9.2

儲存器中的 *IBM MQ*

附註

使用本資訊及其支援的產品之前，請先閱讀第 143 頁的『注意事項』中的資訊。

除非新版中另有指示，否則此版本適用於 IBM® MQ 6.2 版及所有後續版本與修訂版本。

當您將資訊傳送至 IBM 時，您授與 IBM 非專屬權限，以任何其認為適當的方式使用或散佈資訊，而無需對您負責。

目錄

Multi IBM MQ 在儲存器及 IBM Cloud Pak for Integration 中

儲存器可讓您將 IBM MQ 佇列管理程式或 IBM MQ 用戶端應用程式及其所有相依關係包裝成標準化單元，以進行軟體開發。

您可以在 Red Hat® OpenShift®上使用 IBM MQ Operator 來執行 IBM MQ 。 這可以使用 IBM Cloud Pak for Integration、 IBM MQ Advanced 或 IBM MQ Advanced for Developers 來完成。

您也可以在自己建置的容器中執行 IBM MQ 。

MQ Adv. CD 如需 IBM MQ Operator 的相關資訊，請參閱下列鏈結。

Multi 規劃儲存器中的 IBM MQ

在儲存器中規劃 IBM MQ 時，請考量 IBM MQ 提供的各種架構選項支援，例如如何管理高可用性，以及如何保護佇列管理程式的安全。

關於這項作業

在儲存器架構中規劃 IBM MQ 之前，您應該先熟悉基本 IBM MQ 概念 (請參閱 IBM MQ 技術概觀) 以及基本 Kubernetes/Red Hat OpenShift 概念 (請參閱 Red Hat OpenShift Container Platform 架構)。

程序

- 第 5 頁的『選擇您要如何在儲存器中使用 IBM MQ』.
- 第 6 頁的『支援 IBM MQ Operator』.
- 第 12 頁的『支援建置您自己的 IBM MQ 佇列管理程式儲存器映像檔』.
- 第 10 頁的『IBM MQ Operator 的儲存體考量』.
- 第 15 頁的『儲存器中 IBM MQ 的高可用性』.
- 第 16 頁的『儲存器中 IBM MQ 的災難回復』.
- 第 17 頁的『儲存器中 IBM MQ 的使用者鑑別和授權』.

選擇您要如何在儲存器中使用 IBM MQ

在容器中使用 IBM MQ 有多個選項: 您可以選擇使用 IBM MQ Operator，它會使用預先包裝的容器映像檔，或者您可以建置自己的映像檔及部署程式碼。

使用 IBM MQ Operator

OpenShift CP4I

如果您計劃在 Red Hat OpenShift Container Platform 上部署，則可能想要使用 IBM MQ Operator。

IBM MQ Operator 會將新的 `QueueManager` 自訂資源新增至 Red Hat OpenShift Container Platform。 操作員監看新的佇列管理程式定義，然後將它們轉換成必要的低階資源，例如 `StatefulSet` 和 `Service` 資源。 如果是原生 HA，操作員也可以執行佇列管理程式實例的複式漸進式更新。 請參閱 第 140 頁的『執行您自己的原生 HA 佇列管理程式漸進式更新的考量』

使用 IBM MQ Operator 時，不支援部分 IBM MQ 特性。 如果您想要執行下列任何動作，則需要建置您自己的映像檔及圖表:

- 使用 REST API 進行管理或傳訊
- 使用下列任何 MQ 元件:
  - Managed File Transfer 代理程式及其資源。 不過，您可以使用 IBM MQ Operator 來提供一或多個「協調」、「指令」或「代理程式」佇列管理程式。

  - AMQP
  - IBM MQ Bridge to Salesforce
  - IBM MQ Bridge to blockchain (在儲存器中不受支援)
  - IBM MQ Telemetry Transport (MQTT).
- 自訂與 **crtmqm**、 **strmqm** 和 **endmqm** 搭配使用的選項，例如配置日誌檔頁面。 大部分選項都可以使用 INI 檔案來配置。

請注意， IBM MQ Operator 及儲存器正在快速發展，因此在 Long Term Support 版本下不受支援。

IBM MQ Operator 包括預先建置的容器映像檔，以及用於在 Red Hat OpenShift Container Platform 上執行的部署程式碼。 IBM MQ Operator 可以用來部署所提供的 IBM MQ 儲存器映像檔，或在其上分層的儲存器映像檔，但無法用來部署自訂建置的 MQ 儲存器映像檔。

建置您自己的映像檔及部署程式碼

Multi

這是最具彈性的容器解決方案，但它需要您具備配置容器的強大技能，以及 "擁有" 產生的容器。 如果您不打算使用 Red Hat OpenShift Container Platform，則需要建置您自己的映像檔及部署程式碼。

提供用於建置您自己的映像檔的範例。 請參閱 第 133 頁的『建置您自己的 IBM MQ 容器及部署程式碼』。

相關概念

第 6 頁的『支援 IBM MQ Operator』
只有在 Red Hat OpenShift Container Platform 上部署時，才支援 IBM MQ Operator 。

第 12 頁的『支援建置您自己的 IBM MQ 佇列管理程式儲存器映像檔』
IBM MQ 提供程式碼以在 GitHub 上建置 IBM MQ 佇列管理程式儲存器。 這是根據 IBM 用來建置其專屬受支援容器的程序，而且您可以使用此 GitHub 儲存庫來簡化並加速建置您專屬容器映像檔。

OpenShift CP4I CD EUS 支援 IBM MQ Operator

只有在 Red Hat OpenShift Container Platform 上部署時，才支援 IBM MQ Operator 。

IBM MQ Operator 使用基於 IBM MQ Continuous Delivery (CD) 版本的映像檔，但 IBM Cloud Pak for Integration 隨附了「延伸更新支援 (EUS)」版本。 支援 CD 版次最多一年，或支援兩個 CD 版次，兩者以較長者為準。 透過 IBM MQ Operator 無法使用 Long Term Support 版本的 IBM MQ 。 IBM Cloud Pak for Integration 2020.4.1 是「延伸更新支援 (EUS)」版本，如果您使用標示為 -eus 的 IBM MQ 版本，則支援時間為 18 個月。 否則， IBM MQ 9.2 會被視為具有 IBM MQ Operator 的 Continuous Delivery 版次。

IBM MQ Operator 使用儲存器映像檔來提供 Red Hat Universal Base Image (UBI) 上的 IBM MQ 安裝，其中包括 IBM MQ 使用的重要 Linux® 程式庫及公用程式。 在 Red Hat OpenShift 上執行時， Red Hat 支援 UBI。

IBM MQ Operator 在 amd64 及 s390x (z/Linux) 架構上受支援。

相關概念

第 12 頁的『支援建置您自己的 IBM MQ 佇列管理程式儲存器映像檔』
IBM MQ 提供程式碼以在 GitHub 上建置 IBM MQ 佇列管理程式儲存器。 這是根據 IBM 用來建置其專屬受支援容器的程序，而且您可以使用此 GitHub 儲存庫來簡化並加速建置您專屬容器映像檔。

OpenShift CP4I CD EUS IBM MQ Operator 的版本支援

IBM MQ、 Red Hat OpenShift Container Platform 和 IBM Cloud Pak for Integration 受支援版本之間的對映。

- 第 7 頁的『可用的 IBM MQ 版本』
- 第 7 頁的『相容的 Red Hat OpenShift Container Platform 版本』
- 第 7 頁的『IBM Cloud Pak for Integration 版本』
- 第 8 頁的『舊版運算子中的可用 IBM MQ 版本』

- 第 8 頁的『舊版運算子的相容 Red Hat OpenShift Container Platform 版本』

可用的 IBM MQ 版本

操作員通道	Operator 版本	IBM MQ 版本							
		9.1.5	9.2.0 CD	9.2.0 EUS	9.2.1	9.2.2	9.2.3	9.2.4	9.2.5
v1.6	1.6	⚠	⚠	→	⚠	●	●		
v1.7	1.7	⚠	⚠	→	⚠	●	●	●	
v1.8	1.8	⚠	⚠	→	⚠	⚠	●	●	●

索引鍵:

●
可用的 Continuous Delivery 支援

▯
Extended Update Support 可用

→
僅在從 Extended Update Support 運算元移轉至 Continuous Delivery 運算元期間可用。

⚠
已棄用。 當 IBM MQ 版本不再支援時，它們可能仍可在操作器中配置，但不再符合支援資格，並可能在未來版本中移除。

如需每一個版本的完整資料，包括每一個版本中的詳細特性、變更及修正程式，請參閱 第 18 頁的『IBM MQ Operator 的發行歷程』 。

相容的 Red Hat OpenShift Container Platform 版本

操作員通道	Operator 版本	Red Hat OpenShift Container Platform 版本 [1]				
		4.6	4.7[2]	4.8	4.9	4.10
v1.6	1.6	●	●	●	●	●
v1.7	1.7	●	●	●	●	●
v1.8	1.8	●	●	●	●	●

索引鍵:

●
可用的 Continuous Delivery 支援

▯
Extended Update Support 可用

IBM Cloud Pak for Integration 版本

支援 IBM MQ Operator 1.8.x 作為 IBM Cloud Pak for Integration 2021.4.1 版的一部分或單獨使用。

支援 IBM MQ Operator 1.7.x 作為 IBM Cloud Pak for Integration 2021.4.1 版的一部分使用，或單獨使用。

支援 IBM MQ Operator 1.6.x 作為 IBM Cloud Pak for Integration 2021.2.1 版、 2021.3.1 版的一部分或單獨使用。

---

[1] Red Hat OpenShift Container Platform 版本取決於其自己的支援日期。 如需相關資訊，請參閱 Red Hat OpenShift Container Platform 生命週期原則 。

[2] IBM MQ Operator 取決於 IBM Cloud Pak foundational services。 如果您想要使用 Red Hat OpenShift Container Platform 4.7，您應該先升級 IBM Cloud Pak foundational services 的版本。

不再支援 IBM MQ Operator 1.5.x。

不再支援 IBM MQ Operator 1.4.x。

不再支援 IBM MQ Operator 1.3.x。

不再支援 IBM MQ Operator 1.2.x。

不再支援 IBM MQ Operators 1.1.x 和 1.0.x。

舊版運算子中的可用 IBM MQ 版本

下表適用於現在已達到「使用期限」的 IBM MQ Operator 版本。

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

索引鍵:

→

僅在從 Extended Update Support 運算元移轉至 Continuous Delivery 運算元期間可用。

⚠

已棄用。 當 IBM MQ 版本不再支援時，它們仍可在 IBM MQ Operator 中配置，但不再適合支援。

如需每一個版本的完整資料，包括每一個版本中的詳細特性、變更及修正程式，請參閱 第 18 頁的『IBM MQ Operator 的發行歷程』。

舊版運算子的相容 Red Hat OpenShift Container Platform 版本

下表適用於現在已達到「使用期限」的 IBM MQ Operator 版本。

操作員通道	Operator 版本	Red Hat OpenShift Container Platform 版本 [3]						
		4.4[4]	4.5[5]	4.6	4.7[6]	4.8	4.9	4.10
v1.0	1.0	⚠	⚠	⚠	⚠			
v1.1	1.1	⚠	⚠	⚠	⚠	⚠		
v1.2	1.2	⚠	⚠	⚠	⚠	⚠		

[3] Red Hat OpenShift Container Platform 版本取決於其自己的支援日期。 如需相關資訊，請參閱 Red Hat OpenShift Container Platform 生命週期原則。

[4] Red Hat OpenShift Container Platform 4.4 已達到「使用期限」。 如需相關資訊，請參閱 Red Hat OpenShift Container Platform 生命週期原則。

[5] Red Hat OpenShift Container Platform 4.5 已達到「使用期限」。 如需相關資訊，請參閱 Red Hat OpenShift Container Platform 生命週期原則。

[6] IBM MQ Operator 取決於 IBM Cloud Pak foundational services。 如果您想要使用 Red Hat OpenShift Container Platform 4.7，您應該先升級 IBM Cloud Pak foundational services 的版本。

操作員通道	Operator 版本	Red Hat OpenShift Container Platform 版本 [3]						
		4.4[4]	4.5[5]	4.6	4.7[6]	4.8	4.9	4.10
v1.3-eus	1.3			⚠	→	→	→	→
v1.4	1.4			⚠	⚠	⚠	⚠	
v1.5	1.5			⚠	⚠	⚠	⚠	⚠

索引鍵:

→
僅在從 Extended Update Support 運算元移轉至 Continuous Delivery 運算元期間可用。

⚠
IBM MQ Operator 版本已達到「使用期限」，但先前已在此版本的 Red Hat OpenShift Container Platform 上提供。

IBM MQ Operator 的相依關係

IBM MQ Operator 取決於 IBM Cloud Pak foundational services 操作器，它也會安裝 IBM Operand Deployment Lifecycle Manager (ODLM) 操作器。 當您安裝 IBM MQ Operator 時，會自動安裝這些操作器。這些相依運算子具有較小的 CPU 及記憶體覆蓋區，並在某些情況下用來部署其他資源。

當您建立 `QueueManager` 時， IBM MQ Operator 會針對它需要的其他服務建立 `OperandRequest`。 `OperandRequest` 由 ODLM 操作器履行，必要的話，將安裝並實例化必要的服務。 需要哪些服務取決於部署佇列管理程式時所接受的授權合約，以及要求哪些佇列管理程式元件。

- 如果您選擇 IBM MQ Advanced 或 IBM MQ Advanced for Developers 授權，則不會要求其他服務。 例如，在下列情況下，不會使用 IBM Cloud Pak foundational services :

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

- 如果您選擇 IBM Cloud Pak for Integration 授權並選擇啟用 Web 伺服器，則 IBM MQ Operator 也會實例化「IBM Identity and Access Management (IAM) 操作員」，以啟用單一登入。 如果您已安裝 IBM Cloud Pak for Integration 操作器，則 IAM 操作器將已可用。 例如:

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

不過，如果您停用 Web 伺服器，則不會要求任何 IBM Cloud Pak foundational services。 例如:

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

---

[3] Red Hat OpenShift Container Platform 版本取決於其自己的支援日期。 如需相關資訊，請參閱 Red Hat OpenShift Container Platform 生命週期原則。

[4] Red Hat OpenShift Container Platform 4.4 已達到「使用期限」。 如需相關資訊，請參閱 Red Hat OpenShift Container Platform 生命週期原則。

[5] Red Hat OpenShift Container Platform 4.5 已達到「使用期限」。 如需相關資訊，請參閱 Red Hat OpenShift Container Platform 生命週期原則。

[6] IBM MQ Operator 取決於 IBM Cloud Pak foundational services。 如果您想要使用 Red Hat OpenShift Container Platform 4.7，您應該先升級 IBM Cloud Pak foundational services 的版本。

```
  web:
    enabled: false
```

舊版 IBM MQ Operator 一律要求安裝 IBM Licensing Operator (及其相依關係)，以追蹤授權使用。 從 IBM MQ Operator 1.5 開始，不會要求授權服務，您需要個別要求。

IBM MQ Operator 需要 1 個 CPU 核心和 1 GB 記憶體。 如需相依操作員的軟硬體需求明細，請參閱 基礎服務的硬體需求及建議。

您可以選擇佇列管理程式所使用的 CPU 及記憶體數量。 如需相關資訊，請參閱 第 123 頁的『.spec.queueManager.resources』。

相關參考

第 115 頁的『mq.ibm.com/v1beta1 的授權參考手冊』

IBM MQ Operator 所需的叢集範圍許可權

IBM MQ Operator 需要以叢集為範圍的許可權，才能管理許可 Webhook 和範例，以及讀取儲存類別和叢集版本資訊。

IBM MQ Operator 需要下列叢集範圍的許可權:

- 管理許可 Webhook 的許可權。 這容許建立、擷取及更新特定 Webhook，這些 Webhook 用於建立及管理「操作員」提供的儲存器的程序。
  - API 群組: **admissionregistration.k8s.io**
  - 資源: **validatingwebhookconfigurations**
  - verbs: **create, get, update**
- 建立及管理在 Red Hat OpenShift 主控台中用來在建立自訂資源時提供範例和 Snippet 的資源的許可權。
  - API 群組: **console.openshift.io**
  - 資源: **consoleyamlsamples**
  - verbs: **create, get, update, delete**
- 讀取叢集版本的許可權。 這可讓「操作員」回復叢集環境的任何問題。
  - API 群組: **config.openshift.io**
  - 資源: **clusterversions**
  - verbs: **get, list, watch**
- 在叢集上讀取儲存類別的許可權。 這可讓「操作員」回復儲存器中所選取儲存類別的任何問題。
  - API 群組: **storage.k8s.io**
  - 資源: **storageclasses**
  - verbs: **get, list**

IBM MQ Operator 的儲存體考量

IBM MQ Operator 以兩種儲存模式執行:

- 當容器重新啟動時，可以捨棄容器的所有狀態資訊時，會使用 **暫時儲存體**。 這通常在建立環境以示範時使用，或在使用獨立式佇列管理程式進行開發時使用。
- **持續性儲存體** 是 IBM MQ 的一般配置，可確保在容器重新啟動時，現有配置、日誌及持續訊息在已重新啟動的容器中可用。

IBM MQ Operator 提供自訂儲存體性質的功能，視環境及想要的儲存體模式而定，這些儲存體性質可能會有很大差異。

臨時儲存空間

IBM MQ 是有狀態的應用程式，並將此狀態持續保存至儲存體，以在重新啟動時進行回復。 如果使用暫時儲存體，則在重新啟動時，佇列管理程式的所有狀態資訊都會遺失。 這包括：

- 所有訊息
- 所有佇列管理程式至佇列管理程式的通訊狀態 (通道訊息序號)
- 佇列管理程式的 MQ 叢集身分
- 所有交易狀態
- 所有佇列管理程式配置
- 所有本端診斷資料

因此，您需要考量暫時儲存體是否適合正式作業、測試或開發實務範例。 例如，已知所有訊息都是非持續性，且佇列管理程式不是「MQ 叢集」的成員。 除了在重新啟動時刪除所有傳訊狀態之外，也會捨棄佇列管理程式的配置。 若要啟用完全暫時儲存器，必須將 IBM MQ 配置新增至儲存器映像檔本身 (如需相關資訊，請參閱 第 102 頁的『使用 Red Hat OpenShift CLI 以自訂 MQSC 及 INI 檔案建置映像檔』)。 如果未完成此作業，則每次容器重新啟動時都需要配置 IBM MQ 。

OpenShift CP4I 例如，若要將 IBM MQ 配置為暫時儲存體， `QueueManager` 的儲存體類型應該包括下列：

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

持續性儲存體

OpenShift CP4I

IBM MQ 通常會使用持續性儲存體來執行，以確保在重新啟動之後佇列管理程式會保留其持續訊息及配置。 因此，這是預設行為。 由於各種儲存體提供者及每一個支援的不同功能，這通常表示需要自訂配置。 下列範例概述在 v1beta1 API 中自訂 MQ 儲存體配置的一般欄位：

- spec.queueManager.availability 控制可用性模式。 如果您使用 `SingleInstance` ，則只需要 `ReadWriteOnce` 儲存體，而 `multiInstance` 需要具有正確檔案鎖定性質且支援 `ReadWriteMany` 的儲存類別。 IBM MQ 提供 支援陳述式 及 測試陳述式。 可用性模式也會影響持續性磁區佈置。 如需相關資訊，請參閱： 第 15 頁的『儲存器中 IBM MQ 的高可用性』
- spec.queueManager.storage 控制個別儲存體設定。 佇列管理程式可以配置成在一到四個持續性磁區之間使用

下列範例顯示使用單一實例佇列管理程式之簡式配置的 Snippet：

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

下列範例顯示多重實例佇列管理程式配置的 Snippet ， 其中含有非預設儲存類別，以及需要補充群組的檔案儲存體：

```
spec:
  queueManager:
    availability:
      type: MultiInstance
    storage:
      queueManager:
        class: ibmc-file-gold-gid
      persistedData:
        enabled: true
        class: ibmc-file-gold-gid
      recoveryLogs:
        enabled: true
        class: ibmc-file-gold-gid
```

```
  securityContext:
    supplementalGroups: [99]
```

註: 您也可以使用單一實例佇列管理程式來配置增補群組。

註: 如果您使用原生 HA，則不需要共用檔案系統 (請參閱 第 15 頁的『儲存器中 IBM MQ 的高可用性』)。特別是您不應使用 NFSv3。

Linux 支援建置您自己的 IBM MQ 佇列管理程式儲存器映像檔

IBM MQ 提供程式碼以在 GitHub 上建置 IBM MQ 佇列管理程式儲存器。 這是根據 IBM 用來建置其專屬受支援容器的程序，而且您可以使用此 GitHub 儲存庫來簡化並加速建置您專屬容器映像檔。

程式碼在下列 mq-container GitHub 儲存庫中提供: https://github.com/ibm-messaging/mq-container。 這是根據 Apache 2.0 授權提供，並由社群提供支援。

儲存庫不會使用標準 Linux RPM 套件; 它會使用壓縮套件來進行儲存器部署。 此方法的好處是您可以在更安全的儲存器環境中執行，而不需要呈報的許可權。 不過，這確實會影響可用的安全選項，因為 IBM MQ 傳統上會使用提升的許可權來進行作業系統型鑑別。 對於儲存器部署，使用作業系統型鑑別通常不是良好的作法; 您可以改用相互 TLS 或 LDAP 鑑別。 使用 IBM MQ Advanced for Developers，您也可以使用檔案型鑑別，讓您的使用者可以快速開始使用。

儲存器環境內不支援抄寫的資料佇列管理程式 (RDQM)。 您可以使用 第 84 頁的『原生 HA』來取得與 RDQM 類似的功能。

相關概念

第 6 頁的『支援 IBM MQ Operator』
只有在 Red Hat OpenShift Container Platform 上部署時，才支援 IBM MQ Operator 。

IBM MQ 非安裝映像檔

Linux 建置您自己的 IBM MQ 儲存器映像檔時的授權註釋

授權註釋可讓您根據儲存器上定義的限制 (而非基礎機器上定義的限制) 來追蹤使用情形。 您可以將用戶端配置為使用特定註釋來部署儲存器，然後 IBM License Service 會使用這些註釋來追蹤使用情形。

部署自行建置 IBM MQ 儲存器映像檔時，有兩種常見的授權方法:

- 授權執行儲存器的整個機器。
- 根據相關聯的限制來授權儲存器。

這兩個選項都可供用戶端使用，您可以在 Passport Advantage 上的 IBM 儲存器授權」頁面 中找到進一步詳細資料。

如果要根據儲存器限制來授權 IBM MQ 儲存器，則需要安裝 IBM License Service 以追蹤使用情形。 如需受支援環境及安裝指示的相關資訊，請參閱 GitHub 上的 ibm-licensing-operator 頁面。

IBM License Service 安裝在部署 IBM MQ 儲存器的 Kubernetes 叢集上，並使用 Pod 註釋來追蹤使用情形。因此，用戶端需要使用 IBM License Service 隨後使用的特定註釋來部署 Pod。 根據您在儲存器內部署的授權及功能，使用下列一或多個註釋:

- 第 13 頁的『IBM MQ 進階儲存器』
- 第 13 頁的『IBM MQ 進階高可用性抄本儲存器』
- 第 13 頁的『IBM MQ 基本儲存器』
- 第 13 頁的『IBM MQ 基本高可用性抄本儲存器』
- 第 13 頁的『IBM MQ Advanced for Developers 儲存器』
- 第 13 頁的『IBM MQ 具有 CP4I 授權的進階儲存器 (正式作業)』
- 第 13 頁的『IBM MQ 具有 CP4I 授權的進階高可用性抄本儲存器 (正式作業)』
- 第 14 頁的『IBM MQ 具有 CP4I 授權的進階儲存器 (非正式作業)』
- 第 14 頁的『IBM MQ 具有 CP4I 授權的進階高可用性抄本儲存器 (非正式作業)』
- 第 14 頁的『IBM MQ 基本程式 (含 CP4I 授權) (正式作業)』

- 第 14 頁的『IBM MQ 具有 CP4I 授權的基本高可用性抄本 (正式作業)』
- 第 14 頁的『IBM MQ 具有 CP4I 授權的基本程式 (非正式作業)』
- 第 14 頁的『IBM MQ 具有 CP4I 授權的基本高可用性抄本 (非正式作業)』

IBM MQ 進階儲存器

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

IBM MQ 進階高可用性抄本儲存器

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

IBM MQ 基本儲存器

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

IBM MQ 基本高可用性抄本儲存器

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

IBM MQ Advanced for Developers 儲存器

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

IBM MQ 具有 CP4I 授權的進階儲存器 (正式作業)

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

IBM MQ 具有 CP4I 授權的進階高可用性抄本儲存器 (正式作業)

```
productName: "IBM MQ Advanced High Availability Replica with CP4I License"
productID: "546cb719714942c18748137ddd8d5659"
productChargedContainers: "All" | "NAME_OF_CONTAINER"
productMetric: "VIRTUAL_PROCESSOR_CORE"
productCloudpakRatio: "10:1"
```

```
cloudpakName: "IBM Cloud Pak for Integration"
cloudpakId: "c8b82d189e7545f0892db9ef2731b90d"
```

IBM MQ 具有 CP4I 授權的進階儲存器 (非正式作業)

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

IBM MQ 具有 CP4I 授權的進階高可用性抄本儲存器 (非正式作業)

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

IBM MQ 基本程式 (含 CP4I 授權) (正式作業)

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

IBM MQ 具有 CP4I 授權的基本高可用性抄本 (正式作業)

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

IBM MQ 具有 CP4I 授權的基本程式 (非正式作業)

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

IBM MQ 具有 CP4I 授權的基本高可用性抄本 (非正式作業)

```
productName: "IBM MQ High Availability Replica with CP4I License Non-Production"
productID: "f5d0e21c013c4d4b8b9b2ce701f31928"
productChargedContainers: "All" | "NAME_OF_CONTAINER"
productMetric: "VIRTUAL_PROCESSOR_CORE"
productCloudpakRatio: "40:1"
```

```
cloudpakName: "IBM Cloud Pak for Integration"
cloudpakId: "c8b82d189e7545f0892db9ef2731b90d"
```

OpenShift CP4I Kubernetes 儲存器中 IBM MQ 的高可用性

IBM MQ Operator 有三個高可用性選項: **原生 HA 佇列管理程式** (具有作用中抄本及兩個待命抄本)、 **多重實例佇列管理程式** (使用共用網路檔案系統的主動-待命配對) 或 **單一具復原力佇列管理程式** (提供使用網路儲存體的 HA 簡單方法)。 後兩者依賴檔案系統來確保可回復資料的可用性，但原生 HA 不會。 因此，當不使用「原生 HA」時，檔案系統的可用性對佇列管理程式可用性而言很重要。 當資料回復很重要時，檔案系統應該透過抄寫來確保備援。

您應該分別考量 **訊息** 和 **服務** 可用性。 使用 IBM MQ for Multiplatforms，訊息只會儲存在一個佇列管理程式上。 因此，如果該佇列管理程式變成無法使用，您會暫時無法存取它所保留的訊息。 若要達到高 **訊息** 可用性，您需要能夠儘快回復佇列管理程式。 您可以讓多個佇列實例供用戶端應用程式使用，例如使用 IBM MQ 統一叢集，以達到 **服務** 可用性。

佇列管理程式可以分為兩部分: 儲存在磁碟上的資料，以及容許存取資料的執行中處理程序。 任何佇列管理程式都可以移至不同的 Kubernetes 節點，只要它保留相同的資料 (由 Kubernetes 持續性磁區提供)，且仍可由用戶端應用程式在網路中定址。 在 Kubernetes 中，「服務」是用來提供一致的網路身分。

IBM MQ 依賴於持續性磁區上資料的可用性。 因此，提供持續性磁區的儲存體可用性對佇列管理程式可用性而言很重要，因為 IBM MQ 的可用程度不能超過它所使用的儲存體。 如果您想要容忍整個可用性區域的中斷，則需要使用將磁碟寫入抄寫至另一個區域的磁區提供者。

原生 HA 佇列管理程式

CP4I V 9.2.3

原生 HA 佇列管理程式可從 IBM Cloud Pak for Integration 2021.2.1 使用 IBM MQ Operator 1.6 或更高版本，以及 IBM MQ 9.2.3 或更高版本。

原生 HA 佇列管理程式涉及一個 **作用中** 及兩個 **抄本** Kubernetes Pod，它們會作為 Kubernetes StatefulSet 的一部分執行，每一個只有三個抄本，並具有自己的 Kubernetes 持續性磁區集。 使用原生 HA 佇列管理程式 (租賃型鎖定除外) 時，共用檔案系統的 IBM MQ 需求也適用，但您不需要使用共用檔案系統。 您可以使用區塊儲存體，並在頂端安裝適當的檔案系統。 例如， *xfs* 或 *ext4*。 原生 HA 佇列管理程式的回復時間由下列因素控制:

1. 抄本實例偵測作用中實例失敗所花費的時間。 這是可配置的。
2. Kubernetes Pod 就緒探測偵測備妥容器已變更並重新導向網路資料流量所需的時間。 這是可配置的。
3. IBM MQ 用戶端重新連接所需的時間。

如需相關資訊，請參閱 第 84 頁的『原生 HA』

多重實例佇列管理程式

Multi

多重實例佇列管理程式涉及一個 **作用中** 及一個 **待命** Kubernetes Pod，該 Pod 作為具有正好兩個抄本及一組 Kubernetes 持續性磁區的 Kubernetes 有狀態集的一部分執行。 使用共用檔案系統，將佇列管理程式交易日誌及資料保留在兩個持續性磁區上。

多重實例佇列管理程式需要 **作用中** 及 **待命** Pod 都具有持續性磁區的並行存取權。 若要配置此項目，您可以使用 Kubernetes 持續性磁區，並將 **access mode** 設為 `ReadWriteMany`。 磁區也必須符合 IBM MQ 共用檔案系統的需求，因為 IBM MQ 依賴自動釋放檔案鎖定來進行佇列管理程式失效接手。 IBM MQ 會產生 已測試檔案系統的清單。

多重實例佇列管理程式的回復時間由下列因素控制:

1. 共用檔案系統在失敗之後釋放作用中實例最初所花費的鎖定所花費的時間。
2. 待命實例獲得鎖定然後啟動所需的時間。
3. Kubernetes Pod 就緒探測偵測備妥容器已變更並重新導向網路資料流量所需的時間。 這是可配置的。
4. IBM MQ 用戶端重新連接所花費的時間。

單一回復型佇列管理程式

Multi

單一回復佇列管理程式是在單一 Kubernetes Pod 中執行之佇列管理程式的單一實例，其中 Kubernetes 會監視佇列管理程式，並視需要取代 Pod。

使用單一回復型佇列管理程式 (租賃型鎖定除外) 時，也適用 IBM MQ 共用檔案系統的需求，但您不需要使用共用檔案系統。 您可以使用區塊儲存體，並在頂端安裝適當的檔案系統。 例如， *xfs* 或 *ext4*。

單一回復型佇列管理程式的回復時間由下列因素控制:

1. 活性探測執行所花費的時間，以及它所容忍的失敗次數。 這是可配置的。
2. Kubernetes 排程器將失敗 Pod 重新排定到新節點所需的時間。
3. 將儲存器映像檔下載至新「節點」所需的時間。 如果您使用 **imagePullPolicy** 值 `IfNotPresent`，則映像檔可能已在該節點上可用。
4. 啟動新佇列管理程式實例所需的時間。
5. Kubernetes Pod 就緒性探測偵測容器已備妥所花費的時間。 這是可配置的。
6. IBM MQ 用戶端重新連接所花費的時間。

重要:

雖然單一具復原力的佇列管理程式型樣提供一些好處，但您需要瞭解是否可以在「節點」失敗的相關限制下達成可用性目標。

在 Kubernetes 中，故障 Pod 通常會快速回復; 但整個節點的故障會以不同方式處理。 當搭配使用有狀態工作量 (例如 IBM MQ 與 Kubernetes StatefulSet) 時，如果 Kubernetes 主要節點與工作者節點失去聯絡，則無法判定節點是否失敗，或是否僅失去網路連線功能。 因此，在此情況下，除非發生下列其中一個事件，否則 Kubernetes 不會採取 **任何動作** :

1. 節點會回復到 Kubernetes 主要節點可以與其通訊的狀態。
2. 已採取管理動作來明確刪除 Kubernetes 「主要節點」上的 Pod。 這不一定會停止 Pod 執行，但只會從 Kubernetes 儲存庫中刪除它。 因此，當局必須小心採取這項行政行動。

相關工作

第 84 頁的『使用 IBM MQ Operator 來配置佇列管理程式的高可用性』

相關參考

高可用性配置

OpenShift CP4I Kubernetes 儲存器中 IBM MQ 的災難回復

您需要考量您正在準備的災難類型。 在雲端環境中，使用可用性區域可提供特定層次的災難容錯，而且更容易使用。 如果您有奇數資料中心 (用於仲裁) 和低延遲網路鏈結，則可能執行具有多個可用性區域的單一 Red Hat OpenShift Container Platform 或 Kubernetes 叢集，每個區域位於個別實體位置。 本主題討論無法符合這些準則的災難回復考量: 即偶數個資料中心或高延遲網路鏈結。

對於災難回復，您需要考量下列各項:

- 將 IBM MQ 資料 (保留在一或多個 `PersistentVolume` 資源中) 抄寫至災難回復位置
- 使用抄寫的資料重建佇列管理程式
- IBM MQ 用戶端應用程式及其他佇列管理程式可看見的佇列管理程式網路 ID。 例如，此 ID 可以是 DNS 項目。

持續資料需要同步或非同步抄寫至災難回復站台。 這通常是儲存體提供者特有的，但也可以使用 `VolumeSnapshot` 來完成。 如需磁區 Snapshot 的相關資訊，請參閱 CSI 磁區 Snapshot 。

從災難回復時，您需要使用抄寫的資料，在新的 Kubernetes 叢集上重建佇列管理程式實例。 如果您使用 IBM MQ Operator，則需要 `QueueManager` YAML 以及 YAML ，以用於其他支援資源，例如 `ConfigMap` 或 `Secret`。

相關資訊

ha_for_ctr.dita

儲存器中 IBM MQ 的使用者鑑別和授權

IBM MQ 可以配置成使用 LDAP 使用者和群組。 或者，您可以在容器映像檔內使用本端作業系統使用者和群組。 基於安全考量， IBM MQ Operator 不容許作業系統使用者和群組的使用者。

在多租戶容器化環境中，通常會設定安全限制，以防止潛在的安全問題，例如:

- **防止在容器內使用 "root" 使用者**
- **強制使用隨機 UID**。 例如，在 Red Hat OpenShift Container Platform 中，預設 `SecurityContextConstraints` (稱為 `restricted`) 會針對每一個儲存器使用隨機化使用者 ID。
- **防止使用專用權升級**。 Linux 上的 IBM MQ 會使用專用權提升來檢查使用者的密碼-它會使用 "setuid" 程式來變成 "root" 使用者來執行此動作。

OpenShift CP4I 為了確保符合這些安全措施， IBM MQ Operator 不容許使用儲存器內作業系統程式庫上定義的 ID。 儲存器中未定義任何 `mqm` 使用者 ID 或群組。 在 IBM Cloud Pak for Integration 和 Red Hat OpenShift 中使用 IBM MQ 時，您需要將佇列管理程式配置成使用 LDAP 進行使用者鑑別和授權。 如需配置 IBM MQ 來執行此動作的相關資訊，請參閱 連線鑑別: 使用者儲存庫 和 LDAP 授權 。

Multi 規劃儲存器中 IBM MQ 的可調整性及效能

在大部分情況下，儲存器中 IBM MQ 的調整大小及效能與 IBM MQ for Multiplatforms 相同。 不過，儲存器平台還有一些額外的限制。

關於這項作業

在儲存器中規劃 IBM MQ 的可調整性及效能時，請考量下列選項:

程序

- **限制執行緒及處理程序的數目**。

  IBM MQ 使用執行緒來管理並行。 在 Linux 中，執行緒會實作為處理程序，因此您可能會遇到儲存器平台或作業系統對處理程序數目上限所強制的限制。 在 Red Hat OpenShift Container Platform 中，每個儲存器的預設限制為 4096 個處理程序 (OpenShift 4.11 之前為 1024 個處理程序)。 雖然這適用於絕大多數的實務範例，但在某些情況下，這可能會影響佇列管理程式的用戶端連線數目。

  叢集管理者可以使用 kubelet 配置設定 **podPidsLimit** 來配置 Kubernetes 中的程序限制。 請參閱 Kubernetes 文件中的 處理程序 ID 限制和保留 。 在 Red Hat OpenShift Container Platform 中，您也可以 建立 **ContainerRuntimeConfig** 自訂資源以編輯 CRI-O 參數。

  在 IBM MQ 配置中，您也可以設定佇列管理程式的用戶端連線數目上限。 請參閱 伺服器連線通道限制 ，以瞭解將限制套用至個別伺服器連線通道，以及 MAXCHANNELS INI 屬性 ，以瞭解將限制套用至整個佇列管理程式。

- **限制磁區數目**。

  在雲端及儲存器系統中，通常會使用網路連接儲存磁區。 可以連接至 Linux 節點的磁區數目有一些限制。 例如， AWS EC2 限制為每個 VM 不超過 30 個磁區。 Red Hat OpenShift Container Platform 有類似的限制，與 Microsoft Azure 和 Google Cloud Platform 一樣。

  「原生 HA」佇列管理程式需要三個實例中的每一個實例都有一個磁區，並強制實例分散在節點之間。 不過，您可以將佇列管理程式配置成每個實例使用三個磁區 (佇列管理程式資料、回復日誌及持續保存資料)。

- **使用 IBM MQ 調整大小技術**。

  使用 IBM MQ 調整大小技術 (例如 IBM MQ 統一叢集) 來執行多個具有相同配置的佇列管理程式，會有助於取代少數大型佇列管理程式。 這會增加減少單一容器重新啟動 (例如，作為容器平台維護的一部分) 的影響的好處。

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

IBM MQ Operator 會將 IBM MQ 部署並管理為 IBM Cloud Pak for Integration 的一部分，或在 Red Hat OpenShift Container Platform 上獨立部署並管理

程序

OpenShift CP4I CD EUS IBM MQ Operator 的發行歷程

IBM MQ Operator

IBM MQ Operator 1.8.2

CD

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

操作員通道
v1.8

`.spec.version` 容許的值
9.1.5.0-r2, 9.2.0.0-r1, 9.2.0.0-r2, 9.2.0.0-r3, 9.2.0.1-r1-eus, 9.2.0.2-r1-eus, 9.2.0.2-r2-eus, 9.2.0.5-r1-eus, 9.2.0.5-r2-eus, 9.2.0.5-r3-eus, 9.2.1.0-r1, 9.2.1.0-r2, 9.2.2.0-r1, 9.2.3.0-r1, 9.2.4.0-r1, 9.2.5.0-r1, 9.2.5.0-r2, 9.2.5.0-r3

Red Hat OpenShift Container Platform 版本
Red Hat OpenShift Container Platform 4.6 及以上版本

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

新增功能

- 以 IBM MQ Operator 1.8.0 為建置基礎的僅安全更新。
- 此 Security Bulletin 中詳述已解決的漏洞。

IBM MQ Operator 1.8.1

CD

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

操作員通道
v1.8

`.spec.version` 容許的值
9.1.5.0-r2, 9.2.0.0-r1, 9.2.0.0-r2, 9.2.0.0-r3, 9.2.0.1-r1-eus, 9.2.0.2-r1-eus, 9.2.0.2-r2-eus, 9.2.1.0-r1, 9.2.1.0-r2, 9.2.2.0-r1, 9.2.3.0-r1, 9.2.4.0-r1, 9.2.5.0-r1, 9.2.5.0-r2

Red Hat OpenShift Container Platform 版本
Red Hat OpenShift Container Platform 4.6 及以上版本

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

新增功能

- 以 IBM MQ Operator 1.8.0 為建置基礎的僅安全更新。
- 此 Security Bulletin 中詳述已解決的漏洞。

IBM MQ Operator 1.8.0

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

操作員通道
`v1.8`

`.spec.version` 容許的值
9.1.5.0-r2, 9.2.0.0-r1, 9.2.0.0-r2, 9.2.0.0-r3, 9.2.0.1-r1-eus, 9.2.0.2-r1-eus, 9.2.0.2-r2-eus, 9.2.1.0-r1, 9.2.1.0-r2, 9.2.2.0-r1, 9.2.3.0-r1, 9.2.4.0-r1, 9.2.5.0-r1

Red Hat OpenShift Container Platform 版本
Red Hat OpenShift Container Platform 4.6 及以上版本

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

新增功能

- 新增已淘汰 IBM MQ 版本的狀態條件。

變更的內容

- 映像檔已從 Docker Hub 移至 IBM Container Registry。
  - 具有防火牆規則的客戶可能需要調整它們，才能存取 IBM Container Registry 上的映像檔。
  - 當升級至 IBM MQ Operator 1.8.0 時，氣隙客戶會經歷節點重新啟動。
- 已淘汰的版本: IBM MQ 9.1.5、 9.2.0 CD、 9.2.1、 9.2.2。 未來版本的 IBM MQ Operator 可能不會核對這些版本。
- 授權邏輯的變更: 升級至 IBM MQ 9.2.5 的客戶只能使用指定的授權來使用 IBM MQ 9.2.5。 請參閱第 115 頁的『mq.ibm.com/v1beta1 的授權參考手冊』。
- 此 Security Bulletin 中詳述已解決的漏洞。

IBM MQ Operator 1.7.0

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

操作員通道
`v1.7`

`.spec.version` 容許的值
9.1.5.0-r2, 9.2.0.0-r1, 9.2.0.0-r2, 9.2.0.0-r3, 9.2.0.1-r1-eus, 9.2.0.2-r1-eus, 9.2.0.2-r2-eus, 9.2.1.0-r1, 9.2.1.0-r2, 9.2.2.0-r1, 9.2.3.0-r1, 9.2.4.0-r1

Red Hat OpenShift Container Platform 版本
Red Hat OpenShift Container Platform 4.6 及以上版本

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

新增功能

- 新增 IBM MQ 9.2.4 作為持續交付版本

IBM MQ Operator 1.6.0

CD

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

操作員通道
`v1.6`

`.spec.version` 容許的值
9.1.5.0-r2, 9.2.0.0-r1, 9.2.0.0-r2, 9.2.0.0-r3, 9.2.0.1-r1-eus, 9.2.0.2-r1-eus, 9.2.0.2-r2-eus, 9.2.1.0-r1, 9.2.1.0-r2, 9.2.2.0-r1, 9.2.3.0-r1

Red Hat OpenShift Container Platform 版本
Red Hat OpenShift Container Platform 4.6 及以上版本

IBM Cloud Pak foundational services 版本
IBM Cloud Pak foundational services 3.7 及以上版本 (v3 通道)

新增功能

- 將 IBM MQ 9.2.3 新增為持續交付版本 (amd64 僅適用於 IBM Cloud Pak for Integration 2021.2.1; amd64 或 s390x 使用 IBM MQ 授權時)
- 佇列管理程式的新可用性類型: 原生 HA。 作為 IBM Cloud Pak for Integration 2021.2.1 的一部分，可用於正式作業。

變更的內容

- IBM MQ Operator 1.6 及更新版本使用 IBM Container Registry，而非 Docker Hub。 這表示您需要使用 `icr.io` 中的 `CatalogSource`。 請參閱 第 55 頁的『在 Red Hat OpenShift 上安裝及解除安裝 IBM MQ Operator』。
- 在移至下一個抄本之前，原生 HA 漸進式更新不再等待抄本同步。
- 修正 OCP 4.7 及以上版本上的原生 HA 親緣性問題。
- 使用 CA 簽章憑證與原生 HA 搭配時的修正程式問題。

IBM MQ Operator 1.5.0

CD

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

操作員通道
`v1.5`

`.spec.version` 容許的值
9.1.5.0-r2, 9.2.0.0-r1, 9.2.0.0-r2, 9.2.0.0-r3, 9.2.0.1-r1-eus, 9.2.1.0-r1, 9.2.1.0-r2, 9.2.2.0-r1

Red Hat OpenShift Container Platform 版本
Red Hat OpenShift Container Platform 4.6 及以上版本

IBM Cloud Pak foundational services 版本
IBM Cloud Pak foundational services 3.7 及以上版本 (v3 通道)

新增功能

- 新增 IBM MQ 9.2.2 作為持續交付版本 (amd64 僅適用於 IBM Cloud Pak for Integration 2021.1.1; amd64 或 s390x 使用 IBM MQ 授權時)
- 佇列管理程式的新可用性類型: 原生 HA。 僅供評估之用，作為 IBM Cloud Pak for Integration 2021.1.1 的一部分。

- 透過提供 `ServiceMonitor` 資源，與 Red Hat OpenShift Container Platform Cluster Monitoring for Prometheus 度量整合

變更的內容

- 當您建立佇列管理程式時，依預設不再建立 IBM Licensing Operator
- 多重實例佇列管理程式的更新項目現在以漸進式順序處理。 作為此變更的一部分，已引進 Kubernetes 啟動探測，其會影響配置存活性探測時所使用的值。 啟動探測會立即啟動，然後等待佇列管理程式順利啟動。 如果啟動探測在此等待期間內的任何時間通過，則活性及就緒性探測會啟動。 先前，如果您具有啟動緩慢的佇列管理程式，則可能已增加活性探測上的 `initialDelaySeconds` 設定。 如果您這麼做，現在應該將 `initialDelaySeconds` 回復為先前的設定。
- `CustomResourceDefinition` 已從 `apiextensions.k8s.io/v1beta1` 升級至 `apiextensions.k8s.io/v1`

已知的問題和限制

- 需要 IBM Cloud Pak foundational services 3.7，其在 Identity and Access Management (IAM) 元件中包含不相容的變更。 如果您有任何使用 IBM Cloud Pak for Integration 授權的佇列管理程式，則在此升級之後，需要重新啟動佇列管理程式才能存取 Web 主控台，您也會看到登入 Web 主控台的 其他錯誤 。 在操作器升級完成之後，您可以針對您選擇的 IBM MQ 版本升級至 `.spec.version` 的最新值，來修正這些錯誤。
- 如果您是升級 MQ 版本，則漸進式更新不會自動啟動。 您需要手動刪除 Pod。

IBM MQ Operator 1.4.0

CD

IBM Cloud Pak for Integration 版本
IBM Cloud Pak for Integration 2020.4.1 (IBM MQ Operator 1.4.0 是 CD 版次，不適用於「延伸更新支援」)

操作員通道
`v1.4`

`.spec.version` 容許的值
9.1.5.0-r2, 9.2.0.0-r1, 9.2.0.0-r2, 9.2.0.1-r1-eus, 9.2.1.0-r1

Red Hat OpenShift Container Platform 版本
Red Hat OpenShift Container Platform 4.6 及以上版本

新增功能

- 新增 IBM MQ 9.2.1 作為持續交付版本
- 您現在可以將 `.spec.queueManager.route.enabled` 設為 `false` ， 以防止建立預設佇列管理程式路徑

已知的問題和限制

- 更新可用性類型為 `MultiInstance` 的 `QueueManager` 時，將會立即刪除這兩個 Pod。 它們應該都由 Red Hat OpenShift Container Platform 快速重新啟動。

IBM MQ Operator 1.3.8 (EUS)

EUS

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

操作員通道
`v1.3-eus`

`.spec.version` 容許的值
9.1.5.0-r2, 9.2.0.0-r1, 9.2.0.0-r2, 9.2.0.1-r1-eus, 9.2.0.2-r1-eus, 9.2.0.4-r1-eus, 9.2.0.5-r1-eus, 9.2.0.5-r2-eus, 9.2.0.5-r3-eus, 9.2.0.6-r1-eus, 9.2.0.6-r2-eus, 9.2.0.6-r3-eus

Red Hat OpenShift Container Platform 版本
僅限 Red Hat OpenShift Container Platform 4.6

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

新增功能

- 新增運算元版本 9.2.0.6-r3-eus。
- 此 Security Bulletin 中詳述已解決的漏洞。

IBM MQ Operator 1.3.7 (EUS)

EUS

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

操作員通道
`v1.3-eus`

`.spec.version` 容許的值
9.1.5.0-r2, 9.2.0.0-r1, 9.2.0.0-r2, 9.2.0.1-r1-eus, 9.2.0.2-r1-eus, 9.2.0.4-r1-eus, 9.2.0.5-r1-eus, 9.2.0.5-r2-eus, 9.2.0.5-r3-eus, 9.2.0.6-r1-eus, 9.2.0.6-r2-eus

Red Hat OpenShift Container Platform 版本
僅限 Red Hat OpenShift Container Platform 4.6

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

新增功能

- 新增運算元版本 9.2.0.6-r2-eus。
- 此 Security Bulletin 中詳述已解決的漏洞。

IBM MQ Operator 1.3.6 (EUS)

EUS

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

操作員通道
`v1.3-eus`

`.spec.version` 容許的值
9.1.5.0-r2, 9.2.0.0-r1, 9.2.0.0-r2, 9.2.0.1-r1-eus, 9.2.0.2-r1-eus, 9.2.0.4-r1-eus, 9.2.0.5-r1-eus, 9.2.0.5-r2-eus, 9.2.0.5-r3-eus, 9.2.0.6-r1-eus

Red Hat OpenShift Container Platform 版本
僅限 Red Hat OpenShift Container Platform 4.6

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

新增功能

- 新增運算元版本 9.2.0.6-r1-eus。
- 此 Security Bulletin 中詳述已解決的漏洞。

IBM MQ Operator 1.3.5 (EUS)

EUS

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

操作員通道
`v1.3-eus`

`.spec.version` 容許的值
9.1.5.0-r2, 9.2.0.0-r1, 9.2.0.0-r2, 9.2.0.1-r1-eus, 9.2.0.2-r1-eus, 9.2.0.4-r1-eus, 9.2.0.5-r1-eus, 9.2.0.5-r2-eus, 9.2.0.5-r3-eus

Red Hat OpenShift Container Platform 版本
僅限 Red Hat OpenShift Container Platform 4.6

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

新增功能

- 新增運算元版本 9.2.0.5-r3-eus。
- 此 Security Bulletin 中詳述已解決的漏洞。

IBM MQ Operator 1.3.4 (EUS)

EUS

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

操作員通道
`v1.3-eus`

`.spec.version` 容許的值
9.1.5.0-r2, 9.2.0.0-r1, 9.2.0.0-r2, 9.2.0.1-r1-eus, 9.2.0.2-r1-eus, 9.2.0.4-r1-eus, 9.2.0.5-r1-eus, 9.2.0.5-r2-eus

Red Hat OpenShift Container Platform 版本
僅限 Red Hat OpenShift Container Platform 4.6

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

新增功能

- 新增運算元版本 9.2.0.5-r2-eus
- 此 Security Bulletin 中詳述已解決的漏洞。

IBM MQ Operator 1.3.3 (EUS)

EUS

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

操作員通道
`v1.3-eus`

`.spec.version` 容許的值
9.1.5.0-r2, 9.2.0.0-r1, 9.2.0.0-r2, 9.2.0.1-r1-eus, 9.2.0.2-r1-eus, 9.2.0.4-r1-eus, 9.2.0.5-r1-eus

Red Hat OpenShift Container Platform 版本
僅限 Red Hat OpenShift Container Platform 4.6

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

新增功能

- 新增運算元版本 9.2.0.5-r1-eus
- 此 Security Bulletin 中詳述已解決的漏洞。

IBM MQ Operator 1.3.2 (EUS)

EUS

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

操作員通道
`v1.3-eus`

`.spec.version` 容許的值
9.1.5.0-r2, 9.2.0.0-r1, 9.2.0.0-r2, 9.2.0.1-r1-eus, 9.2.0.2-r1-eus, 9.2.0.4-r1-eus

Red Hat OpenShift Container Platform 版本
僅限 Red Hat OpenShift Container Platform 4.6

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

新增功能

- 新增運算元版本 9.2.0.4-r1-eus

IBM MQ Operator 1.3.1 (EUS)

EUS

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

操作員通道
`v1.3-eus`

`.spec.version` 容許的值
9.1.5.0-r2, 9.2.0.0-r1, 9.2.0.0-r2, 9.2.0.1-r1-eus, 9.2.0.2-r1-eus

Red Hat OpenShift Container Platform 版本
僅限 Red Hat OpenShift Container Platform 4.6

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

新增功能

- 新增運算元版本 9.2.0.2-r1-eus

IBM MQ Operator 1.3.0 (EUS)

EUS

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

操作員通道
`v1.3-eus`

`.spec.version` 容許的值
9.1.5.0-r2, 9.2.0.0-r1, 9.2.0.0-r2, 9.2.0.1-r1-eus

Red Hat OpenShift Container Platform 版本
僅限 Red Hat OpenShift Container Platform 4.6

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

新增功能

- 使用 IBM Cloud Pak for Integration 授權時，針對以 `-eus` 結尾的 `.spec.version` 欄位提供「延伸更新支援 (EUS)」
- 使用 `.spec.labels` 和 `.spec.annotations` 在 `QueueManager` 資源上新增設定標籤和註釋的新方式

變更的內容

- 嘗試從單一實例變更為多重實例時，改善錯誤處理
- 改良 QueueManager 內容在 IBM Cloud Pak for Integration Platform Navigator 中的呈現方式，以及 Red Hat OpenShift Container Platform Web 主控台的 "表單視圖"
- 將使用 IBM Cloud Pak for Integration 授權時的預設授權標準修正為 VirtualProcessorCore
- 修正 Red Hat OpenShift Container Platform Web 主控台中 QueueManager 的 **資源** 標籤，現在可正確顯示該佇列管理程式的 IBM MQ Operator 所管理的資源

已知的問題和限制

- 更新可用性類型為 MultiInstance 的 QueueManager 時，將會立即刪除這兩個 Pod。 它們應該都由 Red Hat OpenShift Container Platform 快速重新啟動。

IBM MQ Operator 1.2.0

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

操作員通道
v1.2

.spec.version 容許的值
9.1.5.0-r2, 9.2.0.0-r1, 9.2.0.0-r2

Red Hat OpenShift Container Platform 版本
Red Hat OpenShift Container Platform 4.4 及以上版本

新增功能

- 新增 z/Linux 支援
- 將更詳細的狀態條件新增至 QueueManager 資源。 如需相關資訊，請參閱 第 131 頁的『QueueManager 的狀態條件 (mq.ibm.com/v1beta1)』
- 新增其他執行時期檢查，以防止使用無效的儲存類別。 如需相關資訊，請參閱 第 104 頁的『停用執行時期 Webhook 檢查』
- 簡化多重實例佇列管理程式的體驗: 現在只能在 QueueManager 資源中選擇一個內容 (.spec.queueManager.availability.type)
- 透過在 QueueManager 資源中引進 .spec.queueManager.storage.defaultClass 內容，簡化選擇非預設儲存類別的作業

變更的內容

- 改良 QueueManager 內容在 IBM Cloud Pak for Integration Platform Navigator 中的呈現方式，以及 Red Hat OpenShift Container Platform Web 主控台的 "表單視圖"
- 如果升級的佇列管理程式版本可用，現在會在 IBM Cloud Pak for Integration Platform Navigator 中標示它

IBM MQ Operator 1.1.0

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

操作員通道
v1.1

.spec.version 容許的值
9.1.5.0-r2, 9.2.0.0-r1

Red Hat OpenShift Container Platform 版本
Red Hat OpenShift Container Platform 4.4 及以上版本

新增功能

- 新增 IBM MQ Advanced 9.2.0 作為持續交付版本
- 新增功能以在 ConfigMap 或「密鑰」中指定 INI 及 MQSC 資訊
- 使用 Red Hat OpenShift Container Platform Web 主控台時啟用網目導覽器

變更的內容

- 修正網路原則的問題，影響 IBM Cloud 上的 Red Hat OpenShift
- 改良驗證 Web 連結鉤，以防止 `QueueManager` 資源中的設定組合無效

IBM MQ Operator 1.0.0

CD

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

操作員通道
`v1.0`

`.spec.version` 容許的值
9.1.5.0-r2

Red Hat OpenShift Container Platform 版本
Red Hat OpenShift Container Platform 4.4 及以上版本

新增功能

- 操作器的起始版本，簡介 `mq.ibm.com/v1beta1` API

與 *IBM MQ Operator* 搭配使用的佇列管理程式儲存器映像檔

9.2.5.0-r3

CD

必要的操作員版本
1.8.2 或更高版本

支援的架構
`amd64`, `s390x`

映像檔

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

新增功能

- IBM MQ 9.2.5 的新增功能

變更的內容

- IBM MQ 9.2.5 中的變更內容
- 以 Red Hat Universal Base 映像檔 8.6-751 為基礎

9.2.5.0-r2

CD

必要的操作員版本
1.8.1 或更新版本

支援的架構
amd64, s390x

映像檔

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

新增功能

- IBM MQ 9.2.5 的新增功能

變更的內容

- IBM MQ 9.2.5 中的變更內容
- 以 Red Hat Universal Base 映像檔 8.5-240.1648458092 為基礎

9.2.5.0-r1

CD

必要的操作員版本
1.8.0 或更新版本

支援的架構
amd64, s390x

映像檔

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

新增功能

- IBM MQ 9.2.5 的新增功能

變更的內容

- IBM MQ 9.2.5 中的變更內容
- 現在已從 IBM MQ Console 中移除無效的 **遠端佇列管理程式** 選項
- 以 Red Hat 通用基本映像檔 8.5-240 為基礎

9.2.4.0-r1

CD

必要的操作員版本
1.7.0 或更高版本

支援的架構
amd64, s390x

映像檔

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

新增功能

- IBM MQ 9.2.4 的新增功能

變更的內容

- IBM MQ 9.2.4 中的變更內容

- 以 Red Hat 通用基本映像檔 8.5-204 為基礎

9.2.3.0-r1

CD

必要的操作員版本
1.6.0 或更新版本

支援的架構
amd64, s390x

映像檔

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

新增功能

- IBM MQ 9.2.3 的新增功能
- 與 IBM Cloud Pak for Integration 授權一起使用時，支援 MQ 原生 HA 用於正式作業。 請注意，在具有 IBM MQ 9.2.2 的評估授權下使用原生 HA 的佇列管理程式無法升級至 9.2.3。 評估期間已結束。

變更的內容

- IBM MQ 9.2.3 中的變更內容
- 以 Red Hat 通用基本映像檔 8.4-205 為基礎

9.2.2.0-r1

CD

必要的操作員版本
1.5.0 或更新版本

支援的架構
amd64, s390x

映像檔

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

新增功能

- IBM MQ 9.2.2 的新增功能
- 與 IBM Cloud Pak for Integration 授權搭配使用時，基於評估目的支援 MQ 原生 HA

變更的內容

- IBM MQ 9.2.2 中的變更內容
- 已修正導致 FDC 關閉 IBM MQ Advanced for Developers 佇列管理程式的問題
- 以 Red Hat Universal Base 映像檔 8.3-291 為基礎

9.2.1.0-r2

CD

必要的操作員版本
1.5.0 或更新版本

支援的架構
amd64, s390x

映像檔

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

變更的內容

- 修正 IBM Cloud Pak foundational services 3.7 及更新版本的單一登入問題。
- 以 Red Hat Universal Base 映像檔 8.3-291 為基礎

9.2.1.0-r1

CD

必要的操作員版本
1.4.0 或更新版本

支援的架構
amd64, s390x

映像檔

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

新增功能

- IBM MQ 9.2.1 的新增功能
- MQ Web 主控台中會提供預設路徑的連線資訊

變更的內容

- IBM MQ 9.2.1 中的變更內容
- 以 Red Hat Universal Base 映像檔 8.3-230 為基礎

9.2.0.6-r3-eus

EUS

必要的操作員版本
1.3.8 及未來修正套件

支援的架構
amd64, s390x

映像檔

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

變更的內容

- 包括 IBM MQ 9.2.0 Fix Pack 6。 如需相關資訊，請參閱 IBM MQ 9.2 LTS 版的修正程式清單。
- 以 Red Hat Universal Base 映像檔 8.6-941 為基礎。

9.2.0.6-r2-eus

EUS

必要的操作員版本
1.3.7 及未來修正套件

支援的架構
amd64, s390x

映像檔

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

變更的內容

- 包括 IBM MQ 9.2.0 Fix Pack 6。 如需相關資訊，請參閱 IBM MQ 9.2 LTS 版的修正程式清單。
- 以 Red Hat Universal Base Image 8.6-902 為基礎。

9.2.0.6-r1-eus

EUS

必要的操作員版本
1.3.6 及未來修正套件

支援的架構
amd64, s390x

映像檔

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

變更的內容

- 包括 IBM MQ 9.2.0 Fix Pack 6。 如需相關資訊，請參閱 IBM MQ 9.2 LTS 版的修正程式清單。
- 以 Red Hat Universal Base 映像檔 8.6-854 為基礎。

9.2.0.5-r3-eus

EUS

必要的操作員版本
1.3.5 及未來修正套件

支援的架構
amd64, s390x

映像檔

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

變更的內容

- 包括 IBM MQ 9.2.0 Fix Pack 5。 如需相關資訊，請參閱 IBM MQ 9.2.0 Fix Pack 5 中的變更內容 及 IBM MQ 9.2 版 LTS 的修正程式清單。
- 以 Red Hat Universal Base Image 8.6-751.1655117800 為基礎。

9.2.0.5-r2-eus

EUS

必要的操作員版本
1.3.4 及未來修正套件

支援的架構
amd64, s390x

映像檔

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

變更的內容

- 包括 IBM MQ 9.2.0 Fix Pack 5。 如需相關資訊，請參閱 IBM MQ 9.2.0 Fix Pack 5 中的變更內容 及 IBM MQ 9.2 版 LTS 的修正程式清單
- 以 Red Hat Universal Base 映像檔 8.6-751 為基礎

9.2.0.5-r1-eus

EUS

必要的操作員版本
1.3.3 及未來修正套件

支援的架構
amd64, s390x

映像檔

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

變更的內容

- 包括 IBM MQ 9.2.0 Fix Pack 5。 如需相關資訊，請參閱 IBM MQ 9.2.0 Fix Pack 5 中的變更內容 及 IBM MQ 9.2 版 LTS 的修正程式清單
- 以 Red Hat Universal Base 映像檔 8.5-240.1648458092 為基礎

9.2.0.4-r1-eus

EUS

必要的操作員版本
1.3.2 及未來修正套件

支援的架構
amd64, s390x

映像檔

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

變更的內容

- 包括 IBM MQ 9.2.0 Fix Pack 4。 如需相關資訊，請參閱 IBM MQ 9.2.0 Fix Pack 4 中的變更內容 及 IBM MQ 9.2 版 LTS 的修正程式清單
- 以 Red Hat Universal Base 映像檔 8.5-204 為基礎

9.2.0.2-r2-eus

EUS

必要的操作員版本
1.6.0 或更新版本

支援的架構
amd64, s390x

映像檔

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

變更的內容

- 修正 IBM Cloud Pak foundational services 3.7 及更高版本的單一登入問題，只有在從 EUS 版本移轉至 CD 版本時才需要。
- 以 Red Hat Universal Base 映像檔 8.4-200.1622548483 為基礎

9.2.0.2-r1-eus

EUS

必要的操作員版本
1.3.1 及未來修正套件 ; 1.6.0 或更新版本

支援的架構
amd64, s390x

映像檔

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

變更的內容

- 作業儀表板整合使用追蹤代理程式及收集器 1.0.8 版
- 包括 IBM MQ 9.2.0 Fix Pack 2。 如需相關資訊，請參閱 IBM MQ 9.2.0 Fix Pack 2 中的變更內容 及 IBM MQ 9.2 版 LTS 的修正程式清單
- 以 Red Hat Universal Base 映像檔 8.4-200.1622548483 為基礎

9.2.0.1-r1-eus

EUS

必要的操作員版本
1.3.0 或更新版本

支援的架構
amd64, s390x

映像檔

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

新增功能

- 僅在使用 IBM Cloud Pak for Integration 授權時可用
- 在 Red Hat OpenShift Container Platform 4.6 上使用 IBM MQ Operator 1.3.x 及 IBM Common Services 3.6 時，可以使用「延伸更新支援 (EUS)」。

變更的內容

- 包括 IBM MQ 9.2.0 Fix Pack 1。 如需相關資訊，請參閱 IBM MQ 9.2.0 Fix Pack 1 中的變更內容 及 IBM MQ 9.2 版 LTS 的修正程式清單
- 以 Red Hat Universal Base 映像檔 8.3-201 為基礎
- 在容許專用權提升的 `SecurityContextContraints` 下執行時，已修正存活性探測 (`chkmqhealthy`) 及就緒性探測 (`chkmqready`) 的問題。

9.2.0.0-r3

CD

必要的操作員版本
1.5.0 或更新版本

支援的架構
amd64, s390x

映像檔

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

變更的內容

- 以 Red Hat Universal Base 映像檔 8.3-291 為基礎

9.2.0.0-r2

CD

必要的操作員版本
1.2.0 或更新版本

支援的架構
amd64, s390x

映像檔

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

新增功能

- 現在可在 z/Linux 上使用

變更的內容

- 以 Red Hat Universal Base 映像檔 8.2-349 為基礎

9.2.0.0-r1

必要的操作員版本
1.1.0 或更高版本

支援的架構
amd64

映像檔

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

新增功能

- IBM MQ 9.2.0 的新增功能

變更的內容

- IBM MQ 9.2.0 中的變更內容
- 使用 crtmqm 的 -ic 引數來自動套用 MQSC 檔案。 取代先前使用的 runmqsc 指令
- 根據 Red Hat Universal Base 映像檔 8.2-301.1593113563

9.1.5.0-r2

必要的操作員版本
1.0.0 或更新版本

支援的架構
amd64

映像檔

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

變更的內容

- 以 Red Hat Universal Base 映像檔 8.2-267 為基礎

V 9.2.1 OpenShift CD EUS 將 IBM MQ 移轉至 IBM Cloud Pak for Integration

這組主題說明使用 IBM Cloud Pak for Integration 中的 IBM MQ Operator，將現有 IBM MQ 佇列管理程式移轉至儲存器環境的主要步驟。

關於這項作業

在 Red Hat OpenShift 上部署 IBM MQ 的用戶端可以分成下列實務範例:

1. 在 Red Hat OpenShift 中為新的應用程式建立新的 IBM MQ 部署。
2. 針對 Red Hat OpenShift 中的新應用程式，將 IBM MQ 網路延伸至 Red Hat OpenShift 。
3. 將 IBM MQ 部署移至 Red Hat OpenShift 以繼續支援現有應用程式。

僅適用於實務範例 3，您需要移轉 IBM MQ 配置。 其他實務範例視為新的部署。

這組主題著重於實務範例 3，並說明使用 IBM MQ Operator 將現有 IBM MQ 佇列管理程式移轉至儲存器環境的主要步驟。 由於 IBM MQ 的彈性及廣泛使用，有數個選用步驟。 每一個包括「我是否需要執行此動作」區段。 驗證您的需求應該可以在移轉期間節省您的時間。

您也需要考量要移轉哪些資料:

1. 移轉具有相同配置但沒有任何現有佇列訊息的 IBM MQ 。
2. 移轉具有相同配置及現有訊息的 IBM MQ 。

一般版本至版本移轉可以使用任一方法。 在移轉時的一般 IBM MQ 佇列管理程式中，如果有任何訊息儲存在佇列中，則很少會讓選項 1 適用於許多情況。 在移轉至容器平台的情況下，更常見的情況是使用選項 1，以減少移轉的複雜性並容許藍色綠色部署。 因此，指示著重於此實務範例。

此實務範例的目標是在儲存器環境中建立符合現有佇列管理程式定義的佇列管理程式。 這可讓現有的網路連接應用程式只重新配置成指向新的佇列管理程式，而不變更任何其他配置或應用程式邏輯。

在這項移轉期間，您會產生多個配置檔，以套用至新的佇列管理程式。 若要簡化這些檔案的管理，您應該建立一個目錄，並將它們產生到該目錄中。

程序

1. 第 34 頁的『檢查必要功能是否可用』
2. 第 35 頁的『擷取佇列管理程式配置』
3. 選擇性的: 第 36 頁的『選用項目: 擷取並獲得佇列管理程式金鑰及憑證』
4. 選擇性的: 第 37 頁的『選用項目: 配置 LDAP』
5. 選擇性的: 第 44 頁的『選用項目: 變更 IBM MQ 配置中的 IP 位址和主機名稱』
6. 第 45 頁的『更新儲存器環境的佇列管理程式配置』
7. 第 48 頁的『選取在儲存器中執行之 IBM MQ 的目標 HA 架構』
8. 第 49 頁的『建立佇列管理程式的資源』
9. 第 50 頁的『在 Red Hat OpenShift 上建立新的佇列管理程式』
10. 第 54 頁的『驗證新的儲存器部署』

V 9.2.1 OpenShift CD EUS 檢查必要功能是否可用

IBM MQ Operator 不包含 IBM MQ Advanced 內可用的所有特性，您必須驗證這些特性並非必要。 其他特性部分受支援，並且可以重新配置以符合儲存器中可用的特性。

開始之前

這是 第 34 頁的『將 IBM MQ 移轉至 IBM Cloud Pak for Integration』中的第一個步驟。

程序

1. 請驗證目標儲存器映像檔包含所有必要的功能。

   如需最新資訊，請參閱 第 5 頁的『選擇您要如何在儲存器中使用 IBM MQ』。
2. IBM MQ Operator 具有單一 IBM MQ 資料流量埠，稱為接聽器。 如果您有多個接聽器，請將此簡化為在儲存器中使用單一接聽器。 因為這不是一般實務範例，所以未詳細記載此修改。
3. 如果使用 IBM MQ 結束程式，請在 IBM MQ 結束程式二進位檔中分層將它們移轉至儲存器。 這是進階移轉實務範例，因此不包含在這裡。 如需步驟的大綱，請參閱 第 102 頁的『使用 Red Hat OpenShift CLI 以自訂 MQSC 及 INI 檔案建置映像檔』。
4. 如果 IBM MQ 系統包括「高可用性」，請檢閱可用的選項。

   請參閱 第 15 頁的『儲存器中 IBM MQ 的高可用性』。

下一步

現在您已準備好 擷取佇列管理程式配置。

擷取佇列管理程式配置

大部分配置在佇列管理程式之間是可攜的。 例如，應用程式與之互動的事物，例如佇列、主題及通道的定義。 請利用這項作業，從現有的 IBM MQ 佇列管理程式擷取配置。

開始之前

這項作業假設您已 檢查必要的功能是否可用。

程序

1. 使用現有的 IBM MQ 安裝來登入機器。
2. 備份配置。

   請執行下列指令：

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

   此指令的使用注意事項：

   - 此指令會將備份儲存在 `tmp` 目錄中。 您可以將備份儲存在另一個位置，但此實務範例假設後續指令的 `tmp` 目錄。
   - 將 *QMGR_NAME* 取代為環境中的佇列管理程式名稱。 如果您不確定該值，請執行 **dspmq** 指令，以檢視機器上可用的佇列管理程式。 以下是名為 `qm1` 之佇列管理程式的 **dspmq** 指令輸出範例：

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

     dspmq 指令需要啟動 IBM MQ 佇列管理程式，否則您會收到下列錯誤：

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

     必要的話，請執行下列指令來啟動佇列管理程式：

     ```
     strmqm QMGR_NAME
     ```

下一步

現在您已準備好 擷取並獲得佇列管理程式金鑰及憑證。

V 9.2.1 OpenShift CD EUS

選用項目: 擷取並獲得佇列管理程式金鑰及憑證

可以使用 TLS 來配置 IBM MQ，以加密進入佇列管理程式的資料流量。使用此作業來驗證佇列管理程式是否使用 TLS、擷取金鑰及憑證，以及在移轉的佇列管理程式上配置 TLS。

開始之前

這項作業假設您已 擷取佇列管理程式配置。

關於這項作業

我需要這麼做嗎?

IBM MQ 可以配置為加密進入佇列管理程式的資料流量。此加密是使用佇列管理程式上所配置的金鑰儲存庫來完成。然後，IBM MQ 通道會啟用 TLS 通訊。如果您不確定它是否已在環境中配置，請執行下列指令來驗證:

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

如果找不到任何結果，則不會使用 TLS。不過，這並不表示不應在移轉的佇列管理程式中配置 TLS。您可能想要變更此行為有數個原因:

- 與前一個環境相比，Red Hat OpenShift 環境上的安全方法應該得到加強。
- 如果您需要從 Red Hat OpenShift 環境外部存取已移轉的佇列管理程式，則需要 TLS 才能通過 Red Hat OpenShift 路徑。

程序

1. 從現有儲存庫中擷取任何授信憑證。

   如果目前在佇列管理程式上使用 TLS，則佇列管理程式可能會儲存一些授信憑證。這些需要擷取並複製到新的佇列管理程式。請完成下列其中一個選用步驟:

   - 若要簡化憑證的擷取，請在本端系統上執行下列 Script:

     ```
     #!/bin/bash

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

     執行 Script 時，請指定 IBM MQ 備份的位置作為引數，並擷取憑證。例如，如果 Script 稱為 `extractCert.sh`，且 IBM MQ 備份位於 `/tmp/backup.mqsc`，則執行下列指令:

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

   - 或者，依顯示的順序執行下列指令:

     a. 識別 TLS 儲存庫的位置:

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

範例輸出：

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

金鑰儲存庫位於 /run/runmqserver/tls/key.kdb

b. 根據此位置資訊，查詢金鑰儲存庫以判定儲存的憑證：

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

範例輸出：

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

c. 擷取每一個列出的憑證。 執行下列指令來執行此動作：

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

在先前顯示的範例中，這等同於下列：

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

2. 獲得佇列管理程式的新金鑰和憑證

若要在移轉的佇列管理程式上配置 TLS，您可以產生新的金鑰及憑證。 然後會在部署期間使用此項目。 在許多組織中，這表示聯絡您的安全團隊以要求金鑰及憑證。 在部分組織中無法使用此選項，並使用自簽憑證。

下列範例會產生自簽憑證，其中期限設為 10 年：

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

會建立兩個新檔案：

- qmgr.key 是佇列管理程式的私密金鑰
- qmgr.crt 是公用憑證

下一步

現在您已準備好 配置 LDAP。

V 9.2.1 OpenShift CD EUS

選用項目: 配置 LDAP

IBM MQ Operator 可以配置成使用數種不同的安全方法。 一般而言， LDAP 對企業部署最有效，且 LDAP 用於此移轉實務範例。

開始之前

這項作業假設您已 擷取並獲得佇列管理程式金鑰和憑證。

關於這項作業

我需要這麼做嗎?

如果您已使用 LDAP 進行鑑別及授權，則不需要進行任何變更。

如果您不確定是否正在使用 LDAP，請執行下列指令:

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

範例輸出:

```
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
```

輸出中有兩個特別感興趣的屬性:

AUTHTYPE
如果此值為 IDPWLDAP，則您是使用 LDAP 進行鑑別。

如果值為空白或另一個值，則不會配置 LDAP。 在此情況下，請檢查 AUTHORMD 屬性，以查看是否使用 LDAP 使用者進行授權。

AUTHORMD
如果此值為 OS，則您不會使用 LDAP 進行授權。

若要修改授權及鑑別以使用 LDAP，請完成下列作業:

程序

1. 更新 LDAP 伺服器的 IBM MQ 備份。
2. 更新 IBM MQ 備份以取得 LDAP 授權資訊。

LDAP 第 1 部分: 更新 LDAP 伺服器的 IBM MQ 備份

如何設定 LDAP 的綜合性說明不在此實務範例的範圍內。 本主題提供程序、範例及進一步資訊參照的摘要。

開始之前

這項作業假設您已 擷取並獲得佇列管理程式金鑰和憑證。

關於這項作業

我需要這麼做嗎?

如果您已使用 LDAP 進行鑑別及授權，則不需要進行任何變更。 如果您不確定是否正在使用 LDAP，請參閱第 37 頁的『選用項目: 配置 LDAP』。

設定 LDAP 伺服器有兩個部分:

1. 定義 LDAP 配置。
2. 建立 LDAP 配置與佇列管理程式定義的關聯。

可協助您使用此配置的進一步資訊:

- 使用者儲存庫概觀
- AUTHINFO 指令參考手冊

程序

1. 定義 LDAP 配置。

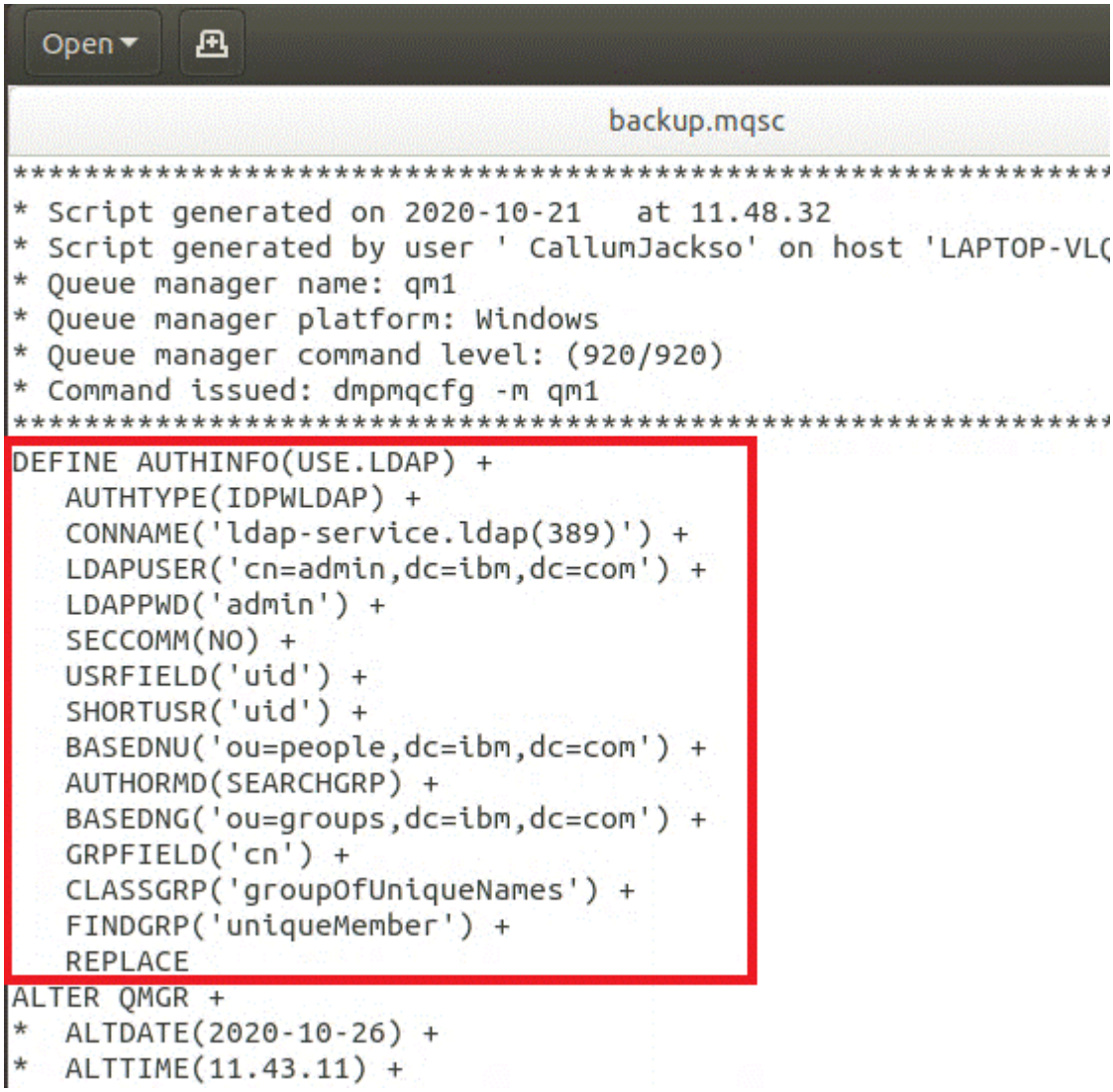
   編輯 backup.mqsc 檔案，為 LDAP 系統定義新的 **AUTHINFO** 物件。 例如:

```
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** 是對應於 LDAP 伺服器的主機名稱及埠。 如果有多個位址用於備援，則可以使用逗點區隔清單來配置這些位址。
   - **LDAPUSER** 是對應於 IBM MQ 在連接至 LDAP 以查詢使用者記錄時所使用之使用者的識別名稱。
   - **LDAPPWD** 是對應於 **LDAPUSER** 使用者的密碼。
   - **SECCOM** 指定與 LDAP 伺服器的通訊是否應該使用 TLS。 可能的值如下:
     - YES: 使用 TLS ， 並由 IBM MQ 伺服器提供憑證。
     - ANON: 使用 TLS 時沒有 IBM MQ 伺服器提供的憑證。
     - NO: 在連線期間不使用 TLS。
   - **USRFIELD** 指定 LDAP 記錄中要比對所呈現使用者名稱的欄位。
   - **SHORTUSR** 是 LDAP 記錄中長度不超過 12 個字元的欄位。 如果鑑別成功，則此欄位內的值是主張的身分。
   - **BASEDNU** 是應該用於搜尋 LDAP 的基本 DN。
   - **BASEDNG** 是 LDAP 內群組的基本 DN。
   - **AUTHORMD** 定義用來解析使用者群組成員資格的機制。 有四個選項:
     - OS: 查詢作業系統中與簡稱相關聯的群組。
     - SEARCHGRP: 在 LDAP 中搜尋已鑑別使用者的群組項目。
     - SEARCHUSR: 在已鑑別使用者記錄中搜尋群組成員資格資訊。
     - SRCHGRPSN: 在 LDAP 中的群組項目中搜尋已鑑別使用者的簡短使用者名稱 (由 SHORTUSR 欄位定義)。
   - **GRPFIELD** 是 LDAP 群組記錄內對應於簡式名稱的屬性。 如果指定的話，這可用來定義授權記錄。
   - **CLASSUSR** 是對應於使用者的 LDAP 物件類別。
   - **CLASSGRP** 是對應於群組的 LDAP 物件類別。
   - **FINDGRP** 是 LDAP 記錄內對應於群組成員資格的屬性。

   新項目可以放置在檔案內的任何位置，不過您可能會發現在檔案開頭具有任何新項目會很有用:



2. 將 LDAP 配置與佇列管理程式定義相關聯。

   您需要將 LDAP 配置與佇列管理程式定義相關聯。 緊接在 DEFINE AUTHINFO 項目下方的是 ALTER QMGR 項目。 修改 CONNAUTH 項目以對應新建立的 AUTHINFO 名稱。 例如，在前一個範例中已定義 AUTHINFO(USE.LDAP)，表示名稱為 USE.LDAP。 因此，將 CONNAUTH('SYSTEM.DEFAULT.AUTHINFO.IDPWOS') 變更為 CONNAUTH('USE.LDAP'):

若要立即切換至 LDAP，請在 ALTER QMGR 指令之後立即新增一行，以呼叫 REFRESH SECURITY 指令:

*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

現在，您已準備好 更新 IBM MQ LDAP 授權備份資訊。

V 9.2.1 OpenShift CD EUS

LDAP 第 2 部分: 更新 LDAP 授權資訊的 IBM MQ 備份

IBM MQ 提供細部授權規則，可控制對 IBM MQ 物件的存取權。 如果您將鑑別及授權變更為 LDAP，則授權規則可能無效且需要更新。

開始之前

這項作業假設您已 更新 LDAP 伺服器的備份。

關於這項作業

我需要這麼做嗎?

如果您已使用 LDAP 進行鑑別及授權，則不需要進行任何變更。 如果您不確定是否正在使用 LDAP，請參閱第 37 頁的『選用項目: 配置 LDAP』。

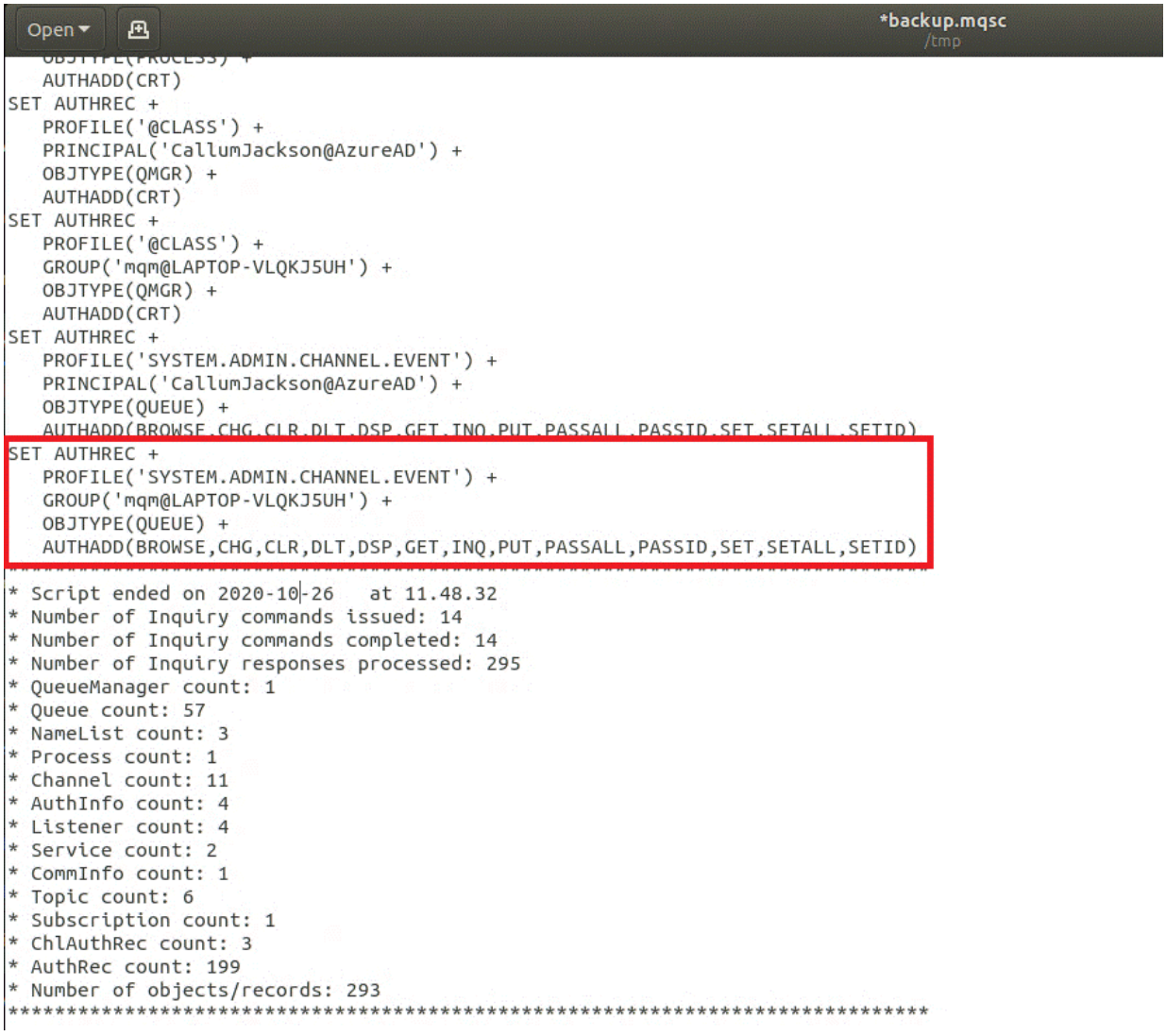
更新 LDAP 授權資訊有兩個部分:

1. 從檔案中移除所有現有的授權。
2. 定義 LDAP 的新授權資訊。

程序

1. 從檔案中移除所有現有的授權。

   在備份檔中，接近檔案結尾時，您應該會看到數個以 SET AUTHREC 開頭的項目:



   尋找現有項目並刪除它們。 最直接明確的方法是移除所有現有的 SET AUTHREC 規則，然後根據 LDAP 項目來建立新項目。

2. 定義 LDAP 的新授權資訊

視您的佇列管理程式配置及資源和群組數目而定，這可能是耗時或直接明確的活動。 下列範例假設您的佇列管理程式只有一個稱為 Q1 的佇列，且您想要容許 LDAP 群組 apps 具有存取權。

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

第一個 AUTHREC 指令會新增存取佇列管理程式的許可權，第二個指令會提供存取佇列的權限。 如果需要存取第二個佇列，則需要第三個 AUTHREC 指令，除非您決定使用萬用字元來提供更通用的存取權。

以下是另一個範例。 如果管理者群組 (稱為 admins) 需要佇列管理程式的完整存取權，請新增下列指令:

```
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

現在，您已準備好 變更 IBM MQ 配置中的 IP 位址和主機名稱。

V 9.2.1 OpenShift CD EUS 選用項目: 變更 IBM MQ 配置中的 IP 位址和主機名稱

IBM MQ 配置可能指定了 IP 位址和主機名稱。 在某些狀況下，這些可以保留，而在其他狀況下則需要更新。

開始之前

這項作業假設您已 配置 LDAP。

關於這項作業

我需要這麼做嗎?

首先，除了前一節所定義的 LDAP 配置之外，請判斷您是否已指定任何 IP 位址或主機名稱。 若要執行此動作，請執行下列指令:

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

範例輸出:

```
*****************************************************
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(' ') +
```

在此範例中，搜尋會傳回三個結果。 一個結果對應於先前定義的 LDAP 配置。 這可以忽略，因為 LDAP 伺服器的主機名稱仍然相同。 另外兩個結果是空的連線項目，因此也可以忽略這些連線項目。 如果您沒有任何其他項目，則可以跳過本主題的其餘部分。

程序

1. 瞭解傳回的項目。

   IBM MQ 可以在配置的許多方面包含 IP 位址、主機名稱和埠。 我們可以將這些分類為兩個種類:

   a. **此佇列管理程式的位置**: 此佇列管理程式使用或發佈的位置資訊，可供 IBM MQ 網路內的其他佇列管理程式或應用程式用於連線功能。

   b. **佇列管理程式相依關係的位置**: 此佇列管理程式需要知道的其他佇列管理程式或系統的位置。

   因為此實務範例僅聚焦於此佇列管理程式配置的變更，所以我們只會處理種類 (a) 的配置更新。 不過，如果其他佇列管理程式或應用程式參照此佇列管理程式位置，則其配置可能需要更新，以符合此佇列管理程式的新位置。

   有兩個金鑰物件可能包含需要更新的資訊:

   - 接聽器: 這些代表 IBM MQ 正在接聽的網址。
   - 叢集接收端通道: 如果佇列管理程式是 IBM MQ 叢集的一部分，則此物件存在。 它指定其他佇列管理程式可以連接的網址。

2. 在 `grep 'CONNAME\|LOCLADDR\|IPADDRV' -B 3 backup.mqsc` 指令的原始輸出中，識別是否定義了任何 CLUSTER RECEIVER 通道。 如果是，請更新 IP 位址。

   若要識別是否已定義任何 CLUSTER RECEIVER 通道，請在原始輸出中尋找具有 CHLTYPE (CLUSRCVR) 的任何項目:

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

   如果項目確實存在，請使用 IBM MQ Red Hat OpenShift 路徑更新 CONNAME 。 此值基於 Red Hat OpenShift 環境，並使用可預測的語法:

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

   例如，如果在 `cp4i` 名稱空間內佇列管理程式部署命名為 `qm1` ，且 *openshift_app_route_hostname* 為 `apps.callumj.icp4i.com`，則路徑 URL 如下:

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

   路徑的埠號通常是 443。 除非 Red Hat OpenShift 管理者以不同方式告訴您，否則這通常是正確的值。 使用此資訊，更新 CONNAME 欄位。 例如:

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

   在 `grep 'CONNAME\|LOCLADDR\|IPADDRV' -B 3 backup.mqsc` 指令的原始輸出中，驗證 LOCLADDR 或 IPADDRV 是否存在任何項目。 如果有的話，請刪除它們。 它們在儲存器環境中不相關。

下一步

現在，您已準備好 更新儲存器環境的佇列管理程式配置。

V 9.2.1 OpenShift CD EUS

更新儲存器環境的佇列管理程式配置

在儲存器中執行時，儲存器會定義某些配置層面，且可能與匯出的配置衝突。

開始之前

這項作業假設您已 變更 IBM MQ IP 位址和主機名稱的配置。

關於這項作業

儲存器定義下列配置層面:

- 接聽器定義 (對應於公開的埠)。
- 任何潛在 TLS 儲存庫的位置。

因此，您需要更新匯出的配置:

1. 移除任何接聽器定義。
2. 定義 TLS 金鑰儲存庫的位置。

程序

1. 移除任何接聽器定義。

   在備份配置中，搜尋 `DEFINE LISTENER`。 這應該介於 `AUTHINFO` 與 `SERVICE` 定義之間。 強調顯示區域，並刪除它。

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

2. 定義 TLS 金鑰儲存庫的位置。

   佇列管理程式備份包含原始環境的 TLS 配置。 這與儲存器環境不同，因此需要一些更新項目:

   - 將 **CERTLABL** 項目變更為 `default`
   - 將 TLS 金鑰儲存庫 (**SSLKEYR**) 的位置變更為: `/run/runmqserver/tls/key`

   若要尋找檔案中 **SSLKEYR** 屬性的位置，請搜尋 **SSLKEYR**。 通常只會找到一個項目。 如果找到多個項目，請確認您正在編輯 **QMGR** 物件，如下圖所示:

*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

現在，您已準備好 選取在儲存器中執行之 IBM MQ 的目標架構。

V 9.2.1 OpenShift CD EUS

選取在儲存器中執行之 IBM MQ 的目標 HA 架構

在單一實例 (單一 Kubernetes Pod) 與多重實例 (兩個 Pod) 之間進行選擇，以符合您的高可用性需求。

開始之前

此作業假設您已 更新儲存器環境的佇列管理程式配置。

關於這項作業

IBM MQ Operator 提供兩個高可用性選項:

- **單一實例**: 單一容器 (Pod) 已啟動，且在失敗時由 Red Hat OpenShift 負責重新啟動。 由於 Kubernetes 內有狀態集的性質，在數個狀況下，此失效接手可能需要較長的時間，或需要完成管理動作。
- **多重實例**: 啟動兩個容器 (每個位於個別 Pod 中)，一個處於作用中模式，另一個處於待命模式。 此拓蹼可啟用更快速的失效接手。 它需要符合 IBM MQ 需求的「讀寫多次」檔案系統。

在此作業中，您只選擇目標 HA 架構。 在此實務範例的後續作業中，會說明配置您所選擇架構的步驟 (第 50 頁的『在 Red Hat OpenShift 上建立新的佇列管理程式』)。

程序

1. 請檢閱兩個選項。

   如需這兩個選項的綜合性說明，請參閱 第 15 頁的『儲存器中 IBM MQ 的高可用性』。
2. 選取目標 HA 架構。

   如果您不確定要選擇哪個選項，請從 **單一實例** 選項開始，並驗證這是否符合您的高可用性需求。

下一步

現在您已準備好 建立佇列管理程式資源。

V 9.2.1 OpenShift CD EUS

建立佇列管理程式的資源

將 IBM MQ 配置以及 TLS 憑證和金鑰匯入 Red Hat OpenShift 環境。

開始之前

這項作業假設您已 選取在儲存器中執行 IBM MQ 的目標架構。

關於這項作業

在先前的區段中，您已擷取、更新並定義兩個資源:

- IBM MQ 配置
- TLS 憑證及金鑰

在部署佇列管理程式之前，您需要將這些資源匯入至 Red Hat OpenShift 環境。

程序

1. 將 IBM MQ 配置匯入至 Red Hat OpenShift。

   下列指示假設您在現行目錄中具有稱為 `backup.mqsc` 的檔案中的 IBM MQ 配置。 否則，您需要根據環境來自訂檔名。

   a) 使用 `oc login` 登入叢集。

   b) 將 IBM MQ 配置載入至 `configmap`。

   請執行下列指令:

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

   c) 驗證已順利載入檔案。

   請執行下列指令:

   ```
   oc describe configmap my-mqsc-migrated
   ```

2. 匯入 IBM MQ TLS 資源

   如 第 36 頁的『選用項目: 擷取並獲得佇列管理程式金鑰及憑證』中所述，佇列管理程式部署可能需要 TLS。 如果是這樣，您應該已有一些檔案以 .crt 和 .key 結尾。 您需要將這些密鑰新增至 Kubernetes 密鑰，以供佇列管理程式在部署時參照。

   例如，如果您具有佇列管理程式的金鑰及憑證，則可能會呼叫它們:

   - qmgr.crt
   - qmgr.key

   若要匯入這些檔案，請執行下列指令:

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

   當您匯入相符的公開和私密金鑰時， Kubernetes 會提供這個有用的公用程式。 如果您有其他憑證要新增 (例如，新增至佇列管理程式信任儲存庫)，請執行下列指令:

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

   例如，如果要匯入的檔案是 trust1.crt、 trust2.crt 和 trust3.crt，則指令如下:

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

下一步

現在您已準備好 在 Red Hat OpenShift 上建立新的佇列管理程式。

V 9.2.1 OpenShift CD EUS 在 Red Hat OpenShift 上建立新的佇列管理程式

在 Red Hat OpenShift 上部署單一實例或多重實例佇列管理程式。

開始之前

這項作業假設您已 建立佇列管理程式資源，且 已將 IBM MQ Operator 安裝至 Red Hat OpenShift。

關於這項作業

如 第 48 頁的『選取在儲存器中執行之 IBM MQ 的目標 HA 架構』中所述，有兩種可能的部署拓蹼。 因此，本主題提供兩個不同的範本:

- 部署單一實例佇列管理程式。
- 部署多重實例佇列管理程式。

重要: 根據您偏好的拓蹼，只完成兩個範本中的一個。

程序

- 部署單一實例佇列管理程式。

  移轉的佇列管理程式會使用 YAML 檔案部署至 Red Hat OpenShift 。 以下是根據先前主題中使用的名稱的範例:

  ```
  apiVersion: mq.ibm.com/v1beta1
  kind: QueueManager
  metadata:
    name: qm1
  spec:
    version: 9.2.5.0-r3
    license:
  ```

```
    accept: true
    license: L-RJON-C7QG3S
    use: "Production"
  pki:
    keys:
    - name: default
      secret:
        secretName: my-tls-migration
        items:
          - tls.key
          - tls.crt
  web:
    enabled: true
  queueManager:
    name: QM1
    mqsc:
      - configMap:
          name: my-mqsc-migrated
          items:
            - backup.mqsc
```

視您執行的步驟而定，可能需要自訂先前的 YAML。 為了協助您解決此問題，以下是此 YAML 的說明:

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

這會定義 Kubernetes 物件、類型及名稱。 唯一需要自訂作業的欄位是 name 欄位。

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

這對應於部署的版本和授權資訊。 如果您需要自訂此項，請使用 第 115 頁的『mq.ibm.com/v1beta1 的授權參考手冊』中提供的資訊。

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

若要將佇列管理程式配置為使用 TLS，它必須參照相關憑證及金鑰。 `secretName` 欄位會參照在 匯入 IBM MQ TLS 資源 區段內建立的 Kubernetes 密鑰，並且項目清單 (`tls.key` 和 `tls.crt`) 是 Kubernetes 使用 `oc create secret tls` 語法時指派的標準名稱。 如果您有其他憑證要新增至信任儲存庫，則可以用類似方式新增這些憑證，但這些項目是匯入期間使用的對應檔名。 例如，下列程式碼可用來建立信任儲存庫憑證:

```
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
```

重要: 如果不需要 TLS，請刪除 YAML 的 TLS 區段。

```
  web:
    enabled: true
```

這會啟用部署的 Web 主控台

```
  queueManager:
    name: QM1
```

這會將佇列管理程式的名稱定義為 QM1。 佇列管理程式會根據您的需求自訂，例如原始佇列管理程式名稱。

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

前一個程式碼會取回在 匯入 IBM MQ 配置 區段中匯入的佇列管理程式配置。 如果您使用不同的名稱，則需要修改 `my-mqsc-migrated` 和 `backup.mqsc`。

請注意，範例 YAML 假設 Red Hat OpenShift 環境的預設儲存類別定義為 RWX 或 RWO 儲存類別。 如果環境中未定義預設值，則您需要指定要使用的儲存類別。 您可以透過延伸 YAML 來執行此動作，如下所示:

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

新增強調顯示的文字，並自訂類別屬性以符合您的環境。 若要探索環境內的儲存類別名稱，請執行下列指令:

```
oc get storageclass
```

以下是此指令傳回的範例輸出:

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

下列程式碼顯示如何參照在 匯入 IBM MQ 配置 區段中匯入的 IBM MQ 配置。 如果您使用不同的名稱，則需要修改 `my-mqsc-migrated` 和 `backup.mqsc`。

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

您已部署單一實例佇列管理程式。 這會完成範本。 現在，您已準備好 驗證新的儲存器部署。

- 部署多重實例佇列管理程式。

  移轉的佇列管理程式會使用 YAML 檔案部署至 Red Hat OpenShift 。 下列範例基於前幾節中使用的名稱。

```
apiVersion: mq.ibm.com/v1beta1
kind: QueueManager
metadata:
  name: qm1mi
spec:
  version: 9.2.5.0-r3
  license:
```

```
    accept: true
    license: L-RJON-C7QG3S
    use: "Production"
  pki:
    keys:
    - name: default
      secret:
        secretName: my-tls-migration
        items:
          - tls.key
          - tls.crt
  web:
    enabled: true
  queueManager:
    name: QM1
    availability: MultiInstance
    storage:
      defaultClass: aws-efs
      persistedData:
        enabled: true
      queueManager:
        enabled: true
      recoveryLogs:
        enabled: true
    mqsc:
      - configMap:
          name: my-mqsc-migrated
          items:
            - backup.mqsc
```

以下是此 YAML 的說明。 大部分配置遵循與 部署單一實例佇列管理程式相同的方法，因此這裡只會說明佇列管理程式可用性及儲存體方面。

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

這會將佇列管理程式名稱指定為 QM1，並將部署設為 MultiInstance，而不是預設單一實例。

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

IBM MQ 多重實例佇列管理程式取決於 RWX 儲存體。 依預設，佇列管理程式會以單一實例模式部署，因此變更為多重實例模式時需要其他儲存體選項。 在先前的 YAML 範例中，會定義三個儲存體持續性磁區及一個持續保存磁區類別。 這個持續保存的磁區類別必須是 RWX 儲存類別。 如果您不確定環境內的儲存類別名稱，則可以執行下列指令來探索它們:

```
oc get storageclass
```

以下是此指令傳回的範例輸出:

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

下列程式碼顯示如何參照在 匯入 IBM MQ 配置 區段中匯入的 IBM MQ 配置。 如果您使用不同的名稱，則需要修改 my-mqsc-migrated 和 backup.mqsc。

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

您已部署多重實例佇列管理程式。 這會完成範本。 現在，您已準備好 驗證新的儲存器部署。

V 9.2.1 OpenShift CD EUS 驗證新的儲存器部署

既然 IBM MQ 已部署在 Red Hat OpenShift 上，您可以使用 IBM MQ 範例來驗證環境。

開始之前

這項作業假設您已 在 Red Hat OpenShift 上建立新的佇列管理程式。

重要: 此作業假設未在佇列管理程式中啟用 TLS。

關於這項作業

在這項作業中，您可以從移轉的佇列管理程式儲存器內執行 IBM MQ 範例。 不過，您可能偏好使用您自己從另一個環境執行的應用程式。

您需要下列資訊:

- LDAP 使用者名稱
- LDAP 密碼
- IBM MQ 通道名稱
- 佇列名稱

此程式碼範例使用下列設定。 請注意，您的設定將會不同。

- LDAP 使用者名稱 :mqapp
- LDAP 密碼 :mqapp
- IBM MQ 通道名稱: DEV.APP.SVRCONN
- 佇列名稱: Q1

程序

1. 對執行中 IBM MQ 儲存器執行 exec 指令。

   使用下列指令:

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

   其中 `qm1-ibm-mq-0` 是我們在 第 50 頁的『在 Red Hat OpenShift 上建立新的佇列管理程式』中部署的 Pod。 如果您呼叫不同的部署，則自訂此值。
2. 傳送訊息。

   執行下列指令:

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

   系統會提示您輸入密碼，然後您可以傳送訊息。
3. 請驗證已順利收到訊息。

   執行 GET 範例:

   ```
   ./amqsgetc Q1 QM1
   ```

結果

您已完成 第 34 頁的『將 IBM MQ 移轉至 IBM Cloud Pak for Integration』。

下一步

請使用下列資訊來協助您處理更複雜的移轉實務範例:

移轉排入佇列的訊息
若要移轉現有的佇列訊息，請遵循下列主題中的指引，在新的佇列管理程式就緒之後匯出及匯入訊息: 在兩個系統之間使用 dmpmqmsg 公用程式。

從 Red Hat OpenShift 環境外部連接至 IBM MQ
已部署的佇列管理程式可以向 Red Hat OpenShift 環境外部的 IBM MQ 用戶端及佇列管理程式公開。 此處理程序視連接至 Red Hat OpenShift 環境的 IBM MQ 版本而定。 請參閱 第 98 頁的『配置路徑以從 Red Hat OpenShift 叢集外部連接至佇列管理程式』。

OpenShift CP4I 在 Red Hat OpenShift 上安裝及解除安裝 IBM MQ Operator

可以使用 Operator Hub 將 IBM MQ Operator 安裝至 Red Hat OpenShift 。

程序

- 第 9 頁的『IBM MQ Operator 的相依關係』.
- 第 10 頁的『IBM MQ Operator 所需的叢集範圍許可權』.
- 第 55 頁的『使用 Red Hat OpenShift Web 主控台安裝 IBM MQ Operator』.
- 第 57 頁的『使用 Red Hat OpenShift CLI 安裝 IBM MQ Operator』.
- 第 61 頁的『在氣隙環境中安裝 IBM MQ Operator』.

相關工作
第 57 頁的『使用 Red Hat OpenShift Web 主控台解除安裝 IBM MQ Operator』
您可以使用 Red Hat OpenShift Web 主控台，從 Red Hat OpenShift 解除安裝 IBM MQ Operator 。

第 59 頁的『使用 Red Hat OpenShift CLI 解除安裝 IBM MQ Operator』
您可以使用 Red Hat OpenShift CLI ，從 Red Hat OpenShift 解除安裝 IBM MQ Operator 。 解除安裝程序有一些差異，視 IBM MQ Operator 是安裝在單一名稱空間中，還是已安裝且可供叢集上的所有名稱空間使用而定。

OpenShift CP4I 使用 Red Hat OpenShift Web 主控台安裝 IBM MQ Operator

可以使用 Operator Hub 將 IBM MQ Operator 安裝至 Red Hat OpenShift 。

開始之前
登入 Red Hat OpenShift 叢集 Web 主控台。

程序

1. EUS
選擇性的: 將 IBM Common Services 操作器新增至可安裝操作器清單。

註:

此步驟適用於 IBM MQ Operator 1.5 及更舊版本。 此步驟會新增個別 Common Services 型錄。 對於更新版本的操作器， Common Services 包含在 IBM 型錄中。

a) 按一下畫面右上方的加號圖示。 您會看到 **匯入 YAML** 對話框。

b) 在對話框中貼上下列資源定義。

```
apiVersion: operators.coreos.com/v1alpha1
kind: CatalogSource
metadata:
  name: opencloud-operators
  namespace: openshift-marketplace
spec:
  displayName: IBMCS Operators
```

```
  publisher: IBM
  sourceType: grpc
  image: icr.io/cpopen/ibm-common-service-catalog:latest
  updateStrategy:
    registryPoll:
      interval: 45m
```

c) 按一下**建立**。

2. 將 IBM 操作器新增至可安裝操作器清單

a) 按一下畫面右上方的加號圖示。 您會看到 **匯入 YAML** 對話框。

b) 在對話框中貼上下列資源定義。

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

c) 按一下**建立**。

3. 建立要用於 IBM MQ Operator 的名稱空間

IBM MQ Operator 可以安裝範圍為單一名稱空間或所有名稱空間。 只有在您想要安裝至尚不存在的特定名稱空間時，才需要此步驟。

a) 從導覽窗格中，按一下 **首頁** > **專案**。

即會顯示「專案」頁面。

b) 按一下**建立專案**。 即會顯示「建立專案」區域。

c) 輸入您要建立之名稱空間的詳細資料。 例如，您可以指定 "ibm-mq" 作為名稱。

d) 按一下**建立**。 即會建立 IBM MQ Operator 的名稱空間。

4. 安裝「IBM MQ Operator」。

a) 從導覽窗格中，按一下 **操作器** > **OperatorHub**。

即會顯示 OperatorHub 頁面。

b) 在 **所有項目** 欄位中，輸入 "IBM MQ"。

這時會顯示 IBM MQ 型錄項目。

c) 選取 **IBM MQ**。

即會顯示 IBM MQ 視窗。

d) 按一下**安裝**。

您會看到「建立操作員訂閱」頁面。

e) 檢閱 第 6 頁的『IBM MQ Operator 的版本支援』，以判定要選擇哪個操作員通道。

f) 將「安裝模式」設為您所建立的特定名稱空間，或整個叢集範圍。

建議選擇叢集層面範圍，因為在不同名稱空間中安裝不同版本的操作員可能會導致問題。 操作員設計為控制平面的延伸。

g) 按一下**訂閱**。

您會在「已安裝的操作器」頁面上看到 IBM MQ 。

h) 在「已安裝的操作器」頁面上檢查「操作器」的狀態，當安裝完成時，狀態會變更為「成功」。

下一步

第 73 頁的『使用 Red Hat OpenShift Web 主控台為 IBM MQ 準備 Red Hat OpenShift 專案』

OpenShift CP4I 使用 *Red Hat OpenShift Web* 主控台解除安裝 *IBM MQ Operator*

您可以使用 Red Hat OpenShift Web 主控台，從 Red Hat OpenShift 解除安裝 IBM MQ Operator 。

開始之前

登入 Red Hat OpenShift 叢集的 Web 主控台。

如果 IBM MQ Operator 安裝在叢集上的所有專案/名稱空間之間，請針對您要刪除佇列管理程式的每一個專案，重複下列程序的步驟 1-5。

程序

1. 選取 **操作器** > **已安裝的操作器**。
2. 從 **專案** 下拉清單中，選取專案。
3. 按一下 **IBM MQ** 運算子。
4. 按一下 **佇列管理程式** 標籤，以檢視此「IBM MQ Operator」所管理的佇列管理程式。
5. 刪除一或多個佇列管理程式。

   請注意，雖然這些佇列管理程式會繼續執行，但如果沒有 IBM MQ Operator，它們可能無法如預期般運作。
6. 選擇性的: 適當的話，針對您要刪除佇列管理程式的每一個專案，重複步驟 1-5。
7. 回到 **操作器** > **已安裝的操作器**。
8. 在 **IBM MQ** 操作器旁邊，按一下三個點功能表，然後選取 **解除安裝操作器**。
9. 如果您使用 Red Hat OpenShift Container Platform 4.7，您可能需要從指令行手動刪除驗證 Web 連結鉤:

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

OpenShift CP4I 使用 Red Hat OpenShift CLI 安裝 IBM MQ Operator

可以使用 Operator Hub 將 IBM MQ Operator 安裝至 Red Hat OpenShift 。

開始之前

使用 **oc login** 登入 Red Hat OpenShift 指令行介面 (CLI)。 對於這些步驟，您將需要成為叢集管理者。

程序

1. EUS

   選擇性的: 為 IBM Common Services 操作員建立 **CatalogSource** 。

   註:

   此步驟適用於 IBM MQ Operator 1.5 及更舊版本。 此步驟會新增個別 Common Services 型錄。 對於更新版本的操作器， Common Services 包含在 IBM 型錄中。

   a) 建立定義 **CatalogSource** 資源的 YAML 檔。

   建立名為 "operator-source-cs.yaml" 的檔案，其中包含下列內容:

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

```
    registryPoll:
      interval: 45m
```

b) 將 **CatalogSource** 套用至伺服器。

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

2. 為 IBM 操作員建立 **CatalogSource**

a) 建立定義 **CatalogSource** 資源的 YAML 檔

建立名為 "operator-source-ibm.yaml" 的檔案，其中包含下列內容:

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

b) 將 **CatalogSource** 套用至伺服器。

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

3. 建立要用於 IBM MQ Operator 的名稱空間

IBM MQ Operator 可以安裝範圍為單一名稱空間或所有名稱空間。 只有在您想要安裝至尚不存在的特定名稱空間時，才需要此步驟。

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

4. 從 OperatorHub 檢視叢集可用的「運算子」清單

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

5. 檢查 IBM MQ Operator ， 以驗證其支援的 InstallModes 及可用的通道

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

6. 建立 **OperatorGroup** 物件 YAML 檔案

OperatorGroup 是一個 OLM 資源，用於選取目標名稱空間，以在其中為與 **OperatorGroup** 相同的名稱空間中的所有操作員產生必要的 RBAC 存取權。

您訂閱操作員的名稱空間必須具有符合操作員 **InstallMode** 的 **OperatorGroup** ，可以是 `AllNamespaces` 或 `SingleNamespace` 模式。 如果您想要安裝的操作器使用 `AllNamespaces`，則 `openshift-operators` 名稱空間已備妥適當的 **OperatorGroup** 。

不過，如果操作器使用 `SingleNamespace` 模式，且您還沒有適當的 **OperatorGroup** ，則必須建立一個。

a) 建立名為 "mq-operator-group.yaml" 的檔案，並包含下列內容:

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

b) 建立 **OperatorGroup** 物件

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

7. 建立 **Subscription** 物件 YAML 檔，以訂閱 IBM MQ Operator 的名稱空間
   a) 檢閱 第 6 頁的『IBM MQ Operator 的版本支援』，以判定要選擇哪個操作員通道。
   b) 建立名為 "mq-sub.yaml" 的檔案，包含下列內容，但變更 **channel** 以符合您要安裝之 IBM MQ Operator 版本的通道。

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

   如需 AllNamespaces **InstallMode** 用法，請在名稱空間中指定 **openshift-operators**。否則，請指定 SingleNamespace **InstallMode** 用法的相關單一名稱空間。請注意，您應該只變更 **namespace** 欄位，而保留 **sourceNamespace** 欄位不變。
   c) 建立 **Subscription** 物件

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

8. 檢查操作員的狀態
   在操作器安裝成功之後，Pod 狀態會顯示為 *Running*。對於 AllNamespaces **InstallMode** 用法，請指定 **openshift-operators** 作為名稱空間。否則，請指定 SingleNamespace **InstallMode** 用法的相關單一名稱空間。

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

下一步

第 74 頁的『使用 Red Hat OpenShift CLI 為 IBM MQ 準備 Red Hat OpenShift 專案』

OpenShift CP4I 使用 *Red Hat OpenShift CLI* 解除安裝 *IBM MQ Operator*

您可以使用 Red Hat OpenShift CLI，從 Red Hat OpenShift 解除安裝 IBM MQ Operator。解除安裝程序有一些差異，視 IBM MQ Operator 是安裝在單一名稱空間中，還是已安裝且可供叢集上的所有名稱空間使用而定。

開始之前

使用 oc login 登入 Red Hat OpenShift 叢集。

程序

- 如果 IBM MQ Operator 安裝在單一名稱空間中，請完成下列子步驟:
  a) 請確定您在正確的專案中:

```
oc project <project_name>
```

  b) 檢視專案中安裝的佇列管理程式:

```
oc get qmgr
```

  c) 刪除一或多個佇列管理程式:

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

  請注意，雖然這些佇列管理程式會繼續執行，但如果沒有 IBM MQ Operator，它們可能無法如預期般運作。
  d) 檢視 **ClusterServiceVersion** 實例:

```
oc get csv
```

e) 刪除 IBM MQ **ClusterServiceVersion**:

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

f) 檢視訂閱:

```
oc get subscription
```

g) 刪除所有訂閱:

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

h) 選擇性的: 如果沒有其他項目使用共用服務，則您可能想要解除安裝共用服務操作器，並刪除操作器群組:

a. 遵循 IBM Cloud Pak foundational services 產品說明文件中 解除安裝共用服務 的指示，解除安裝共用服務操作器。

b. 檢視操作員群組:

```
oc get operatorgroup
```

c. 刪除操作員群組:

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

- 如果 IBM MQ Operator 已安裝且可用於叢集上的所有名稱空間，請完成下列子步驟:

a) 檢視所有已安裝的佇列管理程式:

```
oc get qmgr -A
```

b) 刪除一或多個佇列管理程式:

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

請注意，雖然這些佇列管理程式會繼續執行，但如果沒有 IBM MQ Operator，它們可能無法如預期般運作。

c) 檢視 **ClusterServiceVersion** 實例:

```
oc get csv -A
```

d) 從叢集中刪除 IBM MQ **ClusterServiceVersion** :

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

e) 檢視訂閱:

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

f) 刪除訂閱:

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

g) 如果您使用 Red Hat OpenShift Container Platform 4.7，則可能需要手動刪除驗證 Web 連結鉤:

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

h) 選擇性的: 如果沒有其他項目正在使用共用服務，您可能想要解除安裝共用服務操作器:

遵循 IBM Cloud Pak foundational services 產品說明文件中 解除安裝共用服務 的指示。

OpenShift CP4I Linux 在氣隙環境中安裝 IBM MQ Operator

本指導教學引導您將 IBM MQ Operator 安裝至沒有網際網路連線功能的 Red Hat OpenShift 叢集。 您可以使用可攜式儲存裝置或使用防禦機器，在氣隙環境中安裝 IBM MQ Operator 。

使用可攜式儲存裝置在氣隙環境中安裝 IBM MQ Operator

如需完成安裝的步驟，請參閱 IBM Cloud Pak for Integration 說明文件中的 使用可攜式儲存裝置鏡映映像檔 。 如果您只安裝 IBM MQ，請將下列環境變數的所有出現項目取代為這裡提供的值:

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

其中 *version_number* 是您要用來執行氣隙安裝的案例版本。 如需可用案例版本的清單，請參閱 https://github.com/IBM/cloud-pak/tree/master/repo/case/ibm-mq。 檢閱 第 6 頁的『IBM MQ Operator 的版本支援』，以判定要選擇哪個操作員通道。

使用防禦機器在氣隙環境中安裝 IBM MQ Operator

1. 第 61 頁的『必要條件』
2. 第 61 頁的『準備 Docker 登錄』
3. 第 62 頁的『準備防禦主機』
4. 第 63 頁的『建立安裝程式和映像檔庫存的環境變數』
5. 第 63 頁的『下載 IBM MQ 安裝程式及映像檔庫存』
6. 第 63 頁的『以叢集管理者身分登入 Red Hat OpenShift Container Platform 叢集』
7. 第 63 頁的『建立 IBM MQ Operator 的 Kubernetes 名稱空間』
8. 第 63 頁的『鏡映映像檔並配置叢集』
9. 第 65 頁的『安裝「IBM MQ Operator」。』
10. 第 66 頁的『部署 IBM MQ 佇列管理程式』

必要條件

1. 必須安裝 Red Hat OpenShift Container Platform 叢集。 如需支援的 Red Hat OpenShift Container Platform 版本，請參閱 第 6 頁的『IBM MQ Operator 的版本支援』。
2. Docker 登錄必須可用。 如需相關資訊，請參閱 第 61 頁的『準備 Docker 登錄』。
3. 必須配置防禦伺服器。 如需相關資訊，請參閱 第 62 頁的『準備防禦主機』。

準備 Docker 登錄

本端 Docker 登錄用來儲存本端環境中的所有映像檔。 您必須建立此類登錄，且必須確保它符合下列需求:

- 支援 Docker 資訊清單 V2 綱目 2。
- 支援多架構映像檔。
- 可從防禦伺服器及 Red Hat OpenShift Container Platform 叢集節點存取。
- 具有可以從防禦主機寫入目標登錄的使用者名稱及密碼。
- 具有可以從 Red Hat OpenShift 叢集節點上的目標登錄讀取之使用者的使用者名稱及密碼。
- 容許在映像檔名稱中使用路徑分隔字元。

建立 Docker 登錄之後，您必須配置登錄:

1. 建立登錄名稱空間
   - `ibmcom` -用來儲存 `dockerhub.io/ibmcom` 名稱空間中所有映像檔的名稱空間。

     `ibmcom` 名稱空間適用於所有公開可用的 IBM 映像檔，且不需要認證即可取回。

- cp -用來儲存 cp.icr.io/cp 儲存庫中的 IBM 映像檔的名稱空間。

  cp 名稱空間適用於 IBM Entitled Registry 中需要產品授權金鑰和認證才能取回的映像檔。 若要取得授權金鑰，請使用與授權軟體相關聯的 IBM ID 及密碼登入 MyIBM Container Software Library 。 在 **授權金鑰** 區段中，選取 **複製金鑰** 以將授權金鑰複製到剪貼簿，然後儲存它以在下列步驟中使用。
- opencloudio -從 quay.io/opencloudio 儲存映像檔的名稱空間。

  opencloudio 名稱空間用於選取 quay.io 上可用的 IBM 開放程式碼元件映像檔。 IBM Cloud Pak foundational services 映像檔是在 opencloudio 上管理。

2. 請確認每一個名稱空間都符合下列需求：
   - 支援自動建立儲存庫。
   - 具有可以寫入及建立儲存庫之使用者的認證。 防禦主機會使用這些認證。
   - 具有可以讀取所有儲存庫之使用者的認證。 Red Hat OpenShift Container Platform 叢集使用這些認證。

準備防禦主機

準備可以存取 Red Hat OpenShift Container Platform 叢集、本端 Docker 登錄及網際網路的防禦主機。 防禦主機必須位於具有 IBM Cloud Pak CLI 及 Red Hat OpenShift Container Platform CLI 支援之任何作業系統的 Linux for x86-64 平台上。

在防禦主機節點上完成下列步驟:

1. 安裝 OpenSSL 1.11.1 版或更新版本。
2. 在防禦主機節點上安裝 Docker 或 Podman 。
   - 若要安裝 Docker，請執行下列指令:

     ```
     yum check-update
     yum install docker
     ```
   - 若要安裝 Podman，請參閱 Podman 安裝指示
3. 在防禦主機節點上安裝 skopeo 1.x.x 版。 若要安裝 skopeo ，請執行下列指令:

   ```
   yum check-update
   yum install skopeo
   ```
4. 安裝 IBM Cloud Pak CLI。 為您的平台安裝最新版本的二進位檔。 如需相關資訊，請參閱 cloud-pak-cli。
   a. 下載該二進位檔。

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

      例如:

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

   b. 解壓縮該二進位檔。

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

   c. 執行下列指令來修改及移動檔案

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

   d. 確認已安裝 cloudctl:

      ```
      cloudctl --help
      ```

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

   如需相關資訊，請參閱 Red Hat OpenShift Container Platform CLI 工具

6. 建立一個目錄，用來作為離線儲存庫。

   下列是範例目錄。 此範例在後續步驟中使用。

   ```
   mkdir $HOME/offline
   ```

   附註: 此離線儲存庫必須持續保存，以避免多次傳送資料。 持續性也有助於多次執行或依排程執行鏡映程序。

建立安裝程式和映像檔庫存的環境變數

使用安裝程式映像檔名稱及映像檔庫存來建立下列環境變數:

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

其中 *version_number* 是您要用來執行氣隙安裝的案例版本。 如需可用案例版本的清單，請參閱 https://github.com/IBM/cloud-pak/tree/master/repo/case/ibm-mq。 檢閱 IBM MQ Operator 的版本支援，以判定要選擇的操作員通道。

下載 IBM MQ 安裝程式及映像檔庫存

將 ibm-mq 安裝程式及映像檔庫存下載至防禦主機:

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

以叢集管理者身分登入 Red Hat OpenShift Container Platform 叢集

下列是登入 Red Hat OpenShift Container Platform 叢集的範例指令:

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

建立 IBM MQ Operator 的 Kubernetes 名稱空間

建立具有名稱空間的環境變數以安裝 IBM MQ Operator，然後建立名稱空間:

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

鏡映映像檔並配置叢集

完成下列步驟，以鏡映映像檔並配置叢集:

註: 請勿在任何指令中的雙引號內使用波狀符號。 例如，請勿使用 `args "--registry registry --user registry_userid --pass registry_password --inputDir ~/offline"`。 波狀符號不會展開，您的指令可能會失敗。

1. 儲存所有來源 Docker 登錄的鑑別認證。

   所有 IBM Cloud Platform Common Services、 IBM MQ Operator 映像檔和 IBM MQ Advanced Developer 映像檔都儲存在不需要鑑別的公用登錄中。 不過， IBM MQ Advanced Server (非開發人員)、其他產品和協力廠商元件需要一或多個鑑別登錄。 下列登錄需要鑑別:

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

- registry.access.redhat.com

如需這些登錄的相關資訊，請參閱 建立登錄名稱空間。

您必須執行下列指令，為所有需要鑑別的登錄配置認證。 針對每一個此類登錄分別執行指令:

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

此指令會將登錄認證儲存並快取在檔案系統上位於 $HOME/.airgap/secrets 位置的檔案中。

2. 使用本端 Docker 登錄連線資訊來建立環境變數。

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

附註: Docker 登錄使用標準埠，例如 80 或 443。 如果 Docker 登錄使用非標準埠，請使用語法 *host*:*port* 來指定埠。 例如:

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

3. 配置本端 Docker 登錄的鑑別密碼。

附註: 此步驟只需要執行一次。

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

此指令會將登錄認證儲存並快取在檔案系統上位於 $HOME/.airgap/secrets 位置的檔案中。

4. 配置廣域映像檔取回密鑰和 **ImageContentSourcePolicy**。

a. 檢查是否需要重新啟動節點。

- 在 Red Hat OpenShift Container Platform 4.4 版及更新版本中，以及在使用氣隙的 IBM MQ Operator 新安裝上，此步驟會重新啟動所有叢集節點。 可能要到套用新的取回密鑰之後，叢集資源才可供使用。
- 在 IBM MQ Operator 1.8 中， CASE 已更新為包含映像檔的其他鏡映來源。 因此，當您從舊版 IBM MQ Operator 升級至 1.8 版或更高版本時，會觸發節點重新啟動。
- 若要檢查此步驟是否需要重新啟動節點，請將 --dry-run 選項新增至此步驟的程式碼。 這會產生最新的 **ImageContentSourcePolicy** ，並將它顯示在主控台視窗 (**stdout**) 中。 如果此 **ImageContentSourcePolicy** 不同於配置的叢集 **ImageContentSourcePolicy**，則會進行重新啟動。

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

b. 若要配置廣域映像檔取回密碼及 **ImageContentSourcePolicy**，請在沒有 --dry-run 選項的情況下執行此步驟的程式碼:

```
cloudctl case launch \
--case $HOME/offline/${CASE_ARCHIVE} \
--inventory ${CASE_INVENTORY} \
--action configure-cluster-airgap \
```

```
--namespace ${NAMESPACE} \
--args "--registry ${LOCAL_DOCKER_REGISTRY} --user ${LOCAL_DOCKER_USER} --pass $
{LOCAL_DOCKER_PASSWORD} --inputDir $HOME/offline"
```

5. 確認 **ImageContentSourcePolicy** 資源已建立。

```
oc get imageContentSourcePolicy
```

6. 選用項目: 如果您使用不安全的登錄，則必須將本端登錄新增至叢集 **insecureRegistries** 清單。

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

7. 驗證叢集節點狀態。

```
oc get nodes
```

在套用 **imageContentsourcePolicy** 及廣域映像檔取回密鑰之後，您可能會看到節點狀態為 **Ready**、**Scheduling** 或 **Disabled**。 請等待所有節點都顯示 **Ready** 狀態。

8. 將映像檔鏡映至本端登錄。

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

安裝「IBM MQ Operator」。

1. 登入 Red Hat OpenShift 叢集 Web 主控台。
2. 建立型錄來源。 請使用執行先前步驟的相同終端機。

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

3. 驗證已為 Common Services 安裝程式操作員建立 **CatalogSource** 。

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

4. 使用 OLM 來安裝 IBM MQ Operator 。
   a. 從導覽窗格中，按一下 **操作器** > **OperatorHub**。

      即會顯示 **OperatorHub** 頁面。
   b. 在 **所有項目** 欄位中，輸入 IBM MQ。

      即會顯示 IBM MQ 型錄項目。
   c. 選取 **IBM MQ**。

      即會顯示 **IBM MQ** 視窗。
   d. 按一下**安裝**。

      即會顯示「 **建立操作員訂閱** 」頁面。
   e. 檢閱 第 6 頁的『IBM MQ Operator 的版本支援』 ，以判定要選擇哪個操作員通道。
   f. 將 **安裝模式** 設為您建立的特定名稱空間或叢集範圍。
   g. 按一下**訂閱**。

      IBM MQ 會新增至「 **已安裝的操作器** 」頁面。

h. 在「**已安裝的操作器**」頁面上檢查操作器的狀態。 安裝完成時，狀態會變更為 **Succeeded**。

部署 IBM MQ 佇列管理程式

若要在已安裝的操作器下建立新的佇列管理程式，請參閱 第 73 頁的『使用 IBM MQ Operator 來部署及配置佇列管理程式』。

相關工作

第 66 頁的『準備在氣隙環境中升級 IBM MQ Operator 或佇列管理程式』
在沒有網際網路連線功能的 Red Hat OpenShift 叢集中，在升級 IBM MQ Operator 之前需要採取一些準備步驟。

OpenShift CP4I 升級 IBM MQ Operator 及佇列管理程式

升級「IBM MQ Operator」可讓您接著升級佇列管理程式。

程序

- 第 69 頁的『使用 Red Hat OpenShift Web 主控台升級 IBM MQ Operator』.
- 第 70 頁的『使用 Red Hat OpenShift CLI 升級 IBM MQ Operator』.
- 第 71 頁的『使用 Red Hat OpenShift Web 主控台升級 IBM MQ 佇列管理程式』.
- 第 72 頁的『使用 Red Hat OpenShift CLI 升級 IBM MQ 佇列管理程式』.

OpenShift CP4I Linux 準備在氣隙環境中升級 IBM MQ Operator 或佇列管理程式

在沒有網際網路連線功能的 Red Hat OpenShift 叢集中，在升級 IBM MQ Operator 之前需要採取一些準備步驟。

開始之前

本主題假設您已配置本端映像檔登錄，其中鏡映先前發行的 IBM Cloud Pak for Integration 映像檔。

關於這項作業

您必須先鏡映最新的 IBM Cloud Pak for Integration 映像檔，然後才能在氣隙環境中升級 IBM MQ Operator 或佇列管理程式。

請注意，此作業中的前四個步驟與您在 第 61 頁的『在氣隙環境中安裝 IBM MQ Operator』時採取的步驟相同。

程序

1. 建立安裝程式及映像檔庫存的環境變數。

   使用安裝程式映像檔名稱及映像檔庫存來建立下列環境變數:

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

   其中 *version_number* 是您要用來執行氣隙安裝的案例版本。 如需可用案例版本的清單，請參閱 https://github.com/IBM/cloud-pak/tree/master/repo/case/ibm-mq。 檢閱 IBM MQ Operator 的版本支援，以判定要選擇的操作員通道。
2. 下載 IBM MQ 安裝程式及映像檔庫存。

   將 `ibm-mq` 安裝程式及映像檔庫存下載至防禦主機:

   ```
   cloudctl case save \
       --case https://github.com/IBM/cloud-pak/raw/master/repo/case/ibm-mq/
   ```

```
$CASE_ARCHIVE_VERSION/$CASE_ARCHIVE \
    --outputdir $HOME/offline/
```

3. 以叢集管理者身分登入 Red Hat OpenShift Container Platform 叢集。

   下列是登入 Red Hat OpenShift Container Platform 叢集的範例指令:

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

4. 鏡映映像檔並配置叢集。

   完成下列步驟，以鏡映映像檔並配置叢集:

   註: 請勿在任何指令中的雙引號內使用波狀符號。 例如，請勿使用 `args "--registry registry --user registry_userid --pass registry_password --inputDir ~/offline"`。 波狀符號不會展開，您的指令可能會失敗。

   a. 儲存所有來源 Docker 登錄的鑑別認證。

      所有 IBM Cloud Platform Common Services、 IBM MQ Operator 映像檔和 IBM MQ Advanced Developer 映像檔都儲存在不需要鑑別的公用登錄中。 不過， IBM MQ Advanced Server (非開發人員)、其他產品和協力廠商元件需要一或多個鑑別登錄。 下列登錄需要鑑別:

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

      如需這些登錄的相關資訊，請參閱 建立登錄名稱空間。

      您必須執行下列指令，為所有需要鑑別的登錄配置認證。 針對每一個此類登錄分別執行指令:

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

      此指令會將登錄認證儲存並快取在檔案系統上位於 `$HOME/.airgap/secrets` 位置的檔案中。

   b. 使用本端 Docker 登錄連線資訊來建立環境變數。

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

      附註: Docker 登錄使用標準埠，例如 80 或 443。 如果 Docker 登錄使用非標準埠，請使用語法 *host*:*port* 來指定埠。 例如:

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

   c. 配置本端 Docker 登錄的鑑別密碼。

      附註: 此步驟只需要執行一次。

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

      此指令會將登錄認證儲存並快取在檔案系統上位於 `$HOME/.airgap/secrets` 位置的檔案中。

   d. 配置廣域映像檔取回密鑰和 **ImageContentSourcePolicy**。

      i) 檢查是否需要重新啟動節點。

- 在 Red Hat OpenShift Container Platform 4.4 版及更新版本中，以及在使用氣隙的 IBM MQ Operator 新安裝上，此步驟會重新啟動所有叢集節點。 可能要到套用新的取回密鑰之後，叢集資源才可供使用。
- 在 IBM MQ Operator 1.8 中， CASE 已更新為包含映像檔的其他鏡映來源。 因此，當您從舊版 IBM MQ Operator 升級至 1.8 版或更高版本時，會觸發節點重新啟動。
- 若要檢查此步驟是否需要重新啟動節點，請將 `--dry-run` 選項新增至此步驟的程式碼。 這會產生最新的 **ImageContentSourcePolicy**，並將它顯示在主控台視窗 (**stdout**) 中。 如果此 **ImageContentSourcePolicy** 不同於配置的叢集 **ImageContentSourcePolicy**，則會進行重新啟動。

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

ii) 若要配置廣域映像檔取回密碼及 **ImageContentSourcePolicy**，請在沒有 `--dry-run` 選項的情況下執行此步驟的程式碼:

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

e. 確認 **ImageContentSourcePolicy** 資源已建立。

```
oc get imageContentSourcePolicy
```

f. 選用項目: 如果您使用不安全的登錄，則必須將本端登錄新增至叢集 **insecureRegistries** 清單。

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

g. 驗證叢集節點狀態。

```
oc get nodes
```

在套用 **imageContentsourcePolicy** 及廣域映像檔取回密鑰之後，您可能會看到節點狀態為 **Ready**、**Scheduling** 或 **Disabled**。 請等待所有節點都顯示 **Ready** 狀態。

h. 將映像檔鏡映至本端登錄。

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

5. 升級型錄來源。

請使用執行先前步驟的相同終端機。

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

下一步

現在，您已準備好完成下列其中一項作業，來升級 IBM MQ Operator 和佇列管理程式:

- 第 69 頁的『使用 Red Hat OpenShift Web 主控台升級 IBM MQ Operator』
- 第 70 頁的『使用 Red Hat OpenShift CLI 升級 IBM MQ Operator』
- 第 71 頁的『使用 Red Hat OpenShift Web 主控台升級 IBM MQ 佇列管理程式』
- 第 72 頁的『使用 Red Hat OpenShift CLI 升級 IBM MQ 佇列管理程式』
- 第 72 頁的『使用平台 Navigator 在 Red Hat OpenShift 中升級 IBM MQ 佇列管理程式』

OpenShift CP4I 使用 Red Hat OpenShift Web 主控台升級 IBM MQ Operator

可以使用 Operator Hub 來升級 IBM MQ Operator 。

開始之前

登入 Red Hat OpenShift 叢集 Web 主控台。

您必須先鏡映最新的 IBM Cloud Pak for Integration 映像檔，然後才能在氣隙環境中升級 IBM MQ Operator 。 請參閱 在氣隙環境中準備升級 IBM MQ Operator 或佇列管理程式。

程序

1. 檢閱 第 6 頁的『IBM MQ Operator 的版本支援』 ，以判定要升級至哪個操作員通道。
2. 選擇性的: 如果您要從早於 1.5 的 IBM MQ Operator 版本升級至 IBM MQ Operator 1.5 或更新版本，則必須先升級 IBM Cloud Pak foundational services 版本。

   如需相關資訊，請參閱 第 69 頁的『使用 Red Hat OpenShift Web 主控台升級 IBM Cloud Pak foundational services』。
3. 升級 IBM MQ Operator。 新的主要或次要 IBM MQ Operator 版本是透過新的「訂閱通道」來遞送。 若要將您的操作器升級至新的主要或次要版本，您需要在 IBM MQ Operator Subscription 中更新選取的頻道。
   a) 從導覽窗格中，按一下 **操作器 > 已安裝的操作器**。
      會顯示指定專案中所有已安裝的操作員。
   b) 選取 **IBM MQ 操作器**
   c) 導覽至 **訂閱** 標籤
   d) 按一下 **通道**
      即會顯示「 **變更訂閱更新通道** 」視窗。
   e) 選取所需的通道，然後按一下 **儲存**。
      操作員將升級至新通道可用的最新版本。 請參閱 第 6 頁的『IBM MQ Operator 的版本支援』。

下一步

如果您已升級至 IBM Cloud Pak foundational services 3.7，則需要升級或重新啟動任何使用 IBM Cloud Pak for Integration 授權的佇列管理程式。 如需如何執行此動作的相關資訊，請參閱 第 71 頁的『使用 Red Hat OpenShift Web 主控台升級 IBM MQ 佇列管理程式』。

OpenShift CP4I 使用 *Red Hat OpenShift Web* 主控台升級 *IBM Cloud Pak foundational services*

如果您要從早於 1.5 的 IBM MQ Operator 版本升級至 IBM MQ Operator 1.5 或更新版本，則必須先升級 IBM Cloud Pak foundational services 版本。

開始之前

註: 僅當您從早於 1.5 的 IBM MQ Operator 版本升級至 IBM MQ Operator 1.5 或更新版本時，才需要完成此作業。

CP4I 如果您有任何使用 IBM Cloud Pak for Integration 授權的佇列管理程式，則在此升級之後，需要重新啟動佇列管理程式才能存取 Web 主控台，您也會看到登入 Web 主控台的 其他錯誤 。 在操作器升級完成之後，您可以針對您選擇的 IBM MQ 版本升級至 `.spec.version` 的最新值，以修正這些錯誤。

CP4I 如果您有現有的佇列管理程式，且您使用「IBM Cloud Pak for Integration 作業儀表板」，請在升級之前先參閱 第 101 頁的『在 IBM Cloud Pak for Integration 2021.4 中使用「作業儀表板」整合來部署或升級 IBM MQ 9.2.2 或 9.2.3』 。

程序

1. 登入 Red Hat OpenShift 叢集 Web 主控台。
2. 從導覽窗格中，按一下 **操作器** > **已安裝的操作器**。
   會顯示指定專案中所有已安裝的操作員。
3. 選取 **IBM Cloud Pak foundational services 運算子**。 請注意，在 3.7 版之前，這稱為 **IBM Common Services 操作器**
4. 導覽至 **訂閱** 標籤。
5. 按一下 **通道**。
   即會顯示「**變更訂閱更新通道**」視窗。
6. 選取 **v3** 通道，然後按一下 **儲存**。
   IBM Cloud Pak foundational services 操作器會升級至新通道可用的最新版本。 請參閱 第 6 頁的『IBM MQ Operator 的版本支援』 。

下一步

現在您已準備好 升級 IBM MQ Operator。

OpenShift CP4I 使用 Red Hat OpenShift CLI 升級 IBM MQ Operator

可以從指令行升級 IBM MQ Operator 。

開始之前

使用 **`cloudctl login`** (適用於 IBM Cloud Pak for Integration) 或 **`oc login`** 登入叢集。

您必須先鏡映最新的 IBM Cloud Pak for Integration 映像檔，然後才能在氣隙環境中升級 IBM MQ Operator 。 請參閱 在氣隙環境中準備升級 IBM MQ Operator 或佇列管理程式。

程序

1. 檢閱 第 6 頁的『IBM MQ Operator 的版本支援』 ，以判定要升級至哪個操作員通道。
2. 選擇性的: 如果您要從早於 1.5 的 IBM MQ Operator 版本升級至 IBM MQ Operator 1.5 或更新版本，則必須先升級 IBM Cloud Pak foundational services 版本。

   如需相關資訊，請參閱 第 71 頁的『使用 Red Hat OpenShift CLI 升級 IBM Cloud Pak foundational services』 。
3. 升級 IBM MQ Operator。 新的主要/次要 IBM MQ Operator 版本會透過新的「訂閱通道」來遞送。 若要將操作器升級至新的主要/次要版本，您將需要在 IBM MQ Operator Subscription 中更新選取的通道。
   a) 請確定必要的「IBM MQ Operator 升級通道」可用。

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

   b) 修補 `Subscription` 以移至所需的更新通道 (其中 *vX。Y* 是前一個步驟中所識別的所需更新通道。

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

下一步

如果您已升級至 IBM Cloud Pak foundational services 3.7，則需要升級或重新啟動任何使用 IBM Cloud Pak for Integration 授權的佇列管理程式。 如需如何執行此動作的相關資訊，請參閱 第 72 頁的『使用 Red Hat OpenShift CLI 升級 IBM MQ 佇列管理程式』。

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

如果您要從早於 1.5 的 IBM MQ Operator 版本升級至 IBM MQ Operator 1.5 或更新版本，則必須先升級 IBM Cloud Pak foundational services 版本。

開始之前

註: 僅當您從早於 1.5 的 IBM MQ Operator 版本升級至 IBM MQ Operator 1.5 或更新版本時，才需要完成此作業。

CP4I 如果您有任何佇列管理程式使用 IBM Cloud Pak for Integration 授權，則在此升級之後，需要重新啟動佇列管理程式才能存取 Web 主控台，而且您也會看到登入 Web 主控台的 其他錯誤 。 在操作器升級完成之後，您可以針對您選擇的 IBM MQ 版本升級至 `.spec.version` 的最新值，以修正這些錯誤。

CP4I 如果您有現有的佇列管理程式，且您使用「IBM Cloud Pak for Integration 作業儀表板」，請在升級之前先參閱 第 101 頁的『在 IBM Cloud Pak for Integration 2021.4 中使用「作業儀表板」整合來部署或升級 IBM MQ 9.2.2 或 9.2.3』 。

程序

1. 使用 **`cloudctl login`** (適用於 IBM Cloud Pak for Integration) 或 **`oc login`** 登入叢集。
2. 確保 v3 IBM Cloud Pak foundational services 升級通道可用。

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

3. 修補 `Subscription` 以移至所需的更新通道: `v3`

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

下一步

現在您已準備好 升級 IBM MQ Operator。

OpenShift CP4I 使用 Red Hat OpenShift Web 主控台升級 IBM MQ 佇列管理程式

可以使用 Operator Hub 在 Red Hat OpenShift 中升級使用 IBM MQ Operator 部署的 IBM MQ 佇列管理程式。

開始之前

- 登入 Red Hat OpenShift 叢集 Web 主控台。
- 確定您的 IBM MQ Operator 正在使用想要的「更新通道」。 請參閱 第 66 頁的『升級 IBM MQ Operator 及佇列管理程式』。

您必須先鏡映最新的 IBM Cloud Pak for Integration 映像檔，然後才能在氣隙環境中升級佇列管理程式。 請參閱 在氣隙環境中準備升級 IBM MQ Operator 或佇列管理程式。

程序

1. 從導覽窗格中，按一下 **操作器 > 已安裝的操作器**。
   會顯示指定專案中所有已安裝的操作員。
2. 選取 **IBM MQ 操作器**。
   即會顯示「**IBM MQ 操作員**」視窗。
3. 導覽至 **佇列管理程式** 標籤。
   即會顯示「**佇列管理程式詳細資料**」視窗。
4. 選取您要升級的佇列管理程式。
5. 導覽至 YAML 標籤。
6. 必要的話，請更新下列欄位，以符合所需的 IBM MQ 佇列管理程式版本升級。
   - spec.version
   - spec.license.licence

   如需通道與 IBM MQ Operator 版本及 IBM MQ 佇列管理程式版本的對映，請參閱 第 6 頁的『IBM MQ Operator 的版本支援』。
7. 儲存更新的佇列管理程式 YAML。

OpenShift CP4I 使用 Red Hat OpenShift CLI 升級 IBM MQ 佇列管理程式

使用 IBM MQ Operator 部署的 IBM MQ 佇列管理程式可以使用指令行在 Red Hat OpenShift 中升級。

開始之前

您必須是叢集管理者，才能完成這些步驟。

- 使用 `oc login` 登入 Red Hat OpenShift 指令行介面 (CLI)。
- 確定您的 IBM MQ Operator 正在使用想要的「更新通道」。 請參閱 第 66 頁的『升級 IBM MQ Operator 及佇列管理程式』。

您必須先鏡映最新的 IBM Cloud Pak for Integration 映像檔，然後才能在氣隙環境中升級佇列管理程式。 請參閱 在氣隙環境中準備升級 IBM MQ Operator 或佇列管理程式。

程序

必要的話，編輯 **QueueManager** 資源以更新下列欄位，以符合所需的 IBM MQ 佇列管理程式版本升級。

- spec.version
- spec.license.licence

如需通道與 IBM MQ Operator 版本及 IBM MQ 佇列管理程式版本的對映，請參閱 第 6 頁的『IBM MQ Operator 的版本支援』。

使用下列指令：

```
oc edit queuemanager my_qmgr
```

其中 *my_qmgr* 是您要升級的 QueueManager 資源的名稱。

CP4I 使用平台 Navigator 在 Red Hat OpenShift 中升級 IBM MQ 佇列管理程式

使用 IBM MQ Operator 部署的 IBM MQ 佇列管理程式，可以使用 IBM Cloud Pak for Integration Platform Navigator 在 Red Hat OpenShift 中升級。

開始之前

- 登入包含您要升級之佇列管理程式的名稱空間中的 IBM Cloud Pak for Integration Platform Navigator。

- 確定您的 IBM MQ Operator 正在使用想要的「更新通道」。 請參閱 第 66 頁的『升級 IBM MQ Operator 及佇列管理程式』。

您必須先鏡映最新的 IBM Cloud Pak for Integration 映像檔，然後才能在氣隙環境中升級佇列管理程式。 請參閱 在氣隙環境中準備升級 IBM MQ Operator 或佇列管理程式。

程序

1. 從 IBM Cloud Pak for Integration Platform Navigator 首頁中，按一下 **執行時期** 標籤。
2. 具有可用升級項目的佇列管理程式在 版旁邊具有藍色 **i**。 按一下 **i** 以顯示 **可用的新版本**。
3. 按一下您要升級之佇列管理程式最右邊的三個點，然後按一下 **變更版本**。
4. 在 **選取新通道或版本**下，選取所需的升級版本。
5. 按一下 **變更版本**。

結果
佇列管理程式已升級。

OpenShift CP4I 使用 IBM MQ Operator 來部署及配置佇列管理程式

IBM MQ 9.1.5 以及更新版本會使用 IBM MQ Operator 部署至 Red Hat OpenShift 。

關於這項作業

程序

- 第 73 頁的『準備 IBM MQ 的 Red Hat OpenShift 專案』.
- 第 75 頁的『將佇列管理程式部署至 Red Hat OpenShift Container Platform 叢集』.

OpenShift CP4I 準備 IBM MQ 的 Red Hat OpenShift 專案

準備 Red Hat OpenShift Container Platform 叢集，以便它可以部署佇列管理程式。

程序

- 第 73 頁的『使用 Red Hat OpenShift Web 主控台為 IBM MQ 準備 Red Hat OpenShift 專案』.
- 第 74 頁的『使用 Red Hat OpenShift CLI 為 IBM MQ 準備 Red Hat OpenShift 專案』.

相關工作
第 75 頁的『將佇列管理程式部署至 Red Hat OpenShift Container Platform 叢集』
使用 QueueManager 自訂資源，將佇列管理程式部署至 Red Hat OpenShift Container Platform 叢集。

OpenShift CP4I 使用 *Red Hat OpenShift Web* 主控台為 *IBM MQ* 準備 *Red Hat OpenShift* 專案

準備 Red Hat OpenShift Container Platform 叢集，以便它可以使用「IBM MQ Operator」來部署佇列管理程式。 此作業應由專案管理者完成。

開始之前

註: 如果您計劃在已安裝其他 IBM Cloud Pak for Integration 元件的專案中使用 IBM MQ，則可能不需要遵循這些指示。

登入 Red Hat OpenShift 叢集 Web 主控台。

關於這項作業
IBM MQ Operator 映像檔是從執行授權檢查的容器登錄中取回。 此檢查需要儲存在 `docker-registry` 取回密碼中的授權金鑰。 如果您還沒有授權金鑰，請遵循下列指示來取得授權金鑰並建立取回密碼。

程序

1. 取得指派給您 ID 的授權金鑰。
   a) 使用與授權軟體相關聯的 IBM ID 及密碼，登入 MyIBM Container Software Library 。
   b) 在**授權金鑰**區段，選取**複製金鑰**，以將該授權金鑰複製至剪貼簿。
2. 在您要部署佇列管理程式的專案中，建立包含授權金鑰的密鑰。
   a) 從導覽窗格中，按一下 **工作量** > **密鑰**。
   即會顯示「密鑰」頁面。
   b) 在 **專案** 下拉清單中，選取您要安裝 IBM MQ 的專案
   c) 按一下 **建立** 按鈕，然後選取 **映像檔取回密碼**
   d) 在 **名稱** 欄位中，輸入 `ibm-entitlement-key`
   e) 在 **登錄伺服器位址** 欄位中，輸入 `cp.icr.io`
   f) 在 **使用者名稱** 欄位中，輸入 `cp`
   g) 在 **密碼** 欄位中，輸入您在前一個步驟中複製的授權金鑰。
   h) 在 **電子郵件** 欄位中，輸入與授權軟體相關聯的 IBM ID

下一步

第 76 頁的『使用 Red Hat OpenShift Web 主控台部署佇列管理程式』

OpenShift CP4I 使用 *Red Hat OpenShift CLI* 為 *IBM MQ* 準備 *Red Hat OpenShift* 專案

準備 Red Hat OpenShift Container Platform 叢集，以便它可以使用「IBM MQ Operator」來部署佇列管理程式。 此作業應由專案管理者完成。

開始之前

註: 如果您計劃在已安裝其他 IBM Cloud Pak for Integration 元件的專案中使用 IBM MQ，則可能不需要遵循這些指示。

使用 **`cloudctl login`** (適用於 IBM Cloud Pak for Integration) 或 **`oc login`** 登入叢集。

關於這項作業

IBM MQ Operator 映像檔是從執行授權檢查的容器登錄中取回。 此檢查需要儲存在 `docker-registry` 取回密碼中的授權金鑰。 如果您還沒有授權金鑰，請遵循下列指示來取得授權金鑰並建立取回密碼。

程序

1. 取得指派給您 ID 的授權金鑰。
   a) 使用與授權軟體相關聯的 IBM ID 及密碼，登入 MyIBM Container Software Library 。
   b) 在**授權金鑰**區段，選取**複製金鑰**，以將該授權金鑰複製至剪貼簿。
2. 在您要部署佇列管理程式的專案中，建立包含授權金鑰的密鑰。
   執行下列指令，其中 *<entitlement-key>* 是在步驟 1 中擷取的金鑰， *<user-email>* 是與授權軟體相關聯的 IBM ID。

```
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

第 77 頁的『使用 Red Hat OpenShift CLI 部署佇列管理程式』

OpenShift CP4I 將佇列管理程式部署至 Red Hat OpenShift Container Platform 叢集

使用 QueueManager 自訂資源，將佇列管理程式部署至 Red Hat OpenShift Container Platform 叢集。

程序

- CP4I
  第 75 頁的『使用 IBM Cloud Pak for Integration Platform Navigator 部署佇列管理程式』.
- OpenShift
  第 76 頁的『使用 Red Hat OpenShift Web 主控台部署佇列管理程式』.
- OpenShift
  第 77 頁的『使用 Red Hat OpenShift CLI 部署佇列管理程式』.

相關工作

第 78 頁的『配置佇列管理程式的範例』
可以透過調整 QueueManager 自訂資源的內容來配置佇列管理程式。

CP4I 使用 *IBM Cloud Pak for Integration Platform Navigator* 部署佇列管理程式

使用 QueueManager 自訂資源，可使用 IBM Cloud Pak for Integration Platform Navigator 將佇列管理程式部署至 Red Hat OpenShift Container Platform 叢集。 此作業應由專案管理者完成

開始之前

在瀏覽器中，啟動 IBM Cloud Pak for Integration Platform Navigator。

如果這是第一次將佇列管理程式部署至這個 Red Hat OpenShift 專案，請遵循 第 73 頁的『準備 IBM MQ 的 Red Hat OpenShift 專案』的步驟。

程序

1. 部署佇列管理程式。

   下列範例會部署「快速入門」佇列管理程式，其使用暫時 (非持續性) 儲存體，並關閉 MQ 安全。 在重新啟動佇列管理程式之後，將不會持續保存訊息。 您可以調整配置來變更許多佇列管理程式設定。

   a) 在 IBM Cloud Pak for Integration Platform Navigator 中，按一下 **管理**，然後按一下 **整合執行時期**。 在舊版 IBM Cloud Pak for Integration Platform Navigator 中，按一下 **執行時期及實例**。

   b) 按一下**建立實例**。

   c) 選取 **傳訊**，然後按 **下一步**。 在舊版 IBM Cloud Pak for Integration Platform Navigator 中，按一下 **佇列管理程式**，然後按 **下一步**。

   即會顯示用來建立 `QueueManager` 實例的表單。

   註: 您也可以按一下 **程式碼**，以檢視或變更 `QueueManager` 配置 YAML。

   d) 在 **詳細資料** 區段中，檢查或更新 **名稱** 欄位，並指定要在其中建立佇列管理程式實例的 **名稱空間**。

   e) 如果您接受 IBM Cloud Pak for Integration 授權合約，請將 **授權接受** 變更為 **開啟**。

   您必須接受授權，才能部署佇列管理程式。

   f) 在 **佇列管理程式** 區段中，檢查或更新基礎佇列管理程式的 **名稱**。 在舊版 IBM Cloud Pak for Integration Platform Navigator 中，請使用 **佇列管理程式配置** 區段。

   依預設， IBM MQ 用戶端應用程式使用的佇列管理程式名稱將與 `QueueManager` 的名稱相同，但會移除任何無效字元 (例如連字號)。

   g) 按一下**建立**

   現在會顯示現行專案 (名稱空間) 中的佇列管理程式清單。 新的 `QueueManager` 應該具有 `Pending` 狀態

2. 請檢查佇列管理程式是否在執行中

當 QueueManager 狀態為 Running 時，建立完成。

相關工作

第 98 頁的『配置路徑以從 Red Hat OpenShift 叢集外部連接至佇列管理程式』
您需要 Red Hat OpenShift 路徑，才能從 Red Hat OpenShift 叢集外部將應用程式連接至 IBM MQ 佇列管理程式。 您必須在 IBM MQ 佇列管理程式及用戶端應用程式上啟用 TLS，因為只有在使用 TLS 1.2 或更高版本的通訊協定時，才能在 TLS 通訊協定中使用 SNI。 Red Hat OpenShift Container Platform Router 使用 SNI 將要求遞送至 IBM MQ 佇列管理程式。

第 104 頁的『連接至 Red Hat OpenShift 叢集中部署的 IBM MQ Console』
如何連接至已部署至 Red Hat OpenShift Container Platform 叢集之佇列管理程式的 IBM MQ Console 。

OpenShift CP4I 使用 *Red Hat OpenShift Web* 主控台部署佇列管理程式

使用 QueueManager 自訂資源，利用 Red Hat OpenShift Web 主控台，將佇列管理程式部署至 Red Hat OpenShift Container Platform 叢集。 此作業應由專案管理者完成

開始之前

登入 Red Hat OpenShift 叢集 Web 主控台。 您需要選取要使用的現有專案 (名稱空間)，或建立新的專案 (名稱空間)。

如果這是第一次將佇列管理程式部署至這個 Red Hat OpenShift 專案，請遵循 第 73 頁的『準備 IBM MQ 的 Red Hat OpenShift 專案』的步驟。

程序

1. 部署佇列管理程式。

   下列範例會部署「快速入門」佇列管理程式，其使用暫時 (非持續性) 儲存體，並關閉 MQ 安全。 在重新啟動佇列管理程式之後，將不會持續保存訊息。 您可以調整配置來變更許多佇列管理程式設定。

   a) 在 Red Hat OpenShift Web 主控台中，從導覽窗格中按一下 **操作器 > 已安裝的操作器**

   b) 按一下 **IBM MQ**。

   c) 按一下 **佇列管理程式** 標籤。

   d) 按一下 **建立 QueueManager** 按鈕。

   即會顯示 YAML 編輯器，其中包含 QueueManager 資源的範例 YAML。

   註: 您也可以按一下 **編輯表單**，以檢視或變更 QueueManager 配置。

   e) 如果您接受授權合約，請將 **授權接受** 變更為 **開啟**。

   IBM MQ 在數個不同的授權下可用。 如需有效授權的相關資訊，請參閱 第 115 頁的『mq.ibm.com/v1beta1 的授權參考手冊』。 您必須接受授權，才能部署佇列管理程式。

   f) 按一下**建立**

   現在會顯示現行專案 (名稱空間) 中的佇列管理程式清單。 新的 QueueManager 應該處於 Pending 狀態。

2. 請檢查佇列管理程式是否在執行中

   當 QueueManager 狀態為 Running 時，建立完成。

相關工作

第 98 頁的『配置路徑以從 Red Hat OpenShift 叢集外部連接至佇列管理程式』
您需要 Red Hat OpenShift 路徑，才能從 Red Hat OpenShift 叢集外部將應用程式連接至 IBM MQ 佇列管理程式。 您必須在 IBM MQ 佇列管理程式及用戶端應用程式上啟用 TLS，因為只有在使用 TLS 1.2 或更高版本的通訊協定時，才能在 TLS 通訊協定中使用 SNI。 Red Hat OpenShift Container Platform Router 使用 SNI 將要求遞送至 IBM MQ 佇列管理程式。

第 104 頁的『連接至 Red Hat OpenShift 叢集中部署的 IBM MQ Console』
如何連接至已部署至 Red Hat OpenShift Container Platform 叢集之佇列管理程式的 IBM MQ Console 。

OpenShift CP4I 使用 *Red Hat OpenShift CLI* 部署佇列管理程式

使用 QueueManager 自訂資源，可使用指令行介面 (CLI) 將佇列管理程式部署至 Red Hat OpenShift Container Platform 叢集。 此作業應由專案管理者完成

開始之前

您需要安裝 Red Hat OpenShift Container Platform 指令行介面。

使用 **cloudctl login** (適用於 IBM Cloud Pak for Integration) 或 **oc login** 登入叢集。

如果這是第一次將佇列管理程式部署至這個 Red Hat OpenShift 專案，請遵循 第 73 頁的『準備 IBM MQ 的 Red Hat OpenShift 專案』的步驟。

程序

1. 部署佇列管理程式。

   下列範例會部署「快速入門」佇列管理程式，其使用暫時 (非持續性) 儲存體，並關閉 MQ 安全。 在重新啟動佇列管理程式之後，將不會持續保存訊息。 您可以調整 YAML 的內容，以變更許多佇列管理程式設定。

   a) 建立 QueueManager YAML 檔案

   例如，若要在 IBM Cloud Pak for Integration 中安裝基本佇列管理程式，請建立具有下列內容的檔案 "mq-quickstart.yaml":

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

   重要: 如果您接受 IBM Cloud Pak for Integration 授權合約，請將 `accept: false` 變更為 `accept: true`。 如需授權的詳細資料，請參閱 第 115 頁的『mq.ibm.com/v1beta1 的授權參考手冊』。

   此範例還包括隨佇列管理程式一起部署的 Web 伺服器，其中已啟用 Web 主控台並使用 IBM Cloud Pak Identity and Access Manager 啟用「單一登入」。

   若要獨立於 IBM Cloud Pak for Integration 安裝基本佇列管理程式，請建立具有下列內容的檔案 "mq-quickstart.yaml":

   ```
   apiVersion: mq.ibm.com/v1beta1
   kind: QueueManager
   metadata:
     name: quickstart
   spec:
     version: 9.2.5.0-r3
     license:
       accept: false
       license: L-APIG-BZDDDY
     web:
       enabled: true
     queueManager:
   ```

```
    name: "QUICKSTART"
    storage:
      queueManager:
        type: ephemeral
  template:
    pod:
      containers:
       - name: qmgr
         env:
         - name: MQSNOAUT
           value: "yes"
```

重要事項:如果您接受 MQ 授權合約，請將 `accept: false` 變更為 `accept: true`。 如需授權的詳細資料，請參閱 第 115 頁的『mq.ibm.com/v1beta1 的授權參考手冊』 。

b) 建立 `QueueManager` 物件

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

2. 請檢查佇列管理程式是否在執行中

   您可以執行下列指令來驗證部署:

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

   ，然後檢查狀態。

   例如，執行

   ```
   oc describe queuemanager quickstart
   ```

   ，並檢查 `status.Phase` 欄位是否指出 `Running`

相關工作

第 98 頁的『配置路徑以從 Red Hat OpenShift 叢集外部連接至佇列管理程式』
您需要 Red Hat OpenShift 路徑，才能從 Red Hat OpenShift 叢集外部將應用程式連接至 IBM MQ 佇列管理程式。 您必須在 IBM MQ 佇列管理程式及用戶端應用程式上啟用 TLS，因為只有在使用 TLS 1.2 或更高版本的通訊協定時，才能在 TLS 通訊協定中使用 SNI。 Red Hat OpenShift Container Platform Router 使用 SNI 將要求遞送至 IBM MQ 佇列管理程式。

第 104 頁的『連接至 Red Hat OpenShift 叢集中部署的 IBM MQ Console』
如何連接至已部署至 Red Hat OpenShift Container Platform 叢集之佇列管理程式的 IBM MQ Console 。

OpenShift CP4I 配置佇列管理程式的範例

可以透過調整 QueueManager 自訂資源的內容來配置佇列管理程式。

關於這項作業

請使用下列範例，以協助您使用 QueueManager YAML 檔案來配置佇列管理程式。

程序

- 第 78 頁的『範例: 提供 MQSC 及 INI 檔案』
- 第 80 頁的『範例: 配置 TLS』

OpenShift CP4I 範例: 提供 *MQSC* 及 *INI* 檔案

此範例會建立 Kubernetes ConfigMap，其中包含兩個 MQSC 檔及一個 INI 檔。 然後會部署佇列管理程式來處理這些 MQSC 及 INI 檔案。

關於這項作業

部署佇列管理程式時，可以提供 MQSC 及 INI 檔。 MQSC 及 INI 資料必須定義在一或多個 Kubernetes ConfigMaps 及 Secrets 中。 這些必須建立在您將在其中部署佇列管理程式的名稱空間 (專案) 中。

註: 當 MQSC 或 INI 檔案包含機密資料時，應該使用 Kubernetes 密鑰。

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

範例

下列範例會建立 Kubernetes ConfigMap，其中包含兩個 MQSC 檔及一個 INI 檔。 然後會部署佇列管理程式來處理這些 MQSC 及 INI 檔案。

範例 ConfigMap -在叢集裡套用下列 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
```

範例 QueueManager -使用指令行或 IBM Cloud Pak for Integration Platform Navigator，以下列配置部署佇列管理程式:

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

重要: 如果您接受 IBM Cloud Pak for Integration 授權合約，請將 `accept: false` 變更為 `accept: true`。 如需授權的詳細資料，請參閱 mq.ibm.com/v1beta1 的授權參考資料 。

其他資訊:

- 佇列管理程式可以配置為使用單一 Kubernetes ConfigMap 或「密鑰」(如本範例所示) 或多個 Kubernetes ConfigMaps 及「密鑰」。
- 您可以選擇使用 Kubernetes ConfigMap 或 Secret 中的所有 MQSC 及 INI 資料 (如本範例所示)，或將每一個佇列管理程式配置為只使用可用檔案的子集。
- MQSC 及 INI 檔案會根據其索引鍵按字母順序進行處理。 因此，不論 `example1.mqsc` 在佇列管理程式配置中出現的順序為何，一律會在 `example2.mqsc` 之前處理。
- 如果多個 MQSC 或 INI 檔案在多個 Kubernetes ConfigMaps 或「密鑰」之間具有相同的索引鍵，則會根據這些檔案在佇列管理程式配置中的定義順序來處理這組檔案。

OpenShift CP4I Linux 範例: 配置 *TLS*

此範例使用 IBM MQ Operator 將佇列管理程式部署至 Red Hat OpenShift Container Platform 。 在範例用戶端與佇列管理程式之間配置單向 TLS 通訊。 此範例透過放置及取得訊息來示範成功的配置。

開始之前

若要完成此範例，您必須先完成下列必要條件:

- 安裝 IBM MQ client，並將 `samp/bin` 及 `bin` 新增至 *PATH*。 您需要 **runmqakm**、 **amqsputc** 和 **amqsgetc** 應用程式，可以安裝成 IBM MQ client 的一部分，如下所示:
  - Windows Linux Windows 和 Linux: 從 https://ibm.biz/mq92redistclients 安裝適用於您作業系統的 IBM MQ 可重新配送用戶端。
  - macOS Mac: 下載並設定 IBM MQ MacOS Toolkit: https://developer.ibm.com/tutorials/mq-macos-dev/
- 安裝適用於您作業系統的 OpenSSL 工具。
- 針對此範例建立 Red Hat OpenShift Container Platform (OCP) 專案/名稱空間。
- 在指令行上，登入 OCP 叢集，並切換至上述名稱空間。
- 請確定上述名稱空間中已安裝 IBM MQ Operator 且可供使用。

關於這項作業

此範例提供自訂資源 YAML ，定義要部署至 Red Hat OpenShift Container Platform 的佇列管理程式。 它也會詳細說明在啟用 TLS 的情況下部署佇列管理程式所需的其他步驟。 完成時，放置及取得訊息會驗證佇列管理程式是否已配置 TLS。

為 IBM MQ 伺服器建立 TLS 私密金鑰及憑證

下列程式碼範例顯示如何建立佇列管理程式的自簽憑證，以及如何將憑證新增至金鑰資料庫，以充當用戶端的信任儲存庫。 如果您已有私密金鑰和憑證，則可以改用那些憑證。

請注意，自簽憑證只能用於開發目的。

在現行目錄中建立自簽私密金鑰及公用憑證

請執行下列指令:

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

將伺服器公開金鑰新增至用戶端金鑰資料庫

金鑰資料庫用作用戶端應用程式的信任儲存庫。

建立用戶端金鑰資料庫:

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

將先前產生的公開金鑰新增至用戶端金鑰資料庫:

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

配置用於佇列管理程式部署的 TLS 憑證

為了讓佇列管理程式可以參照並套用金鑰及憑證，請建立 Kubernetes TLS 密鑰，並參照上面建立的檔案。 當您這麼做時，請確定您在開始此作業之前所建立的名稱空間中。

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

建立包含 MQSC 指令的配置對映

建立包含 MQSC 指令的 Kubernetes 配置對映，以建立新佇列及「SVRCONN 通道」，並新增通道鑑別記錄，只封鎖那些稱為 *nobody* 的使用者來容許存取通道。

請注意，此方法應該僅用於開發目的。

請確定您位於先前建立的名稱空間中 (請參閱 開始之前) ，然後在 OCP 使用者介面中輸入下列 YAML ，或使用指令行。

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

建立必要的 OCP 路徑

請確定您在開始此作業之前所建立的名稱空間中，然後在 OCP 使用者介面中輸入下列 YAML ，或使用指令行。

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

請注意， Red Hat OpenShift Container Platform Router 會使用 SNI 將要求遞送至 IBM MQ 佇列管理程式。 如果您在先前建立的配置對映中變更 MQSC 中指定的通道名稱，則也需要在這裡及稍後建立的 CCDT 檔案中變更主機欄位。 如需相關資訊，請參閱第 98 頁的『配置路徑以從 Red Hat OpenShift 叢集外部連接至佇列管理程式』。

部署佇列管理程式

重要事項: 在此範例中，我們使用 *MQSNAUT* 變數來停用佇列管理程式上的授權，這可讓我們聚焦於使用 TLS 連接用戶端所需的步驟。 在 IBM MQ 的正式作業部署中不建議這樣做，因為它會導致任何連接的應用程式都具有完整管理權力，且沒有降低個別應用程式許可權的機制。

使用下列自訂資源 YAML 建立新的佇列管理程式。 請注意，它會參照先前建立的配置對映和密鑰，以及 *MQSNAUT* 變數。

請確定您在開始此作業之前建立的名稱空間中，然後在 OCP 使用者介面中使用指令行或使用 IBM Cloud Pak for Integration Platform Navigator 輸入下列 YAML。 請檢查是否指定正確的授權，並將 `false` 變更為 `true` 以接受授權。

```
apiVersion: mq.ibm.com/v1beta1
kind: QueueManager
metadata:
  name: secureqm
spec:
  license:
    accept: false
    license: L-RJON-C7QG3S
    use: Production
  queueManager:
    name: SECUREQM
    mqsc:
    - configMap:
        name: example-tls-configmap
        items:
        - tls.mqsc
    storage:
      queueManager:
        type: ephemeral
  template:
```

```
    pod:
      containers:
        - env:
            - name: MQSNOAUT
              value: 'yes'
          name: qmgr
  version: 9.2.5.0-r3
  web:
    enabled: true
  pki:
    keys:
      - name: example
        secret:
          secretName: example-tls-secret
          items:
          - tls.key
          - tls.crt
```

確認佇列管理程式正在執行中

現在正在部署佇列管理程式。 請先確認它處於 Running 狀態，然後再繼續。 例如：

```
oc get qmgr secureqm
```

測試佇列管理程式的連線

若要確認佇列管理程式已配置單向 TLS 通訊，請使用 **amqsputc** 及 **amqsgetc** 範例應用程式：

尋找佇列管理程式主機名稱

使用下列指令來尋找路徑 secureqm-ibm-mq-qm 的佇列管理程式完整主機名稱：

```
oc get routes secureqm-ibm-mq-qm
```

指定佇列管理程式詳細資料

建立 CCDT.JSON 檔案，以指定佇列管理程式詳細資料。 將主機值取代為前一個步驟中的主機名稱。

```
{
    "channel":
    [
        {
            "name": "SECUREQMCHL",
            "clientConnection":
            {
                "connection":
                [
                    {
                        "host": "<hostname from previous step>",
                        "port": 443
                    }
                ],
                "queueManager": "SECUREQM"
            },
            "transmissionSecurity":
            {
              "cipherSpecification": "ECDHE_RSA_AES_128_CBC_SHA256"
            },
            "type": "clientConnection"
        }
   ]
}
```

匯出環境變數

以適合您作業系統的方式匯出下列環境變數。 **amqsputc** 和 **amqsgetc** 將會讀取這些變數。

更新系統上檔案的路徑：

```
export MQCCDTURL='<full path to file>/CCDT.JSON'
export MQSSLKEYR='<full path to file>/clientkey'
```

將訊息放入佇列

請執行下列指令：

```
amqsputc EXAMPLE.QUEUE SECUREQM
```

如果佇列管理程式連線成功，則會輸出下列回應:

```
target queue is EXAMPLE.QUEUE
```

透過輸入一些文字，然後每次按 **Enter** 鍵，將數個訊息放入佇列。

若要完成，請按 **Enter** 鍵兩次。

從佇列擷取訊息

請執行下列指令:

```
amqsgetc EXAMPLE.QUEUE SECUREQM
```

您在前一個步驟中新增的訊息已耗用且為輸出。

在幾秒鐘之後，指令結束。

恭喜，您已順利部署已啟用 TLS 的佇列管理程式，並顯示您可以從用戶端安全地放置訊息並取得訊息至佇列管理程式。

OpenShift CP4I 範例: 自訂授權服務註釋

IBM MQ Operator 會自動將 IBM License Service 註釋新增至已部署的資源。 這些由 IBM License Service 監視，並產生對應於所需授權的報告。

關於這項作業

IBM MQ Operator 所新增的註釋是標準狀況中所預期的註釋，並以部署佇列管理程式期間所選取的授權值為基礎。

範例

如果 **License** 設為 `L-RJON-BZFQU2 (IBM Cloud Pak for Integration 2021.2.1)`，且 **Use** 設為 `NonProduction`，則會套用下列註釋:

- cloudpakId: c8b82d189e7545f0892db9ef2731b90d
- cloudpakName: IBM Cloud Pak for Integration
- productCharged 儲存器 :qmgr
- productCloudpak 比例: '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 內， IBM App Connect Enterprise 的部署包括 IBM MQ 的受限授權。 在這些狀況下，需要置換這些註釋，以確保 IBM License Service 擷取正確的用法。 若要執行此動作，請使用 第 103 頁的『將自訂註釋和標籤新增至佇列管理程式資源』中說明的方法。

例如，如果在 IBM App Connect Enterprise 授權下部署 IBM MQ ，請使用下列程式碼片段中顯示的方法:

```
apiVersion: mq.ibm.com/v1beta1
kind: QueueManager
metadata:
  name: mq4ace
  namespace: cp4i
spec:
  annotations:
    productMetric: FREE
```

授權註釋可能需要修改的其他兩個常見原因:

1. IBM MQ Advanced 包含在另一個 IBM 產品的授權中。
   - 在此狀況下，請使用先前針對 IBM App Connect Enterprise 所說明的方法。
2. IBM MQ 是在 IBM Cloud Pak for Integration 授權下部署。

- 如果您具有 IBM Cloud Pak for Integration 授權，則可以決定以 IBM MQ 或 IBM MQ Advanced 比例來部署佇列管理程式。 如果您在 IBM MQ 比例下部署，則必須確保您未使用任何進階功能，例如原生 HA 或 Advanced Message Security。
- 在此狀況下，請使用下列註釋供正式作業使用:

```
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
```

- 將下列註釋用於非正式作業用途:

```
apiVersion: mq.ibm.com/v1beta1
kind: QueueManager
metadata:
  name: mq4ace
  namespace: cp4i
spec:
  annotations:
    productID: 151bec68564a4a47a14e6fa99266deff
    productCloudpakRatio: '8:1'
    productName: IBM MQ for Non-Production
    productMetric: VIRTUAL_PROCESSOR_CORE
```

OpenShift CP4I 使用 IBM MQ Operator 來配置佇列管理程式的高可用性

關於這項作業

程序

- V 9.2.3
  第 84 頁的『原生 HA』.
- V 9.2.3
  第 86 頁的『範例: 配置原生 HA 佇列管理程式』.
- 第 94 頁的『範例: 配置多重實例佇列管理程式』.

CP4I CD V 9.2.3 原生 *HA*

原生 HA 是適用於 IBM MQ 的原生 (內建) 高可用性解決方案，適合與雲端區塊儲存體搭配使用。

「原生 HA」配置提供高可用性佇列管理程式，其中可回復的 MQ 資料 (例如，訊息) 會在多組儲存體之間抄寫，以防止因儲存體故障而流失。 佇列管理程式由多個執行中的實例組成，其中一個實例是主導者，其他實例則準備好在失敗時快速接管，以最大化對佇列管理程式及其訊息的存取權。

原生 HA 配置由三個 Kubernetes Pod 組成，每一個 Pod 都有一個佇列管理程式實例。 一個實例是作用中佇列管理程式，處理訊息並寫入其回復日誌。 每當寫入回復日誌時，作用中佇列管理程式就會將資料傳送至另外兩個實例，稱為抄本。 每一個抄本都會寫入自己的回復日誌，確認資料，然後從抄寫的回復日誌更新自己的佇列資料。 如果執行作用中佇列管理程式的 Pod 失敗，則佇列管理程式的其中一個抄本實例會接管作用中角色，並具有可操作的現行資料。

日誌類型稱為「抄寫日誌」。 抄寫的日誌本質上是線性日誌，已啟用自動日誌管理及自動媒體映像檔。 請參閱 記載類型。 您可以使用相同的技術來管理用於管理線性日誌的抄寫日誌。

Kubernetes Service 用來將 TCP/IP 用戶端連線遞送至現行作用中實例，該實例被識別為備妥可供網路資料流量使用的唯一 Pod。 這會發生，而不需要用戶端應用程式知道不同的實例。

3 個 Pod 被用來大大減少出現腦分裂情況的可能性。 在雙 Pod 高可用性中，當兩個 Pod 之間的連線功能中斷時，可能會發生核心分裂。 沒有連線功能，兩個 Pod 可以同時執行佇列管理程式，累計不同的資料。 當連線回復時，會有兩個不同版本的資料 ('spit-brain ')，且需要人為介入來決定要保留哪些資料集，以及要捨棄哪些資料集。

原生 HA 使用具有額定的三個 Pod 系統，以避免核心分裂狀況。 至少可以與其中一個其他 Pod 通訊的 Pod 會形成仲裁。 佇列管理程式只能在具有仲裁的 Pod 上變成作用中實例。 佇列管理程式無法在未連接至至少一個其他 Pod 的 Pod 上變成作用中，因此永遠不會同時有兩個作用中實例:

- 如果單一 Pod 失敗，則其他兩個 Pod 中的其中一個 Pod 上的佇列管理程式可以接管。 如果兩個 Pod 失敗，則佇列管理程式無法變成其餘 Pod 上的作用中實例，因為該 Pod 沒有仲裁 (其餘 Pod 無法指出其他兩個 Pod 是否失敗，或它們仍在執行中且失去連線功能)。
- 如果單一 Pod 失去連線功能，則佇列管理程式無法在此 Pod 上變成作用中，因為 Pod 沒有仲裁。 其餘兩個 Pod 中的其中一個 Pod 上的佇列管理程式可以接管具有額定的佇列管理程式。 如果所有 Pod 都失去連線功能，則佇列管理程式無法在任何 Pod 上變成作用中，因為所有 Pod 都沒有仲裁。

如果作用中 Pod 失敗並隨後回復，則可以在抄本角色中重新結合群組。

下圖顯示一般部署，其中有三個佇列管理程式實例部署在三個儲存器中。

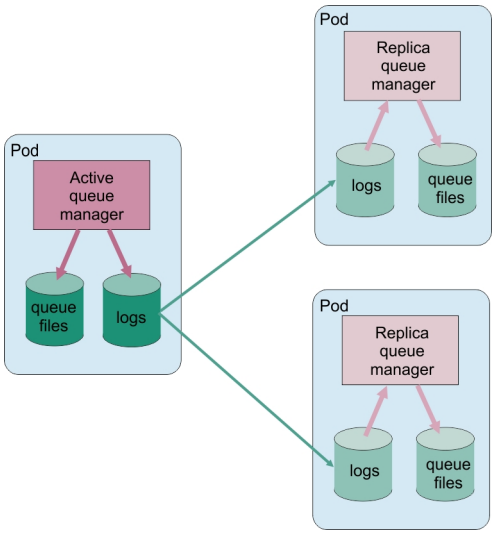


圖 *1:* 原生 *HA* 配置範例

CP4I CD V 9.2.3 使用 *IBM MQ Operator* 來配置原生 *HA*

原生 HA 是使用 `QueueManager` API 來配置，而進階選項是使用 INI 檔案來提供。

原生 HA 是使用 `QueueManager` API 的 `.spec.queueManager.availability` 來配置，例如:

```
apiVersion: mq.ibm.com/v1beta1
kind: QueueManager
metadata:
name: nativeha-example
spec:
  license:
    accept: false
    license: L-RJON-C7QG3S
    use: Production
  queueManager:
    availability:
      type: NativeHA
  version: 9.2.5.0-r3
```

`.spec.queueManager.availability.type` 欄位必須設為 `NativeHA`。

原生 HA 在 IBM MQ 9.2.3 或更高版本中可用。

在 `.spec.queueManager.availability` 下，您也可以配置 TLS 密鑰及密碼，以在抄寫時在佇列管理程式實例之間使用。 強烈建議這樣做，並在 第 86 頁的『範例: 配置原生 HA 佇列管理程式』中提供逐步手冊。

相關參考

第 86 頁的『範例: 配置原生 HA 佇列管理程式』
此範例顯示如何使用 IBM MQ Operator，將使用原生高可用性特性的佇列管理程式部署至 Red Hat OpenShift Container Platform (OCP)。

範例: 配置原生 *HA* 佇列管理程式

此範例顯示如何使用 IBM MQ Operator，將使用原生高可用性特性的佇列管理程式部署至 Red Hat OpenShift Container Platform (OCP)。

開始之前

若要完成此範例，您必須先完成下列必要條件:

- 安裝 IBM MQ client，並將已安裝的 `samp/bin` 及 `bin` 目錄新增至 *PATH*。 用戶端提供此範例所需的 **runmqakm**、 **amqsputc** 及 **amqsgetc** 應用程式。 安裝 IBM MQ client ，如下所示:
  - 

Windows 和 Linux: 從 https://ibm.biz/mq92redistclients 安裝適用於您作業系統的 IBM MQ 可重新配送用戶端。
  - 

Mac: 下載並設定 IBM MQ MacOS Toolkit。 請參閱 https://ibm.biz/mqdevmacclient。
- 安裝適用於您作業系統的 OpenSSL 工具。 如果您還沒有私密金鑰和憑證，則需要這樣才能為佇列管理程式產生自簽憑證。
- 為此範例建立 Red Hat OpenShift Container Platform (OCP) 專案/名稱空間，並遵循作業 第 73 頁的『準備 IBM MQ 的 Red Hat OpenShift 專案』中的步驟
- 在指令行上，登入 OCP 叢集，並切換至您剛建立的名稱空間。
- 請確定 IBM MQ Operator 已安裝且可在名稱空間中使用。
- 在 OCP 中配置要由佇列管理程式使用的預設儲存類別。 如果您想要在不設定預設儲存空間類別的情況下完成本指導教學，請參閱 附註 2: 使用非預設儲存空間類別。

關於此作業

原生 HA 佇列管理程式涉及一個作用中及兩個抄本 Kubernetes Pod。 這些會作為 Kubernetes 有狀態集的一部分執行，其中正好有三個抄本及一組 Kubernetes 持續性磁區。 如需原生 HA 佇列管理程式的相關資訊，請參閱 第 15 頁的『儲存器中 IBM MQ 的高可用性』。

此範例提供自訂資源 YAML，其定義使用持續性儲存體且已配置 TLS 的原生 HA 佇列管理程式。 將佇列管理程式部署至 OCP 之後，您會模擬作用中佇列管理程式 Pod 的失敗。 您會看到自動回復發生，並在失敗之後放置並取得訊息，以證明它已成功。

範例

為 MQ 伺服器建立 TLS 私密金鑰和憑證

您可以為佇列管理程式建立自簽憑證，並將憑證新增至金鑰資料庫，以充當用戶端的信任儲存庫。 如果您已有私密金鑰和憑證，則可以改用那些憑證。 請注意，您只應該將自簽憑證用於開發目的。

若要在現行目錄中建立自簽私密金鑰及公用憑證，請執行下列指令:

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

建立 TLS 私密金鑰及憑證，以供原生 HA 內部使用

原生 HA 佇列管理程式中的三個 Pod 會透過網路抄寫資料。 您可以建立自簽憑證，以在內部抄寫時使用。 請注意，您只應該將自簽憑證用於開發目的。

若要在現行目錄中建立自簽私密金鑰及公用憑證，請執行下列指令:

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

將佇列管理程式公開金鑰新增至用戶端金鑰資料庫

用戶端金鑰資料庫用來作為用戶端應用程式的信任儲存庫。

建立用戶端金鑰資料庫:

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

將先前產生的公開金鑰新增至用戶端金鑰資料庫:

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

為佇列管理程式部署建立包含 TLS 憑證的密鑰

為了讓您的佇列管理程式可以參照及套用金鑰和憑證，請建立 Kubernetes TLS 密鑰，並參照上面建立的檔案。 當您這麼做時，請確定您在開始此作業之前所建立的名稱空間中。

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

建立密鑰，其中包含內部原生 HA TLS 憑證及金鑰

為了讓您的佇列管理程式可以參照及套用金鑰和憑證，請建立 Kubernetes TLS 密鑰，並參照上面建立的檔案。 當您這麼做時，請確定您在開始此作業之前所建立的名稱空間中。

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

建立包含 MQSC 指令的配置對映

建立包含 MQSC 指令的 Kubernetes 配置對映，以建立新的佇列及「SVRCONN 通道」，以及新增通道鑑別記錄，藉由只封鎖那些稱為 *nobody* 的使用者來容許存取通道。

請注意，此方法應該僅用於開發目的。

請確定您位於先前建立的名稱空間中 (請參閱 第 86 頁的『開始之前』)，然後在 OCP 使用者介面中輸入下列 YAML，或使用指令行:

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

配置遞送

如果您使用 IBM MQ 9.2.1 或更新版本的 IBM MQ client 或工具箱，您可以使用佇列管理程式配置檔 (INI 檔案) 來配置遞送至佇列管理程式。 在檔案內，您將 *OutboundSNI* 變數設為根據主機名稱而非通道名稱來遞送。

在您執行指令的目錄中建立名為 `mqclient.ini` 的檔案，其中完全包含下列文字:

```
SSL:
   OutboundSNI=HOSTNAME
```

請勿變更此 INI 檔案中的任何值。 例如，不得變更字串 `HOSTNAME`。

如需進一步詳細資料，請參閱 用戶端配置檔的 SSL 段落。

如果您使用 IBM MQ 9.2.1 之前的 IBM MQ client 或工具箱，則需要建立 OCP 路徑，而不是先前的配置檔。 遵循 附註 1: 建立路徑中的步驟。

部署佇列管理程式

重要事項: 在此範例中，我們使用 *MQSNAUT* 變數來停用佇列管理程式上的授權，這可讓我們聚焦於使用 TLS 連接用戶端所需的步驟。 在 IBM MQ 的正式作業部署中不建議這樣做，因為它會導致任何連接的應用程式都具有完整管理權力，且沒有降低個別應用程式許可權的機制。

複製並更新下列 YAML。

- 請確定已指定正確的授權。 請參閱 mq.ibm.com/v1beta1 的授權參考手冊。 在 IBM Cloud Pak for Integration 2021.1.1 中，授權必須是評估授權 `L-RJON-BYRMYW`
- 將 `false` 變更為 `true` 以接受授權。

佇列管理程式自訂資源 YAML:

```
apiVersion: mq.ibm.com/v1beta1
kind: QueueManager
metadata:
  name: nativeha-example
spec:
  license:
    accept: false
    license: L-RJON-C7QG3S
    use: Production
  queueManager:
    name: HAEXAMPLE
    availability:
      type: NativeHA
      tls:
        secretName: example-ha-secret-internal
    mqsc:
    - configMap:
        name: example-mi-configmap
        items:
        - tls.mqsc
  template:
    pod:
      containers:
        - env:
            - name: MQSNOAUT
              value: 'yes'
          name: qmgr
  version: 9.2.5.0-r3
  pki:
    keys:
      - name: example
        secret:
          secretName: example-ha-secret
          items:
          - tls.key
          - tls.crt
```

確保您位於先前建立的名稱空間中，使用 Red Hat OpenShift Container Platform Web 主控台、指令行或使用 IBM Cloud Pak for Integration Platform Navigator 來部署更新的 YAML。

當系統配置「原生 HA」佇列管理程式時，會有短暫延遲，在此之後佇列管理程式應該可供使用。

驗證

在本節中，我們驗證佇列管理程式的行為符合預期。

確認佇列管理程式正在執行中

現在正在部署佇列管理程式。 請先確認它處於 `Running` 狀態，然後再繼續。 例如:

```
oc get qmgr nativeha-example
```

測試佇列管理程式的連線

若要確認佇列管理程式已配置單向 TLS 通訊，請使用 **amqsputc** 及 **amqsgetc** 範例應用程式:

尋找佇列管理程式主機名稱

若要尋找路徑 `nativeha-example-ibm-mq-qm` 的佇列管理程式主機名稱，請執行下列指令。 主機名稱會在 HOST 欄位中傳回。

```
oc get routes nativeha-example-ibm-mq-qm
```

指定佇列管理程式詳細資料

建立 CCDT.JSON 檔案，以指定佇列管理程式詳細資料。 將主機值取代為前一個步驟所傳回的主機名稱。

```
{
    "channel":
    [
        {
            "name": "HAQMCHL",
            "clientConnection":
            {
                "connection":
                [
                    {
                        "host": "<host from previous step>",
                        "port": 443
                    }
                ],
                "queueManager": "HAEXAMPLE"
            },
            "transmissionSecurity":
            {
              "cipherSpecification": "ECDHE_RSA_AES_128_CBC_SHA256"
            },
            "type": "clientConnection"
        }
   ]
}
```

匯出環境變數

以適合您作業系統的方式匯出下列環境變數。 **amqsputc** 和 **amqsgetc** 將會讀取這些變數。

更新系統上檔案的路徑:

```
export MQCCDTURL='<full_path_to_file>/CCDT.JSON'
export MQSSLKEYR='<full_path_to_file>/clientkey'
```

將訊息放入佇列

請執行下列指令:

```
amqsputc EXAMPLE.QUEUE HAEXAMPLE
```

如果佇列管理程式連線成功，則會輸出下列回應:

```
target queue is EXAMPLE.QUEUE
```

透過輸入一些文字，然後每次按 **Enter** 鍵，將數個訊息放入佇列。

若要完成，請按 **Enter** 鍵兩次。

從佇列擷取訊息

請執行下列指令:

```
amqsgetc EXAMPLE.QUEUE HAEXAMPLE
```

您在前一個步驟中新增的訊息已耗用且為輸出。

在幾秒鐘之後，指令結束。

強制作用中 Pod 失敗

若要驗證佇列管理程式的自動回復，請模擬 Pod 失敗:

檢視作用中及待命 Pod

請執行下列指令:

```
oc get pods --selector app.kubernetes.io/instance=nativeha-example
```

請注意，在 **READY** 欄位中，作用中 Pod 會傳回值 1/1，而抄本 Pod 會傳回值 0/1。

刪除作用中 Pod

執行下列指令，並指定作用中 Pod 的完整名稱:

```
oc delete pod nativeha-example-ibm-mq-<value>
```

再次檢視 Pod 狀態

請執行下列指令:

```
oc get pods --selector app.kubernetes.io/instance=nativeha-example
```

檢視佇列管理程式狀態

執行下列指令，並指定其中一個其他 Pod 的完整名稱:

```
oc exec -t Pod -- dspmq -o nativeha -x -m HAEXAMPLE
```

您應該會看到狀態顯示作用中實例已變更，例如:

```
QMNAME(HAEXAMPLE) ROLE(Active) INSTANCE(inst1) INSYNC(Yes) QUORUM(3/3)
INSTANCE(inst1) ROLE(Active) REPLADDR(9.20.123.45) CONNACTV(Yes) INSYNC(Yes) BACKLOG(0)
CONNINST(Yes) ALTDATE(2022-01-12) ALTTIME(12.03.44)
INSTANCE(inst2) ROLE(Replica) REPLADDR(9.20.123.46) CONNACTV(Yes) INSYNC(Yes) BACKLOG(0)
CONNINST(Yes) ALTDATE(2022-01-12) ALTTIME(12.03.44)
INSTANCE(inst3) ROLE(Replica) REPLADDR(9.20.123.47) CONNACTV(Yes) INSYNC(Yes) BACKLOG(0)
CONNINST(Yes) ALTDATE(2022-01-12) ALTTIME(12.03.44)
```

再次放置並取得訊息

在待命 Pod 變成作用中 Pod (亦即，在 READY 欄位值變成 1/1 之後) 之後，如先前所述，再次使用下列指令將訊息放置到佇列管理程式，然後從佇列管理程式擷取訊息:

```
amqsputc EXAMPLE.QUEUE HAEXAMPLE
```

```
amqsgetc EXAMPLE.QUEUE HAEXAMPLE
```

恭喜，您已順利部署原生 HA 佇列管理程式，並顯示它可以自動從 Pod 失敗中回復。

其他資訊

附註 1: 建立路徑

如果您使用早於 IBM MQ 9.2.1 的 IBM MQ client 或工具箱，則需要建立路徑。

若要建立路徑，請確保您位於先前建立的名稱空間中 (請參閱 第 86 頁的『開始之前』)，然後在 Red Hat OpenShift Container Platform Web 主控台或使用指令行輸入下列 YAML:

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

請注意， Red Hat OpenShift Container Platform Router 會使用 SNI 將要求遞送至 IBM MQ 佇列管理程式。 如果您變更 包含 MQSC 指令的配置對映中指定的通道名稱，則也必須在這裡變更主機欄位，以及在 CCDT.JSON 檔案中指定佇列管理程式詳細資料。 如需相關資訊，請參閱 第 98 頁的『配置路徑以從 Red Hat OpenShift 叢集外部連接至佇列管理程式』。

附註 2: 使用非預設儲存類別

此範例預期已在 Red Hat OpenShift Container Platform 中配置預設儲存類別，因此 佇列管理程式自訂資源 YAML 中不需要任何儲存體資訊。 如果您未將儲存類別配置為預設值，或想要使用不同的儲存類別，請在 spec.queueManager.storage 下新增 defaultClass: *<storage_class_name>*。

儲存類別名稱必須完全符合已存在的儲存類別名稱。 也就是說，它必須符合指令 oc get storageclass 所傳回的名稱。 它也必須支援 ReadWriteMany。 如需相關資訊，請參閱第 10 頁的『IBM MQ Operator 的儲存體考量』。

相關工作

第 91 頁的『檢視 IBM MQ 認證儲存器的原生 HA 佇列管理程式狀態』
對於 IBM MQ 認證容器，您可以透過在其中一個執行中 Pod 內執行 **dspmq** 指令，來檢視原生 HA 實例的狀態。


檢視 *IBM MQ* 認證儲存器的原生 *HA* 佇列管理程式狀態

對於 IBM MQ 認證容器，您可以透過在其中一個執行中 Pod 內執行 **dspmq** 指令，來檢視原生 HA 實例的狀態。

關於這項作業

重要:

您可以在其中一個執行中 Pod 中使用 **dspmq** 指令，以檢視佇列管理程式實例的作業狀態。 傳回的資訊視實例是作用中還是抄本而定。 作用中實例所提供的資訊是明確的，抄本節點的資訊可能已過期。

您可以執行下列動作:

- 檢視現行節點上的佇列管理程式實例是作用中還是抄本。
- 檢視現行節點上實例的原生 HA 作業狀態。
- 檢視原生 HA 配置中所有三個實例的作業狀態。

下列狀態欄位用來報告原生 HA 配置狀態:

角色
指定實例的現行角色，並且是 `Active`、 `Replica` 或 `Unknown` 之一。

實例
使用 **crtmqm** 指令的 **-lr** 選項建立此佇列管理程式實例時，為其提供的名稱。

INSYNC
指出實例是否可以在必要時接管作為作用中實例。

仲裁
以 *number_of_instances_in-sync*/*number_of_instances_configured* 格式報告仲裁狀態。

REPLADDR
佇列管理程式實例的抄寫位址。

CONNACTV
指出節點是否連接至作用中實例。

BACKLOG
指出實例落後的 KB 數。

CONNINST
指出指定的實例是否連接至此實例。

ALTDATE
指出前次更新此資訊的日期 (如果從未更新過，則為空白)。

ALTTIME
指出前次更新此資訊的時間 (如果從未更新過，則為空白)。

程序

- 尋找屬於佇列管理程式的 Pod。

```
oc get pod --selector app.kubernetes.io/instance=nativeha-qm
```

- 在其中一個 Pod 中執行 `dspmq`

```
oc exec -t Pod dspmq
```

```
oc rsh Pod
```

適用於互動式 Shell，您可以在其中直接執行 dspmq。

- 若要判定佇列管理程式實例是作為作用中實例還是抄本執行，請執行下列動作:

```
oc exec -t Pod dspmq -o status -m QMgrName
```

名為 BOB 之佇列管理程式的作用中實例會報告下列狀態:

```
QMNAME(BOB)                STATUS(Running)
```

名為 BOB 之佇列管理程式的抄本實例會報告下列狀態:

```
QMNAME(BOB)                STATUS(Replica)
```

非作用中實例會報告下列狀態:

```
QMNAME(BOB)                STATUS(Ended Immediately)
```

- 若要判定所指定 Pod 中實例的原生 HA 作業狀態，請執行下列動作:

```
oc exec -t Pod dspmq -o nativeha -m QMgrName
```

名為 BOB 之佇列管理程式的作用中實例可能報告下列狀態:

```
QMNAME(BOB)              ROLE(Active) INSTANCE(inst1) INSYNC(Yes) QUORUM(3/3)
```

佇列管理程式 BOB 的抄本實例可能會報告下列狀態:

```
QMNAME(BOB)              ROLE(Replica) INSTANCE(inst2) INSYNC(Yes) QUORUM(2/3)
```

名為 BOB 之佇列管理程式的非作用中實例可能會報告下列狀態:

```
QMNAME(BOB)              ROLE(Unknown) INSTANCE(inst3) INSYNC(no) QUORUM(0/3)
```

- 若要判定「原生 HA」配置中所有實例的「原生 HA」作業狀態，請執行下列動作:

```
oc exec -t Pod dspmq -o nativeha -x -m QMgrName
```

如果您在執行佇列管理程式 BOB 作用中實例的節點上發出此指令，則可能會收到下列狀態:

```
QMNAME(BOB)            ROLE(Active) INSTANCE(inst1) INSYNC(Yes) QUORUM(3/3)
 INSTANCE(inst1) ROLE(Active)  REPLADDR(9.20.123.45) CONNACTV(Yes) INSYNC(Yes) BACKLOG(0)
CONNINST(Yes) ALTDATE(2022-01-12) ALTTIME(12.03.44)
 INSTANCE(inst2) ROLE(Replica) REPLADDR(9.20.123.46) CONNACTV(Yes) INSYNC(Yes) BACKLOG(0)
CONNINST(Yes) ALTDATE(2022-01-12) ALTTIME(12.03.44)
 INSTANCE(inst3) ROLE(Replica) REPLADDR(9.20.123.47) CONNACTV(Yes) INSYNC(Yes) BACKLOG(0)
CONNINST(Yes) ALTDATE(2022-01-12) ALTTIME(12.03.44)
```

如果您在執行佇列管理程式 BOB 抄本實例的節點上發出此指令，則可能會收到下列狀態，指出其中一個抄本落後:

```
QMNAME(BOB)            ROLE(Replica) INSTANCE(inst2) INSYNC(Yes) QUORUM(2/3)
 INSTANCE(inst2) ROLE(Replica) REPLADDR(9.20.123.46) CONNACTV(Yes) INSYNC(Yes) BACKLOG(0)
CONNINST(Yes) ALTDATE(2022-01-12) ALTTIME(12.03.44)
 INSTANCE(inst1) ROLE(Active)  REPLADDR(9.20.123.45) CONNACTV(Yes) INSYNC(Yes) BACKLOG(0)
CONNINST(Yes) ALTDATE(2022-01-12) ALTTIME(12.03.44)
 INSTANCE(inst3) ROLE(Replica) REPLADDR(9.20.123.47) CONNACTV(Yes) INSYNC(No)  BACKLOG(435)
CONNINST(Yes) ALTDATE(2022-01-12) ALTTIME(12.03.44)
```

如果您在執行佇列管理程式 BOB 非作用中實例的節點上發出此指令，則可能會收到下列狀態:

```
QMNAME(BOB)            ROLE(Unknown) INSTANCE(inst3) INSYNC(no) QUORUM(0/3)
 INSTANCE(inst1) ROLE(Unknown) REPLADDR(9.20.123.45) CONNACTV(Unknown) INSYNC(Unknown)
BACKLOG(Unknown) CONNINST(No) ALTDATE() ALTTIME()
 INSTANCE(inst2) ROLE(Unknown) REPLADDR(9.20.123.46) CONNACTV(Unknown) INSYNC(Unknown)
BACKLOG(Unknown) CONNINST(No) ALTDATE() ALTTIME()
 INSTANCE(inst3) ROLE(Unknown) REPLADDR(9.20.123.47) CONNACTV(No) INSYNC(Unknown)
BACKLOG(Unknown) CONNINST(No) ALTDATE() ALTTIME()
```

如果您在實例仍在協議作用中及抄本時發出指令，則會收到下列狀態:

```
QMNAME(BOB)               STATUS(Negotiating)
```

相關參考

dspmq (顯示佇列管理程式) 指令

第 86 頁的『範例: 配置原生 HA 佇列管理程式』

此範例顯示如何使用 IBM MQ Operator，將使用原生高可用性特性的佇列管理程式部署至 Red Hat OpenShift Container Platform (OCP)。

CP4I CD V 9.2.3 原生 *HA* 的進階調整

調整計時及間隔的進階設定。 除非已知預設值不符合您系統的需求，否則應該不需要使用這些設定。

用於配置原生 HA 的基本選項是使用 QueueManager API 來處理， IBM MQ Operator 會使用該 API 來為您配置基礎佇列管理程式 INI 檔。 在 NativeHALocal 實例段落下，有一些其他進階選項只能使用 INI 檔案來配置。 如需如何配置 INI 檔案的相關資訊，請參閱 第 78 頁的『範例: 提供 MQSC 及 INI 檔案』 。

HeartbeatInterval
活動訊號間隔定義原生 HA 佇列管理程式的作用中實例傳送網路活動訊號的頻率 (毫秒)。 活動訊號間隔值的有效範圍是 500 (0.5 秒) 至 60000 (1 分鐘) ，超出此範圍的值會導致佇列管理程式無法啟動。 如果省略此屬性，則會使用預設值 5000 (5 秒)。 每一個實例都必須使用相同的活動訊號間隔。

HeartbeatTimeout
活動訊號逾時值定義原生 HA 佇列管理程式的抄本實例在決定作用中實例無回應之前等待的時間。 活動訊號間隔逾時值的有效範圍是 500 (0.5 秒) 至 120000 (2 分鐘)。 活動訊號逾時值必須大於或等於活動訊號間隔。

無效值會導致佇列管理程式無法啟動。 如果省略此屬性，則抄本會等待 2 x HeartbeatInterval ，然後再啟動處理程序來選取新的作用中實例。 每一個實例都必須使用相同的活動訊號逾時。

RetryInterval
重試間隔定義原生 HA 佇列管理程式應該重試失敗抄寫鏈結的頻率 (毫秒)。 重試間隔的有效範圍是 500 (0.5 秒) 至 120000 (2 分鐘)。 如果省略此屬性，在重試失敗的抄寫鏈結之前，抄本會等待 2 x HeartbeatInterval 。

CP4I 正在結束原生 *HA* 佇列管理程式

您可以使用 **endmqm** 指令來結束屬於「原生 HA」群組的作用中或抄本佇列管理程式。

程序

- 若要結束佇列管理程式的作用中實例，請參閱本文件「配置」一節中的 結束原生 HA 佇列管理程式 。

CP4I CD V 9.2.2 在 *IBM Cloud Pak for Integration 2021.1.1* 中評估原生 *HA* 特性

IBM Cloud Pak for Integration 2021.1.1 原生 HA 評估期間已結束。 請使用 IBM Cloud Pak for Integration 2021.2.1 中提供的已更新「原生 HA」特性，搭配使用 IBM MQ Operator 1.6 或更高版本與 IBM MQ 9.2.3 或更高版本。

相關工作

第 91 頁的『檢視 IBM MQ 認證儲存器的原生 HA 佇列管理程式狀態』

對於 IBM MQ 認證容器，您可以透過在其中一個執行中 Pod 內執行 **dspmq** 指令，來檢視原生 HA 實例的狀態。

相關參考

第 86 頁的『範例: 配置原生 HA 佇列管理程式』

此範例顯示如何使用 IBM MQ Operator，將使用原生高可用性特性的佇列管理程式部署至 Red Hat OpenShift Container Platform (OCP)。

OpenShift CP4I 範例: 配置多重實例佇列管理程式

此範例顯示如何使用 IBM MQ Operator 將多重實例佇列管理程式部署至 Red Hat OpenShift Container Platform (OCP)。 在此範例中，您還可以在範例用戶端與佇列管理程式之間配置單向 TLS 通訊。 此範例示範如何在模擬 Pod 失敗之前及之後放置及取得訊息，以順利完成配置。

開始之前

若要完成此範例，您必須先完成下列必要條件:

- 安裝 IBM MQ client，並將已安裝的 `samp/bin` 及 `bin` 目錄新增至 *PATH*。 用戶端提供此範例所需的 **runmqakm**、 **amqsputc** 及 **amqsgetc** 應用程式。 安裝 IBM MQ client ，如下所示:
  - Windows Linux Windows 和 Linux: 從 https://ibm.biz/mq92redistclients 安裝適用於您作業系統的 IBM MQ 可重新配送用戶端。
  - macOS Mac: 下載並設定 IBM MQ MacOS Toolkit。 請參閱 https://developer.ibm.com/tutorials/mq-macos-dev/。
- 安裝適用於您作業系統的 OpenSSL 工具。 如果您還沒有私密金鑰和憑證，則需要這樣才能為佇列管理程式產生自簽憑證。
- 針對此範例建立 Red Hat OpenShift Container Platform (OCP) 專案/名稱空間。
- 在指令行上，登入 OCP 叢集，並切換至上述名稱空間。
- 請確定上述名稱空間中已安裝 IBM MQ Operator 且可供使用。
- 在 OCP 中配置要由佇列管理程式使用的預設儲存類別。 如果您想要在不設定預設儲存空間類別的情況下完成本指導教學，請參閱 附註 2: 使用非預設儲存空間類別。

關於此作業

多重實例佇列管理程式涉及作用中及待命 Kubernetes Pod。 這些會作為「Kubernetes 有狀態集」的一部分執行，其中正好有兩個抄本及一組 Kubernetes 持續性磁區。 如需多重實例佇列管理程式的相關資訊，請參閱 第 15 頁的『儲存器中 IBM MQ 的高可用性』。

此範例提供自訂資源 YAML，定義具有持續性儲存體且使用 TLS 配置的多重實例佇列管理程式。 將佇列管理程式部署至 OCP 之後，您會模擬作用中佇列管理程式 Pod 的失敗。 您會看到自動回復發生，並在失敗之後放置並取得訊息，以證明它已成功。

範例

為 MQ 伺服器建立 TLS 私密金鑰和憑證

本節說明如何為佇列管理程式建立自簽憑證，以及如何將憑證新增至金鑰資料庫作為用戶端的信任儲存庫。 如果您已有私密金鑰和憑證，則可以改用那些憑證。 請注意，您只應該將自簽憑證用於開發目的。

若要在現行目錄中建立自簽私密金鑰及公用憑證，請執行下列指令:

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

將佇列管理程式公開金鑰新增至用戶端金鑰資料庫

用戶端金鑰資料庫用來作為用戶端應用程式的信任儲存庫。

建立用戶端金鑰資料庫:

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

將先前產生的公開金鑰新增至用戶端金鑰資料庫:

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

為佇列管理程式部署建立包含 TLS 憑證的密鑰

為了讓您的佇列管理程式可以參照及套用金鑰和憑證，請建立 Kubernetes TLS 密鑰，並參照上面建立的檔案。 當您這麼做時，請確定您在開始此作業之前所建立的名稱空間中。

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

建立包含 MQSC 指令的配置對映

建立包含 MQSC 指令的 Kubernetes 配置對映，以建立新的佇列及「SVRCONN 通道」，以及新增通道鑑別記錄，藉由只封鎖那些稱為 *nobody* 的使用者來容許存取通道。

請注意，此方法應該僅用於開發目的。

請確定您位於先前建立的名稱空間中 (請參閱 第 94 頁的『開始之前』)，然後在 OCP 使用者介面中輸入下列 YAML，或使用指令行:

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

配置遞送

如果您使用 IBM MQ 9.2.1 或更新版本的 IBM MQ client 或工具箱，您可以使用佇列管理程式配置檔 (INI 檔案) 來配置遞送至佇列管理程式。 在檔案內，您將 *OutboundSNI* 變數設為根據主機名稱而非通道名稱來遞送。

在您執行指令的目錄 (稱為 `mqclient.ini`) 中建立檔案，其中包含下列文字:

```
#* Module Name: mqclient.ini                                          *#
#* Type       : IBM MQ MQI client configuration file                  *#
#  Function   : Define the configuration of a client                  *#
#*                                                                    *#
#**********************************************************************#
#* Notes      :                                                       *#
#* 1) This file defines the configuration of a client                 *#
#*                                                                    *#
#**********************************************************************#
SSL:
   OutboundSNI=HOSTNAME
```

附註: 請勿變更此頁面中的任何值。 例如，字串 HOSTNAME 應該依現狀保留。

如需進一步詳細資料，請參閱 用戶端配置檔的 SSL 段落。

如果您使用 IBM MQ 9.2.1 之前的 IBM MQ client 或工具箱，則需要建立 OCP 路徑，而不是先前的配置檔。 遵循 附註 1: 建立路徑中的步驟。

部署佇列管理程式

重要事項: 在此範例中，我們使用 *MQSNAUT* 變數來停用佇列管理程式上的授權，這可讓我們聚焦於使用 TLS 連接用戶端所需的步驟。 在 IBM MQ 的正式作業部署中不建議這樣做，因為它會導致任何連接的應用程式都具有完整管理權力，且沒有降低個別應用程式許可權的機制。

複製並更新下列 YAML。

- 請確定已指定正確的授權。 請參閱 mq.ibm.com/v1beta1 的授權參考手冊。
- 將 `false` 變更為 `true` 以接受授權。
- 如果您使用 IBM Cloud File Storage，請參閱 附註 3: 使用 IBM Cloud File Storage

佇列管理程式自訂資源 YAML:

```
apiVersion: mq.ibm.com/v1beta1
```

```
kind: QueueManager
metadata:
  name: miexample
spec:
  license:
    accept: false
    license: L-RJON-C7QG3S
    use: NonProduction
  queueManager:
    name: MIEXAMPLE
    availability:
      type: MultiInstance
    mqsc:
    - configMap:
        name: example-mi-configmap
        items:
        - tls.mqsc
  template:
    pod:
      containers:
        - env:
            - name: MQSNOAUT
              value: 'yes'
          name: qmgr
  version: 9.2.5.0-r3
  web:
    enabled: true
  pki:
    keys:
      - name: example
        secret:
          secretName: example-mi-secret
          items:
          - tls.key
          - tls.crt
```

確保您位於先前建立的名稱空間中，使用指令行或使用 IBM Cloud Pak for Integration Platform Navigator，在 OCP 使用者介面中部署更新的 YAML。

驗證

在短暫延遲之後，應該配置多重實例佇列管理程式，且可供使用。 在本節中，我們驗證佇列管理程式的行為符合預期。

確認佇列管理程式正在執行中

現在正在部署佇列管理程式。 請先確認它處於 Running 狀態，然後再繼續。 例如：

```
oc get qmgr miexample
```

測試佇列管理程式的連線

若要確認佇列管理程式已配置單向 TLS 通訊，請使用 **amqsputc** 及 **amqsgetc** 範例應用程式：

尋找佇列管理程式主機名稱

若要尋找路徑 miexample-ibm-mq-qm 的佇列管理程式主機名稱，請執行下列指令。 主機名稱會在 HOST 欄位中傳回。

```
oc get routes miexample-ibm-mq-qm
```

指定佇列管理程式詳細資料

建立 CCDT.JSON 檔案，以指定佇列管理程式詳細資料。 將主機值取代為前一個步驟所傳回的主機名稱。

```
{
    "channel":
    [
        {
            "name": "MIQMCHL",
            "clientConnection":
            {
                "connection":
                [
                    {
                        "host": "<host from previous step>",
                        "port": 443
```

```
                    }
                ],
                "queueManager": "MIEXAMPLE"
            },
            "transmissionSecurity":
            {
              "cipherSpecification": "ECDHE_RSA_AES_128_CBC_SHA256"
            },
            "type": "clientConnection"
        }
    ]
}
```

匯出環境變數

以適合您作業系統的方式匯出下列環境變數。 **amqsputc** 和 **amqsgetc** 將會讀取這些變數。

更新系統上檔案的路徑:

```
export MQCCDTURL='<full_path_to_file>/CCDT.JSON'
export MQSSLKEYR='<full_path_to_file>/clientkey'
```

將訊息放入佇列

請執行下列指令:

```
amqsputc EXAMPLE.QUEUE MIEXAMPLE
```

如果佇列管理程式連線成功，則會輸出下列回應:

```
target queue is EXAMPLE.QUEUE
```

透過輸入一些文字，然後每次按 **Enter** 鍵，將數個訊息放入佇列。

若要完成，請按 **Enter** 鍵兩次。

從佇列擷取訊息

請執行下列指令:

```
amqsgetc EXAMPLE.QUEUE MIEXAMPLE
```

您在前一個步驟中新增的訊息已耗用且為輸出。

在幾秒鐘之後，指令結束。

強制作用中 Pod 失敗

若要驗證多重實例佇列管理程式的自動回復，請模擬 Pod 失敗:

檢視作用中及待命 Pod

請執行下列指令:

```
oc get pods
```

請注意，在 **READY** 欄位中，作用中 Pod 會傳回值 `1/1`，而待命 Pod 會傳回值 `0/1`。

刪除作用中 Pod

執行下列指令，並指定作用中 Pod 的完整名稱:

```
oc delete pod miexample-ibm-mq-<value>
```

再次檢視 Pod 狀態

請執行下列指令:

```
oc get pods
```

檢視待命 Pod 日誌

執行下列指令，並指定待命 Pod 的完整名稱:

```
oc logs miexample-ibm-mq-<value>
```

您應該會看到下列訊息:

```
IBM MQ queue manager 'MIEXAMPLE' becoming the active instance.
```

再次放置並取得訊息

在待命 Pod 變成作用中 Pod (亦即，在 READY 欄位值變成 1/1 之後) 之後，如先前所述，再次使用下列指令將訊息放置到佇列管理程式，然後從佇列管理程式擷取訊息:

```
amqsputc EXAMPLE.QUEUE MIEXAMPLE
```

```
amqsgetc EXAMPLE.QUEUE MIEXAMPLE
```

恭喜您，您已順利部署多重實例佇列管理程式，並顯示它可以自動從 Pod 失敗中回復。

其他資訊

附註 1: 建立路徑

如果您使用早於 IBM MQ 9.2.1 的 IBM MQ client 或工具箱，則需要建立 OCP 路徑。

若要建立路徑，請確定您位於先前建立的名稱空間中 (請參閱 第 94 頁的『開始之前』)，然後在 OCP 使用者介面中輸入下列 YAML，或使用指令行:

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

請注意， Red Hat OpenShift Container Platform Router 會使用 SNI 將要求遞送至 IBM MQ 佇列管理程式。 如果您變更 包含 MQSC 指令的配置對映中指定的通道名稱，則也必須在這裡變更主機欄位，以及在 CCDT.JSON 檔案中指定佇列管理程式詳細資料。 如需相關資訊，請參閱 第 98 頁的『配置路徑以從 Red Hat OpenShift 叢集外部連接至佇列管理程式』。

附註 2: 使用非預設儲存類別

此範例預期已在 OCP 中配置預設儲存類別，因此 佇列管理程式自訂資源 YAML 中不需要任何儲存體資訊。 如果您未將儲存類別配置為預設值，或想要使用不同的儲存類別，請在 `spec.queueManager.storage` 下新增 `defaultClass: <storage_class_name>`。

儲存類別名稱必須完全符合 OCP 系統上存在的儲存類別名稱。 也就是說，它必須符合指令 `oc get storageclass` 所傳回的名稱。 它也必須支援 ReadWriteMany。 如需相關資訊，請參閱第 10 頁的『IBM MQ Operator 的儲存體考量』。

附註 3: 使用 IBM Cloud File Storage

在某些情況下，例如使用 IBM Cloud File Storage 時，您也需要在 佇列管理程式自訂資源 YAML 中指定 **securityGroups** 欄位。 例如，直接在 spec: 下新增下列子欄位

```
  securityContext:
    supplementalGroups: [99]
```

如需相關資訊，請參閱 第 10 頁的『IBM MQ Operator 的儲存體考量』。

OpenShift CP4I CD 配置路徑以從 Red Hat OpenShift 叢集外部連接至佇列管理程式

您需要 Red Hat OpenShift 路徑，才能從 Red Hat OpenShift 叢集外部將應用程式連接至 IBM MQ 佇列管理程式。 您必須在 IBM MQ 佇列管理程式及用戶端應用程式上啟用 TLS，因為只有在使用 TLS 1.2 或更高版本的通訊協定時，才能在 TLS 通訊協定中使用 SNI。 Red Hat OpenShift Container Platform Router 使用 SNI 將要求遞送至 IBM MQ 佇列管理程式。

關於這項作業

小心: 本文件適用於 IBM MQ 用戶端 9.2.1 Continuous Delivery 版以及更新版本。 如果您的用戶端是使用 9.2.0 Long Term Support 或更舊版本，請參閱 IBM MQ 9.1 文件頁面 連接至 Red Hat OpenShift 叢集中部署的佇列管理程式。

V 9.2.1 Red Hat OpenShift 路徑 的必要配置取決於用戶端應用程式的 伺服器名稱指示 (SNI) 行為。 IBM MQ 支援兩種不同的 SNI 標頭設定，視配置和用戶端類型而定。 「SNI 標頭」設為用戶端目的地的主機名稱，或設為 IBM MQ 通道名稱。 如需 IBM MQ 如何將通道名稱對映至主機名稱的相關資訊，請參閱 IBM MQ 如何提供多個憑證功能。

V 9.2.1 SNI 標頭是設為 IBM MQ 通道名稱，還是使用 **OutboundSNI** 屬性來控制主機名稱。 可能的值為 `OutboundSNI=CHANNEL` (預設值) 或 `OutboundSNI=HOSTNAME`。 如需相關資訊，請參閱 用戶端配置檔的 SSL 段落。 請注意， `CHANNEL` 和 `HOSTNAME` 是您使用的確切值; 它們不是您取代為實際通道名稱或主機名稱的變數名稱。

V 9.2.1

具有不同 OutboundSNI 設定的用戶端行為

如果 **OutboundSNI** 設為 `HOSTNAME`，則只要在連線名稱中提供主機名稱，下列用戶端就會設定主機名稱 SNI:

- C 用戶端
- 未受管理模式中的 .NET 用戶端
- Java/JMS 用戶端

如果 **OutboundSNI** 設為 `HOSTNAME` ， 且在連線名稱中使用 IP 位址，則下列用戶端會傳送空白 SNI 標頭:

- C 用戶端
- 未受管理模式中的 .NET 用戶端
- Java/JMS 用戶端 (無法對主機名稱執行反向 DNS 查閱)

如果 **OutboundSNI** 設為 `CHANNEL`，則不論使用的是主機名稱還是 IP 位址連線名稱，都會改用 IBM MQ 通道名稱並一律傳送。

下列用戶端類型不支援將 SNI 標頭設為 IBM MQ 通道名稱，因此不論 **OutboundSNI** 設定為何，一律嘗試將 SNI 標頭設為主機名稱:

- AMQP 用戶端
- XR 用戶端
- 受管理模式中的 .NET 用戶端 (IBM MQ 9.2.0 Fix Pack 4 for Long Term Support 之前 在 IBM MQ 9.2.3 for Continuous Delivery 之前。)

V 9.2.0.4 V 9.2.3

從 IBM MQ 9.2.0 Fix Pack 4 for Long Term Support 及 IBM MQ 9.2.3 for Continuous Delivery，如果 **OutboundSNI** 內容設為 `HOSTNAME`，則 IBM MQ 受管理 .NET 用戶端已更新為將 SERVERNAME 設為個別的主機名稱，這容許 IBM MQ 受管理 .NET 用戶端使用 Red Hat OpenShift 路徑來連接至佇列管理程式。 請注意，在 IBM MQ 9.2.0 Fix Pack 4 中，只會從 `mqclient.ini` 檔新增 **OutboundSNI** 內容並予以支援; 您無法從 .NET 應用程式設定該內容。

V 9.2.5

如果用戶端應用程式透過 IBM MQ Internet Pass-Thru (MQIPT) 連接至部署在 Red Hat OpenShift 叢集中的佇列管理程式，則可以使用路徑定義中的 SSLClientOutboundSNI 內容，將 MQIPT 配置為將 SNI 設為主機名稱。

OutboundSNI、多個憑證及 Red Hat OpenShift 路徑

IBM MQ 使用 SNI 標頭來提供多個憑證功能。 如果應用程式連接至透過 CERTLABL 欄位配置為使用不同憑證的 IBM MQ 通道，則應用程式必須使用 CHANNEL 的 **OutboundSNI** 設定進行連接。

如果 Red Hat OpenShift Route 配置需要 HOSTNAME SNI，則您無法使用 IBM MQ 的多個憑證功能，也無法在任何 IBM MQ 通道物件上設定 CERTLABL 設定。

如果 **OutboundSNI** 設定不是 CHANNEL 的應用程式連接至已配置憑證標籤的通道，則會拒絕該應用程式，並在佇列管理程式錯誤日誌中列印 AMQ9673 訊息。

如需 IBM MQ 如何提供多個憑證功能的相關資訊，請參閱 IBM MQ 如何提供多個憑證功能 。

範例

將 SNI 設為 MQ 通道的用戶端應用程式需要針對您要連接的每一個通道建立新的 Red Hat OpenShift 路徑。您也必須在 Red Hat OpenShift Container Platform 叢集中使用唯一通道名稱，以容許遞送至正確的佇列管理程式。

由於 IBM MQ 將通道名稱對映至 SNI 標頭的方式， MQ 通道名稱不能以小寫字母結尾是很重要的。

若要判定每一個新的 Red Hat OpenShift 路徑所需的主機名稱，您需要將每一個通道名稱對映至 SNI 位址。如需相關資訊，請參閱 IBM MQ 如何提供多個憑證功能 。

然後，您必須在叢集裡套用下列 `yaml`，為每一個通道建立新的 Red Hat OpenShift 路徑:

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

配置用戶端應用程式連線詳細資料

您可以執行下列指令來決定要用於用戶端連線的主機名稱:

```
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}"
```

用戶端連線的埠應該設為「 Red Hat OpenShift Container Platform 路由器」所使用的埠-通常是 443。

相關工作

第 104 頁的『連接至 Red Hat OpenShift 叢集中部署的 IBM MQ Console』
如何連接至已部署至 Red Hat OpenShift Container Platform 叢集之佇列管理程式的 IBM MQ Console 。

CP4I 與 IBM Cloud Pak for Integration 作業儀表板整合

透過 IBM Cloud Pak for Integration 追蹤交易的能力由「作業儀表板」提供。

關於這項作業

啟用與「作業儀表板」的整合會將 MQ API 結束程式安裝至佇列管理程式。 API 結束程式會將追蹤資料傳送至「作業儀表板」資料儲存庫，以瞭解流經佇列管理程式的訊息。

請注意，只會追蹤使用 MQ 用戶端連結傳送的訊息。

另請注意，對於 1.5 之前的 IBM MQ Operator 版本，當啟用追蹤時，與佇列管理程式一起部署的追蹤代理程式及收集器映像檔一律是可用的最新版本，如果您未使用最新版本的 IBM Cloud Pak for Integration，可能會導致不相容。

程序

1. 部署已啟用追蹤的佇列管理程式

   依預設，會停用追蹤特性。

   如果您使用 IBM Cloud Pak for Integration Platform Navigator 進行部署，則可以在部署時啟用追蹤，方法是將 **啟用追蹤** 設為 **開啟**，並將 **追蹤名稱空間** 設為已安裝「作業儀表板」的名稱空間。 如需部署佇列管理程式的相關資訊，請參閱 第 75 頁的『使用 IBM Cloud Pak for Integration Platform Navigator 部署佇列管理程式』

   如果您使用 Red Hat OpenShift CLI 或 Red Hat OpenShift Web 主控台進行部署，則可以使用下列 YAML Snippet 來啟用追蹤:

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

   重要事項: 在向「作業儀表板」登錄 MQ 之前，佇列管理程式將不會啟動 (請參閱下一步)。

   請注意，當啟用此特性時，除了佇列管理程式儲存器之外，它還會執行兩個 Sidecar 儲存器 (「代理程式」和「收集器」)。 這些 Sidecar 儲存器的映像檔將在與主要 MQ 映像檔相同的登錄中提供，並且將使用相同的取回原則和取回密碼。 還有其他設定可用來配置 CPU 和記憶體限制。
2. 如果這是第一次在此名稱空間中部署具有「作業儀表板」整合的佇列管理程式，則您需要向「作業儀表板」 登錄 。

   登錄會建立佇列管理程式 Pod 需要順利啟動的「密鑰」物件。

CP4I CD 在 *IBM Cloud Pak for Integration 2021.4* 中使用「作業儀表板」整合來部署或升級 *IBM MQ 9.2.2* 或 *9.2.3*

每一個 IBM MQ 版本都與特定版本的「作業儀表板」代理程式及收集器元件相關聯，這些元件與佇列管理程式一起部署。 IBM Cloud Pak for Integration 2021.4.1 引進了一項變更，導致較舊的代理程式及收集器元件無法使用「作業儀表板」。 若要修正此問題，當您使用 IBM MQ 9.2.2 或 9.2.3 時，必須置換您使用的 Operations Dashboard 代理程式及收集器映像檔的版本。

部署新的 IBM MQ 9.2.2 或 9.2.3 佇列管理程式

將 IBM Cloud Pak for Integration 2021.4.1 與 IBM MQ 9.2.2 或 9.2.3 搭配使用時，您必須將 Operations Dashboard 代理程式及收集器映像檔置換為 `QueueManager` YAML 中的 2.4 版本。 例如:

```
spec:
  tracing:
    agent:
      image: cp.icr.io/cp/icp4i/od/icp4i-od-
agent@sha256:27a211f0f78eff765d1f9520e0f9841f902600bb556827477b206e209cb44d20
    collector:
      image: cp.icr.io/cp/icp4i/od/icp4i-od-
collector@sha256:dc70b1341b23dc72642ce68809811f9db0e8a0c46bda2508e8eb3d4035e04f4b
```

如果不這麼做，您的 `QueueManager` Pod 將停留在 `Pending` 狀態。 當您升級至 IBM MQ 9.2.4 時，可以移除這些置換。

升級至 IBM Cloud Pak for Integration 2021.4.1

註: 如果您要保留 IBM MQ 9.2.2 或 9.2.3 佇列管理程式，請不要完成步驟 3。

1. 更新 `QueueManager` 以置換代理程式及收集器映像檔，如先前所述。
2. 升級 IBM Cloud Pak for Integration 操作員，包括「作業儀表板」及 IBM MQ 操作員，如 第 66 頁的『升級 IBM MQ Operator 及佇列管理程式』中所述。

3. (選用) 若要升級至 IBM MQ 9.2.4 或更新版本，請更新 QueueManager 以將 .spec.version 用於 IBM MQ 版本，然後移除代理程式及收集器映像檔的置換。

OpenShift CP4I 使用 Red Hat OpenShift CLI 以自訂 MQSC 及 INI 檔案建置映像檔

使用「Red Hat OpenShift Container Platform 管線」來建立新的 IBM MQ 儲存器映像檔，並將您想要套用至使用此映像檔之佇列管理程式的 MQSC 及 INI 檔案。 此作業應由專案管理者完成

開始之前

您需要安裝 Red Hat OpenShift Container Platform 指令行介面。

使用 **cloudctl login** (適用於 IBM Cloud Pak for Integration) 或 **oc login** 登入叢集。

如果您在 Red Hat OpenShift 專案中沒有 IBM Entitled Registry 的 Red Hat OpenShift 密鑰，請遵循 第 73 頁的『準備 IBM MQ 的 Red Hat OpenShift 專案』的步驟。

程序

1. 建立 ImageStream

   映像檔串流及其相關聯標籤提供抽象概念，可從 Red Hat OpenShift Container Platform 內參照儲存器映像檔。 映像檔串流及其標籤可讓您查看可用的映像檔，並確保您正在使用您需要的特定映像檔，即使儲存庫中的映像檔變更也一樣。

   ```
   oc create imagestream mymq
   ```

2. 為新映像檔建立 BuildConfig

   BuildConfig 將容許對新映像檔進行建置，該映像檔將基於 IBM 正式映像檔，但將新增您要在容器啟動時執行的任何 MQSC 或 INI 檔案。

   a) 建立定義 BuildConfig 資源的 YAML 檔

   例如，使用下列內容建立稱為 "mq-build-config.yaml" 的檔案:

   ```
   apiVersion: build.openshift.io/v1
   kind: BuildConfig
   metadata:
     name: mymq
   spec:
     source:
       dockerfile: |-
         FROM cp.icr.io/cp/ibm-mqadvanced-server-integration:9.2.5.0-r3
         RUN printf "DEFINE QLOCAL(foo) REPLACE\n" > /etc/mqm/my.mqsc \
           && printf "Channels:\n\tMQIBindType=FASTPATH\n" > /etc/mqm/my.ini
         LABEL summary "My custom MQ image"
     strategy:
       type: Docker
       dockerStrategy:
         from:
           kind: "DockerImage"
           name: "cp.icr.io/cp/ibm-mqadvanced-server-integration:9.2.5.0-r3"
         pullSecret:
           name: ibm-entitlement-key
     output:
       to:
         kind: ImageStreamTag
         name: 'mymq:latest-amd64'
   ```

   您需要取代提及基本 IBM MQ 的兩個位置，以指向您要使用之版本及修正程式的正確基本映像檔 (如需詳細資料，請參閱 第 18 頁的『IBM MQ Operator 的發行歷程』)。 套用修正程式時，您需要重複這些步驟來重建映像檔。

   此範例會根據 IBM 正式映像檔建立新的映像檔，並將稱為 "my.mqsc" 及 "my.ini" 的檔案新增至 /etc/mqm 目錄。 儲存器會在啟動時套用在此目錄中找到的任何 MQSC 或 INI 檔。 INI 檔案會使用 **crtmqm -ii** 選項來套用，並與現有的 INI 檔案合併。 MQSC 檔案按字母順序套用。

MQSC 指令必須可重複，因為 每次 佇列管理程式啟動時都會執行這些指令。 這通常表示在任何 DEFINE 指令上新增 REPLACE 參數，並將 IGNSTATE(YES) 參數新增至任何 START 或 STOP 指令。

b) 將 BuildConfig 套用至伺服器。

```
oc apply -f mq-build-config.yaml
```

3. 執行建置以建立映像檔

a) 開始建置

```
oc start-build mymq
```

您應該會看到類似下列內容的輸出:

```
build.build.openshift.io/mymq-1 started
```

b) 檢查建置的狀態
例如，您可以使用前一個步驟所傳回的建置 ID 來執行下列指令:

```
oc describe build mymq-1
```

4. 使用新映像檔來部署佇列管理程式

遵循 第 75 頁的『將佇列管理程式部署至 Red Hat OpenShift Container Platform 叢集』中說明的步驟，將新的自訂映像檔新增至 YAML。

您可以將下列 YAML Snippet 新增至一般 QueueManager YAML ， 其中 *my-namespace* 是您正在使用的 Red Hat OpenShift 專案/名稱空間，而 *image* 是您先前建立的映像檔名稱 (例如 "mymq:latest-amd64"):

```
spec:
  queueManager:
    image: image-registry.openshift-image-registry.svc:5000/my-namespace/my-image
```

相關工作

第 75 頁的『將佇列管理程式部署至 Red Hat OpenShift Container Platform 叢集』
使用 QueueManager 自訂資源，將佇列管理程式部署至 Red Hat OpenShift Container Platform 叢集。

OpenShift CP4I 將自訂註釋和標籤新增至佇列管理程式資源

您可以將自訂註釋和標籤新增至 QueueManager meta 資料。

關於這項作業

自訂註釋和標籤會新增至 PVC 以外的所有資源。 如果自訂註釋或標籤符合現有的索引鍵，則會使用 IBM MQ Operator 所設定的值。

程序

- 新增自訂註釋。

  若要將自訂註釋新增至佇列管理程式資源 (包括 Pod) ， 請在 metadata 下新增註釋。 例如:

```
apiVersion: mq.ibm.com/v1beta1
kind: QueueManager
metadata:
  name: quickstart-cp4i
  annotations:
    annotationKey: "value"
```

- 新增自訂標籤。

  若要將自訂標籤新增至佇列管理程式資源 (包括 Pod) ， 請在 metadata 下新增標籤。 例如:

```
apiVersion: mq.ibm.com/v1beta1
kind: QueueManager
metadata:
  name: quickstart-cp4i
```

```
  labels:
    labelKey: "value"
```

OpenShift CP4I 停用執行時期 Webhook 檢查

執行時期 Webhook 檢查可確保儲存體類別對您的佇列管理程式而言是可行的。 您可以停用它們以增進效能，或因為它們不適用於您的環境。

關於這項作業

執行時期 Webhook 檢查是在佇列管理程式配置上完成。 它們會檢查儲存類別是否適合您選取的佇列管理程式類型。

您可以選擇停用這些檢查，以減少建立佇列管理程式所花費的時間，或因為這些檢查對您的特定環境無效。

註: 停用執行時期 Webhook 檢查之後，容許任何儲存類別值。 這可能會導致佇列管理程式岔斷。

IBM MQ Operator 1.2 中引進了對執行時期檢查的支援。

程序

- 停用執行時期 Webhook 檢查。

  在 `metadata` 下新增下列註釋。 例如:

```
apiVersion: mq.ibm.com/v1beta1
kind: QueueManager
metadata:
  name: quickstart-cp4i
  annotations:
    "com.ibm.cp4i/disable-webhook-runtime-checks" : "true"
```

OpenShift CP4I 使用 IBM MQ Operator 操作 IBM MQ

關於這項作業

程序

- 第 73 頁的『準備 IBM MQ 的 Red Hat OpenShift 專案』.
- 第 75 頁的『將佇列管理程式部署至 Red Hat OpenShift Container Platform 叢集』.

OpenShift CP4I 連接至 Red Hat OpenShift 叢集中部署的 IBM MQ Console

如何連接至已部署至 Red Hat OpenShift Container Platform 叢集之佇列管理程式的 IBM MQ Console 。

關於這項作業

您可以在 Red Hat OpenShift Web 主控台或 IBM Cloud Pak for Integration Platform Navigator 中的 `QueueManager` 詳細資料頁面上找到 IBM MQ Console URL。 或者，可以透過執行下列指令，從 Red Hat OpenShift CLI 中找到它:

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

如果您使用 IBM Cloud Pak for Integration 授權，則會將 IBM MQ Web 主控台配置為使用 IBM Cloud Pak Identity and Access Manager (IAM)。 叢集管理者可能已設定 IAM 元件。 不過，如果這是第一次在 Red Hat OpenShift 叢集上使用 IAM ，則您需要擷取起始管理者密碼。 如需相關資訊，請參閱 取得起始管理者密碼。

如果您使用 IBM MQ 授權，則 MQ Web 主控台未預先配置，您需要自行配置。 如需相關資訊，請參閱 配置使用者和角色。

相關工作

第 98 頁的『配置路徑以從 Red Hat OpenShift 叢集外部連接至佇列管理程式』
您需要 Red Hat OpenShift 路徑，才能從 Red Hat OpenShift 叢集外部將應用程式連接至 IBM MQ 佇列管理程式。 您必須在 IBM MQ 佇列管理程式及用戶端應用程式上啟用 TLS，因為只有在使用 TLS 1.2 或更高版本的通訊協定時，才能在 TLS 通訊協定中使用 SNI。 Red Hat OpenShift Container Platform Router 使用 SNI 將要求遞送至 IBM MQ 佇列管理程式。

OpenShift CP4I 使用 *IBM Cloud Pak IAM* 授與 *IBM MQ Console* 的許可權

IBM MQ Console 的許可權是透過 IBM Cloud Pak Administration Hub 而非 IBM Cloud Pak for Integration Platform Navigator 來管理。 IBM MQ 不會使用 IBM Cloud Pak for Integration 所提供的「自動化」許可權，而是使用 IBM Cloud Pak Identity and Access Manager (IAM) 所啟用的基本許可權。

程序

1. 開啟 IBM Cloud Pak 管理主控台。

   從 IBM Cloud Pak for Integration Platform UI 中，按一下工具列右上角的 Cloud Pak 切換器 (9 點圖示)，然後按一下 **IBM Cloud Pak 管理** 畫面。
2. 在左上角的導覽功能表中，選取 **身分和存取權**，然後選取 **團隊和服務 ID**。
3. 建立團隊，然後將使用者新增至其中。
   a) 選取 **建立團隊**。
   b) 輸入團隊名稱，然後選取您要管理之使用者的安全網域。
   c) 搜尋使用者。
      這些使用者必須已存在於您的身分提供者中。
   d) 當您找到每一個使用者時，請為他們提供一個角色。 這必須是「管理者」或「叢集管理者」，才能使用 IBM MQ Console 來管理 IBM MQ。
4. 將每一個使用者新增至名稱空間。
   a) 選取要編輯的團隊。
   b) 選取 **資源 > 管理資源**。
   c) 選取您要此團隊管理的名稱空間。 這些可以是具有佇列管理程式的任何名稱空間。

OpenShift CP4I 使用 IBM MQ Operator 時監視

IBM MQ Operator 所管理的佇列管理程式可以產生與 Prometheus 相容的度量值。

您可以使用 Red Hat OpenShift Container Platform (OCP) 監視堆疊來檢視這些度量值。 開啟 OCP 中的 **度量** 標籤，然後按一下 **觀察 > 度量**。 依預設會啟用佇列管理程式度量，但可以透過將 **.spec.metrics.enabled** 設為 `false` 來停用。

Prometheus 是度量值的時間序列資料庫及規則評估引擎。 IBM MQ 容器會公開可由 Prometheus 查詢的度量值端點。 這些度量是從 MQ 系統主題產生，用於監視及活動追蹤。

Red Hat OpenShift Container Platform 包括使用 Prometheus 伺服器的預先配置、預先安裝及自行更新監視堆疊。 需要配置 Red Hat OpenShift Container Platform 監視堆疊來監視使用者定義的專案。 如需相關資訊，請參閱 啟用使用者定義專案的監視。 當您建立已啟用度量值的 `QueueManager` 時， IBM MQ Operator 會建立 `ServiceMonitor`，然後 Prometheus 運算子可以探索該度量值。

在舊版 IBM Cloud Pak for Integration 中，您也可以改用 IBM Cloud Platform Monitoring 服務來提供 Prometheus 伺服器。

OpenShift CP4I 使用 *IBM MQ Operator* 時發佈的度量值

佇列管理程式儲存器可以發佈與 Red Hat OpenShift Monitoring 相容的度量值。

度量	類型	說明
ibmmq_qmgr_commit_total	counter	確定計數
ibmmq_qmgr_cpu_load_fifteen_minute_average_percentage	gauge	CPU 負載 - 十五分鐘平均值
ibmmq_qmgr_cpu_load_five_minute_average_percentage	gauge	CPU 負載 - 五分鐘平均值
ibmmq_qmgr_cpu_load_one_minute_average_percentage	gauge	CPU 負載 - 一分鐘平均值
ibmmq_qmgr_destructive_get_bytes_total	counter	間隔總計破壞性取得 - 位元組計數
ibmmq_qmgr_destructive_get_total	counter	間隔總計破壞性取得 - 計數
ibmmq_qmgr_durable_subscription_alter_total	counter	變更可延續訂閱計數
ibmmq_qmgr_durable_subscription_create_total	counter	建立可延續訂閱計數
ibmmq_qmgr_durable_subscription_delete_total	counter	刪除可延續訂閱計數
ibmmq_qmgr_durable_subscription_resume_total	counter	回復可延續訂閱計數
ibmmq_qmgr_errors_file_system_free_space_percentage	gauge	MQ 錯誤檔案系統 - 可用空間
ibmmq_qmgr_errors_file_system_in_use_bytes	gauge	MQ 錯誤檔案系統 - 使用中位元組數
ibmmq_qmgr_expired_message_total	counter	過期訊息計數
ibmmq_qmgr_failed_browse_total	counter	失敗瀏覽計數
ibmmq_qmgr_failed_mqcb_total	counter	失敗 MQCB 計數
ibmmq_qmgr_failed_mqclose_total	counter	失敗 MQCLOSE 計數
ibmmq_qmgr_failed_mqconn_mqconnx_total	counter	失敗 MQCONN/MQCONNX 計數

度量	類型	說明
ibmmq_qmgr_failed_mqget_total	counter	失敗 MQGET - 計數
ibmmq_qmgr_failed_mqinq_total	counter	失敗 MQINQ 計數
ibmmq_qmgr_failed_mqopen_total	counter	失敗 MQOPEN 計數
ibmmq_qmgr_failed_mqput1_total	counter	失敗 MQPUT1 計數
ibmmq_qmgr_failed_mqput_total	counter	失敗 MQPUT 計數
ibmmq_qmgr_failed_mqset_total	counter	失敗 MQSET 計數
ibmmq_qmgr_failed_mqsubrq_total	counter	失敗 MQSUBRQ 計數
ibmmq_qmgr_failed_subscription_create_alter_resume_total	counter	失敗建立/變更/回復訂閱計數
ibmmq_qmgr_failed_subscription_delete_total	counter	訂閱刪除失敗計數
ibmmq_qmgr_failed_topic_mqput_mqput1_total	counter	失敗主題 MQPUT/MQPUT1 計數
ibmmq_qmgr_fdc_files	gauge	MQ FDC 檔案計數
ibmmq_qmgr_log_file_system_in_use_bytes	gauge	日誌檔案系統 - 使用中位元組數
ibmmq_qmgr_log_file_system_max_bytes	gauge	日誌檔案系統 - 位元組數上限
ibmmq_qmgr_log_in_use_bytes	gauge	日誌 - 使用中位元組數
ibmmq_qmgr_log_logical_written_bytes_total	counter	日誌 - 寫入的邏輯位元組數
ibmmq_qmgr_log_max_bytes	gauge	日誌 - 位元組數上限
ibmmq_qmgr_log_physical_written_bytes_total	counter	日誌 - 寫入的實體位元組數
ibmmq_qmgr_log_primary_space_in_use_percentage	gauge	日誌 - 使用中的現行主要空間

度量	類型	說明
ibmmq_qmgr_log_workload_primary_space_utilization_percentage	gauge	日誌 - 工作量主要空間使用率
ibmmq_qmgr_log_write_latency_seconds	gauge	日誌 - 寫入延遲
ibmmq_qmgr_log_write_size_bytes	gauge	日誌 - 寫入大小
ibmmq_qmgr_mqcb_total	counter	MQCB 計數
ibmmq_qmgr_mqclose_total	counter	MQCLOSE 計數
ibmmq_qmgr_mqconn_mqconnx_total	counter	MQCONN/MQCONNX 計數
ibmmq_qmgr_mqctl_total	counter	MQCTL 計數
ibmmq_qmgr_mqdisc_total	counter	MQDISC 計數
ibmmq_qmgr_mqinq_total	counter	MQINQ 計數
ibmmq_qmgr_mqopen_total	counter	MQOPEN 計數
ibmmq_qmgr_mqput_mqput1_bytes_total	counter	間隔總計 MQPUT/MQPUT1 位元組計數
ibmmq_qmgr_mqput_mqput1_total	counter	間隔總計 MQPUT/MQPUT1 計數
ibmmq_qmgr_mqset_total	counter	MQSET 計數
ibmmq_qmgr_mqstat_total	counter	MQSTAT 計數
ibmmq_qmgr_mqsubrq_total	counter	MQSUBRQ 計數
ibmmq_qmgr_non_durable_subscription_create_total	counter	建立不可延續訂閱計數
ibmmq_qmgr_non_durable_subscription_delete_total	counter	刪除不可延續訂閱計數
ibmmq_qmgr_non_persistent_message_browse_bytes_total	counter	非持續訊息瀏覽 - 位元組計數
ibmmq_qmgr_non_persistent_message_browse_total	counter	非持續訊息瀏覽 - 計數

度量	類型	說明
ibmmq_qmgr_non_persistent_message_destructive_get_total	counter	非持續訊息破壞性取得 - 計數
ibmmq_qmgr_non_persistent_message_get_bytes_total	counter	取得非持續訊息 - 位元組計數
ibmmq_qmgr_non_persistent_message_mqput1_total	counter	非持續訊息 MQPUT1 計數
ibmmq_qmgr_non_persistent_message_mqput_total	counter	非持續訊息 MQPUT 計數
ibmmq_qmgr_non_persistent_message_put_bytes_total	counter	放置非持續訊息 - 位元組計數
ibmmq_qmgr_non_persistent_topic_mqput_mqput1_total	counter	非持續 - 主題 MQPUT/MQPUT1 計數
ibmmq_qmgr_persistent_message_browse_bytes_total	counter	持續訊息瀏覽 - 位元組計數
ibmmq_qmgr_persistent_message_browse_total	counter	持續訊息瀏覽 - 計數
ibmmq_qmgr_persistent_message_destructive_get_total	counter	持續訊息破壞性取得 - 計數
ibmmq_qmgr_persistent_message_get_bytes_total	counter	取得持續訊息 - 位元組計數
ibmmq_qmgr_persistent_message_mqput1_total	counter	持續訊息 MQPUT1 計數
ibmmq_qmgr_persistent_message_mqput_total	counter	持續訊息 MQPUT 計數
ibmmq_qmgr_persistent_message_put_bytes_total	counter	放置持續訊息 - 位元組計數
ibmmq_qmgr_persistent_topic_mqput_mqput1_total	counter	持續 - 主題 MQPUT/MQPUT1 計數
ibmmq_qmgr_published_to_subscribers_bytes_total	counter	已發佈至訂閱者 - 位元組計數

度量	類型	說明
ibmmq_qmgr_published_to_subscribers_message_total	counter	已發佈至訂閱者 - 訊息計數
ibmmq_qmgr_purged_queue_total	counter	已清除佇列計數
ibmmq_qmgr_queue_manager_file_system_free_space_percentage	gauge	佇列管理程式檔案系統 - 可用空間
ibmmq_qmgr_queue_manager_file_system_in_use_bytes	gauge	佇列管理程式檔案系統 - 使用中位元組數
ibmmq_qmgr_ram_free_percentage	gauge	RAM 可用百分比
ibmmq_qmgr_ram_usage_estimate_for_queue_manager_bytes	gauge	RAM 位元組數總計 - 佇列管理程式的估計值
ibmmq_qmgr_rollback_total	counter	回復計數
ibmmq_qmgr_system_cpu_time_estimate_for_queue_manager_percentage	gauge	系統 CPU 時間 - 佇列管理程式的百分比估計值
ibmmq_qmgr_system_cpu_time_percentage	gauge	系統 CPU 時間百分比
ibmmq_qmgr_topic_mqput_mqput1_total	counter	主題 MQPUT/MQPUT1 間隔總計
ibmmq_qmgr_topic_put_bytes_total	counter	間隔總計主題放置位元組數
ibmmq_qmgr_trace_file_system_free_space_percentage	gauge	MQ 追蹤檔案系統 - 可用空間
ibmmq_qmgr_trace_file_system_in_use_bytes	gauge	MQ 追蹤檔案系統 - 使用中位元組數
ibmmq_qmgr_user_cpu_time_estimate_for_queue_manager_percentage	gauge	使用者 CPU 時間 - 佇列管理程式的百分比估計值
ibmmq_qmgr_user_cpu_time_percentage	gauge	使用者 CPU 時間百分比

CP4I CD V9.2.2 檢視 IBM MQ 認證儲存器的原生 HA 佇列管理程式狀態

對於 IBM MQ 認證容器，您可以透過在其中一個執行中 Pod 內執行 **dspmq** 指令，來檢視原生 HA 實例的狀態。

關於這項作業

重要:

您可以在其中一個執行中 Pod 中使用 **dspmq** 指令，以檢視佇列管理程式實例的作業狀態。 傳回的資訊視實例是作用中還是抄本而定。 作用中實例所提供的資訊是明確的，抄本節點的資訊可能已過期。

您可以執行下列動作:

- 檢視現行節點上的佇列管理程式實例是作用中還是抄本。
- 檢視現行節點上實例的原生 HA 作業狀態。
- 檢視原生 HA 配置中所有三個實例的作業狀態。

下列狀態欄位用來報告原生 HA 配置狀態:

角色
指定實例的現行角色，並且是 `Active`、 `Replica` 或 `Unknown` 之一。

實例
使用 **crtmqm** 指令的 **-lr** 選項建立此佇列管理程式實例時，為其提供的名稱。

INSYNC
指出實例是否可以在必要時接管作為作用中實例。

仲裁
以 *number_of_instances_in-sync*/*number_of_instances_configured* 格式報告仲裁狀態。

REPLADDR
佇列管理程式實例的抄寫位址。

CONNACTV
指出節點是否連接至作用中實例。

BACKLOG
指出實例落後的 KB 數。

CONNINST
指出指定的實例是否連接至此實例。

ALTDATE
指出前次更新此資訊的日期 (如果從未更新過，則為空白)。

ALTTIME
指出前次更新此資訊的時間 (如果從未更新過，則為空白)。

程序

- 尋找屬於佇列管理程式的 Pod。

  ```
  oc get pod --selector app.kubernetes.io/instance=nativeha-qm
  ```

- 在其中一個 Pod 中執行 `dspmq`

  ```
  oc exec -t Pod dspmq
  ```

  ```
  oc rsh Pod
  ```

  適用於互動式 Shell ，您可以在其中直接執行 `dspmq` 。
- 若要判定佇列管理程式實例是作為作用中實例還是抄本執行，請執行下列動作:

```
oc exec -t Pod dspmq -o status -m QMgrName
```

名為 BOB 之佇列管理程式的作用中實例會報告下列狀態:

```
QMNAME(BOB)                 STATUS(Running)
```

名為 BOB 之佇列管理程式的抄本實例會報告下列狀態:

```
QMNAME(BOB)                 STATUS(Replica)
```

非作用中實例會報告下列狀態:

```
QMNAME(BOB)                 STATUS(Ended Immediately)
```

- 若要判定所指定 Pod 中實例的原生 HA 作業狀態，請執行下列動作:

```
oc exec -t Pod dspmq -o nativeha -m QMgrName
```

名為 BOB 之佇列管理程式的作用中實例可能報告下列狀態:

```
QMNAME(BOB)               ROLE(Active) INSTANCE(inst1) INSYNC(Yes) QUORUM(3/3)
```

佇列管理程式 BOB 的抄本實例可能會報告下列狀態:

```
QMNAME(BOB)               ROLE(Replica) INSTANCE(inst2) INSYNC(Yes) QUORUM(2/3)
```

名為 BOB 之佇列管理程式的非作用中實例可能會報告下列狀態:

```
QMNAME(BOB)               ROLE(Unknown) INSTANCE(inst3) INSYNC(no) QUORUM(0/3)
```

- 若要判定「原生 HA」配置中所有實例的「原生 HA」作業狀態，請執行下列動作:

```
oc exec -t Pod dspmq -o nativeha -x -m QMgrName
```

如果您在執行佇列管理程式 BOB 作用中實例的節點上發出此指令，則可能會收到下列狀態:

```
QMNAME(BOB)            ROLE(Active) INSTANCE(inst1) INSYNC(Yes) QUORUM(3/3)
 INSTANCE(inst1) ROLE(Active)  REPLADDR(9.20.123.45) CONNACTV(Yes) INSYNC(Yes) BACKLOG(0)
CONNINST(Yes) ALTDATE(2022-01-12) ALTTIME(12.03.44)
 INSTANCE(inst2) ROLE(Replica) REPLADDR(9.20.123.46) CONNACTV(Yes) INSYNC(Yes) BACKLOG(0)
CONNINST(Yes) ALTDATE(2022-01-12) ALTTIME(12.03.44)
 INSTANCE(inst3) ROLE(Replica) REPLADDR(9.20.123.47) CONNACTV(Yes) INSYNC(Yes) BACKLOG(0)
CONNINST(Yes) ALTDATE(2022-01-12) ALTTIME(12.03.44)
```

如果您在執行佇列管理程式 BOB 抄本實例的節點上發出此指令，則可能會收到下列狀態，指出其中一個抄本落後:

```
QMNAME(BOB)            ROLE(Replica) INSTANCE(inst2) INSYNC(Yes) QUORUM(2/3)
 INSTANCE(inst2) ROLE(Replica) REPLADDR(9.20.123.46) CONNACTV(Yes) INSYNC(Yes) BACKLOG(0)
CONNINST(Yes) ALTDATE(2022-01-12) ALTTIME(12.03.44)
 INSTANCE(inst1) ROLE(Active)  REPLADDR(9.20.123.45) CONNACTV(Yes) INSYNC(Yes) BACKLOG(0)
CONNINST(Yes) ALTDATE(2022-01-12) ALTTIME(12.03.44)
 INSTANCE(inst3) ROLE(Replica) REPLADDR(9.20.123.47) CONNACTV(Yes) INSYNC(No)  BACKLOG(435)
CONNINST(Yes) ALTDATE(2022-01-12) ALTTIME(12.03.44)
```

如果您在執行佇列管理程式 BOB 非作用中實例的節點上發出此指令，則可能會收到下列狀態:

```
QMNAME(BOB)            ROLE(Unknown) INSTANCE(inst3) INSYNC(no) QUORUM(0/3)
 INSTANCE(inst1) ROLE(Unknown) REPLADDR(9.20.123.45) CONNACTV(Unknown) INSYNC(Unknown)
BACKLOG(Unknown) CONNINST(No) ALTDATE() ALTTIME()
 INSTANCE(inst2) ROLE(Unknown) REPLADDR(9.20.123.46) CONNACTV(Unknown) INSYNC(Unknown)
BACKLOG(Unknown) CONNINST(No) ALTDATE() ALTTIME()
 INSTANCE(inst3) ROLE(Unknown) REPLADDR(9.20.123.47) CONNACTV(No) INSYNC(Unknown)
BACKLOG(Unknown) CONNINST(No) ALTDATE() ALTTIME()
```

如果您在實例仍在協議作用中及抄本時發出指令，則會收到下列狀態:

```
QMNAME(BOB)               STATUS(Negotiating)
```

相關參考

dspmq (顯示佇列管理程式) 指令

第 86 頁的『範例: 配置原生 HA 佇列管理程式』

此範例顯示如何使用 IBM MQ Operator，將使用原生高可用性特性的佇列管理程式部署至 Red Hat OpenShift Container Platform (OCP)。

OpenShift CP4I 使用 Red Hat OpenShift CLI 來備份及還原佇列管理程式配置

如果佇列管理程式配置遺失，備份佇列管理程式配置可協助您從其定義重建佇列管理程式。 此程序不會備份佇列管理程式日誌資料。 由於訊息的暫時性，在還原時，歷程日誌資料可能不相關。

開始之前

使用 **cloudctl login** (適用於 IBM Cloud Pak for Integration) 或 **oc login** 登入叢集。

程序

- 備份佇列管理程式配置。

  您可以使用 **dmpmqcfg** 指令來傾出 IBM MQ 佇列管理程式的配置。

  a) 取得佇列管理程式的 Pod 名稱。
  例如，您可以執行下列指令，其中 *queue_manager_name* 是 QueueManager 資源的名稱:

  ```
  oc get pods --selector app.kubernetes.io/name=ibm-mq,app.kubernetes.io/
  instance=queue_manager_name
  ```

  b) 在 Pod 上執行 **dmpmqcfg** 指令，將輸出導向本端機器上的檔案。

  dmpmqcfg 會輸出佇列管理程式的 MQSC 配置。

  ```
  oc exec -it pod_name -- dmpmqcfg > backup.mqsc
  ```

- 還原佇列管理程式配置。

  遵循前一個步驟所概述的備份程序之後，您應該有一個包含佇列管理程式配置的 backup.mqsc 檔。 您可以將此檔案套用至新的佇列管理程式來還原配置。

  a) 取得佇列管理程式的 Pod 名稱。
  例如，您可以執行下列指令，其中 *queue_manager_name* 是 QueueManager 資源的名稱:

  ```
  oc get pods --selector app.kubernetes.io/name=ibm-mq,app.kubernetes.io/
  instance=queue_manager_name
  ```

  b) 在 Pod 上執行 **runmqsc** 指令，並在 backup.mqsc 檔案的內容中導向。

  ```
  oc exec -i pod_name -- runmqsc < backup.mqsc
  ```

OpenShift CP4I 對 IBM MQ Operator 的問題進行疑難排解

如果您對 IBM MQ Operator 有問題，請使用說明的技術來協助您診斷及解決它們。

程序

- 第 114 頁的『疑難排解: 取得佇列管理程式資料的存取權』

OpenShift CP4I 疑難排解: 取得佇列管理程式資料的存取權

使用 PVC Inspector 工具來存取佇列管理程式 PVC 上的檔案，其中無法建立佇列管理程式 Pod 的遠端 Shell。 這可能是因為 Pod 處於 **Error** 或 **CrashLoopBackOff** 狀態。 此工具設計用於與 IBM MQ Operator 所部署的佇列管理程式搭配使用。

開始之前

使用 PVC Inspector 工具。 您必須具有佇列管理程式名稱空間的存取權。

關於這項作業

為了協助疑難排解，您可以存取儲存在與給定佇列管理程式相關聯之「持續性磁區要求 (PVC)」上的資料。 若要這樣做，您可以使用工具將 PVC 裝載至一組 Inspector Pod。 然後，您可以將遠端 Shell 放入任何 Inspector Pod 中，以讀取檔案。

視部署類型而定，會在一到三個 Inspector Pod 之間建立。 在相關聯的 PVC 檢查程式 Pod 上，提供 Native-HA 或 Multi-Instance 佇列管理程式的給定 Pod 特定的磁區。 共用磁區在所有檢查程式上都可用。 Inspector Pod 的名稱包含相關聯佇列管理程式 Pod 的名稱。

程序

1. 下載 MQ PVC 檢查程式工具。

   這裡提供此工具: https://github.com/ibm-messaging/mq-pvc-tool。
2. 請確定您已登入叢集。
3. 找出佇列管理程式的名稱，以及佇列管理程式執行所在的名稱空間。
4. 針對佇列管理程式執行檢查程式工具。
   a) 執行下列指令，並指定佇列管理程式名稱及其名稱空間名稱。

   ```
   ./pvc-tool.sh queue_manager_name queue_manager_namespace_name
   ```

   b) 工具完成之後，請執行下列指令，以檢視正在建立的 Inspector Pod。

   ```
   oc get pods
   ```
5. 檢視裝載至 Inspector Pod 的檔案。
   a) 每一個 PVC Inspector Pod 都與佇列管理程式 Pod 相關聯，因此可能有多個 Inspector Pod。 執行下列指令，以存取其中一個 Pod:

   ```
   oc rsh pvc-inspector-pod-name
   ```

   您會被放置在包含已裝載 PVC 目錄的目錄中。
   b) 透過執行下列指令，在 Pod 中開啟遠端 Shell:

   ```
   ls
   ```

   c) 您可以看到與已裝載 PVC 同名的目錄。 透過瀏覽這些目錄來存取佇列管理程式 PVC 上的檔案。 若要查看 PVC 清單，請在遠端 Shell 階段作業外部執行下列指令:

   ```
   oc get pvc
   ```

   d) 執行下列指令，以清除工具所建立的 Pod:

   ```
   'oc delete pods -l tool=mq-pvc-inspector
   ```

OpenShift CP4I IBM MQ Operator 的 API 參考資料

IBM MQ 提供 Kubernetes 操作器，可提供與 Red Hat OpenShift Container Platform 的原生整合。

OpenShift CP4I mq.ibm.com/v1beta1 的 API 參考資料

v1beta1 API 可用來建立及管理 QueueManager 資源。

OpenShift CP4I CD EUS *mq.ibm.com/v1beta1* 的授權參考手冊

現行授權版本

spec.license.license 欄位必須包含您接受之授權的授權 ID。 有效值如下：

下者的值：**spec.license.license**	下者的值：**spec.license.use**	授權資訊	適用的 **IBM MQ** 版本
L-RJON-C7QG3S	Production 或 NonProduction	IBM Cloud Pak for Integration 2021.4.1	9.2.4 或 9.2.5
L-RJON-C7QFZX	Production 或 NonProduction	IBM Cloud Pak for Integration Limited Edition 2021.4.1	9.2.4 或 9.2.5
L-RJON-C5CSNH	Production 或 NonProduction	IBM Cloud Pak for Integration 2021.3.1	9.2.3 或 9.2.4
L-RJON-C5CSM2	Production 或 NonProduction	IBM Cloud Pak for Integration Limited Edition 2021.3.1	9.2.3 或 9.2.4
L-RJON-BZFQU2	Production 或 NonProduction	IBM Cloud Pak for Integration 2021.2.1	9.2.3
L-RJON-BZFQSB	Production 或 NonProduction	IBM Cloud Pak for Integration Limited Edition 2021.2.1	9.2.3
L-RJON-BUVMQX	Production 或 NonProduction	IBM Cloud Pak for Integration 2020.4.1	9.2.0 EUS 或 9.2.1
L-RJON-BUVMYB	Production 或 NonProduction	IBM Cloud Pak for Integration Limited Edition 2020.4.1	9.2.0 EUS 或 9.2.1
L-APIG-BZDDDY	Production	IBM MQ Advanced 及 IBM MQ Advanced for Non-Production Environment 9.2 -07/2021	9.2.3、 9.2.4 或 9.2.5
L-APIG-BYHCL7	Development	IBM MQ Advanced for Developers (非 Warranted) V9.2 -07/2021	9.2.3、 9.2.4 或 9.2.5
L-APIG-BVJJB3	Production	IBM MQ Advanced 及 IBM MQ Advanced for Non-Production Environment 9.2 -03/2021	9.2.2
L-APIG-BMJJBM	Production	IBM MQ Advanced V9.2	9.2.0 CD 或 9.2.1
L-APIG-BMKG5H	Development	IBM MQ Advanced for Developers (非 Warranted) V9.2	9.2.0 CD、 9.2.1 或 9.2.2

請注意，已指定授權 *version* ，不一定與 IBM MQ 的版本相同。

較舊的授權版本

spec.license.license 欄位必須包含您接受之授權的授權 ID。 有效值如下：

下者的值: **spec.license.license**	下者的值: **spec.license.use**	授權資訊	適用的 IBM MQ 版本
L-RJON-BXUPZ2	Production 或 NonProduction	IBM Cloud Pak for Integration 2021.1.1	9.2.2
L-RJON-BXUQ34	Production 或 NonProduction	IBM Cloud Pak for Integration Limited Edition 2021.1.1	9.2.2
L-RJON-BYRMYW	NonProduction	IBM Cloud Pak for Integration Eval-Demo 2021.1.1。 與 原生 HA 搭配使用的舊版 (僅含 IBM MQ Operator 1.5)。	9.2.2
L-RJON-BQPGWD	Production 或 NonProduction	IBM Cloud Pak for Integration 2020.3.1	9.2.0 CD
L-RJON-BN7PN3	Production 或 NonProduction	IBM Cloud Pak for Integration 2020.2.1	9.1.5 或 9.2.0 CD
L-RJON-BPHL2Y	Production 或 NonProduction	IBM Cloud Pak for Integration Limited Edition 2020.2.1	9.1.5
L-APIG-BJAKBF	Production	IBM MQ Advanced V9.1 -04/2020	9.1.5
L-APIG-BM7GDH	Development	IBM MQ Advanced for Developers (非 Warranted) V9.1 -04/2020	9.1.5

請注意，已指定授權 *version* ，不一定與 IBM MQ 的版本相同。

OpenShift CP4I

QueueManager 的 API 參考資料 (mq.ibm.com/v1beta1)

QueueManager

QueueManager 是 IBM MQ 伺服器，可為應用程式提供佇列作業及發佈/訂閱服務。

欄位	說明
apiVersion 字串	APIVersion 定義這個物件表示法的版本化綱目。 伺服器應該將可辨識的綱目轉換成最新的內部值，且可能會拒絕無法辨識的值。 進一步資訊: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#resources。
kind 字串	Kind 是一個字串值，代表此物件所代表的 REST 資源。 伺服器可能會從用戶端向其提交要求的端點推斷此情況。 無法更新。 在 CamelCase 中。 進一步資訊: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-tind。
metadata	
spec QueueManager 規格	所需的 QueueManager 狀態。
status QueueManager 狀態	QueueManager 的觀察狀態。

.spec

所需的 QueueManager 狀態。

出現在:

- 第 116 頁的『QueueManager』

欄位	說明
affinity	標準 Kubernetes 親緣性規則。 如需相關資訊，請參閱 https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.17/#affinity-v1-core。
annotations 註釋	註釋欄位用作 Pod 註釋的透通。 使用者可以將任何註釋新增至此欄位，並讓它套用至 Pod。 這裡的註釋會改寫預設註釋 (如果有提供的話)。 需要 MQ Operator 1.3.0 或更高版本。
imagePullSecrets LocalObjectReference 陣列	相同名稱空間中密鑰的選用參照清單，用於取回此 QueueManager 所使用的任何映像檔。 如果指定的話，這些密鑰會傳遞給個別的拉取器實作，以供它們使用。 例如，如果是 Docker，則只允許使用 DockerConfig 類型密碼。 如需相關資訊，請參閱 https://kubernetes.io/docs/concepts/containers/images#specifying-imagepullsecrets-on-a-pod。
labels 標籤	標籤欄位用作 Pod 標籤的透通。 使用者可以將任何標籤新增至此欄位，並讓它套用至 Pod。 這裡的標籤會改寫預設標籤 (如果有提供的話)。 需要 MQ Operator 1.3.0 或更高版本。
license 授權	這些設定可控制您是否接受授權，以及要使用哪些授權標準。
pki PKI	公開金鑰基礎架構設定，用於定義與傳輸層安全 (TLS) 或 MQ Advanced Message Security (AMS) 搭配使用的金鑰和憑證。
queueManager QueueManager 配置	「佇列管理程式」儲存器及基礎「佇列管理程式」的設定。
securityContext SecurityContext	要新增至佇列管理程式 Pod 的 securityContext 的安全設定。
template 範本	Kubernetes 資源的進階範本。 範本可讓使用者置換 IBM MQ 如何產生基礎 Kubernetes 資源，例如 StatefulSet、Pod 及服務。 這僅適用於進階使用者，因為如果使用不正確，可能會中斷 MQ 的正常作業。 範本中的設定將置換 QueueManager 資源中的任何其他指定值。
terminationGracePeriodSeconds 整數	Pod 需要溫和終止的選用持續時間 (以秒為單位)。 值必須是非負整數。 值零表示立即刪除。 嘗試結束佇列管理程式的目標時間，會提升應用程式斷線的階段。 必要的話，會岔斷必要的佇列管理程式維護作業。 預設值為 30 秒。
tracing TracingConfig	與 Cloud Pak for Integration 作業儀表板進行追蹤整合的設定。
version 字串	控制將使用之 MQ 版本的設定 (必要)。 例如: 9.1.5.0-r2 會使用儲存器映像檔的第二個修訂來指定 MQ 9.1.5.0 版。 通常會在修訂中套用儲存器特定修正程式，例如基本映像檔的修正程式。
web WebServer 配置	MQ Web 伺服器的設定。

.spec.annotations

註釋欄位用作 Pod 註釋的透通。 使用者可以將任何註釋新增至此欄位，並讓它套用至 Pod。 這裡的註釋會改寫預設註釋 (如果有提供的話)。 需要 MQ Operator 1.3.0 或更高版本。

出現在:

- 第 116 頁的『.spec』

.spec.imagePullSecrets

LocalObjectReference 包含足夠資訊，可讓您在相同名稱空間內找到所參照的物件。

出現在:

- 第 116 頁的『.spec』

欄位	說明
name 字串	參照的名稱。 進一步資訊: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names 待辦事項: 新增其他有用欄位。 apiVersion， kind， uid?。

.spec.labels

標籤欄位用作 Pod 標籤的透通。 使用者可以將任何標籤新增至此欄位，並讓它套用至 Pod。 這裡的標籤會改寫預設標籤 (如果有提供的話)。 需要 MQ Operator 1.3.0 或更高版本。

出現在:

- 第 116 頁的『.spec』

.spec.license

這些設定可控制您是否接受授權，以及要使用哪些授權標準。

出現在:

- 第 116 頁的『.spec』

欄位	說明
accept 布林值	您是否接受與此軟體相關聯的授權 (必要)。
license 字串	您接受的授權 ID。 這必須是您使用之 MQ 版本的正確授權 ID。 如需有效值，請參閱 http://ibm.biz/BdqvCF 。
metric 字串	指定要使用的授權標準的設定。 例如， ProcessorValueUnit、VirtualProcessorCore 或 ManagedVirtualServer。 使用 MQ 授權時預設為 ProcessorValueUnit ， 使用 Cloud Pak for Integration 授權時預設為 VirtualProcessorCore 。
use 字串	控制軟體使用方式的設定，其中授權支援多種用途。 如需有效值，請參閱 http://ibm.biz/BdqvCF 。

.spec.pki

公開金鑰基礎架構設定，用於定義與傳輸層安全 (TLS) 或 MQ Advanced Message Security (AMS) 搭配使用的金鑰和憑證。

出現在:

- 第 116 頁的『.spec』

欄位	說明
keys PKISource 陣列	要新增至佇列管理程式金鑰儲存庫的私密金鑰。
trust PKISource 陣列	要新增至佇列管理程式金鑰儲存庫的憑證。

.spec.pki.keys

PKISource 定義「公開金鑰基礎架構」資訊的來源，例如金鑰或憑證。

出現在:

- 第 118 頁的『.spec.pki』

欄位	說明
name 字串	名稱用作金鑰或憑證的標籤。 必須是小寫英數字串。

欄位	說明
secret 密鑰	使用 Kubernetes 密鑰來提供金鑰。

.spec.pki.keys.secret

使用 Kubernetes 密鑰來提供金鑰。

出現在:

- 第 118 頁的『.spec.pki.keys』

欄位	說明
items 陣列	Kubernetes 密鑰內的金鑰，應該新增至「佇列管理程式」儲存器。
secretName 字串	Kubernetes 密鑰的名稱。

.spec.pki.trust

PKISource 定義「公開金鑰基礎架構」資訊的來源，例如金鑰或憑證。

出現在:

- 第 118 頁的『.spec.pki』

欄位	說明
name 字串	名稱用作金鑰或憑證的標籤。 必須是小寫英數字串。
secret 密鑰	使用 Kubernetes 密鑰來提供金鑰。

.spec.pki.trust.secret

使用 Kubernetes 密鑰來提供金鑰。

出現在:

- 第 119 頁的『.spec.pki.trust』

欄位	說明
items 陣列	Kubernetes 密鑰內的金鑰，應該新增至「佇列管理程式」儲存器。
secretName 字串	Kubernetes 密鑰的名稱。

.spec.queueManager

「佇列管理程式」儲存器及基礎「佇列管理程式」的設定。

出現在:

- 第 116 頁的『.spec』

欄位	說明
availability 可用性	佇列管理程式的可用性設定，例如是否使用主動-待命配對或原生高可用性。
debug 布林值	是否將來自儲存器特定程式碼的除錯訊息記載至儲存器日誌。 預設為 false。
image 字串	將使用的儲存器映像檔。
imagePullPolicy 字串	控制 kubelet 何時嘗試取回指定映像檔的設定。 預設為 IfNotPresent。
ini INISource 陣列	為佇列管理程式提供 INI 的設定。 需要 MQ Operator 1.1.0 或更高版本。

欄位	說明
livenessProbe QueueManagerLivenessProbe	控制存活性探測的設定。
logFormat 字串	要用於此儲存器的日誌格式。 對於儲存器中 JSON 格式的日誌，請使用 JSON 。 對於文字格式的訊息，請使用 Basic 。 預設為 Basic。
metrics QueueManager 度量	Prometheus 樣式度量值的設定。
mqsc MQSCSource 陣列	為佇列管理程式提供 MQSC 的設定。 需要 MQ Operator 1.1.0 或更高版本。
name 字串	基礎「 MQ 佇列管理程式」的名稱 (如果不同於 metadata.name)。 如果您想要的「佇列管理程式」名稱不符合名稱的 Kubernetes 規則 (例如，包含大寫字母的名稱) ，請使用此欄位。
readinessProbe QueueManagerReadinessProbe	控制就緒性探測的設定。
resources 資源	控制資源需求的設定。
route 路徑	佇列管理程式路徑的設定。 需要 MQ Operator 1.4.0 或更高版本。
startupProbe StartupProbe	控制啟動探測的設定。 僅適用於 MultiInstance 及 NativeHA 部署。 需要 MQ Operator 1.5.0 或更高版本。
storage QueueManager 儲存體	儲存體設定，用於控制「佇列管理程式」對持續性磁區及儲存類別的使用。

.spec.queueManager.availability

佇列管理程式的可用性設定，例如是否使用主動-待命配對或原生高可用性。

出現在:

- 第 119 頁的『.spec.queueManager』

欄位	說明
tls TLS	用於在 NativeHA 抄本之間配置安全通訊的選用 TLS 設定。 需要 MQ Operator 1.5.0 或更高版本。
type 字串	要使用的可用性類型。 針對單一 Pod 使用 SingleInstance ， Kubernetes 會自動重新啟動 (在某些情況下)。 針對 Pod 配對使用 MultiInstance ，其中一個是 active 佇列管理程式，另一個是待命。 將 NativeHA 用於原生高可用性抄寫 (需要 MQ Operator 1.5.0 或更高版本)。 預設為 SingleInstance。 如需詳細資料，請參閱 http://ibm.biz/BdqAQa 。
updateStrategy 字串	用於 MultiInstance 及 NativeHA 佇列管理程式的更新策略。 每當佇列管理程式配置變更時，請使用 RollingUpdate 來啟用自動漸進式更新。 使用 OnDelete 來停用自動漸進式更新，只有在刪除 Pod (包括由外部因素觸發的 Pod 刪除) 時，才會套用「佇列管理程式」變更。 預設為 RollingUpdate。 需要 MQ Operator 1.6.0 或更高版本。

.spec.queueManager.availability.tls

用於在 NativeHA 抄本之間配置安全通訊的選用 TLS 設定。 需要 MQ Operator 1.5.0 或更高版本。

出現在:

- 第 120 頁的『.spec.queueManager.availability』

欄位	說明
cipherSpec 字串	NativeHA TLS 的 CipherSpec 名稱。
secretName 字串	Kubernetes 密鑰的名稱。

.spec.queueManager.ini

INI 配置檔的來源。

出現在:

- 第 119 頁的『.spec.queueManager』

欄位	說明
configMap ConfigMapINISource	ConfigMap 代表包含 INI 資訊的 Kubernetes ConfigMap 。
secret SecretINISource	密鑰代表包含 INI 資訊的 Kubernetes 密鑰。

.spec.queueManager.ini.configMap

ConfigMap 代表包含 INI 資訊的 Kubernetes ConfigMap 。

出現在:

- 第 121 頁的『.spec.queueManager.ini』

欄位	說明
items 陣列	Kubernetes 來源內應該套用的金鑰。
name 字串	Kubernetes 來源的名稱。

.spec.queueManager.ini.secret

密鑰代表包含 INI 資訊的 Kubernetes 密鑰。

出現在:

- 第 121 頁的『.spec.queueManager.ini』

欄位	說明
items 陣列	Kubernetes 來源內應該套用的金鑰。
name 字串	Kubernetes 來源的名稱。

.spec.queueManager.livenessProbe

控制存活性探測的設定。

出現在:

- 第 119 頁的『.spec.queueManager』

欄位	說明
failureThreshold 整數	在成功之後，將探測視為失敗的連續失敗次數下限。 預設值為 1。
initialDelaySeconds 整數	在啟動儲存器之後，起始探測之前的秒數。 SingleInstance 預設為 90 秒。 對於 MultiInstance 及 NativeHA 部署，預設為 0 秒。 進一步資訊: https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle#container-probe。

欄位	說明
periodSeconds 整數	執行探測的頻率（以秒為單位）。 預設值為 10 秒。
successThreshold 整數	在失敗之後將探測視為成功的連續成功數下限。 預設值為 1。
timeoutSeconds 整數	探測逾時之前經歷的秒數。 預設值為 5 秒。 進一步資訊: https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle#container-probe。

.spec.queueManager.metrics

Prometheus 樣式度量值的設定。

出現在:

- 第 119 頁的『.spec.queueManager』

欄位	說明
enabled 布林值	是否啟用 Prometheus 相容度量值的端點。 預設為 true。

.spec.queueManager.mqsc

MQSC 配置檔的來源。

出現在:

- 第 119 頁的『.spec.queueManager』

欄位	說明
configMap ConfigMapMQSCSource	ConfigMap 代表包含 MQSC 資訊的 Kubernetes ConfigMap 。
secret SecretMQSCSource	Secret 代表包含 MQSC 資訊的 Kubernetes 「密鑰」。

.spec.queueManager.mqsc.configMap

ConfigMap 代表包含 MQSC 資訊的 Kubernetes ConfigMap 。

出現在:

- 第 122 頁的『.spec.queueManager.mqsc』

欄位	說明
items 陣列	Kubernetes 來源內應該套用的金鑰。
name 字串	Kubernetes 來源的名稱。

.spec.queueManager.mqsc.secret

Secret 代表包含 MQSC 資訊的 Kubernetes 「密鑰」。

出現在:

- 第 122 頁的『.spec.queueManager.mqsc』

欄位	說明
items 陣列	Kubernetes 來源內應該套用的金鑰。
name 字串	Kubernetes 來源的名稱。

.spec.queueManager.readinessProbe

控制就緒性探測的設定。

出現在:

- 第 119 頁的『.spec.queueManager』

欄位	說明
failureThreshold 整數	在成功之後，將探測視為失敗的連續失敗次數下限。 預設值為 1。
initialDelaySeconds 整數	在啟動儲存器之後，起始探測之前的秒數。 SingleInstance 預設為 10 秒。 對於 MultiInstance 和 NativeHA 部署，預設值為 0。 進一步資訊: https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle#container-probe。
periodSeconds 整數	執行探測的頻率（以秒為單位）。 預設值為 5 秒。
successThreshold 整數	在失敗之後將探測視為成功的連續成功數下限。 預設值為 1。
timeoutSeconds 整數	探測逾時之前經歷的秒數。 預設值為 3 秒。 進一步資訊: https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle#container-probe。

.spec.queueManager.resources

控制資源需求的設定。

出現在:

- 第 119 頁的『.spec.queueManager』

欄位	說明
limits 限制	CPU 及記憶體設定。
requests 要求	CPU 及記憶體設定。

.spec.queueManager.resources.limits

CPU 及記憶體設定。

出現在:

- 第 123 頁的『.spec.queueManager.resources』

欄位	說明
cpu	
memory	

.spec.queueManager.resources.requests

CPU 及記憶體設定。

出現在:

- 第 123 頁的『.spec.queueManager.resources』

欄位	說明
cpu	
memory	

.spec.queueManager.route

佇列管理程式路徑的設定。 需要 MQ Operator 1.4.0 或更高版本。

出現在:

- 第 119 頁的『.spec.queueManager』

欄位	說明
enabled 布林值	是否啟用路徑。 預設為 true。

.spec.queueManager.startupProbe

控制啟動探測的設定。 僅適用於 MultiInstance 及 NativeHA 部署。 需要 MQ Operator 1.5.0 或更高版本。

出現在:

- 第 119 頁的『.spec.queueManager』

欄位	說明
failureThreshold 整數	將探測視為失敗的連續失敗次數下限。 預設為 60。
initialDelaySeconds 整數	在啟動儲存器之後，起始探測之前的秒數。 預設值為 0 秒。 進一步資訊: https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle#container-probe。
periodSeconds 整數	執行探測的頻率（以秒為單位）。 預設值為 5 秒。
successThreshold 整數	將探測視為成功的連續成功數下限。 預設值為 1。
timeoutSeconds 整數	探測逾時之前經歷的秒數。 預設值為 5 秒。 進一步資訊: https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle#container-probe。

.spec.queueManager.storage

儲存體設定，用於控制「佇列管理程式」對持續性磁區及儲存類別的使用。

出現在:

- 第 119 頁的『.spec.queueManager』

欄位	說明
defaultClass 字串	依預設套用至此佇列管理程式的所有持續性磁區的儲存類別。 特定持續性磁區可以定義自己的儲存類別，以置換此預設儲存類別設定。 如果 type of availability 是 SingleInstance 或 NativeHA，則儲存類別可以是 ReadWriteOnce 或 ReadWriteMany 類型。 如果 type of availability 是 MultiInstance，則儲存類別必須是 ReadWriteMany 類型。
defaultDeleteClaim 布林值	刪除佇列管理程式時是否應該刪除所有磁區。 特定持續性磁區可以為 deleteClaim 定義自己的值，這將置換此 defaultDeleteClaim 設定。 預設為 false。
persistedData QueueManagerOptionalVolume	MQ 持續保存資料 (包括配置、佇列及訊息) 的 PersistentVolume 詳細資料。 使用多重實例佇列管理程式時需要。
queueManager QueueManager 磁區	通常位於 /var/mqm 下之任何資料的預設 PersistentVolume 。 如果未指定其他磁區，則將包含所有持續保存的資料及回復日誌。
recoveryLogs QueueManagerOptionalVolume	MQ 回復日誌的持續性磁區詳細資料。 使用多重實例佇列管理程式時需要。

.spec.queueManager.storage.persistedData

MQ 持續保存資料 (包括配置、佇列及訊息) 的 PersistentVolume 詳細資料。 使用多重實例佇列管理程式時需要。

出現在:

- 第 124 頁的『.spec.queueManager.storage』

欄位	說明
class 字串	要用於此磁區的儲存類別。 僅當 type 為 persistent-claim 時才有效。 如果 type of availability 是 SingleInstance 或 NativeHA，則儲存類別可以是 ReadWriteOnce 或 ReadWriteMany 類型。 如果 type of availability 是 MultiInstance，則儲存類別必須是 ReadWriteMany 類型。
deleteClaim 布林值	刪除佇列管理程式時是否應刪除此磁區。
enabled 布林值	此磁區是否應該啟用為個別磁區，或放置在預設 queueManager 磁區上。 預設為 false。
size 字串	要傳遞至 Kubernetes(包括 SI 單位) 的 PersistentVolume 大小。 僅當 type 為 persistent-claim 時才有效。 例如，2Gi。 預設為 2Gi。
sizeLimit 字串	使用 ephemeral 磁區時的大小限制。 檔案仍會寫入暫存目錄，因此您可以使用此選項來限制大小。 僅當 type 為 ephemeral 時才有效。
type 字串	要使用的磁區類型。 選擇 ephemeral 以使用非持續性儲存體，或選擇 persistent-claim 以使用持續性磁區。 預設為 persistent-claim。

.spec.queueManager.storage.queueManager

通常位於 /var/mqm 下之任何資料的預設 PersistentVolume 。 如果未指定其他磁區，則將包含所有持續保存的資料及回復日誌。

出現在:

- 第 124 頁的『.spec.queueManager.storage』

欄位	說明
class 字串	要用於此磁區的儲存類別。 僅當 type 為 persistent-claim 時才有效。 如果 type of availability 是 SingleInstance 或 NativeHA，則儲存類別可以是 ReadWriteOnce 或 ReadWriteMany 類型。 如果 type of availability 是 MultiInstance，則儲存類別必須是 ReadWriteMany 類型。
deleteClaim 布林值	刪除佇列管理程式時是否應刪除此磁區。
size 字串	要傳遞至 Kubernetes(包括 SI 單位) 的 PersistentVolume 大小。 僅當 type 為 persistent-claim 時才有效。 例如，2Gi。 預設為 2Gi。
sizeLimit 字串	使用 ephemeral 磁區時的大小限制。 檔案仍會寫入暫存目錄，因此您可以使用此選項來限制大小。 僅當 type 為 ephemeral 時才有效。
type 字串	要使用的磁區類型。 選擇 ephemeral 以使用非持續性儲存體，或選擇 persistent-claim 以使用持續性磁區。 預設為 persistent-claim。

.spec.queueManager.storage.recoveryLogs

MQ 回復日誌的持續性磁區詳細資料。 使用多重實例佇列管理程式時需要。

出現在:

- 第 124 頁的『.spec.queueManager.storage』

欄位	說明
class 字串	要用於此磁區的儲存類別。 僅當 type 為 persistent-claim 時才有效。 如果 type of availability 是 SingleInstance 或 NativeHA，則儲存類別可以是 ReadWriteOnce 或 ReadWriteMany 類型。 如果 type of availability 是 MultiInstance，則儲存類別必須是 ReadWriteMany 類型。
deleteClaim 布林值	刪除佇列管理程式時是否應刪除此磁區。
enabled 布林值	此磁區是否應該啟用為個別磁區，或放置在預設 queueManager 磁區上。 預設為 false。
size 字串	要傳遞至 Kubernetes(包括 SI 單位) 的 PersistentVolume 大小。 僅當 type 為 persistent-claim 時才有效。 例如，2Gi。 預設為 2Gi。
sizeLimit 字串	使用 ephemeral 磁區時的大小限制。 檔案仍會寫入暫存目錄，因此您可以使用此選項來限制大小。 僅當 type 為 ephemeral 時才有效。
type 字串	要使用的磁區類型。 選擇 ephemeral 以使用非持續性儲存體，或選擇 persistent-claim 以使用持續性磁區。 預設為 persistent-claim。

.spec.securityContext

要新增至佇列管理程式 Pod 的 securityContext 的安全設定。

出現在:

- 第 116 頁的『.spec』

欄位	說明
fsGroup 整數	套用至 Pod 中所有容器的特殊增補群組。 部分磁區類型容許 Kubelet 將該磁區的所有權變更為 Pod 所擁有: 1. 擁有的 GID 將是 FSGroup 2。 setgid 位元已設定 (在磁區中建立的新檔案將由 FSGroup 擁有) 3. 許可權位元是 OR ' d with rw-rw ---- 如果取消設定， Kubelet 將不會修改任何磁區的所有權和許可權。
initVolumeAsRoot 布林值	這會影響起始設定 PersistentVolume 的儲存器所使用的 securityContext 。 如果您使用的儲存體提供者需要您是 root 使用者才能存取新佈建的磁區，請將此設為 true 。 將此設為 true 會影響您可以使用的「安全環境定義限制 (SCC)」物件，而且如果您未獲授權使用容許 root 使用者的 SCC ，則「佇列管理程式」可能無法啟動。 預設為 false。 如需相關資訊，請參閱 https://docs.openshift.com/container-platform/latest/authentication/managing-security-context-constraints.html。
supplementalGroups 陣列	除了容器的主要 GID 之外，還在每一個容器中執行的第一個處理程序所套用的群組清單。 如果未指定，則不會將任何群組新增至任何儲存器。

.spec.template

Kubernetes 資源的進階範本。 範本可讓使用者置換 IBM MQ 如何產生基礎 Kubernetes 資源，例如 StatefulSet、Pod 及服務。 這僅適用於進階使用者，因為如果使用不正確，可能會中斷 MQ 的正常作業。 範本中的設定將置換 QueueManager 資源中的任何其他指定值。

出現在:

- 第 116 頁的『.spec』

欄位	說明
pod	置換用於 Pod 的範本。 請參閱 https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.17/#podspec-v1-core。

.spec.tracing

與 Cloud Pak for Integration 作業儀表板進行追蹤整合的設定。

出現在:

- 第 116 頁的『.spec』

欄位	說明
agent TracingAgent	僅在 Cloud Pak for Integration 中，您可以配置選用 Tracing Agent 的設定。
collector TracingCollector	僅在 Cloud Pak for Integration 中，您可以配置選用「追蹤收集器」的設定。
enabled 布林值	是否透過追蹤來啟用與 Cloud Pak for Integration 作業儀表板的整合。 預設為 false。
namespace 字串	Cloud Pak for Integration 作業儀表板安裝所在的名稱空間。

.spec.tracing.agent

僅在 Cloud Pak for Integration 中，您可以配置選用 Tracing Agent 的設定。

出現在:

- 第 127 頁的『.spec.tracing』

欄位	說明
image 字串	將使用的儲存器映像檔。
imagePullPolicy 字串	控制 kubelet 何時嘗試取回指定映像檔的設定。 預設為 IfNotPresent。
livenessProbe TracingProbe	控制存活性探測的設定。
readinessProbe TracingProbe	控制就緒性探測的設定。

.spec.tracing.agent.livenessProbe

控制存活性探測的設定。

出現在:

- 第 127 頁的『.spec.tracing.agent』

欄位	說明
failureThreshold 整數	在成功之後，將探測視為失敗的連續失敗次數下限。 預設值為 1。
initialDelaySeconds 整數	在儲存器啟動之後，起始活性探測之前的秒數。 預設值為 10 秒。 進一步資訊: https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle#container-probe。
periodSeconds 整數	執行探測的頻率（以秒為單位）。 預設值為 10 秒。
successThreshold 整數	在失敗之後將探測視為成功的連續成功數下限。 預設值為 1。
timeoutSeconds 整數	探測逾時之前經歷的秒數。 預設值為 2 秒。 進一步資訊: https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle#container-probe。

.spec.tracing.agent.readinessProbe

控制就緒性探測的設定。

出現在:

- 第 127 頁的『.spec.tracing.agent』

欄位	說明
failureThreshold 整數	在成功之後，將探測視為失敗的連續失敗次數下限。 預設值為 1。
initialDelaySeconds 整數	在儲存器啟動之後，起始活性探測之前的秒數。 預設值為 10 秒。 進一步資訊: https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle#container-probe。
periodSeconds 整數	執行探測的頻率（以秒為單位）。 預設值為 10 秒。
successThreshold 整數	在失敗之後將探測視為成功的連續成功數下限。 預設值為 1。
timeoutSeconds 整數	探測逾時之前經歷的秒數。 預設值為 2 秒。 進一步資訊: https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle#container-probe。

.spec.tracing.collector

僅在 Cloud Pak for Integration 中，您可以配置選用「追蹤收集器」的設定。

出現在:

- 第 127 頁的『.spec.tracing』

欄位	說明
image 字串	將使用的儲存器映像檔。
imagePullPolicy 字串	控制 kubelet 何時嘗試取回指定映像檔的設定。 預設為 IfNotPresent。
livenessProbe TracingProbe	控制存活性探測的設定。
readinessProbe TracingProbe	控制就緒性探測的設定。

.spec.tracing.collector.livenessProbe

控制存活性探測的設定。

出現在:

- 第 128 頁的『.spec.tracing.collector』

欄位	說明
failureThreshold 整數	在成功之後，將探測視為失敗的連續失敗次數下限。 預設值為 1。
initialDelaySeconds 整數	在儲存器啟動之後，起始活性探測之前的秒數。 預設值為 10 秒。 進一步資訊: https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle#container-probe。
periodSeconds 整數	執行探測的頻率（以秒為單位）。 預設值為 10 秒。
successThreshold 整數	在失敗之後將探測視為成功的連續成功數下限。 預設值為 1。
timeoutSeconds 整數	探測逾時之前經歷的秒數。 預設值為 2 秒。 進一步資訊: https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle#container-probe。

.spec.tracing.collector.readinessProbe

控制就緒性探測的設定。

出現在:

- 第 128 頁的『.spec.tracing.collector』

欄位	說明
`failureThreshold` 整數	在成功之後，將探測視為失敗的連續失敗次數下限。 預設值為 1。
`initialDelaySeconds` 整數	在儲存器啟動之後，起始活性探測之前的秒數。 預設值為 10 秒。 進一步資訊: https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle#container-probe。
`periodSeconds` 整數	執行探測的頻率（以秒為單位）。 預設值為 10 秒。
`successThreshold` 整數	在失敗之後將探測視為成功的連續成功數下限。 預設值為 1。
`timeoutSeconds` 整數	探測逾時之前經歷的秒數。 預設值為 2 秒。 進一步資訊: https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle#container-probe。

.spec.web

MQ Web 伺服器的設定。

出現在:

- 第 116 頁的『.spec』

欄位	說明
`enabled` 布林值	是否啟用 Web 伺服器。 預設為 false。

.status

QueueManager 的觀察狀態。

出現在:

- 第 116 頁的『QueueManager』

欄位	說明
`adminUiUrl` 字串	管理使用者介面的 URL。
`availability` 可用性	佇列管理程式的可用性狀態。
`conditions` QueueManagerStatusCondition 陣列	條件代表佇列管理程式狀態的最新可用觀察值。
`endpoints` QueueManagerStatusEndpoint 陣列	此「佇列管理程式」公開之端點 (例如 API 或使用者介面端點) 的相關資訊。
`name` 字串	佇列管理程式的名稱。
`phase` 字串	佇列管理程式狀態的階段。
`versions` QueueManagerStatusVersion	正在使用的 MQ 版本，以及 IBM Entitled Registry 中提供的其他版本。

.status.availability

佇列管理程式的可用性狀態。

出現在:

- 第 129 頁的『.status』

欄位	說明
initialQuorumEstablished 布林值	是否已為 NativeHA 建立起始仲裁。

.status.conditions

QueueManagerStatusCondition 會定義「佇列管理程式」的條件。

出現在:

- 第 129 頁的『.status』

欄位	說明
lastTransitionTime 字串	前次條件從某個狀態轉移至另一個狀態的時間。
message 字串	人類可讀的訊息，指出前次轉移的詳細資料。
reason 字串	此狀態前次轉移的原因。
status 字串	狀況的狀態。
type 字串	條件類型。

.status.endpoints

QueueManagerStatusEndpoint 會定義 QueueManager 的端點。

出現在:

- 第 129 頁的『.status』

欄位	說明
name 字串	端點的名稱。
type 字串	端點的類型，例如使用者介面端點的 'UI'、API 端點的 'API'、API 文件的 'OpenAPI'。
uri 字串	端點的 URI。

.status.versions

正在使用的 MQ 版本，以及 IBM Entitled Registry 中提供的其他版本。

出現在:

- 第 129 頁的『.status』

欄位	說明
available QueueManagerStatusVersion 可用	IBM Entitled Registry 中提供其他版本的 MQ 。
reconciled 字串	正在使用 IBM MQ 的特定版本。 如果指定自訂映像檔，則這可能不符合實際使用的 MQ 版本。

.status.versions.available

IBM Entitled Registry 中提供其他版本的 MQ 。

出現在:

- 第 130 頁的『.status.versions』

欄位	說明
channels 陣列	可用於自動更新 MQ 版本的通道。
versions 版本 陣列	可用的 MQ 特定版本。

.status.versions.available.versions

QueueManagerStatusVersion 會定義 MQ 的版本。

出現在:

- 第 130 頁的『.status.versions.available』

欄位	說明
name 字串	此版本 QueueManager 的 name 版。 這些是 spec.version 欄位的有效值。

OpenShift CP4I *QueueManager* 的狀態條件 *(mq.ibm.com/v1beta1)*

status.conditions 欄位會更新，以反映 QueueManager 資源的狀況。 一般而言，狀況會說明異常狀況。 處於健全且備妥狀態的佇列管理程式沒有 **Error** 或 **Pending** 狀況。 它可能具有一些諮詢 **Warning** 條件。

IBM MQ Operator 1.2 中引進了對條件的支援。

下列是針對 QueueManager 資源所定義的條件:

表 *1:* 佇列管理程式狀態條件			
元件	**條件類型**	**原因碼**	**訊息警告**
QueueManager[7]	擱置	建立	正在部署 MQ 佇列管理程式
	擱置	OidcPending	MQ 佇列管理程式正在等待 OIDC 用戶端登錄
	錯誤	失敗	MQ 佇列管理程式無法部署
	警告	UnsupportedVersion	[8] 運算子已安裝在 OCP *<ocp_version>*版上不受支援的運算元。 不支援此運算元。
	警告	EUSSupport	[9] 已安裝 EUS 運算元 *<mq_version>* ，但由不符合延伸支援持續時間資格的運算子進行管理。 此運算元不適用於延伸支援持續時間。
	警告	EUSSupport	[10] 已安裝 EUS 運算元 *<mq_version>* ，但 OCP 第 4 版*<ocp_version>* 不適用於延伸支援期間。 此運算元不適用於延伸支援持續時間。
	警告	EUSSupport	[11] 已安裝 EUS 運算元 *<mq_version>* ，但 OCP 版本 *<ocp_version>* 不適用於延長的支援持續時間。 根據一般 CD 版次支援此運算元。
Pod[12]	擱置	PodPending	正在部署 MQ 佇列管理程式的 Pod
	錯誤	PodFailed	正在部署 MQ 佇列管理程式的 Pod

[7] 條件 `Creating` 及 `Failed` 會監視佇列管理程式部署的整體進度。 如果您使用 IBM Cloud Pak for Integration 授權，且已啟用 MQ Web 主控台，則 `OidcPending` 條件會在等待 OIDC 用戶端登錄完成並使用 IAM 時，記載佇列管理程式的狀態。

[8] 運算子 1.4.0 以及更新版本

[9] 運算子 1.4.0 以及更新版本

[10] 運算子 1.4.0 以及更新版本

[11] 僅限運算子 1.3.0

[12] 在部署佇列管理程式期間， Pod 條件會監視 Pod 的狀態。 如果您看到任何 `PodFailed` 條件，則整體佇列管理程式條件也會設為 `Failed`。

表 1: 佇列管理程式狀態條件 (繼續)			
元件	**條件類型**	**原因碼**	**訊息警告**
儲存體 [13]	擱置	StoragePending	正在供應 MQ 佇列管理程式的儲存體
	警告	StorageEphemeral	將暫時儲存體用於正式作業 MQ 佇列管理程式
	錯誤	StorageFailed	無法供應 MQ 佇列管理程式的儲存體

Multi 建置您自己的 IBM MQ 容器及部署程式碼

開發自行建置的儲存器。 這是最具彈性的容器解決方案，但它需要您具備配置容器的強大技能，以及 "擁有" 產生的容器。

開始之前

在開發您自己的容器之前，請考量您是否可以改用 IBM 所提供的其中一個預先包裝容器。 請參閱 儲存器中的 IBM MQ

關於這項作業

當您將 IBM MQ 包裝成容器映像檔時，應用程式的變更可以快速且輕鬆地部署至測試和暫置系統。 這可能是企業持續交付的主要好處。

程序

- 第 133 頁的『使用儲存器規劃您自己的 IBM MQ 佇列管理程式映像檔』
- 第 134 頁的『建置範例 IBM MQ 佇列管理程式儲存器映像檔』
- 第 136 頁的『在個別儲存器中執行本端連結應用程式』

相關概念

儲存器中的 IBM MQ

Multi 使用儲存器規劃您自己的 IBM MQ 佇列管理程式映像檔

在儲存器中執行 IBM MQ 佇列管理程式時，有數個需求需要考量。 範例容器映像檔提供處理這些需求的方法，但如果您想要使用自己的映像檔，則需要考量如何處理這些需求。

程序監督

當您執行容器時，基本上是執行單一處理程序 (容器內的 PID 1)，它可以在稍後大量產生子處理程序。

如果主要處理程序結束，則儲存器執行時期會停止儲存器。 IBM MQ 佇列管理程式需要多個處理程序在背景中執行。

因此，您需要確定只要佇列管理程式在執行中，您的主要處理程序就會保持作用中。 例如，透過執行管理查詢來檢查佇列管理程式在此處理程序中是否處於作用中狀態，是很好的作法。

移入資料 /var/mqm

儲存器必須以 /var/mqm 作為磁區來配置。

[13] 儲存體狀況會監視建立持續性儲存體磁區之要求的進度 (StoragePending 狀況)，並報告往回連結錯誤及其他失敗。 如果在儲存體供應期間發生任何錯誤，則 StorageFailed 條件會新增至條件清單，且整體佇列管理程式條件也會設為 Failed。

當您這麼做時，當儲存器第一次啟動時，磁區的目錄是空的。 此目錄通常在安裝時移入，但在使用儲存器時，安裝與執行時期是個別環境。

若要解決此問題，當容器啟動時，您可以使用 **crtmqdir** 指令，在 /var/mqm 第一次執行時移入它。

容器安全

為了將執行時期安全需求降至最低，範例儲存器映像檔是使用 IBM MQ 可解壓縮安裝來安裝。 這可確保未設定任何 setuid 位元，且儲存器不需要使用專用權提升。 部分儲存器系統會定義您可以使用的使用者 ID，而不可壓縮的安裝不會對可用的作業系統使用者做出任何假設。

Multi 建置範例 IBM MQ 佇列管理程式儲存器映像檔

使用此資訊來建置範例儲存器映像檔，以在儲存器中執行 IBM MQ 佇列管理程式。

關於這項作業

首先，您建置基本映像檔，其中包含 Red Hat 通用基本映像檔檔案系統及 IBM MQ 的全新安裝。

其次，您在基本程式之上建置另一個容器映像檔層，這會新增一些 IBM MQ 配置，以容許基本使用者 ID 及密碼安全。

最後，您可以使用此映像檔作為其檔案系統來執行儲存器，並使用主機檔案系統上儲存器特定磁區所提供的 /var/mqm 內容。

程序

- 如需如何建置範例儲存器映像檔以在儲存器中執行 IBM MQ 佇列管理程式的相關資訊，請參閱下列子主題:
  - 第 134 頁的『建置範例基本 IBM MQ 佇列管理程式映像檔』
  - 第 134 頁的『建置已配置的範例 IBM MQ 佇列管理程式映像檔』

Multi 建置範例基本 IBM MQ 佇列管理程式映像檔

為了在您自己的容器映像檔中使用 IBM MQ，您一開始需要使用全新的 IBM MQ 安裝來建置基本映像檔。下列步驟顯示如何使用在 GitHub 上管理的範例程式碼來建置範例基本映像檔。

程序

- 使用 mq-container GitHub 儲存庫 中提供的 make 檔來建置正式作業儲存器映像檔。

  遵循 GitHub 上 建置容器映像檔 中的指示。 如果您計劃使用 Red Hat OpenShift Container Platform "受限" 安全環境定義限制 (SCC) 來配置安全存取，則必須使用 'No-Install' IBM MQ 套件。

結果

您現在具有已安裝 IBM MQ 的基本容器映像檔。

現在，您已準備好 建置已配置的範例 IBM MQ 佇列管理程式映像檔。

Multi 建置已配置的範例 IBM MQ 佇列管理程式映像檔

建置通用基本 IBM MQ 儲存器映像檔之後，您需要套用自己的配置，以容許安全存取。 若要這樣做，您可以使用一般映像檔作為母項來建立自己的容器映像檔層。

開始之前

V 9.2.0 此作業假設當您 建置範例基本 IBM MQ 佇列管理程式映像檔時，已使用 "No-Install" IBM MQ 套件。 否則，您無法使用 Red Hat OpenShift Container Platform "受限" 安全環境定義限制 (SCC) 來配置安

全存取。 依預設使用的 "restricted" SCC 會使用隨機使用者 ID，並透過變更為不同的使用者來防止專用權升級。 IBM MQ 傳統 RPM 型安裝程式依賴於 mqm 使用者和群組，並且也使用可執行程式上的 setuid 位元。 在 IBM MQ 9.2 中，當您使用 "無安裝" IBM MQ 套件時，不再有 mqm 使用者，也沒有 mqm 群組。

程序

1. 建立新的目錄，並新增名為 config.mqsc 的檔案，其內容如下:

```
DEFINE QLOCAL(EXAMPLE.QUEUE.1) REPLACE
```

請注意，前述範例使用簡式使用者 ID 和密碼鑑別。 不過，您可以套用企業需要的任何安全配置。

2. 建立名為 Dockerfile 的檔案，其內容如下:

```
FROM mq
COPY config.mqsc /etc/mqm/
```

3. 使用下列指令來建置自訂容器映像檔:

```
docker build -t mymq .
```

其中 "." 是包含您剛建立的兩個檔案的目錄。

然後， Docker 會使用該映像檔建立暫時儲存器，並執行其餘指令。

註: 在 Red Hat Enterprise Linux (RHEL) 上，您使用指令 **docker** (RHEL V7) 或 **podman** (RHEL V7 或 RHEL V8)。 在 Linux 上，您需要在指令開頭執行 **docker** 指令與 **sudo** ，以取得額外專用權。

4. 執行新的自訂映像檔，以使用您剛才建立的磁碟映像檔來建立新的容器。

您的新映像檔層未指定任何要執行的特定指令，因此已從母項映像檔繼承。 母項的進入點 (程式碼可在 GitHub 上找到):

- 建立佇列管理程式
- 啟動佇列管理程式
- 建立預設接聽器
- 然後從 /etc/mqm/config.mqsc. 執行任何 MQSC 指令

發出下列指令，以執行新的自訂映像檔:

```
docker run \
  --env LICENSE=accept \
  --env MQ_QMGR_NAME=QM1 \
  --volume /var/example:/var/mqm \
  --publish 1414:1414 \
  --detach \
  mymq
```

其中:

第一個 env 參數
將環境變數傳遞至儲存器，以確認您接受 IBM IBM WebSphere MQ 的授權。 您也可以設定要檢視的 LICENSE 變數，以檢視授權。

如需 IBM MQ 授權的進一步詳細資料，請參閱 IBM MQ 授權資訊 。

第二個 env 參數
設定您正在使用的佇列管理程式名稱。

磁區參數
告訴儲存器， MQ 寫入 /var/mqm 的任何內容實際上都應該寫入主機上的 /var/example 。

此選項表示您稍後可以輕鬆地刪除儲存器，並仍然保留任何持續資料。 此選項也可讓您更容易檢視日誌檔。

發佈參數
將主機系統上的埠對映至儲存器中的埠。 依預設，容器會使用其自己的內部 IP 位址來執行，這表示您需要特別對映您要公開的任何埠。

在此範例中，這表示將主機上的埠 1414 對映至儲存器中的埠 1414。

分離參數
在背景中執行儲存器。

結果

您已建置已配置的容器映像檔，並且可以使用 **docker ps** 指令來檢視執行中容器。 您可以使用 **docker top** 指令來檢視在容器中執行的 IBM MQ 處理程序。

小心:

您可以使用 **docker logs ${CONTAINER_ID}** 指令來檢視容器的日誌。

下一步

- 如果在使用 **docker ps** 指令時未顯示容器，則容器可能已失敗。 您可以使用 **docker ps -a** 指令來查看失敗的儲存器。
- 當您使用 **docker ps -a** 指令時，會顯示儲存器 ID。 當您發出 **docker run** 指令時，也會列印此 ID。
- 您可以使用 **docker logs ${CONTAINER_ID}** 指令來檢視容器的日誌。

Multi 在個別儲存器中執行本端連結應用程式

使用 Docker 中儲存器之間的程序名稱空間共用，您可以在 IBM MQ 佇列管理程式的個別儲存器中執行需要本端連結連線至 IBM MQ 的應用程式。

關於這項作業

此功能在 IBM MQ 9.0.3 以及更新版本的佇列管理程式中受支援。

您必須遵守下列限制:

- 您必須使用 `--pid` 引數來共用儲存器 PID 名稱空間。
- 您必須使用 `--ipc` 引數來共用儲存器 IPC 名稱空間。
- 您必須:
  1. 使用 `--uts` 引數與主機共用儲存器 UTS 名稱空間，或
  2. 使用 `-h` 或 `--hostname` 引數，確保儲存器具有相同的主機名稱。
- 您必須在可供 `/var/mqm` 目錄下所有儲存器使用的磁區中裝載 IBM MQ 資料目錄。

在已安裝 Docker 的 Linux 系統上完成下列步驟，即可試用此功能。

下列範例使用範例 IBM MQ 儲存器映像檔。 您可以在 Github 上找到此映像檔的詳細資料。

程序

1. 發出下列指令，以建立暫存目錄來作為您的磁區:

```
mkdir /tmp/dockerVolume
```

2. 發出下列指令，在儲存器中建立名稱為 `sharedNamespace` 的佇列管理程式 (`QM1`):

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

3. 發出下列指令，以啟動稱為 `secondaryContainer` 的第二個儲存器 (以 `ibmcom/mq` 為基礎)，但不建立佇列管理程式:

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

4. 發出下列指令，在第二個儲存器上執行 **dspmq** 指令，以查看兩個佇列管理程式的狀態:

```
docker exec secondaryContainer dspmq
```

5. 執行下列指令，以針對在其他儲存器上執行的佇列管理程式處理 MQSC 指令:

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

結果

現在，您已在個別儲存器中執行本端應用程式，而且現在可以從次要儲存器順利執行 **dspmq**、 **amqsput**、 **amqsget** 及 **runmqsc** 等指令作為 QM1 佇列管理程式的本端連結。

如果您未看到預期的結果，請參閱 第 137 頁的『對名稱空間應用程式進行疑難排解』 以取得相關資訊。

Multi 對名稱空間應用程式進行疑難排解

使用共用名稱空間時，您必須確保共用所有名稱空間 (IPC、PID 及 UTS/hostname) 及裝載磁區，否則您的應用程式將無法運作。

如需您必須遵循的限制清單，請參閱 第 136 頁的『在個別儲存器中執行本端連結應用程式』 。

如果您的應用程式不符合列出的所有限制，您可能會在儲存器啟動時遇到問題，但您預期的功能無法運作。

下列清單概述一些常見原因，以及您在忘記符合其中一項限制時可能看到的行為。

- 如果您忘記共用名稱空間 (UTS/PID/IPC) 或容器的主機名稱，並裝載磁區，則容器將能夠看到佇列管理程式，但無法與佇列管理程式互動。
  - 對於 **dspmq** 指令，您會看到下列:

```
docker exec container dspmq

QMNAME(QM1)                       STATUS(Status not available)
```

  - 對於 **runmqsc** 指令，或嘗試連接至佇列管理程式的其他指令，您可能會收到 AMQ8146 錯誤訊息:

```
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
```

- 如果您共用所有必要的名稱空間，但未將共用磁區裝載至 /var/mqm 目錄，且您具有有效的 IBM MQ 資料路徑，則您的指令也會收到 AMQ8146 錯誤訊息。

  不過， **dspmq** 完全無法看到您的佇列管理程式，而是傳回空白回應:

```
docker exec container dspmq
```

- 如果您共用所有必要的名稱空間，但未將共用磁區裝載至 /var/mqm 目錄，且您沒有有效的 IBM MQ 資料路徑 (或沒有 IBM MQ 資料路徑) ，則會看到各種錯誤，因為資料路徑是 IBM MQ 安裝架構的重要元件。如果沒有資料路徑，則 IBM MQ 無法運作。

  如果您執行下列任何指令，且看到類似這些範例中所示的回應，則應該驗證您已裝載目錄或建立 IBM MQ 資料目錄:

```
docker exec container dspmq
'No such file or directory' from /var/mqm/mqs.ini
AMQ6090: IBM MQ was unable to display an error message FFFFFFFF.
AMQffff

docker exec container dspmqver
AMQ7047: An unexpected error was encountered by a command. Reason code is 0.

docker exec container mqrc
<file path>/mqrc.c[1152]
```

```
lpiObtainQMDetails --> 545261715

docker exec container crtmqm QM1
AMQ8101: IBM MQ error (893) has occurred.

docker exec container strmqm QM1
AMQ6239: Permission denied attempting to access filesystem location '/var/mqm'.
AMQ7002: An error occurred manipulating a file.

docker exec container endmqm QM1
AMQ8101: IBM MQ error (893) has occurred.

docker exec container dltmqm QM1
AMQ7002: An error occurred manipulating a file.

docker exec container strmqweb
<file path>/mqrc.c[1152]
lpiObtainQMDetails --> 545261715
```

CP4I 建立原生 HA 群組 (如果建立您自己的儲存器)

您必須建立、配置及啟動三個佇列管理程式，才能建立「原生 HA」群組。

關於這項作業

建立原生 HA 解決方案的建議方法是使用 IBM MQ 運算子 (請參閱 原生 HA)。 或者，如果您建立自己的儲存器，則可以遵循下列指示。

若要建立原生 HA 群組，您可以在三個節點上建立三個佇列管理程式，並將其日誌類型設為 `log replication`。 然後，您可以編輯每一個佇列管理程式的 `qm.ini` 檔案，以新增三個節點中每一個節點的連線詳細資料，以便它們可以彼此抄寫日誌資料。

然後，您必須啟動這三個佇列管理程式，以便它們可以檢查這三個實例是否可以彼此通訊，並決定其中哪些將是作用中實例，哪些將是抄本。

程序

1. 在三個節點中的每一個節點上，建立佇列管理程式，指定日誌類型的日誌抄本，並提供每一個日誌實例的唯一名稱。 每一個佇列管理程式都具有相同的名稱:

   ```
   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. 順利建立每一個佇列管理程式時，會將另一個名為 `NativeHALocalInstance` 的段落新增至佇列管理程式配置檔 `qm.ini`。 `Name` 屬性會新增至指定所提供實例名稱的段落。

   您可以選擇性地將下列屬性新增至 `qm.ini` 檔中的 `NativeHALocalInstance` 段落:

   KeyRepository
   金鑰儲存庫的位置，該金鑰儲存庫保留用於保護日誌抄寫資料流量的數位憑證。 位置以詞幹格式提供，亦即，它包括不含副檔名的完整路徑和檔名。 如果省略 `KeyRepository` 段落屬性，則會以純文字在實例之間交換日誌抄寫資料。

   CertificateLabel
   憑證標籤，識別用於保護日誌抄寫資料流量的數位憑證。 如果提供 `KeyRepository` ，但省略 `CertificateLabel` ，則會使用預設值 `ibmwebspheremq`*`queue_manager`* 。

   CipherSpec
   用來保護日誌抄寫資料流量的 MQ CipherSpec 。 如果提供此段落屬性，則也必須提供 `KeyRepository` 。 如果提供 `KeyRepository` ，但省略 `CipherSpec` ，則會使用預設值 `ANY` 。

LocalAddress
接受日誌抄寫資料流量的本端網路介面位址。 如果提供此段落屬性，則會使用 "[addr] [(port)]" 格式來識別本端網路介面及/或埠。 網址可以指定為主機名稱、 IPv4 帶點十進位或 IPv6 十六進位格式。 如果省略此屬性，佇列管理程式會嘗試連結至所有網路介面，它會使用 ReplicationAddress 中符合本端實例名稱之 NativeHAInstances 段落中指定的埠。

HeartbeatInterval
活動訊號間隔定義原生 HA 佇列管理程式的作用中實例傳送網路活動訊號的頻率 (毫秒)。 活動訊號間隔值的有效範圍是 500 (0.5 秒) 至 60000 (1 分鐘) ，超出此範圍的值會導致佇列管理程式無法啟動。 如果省略此屬性，則會使用預設值 5000 (5 秒)。 每一個實例都必須使用相同的活動訊號間隔。

HeartbeatTimeout
活動訊號逾時值定義原生 HA 佇列管理程式的抄本實例在決定作用中實例無回應之前等待的時間。 活動訊號間隔逾時值的有效範圍是 500 (0.5 秒) 至 120000 (2 分鐘)。 活動訊號逾時值必須大於或等於活動訊號間隔。

無效值會導致佇列管理程式無法啟動。 如果省略此屬性，則抄本會等待 2 x HeartbeatInterval ，然後再啟動處理程序來選取新的作用中實例。 每一個實例都必須使用相同的活動訊號逾時。

RetryInterval
重試間隔定義原生 HA 佇列管理程式應該重試失敗抄寫鏈結的頻率 (毫秒)。 重試間隔的有效範圍是 500 (0.5 秒) 至 120000 (2 分鐘)。 如果省略此屬性，在重試失敗的抄寫鏈結之前，抄本會等待 2 x HeartbeatInterval 。

3. 編輯每一個佇列管理程式的 qm.ini 檔，並新增連線詳細資料。 您可以新增三個 NativeHAInstance 段落，「原生 HA」群組 (包括本端實例) 中的每一個佇列管理程式實例各一個。 新增下列屬性:

   姓名
   指定您建立佇列管理程式實例時所使用的實例名稱。

   ReplicationAddress
   指定實例的主機名稱 IPv4 帶點十進位或 IPv6 十六進位格式位址。 您可以將位址指定為主機名稱、IPv4 帶點十進位或 IPv6 十六進位格式位址。 抄寫位址必須可解析且可從群組中的每一個實例遞送。 用於日誌抄寫的埠號必須以方括弧 ([]) 指定，例如:

   ```
   ReplicationAddress=host1.example.com(4444)
   ```

   註: NativeHAInstance 段落在每個實例上都相同，且可以使用自動配置 (**crtmqm -ii**) 來提供。

4. 啟動三個實例中的每一個:

   ```
   strmqm QMgrName
   ```

   當實例啟動時，它們會進行通訊以檢查這三個實例是否都在執行中，然後決定這三個實例中的哪一個是作用中實例，而另外兩個實例則繼續以抄本形式執行。

範例

下列範例顯示 qm.ini 檔案的區段，其中指定三個實例之一的必要「原生 HA」詳細資料:

```
NativeHALocalInstance:
  LocalName=node-1

NativeHAInstance:
  Name=node-1
  ReplicationAddress=host1.example.com(4444)
NativeHAInstance:
  Name=node-2
  ReplicationAddress=host2.example.com(4444)
NativeHAInstance:
  Name=node-3
  ReplicationAddress=host3.example.com(4444)
```

Kubernetes 執行您自己的原生 HA 佇列管理程式漸進式更新的考量

對原生 HA 佇列管理程式的 IBM MQ 版本或 Pod 規格進行任何更新都需要您執行佇列管理程式實例的漸進式更新。 IBM MQ Operator 會自動為您處理此問題，但如果您要建置自己的部署程式碼，則有一些重要考量。

註: 範例 Helm Chart 包含 Shell Script 來執行漸進式更新，但 Script **不** 適合正式作業使用，因為它未解決本主題中的考量。

在 Kubernetes 中， `StatefulSet` 資源用來管理依序啟動及漸進式更新。 啟動程序的一部分是個別啟動每一個 Pod，等待它備妥，然後移至下一個 Pod。 這不適用於原生 HA，因為所有 Pod 都需要啟動，才能進行領導者選舉。 因此， `StatefulSet` 上的 `.spec.podManagementPolicy` 欄位需要設為 `Parallel`。 這也表示所有 Pod 也會平行更新，這尤其不需要。 因此， `StatefulSet` 也應該使用 `OnDelete` 更新策略。

無法使用 `StatefulSet` 漸進式更新程式碼會導致需要自訂漸進式更新程式碼，這應該考量下列各項:

- 一般漸進式更新程序
- 以最佳順序更新 Pod 以將關閉時間縮至最短
- 處理叢集狀態中的變更
- 處理錯誤
- 處理計時問題

一般漸進式更新程序

漸進式更新程式碼應該等待每一個實例從 `dspmq` 顯示 `REPLICA` 狀態。 這表示實例已執行某個層次的啟動 (例如，容器已啟動，且 MQ 處理程序正在執行中)，但它還不一定能夠與其他實例交談。 例如: Pod A 會重新啟動，只要它處於 `REPLICA` 狀態，就會重新啟動 Pod B。 一旦 Pod B 以新配置開始，它應該能夠與 Pod A 交談，並且可以形成仲裁， A 或 B 將變成新的作用中實例。

在此過程中，在每一個 Pod 都達到 `REPLICA` 狀態之後，如果要讓它連接至其對等節點並建立仲裁，則延遲很有用。

以最佳順序更新 Pod 以將關閉時間縮至最短

漸進式更新程式碼應該一次刪除一個 Pod，從處於已知錯誤狀態的 Pod 開始，接著是任何未順利啟動的 Pod。 作用中佇列管理程式 Pod 通常應該最後更新。

如果前次更新導致 Pod 進入已知錯誤狀態，則暫停刪除 Pod 也很重要。 這可防止跨所有 Pod 部署中斷的更新。 例如，如果 Pod 更新為使用無法存取 (或包含拼字錯誤) 的新容器映像檔，則會發生此情況。

處理叢集狀態中的變更

漸進式更新程式碼需要適當地回應叢集狀態中的即時變更。 例如，由於「節點」重新開機或「節點」壓力，可能會收回其中一個佇列管理程式的 Pod。 如果叢集忙碌，可能不會立即重新排定收回的 Pod。 在此情況下，在重新啟動任何其他 Pod 之前，漸進式更新程式碼需要適當地等待。

處理錯誤

當呼叫 Kubernetes API 及其他非預期的叢集行為時，漸進式更新程式碼必須健全而不會失敗。

此外，漸進式更新程式碼本身需要容忍重新啟動。 漸進式更新可能長時間執行，且可能需要重新啟動程式碼。

處理計時問題

漸進式更新程式碼需要檢查 Pod 的更新修訂，以便它可以確保 Pod 已重新啟動。 這可避免 Pod 可能指出其「已啟動」但實際上尚未終止的計時問題。

相關概念

第 5 頁的『選擇您要如何在儲存器中使用 IBM MQ』

在容器中使用 IBM MQ 有多個選項: 您可以選擇使用 IBM MQ Operator，它會使用預先包裝的容器映像檔，或者您可以建置自己的映像檔及部署程式碼。

CP4I 檢視自訂建置儲存器的原生 HA 佇列管理程式狀態

對於自訂建置的儲存器，您可以使用 **dspmq** 指令來檢視原生 HA 實例的狀態。

關於這項作業

您可以使用 **dspmq** 指令來檢視節點上佇列管理程式實例的作業狀態。 傳回的資訊視實例是作用中還是抄本而定。 作用中實例所提供的資訊是明確的，抄本節點的資訊可能已過期。

您可以執行下列動作:

- 檢視現行節點上的佇列管理程式實例是作用中還是抄本。
- 檢視現行節點上實例的原生 HA 作業狀態。
- 檢視原生 HA 配置中所有三個實例的作業狀態。

下列狀態欄位用來報告原生 HA 配置狀態:

角色
指定實例的現行角色，並且是 Active、 Replica 或 Unknown 之一。

實例
使用 **crtmqm** 指令的 **-lr** 選項建立此佇列管理程式實例時，為其提供的名稱。

INSYNC
指出實例是否可以在必要時接管作為作用中實例。

仲裁
以 *number_of_instances_in-sync*/*number_of_instances_configured* 格式報告仲裁狀態。

REPLADDR
佇列管理程式實例的抄寫位址。

CONNACTV
指出節點是否連接至作用中實例。

BACKLOG
指出實例落後的 KB 數。

CONNINST
指出指定的實例是否連接至此實例。

ALTDATE
指出前次更新此資訊的日期 (如果從未更新過，則為空白)。

ALTTIME
指出前次更新此資訊的時間 (如果從未更新過，則為空白)。

程序

- 若要判定佇列管理程式實例是作為作用中實例還是抄本執行，請執行下列動作:

```
dspmq -o status -m QMgrName
```

名為 BOB 之佇列管理程式的作用中實例會報告下列狀態:

```
QMNAME(BOB)                   STATUS(Running)
```

名為 BOB 之佇列管理程式的抄本實例會報告下列狀態:

```
QMNAME(BOB)                   STATUS(Replica)
```

非作用中實例會報告下列狀態:

```
QMNAME(BOB)                 STATUS(Ended Immediately)
```

- 若要判定現行節點上實例的原生 HA 作業狀態，請執行下列動作:

  ```
  dspmq -o nativeha -m QMgrName
  ```

  名為 BOB 之佇列管理程式的作用中實例可能報告下列狀態:

  ```
  QMNAME(BOB)               ROLE(Active) INSTANCE(inst1) INSYNC(Yes) QUORUM(3/3)
  ```

  佇列管理程式 BOB 的抄本實例可能會報告下列狀態:

  ```
  QMNAME(BOB)               ROLE(Replica) INSTANCE(inst2) INSYNC(Yes) QUORUM(2/3)
  ```

  名為 BOB 之佇列管理程式的非作用中實例可能會報告下列狀態:

  ```
  QMNAME(BOB)               ROLE(Unknown) INSTANCE(inst3) INSYNC(no) QUORUM(0/3)
  ```

- 若要判定「原生 HA」配置中所有實例的「原生 HA」作業狀態，請執行下列動作:

  ```
  dspmq -o nativeha -x -m QMgrName
  ```

  如果您在執行佇列管理程式 BOB 作用中實例的節點上發出此指令，則可能會收到下列狀態:

  ```
  QMNAME(BOB)            ROLE(Active) INSTANCE(inst1) INSYNC(Yes) QUORUM(3/3)
   INSTANCE(inst1) ROLE(Active)  REPLADDR(9.20.123.45) CONNACTV(Yes) INSYNC(Yes) BACKLOG(0)
  CONNINST(Yes) ALTDATE(2022-01-12) ALTTIME(12.03.44)
   INSTANCE(inst2) ROLE(Replica) REPLADDR(9.20.123.46) CONNACTV(Yes) INSYNC(Yes) BACKLOG(0)
  CONNINST(Yes) ALTDATE(2022-01-12) ALTTIME(12.03.44)
   INSTANCE(inst3) ROLE(Replica) REPLADDR(9.20.123.47) CONNACTV(Yes) INSYNC(Yes) BACKLOG(0)
  CONNINST(Yes) ALTDATE(2022-01-12) ALTTIME(12.03.44)
  ```

  如果您在執行佇列管理程式 BOB 抄本實例的節點上發出此指令，則可能會收到下列狀態，指出其中一個抄本落後:

  ```
  QMNAME(BOB)            ROLE(Replica) INSTANCE(inst2) INSYNC(Yes) QUORUM(2/3)
   INSTANCE(inst2) ROLE(Replica) REPLADDR(9.20.123.46) CONNACTV(Yes) INSYNC(Yes) BACKLOG(0)
  CONNINST(Yes) ALTDATE(2022-01-12) ALTTIME(12.03.44)
   INSTANCE(inst1) ROLE(Active)  REPLADDR(9.20.123.45) CONNACTV(Yes) INSYNC(Yes) BACKLOG(0)
  CONNINST(Yes) ALTDATE(2022-01-12) ALTTIME(12.03.44)
   INSTANCE(inst3) ROLE(Replica) REPLADDR(9.20.123.47) CONNACTV(Yes) INSYNC(No)  BACKLOG(435)
  CONNINST(Yes) ALTDATE(2022-01-12) ALTTIME(12.03.44)
  ```

  如果您在執行佇列管理程式 BOB 非作用中實例的節點上發出此指令，則可能會收到下列狀態:

  ```
  QMNAME(BOB)            ROLE(Unknown) INSTANCE(inst3) INSYNC(no) QUORUM(0/3)
   INSTANCE(inst1) ROLE(Unknown) REPLADDR(9.20.123.45) CONNACTV(Unknown) INSYNC(Unknown)
  BACKLOG(Unknown) CONNINST(No) ALTDATE() ALTTIME()
   INSTANCE(inst2) ROLE(Unknown) REPLADDR(9.20.123.46) CONNACTV(Unknown) INSYNC(Unknown)
  BACKLOG(Unknown) CONNINST(No) ALTDATE() ALTTIME()
   INSTANCE(inst3) ROLE(Unknown) REPLADDR(9.20.123.47) CONNACTV(No) INSYNC(Unknown)
  BACKLOG(Unknown) CONNINST(No) ALTDATE() ALTTIME()
  ```

  如果您在實例仍在協議作用中及抄本時發出指令，則會收到下列狀態:

  ```
  QMNAME(BOB)              STATUS(Negotiating)
  ```

相關參考
dspmq

CP4I 正在結束原生 HA 佇列管理程式

您可以使用 **endmqm** 指令來結束屬於「原生 HA」群組的作用中或抄本佇列管理程式。

程序

- 若要結束佇列管理程式的作用中實例，請參閱本文件「配置」一節中的 結束原生 HA 佇列管理程式 。

注意事項

本資訊係針對 IBM 在美國所提供之產品與服務所開發。

在其他國家中，IBM 可能不會提供本書中所提的各項產品、服務或功能。 請洽當地 IBM 業務代表，以取得當地目前提供的產品和服務之相關資訊。 這份文件在提及 IBM 的產品、程式或服務時，不表示或暗示只能使用 IBM 的產品、程式或服務。 只要未侵犯 IBM 的智慧財產權，任何功能相當的產品、程式或服務都可以取代 IBM 的產品、程式或服務。 不過，任何非 IBM 的產品、程式或服務，使用者必須自行負責作業的評估和驗證責任。

本文件所說明之主題內容，IBM 可能擁有其專利或專利申請案。 提供本文件不代表提供這些專利的授權。 您可以書面提出授權查詢，來函請寄到：

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

如果是有關雙位元組 (DBCS) 資訊的授權查詢，請洽詢所在國的 IBM 智慧財產部門，或書面提出授權查詢，來函請寄到：

智慧財產權授權
法務部與智慧財產權法律
IBM Japan, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

下列段落不適用於英國，若與任何其他國家之法律條款抵觸，亦不適用於該國： International Business Machines Corporation 只依 "現況" 提供本出版品，不提供任何明示或默示之保證，其中包括且不限於不侵權、可商用性或特定目的之適用性的隱含保證。 有些地區在特定交易上，不允許排除明示或暗示的保證，因此，這項聲明不一定適合您。

這項資訊中可能會有技術上或排版印刷上的訛誤。 因此，IBM 會定期修訂；並將修訂後的內容納入新版中。 IBM 隨時會改進及/或變更本出版品所提及的產品及/或程式，不另行通知。

本資訊中任何對非 IBM 網站的敘述僅供參考，IBM 對該網站並不提供任何保證。 這些網站所提供的資料不是 IBM 本產品的資料內容，如果要使用這些網站的資料，您必須自行承擔風險。

IBM 得以各種適當的方式使用或散布由您提供的任何資訊，無需對您負責。

如果本程式的獲授權人為了 (i) 在個別建立的程式和其他程式（包括本程式）之間交換資訊，以及 (ii) 相互使用所交換的資訊，因而需要相關的資訊，請洽詢：

IBM Corporation
軟體交互作業能力協調程式，部門 49XA
3605 公路 52 N
Rochester， MN 55901
U.S.A.

在適當條款與條件之下，包括某些情況下（支付費用），或可使用此類資訊。

IBM 基於雙方之 IBM 客戶合約、IBM 國際程式授權合約或任何同等合約之條款，提供本資訊所提及的授權程式與其所有適用的授權資料。

本文件中所含的任何效能資料都是在受管制的環境下判定。 因此不同作業環境之下所得的結果，可能會有很大的差異。 有些測定已在開發階段系統上做過，不過這並不保證在一般系統上會出現相同結果。 甚至有部分的測量，是利用插補法而得的估計值， 實際結果可能有所不同。 本書的使用者應依自己的特定環境，查證適用的資料。

本文件所提及之非 IBM 產品資訊，取自產品的供應商，或其發佈的聲明或其他公開管道。 IBM 並未測試過這些產品，也無法確認這些非 IBM 產品的執行效能、相容性或任何對產品的其他主張是否完全無誤。 有關非 IBM 產品的性能問題應直接洽詢該產品供應商。

有關 IBM 未來方針或目的之所有聲明，僅代表 IBM 的目標與主旨，隨時可能變更或撤銷，不必另行通知。

這份資訊含有日常商業運作所用的資料和報告範例。 為了要使它們儘可能完整，範例包括個人、公司、 品牌和產品的名稱。 這些名稱全屬虛構，如與實際公司的名稱和住址雷同，純屬巧合。

著作權授權:

本資訊含有原始語言之範例應用程式，用以說明各作業平台中之程式設計技術。 您可以基於研發、使用、銷售或散布符合作業平台（撰寫範例程式的作業平台）之應用程式介面的應用程式等目的，以任何形式複製、修改及散布這些範例程式，而不必向 IBM 付費。 這些範例並未在所 有情況下完整測試。 因此，IBM 不保證或暗示這些程式的可靠性、有用性或功能。

若 貴客戶正在閱讀本項資訊的電子檔，可能不會有照片和彩色說明。

程式設計介面資訊

程式設計介面資訊 (如果有提供的話) 旨在協助您建立與此程式搭配使用的應用軟體。

本書包含預期程式設計介面的相關資訊，可讓客戶撰寫程式以取得 WebSphere MQ 的服務。

不過，本資訊也可能包含診斷、修正和調整資訊。 提供診斷、修正和調整資訊，是要協助您進行應用軟體的除錯。

重要: 請勿使用此診斷、修改及調整資訊作為程式設計介面，因為它可能會變更。

商標

IBM、 IBM 標誌 ibm.com 是 IBM Corporation 在全球許多適用範圍的商標。 IBM 商標的最新清單可在 Web 的 "Copyright and trademark information"www.ibm.com/legal/copytrade.shtml 中找到。 其他產品和服務名稱，可能是 IBM 或其他公司的商標。

Microsoft 及 Windows 是 Microsoft Corporation 在美國及/或其他國家或地區的商標。

UNIX 是 The Open Group 在美國及/或其他國家/地區的註冊商標。

Linux 是 Linus Torvalds 在美國及/或其他國家或地區的註冊商標。

本產品包含 Eclipse Project (https://www.eclipse.org/) 所開發的軟體。

Java 和所有以 Java 為基礎的商標及標誌是 Oracle 及/或其子公司的商標或註冊商標。

產品編號:

(1P) P/N: