# Customising an IBM Provided IBM MQ Container Image with a Metrics Collector and Prometheus Emitter

IBM provide pre-built IBM MQ (MQ) container images for use with the MQ Operator, but what if you want to customise an image by adding your own exits, or the latest MQ Metrics collector? This article will describe some good practices and methods for customising an IBM supplied container image. The notes and examples are based on an IBM MQ Operator controlled deployment running on the Red Hat OpenShift Container Platform (OCP).

## Important considerations and good practice for using a custom image

Customising the IBM supplied container image is supported but you must be aware that there are implications, please read the following notes and reach out to your IBM representative if there is anything you are unsure of.

### Versioning

The first thing to consider is version control. All MQ images are tagged with a version number in the V.M.R.F format, I strongly recommend that you use that same version number for your own images but change the name of the image to reflect the customisation. In following example we will take the image name and version of MQ developer edition and add “-with-metrics” e.g. “mq-with-metrics: 9.3.3.2-r3”, you can add something of your choosing - perhaps a company or project name.

MQ image version history can be found here,

<https://www.ibm.com/docs/en/ibm-mq/9.3?topic=operator-release-history-mq>

Another thing to consider is that the MQ Operator uses a field in the QueueManager Custom Resource (CR) YAML called spec.version. The spec.version field must match the version of the base MQ image that you have customised. Keeping the version number consistent will also help you track and maintain you MQ deployments.

### Specifying a custom image

To use a custom image, you have to make use of another field in the queue manager CR called spec.queueManager.image. When you explicitly specify the MQ image you cannot then simply change the spec.version field in the queue manager YAML to have the Operator automatically upgrade your queue managers anymore. You must change the spec.version field and spec.queueManager.image together when you want to upgrade your queue manager, or you can remove the custom image and revert to a standard image by removing spec.queueManager.image.

## Building the IBM MQ custom image with the metrics collector

All samples can be found here, <https://github.com/ibm-messaging/mq-gitops-samples/tree/main/queue-manager-custom-metrics>

If you want to run the Metrics Collector and Prometheus Emitter inside the same container as MQ you can build on top of the IBM supported container image and then create an MQ Service to control the collection and emitting of metrics. This approach offers some advantages over previously documented approaches that build a separate metrics container image and run the collector/emitter in another Pod which makes an MQ Client connection into the queue manager.

The main advantage of putting the metrics code into the MQ image is that all the metric data is collected within the same container as the queue manager and only the Prometheus data is transferred across the network to the Prometheus pod, this is more efficient and scalable than having hundreds of additional pods receiving the data via client connections. The other advantage is that you don’t have to setup client security and connections on the queue manager as the queue manager itself will run the collector and emitter.

## Steps to build the custom image

1. Go to the OCP OperatorHub and install ‘Red Hat OpenShift Pipelines’, you can take the defaults for this example. Tip seach for ‘pipeline’
2. If you don’t have an existing project create a new one, we will be using ‘mq-demo’ for this example. When you login to the command line make sure you are using the right project as the YAML files do not specify a namespace.

$oc project mq-demo

1. In the mq-demo project under ‘Storage’ - create a PersistentVolumeClaim (PVC) for the pipeline to use as a shared workspace between tasks e.g., 'mq-metrics-workspace', request 4GB, select RWO and volume mode Filesystem. There is a sample PVC YAML file in the repo.

$oc create -f example-pvc.yaml

1. Create Secrets for docker.io and your external image registry, for this example I am using the internal image registry located here, image-registry.openshift-image-registry.svc:5000, in the repo you will find a file called setup-commands.sh with examples for creating the secrets and linking them to the service account. Note, you don’t need to add docker.io unless you get the error saying downloads exceeded. Also note that it is the buildah task that requires the external registry credentials, these are mapped to its workspace for dockerconfig.
2. Create a new Pipeline called ‘build-mq-with-metrics’ using the example YAML from the repo under the pipelines folder.

$oc create -f build-mq-with-metrics.yaml

1. To run the pipeline you can either use the example pipeline-run yaml or you can create a template that allows you to change the name of the pipeline run when you insert it into your cluster, this is handy if you want to keep old runs.

Either,

$oc create -f pipeline-run.yaml

or

$oc create -f pipeline-run-template.yaml

$oc process mq-metrics-pipeline-run-template --param=runNumber=02 | oc create -f -

Example commands are in the example-commands.sh file

Notes:

To find the tag for a task image search here,

[Red Hat Certified Products & Services - Red Hat Ecosystem Catalog](https://catalog.redhat.com/)

-------------------------------------------------

#oc secrets link pipeline docker-login --for=pull

steps:

a. Click ‘Add a task’, call it ‘git clone’ and start typing type git-clone, next select the Red Hat supplied git-clone task and install/add it.

b. Select the task and configure with these parameters:

url: ‘https://github.com/ibm-messaging/mq-metric-samples.git’

revision: ' v5.5.2' refer to the URL above for the latest tag

workspace: select ‘workspace’ in the drop-down menu

c. Click, ‘Add workspace’ and type 'workspace'

d. Create the Pipeline

5. Add the following YAML files, ensure the role binding is added first because the PipelineRun will start the clone task:

kind: RoleBinding

apiVersion: rbac.authorization.k8s.io/v1

metadata:

name: registry-editor

namespace: default

subjects:

- kind: ServiceAccount

name: pipeline

namespace: mq-demo

roleRef:

apiGroup: rbac.authorization.k8s.io

kind: ClusterRole

name: registry-editor

---

apiVersion: tekton.dev/v1beta1

kind: PipelineRun

metadata:

name: build-mq-image-metrics-run

namespace: mq-demo

spec:

pipelineRef:

name: build-mq-image-metrics

serviceAccountName: pipeline

podTemplate:

securityContext:

runAsNonRoot: false

workspaces:

- name: workspace

persistentVolumeClaim:

claimName: mq-demo-workspace-pvc

6. Now deploy the two tasks that are in the tasks folder of this repo, ensure you update the namespace to match yours or mq-demo.

7. Edit pipeline and add the second task ‘build-metrics’ task, select ‘workspace’ from the drop-down for output in the Workspaces section and save.

8. Rerun PipelineRun to test the building of the collector and Prometheus emitter.

9. Edit the pipeline again and add the ‘build-mq-container’ task, on the Workspaces section, as before, select ‘workspace’ for output and ‘workspace’ for dockerconfig.

10. Add your IBM entitlement key to be able to pull from 'cp.icr.io',

a. Create a secret in the mq-demo project/namespace called ' ibm-entitlement-key' and then grant the service account 'pipeline' access

$ oc create secret docker-registry ibm-entitlement-key --namespace=mq-demo --docker-username=cp --docker-server=cp.icr.io --docker-password=your\_key

$ oc secrets link pipeline ibm-mqadvanced-server --for=pull

$ oc secrets link pipeline ibm-mqadvanced-server

9. Test the pipeline by rerunning again.

Notes:

To find the tag for a task image search here,

[Red Hat Certified Products & Services - Red Hat Ecosystem Catalog](https://catalog.redhat.com/)

buidah:latest is available but git-init requires a version.

Add docker login if downloads exceeded.

oc secrets link pipeline docker-login --for=pull

oc secrets link pipeline docker-login

Add internal reg creds