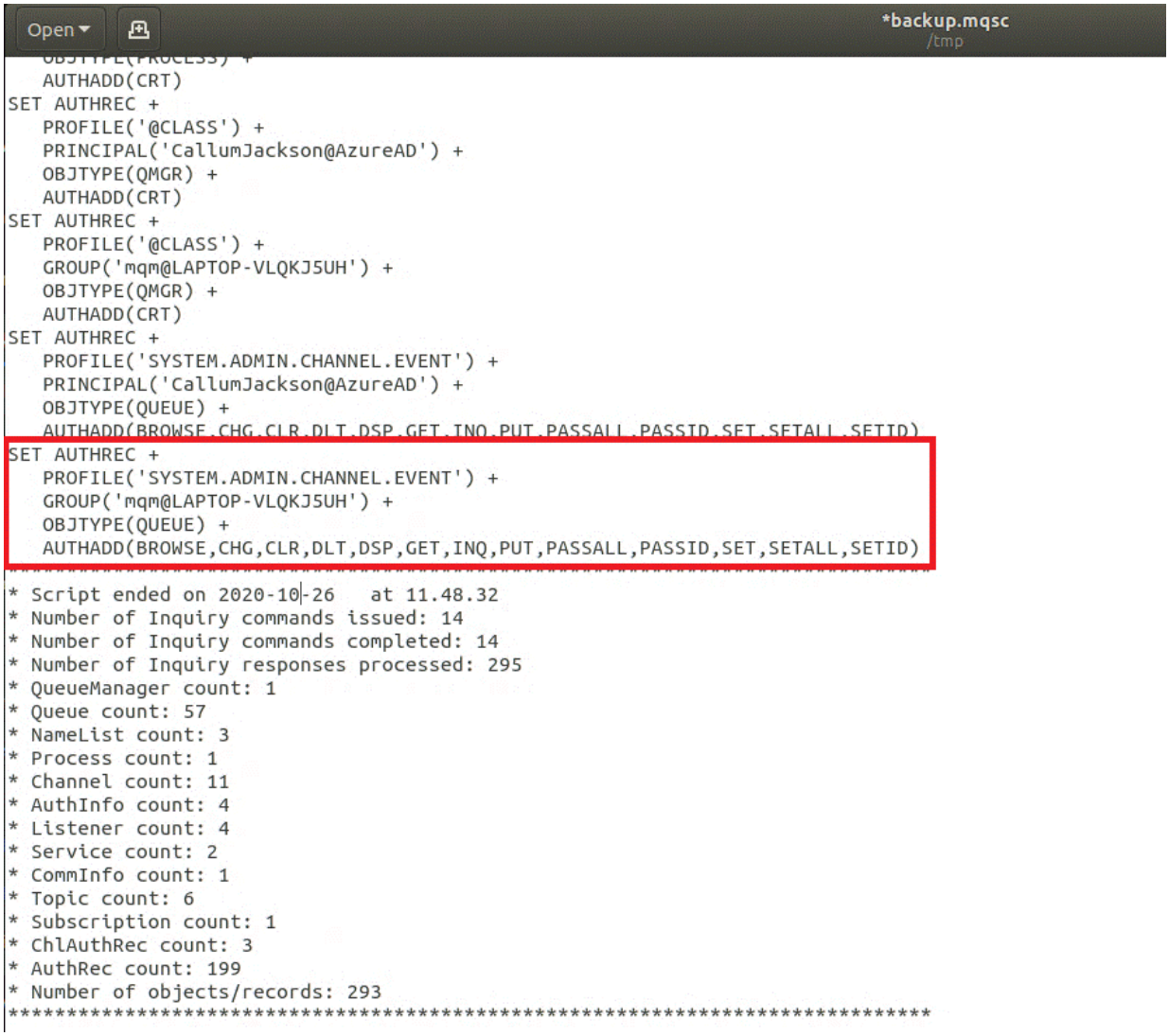
更新 LDAP 授权信息有两个部分:

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:



   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。

OpenShift CD CP4I-SC2 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。

OpenShift CD CP4I-SC2 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。

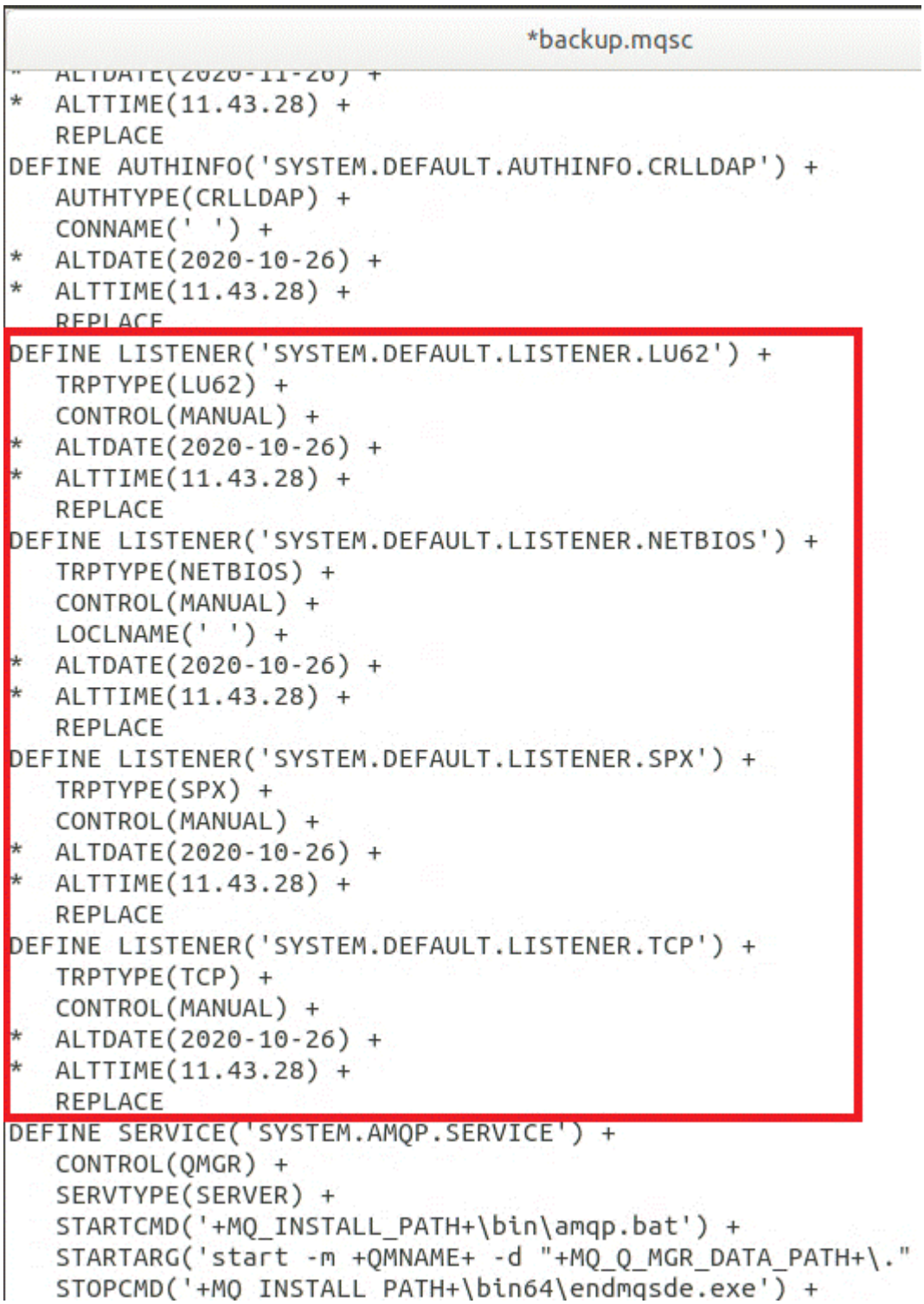
因此，您需要更新导出的配置:

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。



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。

OpenShift CD CP4I-SC2 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)，多实例 (两个 Pod) 和本机 HA (一个活动副本 Pod 和两个备用副本 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 提供了三个高可用性选项:

- **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。
- **本机 HA**: 三个容器 (每个容器位于单独的 Pod 中)，每个容器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)。 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。

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 185 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. 查看三个选项。

   有关这些选项的全面描述，请参阅 第 36 页的『规划容器中 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。

OpenShift CD CP4I-SC2 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 171 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。

OpenShift CD CP4I-SC2 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 183 页的『为在容器中运行的 IBM MQ 选择目标 HA 体系结构』中所述，有三种可能的部署拓扑。 因此，本主题提供了三个不同的模板:

- 模板 1: 部署单个实例队列管理器。
- 模板 2: 部署多实例队列管理器。
- 模板 3: 部署本机 HA 队列管理器。

要点: 根据您的首选拓扑，仅完成三个模板中的一个。

过程

- **模板 1: 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:

  注: 本示例使用 IBM MQ Advanced 许可证，但也可以使用 IBM MQ 许可证。 有关更多信息，请参阅第 140 页的『使用 IBM MQ 许可证注释配置队列管理器。 IBM MQ Operator』。

  ```
  apiVersion: mq.ibm.com/v1beta1
  kind: QueueManager
  ```

```
metadata:
  name: qm1
spec:
  version: 9.4.3.0-r1
  license:
    accept: true
    license: L-CYPF-CRPF3H
    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: 9.4.3.0-r1
  license:
    accept: true
    license: L-CYPF-CRPF3H
    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 219 页的『mq.ibm.com/v1beta1: 当前许可证版本』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。

- **模板 2: 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。

  注: 本示例使用 IBM MQ Advanced 许可证，但也可以使用 IBM MQ 许可证。 有关更多信息，请参阅第 140 页的『使用 IBM MQ 许可证注释配置队列管理器。 IBM MQ Operator』。

```
apiVersion: mq.ibm.com/v1beta1
kind: QueueManager
metadata:
  name: qm1mi
spec:
  version: 9.4.3.0-r1
  license:
    accept: true
    license: L-CYPF-CRPF3H
    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。

- **模板 3: 部署本机 HA 队列管理器**。
  有关创建本机 HA 队列管理器的工作示例，请参阅 第 98 页的『示例: 使用 IBM MQ Operator 配置本机 HA』。

OpenShift CD CP4I-SC2 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 185 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 169 页的『迁移到 IBM MQ Operator』。

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 129 页的『配置路由以从 Red Hat OpenShift 集群外部连接到队列管理器』。

在容器中操作 IBM MQ

如果需要与在容器中运行的 IBM MQ 队列管理器进行操作或交互，请参阅以下主题以获取更多信息。

过程

- 第 190 页的『使用 IBM MQ Operator 操作 IBM MQ』.
- 第 203 页的『查看本机 HA 队列管理器的状态』.
- 第 211 页的『结束本地 HA 队列管理器』.

OpenShift CP4I 使用 IBM MQ Operator 操作 IBM MQ

过程

- 第 191 页的『连接到 Red Hat OpenShift 集群中部署的 IBM MQ Console』.
- 第 191 页的『使用 IBM MQ Operator 时进行监视』.
- 第 202 页的『使用 Red Hat OpenShift CLI 备份和复原队列管理器配置』.

OpenShift CP4I 为 IBM MQ Console 授予许可权

根据您的许可证使用情况，将以不同方式管理 IBM MQ Console 的许可权。

关于此任务

根据队列管理器使用的许可证选择相关选项。

过程

- **方案 1：** 通过 Keycloak 配置用户和角色。

  如果您正在使用 IBM Cloud Pak for Integration 许可证，那么 IBM MQ Console 会将 Keycloak 用于身份和访问权管理。

  - 有关配置 Keycloak 的信息，请参阅 IBM Cloud Pak for Integration 文档中的身份和访问管理。
  - 有关 *MQ* 角色及其相应 Keycloak 角色的信息，请参阅 IBM Cloud Pak for Integration 文档中的 Cloud Pak for Integration 角色和权限。 可以为整个安装配置角色，也可以为每个实例（队列管理器）配置角色。 *MQ* 角色对应项列在**队列管理器**下。
  - 如果您之前在旧版本的 IBM MQ Operator 上使用 IAM 配置了用户，请参阅 IBM Cloud Pak for Integration 文档中的将用户从 IAM 迁移到 Keycloak。
- **方案 2：** 手动配置用户和角色。

  如果您使用的是 IBM MQ 许可证，则 IBM MQ Console 没有预先配置，您需要自行配置。

  - 有关用户和角色的信息，请参阅配置用户和角色。
  - 要获取简单的示例，请参阅 第 145 页的『使用 IBM MQ Operator 使用基本注册表配置 IBM MQ Console』。
  - 另外，如果将队列管理器更改为使用 IBM Cloud Pak for Integration 许可证，则可以按照前面所述配置 Keycloak。

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 。

开始之前

您需要获得许可才能访问 IBM MQ Console。 请参阅 第 190 页的『为 IBM MQ Console 授予许可权』。

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 UI 中找到 IBM MQ Console URL。 另外，也可以通过 Red Hat OpenShiftCLI 运行以下命令来查找：

```
oc get queuemanager QueueManager Name -n namespace_of_your_MQ_deployment --output
jsonpath='{.status.adminUiUrl}'
```

相关任务

第 129 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 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。 队列管理器指标已启用，并通过 HTTP 提供。

开始之前

使用 HTTPS 进行队列管理器指标

队列管理员可以使用创建的 OpenShift 证书通过 HTTPS 从 9.4.2.0-r1 获取指标。 在 9.4.2.0-r1 队列之前，管理人员只能通过 HTTP 提供指标。

注: 所有队列管理器默认通过 HTTP 提供指标。

关于此任务

Prometheus 是一个开源的时间序列数据库和度量规则评估引擎。 IBM MQ 容器暴露了一个指标端点，可通过 Prometheus 进行查询。 这些指标由用于监控和活动追踪的 IBM MQ Operator 系统主题生成。

OpenShift Container Platform 包括一个预先配置、预先安装且可自我更新的监控堆栈，该堆栈使用 服务器。 Prometheus 需要配置 OpenShift Container Platform 监视堆栈以监视用户定义的项目。 请参阅 “启用用户自定义项目的监控 ”。

当您创建启用指标的 QueueManager 时， IBM MQ Operator 会创建一个 ServiceMonitor ，Prometheus 操作员可以发现该。

过程

- 使用 Red Hat OpenShift Container Platform (OCP)监控堆栈查看队列管理器指标。
  a) 在 OCP 中打开 **“指标”选项卡**。
  b) 单击 **“观察 > 指标”**。
- 通过以下方式提供队列管理指标 HTTPS

  请将 **.spec.queueManager.metrics.tls.provider** 设置为 openshift。

```
spec:
  queueManager:
```

```
    metrics:
      enabled: true
      tls:
        provider: openshift
```

注:

使用此设置可创建一个 OpenShift 证书，供队列管理器的 HTTPS 指标服务器使用。 IBM MQ Operator 无法配置其他供应商。

- 禁用队列管理器指标。

  将 **.spec.queueManager.metrics.enabled** 设置为无效。

相关任务

第 217 页的『Troubleshooting container clusters with the OpenShift Monitoring CLI』

As a troubleshooting aid, you can query the available container cluster metrics using the OpenShift Monitoring CLI.

OpenShift CP4I ***Monitoring container clusters with the OpenShift Monitoring console***

You can query the available container cluster metrics using the OpenShift Monitoring web console.

Before you begin

OpenShift Monitoring provides a web console and a CLI. For day-to-day operations you might prefer to use the console, as described in this task. For gathering troubleshooting information you might prefer to use the CLI, as described in “Troubleshooting container clusters with the OpenShift Monitoring CLI” on page 217.

About this task

OpenShift Container Platform includes a pre-configured, pre-installed and self-updating monitoring stack that uses an open source Prometheus server. Queue managers running under an IBM MQ Operator can produce metrics compatible with Prometheus. For a complete list of the metrics available for IBM MQ, see “IBM MQ 容器发布的指标” on page 194.

OpenShift Monitoring is the default cluster monitoring method. It is installed with the cluster. It provides Prometheus, Thanos Querier, and a Grafana dashboard that shows queue manager metrics.

To query the available container cluster metrics using the OpenShift Monitoring console, complete the following steps.

Procedure

1. Enable OpenShift Monitoring for the operands deployed in an IBM MQ container cluster.
   a) Log in to the cluster as a cluster administrator.
   b) Add "`enableUserWorkload: true`" to the configMap `cluster-monitoring-config` in the `openshift-monitoring` namespace. To do this, follow the steps in Enabling monitoring for user-defined projects in the Red Hat OpenShift documentation. If the `cluster-monitoring-config` ConfigMap does not exist in the `openshift-monitoring` namespace, follow the instructions to create it.
2. Query the IBM MQ metrics using the OpenShift Monitoring console.

   Use the static Grafana dashboard, which presents cluster metrics to cluster administrators.

   Query IBM MQ metrics, and examine the data plotted on a chart, as described in Querying metrics in the Red Hat OpenShift documentation.

   For example, if you query the metric **ibmmq_qmgr_cpu_load_fifteen_minute_average_percentage** you get a chart similar to the following screen capture:

This screen capture also shows the web console prompt. You can run PromQL queries from this prompt. For example:

Get a list of queue managers that have a positive FDC count:

```
ibmmq_qmgr_fdc_files > 0
```

Pass the namespace filter in the query to get the results for a specific namespace:

```
ibmmq_qmgr_fdc_files{namespace='your-namespace'} > 0
```

Return the count of all such queue managers:

```
count(ibmmq_qmgr_fdc_files{namespace='your-namespace'} > 0)
```

Get the Write latency of queue manager pods over the last 5 hours:

```
ibmmq_qmgr_log_write_latency_seconds{namespace='your-namespace'}[5h]
```

Get the RAM usage estimate in bytes for a queue manager for a particular namespace summed by "Job":

```
sum by (job)(ibmmq_qmgr_ram_usage_estimate_for_queue_manager_bytes{qmgr='your-queue-manager', namespace='your-namespace'})
```

The previous queries illustrate use of filters "namespace", "job" and "qmgr". For this metric you can also filter on "container", "endpoint", "instance", "pod", and "prometheus". Note that the available filters are the column headings shown in the screen capture, except for “Name” which is the name of the metric itself.

Related tasks

“Troubleshooting container clusters with the OpenShift Monitoring CLI” on page 217
As a troubleshooting aid, you can query the available container cluster metrics using the OpenShift Monitoring CLI.

Thanos Querier Versus Thanos Querier

Related reference

“IBM MQ 容器发布的指标” on page 194
队列管理器容器可以发布与 Red Hat OpenShift Monitoring 兼容的度量。

IBM MQ 容器发布的指标

队列管理器容器可以发布与 Red Hat OpenShift Monitoring 兼容的度量。

指标	类型	描述
ibmmq_nha_recovery_average_network_round_trip_time_seconds	gauge	网络往返平均时间
ibmmq_nha_recovery_backlog_average_bytes	gauge	积压平均字节数
ibmmq_nha_recovery_backlog_bytes	gauge	积压字节数
ibmmq_nha_recovery_compressed_log_sent_bytes	gauge	已发送的压缩日志字节数
ibmmq_nha_recovery_log_data_average_compression_time_seconds	gauge	日志数据平均压缩时间
ibmmq_nha_recovery_log_data_average_decompression_time_seconds	gauge	日志数据平均解压时间
ibmmq_nha_recovery_log_decompressed_bytes	gauge	已解压的日志字节数
ibmmq_nha_recovery_log_sent_bytes	counter	已发送的日志字节数
ibmmq_nha_recovery_rebase_count	gauge	重建计数
ibmmq_nha_recovery_recovery_log_sequence_number	gauge	恢复日志序号
ibmmq_nha_replication_acknowledged_log_sequence_number_total	gauge	已确认的日志序号
ibmmq_nha_replication_average_network_round_trip_time_seconds	gauge	网络往返平均时间
ibmmq_nha_replication_backlog_average_bytes	gauge	积压平均字节数
ibmmq_nha_replication_backlog_bytes	gauge	积压字节数

指标	类型	描述
ibmmq_nha_replication_catch_up_compressed_log_sent_bytes	counter	已发送的同步复制压缩日志字节数
ibmmq_nha_replication_catch_up_log_data_average_compression_time_seconds	gauge	同步复制日志数据平均压缩时间
ibmmq_nha_replication_catch_up_log_data_average_decompression_time_seconds	gauge	同步复制日志数据平均解压时间
ibmmq_nha_replication_catch_up_log_decompressed_bytes	gauge	解压的同步复制日志字节数
ibmmq_nha_replication_catch_up_uncompressed_log_sent_bytes	counter	已发送的同步复制未压缩日志字节数
ibmmq_nha_replication_catchup_log_sent_bytes	counter	已发送的同步复制日志字节数
ibmmq_nha_replication_log_file_system_free_space_percent	gauge	日志文件系统 - 可用空间
ibmmq_nha_replication_log_file_system_in_use_bytes	gauge	日志文件系统 - 使用的字节数
ibmmq_nha_replication_log_write_average_acknowledgement_latency_seconds	gauge	日志写入平均确认等待时间
ibmmq_nha_replication_log_write_average_acknowledgement_size_bytes	gauge	日志写入平均确认大小
ibmmq_nha_replication_mq_fdc_file_count	gauge	MQ FDC 文件数
ibmmq_nha_replication_queue_manager_file_system_free_space_percent	gauge	队列管理器文件系统 - 可用空间

指标	类型	描述
ibmmq_nha_replication_queue_manager_file_system_in_use_bytes	gauge	队列管理器文件系统 - 使用的字节数
ibmmq_nha_replication_synchronous_compressed_log_sent_bytes	counter	已发送的同步压缩日志字节数
ibmmq_nha_replication_synchronous_log_data_average_compression_time_seconds	gauge	同步日志数据平均压缩时间
ibmmq_nha_replication_synchronous_log_data_average_decompression_time_seconds	gauge	同步日志数据平均解压时间
ibmmq_nha_replication_synchronous_log_decompressed_bytes	gauge	解压的同步日志字节数
ibmmq_nha_replication_synchronous_log_sent_bytes	counter	已发送的同步日志字节数
ibmmq_nha_replication_synchronous_uncompressed_log_sent_bytes	counter	已发送的同步未压缩日志字节数
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

指标	类型	描述
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
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

指标	类型	描述
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
ibmmq_qmgr_fdc_files	gauge	MQ FDC 文件数
ibmmq_qmgr_log_file_system_free_space_percentage	gauge	日志文件系统 - 可用空间
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_occupied_by_extents_waiting_to_be_archived_bytes	gauge	日志——等待存档的扩展名
ibmmq_qmgr_log_occupied_by_reusable_extents_bytes	gauge	日志 - 可复用扩展数据块占用的字节数
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_required_for_media_recovery_bytes	gauge	日志 - 介质恢复所需的字节数
ibmmq_qmgr_log_sequence_number_disk_total	gauge	日志 - 写入的磁盘日志序号
ibmmq_qmgr_log_sequence_number_quorum_total	gauge	日志 - 定额日志序号

指标	类型	描述
ibmmq_qmgr_log_slowest_write_since_restart_seconds	gauge	日志 - 自重启后写入速度最慢
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
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

指标	类型	描述
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
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

指标	类型	描述
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
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

相关信息

在系统主题上发布的度量

OpenShift V 9.4.0 V 9.4.0 停止队列管理器 (mq.ibm.com/stop)

通过向队列管理器定义添加注释来停止队列管理器。

关于此任务

由 IBM MQ 操作程序创建的队列管理器具有关联的 StatefulSet。 此 StatefulSet 声明要通过 ".replicas" 字段为给定队列管理器可用性类型部署的 Pods 数。 这将采用 1 (单实例) ， 2 (多实例) 或 3 (NativeHA) 的值。

注: 手动更改 ".replicas" 字段中的值会导致队列管理器无法正常工作。

在某些情况下，您可能想要停止队列管理器，以便 StatefulSet 具有副本计数 0 ，并且不会部署任何 Pods 。 您可能希望执行此操作的示例包括在维护期间或备份过程中。

注: 由于在队列管理器停止时未部署任何队列管理器 Pods ，因此在再次启动队列管理器之前，您和您的应用程序将无法访问该队列管理器。

过程

- 要停止队列管理器，请将以下注释添加到 ".metadata.annotations" 部分下的队列管理器定义中。

```
apiVersion: mq.ibm.com/v1beta1
kind: QueueManager
metadata:
  name: my-qm
  annotations:
    "mq.ibm.com/stop" : "true"
```

- 要重新启动队列管理器并将其返回到正确数量的副本，请从队列管理器中除去注释或将其值设置为 "false'"。

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 **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
```

MQ Adv. 查看本机 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确定的，副本实例提供的信息可能已经过时。

V 9.4.2 对于原生 HA 跨区域复制 (CRR) 配置，提供的信息还取决于实例是实时组还是恢复组的一部分。 实时小组的信息更为准确，但由于恢复小组仅定期发送信息，因此仍可能过时。

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。
- V 9.4.2 查看实例所属或连接的群组状态。

以下状态字段用于报告队列管理器实例的 Native HA 和 Native HA 跨区域复制 (CRR)配置状态。 这些字段始终会出现:

职能部门
指定实例的当前角色。 是 Active、 Replica、 Unknown、 （用于康复小组） Leader 或 Not Configured 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。

V 9.4.2 **GRPLSN**
指定在 HA 组中具有法定人数的实例之间复制的日志序列号（LSN），采用 *nnnnn:nnnnn:nnnnn:nnnnn* 格式。 如果不知道 LSN，则为空。

V 9.4.2 **GRPNAME**
指定 HA 组的名称。 如果未配置名称，则为空。

V 9.4.2 **GRPROLE**
指定 HA 组的当前角色，可以是其中之一:

- Live - 现场乐队
- Recovery - 一个康复小组
- Pending live - 一群等待成为 Live

- Pending recovery - 一群等待成为 Recovery
- Not configured - 本地 HA 尚未配置
- Unknown - 当前角色无法确定

使用 -x 参数请求所有实例的扩展信息时，这些字段就会出现：

REPLADDR
队列管理器实例的复制地址。

连接 ACTV
指示实例是否连接到活动实例。

BACKLOG
表示实例落后于活动实例的 KB 数。

连接
指示指定的实例是否已连接到此实例。

ALTDATE
显示此信息最后更新的日期。 如果从未更新，则留空。

ALTTIME
表示该信息最后一次更新的时间。 如果从未更新，则留空。

V 9.4.2 **ACKLSN**
表示实例已确认写入恢复日志的 LSN，格式为 *nnnnn:nnnnn:nnnnn:nnnnn* 。 如果不知道 LSN 值，则为空。

V 9.4.2 **HAStatus**

表示该实例的运行状态，为以下值之一：

- Normal - 实例正在正常运行。
- Checking - 正在检查该实例，以确保恢复日志的一致性。
- Synchronizing - 实例正在接收数据，使其保持同步。
- Rebasing - 新的恢复日志正在发送到实例。
- Disk full - 实例磁盘已满。
- Disconnected - 无法与实例通信。
- Unknown - 实例状态未知。

V 9.4.2 **同步时间**
以 ISO 8601 格式表示此实例与活动队列管理器保持同步的最后时间。 如果时间未知，则留空。

V 9.4.2 使用 -g 参数请求组信息时，这些字段就会出现。 当组属于原生高可用性 CRR 配置时，这些字段可用于查看状态：

V 9.4.2 **GRPNAME**
指定 HA 组的名称。 如果未配置名称，则为空。

V 9.4.2 **GRPROLE**
指定 HA 组的当前角色，可以是其中之一：

- Live - 现场乐队
- Recovery - 一个康复小组
- Pending live - 一群等待成为 Live
- Pending recovery - 一群等待成为 Recovery
- Unknown - 当前角色无法确定
- Not configured - 本地 HA 尚未配置

V 9.4.2 **GRPADDR**

用于连接到群组的 IP 地址。 如果因为尚未与群组建立连接而不知道地址，请发送邮件至 Unknown 。

V 9.4.2 **GRPVER**

以 V.R.M.F 格式表示当前组长的版本，其中 V.R.M.F 是版本号、发行号、修改号和补丁包号。 如果不知道数值，则将其设置为 ?.?.?.?

V 9.4.2 **CONNGRP**

表示该组是否与本地组相连，并且是其中之一：

- yes - 小组已连接。
- no - 该组已断开连接。
- unknown - 连接状态未知。
- suspended - 组是连接的，但在组之间的配置不兼容问题得到解决之前，组无法复制数据。

此字段仅对本地组以外的组显示。

V 9.4.2 **INSYNC**

此功能仅对具有角色 Recovery 或 Pending live 的群组显示。 它表明在发生故障切换时，该组是否能够 Live 且不丢失数据。 （请注意， Recovery 组可能不会经常与 Live 组同步，因为日志是异步复制的。 使用 RCOVLSN 和 RCOVTIME 值来指示在计划外故障切换时是否可能出现数据丢失。

V 9.4.2 **同步时间**

此功能仅对具有角色 Recovery 或 Pending live 的群组显示。 它以 ISO 8601 格式（如果不知道，则留空）表示该组与 Live 组上次同步的时间。 （请注意， Recovery 组可能不会经常与 Live 组同步，因为日志是异步复制的。 使用 RCOVLSN 和 RCOVTIME 值来指示在计划外故障切换时是否可能出现数据丢失。

V 9.4.2 **BACKLOG**

此功能仅对具有角色 Recovery 或 Pending live 的群组显示。 它表示该组落后于实时组多少 KB。

V 9.4.2 **GRSTATUS**

表示该组的运行状态，为以下值之一：

- Normal - 该小组正在正常运作。
- Checking - 正在检查该组，以确保恢复日志的一致性。
- Synchronizing - 正在向该组发送数据，使其保持同步。
- Rebasing - 新的恢复日志正在发送到实例。
- Waiting for connection - 该群正在等待一个来自 Live 或 Pending Recovery 角色的群。
- Partitioned - 有两个小组认为自己属于“现场”小组（即“大脑分裂”状态）。
- Unknown - 该群状态未知。

V 9.4.2 **RCOVLSN**

组可恢复的日志序列号 (LSN)，格式为 *nnnnn:nnnnn:nnnnn:nnnnn*。 如果发生计划外的故障切换，此后的任何数据都可能丢失。 该值可输入 **dmpmqlog** 命令的 -s 选项。

V 9.4.2 **RCOVTIME**

小组可以恢复的时间，以 ISO 8601 格式表示。 如果发生意外故障切换，在此时间之后写入恢复日志的任何数据都可能丢失。 该值可以传递给 **dmpmqlog** 命令的 -t 或 -u 选项。

V 9.4.2 **INITLSN**

此功能仅对具有角色 Live 或 Pending recovery 的群组显示。 它表示本地高可用性组最初启用时恢复的最后一个日志记录的日志序列号（LSN），格式为 *nnnnn:nnnnn:nnnnn:nnnnn* ，如果该组未启用，则为空。

V 9.4.2 **initTime**

此功能仅对具有角色 Live 或 Pending recovery 的群组显示。 它以 ISO 8601 格式表示本地高可用性组最初启用时恢复的最后一个日志记录的时间。 空白，如果该组不是实时组。

V 9.4.2 **生活时间**
此功能仅对具有角色 Live 或 Pending recovery 的群组显示。 它以 ISO 8601 格式表示该组合首次成为现场组合的时间。 空白，如果该组不是实时组。

V 9.4.2 **ALTDATE**
显示此信息最后更新的日期。

V 9.4.2 **ALTTIME**
表示该信息最后一次更新的时间。

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)
```

V 9.4.2 在本地 HA CRR 配置中作为恢复组组长的 BOB 实例将报告以下状态:

```
QMNAME(BOB)                STATUS(Recovery group leader)
```

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)
GRPLSN(<0:25:365:1000>) GRPNAME() GRPROLE(Live)
```

V 9.4.2

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)
GRPLSN(<0:25:365:1000>) GRPNAME() GRPROLE(Live)
```

V 9.4.2

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)
GRPLSN() GRPNAME() GRPROLE(Live)
```

V 9.4.2 在本地 HA CRR 配置中，属于实时组的名为 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)
GRPLSN(<0:25:365:1000>) GRPNAME(alpha) GRPROLE(Live)
```

V 9.4.2 在组等待成为恢复组的同一实例中，可能会报告以下状态：

```
QMNAME(BOB)         ROLE(Active) INSTANCE(inst1) INSYNC(yes) QUORUM(3/3)
GRPLSN(<0:25:365:1000>) GRPNAME(alpha) GRPROLE(Pending recovery)
```

V 9.4.2 名为 CAROL 的队列管理器不是本地 HA 或本地 HA CRR 配置的实例，将报告以下状态：

```
QMNAME(CAROL)       ROLE(Not configured) INSTANCE() INSYNC() QUORUM() GRPLSN() GRPNAME()
GRPROLE(Not configured)
```

- 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
```

例如，如果您在运行队列管理器 BOB 活跃实例的节点上发出此命令，而该节点上的所有实例都是同步的，则可能会收到以下状态：

```
QMNAME(BOB)         ROLE(Active) INSTANCE(inst1) INSYNC(yes) QUORUM(2/3)
GRPLSN(<0:25:365:1000>) GRPROLE(Live) GRPNAME()
 INSTANCE(inst1) ROLE(Active)  REPLADDR(localhost) CONNACTV(yes) INSYNC(yes) BACKLOG(0)
CONNINST(yes) ACKLSN(<0:26:365:1000>)  HASTATUS(Normal)
SYNCTIME(2023-12-20T17:13:27.853412Z) ALTDATE(2023-12-20) ALTTIME(17.13.27)
 INSTANCE(inst3) ROLE(Replica) REPLADDR(localhost) CONNACTV(yes) INSYNC(yes) BACKLOG(0)
CONNINST(yes) ACKLSN (<0:24:365:1000>) HASTATUS(Normal)
SYNCTIME(2023-12-20T17:13:27.853412Z) ALTDATE(2023-12-20) ALTTIME(17.13.27)
 INSTANCE(inst2) ROLE(Replica) REPLADDR(localhost) CONNACTV(yes) INSYNC(yes) BACKLOG(0)
CONNINST(yes) ACKLSN (<0:25:365:1000>) HASTATUS(Synchronizing)
SYNCTIME(2023-12-20T17:13:22.123321Z) ALTDATE(2023-12-20) ALTTIME(17.13.27)
```

再举一个例子，如果您在运行队列管理器 BOB 副本实例的节点上发出此命令，而该副本实例滞后，则可能会收到以下状态：

```
QMNAME(BOB)         ROLE(Replica) INSTANCE(inst2) INSYNC(no) QUORUM(2/3)
GRPLSN(<0:25:365:1000>) GRPROLE(Live) GRPNAME()
 INSTANCE(inst1) ROLE(Active)  REPLADDR(localhost) CONNACTV(yes) INSYNC(yes) BACKLOG(0)
CONNINST(yes) ACKLSN(<0:26:365:1000>)  HASTATUS(Normal)
SYNCTIME(2023-12-20T17:13:27.853412Z) ALTDATE(2023-12-20) ALTTIME(17.13.27)
 INSTANCE(inst3) ROLE(Replica) REPLADDR(localhost) CONNACTV(yes) INSYNC(yes) BACKLOG(0)
CONNINST(yes) ACKLSN (<0:24:365:1000>) HASTATUS(Normal)
SYNCTIME(2023-12-20T17:13:27.853412Z) ALTDATE(2023-12-20) ALTTIME(17.13.27)
 INSTANCE(inst2) ROLE(Replica) REPLADDR(localhost) CONNACTV(yes) INSYNC(no) BACKLOG(435)
CONNINST(yes) ACKLSN (<0:25:365:1000>) HASTATUS(Synchronizing)
SYNCTIME(2023-12-20T17:13:22.123321Z) ALTDATE(2023-12-20) ALTTIME(17.13.27)
```

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) GRPLSN()
GRPROLE() GRPNAME()
 INSTANCE(inst1) ROLE(Unknown) REPLADDR(9.20.123.45) CONNACTV(Unknown) INSYNC(Unknown)
BACKLOG(Unknown) CONNINST(No) ACKLSN()  HASTATUS(Unknown) SYNCTIME() ALTDATE() ALTTIME()
ALTDATE() ALTTIME()
 INSTANCE(inst2) ROLE(Unknown) REPLADDR(9.20.123.46) CONNACTV(Unknown) INSYNC(Unknown)
BACKLOG(Unknown) CONNINST(No) ACKLSN()  HASTATUS(Unknown) SYNCTIME() ALTDATE() ALTTIME()
ALTDATE() ALTTIME()
 INSTANCE(inst3) ROLE(Unknown) REPLADDR(9.20.123.47) CONNACTV(No)      INSYNC(Unknown)
BACKLOG(Unknown) CONNINST(No) ACKLSN()  HASTATUS(Unknown) SYNCTIME() ALTDATE() ALTTIME()
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)
```

- V 9.4.2

  确定本地 HA CRR 配置中所有本地 HA 组的运行状态：

  ```
  oc exec -t Pod dspmq -o nativeha -g -m QMgrName
  ```

  如果您对实时群组中的成员 alpha 发出此命令，则 alpha 和恢复群组 beta 可能会收到以下状态：

  ```
  GRPNAME(alpha) GRPROLE(Live) GRPADDR(9.20.135.129) GRPVER(09040200)
  GRSTATUS(Normal) RCOVLSN(<0:0:104:65163>) RCOVTIME(2023-12-20T17:37:27.114787Z)
  INITLSN(<0:0:101:352>) INITTIME(2023-12-19T09:13:27.512294)
  LIVETIME(2023-12-19T06:14:27.512294) ALTDATE(2023-12-20) ALTTIME(17.13.27)
  ```

  ```
  GRPNAME(beta) GRPROLE(Recovery) GRPADDR(9.20.135.130) GRPVER(09040300)
  CONNGRP(yes) GRSTATUS(Synchronizing) RCOVLSN(<0:0:19:16723>)
  RCOVTIME(2023-12-20T17:36:23.463209Z) BACKLOG(5619) INSYNC(no) SYNCTIME()
  ALTDATE(2023-12-20) ALTTIME(17.13.27)
  ```

相关参考

dspmq (显示队列管理器) 命令

V 9.4.2 本地 HA CRR 切换和故障切换

本地 HA 跨区域复制（CRR）配置使您能够将队列管理器的运行切换到其他区域。

您可以使用原生高可用性 CRR 配置来提供灾难恢复解决方案。 如果您的现场站点发生灾难，您可以执行故障切换，并在恢复站点继续运行队列管理器。 在这种情况下，您无法保证故障转移时恢复日志是同步的，因此可能会丢失一些数据。 同样，当故障站点恢复时，队列管理器可能会有两个活动实例，因此可能会累积一些分区（分裂脑）数据。

您可以使用原生 HA CRR 配置，以便在执行维护或测试时保持连续性。 在这种情况下，切换过程以不会丢失数据的方式进行控制。 同样，当您切换回来时，也没有需要解析的分区数据。

V 9.4.2 在原生高可用性 CRR 配置上完成计划的切换

在本地高可用性跨区域复制（CRR）配置中，您可以在实时组和恢复组之间进行计划切换。

关于此任务

如果您想测试配置或需要维护，可能需要从实时群组切换到恢复群组。

当您执行计划中的切换时，系统会采取措施确保您不会丢失数据或遇到分区（分割大脑）问题。 您可以通过更改每个群组中的群组角色来请求切换。 在恢复日志同步期间，这两个组会暂时处于待定状态。 当这一目标达成后，小组成员将承担新的角色，新的现场小组将选举出新的组长，组长将发挥积极作用，管理排队。

过程

1. 在 qm.ini 文件的 NativeHALocalInstance 节中将实时群组的 GroupRole 设置为 Recovery ， 使其成为恢复群组。

   如果您使用的是 MQ 操作员，则可以通过在 QueueManager YAML 中设置 .spec.queueManager.availability.nativeHAGroups.local.role 字段来更改 GroupRole 。 例如，要将实时群组更改为恢复群组，请将当前实时群组的 .spec.queueManager.availability.nativeHAGroups.local.role 字段值设置为 Recovery 。

   要验证切换是否成功，请使用 **oc get pods** 命令获取 pod 的状态，其中一个 pod 应处于就绪状态：

   ```
   NAME                 READY   STATUS    RESTARTS   AGE
   exampleqm-ibm-mq-0   1/1     Running   0          2m45s
   ```

```
exampleqm-ibm-mq-1   0/1     Running   0          2m45s
exampleqm-ibm-mq-2   0/1     Running   0          2m45s
```

发出以下命令检查恢复组的状态：

```
oc exec -t <leader-pod> -- dspmq -o nativeha -g
```

其中，<leader-pod> 是就绪 1/1 pod 的名称（示例中为 exampleqm-ibm-mq-0 ）：

检查 GRPROLE 是否待恢复。

2. 在 qm.ini 文件的 NativeHALocalInstance 节中将恢复组的 GroupRole 设置为 Live ，使其成为活动组。

   如果您使用的是 MQ 操作员，则可以通过在 QueueManager YAML 中设置 .spec.queueManager.availability.nativeHAGroups.local.role 字段来更改 GroupRole 。 例如，要将实时组更改为实时，请将当前恢复组的 .spec.queueManager.availability.nativeHAGroups.local.role 字段值设置为 Live 。

   如果在该组中的每个实例都重启后，您为该队列管理器发出 **dspmq -o nativeha -g** 命令，您将看到 GRPROLE 为 Pending live。

   重新启动组中的每个实例后，发出以下命令：

   ```
   oc exec -t <active-pod> -- dspmq -o nativeha -g
   ```

   其中，<active-pod> 是 oc get pods 命令显示的就绪 1/1 pod 的名称。 您可以看到 GRPROLE 是 Pending live ，然后是 live 。
3. 当实时站点和恢复站点的恢复日志同步后，恢复组将变为实时状态，并选举一名组长作为活动队列管理器。 应用程序将从这里自动重新连接到此实例，任何工作负载都将恢复正常。

下一步做什么

维护完成后，您可以通过重复该过程切换回原始的实时群组。

V 9.4.2 在原生高可用性 CRR 配置中完成计划外的故障切换

当实时站点发生灾难且无法继续运行时，您可以在本地高可用性跨区域复制（CRR）配置中，在实时组和恢复组之间执行计划外的故障切换。

关于此任务

当实时组关闭时，恢复组无法检测到这一点，因为它只是停止接收复制的日志数据，这可能只是意味着没有正在进行的工作负载。 因此，必须通过手动方式执行计划外的故障切换，将恢复组的作用更改为实时，并通知它之前处于实时状态的组已被禁用，这样它就不会尝试向其复制数据。

由于日志数据是异步复制的，因此恢复组可能与实时组不同步。 如果恢复组与实时组不同步，当发生计划外的故障切换时，数据就会丢失。 在启动计划外的故障切换之前，请在恢复组上发出 **dspmq -o nativeha -g** 命令，并使用 RCOVLSN 和 RCOVTIME 值查看日志序列号（LSN）和组可以恢复的时间。

过程

1. 确保应用程序已断开与实时群组的连接，并配置为重新连接到当前恢复群组。
2. 请完成以下步骤，指定恢复组成为活动组：

   a. 在 qm.ini 文件的 NativeHALocalInstance 节中将 GroupRole 设置为 Live 。 例如，如果您使用的是 MQ 运营商，请将 .spec.queueManager.availability.nativeHAGroups.local.role 的设置更改为 Live。

   b. 在 qm.ini 文件的 NativeHARecoveryGroup 节中将 Enabled 字段设置为 False 。 例如，如果您使用的是 MQ 运营商，请将 .spec.queueManager.availability.nativeHAGroups.remotes.enabled 的设置更改为 false。

如果在该组中的每个实例都重启后，您为该队列管理器发出 **dspmq -o nativeha -g** 命令，您将看到 GRPROLE 为 Live。 小组成员之间会进行选举，选出一位成员作为活跃实例，负责处理队列管理器的工作。

3. 要在故障转移后重新开始复制，您可以将原始的实时组用作恢复组，也可以在新的站点创建新的恢复组。
   - 如果原始的实时站点已恢复，在重新启动原始的实时群组之前，请将群组角色更改为恢复。 这确保了该小组不会恢复其现场角色。
   - 如果无法恢复原始的现场站点，请在新的站点创建一个新的原生高可用性恢复组。

   要配置恢复组，请完成 第 210 页的『在意外故障转移后恢复复制』 中描述的步骤。
4. 如果原始实时站点的配置没有更改，并且组在恢复时恢复其实时角色，则您将拥有两个活动的队列管理器实例。 如果是这种情况，系统会检测到分区状态，并且不再在两个组之间进行复制。 在还原原始的实时和恢复角色之前，您必须确定要保留哪些数据。 请发送邮件至 第 210 页的『解决分治（分裂脑）问题』 获取指导。

相关信息

dspmq（显示队列管理器）

V 9.4.2 在意外故障转移后恢复复制

在意外故障转移后，将实时组的复制还原到恢复组。

关于此任务

在计划外的故障切换后，如果原始的在线站点恢复，则可以将其重新配置为恢复组。 为了防止队列管理器进入分区（“分裂脑”）状态，请在原始实时站点上重新启动本机高可用性组之前完成此任务。

如果在计划外故障转移后无法恢复原始在线网站，您也可以在创建新的恢复组时完成这些步骤。

过程

1. 请完成以下步骤，指定该群组为恢复群组：
   a) 在 qm.ini 文件的 NativeHALocalInstance 节中将 GroupRole 设置为恢复。 例如，如果您使用的是 MQ 操作员，请将 QueueManager YAML 中 `.spec.queueManager.availability.nativeHAGroups.local.role` 字段的设置更改为“恢复”。

   b) 在 qm.ini 文件的 NativeHARecoveryGroup 节中将“已启用”字段设置为“False”。 例如，如果您使用的是 MQ 操作符，请将 QueueManager YAML 中 `.spec.queueManager.availability.nativeHAGroups.remotes.enabled` 字段的设置更改为 false。

   如果在每个实例重启后，在组实例上发出 **dspmq -o nativeha -g** 命令，您将看到 GRPROLE 为 Recovery。 小组成员之间进行选举，选出一名成员担任康复小组组长。
2. 通过将实时群组的 NativeHARecoveryGroup 文件中的 qm.ini 节段的 Enabled 字段设置为 True ，重新启动从实时群组到恢复群组的复制。 例如，如果您使用的是 MQ 操作符，请将 QueueManager YAML 中 `.spec.queueManager.availability.nativeHAGroups.remotes.enabled` 字段的设置更改为 true。

V 9.4.2 解决分治（分裂脑）问题

如果本地高可用跨区域复制（CRR）配置中的队列管理器同时在两个高可用性组中运行，则可能出现分区或“分裂大脑”的情况。

当发生计划外的故障切换时，就会出现这种情况。 与实时组的通信中断，因此您将队列管理器设为恢复组中的首要任务，使其开始运行。 与此同时，队列管理器仍在原始的 Live 群组中运行。 旧的 Live 小组仍有可能更新其日志，这会在两个小组维护的数据中产生一个无法解决的分支。 您必须先删除其中一个组的数据，才能重新合并。

要解决分区问题，您需要选择保留或删除哪些组数据。 可能是因为在触发计划外的故障切换时，两个小组的工作进度不同步，也可能是因为两个小组都能处理应用程序的工作量。对“最佳”数据的定义很大程度上是主观的。 例如，一个小组可能在日志中存储了更多的数据，但这可能主要是自动记录的 IBM MQ 对象的媒体图像；而另一个小组可能处理了更有价值的业务交易。 无论选择哪个组，在从分区状态恢复时，待删除组中未复制的消息和事务结果在核对可能的数据丢失时可能很有用。

IBM MQ 当具有实时角色的组连接到另一个运行实时角色的组时，会检测到分区状态。 如果您在两个组中发出命令 `dspmq -o nativeha -g`，则状态为 `GRSTATUS(Partitioned)` 。 两个小组之间的日志数据复制暂停。 在团队能够将足够的日志数据复制到另一个具有恢复角色的团队之前， `Partitioned` 状态将继续显示。

如果保留哪个组不是显而易见的选择，并且在计划外故障转移时两个组仍然有日志可用，则可以比较日志内容。

使用 **dspmq -o nativeha -g** 命令可识别日志中某个群组变为“实时”状态的确切时间点（日志序列号）。 LSN 以 `INITLSN`（*nnnnn*:*nnnnn*:*nnnnn*:*nnnnn*）的形式返回。 `INITTIME` 属性也报告了此事件的时间。 然后使用 **dmpmqlog -s** *startLSN* 命令检查从该点开始的日志内容。 请同时使用 **dspmq** 和 **dmpmqlog** 来识别由一个小组编写且未复制到另一个小组的日志记录。 **dmpmqlog** 命令还会报告消息和交易活动的摘要，在选择时可能很有用。

示例

“alpha”和“beta”组最初以实时和恢复配置的形式进行配置。 我们决定将“测试版”小组从“恢复”状态转换为“在线”状态，但意外出现了故障切换。 然而，“alpha”组仍然能够处理一些应用工作负载，而“beta”组则无法复制这些工作负载。 与此同时，一些应用程序连接到“测试版”，并承担了一些消息传递的工作。 “测试版”组中的 INITLSN 表示“测试版”组从“测试版”组的内容中分出的日志点。

比较过程需要队列管理器数据的副本，且队列管理器需要处于非活动状态。 在 Kubernetes 环境中，可以通过将队列管理器的状态集大小缩放为零，或者通过持久卷快照等方式复制数据来实现。

日志数据分析完成后，选择要丢弃的组数据，然后将其组角色更改为恢复，即可删除所有队列管理器持久卷或移除恢复组信息。

相关信息

dspmq

dmpmqlog

OpenShift MQ Adv. 结束本地 HA 队列管理器

对于容器中的 IBM MQ ，可以使用 **endmqm** 命令结束作为本地 HA 组一部分的活动队列管理器或副本队列管理器。

开始之前

注: 此信息仅适用于容器环境。

关于此任务

停止作为本地 HA 组一部分的队列管理器的程序取决于它是活动实例还是副本实例。 结束任何一种类型的实例时，都会进行检查，以确保结束实例不会破坏本地 HA 组的法定人数。 如果法定人数被破坏，**endmqm** 命令将失败。

当您发出 **endmqm** 命令时，组中的其他实例会收到警告，以便在连接中断时不会报错。

如果活动实例因太多副本实例结束或断开连接而失去法定人数，则活动实例在完全结束前会等待一段可配置的时间。 这样就可以有一段时间从容关闭处理程序，而不是让应用程序的连接直接中断。 超时值可由 `qm.ini` 文件 `NativeHALocalInstance` 节中的 `QuorumConnectivityTimeout` 属性指定。 默认值为 0 秒。

过程

- 要结束队列管理器的活动实例，请在运行活动实例的节点上发出以下命令：

  ```
  endmqm -s QMgrName
  ```

  - 指定 -r 选项可帮助客户端应用程序重新连接到另一个实例。
  - 如果该实例不是本地 HA 组中的活动实例，则命令失败。
  - 如果结束此活动实例会导致组法定人数失效，则命令失败。（如果在运行此命令的同时其他实例结束或不可用，则法定人数检查可能无法检测到这一点，本地 HA 组就会结束，只有当有足够的实例可用时才能重新启动）

  当活动队列管理器结束时，其中一个副本实例会接替活动队列管理器的角色。 您不能指定由哪个副本接管，而是由组内协商决定，并取决于哪个副本拥有最新的事务日志。
- 要结束队列管理器的副本实例，请执行以下命令：

  ```
  endmqm -x QMgrName
  ```

  - 如果该实例是活动实例，则命令失败。
  - 如果结束该副本实例会导致组法定人数失效，则命令失败。（如果在运行此命令的同时其他实例结束或不可用，则法定人数检查可能无法检测到这一点，本地 HA 组就会结束，只有当有足够的实例可用时才能重新启动）

  注： 您还可以在 Native HA 实例上使用 -c、-i、-p 或 -w 开关和 **endmqm** 命令，无论它们处于哪个角色。 队列管理器实例结束，忽略对组法定人数的影响。 不过，信息仍会与组内其他实例共享。 您可以将这些开关与活动实例的 -s 一起使用。 这些开关不能与副本实例的 -x 开关一起使用。

对容器中的 IBM MQ 进行故障诊断

如果在容器中运行 IBM MQ 时迁到问题，那么可以使用此处描述的方法来帮助您诊断和解决问题。

过程

- 第 212 页的『对容器中 IBM MQ 的意外重新启动进行故障诊断』.
- 第 213 页的『对 IBM MQ Operator 的问题进行故障诊断』.

OpenShift CP4I Kubernetes 对容器中 IBM MQ 的意外重新启动进行故障诊断

在大多数容器管理系统 (例如 Red Hat OpenShift Container Platform 和 Kubernetes) 中，通常会重新启动容器。 集装箱长寿是不正常的。 本主题说明了容器生命周期，如何调查重新启动以及意外容器重新启动背后的原因。

如果您未看到与 IBM MQ 部署相关的任何问题，并且该部署继续按预期运行，那么该解决方案很可能正在按预期执行。 您可能会在容器日志中看到类似如下的日志消息：

```
Signal received: terminated
```

这意味着 SIGTERM 信号已发送至 MQ 容器，要求其终止。 Linux 容器有责任响应 信号，这些信号是发送给程序的标准消息，用于触发行为。 POSIX SIGTERM 信号由 Kubernetes （以及 Red Hat OpenShift Container Platform ）用于容器正常终止。

当 IBM MQ 容器接收到 SIGTERM 信号时，它发出 `endmqm -w -r -tp` 命令以停止队列管理器。 队列管理器停止后，容器将停止。 如果队列管理器需要很长时间才能停止，那么可能会发送 SIGKILL 信号，这将立即终止 Linux 进程。 在 Kubernetes 中， SIGTERM 与 SIGKILL 之间的时间量称为 "终止宽限期"，可以在 `QueueManager` 资源 (如果您正在使用 IBM MQ Operator) 上配置，也可以直接在 Pod 资源上配置。 缺省值为 30 秒，其中 1 秒保留给容器以关闭，其余部分将提供给 IBM MQ。 例如，在缺省情况下，将发出 `endmqm -w -tp 29`，这将告知队列管理器需要 29 秒才能关闭。

集装箱终止的原因

Kubernetes 可以在现有 Pod 中重启容器，也可以删除 Pod。 Pod 的“状态”部分中的一些字段有助于指示这是否是容器重启，例如：

```
status:
  containerStatuses:
  - restartCount: 2
    started: true
    state:
      running:
        startedAt: "2025-01-20T12:03:15Z"
  startTime: "2024-12-28T13:46:12Z"
```

在这个例子中，您可以看到容器已经重启，并且容器比 Pod 新得多。 这表明一个容器重启了，这很可能意味着该容器无法通过存活性探测。 可以在 Kubernetes 中配置一个活性探测程序，以检查容器是否仍然健康。 IBM MQ Operator 设置队列管理器活动性探测器，该探测器调用 **dspmq** 命令以检查有效的运行状态。 如果队列管理器未处于正常状态，或者如果运行探测器本身需要太长时间，那么 kubelet 将考虑该故障。 可以在 `QueueManager` 资源 (如果您正在使用 IBM MQ Operator) 上或直接在 Pod 资源上配置容错失败次数的阈值。

如果没有看到容器重启，那么很可能整个 Pod 都已被替换。 请参阅 Kubernetes 文档中的 Pods 终止。 Kubernetes 使用术语 “Pod Disruption” 和“eviction”来描述节点上的 Pod 自愿或非自愿终止的过程。 可能驱逐 Pod 的原因有很多，包括：

- **滚动重启**。 如果需要进行配置更改， IBM MQ Operator 或 Stateful Set 控制器可以触发队列管理器 Pod 的滚动更新。 例如，更改图像版本、已安装的文件、环境变量、注释、标签等。
- **由 kubelet 终止**。 这可能有多种原因，包括：
  - 可以终止 Pod ，因为节点正在关闭 (可能是作为滚动集群更新的一部分)
  - 由于节点 "压力" (其中 kubelet 主动终止 Pod 以回收节点上的资源) ，可以终止 Pod。 Kubernetes 集群管理员可以配置可能因集群而异的逐出阈值。
- **由 Kubernetes 调度程序抢占**。 如果 Kubernetes 调度程序必须运行更高优先级的 Pod ，那么可能会发生此情况
- **污点节点**。 节点可以是 "污点" ，而不容许污点的 Pod 将被逐出。 Kubernetes 管理员使用污点从特定节点 "击退" Pod。 例如，为了说明 IBM MQ Pod 不应再在具有现在为其他工作负载保留的特殊硬件的节点上运行。
- **通过逐出 API 请求**。 管理员可以调用此命令来逐出 Pod
- **Pod 垃圾回收**。 如果节点退出服务或通过 Kubernetes API 除去，那么可能会发生此情况。

确定驱逐队列管理器 Pod 的原因

帮助了解 Pod 被逐出原因的潜在信息来源包括：

- **集群事件**。 例如， 在 OpenShift Container Platform 集群中查看系统事件信息。
- **集群审计事件**。 请参阅 在 Red Hat OpenShift Container Platform 中查看审计日志。
- **压力下的节点**。 查找 CPU ，网络或内存压力下的节点。 您可以在 "节点" 状态下看到此信息。 注意，到你来看的时候， Node 可能不再有压力了。
- **Red Hat OpenShift Container Platform 监视** 或其他监视度量值可能能够显示诸如磁盘等待时间问题之类的内容。 有用的 Prometheus 度量值为 `ibmmq_qmgr_log_write_latency_seconds`。 此信息来自 MQ 统计信息主题。

相关信息

有关调度，抢占和逐出的 Kubernetes 文档

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 214 页的『收集使用 IBM MQ Operator 部署的队列管理器的故障诊断信息』
- 第 216 页的『故障诊断: 获取对队列管理器数据的访问权』

OpenShift CP4I 收集使用 IBM MQ Operator 部署的队列管理器的故障诊断信息

收集在提出新的支持案例时应提供给 IBM 支持人员的故障诊断信息。

过程

1. 收集云提供者信息。

   这是托管 Red Hat OpenShift 集群的云提供者 (例如 IBM Cloud)。
2. 收集体系结构信息。

   Red Hat OpenShift 集群的体系结构是下列其中一项:

   - Linux for x86-64
   - Linux on Power Systems (ppc64le)
   - Linux on IBM Z
3. 收集 IBM MQ 部署信息。

   a) 使用 bash/zsh shell 登录到 Red Hat OpenShift 集群。

   b) 设置以下环境变量:

   ```
   export QM=QueueManager_name
   export QM_NAMESPACE=QueueManager_namespace
   export MQ_OPERATOR_NAMESPACE=mq_operator_namespace
   ```

   其中 *QueueManager_name* 是 QueueManager 资源的名称（在队列管理器 YAML 中为 `metadata.name`）、 *QueueManager_namespace* 是部署该资源的命名空间（队列管理器 YAML 中为 `metadata.namespace`）， *mq_operator_namespace* 是部署 IBM MQ Operator 的命名空间。这可能与 QueueManager 命名空间相同。

   c) 运行以下命令，并向 IBM 支持人员提供所有生成的输出文件。

   ```
   # OCP / Kubernetes: Version
   oc version -o yaml > ocversion.yaml

   # QueueManager: YAML
   oc get qmgr $QM -n $QM_NAMESPACE -o yaml > "queue-manager-$QM.yaml"

   # MQ Queue Manager: Pods
   oc get pods -n $QM_NAMESPACE -o wide --selector "app.kubernetes.io/instance=$QM" > "qm-
   pods-$QM.txt"

   # MQ Queue Manager: Pod YAML
   oc get pods -n $QM_NAMESPACE -o yaml --selector "app.kubernetes.io/instance=$QM" > "qm-
   pods-$QM.yaml"

   # MQ Queue Manager: Pod Logs
   for p in $(oc get pods -n $QM_NAMESPACE --no-headers --selector "app.kubernetes.io/
   instance=$QM" | cut -d ' ' -f 1); do oc logs -n $QM_NAMESPACE --previous "$p" > "qm-logs-
   previous-$p.txt"; oc logs -n $QM_NAMESPACE $p > "qm-logs-$p.txt";done

   # MQ Queue Manager: Describe Pods
   for p in $(oc get pods -n $QM_NAMESPACE --no-headers --selector "app.kubernetes.io/
   instance=$QM" | cut -d ' ' -f 1); do oc describe pod $p -n $QM_NAMESPACE > "qm-pod-
   describe-$p.txt"; done

   # MQ Web UI: Console Log
   for p in $(oc get pods -n $QM_NAMESPACE --no-headers --selector "app.kubernetes.io/
   instance=$QM" | cut -d ' ' -f 1); do oc cp -n $QM_NAMESPACE --retries=10 "$p:var/mqm/web/
   installations/Installation1/servers/mqweb/logs/console.log" "web-$p-console.log"; done

   # MQ Web UI: Messages Log
   for p in $(oc get pods -n $QM_NAMESPACE --no-headers --selector "app.kubernetes.io/
   ```

```
instance=$QM" | cut -d ' ' -f 1); do oc cp -n $QM_NAMESPACE --retries=10 "$p:var/mqm/web/
installations/Installation1/servers/mqweb/logs/messages.log" "web-$p-messages.log"; done

# MQ Queue Manager: routes defined by operator
oc get routes -n $QM_NAMESPACE -o yaml --selector "app.kubernetes.io/instance=$QM" > "qm-
routes-$QM.yaml"

# MQ Queue Manager: routes to QM
oc get routes -n $QM_NAMESPACE -o yaml --field-selector "spec.to.name=$QM-ibm-mq" > "qm-
routes2-$QM.yaml"

# MQ Queue Manager: stateful set
oc get statefulset -n $QM_NAMESPACE -o yaml ${QM}-ibm-mq > "qm-statefulset-$QM.yaml"

# MQ Queue Manager: revisions of the stateful set
oc get controllerrevisions.apps -o yaml -n $QM_NAMESPACE --selector "app.kubernetes.io/
instance=$QM" > "qm-statefulset-revisions-$QM.yaml"

# MQ Queue Manager: Pod events
for p in $(oc get pods -n $QM_NAMESPACE --no-headers --selector "app.kubernetes.io/
instance=$QM" | cut -d ' ' -f 1); do oc get -o custom-columns="LAST
SEEN:.lastTimestamp,TYPE:.type,REASON:.reason,KIND:.involvedObject.kind,NAME:.involvedObje
ct.name,MESSAGE:.message" event -n $QM_NAMESPACE --field-selector
involvedObject.name="$p" > "qm-pod-events-$p.txt"; done

# MQ Queue Manager: StatefulSet events
oc get events -n $QM_NAMESPACE -o custom-columns="LAST
SEEN:.lastTimestamp,TYPE:.type,REASON:.reason,KIND:.involvedObject.kind,NAME:.involvedObje
ct.name,MESSAGE:.message" --field-selector involvedObject.name="${QM}-ibm-mq" > "qm-
statefulset-events-$QM.txt"

# MQ Queue Manager: services
oc get services -n $QM_NAMESPACE -o yaml --selector "app.kubernetes.io/instance=$QM" >
"qm-services-$QM.yaml"

# MQ Queue Manager: PVCs
oc get pvc -n $QM_NAMESPACE -o yaml --selector "app.kubernetes.io/instance=$QM" > "qm-
pvcs-$QM.yaml"

# MQ Operator: Version
oc get csv -n $QM_NAMESPACE | grep "^ibm-mq\|NAME" > mq-operator-csv.txt

# Cloud Pak Foundational Services: Version
oc get csv -n $QM_NAMESPACE | grep "^ibm-common-service-operator\|NAME" > common-services-
csv.txt

# Cloud Pak for Integration: Version (if applicable)
oc get csv -n $QM_NAMESPACE | grep "^ibm-integration-platform-navigator\|NAME" > cp4i-
csv.txt

# Output from runmqras (this may take a while to execute)
for p in $(oc get pods -n $QM_NAMESPACE --no-headers --selector "app.kubernetes.io/
instance=$QM" | cut -d ' ' -f 1); do timestamp=$(TZ=UTC date +"%Y%m%d_%H%M%S"); oc exec
-n $QM_NAMESPACE $p -- runmqras -workdirectory "/tmp/runmqras_$timestamp" -section
logger,mqweb,nativeha,trace; oc cp -n $QM_NAMESPACE --retries=10 "$p:tmp/
runmqras_$timestamp/" .; done

# MQ Operator: Pod Log
oc logs -n $MQ_OPERATOR_NAMESPACE $(oc get pods -n $MQ_OPERATOR_NAMESPACE --no-headers --
selector app.kubernetes.io/name=ibm-mq,app.kubernetes.io/managed-by=olm | cut -d ' ' -f
1) > mq-operator-log.txt
```

注意:

- 这些命令中的大多数都需要对部署了队列管理器的名称空间的访问权。 但是，如果 IBM MQ Operator 已安装 **集群作用域**，那么收集 IBM MQ Operator 日志可能还需要 **集群管理员** 访问权。
- 如果有一个正在运行的队列管理器，请确保这些命令生成的输出文件不是空文件（ `qm-logs-previous-`*`pod_name`*`.txt` pod 日志除外，如果容器没有重启，这些文件应该是空的）。

相关任务

收集故障排除信息

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) 通过运行以下命令列出 PVC 目录:

   ```
   ls
   ```

   c) 通过在远程 shell 会话外部运行以下命令来查看 PVC 的列表:

   ```
   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 Troubleshooting container clusters with the OpenShift Monitoring CLI

As a troubleshooting aid, you can query the available container cluster metrics using the OpenShift Monitoring CLI.

Before you begin

OpenShift Monitoring provides a web console and a CLI. For day-to-day operations you might prefer to use the console, as described in “Monitoring container clusters with the OpenShift Monitoring console” on page 192. For gathering troubleshooting information you might prefer to use the CLI, as described in the rest of this task.

About this task

OpenShift Container Platform includes a pre-configured, pre-installed and self-updating monitoring stack that uses an open source Prometheus server. Queue managers running under an IBM MQ Operator can produce metrics compatible with Prometheus. For a complete list of the metrics available for IBM MQ, see “IBM MQ 容器发布的指标” on page 194.

OpenShift Monitoring is the default cluster monitoring method. It is installed with the cluster. To query the available container cluster metrics using the OpenShift Monitoring CLI, complete the following steps.

Procedure

1. Enable OpenShift Monitoring for the operands deployed in an IBM MQ container cluster.
   a) Log in to the cluster as a cluster administrator.
   b) Add "`enableUserWorkload: true`" to the configMap `cluster-monitoring-config` in the `openshift-monitoring` namespace. To do this, follow the steps in Enabling monitoring for user-defined projects in the Red Hat OpenShift documentation. If the `cluster-monitoring-config` ConfigMap does not exist in the `openshift-monitoring` namespace, follow the instructions to create it.
2. Install the OpenShift command line tool (oc).
   See Getting started with the OpenShift CLI.
3. Query the IBM MQ metrics using the OpenShift Monitoring CLI.

   When using the CLI, first declare the variables by using the **`export`** command. For example:

```
export OCP_LOGIN_TOKEN=$(oc whoami -t)
export THANOS_QUERIER_HOST=$(oc -n openshift-monitoring get route thanos-querier -o
jsonpath={.spec.host})
export QM_NAMESPACE=namespace-in-which-the-target-queue-manager-is-installed
export QM_METRIC_NAME=IBM-MQ-metric
```

   To run PromQL queries from the CLI, use the **`PROMQL_QUERY`** command. For example:

   - Get a list of queue managers that have a positive FDC count:

```
export QM_METRIC_NAME=ibmmq_qmgr_fdc_files
export PROMQL_QUERY="$QM_METRIC_NAME{namespace=\"$QM_NAMESPACE\"} > 0"
```

   - Get data for a specific time period:

```
export START=start_time
export END=end_time
export STEP=step_interval
curl -G -s -k -H "Authorization: Bearer $OCP_LOGIN_TOKEN" --data-urlencode
"query=$PROMQL_QUERY" \
--data-urlencode "start=$START" --data-urlencode "end=$END" \
--data-urlencode "step=$STEP" "https://$THANOS_QUERIER_HOST/api/v1/query_range"
```

where *start_time* and *end_time* specify the time in ISO 8601 format, and *step_interval* is (for example) 15m, or 30s, or 1h.

You can also export the data to a CSV file:

a. The following command uses the same headers that display in the web console output shown in the screen capture in "Monitoring container clusters with the OpenShift Monitoring console" on page 192. To export to a CSV file, and pass the same headers that display in the web console output, first download and install `jq`.

b. Run the following command:

```
(
echo
"metric_name,container,endpoint,instance,job,namespace,pod,prometheus,qmgr,service,timesta
mp,value"
curl -G -s -k -H "Authorization: Bearer $OCP_LOGIN_TOKEN" --data-urlencode
"query=$PROMQL_QUERY" \
--data-urlencode "start=$START" --data-urlencode "end=$END" --data-urlencode "step=$STEP"
\
"https://$THANOS_QUERIER_HOST/api/v1/query_range"| jq -r '.data.result[] | .metric as
$metric | .values[] | \
[$metric.__name__,
$metric.container,
$metric.endpoint,
$metric.instance,
$metric.job,
$metric.namespace,
$metric.pod,
$metric.prometheus,
$metric.qmgr,
$metric.service,
(.[0] | strftime("%Y-%m-%dT%H:%M:%SZ")), /* converts the timestamp from milliseconds to
ISO 8601 format */
.[1] ] | @csv'
) > your_filename.csv
```

CLI example: Write latency of Queue manager pods over the last 5 hours with a resolution of 10 minutes

```
export QM_METRIC_NAME=ibmmq_qmgr_log_write_latency_seconds
export PROMQL_QUERY="$QM_METRIC_NAME{namespace=\"$QM_NAMESPACE\"} [5h:10m]"
```

Export the data to a CSV file:

```
(
echo
"metric_name,container,endpoint,instance,job,namespace,pod,prometheus,qmgr,service,timestamp,val
ue"
curl -G -s -k -H "Authorization: Bearer $OCP_LOGIN_TOKEN" \
--data-urlencode "query=$PROMQL_QUERY" \
"https://$THANOS_QUERIER_HOST/api/v1/query" | jq -r '.data.result[] | .metric as $metric
| .values[] | \
[$metric.__name__,
$metric.container,
$metric.endpoint,
$metric.instance,
$metric.job,
$metric.namespace,
$metric.pod,
$metric.prometheus,
$metric.qmgr,
$metric.service,
(.[0] | strftime("%Y-%m-%dT%H:%M:%SZ")), /* converts the timestamp from milliseconds to ISO
8601 format */
.[1] ] | @csv'
) > your_filename.csv
```

Related tasks

"使用 IBM MQ Operator 时进行监视" on page 191
由 IBM MQ Operator 管理的队列管理器可以生成与 Prometheus 兼容的度量。 队列管理器指标已启用，并通过 HTTP 提供。

“Monitoring container clusters with the OpenShift Monitoring console” on page 192
You can query the available container cluster metrics using the OpenShift Monitoring web console.

Related reference
“IBM MQ 容器发布的指标” on page 194
队列管理器容器可以发布与 Red Hat OpenShift Monitoring 兼容的度量。

Kubernetes V 9.4.1 收集部署在 Kubernetes 上的队列管理器的故障排除信息

收集故障排除信息，在提出新的支持案例时提供给 IBM 支持部门。

过程

1. 收集云提供商信息。

   这是托管您的 Kubernetes 集群的云提供商（例如 IBM Cloud）。
2. 收集 Kubernetes 版本信息
   例如，运行 `kubectl version -o yaml > kubeversion.yaml`
3. 收集描述如何部署 IBM MQ 的关键要素的资源 YAML。
   包括以下内容

   a. StatefulSet

   b. Pod （针对每个队列管理器实例）

   c. 服务

   d. PersistentVolumeClaim

   e. ConfigMaps

   例如，为每个队列管理器 Pod 运行 `kubectl get pod` *PodName* `-o yaml`。
4. 收集每个花苞的描述，包括事件。
   例如，为每个队列管理器 pod 运行 `kubectl describe pod`。
5. 收集所有 Pod 日志
   例如，为每个队列管理器 Pod 运行 `kubectl logs` *PodName*。
6. 收集 **runmqras** 命令的输出结果
   使用 `kubectl exec` 在每个队列管理器容器内运行 **runmqras** 命令。

相关任务
收集故障排除信息

OpenShift CP4I 容器中 IBM MQ 的参考信息

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 IBM MQ Operator 的许可和 API 参考资料

IBM MQ 提供了一个 Kubernetes 操作器，可与 Red Hat OpenShift 容器平台进行本机集成。 `v1beta1` API 可用于创建和管理队列管理器资源。

mq.ibm.com/v1beta1: 当前许可证版本

`spec.license.license` 字段必须包含所接受许可证的许可证标识符。 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-CYPF-CRPF3H	Production 或 NonProduction	IBM Cloud Pak for Integration 16.1.2	9.4.3

spec.license.license 的值	spec.license.use 的值	许可证信息	适用的 IBM MQ 版本
L-MQQP-KBWMYE	Production 或 NonProduction	IBM Cloud Pak for Integration 限量版 16.1.2	9.4.3
L-NUUP-23NH8Y	Production	IBM MQ Advanced 9.4 - 2025 年 2 月	9.4.3 或 9.4.2
L-QWBN-T3PKLE	NonProduction	IBM MQ Advanced for Non-Production Environment 9.4 - 2025 年 2 月	9.4.3 或 9.4.2
L-HYGL-6STWD6	Development	IBM MQ Advanced for Developers （非保修） - 10/2024 9.4	9.4.3, 9.4.2, 或 9.4.1
L-QYVA-B365MB	Production 或 NonProduction	IBM Cloud Pak for Integration 16.1.1	9.4.2 或者 9.4.1
L-JVML-UFQVM4	Production 或 NonProduction	IBM Cloud Pak for Integration 限量版 16.1.1	9.4.2 或者 9.4.1
L-JTPV-KYG8TF	Production 或 NonProduction	IBM Cloud Pak for Integration 16.1.0	9.9.4.1 或 9.4.0
L-BMSF-5YDSLR	Production 或 NonProduction	IBM Cloud Pak for Integration 限量版 16.1.0	9.9.4.1 或 9.4.0
L-EHXT-MQCRN9	Production	IBM MQ Advanced 9.4	9.9.4.1 或 9.4.0
L-DRNQ-NGCA76	NonProduction	IBM MQ Advanced for Non-Production Environment 9.4	9.4.1
L-CLXQ-ADXTK3	Development	IBM MQ Advanced for Developers（非授权） 9.4	9.9.4.1 或 9.4.0

请注意，指定的许可证 版本与 IBM MQ 的版本不一定相同。

mq.ibm.com/v1beta1: 旧许可证版本

请参阅 IBM MQ 9.3 文档中的 旧版本许可证。

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。 IBM MQ 文档： https://ibm.biz/BdPZqj. 许可证编号 https://ibm.biz/Bdm9be.

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。
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 QueueManagerSpec	QueueManager 的期望状态。

字段	描述
status QueueManagerStatus	QueueManager 的观察状态。

.规格

QueueManager 的期望状态。

显示在:

- 第 220 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。
imagePullSecretsLocalObjectReference 矩阵	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。 更多信息，请参见 https://kubernetes.io/docs/concepts/containers/images#specifying-imagepullsecrets-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 QueueManagerConfig	队列管理器容器和底层队列管理器的设置。
securityContext 安全背景	要添加到队列管理器 Pod 的 securityContext 的安全设置。
service 服务	适用于所有 QueueManager 服务的设置。 修改或删除本部分或其中的字段可能导致所有 QueueManager 服务的再生。 服务再生将中断通信，导致包括 NativeHA 在内的所有 QueueManager 类型停机。 再生服务通过 Kubernetes 分配新的 ClusterIP 地址。 需要 MQ Operator 3.6.0 或更高版本。
telemetry 遥测	Open Telemetry 配置的设置。 需要 MQ 操作程序 2.2.0 或更高版本。
template 模板	Kubernetes 资源的高级模板。 该模板允许用户覆盖 IBM MQ 生成 Kubernetes 底层资源（如 StatefulSet,Pods 和 Services）的方式。 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。
web WebServerConfig	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 221 页的『.规格』

.spec.imagePullSecrets

LocalObjectReference 包含的信息足以让您在同一命名空间内找到被引用的对象。

显示在:

- 第 221 页的『.规格』

字段	描述
name 字符串	引用的名称。 该字段实际上是必填字段，但由于向后兼容性的原因，允许为空。 这里的空值几乎肯定是错误的。 更多信息 https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names. 默认值： ""。

.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 221 页的『.规格』

.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 221 页的『.规格』

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。 请参见 https://ibm.biz/Bdm9be 了解有效值。
metric 字符串	用于指定要使用的许可证度量的设置。 例如，ProcessorValueUnit、VirtualProcessorCore、ManagedVirtualServer 或 FREE。 使用 MQ Advanced 许可证或 MQ Advancedfor Non-Production 许可证时默认为 ProcessorValueUnit，使用 Cloud Pak for Integration 许可证时默认为 VirtualProcessorCore，使用 MQ Advanced for Developers 许可证时默认为 FREE。
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。 请参见 https://ibm.biz/Bdm9be 了解有效值。

.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 221 页的『.规格』

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 222 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 223 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 222 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 223 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 221 页的『.规格』

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。
env 环境数组	为队列管理器提供环境变量的设置。 需要 MQ 操作员 3.5.0 或更高版本。
files 文件数组	为队列管理器提供文件的设置。 需要 MQ 操作员 3.5.0 或更高版本。
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 QueueManagerMetrics	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)。 如果您希望队列管理器名称不符合 Kubernetes 规则的名称 (例如，包含大写字母的名称)，请使用此字段。
readinessProbe QueueManagerReadinessProbe	用于控制就绪性探测器的设置。
recoveryLogs 恢复日志	MQ 恢复日志的设置。 需要 MQ Operator 2.4.0 或更高版本。
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 启动尝试	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 QueueManagerStorage	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.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。

显示在:

- 第 223 页的『.spec.queueManager』

字段	描述
nativeHAGroups 本地化小组	本地高可用性组设置。 需要将 type of availability 改为 NativeHA。 需要 MQ 操作员 3.5.0 或更高版本。
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。

字段	描述
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。 详情请参见 https://ibm.biz/Bdm9TW 了解更多详情。
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.nativeHAGroups

本地高可用性组设置。 需要将 type of availability 改为 NativeHA。 需要 MQ 操作员 3.5.0 或更高版本。

显示在:

- 第 224 页的『.spec.queueManager.可用性』

字段	描述
local 本地	此 NativeHA 组的设置。
remotes 遥控阵列	用于连接远程 NativeHA 群组的设置。

.spec.queueManager.availability.nativeHAGroups 本地

此 NativeHA 组的设置。

显示在:

- 第 225 页的『.spec.queueManager.availability.nativeHAGroups』

字段	描述
name 字符串	这个 NativeHA 群的名字。
role 字符串	这个 NativeHA 小组应该扮演的角色。 Live 或 Recovery。 有效值为: "Live"; "Recovery"。
route 路由	NativeHA 群路线的设置。
tls Tls	TLS 设置用于在 NativeHA 群组之间配置安全通信。 如果父可用性部分未配置 TLS，则必须填写。

.spec.queueManager.availability.nativeHAGroups.local.route

NativeHA 群路线的设置。

显示在:

- 第 225 页的『.spec.queueManager.availability.nativeHAGroups 本地』

字段	描述
enabled 布尔值	是否启用路由。 缺省值为 true。

.spec.queueManager.availability.nativeHAGroups.local.tls

TLS 设置用于在 NativeHA 群组之间配置安全通信。 如果父可用性部分未配置 TLS，则必须填写。

显示在:

- 第 225 页的『.spec.queueManager.availability.nativeHAGroups 本地』

字段	描述
cipherSpec 字符串	NativeHA 群组 TLS 的 CipherSpec 名称。
key 钥匙	本地 HA 组之间安全通信中使用的 TLS 密钥的设置。

.spec.queueManager.availability.nativeHAGroups.local.tls.key

本地 HA 组之间安全通信中使用的 TLS 密钥的设置。

显示在:

- 第 225 页的『.spec.queueManager.availability.nativeHAGroups.local.tls』

字段	描述
secretName 字符串	Kubernetes 私钥的名称。 密文类型应为 " kubernetes.io/tls "， 包含单个私钥、公共证书和信任链。

.spec.queueManager.availability.nativeHAGroups.遥控器

显示在:

- 第 225 页的『.spec.queueManager.availability.nativeHAGroups』

字段	描述
addresses 阵列	一系列主机名和端口地址，用于与远程 NativeHA 组通信。 仅当本地组角色为 Live 时有效。
enabled 布尔值	是否开始与遥控器进行通信。 仅与本地 NativeHA 组中的“实时”角色兼容。 如果未设置， MQ 容器将使用内部默认值。
name 字符串	远程 NativeHA 群组名称。 这必须与远程集群中配置的 NativeHA 组名一致。
retries 重试	远程重试行为的配置。
statusInterval 整数	恢复组在发送恢复日志状态之前等待的最长时间间隔（以毫秒为单位）。
trust 信任数组	证书将添加到队列管理器的密钥库中，以便与远程 NativeHA 组进行通信。

.spec.queueManager.availability.nativeHAGroups.remotes.retries

远程重试行为的配置。

显示在:

- 第 226 页的『.spec.queueManager.availability.nativeHAGroups.遥控器』

字段	描述
long 长	在较长的重试阶段，重试配置。
reporting 字符串	是报告所有重试尝试还是仅报告重试阶段。
short 简短的	在短重试阶段重试配置。

.spec.queueManager.availability.nativeHAGroups.remotes.retries.long

在较长的重试阶段，重试配置。

显示在:

- 第 226 页的『.spec.queueManager.availability.nativeHAGroups.remotes.retries』

字段	描述
interval 整数	在远程连接长时间重试阶段，两次尝试之间等待的毫秒间隔。

.spec.queueManager.availability.nativeHAGroups.remotes.retries.short

在短重试阶段重试配置。

显示在:

- 第 226 页的『.spec.queueManager.availability.nativeHAGroups.remotes.retries』

字段	描述
count 整数	在短暂的重新尝试阶段尝试连接的次数。
interval 整数	在短重试阶段，尝试连接远程设备之间等待的间隔时间（以毫秒为单位）。

.spec.queueManager.availability.nativeHAGroups.remotes.trust

NativeHAGroupTrust 定义了证书颁发机构公钥的来源。

显示在:

- 第 226 页的『.spec.queueManager.availability.nativeHAGroups.遥控器』

字段	描述
configMap 配置图	使用 Kubenetes 配置图提供 CA 证书。
secret 私钥	使用 Kubernetes 密钥提供 CA 证书。

.spec.queueManager.availability.nativeHAGroups.remotes.trust.configMap

使用 Kubenetes 配置图提供 CA 证书。

显示在:

- 第 227 页的『.spec.queueManager.availability.nativeHAGroups.remotes.trust』

字段	描述
items 阵列	Kubernetes configmap source 中的密钥，应应用这些密钥。 如果未指定，则使用所有按键。
name 字符串	Kubernetes configmap 源的名称。

.spec.queueManager.availability.nativeHAGroups.remotes.trust.secret

使用 Kubernetes 密钥提供 CA 证书。

显示在:

- 第 227 页的『.spec.queueManager.availability.nativeHAGroups.remotes.trust』

字段	描述
items 阵列	钥匙在 Kubernetes 秘密里面，应该打开它。 如果未指定，则使用所有按键。
secretName 字符串	Kubernetes 私钥的名称。

.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 224 页的『.spec.queueManager.可用性』

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.环境

EnvVar 表示容器中存在环境变量。

显示在:

- 第 223 页的『.spec.queueManager』

字段	描述
name 字符串	环境变量的名称。 必须是 C_IDENTIFIER。
value 字符串	变量引用 $(VAR_NAME) 使用容器中预先定义的环境变量和任何服务环境变量进行扩展。 如果变量无法解析，则输入字符串中的引用将保持不变。 双倍$符号被简化为单个$符号，从而避免了使用$(VAR_NAME)语法：即 $$(VAR_NAME) 将生成字符串。 $(VAR_NAME) 无论变量是否存在，跳过的引用都不会被展开。 默认为 。
valueFrom 价值	环境变量值的来源。 如果值不为空，则无法使用。

.spec.queueManager.env.valueFrom

环境变量值的来源。 如果值不为空，则无法使用。

显示在:

- 第 228 页的『.spec.queueManager.环境』

字段	描述
configMapKeyRef 配置映射键参考	选择 ConfigMap 的密钥。
fieldRef Fieldref	选择播客字段：支持 metadata.name、 metadata.namespace、 metadata.labels['<KEY>']、 metadata.annotations['<KEY>']、 spec.nodeName, spec.serviceAccountName, status.hostIP, status.podIP, status.podIPs。
resourceFieldRef 资源字段参考	选择容器中的资源：目前仅支持资源限制和请求（ limits.cpu、 limits.memory、 limits.ephemeral-storage、 requests.cpu、 requests.memory 和 requests.ephemeral-storage ）。
secretKeyRef 密钥参考	在豆荚的命名空间中选择一个密钥。

.spec.queueManager.env.valueFrom.configMapKeyRef

选择 ConfigMap 的密钥。

显示在:

- 第 228 页的『.spec.queueManager.env.valueFrom』

字段	描述
key 字符串	选择的关键。
name 字符串	引用的名称。 该字段实际上是必填字段，但由于向后兼容性的原因，允许为空。 这里的空值几乎肯定是错误的。 更多信息 https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names. 默认值： ""。
optional 布尔值	请指定是否必须定义 ConfigMap 或其密钥。

.spec.queueManager.env.valueFrom.fieldRef

选择播客字段：支持 metadata.name、 metadata.namespace、 metadata.labels['<KEY>']、 metadata.annotations['<KEY>']、 spec.nodeName, spec.serviceAccountName, status.hostIP, status.podIP, status.podIPs。

显示在:

- 第 228 页的『.spec.queueManager.env.valueFrom』

字段	描述
apiVersion 字符串	FieldPath 所使用的模式版本默认为 v1。
fieldPath 字符串	在指定 API 版本中选择字段路径。

.spec.queueManager.env.valueFrom.resourceFieldRef

选择容器中的资源：目前仅支持资源限制和请求（ limits.cpu、 limits.memory、 limits.ephemeral-storage、 requests.cpu、 requests.memory 和 requests.ephemeral-storage ）。

显示在:

- 第 228 页的『.spec.queueManager.env.valueFrom』

字段	描述
containerName 字符串	容器名称：对于卷是必需的，对于环境变量是可选的。
divisor	指定公开资源的输出格式，默认为 1。
resource 字符串	必填：资源选择。

.spec.queueManager.env.valueFrom.secretKeyRef

在豆荚的命名空间中选择一个密钥。

显示在:

- 第 228 页的『.spec.queueManager.env.valueFrom』

字段	描述
key 字符串	选择的关键在于秘诀。 必须是一个有效的密钥。
name 字符串	引用的名称。 该字段实际上是必填字段，但由于向后兼容性的原因，允许为空。 这里的空值几乎肯定是错误的。 更多信息 https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names. 默认值： ""。
optional 布尔值	指定是否必须定义密钥或密钥。

.spec.queueManager.文件

文件来源。

显示在:

- 第 223 页的『.spec.queueManager』

字段	描述
configMap 配置图	ConfigMap 代表一个包含文件的。 Kubernetes ConfigMap
secret 私钥	秘密代表一个包含文件的 Kubernetes。

.spec.queueManager.files.configMap

ConfigMap 代表一个包含文件的。 Kubernetes ConfigMap

显示在:

- 第 229 页的『.spec.queueManager.文件』

字段	描述
defaultMountPath 字符串	Kubernetes 源文件中用于密钥的默认安装路径。 这可以通过为密钥指定单独的安装路径来覆盖。 如果没有特别说明，将使用 /etc/mqm/customfiles 。
items 项目数组	Kubernetes 源代码中的密钥应该应用。
name 字符串	Kubernetes 源的名称。

.spec.queueManager.files.configMap.项目

显示在:

- 第 230 页的『.spec.queueManager.files.configMap』

字段	描述
item 字符串	密钥的名称。
mountPath 字符串	用于存放密钥的挂载路径将覆盖默认挂载路径。

.spec.queueManager.files.secret

秘密代表一个包含文件的 Kubernetes。

显示在:

- 第 229 页的『.spec.queueManager.文件』

字段	描述
defaultMountPath 字符串	Kubernetes 源文件中用于密钥的默认安装路径。 这可以通过为密钥指定单独的安装路径来覆盖。 如果没有特别说明，将使用 /etc/mqm/customfiles 。
items 项目数组	Kubernetes 源代码中的密钥应该应用。
name 字符串	Kubernetes 源的名称。

.spec.queueManager.files.secret.items

显示在:

- 第 230 页的『.spec.queueManager.files.secret』

字段	描述
item 字符串	密钥的名称。
mountPath 字符串	用于存放密钥的挂载路径将覆盖默认挂载路径。

.spec.queueManager.ini

INI 配置文件的源。

显示在:

- 第 223 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 231 页的『.spec.queueManager.ini』

字段	描述
items 阵列	Kubernetes 源代码中的密钥应该应用。
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 231 页的『.spec.queueManager.ini』

字段	描述
items 阵列	Kubernetes 源代码中的密钥应该应用。
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 223 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-probes。

字段	描述
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-probes。

.spec.queueManager.度量

Prometheus 样式度量的设置。

显示在:

- 第 223 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。
serviceMonitor 服务监督员	Prometheus 兼容指标的服务监控器设置。 需要 MQ Operator 3.6.0 或更高版本。
tls Tls	TLS 设置，用于为 Prometheus 兼容的指标配置安全通信。 需要 MQ 操作员 3.5.0 或更高版本。

.spec.queueManager.metrics.serviceMonitor

Prometheus 兼容指标的服务监控器设置。 需要 MQ Operator 3.6.0 或更高版本。

显示在:

- 第 232 页的『.spec.queueManager.度量』

字段	描述
enabled 布尔值	是否启用服务监控器。 缺省值为 true。

.spec.queueManager.metrics.tls

TLS 设置，用于为 Prometheus 兼容的指标配置安全通信。 需要 MQ 操作员 3.5.0 或更高版本。

显示在:

- 第 232 页的『.spec.queueManager.度量』

字段	描述
provider 字符串	用于为 Prometheus 兼容指标配置安全通信的提供商。 缺省值为 none。 使用 none 来获取不安全的 HTTP 指标。 使用 openshift，将 OpenShift-provided 证书与 ServiceMonitor 结合使用。

.spec.queueManager.mqsc

MQSC 配置文件的源。

显示在:

- 第 223 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 232 页的『.spec.queueManager.mqsc』

字段	描述
items 阵列	Kubernetes 源代码中的密钥应该应用。
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 232 页的『.spec.queueManager.mqsc』

字段	描述
items 阵列	Kubernetes 源代码中的密钥应该应用。
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 223 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-probes。
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-probes。

.spec.queueManager.recoveryLogs

MQ 恢复日志的设置。 需要 MQ Operator 2.4.0 或更高版本。

显示在:

- 第 223 页的『.spec.queueManager』

字段	描述
logFilePages 整数	恢复日志数据保存在一系列文件中。 日志文件大小以 4 KB 页面为单位指定。

.spec.queueManager.资源

用于控制资源需求的设置。

显示在:

- 第 223 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 234 页的『.spec.queueManager.资源』

字段	描述
cpu	
memory	

.spec.queueManager.resources.requests

CPU 和内存设置。

显示在:

- 第 234 页的『.spec.queueManager.资源』

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 223 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 223 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 24。
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-probes。
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-probes。

.spec.queueManager.存储

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 223 页的『.spec.queueManager』

字段	描述
allowVolumeExpansion 布尔值	是否允许扩展卷。
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，存储类可以是 ReadWriteOnce 或 ReadWriteMany 类型。 如果 type of availability 为 MultiInstance，存储类必须是 ReadWriteMany 类型。
defaultDeleteClaim 布尔值	删除队列管理器时是否应删除所有卷。 特定的持久卷可以定义自己的 deleteClaim 值，该值将覆盖 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 QueueManagerVolume	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。
scratch 临时	队列管理器的 Scratch 临时卷的设置。 此卷将作为 "/run" 文件夹安装在容器上。 仅当根文件系统设置为只读时才适用。 需要 MQ 操作程序 3.0.0 或更高版本。
tmp 临时	队列管理器的 Tmp 临时卷的设置。 此卷将作为 "/tmp" 文件夹安装在容器上。 将在此卷中创建诊断数据文件，例如 runmqras 命令生成的 zip 文件。 仅当根文件系统设置为只读时才适用。 需要 MQ 操作程序 3.0.0 或更高版本。

.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 235 页的『.spec.queueManager.存储』

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，存储类可以是 ReadWriteOnce 或 ReadWriteMany 类型。 如果 type of availability 为 MultiInstance，存储类必须是 ReadWriteMany 类型。
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 且根文件系统设置为只读时才有效。 需要 MQ 操作程序 3.0.0 或更高版本。
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 235 页的『.spec.queueManager.存储』

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，存储类可以是 ReadWriteOnce 或 ReadWriteMany 类型。 如果 type of availability 为 MultiInstance，存储类必须是 ReadWriteMany 类型。
deleteClaim 布尔值	删除队列管理器时是否应删除此卷。
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 且根文件系统设置为只读时才有效。 需要 MQ 操作程序 3.0.0 或更高版本。
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 235 页的『.spec.queueManager.存储』

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，存储类可以是 ReadWriteOnce 或 ReadWriteMany 类型。 如果 type of availability 为 MultiInstance，存储类必须是 ReadWriteMany 类型。
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 且根文件系统设置为只读时才有效。 需要 MQ 操作程序 3.0.0 或更高版本。
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.scratch

队列管理器的 Scratch 临时卷的设置。 此卷将作为 "/run" 文件夹安装在容器上。 仅当根文件系统设置为只读时才适用。 需要 MQ 操作程序 3.0.0 或更高版本。

显示在:

- 第 235 页的『.spec.queueManager.存储』

字段	描述
sizeLimit 字符串	临时卷的大小限制，包括 SI 单元。 例如，2Gi。 仅当根文件系统设置为只读时才有效。 需要 MQ 操作程序 3.0.0 或更高版本。

.spec.queueManager.storage.tmp

队列管理器的 Tmp 临时卷的设置。 此卷将作为 "/tmp" 文件夹安装在容器上。 将在此卷中创建诊断数据文件，例如 runmqras 命令生成的 zip 文件。 仅当根文件系统设置为只读时才适用。 需要 MQ 操作程序 3.0.0 或更高版本。

显示在:

- 第 235 页的『.spec.queueManager.存储』

字段	描述
sizeLimit 字符串	临时卷的大小限制，包括 SI 单元。 例如，2Gi。 仅当根文件系统设置为只读时才有效。 需要 MQ 操作程序 3.0.0 或更高版本。

.spec.securityContext

要添加到队列管理器 Pod 的 securityContext 的安全设置。

显示在:

- 第 221 页的『.规格』

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。 权限位与 0660 相 OR。 如果未设置，Kubelet 将不会修改任何卷的所有权和权限。
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。
readOnlyRootFilesystem 布尔值	是否为队列管理器启用只读根文件系统设置。 缺省值为 false。 需要 MQ 操作程序 3.0.0 或更高版本。
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.service

适用于所有 QueueManager 服务的设置。 修改或删除本部分或其中的字段可能导致所有 QueueManager 服务的再生。 服务再生将中断通信，导致包括 NativeHA 在内的所有 QueueManager 类型停机。 再生服务通过 Kubernetes 分配新的 ClusterIP 地址。 需要 MQ Operator 3.6.0 或更高版本。

显示在:

- 第 221 页的『.规格』

字段	描述
ipFamilies 阵列	指定用于所有 QueueManager 服务的 IP 家族。 有效选项为 IPv4 和 IPv6。 如果未指定 Kubernetes ，将根据 ipFamilyPolicy 设置默认值。 如果手动设置，如果群集上没有可用的 IP 系列，或其值与 ipFamilyPolicy 设置不兼容，则服务创建将失败。 修改数组中的第一个值（主 IP 家族）或删除该字段将导致所有 QueueManager 服务重新生成。 服务再生将中断通信，导致包括 NativeHA 在内的所有 QueueManager 类型停机。 再生服务通过 Kubernetes 分配新的 ClusterIP 地址。
ipFamilyPolicy 字符串	指定用于所有 QueueManager 服务的 IP 家族策略。 有效选项为 SingleStack, PreferDualStack 和 RequireDualStack。 SingleStack 是针对单个 IP 系列的。 PreferDualStack 在双协议栈配置的集群上适用于两个 IP 系列，在单协议栈集群上适用于一个 IP 系列。 RequireDualStack 是针对双堆栈配置集群上的两个 IP 系列，否则失败。 删除该字段将导致所有 QueueManager 服务的再生。 服务再生将中断通信，导致包括 NativeHA 在内的所有 QueueManager 类型停机。 再生服务通过 Kubernetes 分配新的 ClusterIP 地址。

.spec.telemetry

Open Telemetry 配置的设置。 需要 MQ 操作程序 2.2.0 或更高版本。

显示在:

- 第 221 页的『.规格』

字段	描述
tracing 跟踪	开放式遥测跟踪的设置。

.spec.telemetry.tracing

开放式遥测跟踪的设置。

显示在:

- 第 238 页的『.spec.telemetry』

字段	描述
instana Instana	Instana 跟踪的设置。

.spec.telemetry.tracing.instana

Instana 跟踪的设置。

显示在:

- 第 239 页的『.spec.telemetry.tracing』

字段	描述
agentHost 字符串	要向其发送跟踪数据的 Instana 代理程序的主机名。 这不应包含协议。
enabled 布尔值	是否启用 Instana 跟踪。 缺省值为 false。
protocol 字符串	要用于与 Instana 代理程序通信的协议。 支持 http 和 https。

.spec.template

Kubernetes 资源的高级模板。 该模板允许用户覆盖 IBM MQ 生成 Kubernetes 底层资源（如 StatefulSet,Pods 和 Services）的方式。 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 221 页的『.规格』

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 221 页的『.规格』

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 239 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 239 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-probes。
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-probes。

.spec.tracing.agent.readinessProbe

用于控制就绪性探测器的设置。

显示在:

- 第 239 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-probes。
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-probes。

.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 239 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 241 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-probes。
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-probes。

.spec.tracing.collector.readinessProbe

用于控制就绪性探测器的设置。

显示在:

- 第 241 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-probes。
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。

字段	描述
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-probes。

.spec.web

MQ Web 服务器的设置。

显示在:

- 第 221 页的『.规格』

字段	描述
console 控制台	MQ Web 控制台的设置。 需要 MQ 操作程序 3.0.0 或更高版本。
enabled 布尔值	是否启用 Web 服务器。 缺省值为 false。
manualConfig 手动配置	用于提供 Web 服务器 XML 配置的设置。 需要 MQ 操作程序 3.0.0 或更高版本。
route 路由	网络服务器路由的设置。 需要 MQ Operator 3.6.0 或更高版本。

.spec.web.console

MQ Web 控制台的设置。 需要 MQ 操作程序 3.0.0 或更高版本。

显示在:

- 第 242 页的『.spec.web』

字段	描述
authentication 认证	MQ Web 控制台的认证设置。 需要 MQ 操作程序 3.0.0 或更高版本。
authorization 授权	MQ Web 控制台的授权设置。 需要 MQ 操作程序 3.0.0 或更高版本。

.spec.web.console.authentication

MQ Web 控制台的认证设置。 需要 MQ 操作程序 3.0.0 或更高版本。

显示在:

- 第 242 页的『.spec.web.console』

字段	描述
provider 字符串	要用于 MQ Web 控制台的认证提供程序。 使用 integration-keycloak 可将单点登录与 Cloud Pak for Integration Platform UI (Keycloak) 配合使用。 如果使用 Cloud Pak for Integration 许可证，那么缺省为 integration-keycloak ; 如果使用 MQ 许可证，那么缺省为 manual 。 如果要提供您自己的配置，请使用 manual 。

.spec.web.console.authorization

MQ Web 控制台的授权设置。 需要 MQ 操作程序 3.0.0 或更高版本。

显示在:

- 第 242 页的『.spec.web.console』

字段	描述
provider 字符串	要用于 MQ Web 控制台的授权提供程序。 使用 integration-keycloak 可使用 Cloud Pak for Integration Keycloak 提供的角色。 如果要提供您自己的配置，请使用 manual 。 如果使用 Cloud Pak for Integration 许可证，那么缺省为 integration-keycloak ; 如果使用 MQ 许可证，那么缺省为 manual 。

.spec.web.manualConfig

用于提供 Web 服务器 XML 配置的设置。 需要 MQ 操作程序 3.0.0 或更高版本。

显示在:

- 第 242 页的『.spec.web』

字段	描述
configMap 配置图	ConfigMap 表示包含 Web 服务器 XML 配置的 Kubernetes ConfigMap 。
secret 私钥	私钥表示包含 Web 服务器 XML 配置的 Kubernetes 私钥。 使用 Secret 可保护 Kubernetes 层中的任何凭证，但监视或故障诊断工具可能会过早地公开底层文件。 为提高安全性，请使用“securityUtility” 对凭据进行编码。

.spec.web.manualConfig.configMap

ConfigMap 表示包含 Web 服务器 XML 配置的 Kubernetes ConfigMap 。

显示在:

- 第 243 页的『.spec.web.manualConfig』

字段	描述
name 字符串	Kubernetes 源的名称。

.spec.web.manualConfigsecret

私钥表示包含 Web 服务器 XML 配置的 Kubernetes 私钥。 使用 Secret 可保护 Kubernetes 层中的任何凭证，但监视或故障诊断工具可能会过早地公开底层文件。 为提高安全性，请使用“securityUtility” 对凭据进行编码。

显示在:

- 第 243 页的『.spec.web.manualConfig』

字段	描述
name 字符串	Kubernetes 源的名称。

.spec.web.route

网络服务器路由的设置。 需要 MQ Operator 3.6.0 或更高版本。

显示在:

- 第 242 页的『.spec.web』

字段	描述
enabled 布尔值	是否启用路由。 缺省值为 true。

.状态

QueueManager 的观察状态。

显示在:

- 第 220 页的『QueueManager』

字段	描述
adminUiUrl 字符串	URL 用于管理界面。
availability 可用性	队列管理器的可用性状态。
conditionsQueueManagerStatusCondition 矩阵	条件表示队列管理器状态的最新可用观测值。
customImages 布尔值	操作员选择的队列管理器图像是否已被自定义图像覆盖。
endpointsQueueManagerStatusEndpoint 矩阵	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。
metadata 元数据	元数据表示队列管理器的其他信息，包括集成-Keycloak 状态。
name 字符串	队列管理器的名称。
nativeHACRR 本地人	关于已配置的本机高可用性跨区域复制的信息。
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 244 页的『.状态』

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 244 页的『.状态』

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。 有效值为: "True"; "False"; "Unknown"。
type 字符串	条件的类型。

.status.endpoints

QueueManagerStatusEndpoint 定义了 QueueManager 的端点。

显示在:

- 第 244 页的『.状态』

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 "表示 UI 端点，"API "表示 API 端点，"'OpenAPI'"表示 API 文档。
uri 字符串	端点的 URI。

.status.metadata

元数据表示队列管理器的其他信息，包括集成-Keycloak 状态。

显示在:

- 第 244 页的『.状态』

字段	描述
integrationKeycloak 整合密钥	QueueManagerStatusIntegrationKeycloak 定义 QueueManager 的 Keycloak 关键字状态。

.status.metadata.integrationKeycloak

QueueManagerStatusIntegrationKeycloak 定义 QueueManager 的 Keycloak 关键字状态。

显示在:

- 第 245 页的『.status.metadata』

字段	描述
clientName 字符串	

.status.nativeHACRR

关于已配置的本机高可用性跨区域复制的信息。

显示在:

- 第 244 页的『.状态』

字段	描述
local 本地	本地集群状态。

.status.nativeHACRR 本地

本地集群状态。

显示在:

- 第 245 页的『.status.nativeHACRR』

字段	描述
address 字符串	本地集群的地址。

字段	描述
groupName 字符串	本地群组的群组名称。

.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 244 页的『.状态』

字段	描述
available QueueManagerStatusVersionAvailable	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 246 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 定义了 MQ 的版本。

显示在:

- 第 246 页的『.status.versions.available』

字段	描述
licenses 许可证 数组	适用于此版本的 QueueManager 的许可证。
name 字符串	此版本的 QueueManager 的版本 name 。 这些是 spec.version 字段的有效值。

.status.versions.available.versions.licenses

QueueManagerStatusLicense 定义了许可证。

显示在:

- 第 246 页的『.status.versions.available.versions』

字段	描述
displayName 字符串	许可证的显示名称。
link 字符串	许可证内容的链接。

字段	描述
matchesCurrentType 布尔值	许可证是否与当前使用的许可证类型匹配。
name 字符串	许可证的名称。

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。

为 QueueManager 资源定义了以下条件:

表 *2:* 队列管理器状态条件			
组件	**条件类型**	**原因码**	**消息警告**
QueueManager[2]	已阻止	OperatorDependency	要进行安装，此实例需要由 [IBM Cloud Pak for Integration] 配置 Keycloak 。 此实例将保持为 [暂挂] 状态，直到在此 QueueManager 的 Cp4iServicesBinding 资源中将 Keycloak 报告为 [KeycloakReady] 为止。
			要进行安装，此实例需要操作程序 [IBM IAM]。 在 [IBM Cloud Pak 基础服务] 安装操作程序之前，此实例将保持 [已阻止] 状态。
	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
	暂挂	已停止	MQ 队列管理器已停止，因为在 QueueManager 定义中存在 "mq.ibm.com/stop" 注释并设置为 "true"。 停止 QueueManager StatefulSet 副本计数设置为零时，将除去所有 MQ 队列管理器 pod。
	错误	失败	MQ 队列管理器部署失败
	警告	UnsupportedVersion	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	CP4I-LTS 支持	已安装 CP4I-LTS 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	CP4I-LTS 支持	已安装 CP4I-LTS 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。 此操作数不符合扩展支持持续时间的要求。
展舱 [3]	暂挂	PodPending	正在部署 MQ 队列管理器的 Pod
	错误	PodFailed	正在部署 MQ 队列管理器的 Pod

[2] 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 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。

[3] 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 2: 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 [4]	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
	警告	StorageExpansion 暂挂	Volume expansion is pending for the following PVCs [<list of pvcs>]
	警告	StorageMismatch	Storage sizes defined in the QueueManager resource do not match the capacity of one or more provisioned PVCs [<list of pvcs>]. AllowVolumeExpansion is set to false in the QueueManager resource so the MQ Operator will not attempt to reconcile these differences.
	错误	StorageFailed	Storage for MQ 队列管理器无法供应

Linux IBM MQ 集装箱许可证注释

在 Kubernetes 和 Red Hat OpenShift Container Platform 环境中， IBM License Service 使用 Pod 注释，根据容器而不是底层机器上定义的限制来跟踪使用情况。 IBM MQ Operator 会为您设置这些 Pod 注释，但使用其他方式部署时，您必须自己设置这些注释。

有关 IBM MQ Operator 如何根据队列管理器 YAML 的值注释 pod 的信息，请参阅 第 254 页的『IBM MQ 应用的容器许可注释。 IBM MQ Operator』。

在部署 IBM MQ 容器时，有两种常见的许可方法:

- 许可运行容器的整个机器。
- 根据相关限制许可集装箱（仅限 Kubernetes 和 Red Hat OpenShift Container Platform ）

这两种方案都可供客户选择，更多详情请访问 Passport Advantage 上的 IBM Container Licenses 页面。

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。 根据您在容器中部署的权利和功能，使用以下一个或多个注释。

注: 许多注释包含以下一行或两行:

```
productChargedContainers: "All" | "NAME_OF_CONTAINER"
productMetric: "PROCESSOR_VALUE_UNIT" | "VIRTUAL_PROCESSOR_CORE"
```

在使用注释之前，必须编辑以下行:

- 对于 `productChargedContainers`，必须选择 `"All"`，或者替换容器的实际名称。
- 对于 `productMetric`，必须选择其中一个提供的值。

[4] 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。存储条件还会监视卷扩展的进度，并针对队列管理器定义中定义的存储大小与已部署 PVC 的大小之间的不匹配发出警报。 如果在存储器供应期间发生任何错误，那么会将 `StorageFailed` 条件添加到条件列表，并将整体队列管理器条件设置为 `Failed`。

要与 IBM MQ 产品权利配合使用的注释

如果您具有 IBM MQ 产品权利，请选择下面与您已购买并要使用的权利匹配的注释。

- 第 251 页的『IBM MQ』
- 第 252 页的『IBM MQ 高级』
- 第 252 页的『IBM MQ (对于非生产环境)』
- 第 252 页的『IBM MQ 非生产环境高级』
- 第 252 页的『IBM MQ 面向开发人员的高级功能』

要用于 IBM MQ 多实例高可用性配置的 IBM MQ 注释如下所示。 另请参阅第 250 页的『为高可用性配置选择正确的注释』。

- 第 252 页的『IBM MQ 容器多实例』
- 第 252 页的『IBM MQ 高级容器多实例』
- 第 252 页的『IBM MQ 用于非生产环境的容器多实例』
- 第 252 页的『IBM MQ 针对非生产环境的高级容器多实例』

要与 CP4I 产品权利配合使用的注释

如果您具有 IBM Cloud Pak for Integration (CP4I) 权利，请选择下面与您已购买并要使用的权利相匹配的注释。

- 第 253 页的『具有 CP4I 权利的 IBM MQ』
- 第 253 页的『IBM MQ Advanced with CP4I 权利』
- 第 253 页的『IBM MQ (针对具有 CP4I 权利的非生产环境)』
- 第 253 页的『IBM MQ Advanced for Non-Production Environment with CP4I 权利』

要用于 IBM MQ 多实例高可用性配置的 CP4I 注释如下所示。 另请参阅第 250 页的『为高可用性配置选择正确的注释』。

- 第 253 页的『具有 CP4I 权利的 IBM MQ 容器多实例』
- 第 253 页的『具有 CP4I 权利的 IBM MQ 高级容器多实例』
- 第 254 页的『具有 CP4I 权利的 IBM MQ Container Multi Instance for Non-Production Environment』
- 第 254 页的『具有 CP4I 权利的 IBM MQ Advanced Container Multi Instance for Non-Production Environment』

为高可用性配置选择正确的注释

IBM MQ 多实例

在 IBM MQ 多实例高可用性配置中部署一对队列管理器时，应该在这两个实例上使用相同的注释。 根据购买的权利，应选择下列其中一个注释:

- IBM MQ 或 IBM MQ Advanced 独立权利
  - 第 252 页的『IBM MQ 容器多实例』
  - 第 252 页的『IBM MQ 高级容器多实例』
  - 第 252 页的『IBM MQ 用于非生产环境的容器多实例』
  - 第 252 页的『IBM MQ 针对非生产环境的高级容器多实例』
- IBM Cloud Pak for Integration 权利 (entitlement)
  - 第 253 页的『具有 CP4I 权利的 IBM MQ 容器多实例』
  - 第 253 页的『具有 CP4I 权利的 IBM MQ 高级容器多实例』
  - 第 254 页的『具有 CP4I 权利的 IBM MQ Container Multi Instance for Non-Production Environment』

  - 第 254 页的『具有 CP4I 权利的 IBM MQ Advanced Container Multi Instance for Non-Production Environment』

与 IBM Cloud Pak for Integration 权利一起使用时，注释中的权利比率可确保记录正确的权利使用情况。 与独立 IBM MQ 或 IBM MQ Advanced 权利配合使用时，需要将每个实例的 License Service 中报告的注释映射到 IBM MQ 权利部件，如下所示:

- IBM MQ Advanced Container 多实例
  - 1 x IBM MQ Advanced **和** 1 x IBM MQ Advanced 高可用性副本 **或**
  - 2 x IBM MQ Advanced[5]
- IBM MQ Advanced Container 非生产环境的多实例
  - 1 x IBM MQ Advanced **和** 1 x IBM MQ Advanced 高可用性副本 **或**
  - 2 x IBM MQ Advanced 表示非生产环境)[5]
- IBM MQ 容器多实例
  - 1 x IBM MQ **和** 1 x IBM MQ 高可用性副本 **或**
  - 2 x IBM MQ[5]
- IBM MQ 用于非生产环境的容器多实例
  - 1 x IBM MQ **和** 1 x IBM MQ 高可用性副本 **或**
  - 2 x IBM MQ 表示非生产环境) [5]

IBM MQ 本机 HA

如果要在本机 HA 定额中部署三个队列管理器，那么只有活动实例才会使用权利。 所有实例都应该具有相同的注释。 应根据所购买的权利选择下列其中一项:

- IBM MQ 或 IBM MQ Advanced 独立权利
  - 第 252 页的『IBM MQ 高级』
  - 第 252 页的『IBM MQ 非生产环境高级』
- IBM Cloud Pak for Integration 权利 (entitlement)
  - 第 253 页的『IBM MQ Advanced with CP4I 权利』
  - 第 253 页的『IBM MQ Advanced for Non-Production Environment with CP4I 权利』

注释

本主题的其余部分详细描述了每个注释的内容。

IBM MQ

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

要在使用这些许可证注释的队列管理器上使用本地 HA 和跨区域复制功能，请参阅与您的环境相匹配的主题:

- 第 101 页的『在 IBM MQ 许可的队列管理器上配置本地 HA 和跨区域复制，方法是使用 IBM MQ Operator』
- 第 163 页的『在 IBM MQ 许可的队列管理器上配置本机 HA 和跨区域复制。 Kubernetes』

[5] 此权利选项是次优选项，仅当相关高可用性副本部件的权利不可用时才应使用。

IBM MQ 高级

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

IBM MQ (对于非生产环境)

```
productID: "151bec68564a4a47a14e6fa99266deff"
productName: "IBM MQ for Non-Production Environment"
productMetric: "PROCESSOR_VALUE_UNIT" | "VIRTUAL_PROCESSOR_CORE"
productChargedContainers: "All" | "NAME_OF_CONTAINER"
```

要在使用这些许可证注释的队列管理器上使用本地 HA 和跨区域复制功能，请参阅与您的环境相匹配的主题：

- 第 101 页的『在 IBM MQ 许可的队列管理器上配置本地 HA 和跨区域复制，方法是使用 IBM MQ Operator』
- 第 163 页的『在 IBM MQ 许可的队列管理器上配置本机 HA 和跨区域复制。 Kubernetes』

IBM MQ 非生产环境高级

```
productID: "21dfe9a0f00f444f888756d835334909"
productName: "IBM MQ Advanced for Non-Production Environment"
productMetric: "PROCESSOR_VALUE_UNIT" | "VIRTUAL_PROCESSOR_CORE"
productChargedContainers: "All" | "NAME_OF_CONTAINER"
```

IBM MQ 面向开发人员的高级功能

```
productID: "2f886a3eefbe4ccb89b2adb97c78b9cb"
productName: "IBM MQ Advanced for Developers (Non-Warranted)"
productMetric: "FREE"
productChargedContainers: "All" | "NAME_OF_CONTAINER"
```

IBM MQ 容器多实例

```
productID: "2dea73b866b648b6b4abe2a85eb76964"
productName: "IBM MQ Container Multi Instance"
productMetric: "PROCESSOR_VALUE_UNIT" | "VIRTUAL_PROCESSOR_CORE"
productChargedContainers: "All" | "NAME_OF_CONTAINER"
```

IBM MQ 高级容器多实例

```
productID: "bd35bff411bb47c2a3f3a4590f33a8ef"
productName: "IBM MQ Advanced Container Multi Instance"
productMetric: "PROCESSOR_VALUE_UNIT" | "VIRTUAL_PROCESSOR_CORE"
productChargedContainers: "All" | "NAME_OF_CONTAINER"
```

IBM MQ 用于非生产环境的容器多实例

```
productID: "af11b093f16a4a26806013712b860b60"
productName: "IBM MQ Container Multi Instance for Non-Production Environment"
productMetric: "PROCESSOR_VALUE_UNIT" | "VIRTUAL_PROCESSOR_CORE"
productChargedContainers: "All" | "NAME_OF_CONTAINER"
```

IBM MQ 针对非生产环境的高级容器多实例

```
productID: "31f844f7a96b49749130cd0708fdbb17"
productName: "IBM MQ Advanced Container Multi Instance for Non-Production Environment"
```

```
productMetric: "PROCESSOR_VALUE_UNIT" | "VIRTUAL_PROCESSOR_CORE"
productChargedContainers: "All" | "NAME_OF_CONTAINER"
```

具有 CP4I 权利的 IBM MQ

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

IBM MQ Advanced with CP4I 权利

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

IBM MQ (针对具有 CP4I 权利的非生产环境)

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

IBM MQ Advanced for Non-Production Environment with CP4I 权利

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

具有 CP4I 权利的 IBM MQ 容器多实例

```
productName: "IBM MQ Container Multi Instance"
productID: "2dea73b866b648b6b4abe2a85eb76964"
productChargedContainers: "All" | "NAME_OF_CONTAINER"
productMetric: "VIRTUAL_PROCESSOR_CORE"
productCloudpakRatio: "10:3"
cloudpakName: "IBM Cloud Pak for Integration"
cloudpakId: "c8b82d189e7545f0892db9ef2731b90d"
```

具有 CP4I 权利的 IBM MQ 高级容器多实例

```
cloudpakId: "c8b82d189e7545f0892db9ef2731b90d"
cloudpakName: "IBM Cloud Pak for Integration"
productID: "bd35bff411bb47c2a3f3a4590f33a8ef"
productName: "IBM MQ Advanced Container Multi Instance"
productMetric: "VIRTUAL_PROCESSOR_CORE"
productChargedContainers: "All" | "NAME_OF_CONTAINER"
productCloudpakRatio: "5:3"
```

具有 CP4I 权利的 IBM MQ Container Multi Instance for Non-Production Environment

```
cloudpakId: "c8b82d189e7545f0892db9ef2731b90d"
cloudpakName: "IBM Cloud Pak for Integration"
productID: "af11b093f16a4a26806013712b860b60"
productName: "IBM MQ Container Multi Instance for Non-Production Environment"
productMetric: "VIRTUAL_PROCESSOR_CORE"
productChargedContainers: "All" | "NAME_OF_CONTAINER"
productCloudpakRatio: "20:3"
```

具有 CP4I 权利的 IBM MQ Advanced Container Multi Instance for Non-Production Environment

```
cloudpakId: "c8b82d189e7545f0892db9ef2731b90d"
cloudpakName: "IBM Cloud Pak for Integration"
productID: "31f844f7a96b49749130cd0708fdbb17"
productName: "IBM MQ Advanced Container Multi Instance for Non-Production Environment"
productMetric: "VIRTUAL_PROCESSOR_CORE"
productChargedContainers: "All" | "NAME_OF_CONTAINER"
productCloudpakRatio: "10:3"
```

Linux IBM MQ 应用的容器许可注释。 IBM MQ Operator

在 Kubernetes 和 Red Hat OpenShift Container Platform 环境中， IBM License Service 使用 Pod 注释，根据容器而不是底层机器上定义的限制来跟踪使用情况。 IBM MQ Operator 会为您设置这些 Pod 注释，但使用其他方式部署时，您必须自己设置这些注释。

有关可用注释，请参见 第 249 页的『IBM MQ 集装箱许可证注释』。 下表描述了 IBM MQ Operator 根据队列管理器 YAML 中的这些值对 pod 应用了哪些注释：

- 的价值 `spec.license.license`
- 的价值 `spec.license.use`
- 的价值 `spec.queueManager.availability.type`

注: 本表指的是最新的光盘许可证，适用于 IBM MQ 9.4.3.0。 有关其他 IBM MQ 版本的许可证，请参见 第 219 页的『IBM MQ Operator 的许可和 API 参考资料』。

注释	Value of spec.license.license	spec.license.use 的值	spec.queueManager.availability.type 的值
IBM MQ Advanced	L-NUUP-23NH8Y	Production	SingleInstance 或 NativeHA
IBM MQ Advanced for Non-Production Environment	L-QWBN-T3PKLE	NonProduction	SingleInstance 或 NativeHA
IBM MQ Advanced for Developers	L-HYGL-6STWD6	Development	SingleInstance 或 NativeHA
IBM MQ Advanced Container Multi Instance	L-NUUP-23NH8Y	Production	MultiInstance
IBM MQ Advanced Container Multi Instance for Non-Production Environment	L-QWBN-T3PKLE	NonProduction	MultiInstance
IBM MQ Advanced CP4I	L-CYPF-CRPF3H	Production	SingleInstance 或 NativeHA

注释	Value of spec.license.license	spec.license.use 的值	spec.queueManager.availability.type 的值
IBM MQ Advanced for Non-Production Environment CP4I	L-CYPF-CRPF3H	NonProduction	SingleInstance 或 NativeHA
IBM MQ Advanced Container Multi Instance CP4I	L-CYPF-CRPF3H	Production	MultiInstance
IBM MQ Advanced Container Multi Instance for Non-Production Environment CP4I	L-CYPF-CRPF3H	NonProduction	MultiInstance

Linux V 9.4.1 IBM MQ Advanced 容器映像 (container image)

IBM MQ Advanced 可使用预构建的容器映像。 该图像可从 IBM Container Registry 中获取。

用法

要使用图像，您必须通过设置 **LICENSE** 环境变量来接受 IBM MQ Advanced 许可证的条款。

支持的环境变量

LANG
设置许可证的打印语言。

许可证
设置 accept 以同意 IBM MQ Advanced for Developers 许可条件。

设置 view 查看许可证条件。

MQ_CMDLEVEL
设置为要使用的 MQ 命令级别。

mq_enable_embedded_web_server
设置为 true，以便在容器启动时启动 Web 服务器。

MQ_ENABLE_FIPS
设为 true 可启用 FIPS 模式。 设置为 false 则禁用 FIPS 模式。 默认情况下，如果底层 Linux 主机已启用 FIPS，则会自动启用 FIPS 模式，但该选项对测试很有用。

mq_enable_metrics
设置 true 可为队列管理器生成 Prometheus 指标。

mq_enable_trace_crtmqdir
设置为 "true "可在运行 **crtmqdir** 命令时创建 MQ 诊断跟踪。

mq_enable_trace_crtmqm
设置为 "true "可在运行 **crtmqm** 命令时创建 MQ 诊断跟踪。

mq_enable_trace_strmqm
设置为 "true "可在运行 **strmqm** 命令时创建 MQ 诊断跟踪。

mq_grace_period
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。 设置容器终止时使用的 **endmqm -tp** 选项。 有关更多信息，请参阅 endmqm（结束队列管理器）。

mq_logging_console_source
为镜像到容器的 **stdout** 位置的日志指定一个以逗号分隔的来源列表。

有效值为 qmgr、web 和 mqsc 。

默认值为 qmgr,web。

可选值为 mqsc。 此选项可用于在容器日志中反映 autocfgmqsc.LOG 的内容。

mq_logging_console_format
更改打印到容器 **stdout** 位置的日志格式。

设置 basic 以使用简单的人类可读格式。 这是缺省值。

设置 json 以使用 JSON 格式（每行一个 JSON 对象）。

mq_logging_console_exclude_id
指定以逗号分隔的日志信息 ID 列表。

日志信息仍会显示在磁盘上的日志文件中，但不会打印到容器的 **stdout** 位置。

默认值为 AMQ5041I,AMQ5052I,AMQ5051I,AMQ5037I,AMQ5975I。

V 9.4.3 **已启用 mq_logging_metrics_audit_enabled**
设置为 "true "可启用访问 Prometheus 指标端点的审计日志。 日志输出会发送到一个 JSON 文件中，文件名为 /var/mqm/errors/。 您还必须设置 MQ_ENABLE_METRICS=true ， 以生成 Prometheus 指标。

mq_multi_instance
设置为 "true "可启用作为多实例队列管理器运行。 这会更改与 **endmqm** 一起使用的选项。

MQ_NATIVE_HA
设置为 "true "可启用本地 HA。 这将设置 **crtmqm -lr** 选项。 有关详细信息，请参阅 crtmqm （创建队列管理器）。 您可以通过安装 INI 文件来配置其他 Native HA 设置。 参见 "NativeHAInstance stanza of the qm.ini file 和 "NativeHALocalInstance stanza of the qm.ini file"。

Deprecated **MQ_NATIVE_HA_INSTANCE_0_NAME**
在 NativeHAInstance INI stanza 中为三个本地 HA 实例之一设置 Name 属性。 参见 "NativeHAInstance stanza of the qm.ini file。 从 "IBM MQ 9.4.1 开始，该变量已被弃用，您应在 "/etc/mqm 中提供 INI 文件片段。

Deprecated **MQ_NATIVE_HA_INSTANCE_1_NAME**
在 NativeHAInstance INI stanza 中为三个本地 HA 实例之一设置 Name 属性。 参见 "NativeHAInstance stanza of the qm.ini file。 从 "IBM MQ 9.4.1 开始，该变量已被弃用，您应在 "/etc/mqm 中提供 INI 文件片段。

Deprecated **MQ_NATIVE_HA_INSTANCE_2_NAME**
在 NativeHAInstance INI stanza 中为三个本地 HA 实例之一设置 Name 属性。 参见 "NativeHAInstance stanza of the qm.ini file。 从 "IBM MQ 9.4.1 开始，该变量已被弃用，您应在 "/etc/mqm 中提供 INI 文件片段。

Deprecated **MQ_NATIVE_HA_INSTANCE_0_REPLICATION_ADDRESS**
在 NativeHAInstance INI stanza 中为三个本地 HA 实例之一设置 ReplicationAddress 属性。 参见 "NativeHAInstance stanza of the qm.ini file。 从 "IBM MQ 9.4.1 开始，该变量已被弃用，您应在 "/etc/mqm 中提供 INI 文件片段。

Deprecated **MQ_NATIVE_HA_INSTANCE_1_REPLICATION_ADDRESS**
在 NativeHAInstance INI stanza 中为三个本地 HA 实例之一设置 ReplicationAddress 属性。 参见 "NativeHAInstance stanza of the qm.ini file。 从 "IBM MQ 9.4.1 开始，该变量已被弃用，您应在 "/etc/mqm 中提供 INI 文件片段。

Deprecated **MQ_NATIVE_HA_INSTANCE_2_REPLICATION_ADDRESS**
在 NativeHAInstance INI stanza 中为三个本地 HA 实例之一设置 ReplicationAddress 属性。 参见 "NativeHAInstance stanza of the qm.ini file。 从 "IBM MQ 9.4.1 开始，该变量已被弃用，您应在 "/etc/mqm 中提供 INI 文件片段。

Deprecated **mq_native_ha_cipherspec**
在 NativeHALocalInstance INI stanza 中为三个本地 HA 实例之一设置 CipherSpec 属性。 参见 "NativeHALocalInstance stanza of the qm.ini file。 从 "IBM MQ 9.4.1 开始，该变量已被弃用，您应在 "/etc/mqm 中提供 INI 文件片段。

Deprecated **mq_native_ha_key_repository**
忽略自动生成的 Native HA 密钥存储库，使用指定的密钥存储库。 从 "IBM MQ 9.4.1 开始，该变量已被弃用，您应在 "/etc/mqm 的 INI 文件片段中提供密钥存储库的位置。

mq_qmgr_log_file_pages
MQ 恢复日志数据保存在一系列称为日志文件的文件中。 日志文件大小以 4 KB 页为单位指定。 该环境变量用于设置 **crtmqm -lf** 选项。 有关详细信息，请参阅 crtmqm （创建队列管理器）。

mq_qmgr_name
设置您希望创建的队列管理器名称。

文件系统和挂载点

/etc/mqm
队列管理器启动时将处理该目录下的任何 MQSC 或 INI 文件。 这是自动配置功能使用的目录。 有关更多信息，请参阅 启动时从 MQSC 脚本自动配置 和 启动时自动配置 qm.ini 。

/etc/mqm/pki/keys
包含以 PKCS #1、ASN.11 DER 形式或未加密的 PKCS #8、ASN.1DER 形式提供私钥的子目录。 扫描 /etc/mqm/pki/keys 下的每个子目录，查找扩展名为 .key 的文件。 公共 X.509 证书链应位于同一目录下，以 ASN.1DER 格式存在于一个或多个扩展名为 ".crt 的文件中。 如果以这种方式安装密钥，"/run/runmqserver/tls 中就会自动生成一个密钥存储库。 密钥存储库将在队列管理器的 "SSLKEYR 设置中设置，默认密钥（CERTLABL）将是按词法对密钥名称列表排序后的第一个名称。

/etc/mqm/pki/trust
包含公共 X.509 证书的子目录，证书格式为 ASN.1 DER。 将扫描每个子目录中的任何 .crt 文件，并将其添加为可信证书。 任何私钥的证书都会自动被信任，不需要以这种方式添加。

/etc/mqm/ha/pki/keys
包含以 PKCS #1、ASN.11 DER 形式或未加密的 PKCS #8、ASN.1DER 形式提供私钥的子目录。 扫描 /etc/mqm/ha/pki/keys 下的每个子目录，查找扩展名为 .key 的文件。 公共 X.509 证书链应位于同一目录下，以 ASN.1DER 格式存在于一个或多个扩展名为 ".crt 的文件中。 如果以这种方式挂载密钥，就会自动生成一个密钥存储库，以及使用该密钥存储库的 INI 文件片段。 使用的密钥将是按词典顺序排列密钥名称后的第一个名称。 生成的密钥库被写入 "/run/runmqserver/ha/tls。

/etc/mqm/metrics/pki/keys
如果 9.4.2.0-r1 目录中提供了 TLS 密钥，且队列管理器启用了指标，则会在 /metrics 端口 9157 上启动一个 HTTPS 服务器。 此目录中的文件应为 PEM 编码证书：

- tls.crt 服务器公共证书
- tls.key 服务器专用密钥
- ca.crt CA 公共证书（可选）

如果没有提供 TLS 密钥，或者使用的是早期版本的 IBM MQ ，则启动 HTTP 服务器。

/etc/mqm/web
当容器启动时，/etc/mqm/web 下的目录树会被复制到 /var/mqm/web 的顶部。 例如，您可以创建 /etc/mqm/web/installations/Installation1/servers/mqweb/mqwebuser.xml 来配置网络服务器。 该文件将被复制到 /var/mqm/web 下的正确位置，通常是一个已挂载的卷。 参见 "配置 "IBM MQ Console 和 "REST API。

/mnt/mqm
/var/mqm 目录象征性地链接到此位置。 通过将持久卷挂载到该位置，可以在容器运行期间持久保存 MQ 数据。

/mnt/mqm-log
用于保存队列管理器恢复日志文件的目录。 如果卷被挂载到该位置，容器代码会设置 **crtmqm -ld** 选项。 有关详细信息，请参阅 crtmqm （创建队列管理器）。

/mnt/mqm-data
用于保存队列管理器数据文件的目录。 如果卷被挂载到该位置，容器代码会设置 **crtmqm -md** 选项。 有关详细信息，请参阅 crtmqm （创建队列管理器）。

/run
容器会在此目录下写入临时文件。 如果想在容器中使用只读文件系统，则需要将卷挂载到该位置，并挂载到 /tmp 中

/run/termination-log
如果容器代码遇到会导致队列管理器终止的错误，它会将原因写入该文件。 Kubernetes 中默认使用此文件位置来检索终止信息。

/tmp
容器会在此目录下写入临时文件。 如果想在容器中使用只读文件系统，则需要将卷挂载到该位置，并挂载到 /run 中

Linux IBM MQ 许可的容器映像

预构建的容器映像可用于 IBM MQ。 该图片可从 IBM Container Registry 上获取。

用法

要使用该镜像，必须通过设置 **LICENSE** 环境变量来接受 IBM MQ 许可证的条款。 有关更多信息，请参阅第 140 页的『使用 IBM MQ 许可证注释配置队列管理器。 IBM MQ Operator』。

支持的环境变量

IBM MQ 许可的容器映像支持的环境变量与 IBM MQ Advanced 容器映像支持的环境变量相同。 有关完整列表，请参阅 IBM MQ Advanced 容器镜像支持的环境变量。

注: 从 IBM MQ 9.4.2 ， IBM MQ 许可的队列管理器可通过向队列管理器添加额外许可配置为使用本地 HA 和跨区域复制功能。 有关更多信息，请参阅第 101 页的『在 IBM MQ 许可的队列管理器上配置本地 HA 和跨区域复制，方法是使用 IBM MQ Operator』。 在 IBM MQ 9.4.2 之前，使用 IBM MQ 许可证的队列管理器无法使用本地 HA 功能。

文件系统和挂载点

IBM MQ 许可的容器映像支持的文件系统和挂载点与 IBM MQ Advanced 容器映像支持的文件系统和挂载点相同。 有关完整列表，请参阅 IBM MQ Advanced 容器映像中的文件系统和挂载点。

OpenShift CP4I Kubernetes IBM MQ Advanced for Developers 容器映像 (container image)

预构建的容器映像可用于 IBM MQ Advanced for Developers。 此映像可从 IBM Container Registry 获取。此镜像适用于 Docker、Podman、Kubernetes 和其他容器环境。

可用映像

IBM MQ 映像存储在 IBM Container Registry 中:

- IBM MQ Advanced for Developers 9.4.0.11: `icr.io/ibm-messaging/mq:9.4.0.11-r3`
- IBM MQ Advanced for Developers 9.4.3.0: `icr.io/ibm-messaging/mq:9.4.3.0-r1`

快速参考

- 许可证:
  - 第 219 页的『mq.ibm.com/v1beta1: 当前许可证版本』 和 Apache 许可证 2.0. 请注意， IBM MQ Advanced for Developers 许可证不允许进一步分发，并且这些条款限制开发者机器的使用。
- 记录问题的位置:
  - GitHub
- 可用于以下 CPU 体系结构:

- amd64
- s390x
- ppc64le

用法

在容器中运行 IBM MQ Advanced for Developers 。

有关如何运行容器的详细信息，请参阅 使用情况文档 。

要能够使用该映像，必须通过设置 **LICENSE** 环境变量来接受 IBM MQ 许可证的条款。

环境变量

第 255 页的『IBM MQ Advanced 容器映像 (container image)』 中的所有环境变量都可用，另外还有以下附加变量：

Deprecated **MQ_ADMIN_PASSWORD**
指定管理用户的密码。

长度必须至少为 8 个字符。

没有管理员用户的缺省密码。

V 9.4.0 V 9.4.0 从 IBM MQ 9.4.0 开始，不再提供此变量。 本主题中的 YAML 示例说明了如何通过挂载秘密来提供管理员密码。

Deprecated **MQ_APP_PASSWORD**
指定应用程序用户的密码。

如果设置了此属性，那么将使 **DEV.APP.SVRCONN** 通道成为安全通道，并且仅允许提供有效用户标识和密码的连接。

长度必须至少为 8 个字符。

没有应用程序用户的缺省密码。

V 9.4.0 V 9.4.0 从 IBM MQ 9.4.0 开始，不再提供此变量。 本主题中的 YAML 示例说明了如何通过挂载秘密文件来提供应用程序密码。

MQ_DEV
设置 `false` 以停止正在创建的缺省对象。

有关 IBM MQ Advanced for Developers 映像支持的缺省开发者配置的更多信息，请参阅 缺省开发者配置文档。

用于描述如何为 admin 和 app 用户指定密码的示例队列管理器 YAML

对于 **admin** 和 **app** 用户标识的用户，您必须在使用 `Development` 许可证部署队列管理器时提供密码。 以下是一个示例队列管理器 YAML ，它向您显示如何使用 IBM MQ Operator 执行此操作。

以下命令将创建包含 **admin** 和 **app** 用户密码的私钥。

```
oc create secret generic my-mq-dev-passwords --from-literal=mqAdminPassword=passw0rd --from-
literal=mqAppPassword=passw0rd
```

以下 YAML 在部署队列管理器时使用这些密码。

```
apiVersion: mq.ibm.com/v1beta1
kind: QueueManager
metadata:
  name: qm-dev
  annotations: com.ibm.mq/write-defaults
  spec: 'false'
spec:
  license:
    accept: true
```

```
    license: L-HYGL-6STWD6
    use: Development
  web:
    enabled: true
  queueManager:
    env:
      - name: MQ_DEV
        value: "true"
      - name: MQ_CONNAUTH_USE_HTP
        value: "true"
    files:
      - secret:
          items:
            - item: mqAdminPassword
            - item: mqAppPassword
      name: my-mq-dev-passwords
      defaultMountPath: /run/secrets
    storage:
      queueManager:
        type: persistent-claim
    name: QUICKSTART
  version: 9.4.3.0-r1
```

IBM MQ MFT Agent for Developers 容器映像 (container image)

预构建的容器映像可用于 IBM MQ MFT Agent for Developers。 该图片可从 IBM Container Registry 上获取。 此镜像适用于 Docker, Podman, Kubernetes 以及其他容器环境。

注意: Deprecated IBM MQ MFT Agent for Developers 以前可从 Hub 获取镜像，但现在已不再使用，而且 Hub 上也不再提供更新。 Docker Docker

可用映像

IBM MQ MFT Agent for Developers 图像存储在 中： IBM Container Registry

- IBM MQ 9.3.0.0: `icr.io/ibm-messaging/mqmft:latest`

快速参考

- 许可证:
  - IBM MQ Advanced for Developers 和 Apache 许可证 2.0。 IBM MQ Advanced for Developers 许可证不允许进一步分发，其条款限制只能在开发人员的机器上使用。
- 在哪里提出问题:
  - GitHub
- 适用于以下 CPU 架构
  - amd64

构建容器镜像

请参阅使用文档，在 amd64 架构上使用 MQ Managed File Transfer Redistributable 软件包构建自己的代理容器映像。

用法

有关如何使用 Podman runtime 运行映像容器的详细信息，请参阅使用文档。

有关如何在 OpenShift Container Platform 中部署映像的详细信息，请参阅使用文档。

要使用该镜像，必须通过设置 **LICENSE** 环境变量来接受 IBM MQ 许可证的条款。

环境变量

许可证
设置**接受**表示同意 IBM MQ Advanced for Developers 许可条件。

设置**视图**以查看许可条件。

mft_agent_config_file
必需。

包含设置代理所需信息的 json 文件的路径。 路径必须位于挂载点上。 例如 `configMapOpenShift` 上。 有关属性的详细说明，请参阅附加 MFT 代理配置文件。

MFT_AGENT_NAME
指定管理员用户的密码。

要配置的代理名称。

BFG_JVM_PROPERTIES
可选。

运行代理 JVM 时需要设置的任何 JVM 属性。

MFT_LOG_LEVEL
可选。

显示信息的级别。 支持 `info` 和 `verbose` 值， 默认值为 `info`。 如果 MFT_LOG_LEVEL 设置为 `verbose` ，则会显示代理 `output0.log` 的内容。

mft_agent_start_wait_time
可选。

`fteStartAgent` 命令发出后，代理可能需要一些时间才能启动。 这是容器等待代理启动的时间（以秒为单位）。 如果代理没有在指定的等待时间内完成，容器就会终止。

MFT_MOUNT_PATH
可选。

环境变量，指向代理读取文件或写入文件的路径。

mft_coord_qmgr_cipher
用于安全连接协调队列管理器的 CipherSpec 名称。

mft_cmd_qmgr_cipher
用于安全连接命令队列管理器的 CipherSpec 名称。

mft_agent_qmgr_cipher
用于安全连接代理队列管理器的 CipherSpec 名称。

有关 IBM MQ Advanced for Developers 映像支持的默认开发人员配置的更多信息，请参阅默认开发人员配置文档。

代理配置文件的位置

容器中的代理会在固定目录 `/mnt/mftdata` 下创建代理配置和日志文件。 该文件夹也可以位于持久卷上，在这种情况下，必须将该卷作为 `/mnt/mftdata` 挂载点挂载到容器中。

OpenShift CP4I 在 "Red Hat OpenShift 上进行空隙安装和升级的 IBM MQ CASE

将 "IBM MQ Operator 和 "IBM MQ 队列管理器安装到空气屏蔽的 "Red Hat OpenShift 群集时，使用 "IBM"MQ""CASE。 IBM MQ CASE 包含构成特定 IBM MQ CASE 版本的所有映像和元数据的详细信息。

空气间隙和 IBM MQ CASE 3.5.0 变化

从 IBM MQ CASE 3.5.0 开始， IBM MQ CASE 中仅包含单个队列管理器版本的图像。 这是队列管理器的最新版本，相应的 IBM MQ Operator 可以部署它。 在 IBM MQ CASE 3.5.0 之前， CASE 包含相应操作员可以部署的所有队列管理器版本。 查看 IBM MQ CASE 查询表 ，了解每个 IBM MQ CASE 的确切版本。

您可以使用 IBM MQ CASE 来安装或升级到操作员可以部署的最新队列管理器；即与 CASE 和相应的操作员同时发布的 CD 或 SC2 队列管理器。 执行此操作时，安装或升级过程保持不变，但执行速度更快，因为镜像的图像更少。

但是，如果您安装的队列管理器版本不是操作员可以部署的最新版本，则必须为每个所需的队列管理器版本重复使用单独的 IBM MQ CASE 进行气隙镜像镜像过程。 查看 IBM MQ CASE 查询表 ，了解每个 IBM MQ CASE 的确切版本。

注: IBM MQ Operator 仍然支持旧版本的队列管理器，因此仍然可以要求部署旧版本的队列管理器。 因此，您需要考虑镜像哪些队列管理器版本。

队列管理器相关图像

从 IBM MQ CASE 3.5.0 开始， IBM MQ CASE 中仅包含单个最新队列管理器版本的图像，供相应的操作员部署。 在此之前， CASE 包含相应操作员可以部署的所有队列管理器版本。

注: 从 3.5.0、 IBM MQ CASE 不再包括操作面板图像，因为最新的队列管理器版本不支持操作面板。

使用 IBM MQ CASE 查找表来确定哪个 IBM MQ CASE 包含您所需的 IBM MQ 队列管理器版本。

操作员相关图像

请使用 IBM MQ CASE 查找表来确定哪个 IBM MQ CASE 包含您所需的 IBM MQ Operator 版本。

要点:

- 在气隙环境中安装 IBM MQ Operator 时，请务必完成与要安装的 IBM MQ Operator 版本对应的 IBM MQ CASE 的气隙镜像过程。 请勿从其他 IBM MQ CASE 安装或升级到操作员版本。
- IBM MQ CASE 包含其他 IBM MQ Operator 版本，但只能安装与您镜像的 IBM MQ CASE 对应的操作员。
- 升级 IBM MQ Operator 时，如果您要更改运营商频道，请遵循气隙升级最佳实践，升级到当前频道上可用的最新 IBM MQ Operator 版本。 在更改订阅新的 IBM MQ Operator 频道之前，请务必执行此操作。

IBM MQ CASE 查找表

表 *3: IBM MQ* 最近 *CD* 实例的内容 *CASE*

IBM MQ CASE	**IBM MQ 操作员**	**IBM MQ 队列管理器**
3.6.0	3.6.0	9.4.3.0-r1
3.5.3	3.5.3	9.4.2.1-r2
3.5.2	3.5.2	9.4.2.1-r1
3.5.1	3.5.1	9.4.2.0-r2
3.5.0	3.5.0	9.4.2.0-r1

表 3: *IBM MQ* 最近 *CD* 实例的内容 *CASE (*继续*)*

IBM MQ CASE	**IBM MQ 操作员**	**IBM MQ 队列管理器**
3.4.1	3.4.1	9.4.1.1-r1, 9.4.1.0-r2, 9.4.1.0-r1, 9.4.0.7-r1, 9.4.0.6-r2, 9.4.0.6-r1, 9.4.0.5-r2, 9.4.0.5-r1, 9.4.0.0-r3, 9.4.0.0-r2, 9.4.0.0-r1, 9.3.5.1-r2, 9.3.5.1-r1, 9.3.5.0-r2, 9.3.5.0-r1, 9.3.4.1-r1, 9.3.4.0-r1, 9.3.3.3-r2, 9.3.3.3-r1, 9.3.3.2-r3, 9.3.3.2-r2, 9.3.3.2-r1, 9.3.3.1-r2, 9.3.3.1-r1, 9.3.3.0-r2, 9.3.3.0-r1, 9.3.2.1-r2, 9.3.2.1-r1, 9.3.2.0-r2, 9.3.2.0-r1, 9.3.1.1-r1, 9.3.1.0-r3, 9.3.1.0-r2, 9.3.1.0-r1, 9.3.0.25-r1, 9.3.0.21-r3, 9.3.0.21-r2, 9.3.0.21-r1, 9.3.0.20-r2, 9.3.0.20-r1, 9.3.0.17-r3, 9.3.0.17-r2, 9.3.0.17-r1, 9.3.0.16-r2, 9.3.0.16-r1, 9.3.0.15-r1, 9.3.0.11-r2, 9.3.0.11-r1, 9.3.0.10-r2, 9.3.0.10-r1, 9.3.0.6-r1, 9.3.0.5-r3, 9.3.0.5-r2, 9.3.0.5-r1, 9.3.0.4-r2, 9.3.0.4-r1, 9.3.0.3-r1, 9.3.0.1-r4, 9.3.0.1-r3, 9.3.0.1-r2, 9.3.0.1-r1, 9.3.0.0-r3, 9.3.0.0-r2, 9.3.0.0-r1

表 *4: SC2 IBM MQ* 最近的内容 *CASE*		
IBM MQ CASE	**IBM MQ 操作员**	**IBM MQ 队列管理器**
3.2.13	3.2.13	9.4.0.11-r3, 9.4.0.11-r2, 9.4.0.11-r1, 9.4.0.10-r2, 9.4.0.10-r1, 9.4.0.7-r1, 9.4.0.6-r2, 9.4.0.6-r1, 9.4.0.5-r2, 9.4.0.5-r1, 9.4.0.0-r3, 9.4.0.0-r2, 9.4.0.0-r1, 9.3.5.1-r2, 9.3.5.1-r1, 9.3.5.0-r2, 9.3.5.0-r1, 9.3.4.1-r1, 9.3.4.0-r1, 9.3.3.3-r2, 9.3.3.3-r1, 9.3.3.2-r3, 9.3.3.2-r2, 9.3.3.2-r1, 9.3.3.1-r2, 9.3.3.1-r1, 9.3.3.0-r2, 9.3.3.0-r1, 9.3.2.1-r2, 9.3.2.1-r1, 9.3.2.0-r2, 9.3.2.0-r1, 9.3.1.1-r1, 9.3.1.0-r3, 9.3.1.0-r2, 9.3.1.0-r1, 9.3.0.25-r1, 9.3.0.21-r3, 9.3.0.21-r2, 9.3.0.21-r1, 9.3.0.20-r2, 9.3.0.20-r1, 9.3.0.17-r3, 9.3.0.17-r2, 9.3.0.17-r1, 9.3.0.16-r2, 9.3.0.16-r1, 9.3.0.15-r1, 9.3.0.11-r2, 9.3.0.11-r1, 9.3.0.10-r2, 9.3.0.10-r1, 9.3.0.6-r1, 9.3.0.5-r3, 9.3.0.5-r2, 9.3.0.5-r1, 9.3.0.4-r2, 9.3.0.4-r1, 9.3.0.3-r1, 9.3.0.1-r4, 9.3.0.1-r3, 9.3.0.1-r2, 9.3.0.1-r1, 9.3.0.0-r3, 9.3.0.0-r2, 9.3.0.0-r1

先前的操作员已删除

作为参考，从 IBM MQ CASE 3.4.0 ， 以下频道的 IBM MQ Operator 图片已从 IBM MQ CASE 中删除：

- v1.1
- v1.2
- v1.4
- v1.5
- v1.6
- v1.7
- v2.1

重新创建群集和图像注册表

您可以使用以下命令列出部署在 Red Hat OpenShift 集群中的所有队列管理器及其版本。

```
oc get queuemanager --all-namespaces -o=jsonpath='{range .items[*]}{.metadata.name}{":\t"}
{.spec.version}{"\n"}{end}'
```

您应该定期运行此命令并保存输出。 如果您想重新创建 Red Hat OpenShift 集群或镜像空隙的镜像图像注册表，此列表非常有用。

相关任务

第 67 页的『将 "CD"IBM MQ Operator"升级为当前的 "CD 通道』
将 "CD 部署升级为**只**使用 "IBM MQ 许可证的 "IBM MQ Operator 部署。

第 66 页的『将 3.2.x 的 "SC2"IBM MQ Operator 升级到当前的 "CD 通道』

将 3.2.x 的 "SC2 部署升级到**只**使用 "IBM MQ 许可证的 "IBM MQ Operator。

第 51 页的『安装 IBM MQ Operator』
可以使用 OpenShift 控制台或命令行界面 (CLI) 将 IBM MQ Operator 安装到 Red Hat OpenShift 上。

第 57 页的『安装 IBM MQ Operator 以用于 CP4I』
要与 IBM Cloud Pak for Integration (CP4I) 配合使用，可以通过 OpenShift 控制台或命令行界面 (CLI) 将 IBM MQ Operator 安装到 Red Hat OpenShift 上。

声明

本信息是为在美国国内供应的产品和服务而编写的。

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暗示只能使用 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，则由用户自行负责。

IBM 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。 您可以用书面方式将许可查询寄往：

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：

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, 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 可以随时对本资料中描述的产品和/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。 所有这些名字都是虚构的，若现实生活中实际业务企业使用的名字和地址与此相似，纯属巧合。

版权许可：

本信息包括源语言形式的样本应用程序，这些样本说明不同操作平台上的编程方法。 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 IBM 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 The 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: