Running the MQ docker image on the Kubernetes service in Bluemix

Matt Roberts

Published on 04/09/2017 / Updated on 13/09/2017

In March, we announced the availability of the <u>IBM MQ Advanced for Developers docker</u> <u>image as part of the Bluemix Container service</u>, which provided a simple way to try out IBM MQ using a cloud deployment. Since that time, the <u>original version of the Bluemix Container</u> <u>Service has been deprecated</u> in favour of the new <u>Kubernetes service on Bluemix</u> which provides a single tenant Kubernetes cluster into which you can deploy the Docker containers of your choice.

Importantly, IBM's public images (like MQ) and any private images that you uploaded to the container registry are **still available** for use with the new Kubernetes service. You can continue to try out IBM MQ Advanced for Developers for free, using these three simple steps;

- 1. Launch a free Kubernetes cluster using the Kubernetes service in Bluemix
- 2. Deploy the IBM MQ Advanced for Developers container image into your cluster
- 3. <u>Connect your favourite administration tooling and applications to try it out!</u>

Step 1: Launch a free Kubernetes cluster using the Kubernetes service in Bluemix

You can launch a free Kubernetes cluster using the Bluemix user interface by selecting the <u>Kubernetes Cluster service in the service Catalog</u> as shown in the following screenshots, or otherwise use the Bluemix CLI commands shown below.

Infrastructure	Search ted by creating a Kubernetes cluster, or manage you Looking for container images? Don't worry: Bot Registry tile.			Filter
Compute Get sta Storage Network Security	Looking for container images? Don't worry: Bot			
Storage Network Security	Looking for container images? Don't worry: Bot	oth unit of		
Containers >	negistry tile.	our your pr	private images and IBM's public images can be seen in the registry console. Select the Container	×
VMware Platform	Kubernetes Cluster Provision a cluster of hosts, called worker		Container Registry Store and distributo Docker images by using	
Boilerplates APIs	nodes, to deploy and manage highly available.		this managed private registry. Make inter-	
Application Services Blockchain				

The Containers page in the Bluemix catalog

😑 🤹 IBM Bluemix	ĸ			Catalog	Support	Manage
Dashboard / Clusters / Create a Kube	ernetes Cluster					
mq-test Kubernetes version Lite - 2 CPUs, 4 GB RAM 1 worker node Total	Free	Cluster Name mq-test	Create Cluster			
Cluster type Standard Lite Create a free cluster that com GB memory, and 1 worker noc	nes with 2 CPUs, 4					
Terms Docs						

Creating a "lite" Kubernetes cluster

To create your cluster using the CLI, and to complete the remainder of this tutorial you will need the following command-line tools installed on your laptop;

- Install the Bluemix CLI as described here
- Install the <u>"container-service" CLI plugin</u> to manage the Kubernetes cluster, for example with the following Bluemix CLI command. (Note that this is different to the "IBM-Containers" plugin which is for the original style Bluemix Container Service)

bx plugin install container-service -r Bluemix

• Install the <u>Kubernetes CLI</u> to allow you to deploy containers

Once you have the tools installed you can use the following commands to create a free "lite" cluster in the Kubernetes service in Bluemix.

Log in to Bluemix (use the "--sso" option if you have a federated ID) bx login

Set the Bluemix endpoint for your preferred region bx api https://api.eu-gb.bluemix.net

Set the target org and space.# You can use "bx iam orgs" and "bx iam spaces" to get your org/space names bx target -o "orgName" -s spaceName

Initialise the container service connectivity if you haven't done so already bx cs init

Create a free Kubernetes cluster (equivalent of the UI steps above) bx cs cluster-create --name mq-test

List your clusters to check the status of the new cluster bx cs clusters

Initially your cluster will be in the "deploying" state. You must wait until it reaches the "normal" or "ready" state before you proceed to the next step – this may take some time so be patient!

Once your cluster reaches "normal" or "ready" state you can connect to your cluster using the instructions shown in the "Access" section of the details about your Kubernetes cluster in the Bluemix UI. This downloads the kube-config file to your machine.

bx cs cluster-config mq-test

You must then follow the provided instructions to export the KUBECONFIG environment variable so that you can execute commands against your cluster.

export KUBECONFIG=...

You can then verify your worker nodes by typing the following, and confirm you have a single worker node in status "Ready";

kubectl get nodes

NAME STATUS AGE 10.126.110.230 Ready 2m

If you wish, you can optionally set up a proxy to access the Kubernetes dashboard by typing the following command;

kubectl proxy

With the kubectl proxy running you can then access the Kubernetes console by opening a web browser to <u>http://127.0.0.1:8001/ui</u>.

Step 2: Deploy the IBM MQ Advanced for Developers container image into your cluster

You can see the IBM Public images in the Bluemix user interface by going to the <u>Bluemix</u> <u>Catalog</u> and selecting Containers > Container Service > Registry > <u>IBM Public Repositories</u>.

😑 🧠 IBM Bluemix	Containers		Cat	alog Support Ma
Clusters Containers Registry	Container Service / Registry	c Repositories		
Private Repositories	Repositories			
IBM Public Repositories	8 Repositories		O ₄ Filter	
	REPOSITORY	DESCRIPTION	DOCS	SECURITY REPORT ~
	ibmliberty	Deploy WAR, EAR, or OSGi apps that are based on Java in an IBM WebSphere Application Server Liberty	۵	Secure
	ibmnode	Build and deploy Node is apps, using this image as a parent.		Secure
	ibm-mq	Start developing your own messaging solutions with IBM MQ Advanced for Developers.		Secure
	ibm-integration-bus	Start developing your own integration solutions with IBM Integration Bus for Developers.		Secure
	ibm-node-strong-pm	Run an instance of the StrongLoop Process Manager and deploy Node is apps that were implemented on		Secure
	lbm-websphere-extreme-scale	Run one or more instances of WebSphere eXtreme Scale caching server to provide scalability and predict		Secure

IBM Public images in the Container Registry

Click on the "ibm-mq" image to see how to deploy an instance of that container to your cluster using the command line;

😑 🤹 IBM Bluem	nix Containers					Catalog	Support	Manage
Image Tags		ervice / Registry / IBM Publi						
	Start develo	oping your own messaging sol	utions with IBM MQ Adva	nced for Developers, Do	<u>CS</u>			-
	Tags							
	1 Tags					Q, Filter		
	TAGS	DIGEST	SIZE	CREATED	SECURITY REPORT -			
	latest	df357b115b7e	455 MB	11 days ago	Secure			
	Pull an IBM p	ublic image to your local machine.						
	docker	pull registry.eu-gb.b	luemix.net/ibm-mq					Ð
	Deploy to a k	ubernetes cluster.						
	kubectl	run <service_name> -</service_name>	-image=registry.eu	-gb.bluemix.net/ibm-n	ng			回

IBM MQ image in the public repository

Note: You do NOT have to execute the docker pull command shown in the screenshot above for the purposes of this exercise – that step is only required if you want to use the docker image locally on your laptop, for example to deploy to minikube or similar.

The following command will launch an instance of the container in your Kubernetes cluster, where "my-mq" is a name of your choice that will be given to your Kubernetes deployment (you can choose any name you like that meets the validation rules imposed by Kubernetes). Note that the environment variables ACCEPT and MQ_QMGR_NAME are important as they control the configuration of your container when it first starts up, as described in <u>step 8 of</u> the container image documentation here.

kubectl run my-mq --image=registry.eu-gb.bluemix.net/ibm-mq \ --env="LICENSE=accept" --env="MQ_QMGR_NAME=QM1"

Check that the container has started successfully by confirming that the pod status is "Running";

kubectl get pods

NAME READY STATUS RESTARTS AGE my-mq-1814346958-s0crs 1/1 Running 0 4s

Make a note of the pod name (e.g. "my-mq-1814346958-s0crs"), then wait for a few seconds then check the container logs to confirm that the queue manager has finished being configured and started;

kubectl logs my-mq-1814346958-s0crs

... Monitoring Queue Manager QM1 QMNAME(QM1) STATUS(Running) IBM MQ Queue Manager QM1 is now fully running Server mqweb started with process ID 334.

Step 3: Connect your favourite administration tooling and applications to try it out!

There are two basic options for connecting to your new container;

- a. Set up port forwarding from your local machine directly to the container
- b. Configure Kubernetes to make your container accessible over the public internet

In both cases you will need to know the default credentials that have been configured for you inside the container as described <u>on GitHub here</u>.

Step 3a: Set up port forwarding from your local machine directly to the container

Use the following command to set up port forwarding from your local machine to the container for the 1414 (MQ Channel) and 9443 (MQ Web Console) ports. This has the advantage that your queue manager cannot be accessed over the public internet, but can only be accessed from the laptop where the port forwarding has been configured (unless you set up further network configuration to allow other instances to route through the local machine).

kubectl port-forward my-mq-1814346958-s0crs 9443 1414

You can now access your container by connecting to localhost and the port required for the action you want to carry out, for example;

• To load the MQ Web Console you can point your browser to <u>https://localhost:9443/ibmmq/console</u> and use the default credentials of *admin / passwOrd* • Similarly you can attach MQ Explorer or messaging applications to localhost:1414 (the default channel name is DEV.ADMIN.SVRCONN).

	and nanoword	
Please enter your usernan	ne and password	
User Name:	admin	
Password:		
Please note that after some tim	ne you will be signed out automatically and asked to	sign in again

Log in to the MQ Web Console via port forwarding

Step 3b: Configure Kubernetes to make your container accessible over the public internet

Alternatively you can use Kubernetes to make your container accessible over the public internet. For development and test purposes the simplest approach is to create a Kubernetes service that exposes the necessary endpoints using a NodePort, and since our free cluster only has one worker we don't have to worry about the worker IP address changing.

The following commands create a service for each of the two ports that we want to access in our container;

kubectl expose pod my-mq-1814346958-s0crs --port 1414 --name mqchannel --type NodePort kubectl expose pod my-mq-1814346958-s0crs --port 9443 --name mqwebconsole --type NodePort

Having created the service you now need to look up the port numbers that have been allocated to the NodePort using the "get services" command. In the example below the MQ Channel is exposed publicly on port 30063 and the MQ Web Console on port 32075.

kubectl get services

 NAME
 CLUSTER-IP
 EXTERNAL-IP
 PORT(S)
 AGE

 kubernetes
 10.10.10.1
 none
 443/TCP
 22h

 mqwebconsole
 10.10.10.128
 nodes
 9443:32075/TCP
 2m

 mqchannel
 10.10.10.44
 nodes
 1414:30063/TCP
 2m

Lastly you need to obtain the public IP address of the worker node, for example in the example shown below the public IP address of the worker is 169.51.10.240;

bx cs workers mq-test

```
ID Public IP Private IP Machine Type State Status
kube-par01-pa7f80000007845aaaaf806224d5a53dc-w1 169.51.10.240 10.126.110.230 free normal
Ready
```

Combine the IP address and the port number together to access the relevant endpoint over the internet, for example;

- MQ Web Console: https://169.51.10.240:32075/ibmmq/console/ (admin / passw0rd)
- MQ Explorer: 169.51.10.240, port 30063 (admin / passw0rd, channel=DEV.ADMIN.SVRCONN)

Summary

In this article we described how to try out MQ Advanced for Developers for free using the Docker container image and the Kubernetes service in IBM Bluemix. We described how to create a free Kubernetes cluster, deploy the MQ Docker image into that cluster and successfully connect to the container to use the deployed queue manager.

Happy Messaging!