9.1

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 47 页的『声明』中的信息。

本版本适用于 IBM® MQ V 9 发行版 1 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, 2024.

内容

Linux 容器中的 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。

您可以在 IBM MQ Advanced 和 IBM MQ Advanced for Developers 中提供的预先打包的容器中运行 IBM MQ 。 此 IBM MQ Advanced certified container 提供了受支持的映像和 Helm Chart ， 可用于将生产就绪 IBM MQ 映像部署到 Red Hat® OpenShift®， IBM Cloud Private 或 IBM Cloud Kubernetes Service 中。

您还可以在 IBM Cloud Pak for Integration 容器或您自己构建的容器中运行 IBM MQ 。

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

CD Linux MQ Adv. 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，您应该先熟悉基本 IBM MQ 概念 (请参阅 IBM MQ 技术概述) 以及基本 Kubernetes/OpenShift 概念 (请参阅 OpenShift Container Platform 体系结构)。

过程

- 第 5 页的『选择要在容器中使用 IBM MQ 的方式』.
- 第 9 页的『IBM MQ Advanced certified container 的高可用性』.
- 第 11 页的『IBM MQ Advanced certified container 的用户认证和授权』.

CD Linux MQ Adv. 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: 您可以选择使用预先打包的认证容器，也可以构建自己的映像和部署代码。

使用 IBM MQ Advanced 认证的容器

如果您计划在 Red Hat OpenShift Container Platform 上进行部署，那么可能要使用经过认证的容器。 有三个品种的认证容器:

- IBM Cloud Pak for Integration 的 IBM MQ Advanced certified container 。 这是单独的 IBM 产品，其中包含经过认证的容器的版本。
- IBM MQ Advanced certified container
- IBM MQ Advanced for Developers 认证容器 (无保证)

IBM MQ 9.1.4 和更低版本的 CD 发行版在 IBM Cloud Private 和 IBM Cloud Kubernetes Service 上也受支持。

请注意，已认证的容器正在快速发展，因此仅在 Continuous Delivery 发行版下受支持。

经过认证的容器包含预先构建的容器映像以及用于在 Red Hat OpenShift Container Platform 上运行的部署代码。 从 IBM MQ 9.1.5 开始，使用 IBM MQ 操作程序来管理队列管理器。 IBM MQ 的先前版本 (最高版本和包括版本 9.1.5) 是使用 Helm Chart 进行管理的。

使用经过认证的容器时，不支持某些 IBM MQ 功能部件。 如果要执行以下任何操作，您将需要构建自己的图像和图表:

- 使用 REST API 进行管理或消息传递

© Copyright IBM Corp. 2007, 2024

- 使用以下任何 MQ 组件:
  - Managed File Transfer 代理程序及其资源。 但是，您可以使用经过认证的容器来提供一个或多个协调，命令或代理队列管理器。
  - AMQP
  - IBM MQ Bridge to Salesforce
  - IBM MQ Bridge to blockchain (在容器中不受支持)
- 使用 Helm Chart 进行部署时使用 Web 服务器 (IBM Cloud Pak for Integration 除外)
- 定制用于 **crtmqm**， **strmqm** 和 **endmqm** 的选项，例如配置恢复日志

构建自己的图像和图表

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

提供了用于构建您自己的映像的样本。 请参阅 第 30 页的『构建您自己的 IBM MQ 容器』。 作为认证容器的一部分提供的 Helm 图表发布在 GitHub 上，并且可以用作构建您自己的映像时的样本:

- IBM MQ Advanced certified container 的 Helm 图表
- IBM MQ Advanced for Developers 认证容器的 Helm Chart

相关概念

第 6 页的『支持 IBM MQ 认证容器』
IBM MQ 认证的容器仅在某些 Kubernetes 环境中受支持

第 8 页的『支持构建您自己的 IBM MQ 容器映像和图表』
在 Linux 系统上使用容器时要考虑的信息。

Linux 支持 IBM MQ 认证容器

IBM MQ 认证的容器仅在某些 Kubernetes 环境中受支持

CD V 9.1.4 MQ Adv. 对于 CD 发行版 V9.1.4 和更高版本，支持将 IBM MQ Advanced certified container 与 Red Hat OpenShift 配合使用。 请参阅第 22 页的『使用 Helm CLI 部署队列管理器』。

在以下 Kubernetes 环境中支持低于 V9.1.4 的 CD 发行版:

- IBM Cloud Kubernetes Service
- IBM Cloud Private
- 带 Red Hat OpenShift 的 IBM Cloud Private

有关 Kubernetes 的特定受支持版本，请参阅已下载的 IBM MQ Advanced Helm Chart 中的文件 `qualification.yaml` 和 `Chart.yaml` 。 这些版本因发行版而异。

仅当使用 IBM MQ 操作程序进行部署或使用以下某个 Helm Chart 时，才支持 IBM MQ Advanced certified container :

- `ibm-mqadvanced-server-prod`
- `ibm-mqadvanced-server-integration-prod` 在 IBM Cloud Pak for Integration

注: 在 IBM MQ 操作程序发行版之后，不推荐使用 Helm Chart。

由于容器技术正在快速发展，因此仅在此 Chart 在发布时支持的最新版本的平台上支持 IBM MQ Advanced certified container 。 如果要使用较旧的平台版本，那么可能需要使用较旧的 IBM MQ Advanced certified container 版本。

IBM MQ Advanced certified container 映像基于 IBM MQ Continuous Delivery (CD) 发行版。 这些版本最多支持一年，或支持两个 CD 发行版，以时间较长者为准。 IBM MQ 的 Long Term Support 发行版不可用作经过认证的容器。

从 IBM MQ Advanced certified container V4.0 开始，该映像提供了 Red Hat Universal Base Image (UBI) 上 IBM MQ 的安装，其中包含 IBM MQ 所使用的关键 Linux 库和实用程序。 在 Red Hat Enterprise Linux 主机上运行时， Red Hat 支持 UBI。 较早版本的 IBM MQ Advanced certified container 使用了不受支持的 Ubuntu 基本映像。

相关概念

第 8 页的『支持构建您自己的 IBM MQ 容器映像和图表』
在 Linux 系统上使用容器时要考虑的信息。

CD Linux MQ Adv. IBM MQ Advanced certified container 的版本支持

一组表，显示 IBM MQ Advanced certified container， IBM MQ， IBM Cloud Kubernetes Service， IBM Cloud Pak for Integration 和 IBM Cloud Private 的受支持版本之间的映射。

IBM MQ 操作员

V 9.1.5

支持将 IBM MQ 操作程序用作 IBM Cloud Pak for Integration V 2020.2 的一部分，或者将其单独用于 IBM MQ V 9.1.5 及更高版本。

IBM MQ Operator 在 Red Hat OpenShift Container Platform V 4.4 或更高版本上受支持。

IBM MQ Advanced certified container V 9.1.5 (Helm Chart)-不推荐使用

包含 Helm Chart `ibm-mqadvanced-server-prod`。

V 9.1.5 从 IBM MQ Advanced certified container V5.0.x 开始， Helm Chart ， 映像和修订通过 IBM 授权目录和注册表交付。 先前版本通过 Passport Advantage 提供，修订发行版可从 IBM Fix Central 获取。

表 *1:* 对 *IBM MQ Advanced certified container* 的支持

版本	IBM MQ 版本	终止支持	支持的平台
6.0.x	9.1.5 Continuous Delivery 发行版	2021 年 3 月	详细系统需求
5.0.x	9.1.4 Continuous Delivery 发行版	2020 年 12 月	详细系统需求
4.1.x	9.1.3 Continuous Delivery 发行版	2020 年 7 月	详细系统需求

IBM MQ Advanced certified container IBM Cloud Pak for Integration V 9.1.5 (Helm Chart) 软件-不推荐使用

包含 Helm Chart `ibm-mqadvanced-server-integration-prod`。

表 *2:* 对 *IBM Cloud Pak for Integration* 的 *IBM MQ Advanced certified container* 软件的版本支持

版本	IBM MQ 版本	IBM Cloud Pak for Integration 版本
6.0.x	9.1.4 Continuous Delivery 发行版	2020.1.1 (系统需求)
5.0.x	9.1.3 Continuous Delivery 发行版	2019.4.1 (系统需求)

表 2: 对 *IBM Cloud Pak for Integration* 的 *IBM MQ Advanced certified container* 软件的版本支持 (继续)		
版本	**IBM MQ 版本**	**IBM Cloud Pak for Integration 版本**
4.1.x	9.1.3 Continuous Delivery 发行版	2019.3.2.2 (系统需求)
4.0.x	9.1.3 Continuous Delivery 发行版	2019.3.2 (系统需求)
3.0.x	9.1.3 Continuous Delivery 发行版	2019.3.1 (系统需求)

请参阅 IBM Cloud Pak for Integration 发行说明 以获取受支持的版本信息。

Linux 支持构建您自己的 IBM MQ 容器映像和图表

在 Linux 系统上使用容器时要考虑的信息。

- 容器映像使用的基本映像必须使用受支持的 Linux 操作系统。
- 必须使用 IBM MQ 安装程序在容器映像中安装产品。
- 有关受支持软件包的列表，请参阅 Linux 系统的 IBM MQ rpm 组件。
- V 9.1.0 以下软件包不受支持:
  - `MQSeriesBCBridge`
  - `MQSeriesRDQM`
- 队列管理器数据目录 (缺省情况下为`/var/mqm`) 必须存储在保持持久状态的容器卷上。

  要点: 不能使用并集文件系统。

  必须将主机目录安装为数据卷，或者使用数据卷容器。 有关更多信息，请参阅 管理容器中的数据。
- 您必须能够在容器中运行 IBM MQ 控制命令，例如 **endmqm**。
- 您必须能够从容器中获取文件和目录以进行诊断。
- V 9.1.0 您可以使用名称空间来与其他容器共享队列管理器的容器名称空间，以便在本地将应用程序绑定到在不同容器中运行的队列管理器。 有关更多信息，请参阅第 33 页的『在单独的容器中运行本地绑定应用程序』。

相关概念

第 6 页的『支持 IBM MQ 认证容器』
IBM MQ 认证的容器仅在某些 Kubernetes 环境中受支持

CD V 9.1.5 Linux MQ Adv. IBM MQ Advanced certified container 的存储注意事项

IBM MQ Advanced certified container 以两种存储方式运行:

- 当容器重新启动时可以处理所有容器状态时，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 操作程序提供了定制存储器特征的功能，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，那么重新启动时将丢失所有队列管理器状态。 这包括:

- 全部消息
- 所有队列管理器到队列管理器的通信状态 (通道消息序号)

- 队列管理器的 MQ 集群身份
- 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 19 页的『使用 OpenShift CLI 使用定制 MQSC 和 INI 文件构建映像』)。 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 。

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

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

- spec.queueManager.availability 控制可用性方式。 如果您使用的是 `SingleInstance` ，那么仅需要 `ReadWriteOnce` 存储器，而 `multiInstance` 需要具有正确文件锁定特征的支持 `ReadWriteMany` 的存储类。 IBM MQ 提供了 支持声明 和 测试声明。 可用性方式还会影响持久卷布局。 有关更多信息，请参阅 第 9 页的『IBM MQ Advanced certified container 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:
        enabled: true
        class: ibmc-file-gold-gid
          persistedData:
            enabled: true
            class: ibmc-file-gold-gid
          recoveryLogs:
            enabled: true
            class: ibmc-file-gold-gid
     securityContext:
       supplementalGroups: [99]
```

CD Linux MQ Adv. IBM MQ Advanced certified container 的高可用性

对于 IBM MQ Advanced certified container 的高可用性，您有两个主要选项: **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) 和 **单个弹性队列管理器** (提供了使用网络存储器的简单 HA 方法)。

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。

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。 要对此进行配置，请使用设置为 `ReadWrite` **多项**的 Kubernetes 持久卷 **`access mode`** 。 这些卷还必须满足 IBM MQ 共享文件系统的需求，因为 IBM MQ 依赖于自动释放文件锁定来启动队列管理器故障转移。 IBM MQ 生成 已测试文件系统列表。

多实例队列管理器的恢复时间由以下因素控制:

1. 发生故障后，共享文件系统释放活动实例最初获取的锁定所需的时间。
2. 备用实例获取锁定然后启动所需的时间。
3. Kubernetes Pod 就绪性探测器检测容器是否就绪所需的时间。 这可在 Helm 图表中进行配置。
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。 这可在 Helm 图表中进行配置。
2. Kubernetes 调度程序将失败的 Pod 重新调度到新节点所需的时间。
3. 将容器映像下载到新节点所需的时间。 如果使用 **`imagePullPolicy`** 值 `IfNotPresent`，那么该映像可能已在该节点上可用。
4. 启动新队列管理器实例所需的时间。
5. Kubernetes Pod 就绪性探测器检测容器是否就绪所需的时间。 这可在 Helm 图表中进行配置。
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。 如果 Kubernetes 主节点与工作程序节点失去联系，那么它无法确定该节点是否已发生故障，或者它是否只是失去了网络连接。 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。

相关概念
高可用性配置

CD Linux MQ Adv. IBM MQ Advanced certified container 的用户认证和授权

可以将 IBM MQ 配置为使用 LDAP 用户和组进行授权。 这是 IBM MQ Advanced certified container 的建议方法。

在多租户容器化环境 (例如 Red Hat OpenShift Container Platform) 中，设置了安全性约束以防止潜在的安全性问题。 例如，在 Red Hat OpenShift Container Platform 中，缺省 `SecurityContextConstraints` (称为 `restricted`) 使用随机用户标识，从而阻止容器本身本地的任何用户。 IBM MQ 通常使用特权升级来检查用户的密码，在多租户容器环境中也不建议这样做。 由于这些原因，在 IBM MQ 认证的容器中不支持使用在正在运行的容器内的操作系统库上定义的用户。

您需要配置队列管理器以使用 LDAP 进行用户认证和授权。 有关配置 IBM MQ 以执行此操作的信息，请参阅连接认证: 用户存储库 和 LDAP 授权

CD V 9.1.5 Linux MQ Adv. 在 OpenShift 上安装和卸载 IBM MQ 操作程序

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

开始之前

过程

- 第 12 页的『使用 OpenShift CLI 安装 IBM MQ 操作程序』.
- 第 11 页的『使用 OpenShift Web 控制台安装 IBM MQ 操作程序』.

CD V 9.1.5 Linux MQ Adv. 使用 OpenShift Web 控制台安装 IBM MQ 操作程序

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

开始之前
登录到 OpenShift 集群的 Web 控制台。

过程

1. 将 IBM Common Services 操作程序添加到可安装操作程序列表中
   a) 单击加号图标。 您会看到**导入 YAML** 对话框。
   b) 将以下资源定义粘贴到对话框中。

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

   c) 单击**创建**。

2. 将 IBM 操作程序添加到可安装操作程序列表
   a) 单击加号图标。 您会看到**导入 YAML** 对话框。
   b) 将以下资源定义粘贴到对话框中。

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

   c) 单击**创建**。
3. 创建要用于 IBM MQ 操作程序的名称空间
   可以将 IBM MQ 操作程序安装到单个名称空间或所有名称空间。 仅当要安装到尚不存在的特定名称空间时，才需要执行此步骤。
   a) 在导航窗格中，单击 **主页** > **项目**。
      此时将显示 "项目" 页面。
   b) 单击**创建项目**。 将显示 "创建项目" 区域。
   c) 输入正在创建的名称空间的详细信息。 例如，可以指定 "ibm-mq" 作为名称。
   d) 单击**创建**。 将创建 IBM MQ 操作程序的名称空间。
4. 安装 IBM MQ 操作程序。
   a) 在导航窗格中，单击 **操作程序** > **OperatorHub**。
      此时将显示 " OperatorHub " 页面。
   b) 在 **所有项** 字段中，输入 "IBM MQ"。
      这将显示 IBM MQ 目录条目。
   c) 选择 **IBM MQ**。
      此时将显示 " IBM MQ " 窗口。
   d) 单击**安装**。
      您将看到 "创建操作员预订" 页面。
   e) 将安装方式设置为您创建的特定名称空间或集群范围
   f) 单击**预订**。
      您将在 "已安装的操作程序" 页面上看到 IBM MQ 。
   g) 在 "已安装的操作程序" 页面上检查操作程序的状态，当安装完成时，状态将更改为 "已成功"。

下一步做什么

第 14 页的『部署 IBM MQ 认证的容器』

CD V 9.1.5 Linux MQ Adv. 使用 OpenShift CLI 安装 IBM MQ 操作程序

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

开始之前

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

过程

1. 为 IBM Common Services 操作程序创建 OperatorSource

   a) 创建用于定义 OperatorSource 资源的 YAML 文件

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

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

   b) 将 OperatorSource 应用于服务器。

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

2. 为 IBM 操作程序创建 OperatorSource

   a) 创建用于定义 OperatorSource 资源的 YAML 文件

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

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

   b) 将 OperatorSource 应用于服务器。

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

3. 创建要用于 IBM MQ 操作程序的名称空间

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

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

4. 从 OperatorHub 查看可用于集群的操作程序列表

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

5. 检查 IBM MQ 操作程序以验证其受支持的 InstallModes 和可用通道

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

6. 创建 OperatorGroup 对象 YAML 文件

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

   您预订操作程序的名称空间必须具有与操作程序的 InstallMode(AllNamespaces 或 SingleNamespace 方式) 匹配的 OperatorGroup 。 如果要安装的操作程序使用 AllNamespaces，那么 openshift-operators 名称空间已具有相应的 OperatorGroup 。

但是，如果操作程序使用 SingleNamespace 方式，并且尚未安装相应的 OperatorGroup，那么必须创建一个操作程序。

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

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

b) 创建 OperatorGroup 对象

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

7. 创建预订对象 YAML 文件以预订 MQ 操作程序的名称空间

a) 使用以下内容创建名为 "mq-sub.yaml" 的文件:

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

对于 AllNamespaces **InstallMode** 用法，请指定 openshift-operators 名称空间。 否则，请为 SingleNamespace **InstallMode** 用法指定相关的单个名称空间。

b) 创建 Subscription 对象

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

8. 检查操作员的状态

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

下一步做什么

第 14 页的『部署 IBM MQ 认证的容器』

CD Linux MQ Adv. 部署 IBM MQ 认证的容器

可以使用 IBM MQ 操作程序将 IBM MQ V 9.1.5 和更高版本部署到 Red Hat OpenShift。 可以使用 Helm 将 IBM MQ 版本 9.1.5 和 9.1.4 部署到 Red Hat OpenShift。 可以使用 Helm 将较早的 CD 版本部署到 IBM Cloud Private 集群或 IBM Cloud Kubernetes Service 集群。

关于此任务

过程

- 第 22 页的『使用 Helm CLI 部署队列管理器』.
- 第 24 页的『将 IBM MQ 的先前 CD 发行版部署到 IBM Cloud Private 集群』.
- 第 25 页的『将 IBM MQ 映像的先前 CD 发行版添加到 IBM Cloud Private 集群』.
- 第 26 页的『将 IBM MQ 映像的先前 CD 发行版添加到 IBM Cloud Kubernetes Service 集群』.

CD Linux MQ Adv. 使用 OpenShift CLI 为 IBM MQ 准备 OpenShift 项目

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

开始之前

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

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

关于此任务

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

过程

1. 获取分配给您的标识的权利密钥。
   a) 使用与授权软件关联的 IBM 标识和密码登录到 MyIBM Container Software Library 。
   b) 在**权利密钥**部分中，选择**复制密钥**以将权利密钥复制到剪贴板。
2. 在要部署队列管理器的项目中创建包含权利密钥的私钥。
   运行以下命令，其中 *<entitlement-key>* 是在步骤 1 中检索的密钥， *<user-email>* 是与授权软件关联的 IBM 标识。

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

下一步做什么

第 17 页的『使用 OpenShift CLI 部署队列管理器』

CD V 9.1.5 Linux MQ Adv. 使用 IBM Cloud Pak for Integration Platform Navigator 部署队列管理器

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

开始之前

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

如果这是首次将队列管理器部署到此 Red Hat OpenShift 项目中，请遵循 第 15 页的『使用 OpenShift CLI 为 IBM MQ 准备 OpenShift 项目』的步骤。

过程

1. 部署队列管理器。
   以下示例部署 "快速启动" 队列管理器，该队列管理器使用临时 (非持久) 存储器，并关闭 MQ 安全性。 消息不会在队列管理器重新启动时持久存储。 您可以调整配置以更改许多队列管理器设置。
   a) 在 IBM Cloud Pak for Integration Platform Navigator 中，单击 **运行时和实例**。
   b) 单击**创建实例**。

c) 选择 **队列管理器**，然后单击 **下一步**。

将显示用于创建 QueueManager 实例的表单。

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

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

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

您必须接受部署队列管理器的许可证。

f) 在 **队列管理器配置** 部分中，检查或更新底层队列管理器的 **名称** 。

缺省情况下， IBM MQ 客户机应用程序使用的队列管理器的名称将与 QueueManager 的名称相同，但除去了任何无效字符 (例如连字符)。 如果要强制使用特定名称，可以在此处对此进行编辑。

g) 单击**创建**

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

2. 检查队列管理器是否正在运行

当 QueueManager 状态为 Running 时，将完成创建。

相关任务

第 26 页的『连接到部署在 OpenShift 集群中的队列管理器』
一组用于连接到 Red Hat OpenShift 集群中部署的队列管理器的配置示例。

第 28 页的『连接到 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 。

CD V 9.1.5 Linux MQ Adv. 使用 OpenShift Web 控制台部署队列管理器

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

开始之前

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

如果这是首次将队列管理器部署到此 Red Hat OpenShift 项目中，请遵循 第 15 页的『使用 OpenShift CLI 为 IBM MQ 准备 OpenShift 项目』的步骤。

过程

1. 部署队列管理器。

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

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

b) 单击 **IBM MQ**。

c) 单击 **队列管理器** 选项卡。

d) 单击 **创建 QueueManager** 按钮。

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

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

e) 如果您接受许可协议，请将 **许可接受** 更改为 **开启**。

IBM MQ 在多个不同的许可证下可用。 有关有效许可证的更多信息，请参阅 第 35 页的『mq.ibm.com/v1beta1 的许可参考』。 您必须接受部署队列管理器的许可证。

f) 单击**创建**

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

2. 检查队列管理器是否正在运行

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

相关任务

第 26 页的『连接到部署在 OpenShift 集群中的队列管理器』
一组用于连接到 Red Hat OpenShift 集群中部署的队列管理器的配置示例。

第 28 页的『连接到 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 。

CD V 9.1.5 Linux MQ Adv.

使用 OpenShift CLI 部署队列管理器

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

开始之前

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

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

如果这是首次将队列管理器部署到此 Red Hat OpenShift 项目中，请遵循 第 15 页的『使用 OpenShift CLI 为 IBM MQ 准备 OpenShift 项目』的步骤。

过程

1. 部署队列管理器。

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

   a) 创建 QueueManager YAML 文件

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

```
apiVersion: mq.ibm.com/v1beta1
kind: QueueManager
metadata:
  name: quickstart-cp4i
spec:
  version: 9.1.5.0-r2
  license:
    accept: false
    license: L-RJON-BN7PN3
    use: NonProduction
  web:
    enabled: true
  queueManager:
    name: "QUICKSTART"
    storage:
      queueManager:
        type: ephemeral
  template:
    pod:
      containers:
       - name: qmgr
         env:
         - name: MQSNOAUT
           value: "yes"
```

   重要信息:如果您接受 IBM Cloud Pak for Integration 许可协议，请将 accept: false 更改为 accept: true。 请参阅 第 35 页的『mq.ibm.com/v1beta1 的许可参考』以获取有关许可证的详细信息。

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

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

```
apiVersion: mq.ibm.com/v1beta1
kind: QueueManager
metadata:
  name: quickstart
spec:
  version: 9.1.5.0-r2
  license:
    accept: false
    license: L-APIG-BM7GDH
    use: Development
  web:
    enabled: true
  queueManager:
    name: "QUICKSTART"
    storage:
      queueManager:
        type: ephemeral
  template:
    pod:
      containers:
       - name: qmgr
         env:
         - name: MQSNOAUT
           value: "yes"
```

重要信息:如果您接受 MQ 许可协议，请将 `accept: false` 更改为 `accept: true`。 请参阅 第 35 页的『mq.ibm.com/v1beta1 的许可参考』 以获取有关许可证的详细信息。

b) 创建 `QueueManager` 对象

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

2. 检查队列管理器是否正在运行

   您可以通过运行以下命令来验证部署:

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

   ，然后检查状态。

   例如，运行

   ```
   oc describe queuemanager quickstart
   ```

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

相关任务

第 26 页的『连接到部署在 OpenShift 集群中的队列管理器』
一组用于连接到 Red Hat OpenShift 集群中部署的队列管理器的配置示例。

第 28 页的『连接到 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 。

CD V 9.1.4 Linux MQ Adv. 与 IBM Cloud Pak for Integration 操作仪表板集成

通过 IBM Cloud Pak for Integration 跟踪事务的能力由操作仪表板提供。

关于此任务

启用与 "操作仪表板" 的集成会将 MQ API 出口安装到队列管理器。 API 出口将向 "操作仪表板" 数据存储器发送有关流经队列管理器的消息的跟踪数据。

请注意，仅跟踪使用 MQ 客户机绑定发送的消息。

过程

1. 在启用跟踪的情况下部署队列管理器

   缺省情况下，禁用跟踪功能。

   如果要使用 IBM Cloud Pak for Integration Platform Navigator 进行部署，那么可以在部署时启用跟踪，方法是将 **启用跟踪** 设置为 **开启**，并将 **跟踪名称空间** 设置为安装了操作仪表板的名称空间。 有关部署队列管理器的更多信息，请参阅 第 15 页的『使用 IBM Cloud Pak for Integration Platform Navigator 部署队列管理器』

   如果要使用 OpenShift CLI 或 OpenShift Web 控制台进行部署，那么可以使用以下 YAML 片段来启用跟踪:

   ```
   spec:
     tracing:
       enabled: true
       namespace: <Operations_Dashboard_Namespace
   ```

   如果要使用 Helm 进行部署，那么可以通过设置 `odTracingConfig.enabled=true` 和 `odTracingConfig.odTracingNamespace=<Operations_Dashboard_Namespace` 来启用跟踪。 如果要在现有队列管理器上启用 "操作仪表板" 集成，那么可以在升级 Helm 发行版期间应用此设置。

   重要信息: 在向 "操作仪表板" 注册 MQ 之前，队列管理器将不会启动 (请参阅下一步)。

   请注意，启用此功能后，除队列管理器容器外，它还将运行两个侧柜容器 ("代理程序" 和 "收集器")。 这些侧柜容器的映像将在与主 MQ 映像相同的注册表中可用，并且将使用相同的拉取策略和拉取私钥。 有其他设置可用于配置 CPU 和内存限制。
2. 如果这是首次在此名称空间中部署具有 "操作仪表板" 集成的队列管理器，那么需要向 "操作仪表板" 注册 。

   注册将创建一个密钥对象，队列管理器 Pod 需要该对象才能成功启动。

CD V 9.1.5 Linux MQ Adv. 使用 OpenShift CLI 使用定制 MQSC 和 INI 文件构建映像

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 Red Hat OpenShift 项目中没有 IBM Entitled Registry 的 OpenShift 私钥，请遵循 第 15 页的『使用 OpenShift CLI 为 IBM MQ 准备 OpenShift 项目』的步骤。

过程

1. 创建 `ImageStream`

   映像流及其关联标记提供了用于从 Red Hat OpenShift Container Platform 中引用容器映像的抽象。 映像流及其标记允许您查看可用的映像，并确保您正在使用所需的特定映像，即使存储库中的映像发生更改也是如此。

   ```
   oc create imagestream mymq
   ```
2. 为新映像创建 `BuildConfig`

   `BuildConfig` 将允许构建新映像，该映像将基于 IBM 官方映像，但将添加要在容器启动时运行的任何 MQSC 或 INI 文件。

   a) 创建用于定义 `BuildConfig` 资源的 YAML 文件

   例如，使用以下内容创建名为 "mq-build-config.yaml" 的文件:

```
apiVersion: build.openshift.io/v1
kind: BuildConfig
metadata:
  name: mymq
spec:
  source:
    dockerfile: |-
      FROM cp.icr.io/cp/ibm-mqadvanced-server-integration:9.1.5.0-r2-amd64
      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:9.1.5.0-r2-amd64"
      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。 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 17 页的『使用 OpenShift CLI 部署队列管理器』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 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 17 页的『使用 OpenShift CLI 部署队列管理器』
使用 QueueManager 定制资源，通过命令行界面 (CLI) 将队列管理器部署到 Red Hat OpenShift Container Platform 集群上。 此任务应由项目管理员完成

CD Linux MQ Adv. 使用 Helm 部署 IBM MQ 认证的容器

从 IBM MQ 9.1.5.0 开始，建议使用 IBM MQ 操作程序来部署队列管理器。 可以使用 Helm 通过以下指示信息来部署 IBM MQ 9.1.5.0 和先前 CD 发行版。

关于此任务

过程

- 第 21 页的『使用 Helm 在 OpenShift 上为 IBM MQ 准备 OpenShift 集群』.
- 第 22 页的『使用 Helm CLI 部署队列管理器』.

CD Linux MQ Adv. 使用 Helm 在 OpenShift 上为 IBM MQ 准备 OpenShift 集群

准备 Red Hat OpenShift Container Platform 集群，以便它可以使用 Helm 部署队列管理器。 此任务应由集群管理员完成。

开始之前

注: 如果您正在使用 IBM Cloud Pak for Integration，那么安装程序应该已准备好 OpenShift 项目 (名称空间) 以供您与 IBM MQ 配合使用，因此您可能不需要遵循这些指示信息。

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

过程

1. 确保已将 IBM Helm 存储库添加到 Helm 的本地副本。
   例如，可以运行以下命令:

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

2. 确保您具有 Helm 服务器 (称为 "Tiller") 安装在集群上。
   遵循 OpenShift 上的 Helm 入门 中的指示信息，在集群上安装 Helm 。
3. 确保 OpenShift 项目 (名称空间) 中的服务帐户有权使用正确的安全上下文约束 (SCC)。

   V 9.1.5 IBM MQ 在缺省 SCC "restricted" 下工作，因此通常可以跳过此步骤。

   应用对 SCC 的更改需要由 OpenShift 集群管理员完成。 每个 Helm Chart 版本都有不同的 SCC 需求，这些需求记录在该 Helm Chart 的各个自述文件中:

   ```
   helm inspect readme ibm-entitled-charts/ibm-mqadvanced-server-prod
   ```

   每个自述文件中都有有关设置 SCC 授权的指示信息。 请注意， IBM MQ Helm 图表创建服务帐户以供自己使用，这意味着需要在 "组" 级别 (针对名称空间中的所有服务帐户) 应用 SCC 许可权。
4. 确保您具有有效的 "映像拉取私钥" 以从所选容器注册表拉取映像
   将从执行许可证权利检查的容器注册表中拉取 IBM MQ Advanced certified container 映像。 此检查需要存储在 `docker-registry` 拉取私钥中的权利密钥。 如果您还没有权利密钥，请遵循以下指示信息以获取权利密钥并创建拉取私钥。
   a) 获取分配给您的标识的权利密钥。
      i) 使用与授权软件关联的 IBM 标识和密码登录到 MyIBM Container Software Library 。
      ii) 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。
   b) 在要在其中部署队列管理器的名称空间中创建私钥。

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

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

下一步做什么

第 22 页的『使用 Helm CLI 部署队列管理器』

使用 Helm CLI 部署队列管理器

使用 Helm 将队列管理器部署到 Red Hat OpenShift Container Platform 集群。 此任务应由项目管理员完成。

开始之前

您需要安装 Helm V2 和 Red Hat OpenShift Container Platform 命令行界面。 如果未使用 IBM Cloud Pak for Integration，请遵循 第 21 页的『使用 Helm 在 OpenShift 上为 IBM MQ 准备 OpenShift 集群』的步骤。

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

过程

1. 确保已将 IBM Helm 存储库添加到 Helm 的本地副本。
   例如，可以运行以下命令：

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

2. 查看队列管理器的配置选项

   部署步骤包括安装和配置步骤。 必须在部署时设置队列管理器的某些设置，而更改这些设置需要重新部署。

   您可以通过运行下列其中一个命令来查看 Helm Chart 自述文件，以获取所有可用部署选项的详细信息：

   - 对于 IBM Cloud Pak for Integration 中的 IBM MQ Advanced certified container :

     ```
     helm inspect readme ibm-entitled-charts/ibm-mqadvanced-server-integration-prod
     ```

   - 对于 IBM MQ Advanced certified container：

     ```
     helm inspect readme ibm-entitled-charts/ibm-mqadvanced-server-prod
     ```

   您通常至少需要以下参数：

   a. 发行版名称。 `my-release` 例如：
   b. 远程 Helm 存储库。 `ibm-entitled-charts` 例如：
   c. Helm Chart: 例如 `ibm-mqadvanced-server-prod` 或 `ibm-mqadvanced-server-integration-prod`
   d. 映像拉取私钥名称。 例如：`entitled-registry`。 请注意，如果要部署到 IBM Cloud Pak for Integration 中 MQ 的预定义项目中，那么不需要执行此操作

3. 部署队列管理器。

   请注意，缺省情况下， Helm Chart 假定您在 Red Hat OpenShift Container Platform 集群中设置了缺省存储类 。

例如，要在 IBM Cloud Pak for Integration 中安装基本队列管理器，请运行以下命令:

```
helm install \
--tls \
--name my-release \
ibm-entitled-charts/ibm-mqadvanced-server-integration-prod \
--set license=accept \
--set tls.hostname=my.cluster \
--set tls.generate=true
```

您可以在 `tls.hostname` 字段中输入任何主机名 (这是必填字段，但不会像在此示例中生成新的自签名证书一样使用)

要独立于 IBM Cloud Pak for Integration 安装基本队列管理器，可以运行以下命令:

```
helm install \
--name my-release \
ibm-entitled-charts/ibm-mqadvanced-server-prod \
--set license=accept \
--set image.pullSecret=ibm-entitlement-key
```

相关任务

第 26 页的『连接到部署在 OpenShift 集群中的队列管理器』
一组用于连接到 Red Hat OpenShift 集群中部署的队列管理器的配置示例。

第 28 页的『连接到 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 。

CD V 9.1.5 Linux MQ Adv. **使用 *Helm CLI* 使用 *IBM Cloud File Storage* 部署队列管理器**

使用 Helm 将队列管理器部署到使用 IBM Cloud File Storage 的 IBM Cloud 集群上的 Red Hat OpenShift 的示例方案。 此任务应由项目管理员完成

开始之前

您需要安装 Helm V2 和 Red Hat OpenShift Container Platform 命令行界面。 如果未使用 IBM Cloud Pak for Integration，请遵循 第 21 页的『使用 Helm 在 OpenShift 上为 IBM MQ 准备 OpenShift 集群』的步骤。

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

过程

1. 确保已将 IBM Helm 存储库添加到 Helm 的本地副本。
   例如，可以运行以下命令:

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

2. 部署队列管理器。

   使用 IBM Cloud File Storage 时，您通常会看到使用 `ibmc-file-gold-gid` 存储类的最佳结果。 此存储类允许用户在正确的文件系统组中写入存储器。

   例如，要在 IBM Cloud Pak for Integration 中安装基本队列管理器，请运行以下命令:

   ```
   helm install \
   --tls \
   --name my-release \
   ibm-entitled-charts/ibm-mqadvanced-server-integration-prod \
   --set license=accept \
   --set tls.hostname=my.cluster \
   --set tls.generate=true \
   --set dataPVC.storageClassName=ibmc-file-gold-gid \
   --set security.context.supplementalGroups={99}
   ```

   您可以在 `tls.hostname` 字段中输入任何主机名 (这是必填字段，但此处未使用此字段，因为在此示例中，我们将生成新的自签名证书)。

要独立于 IBM Cloud Pak for Integration 安装基本队列管理器，可以运行以下命令:

```
helm install \
--name my-release \
ibm-entitled-charts/ibm-mqadvanced-server-prod \
--set license=accept \
--set image.pullSecret=ibm-entitlement-key \
--set dataPVC.storageClassName=ibmc-file-gold-gid \
--set security.context.supplementalGroups={99}
```

相关任务

第 26 页的『连接到部署在 OpenShift 集群中的队列管理器』
一组用于连接到 Red Hat OpenShift 集群中部署的队列管理器的配置示例。

第 28 页的『连接到 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 。

CD Linux MQ Adv. 将 IBM MQ 的先前 CD 发行版部署到 IBM Cloud Private 集群

对于 IBM MQ 低于 9.1.4 的 CD 版本，请使用 IBM Cloud Private 管理控制台将队列管理器部署到 IBM Cloud Private 中。

开始之前

注意: V 9.1.4 此部署在 IBM MQ 9.1.4 或更高版本中不受支持。

此任务假定您已 将 IBM MQ 映像添加到 IBM Cloud Private 集群。

Helm Chart README.md 文件可从 IBM Cloud Private 目录条目中获取，该目录条目在您完成 此子步骤后显示，或者从命令行通过 将 IBM Cloud Private 的 **local-charts** 存储库添加为远程 Helm 存储库 并运行以下命令来显示:

```
helm inspect readme remote_repo_name/ibm-mqadvanced-server-prod
```

您必须具有支持必需安全上下文的 PodSecurity 策略或 SecurityContextConstraint (对于 Red Hat OpenShift 上的 IBM Cloud Private)。 可以从 Helm Chart README.md 文件中找到详细信息 (包括示例)。

还可以在 Helm Chart README.md 文件中找到有关如何配置 Helm 发行版的详细信息。

注:

- 如果要部署到缺省情况下不支持必需安全设置的 IBM Cloud Private 环境，请遵循 IBM Cloud Private 产品文档中的 部署需要在非缺省名称空间中提升特权的 Helm Chart 中的指示信息来启用部署。
- 如果您正在使用 SELinux ，那么必须满足 IBM MQ Support for SELinux on Red Hat Enterprise Linux 中描述的 IBM MQ 需求。

关于此任务

IBM Cloud Private 提供了用于管理本地容器化应用程序的平台。 将 IBM MQ 映像添加到 IBM Cloud Private 集群后，可以使用 IBM Cloud Private 管理控制台或命令行来部署队列管理器。

过程

- 使用 IBM Cloud Private 管理控制台
  a) 在 Web 浏览器中打开 IBM Cloud Private 管理控制台，然后单击 **目录**。

  请参阅 IBM Cloud Private 产品文档中的 使用管理控制台访问 IBM Cloud Private 集群 。
  b) 从列表中选择 ibm-mqadvanced-server-prod Chart。
  c) 选择 **配置**，然后完成以下配置步骤:

  a. 输入发布名称。

b. 阅读并接受许可协议。

c. 在 **dataPVC** 部分下，将 **storageclass** 设置为所需的存储类。 留空以选择缺省存储类。

d. 在 **映像** 部分下，将存储库设置为完整映像路径。 例如：

```
mycluster.icp:8500/namespace_name/ibm-mqadvanced-server-prod
```

e. 在 **image** 部分下，将标记设置为 image 标记。 例如：

```
9.1.3.0-r1
```

f. 如果需要 Kubernetes 拉取私钥来访问映像注册表，请将其添加为 **pullSecret**。

g. 在 **queueManager** 部分下，设置队列管理器的名称。

d) 单击 **安装** 以将队列管理器部署为 *Helm* 发行版。

- 使用命令行

a) 配置 **cloudctl** 以访问 IBM Cloud Private 集群。

请参阅 IBM Cloud Private 产品文档中的 安装 IBM Cloud Private CLI 。

b) 确保您已 添加 IBM Cloud Private 的 **local-charts** 存储库作为远程 Helm 存储库。

c) 安装 chart。

运行以下命令并指定这些参数:

a. 发行版名称 (例如 `my-release`)

b. 包含 `ibm-mqadvanced-server-prod` Chart 的远程 Helm 存储库的名称 (例如 `my-repo`)

c. 映像存储库 (例如 `mycluster.icp:8500/`*`namespace_name`*`/ibm-mqadvanced-server-prod`)

d. 图像标记 (例如 `9.1.3.0-r1`)

```
helm install --name my-release --repo my-repo ibm-mqadvanced-server-prod --set
license=accept --set image.repository=mycluster.icp:8500/namespace_name/ibm-mqadvanced-
server-prod --set image.tag=9.1.3.0-r1 --tls
```

相关任务

第 22 页的『使用 Helm CLI 部署队列管理器』
使用 Helm 将队列管理器部署到 Red Hat OpenShift Container Platform 集群。 此任务应由项目管理员完成。

第 25 页的『将 IBM MQ 映像的先前 CD 发行版添加到 IBM Cloud Private 集群』
对于 IBM MQ 低于 9.1.4 的 CD 版本，请准备 IBM Cloud Private 集群以部署 IBM MQ 的生产就绪映像。

第 26 页的『将 IBM MQ 映像的先前 CD 发行版添加到 IBM Cloud Kubernetes Service 集群』
对于 IBM MQ 低于 9.1.4 的 CD 版本，请将 IBM MQ 的生产就绪映像导入到 IBM Cloud Kubernetes Service 中。

CD Linux MQ Adv. 将 IBM MQ 映像的先前 CD 发行版添加到 IBM Cloud Private 集群

对于 IBM MQ 低于 9.1.4 的 CD 版本，请准备 IBM Cloud Private 集群以部署 IBM MQ 的生产就绪映像。

关于此任务

注意: V 9.1.4 此导入在 IBM MQ 9.1.4 或更高版本中不受支持。

您可以从 Passport Advantage 下载 IBM MQ 映像，并将其导入到 IBM Cloud Private 容器中。

过程

1. 从 Passport Advantage 和 Passport Advantage Express Web 站点下载最新的 IBM MQ 映像。

有关可用下载的详细信息，请转至 正在下载 IBM MQ 9.1 ，然后单击要下载的发行版的选项卡。 要下载的部件的名称和编号在表中列出。

2. 将下载的归档文件导入到 IBM Cloud Private 中。

   请参阅 IBM Cloud Private 产品文档中的 将 IBM 软件添加到 IBM Cloud Private 目录 。

下一步做什么

现在，您已准备好 将队列管理器部署到 IBM Cloud Private 中。

相关任务

第 22 页的『使用 Helm CLI 部署队列管理器』
使用 Helm 将队列管理器部署到 Red Hat OpenShift Container Platform 集群。 此任务应由项目管理员完成。

第 24 页的『将 IBM MQ 的先前 CD 发行版部署到 IBM Cloud Private 集群』
对于 IBM MQ 低于 9.1.4 的 CD 版本，请使用 IBM Cloud Private 管理控制台将队列管理器部署到 IBM Cloud Private 中。

第 26 页的『将 IBM MQ 映像的先前 CD 发行版添加到 IBM Cloud Kubernetes Service 集群』
对于 IBM MQ 低于 9.1.4 的 CD 版本，请将 IBM MQ 的生产就绪映像导入到 IBM Cloud Kubernetes Service 中。

CD Linux MQ Adv. 将 IBM MQ 映像的先前 CD 发行版添加到 IBM Cloud Kubernetes Service 集群

对于 IBM MQ 低于 9.1.4 的 CD 版本，请将 IBM MQ 的生产就绪映像导入到 IBM Cloud Kubernetes Service 中。

关于此任务

注意: V 9.1.4 此导入在 IBM MQ 9.1.4 或更高版本中不受支持。

您可以从 Passport Advantage 下载 IBM MQ 映像，并将其导入到 IBM Cloud Kubernetes Service 集群中。

过程

1. 从 Passport Advantage 和 Passport Advantage Express Web 站点下载最新的 IBM MQ 映像。

   有关可用下载的详细信息，请转至 正在下载 IBM MQ 9.1 ，然后单击要下载的发行版的选项卡。 要下载的部件的名称和编号在表中列出。

2. 将下载的归档文件导入到 IBM Cloud Kubernetes Service 中。

   请参阅 在公共 Kubernetes 容器中运行 IBM Cloud Private 映像。

相关任务

第 22 页的『使用 Helm CLI 部署队列管理器』
使用 Helm 将队列管理器部署到 Red Hat OpenShift Container Platform 集群。 此任务应由项目管理员完成。

第 24 页的『将 IBM MQ 的先前 CD 发行版部署到 IBM Cloud Private 集群』
对于 IBM MQ 低于 9.1.4 的 CD 版本，请使用 IBM Cloud Private 管理控制台将队列管理器部署到 IBM Cloud Private 中。

第 25 页的『将 IBM MQ 映像的先前 CD 发行版添加到 IBM Cloud Private 集群』
对于 IBM MQ 低于 9.1.4 的 CD 版本，请准备 IBM Cloud Private 集群以部署 IBM MQ 的生产就绪映像。

CD V 9.1.4 Linux MQ Adv. 连接到部署在 OpenShift 集群中的队列管理器

一组用于连接到 Red Hat OpenShift 集群中部署的队列管理器的配置示例。

关于此任务

您需要 OpenShift Route 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 服务器名称指示 (SNI) 仅在 TLS 协议中可用。 Red Hat OpenShift Container Platform 路由器使用 SNI 将请求路由到 IBM MQ 队列管理器。

OpenShift 路径的必需配置取决于客户机应用程序的 SNI 行为。

要将 SNI 头设置为 TLS 1.2 或更高版本，必须将 CipherSpec 或 CipherSuite 用于 TLS 通信。

如果满足以下条件，那么 SNI 将设置为 MQ 通道:

- IBM MQ C Client 是 V8 或更高版本。
- Java/JMS 客户机是 V9.1.1 或更高版本， Java 安装支持 `javax.net.ssl.SNIHostName` 类。
- .NET 客户机处于非受管方式。

如果提供了主机名作为连接名称，并且满足以下条件，那么 SNI 将设置为主机名:

- .NET 客户机处于受管方式。
- 使用 AMQP 或 XR 客户机。
- Java/JMS 客户机在 **AllowOutboundSNI** 设置为 NO 时使用。

SNI 未设置，在以下条件下为空白:

- IBM MQ C 客户机为 V7.5 或更低版本。
- IBM MQ C 客户机在 **AllowOutboundSNI** 设置为 NO 时使用。
- Java/JMS 客户机与不支持 `javax.net.ssl.SNIHostName` 类的 Java 安装配合使用。

示例

基于主机名的 OpenShift 路由: 对于将 SNI 设置为主机名的客户机应用程序

以下 Helm Chart 自动创建基于主机名的 OpenShift 路径，用于将应用程序连接到 IBM MQ 队列管理器。 将 SNI 设置为主机名的客户机应用程序可以使用此 OpenShift 路由。

- ibm-mqadvanced-server-dev
- ibm-mqadvanced-server-prod
- IBM Cloud Pak for Integration 中的 ibm-mqadvanced-server-integration-prod。

如果您未使用这些图表，并且需要创建自己的基于 OpenShift 路径的主机名，那么可以在集群中应用以下 `yaml`：

```
apiVersion: route.openshift.io/v1
  kind: Route
  metadata:
    name: <provide a unique name for the Route>
    namespace: <namespace of your MQ deployment>
  spec:
    to:
      kind: Service
      name: <name of the Kubernetes Service for your MQ deployment (for example "<Helm Release>-
ibm-mq")>
    port:
      targetPort: 1414
    tls:
      termination: passthrough
```

MQ 基于通道的 OpenShift 路由: 对于将 SNI 设置为 MQ 通道的客户机应用程序

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 OpenShift 路由。 您还必须在 Red Hat OpenShift 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。

要确定每个新 OpenShift 路由所需的主机名，需要将每个通道名称映射到 SNI 地址，如下所述: https://www.ibm.com/support/pages/ibm-websphere-mq-how-does-mq-provide-multiple-certificates-certlabl-capability

然后，必须通过在集群中应用以下 yaml 来创建新的 OpenShift 路径 (针对每个通道):

```
  apiVersion: route.openshift.io/v1
  kind: Route
  metadata:
    name: <provide a unique name for the Route>
    namespace: <the namespace of your MQ deployment>
  spec:
    host: <SNI address mapping for the channel>
    to:
      kind: Service
      name: <the name of the Kubernetes Service for your MQ deployment (for example "<Helm
Release>-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 "<Helm Release>-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 OpenShift Container Platform (OCP) 路由器使用的端口-通常为 443。

相关任务

第 22 页的『使用 Helm CLI 部署队列管理器』
使用 Helm 将队列管理器部署到 Red Hat OpenShift Container Platform 集群。 此任务应由项目管理员完成。

第 28 页的『连接到 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 。

CD V 9.1.4 Linux MQ Adv. 连接到 OpenShift 集群中部署的 IBM MQ Console

如何连接到已部署到 Red Hat OpenShift Container Platform 集群的队列管理器的 IBM MQ Console 。

关于此任务

如果您正在使用 IBM MQ 操作程序，那么可以在 OpenShift Web 控制台的 QueueManager 详细信息页面或 IBM Cloud Pak for Integration Platform Navigator 中找到 IBM MQ Console URL。 或者，可以通过运行以下命令从 OpenShift CLI 中找到此命令:

```
oc get queuemanager <QueueManager Name> -n <namespace of your MQ deployment> --output
jsonpath='{.status.adminUiUrl}'
```

示例

以下 Helm Chart 会自动创建用于访问 IBM MQ Console 的 OpenShift 路径

- ibm-mqadvanced-server-dev
- IBM Cloud Pak for Integration 中的 ibm-mqadvanced-server-integration-prod。

您可以通过运行以下命令来获取 OpenShift 路由的主机名:

```
oc get route <Route Name (for example "<Helm Release>-ibm-mq-web")>
-n <namespace of your MQ deployment> --output jsonpath='{.spec.host}'
```

您可以使用以下 URL 访问 IBM MQ Console :

```
https://<Route Hostname>/ibmmq/console
```

相关任务

第 22 页的『使用 Helm CLI 部署队列管理器』
使用 Helm 将队列管理器部署到 Red Hat OpenShift Container Platform 集群。 此任务应由项目管理员完成。

第 26 页的『连接到部署在 OpenShift 集群中的队列管理器』
一组用于连接到 Red Hat OpenShift 集群中部署的队列管理器的配置示例。

CD Linux MQ Adv. 使用 OpenShift CLI 备份和复原队列管理器配置

如果队列管理器配置丢失，那么备份队列管理器配置可帮助您根据其定义重建队列管理器。 此过程不会备份队列管理器日志数据。 由于消息的瞬态性质，在复原时，历史日志数据很可能不相关。

开始之前

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

过程

- 备份队列管理器配置。

  您可以使用 **dmpmqcfg** 命令来转储 IBM MQ 队列管理器的配置。

  a) 获取队列管理器的 pod 的名称。
  例如，如果您正在使用操作程序，那么可以运行以下命令，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
  ```

  例如，如果您正在使用 Helm，那么可以运行以下命令，其中 *release_name* 是 Helm 发行版的名称。

  ```
  oc get pods --selector release=release_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，如果您正在使用操作程序，那么可以运行以下命令，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
  ```

  例如，如果您正在使用 Helm，那么可以运行以下命令，其中 *release_name* 是 Helm 发行版的名称。

  ```
  oc get pods --selector release=release_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
  ```

构建您自己的 IBM MQ 容器

开发自构建容器，以前称为 “Docker 容器映像”。 这是最灵活的容器解决方案，但这需要您具备配置容器的强大技能，并 “拥有” 生成的容器。

开始之前

在开发自己的容器之前，请考虑是否可以改为使用 IBM 提供的其中一个预先打包的容器。 请参阅容器中的 IBM MQ

关于此任务

将 IBM MQ 打包为容器映像时，可以快速轻松地部署对应用程序的更改以测试和登台系统。 这可能是企业持续交付的主要优势。

过程

- 有关如何使用 Docker 构建 IBM MQ 容器映像的信息，请参阅以下子主题:
  - Linux 第 8 页的『支持构建您自己的 IBM MQ 容器映像和图表』
  - 第 30 页的『使用容器规划您自己的 IBM MQ 队列管理器映像』
  - 第 30 页的『使用 Docker 构建样本 IBM MQ 队列管理器映像』
  - 第 33 页的『在单独的容器中运行本地绑定应用程序』

相关概念

容器中的 IBM MQ

使用容器规划您自己的 IBM MQ 队列管理器映像

在容器中运行 IBM MQ 队列管理器时需要考虑多个需求。 样本容器映像提供了处理这些需求的方法，但是如果要使用您自己的映像，需要考虑如何处理这些需求。

过程监管

运行容器时，本质上是运行单个进程 (容器内的 PID 1) ，该进程稍后会衍生子进程。

如果主进程结束，那么容器运行时将停止该容器。 IBM MQ 队列管理器需要多个进程在后台运行。

因此，只要队列管理器正在运行，您就需要确保主进程保持活动状态。 最好通过执行管理查询等方法来检查此进程中的队列管理器是否处于活动状态。

填充 /var/mqm

容器必须以 /var/mqm 作为卷进行配置。

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。

V 9.1.0 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 。

使用 Docker 构建样本 IBM MQ 队列管理器映像

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 31 页的『构建样本基本 IBM MQ 队列管理器映像』
  - 第 31 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。

结果

现在，您已安装了 IBM MQ 的基本容器映像。

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。 要执行此操作，请使用通用映像作为父代来创建您自己的容器映像层。

开始之前

对于 IBM MQ 9.1 映像，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。 “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 9.1 RPM 的安装程序依赖于 `mqm` 用户和组，并且还在可执行程序上使用 `setuid` 位。

在 IBM MQ 9.2 中除去了此限制。

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 CHANNEL(PASSWORD.SVRCONN) CHLTYPE(SVRCONN)
SET CHLAUTH(PASSWORD.SVRCONN) TYPE(BLOCKUSER) USERLIST('nobody') +
DESCR('Allow privileged users on this channel')
SET CHLAUTH('*') TYPE(ADDRESSMAP) ADDRESS('*') USERSRC(NOACCESS) DESCR('BackStop rule')
SET CHLAUTH(PASSWORD.SVRCONN) TYPE(ADDRESSMAP) ADDRESS('*') USERSRC(CHANNEL) CHCKCLNT(REQUIRED)
ALTER AUTHINFO(SYSTEM.DEFAULT.AUTHINFO.IDPWOS) AUTHTYPE(IDPWOS) ADOPTCTX(YES)
REFRESH SECURITY TYPE(CONNAUTH)
```

   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
RUN useradd johndoe -G mqm && \
    echo johndoe:passw0rd | chpasswd
COPY config.mqsc /etc/mqm/
```

   其中:

   - `johndoe` 是要添加的用户标识
   - `passw0rd` 是原始密码

3. 使用以下命令构建定制容器映像:

```
sudo 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。

RUN 命令使用密码 `passw0rd` 添加名为 `johndoe` 的用户， **COPY** 命令将 `config.mqsc` 文件添加到父映像已知的特定位置。

注: 在 Red Hat Enterprise Linux (RHEL) 上，使用命令 **docker** (RHEL V7) 或 **podman** (RHEL V7 或 RHEL V8)。 对于 **podman**，您不需要在命令开头使用 **sudo** 。

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:

```
sudo 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。 当您发出 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。
- 您可以使用命令 **sysctl fs.file-max=524288** 来设置最大打开文件数。

V 9.1.0 在单独的容器中运行本地绑定应用程序

通过在 Docker 中的容器之间共享进程名称空间，您可以在与 IBM MQ 队列管理器不同的容器中运行需要到 IBM MQ 的本地绑定连接的应用程序。

关于此任务

此功能在 IBM MQ 9.0.3 和更高版本的队列管理器中受支持。

您必须遵守以下限制:

- 必须使用 `--pid` 参数共享容器 PID 名称空间。
- 必须使用 `--ipc` 参数共享容器 IPC 名称空间。
- 您必须:
  1. 使用 `--uts` 参数与主机共享容器 UTS 名称空间，或者
  2. 使用 `-h` 或 `--hostname` 参数确保容器具有相同的主机名。
- 必须将 IBM MQ 数据目录安装在可供 `/var/mqm` 目录下的所有容器使用的卷中。

您可以通过在已安装 Docker 的 Linux 系统上完成以下步骤来尝试此功能。

以下示例使用样本 IBM MQ 容器映像。 您可以在 Github 上找到此图像的详细信息。

过程

1. 通过发出以下命令，创建临时目录以充当卷:

```
mkdir /tmp/dockerVolume
```

2. 通过发出以下命令，在名为 `sharedNamespace` 的容器中创建队列管理器 (`QM1`):

```
docker run -d -e LICENSE=accept -e MQ_QMGR_NAME=QM1 --volume /tmp/dockerVol:/mnt/mqm
--uts host --name sharedNamespace ibmcom/mq
```

3. 通过发出以下命令，启动另一个名为 `secondaryContainer`，基于 `ibmcom/mq` 的容器，但不创建队列管理器:

```
docker run --entrypoint /bin/bash --volumes-from sharedNamespace --pid
container:sharedNamespace --ipc container:sharedNamespace --uts host --name
secondaryContainer -it --detach ibmcom/mq
```

4. 通过发出以下命令，在第二个容器上运行 **dspmq** 命令，以查看两个队列管理器的状态:

```
docker exec secondaryContainer dspmq
```

5. 运行以下命令以针对在另一个容器上运行的队列管理器处理 MQSC 命令:

```
docker exec -it secondaryContainer runmqsc QM1
```

结果

现在，本地应用程序在单独的容器中运行，现在可以成功运行命令 (例如 **dspmq**， **amqsput**， **amqsget** 和 **runmqsc**) 作为从辅助容器到 `QM1` 队列管理器的本地绑定。

如果未看到期望的结果，请参阅 第 34 页的『对名称空间应用程序进行故障诊断』 以获取更多信息。

V 9.1.0 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 33 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, 2024.
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
```

CD V 9.1.5 Linux MQ Adv. IBM MQ 操作程序的 API 参考

IBM MQ 提供了 Kubernetes 操作程序，该操作程序提供与 OpenShift Container Platform 的本机集成。

CD V 9.1.5 Linux MQ Adv. mq.ibm.com/v1beta1 的 API 参考

v1beta1 API 可用于创建和管理 QueueManager 资源。

CD V 9.1.5 Linux MQ Adv. mq.ibm.com/v1beta1 的许可参考

spec.license.license 字段必须包含要接受的许可证的许可证标识。 有效值如下所示：

spec.license.license 的值	spec.license.use 的值	许可证信息
L-RJON-BN7PN3	Production 或 NonProduction	IBM Cloud Pak for Integration 2020.2
L-RJON-BPHL2Y		IBM Cloud Pak for Integration Limited Edition 2020.2
L-APIG-BJAKBF	Production 或 Development	IBM MQ Advanced 9.1.5
L-APIG-BM7GDH	Development	IBM MQ Advanced for Developers 9.1.5

请注意，已指定许可证 版本 ，这并非始终与 IBM MQ 的版本相同。

CD V 9.1.5 Linux MQ Adv. QueueManager (mq.ibm.com/v1beta1) 的 API 参考

QueueManager

QueueManager 是 IBM MQ 服务器，用于向应用程序提供排队和发布/预订服务。

字段	描述
apiVersion 字符串	APIVersion 定义对象的此表示的版本化模式。 服务器应将识别的模式转换为最新的内部值，并可能拒绝无法识别的值。 更多信息: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#resources。
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 第 39 页的『QueueManager 规范』	QueueManager 的期望状态。
status 第 40 页的『QueueManagerStatus』	QueueManager 的观察状态。

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。

显示在:

- 第 37 页的『QueueManager 配置』

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 "活动" 队列管理器，另一个是备用队列管理器。 请参阅最新版本的 IBM MQ 中的 容器中的 IBM MQ 的高可用性 。

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。

显示在:

- 第 39 页的『QueueManager 规范』

字段	描述
use 字符串	用于控制软件使用方式的设置，其中许可证支持多次使用。 请参阅 https://ibm.biz/BdqvCF 以获取有效值。
accept 布尔值	是否接受与此软件关联的许可证 (必需)。
license 字符串	您正在接受的许可证的标识。 这必须是您正在使用的 MQ 版本的正确许可证标识。 请参阅 https://ibm.biz/BdqvCF 以获取有效值。
metric 字符串	用于指定要使用的许可证度量的设置。 例如， "ProcessorValueUnit"，"VirtualProcessorCore" 或 "ManagedVirtualServer"。

限制

QueueManagerResourceList 定义 CPU 和内存设置。

显示在:

- 第 42 页的『资源』

字段	描述
cpu	
memory	

LocalObject 参考

LocalObjectReference 包含足够的信息，使您能够在同一名称空间中找到引用的对象。

显示在:

- 第 39 页的『QueueManager 规范』

字段	描述
name 字符串	引用的名称。 更多信息: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names TODO: 添加其他有用的字段。 apiVersion， kind， uid?。

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 39 页的『QueueManager 规范』

字段	描述
keys 第 37 页的『PKISource』 数组	要添加到队列管理器密钥存储库的专用密钥。
trust 第 37 页的『PKISource』 数组	要添加到队列管理器密钥存储库的证书。

PKISource

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 36 页的『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 第 42 页的『私钥』	使用 Kubernetes 密钥提供密钥。

QueueManager 配置

QueueManagerConfig 定义队列管理器容器和底层队列管理器的设置。

显示在:

- 第 39 页的『QueueManager 规范』

字段	描述
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。 对文本格式的消息使用 "基本"。
metrics 第 38 页的『QueueManager 度量』	Prometheus 样式度量的设置。
readinessProbe 第 38 页的『QueueManagerReadinessProbe』	用于控制就绪性探测器的设置。
resources 第 42 页的『资源』	用于控制资源需求的设置。
storage 第 41 页的『QueueManager 存储器』	用于控制队列管理器对持久卷和存储类的使用的存储器设置。
availability 第 35 页的『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。
imagePullPolicy 字符串	用于控制 kubelet 何时尝试拉取指定映像的设置。
livenessProbe 第 38 页的『QueueManagerLivenessProbe』	用于控制活动性探测器的设置。
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。
image 字符串	将使用的容器映像。

字段	描述
name 字符串	底层 MQ 队列管理器的名称 (如果与 metadata.name 不同)。 如果要使用不符合 Kubernetes 规则的队列管理器名称 (例如，包含字母的名称) ，请使用此字段。

QueueManagerLivenessProbe

用于控制活动性探测器的设置。

显示在:

- 第 37 页的『QueueManager 配置』

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。
initialDelaySeconds 整数	在容器启动之后，启动活动性探测器之前的秒数。 更多信息: https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle#container-探针。
periodSeconds 整数	执行探测的频率（以秒计）。
successThreshold 整数	探测器在失败后被视为成功的最小连续成功数。
timeoutSeconds 整数	秒数，经过此秒数之后探测就会超时。 更多信息: https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle#container-探针。

QueueManager 度量

Prometheus 样式度量的设置。

显示在:

- 第 37 页的『QueueManager 配置』

字段	描述
enabled 布尔值	是否启用兼容 Prometheus 的度量端点。

QueueManagerOptionalVolume

MQ 恢复日志的 PersistentVolume 详细信息。 使用多实例队列管理器时必需。

显示在:

- 第 41 页的『QueueManager 存储器』

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。
enabled 布尔值	是应该将此卷作为单独的卷启用，还是将其放在缺省 "queueManager" 卷上。
size 字符串	要传递到 Kubernetes 的 PersistentVolume 的大小。 仅当 "type" 为 "persistent-claim" 时才有效。
sizeLimit 字符串	使用 "临时" 卷时的大小限制。 文件仍会写入临时目录，因此您可以使用此选项来限制大小。 仅当 type 为 "ephemeral" 时才有效。
type 字符串	要使用的卷的类型。 选择 ephemeral 以创建非持久 "emptyDir" 卷，或者选择 persistent-claim 以使用持久卷。

QueueManagerReadinessProbe

用于控制就绪性探测器的设置。

显示在:

- 第 37 页的『QueueManager 配置』

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。
initialDelaySeconds 整数	在容器启动之后，启动活动性探测器之前的秒数。 更多信息: https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle#container-探针。
periodSeconds 整数	执行探测的频率（以秒计）。
successThreshold 整数	探测器在失败后被视为成功的最小连续成功数。
timeoutSeconds 整数	秒数，经过此秒数之后探测就会超时。 更多信息: https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle#container-探针。

QueueManager 规范

QueueManager 的期望状态。

显示在:

- 第 35 页的『QueueManager』

字段	描述
license 第 36 页的『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 第 36 页的『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 第 37 页的『QueueManager 配置』	QueueManagerConfig 定义队列管理器容器和底层队列管理器的设置。
securityContext 第 42 页的『SecurityContext』	要添加到队列管理器 Pod 的 securityContext 的安全设置。
tracing 第 44 页的『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。
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。
imagePullSecrets 第 36 页的『LocalObject 参考』 数组	对同一名称空间中私钥的引用的可选列表，用于拉取此 QueueManager 所使用的任何映像。 如果指定了这些私钥，那么这些私钥将传递到各个拉取器实现以供其使用。 例如，对于 docker，仅接受 DockerConfig 类型的私钥。 有关更多信息，请参阅 https://kubernetes.io/docs/concepts/containers/images#specifying-imagepullsecret-on-a-pod。
template 第 43 页的『模板』	Kubernetes 资源的高级模板。 该模板允许用户覆盖 IBM MQ 如何生成底层 Kubernetes 资源，例如 StatefulSet， Pod 和服务。 这仅适用于高级用户，因为如果使用不正确，可能会中断 MQ 的正常操作。 在 QueueManager 资源中的任何其他位置指定的任何值都将被模板中的设置覆盖。
terminationGracePeriodSeconds 整数	Pod 需要正常终止的可选持续时间 (以秒计)。 值必须是非负整数。 值 0 指示立即删除。 尝试结束队列管理器的目标时间，将应用程序断开连接的阶段升级。 必要时，将中断基本队列管理器维护任务。

字段	描述
web 第 45 页的『WebServer 配置』	MQ Web 服务器的设置。

QueueManagerStatus

QueueManager 的观察状态。

显示在:

- 第 35 页的『QueueManager』

字段	描述
endpoints 第 40 页的『QueueManagerStatusEndpoint』 数组	有关此队列管理器正在公开的端点 (例如 API 或 UI 端点) 的信息。
name 字符串	队列管理器的名称。
versions 第 41 页的『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。
adminUiUrl 字符串	管理 UI 的 URL。
conditions 第 40 页的『QueueManagerStatusCondition』 数组	条件表示队列管理器状态的最新可用观测值。

QueueManagerStatusCondition

QueueManagerStatusCondition 定义队列管理器的条件。

显示在:

- 第 40 页的『QueueManagerStatus』

字段	描述
message 字符串	指示有关上次转换的详细信息的人类可读消息。
type 字符串	条件的类型。
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。

QueueManagerStatusEndpoint

QueueManagerStatusEndpoint 定义 QueueManager 的端点。

显示在:

- 第 40 页的『QueueManagerStatus』

字段	描述
name 字符串	端点的名称。
type 字符串	端点的类型，例如 UI 端点的 "UI"， API 端点的 "API"， API 文档的 "OpenAPI"。
uri 字符串	端点的 URI。

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。

显示在:

- 第 40 页的『QueueManagerStatus』

字段	描述
`available` 第 41 页的『QueueManagerStatusVersion 可用』	其他版本的 MQ 可从 IBM Entitled Registry 获取。
`reconciled` 字符串	正在使用的 IBM MQ 的特定版本。 如果指定了定制映像，那么这可能与实际使用的 MQ 版本不匹配。

QueueManagerStatusVersion 可用

其他版本的 MQ 可从 IBM Entitled Registry 获取。

显示在:

- 第 41 页的『QueueManagerStatusVersion』

字段	描述
`channels` 阵列	可用于自动更新 MQ 版本的通道。
`versions` 第 44 页的『版本』 数组	可用的特定 MQ 版本。

QueueManager 存储器

用于控制队列管理器对持久卷和存储类的使用的存储器设置。

显示在:

- 第 37 页的『QueueManager 配置』

字段	描述
`persistedData` 第 38 页的『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` 第 41 页的『QueueManager 卷』	通常在 `/var/mqm` 下的任何数据的缺省 PersistentVolume 。 将包含所有持久数据和恢复日志 (如果未指定其他卷)。
`recoveryLogs` 第 38 页的『QueueManagerOptionalVolume』	MQ 恢复日志的 PersistentVolume 详细信息。 使用多实例队列管理器时必需。

QueueManager 卷

通常在 `/var/mqm` 下的任何数据的缺省 PersistentVolume 。 将包含所有持久数据和恢复日志 (如果未指定其他卷)。

显示在:

- 第 41 页的『QueueManager 存储器』

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。

字段	描述
size 字符串	要传递到 Kubernetes 的 PersistentVolume 的大小。 仅当 "type" 为 "persistent-claim" 时才有效。
sizeLimit 字符串	使用 "临时" 卷时的大小限制。 文件仍会写入临时目录，因此您可以使用此选项来限制大小。 仅当 type 为 "ephemeral" 时才有效。
type 字符串	要使用的卷的类型。 选择 ephemeral 以创建非持久 "emptyDir" 卷，或者选择 persistent-claim 以使用持久卷。

请求数

QueueManagerResourceList 定义 CPU 和内存设置。

显示在:

- 第 42 页的『资源』

字段	描述
memory	
cpu	

资源

用于控制资源需求的设置。

显示在:

- 第 37 页的『QueueManager 配置』

字段	描述
limits 第 36 页的『限制』	QueueManagerResourceList 定义 CPU 和内存设置。
requests 第 42 页的『请求数』	QueueManagerResourceList 定义 CPU 和内存设置。

私钥

使用 Kubernetes 密钥提供密钥。

显示在:

- 第 37 页的『PKISource』

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。

SecurityContext

要添加到队列管理器 Pod 的 securityContext 的安全设置。

显示在:

- 第 39 页的『QueueManager 规范』

字段	描述
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。
fsGroup 整数	适用于 pod 中所有容器的特殊补充组。 某些卷类型允许 Kubelet 更改要由 pod 拥有的该卷的所有权: 1。 拥有的 GID 将是 FSGroup 2。 设置了 setgid 位 (在卷中创建的新文件将由 FSGroup 拥有) 3。 许可权位为 OR ' d with rw-rw ---- 如果未设置，那么 Kubelet 将不会修改任何卷的所有权和许可权。
initVolumeAsRoot 布尔值	这会影响初始化 PersistentVolume 的容器所使用的 securityContext 。 如果您正在使用要求您作为 root 用户访问新供应卷的存储器提供者，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。 有关更多信息，请参阅 https://docs.openshift.com/container-platform/latest/authentication/managing-security-context-constraints.html。

模板

Kubernetes 资源的高级模板。 该模板允许用户覆盖 IBM MQ 如何生成底层 Kubernetes 资源，例如 StatefulSet， Pod 和服务。 这仅适用于高级用户，因为如果使用不正确，可能会中断 MQ 的正常操作。 在 QueueManager 资源中的任何其他位置指定的任何值都将被模板中的设置覆盖。

显示在:

- 第 39 页的『QueueManager 规范』

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。

TracingAgent

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 44 页的『TracingConfig』

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。
livenessProbe 第 44 页的『TracingProbe』	用于控制活动性探测器的设置。
readinessProbe 第 44 页的『TracingProbe』	用于控制就绪性探测器的设置。

TracingCollector

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 44 页的『TracingConfig』

字段	描述
image 字符串	将使用的容器映像。

字段	描述
imagePullPolicy 字符串	用于控制 kubelet 何时尝试拉取指定映像的设置。
livenessProbe 第 44 页的『TracingProbe』	用于控制活动性探测器的设置。
readinessProbe 第 44 页的『TracingProbe』	用于控制就绪性探测器的设置。

TracingConfig

用于跟踪与 Cloud Pak for Integration 操作仪表板的集成的设置。

显示在:

- 第 39 页的『QueueManager 规范』

字段	描述
agent 第 43 页的『TracingAgent』	仅在 Cloud Pak for Integration 中，可以配置可选跟踪代理程序的设置。
collector 第 43 页的『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。
namespace 字符串	安装了 Cloud Pak for Integration 操作仪表板的名称空间。

TracingProbe

用于控制就绪性探测器的设置。

显示在:

- 第 43 页的『TracingCollector』

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。
initialDelaySeconds 整数	在容器启动之后，启动活动性探测器之前的秒数。 更多信息: https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle#container-探针。
periodSeconds 整数	执行探测的频率（以秒计）。
successThreshold 整数	探测器在失败后被视为成功的最小连续成功数。
timeoutSeconds 整数	秒数，经过此秒数之后探测就会超时。 更多信息: https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle#container-探针。

版本

QueueManagerStatusVersion 定义 MQ 的版本。

显示在:

- 第 41 页的『QueueManagerStatusVersion 可用』

字段	描述
name 字符串	此版本的 QueueManager 的版本 "name"。 这些是 spec.version 字段的有效值。

WebServer 配置

MQ Web 服务器的设置。

显示在:

- 第 39 页的『QueueManager 规范』

字段	描述
`enabled` 布尔值	是否启用 Web 服务器。

声明

本信息是为在美国提供的产品和服务编写的。

IBM 可能在其他国家或地区不提供本文档中讨论的产品、服务或功能。 有关您当前所在区域的产品和服务的信息，请向您当地的 IBM 代表咨询。 任何对 IBM 产品、程序或服务的引用并非意在明示或默示只能使用 IBM 的产品、程序或服务。 只要不侵犯 IBM 的知识产权，任何同等功能的产品、程序或服务都可以代替 IBM 产品、程序或服务。 但是， 评估和验证任何非 IBM 产品、程序或服务的操作，由用户自行负责。

IBM 公司可能已拥有或正在申请与本文档内容有关的各项专利。 提供本文档并未授予用户使用这些专利的任何许可。 您可以以书面形式将许可查询寄往：

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

有关双字节（DBCS）信息的许可查询，请与您所在国家或地区的 IBM 知识产权部门联系，或用书面方式将查询寄往：

知识产权许可
Legal and Intellectual Property Law
IBM Japan, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 063-8506 Japan

本条款不适用英国或任何这样的条款与当地法律不一致的国家或地区: International Business Machines Corporation “按现状” 提供本出版物，不附有任何种类的 (无论是明示的还是暗含的) 保证，包括但不限于暗含的有关非侵权，适销和适用于某种特定用途的保证。 某些国家或地区在某些交易中不允许免除明示或暗含的保证。 因此本条款可能不适用于您。

本信息中可能包含技术方面不够准确的地方或印刷错误。 此处的信息将定期更改；这些更改将编入本资料的新版本中。 IBM 可以随时对本出版物中描述的产品和/或程序进行改进和/或更改，而不另行通知。

本信息中对非 IBM Web 站点的任何引用都只是为了方便起见才提供的，不以任何方式 充当对那些 Web 站点的保证。 那些 Web 站点中的资料不是 IBM 产品资料的一部分，使用那些 Web 站点带来的风险将由您自行承担。

IBM 可以按它认为适当的任何方式使用或分发您所提供的任何信息而无须对您承担任何责任。

本程序的被许可方如果要了解有关程序的信息以达到如下目的：(i) 允许在独立创建的程序和其他程序（包括本程序）之间进行信息交换，以及 (ii) 允许对已经交换的信息进行相互使用，请与下列地址联系：

IBM Corporation
软件互操作性协调员，部门 49XA
北纬 3605 号公路
罗切斯特，明尼苏达州 55901
U.S.A.

只要遵守适当的条件和条款，包括某些情形下的一定数量的付费，都可获得这方面的信息。

本资料中描述的许可程序及其所有可用的许可资料均由 IBM 依据 IBM 客户协议、IBM 国际软件许可协议或任何同等协议中的条款提供。

此处包含的任何性能数据都是在受控环境中测得的。 因此，在其他操作环境中获得的数据可能会有明显的不同。 有些测量可能是在开发级的系统上进行的，因此不保证与一般可用系统上进行的测量结果相同。 此外，有些测量是通过推算而估计的， 实际结果可能会有差异。 本文档的用户应当验证其特定环境的适用数据。

© Copyright IBM Corp. 2007, 2024

涉及非 IBM 产品的信息可从这些产品的供应商、其出版说明或其他可公开获得的资料中 获取。 IBM 没有对这些产品进行测试，也无法确认其性能的精确性、兼容性或任何其他关于非 IBM 产品的声明。 有关非 IBM 产品性能的问题应当向这些产品的供应商提出。

所有关于 IBM 未来方向或意向的声明都可随时更改或收回，而不另行通知，它们仅仅表示了目标和意愿而已。

本信息包含日常商业运作所使用的数据和报表的示例。 为了尽可能全面地说明这些数据和报表，这些示例包括个人、公司、品牌和产品的名称。 所有这些名称都是虚构的，如与实际商业企业所使用的名称和地址有任何雷同，纯属巧合。

版权许可：

本信息包含源语言形式的样本应用程序，用以阐明在不同操作平台上的编程技术。 如果是为按照在编写样本程序的操作平台上的应用程序编程接口（API）进行应用程序的开发、使用、经销或分发为目的，您可以任何形式对这些样本程序进行复制、修改、分发，而无须向 IBM 付费。 这些示例并未在所有条件下作全面测试。 因此，IBM 不能担保或默示这些程序的可靠性、可维护性或功能。

如果您正在查看本信息的软拷贝，图片和彩色图例可能无法显示。

编程接口信息

编程接口信息 (如果提供) 旨在帮助您创建用于此程序的应用软件。

本书包含有关允许客户编写程序以获取 WebSphere MQ 服务的预期编程接口的信息。

但是，该信息还可能包含诊断、修改和调优信息。 提供诊断、修改和调优信息是为了帮助您调试您的应用程序软件。

要点: 请勿将此诊断，修改和调整信息用作编程接口，因为它可能会发生更改。

商标

IBM IBM 徽标 ibm.com 是 IBM Corporation 在全球许多管辖区域的商标。 当前的 IBM 商标列表可从 Web 上的 “Copyright and trademark information”www.ibm.com/legal/copytrade.shtml 获取。 其他产品和服务名称可能是 IBM 或其他公司的商标。

Microsoft 和 Windows 是 Microsoft Corporation 在美国和/或其他国家或地区的商标。

UNIX 是 Open Group 在美国和其他国家或地区的注册商标。

Linux 是 Linus Torvalds 在美国和/或其他国家或地区的商标。

此产品包含由 Eclipse 项目 (http://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: