9.4

IBM MQ in Containern

Hinweis

Vor Verwendung dieser Informationen und des darin beschriebenen Produkts sollten die Informationen unter „Bemerkungen” auf Seite 183 gelesen werden.

Diese Ausgabe bezieht sich auf Version 9 Release 4 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

IBM MQ in Containern und IBM Cloud Pak for Integration..... 5
Informationen zu.....5
Releaseprotokoll für IBM MQ Operator..... 5
Planung.....7
Verwendung von IBM MQ in Containern auswählen..... 8
Unterstützung für IBM MQ in Containern..... 8
Lizenzierung von IBM MQ in Containern planen.....16
Speicher für IBM MQ Operator planen..... 16
Hochverfügbarkeit für IBM MQ in Containern planen..... 18
Disaster-Recovery für IBM MQ in Containern.....24
Sicherheit für IBM MQ in Containern planen..... 24
Skalierbarkeit und Leistung für IBM MQ in Containern planen.....30
Vorbereitung, Installation und Upgrade.....31
IBM MQ Operator installieren und Upgrade durchführen..... 32
IBM MQ durch Erstellung eines eigenen Container-Image vorbereiten..... 57
Implementierung und Konfiguration.....65
Warteschlangenmanager mithilfe von IBM MQ Operator implementieren und konfigurieren..... 66
Warteschlangenmanager mit Helm implementieren und konfigurieren.....109
Migration auf IBM MQ Operator..... 110
Prüfen, ob erforderliche Funktionen verfügbar sind..... 111
Warteschlangenmanagerkonfiguration extrahieren..... 111
Optional: Warteschlangenmanagerschlüssel und -zertifikate extrahieren und übernehmen.....112
Optional: LDAP konfigurieren.....114
Optional: Ändern der IP-Adressen und Hostnamen in der IBM MQ-Konfiguration..... 122
Warteschlangenmanagerkonfiguration für eine Containerumgebung aktualisieren..... 123
Ziel-HA-Architektur für IBM MQ in Containern auswählen.....127
Ressourcen für den Warteschlangenmanager erstellen.....127
Neuen Warteschlangenmanager unter Red Hat OpenShift erstellen.....129
Neue Containerimplementierung überprüfen.....133
Bedienung.....134
Betreiben von IBM MQ mittels IBM MQ Operator.....134
Status von Warteschlangenmanagern mit Native HA anzeigen.....142
Native HA-Warteschlangenmanagerinstanzen manuell beenden.....144
Referenz..... 144
API-Referenz für IBM MQ Operator.....145
Lizenzanmerkungen zur Erstellung eigener IBM MQ-Container-Images..... 170
IBM MQ Advanced for Developers Container-Image..... 175
Fehlerbehebung.....178
Fehlerbehebung bei ungeplanten Neustarts von IBM MQ in Containern.....178
Fehlerbehebung bei Problemen mit IBM MQ Operator..... 179

Bemerkungen..... 183
Informationen zu Programmierschnittstellen..... 184
Marken..... 185

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.

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

Informationen zu IBM MQ in Containern

Einführende Informationen unterstützen Sie bei den ersten Schritten mit IBM MQ in Containern.

Container sind eine Technologie, die das Packen und Isolieren von Code mit seiner Laufzeitumgebung ermöglicht, die auf eine Weise ausgeführt werden kann, die von anderer Software in derselben Infrastruktur isoliert ist. Dies erleichtert das Verschieben eines Warteschlangenmanagers oder einer Anwendung zwischen Umgebungen (z. B. Entwicklung, Test und Produktion). Moderne Container-Orchestrierer wie Red Hat OpenShift Container Platform und Kubernetes können viele Typen von Containern auf derselben Maschine ausführen, die in Bezug auf Ressourcen, Sicherheit und Fehler voneinander isoliert sind.

Sie können IBM MQ -Warteschlangenmanager oder Ihre IBM MQ -Anwendungen in Containern ausführen.

Zugehörige Informationen
Was sind Container?

Releaseprotokoll für IBM MQ Operator

Anmerkungen:

- Informationen zu früheren IBM MQ -Operatoren finden Sie im Releaseprotokoll für IBM MQ Operator in der IBM MQ 9.3 -Dokumentation.
- Informationen zu zukünftigen IBM MQ Updates finden Sie auf der Seite IBM MQ Recommended Fixes and Planned Maintenance release dates .

IBM MQ Operator 3.2.1

IBM Cloud Pak for Integration Version
IBM Cloud Pak for Integration 16.1.0

Operatorkanal
`v3.2-sc2`

Zulässige Werte für `.spec.version`
9.4.0.0-r1

Zulässige Werte für `.spec.version` während der Migration
9.3.0.0-r1, 9.3.0.0-r2, 9.3.0.0-r3,9.3.3.2-r3 9.3.0.1-r1, 9.3.0.1-r2, 9.3.0.1-r3, 9.3.0.1-r4, 9.3.0.3-r1, 9.3.0.4-r1, 9.3.0.4-r2, 9.3.0.5-r1, 9.3.0.5-r2, 9.3.0.5-r3, 9.3.0.6-r1, 9.3.0.10-r1, 9.3.0.10-r2, 9.3.0.11-r1, 9.3.0.11-r2, 9.3.0.15-r1, 9.3.0.16-r1, 9.3.0.16-r2, 9.3.0.17-r1, 9.3.0.17-r2, 9.3.0.17-r3, 9.3.1.0-r1, 9.3.1.0-r2, 9.3.1.0-r3, 9.3.1.1-r1, 9.3.2.0-r1, 9.3.2.0-r2, 9.3.2.1-r1, 9.3.2.1-r2, 9.3.3.0-r1, 9.3.3.0-r2, 9.3.3.1-r1, 9.3.3.1-r2, 9.3.3.2-r1, 9.3.3.2-r2, 9.3.3.2-r3, 9.3.3.3-r1, 9.3.3.3-r2, 9.3.4.0-r1, 9.3.4.1-r1, 9.3.5.0-r1, 9.3.5.0-r2, 9.3.5.1-r1, 9.3.5.1-r2

Versionen der Red Hat OpenShift Container Platform
OpenShift Container Platform 4.12 und höher.

Versionen der IBM Cloud Pak foundational services
Nur IBM Cloud Pak foundational services Version 4.6 .

Änderungen

- Behebt ein Problem in OpenShift Container Platform 4.12 , bei dem ein Upgrade auf den Kanal v3.2-sc2 zu unerwartetem Verhalten für IBM Cloud Pak for Integration -Benutzer führen könnte. Weitere Informationen finden Sie unter Upgrade von 2023.4 in der IBM Cloud Pak for Integration -Dokumentation.

IBM MQ Operator 3.2.0

IBM Cloud Pak for Integration Version
IBM Cloud Pak for Integration 16.1.0

Operatorkanal
`v3.2-sc2`

Zulässige Werte für `.spec.version`
9.4.0.0-r1

Zulässige Werte für `.spec.version` während der Migration
9.3.0.0-r1, 9.3.0.0-r2, 9.3.0.0-r3,9.3.3.2-r3 9.3.0.1-r1, 9.3.0.1-r2, 9.3.0.1-r3, 9.3.0.1-r4, 9.3.0.3-r1, 9.3.0.4-r1, 9.3.0.4-r2, 9.3.0.5-r1, 9.3.0.5-r2, 9.3.0.5-r3, 9.3.0.6-r1, 9.3.0.10-r1, 9.3.0.10-r2, 9.3.0.11-r1, 9.3.0.11-r2, 9.3.0.15-r1, 9.3.0.16-r1, 9.3.0.16-r2, 9.3.0.17-r1, 9.3.0.17-r2, 9.3.0.17-r3, 9.3.1.0-r1, 9.3.1.0-r2, 9.3.1.0-r3, 9.3.1.1-r1, 9.3.2.0-r1, 9.3.2.0-r2, 9.3.2.1-r1, 9.3.2.1-r2, 9.3.3.0-r1, 9.3.3.0-r2, 9.3.3.1-r1, 9.3.3.1-r2, 9.3.3.2-r1, 9.3.3.2-r2, 9.3.3.2-r3, 9.3.3.3-r1, 9.3.3.3-r2, 9.3.4.0-r1, 9.3.4.1-r1, 9.3.5.0-r1, 9.3.5.0-r2, 9.3.5.1-r1, 9.3.5.1-r2

Versionen der Red Hat OpenShift Container Platform
OpenShift Container Platform 4.12 und höher.

Versionen der IBM Cloud Pak foundational services
Nur IBM Cloud Pak foundational services Version 4.6 .

Neuerungen

- „Persistente Datenträger erweitern” auf Seite 105 wird jetzt unterstützt.
- Warteschlangenmanager können jetzt gestoppt werden, indem die Annotation `mq.ibm.com/stop` hinzugefügt und auf `true`gesetzt wird. Siehe„Warteschlangenmanager stoppen (mq.ibm.com/stop)” auf Seite 109

  Anmerkungen:

  - Für einen gestoppten WS-Manager ist das Feld *.replicas* in `StatefulSet` auf 0 gesetzt.
  - Da das Feld IBM MQ Operator jetzt aktiv das Feld *.replicas* im `StatefulSet`verwaltet, wird es sofort vom Operator zurückgesetzt, wenn Sie dieses Feld ändern.
  - Ältere Versionen von IBM MQ erhalten den Status 'Fehlgeschlagen', wenn Sie das Feld *.replicas* ändern, aber den geänderten Wert beibehalten. Wenn Ihre vorhandenen Betriebsprozeduren auf diesem Verhalten basieren, müssen Sie ab IBM MQ 9.4 die Annotation `mq.ibm.com/stop` verwenden.

Änderungen

- Versionen mit ungeraden Nummern von Red Hat OpenShift Container Platform werden jetzt unterstützt.
- IBM MQ Das Katalogimage wurde aus dem SQLite -Datenbankformat in das dateibasierte Katalogformat versetzt.
- Basiert auf Red Hat Universal Base Image 9.4-949.1716471857. **Hinweis:** Für UBI 9 steht eine FIPS 140-3-Zertifizierung an. UBI 9 wird in der Architektur von Power 8 nicht unterstützt.

- Die Sicherheitslücken, die adressiert werden, werden in diesem Sicherheitsbulletin detailliert beschrieben.

OpenShift CP4I Releaseprotokoll für Warteschlangenmanager-Container-Images zur Verwendung mit IBM MQ Operator

Anmerkung: Informationen zu früheren Warteschlangenmanager-Container-Images finden Sie unter Releaseprotokoll für IBM MQ Operator in der IBM MQ 9.3 -Dokumentation.

9.4.0.0-r1

CD CP4I-SC2

Erforderliche Operatorversion
3.2.0 oder höher

Unterstützte Architekturen
amd64, s390x, ppc64le

Bilder

- `cp.icr.io/cp/ibm-mqadvanced-server-integration:9.4.0.0-r1`
- `cp.icr.io/cp/ibm-mqadvanced-server:9.4.0.0-r1`
- `icr.io/ibm-messaging/mq:9.4.0.0-r1`

Neuerungen

- Neuerungen in IBM MQ 9.4.0 for Multiplatforms-Basis-und Advanced-Berechtigung

Änderungen

- Änderungen in IBM MQ 9.4.0
- Deprecated Wenn Sie IBM MQ Advanced for Developersverwenden, ist das Festlegen der Kennwörter für die Benutzer `admin` und `app` über eine Umgebungsvariable veraltet. Verwenden Sie stattdessen geheime Schlüssel.
- Für die Umgebungsvariable *MQ_LOGGING_CONSOLE_SOURCE*wurde der neue optionale Wert `mqsc` hinzugefügt. Diese Option kann verwendet werden, um den Inhalt von `autocfgmqsc.LOG` im Containerprotokoll wiederzugeben.
- Basiert auf Red Hat Universal Base Image 9.4-949.1716471857. **Hinweis:** Für UBI 9 steht eine FIPS 140-3-Zertifizierung an. UBI 9 wird in der Architektur von Power 8 nicht unterstützt.

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 OpenShift Container Platform-Architektur) vertraut machen.

Prozedur

- „Verwendung von IBM MQ in Containern auswählen” auf Seite 8.
- „Unterstützung für IBM MQ in Containern” auf Seite 8.
- „Speicher für IBM MQ Operator planen” auf Seite 16.
- „Hochverfügbarkeit für IBM MQ in Containern planen” auf Seite 18.

- „Disaster-Recovery für IBM MQ in Containern” auf Seite 24.
- „Benutzerauthentifizierung und -berechtigung für IBM MQ in Containern” auf Seite 25.

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

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

IBM MQ Operator erweitert die Red Hat OpenShift Container Platform -API, um eine neue angepasste `QueueManager` -Ressource hinzuzufügen. 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 Warteschlangenmanagerinstanzen durchführen. Siehe „Hinweise zur Durchführung einer eigenen schrittweisen Aktualisierung eines nativen HA-Warteschlangenmanagers” auf Seite 23

Einige IBM MQ-Features werden bei der Verwendung von IBM MQ Operator nicht unterstützt. Ausführliche Informationen zu den unterstützten Funktionen bei der Verwendung von IBM MQ Operatorfinden Sie unter „Unterstützung für IBM MQ in Containern” auf Seite 8 .

Eigene Images und eigenen Bereitstellungscode erstellen

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 „IBM MQ durch Erstellung eines eigenen Container-Image vorbereiten” auf Seite 57.

Details dazu, was beim Erstellen eines eigenen Image und Bereitstellungscodes unterstützt wird, finden Sie unter „Unterstützung für IBM MQ in Containern” auf Seite 8 .

Zugehörige Verweise

„Unterstützung für IBM MQ in Containern” auf Seite 8
Nicht alle IBM MQ -Funktionen sind in Containern auf dieselbe Weise verfügbar und werden unterstützt.

OpenShift CP4I CD CP4I-SC2 Unterstützung für IBM MQ in Containern

Nicht alle IBM MQ -Funktionen sind in Containern auf dieselbe Weise verfügbar und werden unterstützt.

Die folgende Tabelle zeigt detailliert, wie IBM MQ -Features mit IBM MQ Operatorunterstützt werden oder wenn Sie eigene Container und eigenen Bereitstellungscode erstellen.

Anmerkungen:

- Die vordefinierten IBM MQ Container-Images unter IBM Container Registry (icr.io und cp.icr.io) werden nur unterstützt und sind für Fixes auswählbar, wenn sie mit IBM MQ Operatorverwendet werden.
- Ab IBM MQ Operator Kanal v3.2wird Long Term Support (LTS) in Support Cycle 2 (SC2) umbenannt. Dies liegt daran, dass der einzige verfügbare LTS -Pfad für IBM MQ in Containern zwei Jahre Unterstützung unter IBM Cloud Pak for Integration -Berechtigung ist und IBM Cloud Pak for Integration den Begriff SC2übernommen hat. Hier ist das vollständige Bild der Berechtigung:

- Mit IBM MQ -Berechtigung kann IBM MQ Operator nur die Images von IBM MQ Continuous Delivery (CD) bereitstellen.
- Mit IBM Cloud Pak for Integration -Berechtigung kann IBM MQ Operator CD -oder SC2 (formerly LTS) -Images bereitstellen.

Es ist nicht möglich, die Lizenz des vordefinierten IBM MQ Advanced for Developers -Image auf eine andere Lizenz zu aktualisieren. IBM MQ Operator stellt je nach ausgewählter Lizenz unterschiedliche Images bereit.

In dieser Tabelle gelten die folgenden Bedingungen:

"Code für Containeraktivierung"
Die ausführbaren Dateien **runmqserver**, **runmqintegrationserver**, **chkmqhealthy**, **chkmqready** und **chkmqstarted**. Dieser Code wird als Beispiel bereitgestellt und nur als Teil der vordefinierten Container unterstützt, wenn er mit IBM MQ Operatorverwendet wird.

	IBM MQ Operator und eine IBM Cloud Pak for Integration -Lizenz verwenden	IBM MQ Operator und eine IBM MQ Advanced -Lizenz verwenden	IBM MQ Operator und eine IBM MQ Advanced for Developers -Lizenz verwenden	Vorerstelltes IBM MQ Advanced for Developers -Image	Eigenen Container erstellen
Unterstützte Plattformen	Nur unter Red Hat OpenShift Container Platform unterstützt. Releases von Red Hat OpenShift Container Platform werden von IBM MQ nicht mehr unterstützt, sobald Red Hat die Unterstützung stoppt. Weitere Informationen finden Sie unter „Versionsunterstützung für IBM MQ Operator” auf Seite 14.		Nur unter Red Hat OpenShift Container Platform verfügbar, aber nicht unterstützt.	Funktioniert auf allen Docker-, containerd- oder cri-o-Plattformen, aber nicht unterstützt. Details hierzu finden Sie in Systemvoraussetzungen für IBM MQ.	Jede Docker-, containerd- oder cri-o-Plattform. Details hierzu finden Sie in Systemvoraussetzungen für IBM MQ. Native Hochverfügbarkeit wird nur unter Kubernetes oder Red Hat OpenShift Container Platformunterstützt. Das Beispiel-Container-Image verwendet eine Red Hat Universal Base Image (UBI), die Linux® -Bibliotheken und -Dienstprogramme enthält, die von IBM MQverwendet werden. Das UBI wird von Red Hat unterstützt, wenn die Ausführung unter Red Hat OpenShift erfolgt. Der *Containeraktivierungscode* wird nicht unterstützt.

	IBM MQ Operator und eine IBM Cloud Pak for Integration -Lizenz verwenden	**IBM MQ Operator und eine IBM MQ Advanced -Lizenz verwenden**	**IBM MQ Operator und eine IBM MQ Advanced for Developers -Lizenz verwenden**	**Vorerstelltes IBM MQ Advanced for Developers -Image**	**Eigenen Container erstellen**
CPU-Architekturen	Unterstützt unter `amd64` und `s390x` z/Linux. Wird auch auf Systemen mit `ppc64le` Power Systems Version 9 und höher unterstützt. Beachten Sie, dass alle Knoten im Red Hat OpenShift Container Platform -Cluster dieselbe CPU-Architektur verwenden müssen.		Verfügbar unter `amd64` und `s390x` z/Linux, aber nicht unterstützt. Auch auf Systemen mit `ppc64le` Power Systems Version 9 und höher verfügbar, aber nicht unterstützt. Beachten Sie, dass alle Knoten im Red Hat OpenShift Container Platform -Cluster dieselbe CPU-Architektur verwenden müssen.		Gemäß IBM MQ -Software.
Dauer der Unterstützung	IBM Cloud Pak for Integration -Support Cycle 2 (formerly Long Term Support) oder Continuous Delivery.[1] CD -Bediener und -Warteschlangenmanager werden bis zum nächsten IBM Cloud Pak for Integration CD -oder CP4I-SC2 -Release unterstützt. CP4I-SC2 -Bediener und -Warteschlangenmanager werden bis zum nächsten IBM Cloud Pak for Integration CP4I-SC2 -Release unterstützt, zuzüglich einer Karenzzeit für das Upgrade.	Nur Continuous Delivery -Datenstrom sowohl für IBM MQ Operatorals auch für Warteschlangenmanager. Jede IBM MQ Operator -und Warteschlangenmanagerversion wird nur bis zum nächsten CD -Release unterstützt.	Nicht unterstützt		Gemäß IBM MQ -Software. Siehe IBM MQ -Häufig gestellte Fragen zu Long Term Support-und Continuous Delivery-Releases. Der *Containeraktivierungscode* wird nicht unterstützt.

	IBM MQ Operator und eine IBM Cloud Pak for Integration -Lizenz verwenden	IBM MQ Operator und eine IBM MQ Advanced -Lizenz verwenden	IBM MQ Operator und eine IBM MQ Advanced for Developers -Lizenz verwenden	Vorerstelltes IBM MQ Advanced for Developers -Image	Eigenen Container erstellen
Verfügbarkeit von Sicherheitsfixes	Regelmäßige Fixes als Container-Images unter IBM Container Registry verfügbar				Fixes für IBM MQ -Software sind als Software unter Fix Centralverfügbar. Der *Containeraktivierungscode* wird nicht unterstützt.
Verfügbarkeit vorläufiger Fixes	Es sind Fixes für Warteschlangenmanager als Software verfügbar und ein angepasster Image-Build ist erforderlich. IBM MQ Operator -Fixes sind nicht als vorläufige Fixes verfügbar.		Es sind keine vorläufigen Fixes verfügbar.		Fixes für IBM MQ Software sind als Software unter Fix Central oder über IBM Support verfügbar. Der *Containeraktivierungscode* wird nicht unterstützt.
Komponente: Advanced Message Security	Unterstützt. Beachten Sie, dass die serverseitige Verschlüsselung nicht einfach zu verwenden ist, da IBM MQ Operator nicht direkt die Angabe eines eigenen Keystores für Advanced Message Securityermöglicht.		Verfügbar, aber nicht unterstützt.		Wird gemäß IBM MQ -Software unterstützt, aber es ist kein Beispiel verfügbar.
Komponente: Managed File Transfer	Nicht verfügbar und nicht unterstützt. Sie können jedoch IBM MQ Operator verwenden, um einen oder mehrere Koordinations-, Befehls-oder Agentenwarteschlangenmanager bereitzustellen.			Nicht verfügbar und nicht unterstützt.	Unterstützt gemäß IBM MQ -Software mit sample für den Agenten.

[1] Die IBM MQ Operator wird entweder als IBM MQ CD -Release oder als IBM Cloud Pak for Integration - Support Cycle 2 (formerly Long Term Support) -Release unterstützt:

- IBM MQ 9.4.0.x -Container-Images, die mit IBM MQ Operator 3.2.xbereitgestellt werden und als Teil von IBM Cloud Pak for Integration 16.1.0verwendet werden, sind für den CP4I-LTS Support berechtigt. Das neueste Release von Support Cycle 2 (SC2) von IBM MQ Operator ist 3.2.1und das neueste Container-Image von SC2 ist 9.4.0.0-r1.
- IBM MQ 9.4.0.x -Container-Images, die mit IBM MQ Operator 3.2.xbereitgestellt werden und als Teil von IBM Cloud Pak for Integration 16.1.0verwendet werden, sind für den CD Support berechtigt. Das neueste Release von Continuous Delivery (CD) von IBM MQ Operator ist 3.2.1und das neueste Container-Image von CD ist 9.4.0.0-r1.

	IBM MQ Operator und eine IBM Cloud Pak for Integration -Lizenz verwenden	IBM MQ Operator und eine IBM MQ Advanced -Lizenz verwenden	IBM MQ Operator und eine IBM MQ Advanced for Developers -Lizenz verwenden	Vorerstelltes IBM MQ Advanced for Developers -Image	Eigenen Container erstellen
Komponente: MQTT	Nicht verfügbar und nicht unterstützt.				Wird gemäß IBM MQ -Software unterstützt, aber es ist kein Beispiel verfügbar.
Funktion: AMQP	Nicht verfügbar und nicht unterstützt.				Wird gemäß IBM MQ -Software unterstützt, aber es ist kein Beispiel verfügbar.
Komponente: REST API	Verfügbar und unterstützt.				Verfügbar und unterstützt gemäß IBM MQ -Software.
Feature: Replizierte Datenwarteschlangenmanager	Nicht verfügbar und nicht unterstützt. Replizierte Datenwarteschlangenmanager (RDQM) sind eng mit dem Linux -Kernel verbunden und werden in Containern nicht unterstützt.				
Feature: Native HA	Verfügbar und unterstützt.		Verfügbar, aber nicht unterstützt.		Nur unter Kubernetes und Red Hat OpenShift Container Platformverfügbar Wird gemäß IBM MQ -Software unterstützt.
Feature: Multi-Instanz-Warteschlangenmanager	Verfügbar und unterstützt.		Verfügbar, aber nicht unterstützt.		Verfügbar und unterstützt gemäß IBM MQ -Software.
Feature: Typen von Wiederherstellungsprotokollen	Nur Umlaufprotokollierung oder replizierte Protokolle. Die lineare Protokollierung wird nicht unterstützt.				Verfügbar und unterstützt gemäß IBM MQ -Software. Sie müssen die **crtmqm** -Optionen konfigurieren.

	IBM MQ Operator und eine IBM Cloud Pak for Integration -Lizenz verwenden	IBM MQ Operator und eine IBM MQ Advanced -Lizenz verwenden	IBM MQ Operator und eine IBM MQ Advanced for Developers -Lizenz verwenden	Vorerstelltes IBM MQ Advanced for Developers -Image	Eigenen Container erstellen
Feature: Angepasste Befehlszeilenoptionen für `crtmqdir`, `crtmqm`, `strmqm` und `endmqm` angeben	Nicht verfügbar und nicht unterstützt. Die meisten Optionen können mithilfe einer INI-Datei konfiguriert werden, einige können jedoch nicht konfiguriert werden, z. B. die Verwendung der linearen Protokollierung.				Optional, je nachdem, wie Sie Ihren *Containeraktivierungscode* implementieren.
Feature: Betriebssystembenutzer	Nicht verfügbar und nicht unterstützt.				Möglich und unterstützt gemäß IBM MQ -Software, falls Sie IBM MQ mit RPMs installieren, aber kein Beispiel verfügbar ist. Aufgrund von Sicherheitsrisiken nicht empfohlen.

Anmerkung: Der Ausdruck "gemäß IBM MQ -Software unterstützt" bedeutet, dass der IBM Technical Support auf die Kernsoftware IBM MQ beschränkt ist, die innerhalb des Containers ausgeführt wird.

Zugehörige Konzepte

IBM MQ FAQ für langfristigen Support und Continuous Delivery-Releases

Zugehörige Verweise

IBM Cloud Pak for Integration Software Support Lifecycle Addendum

OpenShift CP4I CD CP4I-SC2 Versionsunterstützung für IBM MQ Operator

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

- „Verfügbare IBM MQ-Versionen” auf Seite 15
- „Kompatible Red Hat OpenShift Container Platform-Versionen” auf Seite 15
- „Versionen der IBM Cloud Pak for Integration” auf Seite 15
- „Verfügbare IBM MQ-Versionen in älteren Operatoren” auf Seite 15
- „Kompatible OpenShift Container Platform-Versionen für ältere Operatoren” auf Seite 15

Verfügbare IBM MQ-Versionen

Operatorkanal	Operatorversion	Versionen der IBM MQ						
		9.4.0	9.3.5	9.3.4	9.3.3	9.3.2	9.3.1	9.3.0
v32-sc2	3.2	CD und SC2	dep	dep	dep	dep	dep	MIG

Key:

- CD: Continuous Delivery -Unterstützung ist verfügbar
- SC2: IBM Cloud Pak for Integration - Support Cycle 2 (formerly Long Term Support) ist verfügbar.
- MIG: Nur während der Migration von einem Operanden IBM Cloud Pak for Integration - Support Cycle 2 (formerly Long Term Support) auf einen Operanden Continuous Delivery verfügbar.
- DEP: Deprecated Veraltet. Da die Unterstützung für IBM MQ -Releases eingestellt wird, sind sie möglicherweise weiterhin im Operator konfigurierbar, kommen jedoch nicht mehr für die Unterstützung infrage und werden möglicherweise in zukünftigen Releases entfernt.

Siehe „Releaseprotokoll für IBM MQ Operator" auf Seite 5 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 OpenShift Container Platform[2]		
		4.15	4.14	4.12
v3.2-sc2	Ab 3.2.0	SC2	SC2	SC2

Key:

- CD: Continuous Delivery -Unterstützung ist verfügbar
- SC2: IBM Cloud Pak for Integration - Support Cycle 2 (formerly Long Term Support) ist verfügbar.
- EOS: Nicht mehr unterstützt. Migrieren Sie auf eine neuere OpenShift Container Platform -Version.

Versionen der IBM Cloud Pak for Integration

Wird für die Verwendung als Teil von IBM Cloud Pak for Integration Version 16.1.0oder unabhängig davon unterstützt:

- IBM MQ Operator 3.2.x

Verfügbare IBM MQ-Versionen in älteren Operatoren

Siehe Verfügbare IBM MQ -Versionen in der IBM MQ 9.3 -Dokumentation.

Kompatible OpenShift Container Platform-Versionen für ältere Operatoren

Siehe Kompatible OpenShift Container Platform -Versionen in der IBM MQ 9.3 -Dokumentation.

OpenShift MQ Adv. Von IBM MQ Operator erstellte Ressourcen bearbeiten

IBM MQ Operator gleicht eine angepasste QueueManager -Ressource ab, indem native Kubernetes -Ressourcen erstellt und verwaltet werden. Diese verwalteten Ressourcen dürfen **nicht** direkt bearbeitet werden.

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

Sie können normalerweise ermitteln, ob eine Ressource zu einer anderen übergeordneten Ressource gehört, indem Sie die `ownerReferences`anzeigen. Die folgenden Metadaten aus einer `StatefulSet` zeigen beispielsweise, dass sie der `QueueManager` -Ressource "qm1" gehören:

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

Beachten Sie, dass nicht alle Ressourcen über diese Metadaten verfügen.

Es liegt in der Verantwortung des IBM MQ Operator s, die zugrunde liegenden Ressourcen wie `StatefulSet`, `Service` und `Route`zu verwalten. Wenn Sie eine dieser zugrunde liegenden Ressourcen ändern, ändert der IBM MQ Operator sie wieder und es kann zu Ausfallzeiten kommen, wenn diese Änderung eine rollierende Aktualisierung erfordert.

Die meisten wichtigen Einstellungen für Warteschlangenmanager sind in der Ressource `QueueManager` verfügbar. Wenn Sie jedoch feststellen, dass Sie die vollständige Kontrolle über die zugrunde liegenden Ressourcen benötigen, gibt es einige Optionen:

- Wenn Sie die Einstellungen für den von IBM MQ Operatorerstellten Pod überschreiben müssen, können Sie eine Vorlage für Pod-Überschreibung im Abschnitt `.spec.template` der YAML-Datei `QueueManager` hinzufügen.
- Wenn Sie Einstellungen auf dem Warteschlangenmanager `Route` überschreiben müssen, der von IBM MQ Operatorerstellt wurde, müssen Sie die Einstellung `.spec.route.enabled` für die Route vollständig inaktivieren und dann Ihre eigene Route erstellen.
- Einstellungen wie Beschriftungen und Anmerkungen sowie Pod Einstellungen wie `security Context`können in der `QueueManager` -Ressource festgelegt werden.
- In anderen Fällen ist IBM MQ Operator möglicherweise nicht für Ihren Anwendungsfall geeignet, wenn Sie uneingeschränkten Zugriff benötigen.

Lizenzierung von IBM MQ in Containern planen

Die Containerlizenzierung ermöglicht es Ihnen, nur die verfügbare Kapazität Ihrer einzelnen IBM MQ -Container zu lizenzieren, anstatt den gesamten Server zu lizenzieren, auf dem Ihre Container ausgeführt werden. Um die Containerlizenzierung nutzen zu können, muss IBM License Service verwendet werden, um die Lizenznutzung zu verfolgen und Ihre erforderliche Berechtigung zu ermitteln.

Zugehörige Verweise

„Lizenzanmerkungen zur Erstellung eigener IBM MQ-Container-Images” auf Seite 170
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.

Zugehörige Informationen

IBM Containerlizenzen

Häufig gestellte Fragen zur Containerlizenzierung

Lizenzservice installieren

Lizenznutzung anzeigen und verfolgen

OpenShift CP4I Kubernetes Speicher für IBM MQ Operator planen

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 98). Geschieht dies nicht, muss IBM MQ bei jedem Neustart des Containers konfiguriert werden.

OpenShift CP4I Um IBM MQ beispielsweise mit ephemerem Speicher zu konfigurieren, sollte der Speichertyp des `QueueManager` Folgendes einschließen:

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

Persistenter Speicher

OpenShift CP4I

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

- **`spec.queueManager.availability`** steuert den Verfügbarkeitsmodus. Wenn Sie `SingleInstance` oder `NativeHA`verwenden, benötigen Sie nur `ReadWriteOnce` -Speicher. Für `multiInstance` benötigen Sie eine Speicherklasse, die `ReadWriteMany` mit den korrekten Dateisperrmerkmalen 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 planen" auf Seite 18.
- **`spec.queueManager.storage`** steuert die einzelnen Speichereinstellungen. Ein Warteschlangenmanager kann für die Verwendung zwischen einem und 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: [65534] # Change to 99 for clusters with RHEL7 or earlier worker nodes
```

Informationen zu Speicheraspekten für native HA-Warteschlangenmanager finden Sie unter „Native HA" auf Seite 21.

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

Speicherkapazität

OpenShift CP4I

Wenn Sie IBM MQ Operator verwenden, sollten Sie sicherstellen, dass Sie Datenträger anfordern, die groß genug für Ihre laufenden Anforderungen sind. Wenn Sie jedoch die Speicherkapazität eines oder mehrerer Datenträger erhöhen müssen, können diese Datenträger erweitert werden, wenn Ihre Speicherklasse die Datenträgererweiterung unterstützt. Datenträger können durch eine Online-oder Offline-Prozedur erweitert werden. Eine Offlineprozedur erfordert, dass `QueueManager` -Pods erneut gestartet werden, eine Onlineprozedur hingegen nicht. Informationen dazu, ob Ihre Speicherklasse die Datenträgererweiterung unterstützt und welche Prozedur die Datenträgererweiterung befolgt, finden Sie in der Dokumentation Ihres Speicheranbieters. Sie sollten diese Informationen berücksichtigen, wenn Sie eine Speicherklasse auswählen. Informationen zur Datenträgererweiterung finden Sie in „Persistente Datenträger erweitern" auf Seite 105.

Verschlüsselung

OpenShift CP4I

IBM MQ verschlüsselt ruhende Daten nicht aktiv. Daher sollten Sie passiv verschlüsselten Speicher und/oder IBM MQ Advanced Message Securityverwenden, um Ihre Nachrichten zu verschlüsseln. Unter IBM Cloud sind Block-und Dateispeicher mit passiver Verschlüsselung im Ruhezustand verfügbar.

OpenShift Kubernetes Hochverfügbarkeit für IBM MQ in Containern planen

Es gibt drei Möglichkeiten für Hochverfügbarkeit mit IBM MQ Operator: **Nativer HA-Warteschlangenmanager** (der über ein aktives Replikat und zwei Standby-Replikate verfügt), **Multi-Instanz-Warteschlangenmanager** (bei dem es sich um ein Aktiv/Standby-Paar handelt, das ein gemeinsam genutztes, vernetztes Dateisystem verwendet) oder **Einzelner ausfallsicherer Warteschlangenmanager** (der eine einfache Methode für Hochverfügbarkeit unter Verwendung eines vernetzten Speichers bietet). Die beiden letzt-

eren 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

MQ Adv.

Native HA-Warteschlangenmanager umfassen einen **aktiven** und zwei **Replikatpods** Kubernetes , die als Teil eines Kubernetes StatefulSet mit genau drei Replikaten mit jeweils eigenen Kubernetes -Persistent Volumes ausgeführt werden. 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 21.

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.

Anmerkung: Die Änderung der Details des StatefulSet eines IBM MQ -Warteschlangenmanagers, einschließlich der Anzahl der Replikate, wird nicht unterstützt, wenn der Warteschlangenmanager über IBM MQ Operatorerstellt wird.

Zugehörige Konzepte

Hochverfügbarkeitskonfigurationen

Zugehörige Tasks

„Hohe Verfügbarkeit für Warteschlangenmanager mithilfe von IBM MQ Operator konfigurieren” auf Seite 78

CP4I MQ Adv. 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.

Für Leistung und Zuverlässigkeit wird der persistente RWO-Speicher (ReadWriteOnce) für die Verwendung mit einer nativen Hochverfügbarkeitskonfiguration empfohlen. RWO-Datenträger von Speicheranbietern werden unterstützt, wenn sie die folgenden Bedingungen erfüllen:

- Von einem Blockspeicheranbieter abgerufen.
- Formatiert als ext4 oder XFS (wodurch die POSIX -Konformität sichergestellt wird)
- Unterstützt die dynamische Datenträgerbereitstellung und den Modus "volumeBinding: WaitForFirst-Consumer".

Folgende Anbieter sind ausdrücklich untersagt:

- NFS
- GlusterFS
- Andere Nicht-Blockprovider.

Die folgende Abbildung zeigt eine typische Implementierung mit drei Instanzen eines Warteschlangenmanagers in drei Containern.

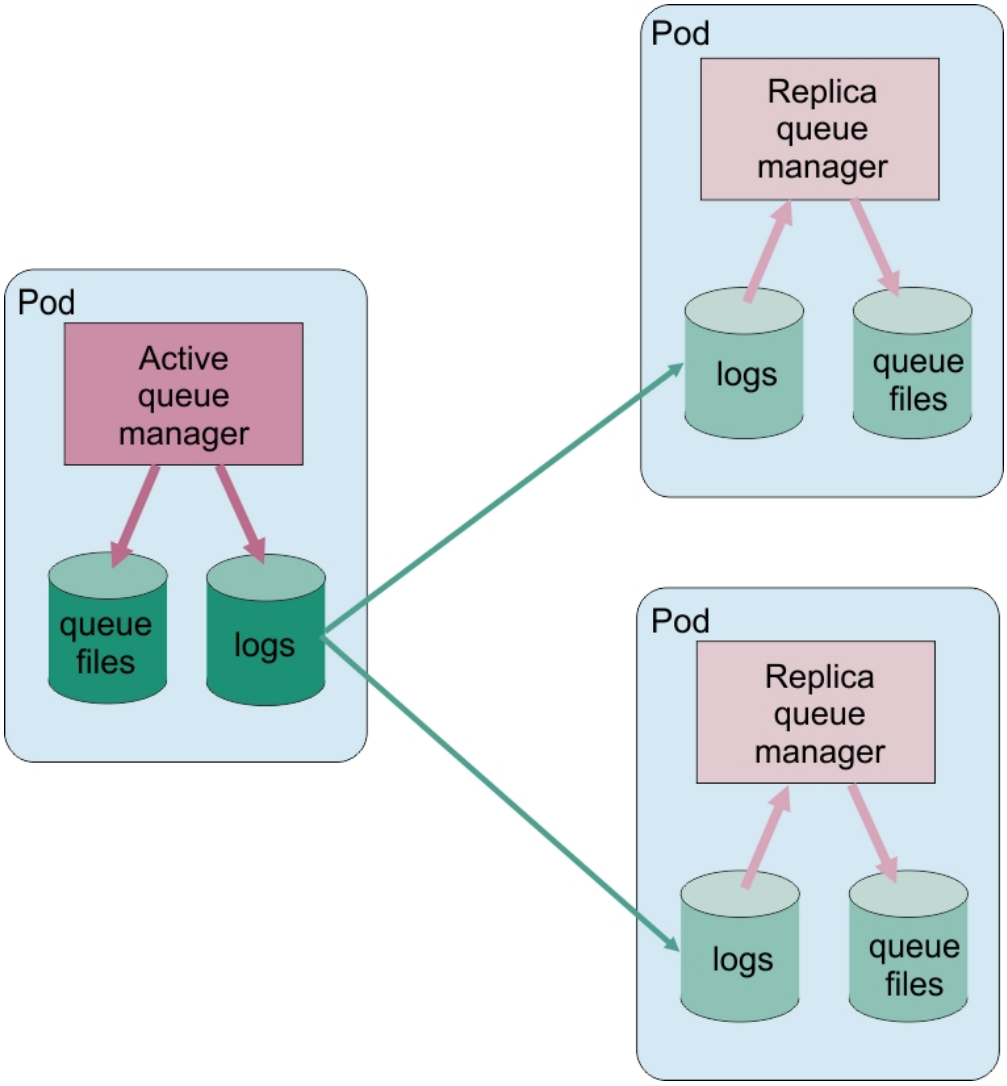


Abbildung 1. Beispiel einer Native HA-Konfiguration

MQ Adv. 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.

Kubernetes In Kuberneteswerden `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 `REPLICA`erreicht 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 8
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.

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

OpenShift CP4I Sicherheit für IBM MQ in Containern planen

Sicherheitsaspekte bei der Planung Ihrer IBM MQ in der Containerkonfiguration.

Prozedur

- „Benutzerauthentifizierung und -berechtigung für IBM MQ in Containern” auf Seite 25
  - „Sicherheitsvorgaben für die Verwendung von Betriebssystembenutzern in Containern” auf Seite 25
- „Hinweise zur Beschränkung des Netzverkehrs auf IBM MQ in Containern” auf Seite 26

Benutzerauthentifizierung und -berechtigung für IBM MQ in Containern

IBM MQ in Containern kann für die Authentifizierung von Benutzern über LDAP, gegenseitige TLS oder ein angepasstes MQ -Plug-in konfiguriert werden.

Beachten Sie, dass IBM MQ Operator die Verwendung von Betriebssystembenutzern und -gruppen im Container-Image nicht zulässt. Weitere Informationen finden Sie unter „Sicherheitsvorgaben für die Verwendung von Betriebssystembenutzern in Containern” auf Seite 25.

LDAP

Informationen zum Konfigurieren von IBM MQ für die Verwendung eines LDAP-Benutzerrepositorys finden Sie unter Verbindungsauthentifizierung: Benutzerrepositorys und LDAP-Berechtigung.

Gegenseitige TLS

Wenn Sie eingehende Verbindungen zu einem Warteschlangenmanager so konfigurieren, dass ein TLS-Zertifikat (gegenseitige TLS-Authentifizierung) erforderlich ist, können Sie den definierten Namen des Zertifikats einem Benutzernamen zuordnen. Sie müssen zwei Dinge tun:

- Konfigurieren Sie einen Kanalauthentifizierungsdatensatz, um die Zuordnung zu einem Benutzernamen mit SSLPEER zu erstellen. Weitere Informationen finden Sie unter Zuordnung eines definierten SSL-oder TLS-Namens zu einer MCAUSER-Benutzer-ID.
- Konfigurieren Sie den Warteschlangenmanager so, dass Sie Berechtigungssätze für einen Benutzernamen definieren, der dem System nicht bekannt ist. Weitere Informationen finden Sie unter Zeilengruppe 'Service' der Datei qm.ini.

JSON-Web-Token

Informationen zum Konfigurieren von IBM MQ für die Verwendung von JSON Web Tokens (JWT) finden Sie unter Mit Authentifizierungstokens arbeiten.

Angepasstes MQ -Plug-in

Dies ist eine fortgeschrittene Technik und erfordert viel mehr Arbeit. Weitere Informationen finden Sie unter Angepassten Berechtigungsservice verwenden.

Zugehörige Tasks

„Beispiel: Warteschlangenmanager mit gegenseitiger TLS-Authentifizierung konfigurieren” auf Seite 73
In diesem Beispiel wird ein Warteschlangenmanager mit IBM MQ Operator in OpenShift Container Platform bereitgestellt. Gegenseitige TLS wird für die Authentifizierung verwendet, um ein TLS-Zertifikat einer Identität im Warteschlangenmanager zuzuordnen.

Sicherheitsvorgaben für die Verwendung von Betriebssystembenutzern in Containern

Die Verwendung von Betriebssystembenutzern in Containern wird nicht empfohlen und ist mit IBM MQ Operator untersagt.

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 on Linux verwendet die Berechtigungseskalation, um die Kennwörter von Benutzern zu überprüfen. Es wird ein "setuid" -Programm verwendet, um zum Rootbenutzer zu werden.

Um die Einhaltung dieser Sicherheitsmaßnahmen sicherzustellen, lässt IBM MQ Operator die Verwendung von IDs, die in den Betriebssystembibliotheken in einem Container definiert sind, nicht zu. Im Container ist keine `mqm`-Benutzer-ID oder -Gruppe definiert.

Hinweise zur Beschränkung des Netzverkehrs auf IBM MQ in Containern

Sie können Netzrichtlinien definieren, um den Datenverkehr auf Pods in Ihrem Cluster in OpenShift Container Platform und Kuberneteszu beschränken. In diesem Abschnitt werden einige Aspekte beschrieben, wie Netzrichtlinien auf IBM MQangewendet werden können.

Für den Netzingress zu einem Warteschlangenmanager müssen mehrere Ports berücksichtigt werden:

- Port 1414 für Datenverkehr des Warteschlangenmanagers
- Port 9414 für native HA
- Port 9157 für Metriken
- Port 9443 für die Webkonsole und REST-APIs

Der ausgehende Netzbetrieb ist komplexer. Beispiele für ausgehende Netzverbindungen, die Sie berücksichtigen sollten:

- DNS-wenn Sie Kanäle oder eine andere Konfiguration haben, die DNS-Namen verwenden
- Andere Warteschlangenmanager
- OCSP (Online Certificate Status Protocol) und CRLs (Certificate Revocation Lists)-festgelegt durch Ihren Zertifikatsprovider.
- Authentifizierungsprovider:
  - LDAP
  - Open ID Connect oder ein anderer konfigurierter Anmeldeprovider für den IBM MQ -Web-Server. Dazu gehört auch die IBM Cloud Pak Keycloak.
- Tracing-Provider:
  - IBM Instana

  Anmerkung: Für frühere IBM MQ Versionen war das IBM Cloud Pak for Integration Operations Dashboard auch als Traceprovider verfügbar. Das Operations Dashboard wurde jedoch in IBM MQ 9.3.3 CD und IBM MQ 9.4.0 LTSentfernt.

Beispiel für Ingress NetworkPolicy

Das folgende Beispiel zeigt eine Netzrichtlinie zur Steuerung von Ingress für einen Warteschlangenmanager mit dem Namen "myqm" zur Verwendung unter Red Hat OpenShift Container Platform.

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

```
      port: 1414
  # Allow access to Native HA port from other instances of the same queue manager
  - from:
      - podSelector:
          matchLabels:
            app.kubernetes.io/instance: myqm
            app.kubernetes.io/name: ibm-mq
    ports:
      - protocol: TCP
        port: 9414
  # Allow access to metrics from monitoring project
  - from:
      - namespaceSelector:
          matchLabels:
            network.openshift.io/policy-group: monitoring
    ports:
      - protocol: TCP
        port: 9157
  # Allow access to web server via Route
  - from:
      - namespaceSelector:
          matchLabels:
            network.openshift.io/policy-group: ingress
    ports:
      - protocol: TCP
        port: 9443
```

FIPS-Konformität für IBM MQ in Containern

Beim Start erkennt IBM MQ in Containern, ob das Betriebssystem, auf dem der Container gestartet wird, FIPS-konform ist, und konfiguriert (falls ja) die FIPS-Unterstützung automatisch. Anforderungen und Einschränkungen sind hier angegeben.

Federal Information Processing Standards

Die US-Regierung produziert technische Beratung zu IT-Systemen und Sicherheit, einschließlich der Datenverschlüsselung. Das National Institute for Standards and Technology (NIST) ist eine Behörde, die sich mit IT-Systemen und Sicherheit befasst. NIST erstellt Empfehlungen und Standards, einschließlich der Federal Information Processing Standards (FIPS).

Ein wichtiger FIPS-Standard ist FIPS 140-2, der die Verwendung starker Verschlüsselungsalgorithmen erfordert. FIPS 140-2 gibt außerdem Anforderungen an Hashing-Algorithmen an, die zum Schutz von Paketen vor Änderungen im Transit verwendet werden sollen.

IBM MQ stellt Unterstützung für FIPS 140-2 bereit, wenn es entsprechend konfiguriert wurde.

Anmerkung: Unter AIX, Linux, and Windowsstellt IBM MQ die Konformität mit FIPS 140-2 über das Verschlüsselungsmodul IBM Crypto for C (ICC) bereit. Das Zertifikat für dieses Modul wurde in den Langzeitstatus versetzt. Kunden sollten das IBM Crypto for C (ICC) -Zertifikat anzeigen und sich über alle Empfehlungen von NIST im Klaren sein. Ein Ersatz-FIPS 140-3-Modul ist derzeit in Bearbeitung und sein Status kann angezeigt werden, indem in der NIST-CMVP-Module in der Prozesslistenach ihm gesucht wird.

IBM MQ Operator 3.2.0 und das Container-Image des Warteschlangenmanagers ab 9.4.0.0 basieren auf UBI 9. Die Konformität mit FIPS 140-3 steht derzeit an und ihr Status kann angezeigt werden, indem Sie in der NIST CMVP-Module in der Prozesslistenach "Red Hat Enterprise Linux 9- OpenSSL FIPS Provider" suchen.

Voraussetzungen

Anforderungen in Bezug auf die Clusterkonfiguration und andere Aspekte finden Sie unter FIPS Wall: Current IBM approach to FIPS compliance.

IBM MQ in Containern kann im Konformitätsmodus FIPS 140-2 ausgeführt werden. Während des Starts erkennt IBM MQ in Containern, ob das Hostbetriebssystem, auf dem der Container gestartet wird, FIPS-konform ist. Wenn das Hostbetriebssystem FIPS-konform ist und private Schlüssel und Zertifikate bereitgestellt wurden, konfiguriert der IBM MQ -Container den Warteschlangenmanager, den IBM MQ

-Web-Server und die Datenübertragung zwischen den Knoten in einer nativen Hochverfügbarkeitsimplementierung für die Ausführung im FIPS-Konformitätsmodus.

Wenn Sie IBM MQ Operator zum Bereitstellen von Warteschlangenmanagern verwenden, erstellt der Bediener eine Route mit dem Beendigungstyp **Passthrough**. Dies bedeutet, dass der Datenverkehr direkt an das Ziel gesendet wird, ohne dass der Router die TLS-Terminierung bereitstellt. Der IBM MQ -Warteschlangenmanager und der IBM MQ -Web-Server sind in diesem Fall die Ziele und stellen bereits eine FIPS-konforme sichere Kommunikation bereit.

Wichtige Anforderungen:

1. Ein privater Schlüssel und Zertifikate, die in einem geheimen Schlüssel für den Warteschlangenmanager und den Web-Server bereitgestellt werden und es externen Clients ermöglichen, eine sichere Verbindung zum Warteschlangenmanager und zum Web-Server herzustellen.
2. Ein privater Schlüssel und Zertifikate für die Datenübertragung zwischen verschiedenen Knoten in einer nativen Hochverfügbarkeitskonfiguration.

Einschränkungen

Beachten Sie bei einer FIPS-konformen Bereitstellung von IBM MQ in Containern Folgendes:

- IBM MQ in Containern stellt einen Endpunkt für die Erfassung von Metriken bereit. Derzeit ist dieser Endpunkt nur HTTP. Sie können den Metrikendpunkt inaktivieren, um den Rest von IBM MQ FIPS-konform zu machen.
- IBM MQ in Containern ermöglicht angepasste Imageüberschreibungen. Das heißt, Sie können angepasste Images mit dem IBM MQ -Container-Image als Basisimage erstellen. FIPS-Konformität gilt möglicherweise nicht für solche angepassten Images.
- Für die Nachrichtenüberwachung mit IBM Instanaerfolgt die Kommunikation zwischen IBM MQ und IBM Instana über HTTP oder HTTPS ohne FIPS-Konformität.
- Der IBM MQ Operator -Zugriff auf IBM Identity and Access Management (IAM) /Zen-Services ist nicht FIPS-konform.

Erkennung der FIPS-Konformität und automatische Konfiguration der FIPS-Unterstützung

Wenn das Betriebssystem, auf dem der Container gestartet wird, FIPS-konform ist, wird die FIPS-Unterstützung automatisch konfiguriert.

Anmerkung: Unter AIX, Linux, and Windowsstellt IBM MQ die Konformität mit FIPS 140-2 über das Verschlüsselungsmodul IBM Crypto for C (ICC) bereit. Das Zertifikat für dieses Modul wurde in den Langzeitstatus versetzt. Kunden sollten das IBM Crypto for C (ICC) -Zertifikat anzeigen und sich über alle Empfehlungen von NIST im Klaren sein. Ein Ersatz-FIPS 140-3-Modul ist derzeit in Bearbeitung und sein Status kann angezeigt werden, indem in der NIST-CMVP-Module in der Prozesslistenach ihm gesucht wird.

IBM MQ Operator 3.2.0 und das Container-Image des Warteschlangenmanagers ab 9.4.0.0 basieren auf UBI 9. Die Konformität mit FIPS 140-3 steht derzeit an und ihr Status kann angezeigt werden, indem Sie in der NIST CMVP-Module in der Prozesslistenach "Red Hat Enterprise Linux 9- OpenSSL FIPS Provider" suchen.

Während des Starts erkennt IBM MQ in Containern, ob das Betriebssystem, unter dem der Container gestartet wird, FIPS-konform ist. Ist dies der Fall, werden die folgenden Aktionen automatisch ausgeführt:

Warteschlangenmanager
Wenn das Hostbetriebssystem FIPS-konform ist und der private Schlüssel und die Zertifikate bereitgestellt werden, wird das Warteschlangenmanagerattribut **SSLFIPS** auf YESgesetzt. Andernfalls wird das Attribut **SSLFIPS** auf N0gesetzt.

IBM MQ Web-Server
Der IBM MQ -Web-Server stellt eine HTTP/HTTPS-Schnittstelle zur Verwaltung von IBM MQbereit. Wenn das Hostbetriebssystem FIPS-konform ist, werden die JVM-Optionen aktualisiert, damit der

Web-Server FIPS-konforme Verschlüsselung verwendet. Damit FIPS verwendet werden kann, müssen der private Schlüssel und die Zertifikate beim Containerstart bereitgestellt werden.

Native HA
Die Sicherheit der zwischen Knoten replizierten Daten wird durch die Zeilengruppe **NativeHALocalInstance** der Datei `qm.ini` gesteuert. For example:

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

Wenn FIPS aktiviert ist, wird das Attribut **SSLFipsRequired** der Zeilengruppe hinzugefügt, wobei der Wert auf `Yes`gesetzt ist:

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

Wenn der Container in einem OpenShift -Cluster ohne FIPS-Unterstützung ausgeführt wird, ist die FIPS-Unterstützung für den Warteschlangenmanager, den IBM MQ -Web-Server und native Hochverfügbarkeitskomponenten nicht automatisch aktiviert. Nur die x86-64 -Architektur wird derzeit von der OpenShift -Plattform für FIPS unterstützt. Für Power -und Linux for IBM Z -Architekturen bietet OpenShift keine FIPS-Unterstützung. Um die FIPS-Unterstützung in den IBM MQ -Komponenten für diese Architekturen explizit zu aktivieren, setzen Sie die Umgebungsvariable *MQ_ENABLE_FIPS* in der YAML-Datei des Warteschlangenmanagers auf `true` . Das folgende YAML-Snippet beschreibt die Verwendung der Umgebungsvariablen *MQ_ENABLE_FIPS* :

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

Automatischen FIPS-Modus für IBM MQ in Containern überschreiben

Verwenden Sie die Umgebungsvariable *MQ_ENABLE_FIPS* , um den FIPS-Modus für die IBM MQ -Komponenten im Container explizit zu aktivieren oder zu inaktivieren.

Vorbereitende Schritte

Anmerkung: Unter AIX, Linux, and Windowsstellt IBM MQ die Konformität mit FIPS 140-2 über das Verschlüsselungsmodul IBM Crypto for C (ICC) bereit. Das Zertifikat für dieses Modul wurde in den Langzeitstatus versetzt. Kunden sollten das IBM Crypto for C (ICC) -Zertifikat anzeigen und sich über alle Empfehlungen von NIST im Klaren sein. Ein Ersatz-FIPS 140-3-Modul ist derzeit in Bearbeitung und sein Status kann angezeigt werden, indem in der NIST-CMVP-Module in der Prozesslistenach ihm gesucht wird.

IBM MQ Operator 3.2.0 und das Container-Image des Warteschlangenmanagers ab 9.4.0.0 basieren auf UBI 9. Die Konformität mit FIPS 140-3 steht derzeit an und ihr Status kann angezeigt werden, indem Sie in der NIST CMVP-Module in der Prozesslistenach "Red Hat Enterprise Linux 9- OpenSSL FIPS Provider" suchen.

Informationen zu diesem Vorgang

MQ_ENABLE_FIPS unterstützt drei Werte:

auto
Dies ist der Standardwert.

Wenn das Hostbetriebssystem FIPS aktiviert ist, werden alle Komponenten (Warteschlangenmanager, IBM MQ -Web-Server und native HA) im FIPS-Modus ausgeführt.

Wenn das Hostbetriebssystem nicht FIPS-fähig ist, werden alle Komponenten nicht im FIPS-Modus ausgeführt.

true

Dieser Wert aktiviert FIPS für ausgewählte Komponenten im Container.

Das Warteschlangenmanagerattribut **SSLFIPS** wird auf YES gesetzt, auch wenn IBM MQ in Containern auf einem Hostbetriebssystem ausgeführt wird, das nicht FIPS-konform ist. Dies ist der Fall, wenn der IBM MQ -Warteschlangenmanager, der Web-Server und die native Hochverfügbarkeit FIPS-konform sind, das Betriebssystem des Containers jedoch nicht.

false

Dieser Wert inaktiviert die FIPS-Konformität.

Das Warteschlangenmanagerattribut **SSLFIPS** wird auf N0gesetzt, auch wenn IBM MQ in Containern auf einer FIPS-kompatiblen Hostmaschine ausgeführt wird. IBM MQ sichert jedoch weiterhin Verbindungen, wenn der private Schlüssel und Zertifikate bereitgestellt werden.

JVM-Optionen werden für den IBM MQ -Web-Server nicht aktualisiert. Der IBM MQ -Web-Server führt jedoch weiterhin einen HTTPS-Endpunkt aus, wenn der private Schlüssel und die Zertifikate bereitgestellt werden.

Die Datenreplikation in der nativen Hochverfügbarkeit verwendet keine FIPS-Verschlüsselung.

Beispiel

Die folgende YAML-Beispieldatei für Warteschlangenmanager beschreibt die Aktivierung von TLS und FIPS für die Warteschlangenmanager-Komponente:

```
apiVersion: mq.ibm.com/v1beta1
kind: QueueManager
metadata:
  namespace: ibm-mq-fips
  name: ibm-mq-qm-ppcle
spec:
  license:
    accept: true
    license: L-EHXT-MQCRN9
    use: Production
  queueManager:
    name: PPCLEQM
    storage:
      queueManager:
        type: ephemeral
  template:
    pod:
      containers:
        - env:
            - name: MQ_ENABLE_FIPS
              value: "true"
          name: qmgr
  version: 9.4.0.0-r1
  web:
    enabled: false
  pki:
    keys:
      - name: ibm-mq-tls-certs
        secret:
          secretName: ibm-mq-tls-secret
          items:
          - tls.key
          - tls.crt
```

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. Ab Red Hat OpenShift Container Platform 4.11gibt es einen Standardgrenzwert von 4096 Prozessen pro Container. 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.

Umgebung für IBM MQ in Containern vorbereiten, installieren und Upgrade durchführen

Sie führen eine Reihe von Tasks aus, um Ihre Umgebung für IBM MQ vorzubereiten.

Informationen zu diesem Vorgang

Wenn Sie IBM MQ Operatorverwenden, bereiten Sie Ihren Red Hat OpenShift Container Platform-Cluster durch Installation des Operators vor. Siehe „IBM MQ Operator installieren und Upgrade durchführen” auf Seite 32

Andernfalls bereiten Sie Ihre Containerumgebung vor, indem Sie eigene Container-Images erstellen. Siehe „IBM MQ durch Erstellung eines eigenen Container-Image vorbereiten” auf Seite 57

IBM MQ Operator installieren und Upgrade durchführen

Sie führen eine Reihe von Tasks aus, um IBM MQ Operatorzu installieren, zu deinstallieren und zu aktualisieren.

Informationen zu diesem Vorgang

Informationen zum Einstieg in die Installation und das Upgrade von IBM MQ Operator unter Red Hat OpenShift Container Platformfinden Sie in den folgenden Abschnitten.

Prozedur

- „Abhängigkeiten für IBM MQ Operator” auf Seite 32
- „Für IBM MQ Operator erforderliche Berechtigungen auf Clusterebene” auf Seite 32
- „Imagesignaturen überprüfen” auf Seite 33
- „IBM MQ Operator installieren” auf Seite 33
- „Upgrade für IBM MQ Operator und Warteschlangenmanager durchführen” auf Seite 44
- „IBM MQ Operator deinstallieren” auf Seite 55

OpenShift CP4I Abhängigkeiten für IBM MQ Operator

Bei der Installation von IBM MQ Operatorwerden keine anderen Operatoren automatisch installiert.

IBM Licensing Operator muss separat installiert werden, um die Lizenznutzung zu überwachen. Weitere Informationen finden Sie unter License Service in der Dokumentation zu IBM Cloud Pak for Integration .

Wenn Sie `QueueManager` mit einer IBM Cloud Pak for Integration -Lizenz erstellen, können Sie auswählen, ob Sie Single Sign-on mit der IBM Cloud Pak for Integration -Instanz von Keycloakverwenden möchten oder nicht. Die Verwendung von Keycloak ist standardmäßig mit einer IBM Cloud Pak for Integration -Lizenz aktiviert, aber wenn sie nicht installiert ist, wechselt `QueueManager` in den Status "Blockiert", bis die richtigen Abhängigkeiten installiert sind. Weitere Details zu den Abhängigkeiten finden Sie unter „IBM MQ Operator installieren” auf Seite 33 .

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`**
  - verbs: **`get, delete`**
- 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`**
  - verbs: **`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**
  - verbs: **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**
  - verbs: **get, list**

Anmerkung: Für IBM MQ Operator sind außerdem Berechtigungen im Geltungsbereich des Namensbereichs erforderlich. Wenn die IBM MQ Operator auf Clusterebene installiert wird, sind die Berechtigungen auf Namensbereichebene in allen Namensbereichen vorhanden.

OpenShift CP4I Imagesignaturen überprüfen

Container-Images für IBM MQ Operator und IBM MQ -Warteschlangenmanager werden digital signiert.

Informationen zu diesem Vorgang

Digitale Signaturen bieten Nutzern von Inhalten eine Möglichkeit, sicherzustellen, dass das, was sie herunterladen, sowohl authentisch ist (es stammt aus der erwarteten Quelle) als auch Integrität hat (es ist das, was wir erwarten).

Prozedur

- Überprüfen Sie die Signaturen der Container-Images für IBM MQ Operator und IBM MQ -Warteschlangenmanager:
  - Weitere Informationen finden Sie unter Imagesignaturen überprüfen in der Dokumentation zu IBM Cloud Pak for Integration (CP4I) 16.1.0 .

OpenShift CP4I IBM MQ Operator installieren

IBM MQ Operator kann mithilfe der OpenShift -Konsole oder der Befehlszeilenschnittstelle (CLI) unter Red Hat OpenShift installiert werden.

Vorbereitende Schritte

Wichtig:

- In diesem Abschnitt wird die Installation von IBM MQ Operator zur eigenständigen Verwendung **ausschließlich**beschrieben. Wenn Sie IBM Cloud Pak for Integrationoder Keycloak -SSO für einen oder mehrere Warteschlangenmanager verwenden möchten, lesen Sie den Abschnitt „IBM MQ Operator zur Verwendung mit CP4I installieren” auf Seite 41.
- Lesen Sie die Anleitung zur Strukturierung Ihrer Implementierung , bevor Sie IBM MQ Operatorinstallieren.

Um sicherzustellen, dass Ihre Installation so reibungslos wie möglich verläuft, müssen Sie alle Voraussetzungen und Voraussetzungen kennen, bevor Sie mit der Installation beginnen. Weitere Informationen finden Sie unter „IBM MQ in Containern planen” auf Seite 7.

Informationen zu diesem Vorgang

Die folgenden Schritte stellen den typischen Aufgabenablauf für die Installation von IBM MQ Operatordar:

1. Installieren Sie Red Hat OpenShift Container Platform.
2. Speicher konfigurieren

3. Spiegelimages (nur Air-Gap).
4. Fügen Sie den IBM MQ Operator -Katalog hinzu.
5. Installieren Sie IBM MQ Operator.
6. Geheimen Schlüssel für Berechtigungsschlüssel erstellen (nur Onlineinstallationen).
7. Implementieren Sie den License Service.
8. Implementieren Sie einen Warteschlangenmanager.

Vorgehensweise

1. Installieren Sie Red Hat OpenShift Container Platform.

   Ausführliche Schritte für die Installation von OpenShiftfinden Sie unter Red Hat -Software installieren 4.6 oder höher.

   Wichtig: Stellen Sie sicher, dass Sie eine unterstützte Version von OpenShift Container Platforminstallieren. Wenn Sie beispielsweise IBM MQ Operator 3.2 oder höher verwenden möchten, müssen Sie OpenShift Container Platform 4.12 oder höher installieren. Weitere Informationen finden Sie unter IBM Cloud Pak and Red Hat OpenShift Container Platform compatibility.

   Für alle Schritte, die die Red Hat OpenShift Container Platform -CLI verwenden, müssen Sie bei Ihrem OpenShift -Cluster mit `oc login`angemeldet sein. Informationen zur Installation der CLI finden Sie unter Einführung in die OpenShift -CLI.

   Nach der Installation von OpenShiftkönnen Sie mithilfe des IBM -Berechtigungsschlüssels, den Sie in Geheimen Schlüssel für Berechtigungsschlüssel erstellenerstellen, den Zugriff auf Ihre Container-Software überprüfen und erhalten.
2. Konfigurieren Sie Speicher.

   Sie müssen Speicherklassen in Red Hat OpenShift Container Platform definieren und Ihre Speicherkonfiguration festlegen, um Ihre Dimensionierungsanforderungen zu erfüllen.

   Wichtig: IBM MQ -Einzelinstanz-und native HA-Warteschlangenmanager können den RWO-Zugriffsmodus verwenden, während Multi-Instanz-Warteschlangenmanager RWX benötigen, wie in „Speicher für IBM MQ Operator planen” auf Seite 16beschrieben. IBM MQ -Warteschlangenmanager mit mehreren Instanzen erfordern bestimmte Dateisystemmerkmale, die anhand der Anweisungen im Abschnitt Gemeinsam genutztes Dateisystem für IBM MQüberprüft werden können.

   Eine Liste bekannter konformer und nicht konformer Dateisysteme sowie Anmerkungen zu anderen Grenzwerten oder Einschränkungen finden Sie in der Testanweisung für IBM MQ -Dateisysteme.

   Empfohlene Speicheranbieter finden Sie auf der Seite CP4I Speicheraspekte .
3. Spiegelbilder (nur Luftspalt).

   Wenn sich Ihr Cluster in einer eingeschränkten (durch eine Air-Gap geschützten) Netzumgebung befindet, müssen Sie die IBM MQ -Images mit den folgenden Werten spiegeln:

   ```
   export OPERATOR_PACKAGE_NAME=ibm-mq
   export OPERATOR_VERSION=3.2.1
   ```

   Informationen zum Erstellen von Spiegelimages finden Sie unter Spiegelimages für einen durch eine Air-Gap geschützten Cluster.
4. Fügen Sie die IBM MQ Operator -Katalogquelle hinzu.

   Fügen Sie die Katalogquelle hinzu, die die Operatoren für Ihren Cluster verfügbar macht. Weitere Informationen finden Sie unter „IBM MQ Operator -Katalogquelle hinzufügen” auf Seite 35.
5. Installieren Sie IBM MQ Operator.

   Wählen Sie eine der folgenden Optionen aus (verwenden Sie die Konsole oder die Befehlszeilenschnittstelle):

   - Option 1: Installieren Sie IBM MQ Operator über die OpenShift -Konsole.

- Option 2: Installieren Sie IBM MQ Operator über die OpenShift -CLI.

6. Geheimen Schlüssel für Berechtigungsschlüssel erstellen (nur Onlineinstallationen).

   IBM MQ Operator implementiert Warteschlangenmanager-Images, die aus einer Container-Registry extrahiert werden, die eine Lizenzberechtigungsprüfung 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 in dem Namensbereich haben, in dem Sie Warteschlangenmanager installieren möchten, befolgen Sie diese Anweisungen, um einen Berechtigungsschlüssel abzurufen und einen geheimen Pull-Schlüssel zu erstellen.

   Anmerkung: Der Berechtigungsschlüssel ist nicht erforderlich, wenn nur IBM MQ Advanced for Developers -Warteschlangenmanager (ohne Gewährleistung) implementiert werden.

   Sie können den geheimen Schlüssel für den Berechtigungsschlüssel entweder über die OpenShift -Konsole oder über die CLI erstellen. Im folgenden Beispiel wird die Befehlszeilenschnittstelle verwendet:

   a. Rufen Sie den Berechtigungsschlüssel ab, der Ihrer IBM ID zugeordnet ist. 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.

   c. Führen Sie über die OpenShift -CLI den folgenden Befehl aus, um einen geheimen Schlüssel für Image-Pull-Operationen namens `ibm-entitlement-key`zu erstellen.

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

   Dabei ist *entitlement_key* der Berechtigungsschlüssel, den Sie in Schritt b kopiert haben, *user_email* ist die IBM -ID, die der berechtigten Software zugeordnet ist, und *namespace* ist der Namensbereich, in dem Sie IBM MQ Operator installiert haben.

7. Implementieren Sie den License Service.

   Dies ist für die Überwachung der Lizenznutzung von Warteschlangenmanagern erforderlich. Befolgen Sie die Anweisungen unter License Service.

8. Implementieren Sie einen Warteschlangenmanager.

   Anweisungen zur Implementierung eines Beispielwarteschlangenmanagers für den "Schnelleinstieg" finden Sie im Abschnitt „Einfachen Warteschlangenmanager mit IBM MQ Operator implementieren” auf Seite 66.

Zugehörige Tasks

„IBM MQ Operator deinstallieren” auf Seite 55
Sie können die Red Hat OpenShift -Konsole oder CLI verwenden, um IBM MQ Operator aus Red Hat OpenShiftzu deinstallieren.

OpenShift *IBM MQ Operator -Katalogquelle hinzufügen*

Fügen Sie die IBM MQ Operator -Katalogquelle zu Ihrem OpenShift -Cluster hinzu, um die IBM MQ Operator für die Installation verfügbar zu machen. Diese Task ist auch erforderlich, wenn Sie Fixpacks für Katalogquellen anwenden, bevor Sie ein Upgrade durchführen.

Informationen zu diesem Vorgang

Ein Operatorkatalog ist ein Index der Operatoren, die verfügbar sind, um die API eines Red Hat OpenShift Container Platform -Clusters zu erweitern und IBM Softwareprodukte zu ermöglichen.

Die folgenden Katalogquellen sind verfügbar:

Option 1: Bestimmte Katalogquelle für IBM MQ Operator.

Durch die Verwendung einer bestimmten IBM MQ Operator -Katalogquelle erhalten Sie volle Kontrolle über die Softwareversionen in einem Cluster und bei Upgrades. Eine neue IBM MQ Operator -Version wird **erst** nach der Aktualisierung der Katalogquelle in einem OpenShift -Cluster verfügbar. Dieser Prozess ermöglicht Ihnen die manuelle Steuerung von Upgrades, sodass Sie die Option `Manuell` für die Einstellung **`Update approval`** für Operatoren nicht verwenden müssen. Die Option **Manuell** erzwingt die gleichzeitige Durchführung aller möglichen Upgrades und kann Upgrades blockieren. Verwenden Sie daher nur die Option **Automatisch** . Weitere Informationen finden Sie im Abschnitt "Automatische Aktualisierungen mit einer Genehmigungsstrategie einschränken" unter Operatoren mithilfe der Red Hat OpenShift -Konsole installieren.

Wählen Sie diese Option aus, wenn Sie ein Upgrade durchführen und die IBM MQ Operator -Katalogquelle einer neueren Version hinzufügen müssen.

Wenn Sie diese Option verwenden möchten, fahren Sie mit Option 1: Bestimmte Katalogquellen für IBM MQ Operatorhinzufügenfort.

Option 2: IBM Operator Catalog.

Mit dieser Option werden neue Operatorversionen verfügbar und **ohne** einen Eingriff von Ihnen angewendet. Verwenden Sie daher diese Option **nur** für Onlineinstallationen, bei denen Sie **automatische** Upgrades von IBM MQ Operatorwünschen und keine deterministischen Installationen erforderlich sind.

Anmerkung: Diese Option kann für Umgebungen mit Machbarkeitsnachweis nützlich sein, ist jedoch **nicht für Produktionsumgebungen geeignet**.

Um diese Option zu verwenden, fahren Sie mit Option 2: IBM -Operatorkatalog hinzufügenfort.

Prozedur

- **Option 1: Fügen Sie bestimmte Katalogquellen für IBM MQ Operatorhinzu.**

  Diese Task setzt voraus, dass Sie die ersten drei Schritte von „IBM MQ Operator installieren” auf Seite 33abgeschlossen haben.

  Diese Task muss von einem Clusteradministrator und über die Befehlszeilenschnittstelle ausgeführt werden.

  a) Nur Upgrade: Wenn Sie Fixpacks für Katalogquellen vor einem Upgrade anwenden, führen Sie die folgenden Schritte aus:

  – Vergewissern Sie sich, dass Ihre Operatoren ordnungsgemäß ausgeführt werden.
  – Wenn anstehende IBM MQ Operator -Aktualisierungen vorliegen, die eine manuelle Genehmigung erfordern, genehmigen Sie diese, bevor Sie mit dieser Prozedur beginnen. Weitere Informationen finden Sie im Abschnitt "Automatische Aktualisierungen mit einer Genehmigungsstrategie einschränken" unter Operatoren mithilfe der Red Hat OpenShift -Konsole installieren.

  b) Wenn Sie es noch nicht installiert haben oder eine Aktualisierung erforderlich ist, laden Sie das IBM Catalog Management-Plug-in (Version 1.6.0 oder höher) von GitHubherunter.

  Mit diesem Plug-in können Sie **`oc ibm-pak`** -Befehle für den Cluster ausführen.

  c) Melden Sie sich mit dem Befehl **`oc login`** und Ihren Benutzerberechtigungsnachweisen an Ihrem Cluster an:

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

  d) Exportieren Sie die folgenden Umgebungsvariablen für IBM MQ Operator:

```
export OPERATOR_PACKAGE_NAME=ibm-mq
export OPERATOR_VERSION=3.2.1
export ARCH=ARCHITECTURE
```

Dabei steht *ARCHITECTURE* für die Architektur des Systems, auf dem Sie IBM MQ Operatorimplementieren, und hat den Wert `amd64`, `ppc64leoder` `s390x`.

Wichtig: Wenn Sie vom IBM -Operatorkatalog zur bestimmten Katalogquelle für IBM MQ Operatorwechseln, setzen Sie `OPERATOR_VERSION` auf die Version Ihrer Bereitstellung von IBM MQ Operator.

e) Laden Sie die Dateien für IBM MQ Operator herunter.

Anmerkung: Wenn Sie eine **durch eine Air-Gap geschützte** Installation durchführen, sollten Sie bereits über die Dateien verfügen, die Sie benötigen, nachdem Sie den Schritt "Spiegelimages" unter " IBM MQ Operatorinstallieren"ausgeführt haben. In diesem Fall können Sie mit Schritt „8” auf Seite 37 " IBM MQ Operator -Katalogquelle auf den Cluster anwenden" fortfahren.

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

f) Generieren Sie die für IBM MQ Operatorerforderliche Katalogquelle:

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

g) Optional: Generieren Sie die Katalogquellen und speichern Sie sie in einem anderen Verzeichnis.

a. Rufen Sie die Katalogquelle ab:

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

b. (Optional) Navigieren Sie zum Verzeichnis in Ihrem Dateibrowser, um diese Artefakte in Dateien zu kopieren, die Sie zur Wiederverwendung oder für Pipelines behalten können.

h) Wenden Sie die IBM MQ Operator -Katalogquelle auf den Cluster an.

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

i) Bestätigen Sie, dass die IBM MQ Operator -Katalogquelle im Namensbereich `openshift-marketplace` erstellt wurde:

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

Beispielausgabe:

```
oc get catalogsource -n openshift-marketplace

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

Sie können jetzt Schritt 5 der Installation von IBM MQ Operatorausführen.

- **Option 2: IBM -Operatorkatalog hinzufügen.**

Wichtig: Verwenden Sie den IBM Operatorkatalog **nur** für Onlineinstallationen, bei denen Sie **automatische** Upgrades von IBM MQ Operatorwünschen und keine deterministischen Installationen erforderlich sind. Diese Option kann für Umgebungen mit Machbarkeitsnachweis nützlich sein, ist jedoch **nicht für Produktionsumgebungen geeignet**.

Der IBM -Operatorkatalog ist ein Index von Operatoren, die verfügbar sind, um die API eines Red Hat OpenShift Container Platform -Clusters zu erweitern und IBM Softwareprodukte zu aktivieren. Durch das Hinzufügen der Katalogquellen zum OpenShift -Cluster werden die IBM -Operatoren zur Liste der Operatoren hinzugefügt, die Sie installieren können.

Diese Task setzt voraus, dass Sie die ersten drei Schritte von „IBM MQ Operator installieren” auf Seite 33abgeschlossen haben.

Diese Task kann über die Befehlszeilenschnittstelle oder über die OpenShift -Webkonsole ausgeführt werden.

Verwendung der Befehlszeilenschnittstelle

1. Kopieren Sie die folgende Ressourcendefinition für IBM -Operatoren in eine lokale Datei auf Ihrem Computer:

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

2. Führen Sie den folgenden Befehl aus. Ersetzen Sie *filename*.yaml durch den Namen der Datei, die Sie im vorherigen Schritt erstellt haben:

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

OpenShift -Webkonsole verwenden

1. Melden Sie sich an der OpenShift -Webkonsole mit den Berechtigungsnachweisen Ihres OpenShift -Clusteradministrators an.
2. Klicken Sie im Banner auf das Pluszeichen ("+") Symbol zum Öffnen des Dialogfensters **YAML importieren** .

   Anmerkung: Sie müssen keinen Wert für **Projekt**auswählen. Der YAML-Code im nächsten Schritt enthält bereits den richtigen Wert für `metadata:namespace`, der sicherstellt, dass die Katalogquelle im richtigen Projekt (Namensbereich) installiert wird.
3. Fügen Sie die 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
```

4. Klicken Sie auf **Erstellen**.

Sie können jetzt Schritt 5 der Installation von IBM MQ Operatorausführen.

OpenShift CP4I *IBM MQ Operator mithilfe der OpenShift -Konsole installieren*

Die IBM MQ Operator kann unter Red Hat OpenShift mithilfe des OperatorHubinstalliert werden.

Vorbereitende Schritte

Diese Task setzt voraus, dass Sie die Schritte 1 bis 4 in „IBM MQ Operator installieren” auf Seite 33ausgeführt haben.

Vorgehensweise

1. Melden Sie sich an der Red Hat OpenShift-Clusterkonsole an.
2. Klicken Sie im Navigationsfenster auf **Operators** > **OperatorHub**.

   Die Seite 'OperatorHub' wird angezeigt.
3. Geben Sie 'IBM MQ' im Feld **All Items** (Alle Elemente) ein.

Der IBM MQ-Katalogeintrag wird angezeigt.

4. Wählen Sie **IBM MQ** aus.

   Das Fenster 'IBM MQ' wird angezeigt.

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

   Die Seite 'Operator installieren' wird angezeigt.

6. Geben Sie die folgenden Werte ein:

   a) Setzen Sie **Kanal** auf die von Ihnen ausgewählte Version.

   Bestimmen Sie anhand der Informationen im Abschnitt „Versionsunterstützung für IBM MQ Operator” auf Seite 14, welcher Operatorkanal ausgewählt werden soll.

   b) Setzen Sie **Installationsmodus** entweder auf einen bestimmten Namensbereich im Cluster (den Sie im nächsten Schritt erstellen können) oder auf den clusterweiten Geltungsbereich.

   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.

   c) Optional: Wenn Sie "einen bestimmten Namensbereich im Cluster" ausgewählt haben, setzen Sie den **Namensbereich** auf den Projektwert (Namensbereich), in dem Sie den Operator installieren möchten.

   Anmerkung: Wenn Sie die Konsole verwenden, um den Operator zu installieren, können Sie entweder einen vorhandenen Namensbereich oder den vom Operator bereitgestellten Standardnamensbereich verwenden oder einen neuen Namensbereich erstellen. Wenn Sie einen neuen Namensbereich erstellen möchten, erstellen Sie ihn wie folgt über dieses Formular: Klicken Sie im Navigationsfenster auf **Home** > **Projekte**, wählen Sie **Projekt erstellen**aus, geben Sie den **Namen** des Projekts (des Namensbereichs) an, das Sie erstellen möchten, und klicken Sie anschließend auf **Erstellen**.

   d) Setzen Sie **Genehmigungsstrategie** auf "Automatisch".

7. Klicken Sie auf **Installieren** und warten Sie, bis Ihr Operator installiert ist.

   Sie erhalten nach Abschluss der Installation eine Bestätigung.

   Navigieren Sie zur Überprüfung der Installation zu **Operatoren** > **Installierte Operatoren**und wählen Sie Ihr Projekt in der Dropdown-Liste **Projekte** aus. Der Status des Operators ändert sich nach Abschluss der Installation in 'Erfolgreich'.

Nächste Schritte

Sie können jetzt den geheimen Schlüssel für Berechtigungsschlüssel erstellen (Schritt 6 in „IBM MQ Operator installieren” auf Seite 33).

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

IBM MQ Operator kann über die Befehlszeilenschnittstelle (CLI) unter Red Hat OpenShift installiert werden.

Vorbereitende Schritte

Diese Task setzt voraus, dass Sie die Schritte 1 bis 4 in „IBM MQ Operator installieren” auf Seite 33ausgeführt haben.

Vorgehensweise

1. Melden Sie sich mit **`oc login`**an der Befehlszeilenschnittstelle (CLI) von Red Hat OpenShift an.
2. Optional: Erstellen Sie einen Namensbereich, der für IBM MQ Operatorverwendet werden soll.

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

Führen Sie den folgenden Befehl aus, um einen neuen Namensbereich in der Befehlszeilenschnittstelle zu erstellen:

```
oc create namespace namespace_name
```

Dabei ist *Namensbereichsname* der Name des zu erstellenden Namensbereichs.

3. Zeigen Sie die Liste der Operatoren an, die für den Cluster über OperatorHubverfügbar sind:

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

4. Überprüfen Sie die IBM MQ Operator , um die unterstützte **InstallModes** und die verfügbare **Channels**zu überprüfen.

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

5. Optional: Erstellen Sie eine **OperatorGroup**.

   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, den Modus `AllNamespaces` verwendet, verfügt der Namensbereich `openshift-operators` bereits über eine entsprechende **OperatorGroup** , und Sie können diesen Schritt überspringen.

   Wenn der Operator den Modus `SingleNamespace` verwendet und Sie noch nicht über eine entsprechende **OperatorGroup** verfügen, erstellen Sie eine solche mit dem folgenden Befehl:

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

6. Bestimmen Sie anhand der Informationen im Abschnitt „Versionsunterstützung für IBM MQ Operator” auf Seite 14, welcher Operatorkanal ausgewählt werden soll.
7. Installieren Sie den Operator.

   Verwenden Sie den folgenden Befehl, indem Sie *ibm-mq-operator-channel* so ändern, dass er dem Kanal für die Version des IBM MQ -Operators entspricht, den Sie installieren möchten, und *namespace_name* in **openshift-operators** ändern, wenn Sie den Modus "AllNamespaces" verwenden, oder in den Namensbereich, für den Sie den IBM MQ -Operator bereitstellen möchten, wenn Sie den Modus "SingleNamespace" verwenden.

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

8. Nach einigen Minuten wird der Operator installiert. Führen Sie den folgenden Befehl aus, um zu überprüfen, ob sich alle Komponenten im Status 'Erfolgreich' befinden:

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

Dabei steht *Namensbereichsname* für **openshift-operators** , wenn Sie den ModusAllNamespaces- verwenden, oder für den Projektnamen (Namensbereich), wenn Sie den Modus "SingleNamespace" verwenden.

Nächste Schritte

Sie können jetzt den geheimen Schlüssel für Berechtigungsschlüssel erstellen (Schritt 6 in „IBM MQ Operator installieren” auf Seite 33).

 IBM MQ Operator zur Verwendung mit CP4I installieren

Zur Verwendung mit IBM Cloud Pak for Integration (CP4I) kann IBM MQ Operator über die OpenShift -Konsole oder Befehlszeilenschnittstelle (CLI) in Red Hat OpenShift installiert werden.

Vorbereitende Schritte

Wichtig:

- In diesem Abschnitt wird die Installation von IBM MQ Operator für die Verwendung mit CP4Ibeschrieben oder wenn Sie mindestens einen Ihrer Warteschlangenmanager mit einer CP4I Lizenz **nur**implementieren möchten. Anweisungen zur Installation von IBM MQ Operator für die eigenständige Verwendung finden Sie in „IBM MQ Operator installieren” auf Seite 33.
- Lesen Sie die Anleitung zur Strukturierung Ihrer Implementierung , bevor Sie IBM MQ Operatorinstallieren.

Um sicherzustellen, dass Ihre Installation so reibungslos wie möglich verläuft, müssen Sie alle Voraussetzungen und Voraussetzungen kennen, bevor Sie mit der Installation beginnen. Weitere Informationen finden Sie unter „IBM MQ in Containern planen” auf Seite 7.

Informationen zu diesem Vorgang

Die folgenden Schritte stellen den typischen Aufgabenablauf für die Installation von IBM MQ Operatordar:

1. Installieren Sie Red Hat OpenShift Container Platform.
2. Speicher konfigurieren
3. Spiegelimages (nur Air-Gap).
4. Fügen Sie den IBM MQ Operator -Katalog hinzu und bereiten Sie Ihren Cluster vor.
5. Installieren Sie IBM MQ Operator.
6. Geheimen Schlüssel für Berechtigungsschlüssel erstellen (nur Onlineinstallationen).
7. Optional: Installieren Sie IBM Cloud Pak for Integration (CP4I) und seine Abhängigkeiten.
8. Implementieren Sie den License Service.
9. Implementieren Sie einen Warteschlangenmanager.

Vorgehensweise

1. Installieren Sie Red Hat OpenShift Container Platform.

   Ausführliche Informationen zur Installation von OpenShiftfinden Sie unter Red Hat -Software installieren 4.6 oder höher.

   Wichtig: Stellen Sie sicher, dass Sie eine unterstützte Version von OpenShift Container Platforminstallieren. Wenn Sie beispielsweise IBM MQ Operator 3.2 oder höher verwenden möchten, müssen Sie OpenShift Container Platform 4.12 oder höher installieren. Weitere Informationen finden Sie unter IBM Cloud Pak and Red Hat OpenShift Container Platform compatibility.

Für alle Schritte, die die Red Hat OpenShift Container Platform -CLI verwenden, müssen Sie bei Ihrem OpenShift -Cluster mit `oc login`angemeldet sein. Informationen zur Installation der Befehlszeilenschnittstelle finden Sie unter Einführung in die Befehlszeilenschnittstelle von OpenShift.

Nach der Installation von OpenShiftkönnen Sie mithilfe des Berechtigungsschlüssels IBM , den Sie in Geheimen Schlüssel für Berechtigungsschlüssel erstellenerstellen, den Zugriff auf Ihre Container-Software überprüfen und erhalten.

2. Konfigurieren Sie Speicher.

   Sie müssen Speicherklassen in Red Hat OpenShift Container Platform definieren und Ihre Speicherkonfiguration festlegen, um Ihre Dimensionierungsanforderungen zu erfüllen.

   Wichtig: IBM MQ -Einzelinstanz-und native HA-Warteschlangenmanager können den RWO-Zugriffsmodus verwenden, während Multi-Instanz-Warteschlangenmanager RWX benötigen, wie in „Speicher für IBM MQ Operator planen” auf Seite 16beschrieben. IBM MQ -Warteschlangenmanager mit mehreren Instanzen erfordern bestimmte Dateisystemmerkmale, die anhand der Anweisungen unter Gemeinsam genutztes Dateisystem für IBM MQüberprüft werden können.

   Eine Liste der bekannten konformen und nicht konformen Dateisysteme sowie Hinweise zu anderen Grenzwerten oder Einschränkungen finden Sie in der Testanweisung für IBM MQ -Dateisysteme.

   Empfohlene Speicheranbieter finden Sie auf der Seite CP4I Speicheraspekte .

3. Spiegelbilder (nur Luftspalt).

   Wenn sich Ihr Cluster in einer eingeschränkten (durch eine Air-Gap geschützten) Netzumgebung befindet, müssen Sie die IBM MQ -Images spiegeln. Je nach Konfiguration müssen Sie möglicherweise auch einige zusätzliche Komponenten spiegeln. Lesen Sie die folgenden Informationen und spiegeln Sie dann die Images nach Bedarf.

   - Sie müssen die IBM MQ -Images spiegeln. Verwenden Sie die folgenden Werte:

     ```
     export OPERATOR_PACKAGE_NAME=ibm-mq
     export OPERATOR_VERSION=3.2.1
     ```

   - Sie müssen auch einige zusätzliche erforderliche Komponenten spiegeln, wenn Sie mindestens einen Warteschlangenmanager implementieren möchten, für den **alle** der folgenden Aussagen zutreffen:
     - Sie verwenden eine CP4I -Lizenz.
     - Die IBM MQ Console ist aktiviert.
     - Sie verwenden den IBM Cloud Pak for Integration Keycloak -Service für die IBM MQ Console -SSO-Authentifizierung und -Autorisierung (Standardeinstellung).

     Wenn alle vorherigen Aussagen zutreffen, wird SSO von Keycloakbereitgestellt. Daher müssen Sie neben der IBM MQ Operator -Katalogquelle auch die Schritte für jede dieser zusätzlich erforderlichen Komponenten wiederholen:

     - IBM Cloud Pak foundational services
     - IBM Cloud Pak for Integration
     - Keycloak (OperatorRed Hat OpenShift)

   Informationen zum Erstellen von Spiegelimages finden Sie unter Spiegelimages für einen durch eine Air-Gap geschützten Cluster.

4. Fügen Sie die IBM MQ Operator -Katalogquelle hinzu.

   Fügen Sie die Katalogquelle hinzu, die IBM MQ Operator für Ihren Cluster verfügbar macht, indem Sie die folgenden Werte verwenden:

   ```
   export OPERATOR_PACKAGE_NAME=ibm-mq
   export OPERATOR_VERSION=3.2.1
   export ARCH=ARCHITECTURE
   ```

   Dabei steht *ARCHITECTURE* für Ihre Systemarchitektur und hat den Wert `amd64`, `ppc64le`oder `s390x`.

Es gibt einige zusätzliche erforderliche Komponenten, wenn Sie mindestens einen Warteschlangenmanager implementieren, bei dem **alle** der folgenden Aussagen zutreffen:

- Sie verwenden eine CP4I -Lizenz.
- Die IBM MQ Console ist aktiviert.
- Sie verwenden den IBM Cloud Pak for Integration Keycloak -Service für die IBM MQ Console -SSO-Authentifizierung und -Autorisierung (Standardeinstellung).

Wenn alle vorherigen Aussagen zutreffen, wird SSO von Keycloakbereitgestellt. Daher müssen Sie neben der IBM MQ Operator -Katalogquelle auch die Schritte für jede dieser zusätzlich erforderlichen Komponenten wiederholen:

- IBM Cloud Pak foundational services
- IBM Cloud Pak for Integration
- Keycloak (OperatorRed Hat OpenShift)

Befolgen Sie die Schritte für Ihre erforderlichen Katalogquellen unter Katalogquellen zu einem Cluster hinzufügen.

5. Installieren Sie IBM MQ Operator.

   Wählen Sie eine der beiden folgenden Optionen aus (verwenden Sie die Konsole oder die Befehlszeilenschnittstelle):

   - Option 1: Installieren Sie IBM MQ Operator über die OpenShift -Konsole.
   - Option 2: Installieren Sie IBM MQ Operator über die OpenShift -CLI.

6. Erstellen Sie den geheimen Schlüssel des Berechtigungsschlüssels (nur Onlineinstallationen).

   IBM MQ Operator implementiert Warteschlangenmanager-Images, die aus einer Container-Registry extrahiert werden, die eine Lizenzberechtigungsprüfung 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 in dem Namensbereich haben, in dem Sie Warteschlangenmanager installieren möchten, befolgen Sie diese Anweisungen, um einen Berechtigungsschlüssel abzurufen und einen geheimen Pull-Schlüssel zu erstellen.

   Anmerkung: Der Berechtigungsschlüssel ist nicht erforderlich, wenn nur IBM MQ Advanced for Developers -Warteschlangenmanager (ohne Gewährleistung) implementiert werden.

   Sie können den geheimen Schlüssel für den Berechtigungsschlüssel entweder über die OpenShift -Konsole oder über die CLI erstellen. Im folgenden Beispiel wird die Befehlszeilenschnittstelle verwendet:

   a. Rufen Sie den Berechtigungsschlüssel ab, der Ihrer IBM ID zugeordnet ist. 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.
   c. Führen Sie in der OpenShift -CLI den folgenden Befehl aus, um einen geheimen Schlüssel für Image-Pull-Operationen namens `ibm-entitlement-key`zu erstellen.

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

      Dabei ist *berechtigungsschlüssel* der Berechtigungsschlüssel, den Sie in Schritt b kopiert haben, *benutzer-email>* ist die IBM -ID, die der berechtigten Software zugeordnet ist, und *namensbereich* ist der Namensbereich, in dem Sie IBM MQ Operator installiert haben.

7. Optional: Installieren Sie CP4I und seine Abhängigkeiten.

Es gibt einige zusätzliche erforderliche Komponenten, wenn Sie mindestens einen Warteschlangenmanager implementieren, bei dem **alle** der folgenden Aussagen zutreffen:

- Sie verwenden eine CP4I -Lizenz.
- Die IBM MQ Console ist aktiviert.
- Sie verwenden den CP4I Keycloak -Service für die IBM MQ Console -SSO-Authentifizierung und -Autorisierung (Standardeinstellung).

Wenn alle vorherigen Aussagen zutreffen, wird SSO von Keycloak bereitgestellt und Sie müssen die folgenden zusätzlichen Schritte ausführen:

- Installieren Sie den IBM Cloud Pak foundational services -Operator in demselben Installationsmodus wie CP4I Operator. Informationen zu unterstützten Versionen finden Sie unter Operatorkanalversionen für dieses Release.
- Installieren Sie den CP4I -Operator.
- Optional: Implementieren Sie die Plattformbenutzerschnittstelle.

  a. Erstellen Sie den Namensbereich `ibm-common-services` . Führen Sie den folgenden Befehl aus, wenn Sie über die CLI bei Ihrem OpenShift -Cluster angemeldet sind:

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

  b. Implementieren Sie die Plattformbenutzerschnittstelle.

8. Implementieren Sie den License Service.

   Dies ist für die Überwachung der Lizenznutzung von Warteschlangenmanagern erforderlich. Befolgen Sie die Anweisungen unter License Service.

9. Implementieren Sie einen Warteschlangenmanager.

   Anweisungen zur Implementierung eines Beispielwarteschlangenmanagers für den "Schnelleinstieg" finden Sie in „Einfachen Warteschlangenmanager mit IBM MQ Operator implementieren” auf Seite 66.

Zugehörige Tasks

„IBM MQ Operator deinstallieren” auf Seite 55
Sie können die Red Hat OpenShift -Konsole oder CLI verwenden, um IBM MQ Operator aus Red Hat OpenShiftzu deinstallieren.

OpenShift Operator2.0.0 CP4I Upgrade für IBM MQ Operator und Warteschlangenmanager durchführen

Es gibt verschiedene Upgradeprozesse für Benutzer der IBM MQ Operator, je nachdem, ob Sie IBM MQ -Lizenzen oder IBM Cloud Pak for Integration -Lizenzen (CP4I) verwenden. Führen Sie den Upgradeschritt für Ihren Implementierungstyp aus.

Informationen zu diesem Vorgang

Führen Sie einen der folgenden Schritte aus, um ein Upgrade für Ihre IBM MQ Operator und Warteschlangenmanager durchzuführen:

Prozedur

- Option 1: **Führen Sie ein Upgrade von Bereitstellungen auf die neueste Version im aktuellen Operatorkanal durch**

  Informationen zum Upgrade von Bereitstellungen von IBM MQ Operator auf die neueste Version in Ihrem aktuellen Operatorkanal finden Sie unter „Upgrade auf das neueste Sicherheitsrelease eines IBM MQ Operator -Kanals durchführen” auf Seite 45.
- Option 2: **Führen Sie ein Upgrade für IBM MQ Operator for IBM MQ -Lizenzen durch** .

Informationen zum Upgrade von Implementierungen von IBM MQ Operator , bei denen **nur** IBM MQ -Lizenzen verwendet werden, finden Sie unter „Upgrade der IBM MQ Operator durchführen” auf Seite 45.

- Option 3: **Führen Sie ein Upgrade für IBM MQ Operator für CP4I -Benutzer durch** .

  Upgrade-Implementierungen von IBM MQ Operator für Benutzer von IBM Cloud Pak for Integration. Dies gilt auch, wenn Sie mindestens einen Warteschlangenmanager unter einer CP4I -Lizenz implementiert haben. Siehe „Upgrade für IBM MQ Operator für CP4I -Benutzer durchführen” auf Seite 50.

OpenShift *Upgrade der IBM MQ Operator durchführen*

Upgrade-Implementierungen von IBM MQ Operator , bei denen **nur** IBM MQ -Lizenzen verwendet werden.

Vorbereitende Schritte

Wichtig: Diese Task ist für Benutzer der Lizenzen IBM MQ Operator und **nur** IBM MQ bestimmt. Wenn Sie ein Benutzer von IBM Cloud Pak for Integration (CP4I) sind oder mindestens einen Ihrer Warteschlangenmanager mit einer CP4I -Lizenz implementiert haben, lesen Sie den Abschnitt „Upgrade für IBM MQ Operator für CP4I -Benutzer durchführen” auf Seite 50.

Informationen zu diesem Vorgang

Führen Sie den Schritt aus, der dem erforderlichen Upgrade entspricht.

Anmerkung: Version 3.2.x von IBM MQ Operator wurde sowohl als CD als auch als SC2 -Release freigegeben.

Prozedur

- Option 1: „Upgrade auf das neueste Sicherheitsrelease eines IBM MQ Operator -Kanals durchführen” auf Seite 45
- Option 2: „Upgrade eines 2.0.x LTS IBM MQ Operator auf den Kanal 3.2.x SC2/CD durchführen” auf Seite 47
- Option 3: „Upgrade eines CD IBM MQ Operator auf den Kanal 3.2.x SC2/CD durchführen” auf Seite 48

OpenShift CP4I *Upgrade auf das neueste Sicherheitsrelease eines IBM MQ Operator -Kanals durchführen*

Das Upgrade von IBM MQ Operator ermöglicht Ihnen das Upgrade Ihrer Warteschlangenmanager.

Vorbereitende Schritte

Wichtig: In diesem Abschnitt wird beschrieben, wie Sie für Implementierungen von IBM MQ Operator ein Upgrade auf das neueste Sicherheitsrelease im Kanal der Implementierung durchführen. Wenn dies für Ihre Implementierung nicht zutrifft, verwenden Sie alternative Upgradepfade, die in „Upgrade für IBM MQ Operator und Warteschlangenmanager durchführen” auf Seite 44beschrieben sind.

Informationen zu diesem Vorgang

Zuerst führen Sie ein Upgrade für die Katalogquelle und anschließend für die Warteschlangenmanager durch. Es gibt zwei Optionen, je nachdem, welche Katalogquelle zum Implementieren der IBM MQ Operator -Instanz verwendet wird, für die das Upgrade durchgeführt wird.

Option 1: Bestimmte Katalogquelle für IBM MQ Operator

Eine neue IBM MQ Operator -Version wird **erst** nach der Aktualisierung der Katalogquelle in einem OpenShift -Cluster verfügbar. Dieser Prozess ermöglicht Ihnen die manuelle Steuerung von Upgrades, sodass Sie die Option **Manuell** für die Einstellung **`Update approval`** für Operatoren nicht verwenden müssen. Die Option **Manuell** erzwingt die gleichzeitige Durchführung aller möglichen Upgrades und kann Upgrades blockieren. Verwenden Sie daher nur die Option **Automatisch** . Weitere Informationen

finden Sie im Abschnitt "Automatische Aktualisierungen mit einer Genehmigungsstrategie einschränken" unter Operatoren mithilfe der Red Hat OpenShift -Konsole installieren.

Wenn Sie diese Option verwenden möchten, fahren Sie mit Upgrade mit der spezifischen Katalogquelle für IBM MQ Operatorfort.

Option 2: IBM -Operatorkatalog

Mit dieser Option werden neue Operatorversionen verfügbar und **ohne** einen Eingriff von Ihnen angewendet. Verwenden Sie diese Option also **nur** für Onlineinstallationen, bei denen Sie **automatische** Upgrades von IBM MQ Operatorwünschen und keine deterministischen Installationen erforderlich sind. Diese Option kann für Umgebungen mit Machbarkeitsnachweis nützlich sein, ist jedoch **nicht für Produktionsumgebungen geeignet**.

Wenn Sie diese Option verwenden möchten, fahren Sie mit Upgrade mit dem IBM -Operatorkatalogfort.

Informationen zum Wechsel von der Verwendung des IBM -Operatorkatalogs zur Verwendung der bestimmten Katalogquelle für IBM MQ Operator, die Ihnen eine größere Kontrolle über Upgrades ermöglicht, finden Sie in „In bestimmte Katalogquelle für IBM MQ Operator verschieben” auf Seite 49.

Prozedur

- **Upgrade mit der bestimmten Katalogquelle für IBM MQ Operatordurchführen**
  a) Wenden Sie die neueste Katalogquelle an.

  Befolgen Sie die Anweisungen unter "Bestimmte Katalogquellen für IBM MQ Operatorhinzufügen" unter " IBM MQ Operator -Katalogquelle hinzufügen".

  b) Wenn der Status **Genehmigung aktualisieren** für IBM MQ Operator auf **Automatisch**gesetzt ist, wird der Operator aktualisiert. Wenn Sie **Update approval** auf **Manual**gesetzt haben, führen Sie die folgenden Schritte aus, um ein Upgrade für IBM MQ Operatordurchzuführen:

  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 **Upgrade verfügbar** .

  e. Klicken Sie auf **Vorschau InstallPlan**

  f. Klicken Sie auf **Genehmigen** , um das Upgrade abzuschließen.

  Der Operator führt ein Upgrade auf die neue Version durch.

  c) Aktualisieren Sie alle IBM MQ -Warteschlangenmanager.

  Fahren Sie mit den Anweisungen im Abschnitt Upgrade für IBM MQ -Warteschlangenmanager durchführenfort.

- **Upgrade mit dem IBM -Operatorkatalog durchführen**
  a) Führen Sie ein Upgrade von IBM MQ Operator auf eine neuere Version durch.

  Wenn Sie automatische Upgrades festgelegt haben, führt Ihr IBM MQ Operator nach dem Release eines neuen Sicherheitsrelease ein Upgrade durch. Wenn keine automatischen Upgrades festgelegt sind, genehmigen Sie Ihr IBM MQ Operator -Upgrade manuell:

  – Wenn ein Upgrade verfügbar ist, lautet die **`Upgrade Status`** möglicherweise "`Upgrade verfügbar`".
  – In diesem Fall ist möglicherweise ein Steuerelement verfügbar, mit dem Sie den **InstallPlan** genehmigen können, der ein Upgrade für IBM MQ Operatordurchführt.

  b) Upgrade für IBM MQ -Warteschlangenmanager durchführen

Fahren Sie mit den Anweisungen im Abschnitt Upgrade für IBM MQ -Warteschlangenmanager durchführenfort.

- **Führen Sie ein Upgrade für IBM MQ -Warteschlangenmanager durch**.

  Nach dem Upgrade von IBM MQ Operatorsollten Sie alle IBM MQ -Warteschlangenmanager auf eine neuere Version aktualisieren.

  In der folgenden Tabelle wird die neueste Version des IBM MQ -Warteschlangenmanagers für jeden aktiven Operatorkanal beschrieben. Verwenden Sie die relevante Version und befolgen Sie die Prozedur in „Upgrade für einen IBM MQ -Warteschlangenmanager mit Red Hat OpenShift durchführen” auf Seite 53.

Operatorkanal	Letzter IBM MQ -Warteschlangenmanager
v3.2 (SC2/CD)	9.4.0.0-r1

Upgrade eines 2.0.x LTS IBM MQ Operator auf den Kanal 3.2.x SC2/CD durchführen

Das Upgrade von IBM MQ Operator ermöglicht Ihnen das Upgrade Ihrer Warteschlangenmanager.

Vorbereitende Schritte

Wichtig:

- Diese Task ist für Benutzer der Lizenzen IBM MQ Operator und **nur** IBM MQ bestimmt. Wenn Sie ein Benutzer von IBM Cloud Pak for Integration (CP4I) sind oder mindestens einen Ihrer Warteschlangenmanager mit einer CP4I -Lizenz implementiert haben, lesen Sie den Abschnitt „Upgrade für IBM MQ Operator für CP4I -Benutzer durchführen” auf Seite 50.
- In diesem Abschnitt wird das Upgrade von Bereitstellungen von 2.0.x Long Term Support (LTS) IBM MQ Operator auf den Kanal Support Cycle 2 (SC2) von IBM MQ Operator 3.2.x **nur**beschrieben. Wenn dies für Ihre Implementierung nicht zutrifft, sehen Sie sich die alternativen Upgradepfade an, die unter „Upgrade für IBM MQ Operator und Warteschlangenmanager durchführen” auf Seite 44beschrieben sind.

Für ein Upgrade auf IBM MQ Operator 3.2.1 muss Red Hat OpenShift Container Platform 4.12 oder höher ausgeführt werden. Informationen zur Überprüfung der kompatiblen Versionen für jeden IBM MQ Operator -Channel finden Sie unter „Kompatible Red Hat OpenShift Container Platform-Versionen” auf Seite 15. Informationen zum Upgrade der Plattform finden Sie unter Upgrade für Red Hat OpenShiftdurchführen.

Vorgehensweise

1. **Spiegelimages (nur Air Gap).**

   Sie müssen die IBM MQ -Images spiegeln. Führen Sie die Schritte unter dem folgenden Link aus und verwenden Sie dabei nur diese Werte:

   ```
   export OPERATOR_PACKAGE_NAME=ibm-mq
   export OPERATOR_VERSION=3.2.1
   ```

   Sie sollten Abschnitt 3.5 "Cluster konfigurieren" weglassen, da die Verbindung zur Image-Registry bei früheren Installationen oder Upgrades eingerichtet werden sollte.

   Link: Images für einen durch eine Air-Gap geschützten Cluster spiegeln.
2. **Führen Sie ein Upgrade für IBM MQ Operator auf 3.2.1durch.**

   Weitere Informationen finden Sie unter „Upgrade für IBM MQ Operator mit Red Hat OpenShift durchführen” auf Seite 51.
3. **Führen Sie ein Upgrade für die Instanzen durch**

   Um die neuesten Features und Sicherheitskorrekturen zu erhalten, führen Sie ein Upgrade für den Operanden IBM MQ (Warteschlangenmanager-Container-Image) auf die neueste CD -Version (9.4.0.0-

r1) durch. Weitere Informationen finden Sie unter „Upgrade für einen IBM MQ -Warteschlangenmanager mit Red Hat OpenShift durchführen” auf Seite 53.

Upgrade eines CD IBM MQ Operator auf den Kanal 3.2.x SC2/CD durchführen

Das Upgrade von IBM MQ Operator ermöglicht Ihnen das Upgrade Ihrer Warteschlangenmanager.

Vorbereitende Schritte

Wichtig:

- Diese Task ist für Benutzer der Lizenzen IBM MQ Operator und **nur** IBM MQ bestimmt. Wenn Sie ein Benutzer von IBM Cloud Pak for Integration (CP4I) sind oder mindestens einen Ihrer Warteschlangenmanager mit einer CP4I -Lizenz implementiert haben, lesen Sie den Abschnitt „Upgrade für IBM MQ Operator für CP4I -Benutzer durchführen” auf Seite 50.
- In diesem Abschnitt wird beschrieben, wie Sie Continuous Delivery (CD) -Implementierungen von IBM MQ Operator vor Version 3.2.0auf Version 3.2.1 **nur**aktualisieren. Wenn dies für Ihre Implementierung nicht zutrifft, sehen Sie sich die alternativen Upgradepfade an, die unter „Upgrade für IBM MQ Operator und Warteschlangenmanager durchführen” auf Seite 44beschrieben sind.

Für ein Upgrade auf IBM MQ Operator 3.2.1 muss Red Hat OpenShift Container Platform 4.12 oder höher ausgeführt werden. Informationen zur Überprüfung der kompatiblen Versionen für jeden IBM MQ Operator -Channel finden Sie unter „Kompatible Red Hat OpenShift Container Platform-Versionen” auf Seite 15. Informationen zum Upgrade der Plattform finden Sie unter Upgrade für Red Hat OpenShiftdurchführen.

Vorgehensweise

1. Optional: **Führen Sie ein Upgrade für einen IBM MQ Operator durch, der derzeit eine CD -Version vor 3.0.0aufweist.**

   Wenn Ihr IBM MQ Operator derzeit eine CD -Version vor 3.0.0aufweist, führen Sie die relevanten Schritte unter Migration auf den aktuellen CD-Kanal von IBM MQ Operator (IBM MQ 9.3 -Dokumentation)aus und kehren Sie dann hierher zurück, um ein Upgrade auf die neueste CD -Version durchzuführen. Beachten Sie, dass dies ein obligatorischer vorausgesetzter Schritt ist, bevor Sie ein Upgrade auf Version 3.2.1durchführen.
2. **Spiegelimages (nur Air-Gap).**

   Sie müssen die IBM MQ -Images spiegeln. Führen Sie die Schritte unter folgendem Link aus und verwenden Sie dabei nur diese Werte:

   ```
   export OPERATOR_PACKAGE_NAME=ibm-mq
   export OPERATOR_VERSION=3.2.1
   ```

   Sie sollten Abschnitt 3.5 "Cluster konfigurieren" weglassen, da die Verbindung zur Image-Registry bei früheren Installationen oder Upgrades eingerichtet werden sollte.

   Link: Images für einen durch eine Air-Gap geschützten Cluster spiegeln.
3. **Führen Sie ein Upgrade für IBM MQ Operator auf 3.2.1durch.**

   Weitere Informationen finden Sie unter „Upgrade für IBM MQ Operator mit Red Hat OpenShift durchführen” auf Seite 51.
4. **Aktualisieren Sie die Instanzen.**

   Um die neuesten Features und Sicherheitskorrekturen zu erhalten, führen Sie ein Upgrade für den Operanden IBM MQ (Warteschlangenmanager-Container-Image) auf die neueste CD -Version (9.4.0.0-r1) durch. Siehe „Upgrade für einen IBM MQ -Warteschlangenmanager mit Red Hat OpenShift durchführen” auf Seite 53.

In bestimmte Katalogquelle für IBM MQ Operator verschieben

Wenn Sie über eine Installation von IBM MQ Operator aus einem früheren Release verfügen und den IBM -Operatorkatalog verwenden, ist die Anwendung der bestimmten Katalogquelle die effektivste Methode, um die Softwareversionen in einem Cluster vollständig zu steuern.

Vorbereitende Schritte

Wichtig: Diese Task muss von einem Clusteradministrator ausgeführt werden. Siehe OpenShift -Rollen und -Berechtigungen.

Die folgenden Schritte werden über die Befehlszeilenschnittstelle ausgeführt.

Informationen zu diesem Vorgang

Der IBM -Operatorkatalog ist ein Index von Operatoren, die verfügbar sind, um die API eines Red Hat OpenShift Container Platform -Clusters zu erweitern und IBM Softwareprodukte zu aktivieren.

Diese Prozedur verschiebt eine Installation von IBM MQ Operator aus dem IBM -Operatorkatalog, sodass Sie die spezielle Katalogquelle für IBM MQ Operatorverwenden können.

Vorgehensweise

1. Fügen Sie den IBM MQ Operator -Katalog hinzu.

   Befolgen Sie die Anweisungen unter "Bestimmte Katalogquellen für IBM MQ Operatorhinzufügen" unter " IBM MQ Operator -Katalogquelle hinzufügen".
2. Vergewissern Sie sich, dass die IBM MQ Operator -Katalogquelle im Namensbereich `openshift-marketplace` erstellt wurde.

   Führen Sie den folgenden Befehl aus:

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

   Beispielausgabe:

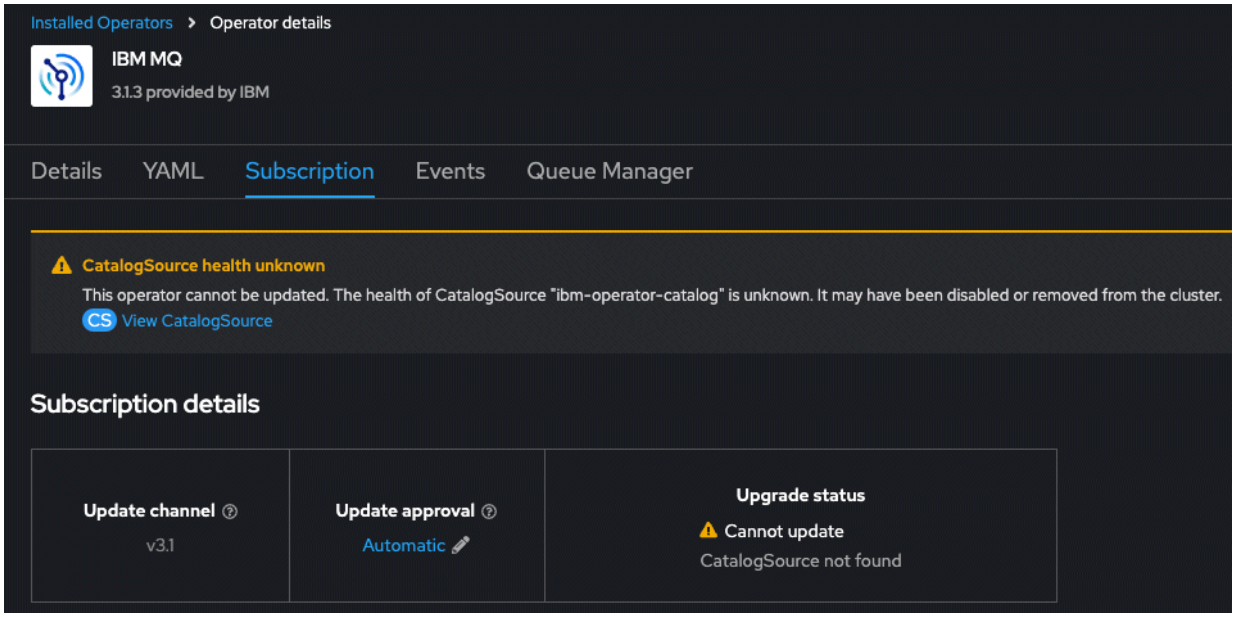
   ```
   oc get catalogsource -n openshift-marketplace

   NAME                           DISPLAY                TYPE   PUBLISHER   AGE
   ibm-operator-catalog           IBM Operator Catalog   grpc   IBM         23h
   ibmmq-operator-catalogsource   ibm-mq-3.1.3           grpc   IBM         23h
   ```
3. Optional: Löschen Sie die IBM Operator-Katalogquelle.

   Warnung: Sie sollten diesen Schritt nur ausführen, wenn Sie sicher sind, dass keine anderen Bediener den IBM -Operatorkatalog verwenden.

   Führen Sie den folgenden Befehl aus:

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

   Der IBM MQ Operator -Status ändert sich in `CatalogSource not found`. Dies ist ein erwartetes Verhalten.



4. Ändern Sie das Abonnement von IBM MQ Operator so, dass es auf die neue spezifische IBM MQ Operator -Katalogquelle verweist.
   a) Bearbeiten Sie die Subskription.

      Führen Sie den folgenden Befehl aus und ersetzen Sie *OPERATOR-NAMESPACE* entweder durch `openshift-operators` für clusterweite Installationen von IBM MQ Operatoroder durch den Namensbereich, in dem IBM MQ Operator bereitgestellt wird:

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

   b) Ändern Sie den Wert `spec.source` von `ibm-operator-catalog` in den Namen der in Schritt „1” auf Seite 49erstellten Katalogquelle.

      For example:

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

   c) Speichern Sie die Änderungen.

      Die IBM MQ Operator -Installation verweist jetzt auf die IBM MQ Operator -Katalogquelle. Wenn Sie den IBM -Operatorkatalog gelöscht haben, wird der Status von "CatalogSource not found" auf "Succeeded" zurückgesetzt.

Ergebnisse

Ihre Installation von IBM MQ Operator verweist jetzt auf die spezielle Katalogquelle für IBM MQ Operator. Dies gibt Ihnen die volle Kontrolle über Upgrades für den Operator.

OpenShift CP4I *Upgrade für IBM MQ Operator für CP4I -Benutzer durchführen*

Upgrade-Implementierungen von IBM MQ Operator , bei denen eine Lizenz für IBM Cloud Pak for Integration (CP4I) verwendet wird.

Vorbereitende Schritte

Wichtig: Diese Task gilt für CP4I -Benutzer. Dies gilt auch, wenn Sie mindestens einen Warteschlangenmanager unter einer CP4I -Lizenz implementiert haben. Wenn dies für Sie nicht zutrifft, lesen Sie den Abschnitt „Upgrade der IBM MQ Operator durchführen” auf Seite 45.

Informationen zu diesem Vorgang

Führen Sie eine der folgenden Optionen aus:

Prozedur

- **Option 1:** Upgrade-Implementierungen von 2.0.x Long Term Support (LTS) IBM MQ Operator

  Befolgen Sie die Schritte im Abschnitt Upgrade von 2022.2 durch Generieren eines Upgradeplans.
- **Option 2:** Upgrade einer 3.0.x -oder 3.1.x -Implementierung von IBM MQ Operator durchführen

  Befolgen Sie die Schritte im Abschnitt Upgrade von 2023.4 durch Generieren eines Upgradeplans.
- **Option 3:** Führen Sie ein Upgrade für andere Implementierungen von IBM MQ Operator durch.

  Befolgen Sie die relevanten Schritte unter Migration auf den aktuellen CD-Kanal von IBM MQ Operator (Dokumentation zuIBM MQ 9.3). Kehren Sie dann hierher zurück und fahren Sie mit **Option 2**fort. Beachten Sie, dass dies ein obligatorischer vorausgesetzter Schritt ist.

OpenShift CP4I ***Upgrade für IBM MQ Operator mit Red Hat OpenShift durchführen***

Sie können ein Upgrade für IBM MQ Operator über die Red Hat OpenShift -Webkonsole oder die Befehlszeilenschnittstelle durchführen.

Prozedur

Führen Sie eine der folgenden Tasks aus, um ein Upgrade für IBM MQ Operator mit Red Hat OpenShiftdurchzuführen:

- „Upgrade für IBM MQ Operator über die Red Hat OpenShift -Konsole durchführen” auf Seite 51
- „Upgrade von IBM MQ Operator über die Red Hat OpenShift-CLI durchführen” auf Seite 52

OpenShift CP4I *Upgrade für IBM MQ Operator über die Red Hat OpenShift -Konsole durchführen*

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

Vorbereitende Schritte

Anmerkung: Die neueste CD -Version von IBM MQ Operator ist 3.2.1und ist sowohl eine SC2 -als auch eine CD -Version. Die neuesten Releaseinformationen zu IBM MQ Operator finden Sie im Releaseprotokoll für IBM MQ Operator.

Melden Sie sich an der Red Hat OpenShift-Clusterkonsole an.

Vorgehensweise

1. Bestimmen Sie anhand der Informationen im Abschnitt „Versionsunterstützung für IBM MQ Operator” auf Seite 14, auf welchen Operatorkanal ein Upgrade durchgeführt werden soll.
2. Wenden Sie die neueste Katalogquelle an.

   Wenn Sie die spezielle Katalogquelle für IBM MQ Operatoranstelle von `ibm-operator-catalog`verwenden, müssen Sie die Katalogquelle für die neue IBM MQ -Version anwenden.

   Wenn Sie von der Verwendung des IBM -Operatorkatalogs zur Verwendung der bestimmten Katalogquelle für IBM MQ Operatorwechseln und eine bessere Kontrolle über Upgrades erhalten möchten, lesen Sie die Schritte in „In bestimmte Katalogquelle für IBM MQ Operator verschieben” auf Seite 49 , bevor Sie zu Schritt „3” auf Seite 52zurückkehren.

   Wenn Sie den IBM -Operatorkatalog verwenden (nur einige Onlineinstallationen), fahren Sie mit Schritt „3” auf Seite 52fort.

   Befolgen Sie die Anweisungen in „IBM MQ Operator -Katalogquelle hinzufügen” auf Seite 35.

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) 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 14.

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

Die IBM MQ Operatorkann über die Befehlszeile aktualisiert werden.

Vorbereitende Schritte

Anmerkung: Die neueste CD -Version von IBM MQ Operator ist 3.2.1und ist sowohl eine SC2 -als auch eine CD -Version. Die neuesten Releaseinformationen zu IBM MQ Operator finden Sie im Releaseprotokoll für IBM MQ Operator.

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

Vorgehensweise

1. Bestimmen Sie anhand der Informationen im Abschnitt „Versionsunterstützung für IBM MQ Operator” auf Seite 14, auf welchen Operatorkanal ein Upgrade durchgeführt werden soll.
2. Wenden Sie die neueste Katalogquelle an.

   Wenn Sie die spezielle Katalogquelle für IBM MQ Operatoranstelle von `ibm-operator-catalog`verwenden, müssen Sie die Katalogquelle für die neue IBM MQ -Version anwenden.

   Wenn Sie von der Verwendung des IBM -Operatorkatalogs zur Verwendung der bestimmten Katalogquelle für IBM MQ Operatorwechseln und eine bessere Kontrolle über Upgrades erhalten möchten, lesen Sie die Schritte in „In bestimmte Katalogquelle für IBM MQ Operator verschieben” auf Seite 49 , bevor Sie zu Schritt „3” auf Seite 52zurückkehren.

   Wenn Sie den IBM -Operatorkatalog verwenden (nur einige Onlineinstallationen), fahren Sie mit Schritt „3” auf Seite 52fort.

   Befolgen Sie die Anweisungen in „IBM MQ Operator -Katalogquelle hinzufügen” auf Seite 35.
3. Führen Sie ein Upgrade für IBM MQ Operator durch. Neue übergeordnete/untergeordnete IBM MQ Operator-Versionen werden 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) 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
      ```

OpenShift CP4I *Upgrade für einen IBM MQ -Warteschlangenmanager mit Red Hat OpenShift durchführen*

Vorbereitende Schritte

Im Rahmen des Upgradeprozesses für die IBM MQ -Warteschlangenmanager wurden Sie möglicherweise aus der IBM Cloud Pak for Integration -Dokumentation an dieses Thema gesendet.

Prozedur

Führen Sie eine der folgenden Tasks aus, um ein Upgrade des IBM MQ -Warteschlangenmanagers mit Red Hat OpenShiftdurchzuführen:

- „Upgrade für einen IBM MQ -Warteschlangenmanager über die Red Hat OpenShift -Konsole durchführen” auf Seite 53
- „Upgrade eines IBM MQ-Warteschlangenmanagers über die Red Hat OpenShift-CLI durchführen” auf Seite 54
- „Upgrade eines IBM MQ -Warteschlangenmanagers in Red Hat OpenShift über die Plattformbenutzerschnittstelle durchführen” auf Seite 54

Nächste Schritte

Zum Durchführen eines IBM Cloud Pak for Integration -Upgrades müssen Sie möglicherweise zur IBM Cloud Pak for Integration -Dokumentation zurückkehren.

OpenShift CP4I *Upgrade für einen IBM MQ -Warteschlangenmanager über die Red Hat OpenShift -Konsole durchführen*

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

Vorbereitende Schritte

Anmerkung: Die neueste Version des IBM MQ -Warteschlangenmanagers ist 9.4.0.0-r1und ist sowohl eine SC2 -als auch eine CD -Version. Die neuesten Releaseinformationen zum IBM MQ -Warteschlangenmanager finden Sie im Releaseprotokoll für Warteschlangenmanager-Container-Images zur Verwendung mit IBM MQ Operator.

- 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 mit Red Hat OpenShift durchführen” auf Seite 51.

Bevor Sie ein Upgrade für den Warteschlangenmanager in einer Air-Gap-Umgebung durchführen können, müssen Sie die neuesten IBM Cloud Pak for Integration -Images über die Airgap-spezifischen Schritte im Abschnitt Upgrade für CD IBM MQ Operator auf den Kanal 3.2.x SC2/CD durchführenspiegeln.

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 **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

   Im Abschnitt „Releaseprotokoll für Warteschlangenmanager-Container-Images zur Verwendung mit IBM MQ Operator" auf Seite 7 finden Sie eine Zuordnung von IBM MQ Operator -Versionen und IBM MQ -Warteschlangenmanager-Container-Images.
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

Anmerkung: Die neueste Version des IBM MQ -Warteschlangenmanagers ist 9.4.0.0-r1und ist sowohl eine SC2 -als auch eine CD -Version. Die neuesten Releaseinformationen zum IBM MQ -Warteschlangenmanager finden Sie im Releaseprotokoll für Warteschlangenmanager-Container-Images zur Verwendung mit IBM MQ Operator.

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 44.

Bevor Sie ein Upgrade für den Warteschlangenmanager in einer Air-Gap-Umgebung durchführen können, müssen Sie die neuesten IBM Cloud Pak for Integration -Images über die Airgap-spezifischen Schritte im Abschnitt Upgrade für CD IBM MQ Operator auf den Kanal 3.2.x SC2/CD durchführenspiegeln.

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 14 .

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 die Plattform-benutzerschnittstelle durchführen*

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

Vorbereitende Schritte

Anmerkung: Die neueste Version des IBM MQ -Warteschlangenmanagers ist 9.4.0.0-r1und ist sowohl eine SC2 -als auch eine CD -Version. Die neuesten Releaseinformationen zum IBM MQ -Warteschlangen-

manager finden Sie im Releaseprotokoll für Warteschlangenmanager-Container-Images zur Verwendung mit IBM MQ Operator.

- Melden Sie sich bei IBM Cloud Pak for Integration Platform UI 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 44.

Bevor Sie ein Upgrade für den Warteschlangenmanager in einer Air-Gap-Umgebung durchführen können, müssen Sie die neuesten IBM Cloud Pak for Integration -Images über die Airgap-spezifischen Schritte im Abschnitt Upgrade für CD IBM MQ Operator auf den Kanal 3.2.x SC2/CD durchführenspiegeln.

Vorgehensweise

1. Klicken Sie auf der IBM Cloud Pak for Integration Platform UI-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.

IBM MQ Operator deinstallieren

Sie können die Red Hat OpenShift -Konsole oder CLI verwenden, um IBM MQ Operator aus Red Hat OpenShiftzu deinstallieren.

Prozedur

- Option 1: Deinstallieren Sie IBM MQ Operator mit der OpenShift -Konsole.

  Anmerkung: Wenn IBM MQ Operator in allen Projekten/Namensbereichen im Cluster installiert ist, wiederholen Sie die Schritte 2 bis 6 der folgenden Prozedur für jedes Projekt, in dem Sie Warteschlangenmanager löschen möchten.

  a) Melden Sie sich bei der Red Hat OpenShift Container Platform -Webkonsole mit den Berechtigungsnachweisen Ihres Red Hat OpenShift Container Platform -Clusteradministrators an.

  b) Ändern Sie **Project** in den Namensbereich, aus dem Sie IBM MQ Operatordeinstallieren möchten. Wählen Sie den Namensbereich aus der Dropdown-Liste **Projekt** aus.

  c) Klicken Sie im Navigationsfenster auf **Operatoren** > **Installierte Operatoren**.

  d) Klicken Sie auf den **IBM MQ**-Operator.

  e) Klicken Sie auf die Registerkarte **Queue Managers** (Warteschlangenmanager), um die von diesem IBM MQ Operator verwalteten Warteschlangenmanager anzuzeigen.

  f) 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.

  g) Optional: Wiederholen Sie gegebenenfalls die Schritte 2 bis 6 für jedes Projekt, in dem Sie Warteschlangenmanager löschen wollen.

  h) Kehren Sie zu **Operators** > **Installed Operators** (Operatoren > Installierte Operatoren) zurück.

i) Klicken Sie neben dem **IBM MQ**-Operator auf das Menü mit den drei Punkten und wählen Sie **Uninstall Operator** (Operator deinstallieren) aus.

- Option 2: IBM MQ Operator über die OpenShift -CLI deinstallieren
  a) Melden Sie sich mit `oc login`bei Ihrem Red Hat OpenShift -Cluster an.
  b) Wenn IBM MQ Operator in einem einzelnen Namensbereich installiert ist, führen Sie die folgenden Schritte aus:
     a. Stellen Sie sicher, dass Sie sich in dem Projekt befinden, das die zu deinstallierende IBM MQ Operator enthält:

     ```
     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. Wenn die Common Services von keinem Prozess mehr verwendet werden, können Sie auch den Common Services-Operator deinstallieren und die Operatorgruppe löschen:
        i) Deinstallieren Sie den Operator für allgemeine Services, indem Sie die Anweisungen in Basis-services deinstallieren in der Produktdokumentation zu IBM Cloud Pak foundational services befolgen.
        ii) Zeigen Sie die Operatorgruppe an:

        ```
        oc get operatorgroup
        ```

        iii) Löschen Sie die Operatorgruppe:

        ```
        oc delete OperatorGroup operator_group_name
        ```

  c) 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. Optional: Wenn keine anderen allgemeinen Services verwendet werden, können Sie den Operator für allgemeine Services deinstallieren. Befolgen Sie dazu die Anweisungen im Abschnitt Basisservices deinstallieren in der Produktdokumentation zu IBM Cloud Pak foundational services .

IBM MQ durch Erstellung eines eigenen Container-Image vorbereiten

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

Bevor Sie einen eigenen Container entwickeln, überlegen Sie, ob Sie stattdessen die IBM MQ Operatorverwenden können. Siehe „Verwendung von IBM MQ in Containern auswählen” auf Seite 8

Informationen zu diesem Vorgang

Prozedur

- „Allgemeine Hinweise zur Erstellung eines eigenen Warteschlangenmanager-Image” auf Seite 57
- „Beispielcontainer-Image für IBM MQ-Warteschlangenmanager erstellen” auf Seite 58
- „Lokale Bindungsanwendungen in separaten Containern ausführen” auf Seite 61
- Sehen Sie sich das IBM MQ Beispieldiagramm Helm an.

Allgemeine Hinweise zur Erstellung eines eigenen Warteschlangenmanager-Image

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 dieses Problem zu lösen, können Sie beim Start Ihres Containers den Befehl **crtmqdir** verwenden, um `/var/mqm` bei der ersten Ausführung zu füllen.

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.

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.

Schließlich führen Sie einen Container unter Verwendung dieses Image als Dateisystem aus, wobei der Inhalt von `/var/mqm` von einem containerspezifischen Datenträger auf dem Hostdateisystem bereitgestellt wird.

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 58
  - „Beispiel für ein konfiguriertes IBM MQ-Warteschlangenmanagerimage erstellen” auf Seite 59

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.

- Optional: Wenn Sie den sicheren Zugriff mit der Sicherheitskontexteinschränkung Red Hat OpenShift Container Platform "restricted" (SCC) konfigurieren möchten, verwenden Sie eines der IBM MQ -Images, die nicht installiert sind.

  Links zum Herunterladen dieser Images sind im Abschnitt "Container" unter IBM MQ Downloadsverfügbar.

Ergebnisse

Sie verfügen nun über ein Container-Image mit installiertem IBM MQ.

Sie können jetzt ein Beispiel für ein IBM MQ-Warteschlangenmanager-Image erstellen.

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

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 der aktuellen Version von IBM MQdas Paket "No-Install" IBM MQ verwenden, gibt es weder einen `mqm` -Benutzer noch eine `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.

Lokale Bindungsanwendungen in separaten Containern ausführen

Mit der gemeinsamen Nutzung von Prozessnamensbereichen durch mehrere Container können Sie Anwendungen, die eine lokale Bindungsverbindung zu IBM MQ erfordern, in separaten Containern vom IBM MQ -Warteschlangenmanager ausführen.

Informationen zu diesem Vorgang

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.

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 62 weitere Informationen.

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 61 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
```

MQ Adv. 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.

Anmerkung: Auf diese Weise können Sie eine native HA-Gruppe in Ihren eigenen Containern nur erstellen, wenn Sie Kubernetes oder Red Hat OpenShiftausführen.

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
   ```

   For example:

   ```
   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)
```

Warteschlangenmanager in Containern implementieren und konfigurieren

Sie führen verschiedene Tasks aus, um IBM MQ -Warteschlangenmanager zu implementieren und zu konfigurieren.

Informationen zu diesem Vorgang

Informationen zum Einstieg in die Implementierung und Konfiguration von Warteschlangenmanagern finden Sie in den folgenden Abschnitten.

Prozedur

- „Warteschlangenmanager mithilfe von IBM MQ Operator implementieren und konfigurieren” auf Seite 66
- „Warteschlangenmanager mit Helm implementieren und konfigurieren” auf Seite 109

OpenShift Warteschlangenmanager mithilfe von IBM MQ Operator implementieren und konfigurieren

Konfigurationsbeispiele; Konfiguration der Hochverfügbarkeit; Verbindung von außerhalb eines OpenShift -Clusters; Integration mit dem CP4i -Dashboard; Integration mit Instana-Traceerstellung; Erstellung eines Image mit angepassten MQSC-und INI-Dateien; Hinzufügen von angepassten Anmerkungen und Bezeichnungen.

Informationen zu diesem Vorgang

Prozedur

- „Beispiele für die Konfiguration eines Warteschlangenmanagers” auf Seite 69.
- „Hohe Verfügbarkeit für Warteschlangenmanager mithilfe von IBM MQ Operator konfigurieren” auf Seite 78.
- „Route für Verbindung zu einem Warteschlangenmanager von außerhalb eines Red Hat OpenShift-Clusters konfigurieren” auf Seite 88.
- „IBM MQ mit IBM Instana -Traceerstellung integrieren” auf Seite 90.
- „Image mit benutzerdefinierten MQSC- und INI-Dateien über die Red Hat OpenShift-CLI erstellen” auf Seite 98.
- „Angepasste Anmerkungen und Beschriftungen zu Warteschlangenmanagerressourcen hinzufügen” auf Seite 100.
- „Laufzeit-Webhook-Prüfungen inaktivieren” auf Seite 100.
- „Inaktivieren von Standardwertaktualisierungen für die Warteschlangenmanagerspezifikation” auf Seite 101.

OpenShift CP4I Einfachen Warteschlangenmanager mit IBM MQ Operator implementieren

In diesem Beispiel wird ein Warteschlangenmanager für den Schnelleinstieg implementiert, der ephemeren (nicht persistenten) Speicher verwendet und die IBM MQ -Sicherheit inaktiviert. Nachrichten werden bei Neustarts des Warteschlangenmanagers nicht persistent gespeichert. Sie können die Konfiguration anpassen, um viele Warteschlangenmanagereinstellungen zu ändern.

Informationen zu diesem Vorgang

Diese Task bietet drei Optionen für die Implementierung eines Warteschlangenmanagers in OpenShift:

1. Implementieren Sie einen Warteschlangenmanager mit der OpenShift -Konsole.
2. Implementieren Sie einen Warteschlangenmanager mit der OpenShift -CLI.
3. Implementieren Sie einen Warteschlangenmanager mit IBM Cloud Pak for Integration Platform UI.

Prozedur

- **Option 1: Warteschlangenmanager über die OpenShift -Konsole implementieren.**
  a) Implementieren Sie einen Warteschlangenmanager.
    a. Melden Sie sich mit den Berechtigungsnachweisen Ihres Red Hat OpenShift Container Platform -Clusteradministrators an der OpenShift -Konsole an.
    b. Ändern Sie **Project** in den Namensbereich, in dem Sie IBM MQ Operatorinstalliert haben. Wählen Sie den Namensbereich aus der Dropdown-Liste **Projekt** aus.
    c. Klicken Sie im Navigationsfenster auf **Operatoren** > **Installierte Operatoren**.
    d. Suchen Sie in der Liste in der Anzeige "Installierte Operatoren" nach **IBM MQ**und klicken Sie darauf.

e. Klicken Sie auf die Registerkarte **Queue Manager** (Warteschlangenmanager).

f. Klicken Sie auf die Schaltfläche **Create QueueManager** (QueueManager erstellen). Die Instanzerstellungsanzeige wird angezeigt und bietet zwei Methoden zum Konfigurieren der Ressource: die **Formularansicht** und die **YAML-Ansicht**. Die **Formularansicht** ist standardmäßig ausgewählt.

b) Konfigurieren Sie den Warteschlangenmanager.

Schritt 2 Option 1: Konfigurieren Sie in der **Formularansicht**.

Die **Formularansicht** öffnet ein Formular, in dem Sie die Ressourcenkonfiguration anzeigen oder ändern können.

a. Klicken Sie neben **Lizenz**auf den Pfeil, um den Abschnitt zur Lizenzannahme zu erweitern.

b. Setzen Sie **License accept** auf **true** , wenn Sie die Lizenzvereinbarung akzeptieren.

c. Klicken Sie auf den Pfeil, um die Dropdown-Liste zu öffnen, und wählen Sie eine Lizenz aus. 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 145. Um einen Warteschlangenmanager implementieren zu können, müssen Sie die Lizenz akzeptieren.

d. 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.

Schritt 2 Option 2: Konfigurieren Sie in der **YAML-Sicht**.

Die **YAML-Ansicht** öffnet einen Editor, der eine YAML-Beispieldatei für eine `QueueManager`enthält. Aktualisieren Sie die Werte in der Datei, indem Sie die folgenden Schritte ausführen.

a. Ändern Sie `metadata.namespace` in Ihren Projektnamen (Namensbereich).

b. Ändern Sie den Wert von `spec.license.license` in die Lizenzzeichenfolge, die Ihren Anforderungen entspricht. Details zur Lizenz finden Sie in „Lizenzierungsreferenz für mq.ibm.com/v1beta1" auf Seite 145 .

c. Ändern Sie `spec.license.accept` in `true` , wenn Sie die Lizenzvereinbarung akzeptieren.

d. 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.

c) Überprüfen Sie die Erstellung des Warteschlangenmanagers.

Sie können überprüfen, ob Sie einen Warteschlangenmanager erstellt haben, indem Sie die folgenden Schritte ausführen:

a. Stellen Sie sicher, dass Sie sich in dem Namensbereich befinden, in dem Sie IBM MQ Operator erstellt haben.

b. Klicken Sie in der **Hauptanzeige** auf **Operatoren** > **Installierte Operatoren**und wählen Sie dann die installierte IBM MQ Operator -Instanz aus, für die Sie den Warteschlangenmanager erstellt haben.

c. Klicken Sie auf die Registerkarte **Queue Manager** (Warteschlangenmanager). Die Erstellung ist beendet, wenn der `QueueManager` den Status `Running` (Aktiv) hat.

- **Option 2: Implementieren Sie einen Warteschlangenmanager über die OpenShift -CLI.**

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: 9.4.0.0-r1
```

```
  license:
    accept: false
    license: L-BMSF-5YDSLR
    use: NonProduction
  web:
    enabled: true
  queueManager:
    name: "QUICKSTART"
    storage:
      queueManager:
        type: ephemeral
```

Wichtig: Wenn Sie die Lizenzvereinbarung akzeptieren, ändern Sie `accept: false` in `accept: true`. Details zur Lizenz finden Sie im Abschnitt „Lizenzierungsreferenz für mq.ibm.com/v1beta1" auf Seite 145.

Dieses Beispiel umfasst auch einen Web-Server, der mit dem Warteschlangenmanager implementiert wird, wobei die Webkonsole mit Single Sign-on in IBM Cloud Pak for Integrationaktiviert ist. Damit Single Sign-on funktioniert, müssen zunächst andere IBM Cloud Pak for Integration -Komponenten installiert werden. Weitere Informationen finden Sie im Abschnitt „IBM MQ Operator zur Verwendung mit CP4I installieren" auf Seite 41.

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: 9.4.0.0-r1
  license:
    accept: false
    license: L-EHXT-MQCRN9
  web:
    enabled: true
  queueManager:
    name: "QUICKSTART"
    storage:
      queueManager:
        type: ephemeral
```

Wichtig: Wenn Sie die Lizenzvereinbarung für MQ akzeptieren, ändern Sie `accept: false` in `accept: true`. Details zur Lizenz finden Sie im Abschnitt „Lizenzierungsreferenz für mq.ibm.com/v1beta1" auf Seite 145.

b) Erstellen Sie das Objekt `QueueManager` .

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

c) Überprüfen Sie die Erstellung des Warteschlangenmanagers.

Überprüfen Sie, ob Sie einen Warteschlangenmanager erstellt haben, indem Sie die folgenden Schritte ausführen:

a. Überprüfen Sie die Bereitstellung:

```
oc describe queuemanager Queue_Manager_Resource_Name
```

b. Überprüfen Sie den Status:

```
oc describe queuemanager quickstart
```

- **Option 3: Implementieren Sie einen WS-Manager mit dem IBM Cloud Pak for Integration Platform UI.**

  Befolgen Sie die Anweisungen unter Instanz über die Plattformbenutzerschnittstelle bereitstellen.

Zugehörige Tasks

„Route für Verbindung zu einem Warteschlangenmanager von außerhalb eines Red Hat OpenShift-Clusters konfigurieren" auf Seite 88

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 134
Vorgehensweise zum Herstellen einer Verbindung zur IBM MQ Console eines Warteschlangenmanagers, der in einem Red Hat OpenShift Container Platform -Cluster implementiert wurde.

„Beispiele für die Konfiguration eines Warteschlangenmanagers" auf Seite 69
Ein Warteschlangenmanager kann durch Anpassung des Inhalts der angepassten QueueManager-Ressource konfiguriert werden.

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 69
- „Beispiel: Warteschlangenmanager mit gegenseitiger TLS-Authentifizierung konfigurieren" auf Seite 73

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.

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 QueueManager -Implementieren Sie Ihren Warteschlangenmanager mit der folgenden Konfiguration über die Befehlszeile oder über die Red Hat OpenShift Container Platform -Webkonsole:

```
apiVersion: mq.ibm.com/v1beta1
kind: QueueManager
metadata:
  name: mqsc-ini-qm
spec:
  version: 9.4.0.0-r1
  license:
    accept: false
    license: L-EHXT-MQCRN9
    use: Production
  web:
    enabled: true
  queueManager:
    name: "MQSCINI"
    mqsc:
    - configMap:
        name: mqsc-ini-example
        items:
        - example1.mqsc
        - example2.mqsc
    ini:
    - configMap:
        name: mqsc-ini-example
        items:
        - example.ini
    storage:
      queueManager:
        type: ephemeral
```

Wichtig: Wenn Sie die Lizenzvereinbarung für IBM MQ Advancedakzeptieren, 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 die Verwendung einer einzelnen Kubernetes ConfigMap oder eines geheimen Schlüssels (wie in diesem Beispiel gezeigt) oder für mehrere ConfigMaps und geheime Schlüssel 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.
- Wenn ein Warteschlangenmanager-Pod ausgeführt wird, werden alle Änderungen an der Kubernetes ConfigMap nicht berücksichtigt, weil IBM MQ Operator die Änderung nicht erkennt. Wenn Sie Änderungen an der ConfigMapvornehmen, z. B. an den MQSC-Befehlen oder den INI-Dateien, müssen Sie die Warteschlangenmanager manuell erneut starten, um diese Änderungen zu übernehmen. Löschen Sie bei Einzelinstanz-Warteschlangenmanagern den Pod, um den erforderlichen Neustart auszulösen. Starten Sie bei nativen HA-Implementierungen zuerst die Standby-Pods erneut, indem Sie sie löschen. Wenn sie sich wieder in einem aktiven Status befinden, löschen Sie den aktiven Pod, um ihn neu zu starten. Diese Reihenfolge der Neustarts stellt die minimale Ausfallzeit für den Warteschlangenmanager sicher.

OpenShift CP4I *Selbst signierte PKI mit OpenSSL erstellen*

IBM MQ ermöglicht Ihnen die Verwendung der gegenseitigen TLS-Authentifizierung, bei der beide Enden einer Verbindung ein Zertifikat bereitstellen und Details im Zertifikat verwendet werden, um eine Identität mit dem Warteschlangenmanager herzustellen. In diesem Abschnitt wird beschrieben, wie Sie mit dem OpenSSL -Befehlszeilentool ein Beispiel für eine PKI (Public Key Infrastructure) erstellen und zwei Zertifikate erstellen, die in anderen Beispielen verwendet werden können.

Vorbereitende Schritte

Stellen Sie sicher, dass das Befehlszeilentool OpenSSL installiert ist.

Installieren Sie den IBM MQ client und fügen Sie `samp/bin` und `bin` Ihrem *PATH* hinzu. Sie benötigen den Befehl **`runmqicred`**, der als Teil von IBM MQ client wie folgt installiert werden kann:

- Windows Linux Für Windows und Linux: Installieren Sie den IBM MQ Redistributable Client für Ihr Betriebssystem von https://ibm.biz/mq94redistclients
- macOS Für Mac: Laden Sie IBM MQ MacOS Toolkitherunter und richten Sie es ein: https://developer.ibm.com/tutorials/mq-macos-dev/

Informationen zu diesem Vorgang

Wichtig: Die hier beschriebenen Beispiele eignen sich nicht für eine Produktionsumgebung und sind lediglich als Beispiele für einen schnellen Einstieg gedacht. Das Zertifikatsmanagement ist ein komplexes Subjekt für fortgeschrittene Benutzer. Für die Produktion müssen Sie Dinge wie Rotation, Widerruf, Schlüssellänge, Disaster-Recovery und vieles mehr berücksichtigen.

Diese Schritte wurden mit OpenSSL 3.1.4getestet.

Vorgehensweise

1. Erstellen Sie einen privaten Schlüssel für Ihre interne Zertifizierungsstelle.

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

   Ein privater Schlüssel für die interne Zertifizierungsstelle wird in einer Datei namens *ca.key*erstellt. Diese Datei sollte sicher und geheim gehalten werden-sie wird zum Signieren von Zertifikaten für Ihre interne Zertifizierungsstelle verwendet.
2. Ein selbst signiertes Zertifikat für Ihre interne Zertifizierungsstelle ausstellen

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

   `-days` gibt an, wie viele Tage das Zertifikat der Stammzertifizierungsstelle gültig sein soll.

   Ein Zertifikat wird in einer Datei namens *ca.crt*erstellt. Dieses Zertifikat enthält die öffentlichen Informationen über die interne Zertifizierungsstelle und kann frei gemeinsam genutzt werden.
3. Privaten Schlüssel und Zertifikat für einen Warteschlangenmanager erstellen

   a) Privaten Schlüssel und Zertifikatssignieranforderung für einen Warteschlangenmanager erstellen

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

   Ein privater Schlüssel wird in einer Datei namens *example-qm.key*erstellt und eine Zertifikatssignieranforderung in einer Datei namens *example-qm.csr* .

   b) Signieren Sie den Warteschlangenmanagerschlüssel mit Ihrer internen Zertifizierungsstelle.

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

   `-days` gibt die Anzahl der Tage an, die das Zertifikat gültig ist.

Ein signiertes Zertifikat wird in einer Datei namens *example-qm.crt* erstellt.

c) Erstellen Sie einen geheimen Kubernetes -Schlüssel mit dem Warteschlangenmanagerschlüssel und -Zertifikat.

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

Ein geheimer Kubernetes -Schlüssel namens *example-qm-tls* wird erstellt. Dieser geheime Schlüssel enthält den privaten Schlüssel für den Warteschlangenmanager, das öffentliche Zertifikat und das CA-Zertifikat.

4. Privaten Schlüssel und Zertifikat für eine Anwendung erstellen

a) Privaten Schlüssel und Zertifikatssignieranforderung für eine Anwendung erstellen

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

Ein privater Schlüssel wird in einer Datei namens *example-app1.key*erstellt und eine Zertifikatssignieranforderung wird in einer Datei namens *example-app1.csr* erstellt.

b) Signieren Sie den Warteschlangenmanagerschlüssel mit Ihrer internen Zertifizierungsstelle.

```
openssl x509 -req -in example-app1.csr -CA ca.crt -CAkey ca.key -CAcreateserial -out exam
ple-app1.crt -days 7 -sha512
```

`-days` gibt die Anzahl der Tage an, die das Zertifikat gültig ist.

Ein signiertes Zertifikat wird in einer Datei namens *example-app1.crt* erstellt.

c) Erstellen Sie einen PKCS#12 -Keystore mit dem Schlüssel und Zertifikat der Anwendung.

IBM MQ verwendet eine Schlüsseldatenbank und keine einzelnen Schlüsseldateien. Der containerisierte Warteschlangenmanager erstellt die Schlüsseldatenbank für den WS-Manager aus einem geheimen Schlüssel, aber für Clientanwendungen müssen Sie die Schlüsseldatenbank manuell erstellen.

```
openssl pkcs12 -export -in "example-app1.crt" -name "example-app1" -certfile "ca.crt" -in
key "example-app1.key" -out "example-app1.p12" -passout pass:PASSWORD
```

Dabei ist *PASSWORD* ein Kennwort Ihrer Wahl.

Ein Keystore wird in einer Datei mit dem Namen *example-app1.p12*erstellt. Der Schlüssel und das Zertifikat der Anwendung werden in gespeichert, mit einem "Kennsatz" oder "Anzeigenamen" von "example-app1" sowie dem CA-Zertifikat.

d) Wenn Sie einen arm64 Apple Mac verwenden, müssen Sie eine zusätzliche Datei konfigurieren, die die Anwendung und die CA-Zertifikate kombiniert.
For example:

```
cat example-app1.crt ca.crt > example-app1-chain.crt
```

Zugehörige Tasks

„Beispiel: Warteschlangenmanager mit gegenseitiger TLS-Authentifizierung konfigurieren” auf Seite 73
In diesem Beispiel wird ein Warteschlangenmanager mit IBM MQ Operator in OpenShift Container Platform bereitgestellt. Gegenseitige TLS wird für die Authentifizierung verwendet, um ein TLS-Zertifikat einer Identität im Warteschlangenmanager zuzuordnen.

„Beispiel: Native HA mit IBM MQ Operator konfigurieren” auf Seite 79
In diesem Beispiel wird ein WS-Manager mit der nativen Hochverfügbarkeitsfunktion in OpenShift Container Platform unter Verwendung von IBM MQ Operatorimplementiert. Gegenseitige TLS wird für die Authentifizierung verwendet, um ein TLS-Zertifikat einer Identität im Warteschlangenmanager zuzuordnen.

„Warteschlangenmanager mit mehreren Instanzen mit IBM MQ Operator konfigurieren” auf Seite 85
In diesem Beispiel wird ein Warteschlangenmanager mit mehreren Instanzen unter Verwendung von IBM MQ Operatorin OpenShift Container Platform implementiert. Gegenseitige TLS wird für die Authentifizierung verwendet, um ein TLS-Zertifikat einer Identität im Warteschlangenmanager zuzuordnen.

OpenShift CP4I Linux **Beispiel: Warteschlangenmanager mit gegenseitiger TLS-Authentifizierung konfigurieren**

In diesem Beispiel wird ein Warteschlangenmanager mit IBM MQ Operator in OpenShift Container Platform bereitgestellt. Gegenseitige TLS wird für die Authentifizierung verwendet, um ein TLS-Zertifikat einer Identität im Warteschlangenmanager zuzuordnen.

Vorbereitende Schritte

Die folgenden Schritte müssen als Voraussetzung für dieses Beispiel durchgeführt worden sein:

- Erstellen Sie für dieses Beispiel ein Projekt bzw. einen Namensbereich für 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 OpenShift Container Platform bereitzustellenden Warteschlangenmanager definiert. Darüber hinaus beschreibt es alle weiteren Schritte, die zur Bereitstellung des Warteschlangenmanagers mit aktiviertem TLS erforderlich sind.

Vorgehensweise

1. Erstellen Sie ein Zertifikatspaar wie in „Selbst signierte PKI mit OpenSSL erstellen” auf Seite 71beschrieben.
2. Konfigurationsübersicht mit MQSC-Befehlen und einer INI-Datei erstellen

   Erstellen Sie eine Kubernetes ConfigMap mit den MQSC-Befehlen, um eine neue Warteschlange und einen SVRCONN-Kanal zu erstellen und einen Kanalauthentifizierungsdatensatz hinzuzufügen, der den Zugriff auf den Kanal ermöglicht.

   Stellen Sie sicher, dass Sie sich in dem Namensbereich befinden, den Sie zuvor erstellt haben (siehe Vorbereitungen), und geben Sie dann die folgende YAML-Datei in der OCP-Webkonsole oder über die Befehlszeile ein.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: example-tls-configmap
data:
  example-tls.mqsc: |
    DEFINE CHANNEL('MTLS.SVRCONN') CHLTYPE(SVRCONN) SSLCAUTH(REQUIRED) SSLCIPH('A□
NY_TLS13_OR_HIGHER') REPLACE
    SET CHLAUTH('MTLS.SVRCONN') TYPE(SSLPEERMAP) SSLPEER('CN=*') USERSRC(NOACCESS) ACTION(RE□
PLACE)
    SET CHLAUTH('MTLS.SVRCONN') TYPE(SSLPEERMAP) SSLPEER('CN=example-app1') USERSRC(MAP)
MCAUSER('app1') ACTION(REPLACE)
    SET AUTHREC PRINCIPAL('app1') OBJTYPE(QMGR) AUTHADD(CONNECT,INQ)
    DEFINE QLOCAL('EXAMPLE.QUEUE') REPLACE
    SET AUTHREC PROFILE('EXAMPLE.QUEUE') PRINCIPAL('app1') OBJTYPE(QUEUE) AUTHADD(BROW□
SE,PUT,GET,INQ)
  example-tls.ini: |
    Service:
        Name=AuthorizationService
        EntryPoints=14
        SecurityPolicy=UserExternal
```

   Der MQSC definiert einen Kanal mit dem Namen *MTLS.SVRCONN* und eine Warteschlange mit dem Namen *EXAMPLE.QUEUE*. Der Kanal ist so konfiguriert, dass nur auf Clients zugegriffen werden kann, die ein Zertifikat mit dem allgemeinen Namen *example-app1*vorlegen. Dies ist der allgemeine Name, der in einem der in Schritt „1” auf Seite 73erstellten Zertifikate verwendet wird. Verbindungen dieses Kanals mit diesem allgemeinen Namen werden der Benutzer-ID *app1*zugeordnet, die berechtigt ist, eine

Verbindung zum Warteschlangenmanager herzustellen und auf die Beispielwarteschlange zuzugreifen. Die INI-Datei aktiviert eine Sicherheitsrichtlinie, was bedeutet, dass die Benutzer-ID *app1* nicht in einer externen Benutzerregistry vorhanden sein muss, sondern nur als Name in dieser Konfiguration.

3. Stellen Sie den Warteschlangenmanager bereit.

   Erstellen Sie mit der folgenden angepassten YAML-Ressource einen neuen Warteschlangenmanager. Stellen Sie sicher, dass Sie sich in dem Namensbereich befinden, den Sie vor Beginn dieser Aufgabe erstellt haben. Geben Sie dann die folgende YAML-Datei in der OCP-Webkonsole oder über die Befehlszeile ein. Vergewissern Sie sich, dass die richtige Lizenz angegeben ist, und bestätigen Sie die Lizenz, indem Sie `false` in `true` ändern.

```
apiVersion: mq.ibm.com/v1beta1
kind: QueueManager
metadata:
  name: exampleqm
spec:
  license:
    accept: false
    license: L-EHXT-MQCRN9
    use: Production
  queueManager:
    name: EXAMPLEQM
    mqsc:
    - configMap:
        name: example-tls-configmap
        items:
        - example-tls.mqsc
    ini:
    - configMap:
        name: example-tls-configmap
        items:
        - example-tls.ini
    storage:
      queueManager:
        type: ephemeral
  version: 9.4.0.0-r1
  pki:
    keys:
      - name: default
        secret:
          secretName: example-qm-tls
          items:
            - tls.key
            - tls.crt
            - ca.crt
```

   Beachten Sie, dass der geheime Schlüssel *example-qm-tls* in Schritt „1” auf Seite 73und die ConfigMap *example-tls-configmap* in Schritt „2” auf Seite 73 erstellt wurden.

4. 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. For example:

```
oc get qmgr exampleqm
```

5. Testen Sie die Verbindung zum Warteschlangenmanager.

   Um sicherzustellen, dass der Warteschlangenmanager für die gegenseitige TLS-Kommunikation konfiguriert ist, führen Sie die Schritte in „Gegenseitige TLS-Verbindung zu einem Warteschlangenmanager von Ihrem Laptop testen” auf Seite 75aus.

Ergebnisse

Herzlichen Glückwunsch! Sie haben erfolgreich einen Warteschlangenmanager mit aktiviertem TLS implementiert, der Details aus dem TLS-Zertifikat verwendet, um sich beim Warteschlangenmanager zu authentifizieren und eine Identität bereitzustellen.

OpenShift CP4I Linux *Gegenseitige TLS-Verbindung zu einem Warteschlangenmanager von Ihrem Laptop testen*

Nachdem Sie mit IBM MQ Operatoreinen Warteschlangenmanager erstellt haben, können Sie testen, ob er funktioniert, indem Sie eine Verbindung zu ihm herstellen und eine Nachricht einreihen und abrufen. In dieser Task erfahren Sie, wie Sie mithilfe der IBM MQ -Beispielprogramme eine Verbindung herstellen, indem Sie sie auf einer Maschine außerhalb des Kubernetes -Clusters ausführen, z. B. auf Ihrem Laptop.

Vorbereitende Schritte

Die folgenden Schritte müssen als Voraussetzung für dieses Beispiel durchgeführt worden sein:

- Installieren Sie IBM MQ client. Sie benötigen die Befehle **amqsputc** und **amqsgetc** , die wie folgt als Teil von IBM MQ client installiert werden können:
  - Windows Linux Für Windows und Linux: Installieren Sie den IBM MQ Redistributable Client für Ihr Betriebssystem von https://ibm.biz/mq94redistclients
  - macOS Für Mac: Laden Sie IBM MQ MacOS Toolkitherunter und richten Sie es ein: https://developer.ibm.com/tutorials/mq-macos-dev/
- Stellen Sie sicher, dass Sie die erforderlichen Schlüssel-und Zertifikatsdateien in ein Verzeichnis auf Ihrer Maschine heruntergeladen und das Kennwort für den Keystore kennen. Diese Dateien werden beispielsweise in „Selbst signierte PKI mit OpenSSL erstellen” auf Seite 71erstellt:
  - `example-app1.p12`
  - `example-app1-chain.crt` (nur bei Verwendung von arm64 Apple Mac)
- Stellen Sie einen Warteschlangenmanager, der mit TLS konfiguriert ist, im OCP-Cluster bereit, indem Sie beispielsweise die Schritte in „Beispiel: Warteschlangenmanager mit gegenseitiger TLS-Authentifizierung konfigurieren” auf Seite 73 ausführen.

Informationen zu diesem Vorgang

In diesem Beispiel werden die IBM MQ -Beispielprogramme verwendet, die auf einer Maschine außerhalb des Kubernetes -Clusters ausgeführt werden (z. B. Ihr Laptop), um eine Verbindung zu einem QueueManager herzustellen, der mit TLS konfiguriert ist, und um Nachrichten einzureihen und abzurufen.

Vorgehensweise

1. 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. For example:

   ```
   oc get qmgr exampleqm
   ```

2. Suchen Sie den Hostnamen des Warteschlangenmanagers.

   Verwenden Sie den folgenden Befehl, um den vollständig qualifizierten Hostnamen des Warteschlangenmanagers für den Warteschlangenmanager von außerhalb des OCP-Clusters unter Verwendung der automatisch erstellten Route zu suchen: `exampleqm-ibm-mq-qm`:

   ```
   oc get route exampleqm-ibm-mq-qm --template="{{.spec.host}}"
   ```

3. IBM MQ -Definitionstabelle für Clientkanäle (CCDT) erstellen

   Erstellen Sie eine Datei namens `ccdt.json` mit folgendem Inhalt:

   ```
   {
       "channel":
       [
           {
               "name": "MTLS.SVRCONN",
               "clientConnection":
               {
   ```

```
                "connection":
                [
                    {
                        "host": "hostname from previous step",
                        "port": 443
                    }
                ],
                "queueManager": "EXAMPLEQM"
            },
            "transmissionSecurity":
            {
              "cipherSpecification": "ANY_TLS13",
              "certificateLabel": "example-app1"
            },
            "type": "clientConnection"
        }
   ]
}
```

Die Verbindung verwendet Port 443, da dies der Port ist, an dem der Red Hat OpenShift Container Platform -Router empfangsbereit ist. Der Datenverkehr wird an den Warteschlangenmanager an Port 1414 weitergeleitet.

Wenn Sie einen anderen Kanalnamen verwendet haben, müssen Sie diesen anpassen. Die Beispiele für gegenseitige TLS verwenden einen Kanal mit dem Namen *MTLS.SVRCONN*

Weitere Details finden Sie unter Configuring a JSON format CCDT .

4. Erstellen Sie eine Client-INI-Datei, um die Verbindungsdetails zu konfigurieren

   Erstellen Sie eine Datei mit dem Namen `mqclient.ini` im aktuellen Verzeichnis. Diese Datei wird von **amqsputc** und **amqsgetc**gelesen.

   ```
   Channels:
     ChannelDefinitionDirectory=.
     ChannelDefinitionFile=ccdt.json
   SSL:
     OutboundSNI=HOSTNAME
     SSLKeyRepository=example-app1.p12
     SSLKeyRepositoryPassword=password you used when creating the p12 file
   ```

   Stellen Sie sicher, dass Sie *SSLKeyRepositoryKennwort* auf das Kennwort aktualisieren, das Sie beim Erstellen der Datei PKCS#12 ausgewählt haben. Es gibt andere Möglichkeiten, das Keystore-Kennwort festzulegen, einschließlich der Verwendung eines verschlüsselten Kennworts. Weitere Informationen finden Sie unter Kennwort des Schlüsselrepositorys für einen IBM MQ MQI client unter AIX, Linux, and Windowsangeben

   Beachten Sie, dass Red Hat OpenShift Container Platform Router die SNI für die Weiterleitung von Anforderungen an den IBM MQ-Warteschlangenmanager verwendet. Das Attribut *OutboundSNI= HOSTNAME* stellt sicher, dass der IBM MQ -Client die erforderlichen Informationen enthält, damit der Router mit der von IBM MQ Operatorkonfigurierten Standardroute arbeiten kann. Weitere Informationen finden Sie unter „Route für Verbindung zu einem Warteschlangenmanager von außerhalb eines Red Hat OpenShift-Clusters konfigurieren” auf Seite 88.

5. Wenn Sie einen arm64 Apple Macverwenden, müssen Sie eine zusätzliche Umgebungsvariable konfigurieren.

   ```
   export MQSSLTRUSTSTORE=example-app1-chain.crt
   ```

   Diese Datei enthält die vollständige Zertifikatskette, einschließlich der Anwendungs-und CA-Zertifikate.

6. Reihen Sie Nachrichten in die Warteschlange ein.

   Führen Sie den folgenden Befehl aus:

   ```
   /opt/mqm/samp/bin/amqsputc EXAMPLE.QUEUE EXAMPLEQM
   ```

   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.

7. Rufen Sie die Nachrichten aus der Warteschlange ab.

   Führen Sie den folgenden Befehl aus:

   ```
   /opt/mqm/samp/bin/amqsgetc EXAMPLE.QUEUE EXAMPLEQM
   ```

   Die zuvor hinzugefügten Nachrichten wurden verarbeitet und werden nun ausgegeben. Der Befehl wird nach wenigen Sekunden automatisch beendet.

Ergebnisse

Herzlichen Glückwunsch! Sie haben die Verbindung zu einem Warteschlangenmanager mit aktiviertem TLS erfolgreich getestet und gezeigt, dass Sie Nachrichten sicher von einem Client einreihen und abrufen können.

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 Warteschlangenmanager-ressourcen hinzufügen” auf Seite 100 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 MQ Adv. Hohe Verfügbarkeit für Warteschlangenmanager mithilfe von IBM MQ Operator konfigurieren

Informationen zu diesem Vorgang

Prozedur

- „Native HA” auf Seite 21.
- „Beispiel: Native HA mit IBM MQ Operator konfigurieren” auf Seite 79.
- „Warteschlangenmanager mit mehreren Instanzen mit IBM MQ Operator konfigurieren” auf Seite 85.

OpenShift MQ Adv. *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-EHXT-MQCRN9
    use: Production
  queueManager:
    availability:
      type: NativeHA
  version: 9.4.0.0-r1
```

Das Feld `.spec.queueManager.availability.type`muss auf `NativeHA`gesetzt sein.

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: Native HA mit IBM MQ Operator konfigurieren” auf Seite 79 ist eine Schritt-für-Schritt-Anleitung verfügbar.

Zugehörige Tasks

„Beispiel: Native HA mit IBM MQ Operator konfigurieren” auf Seite 79
In diesem Beispiel wird ein WS-Manager mit der nativen Hochverfügbarkeitsfunktion in OpenShift Container Platform unter Verwendung von IBM MQ Operatorimplementiert. Gegenseitige TLS wird für die Authentifizierung verwendet, um ein TLS-Zertifikat einer Identität im Warteschlangenmanager zuzuordnen.

Beispiel: Native HA mit IBM MQ Operator konfigurieren

In diesem Beispiel wird ein WS-Manager mit der nativen Hochverfügbarkeitsfunktion in OpenShift Container Platform unter Verwendung von IBM MQ Operatorimplementiert. Gegenseitige TLS wird für die Authentifizierung verwendet, um ein TLS-Zertifikat einer Identität im Warteschlangenmanager zuzuordnen.

Vorbereitende Schritte

Die folgenden Schritte müssen als Voraussetzung für dieses Beispiel durchgeführt worden sein:

- Erstellen Sie für dieses Beispiel ein Projekt bzw. einen Namensbereich für 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 OpenShift Container Platform bereitzustellenden Warteschlangenmanager definiert. Darüber hinaus beschreibt es alle weiteren Schritte, die zur Bereitstellung des Warteschlangenmanagers mit aktiviertem TLS erforderlich sind.

Vorgehensweise

1. Erstellen Sie ein Zertifikatspaar wie in „Selbst signierte PKI mit OpenSSL erstellen” auf Seite 71beschrieben.
2. Konfigurationsübersicht mit MQSC-Befehlen und einer INI-Datei erstellen

Erstellen Sie eine Kubernetes ConfigMap mit den MQSC-Befehlen, um eine neue Warteschlange und einen SVRCONN-Kanal zu erstellen und einen Kanalauthentifizierungsdatensatz hinzuzufügen, der den Zugriff auf den Kanal ermöglicht.

Stellen Sie sicher, dass Sie sich in dem Namensbereich befinden, den Sie zuvor erstellt haben (siehe Vorbereitungen), und geben Sie dann die folgende YAML-Datei in der OCP-Webkonsole oder über die Befehlszeile ein.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: example-nativeha-configmap
data:
  example-tls.mqsc: |
    DEFINE CHANNEL('MTLS.SVRCONN') CHLTYPE(SVRCONN) SSLCAUTH(REQUIRED) SSLCIPH('A
NY_TLS13_OR_HIGHER') REPLACE
    SET CHLAUTH('MTLS.SVRCONN') TYPE(SSLPEERMAP) SSLPEER('CN=*') USERSRC(NOACCESS) ACTION(RE
PLACE)
    SET CHLAUTH('MTLS.SVRCONN') TYPE(SSLPEERMAP) SSLPEER('CN=example-app1') USERSRC(MAP)
MCAUSER('app1') ACTION(REPLACE)
    SET AUTHREC PRINCIPAL('app1') OBJTYPE(QMGR) AUTHADD(CONNECT,INQ)
    DEFINE QLOCAL('EXAMPLE.QUEUE') REPLACE
    SET AUTHREC PROFILE('EXAMPLE.QUEUE') PRINCIPAL('app1') OBJTYPE(QUEUE) AUTHADD(BROW
SE,PUT,GET,INQ)
  example-tls.ini: |
    Service:
        Name=AuthorizationService
        EntryPoints=14
        SecurityPolicy=UserExternal
```

Der MQSC definiert einen Kanal mit dem Namen *MTLS.SVRCONN* und eine Warteschlange mit dem Namen *EXAMPLE.QUEUE*. Der Kanal ist so konfiguriert, dass nur auf Clients zugegriffen werden kann, die ein Zertifikat mit dem allgemeinen Namen *example-app1*vorlegen. Dies ist der allgemeine Name, der in einem der in Schritt „1” auf Seite 79erstellten Zertifikate verwendet wird. Verbindungen dieses Kanals mit diesem allgemeinen Namen werden der Benutzer-ID *app1*zugeordnet, die berechtigt ist, eine Verbindung zum Warteschlangenmanager herzustellen und auf die Beispielwarteschlange zuzugreifen. Die INI-Datei aktiviert eine Sicherheitsrichtlinie, was bedeutet, dass die Benutzer-ID *app1* nicht in einer externen Benutzerregistry vorhanden sein muss, sondern nur als Name in dieser Konfiguration.

3. Stellen Sie den Warteschlangenmanager bereit.

   Erstellen Sie mit der folgenden angepassten YAML-Ressource einen neuen Warteschlangenmanager. Stellen Sie sicher, dass Sie sich in dem Namensbereich befinden, den Sie vor Beginn dieser Aufgabe erstellt haben. Geben Sie dann die folgende YAML-Datei in der OCP-Webkonsole oder über die Befehlszeile ein. Vergewissern Sie sich, dass die richtige Lizenz angegeben ist, und bestätigen Sie die Lizenz, indem Sie `false` in `true` ändern.

```
apiVersion: mq.ibm.com/v1beta1
kind: QueueManager
metadata:
  name: exampleqm
spec:
  license:
    accept: false
    license: L-EHXT-MQCRN9
    use: Production
  queueManager:
    name: EXAMPLEQM
    availability:
      type: NativeHA
      tls:
        secretName: example-qm-tls
    mqsc:
    - configMap:
        name: example-nativeha-configmap
        items:
        - example-tls.mqsc
    ini:
    - configMap:
        name: example-nativeha-configmap
        items:
        - example-tls.ini
    storage:
      queueManager:
```

```
      type: persistent-claim
version: 9.4.0.0-r1
pki:
  keys:
    - name: default
      secret:
        secretName: example-qm-tls
        items:
          - tls.key
          - tls.crt
          - ca.crt
```

Beachten Sie, dass der geheime Schlüssel *example-qm-tls* in Schritt „1” auf Seite 79und die Config-Map *example-nativeha-configmap* in Schritt „2” auf Seite 79 erstellt wurden.

Der Verfügbarkeitstyp wird auf *NativeHA*gesetzt und persistenter Speicher ausgewählt. Die in Ihrem Kubernetes -Cluster konfigurierte Standardspeicherklasse wird verwendet. Wenn Sie keine Speicherklasse als Standard konfiguriert haben oder eine andere Speicherklasse verwenden möchten, fügen Sie `defaultClass: storage_class_name`unter `spec.queueManager.storage`ehinzu.

Die drei Pods in einem Warteschlangenmanager mit Native HA replizieren Daten über das Netz. Diese Verbindung ist standardmäßig nicht verschlüsselt, aber in diesem Beispiel wird das Zertifikat des Warteschlangenmanagers für die Verschlüsselung des Datenverkehrs verwendet. Für zusätzliche Sicherheit können Sie ein anderes Zertifikat angeben. Der geheime TLS-Schlüssel für die native Hochverfügbarkeit muss ein geheimer Kubernetes -TLS-Schlüssel mit einer bestimmten Struktur sein (beispielsweise muss der private Schlüssel *tls.key*genannt werden).

4. 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. For example:

   ```
   oc get qmgr exampleqm
   ```

5. Testen Sie die Verbindung zum Warteschlangenmanager.

   Um zu bestätigen, dass der Warteschlangenmanager konfiguriert und verfügbar ist, führen Sie die Schritte in „Gegenseitige TLS-Verbindung zu einem Warteschlangenmanager von Ihrem Laptop testen” auf Seite 75aus.

6. Erzwingen Sie einen Ausfall des aktiven Pods.

   Simulieren Sie einen Pod-Ausfall, um die automatische Wiederherstellung des Warteschlangenmanagers zu überprüfen:

   a) Zeigen Sie den aktiven und den Standby-Pod an.

      Führen Sie den folgenden Befehl aus:

      ```
      oc get pods --selector app.kubernetes.io/instance=exampleqm
      ```

      Beachten Sie, dass der aktive Pod im Feld **READY**den Wert `1/1`zurückgibt, während die Replikatpods den Wert `0/1`zurückgeben.

   b) Löschen Sie den aktiven Pod.

      Führen Sie folgenden Befehl aus und geben Sie dabei den vollständigen Namen des aktiven Pods an:

      ```
      oc delete pod exampleqm-ibm-mq-value
      ```

   c) Zeigen Sie den Pod-Status erneut an.

      Führen Sie den folgenden Befehl aus:

      ```
      oc get pods --selector app.kubernetes.io/instance=exampleqm
      ```

   d) 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 EXAMPLEQM
```

Der Status sollte anzeigen, dass sich die aktive Instanz geändert hat, z. B.:

```
QMNAME(EXAMPLEQM) ROLE(Active) INSTANCE(inst1) INSYNC(Yes) QUORUM(3/3)
INSTANCE(inst1) ROLE(Active) REPLADDR(9.20.123.45) CONNACTV(Yes) INSYNC(Yes) BACKLOG(0)
CONNINST(Yes) ALTDATE(2022-01-12) ALTTIME(12.03.44)
INSTANCE(inst2) ROLE(Replica) REPLADDR(9.20.123.46) CONNACTV(Yes) INSYNC(Yes) BACKLOG(0)
CONNINST(Yes) ALTDATE(2022-01-12) ALTTIME(12.03.44)
INSTANCE(inst3) ROLE(Replica) REPLADDR(9.20.123.47) CONNACTV(Yes) INSYNC(Yes) BACKLOG(0)
CONNINST(Yes) ALTDATE(2022-01-12) ALTTIME(12.03.44)
```

e) Testen Sie die Verbindung zum Warteschlangenmanager erneut.

Um zu bestätigen, dass der Warteschlangenmanager wiederhergestellt wurde, führen Sie die Schritte in „Gegenseitige TLS-Verbindung zu einem Warteschlangenmanager von Ihrem Laptop testen” auf Seite 75aus.

Ergebnisse

Herzlichen Glückwunsch! Sie haben erfolgreich einen Warteschlangenmanager mit nativer Hochverfügbarkeit und gegenseitiger TLS-Authentifizierung implementiert und überprüft, dass er automatisch wiederhergestellt wird, wenn der aktive Pod fehlschlägt.

OpenShift MQ Adv. *Status von nativen HA-Warteschlangenmanagern für IBM MQ -Container anzeigen*

Für 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

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 Tasks

„Beispiel: Native HA mit IBM MQ Operator konfigurieren” auf Seite 79
In diesem Beispiel wird ein WS-Manager mit der nativen Hochverfügbarkeitsfunktion in OpenShift Container Platform unter Verwendung von IBM MQ Operatorimplementiert. Gegenseitige TLS wird für die Authentifizierung verwendet, um ein TLS-Zertifikat einer Identität im Warteschlangenmanager zuzuordnen.

Zugehörige Verweise

Befehl dspmq (Warteschlangenmanager anzeigen)

OpenShift MQ Adv. *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 69 .

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.

OpenShift MQ Adv. ***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.

OpenShift CP4I MQ Adv. Kubernetes ***Warteschlangenmanager mit mehreren Instanzen mit IBM MQ Operator konfigurieren***

In diesem Beispiel wird ein Warteschlangenmanager mit mehreren Instanzen unter Verwendung von IBM MQ Operatorin OpenShift Container Platform implementiert. Gegenseitige TLS wird für die Authentifizierung verwendet, um ein TLS-Zertifikat einer Identität im Warteschlangenmanager zuzuordnen.

Vorbereitende Schritte

Die folgenden Schritte müssen als Voraussetzung für dieses Beispiel durchgeführt worden sein:

- Erstellen Sie für dieses Beispiel ein Projekt bzw. einen Namensbereich für 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 OpenShift Container Platform bereitzustellenden Warteschlangenmanager definiert. Darüber hinaus beschreibt es alle weiteren Schritte, die zur Bereitstellung des Warteschlangenmanagers mit aktiviertem TLS erforderlich sind.

Vorgehensweise

1. Geeignete Speicherklasse ermitteln

   Auf den Speicher in einem Kubernetes -Cluster kann über mehrere Zugriffsmodi für persistente Datenträgerzugegriffen werden. Ein Warteschlangenmanager mit mehreren Instanzen erstellt mehrere persistente Datenträger: einen für jeden Warteschlangenmanager und mindestens einen gemeinsam genutzten Datenträger. Der gemeinsam genutzte Datenträger für einen Warteschlangenmanager mit mehreren Instanzen muss eine `ReadWriteMany` -Speicherklasse verwenden. Die Standardspeicherklasse in einem Kubernetes -Cluster gilt normalerweise für eine `ReadWriteOnce` -Speicherklasse (Blockspeicher). Wenn Sie beispielsweise Red Hat OpenShift Data Foundationverwenden, stellt die Speicherklasse *ocs-storagecluster-cephfs* ein geeignetes gemeinsam genutztes Dateisystem bereit. Die Auswahl des Dateisystems ist sehr wichtig, da nicht alle gemeinsam genutzten Dateisysteme das Sperren von Dateien auf dieselbe Weise handhaben. Siehe Planning file system support on Multiplatforms and Testing statement for IBM MQ multi-instance queue manager file systems.
2. Erstellen Sie ein Zertifikatspaar wie in „Selbst signierte PKI mit OpenSSL erstellen” auf Seite 71beschrieben.
3. Konfigurationsübersicht mit MQSC-Befehlen und einer INI-Datei erstellen

   Erstellen Sie eine Kubernetes ConfigMap mit den MQSC-Befehlen, um eine neue Warteschlange und einen SVRCONN-Kanal zu erstellen und einen Kanalauthentifizierungsdatensatz hinzuzufügen, der den Zugriff auf den Kanal ermöglicht.

   Stellen Sie sicher, dass Sie sich in dem Namensbereich befinden, den Sie zuvor erstellt haben (siehe Vorbereitungen), und geben Sie dann die folgende YAML-Datei in der OCP-Webkonsole oder über die Befehlszeile ein.

   ```
   apiVersion: v1
   kind: ConfigMap
   metadata:
     name: example-miqm-configmap
   data:
     example-tls.mqsc: |
       DEFINE CHANNEL('MTLS.SVRCONN') CHLTYPE(SVRCONN) SSLCAUTH(REQUIRED) SSLCIPH('A□
   NY_TLS13_OR_HIGHER') REPLACE
       SET CHLAUTH('MTLS.SVRCONN') TYPE(SSLPEERMAP) SSLPEER('CN=*') USERSRC(NOACCESS) ACTION(RE□
   PLACE)
       SET CHLAUTH('MTLS.SVRCONN') TYPE(SSLPEERMAP) SSLPEER('CN=example-app1') USERSRC(MAP)
   MCAUSER('app1') ACTION(REPLACE)
       SET AUTHREC PRINCIPAL('app1') OBJTYPE(QMGR) AUTHADD(CONNECT,INQ)
       DEFINE QLOCAL('EXAMPLE.QUEUE') REPLACE
       SET AUTHREC PROFILE('EXAMPLE.QUEUE') PRINCIPAL('app1') OBJTYPE(QUEUE) AUTHADD(BROW□
   SE,PUT,GET,INQ)
     example-tls.ini: |
       Service:
           Name=AuthorizationService
           EntryPoints=14
           SecurityPolicy=UserExternal
   ```

   Der MQSC definiert einen Kanal mit dem Namen *MTLS.SVRCONN* und eine Warteschlange mit dem Namen *EXAMPLE.QUEUE*. Der Kanal ist so konfiguriert, dass nur auf Clients zugegriffen werden kann, die ein Zertifikat mit dem allgemeinen Namen *example-app1*vorlegen. Dies ist der allgemeine Name, der in einem der in Schritt „2” auf Seite 86erstellten Zertifikate verwendet wird. Verbindungen dieses Kanals mit diesem allgemeinen Namen werden der Benutzer-ID *app1*zugeordnet, die berechtigt ist, eine Verbindung zum Warteschlangenmanager herzustellen und auf die Beispielwarteschlange zuzugreifen. Die INI-Datei aktiviert eine Sicherheitsrichtlinie, was bedeutet, dass die Benutzer-ID *app1* nicht in einer externen Benutzerregistry vorhanden sein muss, sondern nur als Name in dieser Konfiguration.
4. Stellen Sie den Warteschlangenmanager bereit.

   Erstellen Sie mit der folgenden angepassten YAML-Ressource einen neuen Warteschlangenmanager. Stellen Sie sicher, dass Sie sich in dem Namensbereich befinden, den Sie vor Beginn dieser Aufgabe erstellt haben. Geben Sie dann die folgende YAML-Datei in der OCP-Webkonsole oder über die Befehlszeile ein. Vergewissern Sie sich, dass die richtige Lizenz angegeben ist, und bestätigen Sie die Lizenz, indem Sie `false` in `true` ändern.

   ```
   apiVersion: mq.ibm.com/v1beta1
   kind: QueueManager
   ```

```
metadata:
  name: exampleqm
spec:
  license:
    accept: false
    license: L-EHXT-MQCRN9
    use: Production
  queueManager:
    name: EXAMPLEQM
    availability:
      type: MultiInstance
    mqsc:
    - configMap:
        name: example-miqm-configmap
        items:
        - example-tls.mqsc
    ini:
    - configMap:
        name: example-miqm-configmap
        items:
        - example-tls.ini
    storage:
      defaultClass: STORAGE_CLASS
  version: 9.4.0.0-r1
  pki:
    keys:
      - name: default
        secret:
          secretName: example-qm-tls
          items:
            - tls.key
            - tls.crt
            - ca.crt
```

Ändern Sie *STORAGE_CLASS* in die in Schritt „1” auf Seite 86angegebene Speicherklasse.

Beachten Sie, dass der geheime Schlüssel *example-qm-tls* in Schritt „2” auf Seite 86und die Config-Map *example-miqm-configmap* in Schritt „3” auf Seite 86 erstellt wurden.

Der Verfügbarkeitstyp wird auf *MultiInstance*gesetzt, wodurch der persistente Speicher automatisch ausgewählt wird.

5. 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. For example:

   ```
   oc get qmgr exampleqm
   ```

6. Testen Sie die Verbindung zum Warteschlangenmanager.

   Um zu bestätigen, dass der Warteschlangenmanager konfiguriert und verfügbar ist, führen Sie die Schritte in „Gegenseitige TLS-Verbindung zu einem Warteschlangenmanager von Ihrem Laptop testen” auf Seite 75aus.

7. Erzwingen Sie einen Ausfall des aktiven Pods.

   Simulieren Sie einen Pod-Ausfall, um die automatische Wiederherstellung des Warteschlangenmanagers zu überprüfen:

   a) Zeigen Sie den aktiven und den Standby-Pod an.

      Führen Sie den folgenden Befehl aus:

      ```
      oc get pods --selector app.kubernetes.io/instance=exampleqm
      ```

      Beachten Sie, dass der aktive Pod im Feld **READY**den Wert `1/1`zurückgibt, während der Standby-Pod den Wert `0/1`zurückgibt.

   b) Löschen Sie den aktiven Pod.

      Führen Sie folgenden Befehl aus und geben Sie dabei den vollständigen Namen des aktiven Pods an:

      ```
      oc delete pod exampleqm-ibm-mq-value
      ```

c) Zeigen Sie den Pod-Status erneut an.

Führen Sie den folgenden Befehl aus:

```
oc get pods --selector app.kubernetes.io/instance=exampleqm
```

d) Zeigen Sie den Status des Warteschlangenmanagers an.

Führen Sie den folgenden Befehl aus und geben Sie dabei den vollständigen Namen des anderen Pods an:

```
oc exec -t Pod -- dspmq -x
```

Der Status sollte anzeigen, dass sich die aktive Instanz geändert hat, z. B.:

```
QMNAME(EXAMPLEQM)                                         STATUS(Running as standby)
    INSTANCE(exampleqm-ibm-mq-1) MODE(Active)
    INSTANCE(exampleqm-ibm-mq-0) MODE(Standby)
```

e) Testen Sie die Verbindung zum Warteschlangenmanager erneut.

Um zu bestätigen, dass der Warteschlangenmanager wiederhergestellt wurde, führen Sie die Schritte in „Gegenseitige TLS-Verbindung zu einem Warteschlangenmanager von Ihrem Laptop testen" auf Seite 75aus.

Ergebnisse

Herzlichen Glückwunsch! Sie haben erfolgreich einen Multi-Instanz-Warteschlangenmanager mit gegenseitiger TLS-Authentifizierung implementiert und überprüft, ob er automatisch wiederhergestellt wird, wenn der aktive Pod fehlschlägt.

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

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.

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.

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

Der IBM MQ verwaltete .NET -Client setzt SERVERNAME auf den entsprechenden Hostnamen, wenn die Eigenschaft **OutboundSNI** auf HOSTNAMEgesetzt ist, wodurch ein IBM MQ verwalteter .NET -Client über Red Hat OpenShift -Routen eine Verbindung zu einem Warteschlangenmanager herstellen kann.

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.

Es ist wichtig, dass die Namen von MQ -Kanälen nicht in Kleinbuchstaben enden, da IBM MQ Kanalnamen SNI-Headern zuordnet.

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: unique_name_for_the_route
  namespace: namespace_of_your_MQ_deployment
spec:
  host: SNI_address_mapping_for_the_channel
  to:
    kind: Service
    name: name_of_Kubernetes_Service_for_your_MQ_deployment (for example "queue_manager_name-
ibm-mq")
  port:
    targetPort: 1414
  tls:
    termination: passthrough
```

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 134
Vorgehensweise zum Herstellen einer Verbindung zur IBM MQ Console eines Warteschlangenmanagers, der in einem Red Hat OpenShift Container Platform -Cluster implementiert wurde.

OpenShift CP4I IBM MQ mit IBM Instana -Traceerstellung integrieren

IBM Instana kann verwendet werden, um Transaktionen in IBM Cloud Pak for Integrationzu verfolgen.

Vorbereitende Schritte

In diesem Dokument wird die IBM Instana -Traceerstellung behandelt. Dies ist der Prozess, bei dem Nachrichten über ein System verfolgt werden. Es umfasst nicht die IBM Instana -Überwachung, bei der Details zum Status eines IBM MQ -Warteschlangenmanagers abgerufen werden. Weitere Informationen zur Überwachung von IBM MQ durch IBM Instana finden Sie im Abschnitt IBM MQ. Ausführliche Anweisungen zur authentifizierten Überwachung finden Sie unter „Authentifizierte IBM Instana -Überwachung mit TLS konfigurieren" auf Seite 92.

Anmerkung:

- Diese Funktion wird nur in Operanden von IBM MQ Version 9.3.1.0-r2 oder höher unterstützt.
- Sie können die IBM Instana -Traceerstellung in früheren Versionen von IBM MQ Operator und Warteschlangenmanager ausführen, jedoch nicht nativ. Weitere Informationen finden Sie unter Configuring IBM MQ Tracing in der IBM Instana -Dokumentation.

Bevor Sie die IBM Instana -Traceerstellung mit IBM MQ Operator ausführen können, müssen Sie sowohl ein IBM Instana -Back-End als auch IBM Instana -Agenten implementieren. Standardmäßig kommuniziert ein Warteschlangenmanager von IBM MQ mit einem IBM Instana -Agenten, der auf demselben Knoten wie der Warteschlangenmanager-Pod implementiert ist.

Informationen zu diesem Vorgang

Wenn Sie die Integration mit IBM Instana aktivieren, wird ein IBM MQ -API-Exit in Ihrem Warteschlangenmanager installiert. Der API-Exit sendet Tracedaten zu Nachrichten, die durch den Warteschlangenmanager fließen, an IBM Instana -Agenten.

Der API-Exit fügt jeder Nachricht RFH2 -Header hinzu. Diese Header enthalten Traceinformationen.

Die IBM Instana -Agenten sind für das Senden der Tracedaten an das IBM Instana -Back-End verantwortlich.

Informationen zum Implementieren eines IBM Instana -Back-End und IBM Instana -Agenten finden Sie unter Instana-Überwachungslinks in der Plattformbenutzerschnittstelle aktivieren in der IBM Instana -Dokumentation.

Prozedur

Standardimplementierung

- Implementieren Sie einen Warteschlangenmanager mit aktivierter IBM Instana -Traceerstellung.

  Standardmäßig ist das IBM Instana -Tracing inaktiviert.

  Wenn Sie die IBM Cloud Pak for Integration Platform UI oder die OpenShift -Webkonsole verwenden:

  1. Klicken Sie auf **Telemetrie** > **Tracefunktion** > **Instana**.
  2. Setzen Sie die Umschaltfläche **Traceerstellung für Instanz aktivieren** auf `true`.

  Wenn Sie über YAML bereitstellen, verwenden Sie das folgende Snippet:

```
spec:
  telemetry:
    tracing:
      instana:
        enabled: true
```

Erweiterte Bereitstellung

- Kommunikation mit dem IBM Instana -Agenten über HTTPS.

  Standardmäßig kommuniziert der IBM Instana -Exit für IBM MQ über HTTP mit dem IBM Instana -Agenten. Die Hostadresse des Agenten wird auf die IP-Adresse des Knotens gesetzt, auf dem der Warteschlangenmanager ausgeführt wird. Dieser Wert entspricht der Konfiguration, die im Abschnitt IBM Instana -Überwachung aktivieren in der IBM Instana -Dokumentation beschrieben ist. Dabei werden IBM Instana -Agenten vom IBM Instana -Agentenoperator als DaemonSet bereitgestellt.

  Derzeit unterstützt die Kommunikation zwischen dem IBM Instana -Exit für IBM MQ und dem IBM Instana -Agenten HTTP-oder HTTPS-Protokolle. Für die Verwendung von HTTPS muss der IBM Instana -Agent zuerst für die Verwendung der TLS-Verschlüsselung konfiguriert werden. Siehe TLS-Verschlüsselung für Agentenendpunkt einrichten in der IBM Instana -Dokumentation. Das Protokoll kann dann wie folgt auf https gesetzt werden:

  Wenn Sie die OpenShift -Webkonsole verwenden:

  1. Klicken Sie auf **Telemetrie** > **Instanziierung**.
  2. Erweitern Sie die Dropdown-Liste **Erweiterte Konfiguration** .
  3. Setzen Sie **Instana agent communication protocol** auf `https`.

  Wenn Sie über YAML bereitstellen, verwenden Sie das folgende Snippet:

```
spec:
  telemetry:
    instana:
      enabled: true
      protocol: https
```

- Legen Sie die **`agentHost`** fest.

Wenn IBM Instana -Agenten nicht als DaemonSet im OpenShift-Cluster bereitgestellt wurden, in dem der Warteschlangenmanager ausgeführt wird, müssen Sie den Wert für **agentHost** auf den Hostnamen oder die IP-Adresse setzen, unter dem bzw. der der IBM Instana -Agent ausgeführt wird. Der Wert **agentHost** darf kein Protokoll oder Port enthalten.

Wenn Sie die OpenShift -Webkonsole verwenden:

1. Klicken Sie auf **Telemetrie** > **Instanziierung**.
2. Erweitern Sie die Dropdown-Liste **Erweiterte Konfiguration** .
3. Geben Sie Ihren Hostnamen in das Textfeld **Instana agent host** ein.

Wenn Sie über YAML bereitstellen, verwenden Sie das folgende Snippet:

```
spec:
  telemetry:
    instana:
      enabled: true
      agentHost: 9.9.9.9
```

Nächste Schritte

Weitere Informationen hierzu finden Sie im Abschnitt „Einfachen Warteschlangenmanager mit IBM MQ Operator implementieren” auf Seite 66.

OpenShift CP4I Operator2.2.0 Authentifizierte IBM Instana -Überwachung mit TLS konfigurieren

Um einen Warteschlangenmanager über einen IBM Instana -Agenten überwachen zu können, müssen Sie sowohl den Agenten als auch den Warteschlangenmanager konfigurieren.

Vorbereitende Schritte

Der Abschnitt "Configuration" von "Monitoring IBM MQ" in der IBM Instana -Dokumentation enthält allgemeine Informationen zur IBM Instana -Überwachungskonfiguration. Es enthält jedoch keine Details zur Konfiguration des Warteschlangenmanagers.

Bevor Sie die IBM Instana -Traceerstellung mit IBM MQ Operator ausführen können, müssen Sie sowohl ein IBM Instana -Back-End als auch IBM Instana -Agenten implementieren. Informationen hierzu finden Sie unter Überwachung von IBM Instana in der Benutzerschnittstelle von CP4I Platform aktivieren in der IBM Instana -Dokumentation.

Vorgehensweise

1. Zertifikate generieren.
2. Konfigurieren Sie die IBM Instana -Agenten.
3. Konfigurieren Sie den Warteschlangenmanager..
4. Prüfen und debuggen.

Zugehörige Tasks

„IBM MQ mit IBM Instana -Traceerstellung integrieren” auf Seite 90
IBM Instana kann verwendet werden, um Transaktionen in IBM Cloud Pak for Integrationzu verfolgen.

OpenShift CP4I Operator2.2.0 *Zertifikat und Schlüssel für den IBM Instana -Agenten und den Warteschlangenmanager generieren*

Für die TLS-Kommunikation zwischen dem IBM Instana -Agenten und dem Warteschlangenmanager müssen beide ein Zertifikat und einen entsprechenden privaten Schlüssel haben.

Vorbereitende Schritte

Dies ist die erste von vier Tasks zum Konfigurieren der authentifizierten IBM Instana Überwachung mit TLS.

Anmerkung: Die Werte, die bei der Generierung dieser Zertifikate verwendet werden, dienen Demonstrationszwecken. Stellen Sie bei der Implementierung in einer Produktionsumgebung sicher, dass Subjekt und Ablauf des Zertifikats angemessen sind.

Vorgehensweise

IBM MQ Warteschlangenmanager

Für die Kommunikation mit dem IBM Instana -Agenten über TLS muss der Warteschlangenmanager über ein Zertifikat und einen entsprechenden privaten Schlüssel verfügen. Wenn Sie diese bereits haben, überspringen Sie diesen Abschnitt.

1. Generieren Sie ein Zertifikat und einen privaten Schlüssel für den Warteschlangenmanager.

   Führen Sie den folgenden Befehl aus:

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

IBM Instana Agent

Damit der Agent die TLS-Kommunikation mit dem Warteschlangenmanager IBM MQ ausführen kann, muss er über ein Zertifikat und einen entsprechenden privaten Schlüssel verfügen. Wenn Sie bereits einen privaten Schlüssel und ein Zertifikat in einem JKS-Keystore haben, den Sie verwenden möchten, überspringen Sie diesen Abschnitt.

2. Generieren Sie ein Zertifikat und einen privaten Schlüssel für den IBM Instana -Agenten.

   Führen Sie den folgenden Befehl aus:

   ```
   openssl req \
          -newkey rsa:2048 -nodes -keyout application.key \
          -subj "/CN=instana-agent/OU=app team1" \
          -x509 -days 3650 -out application.crt
   ```

3. Speichern Sie das Zertifikat und den privaten Schlüssel in einem PKCS12 -Keystore.

   Führen Sie den folgenden Befehl aus und ersetzen Sie *ihr_kennwort* durch das Kennwort, das Sie zum Sichern des Keystores verwenden möchten. Führen Sie diesen Austausch in allen nachfolgenden Schritten durch.

   ```
   openssl pkcs12 -export -out application.p12 -inkey application.key -in application.crt -pas
   sout pass:your_password
   ```

4. Konvertieren Sie den Keystore PKCS12 in einen JKS-Keystore.

   Führen Sie den folgenden Befehl aus:

   ```
   keytool -importkeystore \
           -srckeystore application.p12 \
           -srcstoretype pkcs12 \
           -destkeystore application.jks \
           -deststoretype JKS \
           -srcstorepass your_password \
           -deststorepass your_password \
           -noprompt
   ```

5. Bezeichnen Sie das Zertifikat.

   Führen Sie den folgenden Befehl aus:

   ```
   keytool -changealias -alias "1" -destalias "instana" -keypass your_password -keystore appli
   cation.jks -storepass your_password -noprompt
   ```

6. Importieren Sie das Warteschlangenmanagerzertifikat in den Keystore.

   Führen Sie den folgenden Befehl aus:

   ```
   keytool -importcert -file server.crt -keystore application.jks -storepass your_password -ali
   as myca -noprompt
   ```

Nächste Schritte

Jetzt können Sie die Agenten für die Überwachung von IBM Instana konfigurieren.

OpenShift CP4I Operator 2.2.0 ***Instana-Überwachung: Agenten konfigurieren***

Hängen Sie den Keystore an die IBM Instana -Agenten an und konfigurieren Sie anschließend die Überwachung für einen bestimmten Warteschlangenmanager.

Vorbereitende Schritte

Bei dieser Task wird vorausgesetzt, dass Sie ein Zertifikat und einen Schlüssel für die IBM Instana -Agenten und den Warteschlangenmanagergeneriert haben.

Vorgehensweise

Keystore an die IBM Instana -Agenten anhängen

1. Erstellen Sie einen geheimen Schlüssel aus Ihrem JKS-Keystore im Namensbereich des IBM Instana -Agenten.

   Führen Sie den folgenden Befehl aus und ersetzen Sie *Name_des_geheimen_Schlüsselspeichers* durch den Namen, den Sie verwenden möchten. Führen Sie diesen Austausch in allen nachfolgenden Schritten durch.

   ```
   oc create secret generic keystore_secret_name --from-file=./application.jks -n instana-agent
   ```

2. Verwenden Sie im Namensbereich 'instana-agent' den Befehl `oc edit daemonset instana-agent` , um das DaemonSet 'instana-agent' so zu bearbeiten, dass es den folgenden zusätzlichen volumeMount und den folgenden Datenträger enthält:

   ```
   volumeMounts:
   - name: mq-key-jks-name
      subPath: application.jks
      mountPath: /opt/instana/agent/etc/application.jks
   volumes:
   - name: mq-key-jks-name
      secret:
         secretName: keystore_secret_name
   ```

Überwachung für einen bestimmten Warteschlangenmanager konfigurieren

3. Verwenden Sie im Namensbereich 'instana-agent' den Befehl `oc edit configmap instana-agent` , um die Konfigurationszuordnung 'instana-agent' zu bearbeiten.
4. Fügen Sie den folgenden Abschnitt unter `configuration.yaml: |` hinzu. Wenn Sie diesen Abschnitt bereits definiert haben, fügen Sie einfach den neuen WS-Manager zur Liste hinzu.

   ```
       com.instana.plugin.ibmmq:
         enabled: true
         poll_rate: 60
         queueManagers:
            QUEUE_MANAGER_NAME:
               channel: 'INSTANA.A.SVRCONN'
               keystorePassword: 'your_password'
               keystore: '/opt/instana/agent/etc/application.jks'
               cipherSuite: 'TLS_RSA_WITH_AES_256_CBC_SHA256'
   ```

   Dabei gilt Folgendes:

   - *your_password* ist das Kennwort für Ihren JKS-Keystore.

- *QUEUE_MANAGER_NAME* ist der Name des zu implementierenden IBM MQ -Warteschlangenmanagers und nicht der Name des WS-Manager-Operanden.

Anmerkung: Wenn *QUEUE_MANAGER_NAME* nicht auf den Namen des zugrunde liegenden Warteschlangenmanagers und stattdessen auf den Operanden gesetzt ist, funktioniert die Überwachung nicht. Der zugrunde liegende Name ist in `spec.queuemanager.name` für den Operanden des Warteschlangenmanagers definiert.

5. Löschen Sie die Pods 'instana-agent' im Namensbereich 'instana-agent'. Dies führt dazu, dass sie erneut gestartet werden und die Überwachung mit den neuen Einstellungen beginnt.

Nächste Schritte

Jetzt können Sie den Warteschlangenmanager für IBM Instana -Überwachung konfigurieren.

OpenShift CP4I Operator2.2.0 *Instana-Überwachung: Warteschlangenmanager konfigurieren*

Richten Sie einen Warteschlangenmanager ein, der TLS für die Kommunikation mit dem IBM Instana -Agenten verwendet.Die Authentifizierung für diese Verbindung erfolgt unter Verwendung von SSLPEERMAP.

Vorbereitende Schritte

Diese Task setzt voraus, dass Sie die Agenten für die IBM Instana -Überwachung konfiguriert haben.

Vorgehensweise

1. Konfigurieren Sie den Warteschlangenmanager über MQSC und INI.

   MQSC wird verwendet, um einen neuen TLS-fähigen Kanal einzurichten, und dann diesen Kanal so konfigurieren, dass er den verbindenden IBM Instana -Agenten authentifiziert, wenn er über ein Zertifikat mit den erforderlichen Feldern verfügt. In diesem Fall wird jeder verbundene Client mit einem Zertifikat, das die Felder `CN=instana-agent,OU=app team1` enthält, dem Benutzer `app1`zugeordnet. Anschließend erteilt MQSC dem Benutzer `app1` die Berechtigung, die erforderlichen Operationen für die IBM Instana -Überwachung auszuführen.

   Die INI-Datei wird verwendet, um dem externen Benutzer `app1`Berechtigungen zu erteilen.

   Die folgende ConfigMap enthält die erforderlichen MQSC-und INI-Einstellungen. Stellen Sie sie im Namensbereich Ihres Warteschlangenmanagers bereit.

```
apiVersion: v1
data:
  channel.mqsc: |-
    DEFINE CHANNEL('INSTANA.A.SVRCONN') CHLTYPE(SVRCONN) SSLCAUTH(REQUIRED) SSLCIPH('A
NY_TLS12_OR_HIGHER')
    ALTER QMGR CONNAUTH(' ')
    REFRESH SECURITY
    SET CHLAUTH('INSTANA.A.SVRCONN') TYPE(SSLPEERMAP) SSLPEER('CN=*') USERSRC(NOACCESS) AC
TION(REPLACE)
    SET CHLAUTH('*') TYPE(ADDRESSMAP) ADDRESS('*') USERSRC(NOACCESS) ACTION(REPLACE)
    SET CHLAUTH('INSTANA.A.SVRCONN') TYPE(SSLPEERMAP) SSLPEER('CN=instana-agent,OU=app
team1') USERSRC(MAP) MCAUSER('app1')
    SET AUTHREC PRINCIPAL('app1') OBJTYPE(QMGR) AUTHADD(ALL)
    SET AUTHREC PROFILE('SYSTEM.ADMIN.COMMAND.QUEUE') PRINCIPAL('app1') OBJTYPE(QUEUE) AU
THADD(PUT,INQ,DSP,CHG)
    SET AUTHREC PROFILE('SYSTEM.**') PRINCIPAL('app1') OBJTYPE(TOPIC) AUTHADD(DSP)
    SET AUTHREC PROFILE('*') PRINCIPAL('app1') OBJTYPE(TOPIC) AUTHADD(DSP)
    SET AUTHREC PROFILE('SYSTEM.**') PRINCIPAL('app1') OBJTYPE(QUEUE) AUTHADD(DSP, CHG, GET)
    SET AUTHREC PROFILE('SYSTEM.**') PRINCIPAL('app1') OBJTYPE(LISTENER) AUTHADD(DSP)
    SET AUTHREC PROFILE('AMQ.*') PRINCIPAL('app1') OBJTYPE(QUEUE) AUTHADD(DSP, CHG)
    REFRESH SECURITY TYPE(CONNAUTH)
  auth.ini: |-
    Service:
      Name=AuthorizationService
      EntryPoints=14
      SecurityPolicy=UserExternal
kind: ConfigMap
metadata:
```

```
  namespace: your-queue-manager-namespace
  name: qmgr-monitoring-config
```

Dabei ist *namensbereich-des-warteschlangenmanagers* der Namensbereich, in dem Ihr Warteschlangenmanager bereitgestellt wird.

Anmerkung: Wenn Sie benutzerdefinierte Warteschlangen überwachen, müssen Sie der ConfigMap MQSC zusätzliche Zeilen hinzufügen und diesen Warteschlangen DSP-, CHG-und GET-Berechtigungen erteilen. For example:

```
SET AUTHREC PROFILE('MYQUEUE') PRINCIPAL('app1') OBJTYPE(QUEUE) AUTHADD(DSP, CHG, GET).
```

In diesem Beispiel wird eine ConfigMap für die MQSC-und INI-Daten verwendet, aber Sie können einen geheimen Schlüssel verwenden, wenn Zusätze, die Sie vornehmen, vertraulich sind. Allgemeine Informationen zur Implementierung mit MQSC und INI finden Sie unter „Beispiel: Unterstützung von MQSC- und INI-Dateien” auf Seite 69.

2. Damit eine TLS-Verbindung hergestellt werden kann, muss der Warteschlangenmanager dem Zertifikat des IBM Instana -Agenten vertrauen. Erstellen Sie dazu einen geheimen Schlüssel, der nur das Zertifikat des IBM Instana -Agenten enthält:

```
oc create secret generic instana-certificate-secret --from-file=./application.crt -n your-
queue-manager-namespace
```

3. Der Warteschlangenmanager muss ein eigenes Zertifikat für den TLS-Handshake vorlegen und benötigt Zugriff auf den zugehörigen privaten Schlüssel. Stellen Sie einen geheimen Schlüssel bereit, der den Schlüssel und das Zertifikat enthält, die Sie entweder zuvor erstellt haben oder bereits besitzen:

```
oc create secret tls qm-tls-secret --cert server.crt --key server.key -n your-queue-manager-
namespace
```

Nachdem die Konfigurationszuordnung und der geheime Schlüssel erstellt wurden, können Sie den Warteschlangenmanager selbst erstellen.

4. Stellen Sie sicher, dass Ihr Warteschlangenmanager-YAML die Umgebungsvariable **MQSNOAUT** im Warteschlangenmanager-Container nicht definiert.

   Andernfalls funktioniert das Authentifizierungsverfahren nach seiner Aktivierung nicht mehr. Wird die Variable nach der Implementierung entfernt, wird der Mechanismus nicht erneut aktiviert und der Warteschlangenmanager muss erneut erstellt werden.

5. Fügen Sie Ihrer Warteschlangenmanagerdefinition die folgenden Abschnitte hinzu, wobei *MYQM* der Name Ihres Warteschlangenmanagers ist:

```
spec:
  queueManager:
    name: MYQM   #(a)
    ini:   #(b)
      - configMap:
          items:
            - auth.ini
          name: qmgr-monitoring-config
    mqsc:   #(c)
      - configMap:
          items:
            - channel.mqsc
          name: qmgr-monitoring-config
  pki:
    keys:   #(d)
        - name: default
          secret:
            items:
            - tls.key
            - tls.crt
            secretName: qm-tls-secret
    trust:   #(e)
      - name: app
        secret:
          items:
            - application.crt
          secretName: instana-certificate-secret
```

Die markierten Abschnitte der Spezifikation werden wie folgt beschrieben:

a. Stellen Sie sicher, dass Sie Ihrem zugrunde liegenden Warteschlangenmanager einen eindeutigen Namen gegeben haben. Wenn der zugrunde liegende Warteschlangenmanager keinen eindeutigen Namen hat, funktioniert die Überwachung möglicherweise nicht wie beabsichtigt. Dieser Name muss mit dem Namen in der Konfigurationszuordnung des IBM Instana -Agenten übereinstimmen, die zuvor bearbeitet wurde.

b. Die INI-Informationen, die in die ConfigMap geschrieben wurden, werden dem Warteschlangenmanager hinzugefügt.

c. Die MQSC-Informationen, die in die Konfigurationszuordnung geschrieben wurden, werden dem WS-Manager hinzugefügt.

d. Das Warteschlangenmanagerzertifikat und der private Schlüssel werden dem Warteschlangenmanager-Keystore hinzugefügt.

e. Das Zertifikat des IBM Instana -Agenten wird dem Truststore des Warteschlangenmanagers hinzugefügt.

6. Optional: Aktivieren Sie die IBM Instana -Traceerstellung auf Ihrem überwachten Warteschlangenmanager.

   Wenn Sie dies wünschen, lesen Sie den Abschnitt „IBM MQ mit IBM Instana -Traceerstellung integrieren” auf Seite 90.

7. Implementieren Sie den Warteschlangenmanager.

Nächste Schritte

Jetzt können Sie die IBM Instana -Überwachung überprüfen und debuggen.

OpenShift CP4I Operator2.2.0

Instana-Überwachung: Prüfen und Debugging

Um einen Warteschlangenmanager über einen IBM Instana -Agenten überwachen zu können, müssen Sie sowohl den Agenten als auch den Warteschlangenmanager konfigurieren.

Vorbereitende Schritte

Diese Task setzt voraus, dass Sie den Warteschlangenmanager für die IBM Instana -Überwachung konfiguriert haben.

Vorgehensweise

Verifizierung

1. Um zu überprüfen, ob Ihre Implementierung erfolgreich war, zeigen Sie Ihren Warteschlangenmanager im IBM Instana -Dashboard an.

   Der Warteschlangenmanager sollte im Servicebereich der Anwendungsseite und auch in der Infrastrukturansicht sichtbar sein.

Debugging

Anmerkung: Bei diesen Debugschritten wird eine OpenShift-Implementierung des IBM Instana -Agenten vorausgesetzt, der als DaemonSet ausgeführt wird.

Wenn Ihr Warteschlangenmanager nicht im IBM Instana -Dashboard angezeigt wird, ist Ihr Warteschlangenmanager möglicherweise fehlerhaft konfiguriert. Führen Sie die folgenden Schritte aus, um dies zu untersuchen.

2. Ermitteln Sie den Knoten, auf dem Ihr aktiver Warteschlangenmanager-Pod ausgeführt wird.

   Führen Sie den folgenden Befehl im Namensbereich Ihres Warteschlangenmanagers aus:

   ```
   oc get pods -o wide -n your-queue-manager-namespace
   ```

3. Um festzustellen, welcher IBM Instana -Agentenpod auf demselben Knoten wie Ihr Warteschlangenmanager ausgeführt wird, führen Sie denselben Befehl im Namensbereich 'instana-agent' aus:

```
oc get pods -o wide -n instana-agent-namespace
```

4. Um Probleme auf der Seite des IBM Instana -Agenten zu verstehen, rufen Sie die Protokolle des IBM Instana -Agentenpods ab und suchen Sie nach Einträgen, die sich auf 'mq' oder auf den Namen Ihres Warteschlangenmanagers beziehen.

   Führen Sie den folgenden Befehl aus:

   ```
   oc logs instana-agent-pod -c instana-agent -n instana-agent
   ```

5. Überprüfen Sie die Protokolle des Warteschlangenmanagers.

   Wenn der Agent versucht hat, eine Verbindung zum WS-Manager herzustellen, sollten die Warteschlangenmanagerprotokolle angeben, warum die Verbindung nicht erfolgreich war. Führen Sie den folgenden Befehl aus:

   ```
   oc logs your-queue-manager-name -n your-queue-manager-namespace
   ```

Ergebnisse

Sie haben alle vier Tasks zum Konfigurieren der authentifizierten IBM Instana Überwachung mit TLSausgeführt.

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 keinen geheimen Schlüssel Red Hat OpenShift für die IBM Entitled Registry in Ihrem Red Hat OpenShift -Projekt haben, führen Sie die Schritte unter Geheimen Schlüssel für Berechtigungsschlüssel erstellenaus.

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 `BuildConfig`ermöglicht 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:9.4.0.0-r1
      RUN printf "DEFINE QLOCAL(foo) REPLACE\n" > /etc/mqm/my.mqsc \
        && printf "Channels:\n\tMQIBindType=FASTPATH\n" > /etc/mqm/my.ini
      LABEL summary "My custom MQ image"
  strategy:
    type: Docker
    dockerStrategy:
      from:
        kind: "DockerImage"
        name: "cp.icr.io/cp/ibm-mqadvanced-server-integration:9.4.0.0-r1"
      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 5). 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 „Einfachen Warteschlangenmanager mit IBM MQ Operator implementieren" auf Seite 66 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

„Einfachen Warteschlangenmanager mit IBM MQ Operator implementieren” auf Seite 66
In diesem Beispiel wird ein Warteschlangenmanager für den Schnelleinstieg implementiert, der ephemeren (nicht persistenten) Speicher verwendet und die IBM MQ -Sicherheit inaktiviert. Nachrichten werden bei Neustarts des Warteschlangenmanagers nicht persistent gespeichert. Sie können die Konfiguration anpassen, um viele Warteschlangenmanagereinstellungen zu ändern.

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. For example:

  ```
  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. For example:

  ```
  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.

Prozedur

- Inaktivieren Sie Laufzeit-Webhook-Prüfungen.

  Fügen Sie unter `metadata` folgende Anmerkung hinzu. For example:

```
apiVersion: mq.ibm.com/v1beta1
kind: QueueManager
metadata:
  name: quickstart-cp4i
  annotations:
    "com.ibm.cp4i/disable-webhook-runtime-checks" : "true"
```

OpenShift CP4I Operator2.1.0 Inaktivieren von Standardwertaktualisierungen für die Warteschlangenmanagerspezifikation

IBM MQ Operator aktualisiert alle nicht angegebenen Werte in der Warteschlangenmanagerspezifikation mit ihren Standardwerten. Sie können dieses Verhalten inaktivieren, wenn Sie Änderungen an der Warteschlangenmanagerspezifikation vermeiden möchten. Die Statusfelder des Warteschlangenmanagers werden weiterhin aktualisiert.

Prozedur

- Inaktivieren Sie die Warteschlangenmanager-Standardwertaktualisierungen.

  Fügen Sie unter `metadata` folgende Anmerkung hinzu. For example:

```
apiVersion: mq.ibm.com/v1beta1
kind: QueueManager
metadata:
  name: quickstart-cp4i
  annotations:
    "com.ibm.mq/write-defaults-spec" : "false"
```

  Anmerkung: In den Quickstart-Beispielen wird diese Annotation standardmäßig angewendet.

IBM MQ -Container mit einem schreibgeschützten Stammdateisystem ausführen

Sie können den IBM MQ -Container so konfigurieren, dass er mit einem schreibgeschützten Stammdateisystem ausgeführt wird. Dies verhindert, dass Angreifer zerstörerischen Programmcode im Container kopieren und ausführen.

Informationen zu diesem Vorgang

Die Aktivierung des schreibgeschützten Stammdateisystems macht die Containerdateien unveränderlich. Das heißt, im Containerdateisystem können Dateien angezeigt, aber nicht geändert werden, und es können keine neuen Dateien erstellt werden. Dateien können nur in einem angehängten Dateisystem geändert bzw. erstellt werden.

Wenn ein schreibgeschütztes Stammdateisystem aktiviert ist, werden zwei ephemere Datenträger `Scratch` und `Tmp` erstellt und in den Verzeichnissen `/run` und `/tmp` im Container angehängt.

- Der `Arbeitsdatenträger` enthält die Dateien, Keystores und anderen Dateien, die für die Konfiguration des Warteschlangenmanagers verwendet wurden.
- Der Datenträger `Tmp` enthält Diagnosedateien, z. B. die RAS-Dateien des Warteschlangenmanagers.

Da diese Datenträger ephemere Datenträger sind, gehen die Dateien auf diesen Datenträgern beim Neustart des Pods verloren.

Der Typ des für WS-Manager-Daten erstellten Datenträgers hängt vom Speichertyp ab. Standardmäßig wird ein persistenter Datenträger angehängt. Wenn der Speichertyp `ephemeral`lautet, wird ein ephemerer Datenträger angehängt. Wenn die Größe der Daten auf dem Datenträger den für die Eigenschaft **`sizeLimit`** angegebenen Wert überschreitet, kann Kubernetes den Container ausgeben und einen neuen erstellen.

Ein schreibgeschütztes Stammdateisystem ist standardmäßig nicht aktiviert. Führen Sie die folgenden Schritte aus, um sie zu aktivieren:

Vorgehensweise

1. Verwenden Sie die API `spec.securityContext` , um das schreibgeschützte Stammdateisystem zu aktivieren.

   Setzen Sie für Ihren Warteschlangenmanager die Eigenschaft **`readOnlyRootFilesystem`** in „.spec.securityContext” auf Seite 158 auf `true`.

   IBM MQ Operator erstellt zwei ephemere Datenträger: `Scratch` und `Tmp`.
2. Optional: Legen Sie den Datenspeichertyp des Warteschlangenmanagers fest oder ändern Sie ihn.

   Standardmäßig wird ein Persistent Volume Claim (PVC) an `/mnt/mqm`angehängt. Wenn die Eigenschaft **`type`** in „.spec.queueManager.storage.queueManager” auf Seite 157auf `ephemeral` gesetzt ist, wird ein ephemerer Datenträger erstellt und angehängt.
3. Berücksichtigen Sie für jeden ephemeren Datenträger sorgfältig, wie groß die Datenmenge werden könnte. Legen Sie den Wert der Eigenschaft **`sizeLimit`** entsprechend fest, einschließlich SI-Einheiten.
   - Legen Sie für den ephemeren `Arbeitsdatenträger` die Eigenschaft **`sizeLimit`** in „.spec.queueManager.storage.scratch” auf Seite 158fest. Der Standardwert ist "100M".
   - Legen Sie für den ephemeren `Tmp` -Datenträger die Eigenschaft **`sizeLimit`** in „.spec.queueManager.storage.tmp” auf Seite 158fest. Der Standardwert ist "2Gi".
   - Wenn **`type`** des Warteschlangenmanagerdatenträgers auf `ephemeral`gesetzt ist, legen Sie die Eigenschaft **`sizeLimit`** in „.spec.queueManager.storage.queueManager” auf Seite 157fest. Der Standardwert ist "2Gi".

OpenShift V 9.4.0 IBM MQ Console mit einer Basisregistry mit IBM MQ Operator konfigurieren

Für die Anmeldung bei IBM MQ Consolekönnen Sie dem Warteschlangenmanager Ihre eigene Konfiguration bereitstellen.

Vorbereitende Schritte

Wenn Sie einen Warteschlangenmanager mit einer IBM MQ Advanced for Developers -Lizenz implementieren, ist eine einfache Konfiguration integriert. Siehe „Beispiel-YAML-Datei für Warteschlangenmanager, die beschreibt, wie Kennwörter für admin -und app -Benutzer angegeben werden” auf Seite 177. Wenn Sie einen IBM Cloud Pak for Integration -Lizenzwarteschlangenmanager implementieren, können Sie die Integration mit IBM Cloud Pak for Integration Keycloak aktivieren, um sich über Single Sign-on bei IBM MQ Console anzumelden. Weitere Informationen finden Sie in „Verbindung zur IBM MQ Console herstellen, die in einem Red Hat OpenShift-Cluster implementiert ist” auf Seite 134.

Vorgehensweise

1. **Erstellen Sie ein Kennwort und verschlüsseln Sie es mit `securityUtility`.**

Ein ConfigMap wird zum Speichern der Berechtigungsnachweise verwendet, die Sie für den Zugriff auf Ihren Warteschlangenmanager verwenden. Zur Verbesserung der Sicherheit codieren Sie diese Berechtigungsnachweise mit dem BefehlsecurityUtility.

Alternativ können Sie einen geheimen Schlüssel verwenden, der Berechtigungsnachweise in der Kubernetes -Ebene schützt. Überwachungs-oder Fehlerbehebungstools können jedoch die zugrunde liegende Datei unsicher machen.

2. Optional: **Melden Sie sich bei der Red Hat OpenShift -Befehlszeilenschnittstelle an.**

   Wenn Sie die OpenShift -Befehlszeilenschnittstelle verwenden, melden Sie sich mit `oc login`an.

   Alternativ können Sie die OpenShift -Konsole verwenden.

3. **Erstellen Sie eine ConfigMap mit Ihrer Konfiguration.**

   Hilfe zum Erstellen der XML-Konfiguration finden Sie unter IBM MQ Console und REST API Sicherheit.

   Das folgende Beispiel erstellt einen Benutzer in der Gruppe MQWebAdminGroup. Mitgliedern der MQWebAdminGroup wird die Rolle MQWebAdmin zugewiesen. In diesem Beispiel gilt Folgendes:

   - Sie **müssen** *USERNAME* und *PASSWORD* durch Ihre eigenen Werte ersetzen. Beachten Sie, dass *USERNAME* im Beispiel zweimal verwendet wird.

     Sie **müssen** den *NAMESPACE* als den Namensbereich angeben, in dem Ihr IBM MQ Operator bereitgestellt wird und in dem Ihr Warteschlangenmanager bereitgestellt wird.

   a) Verwenden Sie die OpenShift -Konsole oder die Befehlszeile, um die folgende ConfigMapzu erstellen:

   ```
   kind: ConfigMap
   apiVersion: v1
   metadata:
     name: mqwebuserconfigmap
     namespace: NAMESPACE
   data:
     mqwebuser.xml: |
       <?xml version="1.0" encoding="UTF-8"?>
       <server>
         <featureManager>
           <feature>appSecurity-2.0</feature>
           <feature>basicAuthenticationMQ-1.0</feature>
         </featureManager>
         <enterpriseApplication id="com.ibm.mq.console">
           <application-bnd>
             <security-role name="MQWebAdmin">
               <group name="MQWebAdminGroup" realm="defaultRealm"/>
             </security-role>
           </application-bnd>
         </enterpriseApplication>
         <basicRegistry id="basic" realm="defaultRealm">
           <user name="USERNAME" password="PASSWORD"/>
           <group name="MQWebAdminGroup">
             <member name="USERNAME"/>
           </group>
         </basicRegistry>
         <sslDefault sslRef="mqDefaultSSLConfig"/>
       </server>
   ```

   b) Optional: Wenn Sie die Befehlszeile verwenden, wenden Sie ConfigMapan:

   ```
   oc apply -f mqwebuserconfigmap.yaml
   ```

   Wählen Sie für die übrigen Schritte eine der folgenden Optionen aus:

   - Implementieren Sie einen neuen Warteschlangenmanager mit der Konfiguration für den Zugriff auf IBM MQ Console.
   - Wenden Sie die Konfiguration an, die IBM MQ Console den Zugriff auf einen vorhandenen Warteschlangenmanager ermöglicht.

4. Optional: **Implementieren Sie einen neuen Warteschlangenmanager mit der Konfiguration, um auf IBM MQ Consolezuzugreifen.**

   a) Erstellen Sie einen Warteschlangenmanager.

Setzen Sie den Authentifizierungs-und Berechtigungsprovider auf `manual` und geben Sie den neu erstellten `ConfigMap` `mqwebuserconfigmap` über eine der folgenden Optionen an:

- Option 1: Über die YAML-Datei des Warteschlangenmanagers

  Fügen Sie den folgenden Code im Abschnitt `web` der YAML-Datei des Warteschlangenmanagers hinzu:

```
...
web:
    enabled: true
    console:
      authentication:
        provider: manual
      authorization:
        provider: manual
    manualConfig:
      configMap:
        name: mqwebuserconfigmap
```

- Option 2: Über die Formularansicht der OpenShift -Konsole:

  i) Wählen Sie in der OpenShift -Konsole **Operatoren** > **Installierte Operatoren**aus.

  ii) Wählen Sie Ihre Implementierung von IBM MQ Operatoraus.

  iii) Wählen Sie **Warteschlangenmanager** aus und klicken Sie auf **QueueManager**.

  iv) Wählen Sie die relevanten Optionen für Ihren Warteschlangenmanager aus.

  v) Wählen Sie **Web** aus und setzen Sie **Web-Server aktivieren** auf `true`.

  vi) Öffnen Sie das Listenfenster **Erweiterte Konfiguration** .

  vii) Setzen Sie im Listenfeld **Konsole** den **Provider** für **Authentifizierung** und **Berechtigung** auf `manuell`.

  viii) Öffnen Sie das Listenfeld **Konfiguration** .

  ix) Öffnen Sie das Listenfeld **ConfigMap** und wählen Sie die in Schritt „3” auf Seite 103erstellte ConfigMap `mqwebuserconfigmap` aus.

  x) Klicken Sie auf **Erstellen**.

  Sie können jetzt über die Berechtigungsnachweise, die in dem in Schritt „3” auf Seite 103erstellten `ConfigMap` angegeben sind, auf die IBM MQ Console Ihres neuen Warteschlangenmanagers zugreifen.

5. Optional: **Apply-Konfiguration, die den IBM MQ Console für einen vorhandenen Warteschlangenmanager aktiviert.**

   Bearbeiten Sie die YAML-Datei des Warteschlangenmanagers, für den Sie die IBM MQ Consoleaktivieren:

   a. Wählen Sie in der OpenShift -Konsole **Operatoren** > **Installierte Operatoren**aus.

   b. Wählen Sie Ihre Implementierung von IBM MQ Operatoraus.

   c. Wählen Sie **Queue Mananger** und dann den Namen Ihres Warteschlangenmanagers aus.

   d. Wählen Sie **YAML**aus.

   e. Ersetzen Sie den vorhandenen Abschnitt `web` der YAML-Datei des Warteschlangenmanagers durch den folgenden Code:

```
...
web:
    enabled: true
    console:
      authentication:
        provider: manual
      authorization:
        provider: manual
    manualConfig:
      configMap:
        name: mqwebuserconfigmap
```

f. Klicken Sie auf **Speichern**.

Sie können jetzt über die Berechtigungsnachweise, die in dem in Schritt „3” auf Seite 103erstellten `ConfigMap` angegeben wurden, auf die IBM MQ Console Ihres vorhandenen Warteschlangenmanagers zugreifen.

OpenShift V 9.4.0 V 9.4.0 Persistente Datenträger erweitern

Wenn Ihr Speicheranbieter die Datenträgererweiterung unterstützt, können Sie mit dieser Task einen persistenten Datenträger erweitern. Je nach Speicheranbieter kann die Erweiterung online oder offline erfolgen.

Vorbereitende Schritte

Die erfolgreiche Datenträgererweiterung hängt von Ihrem Speicheranbieter ab, um die Erweiterungsanforderung zu erfüllen. Ziehen Sie die Dokumentation zu Ihren Speicheranbietern zu Rate, um festzustellen, ob die Onlinegrößenänderung unterstützt wird, und um Informationen zu Verfahren zur Offlinegrößenänderung zu erhalten.

Wenn Ihr Speicheranbieter die Erweiterungsanforderung nicht erfüllen kann, wird Ihr Persistent Volume Claim möglicherweise in einen Status mit Warnungen oder Fehlern versetzt. Wenn die Erweiterung fehlschlägt, kann ein OpenShift -Administrator den Persistent Volume Claim-Status manuell wiederherstellen und die Erweiterung abbrechen. Siehe Fehlerbehebung beim Erweitern von Datenträgern in der Red Hat OpenShift -Dokumentation.

Informationen zu diesem Vorgang

Zur Unterstützung bei der Verwaltung des persistenten Speichers definiert Kubernetes zwei API-Ressourcen:

- Ein PersistentVolume (PV), bei dem es sich um einen Teil des Speichers im Cluster handelt, der von einem Administrator oder mithilfe von Speicherklassen dynamisch bereitgestellt wurde. Sie kann statisch oder dynamisch bereitgestellt werden.
- Ein PersistentVolume-Claim (PVC), bei dem es sich um eine Speicheranforderung eines Benutzers handelt. Sie fungiert auch als Anspruchsüberprüfung für die Ressource.

Weitere Informationen finden Sie unter Persistent Volumes in der Kubernetes -Dokumentation.

Warnung:

- Wenn die Speicherklasse, die zum Erstellen von PVCs des Warteschlangenmanagers verwendet wird, keine Onlinegrößenänderung unterstützt, findet eine Offlinegrößenänderung statt. Während der Offlinegrößenänderung ist ein Benutzereingriff erforderlich, um die Datenträgererweiterung abzuschließen, sodass bei Warteschlangenmanagern Ausfallzeiten auftreten.
- Für die Offlinegrößenänderung gemeinsam genutzter Datenträger für Multi-Instanz-Warteschlangenmanagermüssen sowohl aktive als auch Standby-Pods gleichzeitig heruntergefahren werden, wenn der Benutzereingriff ausgeführt wird.
- OpenShift unterstützt keine Reduzierung der Größe von PVCs. Der Versuch, die Größe der persistenten Datenträger zu reduzieren, versetzt den Warteschlangenmanager in den Status 'Fehlgeschlagen'.
- Diese Prozedur gilt nicht für ephemere Datenträger.

Führen Sie die folgenden Schritte aus, um einen PV zu erweitern, der vom IBM MQ -Container verwendet wird.

Vorgehensweise

1. **Erweitern von Datenträgern vorbereiten**
   a) Entscheiden Sie, welche Datenträger erweitert werden sollen.

b) Bestimmen Sie die Speicherklassen, die von Ihren Datenträgern verwendet werden.

For example:

```
spec:
  queueManager:
    storage:
      persistedData:
        enabled: true
        type: persistent-claim
        class: ocs-storagecluster-cephfs (1)
      queueManager:
        type: persistent-claim
      recoveryLogs:
        enabled: true
        type: persistent-claim
      defaultClass: ocs-storagecluster-ceph-rbd (2)
```

Anmerkungen:

- (1) Wenn der Datenträger eine bestimmte Speicherklasse definiert, wird diese von PVCs dieses Typs verwendet.
- (2) Wenn **defaultClass** festgelegt ist, wird diese Speicherklasse für alle Datenträger ohne eine bestimmte Speicherklasse verwendet. Wenn **defaultClass** nicht festgelegt ist und ein Datenträgertyp keine Klasse angegeben hat, dann wird die Standardspeicherklasse für den Cluster verwendet.

Sie können auch die verwendete Speicherklasse bestätigen, indem Sie die zugrunde liegenden PVCs beschreiben. For example:

```
oc describe pvc pvc-name
```

c) Überprüfen Sie, ob Ihre Speicherklasse die Datenträgererweiterung unterstützt.

Für eine Speicherklasse kann die Eigenschaft **.allowVolumeExpansion** definiert sein:

- Wenn diese Eigenschaft auf `true`gesetzt ist, wird die Datenträgererweiterung unterstützt.
- Wenn diese Eigenschaft auf `false`gesetzt oder nicht definiert ist, lässt die Speicherklasse keine Datenträgererweiterung zu. Ziehen Sie in diesem Fall die Dokumentation zu Ihrem Speicheranbieter zu Rate, um festzustellen, ob diese Funktion aktiviert werden kann.

Sie können eine Speicherklasse auch beschreiben, um festzustellen, ob sie die Datenträgererweiterung unterstützt. For example:

```
oc describe sc storage-class-name
```

d) Lesen Sie in der Dokumentation Ihres Speicheranbieters nach, ob eine Online-oder Offlineprozedur für die Datenträgererweiterung verwendet wird.

Eine Offlineprozedur erfordert, dass Warteschlangenmanager-Pods manuell erneut gestartet werden, eine Onlineprozedur hingegen nicht. Informationen zum Ändern der Offlinegröße finden Sie in der Dokumentation zu Ihrem Speicheranbieter.

e) Überprüfen Sie, ob Ihr Warteschlangenmanager eine Statusbedingung mit der Ursache 'Storage-Mismatch' aufweist.

Wenn Ihr Warteschlangenmanager diese Statusbedingung aufweist, werden die in der Bedingung aufgelisteten Datenträger erweitert, wenn Sie die Datenträgererweiterung aktivieren. Wenn Sie dies nicht wünschen, ändern Sie die Größenfelder, die den einzelnen Datenträgertypen in Ihrer Warteschlangenmanagerdefinition zugeordnet sind, so, dass sie den bereitgestellten PVCs entsprechen. Die Statusbedingung wird entfernt, wenn dies für alle abweichenden Datenträger erfolgt.

2. **Datenträger erweitern**

Warnung:

- Wenn Sie zuvor Felder für die Datenträgergröße in Ihrer Warteschlangenmanagerdefinition geändert haben, beginnen die Datenträger mit der Erweiterung, wenn **.allowVolumeExpansion** in Ihrer Warteschlangenmanagerdefinition auf `true` gesetzt ist.

- Ihr Speicheranbieter kann aufgrund von Dateisystemeinschränkungen oder der Verfügbarkeit lokaler Hardware Einschränkungen für die maximale Größe eines Datenträgers haben. Um Fehler zu vermeiden, überprüfen Sie diese Einschränkungen in der Dokumentation zu Ihrem Speicheranbieter, bevor Sie Datenträger erweitern.
- Eine Reduzierung der PVC-Größe wird von OpenShiftnicht unterstützt. Wenn Sie die Größe eines Datenträgers erweitern, können Sie ihn nicht reduzieren. Wenn Ihr Versuch fehlschlägt, kann der IBM MQ Operator den PVC nicht in seinen ursprünglichen Zustand zurückversetzen.

Beispiel für eine Warteschlangenmanagerdefinition zur Datenträgererweiterung:

```
spec:
  queueManager:
    storage:
      allowVolumeExpansion: true (A)
      persistedData:
        enabled: true
        type: persistent-claim
        size: 3Gi (B)
      queueManager:
        type: persistent-claim
        size: 4Gi (B)
      recoveryLogs:
        enabled: true
        type: persistent-claim
        size: 3Gi (B)
```

a) Um die Datenträgererweiterung für den Warteschlangenmanager zuzulassen, setzen Sie das Feld **`.spec.queueManager.storage.allowVolumeExpansion`** (A) auf Ihrem Warteschlangenmanager auf `true`.

b) Sie können jetzt die Größe der Felder (B) für alle aktivierten Datenträgertypen erhöhen. Durch Anwenden dieser Änderungen wird die Datenträgererweiterung gestartet.

3. **Überprüfen Sie, ob die Größe Ihrer PVCs geändert wurde.**

Anmerkungen:

- Die Datenträgererweiterung kann einige Zeit dauern. Wenn die Validierung nicht erfolgreich ist, sollten Sie einige Minuten warten und die Validierung erneut durchführen.
- Die Datenträgererweiterung wird nur ohne Benutzeraktion ausgeführt, wenn eine Onlinegrößenänderung ausgeführt wird.
- Einige Speicheranbieter runden die angeforderte Speichergröße auf. Der erweiterte Datenträger sollte mindestens dieselbe Größe wie Ihre Anforderung haben.

a) Überprüfen Sie Ihren Warteschlangenmanager auf Statusbedingungen. In der folgenden Tabelle finden Sie Bedingungen, Erläuterungen und vorgeschlagene Aktionen.

Tabelle 1. Statusbedingungen für die Speicherung

CONDITION	**NACHRICHT**	**Erläuterung**
`StorageMismatch`	`Storage sizes defined in the QueueManager resource do not match the capacity of one or more provisioned PVCs [pvc-list]. AllowVolumeExpansion is set to false in the QueueManager resource so the MQ Operator will not attempt to reconcile these differences.`	Die Datenträgererweiterung findet nicht statt, da **`.allowVolumeExpansion`** in der Warteschlangenmanagerdefinition nicht auf `true` gesetzt wurde.

Tabelle 1. Statusbedingungen für die Speicherung (Forts.)

CONDITION	NACHRICHT	Erläuterung
`StorageExpansionPending`	`Volume expansion is pending for the following PVCs [pvc-list]`	Die Datenträgererweiterung findet noch statt. Wenn diese Statusbedingung über einen längeren Zeitraum bestehen bleibt, führen Sie die folgenden Schritte aus, um weitere Informationen zu sammeln, da möglicherweise eine Offlinegrößenänderung oder eine Größenänderung stattfindet.
`Failed`	Es gibt viele mögliche speicherbezogene Nachrichten, die eine `'Failed'` -Statusbedingung erstellen können. Beispiel: `'MQ Queue Manager failed to deploy: persistentvolumeclaims "<pvc>" is forbidden: only dynamically provisioned pvc can be resized and the storageclass the provisions the pvc must support resize.'`	Wenn der Warteschlangenmanager `'Failed'` -Statusbedingungen mit Text hat, der sich auf Speicher bezieht, lesen Sie die Nachricht innerhalb der Statusbedingung. Die hier angegebene Beispielnachricht wird durch die Verwendung einer Speicherklasse verursacht, die keine Erweiterung unterstützt.

b) Überprüfen Sie für jedes PVC, das Sie erweitert haben, ob die Kapazität so erhöht wurde, dass sie dem in der Warteschlangenmanagerdefinition angegebenen Wert entspricht oder größer ist.

HA-Warteschlangenmanager können mehrere PVCs jedes Typs haben. Führen Sie den folgenden Befehl aus, um die Kapazität eines PVC abzurufen:

```
oc get pvc pvc-name -o template --template '{{.status.capacity.storage}}'
```

c) Überprüfen Sie, ob der PVC keine Statusbedingungen oder Ereignisse aufweist, die eine fehlgeschlagene Größenänderung vorschlagen:

```
oc describe pvc pvc-name
```

- Ihr PVC verfügt möglicherweise über die Statusbedingung `FileSystemResizePending` mit der Nachricht 'Waiting for user to (re-) start a pod to finish file system resize of volume on node'. Diese Statusbedingung tritt bei Größenänderungen im Online-und Offlinestatus auf. Bei einer Onlinegrößenänderung verschwindet diese Statusbedingung ohne Benutzeraktion, nachdem die Onlinegrößenänderung abgeschlossen ist.
- Wenn Ihr PVC über eine Ereignis-oder Statusbedingung verfügt, die auf eine fehlgeschlagene Größenänderung hinweist, lesen Sie den Abschnitt Wiederherstellung nach einem Fehler beim Erweitern von Datenträgern in der Red Hat OpenShift -Dokumentation.

d) Stellen Sie sicher, dass die Warteschlangenmanager-Pods keine Statusbedingungen oder Ereignisse haben, die eine fehlgeschlagene Größenänderung vorschlagen. Überprüfen Sie bei Hochverfügbarkeitsimplementierungen jedes Replikat.

```
oc describe pod queue-manager-pod-name
```

- Wenn Ihr Pod über eine Ereignis-oder Statusbedingung verfügt, die auf eine fehlgeschlagene Größenänderung hinweist, finden Sie weitere Informationen in Wiederherstellung nach Fehlern beim Erweitern von Datenträgern in der Red Hat OpenShift -Dokumentation. Der Fehlertext kann

Ihnen helfen, das Problem zu lösen, oder verhindern, dass dasselbe Problem auftritt, wenn Sie versuchen, die Größe nach der Wiederherstellung erneut zu ändern.

4. **Pods erneut starten, wenn Größe offline geändert wird**

   Wenn Ihr Speicheranbieter beim Erweitern von Datenträgern eine Prozedur zur Offlinegrößenänderung verwendet, müssen Sie nach Abschluss der Datenträgererweiterung die Warteschlangenmanager-Pods erneut starten, die die Datenträger anhängen, deren Größe geändert wird.

   Bei Warteschlangenmanagern mit mehreren Instanzen werden die Wiederherstellungsprotokolle und persistenten Datenträger für Daten von den aktiven Pods und den Standby-Pods gemeinsam genutzt. Um die Größe dieser Datenträger zu ändern, müssen Sie beide Pods gleichzeitig herunterfahren.

   Informationen zur Vorgehensweise beim Ändern der Offlinegröße finden Sie in der Dokumentation Ihres Speicheranbieters.

OpenShift V 9.4.0 V 9.4.0 Warteschlangenmanager stoppen (`mq.ibm.com/stop`)

Stoppen Sie einen Warteschlangenmanager, indem Sie der Warteschlangenmanagerdefinition eine Anmerkung hinzufügen.

Informationen zu diesem Vorgang

Warteschlangenmanagern, die vom IBM MQ -Operator erstellt wurden, ist `StatefulSet`zugeordnet. Diese `StatefulSet` deklariert die Anzahl der `Pods` , die für einen bestimmten Warteschlangenmanagerverfügbarkeitstyp implementiert werden sollen, über das Feld '`.replicas`'. Gültige Werte sind 1 (Einzelinstanz), 2 (Mehrfachinstanz) oder 3 (NativeHA).

Anmerkung: Durch manuelle Änderung des Werts im Feld '`.replicas`' wird verhindert, dass der Warteschlangenmanager ordnungsgemäß funktioniert.

In einigen Fällen können Sie Ihren Warteschlangenmanager stoppen, damit der `StatefulSet` eine Replikatanzahl von 0 hat und keine `Pods` implementiert werden. Beispiele für den Fall, dass Sie dies tun möchten, sind beispielsweise während der Wartung oder einer Sicherungsprozedur.

Anmerkung: Da kein Warteschlangenmanager `Pods` implementiert ist, wenn der Warteschlangenmanager gestoppt wird, können Sie und Ihre Anwendungen erst wieder auf den Warteschlangenmanager zugreifen, wenn er erneut gestartet wird.

Prozedur

- Um Ihren Warteschlangenmanager zu stoppen, fügen Sie die folgende Anmerkung zur Warteschlangenmanagerdefinition im Abschnitt '`.metadata.annotations`' hinzu.

```
apiVersion: mq.ibm.com/v1beta1
kind: QueueManager
metadata:
  name: my-qm
  annotations:
    "mq.ibm.com/stop" : "true"
```

- Um den Warteschlangenmanager erneut zu starten und auf die korrekte Anzahl von Replikaten zurückzusetzen, entfernen Sie die Anmerkung aus dem Warteschlangenmanager oder setzen Sie ihren Wert auf '`false`'.

Warteschlangenmanager mit Helm implementieren und konfigurieren

Sie können einen Warteschlangenmanager unter Kubernetes mithilfe des Beispieldiagramms Helm bereitstellen und konfigurieren.

Informationen zu diesem Vorgang

Wenn Sie Red Hat OpenShift Container Platformnicht verwenden, wird IBM MQ Operator nicht unterstützt. Sie können das Helm -Beispieldiagramm für die Bereitstellung in anderen Typen von Kubernetes -Clustern verwenden.

Prozedur

- Informationen zur Verwendung von Helm für die Bereitstellung Ihres eigenen IBM MQ -Container-Image finden Sie im Beispieldiagramm IBM MQ Helm .

Zugehörige Verweise

„Unterstützung für IBM MQ in Containern” auf Seite 8
Nicht alle IBM MQ -Funktionen sind in Containern auf dieselbe Weise verfügbar und werden unterstützt.

OpenShift CD CP4I-SC2 Migration auf IBM MQ Operator

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 Red Hat OpenShift Container Platform 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 111
2. „Warteschlangenmanagerkonfiguration extrahieren” auf Seite 111
3. Optional: „Optional: Warteschlangenmanagerschlüssel und -zertifikate extrahieren und übernehmen” auf Seite 112
4. Optional: „Optional: LDAP konfigurieren” auf Seite 114
5. Optional: „Optional: Ändern der IP-Adressen und Hostnamen in der IBM MQ-Konfiguration” auf Seite 122
6. „Warteschlangenmanagerkonfiguration für eine Containerumgebung aktualisieren” auf Seite 123
7. „Ziel-HA-Architektur für IBM MQ in Containern auswählen” auf Seite 127
8. „Ressourcen für den Warteschlangenmanager erstellen” auf Seite 127
9. „Neuen Warteschlangenmanager unter Red Hat OpenShift erstellen” auf Seite 129
10. „Neue Containerimplementierung überprüfen” auf Seite 133

OpenShift CD CP4I-SC2 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 auf IBM MQ Operator” auf Seite 110.

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 8.
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 98.
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 planen” auf Seite 18.

Nächste Schritte

Sie können jetzt die Warteschlangenmanagerkonfiguration extrahieren.

OpenShift CD CP4I-SC2 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 CD CP4I-SC2 Optional: Warteschlangenmanagerschlüssel und -zertifikate extrahieren und übernehmen

IBM MQ kann so konfiguriert werden, dass der Netzverkehr im Warteschlangenmanager mit TLS verschlüsselt wird. Verwenden Sie diese Task, um zu überprüfen, ob Ihr Warteschlangenmanager TLS verwendet, Schlüssel und Zertifikate zu extrahieren und 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 wird mit einem Schlüsselrepository ausgeführt, das auf dem Warteschlangenmanager konfiguriert ist. Die TLS-Kommunikation wird dann über IBM MQ-Kanäle ermöglicht. Wenn Sie nicht sicher sind, ob die TLS-Kommunikation in Ihrer Umgebung konfiguriert ist, führen Sie den folgenden Befehl aus, um dies zu überprüfen:

```
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.

Anmerkung: Warteschlangenmanagerzertifikate mit demselben registrierten Namen (Distinguished Name, DN) wie das Ausstellerzertifikat (CA) werden nicht unterstützt. Ein Zertifikat muss einen eindeutigen registrierten Namen haben. Das Produkt prüft, ob die DNs nicht identisch sind.

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 runmqakmResult < <(runmqakm -cert -list -db ${keyrlocation}.kdb -stashed)
        cert=1
        for i in "${runmqakmResult[@]:2}"
        do
           certlabel=$(echo ${i:2} | xargs)
           echo Extracting certificate $certlabel to $cert.cert
           runmqakm -cert -extract -db ${keyrlocation}.kdb -label "$certlabel" -target $
     {cert}.cert -stashed
           cert=$[$cert+1]
        done
     fi
     ```

     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-Schlüsselrepositorys für den Warteschlangenmanager:

        ```
        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:

```
runmqakm -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:

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

In den zuvor gezeigten Beispielen entspricht dies den folgenden Befehlen:

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

2. 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 CD CP4I-SC2 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 CD CP4I-SC2 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 114.

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. For example:

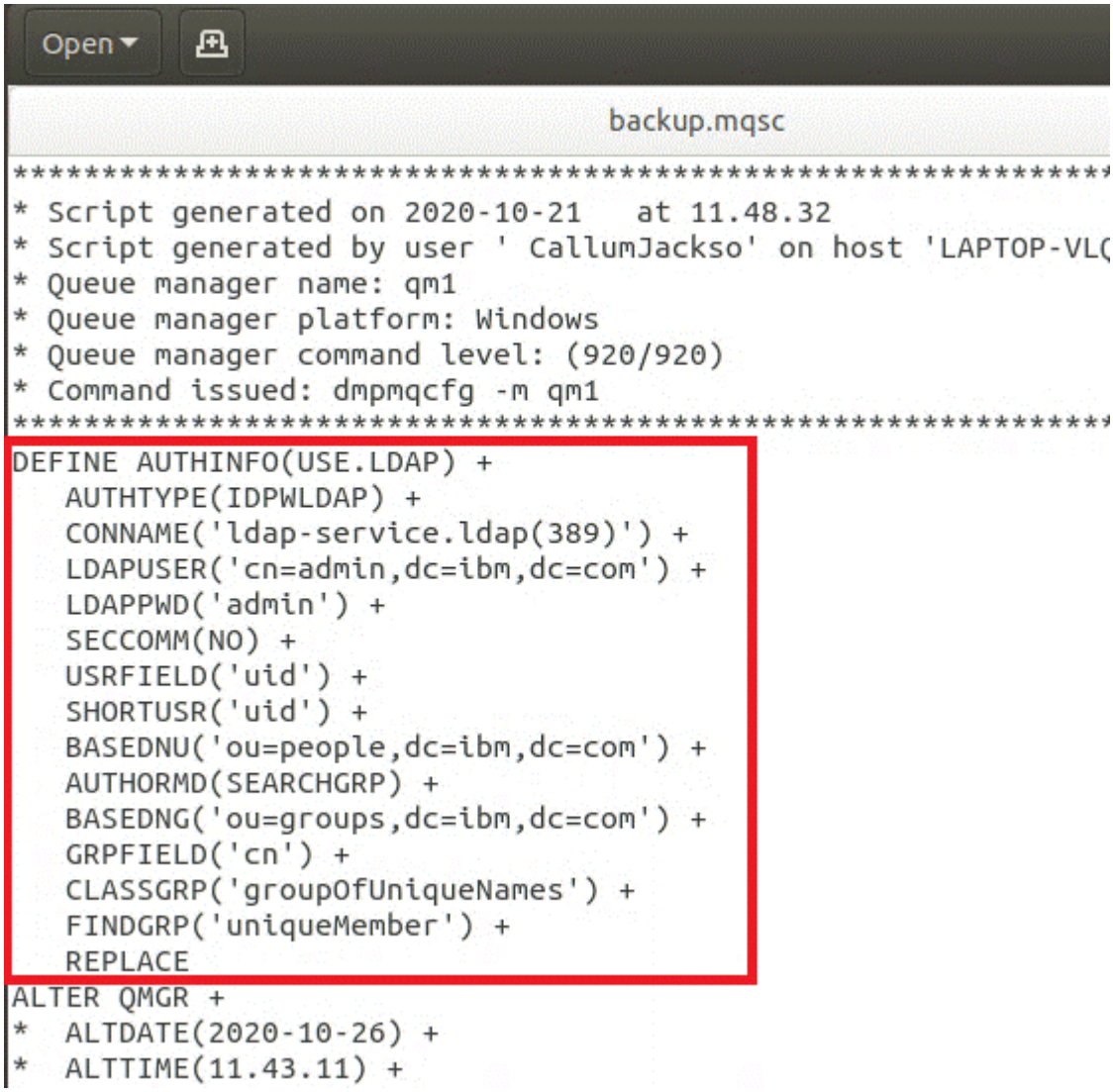
   ```
   DEFINE AUTHINFO(USE.LDAP) +
      AUTHTYPE(IDPWLDAP) +
      CONNAME('ldap-service.ldap(389)') +
      LDAPUSER('cn=admin,dc=ibm,dc=com') +
      LDAPPWD('admin') +
      SECCOMM(NO) +
      USRFIELD('uid') +
      SHORTUSR('uid') +
      BASEDNU('ou=people,dc=ibm,dc=com') +
      AUTHORMD(SEARCHGRP) +
      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 CD CP4I-SC2 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 114.

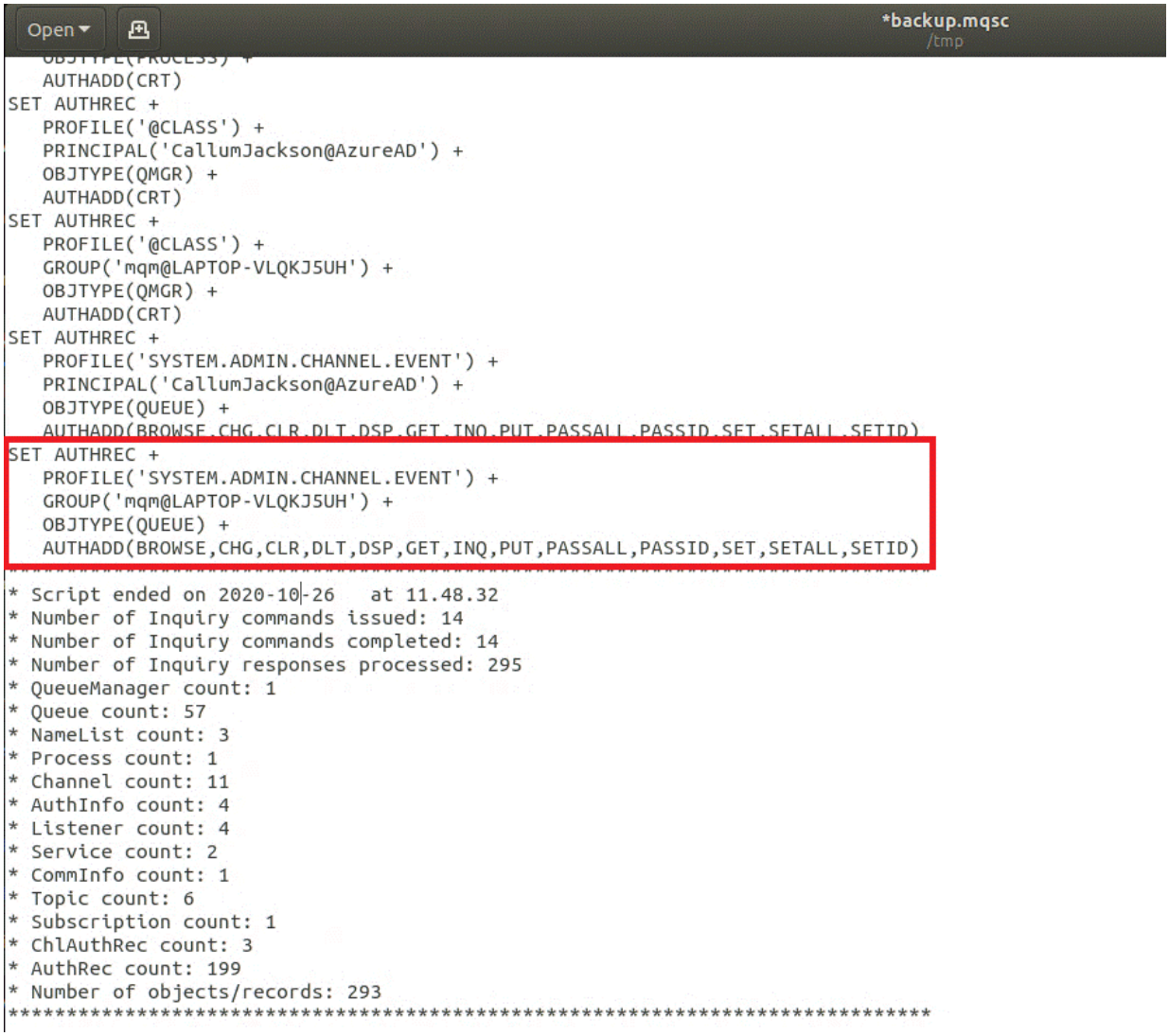
Die Aktualisierung der LDAP-Berechtigungsinformationen besteht aus zwei Teilen:

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 CD CP4I-SC2 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. For example:

   ```
   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 CD CP4I-SC2 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.

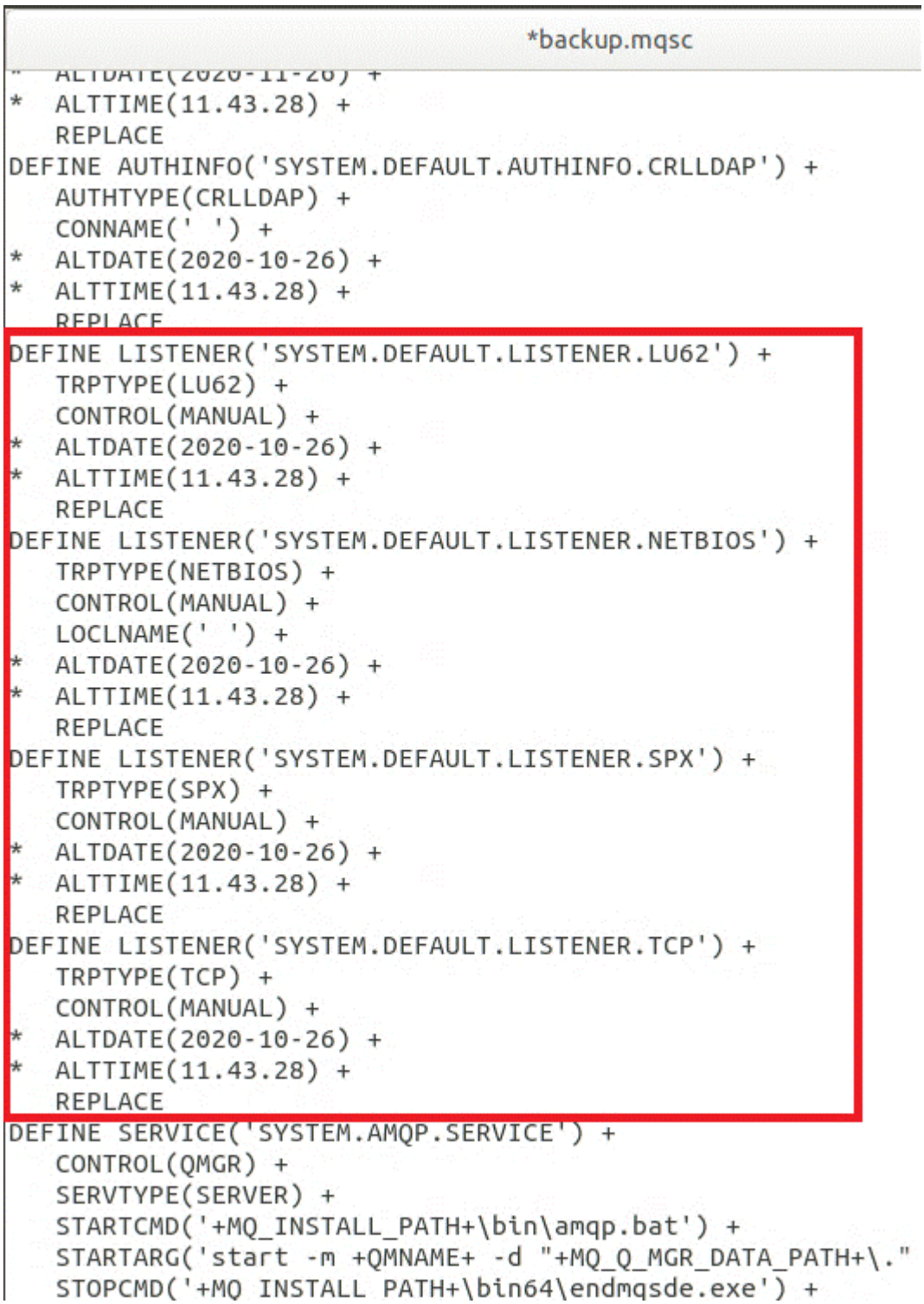
Daher müssen Sie die exportierte Konfiguration aktualisieren:

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 CD CP4I-SC2 Ziel-HA-Architektur für IBM MQ in Containern auswählen

Wählen Sie zwischen einer einzelnen Instanz (einem einzelnen Kubernetes -Pod), mehreren Instanzen (zwei Pods) und nativer Hochverfügbarkeit (ein aktiver Replikations-Pod und zwei Standby-Replikations-Pods), um Ihre Hochverfügbarkeitsanforderungen zu erfüllen.

Vorbereitende Schritte

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

Informationen zu diesem Vorgang

IBM MQ Operator bietet drei Hochverfügbarkeitsoptionen:

- **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.
- **Native HA**: Drei Container (jeweils in einem separaten Pod) mit jeweils 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. 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.

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 129).

Vorgehensweise

1. Überprüfen Sie die drei Optionen.

   Eine umfassende Beschreibung dieser Optionen enthält „Hochverfügbarkeit für IBM MQ in Containern planen” auf Seite 18.
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 CD CP4I-SC2 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 112 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 CD CP4I-SC2 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 in „Ziel-HA-Architektur für IBM MQ in Containern auswählen" auf Seite 127beschrieben, gibt es drei mögliche Implementierungstopologien. Daher enthält dieser Abschnitt drei verschiedene Vorlagen:

- Vorlage 1: Implementieren Sie einen Einzel-Instanz-Warteschlangenmanager.
- Vorlage 2: Stellen Sie einen Warteschlangenmanager mit mehreren Instanzen bereit.
- Vorlage 3: Native HA-Warteschlangenmanager implementieren.

Wichtig: Führen Sie nur eine der drei Vorlagen auf der Basis Ihrer bevorzugten Topologie aus.

Prozedur

- **Vorlage 1. 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: 9.4.0.0-r1
  license:
    accept: true
    license: L-BMSF-5YDSLR
    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: 9.4.0.0-r1
  license:
    accept: true
    license: L-BMSF-5YDSLR
    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 145 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.

- **Vorlage 2: Warteschlangenmanager mit mehreren Instanzen implementieren**

  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: 9.4.0.0-r1
  license:
    accept: true
    license: L-BMSF-5YDSLR
    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.

- **Vorlage 3. Implementieren Sie einen nativen HA-Warteschlangenmanager.**

Ein Beispiel für die Erstellung eines nativen HA-Warteschlangenmanagers finden Sie im Artikel „Beispiel: Native HA mit IBM MQ Operator konfigurieren” auf Seite 79.

OpenShift CD CP4I-SC2 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 129 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 auf IBM MQ Operator” auf Seite 110 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 88.

IBM MQ in Containern betreiben

Weitere Informationen zum Betrieb oder zur Interaktion mit IBM MQ -Warteschlangenmanagern, die in Containern ausgeführt werden, finden Sie in den folgenden Abschnitten.

Prozedur

- „Betreiben von IBM MQ mittels IBM MQ Operator” auf Seite 134.
- „Status von Warteschlangenmanagern mit Native HA anzeigen” auf Seite 142.
- „Native HA-Warteschlangenmanagerinstanzen manuell beenden” auf Seite 144.

OpenShift CP4I Betreiben von IBM MQ mittels IBM MQ Operator

Prozedur

- „Verbindung zur IBM MQ Console herstellen, die in einem Red Hat OpenShift-Cluster implementiert ist” auf Seite 134.
- „Überwachung bei Verwendung von IBM MQ Operator” auf Seite 135.
- „Warteschlangenmanagerkonfiguration über die Red Hat OpenShift-CLI sichern und wiederherstellen” auf Seite 141.

OpenShift CP4I Verbindung zur IBM MQ Console herstellen, die in einem Red Hat OpenShift-Cluster implementiert ist

Vorgehensweise zum Herstellen einer Verbindung zur IBM MQ Console eines Warteschlangenmanagers, der 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 UI. 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, verwendet IBM MQ Console Keycloak für das Identitäts-und Zugriffsmanagement. Siehe Identitäts-und Zugriffsmanagement in der IBM Cloud Pak for Integration -Dokumentation.

Wenn Sie eine IBM MQ -Lizenz verwenden, ist IBM MQ Console nicht vorkonfiguriert und Sie müssen es selbst konfigurieren. Weitere Informationen hierzu finden Sie im Abschnitt Benutzer und Rollen konfigurieren . Ein Beispiel hierzu finden Sie im Thema „IBM MQ Console mit einer Basisregistry mit IBM MQ Operator konfigurieren” auf Seite 102.

Zugehörige Tasks

„Route für Verbindung zu einem Warteschlangenmanager von außerhalb eines Red Hat OpenShift-Clusters konfigurieren” auf Seite 88
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 erteilen

Berechtigungen für IBM MQ Console werden abhängig von Ihrer Lizenznutzung unterschiedlich verwaltet.

Informationen zu diesem Vorgang

- Wenn Sie eine IBM Cloud Pak for Integration -Lizenz verwenden, verwendet IBM MQ Console Keycloak für das Identitäts-und Zugriffsmanagement.
  - Siehe Identitäts-und Zugriffsmanagement in der IBM Cloud Pak for Integration -Dokumentation.
  - Wenn Sie zuvor Benutzer mit IAM in älteren Versionen von IBM MQ Operatorkonfiguriert hatten, lesen Sie den Abschnitt Benutzer von IAM auf Keycloakmigrieren.
- Wenn Sie eine IBM MQ -Lizenz verwenden, ist IBM MQ Console nicht vorkonfiguriert und Sie müssen es selbst konfigurieren.
  - Weitere Informationen zu Benutzern und Rollen finden Sie unter Benutzer und Rollen konfigurieren.
  - Ein einfaches Beispiel finden Sie unter „IBM MQ Console mit einer Basisregistry mit IBM MQ Operator konfigurieren” auf Seite 102.
  - Alternativ können Sie IBM Cloud Pak for Integration Operator installieren, um Keycloak wie zuvor beschrieben zu konfigurieren.

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.

OpenShift Container Platform enthält einen bereits konfigurierten, vorinstallierten und selbst aktualisierten Überwachungsstack, der einen Prometheus-Server verwendet. Der Monitoring-Stack der 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.

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

Metrik	Typ	Beschreibung
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
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

Metrik	Typ	Beschreibung
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
ibmmq_qmgr_log_max_bytes	gauge	Protokoll - max. Bytes
ibmmq_qmgr_log_occupied_by_reusable_extents_bytes	gauge	Protokoll - durch wiederverwendbare Speicherbereiche belegte 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_required_for_media_recovery_bytes	gauge	Protokoll - für Datenträgerwiederherstellung erforderliche Bytes
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

Metrik	Typ	Beschreibung
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
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

Metrik	Typ	Beschreibung
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
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

Metrik	Typ	Beschreibung
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
ibmmq_qmgr_topic_put_bytes_total	counter	Gesamtbytes eingereihter Themen in Intervall
ibmmq_qmgr_trace_file_system_free_space_percentage	gauge	MQ-Tracedateisystem - freier Speicherplatz
ibmmq_qmgr_trace_file_system_in_use_bytes	gauge	MQ-Tracedateisystem - belegte Bytes
ibmmq_qmgr_user_cpu_time_estimate_for_queue_manager_percentage	gauge	Benutzer-CPU-Zeit - geschätzter Prozentsatz für Warteschlangenmanager
ibmmq_qmgr_user_cpu_time_percentage	gauge	Prozentsatz der Benutzer-CPU-Zeit

Zugehörige Informationen

Zu den Systemthemen veröffentlichte Metriken

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 mithilfe von **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
  ```

MQ Adv. Status von Warteschlangenmanagern mit Native HA 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

Befehl dspmq (Warteschlangenmanager anzeigen)

MQ Adv. Native HA-Warteschlangenmanagerinstanzen manuell beenden

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.

OpenShift CP4I Referenzinformationen für IBM MQ in Containern

IBM MQ stellt einen Kubernetes-Operator für die native Integration mit Red Hat OpenShift Container Platform bereit.

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 CP4I-SC2 *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`	Lizenzinformationen	Zutreffende IBM MQ-Versionen
`L-JTPV-KYG8TF`	`Production` oder `NonProduction`	IBM Cloud Pak for Integration 16.1.0	9.4.0
`L-BMSF-5YDSLR`	`Production` oder `NonProduction`	IBM Cloud Pak for Integration Limited Edition 16.1.0	9.4.0
`L-EHXT-MQCRN9`	`Production`	IBM MQ Advanced 9.4	9.4.0
`L-CLXQ-ADXTK3`	`Development`	IBM MQ Advanced for Developers (ohne Gewährleistung) 9.4	9.4.0

Beachten Sie, dass die *Version* der Lizenz angegeben ist, die nicht immer mit der Version von IBM MQ identisch ist.

Ältere Lizenzversionen

Siehe Ältere Lizenzversionen in der IBM MQ 9.3 -Dokumentation.

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. IBM MQ : https://ibm.biz/BdPZqj. Lizenzreferenz: https://ibm.biz/BdPZfq..

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.

Feld	Beschreibung
`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`	
`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 145

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.
`telemetry` Telemetrie	Einstellungen für die Open Telemetry-Konfiguration Erfordert MQ Operator 2.2.0 oder höher.

Feld	Beschreibung
`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.
`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 146

.spec.imagePullSecrets

LocalObjectReference enthält genug Informationen, damit Sie das referenzierte Objekt innerhalb desselben Namensbereichs lokalisieren können.

Wird angezeigt in:

- „.spec" auf Seite 146

Feld	Beschreibung
`name` Zeichenfolge	Name des Referenten. Weitere Informationen: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names TODO: Weitere nützliche Felder hinzufügen. 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 146

.spec.license

Einstellungen zur Steuerung der Annahme der Lizenz und der zu verwendenden Lizenzmetrik.

Wird angezeigt in:

- „.spec" auf Seite 146

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 finden Sie unter https://ibm.biz/BdPZfq .
`metric` Zeichenfolge	Einstellung zur Angabe der zu verwendenden Lizenzmetrik. Beispiel: `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 finden Sie unter https://ibm.biz/BdPZfq .

.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 146

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 148

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 148

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 148

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 149

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 146

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.

Feld	Beschreibung
`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`.
`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 Name eines Warteschlangenmanagers nicht den Kubernetes -Regeln für Namen entsprechen soll (z. B. ein Name, der Großbuchstaben enthält).
`readinessProbe` QueueManagerReadinessProbe	Einstellungen zur Steuerung des Bereitschaftstests.
`recoveryLogs` RecoveryLogs	Einstellungen für MQ -Wiederherstellungsprotokolle. Erfordert MQ Operator 2.4.0 oder höher.
`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 149

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 von Kubernetesautomatisch (in einigen Fällen) erneut gestartet wird. Verwenden Sie `MultiInstance` für ein Paar von Pods, von denen einer der `active` -Warteschlangenmanager und der andere ein Standby-Warteschlangenmanager 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.

Feld	Beschreibung
`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 150

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 149

Feld	Beschreibung
`configMap` ConfigMapINISource	'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 151

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 151

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 149

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 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 149

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 149

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 152

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 152

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 149

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.recoveryLogs

Einstellungen für MQ -Wiederherstellungsprotokolle. Erfordert MQ Operator 2.4.0 oder höher.

Wird angezeigt in:

- „.spec.queueManager” auf Seite 149

Feld	Beschreibung
`logFilePages` Ganzzahl	Die Wiederherstellungsprotokolldaten werden in einer Reihe von Dateien gespeichert. Die Protokolldateigröße wird in Einheiten von 4-KB-Seiten angegeben.

.spec.queueManager.resources

Einstellungen zur Steuerung der Ressourcenanforderungen.

Wird angezeigt in:

- „.spec.queueManager” auf Seite 149

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 154

Feld	Beschreibung
`cpu`	
`memory`	

.spec.queueManager.resources.requests

CPU- und Speichereinstellungen.

Wird angezeigt in:

- „.spec.queueManager.resources” auf Seite 154

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 149

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 149

Feld	Beschreibung
`failureThreshold` Ganzzahl	Mindestanzahl aufeinanderfolgender Fehler für den Test, damit er als fehlgeschlagen betrachtet wird. Der Standardwert ist 24.
`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 149

Feld	Beschreibung
`allowVolumeExpansion` boolesch	Gibt an, ob Datenträger erweitert werden dürfen.
`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.

Feld	Beschreibung
`recoveryLogs` QueueManagerOptionalVolume	Details zu Persistent Volumes, die in die MQ-Wiederherstellungsprotokolle geschrieben werden. Erforderlich bei Verwendung eines Multi-Instanz-Warteschlangenmanagers.
`scratch` Arbeitsdatenträger	Einstellungen für den temporären Datenträger des Warteschlangenmanagers. Dieser Datenträger wird als Ordner '/run' im Container angehängt. Nur anwendbar, wenn das Stammdateisystem schreibgeschützt ist. Erfordert MQ Operator 3.0.0 oder höher.
`tmp` Tmp	Einstellungen für den temporären Datenträger des Warteschlangenmanagers. Dieser Datenträger wird an den Container als Ordner/tmpangehängt. Die Diagnosedatendateien, wie z. B. die ZIP-Datei, die mit dem Befehl runmqras erstellt wird, werden auf diesem Datenträger erstellt. Nur anwendbar, wenn das Stammdateisystem schreibgeschützt ist. Erfordert MQ Operator 3.0.0 oder höher.

.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 155

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 `ephemeral` -Datenträgers. 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 und das Stammdateisystem schreibgeschützt ist. Erfordert MQ Operator 3.0.0 oder höher.
`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 155

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 `ephemeral` -Datenträgers. 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 und das Stammdateisystem schreibgeschützt ist. Erfordert MQ Operator 3.0.0 oder höher.
`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 155

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'.

Feld	Beschreibung
`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 `ephemeral` -Datenträgers. 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 und das Stammdateisystem schreibgeschützt ist. Erfordert MQ Operator 3.0.0 oder höher.
`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.scratch

Einstellungen für den temporären Datenträger des Warteschlangenmanagers. Dieser Datenträger wird als Ordner '/run' im Container angehängt. Nur anwendbar, wenn das Stammdateisystem schreibgeschützt ist. Erfordert MQ Operator 3.0.0 oder höher.

Wird angezeigt in:

- „.spec.queueManager.storage” auf Seite 155

Feld	Beschreibung
`sizeLimit` Zeichenfolge	Größenbegrenzung des ephemeren Datenträgers, einschließlich SI-Einheiten. Beispiel: `2Gi`. Nur gültig, wenn das Stammdateisystem schreibgeschützt ist. Erfordert MQ Operator 3.0.0 oder höher.

.spec.queueManager.storage.tmp

Einstellungen für den temporären Datenträger des Warteschlangenmanagers. Dieser Datenträger wird an den Container als Ordner/tmpangehängt. Die Diagnosedatendateien, wie z. B. die ZIP-Datei, die mit dem Befehl runmqras erstellt wird, werden auf diesem Datenträger erstellt. Nur anwendbar, wenn das Stammdateisystem schreibgeschützt ist. Erfordert MQ Operator 3.0.0 oder höher.

Wird angezeigt in:

- „.spec.queueManager.storage” auf Seite 155

Feld	Beschreibung
`sizeLimit` Zeichenfolge	Größenbegrenzung des ephemeren Datenträgers, einschließlich SI-Einheiten. Beispiel: `2Gi`. Nur gültig, wenn das Stammdateisystem schreibgeschützt ist. Erfordert MQ Operator 3.0.0 oder höher.

.spec.securityContext

Sicherheitseinstellungen, die dem Sicherheitskontext (securityContext) des Warteschlangenmanager-Pods hinzugefügt werden sollen.

Wird angezeigt in:

- „.spec” auf Seite 146

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.
`readOnlyRootFilesystem` boolesch	Gibt an, ob die Einstellungen des schreibgeschützten Stammdateisystems für den Warteschlangenmanager aktiviert werden sollen. Der Standardwert ist 'false'. Erfordert MQ Operator 3.0.0 oder höher.
`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.telemetry

Einstellungen für die Open Telemetry-Konfiguration Erfordert MQ Operator 2.2.0 oder höher.

Wird angezeigt in:

- „.spec" auf Seite 146

Feld	Beschreibung
`tracing` Traceerstellung	Einstellungen für die OpenTelemetry-Tracefunktion

.spec.telemetry.tracing

Einstellungen für die OpenTelemetry-Tracefunktion

Wird angezeigt in:

- „.spec.telemetry" auf Seite 159

Feld	Beschreibung
`instana` Exemplar	Einstellungen für Instana-Traceerstellung.

.spec.telemetry.tracing.instana

Einstellungen für Instana-Traceerstellung.

Wird angezeigt in:

- „.spec.telemetry.tracing" auf Seite 159

Feld	Beschreibung
`agentHost` Zeichenfolge	Der Hostname des Instana-Agenten, an den Tracedaten gesendet werden. Dies sollte kein Protokoll enthalten.
`enabled` boolesch	Gibt an, ob die Instana-Tracefunktion aktiviert werden soll. Der Standardwert ist 'false'.
`protocol` Zeichenfolge	Das Protokoll, das für die Kommunikation mit dem Instanzagenten verwendet werden soll. http und https werden unterstützt.

.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 146

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 146

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 160

Feld	Beschreibung
`image` Zeichenfolge	Das Container-Image, das verwendet wird.

Feld	Beschreibung
`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.agent.livenessProbe

Einstellungen zur Steuerung des Livetests.

Wird angezeigt in:

- „.spec.tracing.agent" auf Seite 160

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 160

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.

Feld	Beschreibung
`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 160

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 162

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 162

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.web

Einstellungen für den MQ-Web-Server.

Wird angezeigt in:

- „.spec" auf Seite 146

Feld	Beschreibung
`console` -Konsole	Einstellungen für die MQ -Webkonsole. Erfordert MQ Operator 3.0.0 oder höher.
`enabled` boolesch	Gibt an, ob der Web-Server aktiviert werden soll oder nicht. Der Standardwert ist 'false'.
`manualConfig` ManualConfig	Einstellungen für die Bereitstellung der Web-Server-XML-Konfiguration Erfordert MQ Operator 3.0.0 oder höher.

.spec.web.console

Einstellungen für die MQ -Webkonsole. Erfordert MQ Operator 3.0.0 oder höher.

Wird angezeigt in:

- „.spec.web" auf Seite 163

Feld	Beschreibung
`authentication` -Authentifizierung	Authentifizierungseinstellungen für die MQ -Webkonsole. Erfordert MQ Operator 3.0.0 oder höher.
`authorization` -Autorisierung	Berechtigungseinstellungen für die MQ -Webkonsole. Erfordert MQ Operator 3.0.0 oder höher.

.spec.web.console.authentication

Authentifizierungseinstellungen für die MQ -Webkonsole. Erfordert MQ Operator 3.0.0 oder höher.

Wird angezeigt in:

- „.spec.web.console" auf Seite 163

Feld	Beschreibung
`provider` Zeichenfolge	Der für die MQ -Webkonsole zu verwendende Authentifizierungsprovider Verwenden Sie `integration-keycloak` , um Single Sign-on mit der Benutzerschnittstelle von Cloud Pak for Integration Platform (Keycloak) zu verwenden. Standardwert ist `integration-keycloak` , wenn Sie eine Lizenz für Cloud Pak for Integration verwenden, oder `manual` , wenn Sie eine Lizenz für MQ verwenden Verwenden Sie `manual` , wenn Sie Ihre eigene Konfiguration bereitstellen möchten.

.spec.web.console.authorization

Berechtigungseinstellungen für die MQ -Webkonsole. Erfordert MQ Operator 3.0.0 oder höher.

Wird angezeigt in:

- „.spec.web.console" auf Seite 163

Feld	Beschreibung
`provider` Zeichenfolge	Der Berechtigungsprovider, der für die MQ -Webkonsole verwendet wird. Verwenden Sie `integration-keycloak` , um Rollen zu verwenden, die von Cloud Pak for Integration Keycloakbereitgestellt werden. Verwenden Sie `manual` , wenn Sie Ihre eigene Konfiguration bereitstellen möchten. Standardwert ist `integration-keycloak` , wenn Sie eine Lizenz für Cloud Pak for Integration verwenden, oder `manual` , wenn Sie eine Lizenz für MQ verwenden

.spec.web.manualConfig

Einstellungen für die Bereitstellung der Web-Server-XML-Konfiguration Erfordert MQ Operator 3.0.0 oder höher.

Wird angezeigt in:

- „.spec.web" auf Seite 163

Feld	Beschreibung
`configMap` ConfigMap	ConfigMap stellt eine Kubernetes ConfigMap dar, die die XML-Konfiguration des Web-Servers enthält.
`secret` Secret	Der geheime Schlüssel stellt einen geheimen Kubernetes -Schlüssel dar, der die XML-Konfiguration des Web-Servers enthält. Die Verwendung eines geheimen Schlüssels schützt alle Berechtigungsnachweise in der Kubernetes -Schicht, aber es ist möglich, dass Überwachungs-oder Fehlerbehebungstools die zugrunde liegende Datei unsicher zugänglich machen. Codieren Sie Berechtigungsnachweise mit "securityUtility", um die Sicherheit zu verbessern.

.spec.web.manualConfig.configMap

ConfigMap stellt eine Kubernetes ConfigMap dar, die die XML-Konfiguration des Web-Servers enthält.

Wird angezeigt in:

- „.spec.web.manualConfig" auf Seite 164

Feld	Beschreibung
`name` Zeichenfolge	Der Name der Kubernetes-Quelle.

.spec.web.manualConfig.secret

Der geheime Schlüssel stellt einen geheimen Kubernetes -Schlüssel dar, der die XML-Konfiguration des Web-Servers enthält. Die Verwendung eines geheimen Schlüssels schützt alle Berechtigungsnachweise in der Kubernetes -Schicht, aber es ist möglich, dass Überwachungs-oder Fehlerbehebungstools die zugrunde liegende Datei unsicher zugänglich machen. Codieren Sie Berechtigungsnachweise mit "securityUtility", um die Sicherheit zu verbessern.

Wird angezeigt in:

- „.spec.web.manualConfig” auf Seite 164

Feld	Beschreibung
name Zeichenfolge	Der Name der Kubernetes-Quelle.

.Status

Der Ist-Status des QueueManager.

Wird angezeigt in:

- „QueueManager” auf Seite 145

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.
metadata Metadaten	Metadaten stellen zusätzliche Informationen für den Warteschlangenmanager dar, einschließlich Integration-StatusKeycloak .
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 165

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 165

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 165

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.metadata

Metadaten stellen zusätzliche Informationen für den Warteschlangenmanager dar, einschließlich Integration-StatusKeycloak .

Wird angezeigt in:

- „.Status” auf Seite 165

Feld	Beschreibung
`integrationKeycloak` IntegrationKeycloak	QueueManagerStatusIntegrationKeycloak definiert die Integration-Keycloak -Status für QueueManager.

.status.metadata.integrationKeycloak

QueueManagerStatusIntegrationKeycloak definiert die Integration-Keycloak -Status für QueueManager.

Wird angezeigt in:

- „.status.metadata” auf Seite 166

Feld	Beschreibung
`clientName` Zeichenfolge	

.status.versions

Verwendete MQ-Version und weitere Versionen, die aus der IBM Entitled Registry verfügbar sind.

Wird angezeigt in:

- „.Status” auf Seite 165

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 166

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 167

Feld	Beschreibung
Array `licenses` Licenses	Lizenzen, die für diese Version von QueueManagergelten
`name` Zeichenfolge	Version `name` für diese Version von QueueManager. Hierbei handelt es sich um gültige Werte für das Feld `spec.version`.

.status.versions.available.versions.licenses

QueueManagerStatusLicense definiert eine Lizenz.

Wird angezeigt in:

- „.status.versions.available.versions" auf Seite 167

Feld	Beschreibung
`displayName` Zeichenfolge	Anzeigename für die Lizenz.
`link` Zeichenfolge	Link zum Lizenzinhalt.
`matchesCurrentType` boolesch	Gibt an, ob die Lizenz mit dem derzeit verwendeten Lizenztyp übereinstimmt.
`name` Zeichenfolge	Name der Lizenz.

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`**-Bedin-

gungen ausgegeben. Für einen solchen Warteschlangenmanager können jedoch **Warning**-Bedingungen vorliegen, die lediglich empfehlenden Charakter haben.

Die folgenden Bedingungen sind für eine `QueueManager`-Ressource definiert:

Tabelle 2. Statusbedingungen für Warteschlangenmanager

Komponente	Bedingungstyp	Ursachencode	Nachrichtenwarnung
QueueManager[3]	Geblockt	OperatorDependency	Für die Installation dieser Instanz muss Keycloak von [IBM Cloud Pak for Integration] konfiguriert werden. Diese Instanz verbleibt im Status [Pending], bis Keycloak als [KeycloakReady] in der Ressource Cp4iServicesBinding für diesen QueueManagergemeldet wird.
			Für die Installation ist für diese Instanz der Operator [IBM IAM] erforderlich. Diese Instanz verbleibt im Status [Blockiert], bis der Operator von [IBM Cloud Pak Foundational Services] installiert wird.
	Anstehend	Wird erstellt	Der MQ-Warteschlangenmanager wird bereitgestellt.
	Anstehend	OidcPending	Der MQ-Warteschlangenmanager wartet auf die OIDC-Clientregistrierung.
	Anstehend	Gestoppt	Der MQ -Warteschlangenmanager wurde gestoppt, da die Annotation 'mq.ibm.com/stop' vorhanden und in der Definition QueueManager auf 'true' gesetzt ist. Wenn die Anzahl der Replikate für QueueManager StatefulSet auf null gesetzt wird, werden alle Pods des MQ -Warteschlangenmanagers entfernt.
	Fehler	Fehlgeschlagen	Der MQ-Warteschlangenmanager konnte nicht bereitgestellt werden.
	Warnung	UnsupportedVersion	Ein Operand wurde von einem Operator installiert, der von OCP-Version *<ocp_version>* nicht unterstützt wird. Dieser Operand wird nicht unterstützt.
	Warnung	Unterstützung von CP4I-LTS	Ein CP4I-LTS Operand *< mq-version>* wurde installiert, wird aber von einem Operator verwaltet, der nicht für die Dauer der erweiterten Unterstützung qualifiziert ist. Dieser Operand ist nicht für die erweiterte Unterstützungsdauer qualifiziert.
	Warnung	Unterstützung von CP4I-LTS	Ein CP4I-LTS Operand *< mq_version>* wurde installiert, aber die OCP-Version *< ocp_version>* ist nicht für diese erweiterte Unterstützungsdauer qualifiziert. Dieser Operand ist nicht für die erweiterte Unterstützungsdauer qualifiziert.

[3] 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 Webkonsole aktiviert ist, protokolliert die Bedingung `OidcPending` den Status des Warteschlangenmanagers, während darauf gewartet wird, dass die OIDC-Clientregistrierung bei IAM abgeschlossen wird.

Tabelle 2. Statusbedingungen für Warteschlangenmanager (Forts.)

Komponente	Bedingungstyp	Ursachencode	Nachrichtenwarnung
Pod[4]	Anstehend	PodPending	Der Pod für den MQ-Warteschlangenmanager wird bereitgestellt.
	Fehler	PodFailed	Der Pod für den MQ-Warteschlangenmanager wird bereitgestellt.
Speicher[5]	Anstehend	StoragePending	Speicher für MQ-Warteschlangenmanager wird bereitgestellt.
	Warnung	StorageEphemeral	Für einen MQ-Warteschlangenmanager in einer Produktionsumgebung wird ephemerer Speicher verwendet.
	Warnung	StorageExpansionAnstehend	Datenträgererweiterung steht für die folgenden PVCs an [< Liste der PVCs >]
	Warnung	StorageMismatch	Die in der Ressource QueueManager definierten Speichergrößen stimmen nicht mit der Kapazität eines oder mehrerer bereitgestellter PVCs [< Liste der PVCs >] überein. AllowVolumeExpansion ist in der Ressource QueueManager auf ' false' gesetzt, sodass der MQ -Operator nicht versucht, diese Unterschiede auszugleichen.
	Fehler	StorageFailed	Der Speicher für den MQ-Warteschlangenmanager konnte nicht bereitgestellt werden.

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.

---

[4] 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).

[5] Speicherbedingungen überwachen den Fortschritt (Bedingung `StoragePending`) von Anforderungen für die Erstellung von Datenträgern für persistenten Speicher und sie melden Bindungs- und andere Fehler zurück. Speicherbedingungen überwachen auch den Fortschritt von Datenträgererweiterungen und warnen vor Abweichungen zwischen Speichergrößen, die in der Warteschlangenmanagerdefinition definiert sind, und der Größe implementierter PVCs. Wenn während der Speicherbereitstellung ein Fehler auftritt, wird die Bedingung `StorageFailed` zur Bedingungsliste hinzugefügt und die Gesamtbedingung des Warteschlangenmanagers wird auf `Failed`gesetzt.

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. Verwenden Sie abhängig von Ihrer Berechtigung und Ihren Funktionen, die im Container bereitgestellt werden, eine oder mehrere der folgenden Annotationen.

Anmerkung: Viele Anmerkungen enthalten eine oder beide der folgenden Zeilen:

```
productChargedContainers: "All" | "NAME_OF_CONTAINER"
productMetric: "PROCESSOR_VALUE_UNIT" | "VIRTUAL_PROCESSOR_CORE"
```

Sie müssen die folgenden Zeilen bearbeiten, bevor Sie die Anmerkung verwenden können:

- Für `productChargedContainers`müssen Sie `"All"`auswählen oder den tatsächlichen Namen des Containers ersetzen.
- Für `productMetric`müssen Sie einen der angebotenen Werte auswählen.

Annotationen zur Verwendung mit einer IBM MQ -Produktberechtigung

Wenn Sie über eine IBM MQ Produktberechtigung verfügen, wählen Sie die Anmerkung unten aus, die der erworbenen Berechtigung entspricht und verwendet werden soll.

- „IBM MQ” auf Seite 173
- „IBM MQ Erweitert” auf Seite 173
- „IBM MQ für Nicht-Produktionsumgebung” auf Seite 173
- „IBM MQ Advanced for Non-Production Environment” auf Seite 173
- „IBM MQ Advanced für Entwickler” auf Seite 173

Die IBM MQ -Annotationen, die mit IBM MQ -Hochverfügbarkeitskonfigurationen mit mehreren Instanzen verwendet werden sollen, lauten wie folgt. Weitere Informationen hierzu finden Sie im Abschnitt „Richtige Annotationen für Hochverfügbarkeitskonfigurationen auswählen” auf Seite 172.

- „IBM MQ -Container mit mehreren Instanzen” auf Seite 173
- „IBM MQ Advanced Container-Multi-Instanz” auf Seite 174
- „IBM MQ Container Multi Instance for Non-Production Environment” auf Seite 174
- „IBM MQ Advanced Container Multi Instance for Non-Production Environment” auf Seite 174

Anmerkungen zur Verwendung mit CP4I -Produktberechtigung

Wenn Sie über eine IBM Cloud Pak for Integration -Berechtigung (CP4I) verfügen, wählen Sie unten die Anmerkung aus, die mit der erworbenen Berechtigung übereinstimmt und verwendet werden soll.

- „IBM MQ mit CP4I -Berechtigung” auf Seite 174
- „IBM MQ Advanced mit CP4I -Berechtigung” auf Seite 174
- „IBM MQ for Non-Production Environment mit CP4I -Berechtigung” auf Seite 174
- „IBM MQ Advanced for Non-Production Environment mit Berechtigung CP4I” auf Seite 174

Die CP4I -Annotationen, die mit IBM MQ -Hochverfügbarkeitskonfigurationen mit mehreren Instanzen verwendet werden sollen, lauten wie folgt. Weitere Informationen hierzu finden Sie im Abschnitt „Richtige Annotationen für Hochverfügbarkeitskonfigurationen auswählen” auf Seite 172.

- „IBM MQ Containermehrinstanz mit CP4I -Berechtigung” auf Seite 175

- „IBM MQ Advanced Container Multi Instance mit CP4I -Berechtigung” auf Seite 175
- „IBM MQ Container Multi Instance for Non-Production Environment mit CP4I -Berechtigung” auf Seite 175
- „IBM MQ Advanced Container Multi Instance for Non-Production Environment with CP4I -Berechtigung” auf Seite 175

Richtige Annotationen für Hochverfügbarkeitskonfigurationen auswählen

IBM MQ Mehrfachinstanz

Wenn Sie ein Warteschlangenmanagerpaar in einer IBM MQ -Hochverfügbarkeitskonfiguration mit mehreren Instanzen implementieren, sollten Sie dieselbe Annotation für beide Instanzen verwenden. Je nach erworbener Berechtigung sollte eine der folgenden Annotationen ausgewählt werden:

- Eigenständige IBM MQ -oder IBM MQ Advanced -Berechtigung
  - „IBM MQ -Container mit mehreren Instanzen” auf Seite 173
  - „IBM MQ Advanced Container-Multi-Instanz” auf Seite 174
  - „IBM MQ Container Multi Instance for Non-Production Environment” auf Seite 174
  - „IBM MQ Advanced Container Multi Instance for Non-Production Environment” auf Seite 174
- IBM Cloud Pak for Integration Nutzungsrecht
  - „IBM MQ Containermehrinstanz mit CP4I -Berechtigung” auf Seite 175
  - „IBM MQ Advanced Container Multi Instance mit CP4I -Berechtigung” auf Seite 175
  - „IBM MQ Container Multi Instance for Non-Production Environment mit CP4I -Berechtigung” auf Seite 175
  - „IBM MQ Advanced Container Multi Instance for Non-Production Environment with CP4I -Berechtigung” auf Seite 175

Bei Verwendung mit IBM Cloud Pak for Integration -Berechtigung stellen die Berechtigungsverhältnisse in den Annotationen sicher, dass die korrekte Nutzungsrechtnutzung aufgezeichnet wird. Bei Verwendung mit eigenständigen IBM MQ -oder IBM MQ Advanced -Berechtigungen müssen die Annotationen, die im License Service für jede Instanz gemeldet werden, den IBM MQ -Berechtigungskomponenten wie folgt zugeordnet werden:

- IBM MQ Advanced container Multi-Instanz
  - 1 x IBM MQ Advanced **und** 1 x IBM MQ Advanced High Availability Replica **oder**
  - 2 x IBM MQ Advanced[6]
- IBM MQ Advanced container Multi Instance for Non-Production Environment
  - 1 x IBM MQ Advanced **und** 1 x IBM MQ Advanced High Availability Replica **oder**
  - 2 x IBM MQ Advanced für Nicht-Produktionsumgebung)[6]
- IBM MQ -Container mit mehreren Instanzen
  - 1 x IBM MQ **und** 1 x IBM MQ High Availability Replica **oder**
  - 2 x IBM MQ[6]
- IBM MQ Container Multi Instance for Non-Production Environment
  - 1 x IBM MQ **und** 1 x IBM MQ High Availability Replica **oder**
  - 2 x IBM MQ für Nicht-Produktionsumgebung) [6]

IBM MQ Native HA

[6] Diese Berechtigungsoption ist nicht optimal und sollte nur verwendet werden, wenn keine Berechtigung der relevanten High Availability Replica-Komponente verfügbar ist.

Wenn Sie drei Warteschlangenmanager in einem nativen HA-Quorum implementieren, verbraucht nur die aktive Instanz die Berechtigung. Alle Instanzen sollten dieselbe Annotation haben. Abhängig von der erworbenen Berechtigung sollte eine der folgenden Optionen ausgewählt werden:

- Eigenständige IBM MQ -oder IBM MQ Advanced -Berechtigung
  - „IBM MQ Erweitert” auf Seite 173
  - „IBM MQ Advanced for Non-Production Environment” auf Seite 173
- IBM Cloud Pak for Integration Nutzungsrecht
  - „IBM MQ Advanced mit CP4I -Berechtigung” auf Seite 174
  - „IBM MQ Advanced for Non-Production Environment mit Berechtigung CP4I” auf Seite 174

Anmerkungen

Der Rest dieses Abschnitts enthält Details zum Inhalt der einzelnen Annotationen.

IBM MQ

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

IBM MQ Erweitert

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

IBM MQ für Nicht-Produktionsumgebung

```
productID: "151bec68564a4a47a14e6fa99266deff"
productName: "IBM MQ for Non-Production Environment"
productMetric: "PROCESSOR_VALUE_UNIT" | "VIRTUAL_PROCESSOR_CORE"
productChargedContainers: "All" | "NAME_OF_CONTAINER"
```

IBM MQ Advanced for Non-Production Environment

```
productID: "21dfe9a0f00f444f888756d835334909"
productName: "IBM MQ Advanced for Non-Production Environment"
productMetric: "PROCESSOR_VALUE_UNIT" | "VIRTUAL_PROCESSOR_CORE"
productChargedContainers: "All" | "NAME_OF_CONTAINER"
```

IBM MQ Advanced für Entwickler

```
productID: "2f886a3eefbe4ccb89b2adb97c78b9cb"
productName: "IBM MQ Advanced for Developers (Non-Warranted)"
productMetric: "FREE"
productChargedContainers: "All" | "NAME_OF_CONTAINER"
```

IBM MQ -Container mit mehreren Instanzen

```
productID: "2dea73b866b648b6b4abe2a85eb76964"
productName: "IBM MQ Container Multi Instance"
```

```
productMetric: "PROCESSOR_VALUE_UNIT" | "VIRTUAL_PROCESSOR_CORE"
productChargedContainers: "All" | "NAME_OF_CONTAINER"
```

IBM MQ Advanced Container-Multi-Instanz

```
productID: "bd35bff411bb47c2a3f3a4590f33a8ef"
productName: "IBM MQ Advanced Container Multi Instance"
productMetric: "PROCESSOR_VALUE_UNIT" | "VIRTUAL_PROCESSOR_CORE"
productChargedContainers: "All" | "NAME_OF_CONTAINER"
```

IBM MQ Container Multi Instance for Non-Production Environment

```
productID: "af11b093f16a4a26806013712b860b60"
productName: "IBM MQ Container Multi Instance for Non-Production Environment"
productMetric: "VIRTUAL_PROCESSOR_CORE"
productChargedContainers: "All" | "NAME_OF_CONTAINER"
```

IBM MQ Advanced Container Multi Instance for Non-Production Environment

```
productID: "31f844f7a96b49749130cd0708fdbb17"
productName: "IBM MQ Advanced Container Multi Instance for Non-Production Environment"
productMetric: "VIRTUAL_PROCESSOR_CORE"
productChargedContainers: "All" | "NAME_OF_CONTAINER"
```

IBM MQ mit CP4I -Berechtigung

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

IBM MQ Advanced mit CP4I -Berechtigung

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

IBM MQ for Non-Production Environment mit CP4I -Berechtigung

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

IBM MQ Advanced for Non-Production Environment mit Berechtigung CP4I

```
cloudpakId: "c8b82d189e7545f0892db9ef2731b90d"
```

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

IBM MQ Containermehrinstanz mit CP4I -Berechtigung

```
productName: "IBM MQ Container Multi Instance"
productID: "2dea73b866b648b6b4abe2a85eb76964"
productChargedContainers: "All" | "NAME_OF_CONTAINER"
productMetric: "VIRTUAL_PROCESSOR_CORE"
productCloudpakRatio: "10:3"
cloudpakName: "IBM Cloud Pak for Integration"
cloudpakId: "c8b82d189e7545f0892db9ef2731b90d"
```

IBM MQ Advanced Container Multi Instance mit CP4I -Berechtigung

```
cloudpakId: "c8b82d189e7545f0892db9ef2731b90d"
cloudpakName: "IBM Cloud Pak for Integration"
productID: "bd35bff411bb47c2a3f3a4590f33a8ef"
productName: "IBM MQ Advanced Container Multi Instance"
productMetric: "VIRTUAL_PROCESSOR_CORE"
productChargedContainers: "All" | "NAME_OF_CONTAINER"
productCloudpakRatio: "5:3"
```

IBM MQ Container Multi Instance for Non-Production Environment mit CP4I -Berechtigung

```
cloudpakId: "c8b82d189e7545f0892db9ef2731b90d"
cloudpakName: "IBM Cloud Pak for Integration"
productID: "af11b093f16a4a26806013712b860b60"
productName: "IBM MQ Container Multi Instance for Non-Production Environment"
productMetric: "VIRTUAL_PROCESSOR_CORE"
productChargedContainers: "All" | "NAME_OF_CONTAINER"
productCloudpakRatio: "20:3"
```

IBM MQ Advanced Container Multi Instance for Non-Production Environment with CP4I -Berechtigung

```
cloudpakId: "c8b82d189e7545f0892db9ef2731b90d"
cloudpakName: "IBM Cloud Pak for Integration"
productID: "31f844f7a96b49749130cd0708fdbb17"
productName: "IBM MQ Advanced Container Multi Instance for Non-Production Environment"
productMetric: "VIRTUAL_PROCESSOR_CORE"
productChargedContainers: "All" | "NAME_OF_CONTAINER"
productCloudpakRatio: "10:3"
```

OpenShift CP4I Kubernetes IBM MQ Advanced for Developers Container-Image

Für IBM MQ Advanced for Developersist ein vordefinierter Container-Image verfügbar. Dieses Image ist in IBM Container Registryverfügbar. Dieses Image ist für die Verwendung mit Docker, Podman, Kubernete-sund anderen Containerumgebungen geeignet.

Verfügbare Images

IBM MQ -Images werden in IBM Container Registrygespeichert:

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

Kurzübersicht

- Lizenz:
  - Lizenzierungsreferenz für mq.ibm.com/v1beta1 und Apache License 2.0. Beachten Sie, dass die IBM MQ Advanced for Developers -Lizenz keine weitere Verteilung zulässt und die Bedingungen die Nutzung auf eine Entwicklermaschine beschränken.
- Speicherposition für Probleme:
  - GitHub
- Verfügbar für die folgenden CPU-Architekturen:
  - amd64
  - s390x
  - ppc64le

Verwendung

Führen Sie IBM MQ Advanced for Developers in einem Container aus.

Details zur Ausführung eines Containers finden Sie in der Dokumentation zur Verwendung .

Um das Image verwenden zu können, müssen Sie die Bedingungen der IBM MQ -Lizenz akzeptieren, indem Sie die Umgebungsvariable **LICENSE** festlegen.

Unterstützte Umgebungsvariablen

LANG
Legen Sie die Sprache fest, in der die Lizenz gedruckt werden soll.

LICENSE
Legen Sie `accept` fest, um den IBM MQ Advanced for Developers -Lizenzbedingungen zuzustimmen.

Legen Sie `view` fest, damit die Lizenzbedingungen angezeigt werden.

Deprecated **MQ_ADMIN_PASSWORD**
Geben Sie das Kennwort des Benutzers mit Administratorberechtigung an.

Muss mindestens 8 Zeichen lang sein.

Es gibt kein Standardkennwort für den Benutzer mit Administratorberechtigung.

V 9.4.0 V 9.4.0 Ab IBM MQ 9.4.0wird diese Variable nicht mehr bereitgestellt. Das YAML-Beispiel in diesem Abschnitt zeigt, wie Sie diese Variable selbst erstellen und mit einem geheimen Schlüssel schützen können.

Deprecated **MQ_APP_PASSWORD**
Geben Sie das Kennwort des App-Benutzers an.

Wenn diese Option festgelegt ist, wird der **DEV.APP.SVRCONN** -Kanal geschützt und ermöglicht nur Verbindungen, die eine gültige Benutzer-ID und ein gültiges Kennwort bereitstellen.

Muss mindestens 8 Zeichen lang sein.

Es gibt kein Standardkennwort für den App-Benutzer.

V 9.4.0 V 9.4.0 Ab IBM MQ 9.4.0wird diese Variable nicht mehr bereitgestellt. Das YAML-Beispiel in diesem Abschnitt zeigt, wie Sie diese Variable selbst erstellen und mit einem geheimen Schlüssel schützen können.

MQ_DEV
Legen Sie `false` fest, um die Erstellung der Standardobjekte zu stoppen.

MQ_ENABLE_, METRIKEN
Legen Sie `true` fest, um Prometheus -Metriken für Ihren Warteschlangenmanager zu generieren.

MQ_LOGGING_CONSOLE_QUELLE
Geben Sie eine durch Kommas getrennte Liste mit Quellen für Protokolle an, die an der Position **stdout** des Containers gespiegelt werden.

Gültige Werte sind `qmgr`, `web` und `mqsc`.

Der Standardwert ist `qmgr, web`.

Der optionale Wert ist `mqsc`. Diese Option kann verwendet werden, um den Inhalt von `autocfgmqsc.LOG` im Containerprotokoll wiederzugeben.

MQ_LOGGING_CONSOLE_FORMAT
Ändern Sie das Format der Protokolle, die an die **stdout** -Position des Containers ausgegeben werden.

Legen Sie `basic` fest, um ein einfaches lesbares Format zu verwenden. Dies ist der Standardwert.

Legen Sie `json` fest, um das JSON-Format zu verwenden (ein JSON-Objekt in jeder Zeile).

MQ_LOGGING_CONSOLE_EXCLUDE_ID
Geben Sie eine durch Kommas getrennte Liste von Nachrichten-IDs für ausgeschlossene Protokollnachrichten an.

Die Protokollnachrichten werden weiterhin in der Protokolldatei auf Platte angezeigt, aber nicht an der Position **stdout** des Containers ausgegeben.

Standardwert: `AMQ5041I,AMQ5052I,AMQ5051I,AMQ5037I,AMQ5975I`.

mq_qmgr_name
Legen Sie den Namen fest, mit dem Ihr WS-Manager erstellt werden soll.

Weitere Informationen zur Standardentwicklerkonfiguration, die vom IBM MQ Advanced for Developers -Image unterstützt wird, finden Sie in der Dokumentation zur Standardentwicklerkonfiguration.

Beispiel-YAML-Datei für Warteschlangenmanager, die beschreibt, wie Kennwörter für admin -und app -Benutzer angegeben werden

Für Benutzer der Benutzer-IDs **admin** und **app** müssen Sie Kennwörter angeben, wenn Sie einen Warteschlangenmanager mit der Lizenz für `Development` implementieren. Im Folgenden sehen Sie ein Beispiel für einen Warteschlangenmanager-YAML-Code, der Ihnen zeigt, wie dies mit dem IBM MQ Operatorgeschieht.

Der folgende Befehl erstellt einen geheimen Schlüssel, der Kennwörter für **admin** -und **app** -Benutzer enthält.

```
oc create secret generic my-mq-dev-passwords --from-literal=dev-admin-password=passw0rd --from-
literal=dev-app-password=passw0rd
```

Die folgende YAML-Datei verwendet diese Kennwörter bei der Implementierung eines Warteschlangenmanagers.

```
apiVersion: mq.ibm.com/v1beta1
kind: QueueManager
metadata:
  name: qm-dev
spec:
  license:
    accept: false
    license: L-CLXQ-ADXTK3
    use: Development
  web:
    enabled: true
  template:
    pod:
      containers:
        - env:
            - name: MQ_DEV
              value: "true"
            - name: MQ_CONNAUTH_USE_HTP
              value: "true"
            - name: MQ_ADMIN_PASSWORD
              valueFrom:
```

```
                secretKeyRef:
                  name: my-mq-dev-passwords
                  key: dev-admin-password
            - name: MQ_APP_PASSWORD
              valueFrom:
                secretKeyRef:
                  name: my-mq-dev-passwords
                  key: dev-app-password
          name: qmgr
  queueManager:
    storage:
      queueManager:
        type: persistent-claim
    name: QUICKSTART
  version: 9.4.0.0-r1
```

Fehlerbehebung für IBM MQ in Containern

Wenn bei der Ausführung von IBM MQ in einem Container Probleme auftreten, können Sie die hier beschriebenen Verfahren verwenden, um die Probleme zu diagnostizieren und zu beheben.

Prozedur

- „Fehlerbehebung bei ungeplanten Neustarts von IBM MQ in Containern” auf Seite 178.
- „Fehlerbehebung bei Problemen mit IBM MQ Operator” auf Seite 179.

OpenShift CP4I Kubernetes Fehlerbehebung bei ungeplanten Neustarts von IBM MQ in Containern

In den meisten Containerverwaltungssystemen wie Red Hat OpenShift Container Platform und Kuberneteswerden Container in der Regel erneut gestartet. Es ist nicht normal, dass ein Container langlebig ist. In diesem Abschnitt werden der Containerlebenszyklus, die Vorgehensweise zum Untersuchen eines Neustarts und die Ursachen für einen ungeplanten Containerneustart erläutert.

Wenn Sie keine Probleme mit Ihrer IBM MQ -Implementierung festgestellt haben und sie weiterhin wie erwartet ausgeführt wird, wird die Lösung wahrscheinlich wie beabsichtigt ausgeführt. Im Containerprotokoll wird möglicherweise eine Protokollnachricht wie die folgende angezeigt:

```
Signal received: terminated
```

Dies bedeutet, dass das Signal SIGTERM an den MQ -Container gesendet wurde und ihn zur Beendigung auffordert. Linux -Container sind dafür zuständig, auf POSIX -Signale zu antworten, bei denen es sich um standardisierte Nachrichten handelt, die an ein Programm gesendet werden, um Verhalten auszulösen.

Wenn der IBM MQ -Container ein SIGTERM-Signal empfängt, gibt er einen `endmqm -w -r -tp` -Befehl aus, um den Warteschlangenmanager zu stoppen. Sobald der Warteschlangenmanager gestoppt wurde, wird der Container gestoppt. Wenn das Stoppen des Warteschlangenmanagers lange dauert, wird möglicherweise ein SIGKILL-Signal gesendet, das die Linux -Prozesse sofort beendet. Die Zeit zwischen SIGTERM und SIGKILL wird in Kubernetesals "Beendigungskarenzzeit" bezeichnet und kann in der `QueueManager` -Ressource (wenn Sie IBM MQ Operatorverwenden) oder direkt in der Pod-Ressource konfiguriert werden. Der Standardwert ist 30 Sekunden, von denen eine Sekunde für die Beendigung des Containers reserviert ist und der Rest an IBM MQübergeben wird. Im Standardfall wird beispielsweise ein `endmqm -w -tp 29` ausgegeben, der dem Warteschlangenmanager mitteilt, dass er 29 Sekunden zum Beenden benötigt.

Gründe für die Pod-Bereinigung

Das Signal SIGTERM wird von Kubernetes (und damit Red Hat OpenShift Container Platform) verwendet, um einen Pod ordnungsgemäß zu beenden. Weitere Informationen finden Sie unter Beendigung von Pods in der Dokumentation zu Kubernetes . Kubernetes verwendet die Begriffe "Pod-Unterbrechung" und "Bereinigung" für den Prozess, durch den Pods auf Knoten entweder freiwillig oder unfreiwillig beendet werden. Es gibt viele Gründe, warum ein Pod entfernt werden kann, darunter:

- **Beendigung durch kubelet**. Dies kann verschiedene Ursachen haben, darunter die folgenden:
  - Der Pod kann beendet werden, weil der Knoten heruntergefahren wird (möglicherweise als Teil einer rollierenden Clusteraktualisierung)
  - Der Pod kann aufgrund eines Knotendrucks beendet werden (wobei das Kubelet Pods proaktiv beendet, um Ressourcen auf einem Knoten freizugeben). Der Kubernetes -Clusteradministrator kann Bereinigungsschwellenwerte konfigurieren, die je nach Cluster variieren können.
  - Der Pod kann beendet werden, weil der Aktivitätsprüfung für den Pod fehlgeschlagen ist. In Kubernetes kann ein Aktivitätsprüfung konfiguriert werden, um zu überprüfen, ob ein Pod noch in einwandfreiem Zustand ist. Der IBM MQ Operator richtet einen Liveness-Test des Warteschlangenmanagers ein, der den Befehl **dspmq** aufruft, um nach einem gültigen Ausführungsstatus zu suchen. Wenn sich der Warteschlangenmanager nicht in einwandfreiem Zustand befindet oder wenn die Ausführung des Testmonitors selbst zu lange dauert, wird der Fehler vom Kubelet berücksichtigt. Schwellenwerte für die Anzahl der zu tolerierenden Fehler können in der `QueueManager` -Ressource (wenn Sie die IBM MQ Operatorverwenden) oder direkt in der Pod-Ressource konfiguriert werden.
- **Zurückstellung durch den Kubernetes -Scheduler**. Dies kann auftreten, wenn der Kubernetes -Scheduler einen Pod mit höherer Priorität ausführen muss.
- **Getönter Knoten**. Ein Knoten kann einen "Taint" aufweisen und Pods, die den Taint nicht tolerieren, werden entfernt. Taints werden von Kubernetes -Administratoren verwendet, um Pods von bestimmten Knoten abzuwehren. Um beispielsweise zu sagen, dass IBM MQ -Pods nicht mehr auf Knoten ausgeführt werden sollen, die über spezielle Hardware verfügen, die jetzt für andere Workloads reserviert ist.
- **Anforderung über die Räumungs-API**. Dies kann von einem Administrator aufgerufen werden, um Pods zu bereinigen.
- **Pod-Garbage-Collection**. Dies kann auftreten, wenn der Knoten außer Betrieb geht oder über die API Kubernetes entfernt wird.

Feststellen, warum ein Warteschlangenmanager-Pod entfernt wurde

Mögliche Informationsquellen, um zu verstehen, warum ein Pod entfernt wurde:

- **Clusterereignisse**. Beispiel: Anzeigen von Systemereignisinformationen in einem OpenShift Container Platform -Cluster.
- **Clusterprüfereignisse**. Siehe Prüfprotokolle in Red Hat OpenShift Container Platformanzeigen.
- **Knoten unter Druck**. Suchen Sie nach Knoten unter CPU-, Netz-oder Speicherdruck. Dies wird im Knotenstatus angezeigt. Beachten Sie, dass der Knoten bei der Suche möglicherweise nicht mehr unter Druck steht.
- **Red Hat OpenShift Container Platform Monitoring** oder andere Überwachungsmetriken können möglicherweise Dinge wie Plattenlatenzprobleme anzeigen. Eine nützliche Prometheus -Metrik ist `ibmmq_qmgr_log_write_latency_seconds`. Diese Informationen stammen aus den MQ -Statistikthemen.

Zugehörige Informationen

Kubernetes -Dokumentation zur Planung, Zurückstellung und Räumung

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

- „Fehlerbehebungsinformationen für Warteschlangenmanager erfassen, die mit IBM MQ Operator implementiert wurden” auf Seite 180
- „Fehlerbehebung: Zugriff auf Warteschlangenmanagerdaten erhalten” auf Seite 181

OpenShift CP4I Fehlerbehebungsinformationen für Warteschlangenmanager erfassen, die mit IBM MQ Operator implementiert wurden

Erfassen von Fehlerbehebungsinformationen, die dem IBM Support beim Auslösen eines neuen Supportfalls bereitgestellt werden sollten.

Vorgehensweise

1. Informationen zum Cloud-Provider erfassen.

   Dies ist der Cloud-Provider, der Ihren Red Hat OpenShift -Cluster hostet (z. B. IBM Cloud).
2. Architekturinformationen erfassen.

   Die Architektur Ihres Red Hat OpenShift -Clusters ist eine der folgenden:

   - Linux for x86-64
   - Linux on Power Systems (ppc64le)
   - Linux for IBM Z
3. IBM MQ -Implementierungsinformationen erfassen.

   a) Melden Sie sich mit einer `bash/zsh` -Shell bei Ihrem Red Hat OpenShift -Cluster an.

   b) Setzen Sie die folgenden Umgebungsvariablen:

   ```
   export QM=QueueManager_name
   export QM_NAMESPACE=QueueManager_namespace
   export MQ_OPERATOR_NAMESPACE=mq_operator_namespace
   ```

   Dabei ist *Warteschlangenmanagername* der Name Ihrer QueueManager -Ressource, *Warteschlangenmanager-Namensbereich* der Namensbereich, in dem sie bereitgestellt wird, und *mq_operator_namespace* der Namensbereich, in dem IBM MQ Operator bereitgestellt wird. Dies kann mit dem Namensbereich QueueManager identisch sein.

   c) Führen Sie die folgenden Befehle aus und stellen Sie dem IBM Support alle resultierenden Ausgabedateien bereit.

   ```
   # OCP / Kubernetes: Version
   oc version -o yaml > ocversion.yaml

   # QueueManager: YAML
   oc get qmgr $QM -n $QM_NAMESPACE -o yaml > "queue-manager-$QM.yaml"

   # MQ Queue Manager: Pods
   oc get pods -n $QM_NAMESPACE -o wide --selector "app.kubernetes.io/instance=$QM" > "qm-
   pods-$QM.txt"

   # MQ Queue Manager: Pod YAML
   oc get pods -n $QM_NAMESPACE -o yaml --selector "app.kubernetes.io/instance=$QM" > "qm-
   pods-$QM.yaml"

   # MQ Queue Manager: Pod Logs
   for p in $(oc get pods -n $QM_NAMESPACE --no-headers --selector "app.kubernetes.io/ins
   tance=$QM" | cut -d ' ' -f 1); do oc logs -n $QM_NAMESPACE --previous "$p" > "qm-logs-pre
   vious-$p.txt"; oc logs -n $QM_NAMESPACE $p > "qm-logs-$p.txt";done

   # MQ Queue Manager: Describe Pods
   for p in $(oc get pods -n $QM_NAMESPACE --no-headers --selector "app.kubernetes.io/ins
   tance=$QM" | cut -d ' ' -f 1); do oc describe pod $p -n $QM_NAMESPACE > "qm-pod-describe-
   $p.txt"; done

   # MQ Web UI: Console Log
   for p in $(oc get pods -n $QM_NAMESPACE --no-headers --selector "app.kubernetes.io/ins
   tance=$QM" | cut -d ' ' -f 1); do oc cp -n $QM_NAMESPACE --retries=10 "$p:var/mqm/web/
   installations/Installation1/servers/mqweb/logs/console.log" "web-$p-console.log"; done

   # MQ Web UI: Messages Log
   for p in $(oc get pods -n $QM_NAMESPACE --no-headers --selector "app.kubernetes.io/ins
   tance=$QM" | cut -d ' ' -f 1); do oc cp -n $QM_NAMESPACE --retries=10 "$p:var/mqm/web/
   installations/Installation1/servers/mqweb/logs/messages.log" "web-$p-messages.log"; done

   # MQ Queue Manager: routes defined by operator
   ```

```
oc get routes -n $QM_NAMESPACE -o yaml --selector "app.kubernetes.io/instance=$QM" > "qm-
routes-$QM.yaml"

# MQ Queue Manager: routes to QM
oc get routes -n $QM_NAMESPACE -o yaml --field-selector "spec.to.name=$QM-ibm-mq" > "qm-
routes2-$QM.yaml"

# MQ Queue Manager: stateful set
oc get statefulset -n $QM_NAMESPACE -o yaml ${QM}-ibm-mq > "qm-statefulset-$QM.yaml"

# MQ Queue Manager: revisions of the stateful set
oc get controllerrevisions.apps -o yaml -n $QM_NAMESPACE --selector "app.kuberne
tes.io/instance=$QM" > "qm-statefulset-revisions-$QM.yaml"

# MQ Queue Manager: Pod events
for p in $(oc get pods -n $QM_NAMESPACE --no-headers --selector
"app.kubernetes.io/instance=$QM" | cut -d ' ' -f 1); do oc get
-o custom-columns="LAST SEEN:.lastTimestamp,TYPE:.type,REASON:.reason,KIND:.involvedOb
ject.kind,NAME:.involvedObject.name,MESSAGE:.message" event -n $QM_NAMESPACE --field-se
lector involvedObject.name="$p" > "qm-pod-events-$p.txt"; done

# MQ Queue Manager: StatefulSet events
oc get events -n $QM_NAMESPACE -o custom-columns="LAST SEEN:.lastTimestamp,TYPE:.type,REA
SON:.reason,KIND:.involvedObject.kind,NAME:.involvedObject.name,MESSAGE:.message" --field-
selector involvedObject.name="${QM}-ibm-mq" > "qm-statefulset-events-$QM.txt"

# MQ Queue Manager: services
oc get services -n $QM_NAMESPACE -o yaml --selector "app.kubernetes.io/instance=$QM" >
"qm-services-$QM.yaml"

# MQ Queue Manager: PVCs
oc get pvc -n $QM_NAMESPACE -o yaml --selector "app.kubernetes.io/instance=$QM" > "qm-
pvcs-$QM.yaml"

# MQ Operator: Version
oc get csv -n $QM_NAMESPACE | grep "^ibm-mq\|NAME" > mq-operator-csv.txt

# Cloud Pak Foundational Services: Version
oc get csv -n $QM_NAMESPACE | grep "^ibm-common-service-operator\|NAME" > common-services-
csv.txt

# Cloud Pak for Integration: Version (if applicable)
oc get csv -n $QM_NAMESPACE | grep "^ibm-integration-platform-navigator\|NAME" > cp4i-
csv.txt

# Output from runmqras (this may take a while to execute)
for p in $(oc get pods -n $QM_NAMESPACE --no-headers --selector "app.kubernetes.io/ins
tance=$QM" | cut -d ' ' -f 1); do timestamp=$(TZ=UTC date +"%Y%m%d_%H%M%S"); oc
exec -n $QM_NAMESPACE $p -- runmqras -workdirectory "/tmp/runmqras_$timestamp" -section
logger,mqweb,nativeha,trace; oc cp -n $QM_NAMESPACE --retries=10 "$p:tmp/runmqras_$time
stamp/" .; done

# MQ Operator: Pod Log
oc logs -n $MQ_OPERATOR_NAMESPACE $(oc get pods -n $MQ_OPERATOR_NAMESPACE --no-headers
--selector app.kubernetes.io/name=ibm-mq,app.kubernetes.io/managed-by=olm | cut -d ' ' -f
1) > mq-operator-log.txt
```

Anmerkung:

Die meisten dieser Befehle erfordern Zugriff auf den Namensbereich, in dem der Warteschlangenmanager bereitgestellt wird. Für die Erfassung des IBM MQ Operator -Protokolls kann jedoch zusätzlich der Zugriff eines **Clusteradministrators** erforderlich sein, wenn IBM MQ Operator auf **Clusterebene**installiert ist.

Zugehörige Tasks

Fehlerbehebungsinformationen für IBM Support erfassen

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) Listen Sie die PVC-Verzeichnisse mit dem folgenden Befehl auf:

   ```
   ls
   ```

   c) 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
   ```

Bemerkungen

Die vorliegenden Informationen wurden für Produkte und Services entwickelt, die auf dem deutschen Markt angeboten werden.

Möglicherweise bietet IBM die in diesem Dokument 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 Produkte, Programme oder Services von IBM bedeuten nicht, dass nur Produkte, Programme 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 dieses Handbuchs ist keine Lizenzierung dieser Patente verbunden. Lizenzanforderungen sind schriftlich an folgende Adresse zu richten (Anfragen an diese Adresse müssen auf Englisch formuliert werden):

IBM Director of Licensing
IBM Europe, Middle East & 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:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

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 hier enthaltenen Informationen werden in regelmäßigen Zeitabständen aktualisiert und als Neuausgabe veröffentlicht. IBM kann ohne weitere Mitteilung 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. Die Verwendung dieser Websites geschieht auf eigene Verantwortung.

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 & 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.

Alle 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. Sie sollen nur die Funktionen des Lizenzprogramms illustrieren und können Namen von Personen, Firmen, Marken oder Produkten enthalten. Sämtliche dieser Namen sind fiktiv. Ähnlichkeiten mit Namen und Adressen tatsächlicher Unternehmen oder Personen sind zufällig.

COPYRIGHTLIZENZ:

Diese Veröffentlichung enthält Beispielanwendungsprogramme, die in Quellensprache geschrieben sind und Programmiertechniken in verschiedenen Betriebsumgebungen veranschaulichen. Sie dürfen diese Beispielprogramme kostenlos ohne Zahlung an IBM in jeder Form 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 Beispielprogramme geschrieben sind. 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 zu geplanten Programmierschnittstellen, die es dem Kunden ermöglichen, Programme zum Abrufen der Services von IBM MQzu schreiben.

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 eingetragene 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: