9.2

IBM MQ in Containern

Hinweis

Vor Verwendung dieser Informationen und des darin beschriebenen Produkts sollten die Informationen unter „Bemerkungen” auf Seite 167 gelesen werden.

Diese Ausgabe bezieht sich auf Version 9 Release 2 von IBM® MQ und alle nachfolgenden Releases und Modifikationen, bis dieser Hinweis in einer Neuausgabe geändert wird.

Wenn Sie Informationen an IBMsenden, erteilen Sie IBM ein nicht ausschließliches Recht, die Informationen in beliebiger Weise zu verwenden oder zu verteilen, ohne dass eine Verpflichtung für Sie entsteht.

Inhaltsverzeichnis

Multi IBM MQ in Containern und IBM Cloud Pak for Integration

Mit Containern können Sie einen IBM MQ-Warteschlangenmanager oder eine IBM MQ-Clientanwendung mit allen Abhängigkeiten in eine standardisierte Einheit für die Softwareentwicklung packen.

Sie können IBM MQ mit dem IBM MQ Operator unter Red Hat® OpenShift® ausführen. Dies kann unter Verwendung von IBM Cloud Pak for Integration, IBM MQ Advanced oder IBM MQ Advanced for Developers erfolgen.

Sie können IBM MQ auch in einem selbst erstellten Container ausführen.

MQ Adv. CD Weitere Informationen zu IBM MQ Operator finden Sie unter den folgenden Links.

Multi IBM MQ in Containern planen

Bei der Planung von IBM MQ in Containern sollten Sie die Unterstützung berücksichtigen, die IBM MQ für verschiedene Architekturoptionen bereitstellt, z. B. die Verwaltung der Hochverfügbarkeit und die Sicherung Ihrer Warteschlangenmanager.

Informationen zu diesem Vorgang

Bevor Sie Ihre IBM MQ in Container-Architektur planen, sollten Sie sich mit den grundlegenden IBM MQ-Konzepten (siehe IBM MQ - Technische Übersicht) sowie mit grundlegenden Konzepten von Kubernetes/Red Hat OpenShift (siehe Red Hat OpenShift Container Platform-Architektur) vertraut machen.

Prozedur

- „Verwendung von IBM MQ in Containern auswählen” auf Seite 5.
- „Unterstützung für IBM MQ Operator” auf Seite 6.
- „Unterstützung für die Erstellung eigener Container-Images für den IBM MQ-Warteschlangenmanager” auf Seite 13.
- „Speicheraspekte für IBM MQ Operator” auf Seite 11.
- „Hochverfügbarkeit für IBM MQ in Containern” auf Seite 16.
- „Disaster-Recovery für IBM MQ in Containern” auf Seite 19.
- „Benutzerauthentifizierung und -berechtigung für IBM MQ in Containern” auf Seite 19.

Verwendung von IBM MQ in Containern auswählen

Es gibt mehrere Optionen für die Verwendung von IBM MQ in Containern: Sie können die IBM MQ Operator verwenden, die vorverpackte Containerimages verwendet, oder Sie können Ihre eigenen Images und Ihren eigenen Implementierungscode erstellen.

IBM MQ Operator verwenden

OpenShift CP4I

Wenn Sie die Implementierung in Red Hat OpenShift Container Platform planen, dann möchten Sie wahrscheinlich die IBM MQ Operator verwenden.

IBM MQ Operatorfügt `QueueManager`eine neue angepasste Red Hat OpenShift Container Platform-Ressource hinzu. Der Operator überwacht neue Warteschlangenmanagerdefinitionen und wandelt sie dann in erforderliche Low-Level-Ressourcen wie `StatefulSet`- und `Service`-Ressourcen um. Bei nativer Hochverfügbarkeit kann der Operator auch die komplexe rollierende Aktualisierung von Warteschlangenmana-

gerinstanzen durchführen. Siehe „Hinweise zur Durchführung einer eigenen schrittweisen Aktualisierung eines nativen HA-Warteschlangenmanagers” auf Seite 162.

Einige IBM MQ-Features werden bei der Verwendung von IBM MQ Operator nicht unterstützt. Wenn Sie eine der folgenden Funktionen verwenden möchten, müssen Sie Ihre eigenen Images und Diagramme erstellen:

- REST-APIs für die Verwaltung oder Nachrichtenübermittlung verwenden
- Eine der folgenden MQ-Komponenten verwenden:
  - Managed File Transfer-Agenten und ihre Ressourcen. Sie können jedoch IBM MQ Operator verwenden, um einen oder mehrere Koordinations-, Befehls-oder Agentenwarteschlangenmanager bereitzustellen.
  - AMQP
  - IBM MQ Bridge to Salesforce
  - IBM MQ Bridge to blockchain (nicht unterstützt in Containern)
  - IBM MQ Telemetry Transport (MQTT).
- Anpassen von Optionen, die mit **crtmqm**, **strmqm** und **endmqm**verwendet werden, wie z. B. die Konfiguration von Protokolldateiseiten. Die meisten Optionen können mit einer INI-Datei konfiguriert werden.

Beachten Sie, dass sich die IBM MQ Operator und die Container schnell entwickeln und daher unter Long Term Support-Releases nicht unterstützt werden.

Der IBM MQ Operator enthält sowohl vorgefertigte Container-Images als auch Implementierungscode für die Ausführung unter Red Hat OpenShift Container Platform. Der IBM MQ Operator kann verwendet werden, um das bereitgestellte IBM MQ-Containerimage zu implementieren, oder ein Containerimage, das auf dieses aufgeschichtet is, aber nicht zum Implementieren von angepassten MQ-Containerimages verwendet werden kann.

Eigene Images und eigenen Bereitstellungscode erstellen

Multi

Dies ist die flexibelste Containerlösung, Sie benötigen jedoch umfassende Kenntnisse für die Konfiguration von Containern und der resultierende Container sollte sich in Ihrem "Eigentum" befinden. Wenn Sie Red Hat OpenShift Container Platform nicht verwenden möchten, müssen Sie Ihre eigenen Images und einen eigenen Implementierungscode erstellen.

Es sind Beispiele für die Erstellung Ihrer eigenen Images verfügbar. Weitere Informationen finden Sie unter „Eigenen IBM MQ-Container und Bereitstellungscode erstellen” auf Seite 153.

Zugehörige Konzepte

„Unterstützung für IBM MQ Operator” auf Seite 6
Die IBM MQ Operator wird nur unterstützt, wenn sie unter Red Hat OpenShift Container Platform implementiert wird.

„Unterstützung für die Erstellung eigener Container-Images für den IBM MQ-Warteschlangenmanager” auf Seite 13
IBM MQ stellt Code zum Erstellen eines IBM MQ-Warteschlangenmanagercontainers in GitHub bereit. Basis ist hierbei der Prozess, mit dem IBM einen eigenen unterstützten Container erstellt. Sie können dieses GitHub-Repository verwenden, um die Erstellung Ihrer eigenen Container-Images zu vereinfachen und zu beschleunigen.

OpenShift CP4I CD EUS Unterstützung für IBM MQ Operator

Die IBM MQ Operator wird nur unterstützt, wenn sie unter Red Hat OpenShift Container Platform implementiert wird.

IBM MQ Operator verwendet Images auf der Basis von IBM MQ Continuous Delivery-Releases (CD-Releases), obwohl ein EUS-Release (Extended Update Support, erweiterte Aktualisierungsunterstützung) mit IBM Cloud Pak for Integration verfügbar ist. CD-Releases werden bis zu einem Jahr lang oder für zwei

CD-Releases unterstützt, je nachdem, was länger ist. Long Term Support-Releases von IBM MQsind nicht über IBM MQ Operatorverfügbar. IBM Cloud Pak for Integration 2020.4.1 ist ein Release von Extended Update Support (EUS), das 18 Monate lang unterstützt wird, wenn Sie eine Version von IBM MQverwenden, die als `-eus`markiert ist. Andernfalls wird IBM MQ 9.2 als Continuous Delivery-Release mit dem IBM MQ Operator betrachtet.

IBM MQ Operator verwendet Container-Images, die eine Installation von IBM MQ auf einem Red Hat Universal Base Image (UBI) bereitstellen. Dies umfasst die wichtigsten Linux®-Bibliotheken und Dienstprogramme, die von IBM MQ verwendet werden. Das UBI wird von Red Hat unterstützt, wenn die Ausführung unter Red Hat OpenShift erfolgt.

IBM MQ Operator wird auf den Architekturen `amd64`und `s390x`(z/Linux) unterstützt.

Zugehörige Konzepte

„Unterstützung für die Erstellung eigener Container-Images für den IBM MQ-Warteschlangenmanager" auf Seite 13

IBM MQ stellt Code zum Erstellen eines IBM MQ-Warteschlangenmanagercontainers in GitHub bereit. Basis ist hierbei der Prozess, mit dem IBM einen eigenen unterstützten Container erstellt. Sie können dieses GitHub-Repository verwenden, um die Erstellung Ihrer eigenen Container-Images zu vereinfachen und zu beschleunigen.

OpenShift CP4I CD EUS

Versionsunterstützung für IBM MQ Operator

Eine Zuordnung zwischen unterstützten Versionen von IBM MQ, Red Hat OpenShift Container Platform und IBM Cloud Pak for Integration.

- „Verfügbare IBM MQ-Versionen" auf Seite 7
- „Kompatible Red Hat OpenShift Container Platform-Versionen" auf Seite 8
- „Versionen der IBM Cloud Pak for Integration" auf Seite 8
- „Verfügbare IBM MQ-Versionen in älteren Operatoren" auf Seite 8
- „Kompatible Red Hat OpenShift Container Platform-Versionen für ältere Operatoren" auf Seite 9

Verfügbare IBM MQ-Versionen

Operatorkanal	Operatorversion	Versionen der IBM MQ							
		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	⚠	⚠	→	⚠	⚠	●	●	●

Schlüssel:

● Continuous Delivery Support verfügbar

▯ Extended Update Support verfügbar

→ Nur während der Migration von Extended Update Support-Operand zu einem Continuous Delivery-Operand verfügbar.

⚠ Veraltet. Wenn die Unterstützung von IBM MQ-Releases ausläuft, können sie dennoch weiterhin im Operator konfiguriert werden. Allerdings kommen sie für die Unterstützung nicht mehr infrage und können in zukünftigen Releases entfernt werden.

Siehe „Releaseprotokoll für IBM MQ Operator” auf Seite 21 für vollständige Details zu jeder Version, einschließlich detaillierter Funktionen, Änderungen und Korrekturen in jeder Version.

Kompatible Red Hat OpenShift Container Platform-Versionen

Operatorkanal	Operatorversion	Versionen der Red Hat OpenShift Container Platform[1]				
		4.6	4.7[2]	4.8	4.9	4.10
v1.6	1.6	●	●	●	●	●
v1.7	1.7	●	●	●	●	●
v1.8	1.8	●	●	●	●	●

Schlüssel:

● Continuous Delivery Support verfügbar

▯ Extended Update Support verfügbar

Versionen der IBM Cloud Pak for Integration

Die IBM MQ Operator1.8.x wird als Teil von IBM Cloud Pak for IntegrationVersion 2021.4.1 oder unabhängig davon unterstützt.

Die Verwendung von IBM MQ Operator 1.7.x wird im Rahmen von IBM Cloud Pak for Integration Version 2021.4.1 oder als unabhängige Komponente unterstützt.

Die Verwendung von IBM MQ Operator 1.6.x wird im Rahmen von IBM Cloud Pak for Integration Version 2021.2.1 oder 2021.3.1 oder als unabhängige Komponente unterstützt.

IBM MQ Operator 1.5.x wird nicht länger unterstützt.

IBM MQ Operator 1.4.x wird nicht mehr unterstützt.

IBM MQ Operator 1.3.x wird nicht mehr unterstützt.

IBM MQ Operator 1.2.x wird nicht mehr unterstützt.

Die IBM MQ Operators 1.1.x und 1.0.x werden nicht mehr unterstützt.

Verfügbare IBM MQ-Versionen in älteren Operatoren

Die folgende Tabelle gilt für Versionen von IBM MQ Operator, die nun das "Ende des Lebenszyklus" erreicht haben.

Operatorkanal	Operatorversion	Versionen der IBM MQ							
		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	⚠	⚠						

[1] Für Red Hat OpenShift Container Platform-Versionen gelten eigene Unterstützungsdaten. Weitere Informationen siehe Red Hat OpenShift Container Platform Life Cycle Policy.

[2] IBM MQ Operator ist von IBM Cloud Pak foundational services abhängig. Wenn Sie Red Hat OpenShift Container Platform 4.7 verwenden möchten, müssen Sie zunächst ein Upgrade der Version von IBM Cloud Pak foundational services durchführen.

Operatorkanal	Operatorversion	Versionen der IBM MQ							
		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
v1.3-eus	1.3	⚠	⚠	⚠					
v1.4	1.4	⚠	⚠	→	⚠				
v1.5	1.5	⚠	⚠	→	⚠	⚠			

Schlüssel:

→
Nur während der Migration von Extended Update Support-Operand zu einem Continuous Delivery-Operand verfügbar.

⚠
Veraltet. Wenn die Unterstützung von IBM MQ-Releases ausläuft, können sie dennoch weiterhin im IBM MQ Operator konfiguriert werden; allerdings kommen sie für die Unterstützung nicht mehr infrage.

Siehe „Releaseprotokoll für IBM MQ Operator” auf Seite 21 für vollständige Details zu jeder Version, einschließlich detaillierter Funktionen, Änderungen und Korrekturen in jeder Version.

Kompatible Red Hat OpenShift Container Platform-Versionen für ältere Operatoren

Die folgende Tabelle gilt für Versionen von IBM MQ Operator, die nun das "Ende des Lebenszyklus" erreicht haben.

Operatorkanal	Operatorversion	Versionen der Red Hat OpenShift Container Platform[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	⚠	⚠	⚠	⚠	⚠		
v1.3-eus	1.3			⚠	→	→	→	→
v1.4	1.4			⚠	⚠	⚠	⚠	
v1.5	1.5			⚠	⚠	⚠	⚠	⚠

Schlüssel:

→
Nur während der Migration von Extended Update Support-Operand zu einem Continuous Delivery-Operand verfügbar.

[3] Für Red Hat OpenShift Container Platform-Versionen gelten eigene Unterstützungsdaten. Weitere Informationen siehe Red Hat OpenShift Container Platform Life Cycle Policy.

[4] Red Hat OpenShift Container Platform 4.4 hat das "Ende des Lebenszyklus" erreicht. Weitere Informationen siehe Red Hat OpenShift Container Platform Life Cycle Policy.

[5] Red Hat OpenShift Container Platform 4.5 hat das "Ende des Lebenszyklus" erreicht. Weitere Informationen siehe Red Hat OpenShift Container Platform Life Cycle Policy.

[6] IBM MQ Operator ist von IBM Cloud Pak foundational services abhängig. Wenn Sie Red Hat OpenShift Container Platform 4.7 verwenden möchten, müssen Sie zunächst ein Upgrade der Version von IBM Cloud Pak foundational services durchführen.

⚠

Die IBM MQ Operator-Version hat das "Ende des Lebenszyklus" erreicht, war jedoch bisher in dieser Version von Red Hat OpenShift Container Platform verfügbar.

OpenShift CP4I Abhängigkeiten für IBM MQ Operator

Der IBM MQ Operator ist vom IBM Cloud Pak foundational services-Operator abhängig, der auch den ODLM-Operator installiert (ODLM = IBM Operand Deployment Lifecycle Manager). Diese Operatoren werden bei der Installation des IBM MQ Operator automatisch installiert. Diese abhängigen Operatoren haben einen geringen CPU- und Speicherbedarf und werden unter einigen Umständen zum Bereitstellen zusätzlicher Ressourcen verwendet.

Wenn Sie einen `QueueManager` erstellen, erstellt der IBM MQ Operator ein `OperandRequest` für von diesem benötigte zusätzliche Services. Das `OperandRequest` wird vom ODLM-Operator ausgeführt und installiert und instanziiert ggf. die erforderlichen Services. Welche Services erforderlich sind, wird auf der Basis der Lizenzvereinbarung, die bei der Implementierung des Warteschlangenmanagers akzeptiert wurde, und anhand der angeforderten Warteschlangenmanagerkomponenten festgelegt.

- Wenn Sie eine IBM MQ Advanced- oder IBM MQ Advanced for Developers-Lizenz auswählen, werden keine zusätzlichen Services angefordert. In dem folgenden Fall werden die IBM Cloud Pak foundational services beispielsweise nicht verwendet:

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

- Wenn Sie eine IBM Cloud Pak for Integration-Lizenz auswählen und den Web-Server aktivieren wollen, instanziiert der IBM MQ Operator auch den IAM-Operator (IAM = IBM Identity and Access Management), um Single Sign-on zu aktivieren. Der IAM-Operator ist bereits verfügbar, wenn Sie den IBM Cloud Pak for Integration Operator installiert haben. Beispiel:

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

  Wenn Sie den Web-Server inaktivieren, werden jedoch keine IBM Cloud Pak foundational services angefordert. Beispiel:

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

In älteren Versionen von IBM MQ Operator wurde immer die Installation des IBM Licensing Operator (und der zugehörigen Abhängigkeiten) angefordert, um die Lizenznutzung zu verfolgen. Ab IBM MQ Operator 1.5 wird der Lizenzierungsservice nicht angefordert, und Sie müssen ihn separat anfordern.

Für den IBM MQ Operator sind 1 CPU-Kern und 1 GB Speicher erforderlich. Eine detaillierte Aufgliederung der Hardware- und Softwarevoraussetzungen für die abhängigen Operatoren finden Sie in dem Abschnitt über die Hardwareanforderungen und -empfehlungen für Foundational Services.

Sie können die von Ihren Warteschlangenmanagern genutzte CPU-Anzahl und Speichergröße auswählen. Weitere Informationen finden Sie unter „.spec.queueManager.resources” auf Seite 141.

Zugehörige Verweise

„Lizenzierungsreferenz für mq.ibm.com/v1beta1” auf Seite 132

OpenShift CP4I Für IBM MQ Operator erforderliche Berechtigungen auf Clusterebene

IBM MQ Operator benötigt Berechtigungen auf Clusterebene, um Zulassungswebhooks und Beispiele verwalten und Informationen zu Speicherklassen und Clusterversionen lesen zu können.

Für IBM MQ Operator sind die folgenden Berechtigungen auf Clusterebene erforderlich:

- Berechtigung zur Verwaltung von Zulassungswebhooks. Diese Berechtigung ermöglicht das Erstellen, Abrufen und Aktualisieren bestimmter Webhooks, die für die Erstellung und Verwaltung der vom Operator bereitgestellten Container verwendet werden.
  - API-Gruppen: **admissionregistration.k8s.io**
  - Ressourcen: **validatingwebhookconfigurations**
  - Verben: **create, get, update**
- Berechtigung zum Erstellen und Verwalten von Ressourcen, die bei der Erstellung angepasster Ressourcen in der Red Hat OpenShift-Konsole zum Bereitstellen von Beispielen und Snippets verwendet werden.
  - API-Gruppen: **console.openshift.io**
  - Ressourcen: **consoleyamlsamples**
  - Verben: **create, get, update, delete**
- Berechtigung zum Lesen der Clusterversion. Mit dieser Berechtigung kann der Operator Probleme mit der Clusterumgebung zurückmelden.
  - API-Gruppen: **config.openshift.io**
  - Ressourcen: **clusterversions**
  - Verben: **get, list, watch**
- Berechtigung zum Lesen der Speicherklassen im Cluster. Mit dieser Berechtigung kann der Operator Probleme mit den in Containern ausgewählten Speicherklassen zurückmelden.
  - API-Gruppen: **storage.k8s.io**
  - Ressourcen: **storageclasses**
  - Verben: **get, list**

OpenShift CP4I Kubernetes Speicheraspekte für IBM MQ Operator

IBM MQ Operator wird in zwei Speichermodi ausgeführt:

- **Ephemerer Speicher** wird verwendet, wenn alle Statusinformationen für den Container bei einem Neustart des Containers gelöscht werden können. Dies ist der gängige Modus, wenn Umgebungen für Vorführungen erstellt werden oder die Entwicklung mit eigenständigen Warteschlangenmanagern erfolgt.
- **Persistenter Speicher** ist die gängige Konfiguration für IBM MQ und stellt sicher, dass bei einem Neustart des Containers die bestehende Konfiguration, Protokolle und persistente Nachrichten im erneut gestarteten Container verfügbar sind.

IBM MQ Operator bietet die Möglichkeit, die Speichermerkmale, die sich je nach Umgebung erheblich voneinander unterscheiden können, und den gewünschten Speichermodus anzupassen.

Ephemerer Speicher

IBM MQ ist eine statusabhängige Anwendung und speichert diesen Status auf Platte, um ihn im Falle eines Neustarts wiederherstellen zu können. Wenn der ephemere Speicher verwendet wird, gehen alle Statusinformationen für den Warteschlangenmanager bei einem Neustart verloren. Hierzu zählt:

- Alle Nachrichten

- Status der Kommunikation zwischen den Warteschlangenmanagern (Kanalnachrichtensequenznummern)
- MQ-Cluster-Identität des Warteschlangenmanagers
- Alle Transaktionsstatus
- Gesamte Warteschlangenmanagerkonfiguration
- Alle lokalen Diagnosedaten

Aus diesem Grund müssen Sie überlegen, ob ephemerer Speicher ein geeigneter Ansatz für ein Produktions-, Test- oder Entwicklungsszenario ist. Dies betrifft beispielsweise ein Szenario, in dem alle Nachrichten bekanntermaßen nicht persistent sind und der Warteschlangenmanager nicht Mitglied eines MQ-Clusters ist. Neben dem gesamten Messaging-Status wird bei einem Neustart auch die Konfiguration des Warteschlangenmanagers gelöscht. Um einen vollständig ephemeren Container zu aktivieren, muss die IBM MQ-Konfiguration zum Container-Image selbst hinzugefügt werden (weitere Informationen siehe „Image mit benutzerdefinierten MQSC- und INI-Dateien über die Red Hat OpenShift-CLI erstellen" auf Seite 117). Geschieht dies nicht, muss IBM MQ bei jedem Neustart des Containers konfiguriert werden.

Um IBM MQ beispielsweise mit ephemerem Speicher zu konfigurieren, sollte der Speichertyp des `QueueManager` Folgendes einschließen:

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

Persistenter Speicher

IBM MQ wird normalerweise mit persistentem Speicher ausgeführt, um sicherzustellen, dass der Warteschlangenmanager seine persistenten Nachrichten und die Konfiguration nach einem Neustart beibehält. Deshalb ist dies das Standardverhalten. Da es viele Speicheranbieter gibt, die unterschiedliche Funktionen unterstützen, ist häufig eine Anpassung der Konfiguration erforderlich. Im folgenden Beispiel werden die allgemeinen Felder für die Anpassung der MQ-Speicherkonfiguration in der API v1beta1 beschrieben:

- spec.queueManager.availability steuert den Verfügbarkeitsmodus. Bei Verwendung von `SingleInstance` ist nur `ReadWriteOnce`-Speicher erforderlich, während für `multiInstance` eine Speicherklasse erforderlich ist, die `ReadWriteMany` mit den richtigen Dateisperrungsmerkmalen unterstützt. IBM MQ stellt ein Support Statement und ein Testing Statement bereit. Der Verfügbarkeitsmodus wirkt sich auch auf das Layout des persistenten Datenträgers aus. Weitere Informationen finden Sie im Abschnitt „Hochverfügbarkeit für IBM MQ in Containern" auf Seite 16.
- spec.queueManager.storage steuert die individuellen Speichereinstellungen. Ein Warteschlangenmanager kann für die Verwendung von bis zu vier persistenten Datenträgern konfiguriert werden.

Das folgende Beispiel zeigt ein Snippet einer einfachen Konfiguration für einen Einzel-Instanz-Warteschlangenmanager:

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

Das folgende Beispiel zeigt ein Snippet einer Konfiguration für einen Multi-Instanz-Warteschlangenmanager mit einer vom Standard abweichenden Speicherklasse und mit Dateispeicher, der zusätzliche Gruppen erfordert:

```
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]
```

Anmerkung: Sie können auch ergänzende Gruppen mit einzelnen Warteschlangenmanagern für einzelne Instanzen konfigurieren.

Anmerkung: Sie benötigen keine gemeinsam genutzten Dateisysteme, wenn Sie native Hochverfügbarkeit verwenden (siehe „Hochverfügbarkeit für IBM MQ in Containern” auf Seite 16). Verwenden Sie insbesondere nicht NFSv3.

Linux Unterstützung für die Erstellung eigener Container-Images für den IBM MQ-Warteschlangenmanager

IBM MQ stellt Code zum Erstellen eines IBM MQ-Warteschlangenmanagercontainers in GitHub bereit. Basis ist hierbei der Prozess, mit dem IBM einen eigenen unterstützten Container erstellt. Sie können dieses GitHub-Repository verwenden, um die Erstellung Ihrer eigenen Container-Images zu vereinfachen und zu beschleunigen.

Der Code wird im GitHub-Repository 'mq-container' hier bereitgestellt: https://github.com/ibm-messaging/mq-container. Die Bereitstellung unterliegt einer Lizenz für Apache 2.0 mit Unterstützung durch die Community.

Das Repository verwendet nicht die standardmäßigen Linux-RPM-Pakete, sondern das komprimierte Paket für Containerbereitstellungen. Der Vorteil bei diesem Ansatz besteht darin, dass eine Ausführung in sichereren Containerumgebungen möglich ist, ohne dass eskalierte Berechtigungen erforderlich sind. Dies wirkt sich jedoch auf die verfügbaren Sicherheitsoptionen aus, da IBM MQ traditionell eskalierte Berechtigungen für die betriebssystembasierte Authentifizierung verwendet. Bei einer Containerbereitstellung ist die Verwendung einer betriebssystembasierten Authentifizierung normalerweise keine gute Vorgehensweise. Stattdessen können Sie die gegenseitige TLS- oder LDAP-Authentifizierung verwenden. Bei IBM MQ Advanced for Developers können Sie auch die dateibasierte Authentifizierung verwenden, die Ihren Benutzern einen schnellen Einstieg ermöglicht.

Der Warteschlangenmanager für replizierte Daten (RDQM) wird in einer Containerumgebung nicht unterstützt. Bei Verwendung von „Native HA” auf Seite 96 stehen Ihnen ähnliche Funktionen wie bei RDQM zur Verfügung.

Zugehörige Konzepte

„Unterstützung für IBM MQ Operator” auf Seite 6
Die IBM MQ Operator wird nur unterstützt, wenn sie unter Red Hat OpenShift Container Platform implementiert wird.

IBM MQ

Linux Lizenzanmerkungen zur Erstellung eigener IBM MQ-Container-Images

Mit Lizenzanmerkungen können Sie zur Nutzungsüberwachung statt der Grenzwerte der zugrunde liegenden Maschine die für den Container definierten Grenzwerte verwenden. Hierzu konfigurieren Sie Ihre Clients so, dass der Container mit bestimmten Anmerkungen bereitgestellt wird, die von IBM License Service zur Nutzungsüberwachung verwendet werden.

Bei der Bereitstellung eines selbst erstellten IBM MQ-Container-Images wird in der Regel nach einer von zwei gängigen Lizenzierungsmethoden vorgegangen:

- Lizenzierung der gesamten Maschine, auf der der Container ausführt wird.
- Lizenzierung des Containers entsprechend der zugehörigen Grenzwerte.

Beide Optionen sind für Clients möglich. Weitere Details finden Sie auf der Seite IBM Container Licenses auf Passport Advantage.

Wenn der IBM MQ-Container auf Basis der für den Container geltenden Grenzwerte lizenziert werden soll, muss für die Nutzungsüberwachung IBM License Service installiert werden. Informationen zu den unterstützten Umgebungen und Installationsanweisungen finden Sie auf der Seite ibm-licensing-operator auf GitHub.

IBM License Service wird für das Kubernetes-Cluster installiert, in dem der IBM MQ-Container bereitgestellt wird, wobei Pod-Anmerkungen die Nutzung überwachen. Die Clients müssen daher den Pod mit den von IBM License Service verwendeten Anmerkungen bereitstellen. In Abhängigkeit der im Container bereitgestellten Berechtigungen und Funktionen können Sie eine oder mehrere der folgenden Anmerkungen verwenden:

- „IBM MQ Advanced-Container" auf Seite 14
- „IBM MQ Advanced High Availability Replica-Container" auf Seite 14
- „IBM MQ Base-Container" auf Seite 14
- „IBM MQ Base High Availability Replica-Container" auf Seite 15
- „IBM MQ Advanced for Developers-Container" auf Seite 15
- „IBM MQ Advanced-Container mit CP4I-Berechtigung (Produktion)" auf Seite 15
- „IBM MQ Advanced High Availability Replica-Container mit CP4I-Berechtigung (Produktion)" auf Seite 15
- „IBM MQ Advanced-Container mit CP4I-Berechtigung (keine Produktion)" auf Seite 15
- „IBM MQ Advanced High Availability Replica-Container mit CP4I-Berechtigung (keine Produktion)" auf Seite 15
- „IBM MQ Base mit CP4I-Berechtigung (Produktion)" auf Seite 16
- „IBM MQ Base High Availability Replica mit CP4I-Berechtigung (Produktion)" auf Seite 16
- „IBM MQ Base mit CP4I-Berechtigung (keine Produktion)" auf Seite 16
- „IBM MQ Base High Availability Replica mit CP4I-Berechtigung (keine Produktion)" auf Seite 16

IBM MQ Advanced-Container

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

IBM MQ Advanced High Availability Replica-Container

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

IBM MQ Base-Container

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

IBM MQ Base High Availability Replica-Container

```
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-Container

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

IBM MQ Advanced-Container mit CP4I-Berechtigung (Produktion)

```
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 Advanced High Availability Replica-Container mit CP4I-Berechtigung (Produktion)

```
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 Advanced-Container mit CP4I-Berechtigung (keine Produktion)

```
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 Advanced High Availability Replica-Container mit CP4I-Berechtigung (keine Produktion)

```
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 Base mit CP4I-Berechtigung (Produktion)

```
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 Base High Availability Replica mit CP4I-Berechtigung (Produktion)

```
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 Base mit CP4I-Berechtigung (keine Produktion)

```
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 Base High Availability Replica mit CP4I-Berechtigung (keine Produktion)

```
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 Hochverfügbarkeit für IBM MQ in Containern

Es gibt drei Optionen für die hohe Verfügbarkeit mit IBM MQ Operator: **Native HA-Warteschlangenmanager** (mit einem aktiven Replikat und zwei Standby-Replikaten), **Multi-Instanz-Warteschlangenmanager** (ein aktives Standby-Paar, ein gemeinsam genutztes, vernetztes Dateisystem) oder **Single elastischer Warteschlangenmanager** (der einen einfachen Ansatz für HA mit vernetztem Speicher bietet). Die beiden letzteren verlassen sich auf das Dateisystem, um sicherzustellen, dass die wiederherstellbaren Daten verfügbar sind, aber die native HA-Daten nicht. Wenn Sie native HA nicht verwenden, ist die Verfügbarkeit des Dateisystems für die Verfügbarkeit des Warteschlangenmanagers von entscheidender Bedeutung. Wenn Datenwiederherstellung wichtig ist, sollte das Dateisystem Redundanz durch Replikation sicherstellen.

Sie sollten eine separate Verfügbarkeit von **Nachricht** und **Service** in Betracht ziehen. Mit IBM MQ for Multiplatforms wird eine Nachricht auf genau einem Warteschlangenmanager gespeichert. Falls dieser Warteschlangenmanager ausfällt, haben Sie temporär keinen Zugriff auf die dort gespeicherten Nachrichten. Um eine hohe Verfügbarkeit von `Nachrichten` zu erreichen, müssen Sie in der Lage sein, einen Warteschlangenmanager so schnell wie möglich wiederherzustellen. Sie können `Service`-Verfügbarkeit erreichen, indem Sie über mehrere Warteschlangeninstanzen zur Verwendung durch Clientanwendungen verfügen, zum Beispiel mithilfe eines IBM MQ-Uniform-Clusters.

Ein Warteschlangenmanager besteht im Prinzip aus zwei Teilen: den auf der Platte gespeicherten Daten und den aktiven Prozessen, die den Zugriff auf die Daten ermöglichen. Jeder Warteschlangenmanager kann auf einen anderen Kubernetes-Knoten verschoben werden, solange er über dieselben Daten verfügt (die von Kubernetes Persistent Volumes bereitgestellt werden) und weiterhin von Clientanwendungen über das Netz adressierbar ist. In Kubernetes dient ein Service dazu, eine konsistente Netzidentität bereitzustellen.

IBM MQ stützt sich auf die Verfügbarkeit der Daten auf Persistent Volumes. Deshalb ist die Verfügbarkeit des Speichers, der die Persistent Volumes bereitstellt, für die Verfügbarkeit des Warteschlangenmanagers von entscheidender Bedeutung, da IBM MQ nur verfügbar sein kann, wenn der von ihm verwendete Speicher verfügbar ist. Wenn ein Ausfall einer gesamten Verfügbarkeitszone toleriert werden soll, müssen Sie einen Volumeprovider verwenden, der die Plattenschreibvorgänge in einer anderen Zone repliziert.

Native HA-Warteschlangenmanager

Native HA-Warteschlangenmanager sind ab IBM Cloud Pak for Integration 2021.2.1 unter Verwendung von IBM MQ Operator 1.6 oder höher mit IBM MQ 9.2.3 oder höher verfügbar.

Für Warteschlangenmanager mit Native HA werden ein **aktiver** und zwei **Replikat**-Pods von Kubernetes benötigt, die als Teil eines Kubernetes StatefulSet mit drei Replikaten ausgeführt werden, die jeweils über ihren eigenen Satz an Kubernetes Persistent Volumes verfügen. Die Voraussetzungen für IBM MQ für gemeinsam genutzte Dateisysteme gelten auch bei Verwendung eines nativen HA-Warteschlangenmanagers (außer für mietbasierte Sperrung), allerdings müssen Sie kein gemeinsam genutztes Dateisystem verwenden. Sie können Blockspeicher verwenden, mit einem darauf aufgesetzten, geeigneten Dateisystem, z. B. *xfs* oder *ext4*. Die Wiederherstellungszeiten für einen nativen HA-Warteschlangenmanager werden durch folgende Faktoren gesteuert:

1. Wie lange es dauert, bis die Replikatinstanzen einen Ausfall der aktiven Instanz erkennen. Dies ist konfigurierbar.
2. Wie lange dauert es, bis der Bereitschaftstest des Kubernetes-Pods erkennt, dass der bereite Container geändert wurde, und den Netzverkehr umleitet. Dies ist konfigurierbar.
3. Wie lange IBM MQ für die Wiederherstellung der Clientverbindungen braucht.

Weitere Informationen finden Sie unter „Native HA” auf Seite 96

Multi-Instanz-Warteschlangenmanager

Warteschlangenmanager mit mehreren Instanzen enthalten einen **aktiven** und einen **Standby-** Kubernetes -Pod, die als Teil einer statusabhängigen Gruppe von Kubernetes mit genau zwei Replikaten und einer Gruppe persistenter Kubernetes -Datenträger ausgeführt werden. Die Transaktionsprotokolle und Daten des Warteschlangenmanagers werden auf zwei Persistent Volumes mit einem gemeinsam genutzten Dateisystem gespeichert.

Für Multi-Instanz-Warteschlangenmanager ist sowohl der **aktive** als auch der **Standby**-Pod erforderlich, um über gleichzeitigen Zugriff auf den Persistent Volumes zu verfügen. Um dies zu konfigurieren, verwenden Sie Kubernetes Persistent Volumes, für die **`access mode`** (Zugriffsmodus) auf `ReadWriteMany` gesetzt ist. Die Volumes müssen auch die IBM MQ Anforderungen für gemeinsam genutzte Dateisysteme erfüllen, da sich IBM MQ bei der Einleitung einer Warteschlangenmanagerübernahme auf die automatische Freigabe von Dateisperren stützt. IBM MQ erstellt eine Liste der getesteten Dateisysteme.

Die Wiederherstellungszeiten für einen Multi-Instanz-Warteschlangenmanager werden durch folgende Faktoren gesteuert:

1. Wie lange dauert es nach einem Ausfall, bis das gemeinsam genutzte Dateisystem die Sperren freigibt, die ursprünglich von der aktiven Instanz eingerichtet wurden.
2. Wie lange dauert es, bis die Standby-Instanz die Sperren übernommen hat und gestartet wird.

3. Wie lange dauert es, bis der Bereitschaftstest des Kubernetes-Pods erkennt, dass der bereite Container geändert wurde, und den Netzverkehr umleitet. Dies ist konfigurierbar.
4. Wie lange dauert es, bis IBM MQ-Clients Verbindungen wiederhergestellt haben.

Einzelner ausfallsicherer Warteschlangenmanager

Ein einzelner ausfallsicherer Warteschlangenmanager ist eine einzelne Instanz eines Warteschlangenmanagers, der in einem einzelnen Kubernetes-Pod ausgeführt wird, wobei Kubernetes den Warteschlangenmanager überwacht und den Pod nach Bedarf ersetzt.

Die IBM MQ Voraussetzungen für gemeinsam genutzte Dateisysteme gelten auch bei Verwendung eines einzelnen ausfallsicheren Warteschlangenmanagers (außer für mietbasierte Sperrung), allerdings müssen Sie kein gemeinsam genutztes Dateisystem verwenden. Sie können Blockspeicher verwenden, mit einem darauf aufgesetzten, geeigneten Dateisystem, z. B. *xfs* oder *ext4*.

Die Wiederherstellungszeiten für einen einzelnen ausfallsicheren Warteschlangenmanager werden durch folgende Faktoren gesteuert:

1. Wie lange dauert die Ausführung des Livetests und wie viele Fehler toleriert sie. Dies ist konfigurierbar.
2. Wie lange benötigt der Kubernetes Scheduler, um den ausgefallenen Pod auf einem neuen Knoten neu zu planen.
3. Wie lange dauert es, das Container-Image auf den neuen Knoten herunterzuladen. Wenn der Parameter **`imagePullPolicy`** den Wert `IfNotPresent` hat, ist das Image möglicherweise bereits auf dem Knoten verfügbar.
4. Wie lange dauert es, bis die neue Warteschlangenmanagerinstanz gestartet wird.
5. Wie lange dauert es, bis der Bereitschaftstest des Kubernetes-Pods erkennt, dass der Container bereit ist. Dies ist konfigurierbar.
6. Wie lange dauert es, bis IBM MQ-Clients Verbindungen wiederhergestellt haben.

Wichtig:

Obwohl das Muster des einzelnen ausfallsicheren Warteschlangenmanagers einige Vorteile bietet, müssen Sie klären, ob Sie Ihre Verfügbarkeitsziele angesichts der Einschränkungen bei Knotenausfällen erreichen können.

In Kubernetes wird ein ausgefallener Pod in der Regel schnell wiederhergestellt; der Ausfall eines vollständigen Knotens wird jedoch anders gehandhabt. Wenn eine statusabhängige Workload wie IBM MQ mit einem Kubernetes StatefulSetverwendet wird und ein Kubernetes -Masterknoten den Kontakt zu einem Workerknoten verliert, kann nicht festgestellt werden, ob der Knoten ausgefallen ist oder ob er einfach die Netzkonnektivität verloren hat. Deshalb wird Kubernetes in diesem Fall **nicht aktiv**, sondern erst, wenn eins der folgenden Ereignisse eintritt:

1. Der Knoten wird in einem Zustand wiederhergestellt, in dem der Kubernetes-Masterknoten mit ihm kommunizieren kann.
2. Es wird eine Verwaltungsaktion ausgeführt, um den Pod auf dem Kubernetes-Masterknoten explizit zu löschen. Dies stoppt nicht zwangsläufig die Ausführung des Pods, sondern löscht ihn nur aus dem Kubernetes-Speicher. Diese Verwaltungsaktion sollte deshalb mit großer Sorgfalt ausgeführt werden.

Zugehörige Tasks

„Hohe Verfügbarkeit für Warteschlangenmanager mithilfe von IBM MQ Operator konfigurieren” auf Seite 96

Zugehörige Verweise

Hochverfügbarkeitskonfigurationen

OpenShift CP4I Kubernetes Disaster-Recovery für IBM MQ in Containern

Sie müssen überlegen, auf welche Art von Katastrophe Sie sich vorbereiten. In Cloudumgebungen bietet die Einrichtung von Verfügbarkeitszonen einen gewissen Grad an Wiederstandsfähigkeit gegen Katastrophen; sie sind auch viel einfacher umsetzbar. Bei einer ungeraden Anzahl von Rechenzentren (für Quorum) und einer Netzverbindung mit niedriger Latenzzeit könnten Sie möglicherweise einen einzelnen Red Hat OpenShift Container Platform- oder Kubernetes-Cluster mit mehreren Verfügbarkeitszonen betreiben, jede an einem separaten physischen Standort. In diesem Abschnitt geht es um Überlegungen für ein Disaster-Recovery, auf das diese Kriterien nicht zutreffen, d. h., es gibt entweder eine gerade Anzahl von Rechenzentren oder eine Netzverbindung mit einer hohen Latenzzeit.

Für ein Disaster-Recovery ist Folgendes zu beachten:

- Replikation von IBM MQ-Daten (in einer oder mehreren `PersistentVolume`-Ressourcen gehalten) an die Disaster-Recovery-Position
- Neuerstellung des Warteschlangenmanagers mithilfe der replizierten Daten
- Die Netz-ID des Warteschlangenmanagers, die für IBM MQ-Client-Anwendungen und andere Warteschlangenmanager sichtbar ist. Diese ID könnte zum Beispiel ein DNS-Eintrag sein.

Persistente Daten müssen entweder synchron oder asynchron am Disaster-Recovery-Standort repliziert werden. Dies ist in der Regel für den Speicheranbieter spezifisch, kann aber auch mit einem `VolumeSnapshot`erfolgen. Weitere Informationen zu Datenträgermomentaufnahmen finden Sie unter CSI-Datenträgermomentaufnahmen.

Bei der Wiederherstellung nach einer Katastrophe müssen Sie die Warteschlangenmanagerinstanz mithilfe der replizierten Daten im neuen Kubernetes-Cluster erneut erstellen. Wenn Sie IBM MQ Operatorverwenden, benötigen Sie die YAML-Datei für `QueueManager`sowie die YAML-Datei für andere unterstützende Ressourcen wie `ConfigMap`oder `Secret`.

Zugehörige Informationen

ha_for_ctr.dita

Benutzerauthentifizierung und -berechtigung für IBM MQ in Containern

IBM MQ können für die Verwendung von LDAP-Benutzern und -Gruppen konfiguriert werden. Alternativ können Sie Benutzer und Gruppen des lokalen Betriebssystems innerhalb des Container-Images verwenden. Der IBM MQ Operator lässt den Benutzer von Betriebssystembenutzern und -gruppen aufgrund von Sicherheitsbedenken nicht zu.

In einer aus Containern bestehenden Multi-Tenant-Umgebung gelten üblicherweise Integritätsbedingungen für die Sicherheit, um potenzielle Sicherheitsprobleme zu verhindern, z. B.:

- **Verhinderung der Verwendung des Benutzers "root" innerhalb eines Containers**
- **Erzwingung der Verwendung einer randomisierten UID**. In Red Hat OpenShift Container Platform wird beispielsweise in der Standardeinstellung für `SecurityContextConstraints` (`restricted`) für jeden Container eine randomisierte Benutzer-ID verwendet.
- **Verhinderung der Verwendung von Berechtigungseskalationen**. IBM MQ unter Linux nutzt die Berechtigungseskalation, um die Kennwörter von Benutzern zu überprüfen — es wird mithilfe eines "setuid"-Programms zum Benutzer "root", um diese Operation auszuführen.

OpenShift CP4I Um die Einhaltung dieser Sicherheitsmaßnahmen zu gewährleisten, lässt die IBM MQ Operator die Verwendung von IDs, die in den Betriebssystembibliotheken innerhalb eines Containers definiert sind, nicht zu. Im Container ist keine `mqm`-Benutzer-ID oder -Gruppe definiert. Bei Verwendung von IBM MQ in IBM Cloud Pak for Integration und Red Hat OpenShift müssen Sie Ihren Warteschlangenmanager so konfigurieren, dass LDAP für die Benutzerauthentifizierung und -berechtigung verwendet wird. Informationen zur Konfiguration von IBM MQ zu diesem Zweck finden Sie in den Abschnitten Verbindungsauthentifizierung: Benutzerrepositorys und LDAP-Berechtigung.

Multi Skalierbarkeit und Leistung für IBM MQ in Containern planen

In den meisten Fällen ist die Skalierung und Leistung von IBM MQ in Containern mit IBM MQ for Multiplatforms identisch. Es gibt jedoch einige zusätzliche Grenzwerte, die von der Containerplattform auferlegt werden können.

Informationen zu diesem Vorgang

Berücksichtigen Sie bei der Planung der Skalierbarkeit und Leistung für IBM MQ in Containern die folgenden Optionen:

Prozedur

- **Begrenzen Sie die Anzahl der Threads und Prozesse.**

  IBM MQ verwendet Threads zur Verwaltung des gemeinsamen Zugriffs. In Linuxwerden Threads als Prozesse implementiert, sodass Sie Grenzwerte feststellen können, die von der Containerplattform oder dem Betriebssystem auf die maximale Anzahl von Prozessen angewendet werden. In Red Hat OpenShift Container Platformgibt es einen Standardgrenzwert von 4096 Prozessen pro Container (1024 Prozesse bis OpenShift 4.11). Obwohl dies für die überwiegende Mehrheit der Szenarios angemessen ist, kann es Fälle geben, in denen sich dies auf die Anzahl der Clientverbindungen für einen Warteschlangenmanager auswirken kann.

  Der Prozessgrenzwert in Kubernetes kann von einem Clusteradministrator mithilfe der kubelet-Konfigurationseinstellung **`podPidsLimit`**konfiguriert werden. Weitere Informationen finden Sie unter Grenzwerte und Reservierungen für Prozess-IDs in der Kubernetes . In Red Hat OpenShift Container Platformkönnen Sie auch eine angepasste **`ContainerRuntimeConfig`** -Ressource erstellen, um CRI-O -Parameter zu bearbeiten.

  In Ihrer IBM MQ -Konfiguration können Sie auch die maximale Anzahl Clientverbindungen für einen Warteschlangenmanager festlegen. Informationen zum Anwenden von Grenzwerten auf einen einzelnen Serververbindungskanal und zum Anwenden des Attributs MAXCHANNELS INI auf den gesamten Warteschlangenmanager finden Sie im Abschnitt Grenzwerte für Serververbindungskanäle .

- **Begrenzen Sie die Anzahl der Datenträger.**

  In Cloud-und Containersystemen werden häufig NAS-Datenträger verwendet. Die Anzahl der Datenträger, die an Linux -Knoten angehängt werden können, ist begrenzt. Beispiel: AWS EC2 begrenzt auf maximal 30 Datenträger pro VM. Red Hat OpenShift Container Platform hat einen ähnlichen Grenzwertwie Microsoft Azure und Google Cloud Platform.

  Ein nativer HA-Warteschlangenmanager erfordert einen Datenträger für jede der drei Instanzen und erzwingt die Verteilung von Instanzen auf Knoten. Sie können den Warteschlangenmanager jedoch so konfigurieren, dass er drei Datenträger pro Instanz verwendet (Warteschlangenmanagerdaten, Wiederherstellungsprotokolle und persistente Daten).

- **IBM MQ -Skalierungsverfahren verwenden.**

  Anstelle einer kleinen Anzahl großer Warteschlangenmanager kann es nützlich sein, IBM MQ -Skalierungsverfahren wie IBM MQ -Uniform-Cluster zu verwenden, um mehrere Warteschlangenmanager mit derselben Konfiguration auszuführen. Dies hat den zusätzlichen Vorteil, dass die Auswirkungen eines Neustarts eines einzelnen Containers (z. B. als Teil der Wartung der Containerplattform) verringert werden.

OpenShift CP4I CD EUS IBM MQ unter IBM Cloud Pak for Integration und Red Hat OpenShift verwenden

IBM MQ Operator implementiert und verwaltet IBM MQ als Teil von IBM Cloud Pak for Integration oder eigenständig auf Red Hat OpenShift Container Platform

Prozedur

- „Releaseprotokoll für IBM MQ Operator” auf Seite 21.
- „Migration von IBM MQ auf IBM Cloud Pak for Integration” auf Seite 37.
- „IBM MQ Operator unter Red Hat OpenShift installieren und deinstallieren” auf Seite 61.
- „Upgrade für IBM MQ Operator und Warteschlangenmanager durchführen” auf Seite 74.
- „Warteschlangenmanager mithilfe von IBM MQ Operator implementieren und konfigurieren” auf Seite 82.
- „Betreiben von IBM MQ mittels IBM MQ Operator” auf Seite 120.
- „API-Referenz für IBM MQ Operator” auf Seite 132.

OpenShift CP4I CD EUS Releaseprotokoll für IBM MQ Operator

IBM MQ Operator

IBM MQ Operator 1.8.2

CD

IBM Cloud Pak for Integration Version
IBM Cloud Pak for Integration 2021.4.1

Operatorkanal
`v1.8`

Zulässige Werte für `.spec.version`
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

Versionen der Red Hat OpenShift Container Platform
Red Hat OpenShift Container Platform 4.6 und höher

Versionen der IBM Cloud Pak foundational services
IBM Cloud Pak foundational services 3.8 und höher (v3-Kanal)

Neuerungen

- Reine Sicherheitsaktualisierung auf Basis von IBM MQ Operator 1.8.0.
- Die Sicherheitslücken, die adressiert werden, werden in diesem Sicherheitsbulletin detailliert beschrieben.

IBM MQ Operator 1.8.1

CD

IBM Cloud Pak for Integration Version
IBM Cloud Pak for Integration 2021.4.1

Operatorkanal
`v1.8`

Zulässige Werte für `.spec.version`
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

Versionen der Red Hat OpenShift Container Platform
Red Hat OpenShift Container Platform 4.6 und höher

Versionen der IBM Cloud Pak foundational services
IBM Cloud Pak foundational services 3.8 und höher (v3-Kanal)

Neuerungen

- Reine Sicherheitsaktualisierung auf Basis von IBM MQ Operator 1.8.0.
- Die Sicherheitslücken, die adressiert werden, werden in diesem Sicherheitsbulletin detailliert beschrieben.

IBM MQ Operator 1.8.0

IBM Cloud Pak for Integration Version
IBM Cloud Pak for Integration 2021.4.1

Operatorkanal
`v1.8`

Zulässige Werte für `.spec.version`
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

Versionen der Red Hat OpenShift Container Platform
Red Hat OpenShift Container Platform 4.6 und höher

Versionen der IBM Cloud Pak foundational services
IBM Cloud Pak foundational services 3.8 und höher (v3-Kanal)

Neuerungen

- Fügt Statusbedingungen für veraltete IBM MQ-Versionen hinzu.

Änderungen

- Images wurden aus Docker Hub in IBM Container Registryverschoben.
  - Kunden mit Firewallregeln müssen sie möglicherweise anpassen, um auf die Images unter IBM Container Registryzuzugreifen.
  - Airgap-Kunden erhalten einen Knotenneustart, wenn ein Upgrade auf IBM MQ Operator1.8.0 durchgeführt wird.
- Veraltete Versionen: IBM MQ 9.1.5, 9.2.0 CD, 9.2.1, 9.2.2. Diese Versionen werden möglicherweise nicht durch zukünftige Versionen von IBM MQ Operatorabgeglichen.
- Änderungen an der Lizenzlogik: Kunden, die ein Upgrade auf IBM MQ 9.2.5durchführen, können nur die angegebenen Lizenzen verwenden, um mit IBM MQ 9.2.5zu arbeiten. Siehe „Lizenzierungsreferenz für mq.ibm.com/v1beta1" auf Seite 132.
- Die Sicherheitslücken, die adressiert werden, werden in diesem Sicherheitsbulletin detailliert beschrieben.

IBM MQ Operator 1.7.0

IBM Cloud Pak for Integration Version
IBM Cloud Pak for Integration 2021.4.1

Operatorkanal
`v1.7`

Zulässige Werte für `.spec.version`
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

Versionen der Red Hat OpenShift Container Platform
Red Hat OpenShift Container Platform 4.6 und höher

Versionen der IBM Cloud Pak foundational services
IBM Cloud Pak foundational services 3.8 und höher (v3-Kanal)

Neuerungen

- Fügt IBM MQ 9.2.4 als Continuous-Delivery-Release hinzu

IBM MQ Operator 1.6.0

IBM Cloud Pak for Integration Version
IBM Cloud Pak for Integration 2021.2.1

Operatorkanal
`v1.6`

Zulässige Werte für `.spec.version`
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

Versionen der Red Hat OpenShift Container Platform
Red Hat OpenShift Container Platform 4.6 und höher

Versionen der IBM Cloud Pak foundational services
IBM Cloud Pak foundational services 3.7 und höher (v3-Kanal)

Neuerungen

- Fügt IBM MQ9.2.3 als Continuous Delivery-Release hinzu (`amd64` nur für IBM Cloud Pak for Integration2021.2.1;`amd64` oder `s390x`bei Verwendung einer IBM MQ-Lizenz)
- Neuer Verfügbarkeitstyp für Warteschlangenmanager: Native HA. Für die Produktionsnutzung verfügbar, im Rahmen von IBM Cloud Pak for Integration 2021.2.1.

Änderungen

- IBM MQ Operator 1.6 und höher verwenden die IBM Container Registry anstelle von Docker Hub. Dies bedeutet, dass Sie `CatalogSource`aus `icr.io`overwenden müssen. Siehe „IBM MQ Operator unter Red Hat OpenShift installieren und deinstallieren” auf Seite 61.
- Die rollende Aktualisierung von native HA wartet nicht mehr darauf, dass ein Replikat synchron ist, bevor es auf das nächste Replikat übertragen wird.
- Behebt Problem mit der nativen HA-Affinität für OCP 4.7 und höher.
- Behebt das Problem, wenn CA-signierte Zertifikate mit Native HA verwendet werden.

IBM MQ Operator 1.5.0

IBM Cloud Pak for Integration Version
IBM Cloud Pak for Integration 2021.1.1

Operatorkanal
`v1.5`

Zulässige Werte für `.spec.version`
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

Versionen der Red Hat OpenShift Container Platform
Red Hat OpenShift Container Platform 4.6 und höher

Versionen der IBM Cloud Pak foundational services
IBM Cloud Pak foundational services 3.7 und höher (v3-Kanal)

Neuerungen

- Fügt IBM MQ9.2.2 als Continuous Delivery-Release hinzu (`amd64` nur für IBM Cloud Pak for Integration2021.1.1; `amd64`oder `s390x`bei Verwendung einer IBM MQ-Lizenz)
- Neuer Verfügbarkeitstyp für Warteschlangenmanager: Native HA. Nur zu Evaluierungszwecken verfügbar, als Teil von IBM Cloud Pak for Integration 2021.1.1.

- Integration mit Red Hat OpenShift Container PlatformCluster Monitoring for Prometheus-Metriken durch Bereitstellung einer `ServiceMonitor`-Ressource

Änderungen

- Der IBM Lizenzierungsoperator wird nicht mehr standardmäßig erstellt, wenn Sie einen Warteschlangenmanager erstellen.
- Aktualisierungen von Multi-Instanz-Warteschlangenmanagern werden jetzt in einem rollierenden Verfahren durchgeführt. Im Rahmen dieser Änderung wurde eine Kubernetes-Startsonde eingeführt, die sich auf die Werte auswirkt, die bei der Konfiguration der Liveness-Sonde verwendet werden. Der Starttest wird sofort gestartet und wartet darauf, dass der Warteschlangenmanager erfolgreich gestartet wird. Wenn der Starttest zu einem beliebigen Zeitpunkt innerhalb dieser Wartezeit durchgeführt wird, starten dann die Testmonitore und die Bereitschaftssonden. Wenn Sie zuvor einen Warteschlangenmanager hatten, der nur langsam gestartet werden konnte, haben Sie möglicherweise die Einstellung `initialDelaySeconds`für den Aktivitätsprüfung erhöht. In diesem Fall sollten Sie `initialDelaySeconds`jetzt auf die frühere Einstellung zurücksetzen.
- Für `CustomResourceDefinition`wird ein Upgrade von `apiextensions.k8s.io/v1beta1`auf `apiextensions.k8s.io/v1`durchgeführt

Bekannte Probleme und Einschränkungen

- Erfordert IBM Cloud Pak foundational services 3.7, das eine inkompatible Änderung in der Komponente Identity and Access Management (IAM) enthält. Wenn Sie Warteschlangenmanager haben, die eine IBM Cloud Pak for Integration-Lizenz verwenden, ist nach diesem Upgrade ein Neustart des Warteschlangenmanagers erforderlich, um auf die Webkonsole zugreifen zu können, und Sie werden auch sehen wir andere Fehler in der Webkonsole geloggt werden. Sie können diese Fehler beheben, indem Sie ein Upgrade auf den neuesten Wert von `.spec.version`für Ihre ausgewählte IBM MQ-Version durchführen, nachdem das Operator-Upgrade abgeschlossen ist.
- Die rollierende Aktualisierung wird nicht automatisch gestartet, wenn Sie ein Upgrade der MQ-Version durchführen. Sie müssen die Pods manuell löschen.

IBM MQ Operator 1.4.0

IBM Cloud Pak for Integration Version
IBM Cloud Pak for Integration 2020.4.1 (IBM MQ Operator 1.4.0 ist ein CD-Release ohne Anspruch auf Extended Update Support)

Operatorkanal
`v1.4`

Zulässige Werte für `.spec.version`
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

Versionen der Red Hat OpenShift Container Platform
Red Hat OpenShift Container Platform 4.6 und höher

Neuerungen

- IBM MQ 9.2.1 wurde als Continuous Delivery (CD)-Release hinzugefügt.
- Sie können jetzt die Erstellung der Standardroute des Warteschlangenmanagers verhindern, indem Sie `.spec.queueManager.route.enabled`auf `false`setzen

Bekannte Probleme und Einschränkungen

- Beim Aktualisieren eines `QueueManagers` mit dem Verfügbarkeitstyp `MultiInstance`werden beide Pods sofort gelöscht. Sie sollten beide schnell von Red Hat OpenShift Container Platform erneut gestartet werden.

IBM MQ Operator 1.3.8 (EUS)

IBM Cloud Pak for Integration Version
IBM Cloud Pak for Integration 2020.4.1

Operatorkanal
`v1.3-eus`

Zulässige Werte für `.spec.version`
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

Versionen der Red Hat OpenShift Container Platform
Nur Red Hat OpenShift Container Platform 4.6

Versionen der IBM Cloud Pak foundational services
IBM Cloud Pak foundational services 3.6 (`stable-v1`-Kanal)

Neuerungen

- Fügt neue Operandenversion 9.2.0.6-r3-eushinzu.
- Die Sicherheitslücken, die adressiert werden, werden in diesem Sicherheitsbulletin detailliert beschrieben.

IBM MQ Operator 1.3.7 (EUS)

EUS

IBM Cloud Pak for Integration Version
IBM Cloud Pak for Integration 2020.4.1

Operatorkanal
`v1.3-eus`

Zulässige Werte für `.spec.version`
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

Versionen der Red Hat OpenShift Container Platform
Nur Red Hat OpenShift Container Platform 4.6

Versionen der IBM Cloud Pak foundational services
IBM Cloud Pak foundational services 3.6 (`stable-v1`-Kanal)

Neuerungen

- Neue Operandenversion 9.2.0.6-r2-eushinzugefügt.
- Die Sicherheitslücken, die adressiert werden, werden in diesem Sicherheitsbulletin detailliert beschrieben.

IBM MQ Operator 1.3.6 (EUS)

EUS

IBM Cloud Pak for Integration Version
IBM Cloud Pak for Integration 2020.4.1

Operatorkanal
`v1.3-eus`

Zulässige Werte für `.spec.version`
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

Versionen der Red Hat OpenShift Container Platform
Nur Red Hat OpenShift Container Platform 4.6

Versionen der IBM Cloud Pak foundational services
IBM Cloud Pak foundational services 3.6 (`stable-v1`-Kanal)

Neuerungen

- Neue Operandenversion 9.2.0.6-r1-eushinzugefügt.
- Die Sicherheitslücken, die adressiert werden, werden in diesem Sicherheitsbulletin detailliert beschrieben.

IBM MQ Operator 1.3.5 (EUS)

IBM Cloud Pak for Integration Version
IBM Cloud Pak for Integration 2020.4.1

Operatorkanal
`v1.3-eus`

Zulässige Werte für `.spec.version`
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

Versionen der Red Hat OpenShift Container Platform
Nur Red Hat OpenShift Container Platform 4.6

Versionen der IBM Cloud Pak foundational services
IBM Cloud Pak foundational services 3.6 (`stable-v1`-Kanal)

Neuerungen

- Fügt die neue Operandenversion 9.2.0.5-r3-eushinzu.
- Die Sicherheitslücken, die adressiert werden, werden in diesem Sicherheitsbulletin detailliert beschrieben.

IBM MQ Operator 1.3.4 (EUS)

IBM Cloud Pak for Integration Version
IBM Cloud Pak for Integration 2020.4.1

Operatorkanal
`v1.3-eus`

Zulässige Werte für `.spec.version`
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

Versionen der Red Hat OpenShift Container Platform
Nur Red Hat OpenShift Container Platform 4.6

Versionen der IBM Cloud Pak foundational services
IBM Cloud Pak foundational services 3.6 (`stable-v1`-Kanal)

Neuerungen

- Neue Operandenversion 9.2.0.5-r2-eus hinzugefügt
- Die Sicherheitslücken, die adressiert werden, werden in diesem Sicherheitsbulletin detailliert beschrieben.

IBM MQ Operator 1.3.3 (EUS)

IBM Cloud Pak for Integration Version
IBM Cloud Pak for Integration 2020.4.1

Operatorkanal
`v1.3-eus`

Zulässige Werte für `.spec.version`
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

Versionen der Red Hat OpenShift Container Platform
Nur Red Hat OpenShift Container Platform 4.6

Versionen der IBM Cloud Pak foundational services
IBM Cloud Pak foundational services 3.6 (`stable-v1`-Kanal)

Neuerungen

- Fügt neue Operandenversion 9.2.0.5-r1-eus hinzu
- Die Sicherheitslücken, die adressiert werden, werden in diesem Sicherheitsbulletin detailliert beschrieben.

IBM MQ Operator 1.3.2 (EUS)

IBM Cloud Pak for Integration Version
IBM Cloud Pak for Integration 2020.4.1

Operatorkanal
`v1.3-eus`

Zulässige Werte für `.spec.version`
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

Versionen der Red Hat OpenShift Container Platform
Nur Red Hat OpenShift Container Platform 4.6

Versionen der IBM Cloud Pak foundational services
IBM Cloud Pak foundational services 3.6 (`stable-v1`-Kanal)

Neuerungen

- Fügt neue Operandenversion 9.2.0.4-r1-eus hinzu

IBM MQ Operator 1.3.1 (EUS)

IBM Cloud Pak for Integration Version
IBM Cloud Pak for Integration 2020.4.1

Operatorkanal
`v1.3-eus`

Zulässige Werte für `.spec.version`
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

Versionen der Red Hat OpenShift Container Platform
Nur Red Hat OpenShift Container Platform 4.6

Versionen der IBM Cloud Pak foundational services
IBM Cloud Pak foundational services 3.6 (`stable-v1`-Kanal)

Neuerungen

- Fügt neue Operandenversion 9.2.0.2-r1-eus hinzu

IBM MQ Operator 1.3.0 (EUS)

IBM Cloud Pak for Integration Version
IBM Cloud Pak for Integration 2020.4.1

Operatorkanal
`v1.3-eus`

Zulässige Werte für `.spec.version`
9.1.5.0-r2, 9.2.0.0-r1, 9.2.0.0-r2, 9.2.0.1-r1-eus

Versionen der Red Hat OpenShift Container Platform
Nur Red Hat OpenShift Container Platform 4.6

Versionen der IBM Cloud Pak foundational services
IBM Cloud Pak foundational services 3.6 (`stable-v1`-Kanal)

Neuerungen

- Extended Update Support (EUS) wird für `.spec.version`-Felder angeboten, die mit `-eus`enden, wenn eine IBM Cloud Pak for Integration-Lizenz verwendet wird
- Fügt eine neue Methode zum Festlegen von Bezeichnungen und Anmerkungen für die `QueueManager`-Ressource mit `.spec.labels`und `.spec.annotations`hinzu

Änderungen

- Die Fehlerbehandlung beim Wechsel von einer einzelnen zu mehreren Instanzen wurde verbessert.
- Verbesserungen bei der Darstellung der `QueueManager`-Eigenschaften im IBM Cloud Pak for Integration Platform Navigatorund im "Formularansicht" der Red Hat OpenShift Container Platform-Webkonsole
- Korrigiert die Standardlizenzmetrik, wenn eine IBM Cloud Pak for Integration-Lizenz verwendet wird, auf `VirtualProcessorCore`
- Behebt die Registerkarte **Ressourcen** für `QueueManager`in der Red Hat OpenShift Container Platform-Webkonsole, auf der jetzt die von IBM MQ Operatorverwalteten Ressourcen für diesen Warteschlangenmanager korrekt angezeigt werden

Bekannte Probleme und Einschränkungen

- Beim Aktualisieren eines `QueueManagers` mit dem Verfügbarkeitstyp `MultiInstance`werden beide Pods sofort gelöscht. Sie sollten beide schnell von Red Hat OpenShift Container Platform erneut gestartet werden.

IBM MQ Operator 1.2.0

CD

IBM Cloud Pak for Integration Version
IBM Cloud Pak for Integration 2020.3.1

Operatorkanal
`v1.2`

Zulässige Werte für `.spec.version`
9.1.5.0-r2, 9.2.0.0-r1, 9.2.0.0-r2

Versionen der Red Hat OpenShift Container Platform
Red Hat OpenShift Container Platform 4.4 und höher

Neuerungen

- Unterstützung für z/Linux wurde hinzugefügt.
- Fügt detailliertere Statusbedingungen zur Ressource `QueueManager`hinzu. Weitere Informationen finden Sie in folgenden Abschnitten „Statusbedingungen der Ressource 'QueueManager' (mq.ibm.com/v1beta1)” auf Seite 151
- Weitere Laufzeitprüfungen zur Verhinderung ungültiger Speicherklassen wurden hinzugefügt. Weitere Informationen finden Sie unter „Laufzeit-Webhook-Prüfungen inaktivieren” auf Seite 119
- Vereinfachung der Benutzererfahrung für Warteschlangenmanager mit mehreren Instanzen: Dies kann jetzt mit nur einer Eigenschaft (`.spec.queueManager.availability.type`) in der `QueueManager`-Ressource ausgewählt werden
- Vereinfachung der Auswahl einer vom Standard abweichenden Speicherklasse durch Einführung der Eigenschaft `.spec.queueManager.storage.defaultClass`in der `QueueManager`-Ressource

Änderungen

- Verbesserungen bei der Darstellung der `QueueManager`-Eigenschaften im IBM Cloud Pak for Integration Platform Navigatorund im "Formularansicht" der Red Hat OpenShift Container Platform-Webkonsole
- Bei Verfügbarkeit einer aktualisierten Warteschlangenmanagerversion wird diese nun in IBM Cloud Pak for Integration Platform Navigator gekennzeichnet.

IBM MQ Operator 1.1.0

IBM Cloud Pak for Integration Version
IBM Cloud Pak for Integration 2020.2.1

Operatorkanal
`v1.1`

Zulässige Werte für `.spec.version`
9.1.5.0-r2, 9.2.0.0-r1

Versionen der Red Hat OpenShift Container Platform
Red Hat OpenShift Container Platform 4.4 und höher

Neuerungen

- IBM MQ Advanced 9.2.0 wurde als Continuous Delivery (CD)-Release hinzugefügt.
- Eine Funktion zur Angabe von INI- und MQSC-Informationen in einer ConfigMap oder einem Secret wurde hinzugefügt.
- Bei Verwendung der Red Hat OpenShift Container Platform-Webkonsole wird der Schemanavigator aktiviert.

Änderungen

- Behebt das Problem mit der Netzrichtlinie, das sich auf Red Hat OpenShift in IBM Cloud auswirkt.
- Verbesserungen beim Validieren von Webhooks, um ungültige Kombinationen von Einstellungen in `QueueManager`-Ressourcen zu vermeiden

IBM MQ Operator 1.0.0

IBM Cloud Pak for Integration Version
IBM Cloud Pak for Integration 2020.2.1

Operatorkanal
`v1.0`

Zulässige Werte für `.spec.version`
9.1.5.0-r2

Versionen der Red Hat OpenShift Container Platform
Red Hat OpenShift Container Platform 4.4 und höher

Neuerungen

- Ursprüngliche Version des Operators, Einführung in die `mq.ibm.com/v1beta1`-API

Images des WS-Manager-Containers für die Verwendung mit dem IBM MQ Operator

9.2.5.0-r3

Erforderliche Operatorversion
1.8.2 oder höher

Unterstützte Architekturen
amd64, s390x

Bilder

- `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`

Neuerungen

- Neuerungen in IBM MQ 9.2.5

Änderungen

- Änderungen in IBM MQ 9.2.5
- Basiert auf Red Hat Universal Base Image 8.6-751

9.2.5.0-r2

Erforderliche Operatorversion
1.8.1 oder höher

Unterstützte Architekturen
amd64, s390x

Bilder

- `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`

Neuerungen

- Neuerungen in IBM MQ 9.2.5

Änderungen

- Änderungen in IBM MQ 9.2.5
- Basiert auf Red Hat Universal Base Image 8.5-240.1648458092

9.2.5.0-r1

Erforderliche Operatorversion
1.8.0 oder höher

Unterstützte Architekturen
amd64, s390x

Bilder

- `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`

Neuerungen

- Neuerungen in IBM MQ 9.2.5

Änderungen

- Änderungen in IBM MQ 9.2.5
- Ungültige Option `Ferne Warteschlangenmanager` wurde jetzt aus IBM MQ Consoleentfernt

- Basiert auf Red Hat Universal Base Image 8.5-240

9.2.4.0-r1

Erforderliche Operatorversion
1.7.0 oder höher

Unterstützte Architekturen
amd64, s390x

Bilder

- `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`

Neuerungen

- Neuerungen in IBM MQ 9.2.4

Änderungen

- Änderungen in IBM MQ 9.2.4
- Basiert auf Red Hat Universal Base Image 8.5-204

9.2.3.0-r1

Erforderliche Operatorversion
1.6.0 oder höher

Unterstützte Architekturen
amd64, s390x

Bilder

- `cp.icr.io/cp/ibm-mqadvanced-server-integration:9.2.3.0-r1` (nuramd64)
- `cp.icr.io/cp/ibm-mqadvanced-server:9.2.3.0-r1`
- `docker.io/ibmcom/mq:9.2.3.0-r1`

Neuerungen

- Neuerungen in IBM MQ 9.2.3
- Unterstützung für MQ Native HA für die Produktionsnutzung, wenn dies mit einer IBM Cloud Pak for Integration-Lizenz verwendet wird. Beachten Sie, dass WS-Manager, die native HA unter einer Auswertungslizenz mit IBM MQ 9.2.2 verwenden, nicht auf 9.2.3 aktualisiert werden können. Der Probezeitraum ist beendet.

Änderungen

- Änderungen in IBM MQ 9.2.3
- Basierend auf Red Hat Universal Base Image 8.4-205

9.2.2.0-r1

Erforderliche Operatorversion
1.5.0 oder höher

Unterstützte Architekturen
amd64, s390x

Bilder

- `cp.icr.io/cp/ibm-mqadvanced-server-integration:9.2.2.0-r1` (nuramd64)
- `cp.icr.io/cp/ibm-mqadvanced-server:9.2.2.0-r1`
- `docker.io/ibmcom/mq:9.2.2.0-r1`

Neuerungen

- Neuerungen in IBM MQ 9.2.2
- Unterstützung für MQ Native HA zu Evaluierungszwecken, bei Verwendung mit einer IBM Cloud Pak for Integration-Lizenz

Änderungen

- Änderungen in IBM MQ 9.2.2
- Problem behoben, das beim Herunterfahren eines IBM MQ Advanced for Developers-Warteschlangenmanagers ein FDC verursachte
- Basiert auf Red Hat Universal Base Image 8.3-291

9.2.1.0-r2

Erforderliche Operatorversion
1.5.0 oder höher

Unterstützte Architekturen
amd64, s390x

Bilder

- `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`

Änderungen

- Behebt das Problem mit Single Sign-on mit IBM Cloud Pak foundational services 3.7 und höher.
- Basiert auf Red Hat Universal Base Image 8.3-291

9.2.1.0-r1

Erforderliche Operatorversion
1.4.0 oder höher

Unterstützte Architekturen
amd64, s390x

Bilder

- `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`

Neuerungen

- Neuerungen in IBM MQ 9.2.1
- Verbindungsinformationen für die Standardroute sind in der MQ-Webkonsole verfügbar.

Änderungen

- Änderungen in IBM MQ 9.2.1

- Basiert auf Red Hat Universal Base Image 8.3-230

9.2.0.6-r3-eus

Erforderliche Operatorversion
1.3.8 und künftige Fixpacks

Unterstützte Architekturen
amd64, s390x

Bilder

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

Änderungen

- Enthält IBM MQ 9.2.0 Fix Pack 6. Weitere Informationen finden Sie unter Programmfixliste für IBM MQ Version 9.2 LTS.
- Basiert auf Red Hat Universal Base Image 8.6-941.

9.2.0.6-r2-eus

Erforderliche Operatorversion
1.3.7 und künftige Fixpacks

Unterstützte Architekturen
amd64, s390x

Bilder

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

Änderungen

- Enthält IBM MQ 9.2.0 Fix Pack 6. Weitere Informationen finden Sie unter Programmfixliste für IBM MQ Version 9.2 LTS.
- Basiert auf Red Hat Universal Base Image 8.6-902.

9.2.0.6-r1-eus

Erforderliche Operatorversion
1.3.6 und künftige Fixpacks

Unterstützte Architekturen
amd64, s390x

Bilder

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

Änderungen

- Enthält IBM MQ 9.2.0 Fix Pack 6. Weitere Informationen finden Sie unter Programmfixliste für IBM MQ Version 9.2 LTS.
- Basiert auf Red Hat Universal Base Image 8.6-854.

9.2.0.5-r3-eus

Erforderliche Operatorversion
1.3.5 und künftige Fixpacks

Unterstützte Architekturen
amd64, s390x

Bilder

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

Änderungen

- Enthält IBM MQ 9.2.0 Fix Pack 5. Weitere Informationen hierzu finden Sie im Abschnitt Änderungen in IBM MQ 9.2.0 Fix Pack 5 und in der -Programmfixliste für IBM MQ Version 9.2 LTS.
- Basiert auf Red Hat Universal Base Image 8.6-751.1655117800.

9.2.0.5-r2-eus

EUS

Erforderliche Operatorversion
1.3.4 und künftige Fixpacks

Unterstützte Architekturen
amd64, s390x

Bilder

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

Änderungen

- Enthält IBM MQ 9.2.0 Fix Pack 5. Weitere Informationen finden Sie in den Abschnitten Änderungen in IBM MQ 9.2.0 Fix Pack 5 und Fixliste für IBM MQ Version 9.2 LTS.
- Basiert auf Red Hat Universal Base Image 8.6-751

9.2.0.5-r1-eus

EUS

Erforderliche Operatorversion
1.3.3 und künftige Fixpacks

Unterstützte Architekturen
amd64, s390x

Bilder

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

Änderungen

- Enthält IBM MQ 9.2.0 Fix Pack 5. Weitere Informationen finden Sie in den Abschnitten Änderungen in IBM MQ 9.2.0 Fix Pack 5 und Fixliste für IBM MQ Version 9.2 LTS.
- Basiert auf Red Hat Universal Base Image 8.5-240.1648458092

9.2.0.4-r1-eus

EUS

Erforderliche Operatorversion
1.3.2 und zukünftige Fixpacks

Unterstützte Architekturen
amd64, s390x

Bilder

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

Änderungen

- Enthält IBM MQ 9.2.0 Fix Pack 4. Weitere Informationen finden Sie in den Abschnitten Änderungen in IBM MQ 9.2.0 Fix Pack 4 und Fixliste für IBM MQ Version 9.2 LTS.
- Basiert auf Red Hat Universal Base Image 8.5-204

9.2.0.2-r2-eus

Erforderliche Operatorversion
1.6.0 oder höher

Unterstützte Architekturen
amd64, s390x

Bilder

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

Änderungen

- Behebt das Problem mit Single Sign-on mit IBM Cloud Pak foundational services 3.7 und höher, was nur erforderlich ist, wenn Sie von einem EUS-Release auf eine CD-Version migrieren.
- Basiert auf Red Hat Universal Base Image 8.4-200.1622548483

9.2.0.2-r1-eus

Erforderliche Operatorversion
1.3.1 und zukünftige Fixpacks ; 1.6.0 oder höher

Unterstützte Architekturen
amd64, s390x

Bilder

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

Änderungen

- Operations-Dashboard-Integration verwendet Trace-Agent und Collector-Version 1.0.8
- Enthält IBM MQ 9.2.0 Fix Pack 2. Weitere Informationen finden Sie in den Abschnitten Änderungen in IBM MQ 9.2.0 Fix Pack 2 und Fixliste für IBM MQ Version 9.2 LTS.
- Basiert auf Red Hat Universal Base Image 8.4-200.1622548483

9.2.0.1-r1-eus

Erforderliche Operatorversion
1.3.0 oder höher

Unterstützte Architekturen
amd64, s390x

Bilder

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

Neuerungen

- Nur mit einer IBM Cloud Pak for Integration-Lizenz verfügbar.
- Extended Update Support (EUS) ist für IBM MQ Operator 1.3.x und IBM Common Services 3.6 unter Red Hat OpenShift Container Platform 4.6 verfügbar.

Änderungen

- Enthält IBM MQ 9.2.0 Fix Pack 1. Weitere Informationen finden Sie in den Abschnitten Änderungen in IBM MQ 9.2.0 Fix Pack 1 und Fixliste für IBM MQ Version 9.2 LTS.
- Basiert auf Red Hat Universal Base Image 8.3-201
- Behebt Probleme mit Liveness-Test (`chkmqhealthy`) und Readiness-Test (`chkmqready`) bei Ausführung unter `SecurityContextContraints`, die eine Berechtigungseskalation zulassen.

9.2.0.0-r3

Erforderliche Operatorversion
1.5.0 oder höher

Unterstützte Architekturen
amd64, s390x

Bilder

- `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`

Änderungen

- Basiert auf Red Hat Universal Base Image 8.3-291

9.2.0.0-r2

Erforderliche Operatorversion
1.2.0 oder höher

Unterstützte Architekturen
amd64, s390x

Bilder

- `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`

Neuerungen

- Nun unter z/Linux verfügbar.

Änderungen

- Basiert auf Red Hat Universal Base Image 8.2-349

9.2.0.0-r1

Erforderliche Operatorversion
1.1.0 oder höher

Unterstützte Architekturen
amd64

Bilder

- `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`

Neuerungen

- Neuerungen in IBM MQ 9.2.0

Änderungen

- Änderungen in IBM MQ 9.2.0
- Verwendet das Argument `-ic`für `crtmqm`, um MQSC-Dateien automatisch anzuwenden. Ersetzt vorherige Verwendung von `runmqsc`-Befehlen
- Basiert auf Red Hat Universal Base Image 8.2-301.1593113563

9.1.5.0-r2

CD

Erforderliche Operatorversion
1.0.0 oder höher

Unterstützte Architekturen
amd64

Bilder

- `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`

Änderungen

- Basiert auf Red Hat Universal Base Image 8.2-267

OpenShift V 9.2.1 CD EUS Migration von IBM MQ auf IBM Cloud Pak for Integration

In dieser Gruppe von Themen werden die wichtigsten Schritte zur Migration eines vorhandenen IBM MQ-Warteschlangenmanagers in eine Containerumgebung unter Verwendung des IBM MQ Operator in IBM Cloud Pak for Integration beschrieben.

Informationen zu diesem Vorgang

Bei Clients, die IBM MQ unter Red Hat OpenShift implementieren, können folgende Szenarios unterschieden werden:

1. Erstellung einer neuen IBM MQ-Implementierung in Red Hat OpenShift für neue Anwendungen.
2. Erweiterung eines IBM MQ-Netzes in Red Hat OpenShift für neue Anwendungen in Red Hat OpenShift.
3. Verschiebung einer IBM MQ-Implementierung in Red Hat OpenShift zur weiteren Unterstützung bestehender Anwendungen.

Nur für Szenario 3 müssen Sie Ihre IBM MQ-Konfiguration migrieren. Die übrigen Szenarios werden als neue Implementierungen betrachtet.

Diese Gruppe von Themen konzentriert sich auf Szenario 3 und beschreibt die wichtigsten Schritte zur Migration eines vorhandenen IBM MQ-Warteschlangenmanagers in eine Containerumgebung unter Verwendung des IBM MQ Operator. Aufgrund der Flexibilität und der breiten Verwendung von IBM MQ gibt es mehrere optionale Schritte. Jeder davon enthält einen Abschnitt mit der Überschrift "Muss ich diesen Schritt ausführen?". Indem Sie vorab die Notwendigkeit prüfen, sparen Sie Zeit bei der Migration.

Sie müssen sich auch überlegen, welche Daten migriert werden sollen:

1. Migration von IBM MQ mit derselben Konfiguration, aber ohne in den Warteschlangen vorhandene Nachrichten

2. Migration von IBM MQ mit derselben Konfiguration und vorhandenen Nachrichten

Für eine typische Migration von Version zu Version sind beide Ansätze geeignet. In einem typischen IBM MQ-Warteschlangenmanager befinden sich zum Zeitpunkt der Migration, wenn überhaupt, nur wenige Nachrichten in den Warteschlangen, was Option 1 in vielen Fällen zur geeigneten Option macht. Im Falle einer Migration auf eine Containerplattform wird noch häufiger Option 1 verwendet, um die Komplexität der Migration zu verringern und eine Blau/Grün-Bereitstellung zu ermöglichen. Deshalb konzentrieren sich die Anweisungen auf dieses Szenario.

Ziel dieses Szenarios ist es, einen Warteschlangenmanager in der Containerumgebung zu erstellen, der mit der Definition des vorhandenen Warteschlangenmanagers übereinstimmt. Es müssen dann lediglich die vorhandenen netzgebundenen Anwendungen so rekonfiguriert werden, dass sie auf den neuen Warteschlangenmanager verweisen. Sonstige Konfigurations- oder Anwendungslogik muss nicht geändert werden.

Während des Migrationsvorgangs generieren Sie mehrere Konfigurationsdateien, die auf den neuen Warteschlangenmanager angewendet werden sollen. Um die Verwaltung dieser Dateien zu vereinfachen, sollten Sie ein Verzeichnis erstellen und sie in diesem Verzeichnis generieren.

Vorgehensweise

1. „Prüfen, ob erforderliche Funktionen verfügbar sind" auf Seite 38
2. „Warteschlangenmanagerkonfiguration extrahieren" auf Seite 39
3. Optional: „Optional: Warteschlangenmanagerschlüssel und -zertifikate extrahieren und übernehmen" auf Seite 40
4. Optional: „Optional: LDAP konfigurieren" auf Seite 42
5. Optional: „Optional: Ändern der IP-Adressen und Hostnamen in der IBM MQ-Konfiguration" auf Seite 49
6. „Warteschlangenmanagerkonfiguration für eine Containerumgebung aktualisieren" auf Seite 51
7. „Ziel-HA-Architektur für IBM MQ in Containern auswählen" auf Seite 54
8. „Ressourcen für den Warteschlangenmanager erstellen" auf Seite 54
9. „Neuen Warteschlangenmanager unter Red Hat OpenShift erstellen" auf Seite 56
10. „Neue Containerimplementierung überprüfen" auf Seite 60

OpenShift V 9.2.1 CD EUS Prüfen, ob erforderliche Funktionen verfügbar sind

Der IBM MQ Operator enthält nicht alle Funktionen, die in IBM MQ Advanced verfügbar sind, und Sie müssen überprüfen, ob diese Funktionen nicht erforderlich sind. Andere Funktionen werden teilweise unterstützt und können so rekonfiguriert werden, dass sie mit dem, was im Container verfügbar ist, übereinstimmen.

Vorbereitende Schritte

Dies ist der erste Schritt im Abschnitt „Migration von IBM MQ auf IBM Cloud Pak for Integration" auf Seite 37.

Vorgehensweise

1. Stellen Sie sicher, dass das Zielcontainerimage alle erforderlichen Funktionen enthält.

   Die neuesten Informationen finden Sie im Abschnitt „Verwendung von IBM MQ in Containern auswählen" auf Seite 5.
2. Der IBM MQ Operator verfügt über einen einzigen IBM MQ-Datenverkehrsport, der als Listener bezeichnet wird. Wenn Sie über mehrere Listener verfügen, vereinfachen Sie diese, um einen einzelnen Listener im Container zu verwenden. Da dies kein übliches Szenario ist, wird diese Änderung nicht im Detail dokumentiert.

3. Wenn IBM MQ-Exits verwendet werden, migrieren Sie sie in den Container, indem Sie in das IBM MQ-Exit-Binärdateien Layering durchführen. Dies ist ein fortgeschrittenes Migrationsszenario, auf das an dieser Stelle nicht weiter eingegangen wird. Eine Beschreibung der Schritte finden Sie im Abschnitt „Image mit benutzerdefinierten MQSC- und INI-Dateien über die Red Hat OpenShift-CLI erstellen" auf Seite 117.
4. Wenn Ihr IBM MQ-System ein Hochverfügbarkeitssystem ist, überprüfen Sie die verfügbaren Optionen.

   Weitere Informationen finden Sie unter „Hochverfügbarkeit für IBM MQ in Containern" auf Seite 16.

Nächste Schritte

Sie können jetzt die Warteschlangenmanagerkonfiguration extrahieren.

OpenShift V 9.2.1 CD EUS Warteschlangenmanagerkonfiguration extrahieren

Der Großteil der Konfiguration ist zwischen Warteschlangenmanagern portierbar. Das gilt beispielsweise für die Dinge, mit denen Anwendungen interagieren, wie etwa Definitionen von Warteschlangen, Themen und Kanälen. Verwenden Sie diese Task, um die Konfiguration aus dem vorhandenen IBM MQ-Warteschlangenmanager zu extrahieren.

Vorbereitende Schritte

Bei dieser Task wird davon ausgegangen, dass Sie überprüft haben, dass erforderliche Funktionen verfügbar sind.

Vorgehensweise

1. Melden Sie sich bei der Maschine mit der bestehenden IBM MQ-Installation an.
2. Erstellen Sie eine Sicherungskopie der Konfiguration.

   Führen Sie den folgenden Befehl aus:

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

   Hinweise zur Verwendung dieses Befehls:

   - Mit diesem Befehl wird die Sicherungskopie im Verzeichnis `tmp` gespeichert. Sie können die Sicherung an einer anderen Position speichern, aber in diesem Szenario wird das Verzeichnis `tmp` in nachfolgenden Befehlen vorausgesetzt.
   - Ersetzen Sie *WS_MANAGER_NAME* durch den Namen des Warteschlangenmanagers für Ihre Umgebung. Wenn Sie sich wegen des Werts nicht sicher sind, führen Sie den Befehl **dspmq** aus, um die verfügbaren Warteschlangenmanager auf der Maschine anzuzeigen. Es folgt eine Beispielausgabe des Befehls **dspmq** für einen Warteschlangenmanager mit dem Namen `qm1`:

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

     Der Befehl **dspmq** setzt voraus, dass der IBM MQ-Warteschlangenmanager gestartet ist, andernfalls erhalten Sie folgenden Fehler:

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

     Falls erforderlich, starten Sie den Warteschlangenmanager mit folgendem Befehl:

     ```
     strmqm QMGR_NAME
     ```

Nächste Schritte

Sie können jetzt die Schlüssel und Zertifikate des Warteschlangenmanagers extrahieren und übernehmen.

OpenShift V 9.2.1 CD EUS Optional: Warteschlangenmanagerschlüssel und -zertifikate extrahieren und übernehmen

IBM MQ kann mithilfe von TLS so konfiguriert werden, dass der Datenverkehr in den Warteschlangenmanager verschlüsselt wird. Verwenden Sie diese Task, um zu überprüfen, ob Ihr Warteschlangenmanager TLS verwendet, um Schlüssel und Zertifikate zu extrahieren, und um TLS auf dem migrierten Warteschlangenmanager zu konfigurieren.

Vorbereitende Schritte

Diese Task setzt voraus, dass Sie die Warteschlangenmanagerkonfiguration extrahiert haben.

Informationen zu diesem Vorgang

Muss ich diesen Schritt ausführen?

IBM MQ kann so konfiguriert werden, dass der Datenverkehr in den Warteschlangenmanager verschlüsselt wird. Diese Verschlüsselung erfolgt mithilfe eines Schlüsselrepositorys, das auf dem Warteschlangenmanager konfiguriert ist. Die TLS-Kommunikation wird dann über IBM MQ-Kanäle ermöglicht. Wenn Sie nicht genau wissen, ob Ihre Umgebung dafür konfiguriert ist, überprüfen Sie es mit folgendem Befehl:

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

Wenn keine Ergebnisse gefunden werden, wird TLS nicht verwendet. Dies bedeutet jedoch nicht, dass TLS nicht im migrierten Warteschlangenmanager konfiguriert werden sollte. Es gibt mehrere Gründe, die dafür sprechen könnten, dieses Verhalten zu ändern:

- Das Sicherheitskonzept in der Red Hat OpenShift-Umgebung soll im Vergleich zur vorherigen Umgebung verbessert werden.
- Wenn Sie von außerhalb der Red Hat OpenShift-Umgebung auf den migrierten Warteschlangenmanager zugreifen müssen, ist TLS für die Nutzung der Red Hat OpenShift-Route erforderlich.

Vorgehensweise

1. Extrahieren Sie alle vertrauenswürdigen Zertifikate aus dem vorhandenen Speicher.

   Wenn TLS aktuell auf dem Warteschlangenmanager verwendet wird, sind auf dem Warteschlangenmanager möglicherweise eine Reihe vertrauenswürdiger Zertifikate gespeichert. Diese müssen extrahiert und in den neuen Warteschlangenmanager kopiert werden. Führen Sie einen der folgenden optionalen Schritte aus:

   - Führen Sie das folgende Script auf dem lokalen System aus, um die Extraktion der Zertifikate zu optimieren:

     ```
     #!/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
```

Geben Sie bei der Ausführung des Scripts die Position der IBM MQ-Sicherung als Argument an und die Zertifikate werden extrahiert. Führen Sie beispielsweise folgenden Befehl aus, wenn das Script `extractCert.sh` heißt und sich die IBM MQ-Sicherung an der Position `/tmp/backup.mqsc` befindet:

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

- Alternativ können Sie folgende Befehle in der angegebenen Reihenfolge ausführen:

  a. Ermitteln Sie die Position des TLS-Speichers:

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

  Beispielausgabe:

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

  Dabei befindet sich der Schlüsselspeicher an der Position `/run/runmqserver/tls/key.kdb`.

  b. Fragen Sie auf Basis dieser Positionsinformationen den Schlüsselspeicher ab, um eine Liste der gespeicherten Zertifikate anzuzeigen:

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

  Beispielausgabe:

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

  c. Extrahieren Sie jedes der aufgelisteten Zertifikate. Führen Sie dazu folgenden Befehl aus:

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

  In den oben gezeigten Beispielen entspricht dies Folgendem:

  ```
  runmqckm -cert -extract -db /run/runmqserver/tls/key.kdb -label "CN=cs-ca-certifica
  te,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. Übernehmen Sie einen neuen Schlüssel und ein neues Zertifikat für den Warteschlangenmanager.

   Um TLS auf dem migrierten Warteschlangenmanager zu konfigurieren, generieren Sie einen neuen Schlüssel und ein neues Zertifikat. Diese werden dann während der Implementierung verwendet. In vielen Organisationen bedeutet dies, dass Sie bei Ihrem Sicherheitsteam einen Schlüssel und ein Zertifikat anfordern müssen. In einigen Organisationen ist diese Option nicht verfügbar und es werden selbst signierte Zertifikate verwendet.

   Im folgenden Beispiel wird ein selbst signiertes Zertifikat mit einer Gültigkeitsdauer von 10 Jahren generiert:

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

Es werden zwei neue Dateien erstellt:

- `qmgr.key` ist der private Schlüssel für den Warteschlangenmanager.
- `qmgr.crt` ist das öffentliche Zertifikat.

Nächste Schritte

Sie können jetzt LDAP konfigurieren.

OpenShift V 9.2.1 CD EUS Optional: LDAP konfigurieren

Der IBM MQ Operator kann so konfiguriert werden, dass er mehrere unterschiedliche Sicherheitskonzepte verwendet. In der Regel ist LDAP das effektivste für den Einsatz in einem Unternehmen und LDAP wird auch für dieses Migrationsszenario verwendet.

Vorbereitende Schritte

Diese Task setzt voraus, dass Sie die Warteschlangenmanagerschlüssel und -zertifikate extrahiert und übernommen haben.

Informationen zu diesem Vorgang

Muss ich diesen Schritt ausführen?

Wenn Sie LDAP bereits für die Authentifizierung und Berechtigung verwenden, sind keine Änderungen erforderlich.

Wenn Sie sich nicht sicher sind, ob LDAP verwendet wird, führen Sie folgenden Befehl aus:

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

Beispielausgabe:

```
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
```

Zwei Attribute in der Ausgabe sind von besonderem Interesse:

AUTHTYPE
Wenn dieses Attribut den Wert `IDPWLDAP` hat, verwenden Sie LDAP für die Authentifizierung.

Ist kein Wert oder ein anderer Wert angegeben, ist LDAP nicht konfiguriert. Überprüfen Sie in diesem Fall anhand des Attributs `AUTHORMD`, ob LDAP-Benutzer für die Berechtigung verwendet werden.

AUTHORMD
Wenn dieses Attribut den Wert `OS` hat, verwenden Sie LDAP nicht für die Berechtigung.

Wenn LDAP für die Berechtigung und Authentifizierung verwendet werden sollen, gehen Sie wie folgt vor:

Vorgehensweise

1. Aktualisieren Sie die IBM MQ-Sicherung für den LDAP-Server.
2. Aktualisieren Sie die IBM MQ-Sicherung für LDAP-Berechtigungsinformationen.

OpenShift V 9.2.1 CD EUS *LDAP Teil 1: IBM MQ-Sicherung für den LDAP-Server aktualisieren*

Eine ausführliche Beschreibung der Vorgehensweise zur Einrichtung von LDAP ist nicht Bestandteil dieses Szenarios. Dieser Abschnitt enthält eine Zusammenfassung des Prozesses, ein Beispiel und Verweise auf weitere Informationen.

Vorbereitende Schritte

Diese Task setzt voraus, dass Sie die Warteschlangenmanagerschlüssel und -zertifikate extrahiert und übernommen haben.

Informationen zu diesem Vorgang

Muss ich diesen Schritt ausführen?

Wenn Sie LDAP bereits für die Authentifizierung und Berechtigung verwenden, sind keine Änderungen erforderlich. Wenn Sie sich nicht sicher sind, ob LDAP verwendet wird, lesen Sie den Abschnitt „Optional: LDAP konfigurieren” auf Seite 42.

Die Einrichtung des LDAP-Servers besteht aus zwei Teilen:

1. Definition einer LDAP-Konfiguration
2. Zuordnung der LDAP-Konfiguration zur Warteschlangenmanagerdefinition

Weitere Informationen, die Sie bei dieser Konfiguration unterstützen:

- Benutzerrepository-Übersicht
- Referenzhandbuch für den Befehl AUTHINFO

Vorgehensweise

1. Definieren Sie eine LDAP-Konfiguration.

   Bearbeiten Sie die Datei `backup.mqsc`, um ein neues **AUTHINFO**-Objekt für das LDAP-System zu definieren. Beispiel:

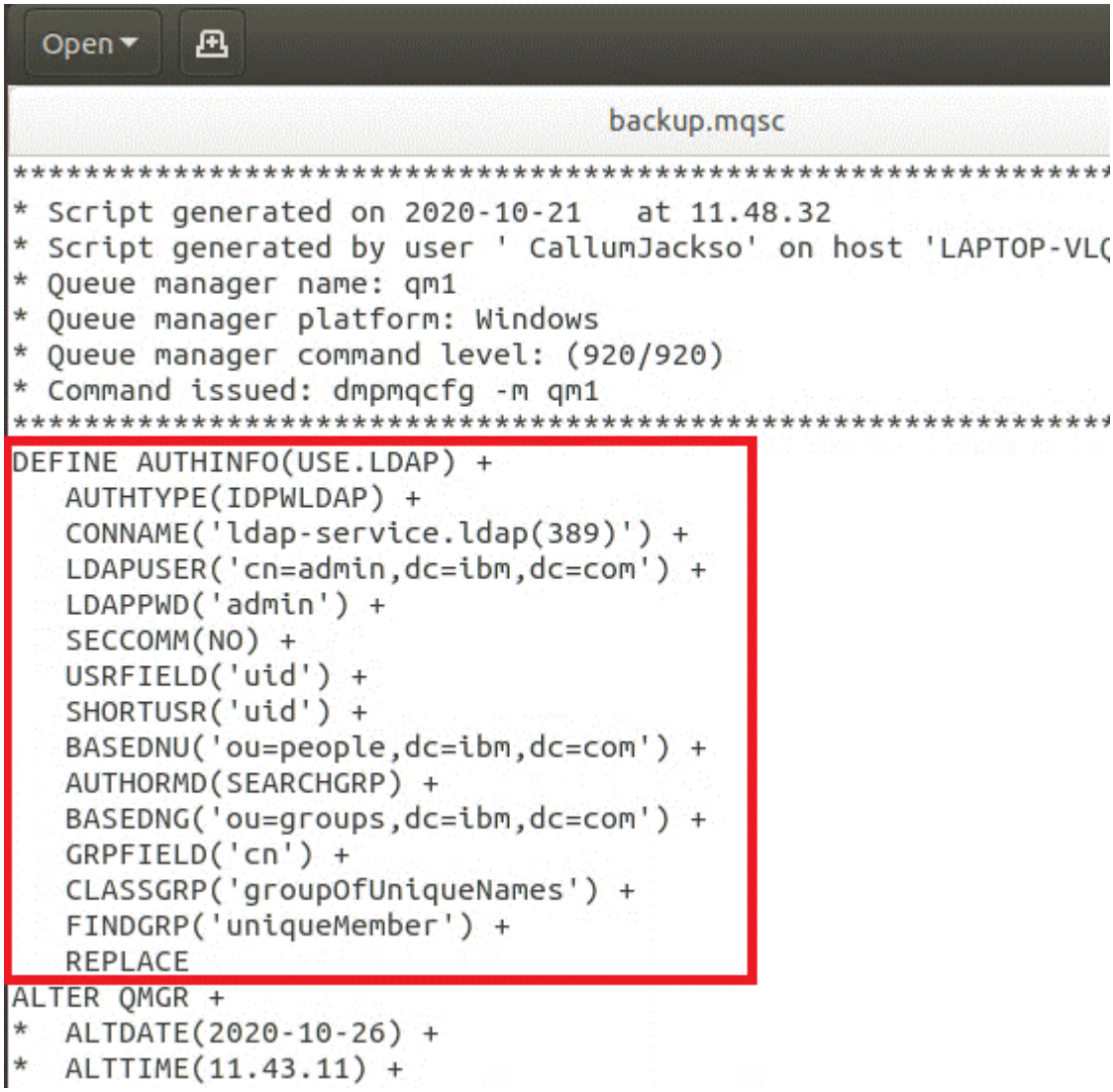
```
DEFINE AUTHINFO(USE.LDAP) +
   AUTHTYPE(IDPWLDAP) +
   CONNAME('ldap-service.ldap(389)') +
   LDAPUSER('cn=admin,dc=ibm,dc=com') +
   LDAPPWD('admin') +
   SECCOMM(NO) +
   USRFIELD('uid') +
   SHORTUSR('uid') +
   BASEDNU('ou=people,dc=ibm,dc=com') +
   AUTHORMD(SEARCHGRP) +
   BASEDNG('ou=groups,dc=ibm,dc=com')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

   Dabei gilt Folgendes:

   - **CONNAME** gibt den Hostnamen und Port des LDAP-Servers an. Wenn aus Gründen der Ausfallsicherheit mehrere Adressen vorhanden sind, können diese mithilfe einer durch Kommas getrennten Liste konfiguriert werden.
   - **LDAPUSER** gibt den definierten Namen des Benutzers an, den IBM MQ bei der Herstellung der Verbindung zu LDAP verwendet, um Benutzerdatensätze abzufragen.

- **LDAPPWD** gibt das Kennwort des Benutzers **LDAPUSER** an.
- **SECCOM** gibt an, ob bei der Kommunikation mit dem LDAP-Server TLS verwendet werden soll. Mögliche Werte:
  - YES: Es wird TLS verwendet und vom IBM MQ-Server ein Zertifikat übergeben.
  - ANON: Es wird TLS verwendet, ohne dass vom IBM MQ-Server ein Zertifikat übergeben wird.
  - NO: TLS wird während der Verbindung nicht verwendet.
- **USRFIELD** gibt das Feld im LDAP-Datensatz an, mit dem der übergebene Benutzername abgeglichen wird.
- **SHORTUSR** gibt ein Feld im LDAP-Datensatz an, das maximal 12 Zeichen lang ist. Der Wert in diesem Feld ist die zugesicherte Identität, wenn die Authentifizierung erfolgreich ist.
- **BASEDNU** gibt den Basis-DN an, der für die Suche in LDAP verwendet werden soll.
- **BASEDNG** gibt den Basis-DN für Gruppen in LDAP an.
- **AUTHORMD** definiert den Mechanismus, der für die Auflösung der Gruppenzugehörigkeit für den Benutzer verwendet wird. Es gibt vier Optionen:
  - OS: Abfrage des Betriebssystems nach den Gruppen, die dem Kurznamen zugeordnet sind
  - SEARCHGRP: Suche in den Gruppeneinträgen in LDAP nach dem authentifizierten Benutzer
  - SEARCHUSR: Suche im Datensatz des authentifizierten Benutzers nach Gruppenzugehörigkeitsinformationen
  - SRCHGRPSN: Suche in den Gruppeneinträgen in LDAP nach dem Kurznamen des authentifizierten Benutzers (definiert durch das Feld SHORTUSR)
- **GRPFIELD** gibt das Attribut im LDAP-Gruppendatensatz an, das einem einfachen Namen entspricht. Falls angegeben, kann dies zur Definition von Berechtigungssätzen verwendet werden.
- **CLASSUSR** gibt die LDAP-Objektklasse für einen Benutzer an.
- **CLASSGRP** gibt die LDAP-Objektklasse für eine Gruppe an.
- **FINDGRP** gibt das Attribut im LDAP-Datensatz an, das der Gruppenzugehörigkeit entspricht.

Der neue Eintrag kann überall in der Datei hinzugefügt werden. Es kann jedoch hilfreich sein, wenn alle neuen Einträge am Anfang der Datei stehen:



2. Ordnen Sie die LDAP-Konfiguration der Warteschlangenmanagerdefinition zu.

   Sie müssen die LDAP-Konfiguration der Warteschlangenmanagerdefinition zuordnen. Direkt unter dem Eintrag `DEFINE AUTHINFO` befindet sich der Eintrag `ALTER QMGR`. Ändern Sie den Eintrag CONNAUTH, sodass er dem neu erstellten AUTHINFO-Namen entspricht. So enthält das vorherige Beispiel etwa die Definition `AUTHINFO(USE.LDAP)`, d. h., der Name lautet `USE.LDAP`. Ändern Sie deshalb `CONNAUTH('SYSTEM.DEFAULT.AUTHINFO.IDPWOS')` in `CONNAUTH('USE.LDAP')`:

Damit der Wechsel zu LDAP sofort erfolgt, rufen Sie einen Befehl `REFRESH SECURITY` auf, indem Sie direkt nach dem Befehl `ALTER QMGR` eine Zeile hinzufügen:

*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
```

Nächste Schritte

Sie können jetzt die IBM MQ-Sicherung für LDAP-Berechtigungsinformationen aktualisieren.

OpenShift V 9.2.1 CD EUS

LDAP Teil 2: IBM MQ-Sicherung für LDAP-Berechtigungsinformationen aktualisieren

IBM MQ stellt differenzierte Berechtigungsregeln für die Steuerung des Zugriffs auf die IBM MQ-Objekte bereit. Wenn Sie die Authentifizierung und Berechtigung für LDAP geändert haben, sind die Berechtigungsregeln möglicherweise ungültig und müssen aktualisiert werden.

Vorbereitende Schritte

Diese Task setzt voraus, dass Sie die Sicherung für den LDAP-Server aktualisiert haben.

Informationen zu diesem Vorgang

Muss ich diesen Schritt ausführen?

Wenn Sie LDAP bereits für die Authentifizierung und Berechtigung verwenden, sind keine Änderungen erforderlich. Wenn Sie sich nicht sicher sind, ob LDAP verwendet wird, lesen Sie den Abschnitt „Optional: LDAP konfigurieren” auf Seite 42.

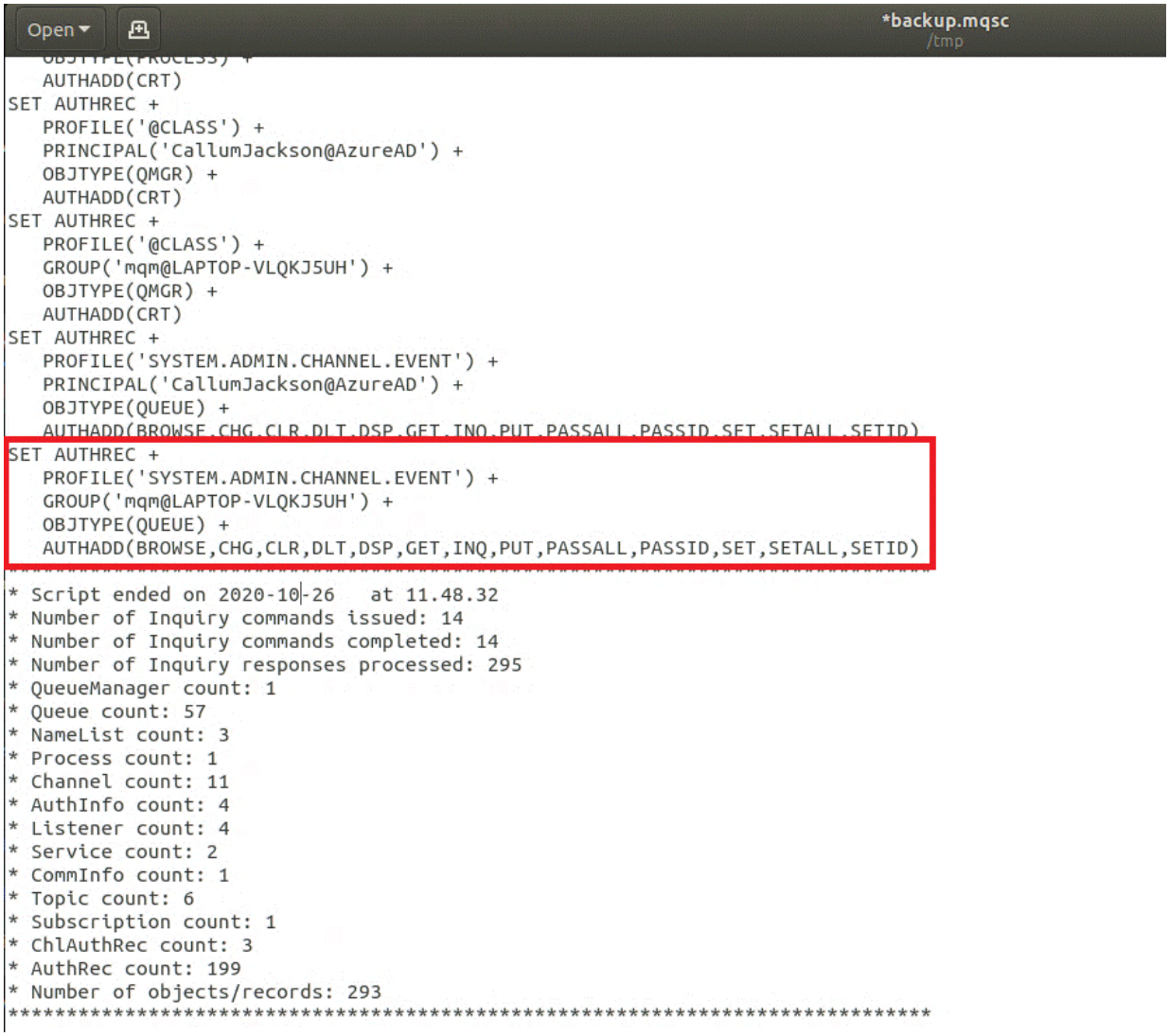
Die Aktualisierung der LDAP-Berechtigungsinformationen besteht aus zwei Teilen:

1. Alle bestehenden Berechtigungen aus der Datei entfernen
2. Neue Berechtigungsinformationen für LDAP definieren

Vorgehensweise

1. Entfernen Sie alle bestehenden Berechtigungen aus der Datei.

   In der Sicherungsdatei am Ende der Datei sollten mehrere Einträge angezeigt werden, die mit SET AUTHRECbeginnen:



Suchen Sie die vorhandenen Einträge und löschen Sie sie. Die einfachste Methode besteht darin, alle vorhandenen SET AUTHREC-Regeln zu entfernen und dann neue Einträge auf Basis der LDAP-Einträge zu erstellen.

2. Definieren Sie neue Berechtigungsinformationen für LDAP.

   Abhängig von der Konfiguration Ihres Warteschlangenmanagers und der Anzahl der Ressourcen und Gruppen kann dies entweder eine zeitaufwendige oder einfache Aktivität sein. Im folgenden Beispiel wird davon ausgegangen, dass Ihr Warteschlangenmanager nur über eine einzige Warteschlange mit dem Namen Q1 verfügt und Sie möchten, dass die LDAP-Gruppe apps auf die Warteschlange zugreifen kann.

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

   Der erste AUTHREC-Befehl fügt die Berechtigung für den Zugriff auf den Warteschlangenmanager hinzu und der zweite erteilt Zugriff auf die Warteschlange. Wenn Zugriff auf eine zweite Warteschlange erforderlich ist, wird ein dritter AUTHREC-Befehl benötigt, es sei denn, Sie haben entschieden, über Platzhalterzeichen einen allgemeineren Zugriff zu erteilen.

   Hier folgt ein weiteres Beispiel. Wenn eine Administratorgruppe (mit dem Namen admins) uneingeschränkten Zugriff auf den Warteschlangenmanager benötigt, fügen Sie folgende Befehle hinzu:

```
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)
```

Nächste Schritte

Sie können jetzt die IP-Adressen und Hostnamen in der IBM MQ-Konfiguration ändern.

OpenShift V 9.2.1 CD EUS Optional: Ändern der IP-Adressen und Hostnamen in der IBM MQ-Konfiguration

In der IBM MQ-Konfiguration können IP-Adressen und Hostnamen angegeben sein. In einigen Situationen können diese beibehalten werden, während sie in anderen Situationen aktualisiert werden müssen.

Vorbereitende Schritte

Diese Task setzt voraus, dass Sie LDAP konfiguriert haben.

Informationen zu diesem Vorgang

Muss ich diesen Schritt ausführen?

Stellen Sie zunächst fest, ob alle IP-Adressen oder Hostnamen angegeben sind, abgesehen von der im vorherigen Abschnitt definierten LDAP-Konfiguration. Führen Sie dazu folgenden Befehl aus:

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

Beispielausgabe:

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

In diesem Beispiel führt die Suche zu drei Ergebnissen. Ein Ergebnis entspricht der zuvor definierten LDAP-Konfiguration. Es kann ignoriert werden, weil der Hostname des LDAP-Servers gleich bleibt. Die anderen beiden Ergebnisse sind leere Verbindungseinträge und können deshalb ebenfalls ignoriert werden. Wenn Sie über keine zusätzlichen Einträge verfügen, können Sie den Rest dieses Abschnitts überspringen.

Vorgehensweise

1. Informieren Sie sich über die zurückgegebenen Einträge.

   IBM MQ kann IP-Adressen, Hostnamen und Ports in vielen Bereichen der Konfiguration einschließen. Diese können in zwei Kategorien eingeteilt werden:

   a. **Position dieses Warteschlangenmanagers**: Positionsinformationen, die dieser Warteschlangenmanager verwendet oder veröffentlicht, die andere Warteschlangenmanager oder Anwendungen innerhalb eines IBM MQ-Netzwerks für die Konnektivität verwenden können.

   b. **Position von Warteschlangenmanagerabhängigkeiten**: Die Positionen anderer Warteschlangenmanager oder Systeme, die diesem Warteschlangenmanager bekannt sein müssen.

   Da es in diesem Szenario nur um die Änderungen an der Konfiguration dieses Warteschlangenmanagers geht, werden nur die Konfigurationsaktualisierungen für Kategorie (a) behandelt. Wenn die Position dieses Warteschlangenmanagers jedoch von anderen Warteschlangenmanagern oder Anwendungen referenziert wird, müssen deren Konfigurationen gegebenenfalls mit der neuen Position dieses Warteschlangenmanagers aktualisiert werden.

   Es gibt zwei Schlüsselobjekte, die Informationen enthalten können, die aktualisiert werden müssen:

   - Listener: Diese stellen die Netzadresse dar, an der IBM MQ empfangsbereit ist.
   - CLUSTER RECEIVER-Kanal: Wenn der Warteschlangenmanager Teil eines IBM MQ-Clusters ist, ist dieses Objekt vorhanden. Es gibt die Netzadresse an, zu der andere Warteschlangenmanager eine Verbindung herstellen können.

2. Geben Sie in der ursprünglichen Ausgabe des Befehls `grep 'CONNAME\|LOCLADDR\|IPADDRV' -B 3 backup.mqsc` an, ob CLUSTER RECEIVER-Kanäle definiert sind. Wenn ja, aktualisieren Sie die IP-Adressen.

   Sie können ermitteln, ob CLUSTER RECEIVER-Kanäle definiert sind, indem Sie in der Originalausgabe nach Einträgen mit CHLTYPE(CLUSRCVR) suchen:

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

   Wenn Einträge vorhanden sind, aktualisieren Sie den CONNAME mit der IBM MQ Red Hat OpenShift-Route. Dieser Wert basiert auf der Red Hat OpenShift-Umgebung und verwendet eine vorhersehbare Syntax:

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

Wenn beispielsweise die Implementierung des Warteschlangenmanagers den Namen `qm1` im Namensbereich `cp4i` hat und *openshift_app_route_hostname* `apps.callumj.icp4i.com` lautet, ergibt sich daraus folgende Routen-URL:

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

Die Portnummer für die Route ist üblicherweise 443. Sofern der Red Hat OpenShift-Administrator nichts anderes sagt, ist dies normalerweise der richtige Wert. Aktualisieren Sie die CONNAME-Felder mit diesen Informationen. Beispiel:

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

Überprüfen Sie in der ursprünglichen Ausgabe des Befehls `grep 'CONNAME\|LOCLADDR\|IPADDRV' -B 3 backup.mqsc` , ob Einträge für LOCLADDR oder IPADDRVvorhanden sind. Wenn ja, löschen Sie sie. Sie sind in einer Containerumgebung nicht relevant.

Nächste Schritte

Sie können jetzt die Warteschlangenmanagerkonfiguration für eine Containerumgebung aktualisieren.

OpenShift V 9.2.1 CD EUS Warteschlangenmanagerkonfiguration für eine Containerumgebung aktualisieren

Bei Ausführung in einem Container werden bestimmte Konfigurationsaspekte vom Container definiert und können in Konflikt mit der exportierten Konfiguration stehen.

Vorbereitende Schritte

Diese Task setzt voraus, dass Sie die IBM MQ-Konfiguration von IP-Adressen und Hostnamen geändert haben.

Informationen zu diesem Vorgang

Die folgenden Konfigurationsaspekte werden vom Container definiert:

- Die Listenerdefinitionen (die den verfügbaren Ports entsprechen).
- Die Position eines potenziellen TLS-Speichers.

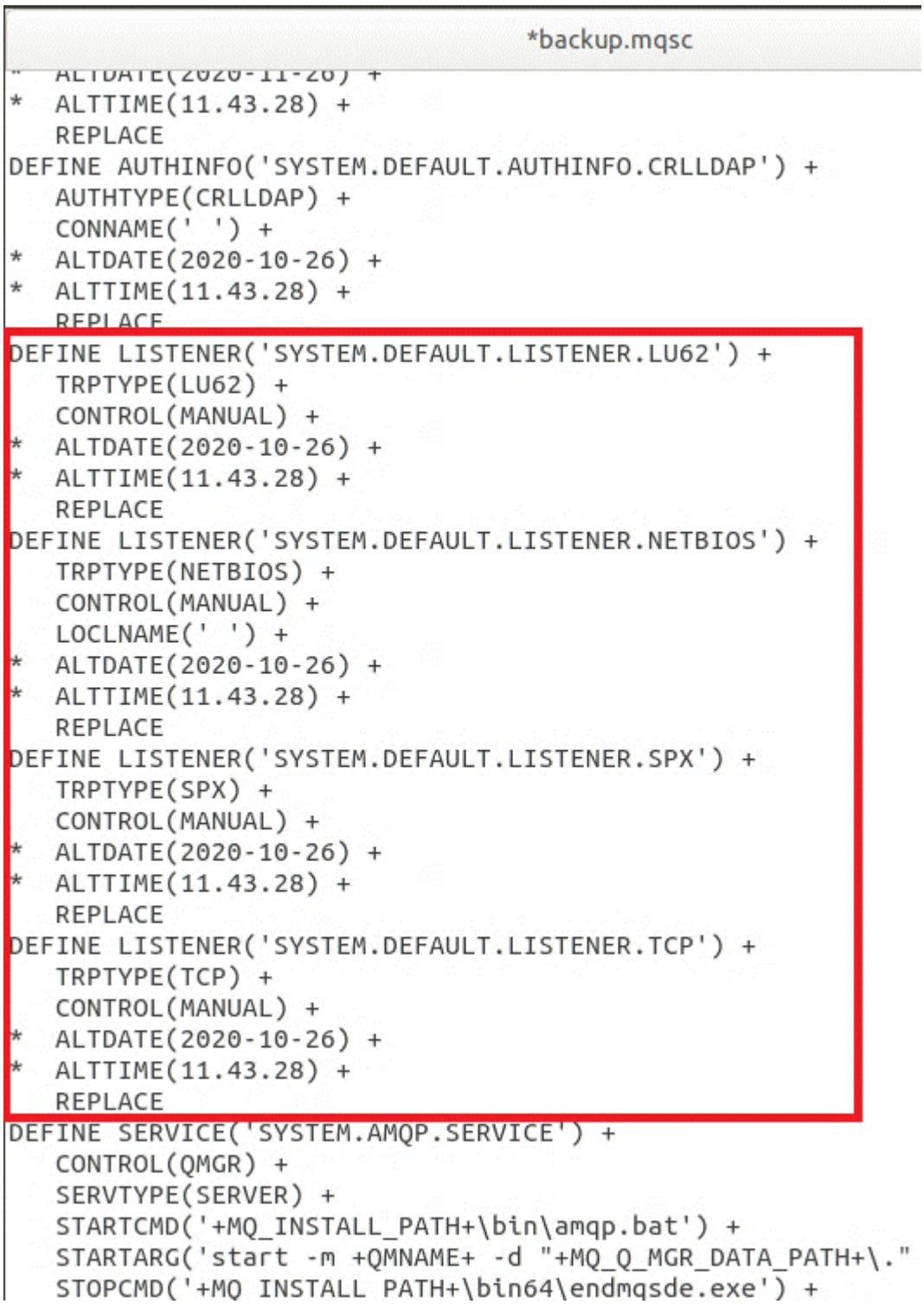
Daher müssen Sie die exportierte Konfiguration aktualisieren:

1. Entfernen Sie alle Listenerdefinitionen.
2. Definieren Sie die Position des TLS-Schlüsselrepositorys.

Vorgehensweise

1. Entfernen Sie alle Listenerdefinitionen.

   Suchen Sie in der Sicherungskonfiguration nach `DEFINE LISTENER`. Diese sollte sich zwischen den Definitionen `AUTHINFO` und `SERVICE` befinden. Markieren Sie den Bereich und löschen Sie ihn.



2. Definieren Sie die Position des TLS-Schlüsselrepositorys.

   Die Sicherung des Warteschlangenmanagers enthält die TLS-Konfiguration für die ursprüngliche Umgebung. Dies unterscheidet sich von der Containerumgebung, und daher sind einige Aktualisierungen erforderlich:

   - Ändern Sie den **CERTLABL**-Eintrag in `default`.
   - Ändern Sie die Position des TLS-Schlüsselrepositorys (**SSLKEYR**) in `/run/runmqserver/tls/key`.

Suchen Sie nach **SSLKEYR**, um die Position des Attributs **SSLKEYR** in der Datei zu finden. In der Regel wird nur ein Eintrag gefunden. Werden mehrere Einträge gefunden, überprüfen Sie, ob Sie das Objekt **QMGR** wie in der folgenden Abbildung dargestellt bearbeiten:

*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
```

Nächste Schritte

Sie können jetzt die Zielarchitektur für IBM MQ in Containern auswählen.

OpenShift V 9.2.1 CD EUS Ziel-HA-Architektur für IBM MQ in Containern auswählen

Wählen Sie Einzelinstanz (ein einzelner Kubernetes-Pod) und Multiinstanz (zwei Pods) aus, um Ihren Hochverfügbarkeitsanforderungen zu entsprechen.

Vorbereitende Schritte

Diese Task setzt voraus, dass Sie die Warteschlangenmanagerkonfiguration für eine Containerumgebung aktualisiert haben.

Informationen zu diesem Vorgang

Der IBM MQ Operator stellt zwei Hochverfügbarkeitsoptionen bereit:

- **Einzelinstanz**: Es wird ein einzelner Container (Pod) gestartet und Red Hat OpenShift ist im Falle eines Ausfalls für den Neustart verantwortlich. Aufgrund der Merkmale eines Stateful Set innerhalb von Kubernetes gibt es mehrere Situationen, in denen diese Funktionsübernahme einen längeren Zeitraum in Anspruch nehmen kann oder durch eine Verwaltungsaktion abgeschlossen werden muss.
- **Multiinstanz**: Es werden zwei Container gestartet (jeder in einem separaten Pod), einer im aktiven und der andere im Standby-Modus. Diese Topologie ermöglicht eine viel schnellere Funktionsübernahme. Dies erfordert ein RWM-Dateisystem (Read Write Many), das die IBM MQ-Anforderungen erfüllt.

In dieser Task wählen Sie nur die Ziel-HA-Architektur aus. Schritte zur Konfiguration der ausgewählten Architektur werden in einer nachfolgenden Task in diesem Szenario beschrieben („Neuen Warteschlangenmanager unter Red Hat OpenShift erstellen” auf Seite 56).

Vorgehensweise

1. Prüfen Sie die beiden Optionen.

   Eine ausführliche Beschreibung dieser beiden Optionen finden Sie im Abschnitt „Hochverfügbarkeit für IBM MQ in Containern” auf Seite 16.
2. Wählen Sie die Ziel-HA-Architektur aus.

   Wenn Sie sich nicht sicher sind, welche Option Sie auswählen sollen, beginnen Sie mit der Option **Einzelinstanz**, und überprüfen Sie, ob diese Ihre Hochverfügbarkeitsanforderungen erfüllt.

Nächste Schritte

Sie können jetzt die Warteschlangenmanagerressourcen erstellen.

OpenShift V 9.2.1 CD EUS Ressourcen für den Warteschlangenmanager erstellen

Importieren Sie die IBM MQ-Konfiguration und die TLS-Zertifikate und -Schlüssel in die Red Hat OpenShift-Umgebung.

Vorbereitende Schritte

Diese Task setzt voraus, dass Sie die Zielarchitektur für die Ausführung von IBM MQ in Containern ausgewählt haben.

Informationen zu diesem Vorgang

In den vorherigen Abschnitten haben Sie zwei Ressourcen extrahiert, aktualisiert und definiert:

- IBM MQ-Konfiguration
- TLS-Zertifikate und -Schlüssel

Sie müssen diese Ressourcen in die Red Hat OpenShift-Umgebung importieren, bevor der Warteschlangenmanager implementiert wird.

Vorgehensweise

1. Importieren Sie die IBM MQ-Konfiguration in Red Hat OpenShift.

   Bei den folgenden Anweisungen wird vorausgesetzt, dass sich die IBM MQ-Konfiguration im aktuellen Verzeichnis in einer Datei mit dem Namen `backup.mqsc` befindet. Andernfalls müssen Sie den Dateinamen der Umgebung entsprechend anpassen.

   a) Melden Sie sich mithilfe von `oc login` bei Ihrem Cluster an.

   b) Laden Sie die IBM MQ-Konfiguration in eine `configmap`.

      Führen Sie den folgenden Befehl aus:

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

   c) Vergewissern Sie sich, dass die Datei erfolgreich geladen wurde.

      Führen Sie den folgenden Befehl aus:

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

2. Importieren Sie die IBM MQ-TLS-Ressourcen.

   Wie in Abschnitt „Optional: Warteschlangenmanagerschlüssel und -zertifikate extrahieren und übernehmen” auf Seite 40 erläutert, ist möglicherweise TLS für die Implementierung des Warteschlangenmanagers erforderlich. Wenn dies der Fall ist, sollten Sie bereits über eine Reihe von Dateien mit den Erweiterungen `.crt` und `.key` verfügen. Diese müssen Sie zu Kubernetes-Secrets hinzufügen, damit der Warteschlangenmanager zum Zeitpunkt der Implementierung referenziert wird.

   Angenommen, Sie verfügen über einen Schlüssel und ein Zertifikat für den Warteschlangenmanager mit den folgenden Namen:

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

   Führen Sie dann folgenden Befehl aus, um diese Dateien zu importieren:

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

   Kubernetes stellt dieses hilfreiche Dienstprogramm beim Importieren eines übereinstimmenden öffentlichen und privaten Schlüssels bereit. Wenn Sie zusätzliche Zertifikate hinzufügen müssen, z. B. zum Truststore des Warteschlangenmanagers, führen Sie folgenden Befehl aus:

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

   Wenn beispielsweise die Dateien `trust1.crt`, `trust2.crt` und `trust3.crt` zu importieren sind, sieht der Befehl wie folgt aus:

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

Nächste Schritte

Sie können jetzt den neuen Warteschlangenmanager unter Red Hat OpenShift erstellen.

OpenShift V 9.2.1 CD EUS Neuen Warteschlangenmanager unter Red Hat OpenShift erstellen

Implementieren Sie entweder einen Einzel-Instanz- oder Multi-Instanz-Warteschlangenmanager unter Red Hat OpenShift.

Vorbereitende Schritte

Diese Task setzt voraus, dass Sie die Warteschlangenmanagerressourcen erstellt und IBM MQ Operator in Red Hat OpenShift installiert haben.

Informationen zu diesem Vorgang

Wie im Abschnitt „Ziel-HA-Architektur für IBM MQ in Containern auswählen” auf Seite 54 beschrieben, gibt es zwei mögliche Implementierungstopologien. Deshalb werden in diesem Abschnitt zwei verschiedene Vorlagen bereitgestellt:

- Implementieren Sie einen Einzel-Instanz-Warteschlangenmanagers.
- Implementieren Sie einen Multi-Instanz-Warteschlangenmanagers.

Wichtig: Führen Sie abhängig von Ihrer bevorzugten Topologie nur eine der beiden Vorlagen aus.

Prozedur

- Implementieren Sie einen Einzel-Instanz-Warteschlangenmanager.

  Der migrierte Warteschlangenmanager wird mithilfe einer YAML-Datei in Red Hat OpenShift implementiert. Es folgt ein Beispiel auf Basis der im vorherigen Abschnitt verwendeten Namen:

```
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
```

  Abhängig von den von Ihnen ausgeführten Schritten muss die vorherige YAML-Datei möglicherweise angepasst werden. Um Sie dabei zu unterstützen, folgt hier eine Erläuterung dieser YAML-Datei:

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

In diesem Block werden Kubernetes-Objekt, Typ und Name definiert. Das einzige Feld, das angepasst werden muss, ist das Feld name.

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

Dieser Block enthält die Versions- und Lizenzinformationen für die Implementierung. Wenn Sie hier Anpassungen vornehmen müssen, verwenden Sie die im Abschnitt „Lizenzierungsreferenz für mq.ibm.com/v1beta1" auf Seite 132 bereitgestellten Informationen.

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

Um den Warteschlangenmanager für die Verwendung von TLS zu konfigurieren, muss er die relevanten Zertifikate und Schlüssel referenzieren. Das Feld `secretName`verweist auf den geheimen Schlüssel Kubernetes, der im Abschnitt IBM MQ-TLS-Ressourcen importieren erstellt wurde, und die Liste der Elemente (`tls.key` und `tls.crt`) sind die Standardnamen, die Kubernetesbei Verwendung der `oc create secret tls`-Syntax zuweist. Wenn Sie zusätzliche Zertifikate zum Truststore hinzufügen müssen, können Sie dies auf ähnliche Weise tun, allerdings handelt es sich bei den Elementen um die entsprechenden Dateinamen, die beim Import verwendet wurden. Es können beispielsweise mit folgendem Code die Truststorezertifikate erstellt werden:

```
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
```

Wichtig: Wenn TLS nicht erforderlich ist, löschen Sie den TLS-Abschnitt der YAML-Datei.

```
  web:
    enabled: true
```

Dies aktiviert die Webkonsole für die Implementierung.

```
  queueManager:
    name: QM1
```

In diesem Abschnitt wird QM1 als Name des Warteschlangenmanagers definiert. Der Warteschlangenmanager wird auf Basis Ihrer Anforderungen angepasst, z. B. was den ursprünglichen Namen des Warteschlangenmanagers betrifft.

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

Der vorherige Code extrahiert die Warteschlangenmanagerkonfiguration, die im Abschnitt IBM MQ-Konfiguration importieren importiert wurde. Wenn Sie unterschiedliche Namen verwendet haben, müssen Sie `my-mqsc-migrated`und `backup.mqsc`ändern.

Beachten Sie, dass der YAML-Beispielcode voraussetzt, dass die Standardspeicherklasse für die Red Hat OpenShift-Umgebung entweder als RWX- oder RWO-Speicherklasse definiert ist. Wenn in der Umgebung keine Standardeinstellung definiert ist, müssen Sie die zu verwendende Speicherklasse angeben. Zu diesem Zweck können Sie die YAML-Datei wie folgt erweitern:

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

Fügen Sie den hervorgehobenen Text hinzu, wobei das Klassenattribut so angepasst wird, dass es mit der Umgebung übereinstimmt. Mit folgendem Befehl können Sie die Speicherklassennamen in der Umgebung ermitteln:

```
oc get storageclass
```

Hier eine von diesem Befehl zurückgegebene Beispielausgabe:

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

Der folgende Code zeigt, wie die IBM MQ-Konfiguration referenziert wird, die im Abschnitt IBM MQ-Konfiguration importieren importiert wurde. Wenn Sie unterschiedliche Namen verwendet haben, müssen Sie `my-mqsc-migrated`und `backup.mqsc`ändern.

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

Sie haben den Einzel-Instanz-Warteschlangenmanager implementiert. Damit ist diese Vorlage abgeschlossen. Sie können jetzt die neue Containerbereitstellung überprüfen.

- Implementieren Sie einen Multi-Instanz-Warteschlangenmanager.

  Der migrierte Warteschlangenmanager wird mithilfe einer YAML-Datei in Red Hat OpenShift implementiert. Das folgende Beispiel basiert auf den in den vorherigen Abschnitten verwendeten Namen.

```
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
```

Es folgt eine Erläuterung dieser YAML. Der Großteil der Konfiguration folgt dem gleichen Ansatz wie bei der Implementierung eines Einzel-Instanz-Warteschlangenmanagers, weshalb hier nur die Verfügbarkeits- und Speicheraspekte des Warteschlangenmanagers erläutert werden.

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

Gibt den Namen des Warteschlangenmanagers als `QM1`an und setzt die Implementierung auf `Multi-Instance`anstelle der Standardeinzelinstanz.

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

Ein IBM MQ-Multi-Instanz-Warteschlangenmanager ist von RWX-Speicher abhängig. Standardmäßig wird ein Warteschlangenmanager im Einzel-Instanz-Modus implementiert. Bei einem Wechsel in den Multi-Instanz-Modus sind deshalb zusätzliche Speicheroptionen erforderlich. Im vorherigen YAML-Beispiel sind drei speicherpersistente Datenträger und eine persistente Datenträgerklasse definiert. Bei der persistenten Datenträgerklasse muss es sich um eine RWX-Speicherklasse handeln. Wenn Sie sich bezüglich der Speicherklassennamen in der Umgebung nicht sicher sind, können Sie die Namen mit folgendem Befehl ermitteln:

```
oc get storageclass
```

Hier eine von diesem Befehl zurückgegebene Beispielausgabe:

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

Der folgende Code zeigt, wie die IBM MQ-Konfiguration referenziert wird, die im Abschnitt IBM MQ-Konfiguration importieren importiert wurde. Wenn Sie unterschiedliche Namen verwendet haben, müssen Sie `my-mqsc-migrated`und `backup.mqsc`ändern.

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

Sie haben den Multi-Instanz-Warteschlangenmanager implementiert. Damit ist diese Vorlage abgeschlossen. Sie können jetzt die neue Containerbereitstellung überprüfen.

OpenShift V 9.2.1 CD EUS Neue Containerimplementierung überprüfen

Jetzt, da IBM MQ unter Red Hat OpenShift implementiert ist, können Sie die Umgebung mithilfe der IBM MQ-Beispiele überprüfen.

Vorbereitende Schritte

Diese Task setzt voraus, dass Sie den neuen Warteschlangenmanager unter Red Hat OpenShift erstellt haben.

Wichtig: Diese Task setzt voraus, dass TLS nicht im Warteschlangenmanager aktiviert ist.

Informationen zu diesem Vorgang

In dieser Task führen Sie die Beispielprogramme von IBM MQ aus dem Container des migrierten Warteschlangenmanagers heraus aus. Möglicherweise ziehen Sie jedoch Ihre eigenen Anwendungen vor, die aus einer anderen Umgebung ausgeführt werden.

Sie benötigen die folgenden Informationen:

- LDAP-Benutzername
- LDAP-Kennwort
- Name des IBM MQ-Kanals
- Warteschlangenname

In diesem Beispielcode werden die folgenden Einstellungen verwendet. Bitte beachten Sie, dass Ihre Einstellungen davon abweichen werden.

- LDAP-Benutzername: mqapp
- LDAP-Kennwort: mqapp
- IBM MQ-Kanalname: DEV.APP.SVRCONN
- Warteschlangenname: Q1

Vorgehensweise

1. Exec in den aktiven IBM MQ-Container.

   Verwenden Sie folgenden Befehl:

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

   Dabei steht `qm1-ibm-mq-0` für den Pod, der in „Neuen Warteschlangenmanager unter Red Hat OpenShift erstellen” auf Seite 56 implementiert wurde. Wenn Sie die Implementierung anders genannt haben, passen Sie diesen Wert an.
2. Senden Sie eine Nachricht.

   Führen Sie folgende Befehle aus:

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

   Sie werden zur Eingabe eines Kennworts aufgefordert und können dann eine Nachricht senden.
3. Überprüfen Sie, ob die Nachricht erfolgreich empfangen wurde.

Führen Sie das GET-Beispiel aus:

```
./amqsgetc Q1 QM1
```

Ergebnisse

Damit ist das „Migration von IBM MQ auf IBM Cloud Pak for Integration” auf Seite 37 abgeschlossen.

Nächste Schritte

Die folgenden Informationen helfen Ihnen bei komplexeren Migrationsszenarios:

Nachrichten in der Warteschlange migrieren
Folgen Sie zum Migrieren vorhandener Nachrichten in der Warteschlange den Anweisungen im folgenden Abschnitt zum Exportieren und Importieren von Nachrichten, nachdem der neue Warteschlangenmanager implementiert wurde: Dienstprogramm 'dmpmqmsg' zwischen zwei Systemen verwenden.

Verbindung zu IBM MQ von außerhalb der Red Hat OpenShift-Umgebung herstellen
Der implementierte Warteschlangenmanager kann IBM MQ-Clients und -Warteschlangenmanagern außerhalb der Red Hat OpenShift-Umgebung zugänglich gemacht werden. Der Prozess ist von der IBM MQ-Version abhängig, die eine Verbindung in die Red Hat OpenShift-Umgebung herstellt. Siehe „Route für Verbindung zu einem Warteschlangenmanager von außerhalb eines Red Hat OpenShift-Clusters konfigurieren” auf Seite 113.

OpenShift CP4I IBM MQ Operator unter Red Hat OpenShift installieren und deinstallieren

Unter Red Hat OpenShift kann IBM MQ Operator über den Operator-Hub installiert werden.

Prozedur

- „Abhängigkeiten für IBM MQ Operator” auf Seite 10.
- „Für IBM MQ Operator erforderliche Berechtigungen auf Clusterebene” auf Seite 11.
- „IBM MQ Operator über die Red Hat OpenShift-Webkonsole installieren” auf Seite 61.
- „IBM MQ Operator über die Red Hat OpenShift-CLI installieren” auf Seite 64.
- „IBM MQ Operator in einer Airgap-Umgebung installieren” auf Seite 68.

Zugehörige Tasks

„IBM MQ Operator über die Red Hat OpenShift-Webkonsole deinstallieren” auf Seite 63
Sie können IBM MQ Operator über die Red Hat OpenShift-Webkonsole in Red Hat OpenShift deinstallieren.

„IBM MQ Operator über die Red Hat OpenShift-CLI deinstallieren” auf Seite 66
Sie können IBM MQ Operator über die Befehlszeilenschnittstelle (CLI) von Red Hat OpenShift in Red Hat OpenShift deinstallieren. Je nachdem, ob IBM MQ Operator in einem einzelnen Namensbereich oder für alle Namensbereiche des Clusters installiert und verfügbar ist, unterscheidet sich der Deinstallationsprozess.

OpenShift CP4I IBM MQ Operator über die Red Hat OpenShift-Webkonsole installieren

Unter Red Hat OpenShift kann IBM MQ Operator über den Operator-Hub installiert werden.

Vorbereitende Schritte

Melden Sie sich bei der Webkonsole Ihres Red Hat OpenShift-Clusters an.

Vorgehensweise

1. EUS

   Optional: Fügen Sie die IBM Common Services Operators der Liste der installierbaren Operatoren hinzu.

   Anmerkung:

   Dieser Schritt gilt für Releases von IBM MQ Operator 1.5 und früher. Der Schritt fügt einen separaten Common Services-Katalog hinzu. Für spätere Releases des Operators sind die Common Services in den IBM Katalog eingeschlossen.

   a) Klicken Sie auf das Plussymbol oben rechts in der Anzeige. Das Dialogfenster **Import YAML** (YAML importieren) wird geöffnet.

   b) Fügen Sie folgende Ressourcendefinition in das Dialogfenster ein:

   ```
   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) Klicken Sie auf **Erstellen**.

2. Fügen Sie die IBM Operatoren zur Liste der installierbaren Operatoren hinzu.

   a) Klicken Sie auf das Plussymbol oben rechts in der Anzeige. Das Dialogfenster **Import YAML** (YAML importieren) wird geöffnet.

   b) Fügen Sie folgende Ressourcendefinition in das Dialogfenster ein:

   ```
   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) Klicken Sie auf **Erstellen**.

3. Erstellen Sie einen Namensbereich für IBM MQ Operator.

   IBM MQ Operator kann für einen einzelnen Namensbereich oder für alle Namensbereiche installiert werden. Dieser Schritt ist nur erforderlich, wenn Sie eine Installation in einem bestimmten Namensbereich ausführen möchten, der noch nicht vorhanden ist.

   a) Klicken Sie im Navigationsfenster auf **Home** > **Projects**.

   Die Seite 'Projects' wird angezeigt.

   b) Klicken Sie auf **Create Project** (Projekt erstellen). Es wird ein Bereich zum Erstellen des Projekts angezeigt.

   c) Geben Sie Details zu dem Namensbereich ein, den Sie erstellen. Sie können zum Beispiel 'ibm-mq' als Name angeben.

   d) Klicken Sie auf **Erstellen**. Der Namensbereich für IBM MQ Operator wird erstellt.

4. Installieren Sie IBM MQ Operator.

a) Klicken Sie im Navigationsfenster auf **Operators** > **OperatorHub**.

   Die Seite 'OperatorHub' wird angezeigt.

b) Geben Sie 'IBM MQ' im Feld **All Items** (Alle Elemente) ein.

   Der IBM MQ-Katalogeintrag wird angezeigt.

c) Wählen Sie **IBM MQ** aus.

   Das Fenster 'IBM MQ' wird angezeigt.

d) Klicken Sie auf **Install** (Installieren).

   Die Seite 'Create Operator Subscription' (Operatorsubskription erstellen) wird angezeigt.

e) Bestimmen Sie anhand der Informationen im Abschnitt „Versionsunterstützung für IBM MQ Operator” auf Seite 7, welcher Operatorkanal ausgewählt werden soll.

f) Legen Sie als Installationsmodus entweder den von Ihnen erstellten Namensbereich oder den clusterweiten Geltungsbereich fest.

   Die Auswahl des clusterweiten Geltungsbereichs wird empfohlen, weil die Installation unterschiedlicher Versionen eines Operators in verschiedenen Namensbereichen zu Problemen führen kann. Operatoren sind als Erweiterungen der Steuerebene konzipiert.

g) Klicken Sie auf **Subscribe** (Subskribieren).

   IBM MQ wird auf der Seite 'Installed Operators' (Installierte Operatoren) angezeigt.

h) Überprüfen Sie den Status des Operators auf der Seite 'Installed Operators'. Der Status ändert sich in 'Succeeded' (Erfolgreich), sobald die Installation abgeschlossen ist.

Nächste Schritte

„Red Hat OpenShift-Projekt für IBM MQ über die Red Hat OpenShift-Webkonsole vorbereiten” auf Seite 83

OpenShift CP4I *IBM MQ Operator über die Red Hat OpenShift-Webkonsole deinstallieren*

Sie können IBM MQ Operator über die Red Hat OpenShift-Webkonsole in Red Hat OpenShift deinstallieren.

Vorbereitende Schritte

Melden Sie sich bei der Webkonsole Ihres Red Hat OpenShift-Clusters an.

Wenn IBM MQ Operator in allen Projekten bzw. Namensbereichen des Clusters installiert ist, wiederholen Sie die Schritte 1-5 des folgenden Verfahrens für jedes Projekt, aus dem Warteschlangenmanager gelöscht werden sollen.

Vorgehensweise

1. Wählen Sie **Operators** > **Installed Operators** (Operatoren > Installierte Operatoren) aus.
2. Wählen Sie in der Dropdown-Liste **Project** (Projekt) ein Projekt aus.
3. Klicken Sie auf den **IBM MQ**-Operator.
4. Klicken Sie auf die Registerkarte **Queue Managers** (Warteschlangenmanager), um die von diesem IBM MQ Operator verwalteten Warteschlangenmanager anzuzeigen.
5. Löschen Sie einen oder mehrere Warteschlangenmanager.

   Beachten Sie, dass diese Warteschlangenmanager zwar weiterhin ausgeführt werden, aber ohne Vorhandensein eines IBM MQ Operator nicht wie erwartet funktionieren.
6. Optional: Wiederholen Sie ggf. die Schritte 1-5 für jedes Projekt, aus dem Sie Warteschlangenmanager löschen möchten.
7. Kehren Sie zu **Operators** > **Installed Operators** (Operatoren > Installierte Operatoren) zurück.

8. Klicken Sie neben dem **IBM MQ**-Operator auf das Menü mit den drei Punkten und wählen Sie **Uninstall Operator** (Operator deinstallieren) aus.
9. Bei Verwendung von Red Hat OpenShift Container Platform 4.7 müssen Sie möglicherweise den Validierungswebhook manuell über die Befehlszeile löschen:

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

OpenShift CP4I IBM MQ Operator über die Red Hat OpenShift-CLI installieren

Unter Red Hat OpenShift kann IBM MQ Operator über den Operator-Hub installiert werden.

Vorbereitende Schritte

Melden Sie sich mit **oc login** an der Befehlszeilenschnittstelle (CLI) von Red Hat OpenShift an. Um diese Schritte ausführen zu können, müssen Sie ein Clusteradministrator sein.

Vorgehensweise

1. EUS

   Optional: Erstellen Sie eine **CatalogSource** für die Operatoren von IBM Common Services.

   Anmerkung:

   Dieser Schritt gilt für Releases von IBM MQ Operator 1.5 und früher. Der Schritt fügt einen separaten Common Services-Katalog hinzu. Für spätere Releases des Operators sind die Common Services in den IBM Katalog eingeschlossen.

   a) Erstellen Sie eine YAML-Datei, die die **CatalogSource**-Ressource definiert.

   Erstellen Sie eine Datei mit dem Namen 'operator-source-cs.yaml' mit folgendem Inhalt:

```
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) Wenden Sie **CatalogSource** auf den Server an.

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

2. Erstellen Sie **CatalogSource** für die IBM-Operatoren

   a) Erstellen Sie eine YAML-Datei, die die **CatalogSource**-Ressource definiert

   Erstellen Sie eine Datei mit dem Namen 'operator-source-ibm.yaml' mit folgendem Inhalt:

```
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) Wenden Sie **CatalogSource**auf den Server an.

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

3. Erstellen Sie einen Namensbereich für IBM MQ Operator.

   IBM MQ Operator kann für einen einzelnen Namensbereich oder für alle Namensbereiche installiert werden. Dieser Schritt ist nur erforderlich, wenn Sie eine Installation in einem bestimmten Namensbereich ausführen möchten, der noch nicht vorhanden ist.

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

4. Zeigen Sie die Liste der Operatoren an, die dem Cluster aus dem OperatorHub zur Verfügung stehen.

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

5. Überprüfen Sie, welche Installationsmodi IBM MQ Operator unterstützt und welche Kanäle verfügbar sind.

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

6. YAML-Datei für **OperatorGroup**-Objekt erstellen

   Eine **OperatorGroup**ist eine OLM-Ressource, die Zielnamensbereiche auswählt, in denen der erforderliche RBAC-Zugriff für alle Operatoren in demselben Namensbereich wie die **OperatorGroup**generiert werden soll.

   Der Namensbereich, für den Sie den Operator subskribieren, muss über eine **OperatorGroup**verfügen, die der **InstallMode**des Operators entspricht (Modus `Alle Namensbereiche` oder `Einzelner Namensbereich`). Wenn der Operator, den Sie installieren möchten, die `Alle Namensbereiche` verwendet, verfügt der Namensbereich `openshift-operators`bereits über eine entsprechende **OperatorGroup**.

   Wenn der Operator jedoch den Modus `Einzelner Namensbereich` verwendet und Sie noch nicht über eine entsprechende **OperatorGroup**verfügen, müssen Sie eine erstellen.

   a) Erstellen Sie eine Datei mit dem Namen 'mq-operator-group.yaml' mit folgendem Inhalt:

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

   b) Erstellen Sie das **OperatorGroup**-Objekt

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

7. Erstellen Sie eine YAML-Datei für **Subscription**-Objekte, um einen Namensbereich für IBM MQ Operatorzu abonnieren

   a) Bestimmen Sie anhand der Informationen im Abschnitt „Versionsunterstützung für IBM MQ Operator” auf Seite 7, welcher Operatorkanal ausgewählt werden soll.

   b) Erstellen Sie eine Datei mit dem Namen "mq-sub.yaml" mit dem folgenden Inhalt, ändern Sie jedoch **channel**so, dass sie dem Kanal für die zu installierende Version von IBM MQ Operatorentspricht.

```
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
```

Geben Sie für die `Alle Namensbereiche` **InstallMode**-Verwendung **openshift-operators**im Namensbereich an. Andernfalls geben Sie den relevanten einzelnen Namensbereich für die Verwendung von `Einzelner Namensbereich`**InstallMode** an. Beachten Sie, dass Sie nur das Feld **namespace**ändern und das Feld **sourceNamespace**unverändert lassen sollten.

c) Erstellen Sie das **Subscription**-Objekt

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

8. Überprüfen Sie den Status des Operators

Sobald die Installation des Operators erfolgreich war, wird als Podstatus *Aktiv* angezeigt. Geben Sie für die `Alle Namensbereiche` **InstallMode**-Verwendung **openshift-operators**als Namensbereich an. Andernfalls geben Sie den relevanten einzelnen Namensbereich für die Verwendung von `Einzelner Namensbereich`**InstallMode** an.

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

Nächste Schritte

„Red Hat OpenShift-Projekt für IBM MQ über die Red Hat OpenShift-CLI vorbereiten” auf Seite 84

OpenShift CP4I *IBM MQ Operator über die Red Hat OpenShift-CLI deinstallieren*

Sie können IBM MQ Operator über die Befehlszeilenschnittstelle (CLI) von Red Hat OpenShift in Red Hat OpenShift deinstallieren. Je nachdem, ob IBM MQ Operator in einem einzelnen Namensbereich oder für alle Namensbereiche des Clusters installiert und verfügbar ist, unterscheidet sich der Deinstallationsprozess.

Vorbereitende Schritte

Melden Sie sich mit `oc login`bei Ihrem Red Hat OpenShift -Cluster an.

Prozedur

- Wenn IBM MQ Operator in einem einzelnen Namensbereich installiert ist, führen Sie die folgenden Schritte aus:

a) Stellen Sie sicher, dass Sie sich im richtigen Projekt befinden:

```
oc project <project_name>
```

b) Zeigen Sie die im Projekt installierten Warteschlangenmanager an:

```
oc get qmgr
```

c) Löschen Sie einen oder mehrere Warteschlangenmanager:

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

Beachten Sie, dass diese Warteschlangenmanager zwar weiterhin ausgeführt werden, aber ohne Vorhandensein eines IBM MQ Operator nicht wie erwartet funktionieren.

d) Zeigen Sie die **ClusterServiceVersion**-Instanzen an:

```
oc get csv
```

e) Löschen Sie die IBM MQ **ClusterServiceVersion**:

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

f) Zeigen Sie die Subskriptionen an:

```
oc get subscription
```

g) Löschen Sie alle Subskriptionen:

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

h) Optional: Wenn die Common Services von keinem Prozess mehr verwendet werden, können Sie auch den Common Services-Operator deinstallieren und die Operatorgruppe löschen:

a. Deinstallieren Sie den Common Services-Operator nach den Anweisungen im Abschnitt Common Services deinstallieren der IBM Cloud Pak foundational services-Produktdokumentation.

b. Zeigen Sie die Operatorgruppe an:

```
oc get operatorgroup
```

c. Löschen Sie die Operatorgruppe:

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

- Wenn IBM MQ Operator für alle Namensbereiche des Clusters installiert und verfügbar ist, führen Sie die folgenden Schritte aus:

a) Zeigen Sie alle installierten Warteschlangenmanager an:

```
oc get qmgr -A
```

b) Löschen Sie einen oder mehrere Warteschlangenmanager:

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

Beachten Sie, dass diese Warteschlangenmanager zwar weiterhin ausgeführt werden, aber ohne Vorhandensein eines IBM MQ Operator nicht wie erwartet funktionieren.

c) Zeigen Sie die **ClusterServiceVersion**-Instanzen an:

```
oc get csv -A
```

d) Löschen Sie die IBM MQ **ClusterServiceVersion** aus dem Cluster:

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

e) Zeigen Sie die Subskriptionen an:

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

f) Löschen Sie die Subskriptionen:

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

g) Bei Verwendung von Red Hat OpenShift Container Platform 4.7 müssen Sie möglicherweise den Validierungswebhook manuell löschen:

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

h) Optional: Wenn die Common Services von keinem Prozess mehr verwendet werden, können Sie auch den Common Services-Operator deinstallieren:

Befolgen Sie hierzu die Anweisungen im Abschnitt Common Services deinstallieren der IBM Cloud Pak foundational services-Produktdokumentation.

OpenShift CP4I Linux IBM MQ Operator in einer Airgap-Umgebung installieren

Dieses Lernprogramm führt Sie durch die Installation von IBM MQ Operator in einem Red Hat OpenShift-Cluster ohne Internetkonnektivität. Sie können IBM MQ Operator mithilfe einer portierbaren Speichereinheit oder einer Bastionsmaschine in einer Airgap-Umgebung installieren.

IBM MQ Operator mithilfe einer portierbaren Speichereinheit in einer Airgap-Umgebung installieren

Informationen zu den Schritten zur Durchführung der Installation finden Sie unter Mirroring Images With A Portable Storage Device in der IBM Cloud Pak for Integration-Dokumentation. Wenn Sie nur IBM MQinstallieren, ersetzen Sie alle Vorkommen der folgenden Umgebungsvariablen durch die hier angegebenen Werte:

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

Dabei steht *Versionsnummer* für die Version des Falls, den Sie zum Installieren der Airgap-Installation verwenden möchten. Eine Liste der verfügbaren Fallversionen finden Sie unter https://github.com/IBM/cloud-pak/tree/master/repo/case/ibm-mq. Bestimmen Sie anhand der Informationen im Abschnitt „Versionsunterstützung für IBM MQ Operator” auf Seite 7, welcher Operatorkanal ausgewählt werden soll.

IBM MQ Operator mithilfe einer Bastionsmaschine in einer Airgap-Umgebung installieren

1. „Voraussetzungen” auf Seite 68
2. „Docker-Registry vorbereiten” auf Seite 68
3. „Bastionshost vorbereiten” auf Seite 69
4. „Umgebungsvariablen für das Installationsprogramm und den Imagebestand erstellen” auf Seite 70
5. „IBM MQ-Installationsprogramm und Imagebestand herunterladen” auf Seite 70
6. „Als Clusteradministrator beim Red Hat OpenShift Container Platform-Cluster anmelden” auf Seite 71
7. „Kubernetes-Namensbereich für den IBM MQ Operator erstellen” auf Seite 71
8. „Images spiegeln und Cluster konfigurieren” auf Seite 71
9. „Installieren Sie IBM MQ Operator.” auf Seite 73
10. „IBM MQ-Warteschlangenmanager implementieren” auf Seite 74

Voraussetzungen

1. Es muss ein Red Hat OpenShift Container Platform-Cluster installiert sein. Informationen zu den unterstützten Red Hat OpenShift Container Platform-Versionen finden Sie im Abschnitt „Versionsunterstützung für IBM MQ Operator” auf Seite 7.
2. Es muss eine Docker-Registry verfügbar sein. Weitere Informationen finden Sie unter „Docker-Registry vorbereiten” auf Seite 68.
3. Es muss ein Bastionsserver konfiguriert sein. Weitere Informationen finden Sie unter „Bastionshost vorbereiten” auf Seite 69.

Docker-Registry vorbereiten

Alle Images in der lokalen Umgebung werden in einer lokalen Docker-Registry gespeichert. Sie müssen eine solche Registry erstellen und sicherstellen, dass sie folgende Anforderungen erfüllt:

- Sie unterstützt Docker Manifest V2, Schema 2.

- Sie unterstützt Images für mehrere Architekturen.
- Sowohl der Bastionsserver als auch die Red Hat OpenShift Container Platform-Clusterknoten können darauf zugreifen.
- Sie hat den Benutzernamen und das Kennwort eines Benutzers, der vom Bastionshost aus in die Zielregistry schreiben kann.
- Sie hat den Benutzernamen und das Kennwort eines Benutzers, der die Zielregistry lesen kann, die sich auf den Red Hat OpenShift-Clusterknoten befindet.
- Sie lässt Trennzeichen im Imagenamen zu.

Nachdem Sie die Docker-Registry erstellt haben, müssen Sie die Registry konfigurieren:

1. Erstellen Sie Registry-Namensbereiche.
   - `ibmcom` - Namensbereich für die Speicherung aller Images aus dem Namensbereich `docker-hub.io/ibmcom`.

     Der Namensbereich `ibmcom` ist für alle IBM Images, die öffentlich verfügbar und für die keine Berechtigungsnachweise zur Durchführung einer Pull-Operationen erforderlich sind.
   - `cp` - Namensbereich für die Speicherung der IBM Images aus dem Repository `cp.icr.io/cp`.

     Der Namensbereich `cp` ist für die Images in der IBM Entitled Registry, für die ein Produktberechtigungsschlüssel sowie Berechtigungsnachweise zur Durchführung einer Pull-Operation erforderlich sind. Um Ihren Berechtigungsschlüssel abzurufen, melden Sie sich bei MyIBM Container Software Library mit der IBM ID und dem Kennwort an, die der berechtigten Software zugeordnet sind. Wählen Sie im Abschnitt **Entitlement keys** (Berechtigungsschlüssel) die Option **Copy key** (Schlüssel kopieren) aus, um den Berechtigungsschlüssel in die Zwischenablage zu kopieren. Speichern Sie ihn dann, um ihn in den folgenden Schritten zu verwenden.
   - `opencloudio` - Namensbereich für die Speicherung der Images aus `quay.io/opencloudio`.

     Der Namensbereich `opencloudio` ist für die Auswahl von IBM Open-Source-Komponentenimages, die auf quay.io verfügbar sind. Die IBM Cloud Pak foundational services-Images werden auf `opencloudio` gehostet.
2. Überprüfen Sie, ob jeder Namensbereich folgende Anforderungen erfüllt:
   - Er unterstützt die Erstellung automatischer Repositorys.
   - Er verfügt über Berechtigungsnachweise eines Benutzers, der Repositorys schreiben und erstellen kann. Der Bastionshost verwendet diese Berechtigungsnachweise.
   - Er verfügt über Berechtigungsnachweise eines Benutzers, der alle Repositorys lesen kann. Der Red Hat OpenShift Container Platform-Cluster verwendet diese Berechtigungsnachweise.

Bastionshost vorbereiten

Bereiten Sie einen Bastionshost vor, der auf den Red Hat OpenShift Container Platform-Cluster, die lokale Docker-Registry und das Internet zugreifen kann. Der Bastionshost muss sich auf einer Linux for x86-64-Plattform mit einem Betriebssystem befinden, das von der IBM Cloud Pak-CLI und der Red Hat OpenShift Container Platform-CLI unterstützt wird.

Führen Sie folgende Schritte auf dem Bastionsknoten aus:

1. Installieren Sie OpenSSL Version 1.11.1 oder höher.
2. Installieren Sie Docker oder Podman auf dem Bastionsknoten.
   - Installieren Sie Docker durch Ausführung der folgenden Befehle:

     ```
     yum check-update
     yum install docker
     ```
   - Informationen zur Installation von Podman finden Sie unter Podman Installation Instructions.
3. Installieren Sie skopeo Version 1.x.x auf der Bastion-Node. Führen Sie die folgenden Befehle aus, um skopeo zu installieren:

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

4. Installieren Sie die Befehlszeilenschnittstelle von IBM Cloud Pak. Installieren Sie die neueste Version der Binärdatei für Ihre Plattform. Weitere Informationen finden Sie unter cloud-pak-cli.

   a. Laden Sie die Binärdatei herunter.

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

   Beispiel:

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

   b. Extrahieren Sie die Binärdatei.

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

   c. Führen Sie folgende Befehle aus, um die Datei zu ändern und zu verschieben:

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

   d. Vergewissern Sie sich, dass `cloudctl` installiert ist:

   ```
   cloudctl --help
   ```

5. Installieren Sie das `oc` Red Hat OpenShift Container Platform -CLI-Tool.

   Weitere Informationen finden Sie unter Red Hat OpenShift Container Platform-CLI-Tools

6. Erstellen Sie ein Verzeichnis, das als Offlinespeicher dient.

   Es folgt der Name eines Beispielverzeichnisses. Dieses Beispiel wird in den nachfolgenden Schritten verwendet.

   ```
   mkdir $HOME/offline
   ```

   Hinweis: Der Offlinespeicher muss persistent sein, um zu verhindern, dass Daten mehr als einmal übertragen werden. Die Persistenz hilft auch dabei, den Spiegelungsprozess mehrfach oder nach einem Zeitplan auszuführen.

Umgebungsvariablen für das Installationsprogramm und den Imagebestand erstellen

Erstellen Sie die folgenden Umgebungsvariablen mit dem Imagenamen des Installationsprogramms und dem Imagebestand:

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

Dabei steht *Versionsnummer* für die Version des Falls, den Sie zum Installieren der Airgap-Installation verwenden möchten. Eine Liste der verfügbaren Fallversionen finden Sie unter https://github.com/IBM/cloud-pak/tree/master/repo/case/ibm-mq. Lesen Sie die Informationen zur Versionsunterstützung für den IBM MQ Operator , um festzustellen, welcher Operatorkanal ausgewählt werden muss.

IBM MQ-Installationsprogramm und Imagebestand herunterladen

Laden Sie das `ibm-mq` -Installationsprogramm und den Imagebestand auf den Bastionshost herunter:

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

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

Als Clusteradministrator beim Red Hat OpenShift Container Platform-Cluster anmelden

Es folgt ein Beispielbefehl für die Anmeldung beim Red Hat OpenShift Container Platform-Cluster:

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

Kubernetes-Namensbereich für den IBM MQ Operator erstellen

Erstellen Sie eine Umgebungsvariable mit einem Namensbereich, um IBM MQ Operatorzu installieren, und erstellen Sie dann den Namensbereich:

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

Images spiegeln und Cluster konfigurieren

Führen Sie die folgenden Schritte aus, um die Images zu spiegeln und Ihren Cluster zu konfigurieren:

Anmerkung: Verwenden Sie in keinem Befehl die Tilde innerhalb der Anführungszeichen. Verwenden Sie beispielsweise nicht `args "--registry registry --user registry_userid --pass registry_password --inputDir ~/offline"`. Die Tilde wird nicht aufgelöst und Ihre Befehle schlagen möglicherweise fehl.

1. Speichern Sie Authentifizierungsnachweise für alle Docker-Quellenregistrys.

   Alle IBM Cloud-Platform Common Services, IBM MQ Operator-Image und IBM MQ-Advanced Developer-Images werden in öffentlichen Registrys gespeichert, für die keine Authentifizierung erforderlich ist. Für IBM MQ Advanced Server (nicht Developer), andere Produkte und Komponenten von Drittanbietern sind jedoch eine oder mehrere authentifizierte Registrys erforderlich. Für folgende Registrys ist eine Authentifizierung erforderlich:

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

   Weitere Informationen zu diesen Registrys finden Sie im Abschnitt Registry-Namensbereiche erstellen.

   Sie müssen folgenden Befehl ausführen, um Berechtigungsnachweise für alle Registrys zu konfigurieren, für die eine Authentifizierung erforderlich ist. Führen Sie den Befehl für jede dieser Registrys gesondert aus:

   ```
   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 $HO
   ME/offline"
   ```

   Der Befehl speichert die Registry-Berechtigungsnachweise in einer Datei auf Ihrem Dateisystem an der Position `$HOME/.airgap/secrets`.

2. Erstellen Sie Umgebungsvariablen mit den Verbindungsinformationen für die lokale Docker-Registry.

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

Hinweis: Die Docker-Registry verwendet Standardports, z. B. 80 oder 443. Wenn Ihre Docker -Registry einen vom Standard abweichenden Port verwendet, geben Sie den Port mit der Syntax *host*:*port*an. For example:

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

3. Konfigurieren Sie einen geheimen Schlüssel für die Authentifizierung für die lokale Docker-Registry.

   Hinweis: Dieser Schritt muss nur einmal ausgeführt werden.

   ```
   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_DO
   CKER_PASSWORD}"
   ```

   Der Befehl speichert die Registry-Berechtigungsnachweise in einer Datei auf Ihrem Dateisystem an der Position $HOME/.airgap/secrets.

4. Konfigurieren Sie einen geheimen Schlüssel für globale Image-Pull-Operationen und eine Richtlinie (**ImageContentSourcePolicy**).

   a. Überprüfen Sie, ob ein Knotenneustart erforderlich ist.

   - In Red Hat OpenShift Container Platform Version 4.4 und höher und in einer neuen Installation von IBM MQ Operator unter Verwendung von Airgap werden mit diesem Schritt alle Clusterknoten erneut gestartet. Die Clusterressourcen sind möglicherweise erst dann verfügbar, wenn der geheime Schlüssel für Pull-Operationen angewendet wird.
   - In IBM MQ Operator 1.8wird CASE aktualisiert, um eine zusätzliche Spiegelungsquelle für Images einzuschließen. Daher wird ein Knotenneustart ausgelöst, wenn Sie ein Upgrade von früheren Versionen von IBM MQ Operator auf Version 1.8 oder höher durchführen.
   - Um zu prüfen, ob dieser Schritt einen Knotenneustart erfordert, fügen Sie dem Code für diesen Schritt die Option `--dry-run` hinzu. Dadurch wird die neueste **ImageContentSourcePolicy** generiert und im Konsolenfenster (**stdout**) angezeigt. Wenn sich diese **ImageContentSourcePolicy** vom konfigurierten Cluster **ImageContentSourcePolicy**unterscheidet, erfolgt ein Neustart.

     ```
     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 ${LO
     CAL_DOCKER_PASSWORD} --inputDir $HOME/offline --dryRun"
     ```

   b. Führen Sie den Code für diesen Schritt ohne die Option `--dry-run` aus, um den globalen geheimen Schlüssel für Image-Pull-Operationen und **ImageContentSourcePolicy**zu konfigurieren:

   ```
   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_DO
   CKER_PASSWORD} --inputDir $HOME/offline"
   ```

5. Überprüfen Sie, ob die **ImageContentSourcePolicy**-Ressource erstellt wurde.

   ```
   oc get imageContentSourcePolicy
   ```

6. Optional: Wenn Sie eine unsichere Registry verwenden, müssen Sie die lokale Registry zur Liste **insecureRegistries** des Clusters hinzufügen.

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

7. Überprüfen Sie den Status Ihres Clusters.

```
oc get nodes
```

   Nachdem die **imageContentsourcePolicy** und der geheime Schlüssel für globale Image-Pull-Operationen angewendet wurden, wird als Knotenstatus entweder **Ready** (Bereit), **Scheduling** (In Planung) oder **Disabled** (Inaktiviert) angezeigt. Warten Sie, bis für alle Knoten der Status **Ready** angezeigt wird.

8. Spiegeln Sie die Images in die lokale Registry.

```
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_DO
CKER_PASSWORD} --inputDir $HOME/offline"
```

Installieren Sie IBM MQ Operator.

1. Melden Sie sich bei der Webkonsole Ihres Red Hat OpenShift-Clusters an.
2. Erstellen Sie eine Katalogquelle. Verwenden Sie dasselbe Terminal wie bei den vorherigen Schritten.

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

3. Überprüfen Sie, ob die **CatalogSource** für den Common Services Installer Operator erstellt wurde.

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

4. Installieren Sie IBM MQ Operator mit OLM.
   a. Klicken Sie im Navigationsfenster auf **Operators** > **OperatorHub**.

      Die Seite **OperatorHub** wird angezeigt.

   b. Geben Sie im Feld **All Items** den Eintrag `IBM MQ` ein.

      Der IBM MQ-Katalogeintrag wird angezeigt.

   c. Wählen Sie **IBM MQ** aus.

      Das Fenster **IBM MQ** wird angezeigt.

   d. Klicken Sie auf **Install** (Installieren).

      Die Seite **Create Operator Subscription** (Operatorsubskription erstellen) wird angezeigt.

   e. Bestimmen Sie anhand der Informationen im Abschnitt „Versionsunterstützung für IBM MQ Operator” auf Seite 7, welcher Operatorkanal ausgewählt werden soll.

   f. Geben Sie als **Installation Mode** (Installationsmodus) entweder den Namensbereich, den Sie erstellt haben, oder den clusterweiten Bereich an.

   g. Klicken Sie auf **Subscribe** (Subskribieren).

      IBM MQ wird zur Seite **Installed Operators** (Installierte Operatoren) hinzugefügt.

   h. Überprüfen Sie den Status des Operators auf der Seite **Installed Operators** (Installierte Operatoren). Der Status ändert sich in **Succeeded** (Erfolgreich), sobald die Installation abgeschlossen ist.

IBM MQ-Warteschlangenmanager implementieren

Informationen zum Erstellen eines neuen Warteschlangenmanagers unter dem installierten Operator finden Sie im Abschnitt „Warteschlangenmanager mithilfe von IBM MQ Operator implementieren und konfigurieren” auf Seite 82.

Zugehörige Tasks

„Upgrade für IBM MQ Operator oder Warteschlangenmanager in einer Airgap-Umgebung vorbereiten” auf Seite 74
In einem Red Hat OpenShift -Cluster ohne Internetkonnektivität müssen Sie vorbereitende Schritte ausführen, bevor Sie ein Upgrade für IBM MQ Operatordurchführen.

OpenShift CP4I Upgrade für IBM MQ Operator und Warteschlangenmanager durchführen

Nach dem Upgrade von IBM MQ Operator können Sie Ihre Warteschlangenmanager aktualisieren.

Prozedur

- „Upgrade von IBM MQ Operator über die Red Hat OpenShift-Webkonsole durchführen” auf Seite 77.
- „Upgrade von IBM MQ Operator über die Red Hat OpenShift-CLI durchführen” auf Seite 79.
- „Upgrade eines IBM MQ MQ-Warteschlangenmanagers über die Red Hat OpenShift-Webkonsole durchführen” auf Seite 80.
- „Upgrade eines IBM MQ-Warteschlangenmanagers über die Red Hat OpenShift-CLI durchführen” auf Seite 81.

OpenShift CP4I Linux Upgrade für IBM MQ Operator oder Warteschlangenmanager in einer Airgap-Umgebung vorbereiten

In einem Red Hat OpenShift -Cluster ohne Internetkonnektivität müssen Sie vorbereitende Schritte ausführen, bevor Sie ein Upgrade für IBM MQ Operatordurchführen.

Vorbereitende Schritte

In diesem Abschnitt wird davon ausgegangen, dass Sie bereits eine lokale Image-Registry konfiguriert haben, in der die zuvor freigegebenen IBM Cloud Pak for Integration -Images gespiegelt werden.

Informationen zu diesem Vorgang

Bevor Sie ein Upgrade für IBM MQ Operator oder den Warteschlangenmanager in einer Airgap-Umgebung durchführen können, müssen Sie die neuesten IBM Cloud Pak for Integration -Images spiegeln.

Beachten Sie, dass die ersten vier Schritte in dieser Task mit den Schritten identisch sind, die Sie bei „IBM MQ Operator in einer Airgap-Umgebung installieren” auf Seite 68ausführen.

Vorgehensweise

1. Umgebungsvariablen für Installationsprogramm und Image-Inventar erstellen

   Erstellen Sie die folgenden Umgebungsvariablen mit dem Imagenamen des Installationsprogramms und dem Imagebestand:

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

   Dabei steht *Versionsnummer* für die Version des Falls, den Sie zum Installieren der Airgap-Installation verwenden möchten. Eine Liste der verfügbaren Fallversionen finden Sie unter https://github.com/IBM/cloud-pak/tree/master/repo/case/ibm-mq. Lesen Sie die Informationen zur Versions-

unterstützung für den IBM MQ Operator , um festzustellen, welcher Operatorkanal ausgewählt werden muss.

2. Laden Sie das IBM MQ -Installationsprogramm und den Imagebestand herunter.

   Laden Sie das `ibm-mq` -Installationsprogramm und den Imagebestand auf den Bastionshost herunter:

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

3. Melden Sie sich als Clusteradministrator am Red Hat OpenShift Container Platform-Cluster an.

   Es folgt ein Beispielbefehl für die Anmeldung beim Red Hat OpenShift Container Platform-Cluster:

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

4. Spiegeln Sie die Images und konfigurieren Sie den Cluster.

   Führen Sie die folgenden Schritte aus, um die Images zu spiegeln und Ihren Cluster zu konfigurieren:

   Anmerkung: Verwenden Sie in keinem Befehl die Tilde innerhalb der Anführungszeichen. Verwenden Sie beispielsweise nicht `args "--registry registry --user registry_userid --pass registry_password --inputDir ~/offline"`. Die Tilde wird nicht aufgelöst und Ihre Befehle schlagen möglicherweise fehl.

   a. Speichern Sie Authentifizierungsnachweise für alle Docker-Quellenregistrys.

      Alle IBM Cloud-Platform Common Services, IBM MQ Operator-Image und IBM MQ-Advanced Developer-Images werden in öffentlichen Registrys gespeichert, für die keine Authentifizierung erforderlich ist. Für IBM MQ Advanced Server (nicht Developer), andere Produkte und Komponenten von Drittanbietern sind jedoch eine oder mehrere authentifizierte Registrys erforderlich. Für folgende Registrys ist eine Authentifizierung erforderlich:

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

      Weitere Informationen zu diesen Registrys finden Sie im Abschnitt Registry-Namensbereiche erstellen.

      Sie müssen folgenden Befehl ausführen, um Berechtigungsnachweise für alle Registrys zu konfigurieren, für die eine Authentifizierung erforderlich ist. Führen Sie den Befehl für jede dieser Registrys gesondert aus:

      ```
      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"
      ```

      Der Befehl speichert die Registry-Berechtigungsnachweise in einer Datei auf Ihrem Dateisystem an der Position `$HOME/.airgap/secrets`.

   b. Erstellen Sie Umgebungsvariablen mit den Verbindungsinformationen für die lokale Docker-Registry.

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

Hinweis: Die Docker-Registry verwendet Standardports, z. B. 80 oder 443. Wenn Ihre Docker -Registry einen vom Standard abweichenden Port verwendet, geben Sie den Port mit der Syntax *host*:*port*an. For example:

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

c. Konfigurieren Sie einen geheimen Schlüssel für die Authentifizierung für die lokale Docker-Registry.

Hinweis: Dieser Schritt muss nur einmal ausgeführt werden.

```
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_DO
CKER_PASSWORD}"
```

Der Befehl speichert die Registry-Berechtigungsnachweise in einer Datei auf Ihrem Dateisystem an der Position $HOME/.airgap/secrets.

d. Konfigurieren Sie einen geheimen Schlüssel für globale Image-Pull-Operationen und eine Richtlinie (**ImageContentSourcePolicy**).

i) Überprüfen Sie, ob ein Knotenneustart erforderlich ist.

- In Red Hat OpenShift Container Platform Version 4.4 und höher und in einer neuen Installation von IBM MQ Operator unter Verwendung von Airgap werden mit diesem Schritt alle Clusterknoten erneut gestartet. Die Clusterressourcen sind möglicherweise erst dann verfügbar, wenn der geheime Schlüssel für Pull-Operationen angewendet wird.
- In IBM MQ Operator 1.8wird CASE aktualisiert, um eine zusätzliche Spiegelungsquelle für Images einzuschließen. Daher wird ein Knotenneustart ausgelöst, wenn Sie ein Upgrade von früheren Versionen von IBM MQ Operator auf Version 1.8 oder höher durchführen.
- Um zu prüfen, ob dieser Schritt einen Knotenneustart erfordert, fügen Sie dem Code für diesen Schritt die Option --dry-run hinzu. Dadurch wird die neueste **ImageContentSourcePolicy** generiert und im Konsolenfenster (**stdout**) angezeigt. Wenn sich diese **ImageContentSourcePolicy** vom konfigurierten Cluster **ImageContentSourcePolicy**unterscheidet, erfolgt ein Neustart.

```
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 ${LO
CAL_DOCKER_PASSWORD} --inputDir $HOME/offline --dryRun"
```

ii) Führen Sie den Code für diesen Schritt ohne die Option --dry-run aus, um den globalen geheimen Schlüssel für Image-Pull-Operationen und **ImageContentSourcePolicy**zu konfigurieren:

```
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 ${LO
CAL_DOCKER_PASSWORD} --inputDir $HOME/offline"
```

e. Überprüfen Sie, ob die **ImageContentSourcePolicy**-Ressource erstellt wurde.

```
oc get imageContentSourcePolicy
```

f. Optional: Wenn Sie eine unsichere Registry verwenden, müssen Sie die lokale Registry zur Liste **insecureRegistries** des Clusters hinzufügen.

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

g. Überprüfen Sie den Status Ihres Clusters.

```
oc get nodes
```

Nachdem die **imageContentsourcePolicy** und der geheime Schlüssel für globale Image-Pull-Operationen angewendet wurden, wird als Knotenstatus entweder **Ready** (Bereit), **Scheduling** (In Planung) oder **Disabled** (Inaktiviert) angezeigt. Warten Sie, bis für alle Knoten der Status **Ready** angezeigt wird.

h. Spiegeln Sie die Images in die lokale Registry.

```
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_DO
CKER_PASSWORD} --inputDir $HOME/offline"
```

5. Aktualisieren Sie die Katalogquelle.

Verwenden Sie dasselbe Terminal wie bei den vorherigen Schritten.

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

Nächste Schritte

Sie können jetzt ein Upgrade für IBM MQ Operator und den Warteschlangenmanager durchführen, indem Sie eine der folgenden Tasks ausführen:

- „Upgrade von IBM MQ Operator über die Red Hat OpenShift-Webkonsole durchführen” auf Seite 77
- „Upgrade von IBM MQ Operator über die Red Hat OpenShift-CLI durchführen” auf Seite 79
- „Upgrade eines IBM MQ MQ-Warteschlangenmanagers über die Red Hat OpenShift-Webkonsole durchführen” auf Seite 80
- „Upgrade eines IBM MQ-Warteschlangenmanagers über die Red Hat OpenShift-CLI durchführen” auf Seite 81
- „Upgrade eines IBM MQ-Warteschlangenmanagers in Red Hat OpenShift über Platform Navigator durchführen” auf Seite 82

OpenShift CP4I Upgrade von IBM MQ Operator über die Red Hat OpenShift-Webkonsole durchführen

Das Upgrade von IBM MQ Operator kann über OperatorHub durchgeführt werden.

Vorbereitende Schritte

Melden Sie sich bei der Webkonsole Ihres Red Hat OpenShift-Clusters an.

Bevor Sie ein Upgrade für IBM MQ Operator in einer Air-Gap-Umgebung durchführen können, müssen Sie die neuesten IBM Cloud Pak for Integration -Images spiegeln. Siehe Upgrade für IBM MQ Operator oder Warteschlangenmanager in einer Airgap-Umgebung vorbereiten.

Vorgehensweise

1. Bestimmen Sie anhand der Informationen im Abschnitt „Versionsunterstützung für IBM MQ Operator” auf Seite 7, auf welchen Operatorkanal ein Upgrade durchgeführt werden soll.
2. Optional: Wenn Sie ein Upgrade von einer Version von IBM MQ Operator, die älter als 1.5 ist, auf IBM MQ Operator1.5 oder höher durchführen, müssen Sie zuerst ein Upgrade für die Version von IBM Cloud Pak foundational servicesdurchführen.

   Weitere Informationen finden Sie unter „Upgrade für IBM Cloud Pak foundational servicesmithilfe der Red Hat OpenShift-Webkonsole durchführen” auf Seite 78.
3. Führen Sie ein Upgrade für IBM MQ Operator durch. Neue Haupt- oder Nebenversionen von IBM MQ Operatorwerden über neue Subskriptionskanäle bereitgestellt. Um für Ihren Operator ein Upgrade auf eine neue Haupt- oder Nebenversion durchzuführen, müssen Sie den ausgewählten Kanal in Ihrer IBM MQ Operator-Subskription aktualisieren.
   a) Klicken Sie im Navigationsfenster auf **Operators** > **Installed Operators** (Operatoren > Installierte Operatoren).

      Es werden alle im angegebenen Projekt installierten Operatoren angezeigt.
   b) Wählen Sie **IBM MQ Operator** aus.
   c) Navigieren Sie zur Registerkarte **Subscription** (Subskription).
   d) Klicken Sie auf den **Channel** (Kanal).

      Das Fenster **Change Subscription Update Channel** (Subskriptionsaktualisierungskanal ändern) wird angezeigt.
   e) Wählen Sie den gewünschten Kanal aus und klicken Sie auf **Save** (Speichern).

      Der Operator wird auf die neueste Version aktualisiert, die für den neuen Kanal verfügbar ist. Weitere Informationen finden Sie unter „Versionsunterstützung für IBM MQ Operator” auf Seite 7.

Nächste Schritte

Wenn Sie ein Upgrade auf IBM Cloud Pak foundational services 3.7 durchgeführt haben, muss für alle Warteschlangenmanager, die eine IBM Cloud Pak for Integration-Lizenz nutzen, ein Upgrade oder Neustart durchgeführt werden. Weitere Informationen hierzu finden Sie unter „Upgrade eines IBM MQ MQ-Warteschlangenmanagers über die Red Hat OpenShift-Webkonsole durchführen” auf Seite 80.

OpenShift CP4I *Upgrade für IBM Cloud Pak foundational servicesmithilfe der Red Hat OpenShift-Webkonsole durchführen*

Wenn Sie ein Upgrade von einer Version von IBM MQ Operator, die älter als 1.5 ist, auf IBM MQ Operator1.5 oder höher durchführen, müssen Sie zuerst ein Upgrade für die Version von IBM Cloud Pak foundational servicesdurchführen.

Vorbereitende Schritte

Anmerkung: Sie müssen diese Task nur ausführen, wenn Sie ein Upgrade von einer Version von IBM MQ Operatordurchführen, die älter als 1.5 auf IBM MQ Operator1.5 oder höher ist.

CP4I Wenn Sie Warteschlangenmanager haben, die eine IBM Cloud Pak for Integration-Lizenz verwenden, ist nach diesem Upgrade ein Neustart des Warteschlangenmanagers erforderlich, um auf die Webkonsole zugreifen zu können, und Sie werden auch sehen wir andere Fehler in der Webkonsole geloggt werden. Sie können diese Fehler beheben, indem Sie ein Upgrade auf den neuesten Wert von `.spec.version`für Ihre ausgewählte IBM MQ-Version durchführen, nachdem das Operator-Upgrade abgeschlossen ist.

CP4I Wenn Sie über vorhandene Warteschlangenmanager verfügen und das IBM Cloud Pak for Integration Operations Dashboard verwenden, lesen Sie die Informationen unter „Implementierung oder Upgrade für IBM MQ 9.2.2 oder 9.2.3 mit Operations Dashboard-Integration in IBM Cloud Pak for Integration 2021.4 durchführen” auf Seite 116, bevor Sie ein Upgrade durchführen.

Vorgehensweise

1. Melden Sie sich bei der Webkonsole Ihres Red Hat OpenShift-Clusters an.
2. Klicken Sie im Navigationsfenster auf **Operators** > **Installed Operators** (Operatoren > Installierte Operatoren).

   Es werden alle im angegebenen Projekt installierten Operatoren angezeigt.
3. Wählen Sie den **IBM Cloud Pak foundational services-Operator** aus. Beachten Sie, dass dies vor Version 3.7 als **IBM Common Services Operator** (IBM Common Services Operator) bezeichnet wurde
4. Navigieren Sie zur Registerkarte **Abonnement**.
5. Klicken Sie auf **Kanal**.

   Das Fenster **Change Subscription Update Channel** (Subskriptionsaktualisierungskanal ändern) wird angezeigt.
6. Wählen Sie den **v3**-Kanal aus und klicken Sie dann auf **Speichern**.

   Der Operator IBM Cloud Pak foundational servicesführt ein Upgrade auf die neueste Version durch, die für den neuen Kanal verfügbar ist. Siehe „Versionsunterstützung für IBM MQ Operator” auf Seite 7.

Nächste Schritte

Sie können jetzt Upgrade für IBM MQ Operatordurchführen.

OpenShift CP4I Upgrade von IBM MQ Operator über die Red Hat OpenShift-CLI durchführen

Die IBM MQ Operatorkann über die Befehlszeile aktualisiert werden.

Vorbereitende Schritte

Melden Sie sich mit **`cloudctl login`** (für IBM Cloud Pak for Integration) oder **`oc login`** bei Ihrem Cluster an.

Bevor Sie ein Upgrade für IBM MQ Operator in einer Air-Gap-Umgebung durchführen können, müssen Sie die neuesten IBM Cloud Pak for Integration -Images spiegeln. Siehe Upgrade für IBM MQ Operator oder Warteschlangenmanager in einer Airgap-Umgebung vorbereiten.

Vorgehensweise

1. Bestimmen Sie anhand der Informationen im Abschnitt „Versionsunterstützung für IBM MQ Operator” auf Seite 7, auf welchen Operatorkanal ein Upgrade durchgeführt werden soll.
2. Optional: Wenn Sie ein Upgrade von einer Version von IBM MQ Operator, die älter als 1.5 ist, auf IBM MQ Operator1.5 oder höher durchführen, müssen Sie zuerst ein Upgrade für die Version von IBM Cloud Pak foundational servicesdurchführen.

   Weitere Informationen finden Sie unter „Upgrade für IBM Cloud Pak foundational servicesüber die Red Hat OpenShift-CLI durchführen” auf Seite 80.
3. Führen Sie ein Upgrade für IBM MQ Operator durch. Neue übergeordnete/untergeordnete IBM MQ Operator-Versionen werden über neue Subskriptionskanäle bereitgestellt. Um ein Upgrade Ihres Operators auf eine neue übergeordnete/untergeordnete Version durchzuführen, müssen Sie den ausgewählten Kanal in Ihrer IBM MQ Operator-Subskription aktualisieren.
   a) Stellen Sie sicher, dass der erforderliche IBM MQ Operator-Upgradekanal verfügbar ist.

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

   b) Korrigieren Sie die `Subscription`, um zum gewünschten Aktualisierungskanal zu wechseln (dabei ist *vX.Y* der im vorherigen Schritt angegebene gewünschte Aktualisierungskanal).

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

Nächste Schritte

Wenn Sie ein Upgrade auf IBM Cloud Pak foundational services 3.7 durchgeführt haben, muss für alle Warteschlangenmanager, die eine IBM Cloud Pak for Integration-Lizenz nutzen, ein Upgrade oder Neustart durchgeführt werden. Weitere Informationen hierzu finden Sie unter „Upgrade eines IBM MQ-Warteschlangenmanagers über die Red Hat OpenShift-CLI durchführen" auf Seite 81.

OpenShift CP4I *Upgrade für IBM Cloud Pak foundational servicesüber die Red Hat OpenShift-CLI durchführen*

Wenn Sie ein Upgrade von einer Version von IBM MQ Operator, die älter als 1.5 ist, auf IBM MQ Operator1.5 oder höher durchführen, müssen Sie zuerst ein Upgrade für die Version von IBM Cloud Pak foundational servicesdurchführen.

Vorbereitende Schritte

Anmerkung: Sie müssen diese Task nur ausführen, wenn Sie ein Upgrade von einer Version von IBM MQ Operatordurchführen, die älter als 1.5 auf IBM MQ Operator1.5 oder höher ist.

CP4I Wenn Sie über Warteschlangenmanager verfügen, die eine IBM Cloud Pak for Integration-Lizenz nutzen, ist nach diesem Upgrade ein Neustart des Warteschlangenmanagers erforderlich, um auf die Webkonsole zugreifen zu können, und Sie werden auch feststellen, dass andere Fehler in der Webkonsole protokolliert werden. Sie können diese Fehler beheben, indem Sie ein Upgrade auf den neuesten Wert von `.spec.version`für Ihre ausgewählte IBM MQ-Version durchführen, nachdem das Operator-Upgrade abgeschlossen ist.

CP4I Wenn Sie über vorhandene Warteschlangenmanager verfügen und das IBM Cloud Pak for Integration Operations Dashboard verwenden, lesen Sie die Informationen unter „Implementierung oder Upgrade für IBM MQ 9.2.2 oder 9.2.3 mit Operations Dashboard-Integration in IBM Cloud Pak for Integration 2021.4 durchführen" auf Seite 116, bevor Sie ein Upgrade durchführen.

Vorgehensweise

1. Melden Sie sich mit **`cloudctl login`** (für IBM Cloud Pak for Integration) oder **`oc login`** bei Ihrem Cluster an.
2. Stellen Sie sicher, dass der v3IBM Cloud Pak foundational services-Upgradekanal verfügbar ist.

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

3. Korrigieren Sie die `Subscription`, um zum gewünschten Aktualisierungskanal zu wechseln: v3

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

Nächste Schritte

Sie können jetzt Upgrade für IBM MQ Operatordurchführen.

OpenShift CP4I Upgrade eines IBM MQ MQ-Warteschlangenmanagers über die Red Hat OpenShift-Webkonsole durchführen

Ein mithilfe von IBM MQ Operator implementierter IBM MQ-Warteschlangenmanager kann in Red Hat OpenShift über den Operator-Hub aktualisiert werden.

Vorbereitende Schritte

- Melden Sie sich bei der Webkonsole Ihres Red Hat OpenShift-Clusters an.
- Stellen Sie sicher, dass Ihr IBM MQ Operator den gewünschten Aktualisierungskanal verwendet. Siehe „Upgrade für IBM MQ Operator und Warteschlangenmanager durchführen" auf Seite 74.

Bevor Sie ein Upgrade für den Warteschlangenmanager in einer Airgap-Umgebung durchführen können, müssen Sie die neuesten IBM Cloud Pak for Integration -Images spiegeln. Siehe Upgrade für IBM MQ Operator oder Warteschlangenmanager in einer Airgap-Umgebung vorbereiten.

Vorgehensweise

1. Klicken Sie im Navigationsfenster auf **Operators** > **Installed Operators** (Operatoren > Installierte Operatoren).

   Es werden alle im angegebenen Projekt installierten Operatoren angezeigt.
2. Wählen Sie **IBM MQ Operator** aus.

   Das Fenster **IBM MQ Operator** wird angezeigt.
3. Navigieren Sie zur Registerkarte **Queue Manager** (Warteschlangenmanager).

   Das Fenster mit den Warteschlangenmanagerdetails wird angezeigt.
4. Wählen Sie den Warteschlangenmanager aus, für den das Upgrade durchgeführt werden soll.
5. Navigieren Sie zur Registerkarte 'YAML'.
6. Passen Sie die Einträge in den folgenden Feldern dem gewünschten Upgrade der IBM MQ-Warteschlangenmanagerversion an, sofern erforderlich.

   - spec.version
   - spec.license.licence

   Eine Zuordnung von Kanälen zu IBM MQ Operator -Versionen und IBM MQ -Warteschlangenmanagerversionen finden Sie im Abschnitt „Versionsunterstützung für IBM MQ Operator” auf Seite 7 .
7. Speichern Sie die aktualisierte Warteschlangenmanager-YAML.

OpenShift CP4I Upgrade eines IBM MQ-Warteschlangenmanagers über die Red Hat OpenShift-CLI durchführen

Ein mithilfe von IBM MQ Operator implementierter IBM MQ-Warteschlangenmanager kann in Red Hat OpenShift über die Befehlszeilenschnittstelle (CLI) aktualisiert werden.

Vorbereitende Schritte

Um diese Schritte ausführen zu können, müssen Sie ein Clusteradministrator sein.

- Melden Sie sich mit `oc login` an der Befehlszeilenschnittstelle von Red Hat OpenShift an.
- Stellen Sie sicher, dass Ihr IBM MQ Operator den gewünschten Aktualisierungskanal verwendet. Siehe „Upgrade für IBM MQ Operator und Warteschlangenmanager durchführen” auf Seite 74.

Bevor Sie ein Upgrade für den Warteschlangenmanager in einer Airgap-Umgebung durchführen können, müssen Sie die neuesten IBM Cloud Pak for Integration -Images spiegeln. Siehe Upgrade für IBM MQ Operator oder Warteschlangenmanager in einer Airgap-Umgebung vorbereiten.

Vorgehensweise

Passen Sie in der Ressource **`QueueManager`** die Einträge in den folgenden Feldern dem gewünschten Upgrade der IBM MQ-Warteschlangenmanagerversion an, sofern erforderlich.

- spec.version
- spec.license.licence

Eine Zuordnung von Kanälen zu IBM MQ Operator -Versionen und IBM MQ -Warteschlangenmanagerversionen finden Sie im Abschnitt „Versionsunterstützung für IBM MQ Operator” auf Seite 7 .

Verwenden Sie folgenden Befehl:

```
oc edit queuemanager my_qmgr
```

Dabei steht *Meine_WSManGr* für den Namen der Warteschlangenmanagerressource, für die das Upgrade durchgeführt werden soll.

CP4I Upgrade eines IBM MQ-Warteschlangenmanagers in Red Hat OpenShift über Platform Navigator durchführen

Ein mithilfe von IBM MQ Operator implementierter IBM MQ-Warteschlangenmanager kann in Red Hat OpenShift über IBM Cloud Pak for Integration Platform Navigator aktualisiert werden.

Vorbereitende Schritte

- Melden Sie sich bei IBM Cloud Pak for Integration Platform Navigator in dem Namensbereich an, der den Warteschlangenmanager enthält, für den Sie ein Upgrade durchführen möchten.
- Stellen Sie sicher, dass Ihr IBM MQ Operator den gewünschten Aktualisierungskanal verwendet. Siehe „Upgrade für IBM MQ Operator und Warteschlangenmanager durchführen" auf Seite 74.

Bevor Sie ein Upgrade für den Warteschlangenmanager in einer Airgap-Umgebung durchführen können, müssen Sie die neuesten IBM Cloud Pak for Integration -Images spiegeln. Siehe Upgrade für IBM MQ Operator oder Warteschlangenmanager in einer Airgap-Umgebung vorbereiten.

Vorgehensweise

1. Klicken Sie auf der IBM Cloud Pak for Integration Platform Navigator-Startseite auf die Registerkarte **Runtimes** (Laufzeiten).
2. Warteschlangenmanager, für die ein Upgrade verfügbar ist, sind neben der **Version** durch den blauen Buchstaben **i** gekennzeichnet. Klicken Sie auf das **i**, um **New version available** (Neue Version verfügbar) anzuzeigen.
3. Klicken Sie auf die drei Punkte rechts neben dem Warteschlangenmanager, für den Sie das Upgrade durchführen möchten, und klicken Sie dann auf **Change version** (Version ändern).
4. Wählen Sie unter **Select a new channel or version** (Neuen Kanal oder neue Version auswählen) die erforderliche Upgrade-Version aus.
5. Klicken Sie auf **Change version** (Version ändern).

Ergebnisse

Der Warteschlangenmanager wird aktualisiert.

OpenShift CP4I Warteschlangenmanager mithilfe von IBM MQ Operator implementieren und konfigurieren

IBM MQ 9.1.5 und höher werden mithilfe von IBM MQ Operator in Red Hat OpenShift implementiert.

Informationen zu diesem Vorgang

Prozedur

- „Red Hat OpenShift-Projekt für IBM MQ vorbereiten" auf Seite 82.
- „Warteschlangenmanager in einem Red Hat OpenShift Container Platform-Cluster bereitstellen" auf Seite 84.

OpenShift CP4I Red Hat OpenShift-Projekt für IBM MQ vorbereiten

Bereiten Sie Ihren Red Hat OpenShift Container Platform-Cluster für die Bereitstellung eines Warteschlangenmanagers vor.

Prozedur

- „Red Hat OpenShift-Projekt für IBM MQ über die Red Hat OpenShift-Webkonsole vorbereiten” auf Seite 83.
- „Red Hat OpenShift-Projekt für IBM MQ über die Red Hat OpenShift-CLI vorbereiten” auf Seite 84.

Zugehörige Tasks

„Warteschlangenmanager in einem Red Hat OpenShift Container Platform-Cluster bereitstellen” auf Seite 84

Verwenden Sie die angepasste QueueManager-Ressource, um einen Warteschlangenmanager in einem Red Hat OpenShift Container Platform-Cluster bereitzustellen.

OpenShift CP4I *Red Hat OpenShift-Projekt für IBM MQ über die Red Hat OpenShift-Webkonsole vorbereiten*

Sie können Ihren Cluster für Red Hat OpenShift Container Platform so vorbereiten, dass er für die Bereitstellung eines Warteschlangenmanagers mithilfe von IBM MQ Operator bereit ist. Diese Aufgabe sollte von einem Projektadministrator ausgeführt werden.

Vorbereitende Schritte

Anmerkung: Wenn Sie planen, IBM MQ in einem Projekt mit anderen, bereits installierten IBM Cloud Pak for Integration-Komponenten zu verwenden, müssen Sie diese Anweisungen möglicherweise nicht befolgen.

Melden Sie sich bei der Webkonsole Ihres Red Hat OpenShift-Clusters an.

Informationen zu diesem Vorgang

Die Images für IBM MQ Operator werden aus einer Container-Registry extrahiert, die eine Überprüfung der Lizenzberechtigung durchführt. Für diese Überprüfung ist ein Berechtigungsschlüssel erforderlich, der in dem geheimen Schlüssel `docker-registry` für Pull-Operationen gespeichert wird. Wenn Sie noch keinen Berechtigungsschlüssel haben, befolgen Sie diese Anweisungen, um einen Berechtigungsschlüssel abzurufen und einen geheimen Schlüssel für Pull-Operationen zu erstellen.

Vorgehensweise

1. Rufen Sie den Berechtigungsschlüssel ab, der Ihrer ID zugeordnet ist.
   a) Melden Sie sich an der MyIBM Container Software Library mit der IBM-ID und dem Kennwort an, die der berechtigten Software zugeordnet sind.
   b) Wählen Sie im Abschnitt **Entitlement keys** (Berechtigungsschlüssel) die Option **Copy key** (Schlüssel kopieren) aus, um den Berechtigungsschlüssel in die Zwischenablage zu kopieren.
2. Erstellen Sie einen geheimen Schlüssel, der Ihren Berechtigungsschlüssel enthält, in dem Projekt, in dem Sie Ihren Warteschlangenmanager implementieren möchten.
   a) Klicken Sie im Navigationsfenster auf **Workloads** > **Secret** (Geheimer Schlüssel).
   Die Seite 'Secrets' wird angezeigt.
   b) Wählen Sie in der Dropdown-Liste **Project** das Projekt aus, in dem IBM MQ installiert werden soll.
   c) Klicken Sie auf die Schaltfläche **Create** (Erstellen) und wählen Sie **Image Pull Secret** (Geheimer Schlüssel für Image-Pull-Operation) aus.
   d) Geben Sie `ibm-entitlement-key` im Feld **Name** ein.
   e) Geben Sie `cp.icr.io` im Feld **Registry Server Address** (Adresse des Registry-Servers) ein.
   f) Geben Sie `cp` im Feld **Username** (Benutzername) ein.
   g) Geben Sie im Feld **Password** (Kennwort) den Berechtigungsschlüssel ein, den Sie im vorherigen Schritt kopiert haben.
   h) Geben Sie im Feld **Email** (E-Mail) die IBM ID ein, die der berechtigten Software zugeordnet ist.

Nächste Schritte

„Warteschlangenmanager über die Red Hat OpenShift-Webkonsole implementieren” auf Seite 86

OpenShift CP4I *Red Hat OpenShift-Projekt für IBM MQ über die Red Hat OpenShift-CLI vorbereiten*

Sie können Ihren Cluster für Red Hat OpenShift Container Platform so vorbereiten, dass er für die Bereitstellung eines Warteschlangenmanagers mithilfe von IBM MQ Operator bereit ist. Diese Aufgabe sollte von einem Projektadministrator ausgeführt werden.

Vorbereitende Schritte

Anmerkung: Wenn Sie planen, IBM MQ in einem Projekt mit anderen, bereits installierten IBM Cloud Pak for Integration-Komponenten zu verwenden, müssen Sie diese Anweisungen möglicherweise nicht befolgen.

Melden Sie sich mit **`cloudctl login`** (für IBM Cloud Pak for Integration) oder **`oc login`** bei Ihrem Cluster an.

Informationen zu diesem Vorgang

Die Images für IBM MQ Operator werden aus einer Container-Registry extrahiert, die eine Überprüfung der Lizenzberechtigung durchführt. Für diese Überprüfung ist ein Berechtigungsschlüssel erforderlich, der in dem geheimen Schlüssel `docker-registry` für Pull-Operationen gespeichert wird. Wenn Sie noch keinen Berechtigungsschlüssel haben, befolgen Sie diese Anweisungen, um einen Berechtigungsschlüssel abzurufen und einen geheimen Schlüssel für Pull-Operationen zu erstellen.

Vorgehensweise

1. Rufen Sie den Berechtigungsschlüssel ab, der Ihrer ID zugeordnet ist.
   a) Melden Sie sich an der MyIBM Container Software Library mit der IBM-ID und dem Kennwort an, die der berechtigten Software zugeordnet sind.
   b) Wählen Sie im Abschnitt **Entitlement keys** (Berechtigungsschlüssel) die Option **Copy key** (Schlüssel kopieren) aus, um den Berechtigungsschlüssel in die Zwischenablage zu kopieren.
2. Erstellen Sie einen geheimen Schlüssel, der Ihren Berechtigungsschlüssel enthält, in dem Projekt, in dem Sie Ihren Warteschlangenmanager implementieren möchten.
   Führen Sie den folgenden Befehl aus, wobei *<entitlement-key>* der in Schritt 1 abgerufene Schlüssel ist und *<user-email>* die IBM ID, die der berechtigten Software zugeordnet ist.

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

Nächste Schritte

„Warteschlangenmanager über die Red Hat OpenShift-CLI implementieren” auf Seite 87

OpenShift CP4I Warteschlangenmanager in einem Red Hat OpenShift Container Platform-Cluster bereitstellen

Verwenden Sie die angepasste QueueManager-Ressource, um einen Warteschlangenmanager in einem Red Hat OpenShift Container Platform-Cluster bereitzustellen.

Prozedur

- CP4I

„Warteschlangenmanager mit dem IBM Cloud Pak for Integration Platform Navigator implementieren” auf Seite 85.

- OpenShift
  „Warteschlangenmanager über die Red Hat OpenShift-Webkonsole implementieren” auf Seite 86.
- OpenShift
  „Warteschlangenmanager über die Red Hat OpenShift-CLI implementieren” auf Seite 87.

Zugehörige Tasks

„Beispiele für die Konfiguration eines Warteschlangenmanagers” auf Seite 89
Ein Warteschlangenmanager kann durch Anpassung des Inhalts der angepassten QueueManager-Ressource konfiguriert werden.

CP4I *Warteschlangenmanager mit dem IBM Cloud Pak for Integration Platform Navigator implementieren*

Verwenden Sie die angepasste QueueManager-Ressource, um einen Warteschlangenmanager mithilfe von IBM Cloud Pak for Integration Platform Navigator in einem Red Hat OpenShift Container Platform-Cluster zu implementieren. Diese Aufgabe sollte von einem Projektadministrator ausgeführt werden

Vorbereitende Schritte

Starten Sie IBM Cloud Pak for Integration Platform Navigator in einem Browser.

Wenn dies das erste Mal ist, dass ein Warteschlangenmanager in diesem Red Hat OpenShift-Projekt implementiert wird, führen Sie die Schritte im Abschnitt „Red Hat OpenShift-Projekt für IBM MQ vorbereiten” auf Seite 82 aus.

Vorgehensweise

1. Implementieren Sie einen Warteschlangenmanager.

   Im folgenden Beispiel wird ein "Schnellstart"-Warteschlangenmanager, der einen ephemeren (nicht persistenten) Speicher verwendet, implementiert und die MQ-Sicherheit inaktiviert. Nachrichten bleiben bei Neustarts des Warteschlangenmanagers nicht erhalten. Sie können die Konfiguration anpassen, um viele Warteschlangenmanagereinstellungen zu ändern.

   a) Klicken Sie in der IBM Cloud Pak for Integration Platform Navigatorauf **Verwaltung** und anschließend auf **Integrationslaufzeiten**. Klicken Sie in älteren Versionen von IBM Cloud Pak for Integration Platform Navigator auf **Laufzeit und Instanzen**.

   b) Klicken Sie auf **Create instance** (Instanz erstellen).

   c) Wählen Sie **Messaging**aus und klicken Sie auf **Weiter**. Klicken Sie in älteren Versionen von IBM Cloud Pak for Integration Platform Navigator auf **Warteschlangenmanager** und klicken Sie auf **Weiter**.

   Das Formular zum Erstellen einer Instanz eines `QueueManager` wird angezeigt.

   Anmerkung: Sie können auch auf **Code** klicken, um die YAML-Konfigurationsdatei für den `QueueManager` anzuzeigen oder zu ändern.

   d) Überprüfen oder aktualisieren Sie im Abschnitt **Details** das Feld **Name** und geben Sie den **Namespace** an, in dem die Warteschlangenmanagerinstanz erstellt werden soll.

   e) Wenn Sie die Lizenzvereinbarung für IBM Cloud Pak for Integration akzeptieren, ändern Sie die Einstellung von **License acceptance** (Lizenzakzeptanz) in **On**.

   Um einen Warteschlangenmanager implementieren zu können, müssen Sie die Lizenz akzeptieren.

   f) Überprüfen oder aktualisieren Sie im Abschnitt **Warteschlangenmanager** den **Namen** des zugrunde liegenden Warteschlangenmanagers. Verwenden Sie in älteren Versionen von IBM Cloud Pak for Integration Platform Navigator den Abschnitt **Warteschlangenmanager-Konfiguration**.

Standardmäßig entspricht der Name des von IBM MQ-Clientanwendungen verwendeten Warteschlangenmanagers dem Namen des `QueueManager`, wobei jedoch alle ungültigen Zeichen (z. B. Bindestriche) entfernt werden.

g) Klicken Sie auf **Erstellen**

Es wird jetzt die Liste der Warteschlangenmanager im aktuellen Projekt (Namensbereich) angezeigt. Der neue `QueueManager` sollte den Status `Pending` (Anstehend) haben.

2. Überprüfen Sie, ob der Warteschlangenmanager aktiv ist.

Die Erstellung ist beendet, wenn der `QueueManager` den Status `Running` (Aktiv) hat.

Zugehörige Tasks

„Route für Verbindung zu einem Warteschlangenmanager von außerhalb eines Red Hat OpenShift-Clusters konfigurieren" auf Seite 113
Sie benötigen eine Red Hat OpenShift -Route, um eine Anwendung von außerhalb eines Red Hat OpenShift -Clusters mit einem IBM MQ -Warteschlangenmanager zu verbinden. Sie müssen TLS auf Ihrem IBM MQ -Warteschlangenmanager und Ihrer Clientanwendung aktivieren, da SNI nur im TLS-Protokoll verfügbar ist, wenn ein TLS 1.2 oder ein höheres Protokoll verwendet wird. Der Red Hat OpenShift Container Platform Router verwendet SNI für Routing-Anforderungen an den IBM MQ-Warteschlangenmanager.

„Verbindung zur IBM MQ Console herstellen, die in einem Red Hat OpenShift-Cluster implementiert ist" auf Seite 120
Sie können eine Verbindung zur IBM MQ Console eines Warteschlangenmanagers herstellen, die in einem Red Hat OpenShift Container Platform-Cluster implementiert wurde.

OpenShift CP4I *Warteschlangenmanager über die Red Hat OpenShift-Webkonsole implementieren*

Verwenden Sie die angepasste QueueManager-Ressource, um einen Warteschlangenmanager über die Red Hat OpenShift-Webkonsole in einem Red Hat OpenShift Container Platform-Cluster zu implementieren. Diese Aufgabe sollte von einem Projektadministrator ausgeführt werden

Vorbereitende Schritte

Melden Sie sich bei der Webkonsole Ihres Red Hat OpenShift-Clusters an. Sie müssen ein vorhandenes Projekt (Namensbereich), das Sie verwenden möchten, auswählen oder ein neues erstellen.

Wenn dies das erste Mal ist, dass ein Warteschlangenmanager in diesem Red Hat OpenShift-Projekt implementiert wird, führen Sie die Schritte im Abschnitt „Red Hat OpenShift-Projekt für IBM MQ vorbereiten" auf Seite 82 aus.

Vorgehensweise

1. Implementieren Sie einen Warteschlangenmanager.

   Im folgenden Beispiel wird ein "Schnellstart"-Warteschlangenmanager, der einen ephemeren (nicht persistenten) Speicher verwendet, implementiert und die MQ-Sicherheit inaktiviert. Nachrichten bleiben bei Neustarts des Warteschlangenmanagers nicht erhalten. Sie können die Konfiguration anpassen, um viele Warteschlangenmanagereinstellungen zu ändern.

   a) Klicken Sie in der Red Hat OpenShift-Webkonsole im Navigationsfenster auf **Operators** > **Installed Operators** (Operatoren > Installierte Operatoren).

   b) Klicken Sie auf **IBM MQ**.

   c) Klicken Sie auf die Registerkarte **Queue Manager** (Warteschlangenmanager).

   d) Klicken Sie auf die Schaltfläche **Create QueueManager** (QueueManager erstellen).

   Es wird ein YAML-Editor mit einer YAML-Beispieldatei für eine `QueueManager`-Ressource angezeigt.

   Anmerkung: Sie können auch auf **Edit Form** (Formular bearbeiten) klicken, um die `QueueManager`-Konfiguration anzuzeigen oder zu ändern.

e) Wenn Sie die Lizenzvereinbarung akzeptieren, ändern Sie die Einstellung von **License acceptance** (Lizenzakzeptanz) in **On**.

IBM MQ ist unter mehreren verschiedenen Lizenzen verfügbar. Weitere Informationen zu den gültigen Lizenzen finden Sie im Abschnitt „Lizenzierungsreferenz für mq.ibm.com/v1beta1" auf Seite 132. Um einen Warteschlangenmanager implementieren zu können, müssen Sie die Lizenz akzeptieren.

f) Klicken Sie auf **Erstellen**

Es wird jetzt die Liste der Warteschlangenmanager im aktuellen Projekt (Namensbereich) angezeigt. Der neue `QueueManager` sollte den Status `Pending` (Anstehend) haben.

2. Überprüfen Sie, ob der Warteschlangenmanager aktiv ist.

Die Erstellung ist beendet, wenn der `QueueManager` den Status `Running` (Aktiv) hat.

Zugehörige Tasks

„Route für Verbindung zu einem Warteschlangenmanager von außerhalb eines Red Hat OpenShift-Clusters konfigurieren" auf Seite 113
Sie benötigen eine Red Hat OpenShift -Route, um eine Anwendung von außerhalb eines Red Hat OpenShift -Clusters mit einem IBM MQ -Warteschlangenmanager zu verbinden. Sie müssen TLS auf Ihrem IBM MQ -Warteschlangenmanager und Ihrer Clientanwendung aktivieren, da SNI nur im TLS-Protokoll verfügbar ist, wenn ein TLS 1.2 oder ein höheres Protokoll verwendet wird. Der Red Hat OpenShift Container Platform Router verwendet SNI für Routing-Anforderungen an den IBM MQ-Warteschlangenmanager.

„Verbindung zur IBM MQ Console herstellen, die in einem Red Hat OpenShift-Cluster implementiert ist" auf Seite 120
Sie können eine Verbindung zur IBM MQ Console eines Warteschlangenmanagers herstellen, die in einem Red Hat OpenShift Container Platform-Cluster implementiert wurde.

OpenShift CP4I *Warteschlangenmanager über die Red Hat OpenShift-CLI implementieren*

Verwenden Sie die angepasste QueueManager-Ressource, um einen Warteschlangenmanager über die Befehlszeilenschnittstelle (Command Line Interface, CLI) in einem Red Hat OpenShift Container Platform-Cluster zu implementieren. Diese Aufgabe sollte von einem Projektadministrator ausgeführt werden

Vorbereitende Schritte

Sie müssen die Red Hat OpenShift Container Platform-Befehlszeilenschnittstelle (CLI) installieren.

Melden Sie sich mit **`cloudctl login`** (für IBM Cloud Pak for Integration) oder **`oc login`** bei Ihrem Cluster an.

Wenn dies das erste Mal ist, dass ein Warteschlangenmanager in diesem Red Hat OpenShift-Projekt implementiert wird, führen Sie die Schritte im Abschnitt „Red Hat OpenShift-Projekt für IBM MQ vorbereiten" auf Seite 82 aus.

Vorgehensweise

1. Implementieren Sie einen Warteschlangenmanager.

Im folgenden Beispiel wird ein "Schnellstart"-Warteschlangenmanager, der einen ephemeren (nicht persistenten) Speicher verwendet, implementiert und die MQ-Sicherheit inaktiviert. Nachrichten bleiben bei Neustarts des Warteschlangenmanagers nicht erhalten. Sie können den Inhalt der YAML-Datei anpassen, um viele Warteschlangenmanagereinstellungen zu ändern.

a) `QueueManager`-YAML-Datei erstellen

Um beispielsweise einen Basiswarteschlangenmanager in IBM Cloud Pak for Integration zu installieren, erstellen Sie die Datei 'mq-quickstart.yaml' mit folgendem Inhalt:

```
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"
```

Wichtig: Wenn Sie die Lizenzvereinbarung für IBM Cloud Pak for Integrationakzeptieren, ändern Sie `accept: false`in `accept: true`. Details zur Lizenz finden Sie im Abschnitt „Lizenzierungsreferenz für mq.ibm.com/v1beta1" auf Seite 132.

Dieses Beispiel schließt auch einen mit dem Warteschlangenmanager bereitgestellten Web-Server ein, wobei die Webkonsole mit Single Sign-On mit IBM Cloud Pak Identity and Access Manager aktiviert ist.

Um einen Basiswarteschlangenmanager unabhängig von IBM Cloud Pak for Integration zu installieren, erstellen Sie die Datei 'mq-quickstart.yaml' mit folgendem Inhalt:

```
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"
```

Wichtig: Wenn Sie die MQ-Lizenzvereinbarung akzeptieren, ändern Sie `accept: false`in `accept: true`. Details zur Lizenz finden Sie im Abschnitt „Lizenzierungsreferenz für mq.ibm.com/v1beta1" auf Seite 132.

b) Erstellen Sie das `QueueManager`-Objekt

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

2. Überprüfen Sie, ob der Warteschlangenmanager aktiv ist.

   Sie können die Implementierung überprüfen, durch die Ausführung von

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

   , und Sie darauffolgend den Status überprüfen.

   Führen Sie beispielsweise

   ```
   oc describe queuemanager quickstart
   ```

und prüfen Sie, ob das Feld `status.Phase`den Wert `Running`enthält

Zugehörige Tasks

„Route für Verbindung zu einem Warteschlangenmanager von außerhalb eines Red Hat OpenShift-Clusters konfigurieren” auf Seite 113
Sie benötigen eine Red Hat OpenShift -Route, um eine Anwendung von außerhalb eines Red Hat OpenShift -Clusters mit einem IBM MQ -Warteschlangenmanager zu verbinden. Sie müssen TLS auf Ihrem IBM MQ -Warteschlangenmanager und Ihrer Clientanwendung aktivieren, da SNI nur im TLS-Protokoll verfügbar ist, wenn ein TLS 1.2 oder ein höheres Protokoll verwendet wird. Der Red Hat OpenShift Container Platform Router verwendet SNI für Routing-Anforderungen an den IBM MQ-Warteschlangenmanager.

„Verbindung zur IBM MQ Console herstellen, die in einem Red Hat OpenShift-Cluster implementiert ist” auf Seite 120
Sie können eine Verbindung zur IBM MQ Console eines Warteschlangenmanagers herstellen, die in einem Red Hat OpenShift Container Platform-Cluster implementiert wurde.

OpenShift CP4I Beispiele für die Konfiguration eines Warteschlangenmanagers

Ein Warteschlangenmanager kann durch Anpassung des Inhalts der angepassten QueueManager-Ressource konfiguriert werden.

Informationen zu diesem Vorgang

Die folgenden Beispiele zeigen, wie ein Warteschlangenmanager mit der YAML-Datei 'QueueManager' konfiguriert wird.

Prozedur

- „Beispiel: Unterstützung von MQSC- und INI-Dateien” auf Seite 89
- „Beispiel: TLS konfigurieren” auf Seite 90

OpenShift CP4I *Beispiel: Unterstützung von MQSC- und INI-Dateien*

In diesem Beispiel wird eine Kubernetes-ConfigMap mit zwei MQSC-Dateien und einer INI-Datei erstellt. Anschließend wird ein Warteschlangenmanager bereitgestellt, der diese MQSC- und INI-Dateien verarbeitet.

Informationen zu diesem Vorgang

MQSC- und INI-Dateien können bei der Bereitstellung eines Warteschlangenmanagers bereitgestellt werden. Die MQSC- und INI-Daten müssen in mindestens einer ConfigMap und mindestens einem Secret von Kubernetes definiert werden. Diese müssen in dem Namensbereich (Projekt) erstellt werden, in dem Sie den Warteschlangenmanager bereitstellen.

Anmerkung: Ein Kubernetes-Secret sollte verwendet werden, wenn die MQSC- oder INI-Dateien sensible Daten enthalten.

Für die Bereitstellung von MQSC- und INI-Dateien auf diese Weise wird IBM MQ Operator 1.1 oder höher vorausgesetzt.

Beispiel

Im folgenden Beispiel wird eine Kubernetes-ConfigMap mit zwei MQSC-Dateien und einer INI-Datei erstellt. Anschließend wird ein Warteschlangenmanager bereitgestellt, der diese MQSC- und INI-Dateien verarbeitet.

Beispiel-ConfigMap: Wenden Sie die folgende YAML in Ihrem Cluster an:

```
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
```

Beispiel-Warteschlangenmanager: Stellen Sie Ihren Warteschlangenmanager über die Befehlszeile oder IBM Cloud Pak for Integration Platform Navigator mit der folgenden Konfiguration bereit:

```
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
```

Wichtig: Wenn Sie die Lizenzvereinbarung für IBM Cloud Pak for Integrationakzeptieren, ändern Sie `accept: false`in `accept: true`. Details zur Lizenz finden Sie im Abschnitt Lizenzreferenz für mq.ibm.com/v1beta1.

Weitere Informationen:

- Ein Warteschlangenmanager kann für eine einzelne Kubernetes-ConfigMap oder ein einzelnes Secret (wie in diesem Beispiel) oder für mehrere Kubernetes-ConfigMaps und -Secrets konfiguriert werden.
- Sie können festlegen, dass alle MQSC- und INI-Daten einer Kubernetes-ConfigMap oder eines Secret verwendet werden (wie in diesem Beispiel) oder Sie können jeden Warteschlangenmanager so konfigurieren, dass er nur einen Teil der verfügbaren Dateien verwendet.
- MQSC- und INI-Dateien werden in alphabetischer Reihenfolge basierend auf ihrem Schlüssel verarbeitet. `example1.mqsc` wird also unabhängig von der Reihenfolge in der Warteschlangenmanagerkonfiguration immer vor `example2.mqsc` verarbeitet.
- Wenn mehrere MQSC- oder INI-Dateien in mehreren Kubernetes-ConfigMaps oder -Secrets den gleichen Schlüssel aufweisen, erfolgt deren Verarbeitung in der Reihenfolge, in der die Dateien in der Warteschlangenmanagerkonfiguration definiert sind.

OpenShift CP4I Linux *Beispiel: TLS konfigurieren*

In diesem Beispiel wird ein Warteschlangenmanager mit IBM MQ Operator in Red Hat OpenShift Container Platform bereitgestellt. Zwischen einem Beispielclient und dem Warteschlangenmanager wird eine unidirektionale TLS-Kommunikation konfiguriert. Die erfolgreiche Konfiguration wird durch Einreihen und Abrufen von Nachrichten nachgewiesen.

Vorbereitende Schritte

Die folgenden Schritte müssen als Voraussetzung für dieses Beispiel durchgeführt worden sein:

- Installieren Sie den IBM MQ client und fügen Sie `samp/bin` und `bin` Ihrem *PATH* hinzu. Sie benötigen die Anwendungen **runmqakm**, **amqsputc** und **amqsgetc**, die wie folgt mit dem IBM MQ client installiert werden können:
  - Windows Linux Für Windows und Linux: Installieren Sie den weiterverteilbaren Client von IBM MQ für Ihr Betriebssystem von https://ibm.biz/mq92redistclients
  - macOS Für Mac: Laden Sie die IBM MQ MacOS Toolkitherunter und richten Sie sie ein: https://developer.ibm.com/tutorials/mq-macos-dev/
- Installieren Sie das OpenSSL-Tool für Ihr Betriebssystem.
- Erstellen Sie für dieses Beispiel ein Projekt bzw. einen Namensbereich für Red Hat OpenShift Container Platform (OCP).
- Melden Sie sich über die Befehlszeile bei dem OCP-Cluster an und wechseln Sie zum oben genannten Namensbereich.
- Stellen Sie sicher, dass IBM MQ Operator installiert und in dem oben genannten Namensbereich verfügbar ist.

Informationen zu diesem Vorgang

Dieses Beispiel stellt eine angepasste Ressourcen-YAML bereit, die den in Red Hat OpenShift Container Platform bereitzustellenden Warteschlangenmanager definiert. Darüber hinaus beschreibt es alle weiteren Schritte, die zur Bereitstellung des Warteschlangenmanagers mit aktiviertem TLS erforderlich sind. Nach Abschluss der Konfiguration des Warteschlangenmanagers mit TLS wird durch Einreihen und Abrufen von Nachrichten die Funktionsfähigkeit der Konfiguration überprüft.

Erstellen Sie für den IBM MQ-Server einen privaten TLS-Schlüssel und Zertifikate

Die folgenden Codebeispiele zeigen, wie ein selbst signiertes Zertifikat für den Warteschlangenmanager erstellt wird und wie das Zertifikat einer Schlüsseldatenbank hinzugefügt wird, die als Truststore für den Client fungiert. Falls Sie bereits über einen privaten Schlüssel und ein Zertifikat verfügen, können Sie diese stattdessen verwenden.

Beachten Sie, dass selbst signierte Zertifikate nur für Entwicklungszwecke verwendet werden sollten.

Erstellen Sie im aktuellen Verzeichnis einen selbst signierten privaten Schlüssel und ein öffentliches Zertifikat.

Führen Sie den folgenden Befehl aus:

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

Fügen Sie den öffentlichen Schlüssel für den Server einer Clientschlüsseldatenbank hinzu.

Die Schlüsseldatenbank dient der Clientanwendung als Truststore.

Erstellen Sie die Clientschlüsseldatenbank:

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

Fügen Sie den zuvor generierten öffentlichen Schlüssel zur Clientschlüsseldatenbank hinzu:

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

Konfigurieren Sie TLS-Zertifikate für die Bereitstellung des Warteschlangenmanagers.

Ziel ist es, dass Ihr Warteschlangenmanager Schlüssel und Zertifikat abrufen und anwenden sowie ein Kubernetes-TLS-Secret erstellen kann, das die zuvor erstellten Dateien referenziert. Achten Sie bei der Konfiguration darauf, dass Sie den zuvor erstellten Namensbereich verwenden.

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

Erstellen Sie eine Konfigurationszuordnung (ConfigMap) mit MQSC-Befehlen.

Erstellen Sie eine Kubernetes-Konfigurationszuordnung (ConfigMap) mit MQSC-Befehlen zur Erstellung einer neuen Warteschlange und eines SVRCONN-Kanals sowie zum Hinzufügen eines Kanalauthentifizierungsdatensatzes, der allen Benutzern mit Ausnahme von Benutzern mit dem Namen *nobody* (diese werden blockiert) Zugriff auf den Kanal erlaubt.

Beachten Sie, dass diese Vorgehensweise ausschließlich für Entwicklungszwecke geeignet ist.

Stellen Sie sicher, dass Sie sich im zuvor erstellten Namensbereich befinden (siehe Vorbereitungen), und geben Sie dann in der OCP-Benutzerschnittstelle oder über die Befehlszeile die folgende YAML ein.

```
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('A□
NY_TLS12_OR_HIGHER')
    SET CHLAUTH(SECUREQMCHL) TYPE(BLOCKUSER) USERLIST('nobody') ACTION(ADD)
```

Erstellen Sie die erforderliche OCP-Route.

Stellen Sie sicher, dass Sie sich im zuvor erstellten Namensbereich befinden, und geben Sie dann in der OCP-Benutzerschnittstelle oder über die Befehlszeile die folgende YAML ein.

```
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
```

Beachten Sie, dass Red Hat OpenShift Container Platform Router die SNI für die Weiterleitung von Anforderungen an den IBM MQ-Warteschlangenmanager verwendet. Wenn Sie den Kanalnamen in der MQSC der zuvor erstellten Konfigurationszuordnung geändert haben, müssen Sie auch hier sowie in der später erstellten CCDT-Datei das Hostfeld ändern. Weitere Informationen finden Sie unter „Route für Verbindung zu einem Warteschlangenmanager von außerhalb eines Red Hat OpenShift-Clusters konfigurieren” auf Seite 113.

Stellen Sie den Warteschlangenmanager bereit.

Wichtig: In diesem Beispiel inaktivieren wir die Autorisierung auf dem Warteschlangenmanager mit der Variablen *MQSNOAUT*. Dadurch können wir uns ganz auf die Schritte konzentrieren, die zur Verbindung eines Clients mit TLS erforderlich sind. In einer IBM MQ-Produktionsumgebung wird hiervon abgeraten. Jede Anwendung, die eine Verbindung herstellt, hätte bei dieser Einstellung vollständige Verwaltungsbefugnisse, ohne jeglichen Mechanismus, die Berechtigungen für einzelne Anwendungen herabzusetzen.

Erstellen Sie mit der folgenden angepassten YAML-Ressource einen neuen Warteschlangenmanager. Diese Ressource verweist auf die zuvor erstellte Konfigurationszuordnung, den zuvor erstellten geheimen Schlüssel und die Variable *MQSNOAUT*.

Stellen Sie sicher, dass Sie sich im zuvor erstellten Namensbereich befinden, und geben Sie dann in der OCP-Benutzerschnittstelle, über die Befehlszeile oder in IBM Cloud Pak for Integration Platform

Navigator die folgende YAML ein. Überprüfen Sie, ob die richtige Lizenz angegeben wurde, und akzeptieren Sie die Lizenz, indem Sie `false`in `true`ändern.

```
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
```

Vergewissern Sie sich, dass der Warteschlangenmanager aktiv ist.

Der Warteschlangenmanager ist nun bereitgestellt. Vergewissern Sie sich, dass er sich im Status `Running`befindet, bevor Sie fortfahren. Beispiel:

```
oc get qmgr secureqm
```

Testen Sie die Verbindung zum Warteschlangenmanager.

Für einen Test der Konfiguration des Warteschlangenmanagers mit unidirektionaler TLS-Kommunikation können Sie die Beispielanwendungen **amqsputc** und **amqsgetc** verwenden:

Suchen Sie den Hostnamen des Warteschlangenmanagers.

Verwenden Sie den folgenden Befehl, um den vollständig qualifizierten Hostnamen des Warteschlangenmanagers für die Route `secureqm-ibm-mq-qm`zu suchen:

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

Geben Sie die Details des Warteschlangenmanagers an.

Erstellen Sie eine Datei `CCDT.JSON`, in der die Details des Warteschlangenmanagers angegeben sind. Ersetzen Sie den Hostwert durch den im vorangegangenen Schritt gefundenen Hostnamen.

```
{
    "channel":
    [
        {
            "name": "SECUREQMCHL",
            "clientConnection":
            {
                "connection":
                [
                    {
                        "host": "<hostname from previous step>",
                        "port": 443
                    }
```

```
                ],
                "queueManager": "SECUREQM"
            },
            "transmissionSecurity":
            {
              "cipherSpecification": "ECDHE_RSA_AES_128_CBC_SHA256"
            },
            "type": "clientConnection"
        }
    ]
}
```

Exportieren Sie Umgebungsvariablen.

Exportieren Sie die folgenden Umgebungsvariablen mit der für Ihr Betriebssystem geeigneten Methode. Diese Variablen werden von **amqsputc** und **amqsgetc** gelesen.

Aktualisieren Sie den Pfad zu den Dateien auf Ihrem System:

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

Reihen Sie Nachrichten in die Warteschlange ein.

Führen Sie den folgenden Befehl aus:

```
amqsputc EXAMPLE.QUEUE SECUREQM
```

Bei einer erfolgreichen Verbindung mit dem Warteschlangenmanager wird die folgende Antwort ausgegeben:

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

Stellen Sie mehrere Nachrichten in die Warteschlange. Geben Sie dazu Text ein und drücken Sie nach jeder Texteingabe die **Eingabetaste**.

Zum Beenden drücken Sie die **Eingabetaste** zweimal hintereinander.

Rufen Sie die Nachrichten aus der Warteschlange ab.

Führen Sie den folgenden Befehl aus:

```
amqsgetc EXAMPLE.QUEUE SECUREQM
```

Die zuvor hinzugefügten Nachrichten wurden verarbeitet und werden nun ausgegeben.

Der Befehl wird nach wenigen Sekunden automatisch beendet.

Herzlichen Glückwunsch. Sie haben einen Warteschlangenmanager mit aktiviertem TLS bereitgestellt und nachgewiesen, dass Nachrichten von einem Client sicher auf dem Warteschlangenmanager eingereiht und wieder abgerufen werden können.

OpenShift CP4I *Beispiel: Lizenzserviceanmerkungen anpassen*

IBM MQ Operator fügt den implementierten Ressourcen automatisch IBM License Service-Anmerkungen hinzu. Diese werden von IBM License Service überwacht, und es werden Berichte generiert, die der erforderlichen Berechtigung entsprechen.

Informationen zu diesem Vorgang

Bei den von IBM MQ Operator hinzugefügten Anmerkungen handelt es sich um die in Standardsituationen erwarteten Anmerkungen, die auf den Lizenzwerten basieren, die während der Implementierung eines Warteschlangenmanagers ausgewählt wurden.

Beispiel

Wenn **License**auf `L-RJON-BZFQU2 (IBM Cloud Pak for Integration 2021.2.1)` und **Use**auf `Nicht-Produktion` gesetzt ist, werden die folgenden Anmerkungen angewendet:

- cloudpakId: c8b82d189e7545f0892db9ef2731b90d
- cloudpakName: IBM Cloud Pak for Integration

- productChargedContainers: qmgr
- productCloudpakRatio: '4:1'
- productID: 21dfe9a0f00f444f888756d835334909
- productName: IBM MQ Advanced for Non-Production
- productMetric: VIRTUAL_PROCESSOR_CORE
- productVersion: 9.2.3.0

Innerhalb von IBM Cloud Pak for Integration verfügen Implementierungen von IBM App Connect Enterprise eine eingeschränkte Berechtigung für IBM MQ. In diesen Fällen müssen diese Anmerkungen außer Kraft gesetzt werden, um sicherzustellen, dass IBM License Service die korrekte Syntax erfasst. Verwenden Sie dazu den in „Angepasste Anmerkungen und Beschriftungen zu Warteschlangenmanagerressourcen hinzufügen” auf Seite 119 beschriebenen Ansatz.

Wenn IBM MQ beispielsweise im Rahmen einer IBM App Connect Enterprise-Berechtigung implementiert wird, verwenden Sie den in dem folgenden Codefragment gezeigten Ansatz:

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

Es gibt zwei weitere häufige Gründe, warum Lizenzanmerkungen möglicherweise geändert werden müssen:

1. IBM MQ Advanced ist in der Berechtigung eines anderen IBM Produkts enthalten.
   - Verwenden Sie in dieser Situation den zuvor beschriebenen Ansatz für IBM App Connect Enterprise.
2. IBM MQ wird im Rahmen einer IBM Cloud Pak for Integration-Lizenz implementiert.
   - Wenn Sie über eine IBM Cloud Pak for Integration-Lizenz verfügen, können Sie entscheiden, ob Sie einen Warteschlangenmanager unter dem IBM MQ-Verhältnis oder unter dem IBM MQ Advanced-Verhältnis implementieren möchten. Wenn Sie die Implementierung unter einem IBM MQ-Verhältnis durchführen, müssen Sie sicherstellen, dass Sie keine erweiterten Funktionen verwenden, z. B. Native HA oder Advanced Message Security.
   - Verwenden Sie in dieser Situation die folgenden Anmerkungen für die produktive Nutzung:

     ```
     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
     ```

   - Verwenden Sie die folgenden Anmerkungen für die nicht produktive Nutzung:

     ```
     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 Hohe Verfügbarkeit für Warteschlangenmanager mithilfe von IBM MQ Operator konfigurieren

Informationen zu diesem Vorgang

Prozedur

- V 9.2.3
  „Native HA” auf Seite 96.
- V 9.2.3
  „Beispiel: Einen nativen HA-Warteschlangenmanager konfigurieren” auf Seite 98.
- „Beispiel: Multi-Instanz-Warteschlangenmanager konfigurieren” auf Seite 107.

CP4I CD V 9.2.3 *Native HA*

Native HA ist eine native (integrierte) Hochverfügbarkeitslösung für IBM MQ, die für die Verwendung mit Cloud-Blockspeicher geeignet ist.

Eine Native HA-Konfiguration stellt einen hoch verfügbaren Warteschlangenmanager bereit, bei dem wiederherstellbare MQ-Daten (z. B. die Nachrichten) über mehrere Speichergruppen hinweg repliziert werden, um so Verluste aufgrund von Speicherfehlern zu vermeiden. Der Warteschlangenmanager besteht aus mehreren aktiven Instanzen, einer führenden Instanz und den anderen Instanzen, die bereit sind, im Falle eines Ausfalls deren Funktion zu übernehmen, wodurch der Zugriff auf den Warteschlangenmanager und seine Nachrichten maximiert wird.

Eine Native HA-Konfiguration besteht aus drei Kubernetes-Pods, jeder mit einer Instanz des Warteschlangenmanagers. Eine Instanz ist der aktive Warteschlangenmanager, der Nachrichten verarbeitet und Daten in sein Wiederherstellungsprotokoll schreibt. Bei jedem Schreibzugriff auf das Wiederherstellungsprotokoll sendet der aktive Warteschlangenmanager die Daten an die anderen zwei Instanzen, die sogenannten Replikate. Jedes Replikat schreibt die Daten in sein eigenes Wiederherstellungsprotokoll, bestätigt die Daten und aktualisiert anschließend die eigenen Warteschlangendaten aus dem replizierten Wiederherstellungsprotokoll. Wenn der Pod mit dem aktiven Warteschlangenmanager ausfällt, übernimmt eine der Replikatinstanzen des Warteschlangenmanagers die aktive Rolle und verfügt über aktuelle Daten für ihre Arbeit.

Der Protokolltyp wird als 'repliziertes Protokoll' bezeichnet. Ein repliziertes Protokoll ist im Wesentlichen ein lineares Protokoll, bei dem die automatische Protokollverwaltung und automatische Medienimages aktiviert sind. Siehe Typen der Protokollierung. Sie verwenden dieselben Verfahren für die Verwaltung des replizierten Protokolls wie für die Verwaltung eines linearen Protokolls.

Es wird ein Kubernetes-Service verwendet, um TCP/IP-Clientverbindungen an die aktuelle aktive Instanz weiterzuleiten, die als der einzige Pod identifiziert wird, der für den Datenaustausch im Netz bereit ist. Dies geschieht, ohne dass der Clientanwendung die verschiedenen Instanzen bekannt sein müssen.

Es werden drei Pods verwendet, um die Möglichkeit einer Split-Brain-Situation deutlich zu reduzieren. In einem Hochverfügbarkeitssystem mit zwei Pods könnte Split Brain auftreten, wenn die Verbindung zwischen den beiden Pods unterbrochen wird. Besteht keine Verbindung, könnten beide Pods den Warteschlangenmanager gleichzeitig ausführen und dabei unterschiedliche Daten kumulieren. Nach Wiederherstellung der Verbindung gäbe es zwei verschiedene Versionen der Daten (ein 'Split Brain') und es wäre ein manueller Eingriff erforderlich, um zu entscheiden, welche Daten beibehalten und welche gelöscht werden sollen.

Bei Native HA wird ein 3-Pod-System mit Quorum verwendet, um die Split-Brain-Situation zu vermeiden. Pods, die mit mindestens einem der anderen Pods kommunizieren können, bilden ein Quorum. Ein Warteschlangenmanager kann nur auf einem Pod mit Quorum zur aktiven Instanz werden. Der Warteschlangenmanager kann nicht auf einem Pod aktiv werden, der nicht mit mindestens einem anderen Pod verbunden ist, sodass es nie zwei aktive Instanzen zur gleichen Zeit geben kann:

- Wenn ein einzelner Pod ausfällt, kann der Warteschlangenmanager auf einem der beiden anderen Pods dessen Aufgabe übernehmen. Wenn zwei Pods ausfallen, kann der Warteschlangenmanager nicht zur aktiven Instanz auf dem verbleibenden Pod werden, da der Pod kein Quorum hat (der verbleibende Pod kann nicht erkennen, ob die beiden anderen Pods ausgefallen sind oder ob sie noch aktiv sind und er nur die Verbindung verloren hat).
- Wenn ein einzelner Pod die Verbindung verliert, kann der Warteschlangenmanager nicht auf diesem Pod aktiv werden, weil der Pod kein Quorum hat. Der Warteschlangenmanager auf einem der verbleibenden zwei Pods mit Quorum kann übernehmen. Wenn alle Pods die Verbindung verlieren, kann der Warteschlangenmanager auf keinem der Pods aktiv werden, da beide Pods kein Quorum haben.

Wenn ein aktiver Pod ausfällt und anschließend wiederhergestellt wird, kann in einer Replikatrolle wieder in die Gruppe eingebunden werden.

Die folgende Abbildung zeigt eine typische Implementierung mit drei Instanzen eines Warteschlangenmanagers in drei Containern.

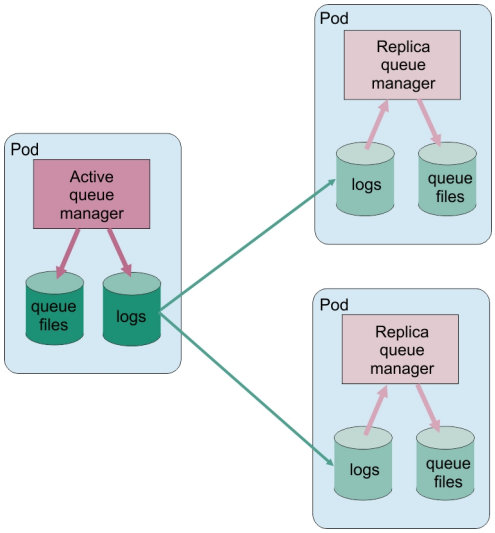


Abbildung 1. Beispiel einer Native HA-Konfiguration

CP4I CD V 9.2.3 *Native HA mit IBM MQ Operator konfigurieren*

Native Hochverfügbarkeit wird über die `QueueManager`-API konfiguriert und erweiterte Optionen sind über eine INI-Datei verfügbar.

Native Hochverfügbarkeit wird mit `.spec.queueManager.availability`der `QueueManager`-API konfiguriert, z. B.:

```
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
```

Das Feld `.spec.queueManager.availability.type`muss auf `NativeHA`gesetzt sein.

Native HA ist in IBM MQ 9.2.3 oder höher verfügbar.

Unter `.spec.queueManager.availability`können Sie auch einen geheimen TLS-Schlüssel und Verschlüsselungen für die Replikation zwischen Warteschlangenmanagerinstanzen konfigurieren. Dies wird

dringend empfohlen. Im Abschnitt „Beispiel: Einen nativen HA-Warteschlangenmanager konfigurieren” auf Seite 98 ist eine Schritt-für-Schritt-Anleitung verfügbar.

Zugehörige Verweise

„Beispiel: Einen nativen HA-Warteschlangenmanager konfigurieren” auf Seite 98
Dieses Beispiel zeigt, wie Sie einen WS-Manager mit der nativen Hochverfügbarkeitsfunktion in Red Hat OpenShift Container Platform (OCP) unter Verwendung von IBM MQ Operatorimplementieren.

Beispiel: Einen nativen HA-Warteschlangenmanager konfigurieren
Dieses Beispiel zeigt, wie Sie einen WS-Manager mit der nativen Hochverfügbarkeitsfunktion in Red Hat OpenShift Container Platform (OCP) unter Verwendung von IBM MQ Operatorimplementieren.

Bevor Sie beginnen

Die folgenden Schritte müssen als Voraussetzung für dieses Beispiel durchgeführt worden sein:

- Installieren Sie den IBM MQ client und fügen Sie die installierten Verzeichnisse `samp/bin` und `bin` Ihrem *PATH* hinzu. Der Client stellt die für dieses Beispiel erforderlichen Anwendungen **`runmqakm`**, **`amqsputc`** und **`amqsgetc`** bereit. Installieren Sie den IBM MQ client wie folgt:
  - 

Für Windows und Linux: Installieren Sie den weiterverteilbaren Client von IBM MQ für Ihr Betriebssystem von https://ibm.biz/mq92redistclients
  - 

Für Mac: Laden Sie das IBM MQ MacOS Toolkit herunter und richten Sie es ein. Siehe https://ibm.biz/mqdevmacclient.
- Installieren Sie das OpenSSL-Tool für Ihr Betriebssystem. Sie benötigen es, um ein selbst signiertes Zertifikat für den Warteschlangenmanager zu generieren, sofern Sie nicht bereits über einen privaten Schlüssel und ein Zertifikat verfügen.
- Erstellen Sie für dieses Beispiel ein Projekt bzw. einen Namensbereich für Red Hat OpenShift Container Platform (OCP) und führen Sie die Schritte in der Task „Red Hat OpenShift-Projekt für IBM MQ vorbereiten” auf Seite 82 aus.
- Melden Sie sich über die Befehlszeile bei dem OCP-Cluster an und wechseln Sie zu dem gerade erstelltn Namensbereich.
- Stellen Sie sicher, dass IBM MQ Operator installiert und im Namensbereich verfügbar ist.
- Konfigurieren Sie eine Standardspeicherklasse in der OCP, die von Ihrem Warteschlangenmanager verwendet werden soll. Wenn Sie dieses Lernprogramm ausführen möchten, ohne eine Standardspeicherklasse festzulegen, lesen Sie die Informationen in Hinweis 2: Nicht standardmäßige Speicherklasse verwenden.

Informationen zu dieser Task

Für Warteschlangenmanager mit Native HA werden ein aktiver und zwei Replikat-Kubernetes-Pods benötigt. Diese werden als Teil eines Kubernetes Stateful Set mit exakt drei Replikaten und einem Satz von Kubernetes Persistent Volumes ausgeführt. Weitere Informationen zu Warteschlangenmanagern mit Native HA finden Sie im Abschnitt „Hochverfügbarkeit für IBM MQ in Containern” auf Seite 16.

Das Beispiel stellt eine angepasste Ressource YAML zur Verfügung, die einen nativen HA-Warteschlangenmanager definiert, der den persistenten Speicher verwendet und mit TLS konfiguriert ist. Nachdem Sie den Warteschlangenmanager in der OCP implementiert haben, simulieren Sie den Ausfall des aktiven Warteschlangenmanager-Pods. Sie können dann eine automatische Wiederherstellung beobachten und deren Erfolg nachweisen, indem Sie nach dem Ausfall Nachrichten einreihen und abrufen.

Beispiel

Privaten TLS-Schlüssel und Zertifikate für MQ-Server erstellen

Sie können ein selbst signiertes Zertifikat für den Warteschlangenmanager erstellen und das Zertifikat zu einer Schlüsseldatenbank hinzufügen, die als Truststore für den Client fungiert. Falls Sie bereits über einen privaten Schlüssel und ein Zertifikat verfügen, können Sie diese stattdessen verwenden. Zu Entwicklungszwecken sollten nur selbst signierte Zertifikate verwendet werden.

Führen Sie folgenden Befehl aus, um im aktuellen Verzeichnis einen selbst signierten privaten Schlüssel und ein öffentliches Zertifikat zu erstellen:

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

Privaten TLS-Schlüssel und Zertifikate für interne Verwendung durch Native HA erstellen

Die drei Pods in einem Warteschlangenmanager mit Native HA replizieren Daten über das Netz. Sie können ein selbst signiertes Zertifikat für die interne Replizierung erstellen. Zu Entwicklungszwecken sollten nur selbst signierte Zertifikate verwendet werden.

Führen Sie folgenden Befehl aus, um im aktuellen Verzeichnis einen selbst signierten privaten Schlüssel und ein öffentliches Zertifikat zu erstellen:

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

Öffentlichen Schlüssel des Warteschlangenmanagers zu einer Clientschlüsseldatenbank hinzufügen

Eine Clientschlüsseldatenbank dient der Clientanwendung als Truststore.

Erstellen Sie die Clientschlüsseldatenbank:

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

Fügen Sie den zuvor generierten öffentlichen Schlüssel zur Clientschlüsseldatenbank hinzu:

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

Secret mit TLS-Zertifikaten für Warteschlangenmanagerimplementierung erstellen

Erstellen Sie ein Kubernetes-TLS-Secret, das auf die zuvor erstellten Dateien verweist, damit Ihr Warteschlangenmanager den Schlüssel und das Zertifikat referenzieren und anwenden kann. Achten Sie bei der Konfiguration darauf, dass Sie den zuvor erstellten Namensbereich verwenden.

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

Secret mit dem internen TLS-Zertifikat und -Schlüssel für Native HA erstellen

Erstellen Sie ein Kubernetes-TLS-Secret, das auf die zuvor erstellten Dateien verweist, damit Ihr Warteschlangenmanager den Schlüssel und das Zertifikat referenzieren und anwenden kann. Achten Sie bei der Konfiguration darauf, dass Sie den zuvor erstellten Namensbereich verwenden.

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

Erstellen Sie eine Konfigurationszuordnung (ConfigMap) mit MQSC-Befehlen.

Erstellen Sie eine Kubernetes-Konfigurationszuordnung (ConfigMap) mit MQSC-Befehlen zum Erstellen einer neuen Warteschlange und eines SVRCONN-Kanals sowie zum Hinzufügen eines Kanalauthentifizierungsdatensatzes, der Zugriff auf den Kanal ermöglicht, wobei nur Benutzer mit dem Namen *nobody* blockiert werden.

Beachten Sie, dass diese Vorgehensweise ausschließlich für Entwicklungszwecke geeignet ist.

Stellen Sie sicher, dass Sie sich im zuvor erstellten Namensbereich befinden (siehe „Bevor Sie beginnen” auf Seite 98), und geben Sie dann in der OCP-Benutzerschnittstelle oder über die Befehlszeile folgende YAML ein:

```
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('A
NY_TLS12_OR_HIGHER')
    SET CHLAUTH(HAQMCHL) TYPE(BLOCKUSER) USERLIST('nobody') ACTION(ADD)
```

Routing konfigurieren

Wenn Sie einen IBM MQ client oder ein Toolkit von IBM MQ 9.2.1 oder höher verwenden, können Sie das Routing zum Warteschlangenmanager mithilfe einer Warteschlangenmanager-Konfigurationsdatei (eine INI-Datei) konfigurieren. In der Datei setzen Sie die Variable *OutboundSNI* für das Routing auf den Hostnamen statt auf den Kanalnamen.

Erstellen Sie in dem Verzeichnis, in dem Sie Befehle ausführen, eine Datei mit dem Namen `mqclient.ini`, die exakt den folgenden Text enthält:

```
SSL:
   OutboundSNI=HOSTNAME
```

Ändern Sie keine Werte in dieser INI-Datei. Die Zeichenfolge HOSTNAMEdarf beispielsweise nicht geändert werden.

Weitere Details finden Sie im Abschnitt SSL-Zeilengruppe der Clientkonfigurationsdatei.

Wenn Sie einen IBM MQ client oder ein Toolkit früher als IBM MQ 9.2.1 verwenden, müssen Sie statt der vorherigen Konfigurationsdatei eine OCP-Route erstellen. Führen Sie die Schritte in Hinweis 1: Route erstellen aus.

Stellen Sie den Warteschlangenmanager bereit.

Wichtig: In diesem Beispiel inaktivieren wir die Autorisierung auf dem Warteschlangenmanager mit der Variablen *MQSNOAUT*. Dadurch können wir uns ganz auf die Schritte konzentrieren, die zur Verbindung eines Clients mit TLS erforderlich sind. In einer IBM MQ-Produktionsumgebung wird hiervon abgeraten. Jede Anwendung, die eine Verbindung herstellt, hätte bei dieser Einstellung vollständige Verwaltungsbefugnisse, ohne jeglichen Mechanismus, die Berechtigungen für einzelne Anwendungen herabzusetzen.

Kopieren und aktualisieren Sie die folgende YAML.

- Stellen Sie sicher, dass die richtige Lizenz angegeben wird. Siehe Lizenzierungsreferenz für mq.ibm.com/v1beta1. In IBM Cloud Pak for Integration2021.1.1 muss die Probelizenz `L-RJON-BYRMYW`
- Akzeptieren Sie die Lizenz, indem Sie `false`in `true`ändern.

Angepasste Ressourcen-YAML des Warteschlangenmanagers:

```
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
```

Stellen Sie sicher, dass Sie sich im zuvor erstellten Namensbereich befinden und implementieren Sie die aktualisierte YAML über die Red Hat OpenShift Container Platform-Webkonsole, die Befehlszeile oder mit dem IBM Cloud Pak for Integration Platform Navigator.

Es gibt eine kurze Verzögerung, während das System den nativen HA-Warteschlangenmanager konfiguriert, nach dem der Warteschlangenmanager für die Verwendung verfügbar sein sollte.

Überprüfung

In diesem Abschnitt wird überprüft, ob sich der Warteschlangenmanager wie erwartet verhält.

Vergewissern Sie sich, dass der Warteschlangenmanager aktiv ist.

Der Warteschlangenmanager ist nun bereitgestellt. Vergewissern Sie sich, dass er sich im Status `Running`befindet, bevor Sie fortfahren. Beispiel:

```
oc get qmgr nativeha-example
```

Testen Sie die Verbindung zum Warteschlangenmanager.

Für einen Test der Konfiguration des Warteschlangenmanagers mit unidirektionaler TLS-Kommunikation können Sie die Beispielanwendungen **`amqsputc`** und **`amqsgetc`** verwenden:

Suchen Sie den Hostnamen des Warteschlangenmanagers.

Führen Sie den folgenden Befehl aus, um den Hostnamen des Warteschlangenmanagers für die Route `nativeha-example-ibm-mq-qm`zu suchen. Der Hostname wird im Feld `HOST`zurückgegeben.

```
oc get routes nativeha-example-ibm-mq-qm
```

Geben Sie die Details des Warteschlangenmanagers an.

Erstellen Sie eine Datei `CCDT.JSON`, in der die Details des Warteschlangenmanagers angegeben sind. Ersetzen Sie den Hostwert durch den Hostnamen, der im vorherigen Schritt zurückgegeben wurde.

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

Exportieren Sie Umgebungsvariablen.

Exportieren Sie die folgenden Umgebungsvariablen mit der für Ihr Betriebssystem geeigneten Methode. Diese Variablen werden von **amqsputc** und **amqsgetc** gelesen.

Aktualisieren Sie den Pfad zu den Dateien auf Ihrem System:

```
export MQCCDTURL='<full_path_to_file>/CCDT.JSON'
export MQSSLKEYR='<full_path_to_file>/clientkey'
```

Reihen Sie Nachrichten in die Warteschlange ein.

Führen Sie den folgenden Befehl aus:

```
amqsputc EXAMPLE.QUEUE HAEXAMPLE
```

Bei einer erfolgreichen Verbindung mit dem Warteschlangenmanager wird die folgende Antwort ausgegeben:

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

Stellen Sie mehrere Nachrichten in die Warteschlange. Geben Sie dazu Text ein und drücken Sie nach jeder Texteingabe die **Eingabetaste**.

Zum Beenden drücken Sie die **Eingabetaste** zweimal hintereinander.

Rufen Sie die Nachrichten aus der Warteschlange ab.

Führen Sie den folgenden Befehl aus:

```
amqsgetc EXAMPLE.QUEUE HAEXAMPLE
```

Die zuvor hinzugefügten Nachrichten wurden verarbeitet und werden nun ausgegeben.

Der Befehl wird nach wenigen Sekunden automatisch beendet.

Erzwingen Sie einen Ausfall des aktiven Pods.

Simulieren Sie einen Pod-Ausfall, um die automatische Wiederherstellung des Warteschlangenmanagers zu überprüfen:

Zeigen Sie den aktiven und den Standby-Pod an.

Führen Sie den folgenden Befehl aus:

```
oc get pods --selector app.kubernetes.io/instance=nativeha-example
```

Beachten Sie, dass der aktive Pod im Feld **READY**den Wert 1/1zurückgibt, während die Replikatpods den Wert 0/1zurückgeben.

Löschen Sie den aktiven Pod.

Führen Sie folgenden Befehl aus und geben Sie dabei den vollständigen Namen des aktiven Pods an:

```
oc delete pod nativeha-example-ibm-mq-<value>
```

Zeigen Sie den Pod-Status erneut an.

Führen Sie den folgenden Befehl aus:

```
oc get pods --selector app.kubernetes.io/instance=nativeha-example
```

Zeigen Sie den Status des Warteschlangenmanagers an.

Führen Sie folgenden Befehl aus und geben Sie dabei den vollständigen Namen eines der anderen Pods an:

```
oc exec -t Pod -- dspmq -o nativeha -x -m HAEXAMPLE
```

Der Status sollte anzeigen, dass sich die aktive Instanz geändert hat, z. B.:

```
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)
```

Führen Sie ein erneutes Einreihen und Abrufen von Nachrichten durch.

Nachdem der Standby-Pod zum aktiven Pod geworden ist (d. h., nachdem der Feldwert READYauf 1/1gesetzt wurde), verwenden Sie die folgenden Befehle erneut, wie zuvor beschrieben, um Nachrichten in den Warteschlangenmanager einzureihen und anschließend Nachrichten vom Warteschlangenmanager abzurufen:

```
amqsputc EXAMPLE.QUEUE HAEXAMPLE
```

```
amqsgetc EXAMPLE.QUEUE HAEXAMPLE
```

Herzlichen Glückwunsch, Sie haben erfolgreich einen Warteschlangenmanager mit Native HA implementiert und gezeigt, dass er nach einem Pod-Ausfall automatisch wiederhergestellt werden kann.

Weitere Informationen

Hinweis 1: Route erstellen

Wenn Sie einen IBM MQ client oder ein Toolkit früher als IBM MQ 9.2.1 verwenden, müssen Sie eine Route erstellen.

Um die Route zu erstellen, müssen Sie sicherstellen, dass Sie sich im zuvor erstellten Namensbereich befinden (siehe „Bevor Sie beginnen" auf Seite 98), und dann in der Red Hat OpenShift Container Platform-Webkonsole oder über die Befehlszeile folgende YAML eingeben:

```
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
```

Beachten Sie, dass Red Hat OpenShift Container Platform Router die SNI für die Weiterleitung von Anforderungen an den IBM MQ-Warteschlangenmanager verwendet. Wenn Sie den Kanalnamen ändern, der in der Konfigurationszuordnung mit MQSC-Befehlen angegeben ist, müssen Sie auch das Hostfeld hier und in der Datei `CCDT.JSON`, in der die Details des Warteschlangenmanagers angegeben sind, ändern. Weitere Informationen finden Sie unter „Route für Verbindung zu einem Warteschlangenmanager von außerhalb eines Red Hat OpenShift-Clusters konfigurieren" auf Seite 113.

Hinweis 2: Nicht standardmäßige Speicherklasse verwenden

In diesem Beispiel wird erwartet, dass eine Standardspeicherklasse in Red Hat OpenShift Container Platform konfiguriert wurde, weshalb in der angepassten Ressourcen-YAML des Warteschlangenmanagers keine Speicherinformationen erforderlich sind. Wenn Sie keine Speicherklasse als Standard konfiguriert haben oder eine andere Speicherklasse verwenden möchten, fügen Sie `defaultClass: <storage_class_name>`unter `spec.queueManager.storage`hinzu.

Der Name der Speicherklasse muss genau mit dem Namen einer Speicherklasse übereinstimmen, die bereits vorhanden ist. Dies bedeutet, dass er mit dem Namen übereinstimmen muss, der vom Befehl `oc get storageclass`zurückgegeben wurde. Es muss auch `ReadWriteMany`unterstützen. Weitere Informationen finden Sie unter „Speicheraspekte für IBM MQ Operator" auf Seite 11.

Zugehörige Tasks

„Status von nativen HA-Warteschlangenmanagern für zertifizierte IBM MQ -Container anzeigen" auf Seite 104

Für zertifizierte IBM MQ -Container können Sie den Status der nativen HA-Instanzen anzeigen, indem Sie den Befehl **dspmq** in einem der aktiven Pods ausführen.

CP4I CD V 9.2.2 *Status von nativen HA-Warteschlangenmanagern für zertifizierte IBM MQ -Container anzeigen*

Für zertifizierte IBM MQ -Container können Sie den Status der nativen HA-Instanzen anzeigen, indem Sie den Befehl **dspmq** in einem der aktiven Pods ausführen.

Informationen zu diesem Vorgang

Wichtig:

Durch Ausführung des Befehls **dspmq** in einem der aktiven Pods können Sie den Betriebsstatus einer Warteschlangenmanagerinstanz anzeigen. Welche Informationen zurückgegeben werden, hängt davon ab, ob die Instanz aktiv oder ein Replikat ist. Die von der aktiven Instanz gelieferten Informationen sind verbindlich, während Informationen von Replikatknoten möglicherweise nicht auf dem neuesten Stand sind.

Sie können folgende Aktionen ausführen:

- Anzeigen, ob die Warteschlangenmanagerinstanz auf dem aktuellen Knoten aktiv oder ein Replikat ist.
- Anzeigen des Native HA-Betriebsstatus der Instanz auf dem aktuellen Knoten.
- Anzeigen des Betriebsstatus aller drei Instanzen in einer Native HA-Konfiguration.

Der Status einer Native HA-Konfiguration wird in folgenden Statusfeldern gemeldet:

ROLE
Gibt die aktuelle Rolle der Instanz an. Die gültigen Werte sind `Active`, `Replica`und `Unknown`.

INSTANCE
Der Name, der für diese Instanz des Warteschlangenmanagers angegeben wurde, als sie mit der Option **-lr** des Befehls **crtmqm** erstellt wurde.

INSYNC
Gibt an, ob die Instanz bei Bedarf die Rolle der aktiven Instanz übernehmen kann.

QUORUM
Gibt den Quorumstatus im Format *Anzahl_synchrone_Instanzen*/*Anzahl_konfigurierte_Instanzen* an.

REPLADDR
Gibt die Replikationsadresse der Warteschlangenmanagerinstanz an.

CONNACTV
Gibt an, ob der Knoten mit der aktiven Instanz verbunden ist.

BACKLOG
Gibt die Anzahl KB an, um die die Instanz zurückliegt.

CONNINST
Gibt an, ob die benannte Instanz mit dieser Instanz verbunden ist.

ALTDATE
Gibt das Datum der letzten Aktualisierung dieser Informationen an (leer, wenn sie noch nie aktualisiert wurden).

ALTTIME
Gibt die Zeit der letzten Aktualisierung dieser Informationen an (leer, wenn sie noch nie aktualisiert wurden).

Prozedur

- Suchen Sie die Pods, die zu Ihrem Warteschlangenmanager gehören.

```
oc get pod --selector app.kubernetes.io/instance=nativeha-qm
```

- Führen Sie `dspmq`in einem der Pods aus

```
oc exec -t Pod dspmq
```

```
oc rsh Pod
```

für eine interaktive Shell, in der Sie `dspmq`direkt ausführen können.

- Stellen Sie fest, ob eine Warteschlangenmanagerinstanz als aktive Instanz oder als Replikat ausgeführt wird:

```
oc exec -t Pod dspmq -o status -m QMgrName
```

Eine aktive Instanz eines Warteschlangenmanagers mit dem Namen BOB würde folgenden Status melden:

```
QMNAME(BOB)                STATUS(Running)
```

Eine Replikatinstanz eines Warteschlangenmanagers mit dem Namen BOB würde folgenden Status melden:

```
QMNAME(BOB)                STATUS(Replica)
```

Eine inaktive Instanz würde folgenden Status melden:

```
QMNAME(BOB)                STATUS(Ended Immediately)
```

- Stellen Sie den Native HA-Betriebsstatus der Instanz im angegebenen Pod fest:

```
oc exec -t Pod dspmq -o nativeha -m QMgrName
```

Die aktive Instanz eines Warteschlangenmanagers mit dem Namen BOB könnte folgenden Status melden:

```
QMNAME(BOB)              ROLE(Active) INSTANCE(inst1) INSYNC(Yes) QUORUM(3/3)
```

Eine Replikatinstanz eines Warteschlangenmanagers mit dem Namen BOB könnte folgenden Status melden:

```
QMNAME(BOB)              ROLE(Replica) INSTANCE(inst2) INSYNC(Yes) QUORUM(2/3)
```

Eine inaktive Instanz eines Warteschlangenmanagers mit dem Namen BOB könnte folgenden Status melden:

```
QMNAME(BOB)              ROLE(Unknown) INSTANCE(inst3) INSYNC(no) QUORUM(0/3)
```

- Stellen Sie den Native HA-Betriebsstatus aller Instanzen in der Native HA-Konfiguration fest:

```
oc exec -t Pod dspmq -o nativeha -x -m QMgrName
```

Wenn Sie diesen Befehl auf dem Knoten mit der aktiven Instanz des Warteschlangenmanagers BOB ausgeben, könnte der folgende Status gemeldet werden:

```
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)
```

Wenn Sie diesen Befehl auf einem Knoten mit einer Replikatinstanz des Warteschlangenmanagers BOB ausgeben, könnte der folgende Status gemeldet werden, der anzeigt, dass eins der Replikate im Rückstand ist:

```
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)
```

Wenn Sie diesen Befehl auf einem Knoten mit einer inaktiven Instanz des Warteschlangenmanagers BOB ausgeben, könnte der folgende Status gemeldet werden:

```
QMNAME(BOB)            ROLE(Unknown) INSTANCE(inst3) INSYNC(no) QUORUM(0/3)
 INSTANCE(inst1) ROLE(Unknown) REPLADDR(9.20.123.45) CONNACTV(Unknown) INSYNC(Unknown) BACK
LOG(Unknown) CONNINST(No) ALTDATE() ALTTIME()
 INSTANCE(inst2) ROLE(Unknown) REPLADDR(9.20.123.46) CONNACTV(Unknown) INSYNC(Unknown) BACK
LOG(Unknown) CONNINST(No) ALTDATE() ALTTIME()
 INSTANCE(inst3) ROLE(Unknown) REPLADDR(9.20.123.47) CONNACTV(No) INSYNC(Unknown) BACKLOG(Un
known) CONNINST(No) ALTDATE() ALTTIME()
```

Wenn Sie den Befehl ausgeben, während die Instanzen aushandeln, welche die aktive Instanz und und welche die Replikate sind, würden Sie folgenden Status empfangen:

```
QMNAME(BOB)             STATUS(Negotiating)
```

Zugehörige Verweise

Befehl dspmq (Warteschlangenmanager anzeigen)

„Beispiel: Einen nativen HA-Warteschlangenmanager konfigurieren” auf Seite 98

Dieses Beispiel zeigt, wie Sie einen WS-Manager mit der nativen Hochverfügbarkeitsfunktion in Red Hat OpenShift Container Platform (OCP) unter Verwendung von IBM MQ Operatorimplementieren.

Erweiterte Optimierung für native HA

Erweiterte Einstellungen zur Optimierung von Abläufen und Intervallen. Diese Einstellungen sollten nur verwendet werden, wenn sich herausstellt, dass die Standardeinstellungen den Anforderungen des Systems nicht genügen.

Die Basisoptionen für die Konfiguration der nativen Hochverfügbarkeit werden mithilfe der `QueueManager`-API behandelt, mit der IBM MQ Operatordie zugrunde liegenden INI-Dateien des Warteschlangenmanagers für Sie konfiguriert. Es gibt einige erweiterte Optionen, die nur mithilfe einer INI-Datei unter der NativeHALocal-Instanzzeilengruppekonfiguriert werden können. Weitere Informationen zur Konfiguration einer INI-Datei finden Sie unter „Beispiel: Unterstützung von MQSC- und INI-Dateien” auf Seite 89 .

`HeartbeatInterval`
: Das Heartbeatintervall legt fest, wie oft in Millisekunden eine aktive Instanz eines Warteschlangenmanagers mit Native HA ein Netzüberwachungssignal sendet. Der gültige Bereich für den Heartbeatintervallwert liegt zwischen 500 (0,5 Sekunden) und 60000 (1 Minute). Ein Wert außerhalb dieses Bereichs führt dazu, dass der Warteschlangenmanager nicht gestartet wird. Wird dieses Attribut nicht angegeben, wird der Standardwert 5000 (5 Sekunden) verwendet. Es muss für alle Instanzen dasselbe Heartbeatintervall festgelegt werden.

`HeartbeatTimeout`
: Das Überwachungssignalzeitlimit legt fest, wie lange eine Replikatinstanz eines Warteschlangenmanagers mit Native HA wartet, bevor sie entscheidet, dass die aktive Instanz nicht mehr reagiert. Der gültige Bereich für den Wert dieses Zeitlimits liegt zwischen 500 (0,5 Sekunden) und 120000 (2 Minuten). Der Wert des Überwachungssignalzeitlimits muss größer-gleich dem Wert des Heartbeatintervalls sein.

  Ein ungültiger Wert führt dazu, dass der Warteschlangenmanager nicht gestartet wird. Wird dieses Attribut nicht angegeben, wartet ein Replikat 2 x `HeartbeatInterval`, bevor es den Prozess startet, um eine neue aktive Instanz zu wählen. Es muss für alle Instanzen dasselbe Überwachungssignalzeitlimit festgelegt werden.

`RetryInterval`
: Das Wiederholungsintervall legt fest, wie oft in Millisekunden ein Warteschlangenmanager mit Native HA eine fehlgeschlagene Replikationsverbindung wiederholen soll. Der gültige Bereich für das Wiederholungsintervall liegt zwischen 500 (0,5 Sekunden) und 120000 (2 Minuten). Wenn dieses Attribut weggelassen wird, wartet ein Replikat 2 x `HeartbeatInterval`, bevor es eine fehlgeschlagene Replikationsverbindung wiederholt.

CP4I *Beenden von nativen HA-Warteschlangenmanagern*

Mit dem Befehl **endmqm** können Sie einen aktiven Warteschlangenmanager oder einen Replikatwarteschlangenmanager beenden, der Teil einer nativen HA-Gruppe ist.

Prozedur

- Informationen zum Beenden der aktiven Instanz eines Warteschlangenmanagers finden Sie unter Native HA-Warteschlangenmanager beenden im Konfigurationsabschnitt dieser Dokumentation.

CP4I CD V 9.2.2 *Funktion 'Native HA' in IBM Cloud Pak for Integration 2021.1.1 evaluieren*

Die Auswertungsperiode für native HA IBM Cloud Pak for Integration 2021.1.1 ist beendet. Verwenden Sie die aktualisierte native HA-Funktion, die ab IBM Cloud Pak for Integration 2021.2.1 verfügbar ist, mit IBM MQ Operator 1.6 oder höher mit IBM MQ 9.2.3 oder höher.

Zugehörige Tasks

„Status von nativen HA-Warteschlangenmanagern für zertifizierte IBM MQ -Container anzeigen" auf Seite 104

Für zertifizierte IBM MQ -Container können Sie den Status der nativen HA-Instanzen anzeigen, indem Sie den Befehl **dspmq** in einem der aktiven Pods ausführen.

Zugehörige Verweise

„Beispiel: Einen nativen HA-Warteschlangenmanager konfigurieren" auf Seite 98

Dieses Beispiel zeigt, wie Sie einen WS-Manager mit der nativen Hochverfügbarkeitsfunktion in Red Hat OpenShift Container Platform (OCP) unter Verwendung von IBM MQ Operatorimplementieren.

OpenShift CP4I

Beispiel: Multi-Instanz-Warteschlangenmanager konfigurieren

Dieses Beispiel zeigt, wie Sie einen Multi-Instanz-Warteschlangenmanager mithilfe von IBM MQ Operatorin Red Hat OpenShift Container Platform (OCP) implementieren. Außerdem konfigurieren Sie in diesem Beispiel eine unidirektionale TLS-Kommunikation zwischen einem Beispielclient und dem Warteschlangenmanager. Im Beispiel wird die erfolgreiche Konfiguration durch das Einreihen und Abrufen von Nachrichten vor und nach einem simulierten Pod-Ausfall nachgewiesen.

Bevor Sie beginnen

Die folgenden Schritte müssen als Voraussetzung für dieses Beispiel durchgeführt worden sein:

- Installieren Sie den IBM MQ client und fügen Sie die installierten Verzeichnisse `samp/bin` und `bin` Ihrem *PATH* hinzu. Der Client stellt die für dieses Beispiel erforderlichen Anwendungen **runmqakm**, **amqsputc** und **amqsgetc** bereit. Installieren Sie den IBM MQ client wie folgt:
  - Windows Linux Für Windows und Linux: Installieren Sie den weiterverteilbaren Client von IBM MQ für Ihr Betriebssystem von https://ibm.biz/mq92redistclients
  - macOS Für Mac: Laden Sie das IBM MQ MacOS Toolkit herunter und richten Sie es ein. Siehe https://developer.ibm.com/tutorials/mq-macos-dev/.
- Installieren Sie das OpenSSL-Tool für Ihr Betriebssystem. Sie benötigen es, um ein selbst signiertes Zertifikat für den Warteschlangenmanager zu generieren, sofern Sie nicht bereits über einen privaten Schlüssel und ein Zertifikat verfügen.
- Erstellen Sie für dieses Beispiel ein Projekt bzw. einen Namensbereich für Red Hat OpenShift Container Platform (OCP).
- Melden Sie sich über die Befehlszeile bei dem OCP-Cluster an und wechseln Sie zum oben genannten Namensbereich.
- Stellen Sie sicher, dass IBM MQ Operator installiert und in dem oben genannten Namensbereich verfügbar ist.
- Konfigurieren Sie eine Standardspeicherklasse in der OCP, die von Ihrem Warteschlangenmanager verwendet werden soll. Wenn Sie dieses Lernprogramm ausführen möchten, ohne eine Standardspei-

cherklasse festzulegen, lesen Sie die Informationen in Hinweis 2: Nicht standardmäßige Speicherklasse verwenden.

Informationen zu dieser Task

Für Multi-Instanz-Warteschlangenmanager werden ein aktiver und ein Standby-Kubernetes-Pod benötigt. Diese werden als Teil eines Kubernetes Stateful Set mit exakt zwei Replikaten und einem Satz von Kubernetes Persistent Volumes ausgeführt. Weitere Informationen zu Multi-Instanz-Warteschlangenmanagern finden Sie im Abschnitt „Hochverfügbarkeit für IBM MQ in Containern" auf Seite 16.

Das Beispiel stellt eine angepasste Ressourcen-YAML bereit, die einen Multi-Instanz-Warteschlangenmanager mit persistentem Speicher definiert und mit TLS konfiguriert ist. Nachdem Sie den Warteschlangenmanager in der OCP implementiert haben, simulieren Sie den Ausfall des aktiven Warteschlangenmanager-Pods. Sie können dann eine automatische Wiederherstellung beobachten und deren Erfolg nachweisen, indem Sie nach dem Ausfall Nachrichten einreihen und abrufen.

Beispiel

Privaten TLS-Schlüssel und Zertifikate für MQ-Server erstellen

In diesem Abschnitt wird dokumentiert, wie ein selbst signiertes Zertifikat für den Warteschlangenmanager erstellt und das Zertifikat zu einer Schlüsseldatenbank hinzugefügt wird, die als Truststore für den Client fungiert. Falls Sie bereits über einen privaten Schlüssel und ein Zertifikat verfügen, können Sie diese stattdessen verwenden. Zu Entwicklungszwecken sollten nur selbst signierte Zertifikate verwendet werden.

Führen Sie folgenden Befehl aus, um im aktuellen Verzeichnis einen selbst signierten privaten Schlüssel und ein öffentliches Zertifikat zu erstellen:

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

Öffentlichen Schlüssel des Warteschlangenmanagers zu einer Clientschlüsseldatenbank hinzufügen

Eine Clientschlüsseldatenbank dient der Clientanwendung als Truststore.

Erstellen Sie die Clientschlüsseldatenbank:

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

Fügen Sie den zuvor generierten öffentlichen Schlüssel zur Clientschlüsseldatenbank hinzu:

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

Secret mit TLS-Zertifikaten für Warteschlangenmanagerimplementierung erstellen

Erstellen Sie ein Kubernetes-TLS-Secret, das auf die zuvor erstellten Dateien verweist, damit Ihr Warteschlangenmanager den Schlüssel und das Zertifikat referenzieren und anwenden kann. Achten Sie bei der Konfiguration darauf, dass Sie den zuvor erstellten Namensbereich verwenden.

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

Erstellen Sie eine Konfigurationszuordnung (ConfigMap) mit MQSC-Befehlen.

Erstellen Sie eine Kubernetes-Konfigurationszuordnung (ConfigMap) mit MQSC-Befehlen zum Erstellen einer neuen Warteschlange und eines SVRCONN-Kanals sowie zum Hinzufügen eines Kanalauthentifizierungsdatensatzes, der Zugriff auf den Kanal ermöglicht, wobei nur Benutzer mit dem Namen *nobody* blockiert werden.

Beachten Sie, dass diese Vorgehensweise ausschließlich für Entwicklungszwecke geeignet ist.

Stellen Sie sicher, dass Sie sich im zuvor erstellten Namensbereich befinden (siehe „Bevor Sie beginnen” auf Seite 107), und geben Sie dann in der OCP-Benutzerschnittstelle oder über die Befehlszeile folgende YAML ein:

```
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('A□
NY_TLS12_OR_HIGHER')
    SET CHLAUTH(MIQMCHL) TYPE(BLOCKUSER) USERLIST('nobody') ACTION(ADD)
```

Routing konfigurieren

Wenn Sie einen IBM MQ client oder ein Toolkit von IBM MQ 9.2.1 oder höher verwenden, können Sie das Routing zum Warteschlangenmanager mithilfe einer Warteschlangenmanager-Konfigurationsdatei (eine INI-Datei) konfigurieren. In der Datei setzen Sie die Variable *OutboundSNI* für das Routing auf den Hostnamen statt auf den Kanalnamen.

Erstellen Sie in dem Verzeichnis, in dem Sie Befehle ausführen, eine Datei mit dem Namen mqclient.ini, die folgenden Text enthält:

```
#* 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
```

Hinweis: Ändern Sie keine Werte auf dieser Seite. Die Zeichenfolge HOSTNAMEsollte beispielsweise unverändert bleiben.

Weitere Details finden Sie im Abschnitt SSL-Zeilengruppe der Clientkonfigurationsdatei.

Wenn Sie einen IBM MQ client oder ein Toolkit früher als IBM MQ 9.2.1 verwenden, müssen Sie statt der vorherigen Konfigurationsdatei eine OCP-Route erstellen. Führen Sie die Schritte in Hinweis 1: Route erstellen aus.

Stellen Sie den Warteschlangenmanager bereit.

Wichtig: In diesem Beispiel inaktivieren wir die Autorisierung auf dem Warteschlangenmanager mit der Variablen *MQSNOAUT*. Dadurch können wir uns ganz auf die Schritte konzentrieren, die zur Verbindung eines Clients mit TLS erforderlich sind. In einer IBM MQ-Produktionsumgebung wird hiervon abgeraten. Jede Anwendung, die eine Verbindung herstellt, hätte bei dieser Einstellung vollständige Verwaltungsbefugnisse, ohne jeglichen Mechanismus, die Berechtigungen für einzelne Anwendungen herabzusetzen.

Kopieren und aktualisieren Sie die folgende YAML.

- Stellen Sie sicher, dass die richtige Lizenz angegeben wird. Siehe Lizenzierungsreferenz für mq.ibm.com/v1beta1.
- Akzeptieren Sie die Lizenz, indem Sie `false`in `true`ändern.
- Wenn Sie IBM Cloud File Storageverwenden, lesen Sie den Abschnitt Anmerkung 3: Nutzen von IBM Cloud File Storage

Angepasste Ressourcen-YAML des Warteschlangenmanagers:

```
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
```

Stellen Sie sicher, dass Sie sich im zuvor erstellten Namensbereich befinden und implementieren Sie die aktualisierte YAML in der OCP-Benutzerschnittstelle. Verwenden Sie dazu die Befehlszeile oder IBM Cloud Pak for Integration Platform Navigator.

Überprüfung

Nach einer kurzen Verzögerung sollte der Multi-Instanz-Warteschlangenmanager konfiguriert und zur Verwendung verfügbar sein. In diesem Abschnitt wird überprüft, ob sich der Warteschlangenmanager wie erwartet verhält.

Vergewissern Sie sich, dass der Warteschlangenmanager aktiv ist.

Der Warteschlangenmanager ist nun bereitgestellt. Vergewissern Sie sich, dass er sich im Status `Running`befindet, bevor Sie fortfahren. Beispiel:

```
oc get qmgr miexample
```

Testen Sie die Verbindung zum Warteschlangenmanager.

Für einen Test der Konfiguration des Warteschlangenmanagers mit unidirektionaler TLS-Kommunikation können Sie die Beispielanwendungen **`amqsputc`** und **`amqsgetc`** verwenden:

Suchen Sie den Hostnamen des Warteschlangenmanagers.

Führen Sie den folgenden Befehl aus, um den Hostnamen des Warteschlangenmanagers für die Route `miexample-ibm-mq-qm`zu suchen. Der Hostname wird im Feld `HOST`zurückgegeben.

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

Geben Sie die Details des Warteschlangenmanagers an.

Erstellen Sie eine Datei `CCDT.JSON`, in der die Details des Warteschlangenmanagers angegeben sind. Ersetzen Sie den Hostwert durch den Hostnamen, der im vorherigen Schritt zurückgegeben wurde.

```
{
    "channel":
    [
        {
            "name": "MIQMCHL",
```

```
            "clientConnection":
            {
                "connection":
                [
                    {
                        "host": "<host from previous step>",
                        "port": 443
                    }
                ],
                "queueManager": "MIEXAMPLE"
            },
            "transmissionSecurity":
            {
              "cipherSpecification": "ECDHE_RSA_AES_128_CBC_SHA256"
            },
            "type": "clientConnection"
        }
    ]
}
```

Exportieren Sie Umgebungsvariablen.

Exportieren Sie die folgenden Umgebungsvariablen mit der für Ihr Betriebssystem geeigneten Methode. Diese Variablen werden von **amqsputc** und **amqsgetc** gelesen.

Aktualisieren Sie den Pfad zu den Dateien auf Ihrem System:

```
export MQCCDTURL='<full_path_to_file>/CCDT.JSON'
export MQSSLKEYR='<full_path_to_file>/clientkey'
```

Reihen Sie Nachrichten in die Warteschlange ein.

Führen Sie den folgenden Befehl aus:

```
amqsputc EXAMPLE.QUEUE MIEXAMPLE
```

Bei einer erfolgreichen Verbindung mit dem Warteschlangenmanager wird die folgende Antwort ausgegeben:

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

Stellen Sie mehrere Nachrichten in die Warteschlange. Geben Sie dazu Text ein und drücken Sie nach jeder Texteingabe die **Eingabetaste**.

Zum Beenden drücken Sie die **Eingabetaste** zweimal hintereinander.

Rufen Sie die Nachrichten aus der Warteschlange ab.

Führen Sie den folgenden Befehl aus:

```
amqsgetc EXAMPLE.QUEUE MIEXAMPLE
```

Die zuvor hinzugefügten Nachrichten wurden verarbeitet und werden nun ausgegeben.

Der Befehl wird nach wenigen Sekunden automatisch beendet.

Erzwingen Sie einen Ausfall des aktiven Pods.

Simulieren Sie einen Pod-Ausfall, um die automatische Wiederherstellung des Multi-Instanz-Warteschlangenmanagers zu überprüfen:

Zeigen Sie den aktiven und den Standby-Pod an.

Führen Sie den folgenden Befehl aus:

```
oc get pods
```

Beachten Sie, dass der aktive Pod im Feld **READY**den Wert 1/1zurückgibt, während der Standby-Pod den Wert 0/1zurückgibt.

Löschen Sie den aktiven Pod.

Führen Sie folgenden Befehl aus und geben Sie dabei den vollständigen Namen des aktiven Pods an:

```
oc delete pod miexample-ibm-mq-<value>
```

Zeigen Sie den Pod-Status erneut an.
Führen Sie den folgenden Befehl aus:

```
oc get pods
```

Zeigen Sie das Protokoll des Standby-Pods an.
Führen Sie folgenden Befehl aus und geben Sie dabei den vollständigen Namen des Standby-Pods an:

```
oc logs miexample-ibm-mq-<value>
```

Es sollte folgende Nachricht angezeigt werden:

```
IBM MQ queue manager 'MIEXAMPLE' becoming the active instance.
```

Führen Sie ein erneutes Einreihen und Abrufen von Nachrichten durch.
Nachdem der Standby-Pod zum aktiven Pod geworden ist (d. h., nachdem der Feldwert READYauf 1/1gesetzt wurde), verwenden Sie die folgenden Befehle erneut, wie zuvor beschrieben, um Nachrichten in den Warteschlangenmanager einzureihen und anschließend Nachrichten vom Warteschlangenmanager abzurufen:

```
amqsputc EXAMPLE.QUEUE MIEXAMPLE
```

```
amqsgetc EXAMPLE.QUEUE MIEXAMPLE
```

Herzlichen Glückwunsch, Sie haben erfolgreich einen Multi-Instanz-Warteschlangenmanager implementiert und gezeigt, dass er nach einem Pod-Ausfall automatisch wiederhergestellt werden kann.

Weitere Informationen

Hinweis 1: Route erstellen
Wenn Sie einen IBM MQ client oder ein Toolkit früher als IBM MQ 9.2.1 verwenden, müssen Sie eine OCP-Route erstellen.

Um die Route zu erstellen, müssen Sie sicherstellen, dass Sie sich im zuvor erstellten Namensbereich befinden (siehe „Bevor Sie beginnen" auf Seite 107), und dann in der OCP-Benutzerschnittstelle oder über die Befehlszeile folgende YAML eingeben:

```
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
```

Beachten Sie, dass Red Hat OpenShift Container Platform Router die SNI für die Weiterleitung von Anforderungen an den IBM MQ-Warteschlangenmanager verwendet. Wenn Sie den Kanalnamen ändern, der in der Konfigurationszuordnung mit MQSC-Befehlen angegeben ist, müssen Sie auch das Hostfeld hier und in der Datei CCDT.JSON, in der die Details des Warteschlangenmanagers angegeben sind, ändern. Weitere Informationen finden Sie unter „Route für Verbindung zu einem Warteschlangenmanager von außerhalb eines Red Hat OpenShift-Clusters konfigurieren" auf Seite 113.

Hinweis 2: Nicht standardmäßige Speicherklasse verwenden
In diesem Beispiel wird erwartet, dass eine Standardspeicherklasse in OCP konfiguriert wurde, weshalb in der angepassten Ressourcen-YAML des Warteschlangenmanagers keine Speicherinformationen erforderlich sind. Wenn Sie keine Speicherklasse als Standard konfiguriert haben oder eine ande-

re Speicherklasse verwenden möchten, fügen Sie `defaultClass: <storage_class_name>` unter `spec.queueManager.storage` hinzu.

Der Name der Speicherklasse muss genau mit dem Namen einer Speicherklasse übereinstimmen, die auf Ihrem OCP-System vorhanden ist. Dies bedeutet, dass er mit dem Namen übereinstimmen muss, der vom Befehl `oc get storageclass` zurückgegeben wurde. Es muss auch `ReadWriteMany` unterstützen. Weitere Informationen finden Sie unter „Speicheraspekte für IBM MQ Operator" auf Seite 11.

Hinweis 3: IBM Cloud File Storage verwenden

In einigen Situationen, z. B. bei Verwendung von IBM Cloud File Storage, müssen Sie auch das Feld **`securityGroups`** im Angepasste YAML-Ressourcendatei für Warteschlangenmanager angeben. Fügen Sie beispielsweise das folgende untergeordnete Feld direkt unter `spec:` hinzu

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

Weitere Informationen finden Sie unter „Speicheraspekte für IBM MQ Operator" auf Seite 11.

OpenShift CP4I CD Route für Verbindung zu einem Warteschlangenmanager von außerhalb eines Red Hat OpenShift-Clusters konfigurieren

Sie benötigen eine Red Hat OpenShift -Route, um eine Anwendung von außerhalb eines Red Hat OpenShift -Clusters mit einem IBM MQ -Warteschlangenmanager zu verbinden. Sie müssen TLS auf Ihrem IBM MQ -Warteschlangenmanager und Ihrer Clientanwendung aktivieren, da SNI nur im TLS-Protokoll verfügbar ist, wenn ein TLS 1.2 oder ein höheres Protokoll verwendet wird. Der Red Hat OpenShift Container Platform Router verwendet SNI für Routing-Anforderungen an den IBM MQ-Warteschlangenmanager.

Informationen zu diesem Vorgang

Achtung: Dieses Dokument gilt für IBM MQ -Clients der Version 9.2.1 Continuous Delivery und höher. Entsprechende Informationen für Clients mit Long Term Support der Version 9.2.0 oder früher finden Sie auf der IBM MQ 9.1-Dokumentationsseite Verbindung zu einem Warteschlangenmanager herstellen, der in einem Red Hat OpenShift-Cluster implementiert ist.

V 9.2.1 Die erforderliche Konfiguration der Red Hat OpenShift -Route hängt vom Verhalten von Server Name Indication (SNI) Ihrer Clientanwendung ab. IBM MQ unterstützt je nach Konfiguration und Clienttyp zwei verschiedene SNI-Headereinstellungen. Ein SNI-Header wird auf den Hostnamen des Ziels des Clients oder alternativ auf den IBM MQ -Kanalnamen gesetzt. Informationen zur Zuordnung eines Kanalnamens zu einem Hostnamen in IBM MQ finden Sie im Abschnitt Wie IBM MQ die Funktionalität mit mehreren Zertifikaten bereitstellt.

V 9.2.1 Ob ein SNI-Header auf einen IBM MQ -Kanalnamen oder einen Hostnamen gesetzt ist, wird über das Attribut **`OutboundSNI`** gesteuert. Mögliche Werte sind `OutboundSNI=CHANNEL` (Standardwert) oder `OutboundSNI=HOSTNAME`. Weitere Informationen finden Sie unter SSL-Zeilengruppe der Clientkonfigurationsdatei. Beachten Sie, dass `CHANNEL` und `HOSTNAME` genau die Werte sind, die Sie verwenden. Es handelt sich nicht um Variablennamen, die Sie durch einen tatsächlichen Kanal-oder Hostnamen ersetzen.

V 9.2.1

Clientverhalten mit unterschiedlichen `OutboundSNI` -Einstellungen

Wenn **`OutboundSNI`** auf `HOSTNAME` gesetzt ist, legen die folgenden Clients für die SNI einen Hostnamen fest, sofern im Verbindungsnamen ein Hostname angegeben ist:

- C-Clients
- .NET-Clients im nicht verwalteten Modus
- Java/JMS-Clients

Wenn **OutboundSNI** auf HOSTNAME gesetzt ist und der Verbindungsname eine IP-Adresse enthält, senden die folgenden Clients einen leeren SNI-Header:

- C-Clients
- .NET-Clients im nicht verwalteten Modus
- Java/JMS-Clients (die kein Reverse-DNS-Lookup des Hostnamens durchführen können)

Wenn **OutboundSNI** auf CHANNEL gesetzt ist (oder hier keine Festlegung besteht), wird stattdessen ein IBM MQ-Kanalname verwendet, der unabhängig davon, ob die Verbindung einen Hostnamen oder eine IP-Adresse angibt, immer gesendet wird.

Die folgenden Clienttypen unterstützen für den SNI-Header keinen IBM MQ-Kanalnamen. Sie versuchen daher unabhängig von der Einstellung von **OutboundSNI**, den SNI-Header immer auf einen Hostnamen zu setzen:

- AMQP-Clients
- XR-Clients
- .NET-Clients im verwalteten Modus (Vor IBM MQ 9.2.0 Fix Pack 4 für Long Term Support und vor IBM MQ 9.2.3 für Continuous Delivery.)

V 9.2.0.4 V 9.2.3

Ab IBM MQ 9.2.0 Fix Pack 4 für Long Term Support und IBM MQ 9.2.3 für Continuous Deliverywurde der IBM MQ verwaltete .NET -Client aktualisiert und SERVERNAME auf den entsprechenden Hostnamen gesetzt, wenn die Eigenschaft **OutboundSNI** auf HOSTNAMEgesetzt ist. Dies ermöglicht einem IBM MQ verwalteten .NET -Client, über Red Hat OpenShift -Routen eine Verbindung zu einem Warteschlangenmanager herzustellen. Beachten Sie, dass die Eigenschaft **OutboundSNI** in IBM MQ 9.2.0 Fix Pack 4nur aus der Datei `mqclient.ini` hinzugefügt und unterstützt wird. Sie können die Eigenschaft nicht über die .NET-Anwendung definieren.

V 9.2.5

Wenn eine Clientanwendung eine Verbindung zu einem Warteschlangenmanager herstellt, der in einem Red Hat OpenShift-Cluster über IBM MQ Internet Pass-Thru (MQIPT) implementiert ist, kann MQIPT durch Verwendung der Eigenschaft SSLClientOutboundSNI in der Routendefinition so konfiguriert werden, dass die SNI auf den Hostnamen gesetzt wird.

OutboundSNI, mehrere Zertifikate und Red Hat OpenShift -Routen

IBM MQ verwendet den SNI-Header, um mehrere Zertifikatsfunktionen bereitzustellen. Wenn eine Anwendung eine Verbindung zu einem IBM MQ -Kanal herstellt, der für die Verwendung eines anderen Zertifikats über das CERTLABL-Feld konfiguriert ist, muss die Anwendung eine Verbindung mit der **OutboundSNI** -Einstellung CHANNEL herstellen.

Wenn Ihre Red Hat OpenShift -Routenkonfiguration einen Hostnamen SNI erfordert, können Sie die Funktionalität für mehrere Zertifikate von IBM MQ nicht verwenden und keine CERTLABL-Einstellung für ein IBM MQ -Kanalobjekt festlegen.

Wenn eine Anwendung mit einer anderen **OutboundSNI** -Einstellung als CHANNEL eine Verbindung zu einem Kanal mit einer konfigurierten Zertifikatsbezeichnung herstellt, wird die Anwendung mit dem Fehler MQRC_SSL_INITIALIZATION_ERROR zurückgewiesen und eine Nachricht AMQ9673 in den Fehlerprotokollen des Warteschlangenmanagers ausgegeben.

Weitere Informationen dazu, wie IBM MQ mehrere Zertifikatsfunktionen bereitstellt, finden Sie unter Funktionalität für mehrere Zertifikate in IBM MQ .

Beispiel

Für Clientanwendungen, die die SNI auf den MQ-Kanal setzen, muss für jeden Kanal, zu dem eine Verbindung hergestellt werden soll, eine neue Red Hat OpenShift-Route erstellt werden. Sie müssen auch eindeutige Kanalnamen in Ihrem Red Hat OpenShift Container Platform-Cluster verwenden, um die Weiterleitung an den richtigen Warteschlangenmanager zu ermöglichen.

Wegen der Art und Weise, wie IBM MQ Kanalnamen SNI-Headern zuordnet, darf das letzte Zeichen von MQ-Kanalnamen auf keinen Fall ein Kleinbuchstabe sein.

Um den erforderlichen Hostnamen für jede Ihrer neuen Red Hat OpenShift-Routen zu ermitteln, müssen Sie jeden Kanalnamen einer SNI-Adresse zuordnen. Informationen hierzu finden Sie im Abschnitt Wie IBM MQ die Funktionalität mit mehreren Zertifikaten bereitstellt.

Anschließend müssen Sie eine neue Red Hat OpenShift -Route für jeden Kanal erstellen, indem Sie die folgende `yaml` in Ihrem Cluster anwenden:

```
  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
```

Konfiguration der Verbindungsdetails der Clientanwendung

Sie können den Hostnamen ermitteln, der für Ihre Clientverbindung verwendet werden soll, indem Sie den folgenden Befehl ausführen:

```
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}"
```

Der Port für Ihre Clientverbindung sollte auf den Port gesetzt werden, der vom Red Hat OpenShift Container Platform-Router verwendet wird – normalerweise 443.

Zugehörige Tasks

„Verbindung zur IBM MQ Console herstellen, die in einem Red Hat OpenShift-Cluster implementiert ist" auf Seite 120
Sie können eine Verbindung zur IBM MQ Console eines Warteschlangenmanagers herstellen, die in einem Red Hat OpenShift Container Platform-Cluster implementiert wurde.

CP4I Integration in das IBM Cloud Pak for Integration-Dashboard 'Operations'

Die Fähigkeit, Transaktionen über IBM Cloud Pak for Integration zu verfolgen, wird durch das Operations-Dashboard bereitgestellt.

Informationen zu diesem Vorgang

Bei der Aktivierung der Integration mit dem Dashboard 'Operations' wird ein MQ-API-Exit für Ihren Warteschlangenmanager installiert. Der API-Exit sendet Tracedaten zu Nachrichten, die über den Warteschlangenmanager übertragen werden, an den Datenspeicher des Dashboards 'Operations'.

Beachten Sie, dass nur für Nachrichten, die mit MQ-Clientbindungen gesendet werden, ein Trace erstellt wird.

Beachten Sie außerdem, dass es sich in Versionen von IBM MQ Operator vor 1.5, in denen die Tracefunktion aktiviert ist, bei den Images für den Tracing-Agent und -Collector, die mit dem Warteschlangenmanager bereitgestellt wurden, immer um die neuesten verfügbaren Versionen handelte, was zu einer Inkompatibilität führen kann, wenn Sie nicht die neueste Version von IBM Cloud Pak for Integration verwenden.

Vorgehensweise

1. Implementieren Sie einen Warteschlangenmanager mit aktiviertem Tracing.

   Das Tracing-Feature ist standardmäßig inaktiviert.

   Wenn Sie die Implementierung mithilfe von IBM Cloud Pak for Integration Platform Navigator durchführen, können Sie das Tracing während der Implementierung aktivieren, indem Sie **Enable Tracing** (Tracing aktivieren) auf **On** setzen und für **Tracing Namespace** (Tracing-Namensbereich) den Namensbereich angeben, in dem das Dashboard 'Operations' installiert ist. Weitere Informationen zur Implementierung eines Warteschlangenmanagers finden Sie im Abschnitt „Warteschlangenmanager mit dem IBM Cloud Pak for Integration Platform Navigator implementieren” auf Seite 85 .

   Wenn Sie die Implementierung über die Red Hat OpenShift-Befehlszeilenschnittstelle (CLI) oder über die Red Hat OpenShift-Webkonsole durchführen, können Sie das Tracing mit dem folgenden YAML-Snippet aktivieren:

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

   Wichtig: Der Warteschlangenmanager startet erst, wenn MQ im Dashboard 'Operations' registriert wurde (siehe nächster Schritt).

   Wenn diese Funktion aktiviert ist, müssen Sie beachten, dass zusätzlich zum Container des Warteschlangenmanagers zwei Sidecar-Container ('Agent' und 'Collector') ausgeführt werden. Die Images für diese Sidecar-Container sind in der gleichen Registry wie das MQ-Hauptimage verfügbar und verwenden die gleichen Richtlinien und geheimen Schlüssel für Pull-Operationen. Es sind weitere Einstellungen zur Konfiguration der CPU und der Speicherbegrenzung verfügbar.
2. Wenn ein Warteschlangenmanager mit der Integration für das Dashboard 'Operations' das erste Mal in diesem Namensbereich implementiert wird, müssen Sie eine Registrierung im Dashboard 'Operations' durchführen.

   Beim Registrieren wird ein Objekt für einen geheimen Schlüssel erstellt, den der Warteschlangenmanager für einen erfolgreichen Start benötigt.

CP4I CD

Implementierung oder Upgrade für IBM MQ 9.2.2 oder 9.2.3 mit Operations Dashboard-Integration in IBM Cloud Pak for Integration 2021.4 durchführen

Jede IBM MQ-Version ist einer bestimmten Version des Operations Dashboard-Agenten und der Kollektorkomponenten zugeordnet, die zusammen mit einem Warteschlangenmanager implementiert werden. IBM Cloud Pak for Integration 2021.4.1 führt eine Änderung ein, die dazu führt, dass ältere Agenten- und Kollektorkomponenten nicht mit dem Operations Dashboard arbeiten. Um dieses Problem zu lösen, müssen Sie die Version der verwendeten Images des Operations Dashboard-Agent und -Collector überschreiben, wenn Sie IBM MQ 9.2.2 oder 9.2.3 verwenden.

Neuen Warteschlangenmanager für IBM MQ 9.2.2 oder 9.2.3 implementieren

Wenn Sie IBM Cloud Pak for Integration2021.4.1 mit IBM MQ 9.2.2oder 9.2.3 verwenden, müssen Sie die Images des Operations Dashboard-Agenten und des Kollektors mit den 2.4-Versionen in Ihrer `QueueManager`-YAML überschreiben. Beispiel:

```
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-collec
tor@sha256:dc70b1341b23dc72642ce68809811f9db0e8a0c46bda2508e8eb3d4035e04f4b
```

Wenn Sie dies nicht tun, verbleibt Ihr QueueManager-Pod im Status Pending. Wenn Sie ein Upgrade auf IBM MQ 9.2.4 durchführen, können Sie diese Überschreibungen entfernen.

Upgrade aufIBM Cloud Pak for Integration 2021.4.1 durchführen

Anmerkung: Wenn Sie Ihren Warteschlangenmanager für IBM MQ 9.2.2oder 9.2.3 beibehalten, führen Sie Schritt 3 nicht aus.

1. Aktualisieren Sie Ihre QueueManager, um die Agenten- und Collector-Images wie zuvor beschrieben zu überschreiben.
2. Führen Sie ein Upgrade Ihrer IBM Cloud Pak for Integration-Operatoren durch, einschließlich des Operations Dashboard und des IBM MQ-Operators (siehe die Beschreibung in „Upgrade für IBM MQ Operator und Warteschlangenmanager durchführen” auf Seite 74).
3. (Optional) Wenn Sie ein Upgrade auf IBM MQ 9.2.4oder höher durchführen möchten, aktualisieren SieQueueManager so, dass .spec.versionfür Ihre Version von IBM MQverwendet wird, und entfernen Sie anschließend die Überschreibung der Agenten- und Collector-Images.

OpenShift CP4I Image mit benutzerdefinierten MQSC- und INI-Dateien über die Red Hat OpenShift-CLI erstellen

Erstellen Sie mithilfe einer Red Hat OpenShift Container Platform-Pipeline ein neues IBM MQ-Container-Image mit MQSC-und INI-Dateien, die mit diesem Image auf Warteschlangenmanager angewendet werden sollen. Diese Aufgabe sollte von einem Projektadministrator ausgeführt werden

Vorbereitende Schritte

Sie müssen die Red Hat OpenShift Container Platform-Befehlszeilenschnittstelle (CLI) installieren.

Melden Sie sich mit **cloudctl login** (für IBM Cloud Pak for Integration) oder **oc login** bei Ihrem Cluster an.

Wenn Sie kein Red Hat OpenShift-Secret (geheimer Schlüssel) für die IBM Entitled Registry in Ihrem Red Hat OpenShift-Projekt haben, führen Sie die Schritte im Abschnitt „Red Hat OpenShift-Projekt für IBM MQ vorbereiten” auf Seite 82 aus.

Vorgehensweise

1. Erstellen:ImageStream

   Ein ImageStream und die ihm zugeordneten Tags stellen eine Abstraktion für die Referenzierung von Container-Images aus Red Hat OpenShift Container Platform heraus bereit. Mithilfe des ImageStream und der zugehörigen Tags können Sie sehen, welche Images verfügbar sind, und sicherstellen, dass Sie genau das Image verwenden, das Sie benötigen, selbst wenn sich das Image im Repository ändert.

   ```
   oc create imagestream mymq
   ```

2. BuildConfig für Ihr neues Image erstellen

   Ein BuildConfigermöglicht Builds für Ihr neues Image, das auf den offiziellen IBM-Images basiert, fügt jedoch alle MQSC- oder INI-Dateien hinzu, die beim Containerstart ausgeführt werden sollen.

   a) Erstellen Sie eine YAML-Datei, die die BuildConfig-Ressource definiert

      Erstellen Sie beispielsweise eine Datei mit dem Namen 'mq-build-config.yaml' mit folgendem Inhalt:

      ```
      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'
```

Sie müssen die zwei Stellen ändern, an denen das IBM MQ-Basisimage genannt wird, sodass auf das richtige Basisimage für die Version und den Fix verwiesen wird, die Sie verwenden möchten (siehe „Releaseprotokoll für IBM MQ Operator" auf Seite 21). Wenn Fixes angewendet werden, müssen Sie diese Schritte wiederholen, um Ihr Image erneut zu erstellen.

In diesem Beispiel wird ein neues Image auf Basis des offiziellen IBM Image erstellt und es werden Dateien mit den Namen "my.mqsc" und "my.ini" im Verzeichnis `/etc/mqm` hinzugefügt. Alle MQSC- oder INI-Dateien, die in diesem Verzeichnis gefunden werden, werden beim Start vom Container ausgeführt. INI-Dateien werden mit der Option **`crtmqm -ii`** ausgeführt und mit den vorhandenen INI-Dateien zusammengeführt. MQSC-Dateien werden in alphabetischer Reihenfolge ausgeführt.

Es ist wichtig, dass Ihre MQSC-Befehle wiederholt anwendbar sind, da sie bei *jedem* Start des Warteschlangenmanagers ausgeführt werden. Dies bedeutet in der Regel, dass der Parameter `REPLACE`in allen `DEFINE`-Befehlen und der Parameter `IGNSTATE(YES)` in allen `START`- oder `STOP`-Befehlen hinzugefügt wird.

b) Wenden Sie `BuildConfig`auf den Server an.

```
oc apply -f mq-build-config.yaml
```

3. Führen Sie einen Build aus, um Ihr Image zu erstellen.

a) Starten Sie den Build.

```
oc start-build mymq
```

Es sollte eine Ausgabe wie die folgende angezeigt werden:

```
build.build.openshift.io/mymq-1 started
```

b) Überprüfen Sie den Status des Builds.
Sie können beispielsweise folgenden Befehl unter Angabe der im vorherigen Schritt zurückgegebenen Build-ID ausführen:

```
oc describe build mymq-1
```

4. Implementieren Sie einen Warteschlangenmanager mit dem neuen Image.

Führen Sie die Schritte im Abschnitt „Warteschlangenmanager in einem Red Hat OpenShift Container Platform-Cluster bereitstellen" auf Seite 84 aus und fügen Sie dabei das neue benutzerdefinierte Image in der YAML-Datei hinzu.

Sie können das folgende YAML-Snippet in Ihrer normalen `QueueManager`-YAML-Datei hinzufügen, wobei *mein_namensbereich* das von Ihnen verwendete Red Hat OpenShift-Projekt/den von Ihnen verwendeten Namensbereich und *Bild* der Name des Image ist, das Sie zuvor erstellt haben (z. B. "mymq:latest-amd64"):

```
spec:
  queueManager:
    image: image-registry.openshift-image-registry.svc:5000/my-namespace/my-image
```

Zugehörige Tasks

„Warteschlangenmanager in einem Red Hat OpenShift Container Platform-Cluster bereitstellen” auf Seite 84

Verwenden Sie die angepasste QueueManager-Ressource, um einen Warteschlangenmanager in einem Red Hat OpenShift Container Platform-Cluster bereitzustellen.

OpenShift CP4I Angepasste Anmerkungen und Beschriftungen zu Warteschlangenmanagerressourcen hinzufügen

Angepasste Anmerkungen und Beschriftungen werden den QueueManager-Metadaten hinzugefügt.

Informationen zu diesem Vorgang

Angepasste Anmerkungen und Beschriftungen werden zu allen Ressourcen mit Ausnahme von PVCs hinzugefügt. Wenn eine angepasste Anmerkung oder eine angepasste Beschriftung mit einem vorhandenen Schlüssel übereinstimmt, wird der von IBM MQ Operator festgelegte Wert verwendet.

Prozedur

- Fügen Sie angepasste Anmerkungen hinzu.

  Angepasste Anmerkungen zu Warteschlangenmanagerressourcen, einschließlich des Pods, fügen Sie unter `metadata` hinzu. Beispiel:

  ```
  apiVersion: mq.ibm.com/v1beta1
  kind: QueueManager
  metadata:
    name: quickstart-cp4i
    annotations:
      annotationKey: "value"
  ```

- Fügen Sie angepasste Beschriftungen hinzu.

  Angepasste Beschriftungen zu Warteschlangenmanagerressourcen, einschließlich des Pods, fügen Sie unter `metadata` hinzu. Beispiel:

  ```
  apiVersion: mq.ibm.com/v1beta1
  kind: QueueManager
  metadata:
    name: quickstart-cp4i
    labels:
      labelKey: "value"
  ```

OpenShift CP4I Laufzeit-Webhook-Prüfungen inaktivieren

Laufzeit-Webhook-Prüfungen stellen die Funktionsfähigkeit der Speicherklassen für Ihren Warteschlangenmanager sicher. Sie können sie jedoch inaktivieren, um die Leistung zu verbessern, oder weil sie ungeeignet für Ihre Umgebung sind.

Informationen zu diesem Vorgang

Mit Laufzeit-Webhook-Prüfungen wird die Warteschlangenmanagerkonfiguration geprüft. Sie überprüfen, ob die Speicherklassen für den von Ihnen gewählten Warteschlangenmanagertyp geeignet sind.

Sie können diese Prüfungen inaktivieren, um die Warteschlangenmanagererstellung zu beschleunigen, oder weil die Prüfungen ungeeignet für Ihre spezifische Umgebung sind.

Anmerkung: Nach der Inaktivierung der Laufzeit-Webhook-Prüfungen sind alle Speicherklassenwerte zulässig. Die Funktionsfähigkeit des Warteschlangenmanagers ist dann nicht sichergestellt.

Die Unterstützung für Laufzeitprüfungen wurde in IBM MQ Operator 1.2 eingeführt.

Prozedur

- Inaktivieren Sie Laufzeit-Webhook-Prüfungen.

  Fügen Sie unter `metadata` folgende Anmerkung hinzu. Beispiel:

```
apiVersion: mq.ibm.com/v1beta1
kind: QueueManager
metadata:
  name: quickstart-cp4i
  annotations:
    "com.ibm.cp4i/disable-webhook-runtime-checks" : "true"
```

OpenShift CP4I Betreiben von IBM MQ mittels IBM MQ Operator

Informationen zu diesem Vorgang

Prozedur

- „Red Hat OpenShift-Projekt für IBM MQ vorbereiten" auf Seite 82.
- „Warteschlangenmanager in einem Red Hat OpenShift Container Platform-Cluster bereitstellen" auf Seite 84.

OpenShift CP4I Verbindung zur IBM MQ Console herstellen, die in einem Red Hat OpenShift-Cluster implementiert ist

Sie können eine Verbindung zur IBM MQ Console eines Warteschlangenmanagers herstellen, die in einem Red Hat OpenShift Container Platform-Cluster implementiert wurde.

Informationen zu diesem Vorgang

Die URL der IBM MQ Console finden Sie auf der `QueueManager`-Detailseite in der Red Hat OpenShift-Webkonsole oder in IBM Cloud Pak for Integration Platform Navigator. Alternativ können Sie sie mit folgendem Befehl über die Befehlszeilenschnittstelle (CLI) von Red Hat OpenShift abrufen:

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

Wenn Sie eine IBM Cloud Pak for Integration-Lizenz verwenden, dann ist die IBM MQ-Webkonsole für die Verwendung von IBM Cloud Pak Identity and Access Manager (IAM) konfiguriert. Die IAM-Komponente wurde möglicherweise bereits vom Clusteradministrator eingerichtet. Wenn dies jedoch das erste Mal ist, dass IAM in Ihrem Red Hat OpenShift-Cluster verwendet wurde, müssen Sie das Anfangskennwort des Administrators abrufen. Weitere Informationen finden Sie im Abschnitt Ursprüngliches Administratorkennwort abrufen.

Bei Verwendung einer IBM MQ-Lizenz ist die MQ-Webkonsole nicht vorkonfiguriert und Sie müssen sie selbst konfigurieren. Weitere Informationen finden Sie im Abschnitt Benutzer und Rollen konfigurieren.

Zugehörige Tasks

„Route für Verbindung zu einem Warteschlangenmanager von außerhalb eines Red Hat OpenShift-Clusters konfigurieren" auf Seite 113
Sie benötigen eine Red Hat OpenShift -Route, um eine Anwendung von außerhalb eines Red Hat OpenShift -Clusters mit einem IBM MQ -Warteschlangenmanager zu verbinden. Sie müssen TLS auf Ihrem IBM MQ -Warteschlangenmanager und Ihrer Clientanwendung aktivieren, da SNI nur im TLS-Protokoll verfügbar ist, wenn ein TLS 1.2 oder ein höheres Protokoll verwendet wird. Der Red Hat OpenShift Container Platform Router verwendet SNI für Routing-Anforderungen an den IBM MQ-Warteschlangenmanager.

OpenShift CP4I *Berechtigungen für IBM MQ Console mithilfe von IBM Cloud Pak IAM erteilen*

Berechtigungen für IBM MQ Console werden über den IBM Cloud Pak Administration Hub und nicht über IBM Cloud Pak for Integration Platform Navigatorverwaltet. IBM MQ verwendet nicht die von IBM Cloud Pak for Integrationbereitgestellten Automatisierungsberechtigungen, sondern stattdessen die Basisberechtigungen, die von IBM Cloud Pak Identity and Access Manager (IAM) aktiviert werden.

Vorgehensweise

1. Öffnen Sie die Administrationskonsole von IBM Cloud Pak.

   Klicken Sie in der IBM Cloud Pak for Integration Platform UIauf den Umschalter Cloud Pak (9-Punkt-Symbol) in der rechten oberen Ecke der Symbolleiste und klicken Sie dann auf die Anzeige **IBM Cloud Pak Administration** .
2. Wählen Sie im Navigationsmenü in der linken oberen Ecke **Identität und Zugriff**und anschließend **Teams und Service-IDs**aus.
3. Erstellen Sie ein Team und fügen Sie anschließend Benutzer hinzu.
   a) Wählen Sie **Team erstellen**aus.
   b) Geben Sie einen Teamnamen ein und wählen Sie dann die Sicherheitsdomäne für die Benutzer aus, die Sie verwalten wollen.
   c) Suchen Sie Benutzer.

      Diese Benutzer müssen bereits in Ihrem Identitätsprovider vorhanden sein.
   d) Wenn Sie jeden Benutzer finden, weisen Sie ihm eine Rolle zu. Dies muss "Administrator" oder "Clusteradministrator" sein, um IBM MQ über die IBM MQ Consoleverwalten zu können.
4. Fügen Sie jeden Benutzer einem Namensbereich hinzu.
   a) Wählen Sie das Team aus, um es zu bearbeiten.
   b) Wählen Sie **Ressourcen** > **Ressourcen verwalten**aus.
   c) Wählen Sie die Namensbereiche aus, die dieses Team verwalten soll. Dies können alle Namensbereiche mit einem Warteschlangenmanager sein.

OpenShift CP4I Überwachung bei Verwendung von IBM MQ Operator

Warteschlangenmanager, die von IBM MQ Operator verwaltet werden, können Messwerte erstellen, die mit Prometheus kompatibel sind.

Sie können diese Metriken mithilfe des Überwachungsstacks vonRed Hat OpenShift Container Platform (OCP)anzeigen. Öffnen Sie die Registerkarte **Metriken** in OCPund klicken Sie auf **Beobachten** > **Metriken**. Die Warteschlangenmanager-Metriken sind standardmäßig aktiviert, können aber inaktiviert werden, indem **`.spec.metrics.enabled`** auf `false`gesetzt wird.

Prometheus ist eine Zeitreihendatenbank und eine Regelauswertungsfunktion für Metriken. Die IBM MQ-Container legen einen Metrikendpunkt offen, der von Prometheus abgefragt werden kann. Die Metriken werden von den MQ-Systemthemen zur Überwachung und Aktivitätsverfolgung generiert.

Red Hat OpenShift Container Platform enthält einen bereits konfigurierten, vorinstallierten und selbst aktualisierten Überwachungsstack, der einen Prometheus-Server verwendet. Der Monitoring-Stack der Red Hat OpenShift Container Platform muss für die Überwachung benutzerdefinierter Projekte konfiguriert werden. Weitere Informationen finden Sie unter Enabling monitoring for user-defined projects. IBM MQ Operator erstellt eine `ServiceMonitor`, wenn Sie eine `QueueManager`mit aktivierten Metriken erstellen, die dann vom Prometheus-Bediener erkannt werden können.

In älteren Versionen von IBM Cloud Pak for Integration könnten Sie stattdessen auch den Service IBM Cloud Platform Monitoring verwenden, um einen Prometheus-Server bereitzustellen.

OpenShift CP4I *Metriken, die bei Verwendung von IBM MQ Operator veröffentlicht werden*

Warteschlangenmanagercontainer können Metriken veröffentlichen, die mit Red Hat OpenShift Monitoring kompatibel sind.

Metrik	Typ	Beschreibung
ibmmq_qmgr_commit_total	counter	Festschreibungszähler
ibmmq_qmgr_cpu_load_fifteen_minute_average_percentage	gauge	CPU-Belastung - 15-Minuten-Durchschnitt
ibmmq_qmgr_cpu_load_five_minute_average_percentage	gauge	CPU-Belastung - 5-Minuten-Durchschnitt
ibmmq_qmgr_cpu_load_one_minute_average_percentage	gauge	CPU-Belastung - 1-Minute-Durchschnitt
ibmmq_qmgr_destructive_get_bytes_total	counter	Gesamtabrufe mit Löschen in Intervall - Byteanzahl
ibmmq_qmgr_destructive_get_total	counter	Gesamtabrufe mit Löschen in Intervall - Anzahl
ibmmq_qmgr_durable_subscription_alter_total	counter	Anzahl der Änderungen permanenter Subskriptionen
ibmmq_qmgr_durable_subscription_create_total	counter	Anzahl der Erstellungen permanenter Subskriptionen
ibmmq_qmgr_durable_subscription_delete_total	counter	Anzahl der Löschungen permanenter Subskriptionen
ibmmq_qmgr_durable_subscription_resume_total	counter	Anzahl der Wiederaufnahmen permanenter Subskriptionen
ibmmq_qmgr_errors_file_system_free_space_percentage	gauge	MQ-Fehlerdateisystem - freier Speicherplatz
ibmmq_qmgr_errors_file_system_in_use_bytes	gauge	MQ-Fehlerdateisystem - belegte Bytes
ibmmq_qmgr_expired_message_total	counter	Anzahl abgelaufener Nachrichten
ibmmq_qmgr_failed_browse_total	counter	Anzahl fehlgeschlagener Anzeigevorgänge

Metrik	Typ	Beschreibung
ibmmq_qmgr_failed_mqcb_total	counter	Anzahl fehlgeschlagener MQCBs
ibmmq_qmgr_failed_mqclose_total	counter	Anzahl fehlgeschlagener MQCLOSE-Vorgänge
ibmmq_qmgr_failed_mqconn_mqconnx_total	counter	Anzahl fehlgeschlagener MQCONN/MQCONNX-Vorgänge
ibmmq_qmgr_failed_mqget_total	counter	Fehlgeschlagene MQGET-Vorgänge - Anzahl
ibmmq_qmgr_failed_mqinq_total	counter	Anzahl fehlgeschlagener MQINQ-Vorgänge
ibmmq_qmgr_failed_mqopen_total	counter	Anzahl fehlgeschlagener MQOPEN-Vorgänge
ibmmq_qmgr_failed_mqput1_total	counter	Anzahl fehlgeschlagener MQPUT1-Vorgänge
ibmmq_qmgr_failed_mqput_total	counter	Anzahl fehlgeschlagener MQPUT-Vorgänge
ibmmq_qmgr_failed_mqset_total	counter	Anzahl fehlgeschlagener MQSET-Vorgänge
ibmmq_qmgr_failed_mqsubrq_total	counter	Anzahl fehlgeschlagener MQSUBRQ-Vorgänge
ibmmq_qmgr_failed_subscription_create_alter_resume_total	counter	Anzahl fehlgeschlagener Erstellungen/Änderungen/Wiederaufnahmen von Subskriptionen
ibmmq_qmgr_failed_subscription_delete_total	counter	Anzahl fehlgeschlagener Löschungen von Subskriptionen
ibmmq_qmgr_failed_topic_mqput_mqput1_total	counter	Anzahl fehlgeschlagener MQPUT/MQPUT1-Vorgänge für Themen
ibmmq_qmgr_fdc_files	gauge	MQ-FDC-Dateizähler
ibmmq_qmgr_log_file_system_in_use_bytes	gauge	Protokolldateisystem - belegte Bytes
ibmmq_qmgr_log_file_system_max_bytes	gauge	Protokolldateisystem - max. Bytes
ibmmq_qmgr_log_in_use_bytes	gauge	Protokoll - belegte Bytes
ibmmq_qmgr_log_logical_written_bytes_total	counter	Protokoll - geschriebene logische Bytes

Metrik	Typ	Beschreibung
ibmmq_qmgr_log_max_bytes	gauge	Protokoll - max. Bytes
ibmmq_qmgr_log_physical_written_bytes_total	counter	Protokoll - geschriebene physische Bytes
ibmmq_qmgr_log_primary_space_in_use_percentage	gauge	Protokoll - aktuell belegter primärer Speicher
ibmmq_qmgr_log_workload_primary_space_utilization_percentage	gauge	Protokoll - Workloadnutzung des primären Speichers
ibmmq_qmgr_log_write_latency_seconds	gauge	Protokoll - Schreiblatenz
ibmmq_qmgr_log_write_size_bytes	gauge	Protokoll - Schreibgröße
ibmmq_qmgr_mqcb_total	counter	MQCB-Anzahl
ibmmq_qmgr_mqclose_total	counter	MQCLOSE-Anzahl
ibmmq_qmgr_mqconn_mqconnx_total	counter	MQCONN/MQCONNX-Anzahl
ibmmq_qmgr_mqctl_total	counter	MQCTL-Anzahl
ibmmq_qmgr_mqdisc_total	counter	MQDISC-Anzahl
ibmmq_qmgr_mqinq_total	counter	MQINQ-Anzahl
ibmmq_qmgr_mqopen_total	counter	MQOPEN-Anzahl
ibmmq_qmgr_mqput_mqput1_bytes_total	counter	MQPUT/MQPUT1-Byteanzahl in Intervall
ibmmq_qmgr_mqput_mqput1_total	counter	MQPUT/MQPUT1-Gesamtanzahl in Intervall
ibmmq_qmgr_mqset_total	counter	MQSET-Anzahl
ibmmq_qmgr_mqstat_total	counter	MQSTAT-Anzahl
ibmmq_qmgr_mqsubrq_total	counter	MQSUBRQ-Anzahl
ibmmq_qmgr_non_durable_subscription_create_total	counter	Anzahl der Erstellungen nicht permanenter Subskriptionen

Metrik	Typ	Beschreibung
ibmmq_qmgr_non_durable_subscription_delete_total	counter	Anzahl der Löschungen nicht permanenter Subskriptionen
ibmmq_qmgr_non_persistent_message_browse_bytes_total	counter	Anzeige nicht persistenter Nachrichten - Byteanzahl
ibmmq_qmgr_non_persistent_message_browse_total	counter	Anzeige nicht persistenter Nachrichten - Anzahl
ibmmq_qmgr_non_persistent_message_destructive_get_total	counter	Abruf nicht persistenter Nachrichten mit Löschen - Anzahl
ibmmq_qmgr_non_persistent_message_get_bytes_total	counter	Abruf nicht persistenter Nachrichten - Byteanzahl
ibmmq_qmgr_non_persistent_message_mqput1_total	counter	Anzahl nicht persistenter MQPUT1-Nachrichten
ibmmq_qmgr_non_persistent_message_mqput_total	counter	Anzahl nicht persistenter MQPUT-Nachrichten
ibmmq_qmgr_non_persistent_message_put_bytes_total	counter	Einreihung nicht persistenter Nachrichten - Byteanzahl
ibmmq_qmgr_non_persistent_topic_mqput_mqput1_total	counter	Nicht persistent - Anzahl der MQPUT/MQPUT1-Vorgänge für Themen
ibmmq_qmgr_persistent_message_browse_bytes_total	counter	Anzeige persistenter Nachrichten - Byteanzahl
ibmmq_qmgr_persistent_message_browse_total	counter	Anzeige persistenter Nachrichten - Anzahl
ibmmq_qmgr_persistent_message_destructive_get_total	counter	Abruf persistenter Nachrichten mit Löschen - Anzahl
ibmmq_qmgr_persistent_message_get_bytes_total	counter	Abruf persistenter Nachrichten - Byteanzahl

Metrik	Typ	Beschreibung
ibmmq_qmgr_persistent_message_mqput1_total	counter	Anzahl persistenter MQPUT1-Nachrichten
ibmmq_qmgr_persistent_message_mqput_total	counter	Anzahl persistenter MQPUT-Nachrichten
ibmmq_qmgr_persistent_message_put_bytes_total	counter	Einreihung persistenter Nachrichten - Byteanzahl
ibmmq_qmgr_persistent_topic_mqput_mqput1_total	counter	Persistent - Anzahl der MQPUT/MQPUT1-Vorgänge für Themen
ibmmq_qmgr_published_to_subscribers_bytes_total	counter	Für Subskribenten veröffentlicht - Byteanzahl
ibmmq_qmgr_published_to_subscribers_message_total	counter	Für Subskribenten veröffentlicht - Nachrichtenanzahl
ibmmq_qmgr_purged_queue_total	counter	Anzahl bereinigter Warteschlangen
ibmmq_qmgr_queue_manager_file_system_free_space_percentage	gauge	Dateisystem des Warteschlangenmanagers - freier Speicherplatz
ibmmq_qmgr_queue_manager_file_system_in_use_bytes	gauge	Dateisystem des Warteschlangenmanagers - belegte Bytes
ibmmq_qmgr_ram_free_percentage	gauge	Prozentsatz des freien RAM
ibmmq_qmgr_ram_usage_estimate_for_queue_manager_bytes	gauge	RAM-Gesamtbytes - Schätzung für Warteschlangenmanager
ibmmq_qmgr_rollback_total	counter	Rollback-Anzahl
ibmmq_qmgr_system_cpu_time_estimate_for_queue_manager_percentage	gauge	System-CPU-Zeit - geschätzter Prozentsatz für Warteschlangenmanager
ibmmq_qmgr_system_cpu_time_percentage	gauge	Prozentsatz der System-CPU-Zeit
ibmmq_qmgr_topic_mqput_mqput1_total	counter	Gesamtzahl der MQPUT/MQPUT1-Vorgänge für Themen in Intervall

Metrik	Typ	Beschreibung
ibmmq_qmgr_to-pic_put_bytes_to-tal	counter	Gesamtbytes eingereihter Themen in Intervall
ibmmq_qmgr_tra-ce_file_sys-tem_free_space_per centage	gauge	MQ-Tracedateisystem - freier Speicherplatz
ibmmq_qmgr_tra-ce_file_sys-tem_in_use_bytes	gauge	MQ-Tracedateisystem - belegte Bytes
ibmmq_qmgr_user_cp u_time_estima-te_for_queue_mana-ger_percentage	gauge	Benutzer-CPU-Zeit - geschätzter Prozentsatz für Warteschlangenmanager
ibmmq_qmgr_user_cp u_time_percentage	gauge	Prozentsatz der Benutzer-CPU-Zeit

CP4I CD V 9.2.2

Status von nativen HA-Warteschlangenmanagern für zertifizierte IBM MQ -Container anzeigen

Für zertifizierte IBM MQ -Container können Sie den Status der nativen HA-Instanzen anzeigen, indem Sie den Befehl **dspmq** in einem der aktiven Pods ausführen.

Informationen zu diesem Vorgang

Wichtig:

Durch Ausführung des Befehls **dspmq** in einem der aktiven Pods können Sie den Betriebsstatus einer Warteschlangenmanagerinstanz anzeigen. Welche Informationen zurückgegeben werden, hängt davon ab, ob die Instanz aktiv oder ein Replikat ist. Die von der aktiven Instanz gelieferten Informationen sind verbindlich, während Informationen von Replikatknoten möglicherweise nicht auf dem neuesten Stand sind.

Sie können folgende Aktionen ausführen:

- Anzeigen, ob die Warteschlangenmanagerinstanz auf dem aktuellen Knoten aktiv oder ein Replikat ist.
- Anzeigen des Native HA-Betriebsstatus der Instanz auf dem aktuellen Knoten.
- Anzeigen des Betriebsstatus aller drei Instanzen in einer Native HA-Konfiguration.

Der Status einer Native HA-Konfiguration wird in folgenden Statusfeldern gemeldet:

ROLE
Gibt die aktuelle Rolle der Instanz an. Die gültigen Werte sind `Active`, `Replica`und Unknown.

INSTANCE
Der Name, der für diese Instanz des Warteschlangenmanagers angegeben wurde, als sie mit der Option **-lr** des Befehls **crtmqm** erstellt wurde.

INSYNC
Gibt an, ob die Instanz bei Bedarf die Rolle der aktiven Instanz übernehmen kann.

QUORUM
Gibt den Quorumstatus im Format *Anzahl_synchrone_Instanzen*/*Anzahl_konfigurierte_Instanzen* an.

REPLADDR
Gibt die Replikationsadresse der Warteschlangenmanagerinstanz an.

CONNACTV
Gibt an, ob der Knoten mit der aktiven Instanz verbunden ist.

BACKLOG
Gibt die Anzahl KB an, um die die Instanz zurückliegt.

CONNINST
Gibt an, ob die benannte Instanz mit dieser Instanz verbunden ist.

ALTDATE
Gibt das Datum der letzten Aktualisierung dieser Informationen an (leer, wenn sie noch nie aktualisiert wurden).

ALTTIME
Gibt die Zeit der letzten Aktualisierung dieser Informationen an (leer, wenn sie noch nie aktualisiert wurden).

Prozedur

- Suchen Sie die Pods, die zu Ihrem Warteschlangenmanager gehören.

```
oc get pod --selector app.kubernetes.io/instance=nativeha-qm
```

- Führen Sie dspmqin einem der Pods aus

```
oc exec -t Pod dspmq
```

```
oc rsh Pod
```

für eine interaktive Shell, in der Sie dspmqdirekt ausführen können.

- Stellen Sie fest, ob eine Warteschlangenmanagerinstanz als aktive Instanz oder als Replikat ausgeführt wird:

```
oc exec -t Pod dspmq -o status -m QMgrName
```

Eine aktive Instanz eines Warteschlangenmanagers mit dem Namen BOB würde folgenden Status melden:

```
QMNAME(BOB)                STATUS(Running)
```

Eine Replikatinstanz eines Warteschlangenmanagers mit dem Namen BOB würde folgenden Status melden:

```
QMNAME(BOB)                STATUS(Replica)
```

Eine inaktive Instanz würde folgenden Status melden:

```
QMNAME(BOB)                STATUS(Ended Immediately)
```

- Stellen Sie den Native HA-Betriebsstatus der Instanz im angegebenen Pod fest:

```
oc exec -t Pod dspmq -o nativeha -m QMgrName
```

Die aktive Instanz eines Warteschlangenmanagers mit dem Namen BOB könnte folgenden Status melden:

```
QMNAME(BOB)              ROLE(Active) INSTANCE(inst1) INSYNC(Yes) QUORUM(3/3)
```

Eine Replikatinstanz eines Warteschlangenmanagers mit dem Namen BOB könnte folgenden Status melden:

```
QMNAME(BOB)              ROLE(Replica) INSTANCE(inst2) INSYNC(Yes) QUORUM(2/3)
```

Eine inaktive Instanz eines Warteschlangenmanagers mit dem Namen BOB könnte folgenden Status melden:

```
QMNAME(BOB)               ROLE(Unknown) INSTANCE(inst3) INSYNC(no) QUORUM(0/3)
```

- Stellen Sie den Native HA-Betriebsstatus aller Instanzen in der Native HA-Konfiguration fest:

```
oc exec -t Pod dspmq -o nativeha -x -m QMgrName
```

Wenn Sie diesen Befehl auf dem Knoten mit der aktiven Instanz des Warteschlangenmanagers BOB ausgeben, könnte der folgende Status gemeldet werden:

```
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)
```

Wenn Sie diesen Befehl auf einem Knoten mit einer Replikatinstanz des Warteschlangenmanagers BOB ausgeben, könnte der folgende Status gemeldet werden, der anzeigt, dass eins der Replikate im Rückstand ist:

```
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)
```

Wenn Sie diesen Befehl auf einem Knoten mit einer inaktiven Instanz des Warteschlangenmanagers BOB ausgeben, könnte der folgende Status gemeldet werden:

```
QMNAME(BOB)            ROLE(Unknown) INSTANCE(inst3) INSYNC(no) QUORUM(0/3)
 INSTANCE(inst1) ROLE(Unknown) REPLADDR(9.20.123.45) CONNACTV(Unknown) INSYNC(Unknown) BACK
LOG(Unknown) CONNINST(No) ALTDATE() ALTTIME()
 INSTANCE(inst2) ROLE(Unknown) REPLADDR(9.20.123.46) CONNACTV(Unknown) INSYNC(Unknown) BACK
LOG(Unknown) CONNINST(No) ALTDATE() ALTTIME()
 INSTANCE(inst3) ROLE(Unknown) REPLADDR(9.20.123.47) CONNACTV(No) INSYNC(Unknown) BACKLOG(Un
known) CONNINST(No) ALTDATE() ALTTIME()
```

Wenn Sie den Befehl ausgeben, während die Instanzen aushandeln, welche die aktive Instanz und und welche die Replikate sind, würden Sie folgenden Status empfangen:

```
QMNAME(BOB)              STATUS(Negotiating)
```

Zugehörige Verweise

Befehl dspmq (Warteschlangenmanager anzeigen)

„Beispiel: Einen nativen HA-Warteschlangenmanager konfigurieren” auf Seite 98

Dieses Beispiel zeigt, wie Sie einen WS-Manager mit der nativen Hochverfügbarkeitsfunktion in Red Hat OpenShift Container Platform (OCP) unter Verwendung von IBM MQ Operatorimplementieren.

OpenShift CP4I Warteschlangenmanagerkonfiguration über die Red Hat OpenShift-CLI sichern und wiederherstellen

Die Warteschlangenmanagerkonfiguration zu sichern, kann dabei helfen, einen Warteschlangenmanager aus seinen Definitionen erneut zu erstellen, wenn die Warteschlangenmanagerkonfiguration verloren geht. Bei dieser Prozedur werden keine Warteschlangenmanagerprotokolldaten gesichert. Aufgrund des temporären Charakters von Nachrichten sind Langzeitprotokolldaten zum Zeitpunkt der Wiederherstellung normalerweise nicht mehr relevant.

Vorbereitende Schritte

Melden Sie sich mit **cloudctl login** (für IBM Cloud Pak for Integration) oder **oc login** bei Ihrem Cluster an.

Prozedur

- Sichern Sie die Warteschlangenmanagerkonfiguration.

  Mit dem Befehl **dmpmqcfg** können Sie einen Speicherauszug der Konfiguration eines IBM MQ-Warteschlangenmanagers erstellen.

  a) Rufen Sie den Namen des Pods für Ihren Warteschlangenmanager ab.
  Sie könnten beispielsweise folgenden Befehl ausführen, wobei *Warteschlangenmanagername* für den Namen Ihrer `QueueManager`-Ressource steht:

  ```
  oc get pods --selector app.kubernetes.io/name=ibm-mq,app.kubernetes.io/instance=queue_ma
  nager_name
  ```

  b) Führen Sie den Befehl **dmpmqcfg** auf dem Pod aus und übertragen Sie dabei die Ausgabe in eine Datei auf Ihrer lokalen Maschine.

  dmpmqcfg erstellt eine Ausgabe der MQSC-Konfiguration des Warteschlangenmanagers.

  ```
  oc exec -it pod_name -- dmpmqcfg > backup.mqsc
  ```

- Stellen Sie die Warteschlangenmanagerkonfiguration wieder her.

  Nachdem Sie die im vorherigen Schritt beschriebene Sicherungsprozedur ausgeführt haben, sollten Sie über eine Datei `backup.mqsc` verfügen, die die Konfiguration des Warteschlangenmanagers enthält. Sie können die Konfiguration wiederherstellen, indem Sie diese Datei auf einen neuen Warteschlangenmanager anwenden.

  a) Rufen Sie den Namen des Pods für Ihren Warteschlangenmanager ab.
  Sie könnten beispielsweise folgenden Befehl ausführen, wobei *Warteschlangenmanagername* für den Namen Ihrer `QueueManager`-Ressource steht:

  ```
  oc get pods --selector app.kubernetes.io/name=ibm-mq,app.kubernetes.io/instance=queue_ma
  nager_name
  ```

  b) Führen Sie den Befehl **runmqsc** auf dem Pod aus und übertragen Sie dabei den Inhalt der Datei `backup.mqsc` aus.

  ```
  oc exec -i pod_name -- runmqsc < backup.mqsc
  ```

OpenShift CP4I Fehlerbehebung bei Problemen mit IBM MQ Operator

Bei Problemen mit IBM MQ Operator helfen Ihnen die erläuterten Verfahren bei der Diagnose und Behebung.

Prozedur

- „Fehlerbehebung: Zugriff auf Warteschlangenmanagerdaten erhalten” auf Seite 130

OpenShift CP4I Fehlerbehebung: Zugriff auf Warteschlangenmanagerdaten erhalten

Verwenden Sie das PVC-Inspektor-Tool, um Zugriff auf die Dateien auf einem Warteschlangenmanager-PVC zu erhalten, wenn keine ferne Shell für den Warteschlangenmanager-Pod eingerichtet werden kann. Dies kann daran liegen, dass sich der Pod im Status **Error** oder **CrashLoopBackOff** befindet. Dieses

Tool ist für die Verwendung mit Warteschlangenmanagern konzipiert, die von IBM MQ Operatorimplementiert werden.

Vorbereitende Schritte

Zum Verwenden des PVC-Inspektortools. Sie müssen Zugriff auf Ihren Warteschlangenmanager-Namensbereich haben.

Informationen zu diesem Vorgang

Zur Unterstützung der Fehlerbehebung können Sie auf die Daten zugreifen, die in den PVCs gespeichert sind, die einem bestimmten Warteschlangenmanager zugeordnet sind. Dazu verwenden Sie ein Tool, um die PVCs an eine Gruppe von Inspector-Pods anzuhängen. Sie können dann eine ferne Shell in jeden der Inspector-Pods abrufen, um die Dateien zu lesen.

Je nach Bereitstellungstyp werden zwischen einem und drei Inspector-Pods erstellt. Datenträger, die für einen bestimmten Pod eines nativen HA-oder Multi-Instanz-Warteschlangenmanagers spezifisch sind, sind im zugehörigen PVC-Inspector-Pod verfügbar. Gemeinsam genutzte Datenträger sind auf allen Prüfern verfügbar. Der Name des Inspector-Pods enthält den Namen des zugehörigen Warteschlangenmanager-Pods.

Vorgehensweise

1. Laden Sie das MQ -Tool PVC Inspector herunter.

   Das Tool ist hier verfügbar: https://github.com/ibm-messaging/mq-pvc-tool.
2. Stellen Sie sicher, dass Sie bei Ihrem Cluster angemeldet sind.
3. Ermitteln Sie den Namen des Warteschlangenmanagers und den Namensbereich, in dem der Warteschlangenmanager ausgeführt wird.
4. Führen Sie das Inspector-Tool für Ihren Warteschlangenmanager aus.
   a) Führen Sie den folgenden Befehl aus und geben Sie dabei den Namen Ihres Warteschlangenmanagers und seinen Namensbereichsnamen an.

   ```
   ./pvc-tool.sh queue_manager_name queue_manager_namespace_name
   ```

   b) Führen Sie nach Abschluss des Tools den folgenden Befehl aus, um die erstellten Inspector-Pods anzuzeigen.

   ```
   oc get pods
   ```

5. Zeigen Sie die Dateien an, die an den Inspector-Pod angehängt sind.
   a) Jeder PVC-Inspector-Pod ist einem Warteschlangenmanager-Pod zugeordnet, sodass es mehrere Inspector-Pods geben kann. Greifen Sie auf einen dieser Pods zu, indem Sie den folgenden Befehl ausführen:

   ```
   oc rsh pvc-inspector-pod-name
   ```

   Sie befinden sich in dem Verzeichnis, das die angehängten PVC-Verzeichnisse enthält.
   b) Öffnen Sie eine ferne Shell im Pod, indem Sie den folgenden Befehl ausführen:

   ```
   ls
   ```

   c) Sie können Verzeichnisse anzeigen, die denselben Namen wie die angehängten PVCs haben. Greifen Sie auf die Dateien auf den PVCs des Warteschlangenmanagers zu, indem Sie diese Verzeichnisse durchsuchen. Führen Sie den folgenden Befehl außerhalb der fernen Shellsitzung aus, um eine Liste der PVCs anzuzeigen:

   ```
   oc get pvc
   ```

   d) Bereinigen Sie die vom Tool erstellten Pods, indem Sie den folgenden Befehl ausführen:

```
'oc delete pods -l tool=mq-pvc-inspector
```

OpenShift CP4I API-Referenz für IBM MQ Operator

IBM MQ stellt einen Kubernetes-Operator für die native Integration mit Red Hat OpenShift Container Platform bereit.

OpenShift CP4I API-Referenz für mq.ibm.com/v1beta1

Über die API v1beta1 können QueueManager-Ressourcen erstellt und verwaltet werden.

OpenShift CP4I CD EUS *Lizenzierungsreferenz für mq.ibm.com/v1beta1*

Aktuelle Lizenzversionen

Das Feld `spec.license.license`muss die Lizenzkennung für die Lizenz enthalten, die Sie akzeptieren. Gültige Werte sind:

Wert von `spec.license.license`	Wert von `spec.license.use`	Lizenzinformation	Zutreffende IBM MQ-Versionen
`L-RJON-C7QG3S`	`Production` oder `NonProduction`	IBM Cloud Pak for Integration 2021.4.1	9.2.4 oder 9.2.5
`L-RJON-C7QFZX`	`Production` oder `NonProduction`	IBM Cloud Pak for Integration Limited Edition 2021.4.1	9.2.4 oder 9.2.5
`L-RJON-C5CSNH`	`Production` oder `NonProduction`	IBM Cloud Pak for Integration 2021.3.1	9.2.3 oder 9.2.4
`L-RJON-C5CSM2`	`Production` oder `NonProduction`	IBM Cloud Pak for Integration Limited Edition 2021.3.1	9.2.3 oder 9.2.4
`L-RJON-BZFQU2`	`Production` oder `NonProduction`	IBM Cloud Pak for Integration 2021.2.1	9.2.3
`L-RJON-BZFQSB`	`Production` oder `NonProduction`	IBM Cloud Pak for Integration Limited Edition 2021.2.1	9.2.3
`L-RJON-BUVMQX`	`Production` oder `NonProduction`	IBM Cloud Pak for Integration 2020.4.1	9.2.0 EUS oder 9.2.1
`L-RJON-BUVMYB`	`Production` oder `NonProduction`	IBM Cloud Pak for Integration Limited Edition 2020.4.1	9.2.0 EUS oder 9.2.1
`L-APIG-BZDDDY`	`Production`	IBM MQ Advanced und IBM MQ Advanced für Non-Production Environment 9.2-07/2021	9.2.3, 9.2.4 oder 9.2.5
`L-APIG-BYHCL7`	`Development`	IBM MQ Advanced for Developers (ohne Gewährleistung) V9.2 -07/2021	9.2.3, 9.2.4 oder 9.2.5
`L-APIG-BVJJB3`	`Production`	IBM MQ Advanced und 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 oder 9.2.1
`L-APIG-BMKG5H`	`Development`	IBM MQ Advanced for Developers (ohne Gewährleistung) V9.2	9.2.0 CD, 9.2.1 oder 9.2.2

Beachten Sie, dass die *Version* der Lizenz angegeben ist, die nicht immer mit der Version von IBM MQ identisch ist.

Ältere Lizenzversionen

Das Feld `spec.license.license`muss die Lizenzkennung für die Lizenz enthalten, die Sie akzeptieren. Gültige Werte sind:

Wert von `spec.license.license`	Wert von `spec.license.use`	Lizenzinformation	Zutreffende IBM MQ-Versionen
`L-RJON-BXUPZ2`	`Production` oder `NonProduction`	IBM Cloud Pak for Integration 2021.1.1	9.2.2
`L-RJON-BXUQ34`	`Production` oder `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. Frühes Release für die Verwendung mit Native HA ausschließlich mit IBM MQ Operator 1.5.	9.2.2
`L-RJON-BQPGWD`	`Production` oder `NonProduction`	IBM Cloud Pak for Integration 2020.3.1	9.2.0 CD
`L-RJON-BN7PN3`	`Production` oder `NonProduction`	IBM Cloud Pak for Integration 2020.2.1	9.1.5 oder 9.2.0 CD
`L-RJON-BPHL2Y`	`Production` oder `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 (ohne Gewährleistung) V9.1 -04/2020	9.1.5

Beachten Sie, dass die *Version* der Lizenz angegeben ist, die nicht immer mit der Version von IBM MQ identisch ist.

OpenShift CP4I ***API-Referenz für QueueManager (mq.ibm.com/v1beta1)***

QueueManager

Ein QueueManager ist ein IBM MQ-Server, der Warteschlangensteuerungs- und Publish/Subscribe-Services für Anwendungen bereitstellt.

Feld	Beschreibung
`apiVersion` Zeichenfolge	'apiVersion' gibt das versionierte Schema dieser Darstellung eines Objekts an. Server sollten erkannte Schemas in den letzten internen Wert umwandeln und können nicht erkannte Werte zurückweisen. Weitere Informationen: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#resources.
`kind` Zeichenfolge	'kind' ist ein Zeichenfolgewert zur Darstellung der REST-Ressource, die dieses Objekt darstellt. Server können dies von dem Endpunkt ableiten, an den der Client Anforderungen übergibt. Kann nicht aktualisiert werden. In Kamelschreibweise. Weitere Informationen: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds.
`metadata`	

Feld	Beschreibung
`spec` QueueManagerSpec	Der Soll-Status des QueueManager.
`status` QueueManagerStatus	Der Ist-Status des QueueManager.

.spec

Der Soll-Status des QueueManager.

Wird angezeigt in:

- „QueueManager” auf Seite 133

Feld	Beschreibung
`affinity`	Standardaffinitätsregeln von Kubernetes. Weitere Informationen: https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.17/#affinity-v1-core.
`annotations` Anmerkungen	Das Feld 'annotations' dient als Durchgriff für Pod-Anmerkungen. Benutzer können diesem Feld beliebige Anmerkungen hinzufügen und sie für ihren Pod verwenden. Durch die hier bereitgestellten Anmerkungen werden die Standardanmerkungen überschrieben. Setzt MQ Operator 1.3.0 oder höher voraus.
`imagePullSecrets` LocalObjectReference Array	Eine optionale Liste mit Verweisen auf Secrets in demselben Namensbereich, die zum Extrahieren eines der von diesem Warteschlangenmanager genutzten Images verwendet werden sollen. Wenn angegeben, werden diese Secrets an Implementierungen individueller Extraktionsprogramme übergeben, damit diese sie verwenden. Beispielsweise werden im Falle von Docker nur Secrets des Typs DockerConfig berücksichtigt. Weitere Informationen siehe https://kubernetes.io/docs/concepts/containers/images#specifying-imagepullsecrets-on-a-pod.
`labels` Beschriftungen	Das Feld 'labels' dient als Durchgriff für Pod-Beschriftungen. Benutzer können diesem Feld beliebige Beschriftungen hinzufügen und sie für ihren Pod verwenden. Durch die hier bereitgestellten Beschriftungen werden die Standardbeschriftungen überschrieben. Setzt MQ Operator 1.3.0 oder höher voraus.
`license` License	Einstellungen zur Steuerung der Annahme der Lizenz und der zu verwendenden Lizenzmetrik.
`pki` PKI	Public Key Infrastructure-Einstellungen zur Definition von Schlüsseln und Zertifikaten für die Verwendung mit Transport Layer Security (TLS) oder MQ Advanced Message Security (AMS).
`queueManager` QueueManagerConfig	Einstellungen für den Warteschlangenmanagercontainer und den zugrunde liegenden Warteschlangenmanager.
`securityContext` SecurityContext	Sicherheitseinstellungen, die dem Sicherheitskontext (securityContext) des Warteschlangenmanager-Pods hinzugefügt werden sollen.
`template` Template	Erweiterte Erstellung anhand einer Vorlage für Kubernetes-Ressourcen. Mit der Vorlage können Benutzer durch Überschreibungen ändern, wie IBM MQ die zugrunde liegenden Kubernetes-Ressourcen, z. B. StatefulSet, Pods und Services, generiert. Dies ist nur für fortgeschrittene Benutzer, da bei falscher Verwendung die Möglichkeit besteht, dass der normale Betrieb von MQ unterbrochen wird. Alle an anderen Stellen in der QueueManager-Ressource angegebenen Werte werden durch Einstellungen in der Vorlage überschrieben.

Feld	Beschreibung
`terminationGracePeriodSeconds` Ganzzahl	Optionale Dauer in Sekunden, die der Pod zum ordnungsgemäßen Beenden benötigt. Der Wert muss eine nicht negative Ganzzahl sein. Null bedeutet sofortiges Löschen. Die Soll-Zeit für den Versuch, den Warteschlangenmanager zu beenden, wobei die Phasen des Anwendungsverbindungsabbaus eskaliert werden. Wichtige Wartungsvorgänge des Warteschlangenmanagers werden, falls nötig, unterbrochen. Der Standardwert ist 30 Sekunden.
`tracing` TracingConfig	Einstellungen für Tracing-Integration mit dem Cloud Pak for Integration-Dashboard 'Operations'.
`version` Zeichenfolge	Einstellung zur Steuerung der MQ-Version, die verwendet wird (erforderlich). Beispiel: `9.1.5.0-r2` gibt MQ-Version 9.1.5.0 unter Verwendung der zweiten Revision des Container-Image an. Containerspezifische Fixes werden häufig in Form von Revisionen angewendet, z. B. Fixes für das Basisimage.
`web` WebServerConfig	Einstellungen für den MQ-Web-Server.

.spec.annotations

Das Feld 'annotations' dient als Durchgriff für Pod-Anmerkungen. Benutzer können diesem Feld beliebige Anmerkungen hinzufügen und sie für ihren Pod verwenden. Durch die hier bereitgestellten Anmerkungen werden die Standardanmerkungen überschrieben. Setzt MQ Operator 1.3.0 oder höher voraus.

Wird angezeigt in:

- „.spec” auf Seite 134

.spec.imagePullSecrets

LocalObjectReference enthält genug Informationen, damit Sie das referenzierte Objekt innerhalb desselben Namensbereichs lokalisieren können.

Wird angezeigt in:

- „.spec” auf Seite 134

Feld	Beschreibung
`name` Zeichenfolge	Name des Referenten. Weitere Informationen: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names Aufgabe: Fügen Sie weitere nützliche Felder hinzu, z. B. apiVersion, kind, uid?.

.spec.labels

Das Feld 'labels' dient als Durchgriff für Pod-Beschriftungen. Benutzer können diesem Feld beliebige Beschriftungen hinzufügen und sie für ihren Pod verwenden. Durch die hier bereitgestellten Beschriftungen werden die Standardbeschriftungen überschrieben. Setzt MQ Operator 1.3.0 oder höher voraus.

Wird angezeigt in:

- „.spec” auf Seite 134

.spec.license

Einstellungen zur Steuerung der Annahme der Lizenz und der zu verwendenden Lizenzmetrik.

Wird angezeigt in:

- „.spec” auf Seite 134

Feld	Beschreibung
`accept` boolesch	Gibt an, ob Sie die Lizenz für diese Software akzeptieren (erforderlich).
`license` Zeichenfolge	Die Kennung der von Ihnen akzeptierten Lizenz. Dies muss die richtige Lizenzkennung für die verwendete MQ-Version sein. Gültige Werte siehe http://ibm.biz/BdqvCF.
`metric` Zeichenfolge	Einstellung zur Angabe der zu verwendenden Lizenzmetrik. Beispiele: `ProcessorValueUnit`, `VirtualProcessorCore` oder `ManagedVirtualServer`. Der Standardwert ist `ProcessorValueUnit` bei einer MQ-Lizenz und `VirtualProcessorCore` bei einer Cloud Pak for Integration-Lizenz.
`use` Zeichenfolge	Einstellung zur Steuerung der Art der Verwendung der Software, wenn die Lizenz Mehrfachverwendungen unterstützt. Gültige Werte siehe http://ibm.biz/BdqvCF.

.spec.pki

Public Key Infrastructure-Einstellungen zur Definition von Schlüsseln und Zertifikaten für die Verwendung mit Transport Layer Security (TLS) oder MQ Advanced Message Security (AMS).

Wird angezeigt in:

- „.spec" auf Seite 134

Feld	Beschreibung
`keys` PKISource Array	Private Schlüssel, die dem Schlüsselrepository des Warteschlangenmanagers hinzugefügt werden sollen.
`trust` PKISource Array	Zertifikate, die dem Schlüsselrepository des Warteschlangenmanagers hinzugefügt werden sollen.

.spec.pki.keys

PKISource definiert eine Quelle von Public Key Infrastructure-Informationen, z. B. Schlüssel oder Zertifikate.

Wird angezeigt in:

- „.spec.pki" auf Seite 136

Feld	Beschreibung
`name` Zeichenfolge	Der Name wird als Bezeichnung für den Schlüssel oder das Zertifikat verwendet. Muss eine alphanumerische Zeichenfolge in Kleinschreibung sein.
`secret` Secret	Übergeben Sie einen Schlüssel mithilfe eines Kubernetes-Secrets.

.spec.pki.keys.secret

Übergeben Sie einen Schlüssel mithilfe eines Kubernetes-Secrets.

Wird angezeigt in:

- „.spec.pki.keys" auf Seite 136

Feld	Beschreibung
`items` Array	Schlüssel im Kubernetes-Secret, die zum Container des Warteschlangenmanagers hinzugefügt werden sollen.
`secretName` Zeichenfolge	Der Name des Kubernetes-Secrets.

.spec.pki.trust

PKISource definiert eine Quelle von Public Key Infrastructure-Informationen, z. B. Schlüssel oder Zertifikate.

Wird angezeigt in:

- „.spec.pki" auf Seite 136

Feld	Beschreibung
name Zeichenfolge	Der Name wird als Bezeichnung für den Schlüssel oder das Zertifikat verwendet. Muss eine alphanumerische Zeichenfolge in Kleinschreibung sein.
secret Secret	Übergeben Sie einen Schlüssel mithilfe eines Kubernetes-Secrets.

.spec.pki.trust.secret

Übergeben Sie einen Schlüssel mithilfe eines Kubernetes-Secrets.

Wird angezeigt in:

- „.spec.pki.trust" auf Seite 137

Feld	Beschreibung
items Array	Schlüssel im Kubernetes-Secret, die zum Container des Warteschlangenmanagers hinzugefügt werden sollen.
secretName Zeichenfolge	Der Name des Kubernetes-Secrets.

.spec.queueManager

Einstellungen für den Warteschlangenmanagercontainer und den zugrunde liegenden Warteschlangenmanager.

Wird angezeigt in:

- „.spec" auf Seite 134

Feld	Beschreibung
availability Availability	Verfügbarkeitseinstellungen für den Warteschlangenmanager, z. B., ob ein aktives Standby-Paar oder eine native Hochverfügbarkeit verwendet werden soll oder nicht.
debug boolesch	Gibt an, ob Debugnachrichten aus dem containerspezifischen Code in das Containerprotokoll übernommen werden sollen oder nicht. Der Standardwert ist 'false'.
image Zeichenfolge	Das Container-Image, das verwendet wird.
imagePullPolicy Zeichenfolge	Diese Einstellung steuert, wann der Kubelet versucht, das angegebene Image zu extrahieren. Der Standardwert ist IfNotPresent.
ini INISource Array	Einstellungen für die Bereitstellung von INI für den Warteschlangenmanager. Erfordert MQ Operator 1.1.0 oder höher.
livenessProbe QueueManagerLivenessProbe	Einstellungen zur Steuerung des Livetests.
logFormat Zeichenfolge	Gibt das Protokollformat für diesen Container an. Verwenden Sie JSON für JSON-formatierte Protokolle von dem Container. Verwenden Sie Basic für textformatierte Nachrichten. Der Standardwert ist Basic.

Feld	Beschreibung
`metrics` QueueManagerMetrics	Einstellungen für Metriken im Prometheus-Stil.
`mqsc` MQSCSource Array	Einstellungen für die Bereitstellung von MQSC für den Warteschlangenmanager. Erfordert MQ Operator 1.1.0 oder höher.
`name` Zeichenfolge	Name des zugrunde liegenden MQ-Warteschlangenmanagers, falls er vom metadata.name abweicht. Verwenden Sie dieses Feld, wenn der gewünschte Warteschlangenmanagername nicht den Kubernetes-Regeln für Namen entspricht (z. B. ein Name, der Großbuchstaben enthält).
`readinessProbe` QueueManagerReadinessProbe	Einstellungen zur Steuerung des Bereitschaftstests.
`resources` Resources	Einstellungen zur Steuerung der Ressourcenanforderungen.
`route` Route	Einstellungen für die Warteschlangenmanagerroute. Setzt MQ Operator 1.4.0 oder höher voraus.
`startupProbe` StartupProbe	Einstellungen zur Steuerung des Starttests. Gilt nur für MultiInstance- und NativeHA-Implementierungen. Erfordert MQ Operator 1.5.0 oder höher.
`storage` QueueManagerStorage	Speichereinstellungen zur Steuerung der Verwendung von Persistent Volumes und Speicherklassen durch den Warteschlangenmanager.

.spec.queueManager.availability

Verfügbarkeitseinstellungen für den Warteschlangenmanager, z. B., ob ein aktives Standby-Paar oder eine native Hochverfügbarkeit verwendet werden soll oder nicht.

Wird angezeigt in:

- „.spec.queueManager" auf Seite 137

Feld	Beschreibung
`tls` Tls	Optionale TLS-Einstellungen für die Konfiguration einer sicheren Kommunikation zwischen NativeHA-Replikaten. Erfordert MQ Operator 1.5.0 oder höher.
`type` Zeichenfolge	Verfügbarkeitstyp, der verwendet werden soll. Verwenden Sie `SingleInstance` für einen einzelnen Pod, der (in einigen Fällen) von Kubernetes automatisch erneut gestartet wird. Verwenden Sie `MultiInstance` für ein Podpaar, von denen einer der `aktive` Warteschlangenmanager und der andere eine Standby-Instanz ist. Verwenden Sie `NativeHA` für eine native Replikation mit hoher Verfügbarkeit (erfordert MQ Operator 1.5.0 oder höher). Der Standardwert ist `SingleInstance`. Weitere Details siehe http://ibm.biz/BdqAQa.
`updateStrategy` -Zeichenfolge	Die Aktualisierungsstrategie, die für MultiInstance- und NativeHA-Warteschlangenmanager verwendet werden soll. Verwenden Sie `RollingUpdate`, um automatisch rollende Aktualisierungen zu aktivieren, wenn sich die Konfiguration des Warteschlangenmanagers ändert. Verwenden Sie `OnDelete`, um automatische rollende Aktualisierungen zu deaktivieren. Warteschlangenmanager-Änderungen werden nur angewendet, wenn Pods gelöscht werden (einschließlich Pod-Löschungen, die durch externe Faktoren ausgelöst werden). Der Standardwert ist `RollingUpdate`. Erfordert MQ-Operator 1.6.0 oder höher.

.spec.queueManager.availability.tls

Optionale TLS-Einstellungen für die Konfiguration einer sicheren Kommunikation zwischen NativeHA-Replikaten. Erfordert MQ Operator 1.5.0 oder höher.

Wird angezeigt in:

- „.spec.queueManager.availability” auf Seite 138

Feld	Beschreibung
`cipherSpec` Zeichenfolge	Der Name der CipherSpec für NativeHA-TLS.
`secretName` Zeichenfolge	Der Name des Kubernetes-Secrets.

.spec.queueManager.ini

Quelle von INI-Konfigurationsdateien.

Wird angezeigt in:

- „.spec.queueManager” auf Seite 137

Feld	Beschreibung
`configMap` ConfigMapINI-Source	'ConfigMap' stellt eine Kubernetes-ConfigMap dar, die INI-Informationen enthält.
`secret` SecretINISource	'Secret' stellt ein Kubernetes-Secret dar, das INI-Informationen enthält.

.spec.queueManager.ini.configMap

'ConfigMap' stellt eine Kubernetes-ConfigMap dar, die INI-Informationen enthält.

Wird angezeigt in:

- „.spec.queueManager.ini” auf Seite 139

Feld	Beschreibung
`items` Array	Schlüssel innerhalb der Kubernetes-Quelle, die angewendet werden sollen.
`name` Zeichenfolge	Der Name der Kubernetes-Quelle.

.spec.queueManager.ini.secret

'Secret' stellt ein Kubernetes-Secret dar, das INI-Informationen enthält.

Wird angezeigt in:

- „.spec.queueManager.ini” auf Seite 139

Feld	Beschreibung
`items` Array	Schlüssel innerhalb der Kubernetes-Quelle, die angewendet werden sollen.
`name` Zeichenfolge	Der Name der Kubernetes-Quelle.

.spec.queueManager.livenessProbe

Einstellungen zur Steuerung des Livetests.

Wird angezeigt in:

- „.spec.queueManager” auf Seite 137

Feld	Beschreibung
`failureThreshold` Ganzzahl	Mindestanzahl aufeinanderfolgender Fehler für den Test, damit er als fehlgeschlagen betrachtet wird, nachdem er erfolgreich war. Der Standardwert ist 1.

Feld	Beschreibung
`initialDelaySeconds` Ganzzahl	Anzahl der Sekunden nach dem Start des Containers, bevor der Test eingeleitet werden. Standardmäßig 90 Sekunden für SingleInstance. Standardmäßig 0 Sekunden für MultiInstance- und NativeHA-Implementierungen. Weitere Informationen: https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle#container-probes.
`periodSeconds` Ganzzahl	Gibt an, wie oft (in Sekunden) der Test durchzuführen ist. Der Standardwert ist 10 Sekunden.
`successThreshold` Ganzzahl	Mindestanzahl aufeinanderfolgender Erfolge für den Test, damit er als erfolgreich betrachtet wird, nachdem er fehlgeschlagen ist. Der Standardwert ist 1.
`timeoutSeconds` Ganzzahl	Anzahl der Sekunden, nach denen der Test das Zeitlimit überschreitet. Der Standardwert ist 5 Sekunden. Weitere Informationen: https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle#container-probes.

.spec.queueManager.metrics

Einstellungen für Metriken im Prometheus-Stil.

Wird angezeigt in:

- „.spec.queueManager” auf Seite 137

Feld	Beschreibung
`enabled` boolesch	Gibt an, ob ein Endpunkt für Prometheus-kompatible Metriken aktiviert wird. Der Standardwert ist 'true'.

.spec.queueManager.mqsc

Quelle von MQSC-Konfigurationsdateien.

Wird angezeigt in:

- „.spec.queueManager” auf Seite 137

Feld	Beschreibung
`configMap` ConfigMapMQSCSource	'ConfigMap' stellt eine Kubernetes-ConfigMap dar, die MQSC-Informationen enthält.
`secret` SecretMQSCSource	'Secret' stellt ein Kubernetes-Secret dar, das MQSC-Informationen enthält.

.spec.queueManager.mqsc.configMap

'ConfigMap' stellt eine Kubernetes-ConfigMap dar, die MQSC-Informationen enthält.

Wird angezeigt in:

- „.spec.queueManager.mqsc” auf Seite 140

Feld	Beschreibung
`items` Array	Schlüssel innerhalb der Kubernetes-Quelle, die angewendet werden sollen.
`name` Zeichenfolge	Der Name der Kubernetes-Quelle.

.spec.queueManager.mqsc.secret

'Secret' stellt ein Kubernetes-Secret dar, das MQSC-Informationen enthält.

Wird angezeigt in:

- „.spec.queueManager.mqsc” auf Seite 140

Feld	Beschreibung
`items` Array	Schlüssel innerhalb der Kubernetes-Quelle, die angewendet werden sollen.
`name` Zeichenfolge	Der Name der Kubernetes-Quelle.

.spec.queueManager.readinessProbe

Einstellungen zur Steuerung des Bereitschaftstests.

Wird angezeigt in:

- „.spec.queueManager” auf Seite 137

Feld	Beschreibung
`failureThreshold` Ganzzahl	Mindestanzahl aufeinanderfolgender Fehler für den Test, damit er als fehlgeschlagen betrachtet wird, nachdem er erfolgreich war. Der Standardwert ist 1.
`initialDelaySeconds` Ganzzahl	Anzahl der Sekunden nach dem Start des Containers, bevor der Test eingeleitet werden. Standardmäßig 10 Sekunden für SingleInstance. Standardmäßig 0 Sekunden für MultiInstance- und NativeHA-Implementierungen. Weitere Informationen: https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle#container-probes.
`periodSeconds` Ganzzahl	Gibt an, wie oft (in Sekunden) der Test durchzuführen ist. Der Standardwert ist 5 Sekunden.
`successThreshold` Ganzzahl	Mindestanzahl aufeinanderfolgender Erfolge für den Test, damit er als erfolgreich betrachtet wird, nachdem er fehlgeschlagen ist. Der Standardwert ist 1.
`timeoutSeconds` Ganzzahl	Anzahl der Sekunden, nach denen der Test das Zeitlimit überschreitet. Der Standardwert ist 3 Sekunden. Weitere Informationen: https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle#container-probes.

.spec.queueManager.resources

Einstellungen zur Steuerung der Ressourcenanforderungen.

Wird angezeigt in:

- „.spec.queueManager” auf Seite 137

Feld	Beschreibung
`limits` Limits	CPU- und Speichereinstellungen.
`requests` Requests	CPU- und Speichereinstellungen.

.spec.queueManager.resources.limits

CPU- und Speichereinstellungen.

Wird angezeigt in:

- „.spec.queueManager.resources” auf Seite 141

Feld	Beschreibung
`cpu`	

Feld	Beschreibung
memory	

.spec.queueManager.resources.requests

CPU- und Speichereinstellungen.

Wird angezeigt in:

- „.spec.queueManager.resources” auf Seite 141

Feld	Beschreibung
cpu	
memory	

.spec.queueManager.route

Einstellungen für die Warteschlangenmanagerroute. Setzt MQ Operator 1.4.0 oder höher voraus.

Wird angezeigt in:

- „.spec.queueManager” auf Seite 137

Feld	Beschreibung
enabled boolesch	Gibt an, ob die Route aktiviert werden soll. Der Standardwert ist 'true'.

.spec.queueManager.startupProbe

Einstellungen zur Steuerung des Starttests. Gilt nur für MultiInstance- und NativeHA-Implementierungen. Erfordert MQ Operator 1.5.0 oder höher.

Wird angezeigt in:

- „.spec.queueManager” auf Seite 137

Feld	Beschreibung
failureThreshold Ganzzahl	Mindestanzahl aufeinanderfolgender Fehler für den Test, damit er als fehlgeschlagen betrachtet wird. Der Standardwert ist 60.
initialDelaySeconds Ganzzahl	Anzahl der Sekunden nach dem Start des Containers, bevor der Test eingeleitet werden. Der Standardwert ist 0 Sekunden. Weitere Informationen: https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle#container-probes.
periodSeconds Ganzzahl	Gibt an, wie oft (in Sekunden) der Test durchzuführen ist. Der Standardwert ist 5 Sekunden.
successThreshold Ganzzahl	Mindestanzahl aufeinanderfolgender Erfolge für den Test, damit er als erfolgreich betrachtet wird. Der Standardwert ist 1.
timeoutSeconds Ganzzahl	Anzahl der Sekunden, nach denen der Test das Zeitlimit überschreitet. Der Standardwert ist 5 Sekunden. Weitere Informationen: https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle#container-probes.

.spec.queueManager.storage

Speichereinstellungen zur Steuerung der Verwendung von Persistent Volumes und Speicherklassen durch den Warteschlangenmanager.

Wird angezeigt in:

- „.spec.queueManager” auf Seite 137

Feld	Beschreibung
`defaultClass` Zeichenfolge	Speicherklasse, die standardmäßig auf alle Persistent Volumes dieses Warteschlangenmanagers angewendet werden soll. Für bestimmte Persistent Volumes kann eine eigene Speicherklasse festgelegt werden, die die Standardspeicherklasse außer Kraft setzt. Wenn `type of availability` (Typ der Verfügbarkeit) auf `SingleInstance` (Einzelinstanz) oder `NativeHA` (Native Hochverfügbarkeit) gesetzt ist, kann die Speicherklasse den Typ 'ReadWriteOnce' oder 'ReadWriteMany' haben. Ist `type of availability` dagegen `MultiInstance`, muss der Speicherklassentyp 'ReadWriteMany' sein.
`defaultDeleteClaim` boolesch	Gibt an, ob beim Löschen des Warteschlangenmanagers auch alle Volumes gelöscht werden sollen. Für bestimmte Persistent Volumes kann 'deleteClaim' gesondert eingestellt werden, wodurch die Einstellung von 'defaultDeleteClaim' für diese Volumes außer Kraft gesetzt wird. Der Standardwert ist 'false'.
`persistedData` QueueManagerOptionalVolume	Details zu Persistent Volumes für persistent gespeicherte MQ-Daten, einschließlich Konfiguration, Warteschlangen und Nachrichten. Erforderlich bei Verwendung eines Multi-Instanz-Warteschlangenmanagers.
`queueManager` QueueManagerVolume	Standardmäßiges Persistent Volume für alle Daten, die normalerweise unter `/var/mqm` gespeichert sind. Enthält alle persistent gespeicherten Daten und Wiederherstellungsprotokolle, wenn keine anderen Volumes angegeben werden.
`recoveryLogs` QueueManagerOptionalVolume	Details zu Persistent Volumes, die in die MQ-Wiederherstellungsprotokolle geschrieben werden. Erforderlich bei Verwendung eines Multi-Instanz-Warteschlangenmanagers.

.spec.queueManager.storage.persistedData

Details zu Persistent Volumes für persistent gespeicherte MQ-Daten, einschließlich Konfiguration, Warteschlangen und Nachrichten. Erforderlich bei Verwendung eines Multi-Instanz-Warteschlangenmanagers.

Wird angezeigt in:

- „.spec.queueManager.storage” auf Seite 142

Feld	Beschreibung
`class` Zeichenfolge	Speicherklasse, die für dieses Volume verwendet werden soll. Nur gültig, wenn `type` auf `persistent-claim` gesetzt ist. Wenn `type of availability` (Typ der Verfügbarkeit) auf `SingleInstance` (Einzelinstanz) oder `NativeHA` (Native Hochverfügbarkeit) gesetzt ist, kann die Speicherklasse den Typ 'ReadWriteOnce' oder 'ReadWriteMany' haben. Ist `type of availability` dagegen `MultiInstance`, muss der Speicherklassentyp 'ReadWriteMany' sein.
`deleteClaim` boolesch	Gibt an, ob dieses Volume beim Löschen des Warteschlangenmanagers gelöscht werden soll.
`enabled` boolesch	Gibt an, ob dieses Volume als separates Volume aktiviert oder auf dem `queueManager`-Standardvolume eingerichtet werden soll. Der Standardwert ist 'false'.
`size` Zeichenfolge	An Kubernetes zu übergebende Größe des Persistent Volumes, einschließlich SI-Einheiten. Nur gültig, wenn `type` auf `persistent-claim` gesetzt ist. Beispiel: `2Gi`. Der Standardwert ist `2Gi`.

Feld	Beschreibung
`sizeLimit` Zeichenfolge	Größenbegrenzung bei Verwendung eines `ephemeren` Volumes. Da Dateien weiterhin in ein temporäres Verzeichnis geschrieben werden, können Sie mit dieser Option die Größe begrenzen. Nur gültig, wenn `type` auf `ephemeral` gesetzt ist.
`type` Zeichenfolge	Typ des zu verwendenden Volumes. Wählen Sie `ephemeral` für einen nicht persistenten Speicher oder `persistent-claim` für ein Persistent Volume aus. Der Standardwert ist `persistent-claim`.

.spec.queueManager.storage.queueManager

Standardmäßiges Persistent Volume für alle Daten, die normalerweise unter `/var/mqm` gespeichert sind. Enthält alle persistent gespeicherten Daten und Wiederherstellungsprotokolle, wenn keine anderen Volumes angegeben werden.

Wird angezeigt in:

- „.spec.queueManager.storage” auf Seite 142

Feld	Beschreibung
`class` Zeichenfolge	Speicherklasse, die für dieses Volume verwendet werden soll. Nur gültig, wenn `type` auf `persistent-claim` gesetzt ist. Wenn `type of availability` (Typ der Verfügbarkeit) auf `SingleInstance` (Einzelinstanz) oder `Native-HA` (Native Hochverfügbarkeit) gesetzt ist, kann die Speicherklasse den Typ 'ReadWriteOnce' oder 'ReadWriteMany' haben. Ist `type of availability` dagegen `MultiInstance`, muss der Speicherklassentyp 'ReadWriteMany' sein.
`deleteClaim` boolesch	Gibt an, ob dieses Volume beim Löschen des Warteschlangenmanagers gelöscht werden soll.
`size` Zeichenfolge	An Kubernetes zu übergebende Größe des Persistent Volumes, einschließlich SI-Einheiten. Nur gültig, wenn `type` auf `persistent-claim` gesetzt ist. Beispiel: `2Gi`. Der Standardwert ist `2Gi`.
`sizeLimit` Zeichenfolge	Größenbegrenzung bei Verwendung eines `ephemeren` Volumes. Da Dateien weiterhin in ein temporäres Verzeichnis geschrieben werden, können Sie mit dieser Option die Größe begrenzen. Nur gültig, wenn `type` auf `ephemeral` gesetzt ist.
`type` Zeichenfolge	Typ des zu verwendenden Volumes. Wählen Sie `ephemeral` für einen nicht persistenten Speicher oder `persistent-claim` für ein Persistent Volume aus. Der Standardwert ist `persistent-claim`.

.spec.queueManager.storage.recoveryLogs

Details zu Persistent Volumes, die in die MQ-Wiederherstellungsprotokolle geschrieben werden. Erforderlich bei Verwendung eines Multi-Instanz-Warteschlangenmanagers.

Wird angezeigt in:

- „.spec.queueManager.storage” auf Seite 142

Feld	Beschreibung
`class` Zeichenfolge	Speicherklasse, die für dieses Volume verwendet werden soll. Nur gültig, wenn `type` auf `persistent-claim` gesetzt ist. Wenn `type of availability` (Typ der Verfügbarkeit) auf `SingleInstance` (Einzelinstanz) oder `Native-HA` (Native Hochverfügbarkeit) gesetzt ist, kann die Speicherklasse den Typ 'ReadWriteOnce' oder 'ReadWriteMany' haben. Ist `type of availability` dagegen `MultiInstance`, muss der Speicherklassentyp 'ReadWriteMany' sein.
`deleteClaim` boolesch	Gibt an, ob dieses Volume beim Löschen des Warteschlangenmanagers gelöscht werden soll.
`enabled` boolesch	Gibt an, ob dieses Volume als separates Volume aktiviert oder auf dem `queueManager`-Standardvolume eingerichtet werden soll. Der Standardwert ist 'false'.
`size` Zeichenfolge	An Kubernetes zu übergebende Größe des Persistent Volumes, einschließlich SI-Einheiten. Nur gültig, wenn `type` auf `persistent-claim` gesetzt ist. Beispiel: `2Gi`. Der Standardwert ist `2Gi`.
`sizeLimit` Zeichenfolge	Größenbegrenzung bei Verwendung eines `ephemeren` Volumes. Da Dateien weiterhin in ein temporäres Verzeichnis geschrieben werden, können Sie mit dieser Option die Größe begrenzen. Nur gültig, wenn `type` auf `ephemeral` gesetzt ist.
`type` Zeichenfolge	Typ des zu verwendenden Volumes. Wählen Sie `ephemeral` für einen nicht persistenten Speicher oder `persistent-claim` für ein Persistent Volume aus. Der Standardwert ist `persistent-claim`.

.spec.securityContext

Sicherheitseinstellungen, die dem Sicherheitskontext (securityContext) des Warteschlangenmanager-Pods hinzugefügt werden sollen.

Wird angezeigt in:

- „.spec” auf Seite 134

Feld	Beschreibung
`fsGroup` Ganzzahl	Eine spezielle zusätzliche Gruppe, die auf alle Container in einem Pod angewendet wird. Bei einigen Datenträgertypen kann der Kubelet das Eigentumsrecht des Datenträgers ändern, sodass er Eigentum des Pods wird: 1. Die FSGroup wird zur Eigner-GID. 2. Das Definitionsbit für Gruppen-ID (setgid) ist gesetzt (im Datenträger erstellte neue Dateien werden Eigentum von FSGroup). 3. Die Berechtigungsbits unterliegen einer OR-Operation mit rw-rw----. Wenn nicht gesetzt, ändert der Kubelet das Eigentumsrecht und die Berechtigungen von Datenträgern nicht.
`initVolumeAsRoot` boolesch	Wirkt sich auf den securityContext aus, der vom Container verwendet wird, der das Persistent Volume initialisiert. Setzen Sie diese Einstellung auf `true`, wenn der verwendete Speicheranbieter verlangt, dass Sie der Rootbenutzer sein müssen, um auf neu bereitgestellte Volumes zugreifen zu können. Der Wert `true` wirkt sich darauf aus, welches SCC-Objekt (Security Context Constraints) Sie verwenden können, und der Warteschlangenmanager wird möglicherweise nicht gestartet, wenn Sie nicht zur Verwendung eines SCC, das den Rootbenutzer zulässt, berechtigt sind. Der Standardwert ist 'false'. Weitere Informationen siehe https://docs.openshift.com/container-platform/latest/authentication/managing-security-context-constraints.html.

Feld	Beschreibung
supplementalGroups Array	Eine Liste der Gruppen, die auf den ersten Prozess angewendet werden, der in jedem Container ausgeführt wird, zusätzlich zur primären GID des Containers. Wenn nicht angegeben, werden zu keinem Container Gruppen hinzugefügt.

.spec.template

Erweiterte Erstellung anhand einer Vorlage für Kubernetes-Ressourcen. Mit der Vorlage können Benutzer durch Überschreibungen ändern, wie IBM MQ die zugrunde liegenden Kubernetes-Ressourcen, z. B. StatefulSet, Pods und Services, generiert. Dies ist nur für fortgeschrittene Benutzer, da bei falscher Verwendung die Möglichkeit besteht, dass der normale Betrieb von MQ unterbrochen wird. Alle an anderen Stellen in der QueueManager-Ressource angegebenen Werte werden durch Einstellungen in der Vorlage überschrieben.

Wird angezeigt in:

- „.spec" auf Seite 134

Feld	Beschreibung
pod	Überschreibungen für die Vorlage, die für den Pod verwendet wird. Siehe https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.17/#podspec-v1-core.

.spec.tracing

Einstellungen für Tracing-Integration mit dem Cloud Pak for Integration-Dashboard 'Operations'.

Wird angezeigt in:

- „.spec" auf Seite 134

Feld	Beschreibung
agent TracingAgent	Einstellungen für den optionalen Tracing-Agenten können nur in Cloud Pak for Integration konfiguriert werden.
collector TracingCollector	Einstellungen für den optionalen Tracing-Collector können nur in Cloud Pak for Integration konfiguriert werden.
enabled boolesch	Gibt an, ob die Integration mit dem Cloud Pak for Integration-Dashboard 'Operations' aktiviert wird oder nicht (über Tracing). Der Standardwert ist 'false'.
namespace Zeichenfolge	Namensbereich, in dem das Cloud Pak for Integration-Dashboard 'Operations' installiert ist.

.spec.tracing.agent

Einstellungen für den optionalen Tracing-Agenten können nur in Cloud Pak for Integration konfiguriert werden.

Wird angezeigt in:

- „.spec.tracing" auf Seite 146

Feld	Beschreibung
image Zeichenfolge	Das Container-Image, das verwendet wird.
imagePullPolicy Zeichenfolge	Diese Einstellung steuert, wann der Kubelet versucht, das angegebene Image zu extrahieren. Der Standardwert ist IfNotPresent.

Feld	Beschreibung
livenessProbe TracingProbe	Einstellungen zur Steuerung des Livetests.
readinessProbe TracingProbe	Einstellungen zur Steuerung des Bereitschaftstests.

.spec.tracing.agent.livenessProbe

Einstellungen zur Steuerung des Livetests.

Wird angezeigt in:

- „.spec.tracing.agent" auf Seite 146

Feld	Beschreibung
failureThreshold Ganzzahl	Mindestanzahl aufeinanderfolgender Fehler für den Test, damit er als fehlgeschlagen betrachtet wird, nachdem er erfolgreich war. Der Standardwert ist 1.
initialDelaySeconds Ganzzahl	Anzahl der Sekunden nach dem Start des Containers, bevor Livetests eingeleitet werden. Der Standardwert ist 10 Sekunden. Weitere Informationen: https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle#container-probes.
periodSeconds Ganzzahl	Gibt an, wie oft (in Sekunden) der Test durchzuführen ist. Der Standardwert ist 10 Sekunden.
successThreshold Ganzzahl	Mindestanzahl aufeinanderfolgender Erfolge für den Test, damit er als erfolgreich betrachtet wird, nachdem er fehlgeschlagen ist. Der Standardwert ist 1.
timeoutSeconds Ganzzahl	Anzahl der Sekunden, nach denen der Test das Zeitlimit überschreitet. Der Standardwert ist 2 Sekunden. Weitere Informationen: https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle#container-probes.

.spec.tracing.agent.readinessProbe

Einstellungen zur Steuerung des Bereitschaftstests.

Wird angezeigt in:

- „.spec.tracing.agent" auf Seite 146

Feld	Beschreibung
failureThreshold Ganzzahl	Mindestanzahl aufeinanderfolgender Fehler für den Test, damit er als fehlgeschlagen betrachtet wird, nachdem er erfolgreich war. Der Standardwert ist 1.
initialDelaySeconds Ganzzahl	Anzahl der Sekunden nach dem Start des Containers, bevor Livetests eingeleitet werden. Der Standardwert ist 10 Sekunden. Weitere Informationen: https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle#container-probes.
periodSeconds Ganzzahl	Gibt an, wie oft (in Sekunden) der Test durchzuführen ist. Der Standardwert ist 10 Sekunden.
successThreshold Ganzzahl	Mindestanzahl aufeinanderfolgender Erfolge für den Test, damit er als erfolgreich betrachtet wird, nachdem er fehlgeschlagen ist. Der Standardwert ist 1.
timeoutSeconds Ganzzahl	Anzahl der Sekunden, nach denen der Test das Zeitlimit überschreitet. Der Standardwert ist 2 Sekunden. Weitere Informationen: https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle#container-probes.

.spec.tracing.collector

Einstellungen für den optionalen Tracing-Collector können nur in Cloud Pak for Integration konfiguriert werden.

Wird angezeigt in:

- „.spec.tracing" auf Seite 146

Feld	Beschreibung
`image` Zeichenfolge	Das Container-Image, das verwendet wird.
`imagePullPolicy` Zeichenfolge	Diese Einstellung steuert, wann der Kubelet versucht, das angegebene Image zu extrahieren. Der Standardwert ist `IfNotPresent`.
`livenessProbe` TracingProbe	Einstellungen zur Steuerung des Livetests.
`readinessProbe` TracingProbe	Einstellungen zur Steuerung des Bereitschaftstests.

.spec.tracing.collector.livenessProbe

Einstellungen zur Steuerung des Livetests.

Wird angezeigt in:

- „.spec.tracing.collector" auf Seite 148

Feld	Beschreibung
`failureThreshold` Ganzzahl	Mindestanzahl aufeinanderfolgender Fehler für den Test, damit er als fehlgeschlagen betrachtet wird, nachdem er erfolgreich war. Der Standardwert ist 1.
`initialDelaySeconds` Ganzzahl	Anzahl der Sekunden nach dem Start des Containers, bevor Livetests eingeleitet werden. Der Standardwert ist 10 Sekunden. Weitere Informationen: https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle#container-probes.
`periodSeconds` Ganzzahl	Gibt an, wie oft (in Sekunden) der Test durchzuführen ist. Der Standardwert ist 10 Sekunden.
`successThreshold` Ganzzahl	Mindestanzahl aufeinanderfolgender Erfolge für den Test, damit er als erfolgreich betrachtet wird, nachdem er fehlgeschlagen ist. Der Standardwert ist 1.
`timeoutSeconds` Ganzzahl	Anzahl der Sekunden, nach denen der Test das Zeitlimit überschreitet. Der Standardwert ist 2 Sekunden. Weitere Informationen: https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle#container-probes.

.spec.tracing.collector.readinessProbe

Einstellungen zur Steuerung des Bereitschaftstests.

Wird angezeigt in:

- „.spec.tracing.collector" auf Seite 148

Feld	Beschreibung
`failureThreshold` Ganzzahl	Mindestanzahl aufeinanderfolgender Fehler für den Test, damit er als fehlgeschlagen betrachtet wird, nachdem er erfolgreich war. Der Standardwert ist 1.

Feld	Beschreibung
`initialDelaySeconds` Ganzzahl	Anzahl der Sekunden nach dem Start des Containers, bevor Livetests eingeleitet werden. Der Standardwert ist 10 Sekunden. Weitere Informationen: https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle#container-probes.
`periodSeconds` Ganzzahl	Gibt an, wie oft (in Sekunden) der Test durchzuführen ist. Der Standardwert ist 10 Sekunden.
`successThreshold` Ganzzahl	Mindestanzahl aufeinanderfolgender Erfolge für den Test, damit er als erfolgreich betrachtet wird, nachdem er fehlgeschlagen ist. Der Standardwert ist 1.
`timeoutSeconds` Ganzzahl	Anzahl der Sekunden, nach denen der Test das Zeitlimit überschreitet. Der Standardwert ist 2 Sekunden. Weitere Informationen: https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle#container-probes.

.spec.web

Einstellungen für den MQ-Web-Server.

Wird angezeigt in:

- „.spec" auf Seite 134

Feld	Beschreibung
`enabled` boolesch	Gibt an, ob der Web-Server aktiviert werden soll oder nicht. Der Standardwert ist 'false'.

.status

Der Ist-Status des QueueManager.

Wird angezeigt in:

- „QueueManager" auf Seite 133

Feld	Beschreibung
`adminUiUrl` Zeichenfolge	URL für die Verwaltungsbenutzerschnittstelle.
`availability` Availability	Verfügbarkeitsstatus für den Warteschlangenmanager.
`conditions` QueueManagerStatusCondition Array	'conditions' (Bedingungen) stellen die letzten verfügbaren Beobachtungen zum Status des Warteschlangenmanagers dar.
`endpoints` QueueManagerStatusEndpoint Array	Informationen zu den Endpunkten, die dieser Warteschlangenmanager zugänglich macht, z. B. API- oder UI-Endpunkte.
`name` Zeichenfolge	Der Name des Warteschlangenmanagers.
`phase` Zeichenfolge	Phase des Status des Warteschlangenmanagers.
`versions` QueueManagerStatusVersion	Verwendete MQ-Version und weitere Versionen, die aus der IBM Entitled Registry verfügbar sind.

.status.availability

Verfügbarkeitsstatus für den Warteschlangenmanager.

Wird angezeigt in:

- „.status" auf Seite 149

Feld	Beschreibung
`initialQuorumEstablished` boolesch	Gibt an, ob für NativeHA ein Ausgangsquorum festgelegt wurde oder nicht.

.status.conditions

QueueManagerStatusCondition definiert die Bedingungen des Warteschlangenmanagers.

Wird angezeigt in:

- „.status" auf Seite 149

Feld	Beschreibung
`lastTransitionTime` Zeichenfolge	Zeitpunkt des letzten Übergangs der Bedingung von einem Status in einen anderen.
`message` Zeichenfolge	Lesbare Nachricht mit Anzeige von Details zum letzten Übergang.
`reason` Zeichenfolge	Ursache für den letzten Übergang in diesen Status.
`status` Zeichenfolge	Status der Bedingung.
`type` Zeichenfolge	Typ der Bedingung.

.status.endpoints

QueueManagerStatusEndpoint definiert die Endpunkte für den QueueManager.

Wird angezeigt in:

- „.status" auf Seite 149

Feld	Beschreibung
`name` Zeichenfolge	Name des Endpunkts.
`type` Zeichenfolge	Der Typ des Endpunkts, z. B. 'UI' für einen Benutzerschnittstellenendpunkt, 'API' für einen API-Endpunkt, 'OpenAPI' für API-Dokumentation.
`uri` Zeichenfolge	URI für den Endpunkt.

.status.versions

Verwendete MQ-Version und weitere Versionen, die aus der IBM Entitled Registry verfügbar sind.

Wird angezeigt in:

- „.status" auf Seite 149

Feld	Beschreibung
`available` QueueManagerStatusVersionAvailable	Weitere MQ-Versionen, die aus der IBM Entitled Registry verfügbar sind.
`reconciled` Zeichenfolge	Die tatsächlich gerade verwendete Version von IBM MQ. Wird ein benutzerdefiniertes Image angegeben, stimmt dieses möglicherweise nicht mit der tatsächlich verwendeten MQ-Version überein.

.status.versions.available

Weitere MQ-Versionen, die aus der IBM Entitled Registry verfügbar sind.

Wird angezeigt in:

- „.status.versions” auf Seite 150

Feld	Beschreibung
`channels` Array	Kanäle, die für eine automatische Aktualisierung der MQ-Version verfügbar sind.
`versions` Versions Array	Bestimmte Versionen von MQ, die verfügbar sind.

.status.versions.available.versions

QueueManagerStatusVersion definiert eine Version von MQ.

Wird angezeigt in:

- „.status.versions.available” auf Seite 150

Feld	Beschreibung
`name` Zeichenfolge	`Name` für diese Version von QueueManager. Hierbei handelt es sich um gültige Werte für das Feld `spec.version`.

OpenShift CP4I *Statusbedingungen der Ressource 'QueueManager' (mq.ibm.com/v1beta1)*

Die Felder **`status.conditions`** aktualisieren sich entsprechend der Statusbedingung der Ressource `QueueManager`. Bedingungen beschreiben im Allgemeinen abnorme Situationen. Für einen Warteschlangenmanager in einem intakten, funktionsbereiten Zustand werden keine **`Error`**- oder **`Pending`**-Bedingungen ausgegeben. Für einen solchen Warteschlangenmanager können jedoch **`Warning`**-Bedingungen vorliegen, die lediglich empfehlenden Charakter haben.

Die Unterstützung für Bedingungen wurde in IBM MQ Operator 1.2 eingeführt.

Die folgenden Bedingungen sind für eine `QueueManager`-Ressource definiert:

Tabelle 1. Statusbedingungen für Warteschlangenmanager

Komponente	**Bedingungstyp**	**Ursachencode**	**Nachrichtenwarnung**
QueueManager[7]	Anstehend	Creating (Wird erstellt)	Der MQ-Warteschlangenmanager wird bereitgestellt.
	Anstehend	OidcPending	Der MQ-Warteschlangenmanager wartet auf die OIDC-Clientregistrierung.
	Fehler	Fehlgeschlagen	Der MQ-Warteschlangenmanager konnte nicht bereitgestellt werden.
	Warnung	UnsupportedVersion	[8]Ein Operand wurde von einem Operator installiert, der von OCP-Version *<ocp_version>* nicht unterstützt wird. Dieser Operand wird nicht unterstützt.
	Warnung	EUSSupport (EUS-Unterstützung)	[9]Ein EUS-Operand der Version *<mq_version>* wurde installiert, der jedoch von einem Operator verwaltet wird, der keinen Anspruch auf erweiterte Unterstützungsdauer hat. Dieser Operand ist nicht für die erweiterte Unterstützungsdauer qualifiziert.
	Warnung	EUSSupport (EUS-Unterstützung)	[10]Ein EUS-Operand der Version *<mq_version>* wurde installiert, jedoch hat OCP-Version 4*<ocp_version>* keinen Anspruch auf erweiterte Unterstützungsdauer. Dieser Operand ist nicht für die erweiterte Unterstützungsdauer qualifiziert.
	Warnung	EUSSupport (EUS-Unterstützung)	[11]Ein EUS-Operand der Version *<mq_version>* wurde installiert, jedoch hat OCP-Version *<ocp_version>* keinen Anspruch auf erweiterte Unterstützungsdauer. Dieser Operand wird wie ein regulärer CD-Release unterstützt.

[7] Mit den Bedingungen `Creating` (Wird erstellt) und `Failed` (Fehlgeschlagen) wird der Gesamtfortschritt der Bereitstellung des Warteschlangenmanagers überwacht. Wenn Sie eine IBM Cloud Pak for Integration-Lizenz verwenden und die MQ-Webkonsole aktiviert ist, protokolliert die Bedingung OidcPending den Status des Warteschlangenmanagers, während die OIDC-Clientregistrierung bei IAM ausgeführt wird.

Tabelle 1. Statusbedingungen für Warteschlangenmanager (Forts.)			
Komponente	**Bedingungstyp**	**Ursachencode**	**Nachrichtenwarnung**
Pod[12]	Anstehend	PodPending	Der Pod für den MQ-Warteschlangenmanager wird bereitgestellt.
	Fehler	PodFailed	Der Pod für den MQ-Warteschlangenmanager wird bereitgestellt.
Speicher[13]	Anstehend	StoragePending	Speicher für MQ-Warteschlangenmanager wird bereitgestellt.
	Warnung	StorageEphemeral	Für einen MQ-Warteschlangenmanager in einer Produktionsumgebung wird ephemerer Speicher verwendet.
	Fehler	StorageFailed	Der Speicher für den MQ-Warteschlangenmanager konnte nicht bereitgestellt werden.

Multi Eigenen IBM MQ-Container und Bereitstellungscode erstellen

Entwickeln Sie einen selbst erstellten Container. Dies ist die flexibelste Containerlösung, Sie benötigen jedoch umfassende Kenntnisse für die Konfiguration von Containern und der resultierende Container sollte sich in Ihrem "Eigentum" befinden.

Vorbereitende Schritte

Vor dem Entwickeln eines eigenen Containers sollten Sie überlegen, ob Sie stattdessen einen der vordefinierten Container verwenden können, die von IBM bereitgestellt werden. Siehe IBM MQ in Containern

8 Operator 1.4.0 und höher
9 Operator 1.4.0 und höher
10 Operator 1.4.0 und höher
11 Nur Operator 1.3.0
12 Pod-Bedingungen überwachen den Status von Pods während der Bereitstellung eines Warteschlangenmanagers. Bei Ausgabe der Bedingung `PodFailed` (Pod fehlgeschlagen) lautet auch die Bedingung für den Warteschlangenmanager insgesamt `Failed` (Fehlgeschlagen).
13 Speicherbedingungen überwachen den Fortschritt (Bedingung `StoragePending`) von Anforderungen für die Erstellung von Datenträgern für persistenten Speicher und sie melden Bindungs- und andere Fehler zurück. Wenn während der Speicherbereitstellung ein Fehler auftritt, wird die Bedingung `StorageFailed` (Speicher fehlgeschlagen) zur Liste der Bedingungen hinzugefügt. In diesem Fall lautet auch die Bedingung für den Warteschlangenmanager insgesamt `Failed` (Fehlgeschlagen).

Informationen zu diesem Vorgang

Wenn Sie IBM MQ als Container-Image packen, können Änderungen an Ihrer Anwendung für Test- und Staging-Systeme ohne großen Aufwand implementiert werden. Dies kann bei einem Continuous Delivery in Ihrem Unternehmen einen großen Vorteil bedeuten.

Prozedur

- „Eigenes Image des IBM MQ-Warteschlangenmanagers mithilfe eines Containers planen” auf Seite 154
- „Beispielcontainer-Image für IBM MQ-Warteschlangenmanager erstellen” auf Seite 155
- „Lokale Bindungsanwendungen in separaten Containern ausführen” auf Seite 157

Zugehörige Konzepte
IBM MQ in Containern

Multi Eigenes Image des IBM MQ-Warteschlangenmanagers mithilfe eines Containers planen

Bei der Ausführung eines IBM MQ-Warteschlangenmanagers in einem Container sind mehrere Anforderungen zu berücksichtigen. Das Beispiel für das Container-Image bietet eine Möglichkeit, diesen Anforderungen gerecht zu werden. Wenn Sie jedoch ein eigenes Image verwenden möchten, müssen Sie berücksichtigen, wie diese Anforderungen erfüllt werden können.

Prozessüberwachung

Wenn Sie einen Container ausführen, führen Sie im Prinzip einen einzelnen Prozess (PID 1 innerhalb des Containers) aus, der später untergeordnete Prozesse generieren kann.

Wenn der Hauptprozess beendet wird, wird der Container von der Container-Laufzeit gestoppt. Für einen IBM MQ-Warteschlangenmanager müssen mehrere Prozesse im Hintergrund ausgeführt werden.

Aus diesem Grund müssen Sie sicherstellen, dass Ihr Hauptprozess aktiv bleibt, solange der Warteschlangenmanager aktiv ist. Es ist sinnvoll, zu überprüfen, ob der Warteschlangenmanager von diesem Prozess aus aktiv ist, z. B. durch Ausführen von Verwaltungsabfragen.

`/var/mqm` füllen

Container müssen mit `/var/mqm` als Datenträger konfiguriert werden.

Dabei ist das Verzeichnis des Datenträgers leer, wenn der Container zuerst gestartet wird. Dieses Verzeichnis wird in der Regel während der Installation gefüllt, bei Verwendung von eines Containers sind Installations- und Laufzeitumgebung jedoch separat.

Um dies zu beheben, können Sie beim Starten Ihres Containers den Befehl **`crtmqdir`** verwenden, um `/var/mqm` zu füllen, wenn er zum ersten Mal ausgeführt wird.

Containersicherheit

Um die Laufzeitsicherheitsanforderungen zu minimieren, werden die Beispielcontainer-Images mit der nicht komprimierbaren IBM MQ-Installation installiert. Dies stellt sicher, dass keine `setuid`-Bits (Definitionsbits für Benutzer-ID) gesetzt werden und der Container keine Berechtigungseskalation verwenden muss. Einige Containersysteme definieren, welche Benutzer-IDs Sie verwenden können, und die nicht komprimierbare Installation stellt keine Annahmen über verfügbare Betriebssystembenutzer an.

Multi Beispielcontainer-Image für IBM MQ-Warteschlangenmanager erstellen

Verwenden Sie diese Informationen, um ein Beispielimage für einen Container für die Ausführung eines IBM MQ-Warteschlangenmanagers in einem Container zu erstellen.

Informationen zu diesem Vorgang

Zunächst erstellen Sie ein Basisimage, das ein Red Hat Universal Base Image-Dateisystem und eine bereinigende Installation von IBM MQ enthält.

Dann erstellen Sie eine weitere Container-Imageebene auf dieser Basis, die einige IBM MQ-Konfigurationen hinzufügt, um eine grundlegende Sicherheit für Benutzer-IDs und Kennwörter zu ermöglichen.

Abschließend führen Sie einen Container unter Verwendung dieses Image als Dateisystem aus, wobei die Inhalte von `/var/mqm` von einem containerspezifischen Datenträger auf dem Hostdateisystem bereitgestellt werden.

Prozedur

- Weitere Informationen zum Erstellen eines Beispiels für ein Containerimage für die Ausführung eines IBM MQ-Warteschlangenmanagers in einem Container finden Sie in den folgenden Unterabschnitten:
  - „Beispielbasisimage eines IBM MQ-Warteschlangenmanagers erstellen” auf Seite 155
  - „Beispiel für ein konfiguriertes IBM MQ-Warteschlangenmanagerimage erstellen” auf Seite 155

Multi Beispielbasisimage eines IBM MQ-Warteschlangenmanagers erstellen

Damit IBM MQ in Ihrem eigenen Container-Image verwendet werden kann, müssen Sie zunächst ein Basisimage mit einer bereinigten IBM MQ-Installation erstellen. In den folgenden Schritten wird gezeigt, wie ein Beispiel für ein Basisimage erstellt wird. Dabei wird der Code verwendet, der auf GitHub gehostet wird.

Prozedur

- Verwenden Sie die im GitHub-Repository zu mq-container bereitgestellten Make-Dateien, um Ihr Container-Image für die Produktion zu erstellen.

  Folgen Sie hierzu den Anweisungen auf GitHub im Abschnitt Container-Image erstellen. Wenn Sie den sicheren Zugriff mit Red Hat OpenShift Container Platform "restricted" Security Context Constraint (SCC) konfigurieren möchten, müssen Sie das Paket 'No-Install' IBM MQ verwenden.

Ergebnisse

Sie verfügen nun über ein Container-Image mit installiertem IBM MQ.

Sie können jetzt ein Beispiel für ein IBM MQ-Warteschlangenmanager-Image erstellen.

Multi Beispiel für ein konfiguriertes IBM MQ-Warteschlangenmanagerimage erstellen

Nachdem Sie Ihr generisches Basisimage für den IBM MQ-Container erstellt haben, müssen Sie Ihre eigene Konfiguration anwenden, um einen sicheren Zugriff zu ermöglichen. Hierzu erstellen Sie eine eigene Container-Imageebene, wobei Sie das generische Image als übergeordnetes Element verwenden.

Vorbereitende Schritte

V 9.2.0 Bei dieser Task wird davon ausgegangen, dass Sie bei der Erstellung des Beispielimage für den IBM MQ-Basiswarteschlangenmanager das Paket "No-Install" IBM MQ verwendet haben. Andernfalls können Sie keinen sicheren Zugriff mit der Sicherheitskontexteinschränkung (SCC) Red Hat OpenShift Container Platform "restricted" konfigurieren. Das standardmäßig verwendete SCC "restricted" verwendet randomisierte Benutzer-IDs und verhindert durch Benutzerwechsel die Eskalation von Berechtigungen. Voraussetzung für das herkömmliche RPM-basierte Installationsprogramm von IBM MQ sind ein `mqm`-Benutzer und eine mqm-Gruppe. Für ausführbare Programme verwendet es zudem `setuid`-Bits. Wenn Sie in IBM MQ 9.2das Paket "No-Install" IBM MQ verwenden, gibt es keinen `mqm` -Benutzer mehr und keine `mqm` -Gruppe.

Vorgehensweise

1. Erstellen Sie ein neues Verzeichnis und fügen Sie eine Datei mit der Bezeichnung `config.mqsc` mit den folgenden Inhalten hinzu:

```
DEFINE QLOCAL(EXAMPLE.QUEUE.1) REPLACE
```

   Beachten Sie, dass im vorigen Beispiel eine einfache Benutzer-ID und Kennwortauthentifizierung verwendet wird. Sie können allerdings auch die für Ihr Unternehmen erforderliche Sicherheitskonfiguration anwenden.

2. Erstellen Sie eine Datei mit der Bezeichnung `Dockerfile` mit den folgenden Inhalten:

```
FROM mq
COPY config.mqsc /etc/mqm/
```

3. Erstellen Sie mit folgendem Befehl ein angepasstes Container-Image:

```
docker build -t mymq .
```

   Dabei steht "`.`" für das Verzeichnis, das die beiden gerade erstellten Dateien enthält.

   Docker erstellt anschließend einen temporären Container mithilfe des Images und führt die verbleibenden Befehle aus.

   Anmerkung: Unter Red Hat Enterprise Linux (RHEL) verwenden Sie den Befehl **`docker`** (RHEL V7) oder **`podman`** (RHEL V7 oder RHEL V8). Unter Linux müssen Sie **`docker`**-Befehle mit **sudo** am Anfang des Befehls ausführen, um zusätzliche Berechtigungen zu erhalten.

4. Führen Sie das neue angepasste Image aus, um einen neuen Container mit dem soeben erstellten Plattenimage zu erstellen.

   In Ihrer neuen Imageebene wurde kein bestimmter Befehl für die Ausführung angegeben, so dass er von dem übergeordneten Image übernommen wurde. Der Eingangspunkt des übergeordneten Elements (der Code ist auf GitHub verfügbar):

   - Erstellen einen Warteschlangenmanager
   - Startet den Warteschlangenmanager
   - Erstellt einen Standardlistener
   - Anschließend werden alle MQSC-Befehle aus `/etc/mqm/config.mqsc.` ausgeführt.

   Geben Sie die folgenden Befehle aus, um Ihr neu angepasstes Image auszuführen:

```
docker run \
  --env LICENSE=accept \
  --env MQ_QMGR_NAME=QM1 \
  --volume /var/example:/var/mqm \
  --publish 1414:1414 \
  --detach \
  mymq
```

   Dabei gilt Folgendes:

Erster Parameter env
Übergibt eine Umgebungsvariable an den Container, der bestätigt, dass die Lizenz für IBM IBM WebSphere MQ akzeptiert wird. Sie können auch die Variable LICENSE für die Anzeige der Lizenz festlegen.

Weitere Informationen zu IBM MQ -Lizenzen finden Sie unter IBM MQ -Lizenzinformationen .

Zweiter Parameter env
Legt den Namen des Warteschlangenmanagers fest, den Sie verwenden.

Parameter 'Volume'
Dem Container wird mitgeteilt, dass alle Informationen, die von MQ in `/var/mqm` geschrieben werden, tatsächlich auf dem Host in `/var/example` geschrieben werden sollten.

Mit dieser Option können Sie den Container später auf einfache Weise löschen und trotzdem alle persistenten Daten beibehalten. Sie vereinfacht auch die Anzeige von Protokolldateien.

Parameter 'Publish'
Ports im Hostsystem werden Ports im Container zugeordnet. Der Container wird standardmäßig mit seiner eigenen internen IP-Adresse ausgeführt, d. h. Sie müssen alle Ports, die Sie zugänglich machen möchten, gezielt zuordnen.

In diesem Beispiel muss Port 1414 auf dem Host Port 1414 im Container zugeordnet werden.

Parameter 'Detach'
Führt den Container im Hintergrund aus.

Ergebnisse

Sie haben ein konfiguriertes Container-Image erstellt und können aktive Container mit dem Befehl **docker ps** anzeigen. Sie können die IBM MQ-Prozesse, die in Ihrem Container ausgeführt werden, mit dem Befehl **docker top** anzeigen.

Achtung:

Sie können die Protokolle eines Containers mit dem Befehl **docker logs ${CONTAINER_ID}** anzeigen.

Nächste Schritte

- Wenn Ihr Container bei Ausführung des Befehls **docker ps** nicht angezeigt wird, ist der Container möglicherweise fehlgeschlagen. Sie können fehlgeschlagene Container mit dem Befehl **docker ps -a** anzeigen.
- Bei Ausführung des Befehls **docker ps -a** wird die Container-ID angezeigt. Diese ID wurde auch gedruckt, als Sie den Befehl **docker run** ausgegeben haben.
- Sie können die Protokolle eines Containers mit dem Befehl **docker logs ${CONTAINER_ID}** anzeigen.

Multi Lokale Bindungsanwendungen in separaten Containern ausführen

Bei der gemeinsame Nutzung von Prozessnamensbereichen zwischen Containern in Docker können Sie Anwendungen, für die eine lokale Bindungsverbindung zu IBM MQ erforderlich ist, in separaten Containern aus dem IBM MQ-Warteschlangenmanager heraus ausführen.

Informationen zu diesem Vorgang

Diese Funktion wird in IBM MQ 9.0.3-Warteschlangenmanagern und späteren Warteschlangenmanagern unterstützt.

Sie müssen die folgenden Einschränkungen beachten:

- Sie müssen den PID-Namensbereich der Container mit dem Argument `--pid` gemeinsam nutzen.
- Sie müssen den IPC-Namensbereich der Container mit dem Argument `--ipc` gemeinsam nutzen.

- Sie müssen eine der folgenden Schritte ausführen:
  1. Nutzen Sie den UTS-Namensbereich der Container gemeinsam mit dem Host, indem Sie das Argument `--uts` verwenden, oder
  2. Stellen Sie sicher, dass die Container denselben Hostnamen verwenden, indem Sie das Argument `-h` oder `--hostname` verwenden.
- Sie müssen das IBM MQ-Datenverzeichnis in einem Datenträger anhängen, der für alle Container unter dem Verzeichnis `/var/mqm` verfügbar ist.

Sie können diese Funktion ausprobieren, indem Sie die folgenden Schritte auf einem Linux-System ausführen, auf dem Docker bereits installiert ist.

Im folgenden Beispiel wird das IBM MQ-Beispielimage für Container verwendet. Einzelheiten zu diesem Image finden Sie unter Github.

Vorgehensweise

1. Erstellen Sie mit folgendem Befehl ein temporäres Verzeichnis, das als Datenträger verwendet werden soll:

```
mkdir /tmp/dockerVolume
```

2. Erstellen Sie mit folgendem Befehl einen Warteschlangenmanager (QM1) in einem Container mit der mit der Bezeichnung `sharedNamespace`:

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

3. Starten Sie einen zweiten Container mit dem Namen `secondaryContainer`, der auf `ibmcom/mq` basiert, aber erstellen Sie keinen Warteschlangenmanager, indem Sie folgenden Befehl ausgeben:

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

4. Führen Sie den Befehl **dspmq** im zweiten Container aus, um den Status der beiden Warteschlangenmanager anzuzeigen:

```
docker exec secondaryContainer dspmq
```

5. Führen Sie den folgenden Befehl aus, um die MQSC-Befehle für den Warteschlangenmanager zu verarbeiten, der im anderen Container aktiv ist:

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

Ergebnisse

Sie verfügen jetzt über lokale Anwendungen, die in separaten Containern ausgeführt werden, und können jetzt Befehle wie **dspmq**, **amqsput**, **amqsget** und **runmqsc** erfolgreich als lokale Bindungen zum QM1-Warteschlangenmanager aus dem sekundären Container ausführen.

Wenn nicht das erwartete Ergebnis angezeigt wird, finden Sie unter „Fehlerbehebung für Ihre Namensbereichsanwendungen” auf Seite 158 weitere Informationen.

Multi Fehlerbehebung für Ihre Namensbereichsanwendungen

Wenn Sie gemeinsam genutzte Namensbereiche verwenden, müssen Sie sicherstellen, dass Sie alle Namensbereiche (IPC, PID und UTS/Hostname) und angehängte Datenträger gemeinsam nutzen, da Ihre Anwendungen ansonsten nicht funktionieren.

Im Abschnitt „Lokale Bindungsanwendungen in separaten Containern ausführen” auf Seite 157 finden Sie eine Liste der Einschränkungen, die Sie beachten müssen.

Wenn Ihre Anwendung nicht alle aufgeführten Einschränkungen erfüllt, kann es zu Problemen kommen, bei denen der Container startet, aber die erwartete Funktion nicht ausgeführt wird.

Die folgende Liste enthält einige häufige Ursachen und das Verhalten, das Sie wahrscheinlich sehen, wenn Sie eine der Einschränkungen vergessen haben.

- Wenn Sie vergessen haben, entweder den Namensbereich (UTS/PID/IPC) oder den Hostnamen der Container gemeinsam zu nutzen, und den Datenträger anhängen, dann kann Ihr Container den Warteschlangenmanager zwar erkennen, aber nicht mit ihm interagieren.
  - Für **dspmq**-Befehle wird Folgendes angezeigt:

```
docker exec container dspmq

QMNAME(QM1)                       STATUS(Status not available)
```

  - Für Befehle des Typs **runmqsc** oder andere Befehle, die versuchen, eine Verbindung zum Warteschlangenmanager herzustellen, wird wahrscheinlich die Fehlernachricht AMQ8146 angezeigt:

```
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
```

- Wenn Sie alle erforderlichen Namensbereiche gemeinsam nutzen, aber keinen gemeinsam genutzten Datenträger an das Verzeichnis `/var/mqm` anhängen, und wenn Sie einen gültigen IBM MQ-Datenpfad haben, empfangen Ihre Befehle auch AMQ8146-Fehlernachrichten.

  dspmq ist jedoch nicht in der Lage, Ihren Warteschlangenmanager zu sehen, und gibt stattdessen eine leere Antwort zurück:

```
docker exec container dspmq
```

- Wenn Sie alle erforderlichen Namensbereiche gemeinsam nutzen, aber keinen gemeinsam genutzten Datenträger an das Verzeichnis `/var/mqm` anhängen, und wenn Sie keinen gültigen IBM MQ-Datenpfad (oder gar keinen IBM MQ-Datenpfad) haben, werden verschiedene Fehler angezeigt, da der Datenpfad eine Schlüsselkomponente einer IBM MQ-Installation ist. Ohne den Datenpfad kann IBM MQ nicht ausgeführt werden.

  Wenn Sie einen der folgenden Befehle ausführen und Antworten ähnlich den Antworten finden, die in diesen Beispielen aufgeführt sind, sollten Sie überprüfen, ob Sie das Verzeichnis angehängt oder ein IBM MQ-Datenverzeichnis erstellt haben:

```
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 Native HA-Gruppe erstellen, wenn eigene Container erstellt werden

Sie müssen drei Warteschlangenmanager erstellen, konfigurieren und starten, um die native HA-Gruppe zu erstellen.

Informationen zu diesem Vorgang

Die empfohlene Methode zum Erstellen einer nativen HA-Lösung ist die Verwendung des Operators IBM MQ (siehe Native HA). Wenn Sie eigene Container erstellen, können Sie alternativ die folgenden Anweisungen befolgen.

Um eine native HA-Gruppe zu erstellen, erstellen Sie drei Warteschlangenmanager auf drei Knoten, deren Protokolltyp auf `log replication`gesetzt ist. Anschließend bearbeiten Sie die Datei `qm.ini` für jeden Warteschlangenmanager, um die Verbindungsdetails für jeden der drei Knoten hinzuzufügen, sodass sie Protokolldaten miteinander replizieren können.

Sie müssen dann alle drei Warteschlangenmanager starten, damit sie überprüfen können, ob alle drei Instanzen miteinander kommunizieren können, und bestimmen, welche von ihnen die aktive Instanz und welche die Replikate sind.

Vorgehensweise

1. Erstellen Sie auf jedem der drei Knoten einen Warteschlangenmanager, geben Sie den Protokolltyp "Protokollreplik" an und geben Sie einen eindeutigen Namen für jede Protokollinstanz an. Jeder Warteschlangenmanager hat denselben Namen:

   ```
   crtmqm -lr instance_name qmname
   ```

   Beispiel:

   ```
   node 1> crtmqm -lr qm1_inst1 qm1
   node 2> crtmqm -lr qm1_inst2 qm1
   node 3> crtmqm -lr qm1_inst3 qm1
   ```

2. Bei erfolgreicher Erstellung jedes Warteschlangenmanagers wird der Konfigurationsdatei des Warteschlangenmanagers `qm.ini`eine zusätzliche Zeilengruppe namens `NativeHALocalInstance` hinzugefügt. Der Zeilengruppe wird ein Attribut `Name` hinzugefügt, das den angegebenen Instanznamen angibt.

   Sie können der Zeilengruppe `NativeHALocalInstance` in der Datei `qm.ini` optional die folgenden Attribute hinzufügen:

   `KeyRepository`
   Die Position des Schlüsselrepositorys, das das digitale Zertifikat enthält, das für den Schutz des Protokollreplikationsverkehrs verwendet werden soll. Die Position wird im Stammformat angegeben, d. h., sie enthält den vollständigen Pfad und Dateinamen ohne Erweiterung. Wenn das Zeilengruppenattribut `KeyRepository` nicht angegeben wird, werden Protokollreplikationsdaten zwischen Instanzen in Klartext ausgetauscht.

   `CertificateLabel`
   Die Zertifikatsbezeichnung, die das digitale Zertifikat angibt, das für den Schutz des Protokollreplikationsverkehrs verwendet werden soll. Wenn `KeyRepository` angegeben wird, aber `CertificateLabel` weggelassen wird, wird der Standardwert `ibmwebspheremq`*`queue_manager`* verwendet.

CipherSpec
Die MQ-CipherSpec, die zum Schutz des Protokollreplikationsverkehrs verwendet werden soll. Wird dieses Zeilengruppenattribut angegeben, muss auch `KeyRepository` angegeben werden. Wenn `KeyRepository` angegeben wird, aber `CipherSpec` weggelassen wird, wird der Standardwert ANY verwendet.

LocalAddress
Die lokale Netzschnittstellenadresse, die den Protokollreplikationsverkehr akzeptiert. Wenn dieses Zeilengruppenattribut angegeben wird, gibt es die lokale Netzschnittstelle und/oder den Port im Format "[addr] [(port)]" an. Die Netzadresse kann als Hostname, als IPv4 -Schreibweise mit Trennzeichen oder als IPv6 -Hexadezimalformat angegeben werden. Wenn dieses Attribut weggelassen wird, versucht der Warteschlangenmanager, eine Bindung zu allen Netzschnittstellen herzustellen. Er verwendet den Port, der in der Zeilengruppe `ReplicationAddress` in der Zeilengruppe `NativeHAInstances` angegeben ist und dem Namen der lokalen Instanz entspricht.

HeartbeatInterval
Das Heartbeatintervall legt fest, wie oft in Millisekunden eine aktive Instanz eines Warteschlangenmanagers mit Native HA ein Netzüberwachungssignal sendet. Der gültige Bereich für den Heartbeatintervallwert liegt zwischen 500 (0,5 Sekunden) und 60000 (1 Minute). Ein Wert außerhalb dieses Bereichs führt dazu, dass der Warteschlangenmanager nicht gestartet wird. Wird dieses Attribut nicht angegeben, wird der Standardwert 5000 (5 Sekunden) verwendet. Es muss für alle Instanzen dasselbe Heartbeatintervall festgelegt werden.

HeartbeatTimeout
Das Überwachungssignalzeitlimit legt fest, wie lange eine Replikatinstanz eines Warteschlangenmanagers mit Native HA wartet, bevor sie entscheidet, dass die aktive Instanz nicht mehr reagiert. Der gültige Bereich für den Wert dieses Zeitlimits liegt zwischen 500 (0,5 Sekunden) und 120000 (2 Minuten). Der Wert des Überwachungssignalzeitlimits muss größer-gleich dem Wert des Heartbeatintervalls sein.

Ein ungültiger Wert führt dazu, dass der Warteschlangenmanager nicht gestartet wird. Wird dieses Attribut nicht angegeben, wartet ein Replikat 2 x `HeartbeatInterval`, bevor es den Prozess startet, um eine neue aktive Instanz zu wählen. Es muss für alle Instanzen dasselbe Überwachungssignalzeitlimit festgelegt werden.

RetryInterval
Das Wiederholungsintervall legt fest, wie oft in Millisekunden ein Warteschlangenmanager mit Native HA eine fehlgeschlagene Replikationsverbindung wiederholen soll. Der gültige Bereich für das Wiederholungsintervall liegt zwischen 500 (0,5 Sekunden) und 120000 (2 Minuten). Wenn dieses Attribut weggelassen wird, wartet ein Replikat 2 x `HeartbeatInterval`, bevor es eine fehlgeschlagene Replikationsverbindung wiederholt.

3. Bearbeiten Sie die Datei `qm.ini` für jeden Warteschlangenmanager und fügen Sie Verbindungsdetails hinzu. Sie fügen drei `NativeHAInstance` -Zeilengruppen hinzu, eine für jede Warteschlangenmanagerinstanz in der nativen HA-Gruppe (einschließlich der lokalen Instanz). Fügen Sie die folgenden Attribute hinzu:

   Name
   Geben Sie den Instanznamen an, den Sie beim Erstellen der Warteschlangenmanagerinstanz verwendet haben.

   ReplicationAddress
   Geben Sie den Hostnamen, die IPv4 -Schreibweise mit Trennzeichen oder die Adresse im IPv6 -Hexadezimalformat der Instanz an. Sie können die Adresse als Hostnamen, als IPv4 -Schreibweise mit Trennzeichen oder als IPv6 -Adresse im Hexadezimalformat angeben. Die Replikationsadresse muss von jeder Instanz in der Gruppe auflösbar und weiterleitbar sein. Die für die Protokollreplikation zu verwendende Portnummer muss in eckigen Klammern angegeben werden. Beispiel:

   ```
   ReplicationAddress=host1.example.com(4444)
   ```

   Anmerkung: Die `NativeHAInstance` -Zeilengruppen sind auf allen Instanzen identisch und können mithilfe der automatischen Konfiguration (**`crtmqm -ii`**) bereitgestellt werden.

4. Starten Sie jede der drei Instanzen:

```
strmqm QMgrName
```

Wenn die Instanzen gestartet werden, kommunizieren sie, um zu überprüfen, ob alle drei Instanzen aktiv sind, und entscheiden, welche der drei Instanzen die aktive Instanz ist, während die beiden anderen Instanzen weiterhin als Replikate ausgeführt werden.

Beispiel

Das folgende Beispiel zeigt den Abschnitt einer Datei `qm.ini` , in dem die erforderlichen nativen HA-Details für eine der drei Instanzen angegeben werden:

```
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 Hinweise zur Durchführung einer eigenen schrittweisen Aktualisierung eines nativen HA-Warteschlangenmanagers

Jede Aktualisierung der IBM MQ-Version oder Pod-Spezifikation für einen nativen HA-Warteschlangenmanager erfordert eine rollierende Aktualisierung der Warteschlangenmanagerinstanzen. IBM MQ Operator verarbeitet dies automatisch für Sie, aber wenn Sie Ihren eigenen Implementierungscode erstellen, gibt es einige wichtige Aspekte.

Anmerkung: Das Helm-Beispieldiagramm enthält ein Shell-Script, um eine rollierende Aktualisierung auszuführen, aber das Script ist **nicht** für den Produktionseinsatz geeignet, da es die Hinweise in diesem Abschnitt nicht berücksichtigt.

In Kubernetes werden `StatefulSet`-Ressourcen verwendet, um geordnete Start und rollierende Aktualisierungen zu verwalten. Ein Teil der Startprozedur besteht darin, jeden Pod einzeln zu starten, zu warten, bis er bereit wird, und dann mit dem nächsten Pod fortzufahren. Dies funktioniert nicht für native HA, da alle Pods gestartet werden müssen, damit sie eine Führungswahl ausführen können. Daher muss das Feld `.spec.podManagementPolicy` in `StatefulSet` auf `Parallel` gesetzt werden. Dies bedeutet auch, dass alle Pods parallel aktualisiert werden, was besonders unerwünscht ist. Aus diesem Grund sollte `StatefulSet` auch die Aktualisierungsstrategie `OnDelete` verwenden.

Die Unfähigkeit, den `StatefulSet`-Rolling-Update-Code zu verwenden, führt zu einem Bedarf an angepasstem Rolling-Update-Code, der Folgendes berücksichtigen sollte:

- Allgemeine Prozedur für rollierende Aktualisierung
- Ausfallzeit durch Aktualisierung von Pods in der besten Reihenfolge minimieren
- Änderungen im Clusterstatus verarbeiten
- Fehler bearbeiten
- Handhabung von Timing-Problemen

Allgemeine Prozedur für rollierende Aktualisierung

Der Code für die rollierende Aktualisierung sollte warten, bis jede Instanz den Status `REPLICA` von `dspmq` anzeigt. Dies bedeutet, dass die Instanz eine gewisse Startstufe ausgeführt hat (z. B. ist der Container gestartet und MQ-Prozesse sind aktiv), aber sie hat noch nicht unbedingt mit den anderen Instanzen kommunizieren können. Beispiel: Pod A wird erneut gestartet und sobald er sich im Status `REPLICA` befindet, wird Pod B erneut gestartet. Sobald Pod B mit der neuen Konfiguration beginnt, sollte

er in der Lage sein, mit Pod A zu kommunizieren und Quorum zu bilden, und entweder A oder B wird zur neuen aktiven Instanz.

Als Teil davon ist es nützlich, eine Verzögerung zu haben, nachdem jeder Pod den Status REPLICAerreicht hat, damit er eine Verbindung zu seinen Peers herstellen und ein Quorum einrichten kann.

Ausfallzeit durch Aktualisierung von Pods in der besten Reihenfolge minimieren

Der Code für die rollierende Aktualisierung sollte Pods nacheinander löschen, beginnend mit Pods, die einen bekannten Fehlerstatus aufweisen, gefolgt von allen Pods, die nicht erfolgreich gestartet wurden. Der aktive Warteschlangenmanager-Pod sollte im Allgemeinen zuletzt aktualisiert werden.

Es ist auch wichtig, das Löschen von Pod anzuhalten, wenn die letzte Aktualisierung dazu geführt hat, dass ein Pod in einen bekannten Fehlerstatus versetzt wurde. Dies verhindert das Rollout einer defekten Aktualisierung über alle Pods hinweg. Dies kann beispielsweise auftreten, wenn der Pod so aktualisiert wird, dass er ein neues Container-Image verwendet, auf das nicht zugegriffen werden kann (oder das einen Schreibfehler enthält).

Änderungen im Clusterstatus verarbeiten

Der Code für die rollierende Aktualisierung muss entsprechend auf Echtzeitänderungen im Clusterstatus reagieren. Beispielsweise kann einer der Pods des Warteschlangenmanagers aufgrund eines Knotenneustarts oder aufgrund von Knotendruck entfernt werden. Möglicherweise wird ein bereinigter Pod nicht sofort neu geplant, wenn der Cluster ausgelastet ist. In diesem Fall müsste der Code für die rollierende Aktualisierung entsprechend warten, bevor andere Pods erneut gestartet werden.

Fehler bearbeiten

Der Code für die rollierende Aktualisierung muss robust gegenüber Fehlern sein, wenn die Kubernetes-API und andere unerwartete Clusterverhalten aufgerufen werden.

Darüber hinaus muss der Code für die rollierende Aktualisierung selbst tolerant gegenüber einem Neustart sein. Eine rollierende Aktualisierung kann eine lange Laufzeit haben und der Code muss möglicherweise erneut gestartet werden.

Handhabung von Timing-Problemen

Der Code für die rollierende Aktualisierung muss die Aktualisierungsrevisionen des Pods überprüfen, damit sichergestellt werden kann, dass der Pod erneut gestartet wurde. Dadurch werden Timing-Probleme vermieden, bei denen ein Pod möglicherweise angibt, dass er "gestartet" ist, aber tatsächlich noch nicht beendet wurde.

Zugehörige Konzepte

„Verwendung von IBM MQ in Containern auswählen” auf Seite 5
Es gibt mehrere Optionen für die Verwendung von IBM MQ in Containern: Sie können die IBM MQ Operator verwenden, die vorverpackte Containerimages verwendet, oder Sie können Ihre eigenen Images und Ihren eigenen Implementierungscode erstellen.

CP4I Status von nativen HA-Warteschlangenmanagern für angepasste Container anzeigen

Für angepasste Container können Sie den Status der nativen HA-Instanzen mit dem Befehl **dspmq** anzeigen.

Informationen zu diesem Vorgang

Sie können den Befehl **dspmq** verwenden, um den Betriebsstatus einer Warteschlangenmanagerinstanz auf einem Knoten anzuzeigen. Welche Informationen zurückgegeben werden, hängt davon ab, ob die

Instanz aktiv oder ein Replikat ist. Die von der aktiven Instanz gelieferten Informationen sind verbindlich, während Informationen von Replikatknoten möglicherweise nicht auf dem neuesten Stand sind.

Sie können folgende Aktionen ausführen:

- Anzeigen, ob die Warteschlangenmanagerinstanz auf dem aktuellen Knoten aktiv oder ein Replikat ist.
- Anzeigen des Native HA-Betriebsstatus der Instanz auf dem aktuellen Knoten.
- Anzeigen des Betriebsstatus aller drei Instanzen in einer Native HA-Konfiguration.

Der Status einer Native HA-Konfiguration wird in folgenden Statusfeldern gemeldet:

ROLE
Gibt die aktuelle Rolle der Instanz an. Die gültigen Werte sind `Active`, `Replica`und `Unknown`.

INSTANCE
Der Name, der für diese Instanz des Warteschlangenmanagers angegeben wurde, als sie mit der Option **`-lr`** des Befehls **`crtmqm`** erstellt wurde.

INSYNC
Gibt an, ob die Instanz bei Bedarf die Rolle der aktiven Instanz übernehmen kann.

QUORUM
Gibt den Quorumstatus im Format *Anzahl_synchrone_Instanzen*/*Anzahl_konfigurierte_Instanzen* an.

REPLADDR
Gibt die Replikationsadresse der Warteschlangenmanagerinstanz an.

CONNACTV
Gibt an, ob der Knoten mit der aktiven Instanz verbunden ist.

BACKLOG
Gibt die Anzahl KB an, um die die Instanz zurückliegt.

CONNINST
Gibt an, ob die benannte Instanz mit dieser Instanz verbunden ist.

ALTDATE
Gibt das Datum der letzten Aktualisierung dieser Informationen an (leer, wenn sie noch nie aktualisiert wurden).

ALTTIME
Gibt die Zeit der letzten Aktualisierung dieser Informationen an (leer, wenn sie noch nie aktualisiert wurden).

Prozedur

- Stellen Sie fest, ob eine Warteschlangenmanagerinstanz als aktive Instanz oder als Replikat ausgeführt wird:

  ```
  dspmq -o status -m QMgrName
  ```

  Eine aktive Instanz eines Warteschlangenmanagers mit dem Namen BOB würde folgenden Status melden:

  ```
  QMNAME(BOB)                 STATUS(Running)
  ```

  Eine Replikatinstanz eines Warteschlangenmanagers mit dem Namen BOB würde folgenden Status melden:

  ```
  QMNAME(BOB)                 STATUS(Replica)
  ```

  Eine inaktive Instanz würde folgenden Status melden:

  ```
  QMNAME(BOB)                 STATUS(Ended Immediately)
  ```

- So ermitteln Sie den nativen HA-Betriebsstatus der Instanz auf dem aktuellen Knoten:

```
dspmq -o nativeha -m QMgrName
```

Die aktive Instanz eines Warteschlangenmanagers mit dem Namen BOB könnte folgenden Status melden:

```
QMNAME(BOB)               ROLE(Active) INSTANCE(inst1) INSYNC(Yes) QUORUM(3/3)
```

Eine Replikatinstanz eines Warteschlangenmanagers mit dem Namen BOB könnte folgenden Status melden:

```
QMNAME(BOB)               ROLE(Replica) INSTANCE(inst2) INSYNC(Yes) QUORUM(2/3)
```

Eine inaktive Instanz eines Warteschlangenmanagers mit dem Namen BOB könnte folgenden Status melden:

```
QMNAME(BOB)               ROLE(Unknown) INSTANCE(inst3) INSYNC(no) QUORUM(0/3)
```

- Stellen Sie den Native HA-Betriebsstatus aller Instanzen in der Native HA-Konfiguration fest:

```
dspmq -o nativeha -x -m QMgrName
```

Wenn Sie diesen Befehl auf dem Knoten mit der aktiven Instanz des Warteschlangenmanagers BOB ausgeben, könnte der folgende Status gemeldet werden:

```
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)
```

Wenn Sie diesen Befehl auf einem Knoten mit einer Replikatinstanz des Warteschlangenmanagers BOB ausgeben, könnte der folgende Status gemeldet werden, der anzeigt, dass eins der Replikate im Rückstand ist:

```
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)
```

Wenn Sie diesen Befehl auf einem Knoten mit einer inaktiven Instanz des Warteschlangenmanagers BOB ausgeben, könnte der folgende Status gemeldet werden:

```
QMNAME(BOB)            ROLE(Unknown) INSTANCE(inst3) INSYNC(no) QUORUM(0/3)
 INSTANCE(inst1) ROLE(Unknown) REPLADDR(9.20.123.45) CONNACTV(Unknown) INSYNC(Unknown) BACK
LOG(Unknown) CONNINST(No) ALTDATE() ALTTIME()
 INSTANCE(inst2) ROLE(Unknown) REPLADDR(9.20.123.46) CONNACTV(Unknown) INSYNC(Unknown) BACK
LOG(Unknown) CONNINST(No) ALTDATE() ALTTIME()
 INSTANCE(inst3) ROLE(Unknown) REPLADDR(9.20.123.47) CONNACTV(No) INSYNC(Unknown) BACKLOG(Un
known) CONNINST(No) ALTDATE() ALTTIME()
```

Wenn Sie den Befehl ausgeben, während die Instanzen aushandeln, welche die aktive Instanz und und welche die Replikate sind, würden Sie folgenden Status empfangen:

```
QMNAME(BOB)              STATUS(Negotiating)
```

Zugehörige Verweise

dspmq

CP4I Beenden von nativen HA-Warteschlangenmanagern

Mit dem Befehl **endmqm** können Sie einen aktiven Warteschlangenmanager oder einen Replikatwarteschlangenmanager beenden, der Teil einer nativen HA-Gruppe ist.

Prozedur

- Informationen zum Beenden der aktiven Instanz eines Warteschlangenmanagers finden Sie unter Native HA-Warteschlangenmanager beenden im Konfigurationsabschnitt dieser Dokumentation.

Bemerkungen

Die vorliegenden Informationen wurden für Produkte und Services entwickelt, die auf dem deutschen Markt angeboten werden.

Möglicherweise bietet IBM die in dieser Dokumentation beschriebenen Produkte, Services oder Funktionen in anderen Ländern nicht an. Informationen über die gegenwärtig im jeweiligen Land verfügbaren Produkte und Services sind beim zuständigen IBM Ansprechpartner erhältlich. Hinweise auf IBM Lizenzprogramme oder andere IBM Produkte bedeuten nicht, dass nur Programme, Produkte oder Services von IBM verwendet werden können. Anstelle der IBM Produkte, Programme oder Services können auch andere, ihnen äquivalente Produkte, Programme oder Services verwendet werden, solange diese keine gewerblichen oder andere Schutzrechte der IBM verletzen. Die Verantwortung für den Betrieb von Fremdprodukten, Fremdprogrammen und Fremdservices liegt beim Kunden.

Für in diesem Handbuch beschriebene Erzeugnisse und Verfahren kann es IBM Patente oder Patentanmeldungen geben. Mit der Auslieferung dieser Dokumentation ist keine Lizenzierung dieser Patente verbunden. Lizenzanforderungen sind schriftlich an folgende Adresse zu richten (Anfragen an diese Adresse müssen auf Englisch formuliert werden):

IBM Europe
IBM Europe, Middle East and Africa
Tour Descartes
2, avenue Gambetta
92066 Paris La Défense
U.S.A.

Bei Lizenzanforderungen zu Double-Byte-Information (DBCS) wenden Sie sich bitte an die IBM Abteilung für geistiges Eigentum in Ihrem Land oder senden Sie Anfragen schriftlich an folgende Adresse:

Lizenzierung von geistigem Eigentum

IBM Japan, Ltd.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Trotz sorgfältiger Bearbeitung können technische Ungenauigkeiten oder Druckfehler in dieser Veröffentlichung nicht ausgeschlossen werden. Die Angaben in dieser Veröffentlichung werden in regelmäßigen Zeitabständen aktualisiert. Die Änderungen werden in Überarbeitungen oder in Technical News Letters (TNLs) bekanntgegeben. IBM kann jederzeit Verbesserungen und/oder Änderungen an den in dieser Veröffentlichung beschriebenen Produkten und/oder Programmen vornehmen.

Verweise in diesen Informationen auf Websites anderer Anbieter werden lediglich als Service für den Kunden bereitgestellt und stellen keinerlei Billigung des Inhalts dieser Websites dar. Das über diese Websites verfügbare Material ist nicht Bestandteil des Materials für dieses IBM Produkt.

Werden an IBM Informationen eingesandt, können diese beliebig verwendet werden, ohne dass eine Verpflichtung gegenüber dem Einsender entsteht.

Lizenznehmer des Programms, die Informationen zu diesem Produkt wünschen mit der Zielsetzung: (i) den Austausch von Informationen zwischen unabhängigen, erstellten Programmen und anderen Programmen (einschließlich des vorliegenden Programms) sowie (ii) die gemeinsame Nutzung der ausgetauschten Informationen zu ermöglichen, wenden sich an folgende Adresse:

IBM Europe, Middle East and Africa
Software Interoperability Coordinator, Department 49XA
3605 Highway 52 N
Rochester, MN 55901
U.S.A.

Die Bereitstellung dieser Informationen kann unter Umständen von bestimmten Bedingungen - in einigen Fällen auch von der Zahlung einer Gebühr - abhängig sein.

Die Lieferung des in diesen Informationen beschriebenen Lizenzprogramms sowie des zugehörigen Lizenzmaterials erfolgt auf der Basis der IBM Rahmenvereinbarung bzw. der Allgemeinen Geschäftsbedingungen von IBM, der IBM Internationalen Nutzungsbedingungen für Programmpakete oder einer äquivalenten Vereinbarung.

Die in diesem Dokument enthaltenen Leistungsdaten stammen aus einer kontrollierten Umgebung. Die Ergebnisse, die in anderen Betriebsumgebungen erzielt werden, können daher erheblich von den hier erzielten Ergebnissen abweichen. Einige Daten stammen möglicherweise von Systemen, deren Entwicklung noch nicht abgeschlossen ist. Eine Gewährleistung, dass diese Daten auch in allgemein verfügbaren Systemen erzielt werden, kann nicht gegeben werden. Darüber hinaus wurden einige Daten unter Umständen durch Extrapolation berechnet. Die tatsächlichen Ergebnisse können davon abweichen. Benutzer dieses Dokuments sollten die entsprechenden Daten in ihrer spezifischen Umgebung prüfen.

Alle Informationen zu Produkten anderer Anbieter stammen von den Anbietern der aufgeführten Produkte, deren veröffentlichten Ankündigungen oder anderen allgemein verfügbaren Quellen. IBM hat diese Produkte nicht getestet und kann daher keine Aussagen zu Leistung, Kompatibilität oder anderen Merkmalen machen. Fragen zu den Leistungsmerkmalen von Produkten anderer Anbieter sind an den jeweiligen Anbieter zu richten.

Aussagen über Pläne und Absichten von IBM unterliegen Änderungen oder können zurückgenommen werden und repräsentieren nur die Ziele von IBM.

Diese Veröffentlichung enthält Beispiele für Daten und Berichte des alltäglichen Geschäftsablaufes. Um diese so realistisch wie möglich zu gestalten, enthalten sie auch Namen von Personen, Firmen, Marken und Produkten. Sämtliche dieser Namen sind fiktiv. Ähnlichkeiten mit Namen und Adressen tatsächlicher Unternehmen oder Personen sind zufällig.

COPYRIGHTLIZENZ:

Diese Veröffentlichung enthält Musterprogramme, die in Quellensprache geschrieben sind. Sie dürfen diese Musterprogramme kostenlos (d. h. ohne Zahlung an IBM) kopieren, ändern und verteilen, wenn dies zu dem Zweck geschieht, Anwendungsprogramme zu entwickeln, zu verwenden, zu vermarkten oder zu verteilen, die mit der Anwendungsprogrammierschnittstelle für die Betriebsumgebung konform sind, für die diese Musterprogramme geschrieben werden. Diese Beispiele wurden nicht unter allen denkbaren Bedingungen getestet. Daher kann IBM die Zuverlässigkeit, Wartungsfreundlichkeit oder Funktion dieser Programme weder zusagen noch gewährleisten.

Wird dieses Buch als Softcopy (Book) angezeigt, erscheinen keine Fotografien oder Farbabbildungen.

Informationen zu Programmierschnittstellen

Die bereitgestellten Informationen zur Programmierschnittstelle sollen Sie bei der Erstellung von Anwendungssoftware für dieses Programm unterstützen.

Dieses Handbuch enthält Informationen über vorgesehene Programmierschnittstellen, die es dem Kunden ermöglichen, Programme zu schreiben, um die Services von WebSphere MQzu erhalten.

Diese Informationen können jedoch auch Angaben über Diagnose, Bearbeitung und Optimierung enthalten. Die Informationen zu Diagnose, Bearbeitung und Optimierung sollten Ihnen bei der Fehlerbehebung für die Anwendungssoftware helfen.

Wichtig: Verwenden Sie diese Diagnose-, Änderungs-und Optimierungsinformationen nicht als Programmierschnittstelle, da sie Änderungen unterliegen.

Marken

IBM, das IBM Logo, ibm.com, sind Marken der IBM Corporation in den USA und/oder anderen Ländern. Eine aktuelle Liste der IBM Marken finden Sie auf der Webseite "Copyright and trademark information"www.ibm.com/legal/copytrade.shtml. Weitere Produkt- und Servicenamen können Marken von IBM oder anderen Unternehmen sein.

Microsoft und Windows sind Marken der Microsoft Corporation in den USA und/oder anderen Ländern.

UNIX ist eine eingetragene Marke von The Open Group in den USA und anderen Ländern.

Linux ist eine eingetragene Marke von Linus Torvalds in den USA und/oder anderen Ländern.

Dieses Produkt enthält Software, die von Eclipse Project (https://www.eclipse.org/) entwickelt wurde.

Java und alle auf Java basierenden Marken und Logos sind Marken oder eingetragene Marken der Oracle Corporation und/oder ihrer verbundenen Unternehmen.

Teilenummer:

(1P) P/N: