9.1

IBM MQ in Containern

Hinweis

Vor Verwendung dieser Informationen und des darin beschriebenen Produkts sollten die Informationen unter „Bemerkungen” auf Seite 55 gelesen werden.

Diese Ausgabe bezieht sich auf Version 9 Release 1 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

Linux IBM MQ in Containern

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 in dem vordefinierten Container ausführen, der in IBM MQ Advanced und IBM MQ Advanced for Developers bereitgestellt wird. Dieser IBM MQ Advanced certified container bietet ein unterstütztes Image und Helm-Diagramm. Damit kann ein einsatzfähiges IBM MQ-Image in Red Hat® OpenShift®, IBM Cloud Private oder IBM Cloud Kubernetes Service bereitgestellt werden.

Sie können IBM MQ auch in einem IBM Cloud Pak for Integration-Container oder in einem selbst erstellten Container ausführen.

MQ Adv. CD Weitere Informationen zu IBM MQ Advanced certified container finden Sie unter den folgenden Links.

Linux MQ Adv. CD 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 die Architektur von IBM MQ in Containern planen, sollten Sie sich mit den Basiskonzepten von IBM MQ (siehe IBM MQ - Technische Übersicht) und mit den Basiskonzepten von Kubernetes/OpenShift (siehe OpenShift Container Platform architecture) vertraut machen.

Prozedur

- „Verwendung von IBM MQ in Containern auswählen” auf Seite 5.
- „Hohe Verfügbarkeit für IBM MQ Advanced certified container” auf Seite 10.
- „Benutzerauthentifizierung und -berechtigung für IBM MQ Advanced certified container” auf Seite 12.

Linux MQ Adv. CD Verwendung von IBM MQ in Containern auswählen

Es gibt mehrere Optionen für die Verwendung von IBM MQ in Containern: Sie können vordefinierte und zertifizierte Container verwenden oder Ihre eigenen Images und einen eigenen Bereitstellungscode erstellen.

Zertifizierte Container für IBM MQ Advanced verwenden

Wenn Sie eine Implementierung auf Red Hat OpenShift Container Platform planen, möchten Sie wahrscheinlich die zertifizierten Container verwenden. Zertifizierte Container gibt es in drei Versionen:

- IBM MQ Advanced certified container für IBM Cloud Pak for Integration. Dabei handelt es sich um ein separates IBM Produkt, das eine Version eines zertifizierten Containers enthält.
- IBM MQ Advanced certified container
- Zertifizierter Container für IBM MQ Advanced for Developers (ohne Gewährleistung)

IBM MQ 9.1.4 und frühere CD-Releases wurde ebenfalls auf IBM Cloud Private und IBM Cloud Kubernetes Service unterstützt.

Beachten Sie, dass sich die zertifizierten Container schnell entwickeln und daher nur unter Continuous Delivery-Releases unterstützt werden.

Die zertifizierten Container enthalten sowohl vordefinierte Container-Images als auch Bereitstellungscode für die Ausführung unter Red Hat OpenShift Container Platform. Ab IBM MQ 9.1.5 werden Warteschlangenmanager mit einem IBM MQ -Operator verwaltet. Frühere Versionen von IBM MQbis einschließlich Version 9.1.5werden mithilfe von Helm -Diagrammen verwaltet.

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

- REST-APIs für die Verwaltung oder Nachrichtenübermittlung verwenden
- Eine der folgenden MQ-Komponenten verwenden:
  - Managed File Transfer-Agenten und ihre Ressourcen. Sie können die zertifizierten Container jedoch verwenden, um einen oder mehrere Koordinations-, Befehls-oder Agentenwarteschlangenmanager bereitzustellen.
  - AMQP
  - IBM MQ Bridge to Salesforce
  - IBM MQ Bridge to blockchain (nicht unterstützt in Containern)
- Verwenden Sie den Web-Server bei der Implementierung mithilfe von Helm -Diagrammen (mit Ausnahme von IBM Cloud Pak for Integration).
- Optionen anpassen, die mit **crtmqm**, **strmqm** und **endmqm** verwendet werden, z. B. Konfiguration von Wiederherstellungsprotokollen

Eigene Images und Diagramme 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 „Eigenen IBM MQ-Container erstellen” auf Seite 35. Die Helm-Diagramme, die als Teil der zertifizierten Container bereitgestellt werden, werden auf GitHub veröffentlicht und können beim Erstellen Ihrer eigenen Images als Beispiele verwenden werden:

- Helm-Diagramm für IBM MQ Advanced certified container
- Helm-Diagramm zu zertifizierten Containern für IBM MQ Advanced for Developers

Zugehörige Konzepte

„Unterstützung für zertifizierte IBM MQ-Container” auf Seite 6
Die zertifizierten IBM MQ-Container werden nur in bestimmten Kubernetes-Umgebungen unterstützt

„Unterstützung für die Erstellung eigener IBM MQ-Container-Images und -Diagramme” auf Seite 8
Berücksichtigen Sie diese Informationen, wenn Sie Containers in einem Linux-System verwenden.

Linux Unterstützung für zertifizierte IBM MQ-Container

Die zertifizierten IBM MQ-Container werden nur in bestimmten Kubernetes-Umgebungen unterstützt

V 9.1.4 MQ Adv. CD Ab CD-Release V9.1.4 wird IBM MQ Advanced certified container für die Verwendung mit Red Hat OpenShift unterstützt. Siehe „Warteschlangenmanager über die Helm -CLI implementieren” auf Seite 25.

CD-Releases vor V9.1.4 wurden in den folgenden Kubernetes-Umgebungen unterstützt:

- IBM Cloud Kubernetes Service

- IBM Cloud Private
- IBM Cloud Private mit Red Hat OpenShift

Informationen zu bestimmten unterstützten Versionen von Kubernetes finden Sie in den Dateien `qualification.yaml` und `Chart.yaml` innerhalb eines heruntergeladenen IBM MQ Advanced-Helm-Diagramms. Diese Versionen variieren von Release zu Release.

Die IBM MQ Advanced certified container wird nur unterstützt, wenn sie mit dem IBM MQ -Operator bereitgestellt wird oder wenn eines der folgenden Helm -Diagramme verwendet wird:

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

Anmerkung: Die Verwendung von Helm -Diagrammen wird ab dem Release von IBM MQ Operator nicht mehr unterstützt.

Da sich die Containertechnologie schnell weiterentwickelt, wird IBM MQ Advanced certified container nur in der neuesten Version der Plattformen unterstützt, die dieses Diagramm zum Zeitpunkt der Freigabe unterstützt. Wenn Sie eine ältere Plattformversion verwenden möchten, benötigen Sie möglicherweise auch eine ältere Version von IBM MQ Advanced certified container.

Das IBM MQ Advanced certified container -Image basiert auf IBM MQ Continuous Delivery -Releases (CD-Releases). Diese werden bis zu einem Jahr lang oder für zwei CD-Releases unterstützt, je nachdem, was länger ist. Long Term Support -Releases von IBM MQ sind nicht als zertifizierter Container verfügbar.

Ab IBM MQ Advanced certified container V4.0 stellt das Image eine Installation von IBM MQ auf einem Red Hat Universal Base Image (UBI) bereit, was die wichtigsten Linux-Bibliotheken und -Dienstprogramme, die von IBM MQ verwendet werden, einschließt. Das UBI wird von Red Hat unterstützt, wenn die Ausführung auf einem Red Hat Enterprise Linux-Host erfolgt. In früheren Versionen von IBM MQ Advanced certified container wurde ein nicht unterstütztes Ubuntu-Basisimage verwendet.

Zugehörige Konzepte

„Unterstützung für die Erstellung eigener IBM MQ-Container-Images und -Diagramme” auf Seite 8
Berücksichtigen Sie diese Informationen, wenn Sie Containers in einem Linux-System verwenden.

Linux MQ Adv. CD Versionsunterstützung für IBM MQ Advanced certified container

Eine Gruppe von Tabellen, die die Zuordnung zwischen unterstützten Versionen von IBM MQ Advanced certified container, IBM MQ, IBM Cloud Kubernetes Service, IBM Cloud Pak for Integration und IBM Cloud Private zeigen.

IBM MQ Operator

V 9.1.5

Der IBM MQ -Operator wird als Teil von IBM Cloud Pak for Integration Version 2020.2oder unabhängig davon mit IBM MQ Version 9.1.5 und höher unterstützt.

IBM MQ Operator wird unter Red Hat OpenShift Container Platform Version 4.4 oder höher unterstützt.

IBM MQ Advanced certified container V 9.1.5 (Helm -Diagramm)-veraltet

Enthält das Helm-Diagramm `ibm-mqadvanced-server-prod`.

V 9.1.5 Ab IBM MQ Advanced certified container V5.0.x werden das Helm-Diagramm, das Image und die Programmkorrekturen über einen von IBM berechtigten Katalog und eine berechtigte Registry bereitgestellt. Frühere Versionen wurden über Passport Advantage bereitgestellt und die Fix-Releases sind auf IBM Fix Central verfügbar.

Tabelle 1. Unterstützung für IBM MQ Advanced certified container			
Version	**IBM MQ Version**	**Ende des Unterstützungszeitraums**	**Unterstützte Plattformen**
6.0.x	9.1.5 Continuous Delivery-Release	March 2021	Ausführliche Systemvoraussetzungen
5.0.x	9.1.4 Continuous Delivery-Release	Dezember 2020	Ausführliche Systemvoraussetzungen
4.1.x	9.1.3 Continuous Delivery-Release	Juli 2020	Ausführliche Systemvoraussetzungen

IBM MQ Advanced certified container -Software für IBM Cloud Pak for Integration V 9.1.5 (Helm -Diagramm)-veraltet

Enthält das Helm-Diagramm `ibm-mqadvanced-server-integration-prod`.

Tabelle 2. Versionsunterstützung für die IBM MQ Advanced certified container-Software für IBM Cloud Pak for Integration		
Version	**IBM MQ Version**	**IBM Cloud Pak for Integration Version**
6.0.x	9.1.4 Continuous Delivery-Release	2020.1.1 (Systemvoraussetzungen)
5.0.x	9.1.3 Continuous Delivery-Release	2019.4.1 (Systemvoraussetzungen)
4.1.x	9.1.3 Continuous Delivery-Release	2019.3.2.2 (Systemvoraussetzungen)
4.0.x	9.1.3 Continuous Delivery-Release	2019.3.2 (Systemvoraussetzungen)
3.0.x	9.1.3 Continuous Delivery-Release	2019.3.1 (Systemvoraussetzungen)

Im Abschnitt IBM Cloud Pak for Integration - Releaseinformationen finden Sie Informationen zu den unterstützten Versionen.

Linux Unterstützung für die Erstellung eigener IBM MQ-Container-Images und -Diagramme

Berücksichtigen Sie diese Informationen, wenn Sie Containers in einem Linux-System verwenden.

- Das vom Container verwendete Basisimage muss ein unterstütztes Linux-Betriebssystem verwenden.
- Sie müssen das Produkt mit den IBM MQ-Installationsprogrammen innerhalb des Container-Images installieren.
- Eine Liste der unterstützten Pakete finden Sie unter IBM MQ-RPM-Komponenten für Linux-Systeme.
- V 9.1.0 Die folgenden Pakete werden nicht unterstützt:
  - `MQSeriesBCBridge`
  - `MQSeriesRDQM`
- Das Warteschlangenmanagerdatenverzeichnis (standardmäßig `/var/mqm` muss auf einem Containerdatenträger gespeichert werden, der einen persistenten Zustand aufrechterhält.

  Wichtig: Das Union-Dateisystem kann nicht verwendet werden.

Sie müssen entweder ein Hostverzeichnis als Datenträger bereitstellen oder einen Datenträgercontainer verwenden. Weitere Informationen finden Sie unter Manage data in containers (Daten in Containern verwalten).

- Sie müssen IBM MQ-Steuerbefehle, z. B. **endmqm**, innerhalb des Containers ausführen können.
- Sie müssen zu Diagnosezwecken Dateien und Verzeichnisse aus dem Container heraus abrufen können.
- V 9.1.0 Sie können Namensbereiche verwenden, um die Namensbereiche des Containers für den Warteschlangenmanager mit anderen Containern gemeinsam zu nutzen, um Anwendungen lokal an einen Warteschlangenmanager binden zu können, der in separaten Containern ausgeführt wird. Weitere Informationen finden Sie unter „Lokale Bindungsanwendungen in separaten Containern ausführen" auf Seite 39.

Zugehörige Konzepte

„Unterstützung für zertifizierte IBM MQ-Container" auf Seite 6
Die zertifizierten IBM MQ-Container werden nur in bestimmten Kubernetes-Umgebungen unterstützt

Linux MQ Adv. CD V 9.1.5 Speicheraspekte für IBM MQ Advanced certified container

IBM MQ Advanced certified container wird in zwei Speichermodi ausgeführt:

- **Ephemerer Speicher** wird verwendet, wenn der gesamte Containerstatus bei einem Neustart des Containers verworfen werden kann. 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. Bei Verwendung des ephemeren Speichers geht der Warteschlangenmanagerstatus beim Neustart in vollem Umfang 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. Ebenso wie der gesamte Nachrichtenübertragungsstatus wird auch die Warteschlangenmanagerkonfiguration beim Neustart verworfen. 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 OpenShift-CLI erstellen" auf Seite 22). Geschieht dies nicht, muss IBM MQ bei jedem Neustart des Containers konfiguriert werden.

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

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

Persistenter Speicher

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

- spec.queueManager.availability steuert den Verfügbarkeitsmodus. Bei Verwendung von `SingleInstance` ist nur `ReadWriteOnce`-Speicher erforderlich, während für `multiInstance` eine Speicherklasse erforderlich ist, die `ReadWriteMany` mit den richtigen Dateisperrungsmerkmalen unterstützt. IBM MQ stellt ein Support Statement und ein Testing Statement bereit. Der Verfügbarkeitsmodus wirkt sich auch auf das Layout des persistenten Datenträgers aus. Weitere Informationen finden Sie im Abschnitt „Hohe Verfügbarkeit für IBM MQ Advanced certified container" auf Seite 10.
- spec.queueManager.storage steuert die individuellen Speichereinstellungen. Ein Warteschlangenmanager kann für die Verwendung von bis zu vier persistenten Datenträgern konfiguriert werden.

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

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

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

```
spec:
  queueManager:
    availability:
      type: MultiInstance
    storage:
      queueManager:
        enabled: true
        class: ibmc-file-gold-gid
          persistedData:
            enabled: true
            class: ibmc-file-gold-gid
          recoveryLogs:
            enabled: true
            class: ibmc-file-gold-gid
     securityContext:
       supplementalGroups: [99]
```

Linux MQ Adv. CD Hohe Verfügbarkeit für IBM MQ Advanced certified container

Es gibt zwei Hauptauswahlmöglichkeiten für Hochverfügbarkeit mit IBM MQ Advanced certified container: **Multi-Instanz-Warteschlangenmanager** (ein Aktiv-Standby-Paar, das ein gemeinsames, vernetztes Dateisystem verwendet) und **Einzelner ausfallsicherer Warteschlangenmanager** (der einen einfachen Ansatz für Hochverfügbarkeit mit vernetztem Speicher bietet).

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.

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 Container bereit ist. Dies ist im Helm-Diagramm 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 im Helm-Diagramm 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 im Helm-Diagramm 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 der Kubernetes-Masterknoten den Kontakt zu einem Workerknoten verliert, kann er nicht feststellen, ob der Knoten ausgefallen ist oder ob lediglich die Netzverbindung unterbrochen wurde. Deshalb wird Kubernetes in diesem Fall **nicht aktiv**, sondern erst, wenn eins der folgenden Ereignisse eintritt:

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

Zugehörige Konzepte

Hochverfügbarkeitskonfigurationen

Linux MQ Adv. CD Benutzerauthentifizierung und -berechtigung für IBM MQ Advanced certified container

IBM MQ kann für die Verwendung von LDAP-Benutzern und -Gruppen für die Berechtigung konfiguriert werden. Dies ist der empfohlene Ansatz für IBM MQ Advanced certified container.

In einer aus Containern bestehenden Multi-Tenant-Umgebung, z. B. Red Hat OpenShift Container Platform, gelten Integritätsbedingungen für die Sicherheit, um potenzielle Sicherheitsprobleme zu verhindern. In Red Hat OpenShift Container Platform wird beispielsweise in der Standardeinstellung für `SecurityContextConstraints` (`restricted`) eine randomisierte Benutzer-ID verwendet, um Benutzer abzuhalten, die für den Container selbst lokale Benutzer sind. In IBM MQ werden Kennwörter von Benutzern in der Regel mithilfe einer Berechtigungseskalation überprüft, was in Multi-Tenant-Containerumgebungen ebenfalls nicht empfohlen wird. Aus diesen Gründen wird die Verwendung von Benutzern, die in den Betriebssystembibliotheken innerhalb eines aktiven Containers definiert sind, in den zertifizierten Containern von IBM MQ nicht unterstützt.

Sie müssen Ihren Warteschlangenmanager so konfigurieren, dass LDAP für die Benutzerauthentifizierung und -berechtigung verwendet wird. Informationen zur Konfiguration von IBM MQ zu diesem Zweck finden Sie in den Abschnitten Verbindungsauthentifizierung: Benutzerrepositorys und LDAP-Berechtigung.

Linux MQ Adv. CD V 9.1.5 IBM MQ Operator unter OpenShift installieren und deinstallieren

IBM MQ Operator kann mithilfe des OperatorHub unter OpenShift installiert werden.

Vorbereitende Schritte

Prozedur

- „IBM MQ Operator über die OpenShift-CLI installieren” auf Seite 14.
- „IBM MQ Operator über die OpenShift-Webkonsole installieren” auf Seite 13.

Linux MQ Adv. CD V 9.1.5 IBM MQ Operator über die OpenShift-Webkonsole installieren

IBM MQ Operator kann mithilfe des OperatorHub unter OpenShift installiert werden.

Vorbereitende Schritte

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

Vorgehensweise

1. Fügen Sie die IBM Common Services-Operatoren zur Liste der installierbaren Operatoren hinzu.
   a) Klicken Sie auf das Pluszeichen. Das Dialogfenster **Import YAML** (YAML importieren) wird geöffnet.
   b) Fügen Sie folgende Ressourcendefinition in das Dialogfenster ein:

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

   c) Klicken Sie auf **Erstellen**.
2. Fügen Sie die IBM Operatoren zur Liste der installierbaren Operatoren hinzu.
   a) Klicken Sie auf das Pluszeichen. Das Dialogfenster **Import YAML** (YAML importieren) wird geöffnet.
   b) Fügen Sie folgende Ressourcendefinition in das Dialogfenster ein:

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

   c) Klicken Sie auf **Erstellen**.
3. Erstellen Sie einen Namensbereich, der für IBM MQ Operator verwendet werden soll.
   IBM MQ Operator kann für einen einzelnen Namensbereich oder für alle Namensbereiche installiert werden. Dieser Schritt ist nur erforderlich, wenn Sie eine Installation in einem bestimmten Namensbereich ausführen möchten, der noch nicht vorhanden ist.
   a) Klicken Sie im Navigationsfenster auf **Home** > **Projects**.
      Die Seite 'Projects' wird angezeigt.
   b) Klicken Sie auf **Create Project** (Projekt erstellen). Es wird ein Bereich zum Erstellen des Projekts angezeigt.

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

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

4. Installieren Sie den IBM MQ-Operator.

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

Die Seite 'OperatorHub' wird angezeigt.

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

Der IBM MQ-Katalogeintrag wird angezeigt.

c) Wählen Sie **IBM MQ** aus.

Das Fenster 'IBM MQ' wird angezeigt.

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

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

e) Geben Sie als Installationsmodus entweder den Namensbereich, den Sie erstellt haben, oder den clusterweiten Bereich an.

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

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

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

Nächste Schritte

„Zertifizierte IBM MQ -Container bereitstellen” auf Seite 16

Linux MQ Adv. CD V 9.1.5 IBM MQ Operator über die OpenShift-CLI installieren

IBM MQ Operator kann mithilfe des OperatorHub unter OpenShift installiert werden.

Vorbereitende Schritte

Melden Sie sich mit **oc login** bei der OpenShift-Befehlszeilenschnittstelle (Command Line Interface, CLI) an. Um diese Schritte ausführen zu können, müssen Sie ein Clusteradministrator sein.

Vorgehensweise

1. Erstellen Sie eine `OperatorSource` (Operatorquelle) für die IBM Common Services-Operatoren.

a) Erstellen Sie eine YAML-Datei, die die `OperatorSource`-Ressource definiert

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

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

b) Wenden Sie `OperatorSource`auf den Server an.

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

2. Erstellen Sie eine OperatorSource (Operatorquelle) für die IBM Operatoren.
   a) Erstellen Sie eine YAML-Datei, die die OperatorSource-Ressource definiert

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

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

   b) Wenden Sie OperatorSourceauf den Server an.

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

3. Erstellen Sie einen Namensbereich, der für IBM MQ Operator verwendet werden soll.

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

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

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

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

5. Untersuchen Sie IBM MQ Operator, um seine unterstützten Installationsmodi und verfügbaren Kanäle zu überprüfen.

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

6. YAML-Datei für OperatorGroup-Objekt erstellen

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

   Der Namensbereich, den Sie für den Operator subskribieren, muss über eine OperatorGroup verfügen, die mit dem InstallMode (Installationsmodus) des Operators übereinstimmt, also entweder dem Modus AllNamespaces (Alle Namensbereiche) oder dem Modus SingleNamespace (Einzelner Namensbereich). Wenn der Operator, den Sie installieren möchten, den Modus AllNamespaces verwendet, verfügt der Namensbereich openshift-operators bereits über eine geeignete Operatorgruppe.

   Wenn der Operator jedoch den Modus SingleNamespace verwendet und noch keine geeignete OperatorGroup vorhanden ist, müssen Sie eine erstellen.

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

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

   b) Erstellen Sie das OperatorGroup-Objekt

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

7. Erstellen Sie eine YAML-Datei für das Subskriptionsobjekt, um einen Namensbereich für MQ Operator zu subskribieren

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

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

   Geben Sie für die Verwendung von `AllNamespaces` **`InstallMode`** den Namensbereich `openshift-operators` an. Andernfalls geben Sie den relevanten einzelnen Namensbereich für die Verwendung von `Einzelner Namensbereich`**`InstallMode`** an.

   b) Erstellen Sie das `Subscription`-Objekt

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

8. Überprüfen Sie den Status des Operators

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

Nächste Schritte

„Zertifizierte IBM MQ -Container bereitstellen” auf Seite 16

Linux MQ Adv. CD Zertifizierte IBM MQ -Container bereitstellen

IBM MQ Version 9.1.5 und höher kann in Red Hat OpenShift mit IBM MQ Operator bereitgestellt werden. Die IBM MQ Versionen 9.1.5 und 9.1.4 können in Red Hat OpenShift mithilfe von Helmbereitgestellt werden. Ältere CD-Versionen können mit Helmin einem IBM Cloud Private -Cluster oder einem IBM Cloud Kubernetes Service -Cluster bereitgestellt werden.

Informationen zu diesem Vorgang

Prozedur

- „Warteschlangenmanager über die Helm -CLI implementieren” auf Seite 25.
- „Frühere CD-Releases von IBM MQ in einem IBM Cloud Private-Cluster bereitstellen” auf Seite 28.
- „Frühere CD-Releases eines IBM MQ-Images zu einem IBM Cloud Private-Cluster hinzufügen” auf Seite 30.
- „Frühere CD-Releases eines IBM MQ-Images zu einem IBM Cloud Kubernetes Service-Cluster hinzufügen” auf Seite 30.

Linux MQ Adv. CD OpenShift-Projekt für IBM MQ über die OpenShift-CLI vorbereiten

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

Vorbereitende Schritte

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

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

Informationen zu diesem Vorgang

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

Vorgehensweise

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

   Führen Sie den folgenden Befehl aus, wobei *<entitlement-key>* der in Schritt 1 abgerufene Schlüssel ist und *<user-email>* die IBM ID, die der berechtigten Software zugeordnet ist.

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

Nächste Schritte

„Warteschlangenmanager über die OpenShift-CLI implementieren” auf Seite 19

Linux MQ Adv. CD V 9.1.5 Warteschlangenmanager mit dem IBM Cloud Pak for Integration Platform Navigator implementieren

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

Vorbereitende Schritte

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

Wenn dies das erste Mal ist, dass ein Warteschlangenmanager in diesem Red Hat OpenShift-Projekt implementiert wird, führen Sie die Schritte im Abschnitt „OpenShift-Projekt für IBM MQ über die OpenShift-CLI vorbereiten” auf Seite 16 aus.

Vorgehensweise

1. Implementieren Sie einen Warteschlangenmanager.

   Im folgenden Beispiel wird ein "Schnellstart"-Warteschlangenmanager, der einen ephemeren (nicht persistenten) Speicher verwendet, implementiert und die MQ-Sicherheit inaktiviert. Nachrichten blei-

ben bei Neustarts des Warteschlangenmanagers nicht erhalten. Sie können die Konfiguration anpassen, um viele Warteschlangenmanagereinstellungen zu ändern.

a) Klicken Sie in IBM Cloud Pak for Integration Platform Navigator auf **Runtime and instances** (Laufzeit und Instanzen).

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

c) Wählen Sie **Queue Manager** (Warteschlangenmanager) aus und klicken Sie auf **Next** (Weiter).

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

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

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

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

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

f) Überprüfen oder aktualisieren Sie im Abschnitt **Queue Manager Config** (Warteschlangenmanagerkonfiguration) im Feld **Name** den Namen des zugrunde liegenden Warteschlangenmanagers.

Standardmäßig entspricht der Name des von IBM MQ-Clientanwendungen verwendeten Warteschlangenmanagers dem Namen des `QueueManager`, wobei jedoch alle ungültigen Zeichen (z. B. Bindestriche) entfernt werden. Falls Sie die Verwendung eines bestimmten Namens erzwingen möchten, können Sie hier die entsprechende Bearbeitung durchführen.

g) Klicken Sie auf **Erstellen**

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

2. Überprüfen Sie, ob der Warteschlangenmanager aktiv ist.

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

Zugehörige Tasks

„Verbindung zu einem Warteschlangenmanager herstellen, der in einem OpenShift-Cluster implementiert ist" auf Seite 31
Konfigurationsbeispiele für die Herstellung einer Verbindung zu einem Warteschlangenmanager, der in einem Red Hat OpenShift-Cluster implementiert ist.

„Verbindung zur in einem OpenShift-Cluster implementierten IBM MQ Console herstellen" auf Seite 33
Sie können eine Verbindung zur IBM MQ Console eines Warteschlangenmanagers herstellen, die in einem Red Hat OpenShift Container Platform-Cluster implementiert wurde.

Warteschlangenmanager über die OpenShift-Webkonsole implementieren

Verwenden Sie die angepasste QueueManager-Ressource, um einen Warteschlangenmanager über die Red Hat OpenShift-Webkonsole in einem Red Hat OpenShift Container Platform-Cluster zu implementieren. Diese Aufgabe sollte von einem Projektadministrator ausgeführt werden

Vorbereitende Schritte

Melden Sie sich bei der Webkonsole Ihres OpenShift-Clusters an. Sie müssen ein vorhandenes Projekt (Namensbereich), das Sie verwenden möchten, auswählen oder ein neues erstellen.

Wenn dies das erste Mal ist, dass ein Warteschlangenmanager in diesem Red Hat OpenShift-Projekt implementiert wird, führen Sie die Schritte im Abschnitt „OpenShift-Projekt für IBM MQ über die OpenShift-CLI vorbereiten" auf Seite 16 aus.

Vorgehensweise

1. Implementieren Sie einen Warteschlangenmanager.

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

a) Klicken Sie in der OpenShift-Webkonsole im Navigationsfenster auf **Operators** > **Installed Operators** (Operatoren > Installierte Operatoren).

b) Klicken Sie auf **IBM MQ**.

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

d) Klicken Sie auf die Schaltfläche **Create QueueManager** (QueueManager erstellen).

Es wird ein YAML-Editor mit einer YAML-Beispieldatei für eine `QueueManager`-Ressource angezeigt.

Anmerkung: Sie können auch auf **Edit Form** (Formular bearbeiten) klicken, um die `QueueManager`-Konfiguration anzuzeigen oder zu ändern.

e) Wenn Sie die Lizenzvereinbarung akzeptieren, ändern Sie die Einstellung von **License acceptance** (Lizenzakzeptanz) in **On**.

IBM MQ ist unter mehreren verschiedenen Lizenzen verfügbar. Weitere Informationen zu den gültigen Lizenzen finden Sie im Abschnitt „Lizenzierungsreferenz für mq.ibm.com/v1beta1" auf Seite 41. Um einen Warteschlangenmanager implementieren zu können, müssen Sie die Lizenz akzeptieren.

f) Klicken Sie auf **Erstellen**

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

2. Überprüfen Sie, ob der Warteschlangenmanager aktiv ist.

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

Zugehörige Tasks

„Verbindung zu einem Warteschlangenmanager herstellen, der in einem OpenShift-Cluster implementiert ist" auf Seite 31
Konfigurationsbeispiele für die Herstellung einer Verbindung zu einem Warteschlangenmanager, der in einem Red Hat OpenShift-Cluster implementiert ist.

„Verbindung zur in einem OpenShift-Cluster implementierten IBM MQ Console herstellen" auf Seite 33
Sie können eine Verbindung zur IBM MQ Console eines Warteschlangenmanagers herstellen, die in einem Red Hat OpenShift Container Platform-Cluster implementiert wurde.

Linux MQ Adv. CD V 9.1.5 Warteschlangenmanager über die OpenShift-CLI implementieren

Verwenden Sie die angepasste QueueManager-Ressource, um einen Warteschlangenmanager über die Befehlszeilenschnittstelle (Command Line Interface, CLI) in einem Red Hat OpenShift Container Platform-Cluster zu implementieren. Diese Aufgabe sollte von einem Projektadministrator ausgeführt werden

Vorbereitende Schritte

Sie müssen die Red Hat OpenShift Container Platform-Befehlszeilenschnittstelle (CLI) installieren.

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

Wenn dies das erste Mal ist, dass ein Warteschlangenmanager in diesem Red Hat OpenShift-Projekt implementiert wird, führen Sie die Schritte im Abschnitt „OpenShift-Projekt für IBM MQ über die OpenShift-CLI vorbereiten" auf Seite 16 aus.

Vorgehensweise

1. Implementieren Sie einen Warteschlangenmanager.

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

   a) `QueueManager`-YAML-Datei erstellen

      Um beispielsweise einen Basiswarteschlangenmanager in IBM Cloud Pak for Integration zu installieren, erstellen Sie die Datei 'mq-quickstart.yaml' mit folgendem Inhalt:

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

      Wichtig: Wenn Sie die Lizenzvereinbarung für IBM Cloud Pak for Integration akzeptieren, ändern Sie `accept: false` in `accept: true`. Details zur Lizenz finden Sie im Abschnitt „Lizenzierungsreferenz für mq.ibm.com/v1beta1” auf Seite 41.

      Dieses Beispiel schließt auch einen mit dem Warteschlangenmanager bereitgestellten Web-Server ein, wobei die Webkonsole mit Single Sign-On mit Cloud Pak Identity and Access Manager aktiviert ist.

      Um einen Basiswarteschlangenmanager unabhängig von IBM Cloud Pak for Integration zu installieren, erstellen Sie die Datei 'mq-quickstart.yaml' mit folgendem Inhalt:

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

Wichtig: Wenn Sie die MQ-Lizenzvereinbarung akzeptieren, ändern Sie `accept: false` in `accept: true`. Details zur Lizenz finden Sie im Abschnitt „Lizenzierungsreferenz für mq.ibm.com/v1beta1" auf Seite 41.

b) Erstellen Sie das `QueueManager`-Objekt

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

2. Überprüfen Sie, ob der Warteschlangenmanager aktiv ist.

   Sie können die Implementierung überprüfen, durch die Ausführung von

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

   , und Sie darauffolgend den Status überprüfen.

   Führen Sie beispielsweise

   ```
   oc describe queuemanager quickstart
   ```

   und prüfen Sie, ob das Feld `status.Phase`den Wert `Running`enthält

Zugehörige Tasks

„Verbindung zu einem Warteschlangenmanager herstellen, der in einem OpenShift-Cluster implementiert ist" auf Seite 31
Konfigurationsbeispiele für die Herstellung einer Verbindung zu einem Warteschlangenmanager, der in einem Red Hat OpenShift-Cluster implementiert ist.

„Verbindung zur in einem OpenShift-Cluster implementierten IBM MQ Console herstellen" auf Seite 33
Sie können eine Verbindung zur IBM MQ Console eines Warteschlangenmanagers herstellen, die in einem Red Hat OpenShift Container Platform-Cluster implementiert wurde.

V 9.1.4 Linux MQ Adv. CD Integration in das IBM Cloud Pak for Integration-Dashboard 'Operations'

Die Fähigkeit, Transaktionen über IBM Cloud Pak for Integration zu verfolgen, wird durch das Operations-Dashboard bereitgestellt.

Informationen zu diesem Vorgang

Bei der Aktivierung der Integration mit dem Dashboard 'Operations' wird ein MQ-API-Exit für Ihren Warteschlangenmanager installiert. Der API-Exit sendet Tracedaten zu Nachrichten, die über den Warteschlangenmanager übertragen werden, an den Datenspeicher des Dashboards 'Operations'.

Beachten Sie, dass nur für Nachrichten, die mit MQ-Clientbindungen gesendet werden, ein Trace erstellt wird.

Vorgehensweise

1. Implementieren Sie einen Warteschlangenmanager mit aktiviertem Tracing.

   Das Tracing-Feature ist standardmäßig inaktiviert.

   Wenn Sie die Implementierung mithilfe von IBM Cloud Pak for Integration Platform Navigator durchführen, können Sie das Tracing während der Implementierung aktivieren, indem Sie **Enable Tracing** (Tracing aktivieren) auf **On** setzen und für **Tracing Namespace** (Tracing-Namensbereich) den Namensbereich angeben, in dem das Dashboard 'Operations' installiert ist. Weitere Informationen zur Implementierung eines Warteschlangenmanagers finden Sie im Abschnitt „Warteschlangenmanager mit dem IBM Cloud Pak for Integration Platform Navigator implementieren" auf Seite 17 .

Wenn Sie die Implementierung über die OpenShift-CLI oder OpenShift-Webkonsole durchführen, können Sie das Tracing mit folgendem YAML-Snippet aktivieren:

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

Wenn Sie die Bereitstellung mit Helmdurchführen, können Sie die Tracefunktion aktivieren, indem Sie `odTracingConfig.enabled=true` und `odTracingConfig.odTracingNamespace=`*`<Operations_Dashboard_Namespace`*festlegen. Wenn Sie die Integration des Dashboards 'Operations' in einem vorhandenen Warteschlangenmanager aktivieren möchten, können Sie diese Einstellung während der Aktualisierung des Helm-Release anwenden.

Wichtig: Der Warteschlangenmanager startet erst, wenn MQ im Dashboard 'Operations' registriert wurde (siehe nächster Schritt).

Wenn diese Funktion aktiviert ist, müssen Sie beachten, dass zusätzlich zum Container des Warteschlangenmanagers zwei Sidecar-Container ('Agent' und 'Collector') ausgeführt werden. Die Images für diese Sidecar-Container sind in der gleichen Registry wie das MQ-Hauptimage verfügbar und verwenden die gleichen Richtlinien und geheimen Schlüssel für Pull-Operationen. Es sind weitere Einstellungen zur Konfiguration der CPU und der Speicherbegrenzung verfügbar.

2. Wenn ein Warteschlangenmanager mit Operations Dashboard-Integration zum ersten Mal in diesem Namensbereich bereitgestellt wurde, müssen Sie Registrieren im Operations Dashboard registrieren.

   Beim Registrieren wird ein Objekt für einen geheimen Schlüssel erstellt, den der Warteschlangenmanager für einen erfolgreichen Start benötigt.

Linux MQ Adv. CD V 9.1.5 Image mit benutzerdefinierten MQSC- und INI-Dateien über die OpenShift-CLI erstellen

Erstellen Sie mithilfe einer Red Hat OpenShift Container Platform-Pipeline ein neues IBM MQ-Container-Image mit MQSC-und INI-Dateien, die mit diesem Image auf Warteschlangenmanager angewendet werden sollen. Diese Aufgabe sollte von einem Projektadministrator ausgeführt werden

Vorbereitende Schritte

Sie müssen die Red Hat OpenShift Container Platform-Befehlszeilenschnittstelle (CLI) installieren.

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

Wenn Sie kein OpenShift-Secret für die IBM Entitled Registry in Ihrem Red Hat OpenShift-Projekt besitzen, führen Sie die Schritte im Abschnitt „OpenShift-Projekt für IBM MQ über die OpenShift-CLI vorbereiten” auf Seite 16 aus.

Vorgehensweise

1. Erstellen:`ImageStream`

   Ein ImageStream und die ihm zugeordneten Tags stellen eine Abstraktion für die Referenzierung von Container-Images aus Red Hat OpenShift Container Platform heraus bereit. Mithilfe des ImageStream und der zugehörigen Tags können Sie sehen, welche Images verfügbar sind, und sicherstellen, dass Sie genau das Image verwenden, das Sie benötigen, selbst wenn sich das Image im Repository ändert.

   ```
   oc create imagestream mymq
   ```

2. `BuildConfig` für Ihr neues Image erstellen

   Ein `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:9.1.5.0-r2-amd64
      RUN printf "DEFINE QLOCAL(foo) REPLACE\n" > /etc/mqm/my.mqsc \
        && printf "Channels:\n\tMQIBindType=FASTPATH\n" > /etc/mqm/my.ini
      LABEL summary "My custom MQ image"
  strategy:
    type: Docker
    dockerStrategy:
      from:
        kind: "DockerImage"
        name: "cp.icr.io/cp/ibm-mqadvanced-server-integration:9.1.5.0-r2-amd64"
      pullSecret:
        name: ibm-entitlement-key
  output:
    to:
      kind: ImageStreamTag
      name: 'mymq:latest-amd64'
```

Sie müssen die zwei Stellen, an denen das IBM MQ-Basisimage genannt wird, ändern, sodass auf das richtige Basisimage für die Version und den Fix verwiesen wird, die Sie verwenden möchten. Wenn Fixes angewendet werden, müssen Sie diese Schritte wiederholen, um Ihr Image erneut zu erstellen.

In diesem Beispiel wird ein neues Image auf Basis des offiziellen IBM Image erstellt und es werden Dateien mit den Namen "my.mqsc" und "my.ini" im Verzeichnis `/etc/mqm` hinzugefügt. Alle MQSC- oder INI-Dateien, die in diesem Verzeichnis gefunden werden, werden beim Start vom Container ausgeführt. INI-Dateien werden mit der Option **`crtmqm -ii`** ausgeführt und mit den vorhandenen INI-Dateien zusammengeführt. MQSC-Dateien werden in alphabetischer Reihenfolge ausgeführt.

Es ist wichtig, dass Ihre MQSC-Befehle wiederholt anwendbar sind, da sie bei *jedem* Start des Warteschlangenmanagers ausgeführt werden. Dies bedeutet in der Regel, dass der Parameter `REPLACE`in allen `DEFINE`-Befehlen und der Parameter `IGNSTATE(YES)` in allen `START`- oder `STOP`-Befehlen hinzugefügt wird.

b) Wenden Sie `BuildConfig`auf den Server an.

```
oc apply -f mq-build-config.yaml
```

3. Führen Sie einen Build aus, um Ihr Image zu erstellen.

a) Starten Sie den Build.

```
oc start-build mymq
```

Es sollte eine Ausgabe wie die folgende angezeigt werden:

```
build.build.openshift.io/mymq-1 started
```

b) Überprüfen Sie den Status des Builds.
Sie können beispielsweise folgenden Befehl unter Angabe der im vorherigen Schritt zurückgegebenen Build-ID ausführen:

```
oc describe build mymq-1
```

4. Implementieren Sie einen Warteschlangenmanager mit dem neuen Image.

Führen Sie die Schritte im Abschnitt „Warteschlangenmanager über die OpenShift-CLI implementieren” auf Seite 19 aus und fügen Sie dabei das neue benutzerdefinierte Image in der YAML-Datei hinzu.

Sie könnten folgendes YAML-Snippet in Ihrer normalen `QueueManager`-YAML-Datei hinzufügen, wobei *my-namespace* für das OpenShift-Projekt oder den OpenShift-Namensbereich, das bzw. den Sie

verwenden, und *my-image* für den Namen des zuvor erstellten Image (z. B. "mymq:latest-amd64") stehen:

```
spec:
  queueManager:
    image: image-registry.openshift-image-registry.svc:5000/my-namespace/my-image
```

Zugehörige Tasks

„Warteschlangenmanager über die OpenShift-CLI implementieren” auf Seite 19
Verwenden Sie die angepasste QueueManager-Ressource, um einen Warteschlangenmanager über die Befehlszeilenschnittstelle (Command Line Interface, CLI) in einem Red Hat OpenShift Container Platform-Cluster zu implementieren. Diese Aufgabe sollte von einem Projektadministrator ausgeführt werden

Zertifizierte IBM MQ -Container mithilfe von Helm implementieren

Ab IBM MQ 9.1.5.0wird empfohlen, einen Warteschlangenmanager mit dem Operator IBM MQ zu implementieren. IBM MQ 9.1.5.0 und frühere CD-Releases können mithilfe von Helmunter Verwendung der folgenden Anweisungen bereitgestellt werden.

Informationen zu diesem Vorgang

Prozedur

- „OpenShift -Cluster für IBM MQ unter OpenShift mithilfe von Helm” auf Seite 24.
- „Warteschlangenmanager über die Helm -CLI implementieren” auf Seite 25.

OpenShift -Cluster für IBM MQ unter OpenShift mithilfe von Helm

Bereiten Sie Ihren Red Hat OpenShift Container Platform-Cluster vor, sodass er bereit ist, einen Warteschlangenmanager mithilfe von Helmbereitzustellen. Diese Task sollte von einem Clusteradministrator ausgeführt werden.

Vorbereitende Schritte

Anmerkung: Wenn Sie IBM Cloud Pak for Integrationverwenden, sollte das Installationsprogramm ein OpenShift -Projekt (Namensbereich) für die Verwendung mit IBM MQvorbereitet haben, sodass Sie diese Anweisungen möglicherweise nicht befolgen müssen.

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

Vorgehensweise

1. Stellen Sie sicher, dass Sie das IBM Helm-Repository zu Ihrer lokalen Kopie von Helm hinzugefügt haben.
   Führen Sie beispielsweise den folgenden Befehl aus:

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

2. Stellen Sie sicher, dass ein Helm-Server (mit der Bezeichnung "Tiller") in Ihren Cluster installiert ist.
   Befolgen Sie die Anweisungen im Abschnitt Erste Schritte mit Helm auf OpenShift, um Helm in Ihrem Cluster zu installieren.
3. Stellen Sie sicher, dass Servicekonten in Ihrem OpenShift-Projekt (Namensbereich) für die Verwendung der korrekten Sicherheitskontexteinschränkungen (Security Context Constraints, SCCs) berechtigt sind.

V 9.1.5 IBM MQ funktioniert mit der standardmäßigen SCC "restricted" (eingeschränkt), daher kann dieser Schritt normalerweise übersprungen werden.

Änderungen an SCCs müssen von einem OpenShift-Clusteradministrator angewendet werden. Jede Version eines Helm-Diagramms hat unterschiedliche Anforderungen an SCCs, die in der jeweiligen Readme-Datei zu diesem Helm-Diagramm dokumentiert sind:

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

In jeder Readme-Datei finden Sie Anweisungen zur Einrichtung der Autorisierung für SCCs. Beachten Sie, dass die Helm-Diagramme für IBM MQ ein Servicekonto zur eigenen Verwendung erstellen. Die SCC-Berechtigungen müssen also auf der Ebene "group" (Gruppe) angewendet werden (für alle Servicekonten m Namensbereich).

4. Stellen Sie sicher, dass Sie über einen gültigen geheimen Schlüssel für Pull-Operationen für das Image verfügen, um Images aus Ihrer angegebenen Container-Registry zu extrahieren.

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

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

      i) Melden Sie sich an der MyIBM Container Software Library mit der IBM-ID und dem Kennwort an, die der berechtigten Software zugeordnet sind.

      ii) Wählen Sie im Abschnitt *Entitlement keys* (Berechtigungsschlüssel) die Option **Copy key** (Schlüssel kopieren) aus, um den Berechtigungsschlüssel in die Zwischenablage zu kopieren.

   b) Erstellen Sie den geheimen Schlüssel in dem Namespace, in dem Sie Ihren Warteschlangenmanager bereitstellen möchten.

   - Führen Sie den folgenden Befehl aus, wobei *<entitlement-key>* der in Schritt 1 abgerufene Schlüssel ist und *<user-email>* die IBM ID, die der berechtigten Software zugeordnet ist.

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

Nächste Schritte

„Warteschlangenmanager über die Helm -CLI implementieren” auf Seite 25

V 9.1.4 Linux MQ Adv. CD Warteschlangenmanager über die Helm -CLI implementieren

Verwenden Sie Helm, um einen Warteschlangenmanager in einem Red Hat OpenShift Container Platform-Cluster bereitzustellen. Diese Aufgabe sollte von einem Projektadministrator ausgeführt werden.

Vorbereitende Schritte

Sie müssen Helm V2 und die Befehlszeilenschnittstelle für Red Hat OpenShift Container Platform installieren. Wenn Sie IBM Cloud Pak for Integration nicht verwenden, führen Sie die Schritte aus, die im Abschnitt „OpenShift -Cluster für IBM MQ unter OpenShift mithilfe von Helm” auf Seite 24 beschrieben sind.

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

Vorgehensweise

1. Stellen Sie sicher, dass Sie das IBM Helm-Repository zu Ihrer lokalen Kopie von Helm hinzugefügt haben.
   Führen Sie beispielsweise den folgenden Befehl aus:

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

2. Überprüfen Sie die Konfigurationsoptionen für Ihren Warteschlangenmanager

   Die Implementierung umfasst Schritte zur Installation und Konfiguration. Einige Einstellungen für Ihren Warteschlangenmanager müssen während der Implementierung estgelegt werden, und Änderungen an den Einstellungen erfordern eine erneute Implementierung.

   Sie können in der Readme-Datei für das Helm-Diagramm Einzelheiten zu allen verfügbaren Implementierungsoptionen anzeigen, indem Sie einen der folgenden Befehle ausführen:

   - Für IBM MQ Advanced certified container in IBM Cloud Pak for Integration:

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

   - Für IBM MQ Advanced certified container:

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

   Sie benötigen typischerweise mindestens einen der folgenden Parameter:

   a. Releasename. Beispiel: `my-release`
   b. Fernes Helm-Repository. Beispiel: `ibm-entitled-charts`
   c. Helm-Diagramm. Beispiel: `ibm-mqadvanced-server-prod` oder `ibm-mqadvanced-server-integration-prod`
   d. Name des geheimen Schlüssels für Image-Pull-Operationen. Beispiel: `entitled-registry`. Beachten Sie, dass dies bei der Implementierung in ein vordefiniertes Projekt für MQ in IBM Cloud Pak for Integration nicht erforderlich ist.

3. Implementieren Sie einen Warteschlangenmanager.

   Beachten Sie, dass das Helm-Diagramm standardmäßig voraussetzt, dass in Ihrem Red Hat OpenShift Container Platform-Cluster eine Standardspeicherklasse (Storage Class) festgelegt ist.

   Wenn Sie beispielsweise einen Basiswarteschlangenmanager in IBM Cloud Pak for Integration installieren möchten, führen Sie den folgenden Befehl aus:

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

   Sie können einen beliebigen Hostnamen im Feld `tls.hostname` eingeben (dieses Feld ist erforderlich, wird in diesem Beispiel zum Generieren eines neuen selbst signierten Zertifikats allerdings nicht verwendet)

   Wenn Sie einen Basiswarteschlangenmanager unabhängig von IBM Cloud Pak for Integration installieren möchten, führen Sie den folgenden Befehl aus:

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

Zugehörige Tasks

„Verbindung zu einem Warteschlangenmanager herstellen, der in einem OpenShift-Cluster implementiert ist" auf Seite 31
Konfigurationsbeispiele für die Herstellung einer Verbindung zu einem Warteschlangenmanager, der in einem Red Hat OpenShift-Cluster implementiert ist.

„Verbindung zur in einem OpenShift-Cluster implementierten IBM MQ Console herstellen" auf Seite 33
Sie können eine Verbindung zur IBM MQ Console eines Warteschlangenmanagers herstellen, die in einem Red Hat OpenShift Container Platform-Cluster implementiert wurde.

Warteschlangenmanager mit IBM Cloud File Storageüber die Helm -CLI bereitstellen

Beispielszenario zur Verwendung von Helm für die Bereitstellung eines Warteschlangenmanagers in einem Red Hat OpenShift on IBM Cloud -Cluster unter Verwendung von IBM Cloud File Storage. Diese Aufgabe sollte von einem Projektadministrator ausgeführt werden

Vorbereitende Schritte

Sie müssen Helm V2 und die Befehlszeilenschnittstelle für Red Hat OpenShift Container Platform installieren. Wenn Sie IBM Cloud Pak for Integration nicht verwenden, führen Sie die Schritte aus, die im Abschnitt „OpenShift -Cluster für IBM MQ unter OpenShift mithilfe von Helm" auf Seite 24 beschrieben sind.

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

Vorgehensweise

1. Stellen Sie sicher, dass Sie das IBM Helm -Repository Ihrer lokalen Kopie von Helmhinzugefügt haben. Führen Sie beispielsweise den folgenden Befehl aus:

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

2. Implementieren Sie einen Warteschlangenmanager.

   Bei Verwendung von IBM Cloud File Storagewerden normalerweise die besten Ergebnisse bei Verwendung der Speicherklasse `ibmc-file-gold-gid` angezeigt. Diese Speicherklasse aktiviert Speicher, in den Benutzer in der richtigen Dateisystemgruppe schreiben können.

   Wenn Sie beispielsweise einen Basiswarteschlangenmanager in IBM Cloud Pak for Integration installieren möchten, führen Sie den folgenden Befehl aus:

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

   Sie können einen beliebigen Hostnamen in das Feld `tls.hostname` eingeben (dies ist ein erforderliches Feld, wird hier jedoch nicht verwendet, da in diesem Beispiel ein neues selbst signiertes Zertifikat generiert wird).

   Wenn Sie einen Basiswarteschlangenmanager unabhängig von IBM Cloud Pak for Integration installieren möchten, führen Sie den folgenden Befehl aus:

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

```
--set dataPVC.storageClassName=ibmc-file-gold-gid \
--set security.context.supplementalGroups={99}
```

Zugehörige Tasks

„Verbindung zu einem Warteschlangenmanager herstellen, der in einem OpenShift-Cluster implementiert ist" auf Seite 31
Konfigurationsbeispiele für die Herstellung einer Verbindung zu einem Warteschlangenmanager, der in einem Red Hat OpenShift-Cluster implementiert ist.

„Verbindung zur in einem OpenShift-Cluster implementierten IBM MQ Console herstellen" auf Seite 33
Sie können eine Verbindung zur IBM MQ Console eines Warteschlangenmanagers herstellen, die in einem Red Hat OpenShift Container Platform-Cluster implementiert wurde.

Linux MQ Adv. CD Frühere CD-Releases von IBM MQ in einem IBM Cloud Private-Cluster bereitstellen

Für CD-Versionen von IBM MQ vor 9.1.4verwenden Sie die IBM Cloud Private -Managementkonsole, um einen Warteschlangenmanager in IBM Cloud Privatezu implementieren.

Vorbereitende Schritte

Achtung: V 9.1.4 Diese Bereitstellung wird ab IBM MQ 9.1.4 nicht unterstützt.

Diese Aufgabe setzt voraus, dass Sie bereits ein IBM MQ-Image zu einem IBM Cloud Private-Cluster hinzugefügt haben.

Die Helm-Diagrammdatei `README.md` ist aus dem IBM Cloud Private-Katalogeintrag verfügbar, der angezeigt wird, nachdem Sie diesen Unterschritt ausgeführt haben oder indem Sie über die Befehlszeile Ihr **`local-charts`**-Repository von IBM Cloud Private als fernes Helm-Repository hinzufügen und folgenden Befehl ausführen:

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

Sie müssen über eine PodSecurityPolicy oder eine SecurityContextConstraint (für IBM Cloud Private on Red Hat OpenShift) verfügen, die den notwendigen Sicherheitskontext unterstützt. Details, einschließlich Beispielen, finden Sie in der Helm-Diagrammdatei `README.md`.

Details zur Konfiguration eines Helm-Release finden Sie ebenfalls in der Helm-Diagrammdatei `README.md`.

Anmerkung:

- Wenn Sie eine Bereitstellung in einer IBM Cloud Private-Umgebung durchführen, die die erforderlichen Sicherheitseinstellungen standardmäßig nicht unterstützt, dann aktivieren Sie Ihre Bereitstellung, indem Sie den Anweisungen im Abschnitt Deploying Helm charts that require elevated privileges in a non-default namespace in der IBM Cloud Private-Produktdokumentation folgen.
- Wenn Sie SELinux verwenden, müssen Sie die IBM MQ-Voraussetzungen erfüllen, die in IBM MQ support for SELinux on Red Hat Enterprise Linux beschrieben sind.

Informationen zu diesem Vorgang

IBM Cloud Private bietet eine Plattform für die Verwaltung containerisierter On-Premises-Anwendungen. Nachdem Sie ein IBM MQ-Image zu einem IBM Cloud Private-Cluster hinzugefügt haben, können Sie entweder über die IBM Cloud Private-Managementkonsole oder die Befehlszeile einen Warteschlangenmanager bereitstellen.

Prozedur

- IBM Cloud Private-Managementkonsole verwenden

a) Öffnen Sie die IBM Cloud PrivateManagementkonsole in einem Web-Browser und klicken Sie auf **Katalog**.

Weitere Informationen finden Sie unter Accessing your IBM Cloud Private cluster by using the management console (Über die Managementkonsole auf IBM Cloud Private-Cluster zugreifen) in der IBM Cloud Private-Produktdokumentation.

b) Wählen Sie das Diagramm `ibm-mqadvanced-server-prod` in der Liste aus.

c) Wählen Sie **Konfigurieren** und führen Sie dann die folgenden Konfigurationsschritte aus:

a. Geben Sie einen Releasenamen ein.

b. Lesen und akzeptieren Sie die Lizenzvereinbarungen.

c. Geben Sie unter dem Abschnitt **dataPVC** für **storageclass** die gewünschte Speicherklasse an. Machen Sie keine Angabe, wenn die Standardspeicherklasse verwendet werden soll.

d. Geben Sie unter dem Abschnitt **image** den vollständigen Imagepfad für das Repository an. Beispiel:

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

e. Geben Sie unter dem Abschnitt **image** den Image-Tag an. Beispiel:

```
9.1.3.0-r1
```

f. Wenn Sie einen geheimen Schlüssel für Pull-Operation für Kubernetes für den Zugriff auf die Image-Registry benötigen, fügen Sie **`pullSecret`** hinzu.

g. Geben Sie unter dem Abschnitt **queueManager** den Namen des Warteschlangenmanagers an.

d) Klicken Sie auf **Installieren**, um Ihren Warteschlangenmanager als *Helm-Release* bereitzustellen.

- Verwenden der Befehlszeile

a) Konfigurieren Sie **`cloudctl`** für den Zugriff auf Ihren IBM Cloud Private-Cluster.

Weitere Informationen finden Sie unter IBM Cloud Private-CLI installieren in der Produktdokumentation zu IBM Cloud Private.

b) Stellen Sie sicher, dass Sie Ihr **`local-charts`**-Repository von IBM Cloud Private als fernes Helm-Repository hinzugefügt haben.

c) Installieren Sie das Diagramm.

Führen Sie folgenden Befehl aus und geben Sie dabei die folgenden Parameter an:

a. Releasename (z. B. `my-release`)

b. Name des fernen Helm-Repositorys, das das Diagramm `ibm-mqadvanced-server-prod` enthält (z. B. `my-repo`)

c. Image-Repository (z. B. `mycluster.icp:8500/`*`namespace_name`*`/ibm-mqadvanced-server-prod`)

d. Image-Tag (z. B. `9.1.3.0-r1`)

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

Zugehörige Tasks

„Warteschlangenmanager über die Helm -CLI implementieren" auf Seite 25
Verwenden Sie Helm, um einen Warteschlangenmanager in einem Red Hat OpenShift Container Platform-Cluster bereitzustellen. Diese Aufgabe sollte von einem Projektadministrator ausgeführt werden.

„Frühere CD-Releases eines IBM MQ-Images zu einem IBM Cloud Private-Cluster hinzufügen" auf Seite 30
Bereiten Sie für CD-Versionen von IBM MQ vor 9.1.4Ihren IBM Cloud Private -Cluster für die Bereitstellung eines produktionsbereiten Images für IBM MQvor.

„Frühere CD-Releases eines IBM MQ-Images zu einem IBM Cloud Kubernetes Service-Cluster hinzufügen” auf Seite 30
Importieren Sie für CD-Versionen von IBM MQ vor 9.1.4ein produktionsbereites Image für IBM MQ in IBM Cloud Kubernetes Service.

Linux MQ Adv. CD Frühere CD-Releases eines IBM MQ-Images zu einem IBM Cloud Private-Cluster hinzufügen

Bereiten Sie für CD-Versionen von IBM MQ vor 9.1.4Ihren IBM Cloud Private -Cluster für die Bereitstellung eines produktionsbereiten Images für IBM MQvor.

Informationen zu diesem Vorgang

Achtung: V 9.1.4 Dieser Import wird ab IBM MQ 9.1.4 nicht unterstützt.

Sie können ein IBM MQ-Image von Passport Advantage herunterladen und in einen IBM Cloud Private-Container importieren.

Vorgehensweise

1. Laden Sie das neueste IBM MQ-Image von Passport Advantage -und Passport Advantage Express -Website herunter.

   Details zu verfügbaren Downloads finden Sie unter IBM MQ 9.1 herunterladen. Klicken Sie dort auf die Registerkarte für das Release, das Sie herunterladen möchten. Name und Teilenummer der herunterzuladenden Komponente werden in einer Tabelle aufgelistet.
2. Importieren Sie die heruntergeladene Archivdatei in IBM Cloud Private.

   Siehe IBM Software zum IBM Cloud Private-Katalog hinzufügen in der Produktdokumentation von IBM Cloud Private.

Nächste Schritte

Sie können jetzt einen Warteschlangenmanager in IBM Cloud Private implementieren.

Zugehörige Tasks

„Warteschlangenmanager über die Helm -CLI implementieren” auf Seite 25
Verwenden Sie Helm, um einen Warteschlangenmanager in einem Red Hat OpenShift Container Platform-Cluster bereitzustellen. Diese Aufgabe sollte von einem Projektadministrator ausgeführt werden.

„Frühere CD-Releases von IBM MQ in einem IBM Cloud Private-Cluster bereitstellen” auf Seite 28
Für CD-Versionen von IBM MQ vor 9.1.4verwenden Sie die IBM Cloud Private -Managementkonsole, um einen Warteschlangenmanager in IBM Cloud Privatezu implementieren.

„Frühere CD-Releases eines IBM MQ-Images zu einem IBM Cloud Kubernetes Service-Cluster hinzufügen” auf Seite 30
Importieren Sie für CD-Versionen von IBM MQ vor 9.1.4ein produktionsbereites Image für IBM MQ in IBM Cloud Kubernetes Service.

Linux MQ Adv. CD Frühere CD-Releases eines IBM MQ-Images zu einem IBM Cloud Kubernetes Service-Cluster hinzufügen

Importieren Sie für CD-Versionen von IBM MQ vor 9.1.4ein produktionsbereites Image für IBM MQ in IBM Cloud Kubernetes Service.

Informationen zu diesem Vorgang

Achtung: V 9.1.4 Dieser Import wird ab IBM MQ 9.1.4 nicht unterstützt.

Sie können ein IBM MQ-Image von Passport Advantage herunterladen und in einen IBM Cloud Kubernetes Service-Cluster importieren.

Vorgehensweise

1. Laden Sie das neueste IBM MQ-Image von Passport Advantage -und Passport Advantage Express -Website herunter.

   Details zu verfügbaren Downloads finden Sie unter IBM MQ 9.1 herunterladen. Klicken Sie dort auf die Registerkarte für das Release, das Sie herunterladen möchten. Name und Teilenummer der herunterzuladenden Komponente werden in einer Tabelle aufgelistet.
2. Importieren Sie die heruntergeladene Archivdatei in IBM Cloud Kubernetes Service.

   Weitere Informationen finden Sie unter IBM Cloud Private in öffentlichen Kubernetes-Containern ausführen.

Zugehörige Tasks

„Warteschlangenmanager über die Helm -CLI implementieren” auf Seite 25
Verwenden Sie Helm, um einen Warteschlangenmanager in einem Red Hat OpenShift Container Platform-Cluster bereitzustellen. Diese Aufgabe sollte von einem Projektadministrator ausgeführt werden.

„Frühere CD-Releases von IBM MQ in einem IBM Cloud Private-Cluster bereitstellen” auf Seite 28
Für CD-Versionen von IBM MQ vor 9.1.4verwenden Sie die IBM Cloud Private -Managementkonsole, um einen Warteschlangenmanager in IBM Cloud Privatezu implementieren.

„Frühere CD-Releases eines IBM MQ-Images zu einem IBM Cloud Private-Cluster hinzufügen” auf Seite 30
Bereiten Sie für CD-Versionen von IBM MQ vor 9.1.4Ihren IBM Cloud Private -Cluster für die Bereitstellung eines produktionsbereiten Images für IBM MQvor.

V 9.1.4 Linux MQ Adv. CD Verbindung zu einem Warteschlangenmanager herstellen, der in einem OpenShift-Cluster implementiert ist

Konfigurationsbeispiele für die Herstellung einer Verbindung zu einem Warteschlangenmanager, der in einem Red Hat OpenShift-Cluster implementiert ist.

Informationen zu diesem Vorgang

Sie benötigen eine OpenShift-Route, um eine Verbindung einer Anwendung zu einem IBM MQ-Warteschlangenmanager von außerhalb eines Red Hat OpenShift-Clusters herzustellen.

Sie müssen TLS auf Ihrem IBM MQ-Warteschlangenmanager und in Ihrer Clientanwendung aktivieren, da Server Name Indication (SNI) nur im TLS-Protokoll verfügbar ist. Der Red Hat OpenShift Container Platform-Router) verwendet SNI für Routing-Anforderungen an den IBM MQ-Warteschlangenmanager.

Die erforderliche Konfiguration der OpenShift-Route hängt vom SNI-Verhalten Ihrer Clientanwendung ab.

Um den SNI-Header als TLS 1.2 oder höher festzulegen, muss eine CipherSpec oder CipherSuite für Ihre TLS-Kommunikation verwendet werden.

Die SNI wird auf den MQ -Kanal gesetzt, wenn die folgenden Bedingungen erfüllt sind:

- Der IBM MQ C-Client ist V8 oder höher.
- Der Java/JMS-Client ist V9.1.1 oder höher und die Java-Installation unterstützt die Klasse `javax.net.ssl.SNIHostName` .
- Der .NET-Client befindet sich im nicht verwalteten Modus.

Die SNI wird auf den Hostnamen gesetzt, wenn als Verbindungsname ein Hostname angegeben wird und die folgenden Bedingungen erfüllt sind:

- Der .NET-Client befindet sich im verwalteten Modus.
- Der AMQP-oder XR-Client wird verwendet.
- Die Java/JMS-Clients werden verwendet, wenn **AllowOutboundSNI** auf NOgesetzt ist.

Die SNI ist unter den folgenden Bedingungen nicht festgelegt und leer:

- Der IBM MQ C-Client ist V7.5 oder früher.
- IBM MQ C Client wird verwendet, wenn **AllowOutboundSNI** auf NOgesetzt ist.
- Die Java/JMS-Clients werden mit einer Java-Installation verwendet, die die Klasse `javax.net.ssl.SNIHostName` nicht unterstützt.

Beispiel

Auf dem Hostnamen basierende OpenShift-Routen: Für Clientanwendungen, die die SNI auf den Hostnamen setzen

Die folgenden Helm-Diagramme erstellen automatisch eine auf dem Hostnamen basierende OpenShift-Route für die Herstellung einer Verbindung einer Anwendung zu einem IBM MQ-Warteschlangenmanager. Clientanwendungen, die die SNI auf den Hostnamen setzen, können diese OpenShift-Route verwenden.

- ibm-mqadvanced-server-dev
- ibm-mqadvanced-server-prod
- ibm-mqadvanced-server-integration-prod in der IBM Cloud Pak for Integration.

Wenn Sie diese Diagramme nicht verwenden und eine eigene auf dem Hostnamen basierende OpenShift-Route erstellen müssen, können Sie die folgende `yaml` in Ihrem Cluster anwenden:

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

Auf dem MQ-Kanal basierende OpenShift-Routen: Für Clientanwendungen, die die SNI auf den MQ-Kanal setzen

Für Clientanwendungen, die die SNI auf den MQ-Kanal setzen, muss für jeden Kanal, zu dem eine Verbindung hergestellt werden soll, eine neue OpenShift-Route erstellt werden. Sie müssen auch eindeutige Kanalnamen in Ihrem Red Hat OpenShift-Cluster verwenden, um die Weiterleitung an den richtigen Warteschlangenmanager zu ermöglichen.

Um den erforderlichen Hostnamen für jede Ihrer neuen OpenShift-Routen zu ermitteln, müssen Sie jeden Kanalnamen einer SNI-Adresse zuordnen, wie hier dokumentiert: https://www.ibm.com/support/pages/ibm-websphere-mq-how-does-mq-provide-multiple-certificates-certlabl-capability

Anschließend müssen Sie eine neue OpenShift-Route (für jeden Kanal) erstellen, indem Sie die folgende `yaml` in Ihrem Cluster anwenden:

```
  apiVersion: route.openshift.io/v1
  kind: Route
  metadata:
    name: <provide a unique name for the Route>
    namespace: <the namespace of your MQ deployment>
  spec:
    host: <SNI address mapping for the channel>
    to:
      kind: Service
      name: <the name of the Kubernetes Service for your MQ deployment (for example "<Helm Release>-
```

```
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 "<Helm Release>-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 OpenShift Container Platform-Router (OCP) verwendet wird – normalerweise 443.

Zugehörige Tasks

„Warteschlangenmanager über die Helm -CLI implementieren” auf Seite 25
Verwenden Sie Helm, um einen Warteschlangenmanager in einem Red Hat OpenShift Container Platform-Cluster bereitzustellen. Diese Aufgabe sollte von einem Projektadministrator ausgeführt werden.

„Verbindung zur in einem OpenShift-Cluster implementierten IBM MQ Console herstellen” auf Seite 33
Sie können eine Verbindung zur IBM MQ Console eines Warteschlangenmanagers herstellen, die in einem Red Hat OpenShift Container Platform-Cluster implementiert wurde.

V 9.1.4 Linux MQ Adv. CD Verbindung zur in einem OpenShift-Cluster implementierten IBM MQ Console herstellen

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

Informationen zu diesem Vorgang

Wenn Sie den IBM MQ -Operator verwenden, finden Sie die URL IBM MQ Console auf der Detailseite `QueueManager` in der OpenShift -Webkonsole oder in der IBM Cloud Pak for Integration Platform Navigator. Alternativ können Sie sie mit folgendem Befehl über die OpenShift-CLI abrufen:

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

Beispiel

Die folgenden Helm-Diagramme erstellen automatisch eine OpenShift-Route für den Zugriff auf die IBM MQ Console

- ibm-mqadvanced-server-dev
- ibm-mqadvanced-server-integration-prod in der IBM Cloud Pak for Integration.

Sie können den Hostnamen der OpenShift-Route abrufen, indem Sie den folgenden Befehl ausführen:

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

Sie können über die folgende URL auf die IBM MQ Console zugreifen:

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

Zugehörige Tasks

„Warteschlangenmanager über die Helm -CLI implementieren” auf Seite 25
Verwenden Sie Helm, um einen Warteschlangenmanager in einem Red Hat OpenShift Container Platform-Cluster bereitzustellen. Diese Aufgabe sollte von einem Projektadministrator ausgeführt werden.

„Verbindung zu einem Warteschlangenmanager herstellen, der in einem OpenShift-Cluster implementiert ist” auf Seite 31
Konfigurationsbeispiele für die Herstellung einer Verbindung zu einem Warteschlangenmanager, der in einem Red Hat OpenShift-Cluster implementiert ist.

Linux MQ Adv. CD Warteschlangenmanagerkonfiguration über die OpenShift-CLI sichern und wiederherstellen

Die Warteschlangenmanagerkonfiguration zu sichern, kann dabei helfen, einen Warteschlangenmanager aus seinen Definitionen erneut zu erstellen, wenn die Warteschlangenmanagerkonfiguration verloren geht. Bei dieser Prozedur werden keine Warteschlangenmanagerprotokolldaten gesichert. Aufgrund des temporären Charakters von Nachrichten sind Langzeitprotokolldaten zum Zeitpunkt der Wiederherstellung normalerweise nicht mehr relevant.

Vorbereitende Schritte

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

Prozedur

- Sichern Sie die Warteschlangenmanagerkonfiguration.

  Mit dem Befehl **dmpmqcfg** können Sie einen Speicherauszug der Konfiguration eines IBM MQ-Warteschlangenmanagers erstellen.

  a) Rufen Sie den Namen des Pods für Ihren Warteschlangenmanager ab.
  Wenn Sie beispielsweise den Operator verwenden, können Sie den folgenden Befehl ausführen, wobei *Warteschlangenmanagername* der Name Ihrer `QueueManager` -Ressource ist:

  ```
  oc get pods --selector app.kubernetes.io/name=ibm-mq,app.kubernetes.io/instance=queue_ma
  nager_name
  ```

  Wenn Sie beispielsweise Helmverwenden, können Sie den folgenden Befehl ausführen, wobei *releasename* der Name Ihres Helm -Release ist.

  ```
  oc get pods --selector release=release_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.
  Wenn Sie beispielsweise den Operator verwenden, können Sie den folgenden Befehl ausführen, wobei *Warteschlangenmanagername* der Name Ihrer `QueueManager` -Ressource ist:

  ```
  oc get pods --selector app.kubernetes.io/name=ibm-mq,app.kubernetes.io/instance=queue_ma
  nager_name
  ```

Wenn Sie beispielsweise Helmverwenden, können Sie den folgenden Befehl ausführen, wobei *releasename* der Name Ihres Helm -Release ist.

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

Eigenen IBM MQ-Container erstellen

Entwickeln Sie einen selbsterstellten Container (früher "Docker-Container-Image"). Dies ist die flexibelste Containerlösung, Sie benötigen jedoch umfassende Kenntnisse für die Konfiguration von Containern und der resultierende Container sollte sich in Ihrem "Eigentum" befinden.

Vorbereitende Schritte

Vor dem Entwickeln eines eigenen Containers sollten Sie überlegen, ob Sie stattdessen einen der vordefinierten Container verwenden können, die von IBM bereitgestellt werden. Siehe IBM MQ in Containern

Informationen zu diesem Vorgang

Wenn Sie IBM MQ als Container-Image packen, können Änderungen an Ihrer Anwendung für Test- und Staging-Systeme ohne großen Aufwand implementiert werden. Dies kann bei einem Continuous Delivery in Ihrem Unternehmen einen großen Vorteil bedeuten.

Prozedur

- Informationen zur Erstellung eines IBM MQ-Container-Images mithilfe von Docker finden Sie in den folgenden Unterabschnitten:
  - Linux „Unterstützung für die Erstellung eigener IBM MQ-Container-Images und -Diagramme” auf Seite 8
  - „Eigenes Image des IBM MQ-Warteschlangenmanagers mithilfe eines Containers planen” auf Seite 35
  - „Beispiel für ein IBM MQ-Warteschlangenmanagerimage mit Docker erstellen” auf Seite 36
  - „Lokale Bindungsanwendungen in separaten Containern ausführen” auf Seite 39

Zugehörige Konzepte

IBM MQ in Containern

Eigenes Image des IBM MQ-Warteschlangenmanagers mithilfe eines Containers planen

Bei der Ausführung eines IBM MQ-Warteschlangenmanagers in einem Container sind mehrere Anforderungen zu berücksichtigen. Das Beispiel für das Container-Image bietet eine Möglichkeit, diesen Anforderungen gerecht zu werden. Wenn Sie jedoch ein eigenes Image verwenden möchten, müssen Sie berücksichtigen, wie diese Anforderungen erfüllt werden können.

Prozessüberwachung

Wenn Sie einen Container ausführen, führen Sie im Prinzip einen einzelnen Prozess (PID 1 innerhalb des Containers) aus, der später untergeordnete Prozesse generieren kann.

Wenn der Hauptprozess beendet wird, wird der Container von der Container-Laufzeit gestoppt. Für einen IBM MQ-Warteschlangenmanager müssen mehrere Prozesse im Hintergrund ausgeführt werden.

Aus diesem Grund müssen Sie sicherstellen, dass Ihr Hauptprozess aktiv bleibt, solange der Warteschlangenmanager aktiv ist. Es ist sinnvoll, zu überprüfen, ob der Warteschlangenmanager von diesem Prozess aus aktiv ist, z. B. durch Ausführen von Verwaltungsabfragen.

`/var/mqm` füllen

Container müssen mit `/var/mqm` als Datenträger konfiguriert werden.

Dabei ist das Verzeichnis des Datenträgers leer, wenn der Container zuerst gestartet wird. Dieses Verzeichnis wird in der Regel während der Installation gefüllt, bei Verwendung von eines Containers sind Installations- und Laufzeitumgebung jedoch separat.

V 9.1.0 Um dies zu beheben, können Sie beim Starten Ihres Containers den Befehl **crtmqdir** verwenden, um `/var/mqm` zu füllen, wenn er zum ersten Mal ausgeführt wird.

Beispiel für ein IBM MQ-Warteschlangenmanagerimage mit Docker 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 36
  - „Beispiel für ein konfiguriertes IBM MQ -Warteschlangenmanagerimage erstellen” auf Seite 37

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.

Ergebnisse

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

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. Erstellen Sie dazu eine eigene Container-Imageebene, wobei Sie das generische Image als übergeordnetes Element verwenden.

Vorbereitende Schritte

Für ein IBM MQ 9.1 -Image können Sie keinen sicheren Zugriff mit der Sicherheitskontexteinschränkung Red Hat OpenShift Container Platform "restricted" (SCC) konfigurieren. Die "eingeschränkte" SCC verwendet wahlfreie Benutzer-IDs und verhindert die Berechtigungseskalation, indem zu einem anderen Benutzer gewechselt wird. Das RPM-basierte IBM MQ 9.1 -Installationsprogramm stützt sich auf einen `mqm` -Benutzer und eine Gruppe und verwendet auch `setuid` Bits in ausführbaren Programmen.

Diese Einschränkung wurde in IBM MQ 9.2entfernt.

Vorgehensweise

1. Erstellen Sie ein neues Verzeichnis und fügen Sie eine Datei mit der Bezeichnung `config.mqsc` mit den folgenden Inhalten hinzu:

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

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

   Dabei gilt:

   - `johndoe` ist die Benutzer-ID, die Sie hinzufügen möchten.
   - `passw0rd` ist das ursprüngliche Kennwort.
3. Erstellen Sie mit folgendem Befehl ein angepasstes Container-Image:

```
sudo 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.

   Der Befehl **RUN** fügt einen Benutzer namens `johndoe` mit Kennwort `passw0rd` hinzu und der Befehl **COPY** fügt die Datei `config.mqsc` an einer bestimmten Position hinzu, die dem übergeordneten Image bekannt ist.

   Anmerkung: Unter Red Hat Enterprise Linux (RHEL) verwenden Sie den Befehl **`docker`** (RHEL V7) oder **podman** (RHEL V7 oder RHEL V8). Bei **podman** ist **sudo** zu Beginn des Befehls nicht erforderlich.
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:

```
sudo 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 Container mit dem Docker-Befehl **ps** ausführen. Sie können die IBM MQ-Prozesse anzeigen, die in Ihrem Container ausgeführt werden, indem Sie den Befehl "docker **top**" verwenden.

Achtung:

Sie können die Protokolle eines Containers mit dem Docker-Befehl **`logs ${CONTAINER_ID}`** anzeigen.

Nächste Schritte

- Wenn Ihr Container bei Verwendung des Docker-Befehls **ps** nicht angezeigt wird, ist der Container möglicherweise fehlgeschlagen. Sie können fehlgeschlagene Container mit dem Docker-Befehl **`ps -a`** anzeigen.

- Wenn Sie den Docker-Befehl **ps -a** verwenden, wird die Container-ID angezeigt. Diese ID wurde auch bei der Ausgabe des Docker-Befehls **run** ausgedruckt.
- Sie können die Protokolle eines Containers mit dem Docker-Befehl **logs ${CONTAINER_ID}** anzeigen.
- Sie können die maximale Anzahl offener Dateien mit dem Befehl **sysctl fs.file-max=524288**festlegen.

V 9.1.0 Lokale Bindungsanwendungen in separaten Containern ausführen

Bei der gemeinsame Nutzung von Prozessnamensbereichen zwischen Containern in Docker können Sie Anwendungen, für die eine lokale Bindungsverbindung zu IBM MQ erforderlich ist, in separaten Containern aus dem IBM MQ-Warteschlangenmanager heraus ausführen.

Informationen zu diesem Vorgang

Diese Funktion wird in IBM MQ 9.0.3-Warteschlangenmanagern und späteren Warteschlangenmanagern unterstützt.

Sie müssen die folgenden Einschränkungen beachten:

- Sie müssen den PID-Namensbereich der Container mit dem Argument `--pid` gemeinsam nutzen.
- Sie müssen den IPC-Namensbereich der Container mit dem Argument `--ipc` gemeinsam nutzen.
- Sie müssen eine der folgenden Schritte ausführen:
  1. Nutzen Sie den UTS-Namensbereich der Container gemeinsam mit dem Host, indem Sie das Argument `--uts` verwenden, oder
  2. Stellen Sie sicher, dass die Container denselben Hostnamen verwenden, indem Sie das Argument `-h` oder `--hostname` verwenden.
- Sie müssen das IBM MQ-Datenverzeichnis in einem Datenträger anhängen, der für alle Container unter dem Verzeichnis `/var/mqm` verfügbar ist.

Sie können diese Funktion ausprobieren, indem Sie die folgenden Schritte auf einem Linux-System ausführen, auf dem Docker bereits installiert ist.

Im folgenden Beispiel wird das IBM MQ-Beispielimage für Container verwendet. Einzelheiten zu diesem Image finden Sie unter Github.

Vorgehensweise

1. Erstellen Sie mit folgendem Befehl ein temporäres Verzeichnis, das als Datenträger verwendet werden soll:

```
mkdir /tmp/dockerVolume
```

2. Erstellen Sie mit folgendem Befehl einen Warteschlangenmanager (`QM1`) in einem Container mit der mit der Bezeichnung `sharedNamespace`:

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

3. Starten Sie einen zweiten Container mit dem Namen `secondaryContainer`, der auf `ibmcom/mq` basiert, aber erstellen Sie keinen Warteschlangenmanager, indem Sie folgenden Befehl ausgeben:

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

4. Führen Sie den Befehl **dspmq** im zweiten Container aus, um den Status der beiden Warteschlangenmanager anzuzeigen:

```
docker exec secondaryContainer dspmq
```

5. Führen Sie den folgenden Befehl aus, um die MQSC-Befehle für den Warteschlangenmanager zu verarbeiten, der im anderen Container aktiv ist:

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

Ergebnisse

Sie verfügen jetzt über lokale Anwendungen, die in separaten Containern ausgeführt werden, und können jetzt Befehle wie **dspmq**, **amqsput**, **amqsget** und **runmqsc** erfolgreich als lokale Bindungen zum QM1-Warteschlangenmanager aus dem sekundären Container ausführen.

Wenn nicht das erwartete Ergebnis angezeigt wird, finden Sie unter „Fehlerbehebung für Ihre Namensbereichsanwendungen” auf Seite 40 weitere Informationen.

V 9.1.0 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 39 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
```

API-Referenz für den IBM MQ-Operator

IBM MQ stellt einen Kubernetes-Operator für die native Integration mit OpenShift Container Platform bereit.

API-Referenz für mq.ibm.com/v1beta1

Über die API v1beta1 können QueueManager-Ressourcen erstellt und verwaltet werden.

Lizenzierungsreferenz für mq.ibm.com/v1beta1

Das Feld `spec.license.license`muss die Lizenzkennung für die Lizenz enthalten, die Sie akzeptieren. Gültige Werte sind:

Wert von `spec.license.license`	Wert von `spec.license.use`	Lizenzinformation
`L-RJON-BN7PN3`	`Production` oder `NonProduction`	IBM Cloud Pak for Integration 2020.2
`L-RJON-BPHL2Y`		IBM Cloud Pak for Integration Limited Edition 2020.2
`L-APIG-BJAKBF`	`Production` oder `Development`	IBM MQ Advanced 9.1.5
`L-APIG-BM7GDH`	`Development`	IBM MQ Advanced for Developers 9.1.5

Beachten Sie, dass die *Version* der Lizenz angegeben ist, die nicht immer mit der Version von IBM MQ identisch ist.

API-Referenz für QueueManager (mq.ibm.com/v1beta1)

QueueManager

Ein QueueManager ist ein IBM MQ-Server, der Warteschlangensteuerungs- und Publish/Subscribe-Services für Anwendungen bereitstellt.

Feld	Beschreibung
`apiVersion` Zeichenfolge	'apiVersion' gibt das versionierte Schema dieser Darstellung eines Objekts an. Server sollten erkannte Schemas in den letzten internen Wert umwandeln und können nicht erkannte Werte zurückweisen. Weitere Informationen: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#resources.
`kind` Zeichenfolge	'kind' ist ein Zeichenfolgewert zur Darstellung der REST-Ressource, die dieses Objekt darstellt. Server können dies von dem Endpunkt ableiten, an den der Client Anforderungen übergibt. Kann nicht aktualisiert werden. In Kamelschreibweise. Weitere Informationen: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds.
`metadata`	
`spec` „Spezifikation QueueManager" auf Seite 46	Der Soll-Status des QueueManager.
`status` „QueueManagerStatus" auf Seite 47	Der Ist-Status des QueueManager.

Verfügbarkeit

Verfügbarkeitseinstellungen für den Warteschlangenmanager, z. B., ob ein aktives Standby-Paar verwendet werden soll oder nicht.

Wird angezeigt in:

- „Konfiguration von QueueManager" auf Seite 44

Feld	Beschreibung
`type` Zeichenfolge	Verfügbarkeitstyp, der verwendet werden soll. Verwenden Sie "SingleInstance" für einen einzelnen Pod, der automatisch (in einigen Fällen) von Kubernetes erneut gestartet wird. Verwenden Sie "MultiInstance" für ein Paar Pods, von denen einer der "aktive" Warteschlangenmanager und der andere ein Standby-Warteschlangenmanager ist. Siehe Hochverfügbarkeit für IBM MQ in Containern in der neuesten Version von IBM MQ.

Lizenz

Einstellungen zur Steuerung der Annahme der Lizenz und der zu verwendenden Lizenzmetrik.

Wird angezeigt in:

- „Spezifikation QueueManager" auf Seite 46

Feld	Beschreibung
`use` Zeichenfolge	Einstellung zur Steuerung der Art der Verwendung der Software, wenn die Lizenz Mehrfachverwendungen unterstützt. Gültige Werte finden Sie unter https://ibm.biz/BdqvCF .
`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/BdqvCF .
`metric` Zeichenfolge	Einstellung zur Angabe der zu verwendenden Lizenzmetrik. Beispiel: "ProcessorValueUnit", "VirtualProcessorCore" oder "ManagedVirtualServer".

Grenzwerte

QueueManagerResourceList definiert CPU-und Speichereinstellungen.

Wird angezeigt in:

- „Ressourcen” auf Seite 50

Feld	Beschreibung
`cpu`	
`memory`	

LocalObjectReferenz

LocalObjectReference enthält genug Informationen, damit Sie das referenzierte Objekt innerhalb desselben Namensbereichs lokalisieren können.

Wird angezeigt in:

- „Spezifikation QueueManager” auf Seite 46

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?

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:

- „Spezifikation QueueManager” auf Seite 46

Feld	Beschreibung
`keys` „PKIS-Quelle” auf Seite 44 -Array	Private Schlüssel, die dem Schlüsselrepository des Warteschlangenmanagers hinzugefügt werden sollen.
`trust` „PKIS-Quelle” auf Seite 44 -Array	Zertifikate, die dem Schlüsselrepository des Warteschlangenmanagers hinzugefügt werden sollen.

PKIS-Quelle

PKISource definiert eine Quelle von Public Key Infrastructure-Informationen, z. B. Schlüssel oder Zertifikate.

Wird angezeigt in:

- „PKI" auf Seite 43

Feld	Beschreibung
name Zeichenfolge	Der Name wird als Bezeichnung für den Schlüssel oder das Zertifikat verwendet. Muss eine alphanumerische Zeichenfolge in Kleinschreibung sein.
secret „Geheimer Schlüssel" auf Seite 50	Übergeben Sie einen Schlüssel mithilfe eines Kubernetes-Secrets.

Konfiguration von QueueManager

QueueManagerConfig definiert die Einstellungen für den Warteschlangenmanager-Container und den zugrunde liegenden Warteschlangenmanager.

Wird angezeigt in:

- „Spezifikation QueueManager" auf Seite 46

Feld	Beschreibung
logFormat Zeichenfolge	Gibt das Protokollformat für diesen Container an. Verwenden Sie "JSON" für JSON-formatierte Protokolle aus dem Container. Verwenden Sie "Basic" für textformatierte Nachrichten.
metrics „Metriken für QueueManager" auf Seite 45	Einstellungen für Metriken im Prometheus-Stil.
readinessProbe „QueueManagerReadinessProbe" auf Seite 46	Einstellungen zur Steuerung des Bereitschaftstests.
resources „Ressourcen" auf Seite 50	Einstellungen zur Steuerung der Ressourcenanforderungen.
storage „QueueManagerSpeicher" auf Seite 49	Speichereinstellungen zur Steuerung der Verwendung von Persistent Volumes und Speicherklassen durch den Warteschlangenmanager.
availability „Verfügbarkeit" auf Seite 42	Verfügbarkeitseinstellungen für den Warteschlangenmanager, z. B., ob ein aktives Standby-Paar verwendet werden soll oder nicht.
imagePullPolicy Zeichenfolge	Diese Einstellung steuert, wann der Kubelet versucht, das angegebene Image zu extrahieren.
livenessProbe „QueueManagerLivenessProbe" auf Seite 45	Einstellungen zur Steuerung des Livetests.
debug boolesch	Gibt an, ob Debugnachrichten aus dem containerspezifischen Code in das Containerprotokoll übernommen werden sollen oder nicht.
image Zeichenfolge	Das Container-Image, das verwendet wird.
name Zeichenfolge	Name des zugrunde liegenden MQ-Warteschlangenmanagers, falls er vom metadata.name abweicht. Verwenden Sie dieses Feld, wenn der gewünschte Warteschlangenmanagername nicht den Kubernetes-Regeln für Namen entspricht (z. B. ein Name, der Großbuchstaben enthält).

QueueManagerLivenessProbe

Einstellungen zur Steuerung des Livetests.

Wird angezeigt in:

- „Konfiguration von QueueManager” auf Seite 44

Feld	Beschreibung
`failureThreshold` Ganzzahl	Mindestanzahl aufeinanderfolgender Fehler für den Test, damit er als fehlgeschlagen betrachtet wird, nachdem er erfolgreich war.
`initialDelaySeconds` Ganzzahl	Anzahl der Sekunden nach dem Start des Containers, bevor Livetests eingeleitet werden. 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.
`successThreshold` Ganzzahl	Mindestanzahl aufeinanderfolgender Erfolge für den Test, damit er als erfolgreich betrachtet wird, nachdem er fehlgeschlagen ist.
`timeoutSeconds` Ganzzahl	Anzahl der Sekunden, nach denen der Test das Zeitlimit überschreitet. Weitere Informationen: https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle#container-probes.

Metriken für QueueManager

Einstellungen für Metriken im Prometheus-Stil.

Wird angezeigt in:

- „Konfiguration von QueueManager” auf Seite 44

Feld	Beschreibung
`enabled` boolesch	Gibt an, ob ein Endpunkt mit Prometheus-kompatiblen Metriken aktiviert werden soll oder nicht.

QueueManagerOptionalVolume

PersistentVolume-Details für MQ-Wiederherstellungsprotokolle. Erforderlich bei Verwendung eines Multi-Instanz-Warteschlangenmanagers.

Wird angezeigt in:

- „QueueManagerSpeicher” auf Seite 49

Feld	Beschreibung
`class` Zeichenfolge	Speicherklasse, die für dieses Volume verwendet werden soll. Nur gültig, wenn "type" "persistent-claim" ist.
`enabled` boolesch	Gibt an, ob dieser Datenträger als separater Datenträger aktiviert oder auf dem Standarddatenträger "queueManager" platziert werden soll.
`size` Zeichenfolge	Größe des PersistentVolume , der an Kubernetesübergeben werden soll Nur gültig, wenn "type" "persistent-claim" ist.
`sizeLimit` Zeichenfolge	Größenbegrenzung bei Verwendung eines "ephemeren" Datenträgers. Da Dateien weiterhin in ein temporäres Verzeichnis geschrieben werden, können Sie mit dieser Option die Größe begrenzen. Nur gültig, wenn `type` "ephemeral" ist.

Feld	Beschreibung
`type` Zeichenfolge	Typ des zu verwendenden Volumes. Wählen Sie `ephemeral` aus, um einen nicht persistenten Datenträger "emptyDir" zu erstellen, oder `persistent-claim` , um einen persistenten Datenträger zu verwenden.

QueueManagerReadinessProbe

Einstellungen zur Steuerung des Bereitschaftstests.

Wird angezeigt in:

- „Konfiguration von QueueManager” auf Seite 44

Feld	Beschreibung
`failureThreshold` Ganzzahl	Mindestanzahl aufeinanderfolgender Fehler für den Test, damit er als fehlgeschlagen betrachtet wird, nachdem er erfolgreich war.
`initialDelaySeconds` Ganzzahl	Anzahl der Sekunden nach dem Start des Containers, bevor Livetests eingeleitet werden. 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.
`successThreshold` Ganzzahl	Mindestanzahl aufeinanderfolgender Erfolge für den Test, damit er als erfolgreich betrachtet wird, nachdem er fehlgeschlagen ist.
`timeoutSeconds` Ganzzahl	Anzahl der Sekunden, nach denen der Test das Zeitlimit überschreitet. Weitere Informationen: https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle#container-probes.

Spezifikation QueueManager

Der Soll-Status des QueueManager.

Wird angezeigt in:

- „QueueManager” auf Seite 42

Feld	Beschreibung
`license` „Lizenz” auf Seite 42	Einstellungen zur Steuerung der Annahme der Lizenz und der zu verwendenden Lizenzmetrik.
`pki` „PKI” auf Seite 43	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` „Konfiguration von QueueManager” auf Seite 44	QueueManagerConfig definiert die Einstellungen für den Warteschlangenmanager-Container und den zugrunde liegenden Warteschlangenmanager.
`securityContext` „SecurityContext” auf Seite 50	Sicherheitseinstellungen, die dem Sicherheitskontext (securityContext) des Warteschlangenmanager-Pods hinzugefügt werden sollen.
`tracing` „TracingConfig” auf Seite 52	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 9.1.5.0unter Verwendung der zweiten Revision des Container-Image an. Containerspezifische Fixes werden häufig in Form von Revisionen angewendet, z. B. Fixes für das Basisimage.

Feld	Beschreibung
affinity	Standardaffinitätsregeln von Kubernetes. Weitere Informationen: https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.17/#affinity-v1-core.
imagePullSecrets „LocalObjectReferenz" auf Seite 43 -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.
template „Schablone" auf Seite 51	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.
web „WebServer-Konfiguration" auf Seite 53	Einstellungen für den MQ-Web-Server.

QueueManagerStatus

Der Ist-Status des QueueManager.

Wird angezeigt in:

- „QueueManager" auf Seite 42

Feld	Beschreibung
endpoints „QueueManagerStatusEndpoint" auf Seite 48 -Array	Informationen zu den Endpunkten, die dieser Warteschlangenmanager zugänglich macht, z. B. API- oder UI-Endpunkte.
name Zeichenfolge	Der Name des Warteschlangenmanagers.
versions „QueueManagerStatusVersion" auf Seite 48	Verwendete MQ-Version und weitere Versionen, die aus der IBM Entitled Registry verfügbar sind.
adminUiUrl Zeichenfolge	URL für die Verwaltungsbenutzerschnittstelle.
conditions „QueueManagerStatusCondition" auf Seite 47 -Array	'conditions' (Bedingungen) stellen die letzten verfügbaren Beobachtungen zum Status des Warteschlangenmanagers dar.

QueueManagerStatusCondition

QueueManagerStatusCondition definiert die Bedingungen des Warteschlangenmanagers.

Wird angezeigt in:

- „QueueManagerStatus” auf Seite 47

Feld	Beschreibung
`message` Zeichenfolge	Lesbare Nachricht mit Anzeige von Details zum letzten Übergang.
`type` Zeichenfolge	Typ der Bedingung.
`lastTransitionTime` Zeichenfolge	Zeitpunkt des letzten Übergangs der Bedingung von einem Status in einen anderen.

QueueManagerStatusEndpoint

QueueManagerStatusEndpoint definiert die Endpunkte für den QueueManager.

Wird angezeigt in:

- „QueueManagerStatus” auf Seite 47

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.

QueueManagerStatusVersion

Verwendete MQ-Version und weitere Versionen, die aus der IBM Entitled Registry verfügbar sind.

Wird angezeigt in:

- „QueueManagerStatus” auf Seite 47

Feld	Beschreibung
`available` „QueueManagerStatusVersionVerfügbar” auf Seite 48	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.

QueueManagerStatusVersionVerfügbar

Weitere MQ-Versionen, die aus der IBM Entitled Registry verfügbar sind.

Wird angezeigt in:

- „QueueManagerStatusVersion” auf Seite 48

Feld	Beschreibung
`channels` Array	Kanäle, die für eine automatische Aktualisierung der MQ-Version verfügbar sind.
`versions` „Versionen” auf Seite 52 -Array	Bestimmte Versionen von MQ, die verfügbar sind.

QueueManagerSpeicher

Speichereinstellungen zur Steuerung der Verwendung von Persistent Volumes und Speicherklassen durch den Warteschlangenmanager.

Wird angezeigt in:

- „Konfiguration von QueueManager" auf Seite 44

Feld	Beschreibung
`persistedData` „QueueManagerOptionalVolume" auf Seite 45	Details zu Persistent Volumes für persistent gespeicherte MQ-Daten, einschließlich Konfiguration, Warteschlangen und Nachrichten. Erforderlich bei Verwendung eines Multi-Instanz-Warteschlangenmanagers.
`queueManager` „QueueManagerDatenträger" auf Seite 49	Standardmäßiges Persistent Volume für alle Daten, die normalerweise unter `/var/mqm` gespeichert sind. Enthält alle persistent gespeicherten Daten und Wiederherstellungsprotokolle, wenn keine anderen Volumes angegeben werden.
`recoveryLogs` „QueueManagerOptionalVolume" auf Seite 45	PersistentVolume-Details für MQ-Wiederherstellungsprotokolle. Erforderlich bei Verwendung eines Multi-Instanz-Warteschlangenmanagers.

QueueManagerDatenträger

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:

- „QueueManagerSpeicher" auf Seite 49

Feld	Beschreibung
`class` Zeichenfolge	Speicherklasse, die für dieses Volume verwendet werden soll. Nur gültig, wenn "type" "persistent-claim" ist.
`size` Zeichenfolge	Größe des PersistentVolume , der an Kubernetesübergeben werden soll Nur gültig, wenn "type" "persistent-claim" ist.
`sizeLimit` Zeichenfolge	Größenbegrenzung bei Verwendung eines "ephemeren" Datenträgers. Da Dateien weiterhin in ein temporäres Verzeichnis geschrieben werden, können Sie mit dieser Option die Größe begrenzen. Nur gültig, wenn `type` "ephemeral" ist.
`type` Zeichenfolge	Typ des zu verwendenden Volumes. Wählen Sie `ephemeral` aus, um einen nicht persistenten Datenträger "emptyDir" zu erstellen, oder `persistent-claim` , um einen persistenten Datenträger zu verwenden.

Anforderungen

QueueManagerResourceList definiert CPU-und Speichereinstellungen.

Wird angezeigt in:

- „Ressourcen" auf Seite 50

Feld	Beschreibung
`memory`	
`cpu`	

Ressourcen

Einstellungen zur Steuerung der Ressourcenanforderungen.

Wird angezeigt in:

- „Konfiguration von QueueManager" auf Seite 44

Feld	Beschreibung
`limits` „Grenzwerte" auf Seite 43	QueueManagerResourceList definiert CPU-und Speichereinstellungen.
`requests` „Anforderungen" auf Seite 49	QueueManagerResourceList definiert CPU-und Speichereinstellungen.

Geheimer Schlüssel

Übergeben Sie einen Schlüssel mithilfe eines Kubernetes-Secrets.

Wird angezeigt in:

- „PKIS-Quelle" auf Seite 44

Feld	Beschreibung
`items` Array	Schlüssel im Kubernetes-Secret, die zum Container des Warteschlangenmanagers hinzugefügt werden sollen.
`secretName` Zeichenfolge	Der Name des Kubernetes-Secrets.

SecurityContext

Sicherheitseinstellungen, die dem Sicherheitskontext (securityContext) des Warteschlangenmanager-Pods hinzugefügt werden sollen.

Wird angezeigt in:

- „Spezifikation QueueManager" auf Seite 46

Feld	Beschreibung
`supplementalGroups` Array	Eine Liste der Gruppen, die auf den ersten Prozess angewendet werden, der in jedem Container ausgeführt wird, zusätzlich zur primären GID des Containers. Wenn nicht angegeben, werden zu keinem Container Gruppen hinzugefügt.
`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.

Feld	Beschreibung
`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 Sie einen Speicheranbieter verwenden, der erfordert, dass Sie der Rootbenutzer für den Zugriff auf neu bereitgestellte Datenträger sind. Wenn Sie diese Eigenschaft auf "true" setzen, wirkt sich dies darauf aus, welches SCC-Objekt (SCC = Security Context Constraints) Sie verwenden können, und der Warteschlangenmanager kann möglicherweise nicht gestartet werden, wenn Sie nicht berechtigt sind, eine SCC zu verwenden, die den Rootbenutzer zulässt. Weitere Informationen siehe https://docs.openshift.com/container-platform/latest/authentication/managing-security-context-constraints.html.

Schablone

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:

- „Spezifikation QueueManager” auf Seite 46

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.

TracingAgent

Einstellungen für den optionalen Tracing-Agenten können nur in Cloud Pak for Integration konfiguriert werden.

Wird angezeigt in:

- „TracingConfig” auf Seite 52

Feld	Beschreibung
`image` Zeichenfolge	Das Container-Image, das verwendet wird.
`imagePullPolicy` Zeichenfolge	Diese Einstellung steuert, wann der Kubelet versucht, das angegebene Image zu extrahieren.
`livenessProbe` „TracingProbe” auf Seite 52	Einstellungen zur Steuerung des Livetests.
`readinessProbe` „TracingProbe” auf Seite 52	Einstellungen zur Steuerung des Bereitschaftstests.

TracingCollector

Einstellungen für den optionalen Tracing-Collector können nur in Cloud Pak for Integration konfiguriert werden.

Wird angezeigt in:

- „TracingConfig” auf Seite 52

Feld	Beschreibung
`image` Zeichenfolge	Das Container-Image, das verwendet wird.
`imagePullPolicy` Zeichenfolge	Diese Einstellung steuert, wann der Kubelet versucht, das angegebene Image zu extrahieren.
`livenessProbe` „TracingProbe" auf Seite 52	Einstellungen zur Steuerung des Livetests.
`readinessProbe` „TracingProbe" auf Seite 52	Einstellungen zur Steuerung des Bereitschaftstests.

TracingConfig

Einstellungen für Tracing-Integration mit dem Cloud Pak for Integration-Dashboard 'Operations'.

Wird angezeigt in:

- „Spezifikation QueueManager" auf Seite 46

Feld	Beschreibung
`agent` „TracingAgent" auf Seite 51	Einstellungen für den optionalen Tracing-Agenten können nur in Cloud Pak for Integration konfiguriert werden.
`collector` „TracingCollector" auf Seite 51	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).
`namespace` Zeichenfolge	Namensbereich, in dem das Cloud Pak for Integration-Dashboard 'Operations' installiert ist.

TracingProbe

Einstellungen zur Steuerung des Bereitschaftstests.

Wird angezeigt in:

- „TracingCollector" auf Seite 51

Feld	Beschreibung
`failureThreshold` Ganzzahl	Mindestanzahl aufeinanderfolgender Fehler für den Test, damit er als fehlgeschlagen betrachtet wird, nachdem er erfolgreich war.
`initialDelaySeconds` Ganzzahl	Anzahl der Sekunden nach dem Start des Containers, bevor Livetests eingeleitet werden. 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.
`successThreshold` Ganzzahl	Mindestanzahl aufeinanderfolgender Erfolge für den Test, damit er als erfolgreich betrachtet wird, nachdem er fehlgeschlagen ist.
`timeoutSeconds` Ganzzahl	Anzahl der Sekunden, nach denen der Test das Zeitlimit überschreitet. Weitere Informationen: https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle#container-probes.

Versionen

QueueManagerStatusVersion definiert eine Version von MQ.

Wird angezeigt in:

- „QueueManagerStatusVersionVerfügbar” auf Seite 48

Feld	Beschreibung
`name` Zeichenfolge	Version "name" für diese Version von QueueManager. Hierbei handelt es sich um gültige Werte für das Feld `spec.version`.

WebServer-Konfiguration

Einstellungen für den MQ-Web-Server.

Wird angezeigt in:

- „Spezifikation QueueManager” auf Seite 46

Feld	Beschreibung
`enabled` boolesch	Gibt an, ob der Web-Server aktiviert werden soll oder nicht.

Bemerkungen

Die vorliegenden Informationen wurden für Produkte und Services entwickelt, die auf dem deutschen Markt angeboten werden.

Möglicherweise bietet IBM die in dieser Dokumentation beschriebenen Produkte, Services oder Funktionen in anderen Ländern nicht an. Informationen über die gegenwärtig im jeweiligen Land verfügbaren Produkte und Services sind beim zuständigen IBM Ansprechpartner erhältlich. Hinweise auf IBM Lizenzprogramme oder andere IBM Produkte bedeuten nicht, dass nur Programme, Produkte oder Services von IBM verwendet werden können. Anstelle der IBM Produkte, Programme oder Services können auch andere, ihnen äquivalente Produkte, Programme oder Services verwendet werden, solange diese keine gewerblichen oder andere Schutzrechte der IBM verletzen. Die Verantwortung für den Betrieb von Fremdprodukten, Fremdprogrammen und Fremdservices liegt beim Kunden.

Für in diesem Handbuch beschriebene Erzeugnisse und Verfahren kann es IBM Patente oder Patentanmeldungen geben. Mit der Auslieferung dieser Dokumentation ist keine Lizenzierung dieser Patente verbunden. Lizenzanforderungen sind schriftlich an folgende Adresse zu richten (Anfragen an diese Adresse müssen auf Englisch formuliert werden):

IBM Europe
IBM Europe, Middle East and Africa
Tour Descartes
2, avenue Gambetta
92066 Paris La Défense
U.S.A.

Bei Lizenzanforderungen zu Double-Byte-Information (DBCS) wenden Sie sich bitte an die IBM Abteilung für geistiges Eigentum in Ihrem Land oder senden Sie Anfragen schriftlich an folgende Adresse:

Lizenzierung von geistigem Eigentum

IBM Japan, Ltd.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Trotz sorgfältiger Bearbeitung können technische Ungenauigkeiten oder Druckfehler in dieser Veröffentlichung nicht ausgeschlossen werden. Die Angaben in dieser Veröffentlichung werden in regelmäßigen Zeitabständen aktualisiert. Die Änderungen werden in Überarbeitungen oder in Technical News Letters (TNLs) bekanntgegeben. IBM kann jederzeit Verbesserungen und/oder Änderungen an den in dieser Veröffentlichung beschriebenen Produkten und/oder Programmen vornehmen.

Verweise in diesen Informationen auf Websites anderer Anbieter werden lediglich als Service für den Kunden bereitgestellt und stellen keinerlei Billigung des Inhalts dieser Websites dar. Das über diese Websites verfügbare Material ist nicht Bestandteil des Materials für dieses IBM Produkt.

Werden an IBM Informationen eingesandt, können diese beliebig verwendet werden, ohne dass eine Verpflichtung gegenüber dem Einsender entsteht.

Lizenznehmer des Programms, die Informationen zu diesem Produkt wünschen mit der Zielsetzung: (i) den Austausch von Informationen zwischen unabhängigen, erstellten Programmen und anderen Programmen (einschließlich des vorliegenden Programms) sowie (ii) die gemeinsame Nutzung der ausgetauschten Informationen zu ermöglichen, wenden sich an folgende Adresse:

IBM Europe, Middle East and Africa
Software Interoperability Coordinator, Department 49XA
3605 Highway 52 N
Rochester, MN 55901
U.S.A.

Die Bereitstellung dieser Informationen kann unter Umständen von bestimmten Bedingungen - in einigen Fällen auch von der Zahlung einer Gebühr - abhängig sein.

Die Lieferung des in diesen Informationen beschriebenen Lizenzprogramms sowie des zugehörigen Lizenzmaterials erfolgt auf der Basis der IBM Rahmenvereinbarung bzw. der Allgemeinen Geschäftsbedingungen von IBM, der IBM Internationalen Nutzungsbedingungen für Programmpakete oder einer äquivalenten Vereinbarung.

Die in diesem Dokument enthaltenen Leistungsdaten stammen aus einer kontrollierten Umgebung. Die Ergebnisse, die in anderen Betriebsumgebungen erzielt werden, können daher erheblich von den hier erzielten Ergebnissen abweichen. Einige Daten stammen möglicherweise von Systemen, deren Entwicklung noch nicht abgeschlossen ist. Eine Gewährleistung, dass diese Daten auch in allgemein verfügbaren Systemen erzielt werden, kann nicht gegeben werden. Darüber hinaus wurden einige Daten unter Umständen durch Extrapolation berechnet. Die tatsächlichen Ergebnisse können davon abweichen. Benutzer dieses Dokuments sollten die entsprechenden Daten in ihrer spezifischen Umgebung prüfen.

Alle Informationen zu Produkten anderer Anbieter stammen von den Anbietern der aufgeführten Produkte, deren veröffentlichten Ankündigungen oder anderen allgemein verfügbaren Quellen. IBM hat diese Produkte nicht getestet und kann daher keine Aussagen zu Leistung, Kompatibilität oder anderen Merkmalen machen. Fragen zu den Leistungsmerkmalen von Produkten anderer Anbieter sind an den jeweiligen Anbieter zu richten.

Aussagen über Pläne und Absichten von IBM unterliegen Änderungen oder können zurückgenommen werden und repräsentieren nur die Ziele von IBM.

Diese Veröffentlichung enthält Beispiele für Daten und Berichte des alltäglichen Geschäftsablaufes. Um diese so realistisch wie möglich zu gestalten, enthalten sie auch Namen von Personen, Firmen, Marken und Produkten. Sämtliche dieser Namen sind fiktiv. Ähnlichkeiten mit Namen und Adressen tatsächlicher Unternehmen oder Personen sind zufällig.

COPYRIGHTLIZENZ:

Diese Veröffentlichung enthält Musterprogramme, die in Quellensprache geschrieben sind. Sie dürfen diese Musterprogramme kostenlos (d. h. ohne Zahlung an IBM) kopieren, ändern und verteilen, wenn dies zu dem Zweck geschieht, Anwendungsprogramme zu entwickeln, zu verwenden, zu vermarkten oder zu verteilen, die mit der Anwendungsprogrammierschnittstelle für die Betriebsumgebung konform sind, für die diese Musterprogramme geschrieben werden. Diese Beispiele wurden nicht unter allen denkbaren Bedingungen getestet. Daher kann IBM die Zuverlässigkeit, Wartungsfreundlichkeit oder Funktion dieser Programme weder zusagen noch gewährleisten.

Wird dieses Buch als Softcopy (Book) angezeigt, erscheinen keine Fotografien oder Farbabbildungen.

Informationen zu Programmierschnittstellen

Die bereitgestellten Informationen zur Programmierschnittstelle sollen Sie bei der Erstellung von Anwendungssoftware für dieses Programm unterstützen.

Dieses Handbuch enthält Informationen über vorgesehene Programmierschnittstellen, die es dem Kunden ermöglichen, Programme zu schreiben, um die Services von WebSphere MQzu erhalten.

Diese Informationen können jedoch auch Angaben über Diagnose, Bearbeitung und Optimierung enthalten. Die Informationen zu Diagnose, Bearbeitung und Optimierung sollten Ihnen bei der Fehlerbehebung für die Anwendungssoftware helfen.

Wichtig: Verwenden Sie diese Diagnose-, Änderungs-und Optimierungsinformationen nicht als Programmierschnittstelle, da sie Änderungen unterliegen.

Marken

IBM, das IBM Logo, ibm.com, sind Marken der IBM Corporation in den USA und/oder anderen Ländern. Eine aktuelle Liste der IBM Marken finden Sie auf der Webseite "Copyright and trademark information"www.ibm.com/legal/copytrade.shtml. Weitere Produkt- und Servicenamen können Marken von IBM oder anderen Unternehmen sein.

Microsoft und Windows sind Marken der Microsoft Corporation in den USA und/oder anderen Ländern.

UNIX ist eine eingetragene Marke von The Open Group in den USA und anderen Ländern.

Linux ist eine eingetragene Marke von Linus Torvalds in den USA und/oder anderen Ländern.

Dieses Produkt enthält Software, die von Eclipse Project (http://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: