Quick guide

RH OpenShift on IBM Cloud

And

Tekton Pipelines

Running up a minimal Openshift 4.3.5 beta on IBM Cloud and installing and testing Tekton.

Dave Arnold

May 2020

Contents

[IBM Cloud – provision a Red Hat OpenShift on IBM Cloud service 4](#_Toc40886562)

[Accessing the Red Hat OpenShift service 7](#_Toc40886563)

[Installing Tekton 8](#_Toc40886564)

[C:\openshift>oc new-project tekton-pipelines 8](#_Toc40886565)

[C:\openshift>oc adm policy add-scc-to-user anyuid -z tekton-pipelines-controller 9](#_Toc40886566)

[C:\openshift>oc apply --filename https://storage.googleapis.com/tekton-releases/latest/release.yaml 9](#_Toc40886567)

[C:\openshift>oc get pods --namespace tekton-pipelines 9](#_Toc40886568)

[Correcting the paths to the image locations 10](#_Toc40886569)

[C:\openshift>oc get pods --namespace tekton-pipelines 12](#_Toc40886570)

[Test Tekton with an Echo example 12](#_Toc40886571)

[C:\openshift>oc apply -f Tekton-ivt\echo-hello-world-task.yaml 14](#_Toc40886572)

[C:\openshift>oc apply -f Tekton-ivt\echo-hello-world-task-run.yaml 14](#_Toc40886573)

[C:\openshift>oc get taskruns/echo-hello-world-task-run -o yaml 14](#_Toc40886574)

[OpenShift Console Pipelines 16](#_Toc40886575)

[Initial Build and run MQ (TLS) from certified base image on RH OpenShift 17](#_Toc40886576)

[Login in to the openshift cluster from the command line. 17](#_Toc40886577)

[oc project da-build-project 17](#_Toc40886578)

[oc create -f c:\openshift\data\build-mq-custom.yaml 17](#_Toc40886579)

[oc adm policy add-scc-to-user anyuid -z default 17](#_Toc40886580)

[oc new-build https://github.com/DAVEXACOM/ibm-mqadvanced-server-tls-build.git 18](#_Toc40886581)

[oc new-app ibm-mqadvanced-server-tls-build --env LICENSE=accept 19](#_Toc40886582)

[oc new-app ibm-mqadvanced-server-tls-build --env LICENSE=accept --env MQ\_QMGR\_NAME=myappqmgr 19](#_Toc40886583)

[oc expose svc/ibm-mqadvanced-server-tls-build 20](#_Toc40886584)

[oc status 20](#_Toc40886585)

[oc create -f c:\openshift\data\create-mq-route.yaml 20](#_Toc40886586)

[oc get routes 21](#_Toc40886587)

[Create CLNTCONN Channel definition 21](#_Toc40886588)

[runmqsc -n < CUSTOMIMAGE.mqsc 22](#_Toc40886589)

[Check the AMQCLCHL.TAB contents 22](#_Toc40886590)

[Setting up the local KeyStore for TLS client connection 22](#_Toc40886591)

[set MQSSLKEYR=C:\Program Files\IBM\MQ\arnold 22](#_Toc40886592)

[runmqsc -c mqappqmgr 23](#_Toc40886593)

[Tekton Pipeline for IBM MQ (tls enabled) 24](#_Toc40886594)

[Create the Pipeline Service Account 24](#_Toc40886595)

[Create the (non executing)Tekton artifacts in RH Openshift 24](#_Toc40886596)

[Create and test the Tekton Taskrun artifacts in RH Openshift 25](#_Toc40886597)

[Create and test the Tekton Pipelinerun artifact in RH Openshift 33](#_Toc40886598)

[Checking the Pipeline Run results 36](#_Toc40886599)

[Creating the Service and Routes 37](#_Toc40886600)

[https://github.com/DAVEXACOM/tekton-mq-example/tree/master/mqdeploymentconfig 37](#_Toc40886601)

[Tekton tt pre-fixed mq service - tt-ibm-mqadvanced-server-tls-build 37](#_Toc40886602)

[Tekton tt prefixed mq route – tt-ibm-mqadvanced-server-tls-build 38](#_Toc40886603)

[Tekton tt prefixed mq tls route – tls-tlsprqm1p-tt 38](#_Toc40886604)

[Clean up to get tls-tlsprqm1p-tt working 39](#_Toc40886605)

[Testing Connection to the queue manager 40](#_Toc40886606)

[Local Client connection MQSC 40](#_Toc40886607)

[Download and copy in the client side Keystore 41](#_Toc40886608)

[Set the MQSSLKEY environment variable 41](#_Toc40886609)

[Use runmqsc client to connect to your queue manager 41](#_Toc40886610)

[Exploring ConfigMaps for MQSC definition abstraction 43](#_Toc40886611)

[Manually investigating Config Maps and Deployment Configs 43](#_Toc40886612)

[Create a config map with an MQSC file 43](#_Toc40886613)

[Tekton Mq Deploy DeploymentConfig 44](#_Toc40886614)

[Final DeploymentConfig for MQ to run MQSCs from ConfigMap 47](#_Toc40886615)

[Check the Pod has restarted based on the Deployment Config change 50](#_Toc40886616)

[Appendix - Debugging 52](#_Toc40886617)

[Updating the Tekton artifacts in RH Openshift 52](#_Toc40886618)

[Deleting the Tekton artifacts in RH Openshift 52](#_Toc40886619)

[Taskrun for build fails 52](#_Toc40886620)

[Inspect the logs 53](#_Toc40886621)

[Taskrun for deploy fails 55](#_Toc40886622)

[Appendix - Changing to a later version of Tekton 60](#_Toc40886623)

[Tekton Controller 60](#_Toc40886624)

[Tekton Webhook 64](#_Toc40886625)

[NOTE: Leave the Webhook deployment alone i.e. v0.10.1 . I could not get later versions of the image to load. Instruction here just for completeness. 64](#_Toc40886626)

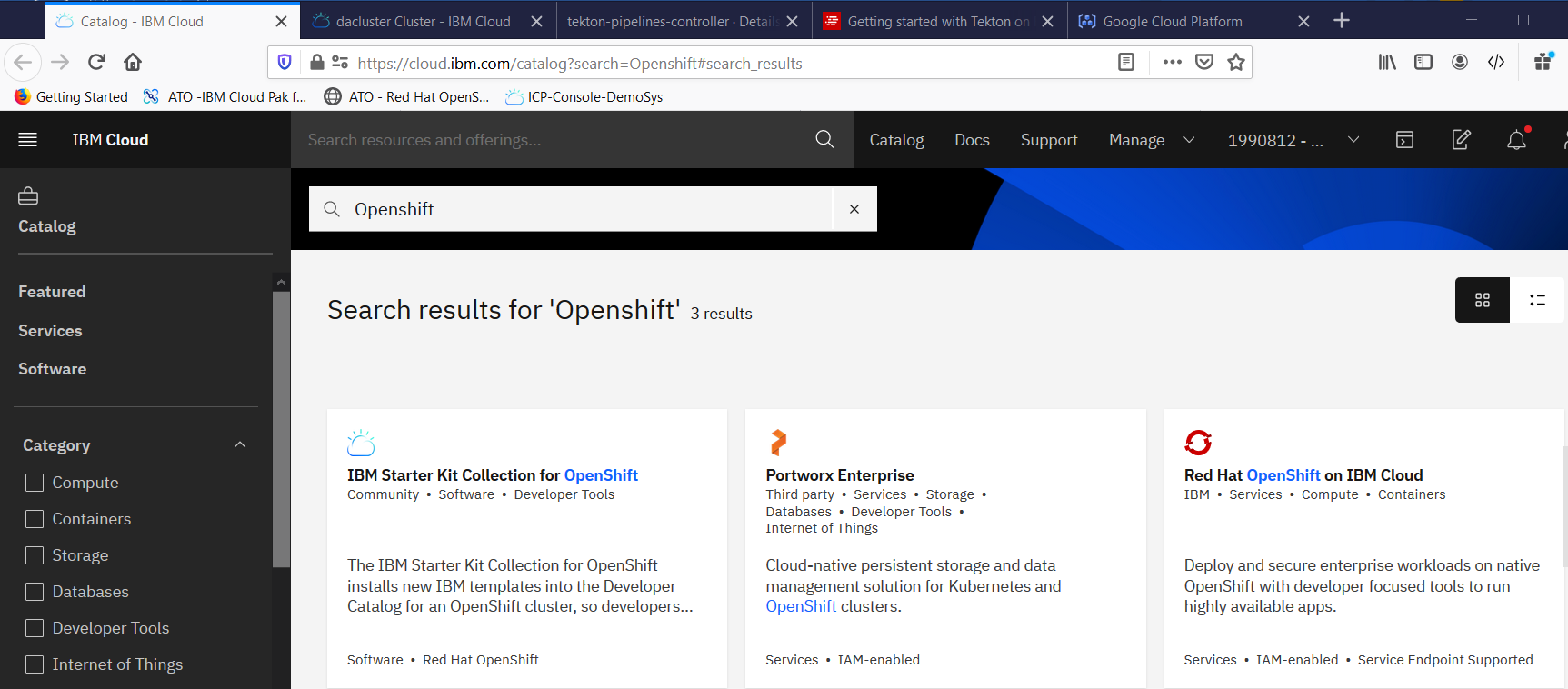
[Test the Tekton Update 64](#_Toc40886627)

## IBM Cloud – provision a Red Hat OpenShift on IBM Cloud service

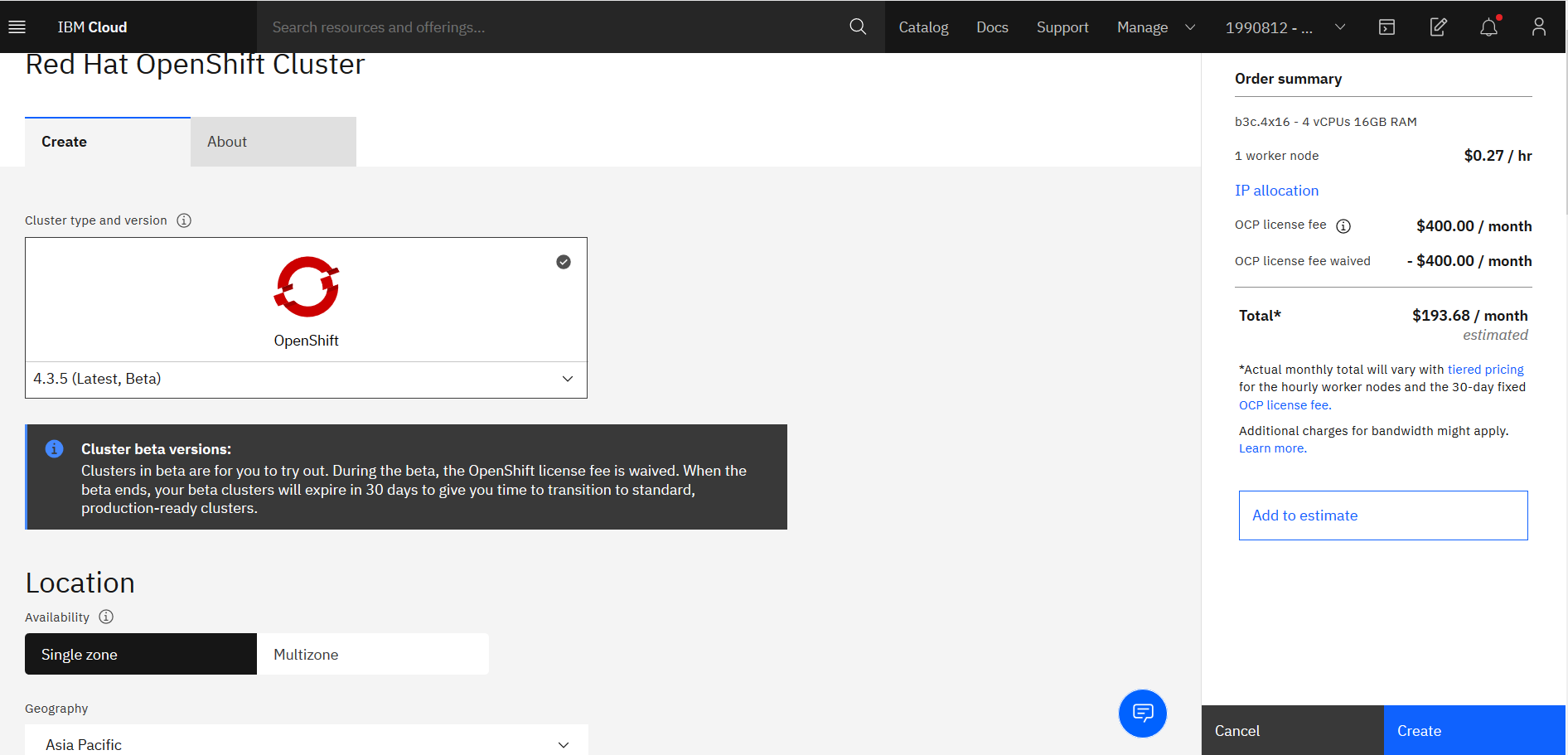
Log into IBM cloud

Goto the catalog and type openshift in the search

Select RedHat Openshift on IBM cloud



Select 4.3.5 Beta as there is no RHOCP license

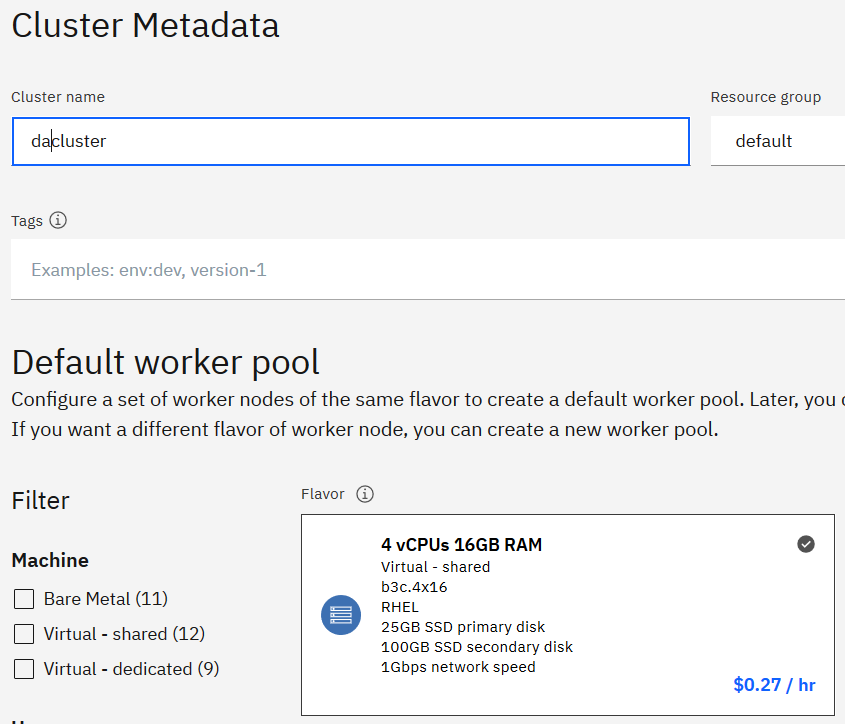


Switch to Single Zone

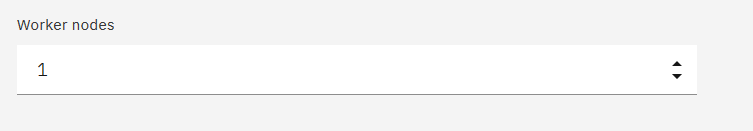
Asia Pacific

Sydney 01

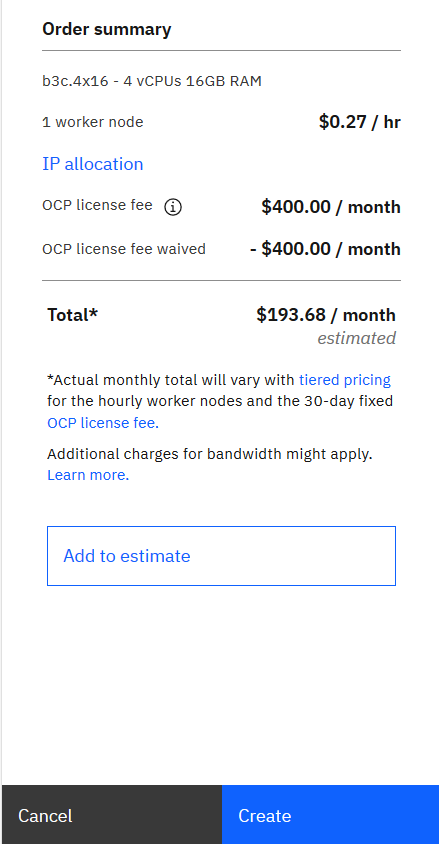
Name your cluster and select the small worker node size



Drop the worker node count from 3 to 1

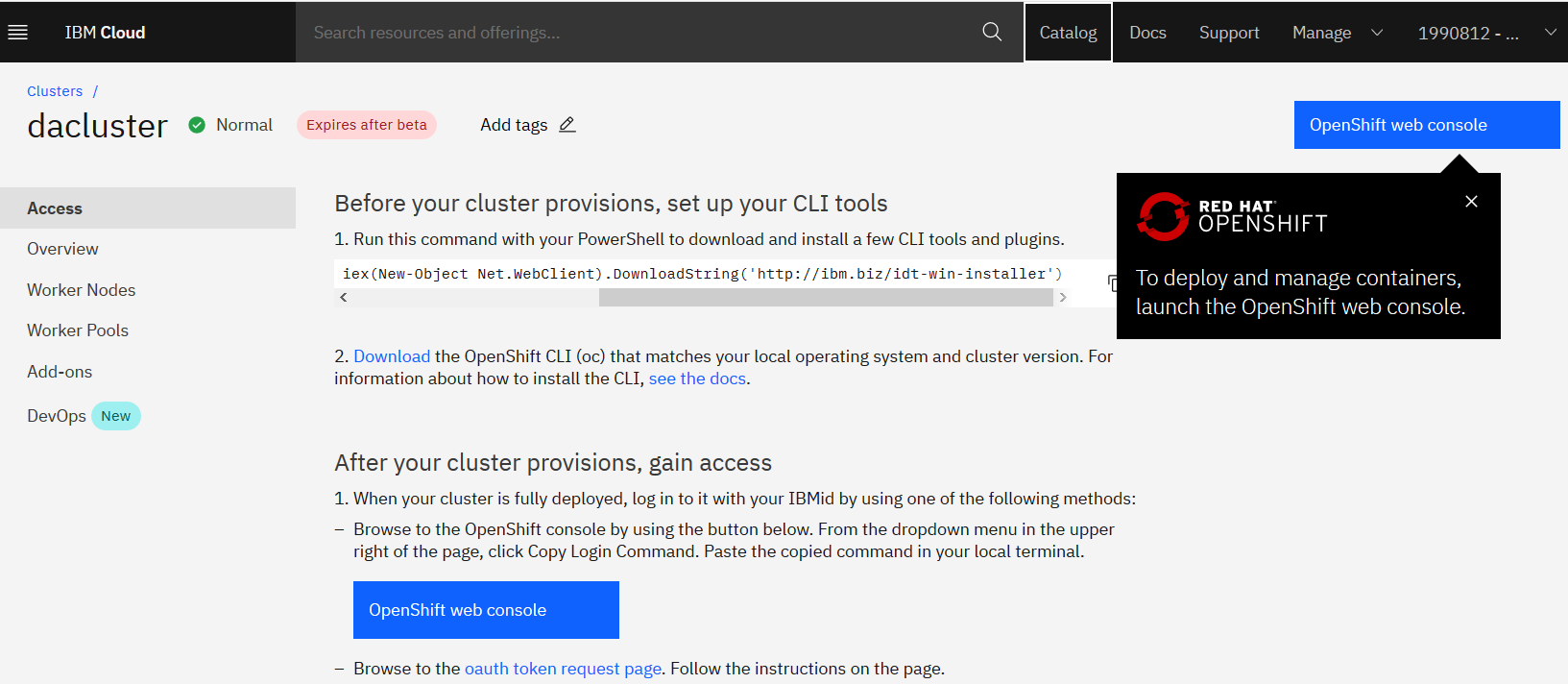


Price for 30 days is low as there is no Openshift license for the Beta version service. **This is point in time.**

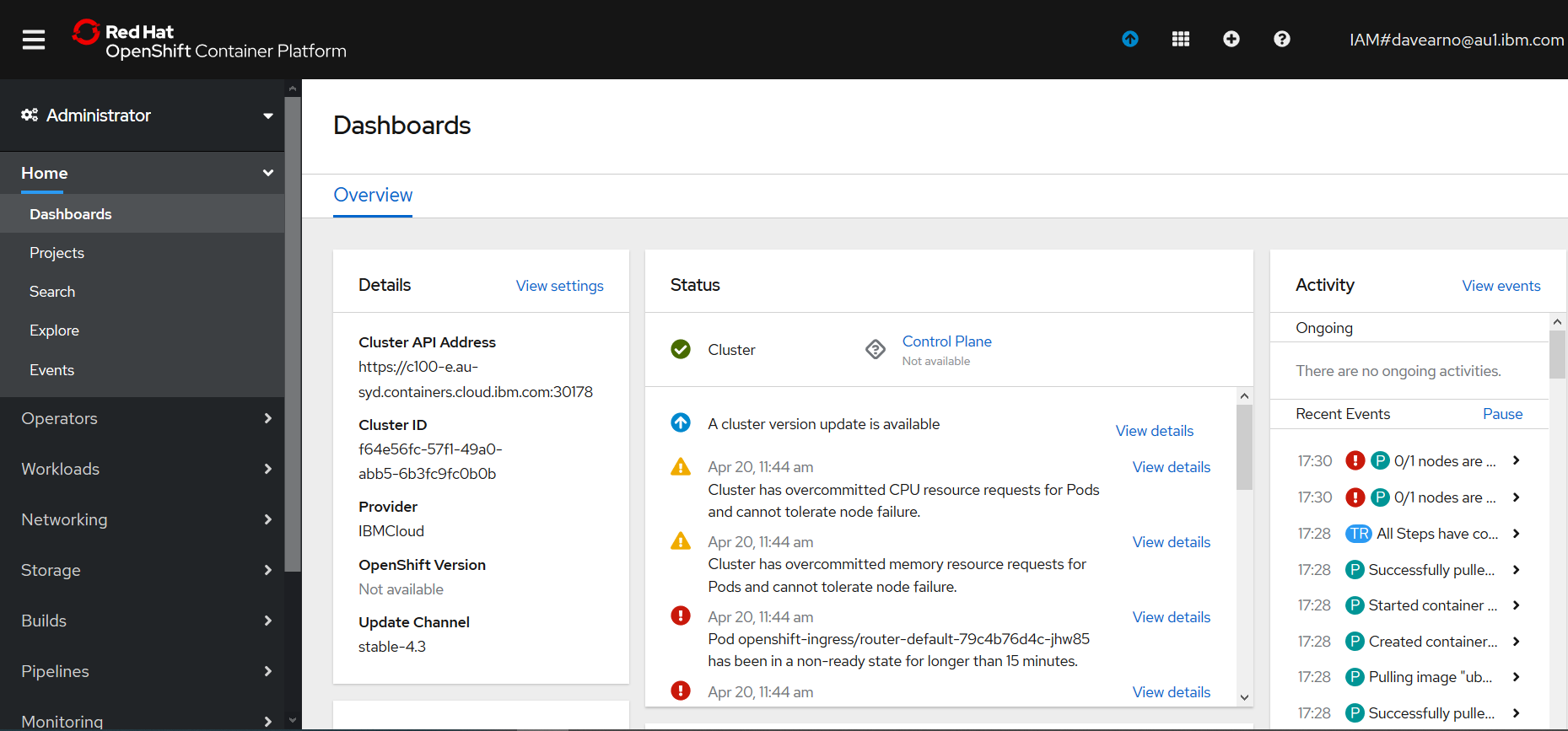


Click create and wait for the service to be provisioned

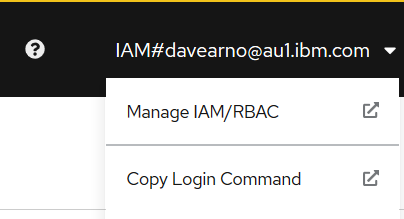
## Accessing the Red Hat OpenShift service

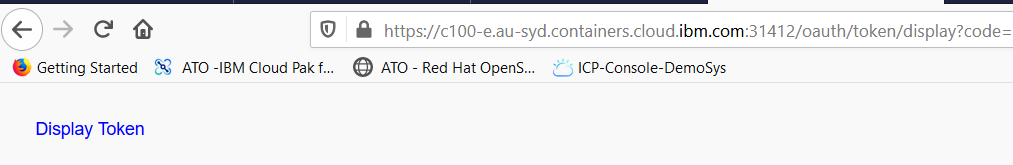


Open the web console



Log in via the command line



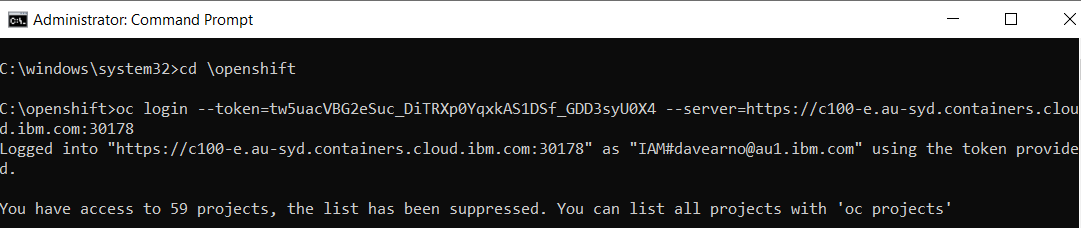


Click on display token



Copy the oc login command

(you need the RH Openshift client installed (the oc.exe))



## Installing Tekton

I followed the instructions in this article

<https://developers.redhat.com/blog/2019/07/19/getting-started-with-tekton-on-red-hat-openshift/>

but also found as a good reference to help with the service account piece

[https://developer.ibm.com/technologies/containers/tutorials/make-continuous-delivery-easier-with-tekton-dashboards/#](https://developer.ibm.com/technologies/containers/tutorials/make-continuous-delivery-easier-with-tekton-dashboards/)

### C:\openshift>oc new-project tekton-pipelines

Now using project "tekton-pipelines" on server "https://c100-e.au-syd.containers.cloud.ibm.com:30178".

You can add applications to this project with the 'new-app' command. For example, try:

oc new-app centos/ruby-25-centos7~https://github.com/sclorg/ruby-ex.git

to build a new example application in Ruby.

### C:\openshift>oc adm policy add-scc-to-user anyuid -z tekton-pipelines-controller

scc "anyuid" added to: ["system:serviceaccount:tekton-pipelines:tekton-pipelines-controller"]

### C:\openshift>oc apply --filename https://storage.googleapis.com/tekton-releases/latest/release.yaml

Warning: oc apply should be used on resource created by either oc create --save-config or oc apply

namespace/tekton-pipelines configured

podsecuritypolicy.policy/tekton-pipelines created

clusterrole.rbac.authorization.k8s.io/tekton-pipelines-admin created

serviceaccount/tekton-pipelines-controller created

clusterrolebinding.rbac.authorization.k8s.io/tekton-pipelines-controller-admin created

customresourcedefinition.apiextensions.k8s.io/clustertasks.tekton.dev created

customresourcedefinition.apiextensions.k8s.io/conditions.tekton.dev created

customresourcedefinition.apiextensions.k8s.io/images.caching.internal.knative.dev created

customresourcedefinition.apiextensions.k8s.io/pipelines.tekton.dev created

customresourcedefinition.apiextensions.k8s.io/pipelineruns.tekton.dev created

customresourcedefinition.apiextensions.k8s.io/pipelineresources.tekton.dev created

customresourcedefinition.apiextensions.k8s.io/tasks.tekton.dev created

customresourcedefinition.apiextensions.k8s.io/taskruns.tekton.dev created

service/tekton-pipelines-controller created

service/tekton-pipelines-webhook created

clusterrole.rbac.authorization.k8s.io/tekton-aggregate-edit created

clusterrole.rbac.authorization.k8s.io/tekton-aggregate-view created

configmap/config-artifact-bucket created

configmap/config-artifact-pvc created

configmap/config-defaults created

configmap/config-logging created

configmap/config-observability created

deployment.apps/tekton-pipelines-controller created

deployment.apps/tekton-pipelines-webhook created

### C:\openshift>oc get pods --namespace tekton-pipelines

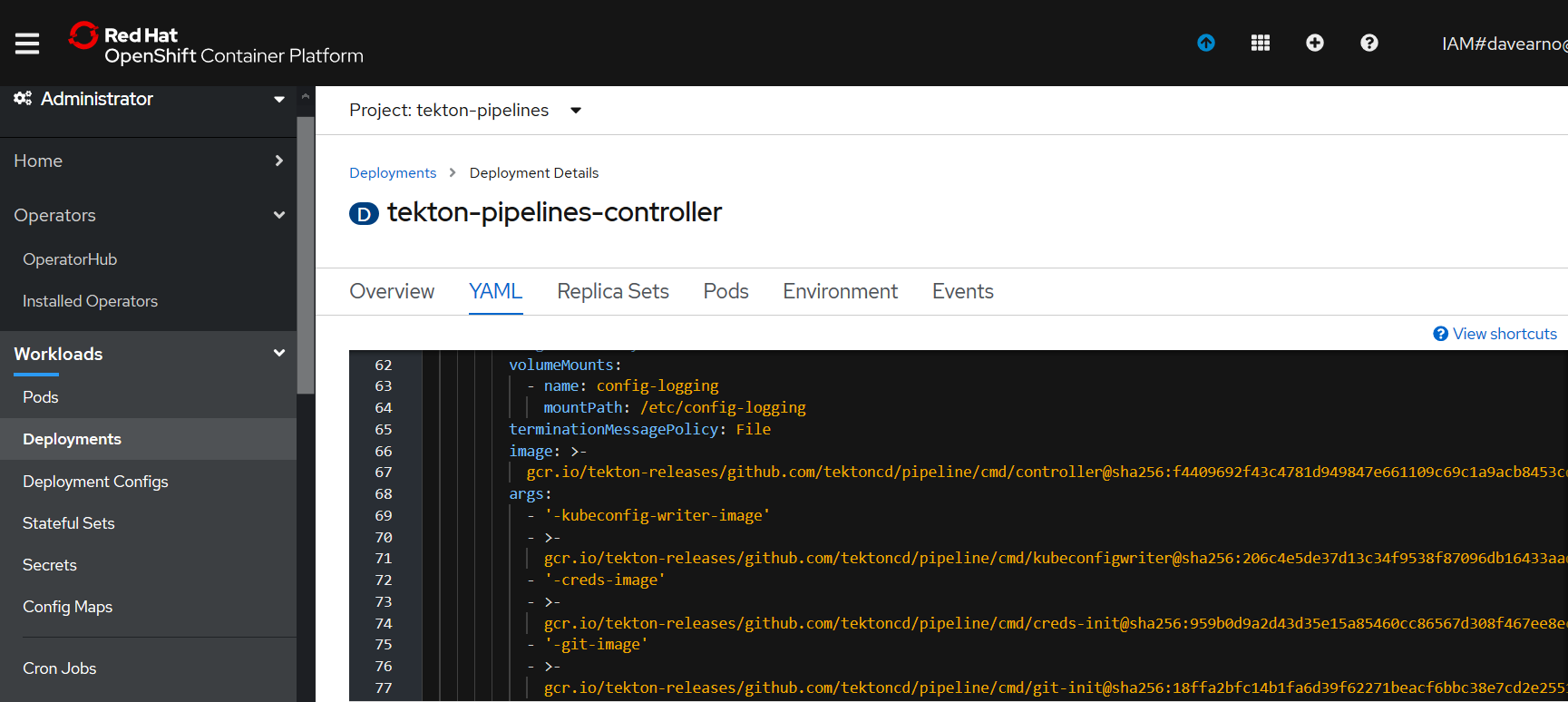
NAME READY STATUS RESTARTS AGE

tekton-pipelines-controller-5b75cdfb95-25gfg 0/1 ImagePullBackOff 0 11m

tekton-pipelines-webhook-b848dcd97-bdhnz 0/1 ImagePullBackOff 0 11m

### Correcting the paths to the image locations

Navigate to the RH Openshift Console->Workloads->Deploymenets->tekton-pipelines-controller



Edit the YAML

The deployment YAML seems to have a mismatch for the actual location of the images on google cloud. The YAML includes **:v0.10.1** this need to be removed as per the example below.

<https://console.cloud.google.com/gcr/images/tekton-releases/GLOBAL/github.com/tektoncd/pipeline/cmd>

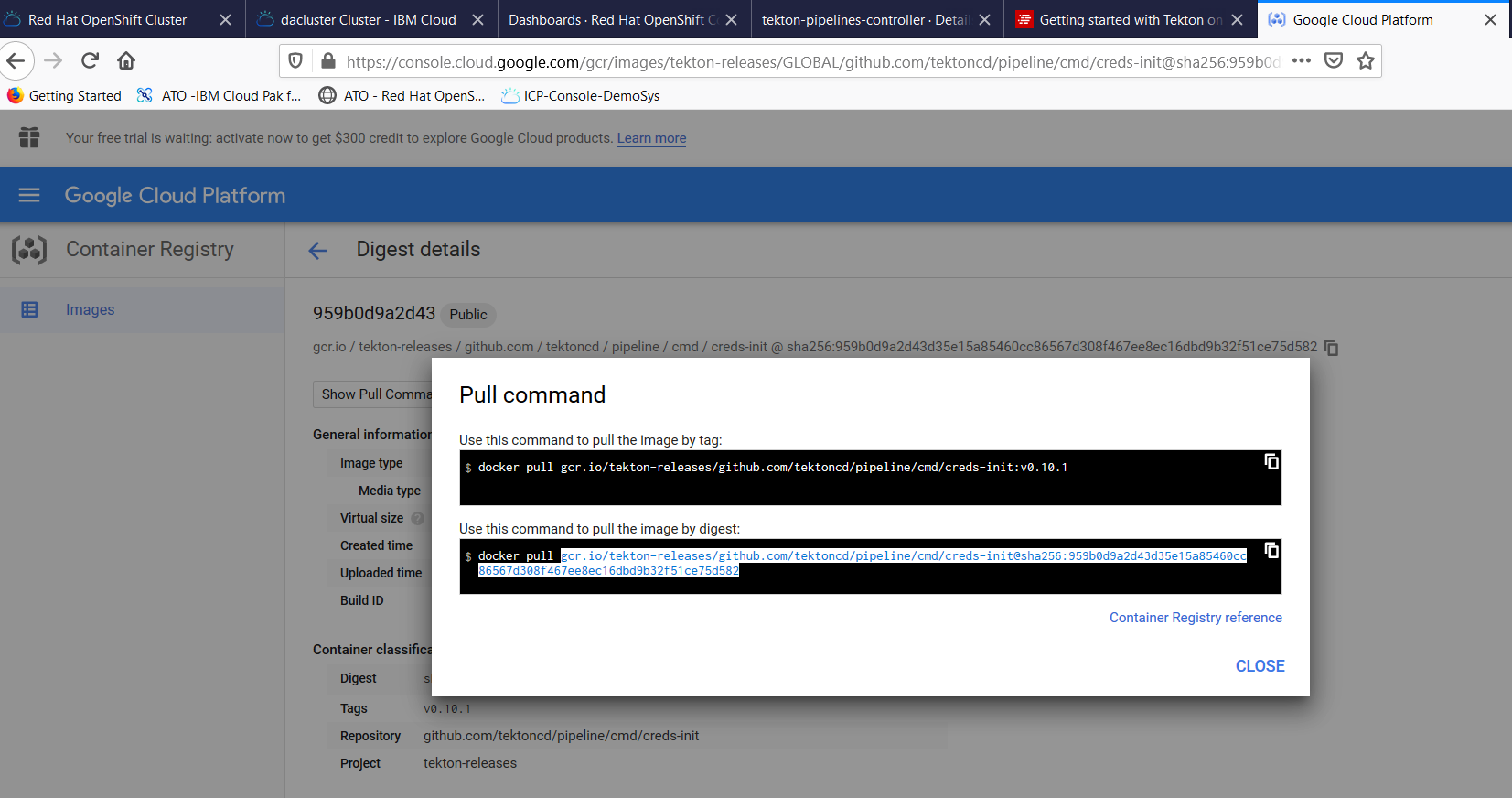
The YAML:

gcr.io/tekton-releases/github.com/tektoncd/pipeline/cmd/creds-init**:v0.10.1**@sha256:959b0d9a2d43d35e15a85460cc86567d308f467ee8ec16dbd9b32f51ce75d582

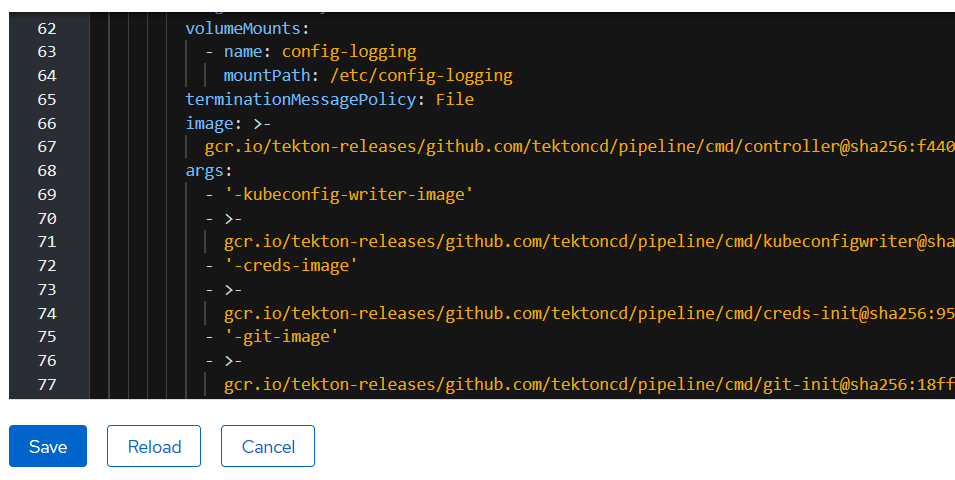
The location of the images on google:

[gcr.io/tekton-releases/github.com/tektoncd/pipeline/cmd/creds-init@sha256:959b0d9a2d43d35e15a85460cc86567d308f467ee8ec16dbd9b32f51ce75d582](mailto:gcr.io/tekton-releases/github.com/tektoncd/pipeline/cmd/creds-init@sha256:959b0d9a2d43d35e15a85460cc86567d308f467ee8ec16dbd9b32f51ce75d582)

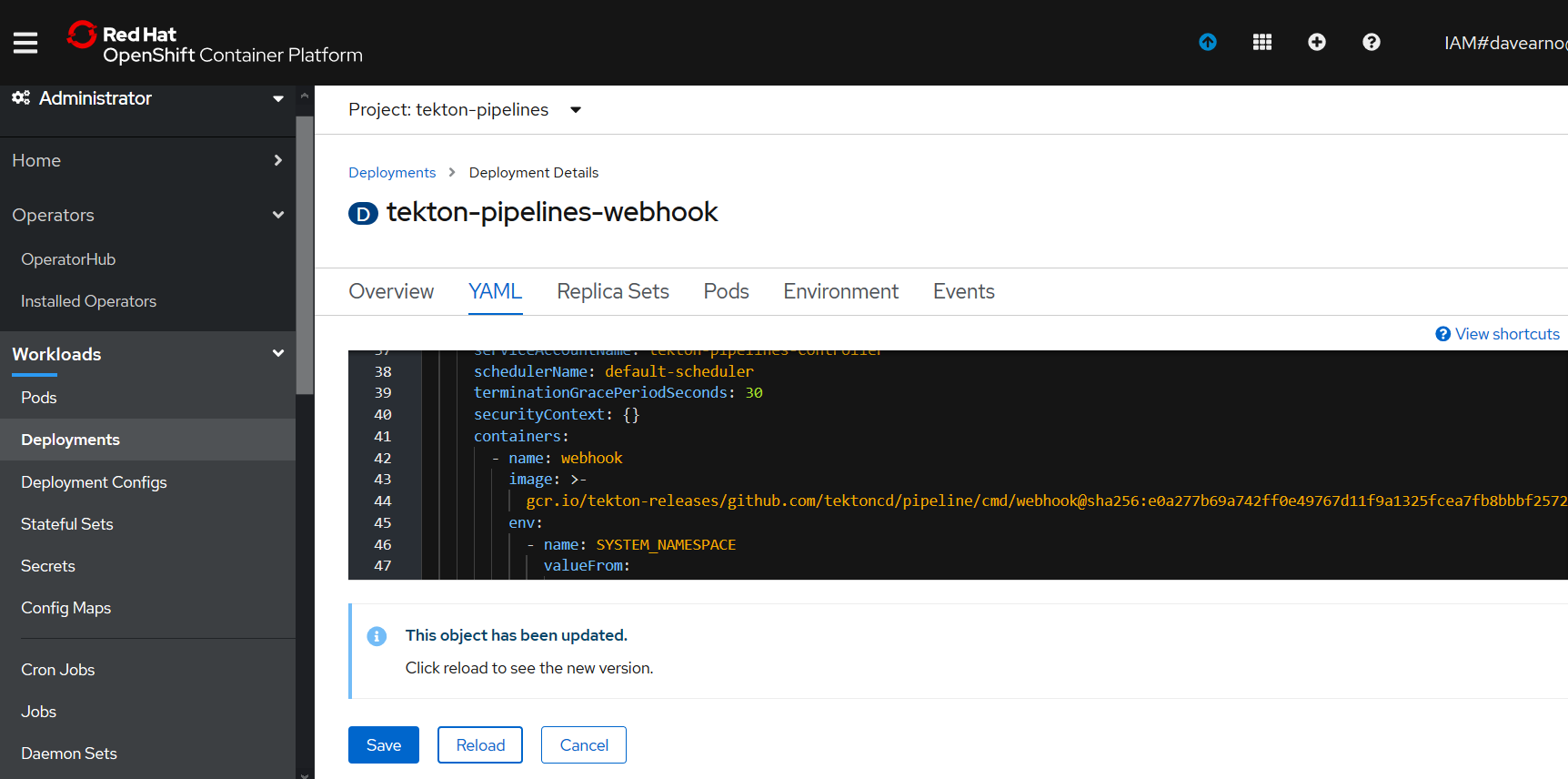
you can navigate to the image and copy and paste from the pull command.



Correct the YAML and hit Save and Reload.

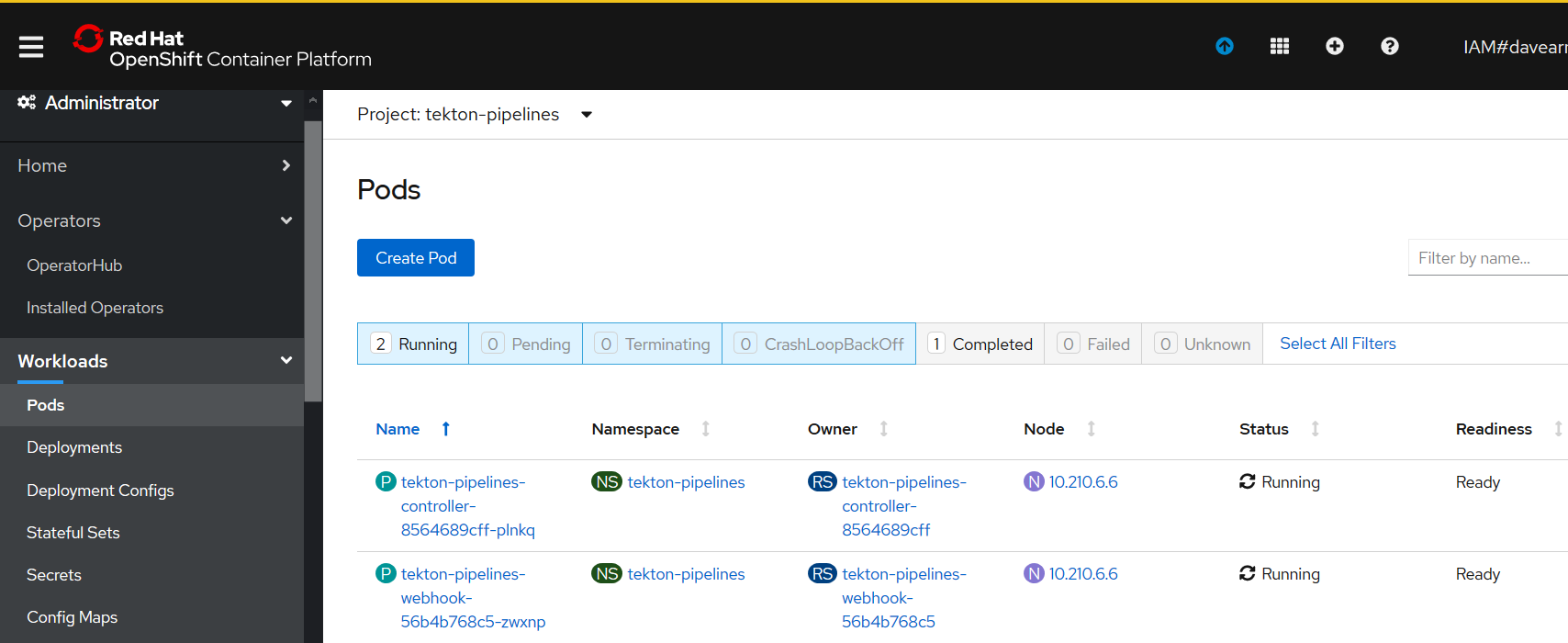


Repeat for the WebHook deployment YAML



[gcr.io/tekton-releases/github.com/tektoncd/pipeline/cmd/webhook@sha256:e0a277b69a742ff0e49767d11f9a1325fcea7fb8bbbf2572af9d49116cbb2385](mailto:gcr.io/tekton-releases/github.com/tektoncd/pipeline/cmd/webhook@sha256:e0a277b69a742ff0e49767d11f9a1325fcea7fb8bbbf2572af9d49116cbb2385)

The Pods should now start up correctly



### C:\openshift>oc get pods --namespace tekton-pipelines

NAME READY STATUS RESTARTS AGE

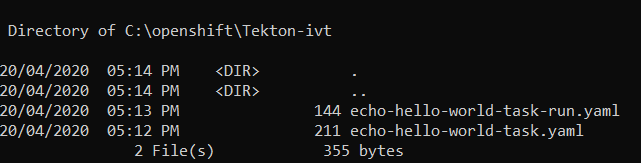
tekton-pipelines-controller-8564689cff-plnkq 1/1 Running 0 50m

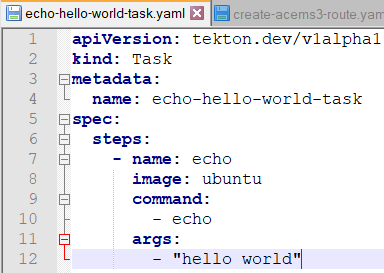
tekton-pipelines-webhook-56b4b768c5-zwxnp 1/1 Running 0 3m12s

## Test Tekton with an Echo example

As we are operating in the tekton-pipelines namespace (where tekton was installed) the default service account will be used as such you don’t need to set up a service account for pipelines at this stage.

Create two YAML files





apiVersion: tekton.dev/v1alpha1

kind: Task

metadata:

name: echo-hello-world-task

spec:

steps:

- name: echo

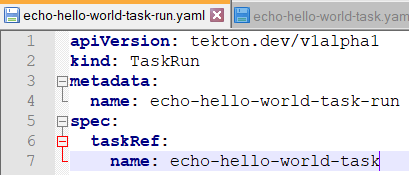
image: ubuntu

command:

- echo

args:

- "hello world"



apiVersion: tekton.dev/v1alpha1

kind: TaskRun

metadata:

name: echo-hello-world-task-run

spec:

taskRef:

name: echo-hello-world-task

### C:\openshift>oc apply -f Tekton-ivt\echo-hello-world-task.yaml

task.tekton.dev/echo-hello-world-task created

### C:\openshift>oc apply -f Tekton-ivt\echo-hello-world-task-run.yaml

taskrun.tekton.dev/echo-hello-world-task-run created

### C:\openshift>oc get taskruns/echo-hello-world-task-run -o yaml

Check the results.

apiVersion: tekton.dev/v1alpha1

kind: TaskRun

metadata:

annotations:

kubectl.kubernetes.io/last-applied-configuration: |

{"apiVersion":"tekton.dev/v1alpha1","kind":"Task","metadata":{"annotations":{},"name":"echo-hello-world-task","namespace":"tekton-pipelines"},"spec":{"steps":[{"args":["hello world"],"command":["echo"],"image":"ubuntu","name":"echo"}]}}

tekton.dev/release: devel

creationTimestamp: 2020-04-20T07:14:58Z

generation: 1

labels:

app.kubernetes.io/managed-by: tekton-pipelines

tekton.dev/task: echo-hello-world-task

name: echo-hello-world-task-run

namespace: tekton-pipelines

resourceVersion: "108232"

selfLink: /apis/tekton.dev/v1alpha1/namespaces/tekton-pipelines/taskruns/echo-hello-world-task-run

uid: 54bf7e0d-b1de-4989-914e-06444e067242

spec:

inputs: {}

outputs: {}

serviceAccountName: ""

taskRef:

kind: Task

name: echo-hello-world-task

timeout: 1h0m0s

status:

completionTime: 2020-04-20T07:28:25Z

conditions:

- lastTransitionTime: 2020-04-20T07:28:25Z

message: All Steps have completed executing

reason: Succeeded

status: "True"

type: Succeeded

podName: echo-hello-world-task-run-pod-pz6vx

startTime: 2020-04-20T07:14:58Z

steps:

- container: step-echo

imageID: docker.io/library/ubuntu@sha256:bec5a2727be7fff3d308193cfde3491f8fba1a2ba392b7546b43a051853a341d

name: echo

terminated:

containerID: cri-o://f04dd508f82092cb3978f14cfc0bedc64aa110082fc557790790577cf265d025

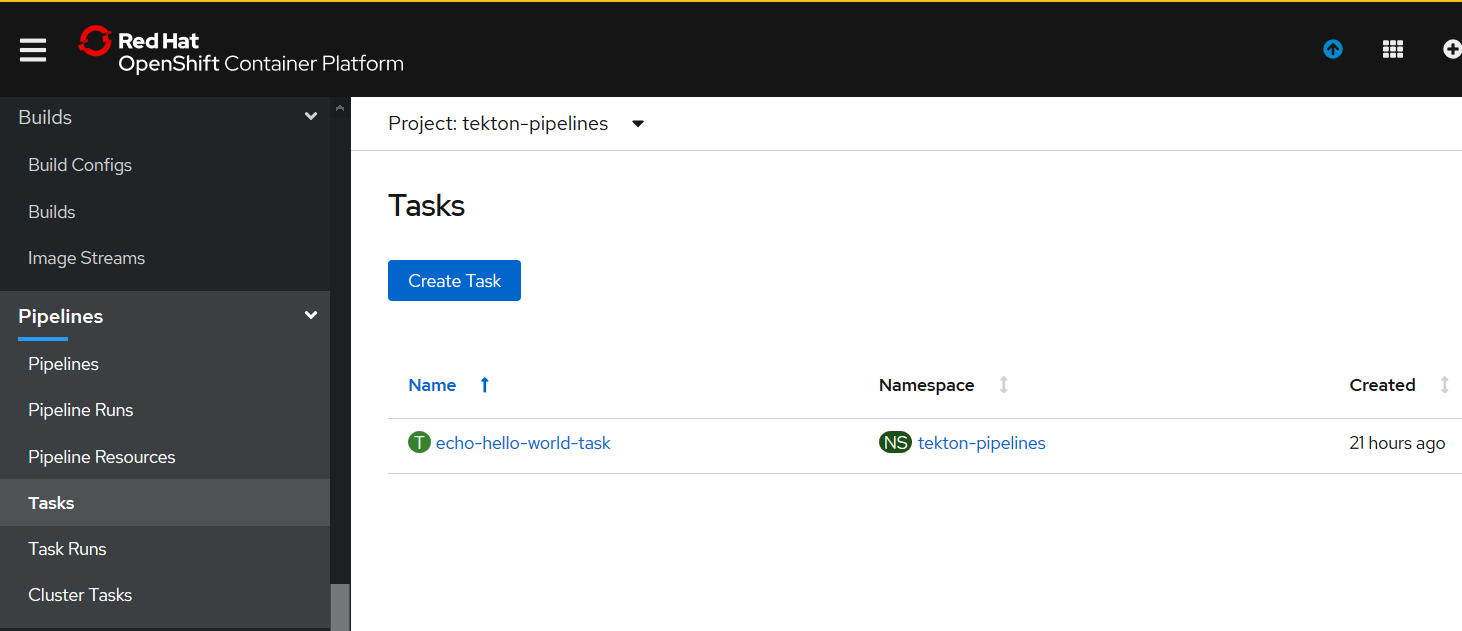
exitCode: 0

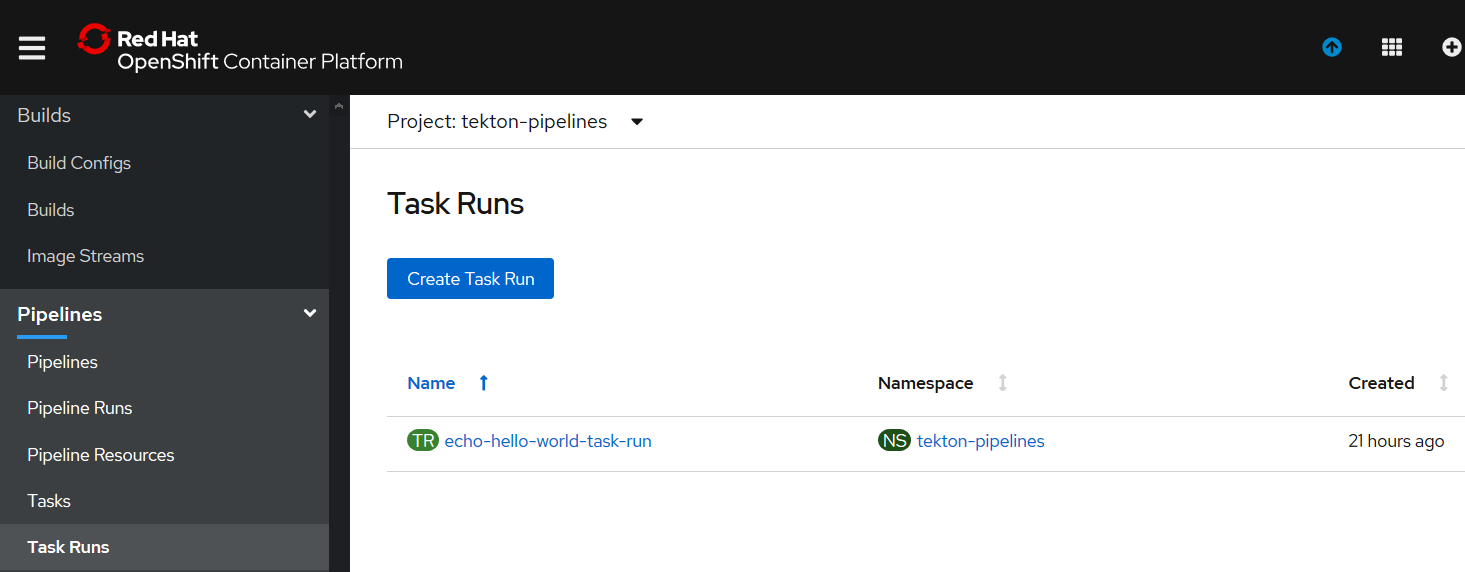
finishedAt: 2020-04-20T07:28:24Z

reason: Completed

startedAt: 2020-04-20T07:28:24Z

### OpenShift Console Pipelines





## Initial Build and run MQ (TLS) from certified base image on RH OpenShift

### Login in to the openshift cluster from the command line.

### oc project da-build-project

### oc create -f c:\openshift\data\build-mq-custom.yaml

**Here is the YAML**

apiVersion: v1

kind: ImageStream

metadata:

name: ibm-mqadvanced-server-integration

spec:

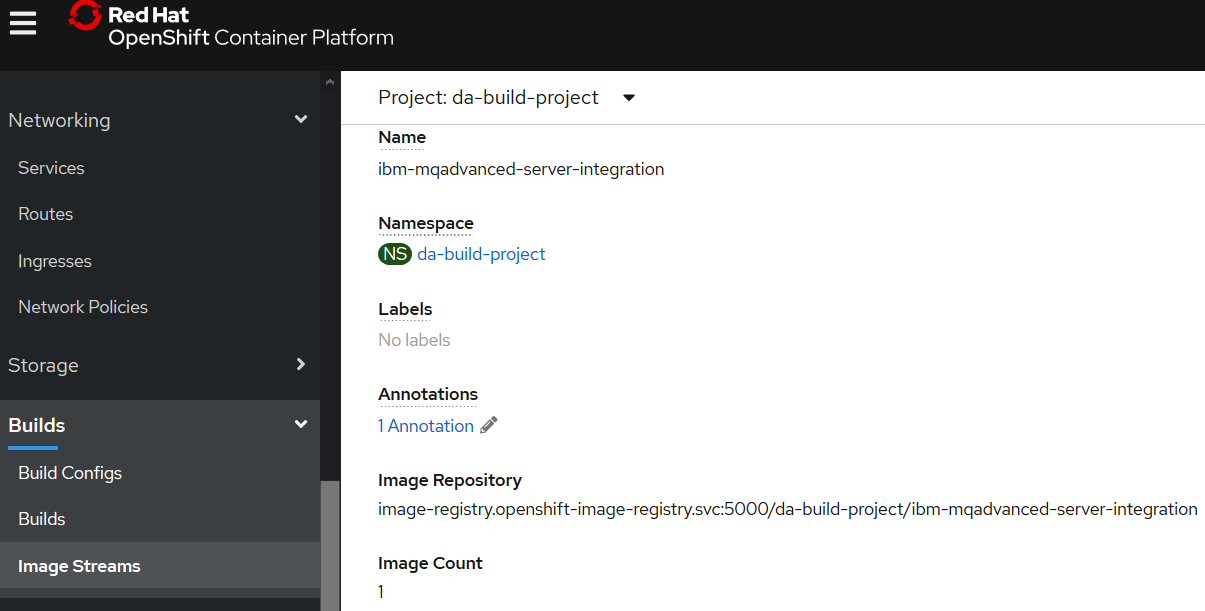
tags:

- name: "9.1.3.0-r4-amd64"

from:

kind: DockerImage

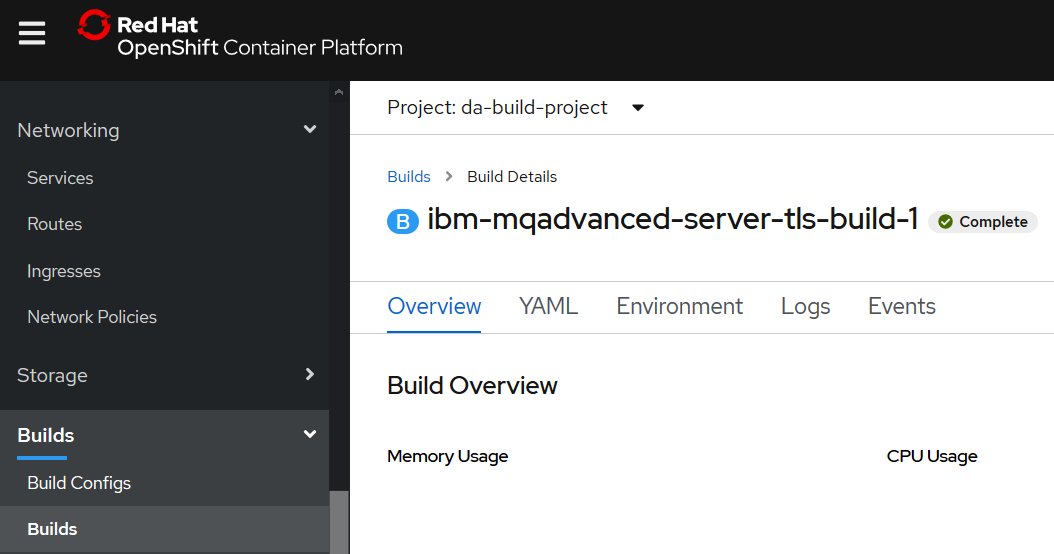
name: davexacom/ibm-mqadvanced-server-integration:9.1.3.0-r4-amd64

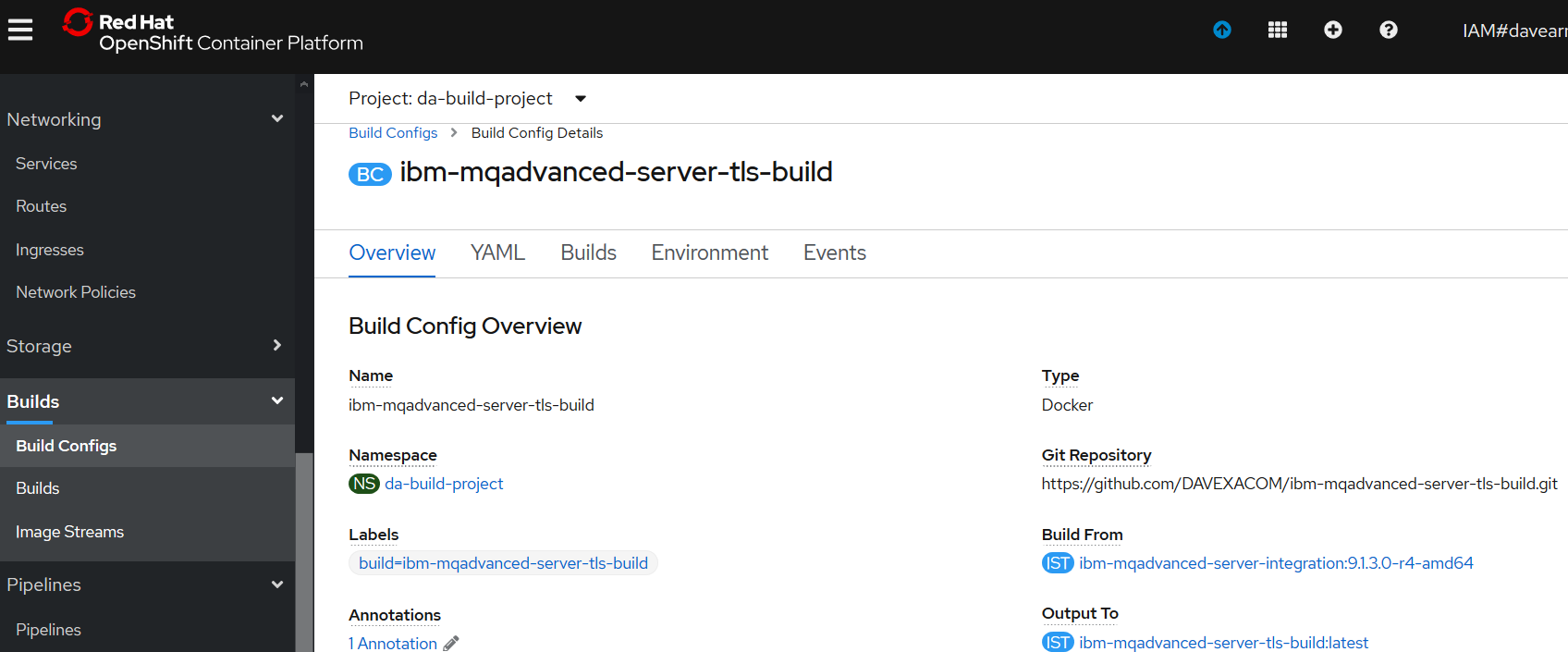


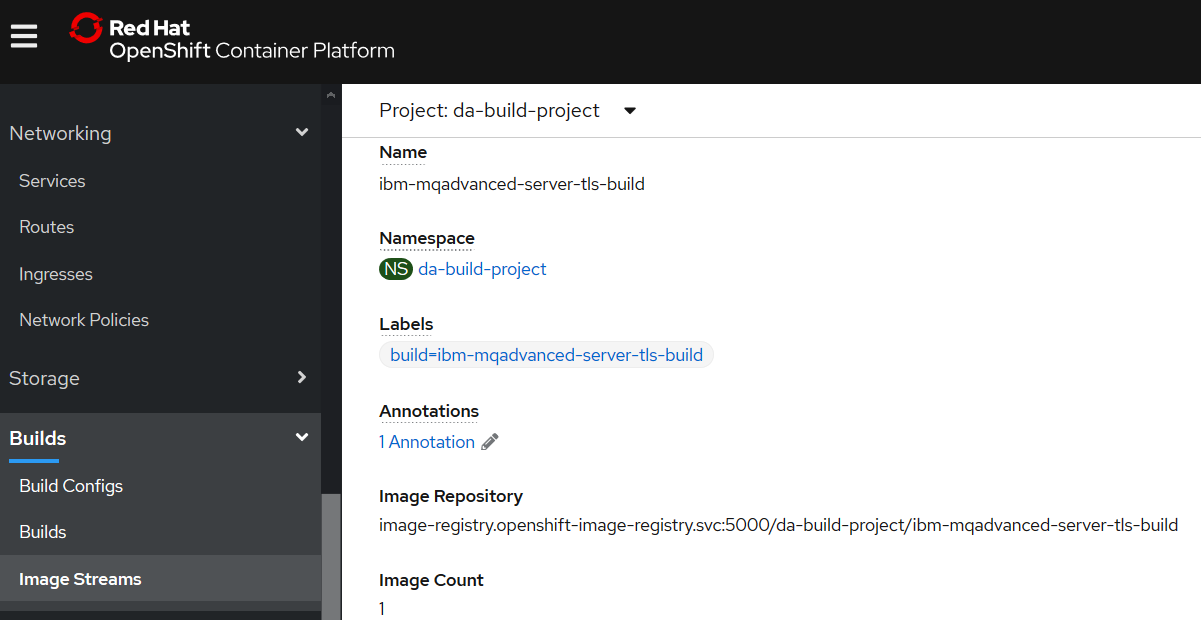


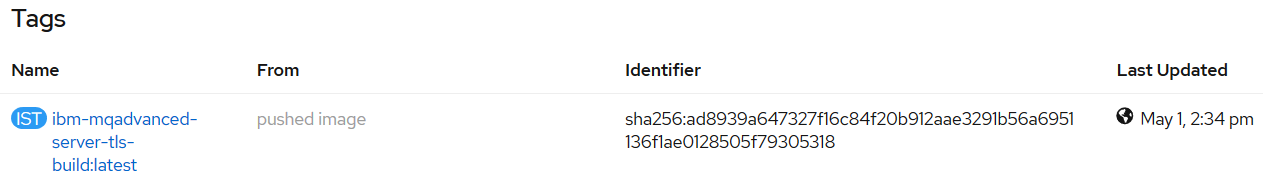
### oc adm policy add-scc-to-user anyuid -z default

### oc new-build <https://github.com/DAVEXACOM/ibm-mqadvanced-server-tls-build.git>







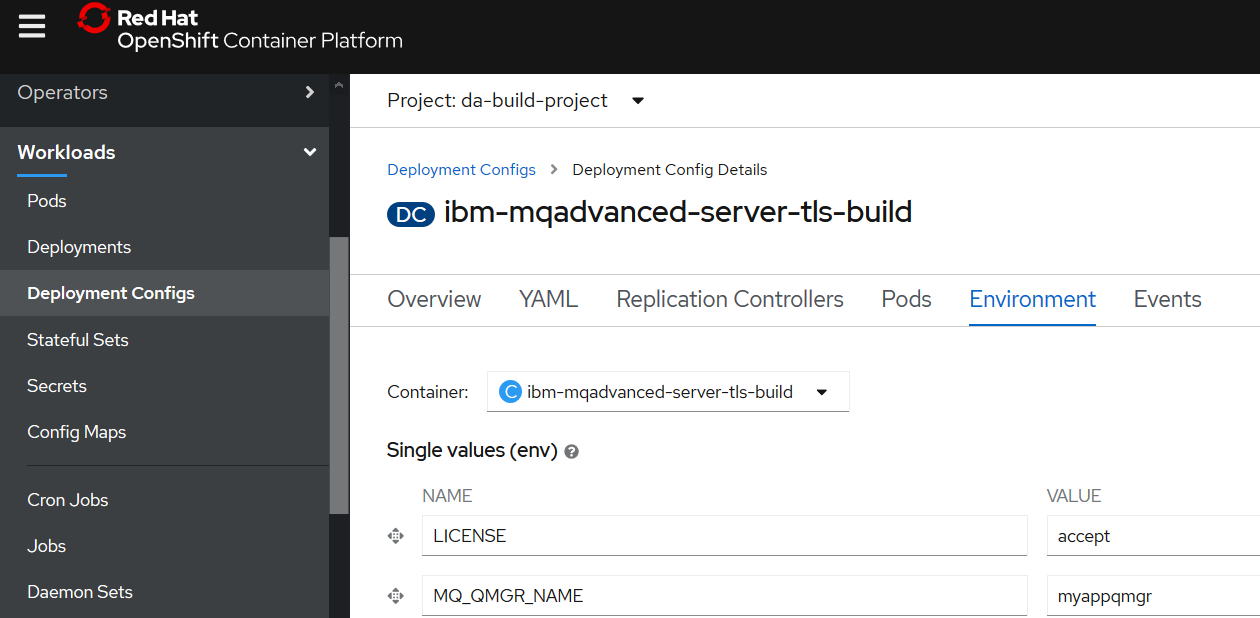


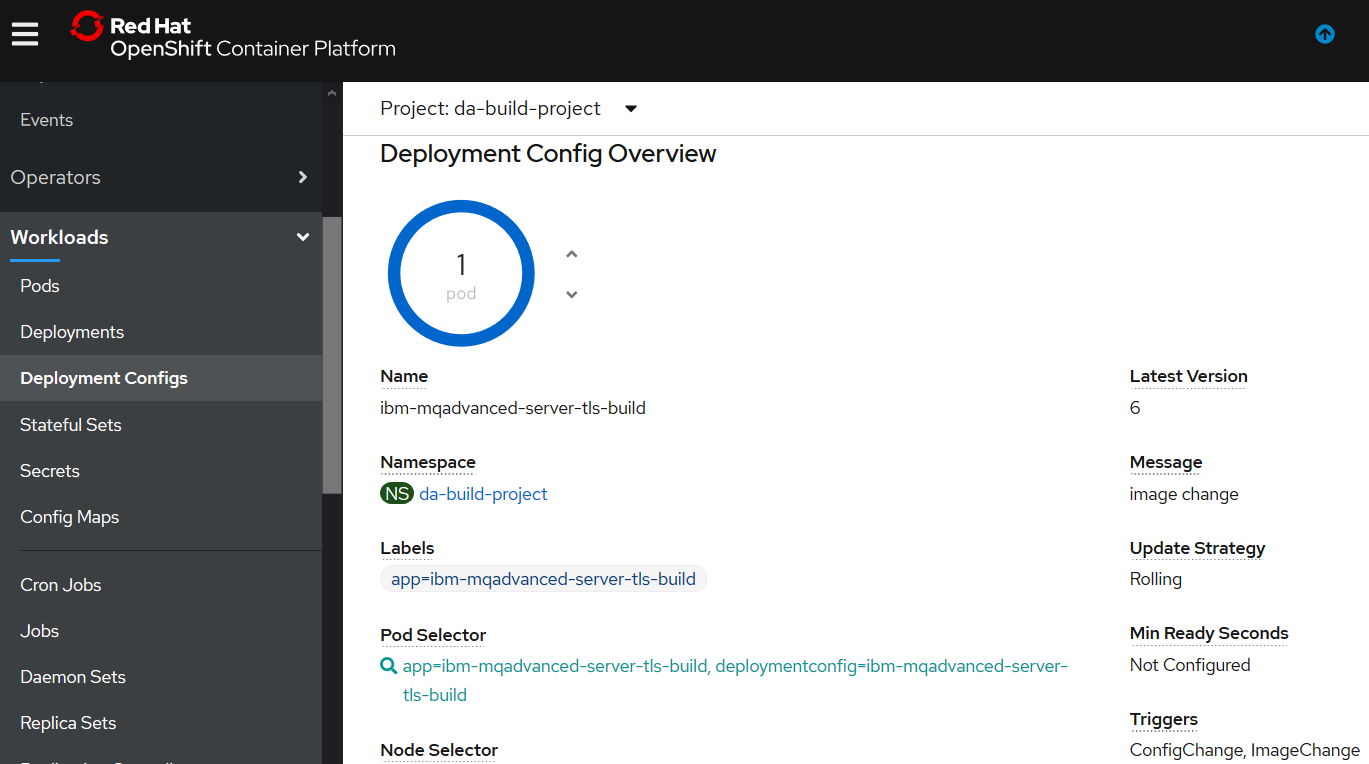
Choose which new-app command

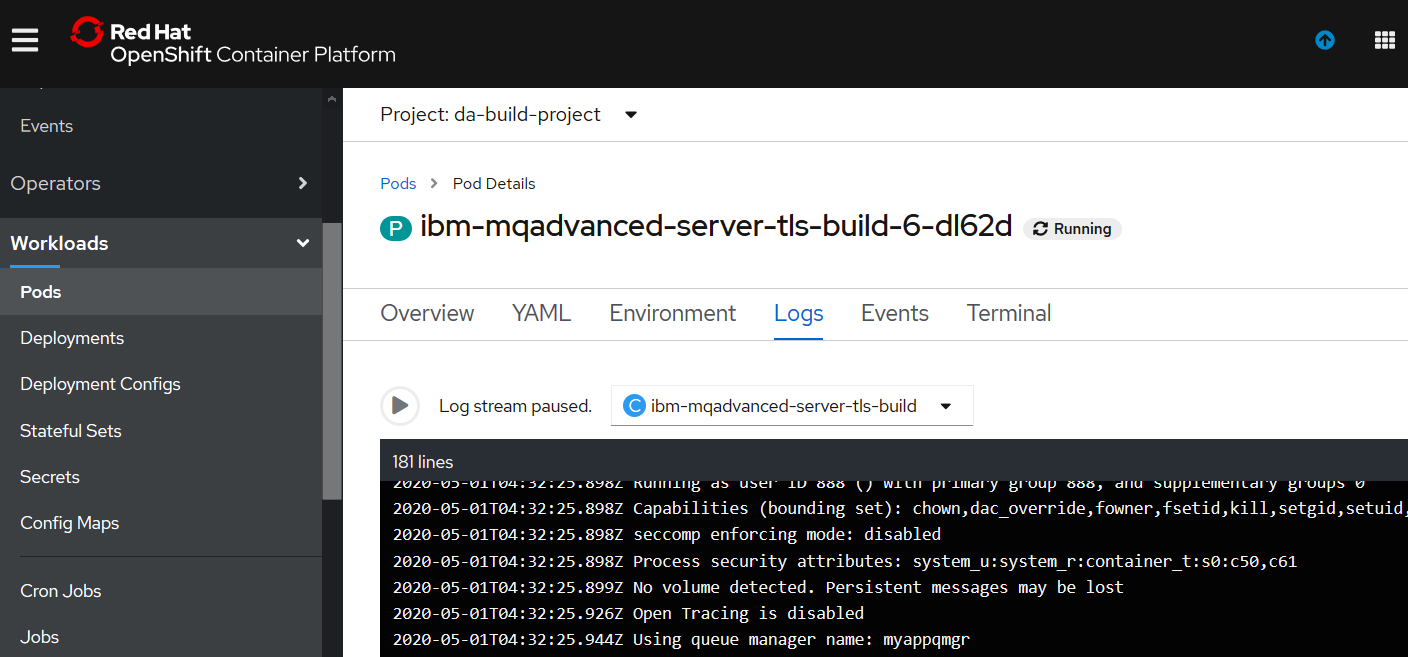
### oc new-app ibm-mqadvanced-server-tls-build --env LICENSE=accept

Note: You could use - -env **MQ\_QMGR\_NAME=myappqmgr to set a queue manager name**

### oc new-app ibm-mqadvanced-server-tls-build --env LICENSE=accept --env MQ\_QMGR\_NAME=myappqmgr

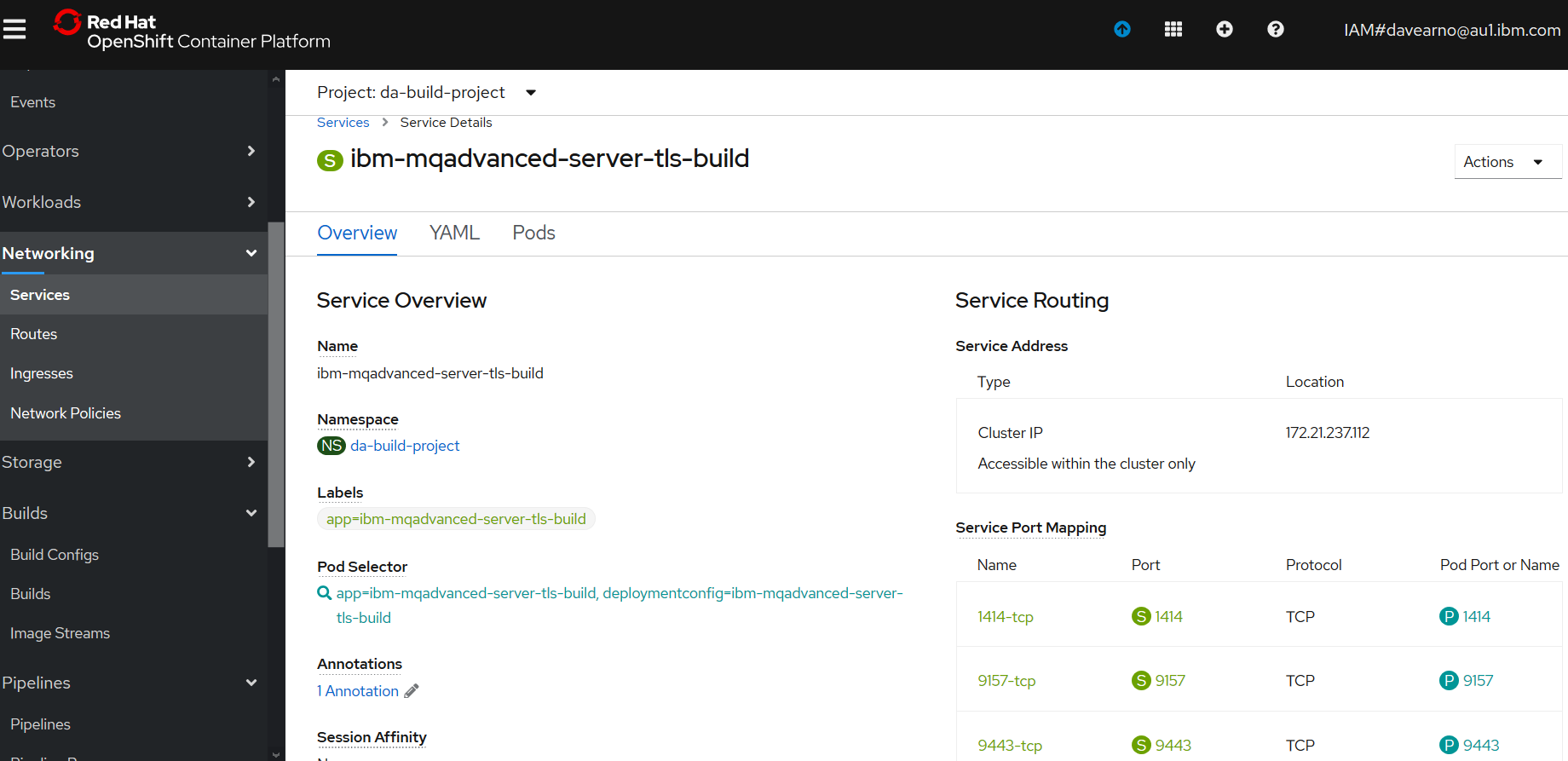






Expose the service as a route

### oc expose svc/ibm-mqadvanced-server-tls-build



Check the status with oc status

### oc status

Create the additional TLS route to complete MQ connectivity

### oc create -f c:\openshift\data\create-mq-route.yaml

**Here is the YAML**

kind: Route

apiVersion: route.openshift.io/v1

metadata:

name: tls-tlsprqm1p

namespace: da-build-project

labels:

app: ibm-mqadvanced-server-tls-build

spec:

host: tlsprqm12e-svrconn.chl.mq.ibm.com

subdomain: ''

to:

kind: Service

name: ibm-mqadvanced-server-tls-build

weight: 100

port:

targetPort: 1414-tcp

tls:

termination: passthrough

wildcardPolicy: None

### oc get routes

**You should have 2 for MQ TLS**

NAME HOST/PORT PATH SERVICES PORT TERMINATION WILDCARD

ibm-mqadvanced-server-tls-build ibm-mqadvanced-server-tls-build-da-build-project.dacluster-0511fc923b97b25b240630e79d362861-0000.au-syd.containers.appdomain.cloud ibm-mqadvanced-server-tls-build 1414-tcp None

tls-tlsprqm1p tlsprqm12e-svrconn.chl.mq.ibm.com ibm-mqadvanced-server-tls-build 1414-tcp passthrough None

Get the CONNANE from the host/port of the route above

Get the queue manager name from the logs workload->pods unless you used

- -env **MQ\_QMGR\_NAME=myappqmgr to set a queue manager name**

### Create CLNTCONN Channel definition

Use your queue manager name and conname from the get routes command.

**Here is the MQSC, create a file called custom.mqsc**

DEFINE CHANNEL(TLSPRQM1.SVRCONN) +

CHLTYPE(CLNTCONN) +

TRPTYPE(TCP) +

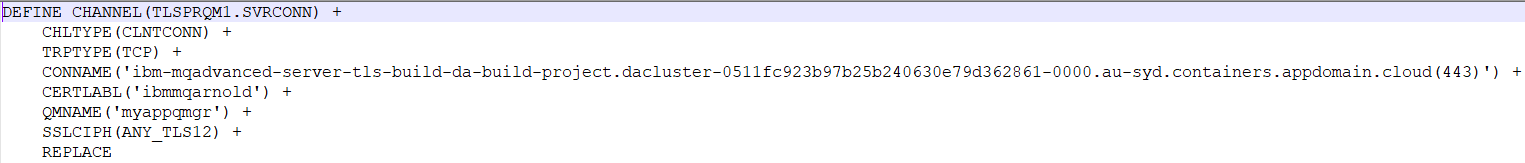
CONNAME(ibm-mqadvanced-server-tls-build-da-build-project.dacluster-0511fc923b97b25b240630e79d362861-0000.au-syd.containers.appdomain.cloud(443)') +

CERTLABL('ibmmqarnold') +

QMNAME('myappqmgr') +

SSLCIPH(ANY\_TLS12) +

REPLACE

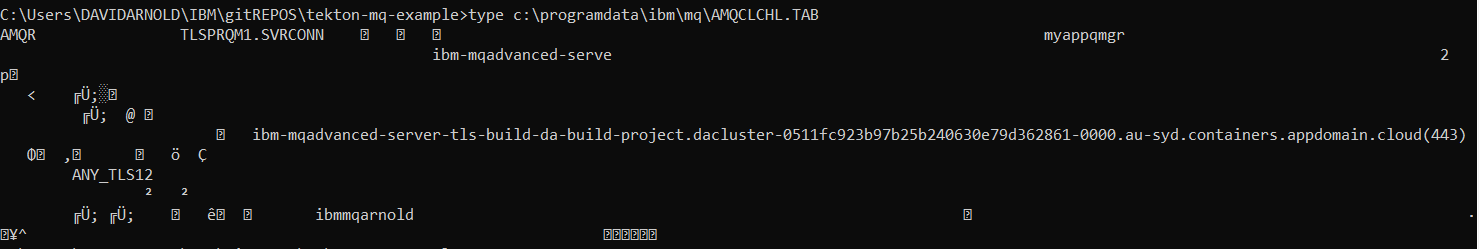


### runmqsc -n < CUSTOMIMAGE.mqsc

### 

### Check the AMQCLCHL.TAB contents

type c:\programdata\ibm\mq\AMQCLCHL.TAB



### Setting up the local KeyStore for TLS client connection

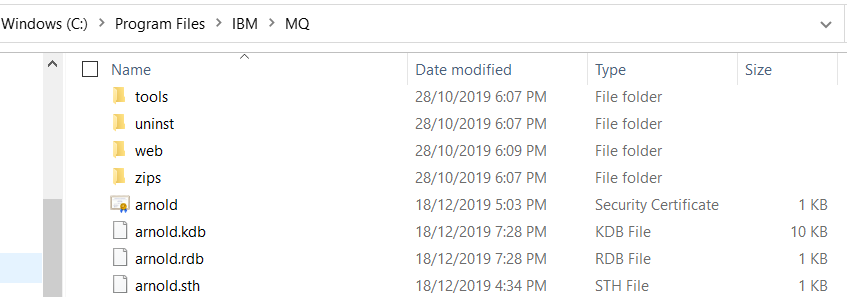
[note: connecting over TLS so need to set the keystore location]

You will need the client side (your laptop with MQ on it) keystore files that match those baked into the image.

Get them from Github below and place them in the directory pointed to be the MQSSLKEYR environment variable.

<https://github.com/DAVEXACOM/Exploring-ICP4i-RHOS/tree/master/MQ%20TLS%20and%20Custom%20Layer%20Image-311and42/clientside%20TLS%20files>

Place the files in the default directory



### set MQSSLKEYR=C:\Program Files\IBM\MQ\arnold

run client connect runmqsc and add a queue

### runmqsc -c mqappqmgr

5724-H72 (C) Copyright IBM Corp. 1994, 2019.

Starting MQSC for queue manager ibmmqadvancedservertlsbuild147zgm.

DIS QL(M\*)

AMQ8409I: Display Queue details.

QUEUE(MY.LOCAL.Q1) TYPE(QLOCAL)

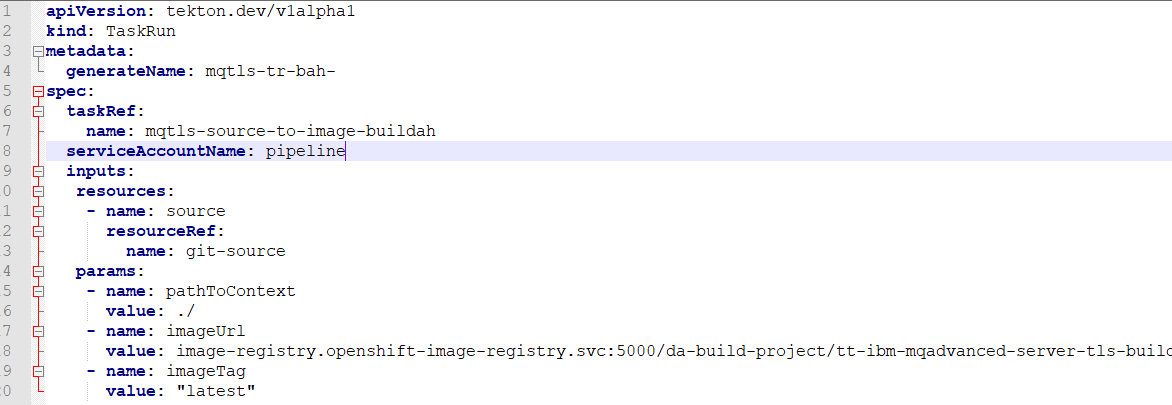
end

## Tekton Pipeline for IBM MQ (tls enabled)

### Create the Pipeline Service Account

We are now going to create and use tekton artifacts in the da-build-project namespace. The default service account for this namespace will not have access to run tekton. So we will create a service account called “pipeline” and this service account name will be used in the “serviceaccountname” parameter in the Spec: for Taskruns and Pipelineruns

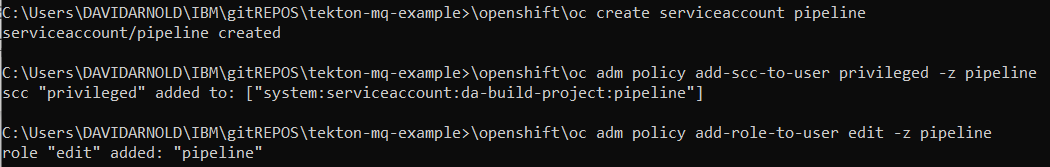
For example:



c:\openshift\oc create serviceaccount pipeline

c:\openshift\oc adm policy add-scc-to-user privileged -z pipeline

c:\openshift\oc adm policy add-role-to-user edit -z pipeline



### Create the (non executing)Tekton artifacts in RH Openshift

#### Resources

c:\openshift\oc apply -f c:\users\DAVIDARNOLD\IBM\gitREPOS\tekton-mq-example\tekton\resources\mqtls-git.yaml

#### Tasks

c:\openshift\oc apply -f c:\users\DAVIDARNOLD\IBM\gitREPOS\tekton-mq-example\tekton\tasks\mqtls-source-to-image-buildah.yaml

c:\openshift\oc apply -f c:\users\DAVIDARNOLD\IBM\gitREPOS\tekton-mq-example\tekton\tasks\mqtls-deploy-using-kubectl.yaml

#### Pipeline

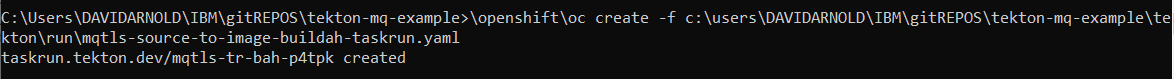
c:\openshift\oc apply -f c:\users\DAVIDARNOLD\IBM\gitREPOS\tekton-mq-example\tekton\pipeline\mqtls-build-and-deploy-pipeline-buildah.yaml

### Create and test the Tekton Taskrun artifacts in RH Openshift

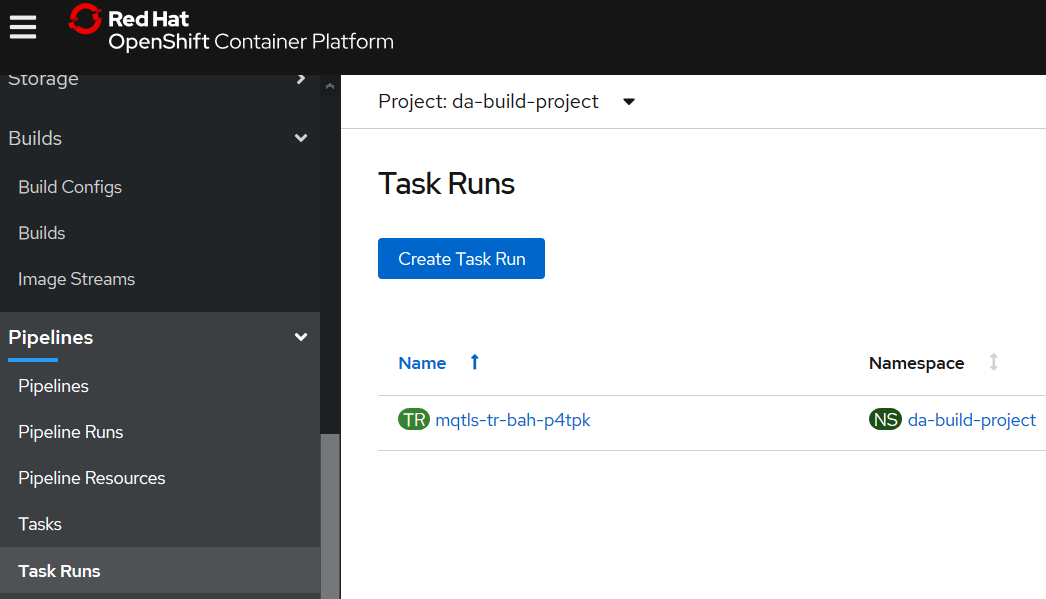
#### Taskrun for Buildah Build task

c:\openshift\oc create -f c:\users\DAVIDARNOLD\IBM\gitREPOS\tekton-mq-example\tekton\run\mqtls-source-to-image-buildah-taskrun.yaml

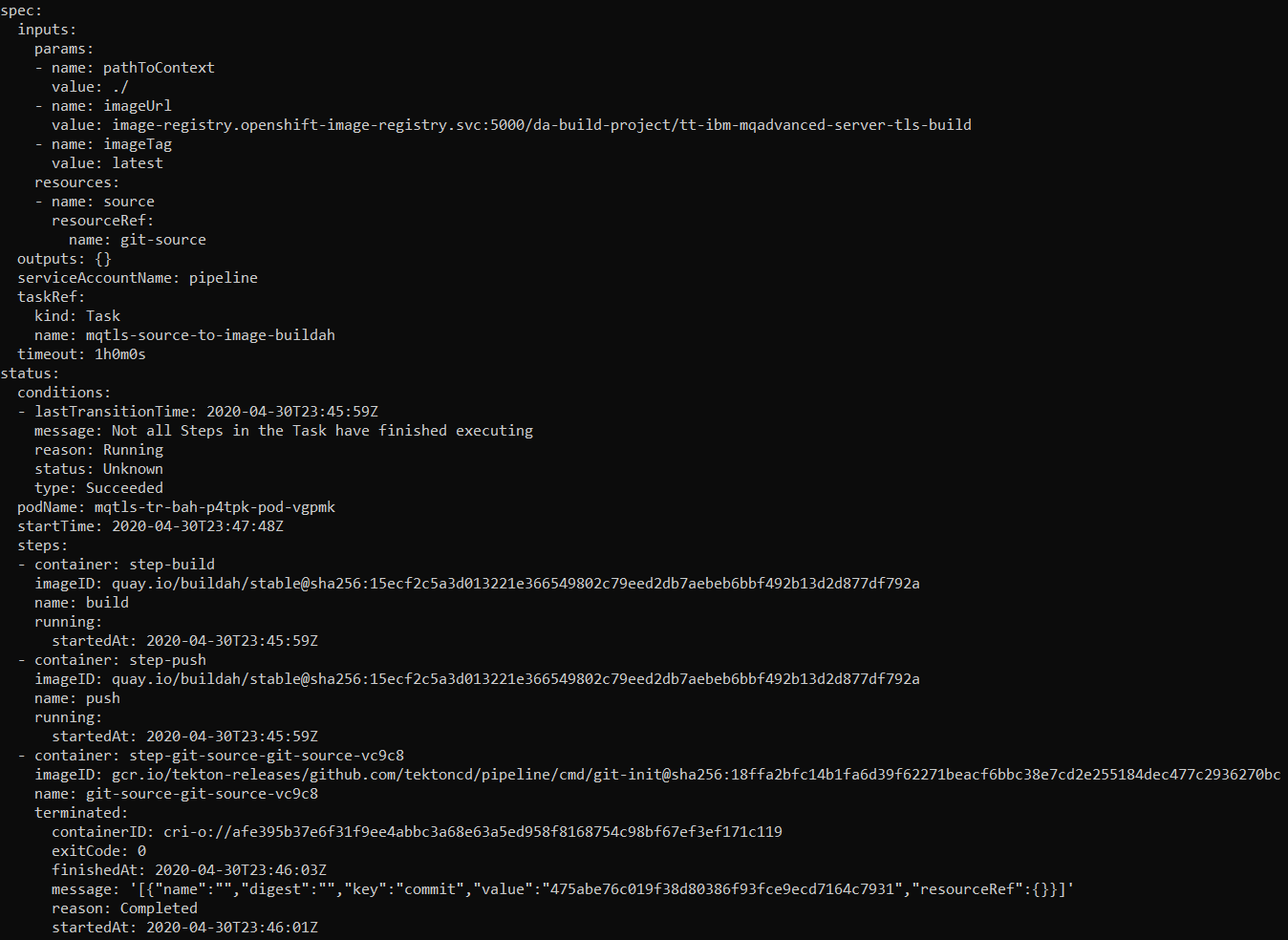
creation of the taskrun will trigger the task.



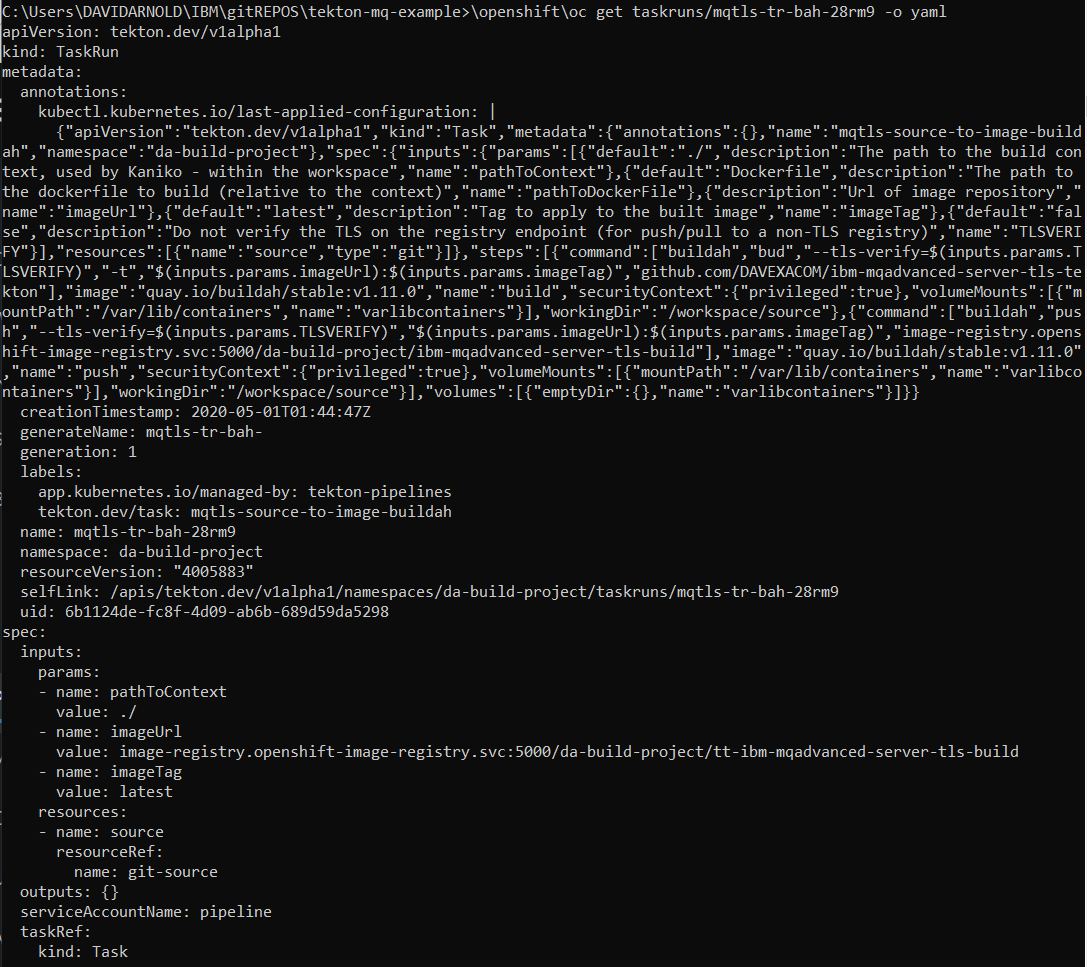
Check the RH Openshift console->pipelines->Task runs

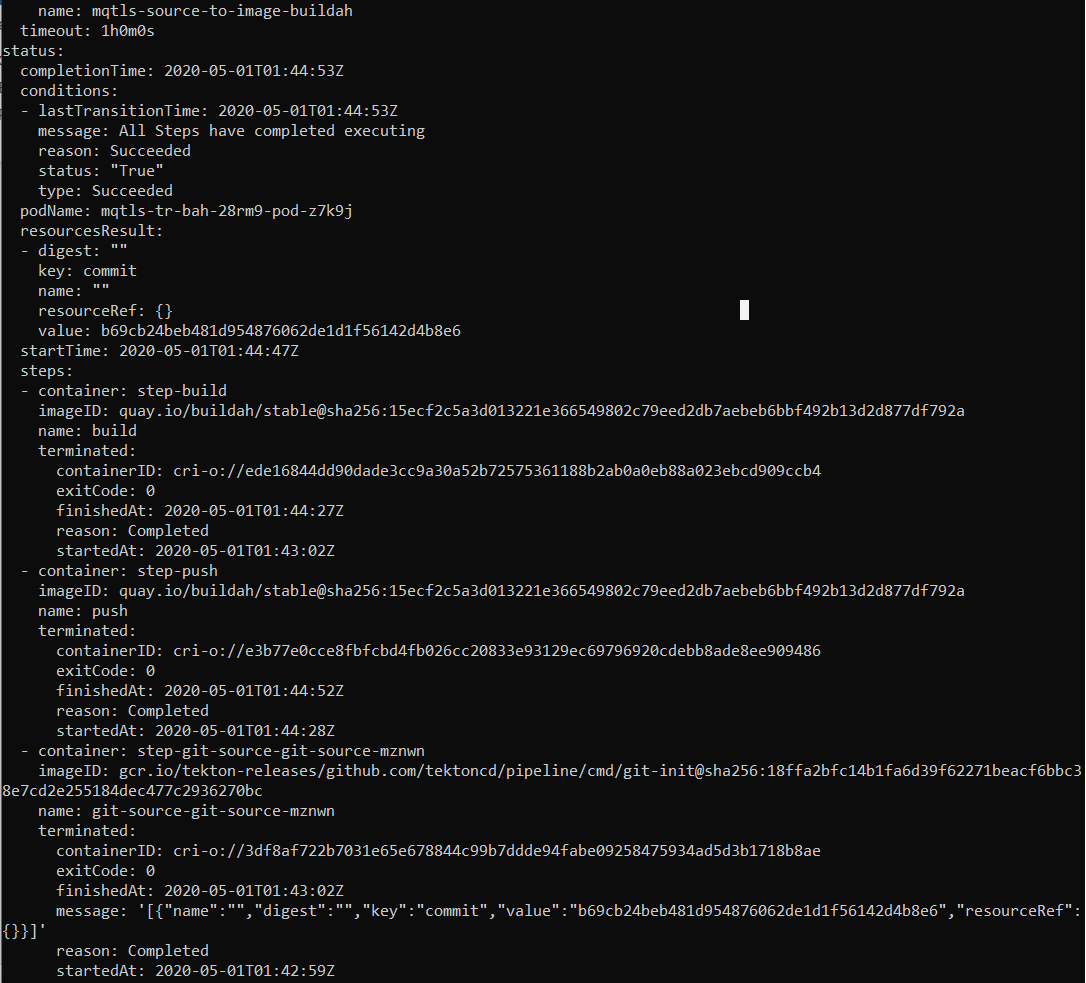


C:\openshift\oc get taskruns/mqtls-tr-bah-p4tpk -o yaml



You may have to run the C:\openshift\oc get taskruns/mqtls-tr-bah-p4tpk -o yaml command a number of times as the task is in flight.



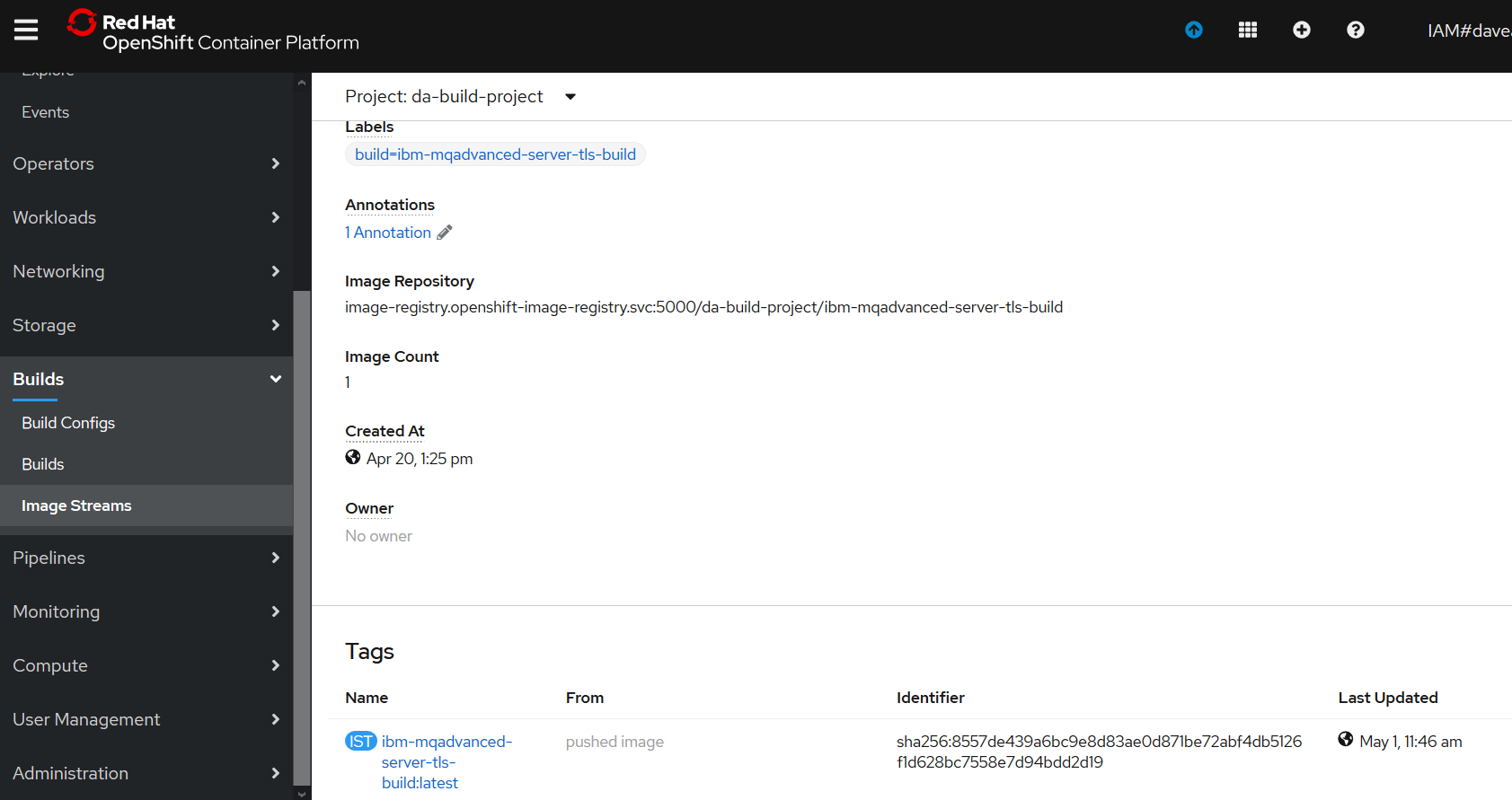


If you get errors jump to the [Debugging Appendix](#_Taskrun_for_build)

If it worked check the results in the RH Openshift console

RH Openshift Console->Builds->ImageStreams->ibm-mqadvanced-server-tls-build

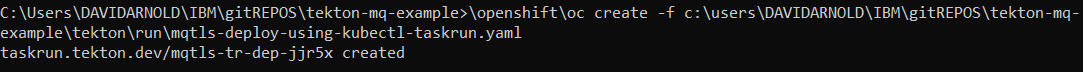
Note last update date and time



#### Taskrun for Kubectl Deploy task

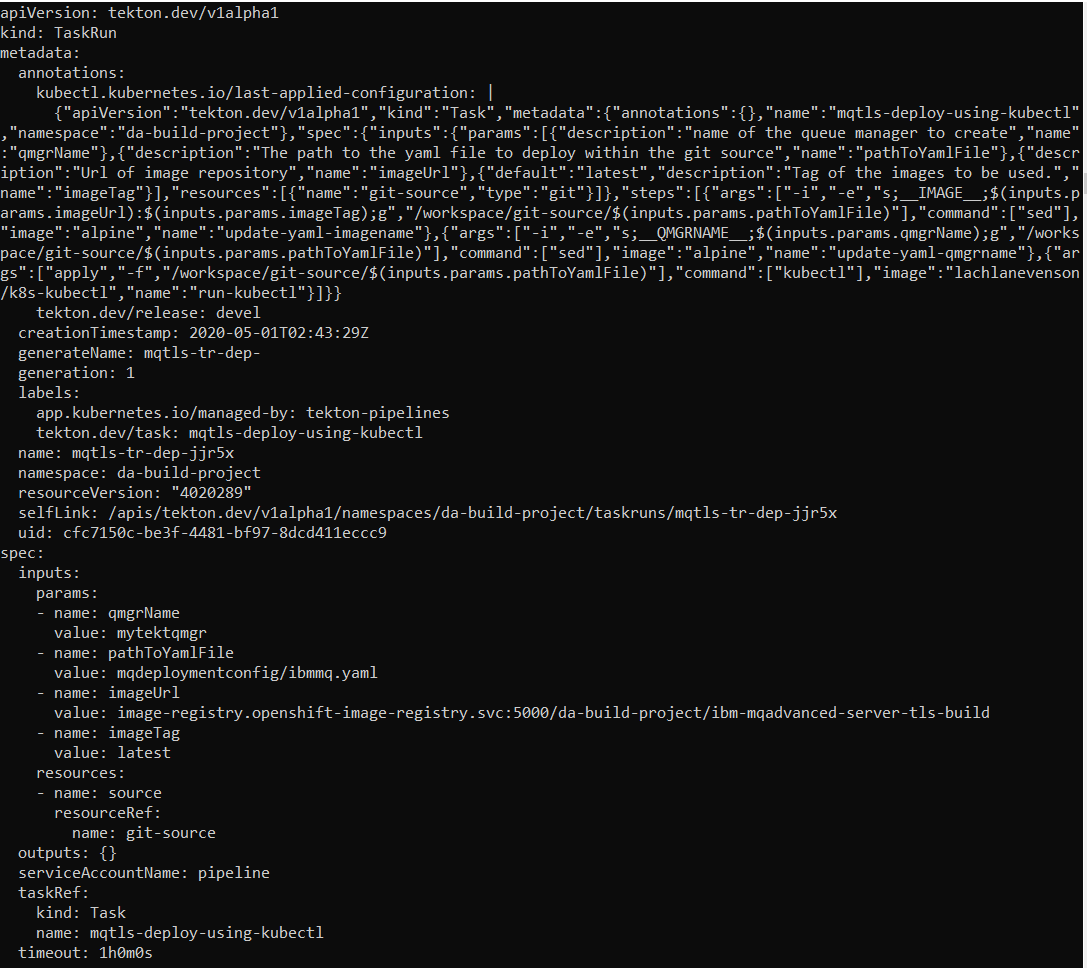
creation of the taskrun will trigger the task.

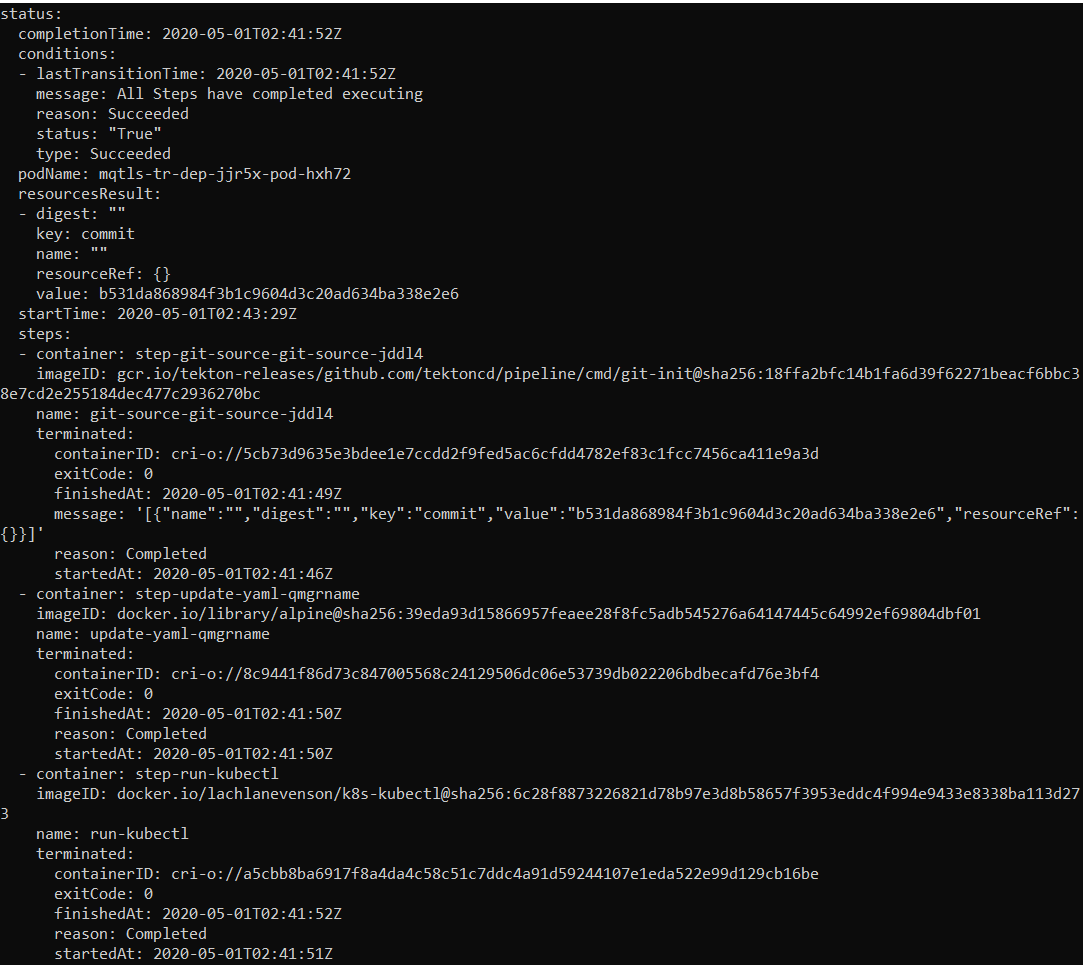
C:\openshift\oc create -f c:\users\DAVIDARNOLD\IBM\gitREPOS\tekton-mq-example\tekton\run\mqtls-deploy-using-kubectl-taskrun.yaml

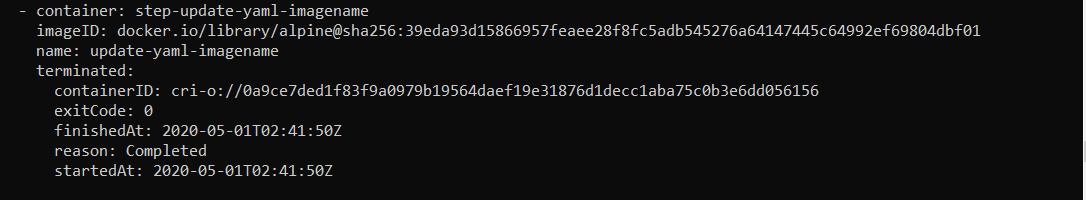


C:\openshift\oc get taskruns/mqtls-tr-dep-jjr5x -o yaml

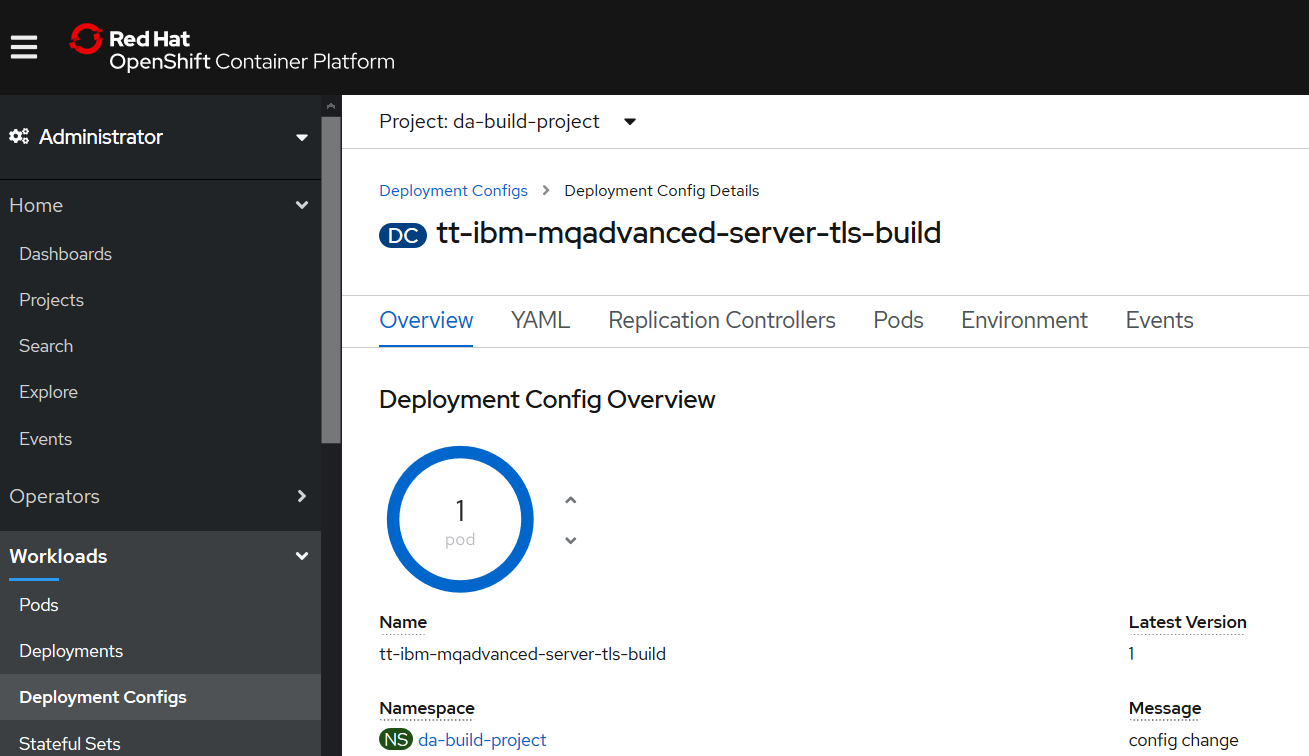
If you get errors jump to the [debug Appendix](#_Taskrun_for_deploy)

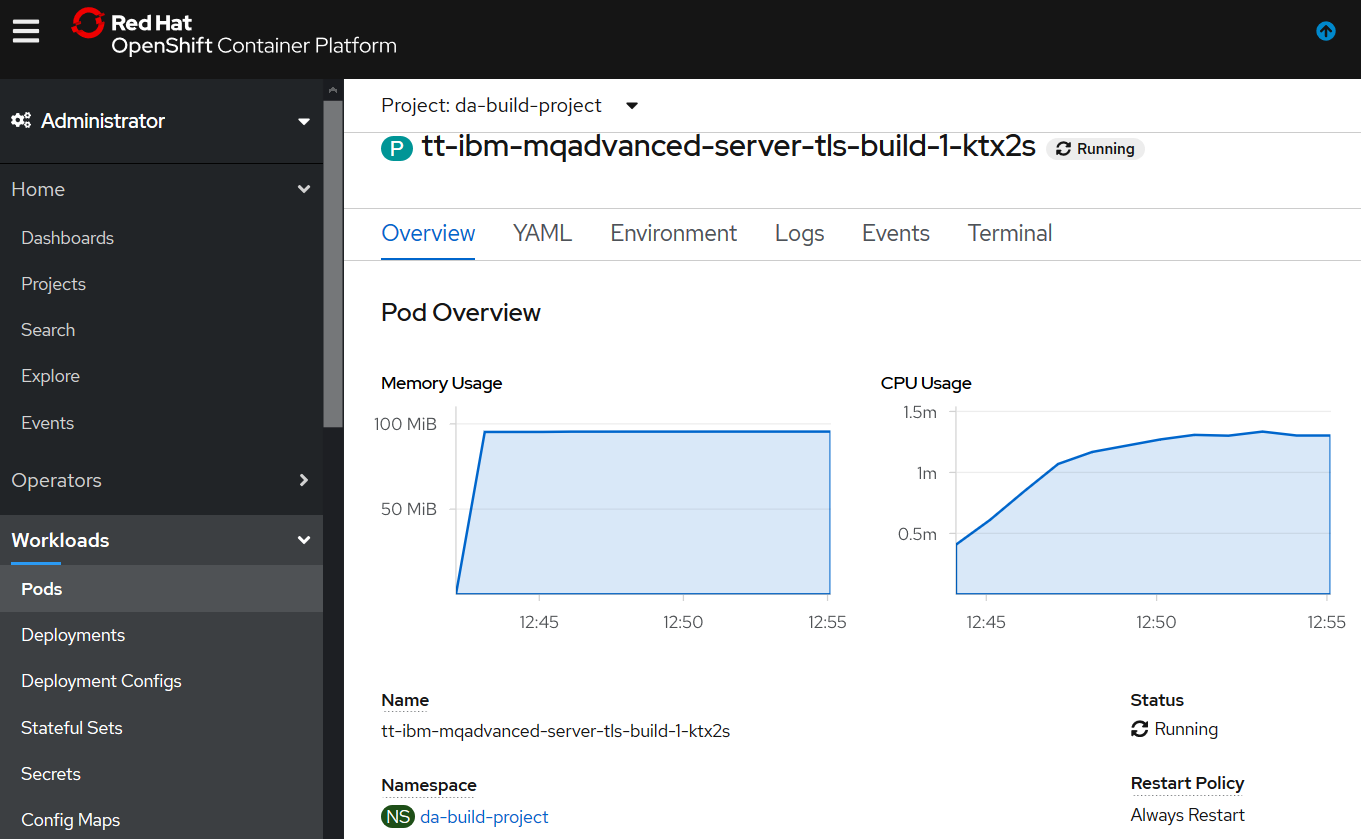


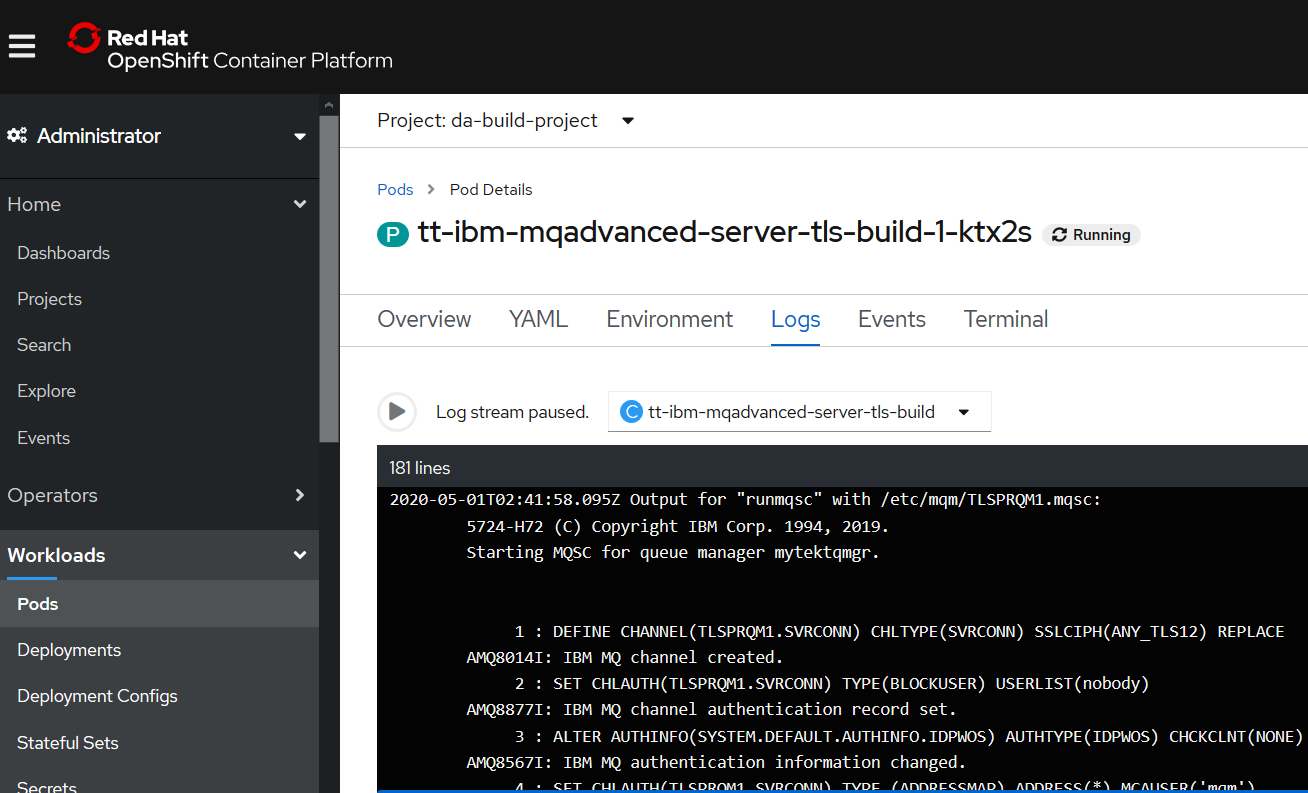




Check the results in the RH Openshift Console->Deployment Config->tt-ibm-mqadvanced-server-tls-build







Note the queue manager mqtektqmgr is running

### Create and test the Tekton Pipelinerun artifact in RH Openshift

#### Pipelinerun YAML – set a different queue manager name

apiVersion: tekton.dev/v1alpha1

kind: PipelineRun

metadata:

generateName: mqtls-pr-bahdep-

spec:

pipelineRef:

name: mqtls-build-and-deploy-pipeline-buildah

serviceAccountName: pipeline

resources:

- name: git-source

resourceRef:

name: git-source

params:

- name: qmgrName

value: **datektqmgr**

- name: pathToYamlFile

value: "mqdeploymentconfig/ibmmq.yaml"

- name: pathToContext

value: ./

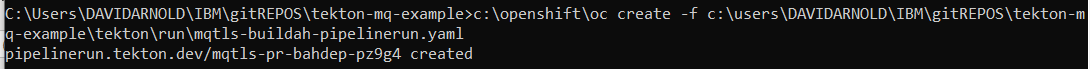
- name: imageUrl

value: image-registry.openshift-image-registry.svc:5000/da-build-project/ibm-mqadvanced-server-tls-build

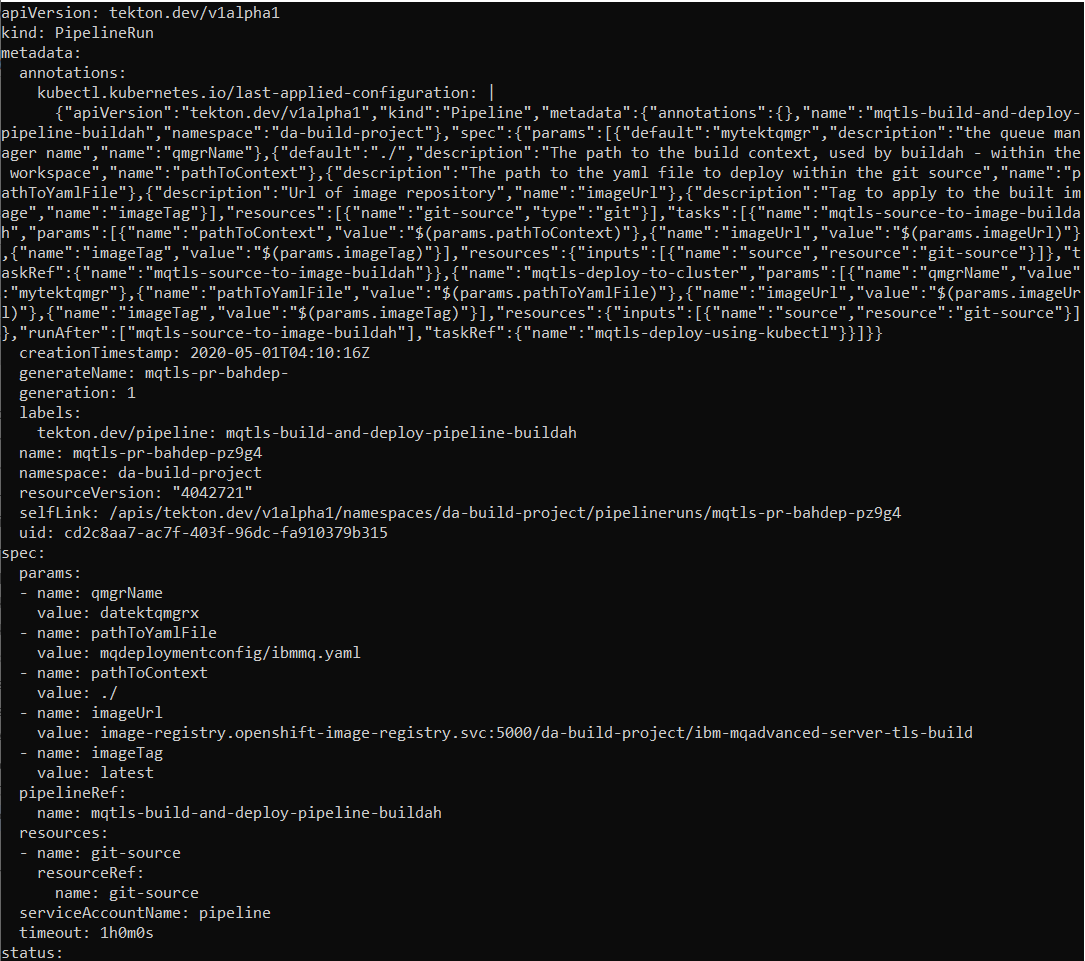
- name: imageTag

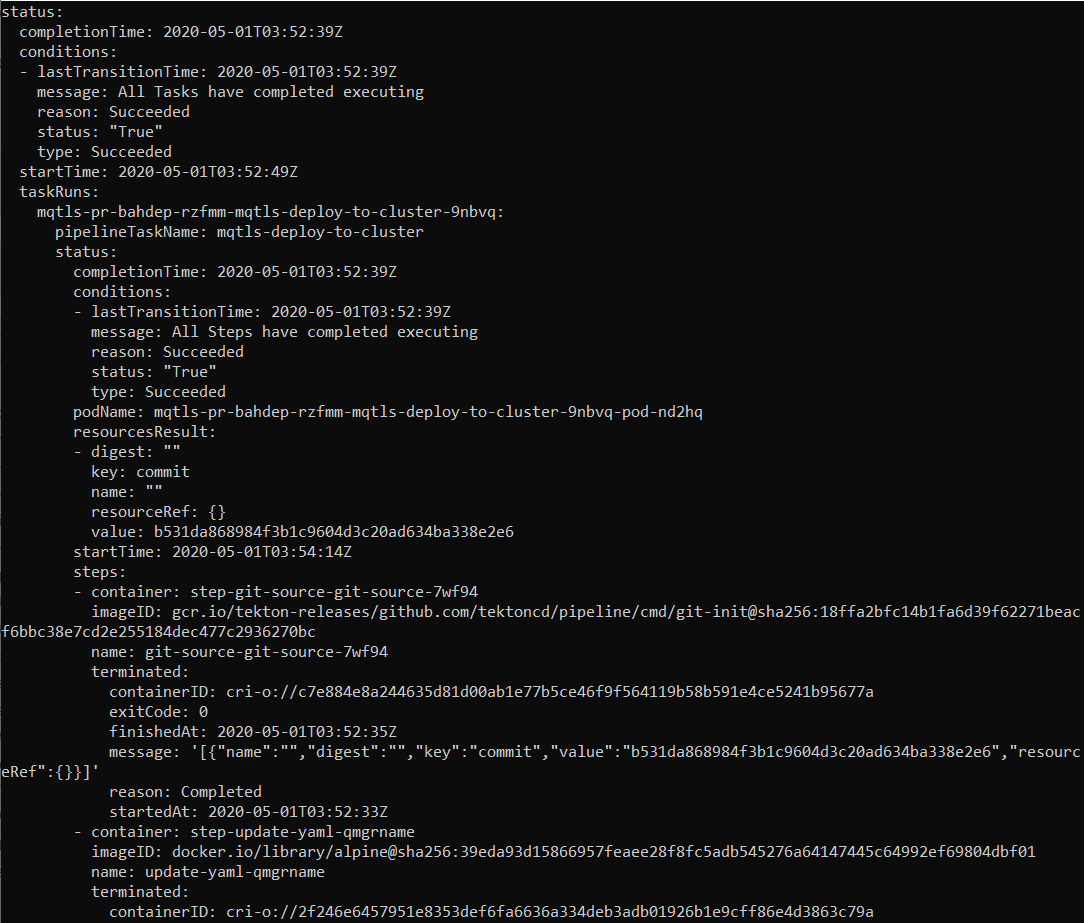
value: "latest"

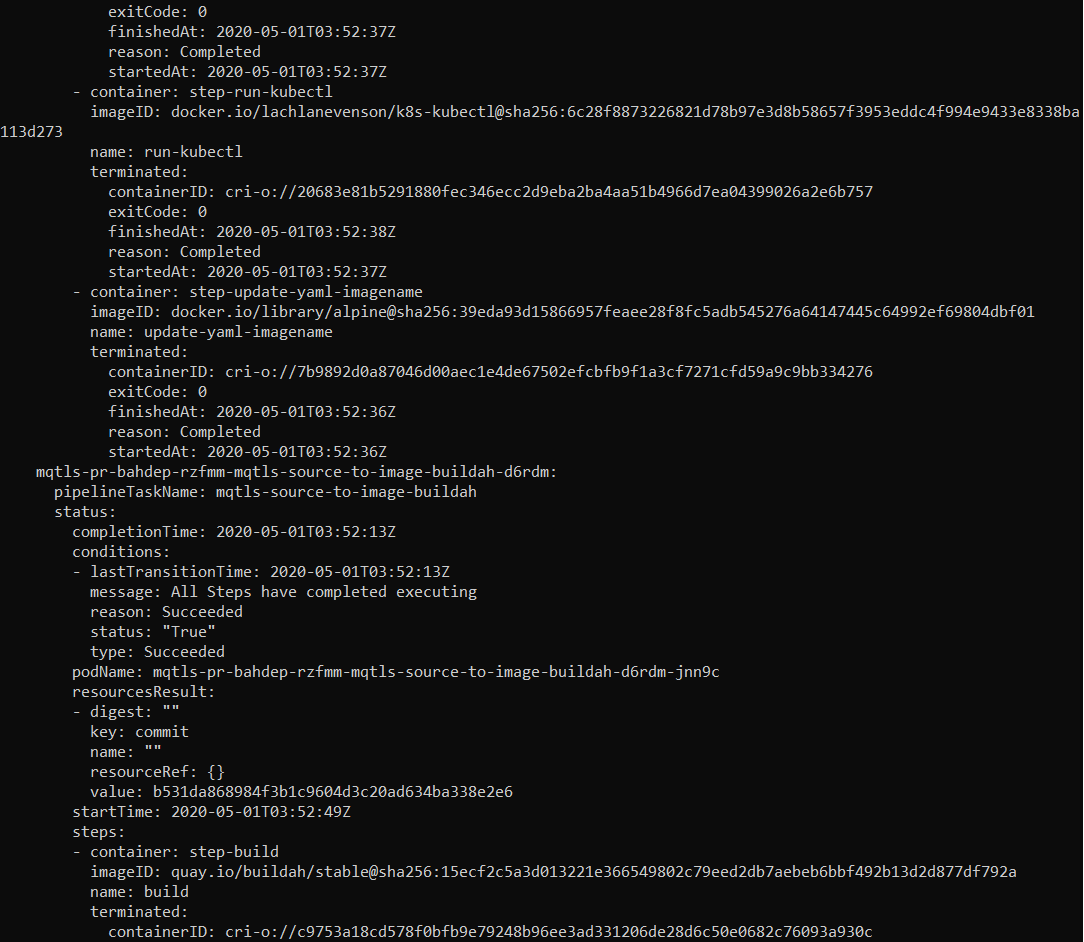
c:\openshift\oc create -f c:\users\DAVIDARNOLD\IBM\gitREPOS\tekton-mq-example\tekton\runs\ mqtls-buildah-pipelinerun.yaml

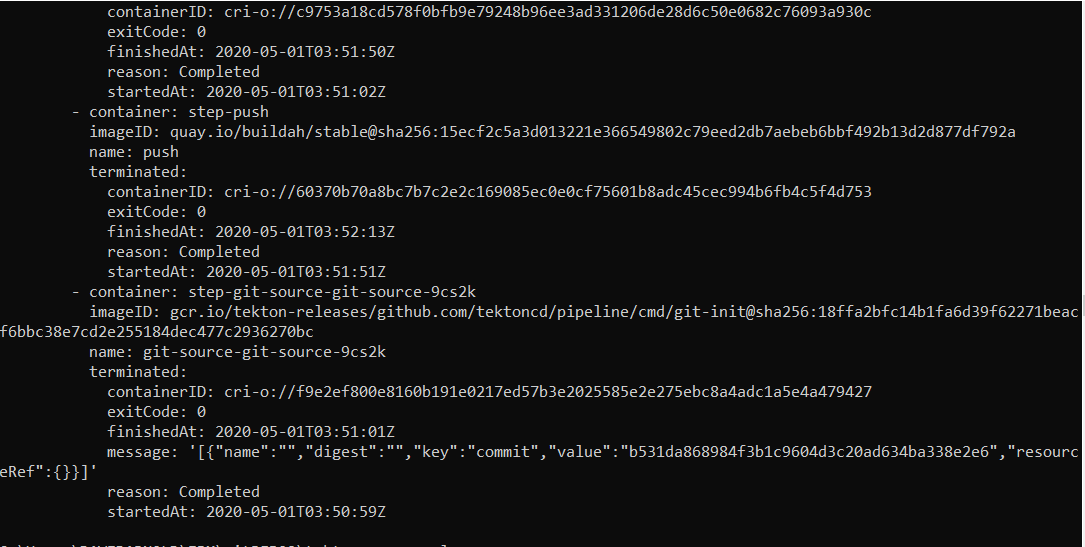


C:\openshift\oc get pipelineruns/mqtls-pr-bahdep-pz9g4 -o yaml



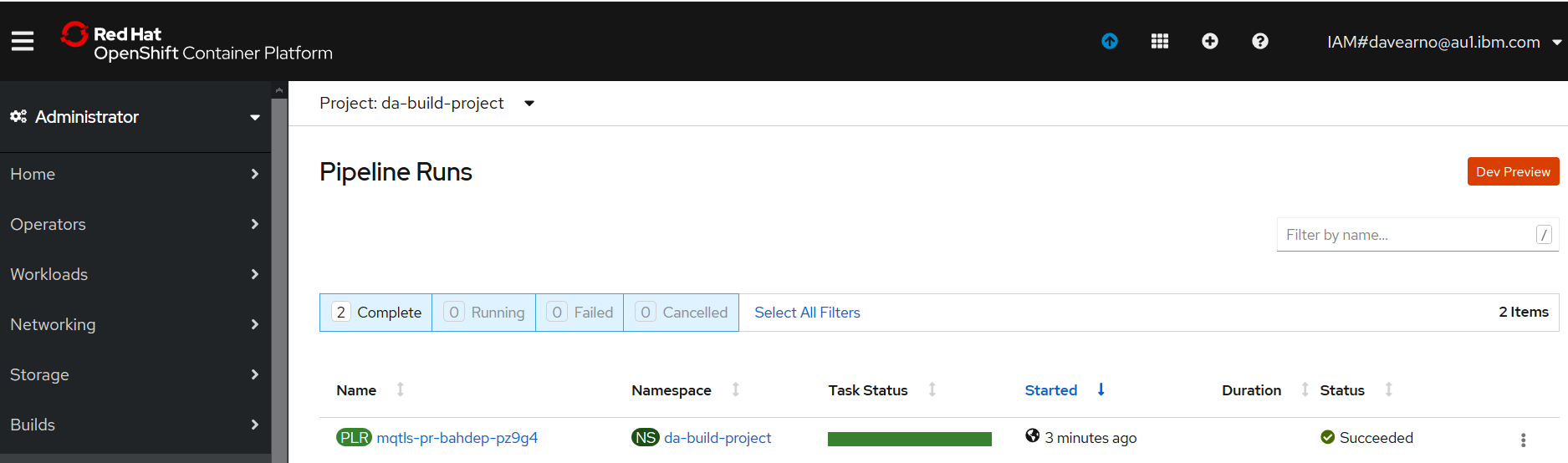






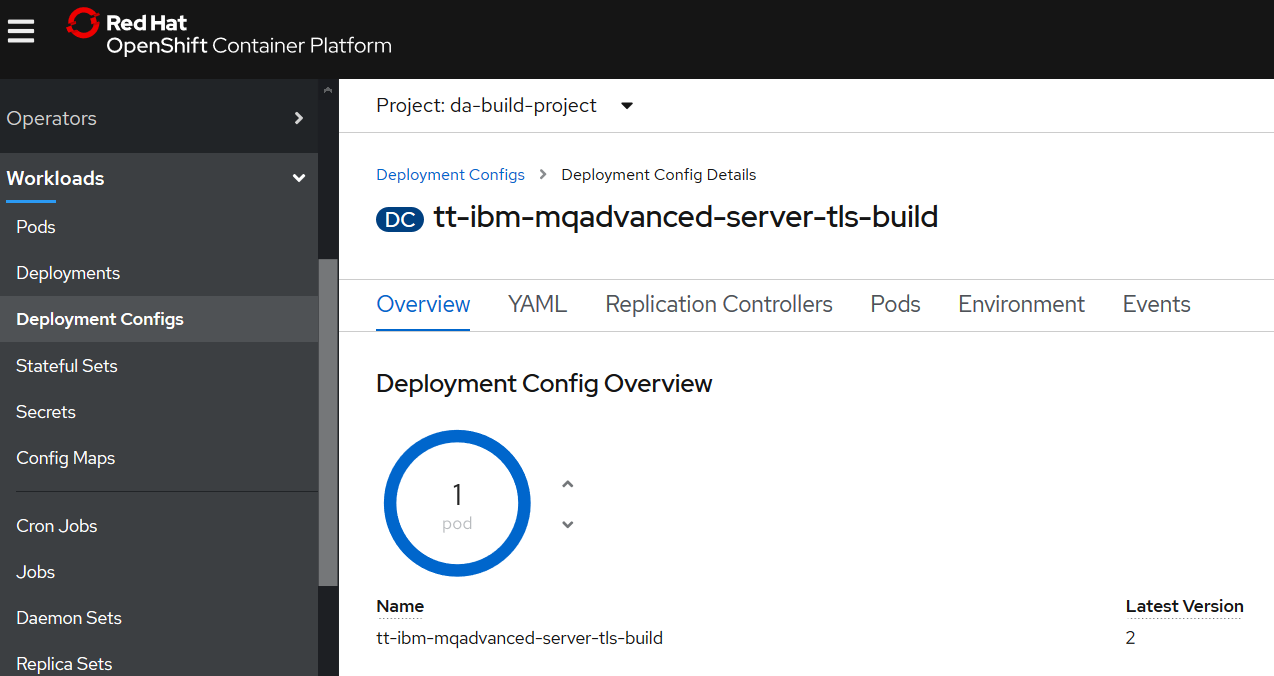
### Checking the Pipeline Run results

RH Openshift Console->Pipeline->pipeline runs->mqtls-pr-bahdep-pz9g4



RH Openshift Console->Workloads->deployment configs->tt-ibm-mqadvanced-server-tls-build

Note: the deployment config is now version 2 (the deploy taskRun created version 1)



## Creating the Service and Routes

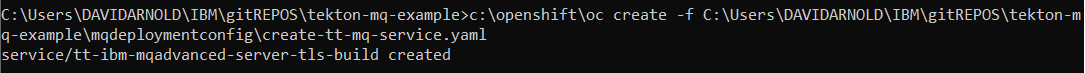
The service and route YAML definition files can be found in

### <https://github.com/DAVEXACOM/tekton-mq-example/tree/master/mqdeploymentconfig>

### Tekton tt pre-fixed mq service - tt-ibm-mqadvanced-server-tls-build

created manually at this point and not automated in the pipeline (but it could be in the same manner as the ibmmq.yaml deployment config in the pipeline)

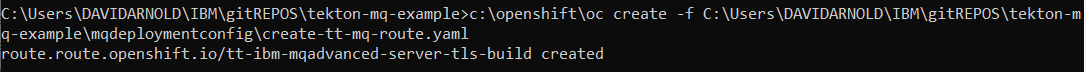
**c:\openshift\oc create -f C:\Users\DAVIDARNOLD\IBM\gitREPOS\tekton-mq-example\mqdeploymentconfig\create-tt-mq-service.yaml**



### Tekton tt prefixed mq route – tt-ibm-mqadvanced-server-tls-build

created manually at this point and not automated in the pipeline (but it could be in the same manner as the ibmace.yaml deployment config in the pipeline)

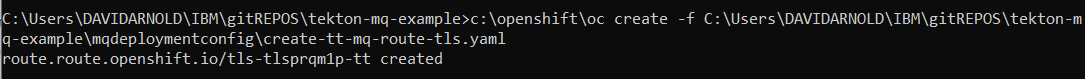
**c:\openshift\oc create -f C:\Users\DAVIDARNOLD\IBM\gitREPOS\tekton-mq-example\mqdeploymentconfig\create-tt-mq-route.yaml**



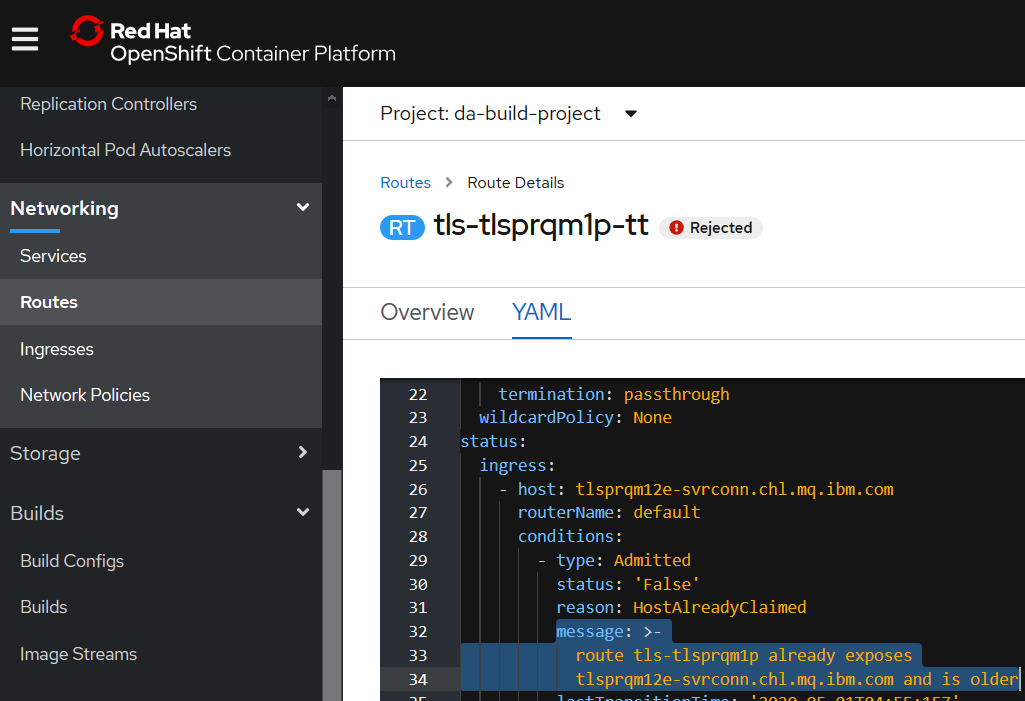
### Tekton tt prefixed mq tls route – tls-tlsprqm1p-tt

created manually at this point and not automated in the pipeline (but it could be in the same manner as the ibmace.yaml deployment config in the pipeline)

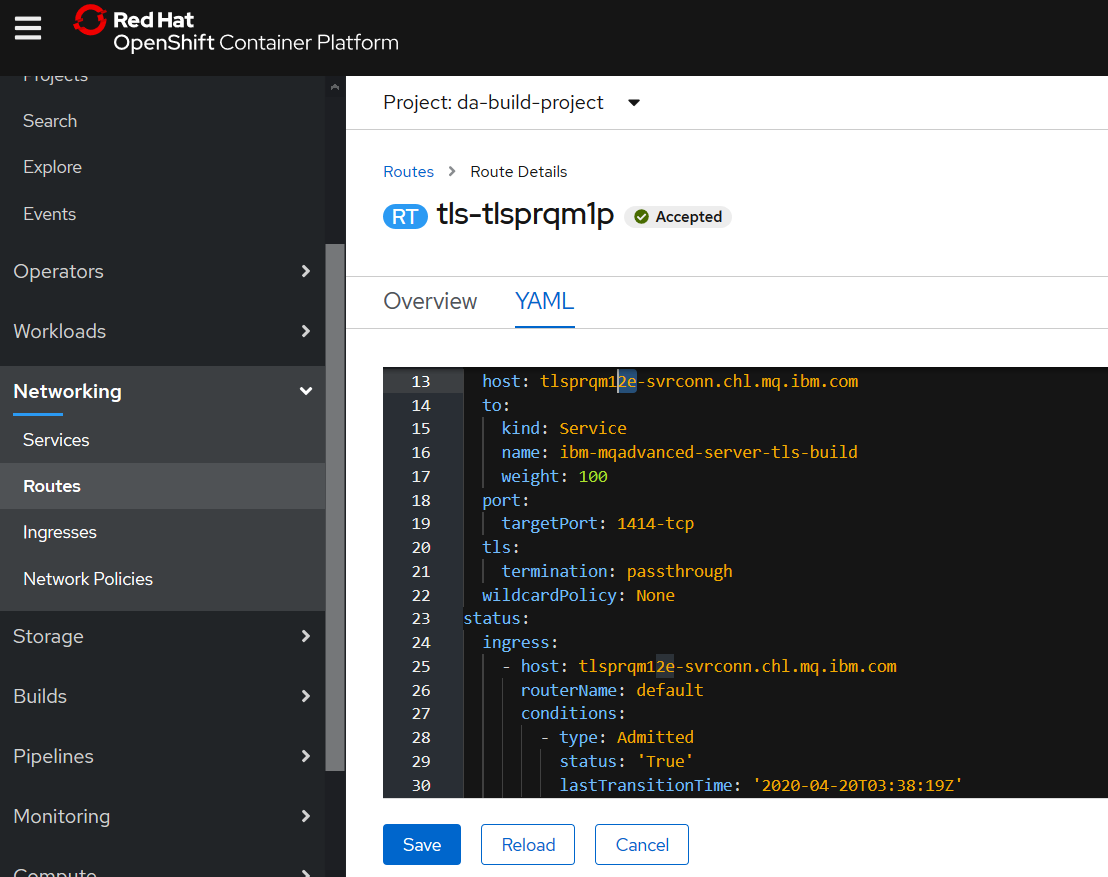
**c:\openshift\oc create -f C:\Users\DAVIDARNOLD\IBM\gitREPOS\tekton-mq-example\mqdeploymentconfig\create-tt-mq-route-tls.yaml**



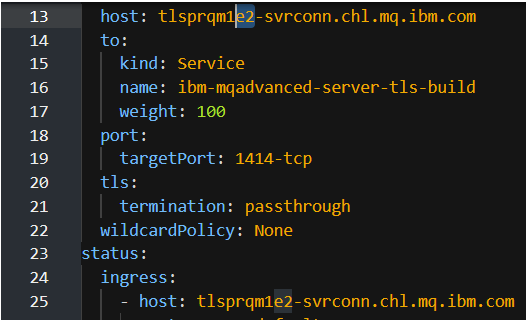
If you created and tested a TLS enabled queue manager using OOTB Openshift earlier the TLS client channel will already be exposed in route tls-tlsprqm1p and you’ll get the following message.



### Clean up to get tls-tlsprqm1p-tt working



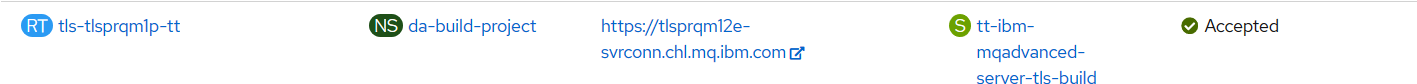
Make an obvious change in case you want to change back later



Save and reload

If you now return to the RH Openshift Web Console->networking routes

Where tls-tlsprqm1p-tt was showing it should now be showing acceped



## Testing Connection to the queue manager

Need to clean up the environment and then perform the tests

### Local Client connection MQSC

Obtain your queue manager name

Obtain you conname from oc get routes

DEFINE CHANNEL(TLSPRQM1.SVRCONN) +

CHLTYPE(CLNTCONN) +

TRPTYPE(TCP) +

CONNAME('ibm-mqadvanced-server-tls-build-da-build-project.dacluster-0511fc923b97b25b240630e79d362861-0000.au-syd.containers.appdomain.cloud(443)') +

CERTLABL('ibmmqarnold') +

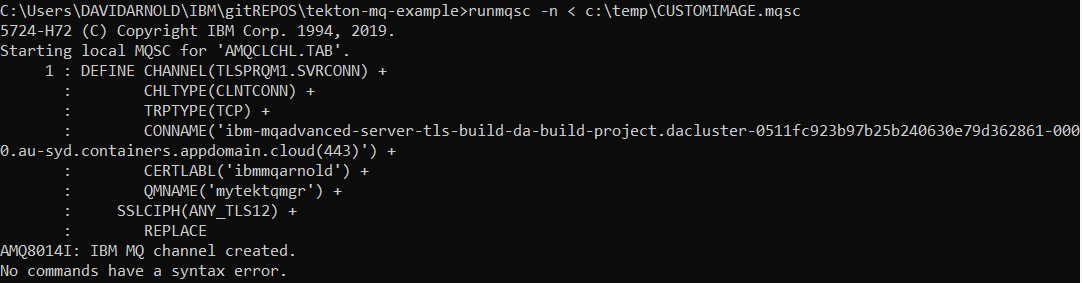
QMNAME('mytektqmgr') +

SSLCIPH(ANY\_TLS12) +

REPLACE

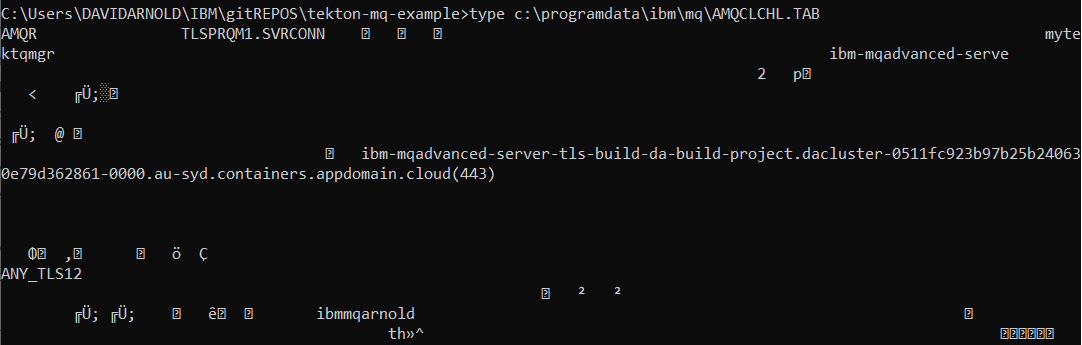
From a command line running as administrator

runmqsc -n < c:\temp\CUSTOMIMAGE.mqsc



Check the CCDT table

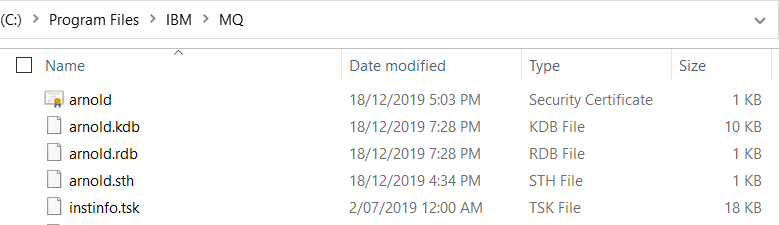
type c:\programdata\ibm\mq\AMQCLCHL.TAB



### Download and copy in the client side Keystore

A matching set of client side key/cert files for those that are “baked into” the server side container can be downloaded from:

<https://github.com/DAVEXACOM/Exploring-ICP4i-RHOS/tree/master/MQ%20TLS%20and%20Custom%20Layer%20Image-311and42/clientside%20TLS%20files>

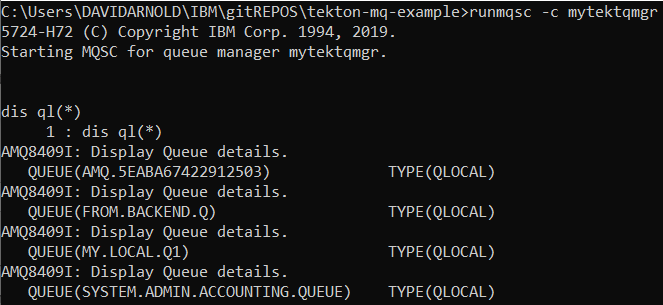


### Set the MQSSLKEY environment variable

set MQSSLKEYR=C:\Program Files\IBM\MQ\arnold

### Use runmqsc client to connect to your queue manager

runmqsc -c mytektqmgr



# Exploring ConfigMaps for MQSC definition abstraction

## Manually investigating Config Maps and Deployment Configs

### Create a config map with an MQSC file

Create a config map where the name is the queue manager name and the MQSC is a simple receiver channel using the naming convention TO.*queuemanagername*

kind: ConfigMap

apiVersion: v1

metadata:

name: mytektqmgr

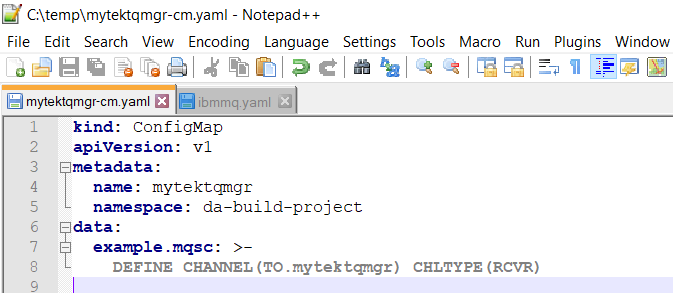
namespace: da-build-project

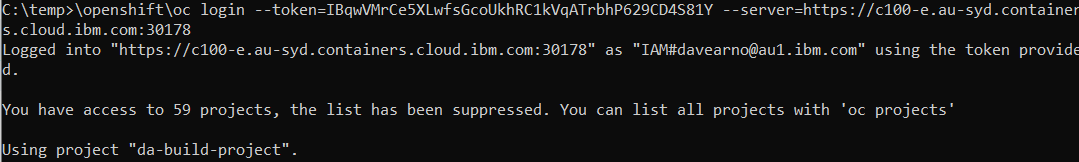
data:

example.mqsc: >-

DEFINE CHANNEL(**TO.mytektqmgr**) CHLTYPE(RCVR)

The above contents should be placed in a file <mytektqmgr-cm.yaml> on your local machine



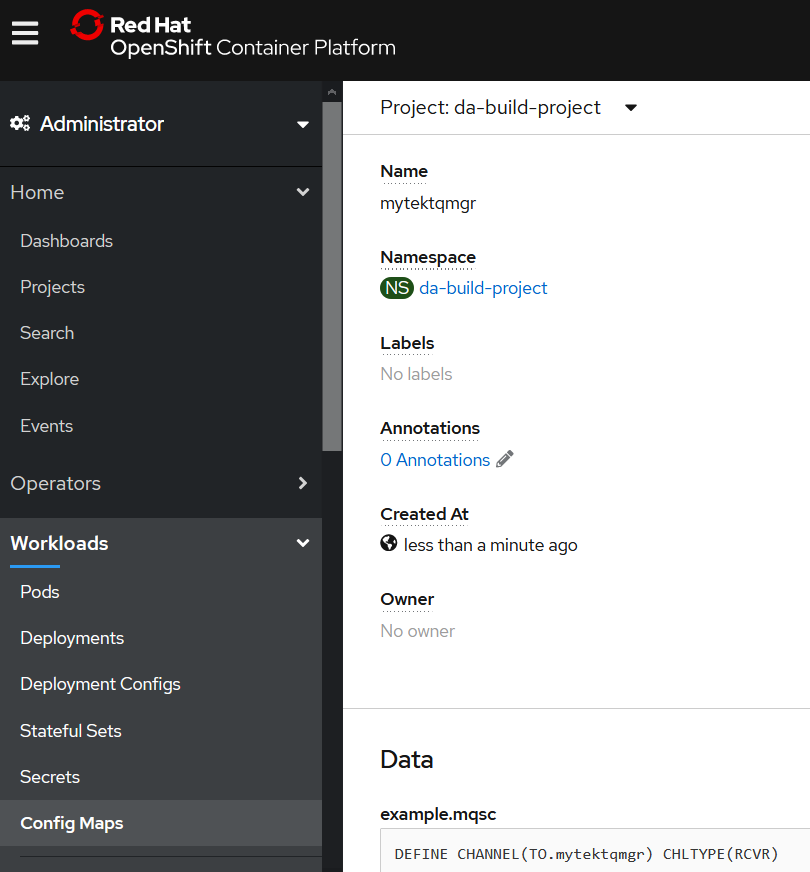


and then use the command:

**oc create -f mytektqmgr-cm.yaml**

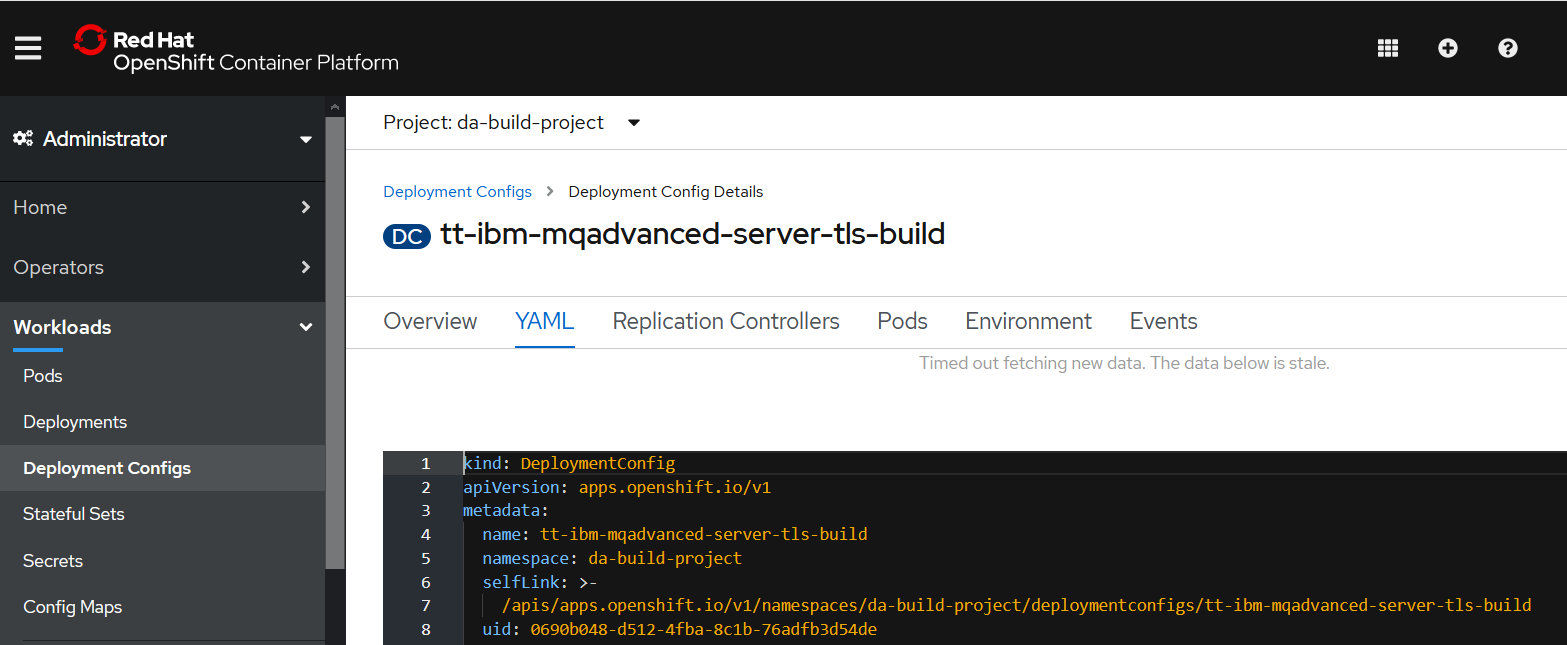


Review in RHOCP Console->workloads->configmaps



### Tekton Mq Deploy DeploymentConfig

DeploymentConfig that is used by my Tekton pipeline that builds an MQ image and deploys it.



kind: DeploymentConfig

apiVersion: apps.openshift.io/v1

metadata:

name: tt-ibm-mqadvanced-server-tls-build

namespace: da-build-project

spec:

strategy:

type: Rolling

rollingParams:

updatePeriodSeconds: 1

intervalSeconds: 1

timeoutSeconds: 600

maxUnavailable: 25%

maxSurge: 25%

resources: {}

activeDeadlineSeconds: 21600

triggers:

- type: ConfigChange

- type: ImageChange

imageChangeParams:

automatic: true

containerNames:

- tt-ibm-mqadvanced-server-tls-build

from:

kind: ImageStreamTag

namespace: da-build-project

name: 'ibm-mqadvanced-server-tls-build:latest'

lastTriggeredImage: >-

image-registry.openshift-image-registry.svc:5000/da-build-project/ibm-mqadvanced-server-tls-build@sha256:06d8aad6d183e82b11390eb3339598cec4b465f5cd61b9b54bb3159b90170f4c

replicas: 1

revisionHistoryLimit: 10

test: false

selector:

app: tt-ibm-mqadvanced-server-tls-build

deploymentconfig: tt-ibm-mqadvanced-server-tls-build

template:

metadata:

creationTimestamp: null

labels:

app: tt-ibm-mqadvanced-server-tls-build

deploymentconfig: tt-ibm-mqadvanced-server-tls-build

spec:

containers:

- name: tt-ibm-mqadvanced-server-tls-build

image: >-

image-registry.openshift-image-registry.svc:5000/da-build-project/ibm-mqadvanced-server-tls-build@sha256:06d8aad6d183e82b11390eb3339598cec4b465f5cd61b9b54bb3159b90170f4c

ports:

- containerPort: 9443

protocol: TCP

- containerPort: 1414

protocol: TCP

- containerPort: 9157

protocol: TCP

env:

- name: LICENSE

value: accept

- name: MQ\_QMGR\_NAME

value: mytektqmgr

resources: {}

terminationMessagePath: /dev/termination-log

terminationMessagePolicy: File

imagePullPolicy: Always

restartPolicy: Always

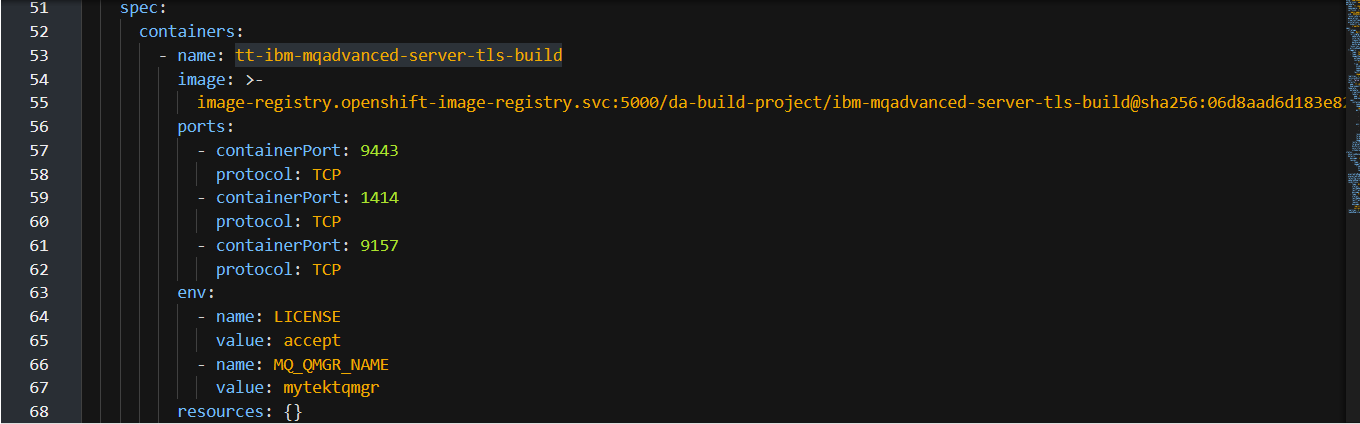
terminationGracePeriodSeconds: 30

dnsPolicy: ClusterFirst

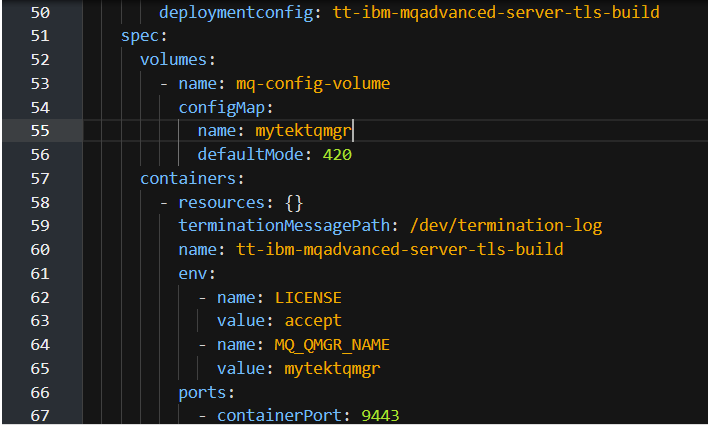
securityContext: {}

schedulerName: default-scheduler

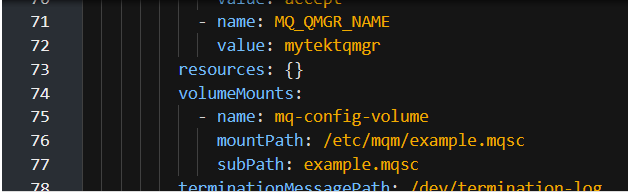
#### MQ container spec section – before making configMap changes



#### MQ container spec section – after making configMap changes

Add the volume under template->spec ahead of containers 

Add volumemounts after resources



Save and reload

### Final DeploymentConfig for MQ to run MQSCs from ConfigMap

kind: DeploymentConfig

apiVersion: apps.openshift.io/v1

metadata:

name: tt-ibm-mqadvanced-server-tls-build

namespace: da-build-project

selfLink: >-

/apis/apps.openshift.io/v1/namespaces/da-build-project/deploymentconfigs/tt-ibm-mqadvanced-server-tls-build

uid: 0690b048-d512-4fba-8c1b-76adfb3d54de

resourceVersion: '10860610'

generation: 8

creationTimestamp: '2020-05-01T04:34:47Z'

annotations:

kubectl.kubernetes.io/last-applied-configuration: >

{"apiVersion":"apps.openshift.io/v1","kind":"DeploymentConfig","metadata":{"annotations":{},"name":"tt-ibm-mqadvanced-server-tls-build","namespace":"da-build-project"},"spec":{"replicas":1,"selector":{"app":"tt-ibm-mqadvanced-server-tls-build","deploymentconfig":"tt-ibm-mqadvanced-server-tls-build"},"strategy":{"activeDeadlineSeconds":21600,"resources":{},"rollingParams":{"intervalSeconds":1,"maxSurge":"25%","maxUnavailable":"25%","timeoutSeconds":600,"updatePeriodSeconds":1},"type":"Rolling"},"template":{"metadata":{"labels":{"app":"tt-ibm-mqadvanced-server-tls-build","deploymentconfig":"tt-ibm-mqadvanced-server-tls-build"}},"spec":{"containers":[{"env":[{"name":"LICENSE","value":"accept"},{"name":"MQ\_QMGR\_NAME","value":"mytektqmgr"}],"image":"image-registry.openshift-image-registry.svc:5000/da-build-project/ibm-mqadvanced-server-tls-build:latest","imagePullPolicy":"Always","name":"tt-ibm-mqadvanced-server-tls-build","ports":[{"containerPort":9443,"protocol":"TCP"},{"containerPort":1414,"protocol":"TCP"},{"containerPort":9157,"protocol":"TCP"}],"resources":{},"terminationMessagePath":"/dev/termination-log","terminationMessagePolicy":"File"}],"dnsPolicy":"ClusterFirst","restartPolicy":"Always","schedulerName":"default-scheduler","securityContext":{},"terminationGracePeriodSeconds":30}},"triggers":[{"type":"ConfigChange"},{"imageChangeParams":{"automatic":true,"containerNames":["tt-ibm-mqadvanced-server-tls-build"],"from":{"kind":"ImageStreamTag","name":"ibm-mqadvanced-server-tls-build:latest","namespace":"da-build-project"}},"type":"ImageChange"}]}}

spec:

strategy:

type: Rolling

rollingParams:

updatePeriodSeconds: 1

intervalSeconds: 1

timeoutSeconds: 600

maxUnavailable: 25%

maxSurge: 25%

resources: {}

activeDeadlineSeconds: 21600

triggers:

- type: ConfigChange

- type: ImageChange

imageChangeParams:

automatic: true

containerNames:

- tt-ibm-mqadvanced-server-tls-build

from:

kind: ImageStreamTag

namespace: da-build-project

name: 'ibm-mqadvanced-server-tls-build:latest'

lastTriggeredImage: >-

image-registry.openshift-image-registry.svc:5000/da-build-project/ibm-mqadvanced-server-tls-build@sha256:06d8aad6d183e82b11390eb3339598cec4b465f5cd61b9b54bb3159b90170f4c

replicas: 1

revisionHistoryLimit: 10

test: false

selector:

app: tt-ibm-mqadvanced-server-tls-build

deploymentconfig: tt-ibm-mqadvanced-server-tls-build

template:

metadata:

creationTimestamp: null

labels:

app: tt-ibm-mqadvanced-server-tls-build

deploymentconfig: tt-ibm-mqadvanced-server-tls-build

spec:

volumes:

- name: mq-config-volume

configMap:

name: mytektqmgr

defaultMode: 420

containers:

- resources: {}

terminationMessagePath: /dev/termination-log

name: tt-ibm-mqadvanced-server-tls-build

env:

- name: LICENSE

value: accept

- name: MQ\_QMGR\_NAME

value: mytektqmgr

ports:

- containerPort: 9443

protocol: TCP

- containerPort: 1414

protocol: TCP

- containerPort: 9157

protocol: TCP

imagePullPolicy: Always

volumeMounts:

- name: mq-config-volume

mountPath: /etc/mqm/example.mqsc

subPath: example.mqsc

terminationMessagePolicy: File

image: >-

image-registry.openshift-image-registry.svc:5000/da-build-project/ibm-mqadvanced-server-tls-build@sha256:06d8aad6d183e82b11390eb3339598cec4b465f5cd61b9b54bb3159b90170f4c

restartPolicy: Always

terminationGracePeriodSeconds: 30

dnsPolicy: ClusterFirst

securityContext: {}

schedulerName: default-scheduler

status:

observedGeneration: 8

details:

message: config change

causes:

- type: ConfigChange

availableReplicas: 1

unavailableReplicas: 0

latestVersion: 5

updatedReplicas: 1

conditions:

- type: Available

status: 'True'

lastUpdateTime: '2020-05-01T04:34:52Z'

lastTransitionTime: '2020-05-01T04:34:52Z'

message: Deployment config has minimum availability.

- type: Progressing

status: 'True'

lastUpdateTime: '2020-05-20T05:52:00Z'

lastTransitionTime: '2020-05-20T05:51:57Z'

reason: NewReplicationControllerAvailable

message: >-

replication controller "tt-ibm-mqadvanced-server-tls-build-5"

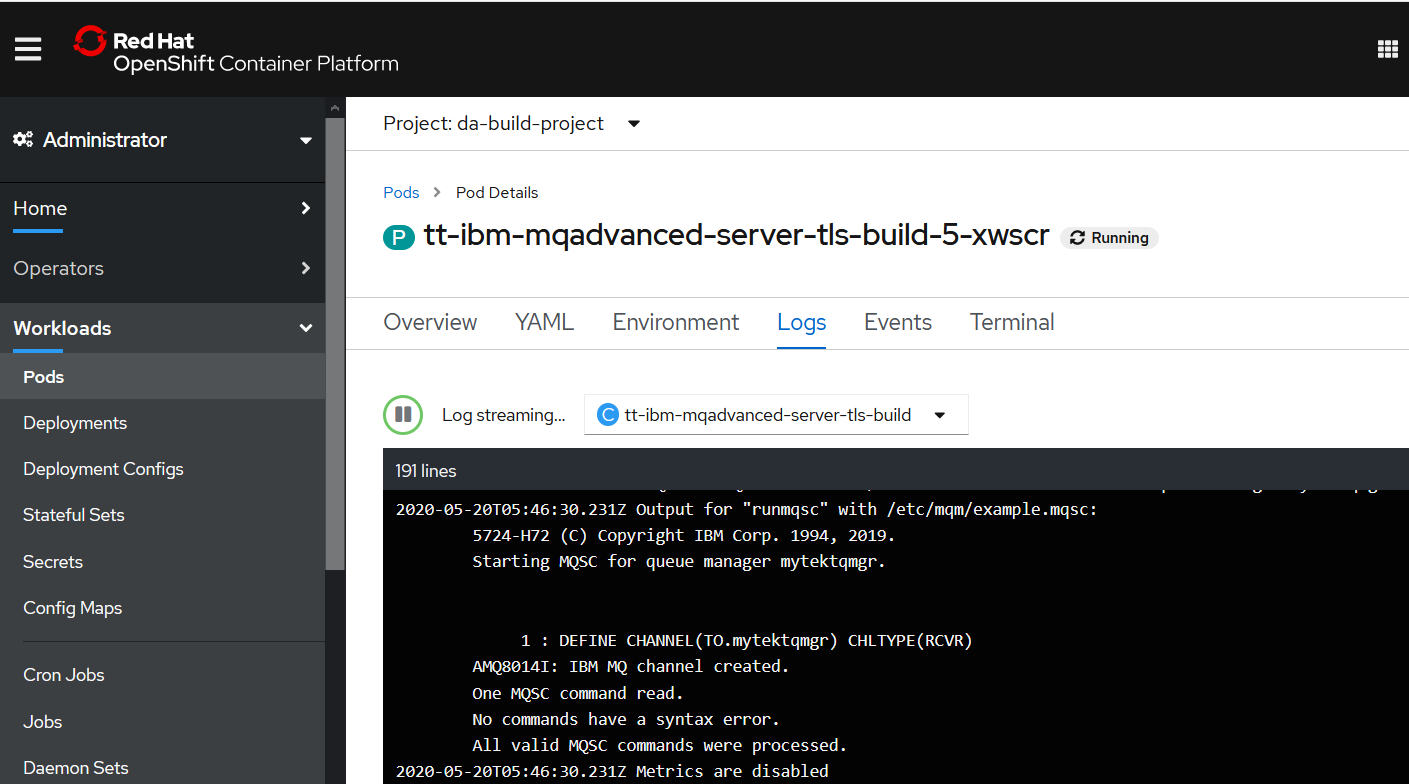
successfully rolled out

replicas: 1

readyReplicas: 1

### Check the Pod has restarted based on the Deployment Config change

Check RHOCP Console->workloads->pods



Note that the MQSC definition from the configMap is executed when the Pod restarts.

# Appendix - Debugging

## Updating the Tekton artifacts in RH Openshift

USE OC REPLACE if you change and files

oc replace -f c:\users\DAVIDARNOLD\IBM\gitREPOS\tekton-ace-example\tekton\tasks\source-to-image-buildah.yamlh

oc apply -f Tekton-ivt\echo-hello-world-task.yaml

## Deleting the Tekton artifacts in RH Openshift

USE OC DELETE -f c:\users\davidarnold\ibm\github\tekton-mq-example\tekton\ other\deleteartifacts.yaml to remove any artifacts from RH OpenShift - use a yaml snippet in a file to identify kind: , name: and namespace: see example deleteartifacts.yaml below

apiVersion: tekton.dev/v1alpha1

kind: Task

metadata:

creationTimestamp: '2020-03-03T00:27:03Z'

generation: 3

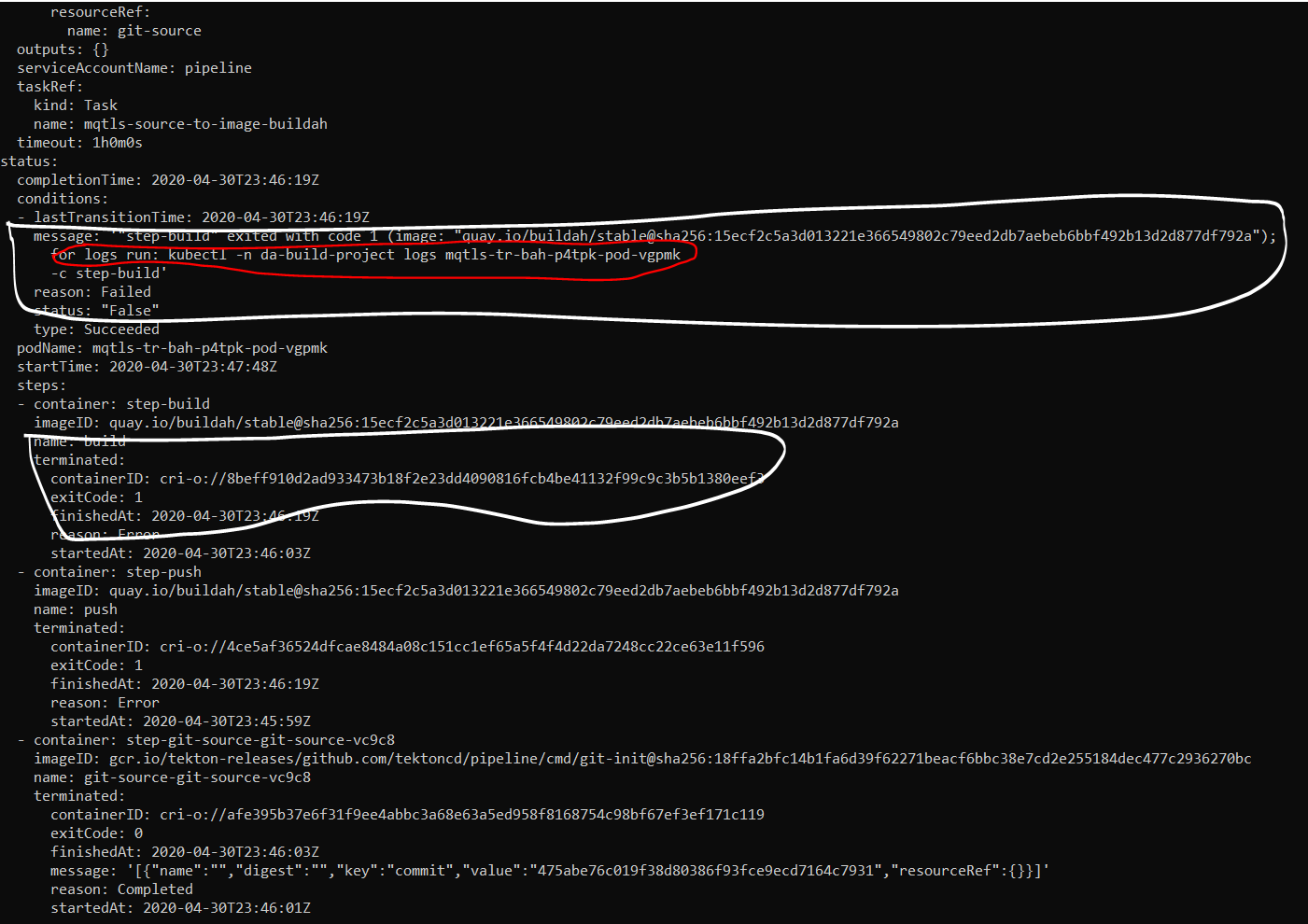
name: mqtls-source-to-image-buildah

namespace: da-build-project

## Taskrun for build fails

When the task runs completes you need to check its logs for success for failure.

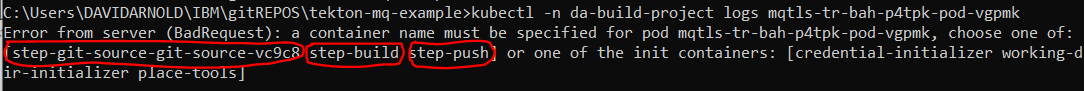
C:\Users\DAVIDARNOLD\IBM\gitREPOS\tekton-mq-example>\openshift\oc get taskruns/mqtls-tr-bah-p4tpk -o yaml



### Inspect the logs

C:\Users\DAVIDARNOLD\IBM\gitREPOS\tekton-mq-example>kubectl -n da-build-project logs mqtls-tr-bah-p4tpk-pod-vgpmk

There are 3 logs to inspect



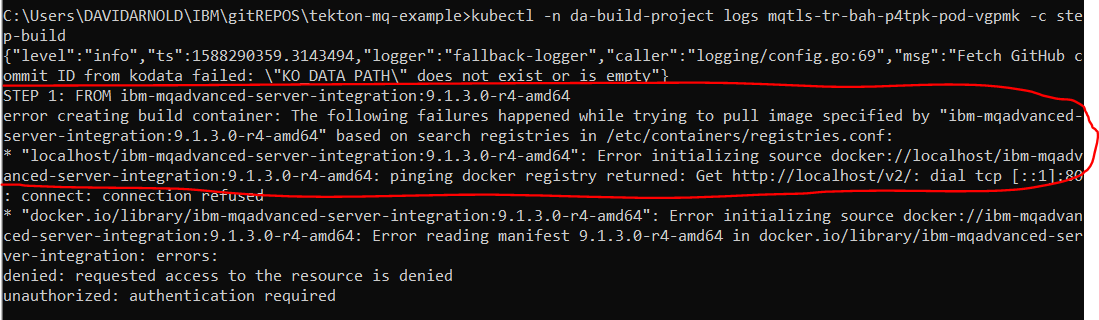
kubectl -n da-build-project logs mqtls-tr-bah-p4tpk-pod-vgpmk -c **step-git-source-git-source-vc9c8**



Looks like we successfully cloned the github source repository into **workspace/source**

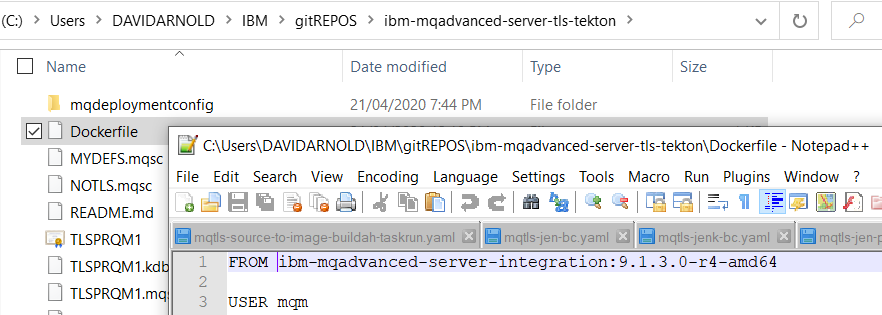
note: potentially subsequent steps and tasks will need to reference **workspace/source**.

kubectl -n da-build-project logs mqtls-tr-bah-p4tpk-pod-vgpmk -c **step-build**



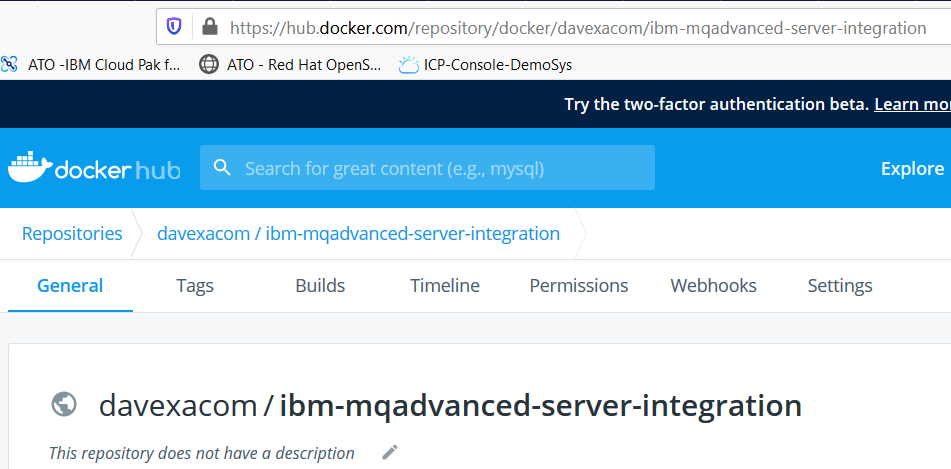
Looks like a problem finding the MQ TLs base image to build FROM in step one of the docker file in the GitHub repos we cloned

Check the docker file



Change the dockerfile to match the image location on dockerhub or other registry

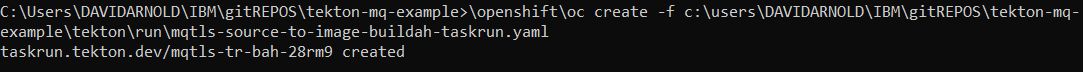




Push your changes to Github

Create a new instance of the taskrun

C:\openshift\oc create -f c:\users\DAVIDARNOLD\IBM\gitREPOS\tekton-mq-example\tekton\run\mqtls-source-to-image-buildah-taskrun.yaml



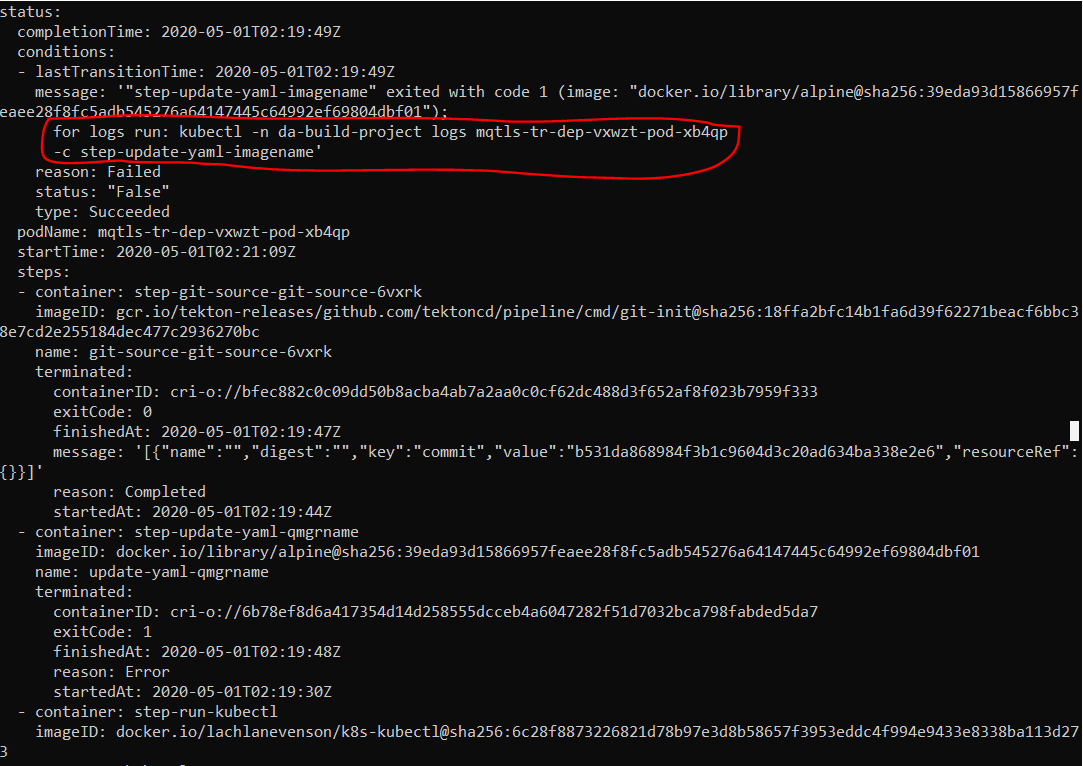
Get the output from run

C:\openshift\oc get taskruns/mqtls-tr-bah-28rm9 -o yaml

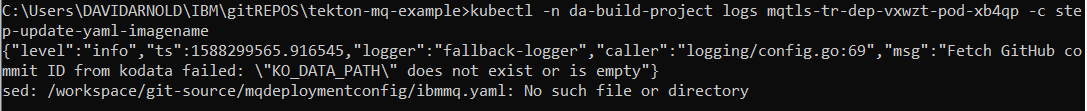
## Taskrun for deploy fails

When the task runs completes you need to check its logs for success for failure.

C:\Users\DAVIDARNOLD\IBM\gitREPOS\tekton-mq-example>\openshift\ oc get taskruns/mqtls-tr-dep-vxwzt -o yaml



kubectl -n da-build-project logs mqtls-tr-dep-vxwzt-pod-xb4qp -c step-update-yaml-imagename



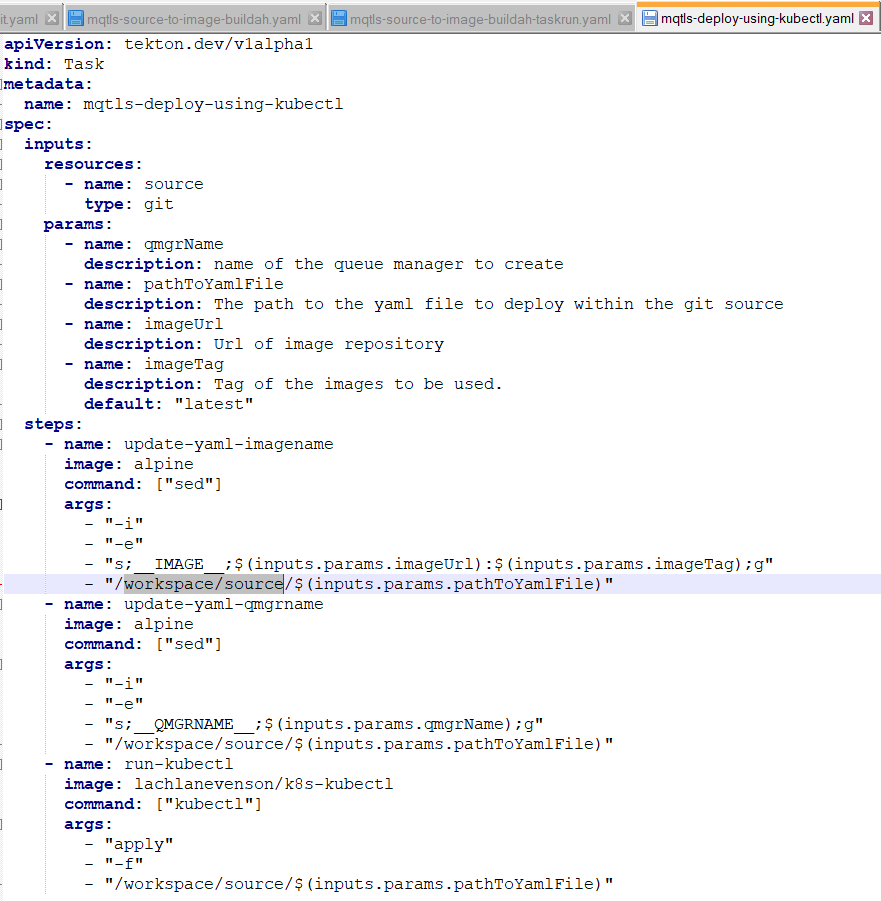
Looks like we have a mismatch between where the Buildah step cloned the git repos to and where this deploy step is looking for the ibmmq.yaml file

Buildah task cloned to: workspace/source

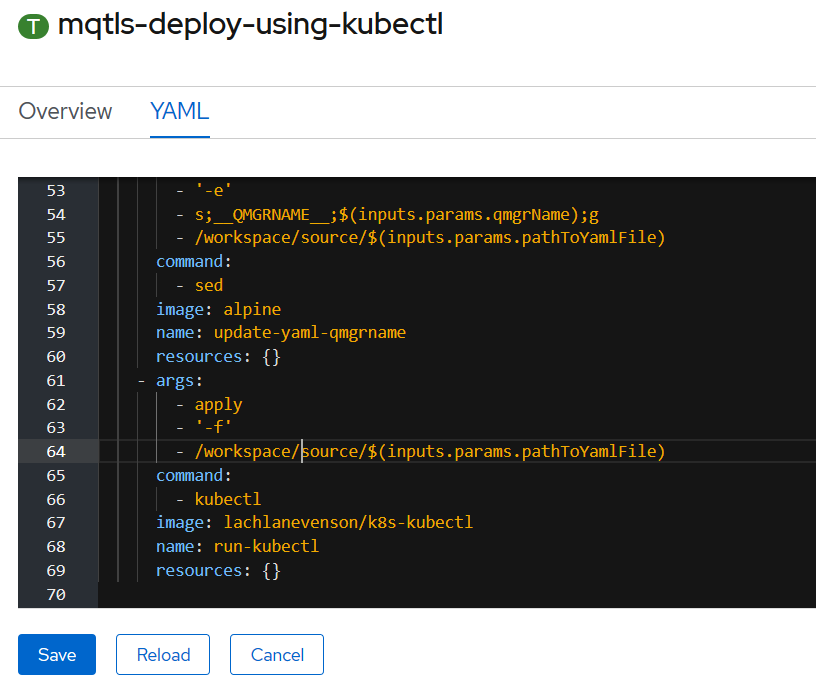
Deploy task is looking in: workspace/git-source

Correct the deploy task YAML.

You can do this in the source and oc apply to update the task



Or edit in the RH Openshift console

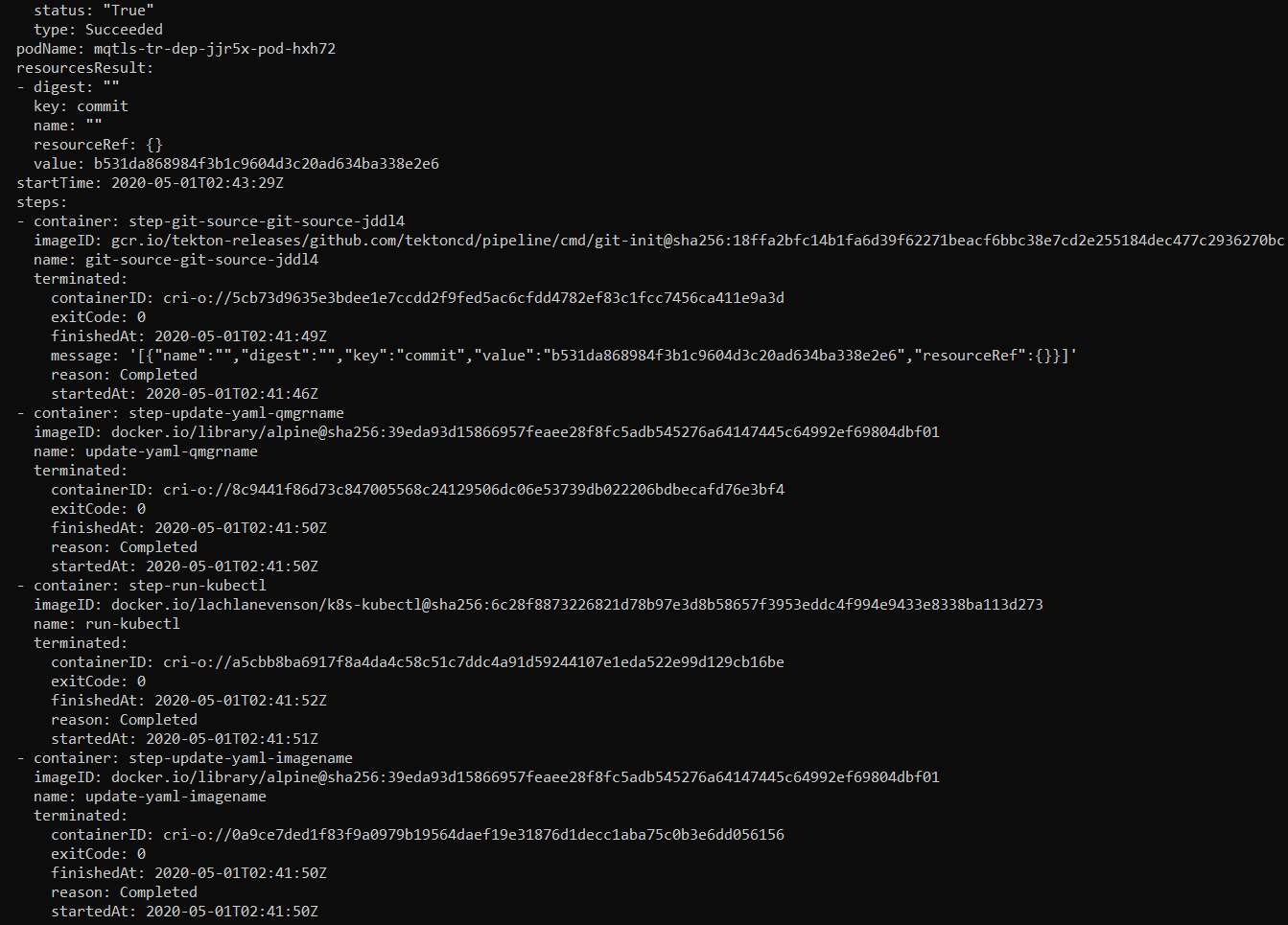


Retry the deploy Taskrun

C:\openshift\oc create -f c:\users\DAVIDARNOLD\IBM\gitREPOS\tekton-mq-example\tekton\run\mqtls-deploy-using-kubectl-taskrun.yaml



C:\openshift\oc get taskruns/mqtls-tr-dep-jjr5x -o yaml



# Appendix - Changing to a later version of Tekton

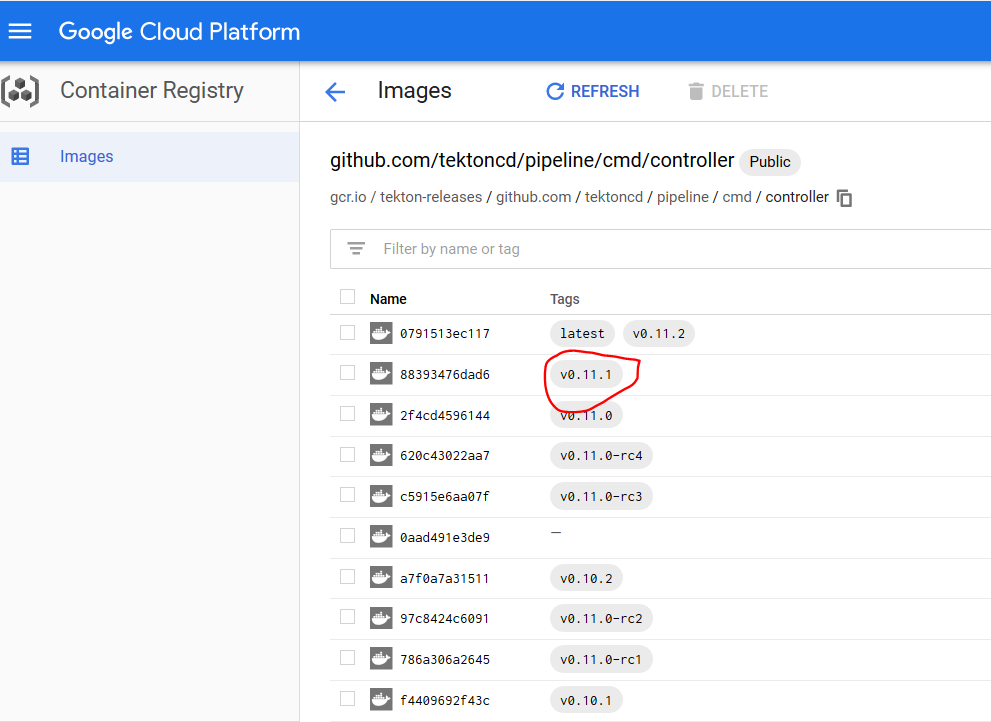
This section is unproven or completely tested. With Tekton still changing I was looking at how one might update the components.

### Tekton Controller

Changing from v0.10.1 to v0.11.1 example

Make a back up of the YAML file in case you mess up the cut and pasting.

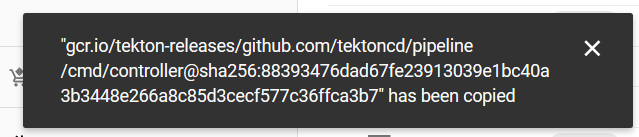
<https://console.cloud.google.com/gcr/images/tekton-releases/GLOBAL/github.com/tektoncd/pipeline/cmd/controller?gcrImageListsize=30>



Hover mouse over the gap



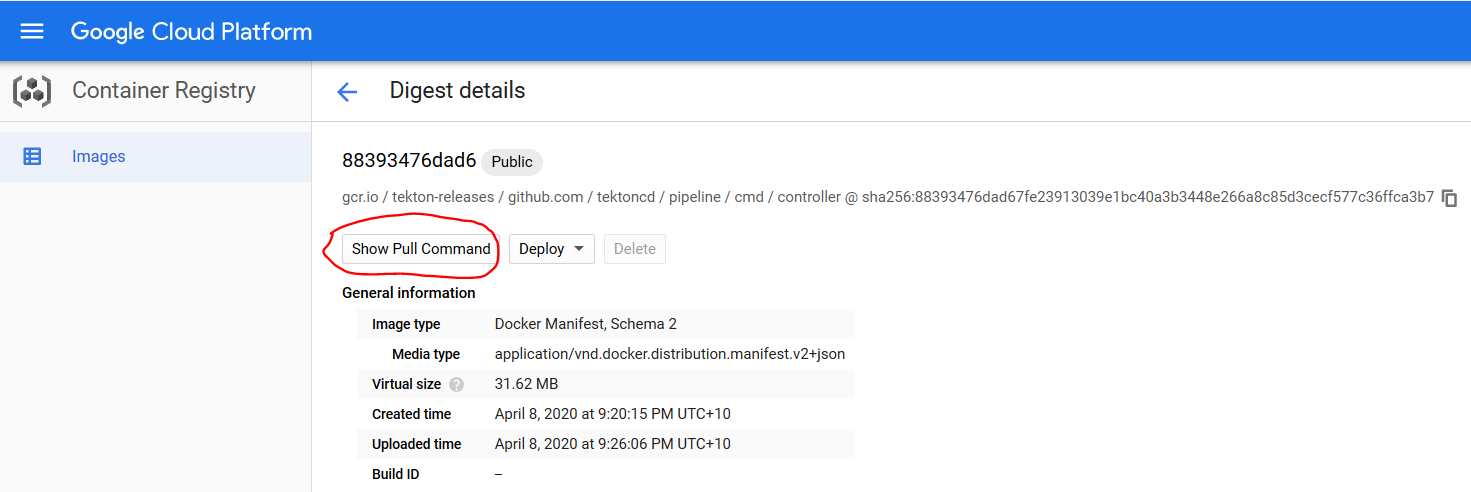
And you will be offered the option to copy the full image name



Or you can click on the



Then Click on show pull command

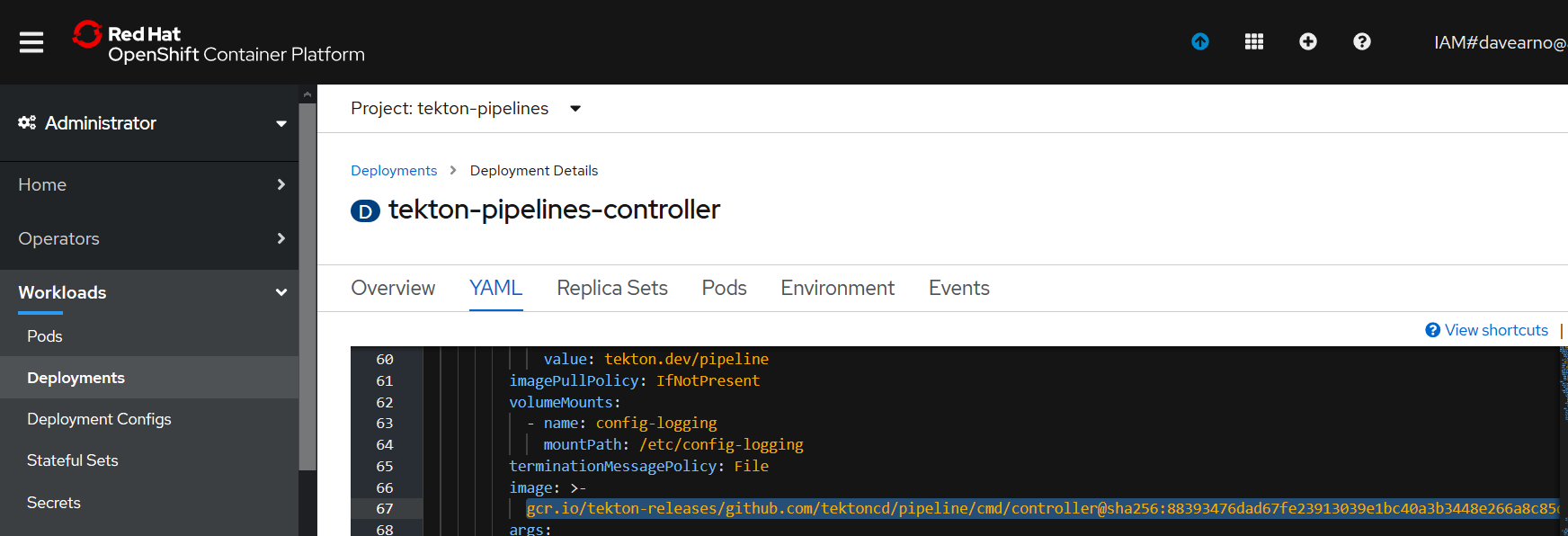


And copy the path to the image digest from there



Paste over the image name in the RH openshift Console->Workloads->deployments->tekton-pipelines-controller YAML

gcr.io/tekton-releases/github.com/tektoncd/pipeline/cmd/controller@sha256:88393476dad67fe23913039e1bc40a3b3448e266a8c85d3cecf577c36ffca3b7



Repeat the process for the other tekton images referenced in the YAML file

**kubeconfig-writer-image**

gcr.io/tekton-releases/github.com/tektoncd/pipeline/cmd/kubeconfigwriter@sha256:206c4e5de37d13c34f9538f87096db16433aadba264f24e9995cbb6b66fb67de

**creds-image**

gcr.io/tekton-releases/github.com/tektoncd/pipeline/cmd/creds-init@sha256:959b0d9a2d43d35e15a85460cc86567d308f467ee8ec16dbd9b32f51ce75d582

**git-image**

gcr.io/tekton-releases/github.com/tektoncd/pipeline/cmd/git-init@sha256:18ffa2bfc14b1fa6d39f62271beacf6bbc38e7cd2e255184dec477c2936270bc

**entrypoint-image**

gcr.io/tekton-releases/github.com/tektoncd/pipeline/cmd/entrypoint@sha256:cf0e81477c45dca0df6253e3239f6f0603700641292bf207503a7b267dc4c916

imagedigest-exporter-image

gcr.io/tekton-releases/github.com/tektoncd/pipeline/cmd/imagedigestexporter@sha256:1cf3f27f3ff7c73782d8a65853e8fc7f0d4aafc6443893e0150bdbe614a9169d

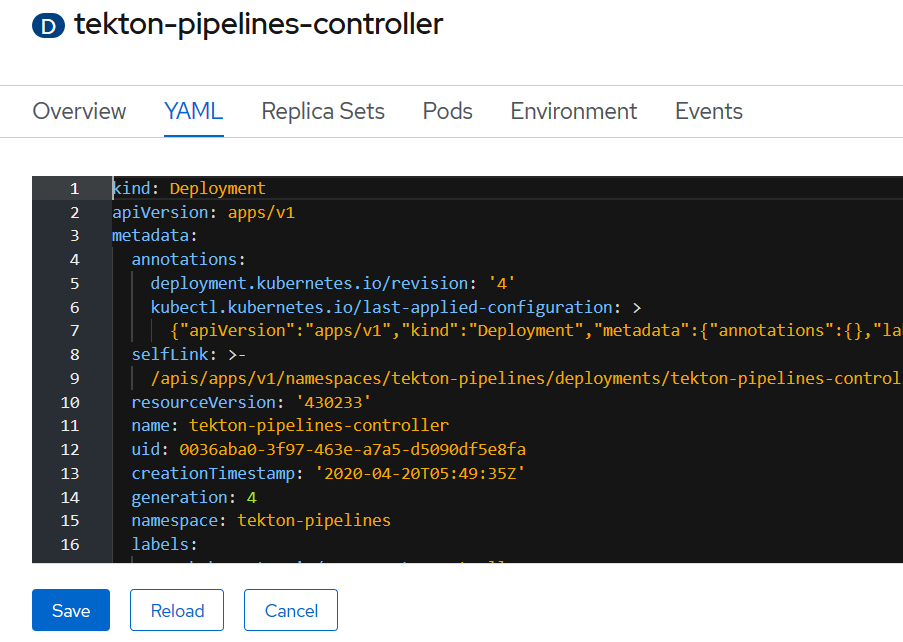
**pr-image**

gcr.io/tekton-releases/github.com/tektoncd/pipeline/cmd/pullrequest-init@sha256:3783254c379b286dd0987674160d3e19a95be1ccf0985788d6dcc0f159199095

**build-gcs-fetcher-image**

gcr.io/tekton-releases/github.com/tektoncd/pipeline/vendor/github.com/googlecloudplatform/cloud-builders/gcs-fetcher/cmd/gcs-fetcher@sha256:70f8d32a572496169d451130541541cbc99434932fd28beea486189af8a2995a

Save and reload



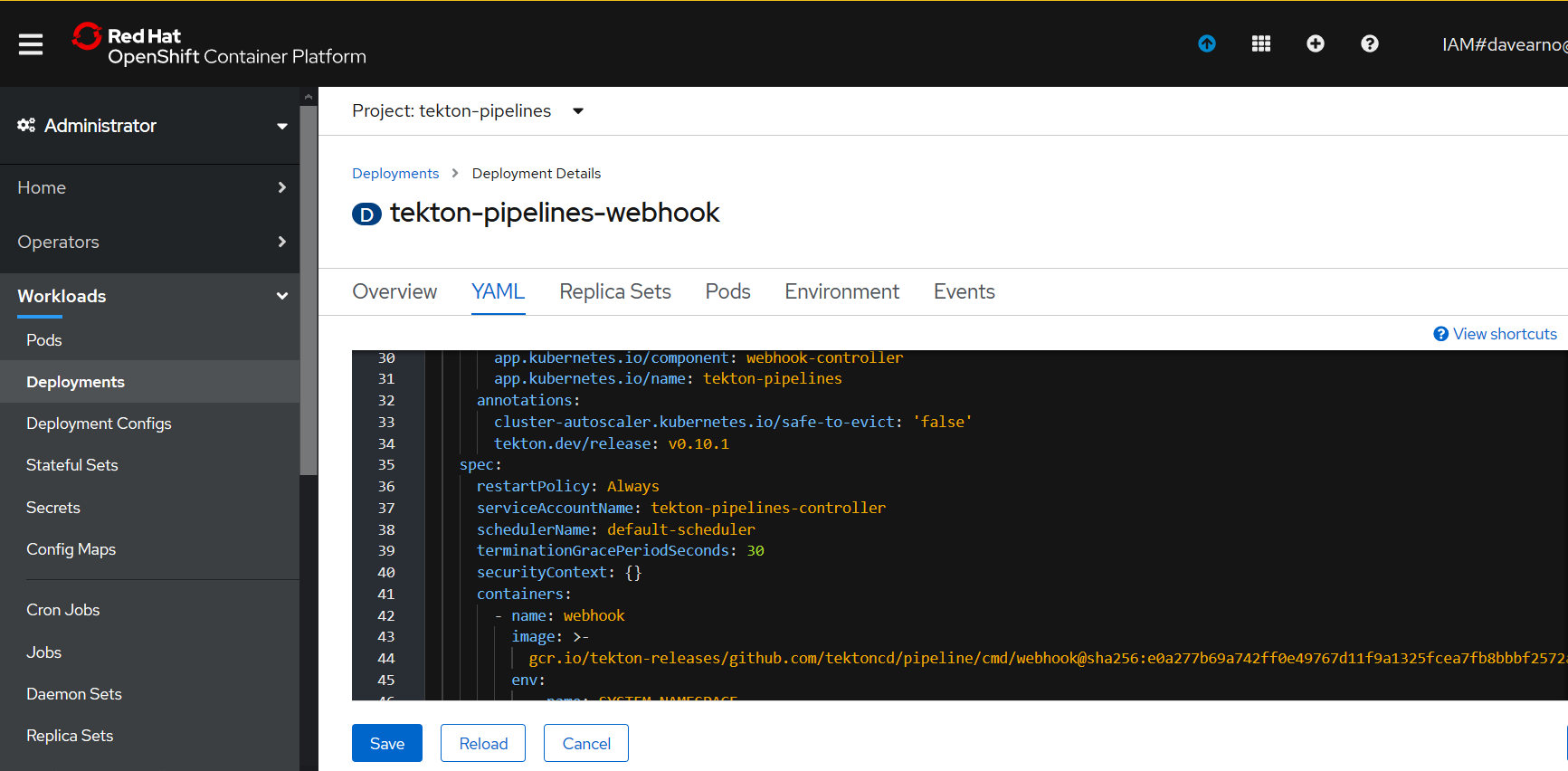
Check the controller pod terminates and a new pod starts.

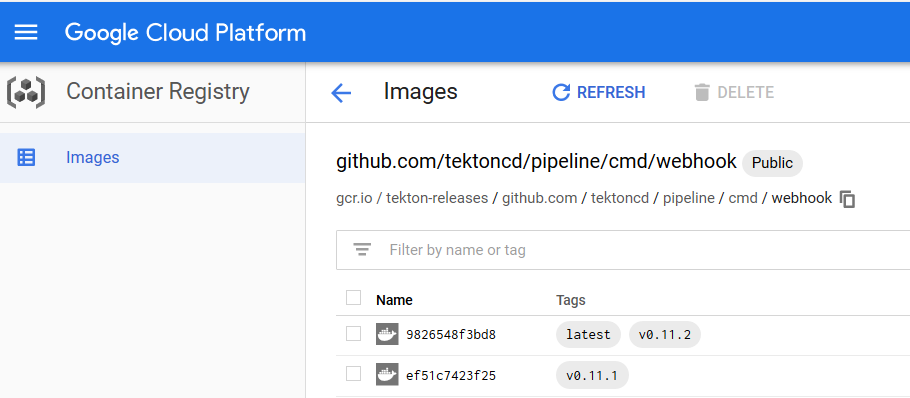


### Tekton Webhook

### NOTE: Leave the Webhook deployment alone i.e. v0.10.1 . I could not get later versions of the image to load. Instruction here just for completeness.

Repeat for tekton-pipelines-webhook



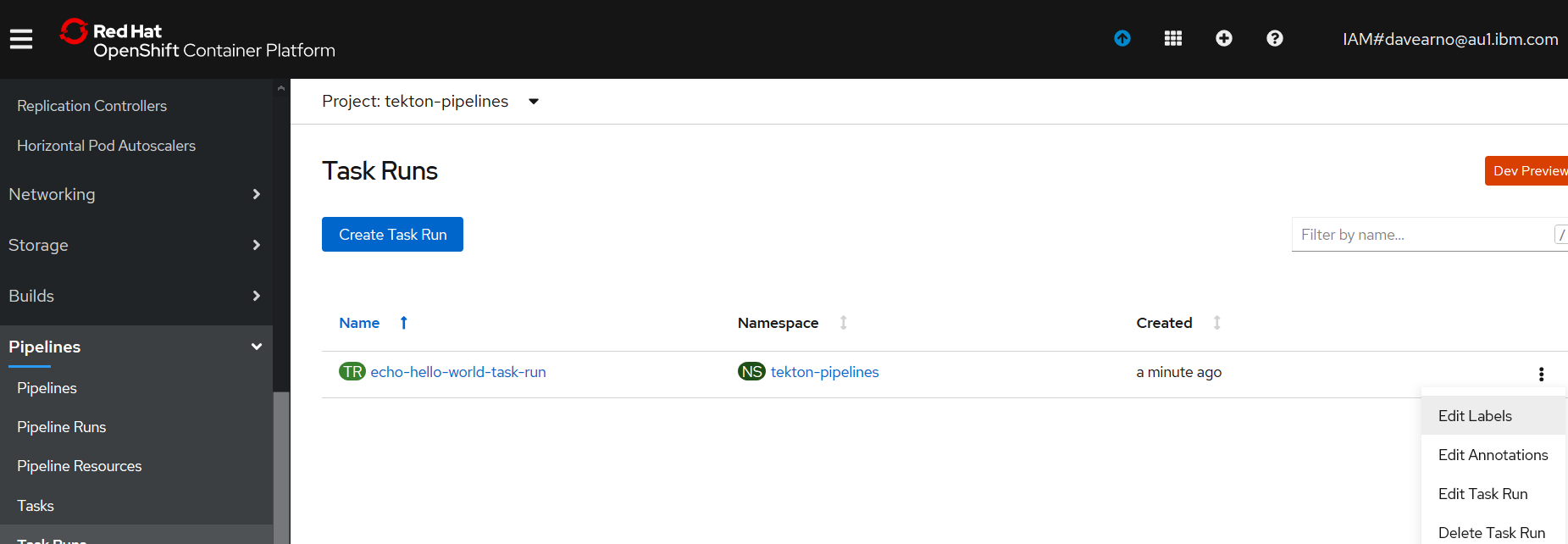


[gcr.io/tekton-releases/github.com/tektoncd/pipeline/cmd/webhook@sha256:ef51c7423f25b257ee63a4a5fb62c9df4721cabb5e34a516e03fb3f3b74f8c8f](mailto:gcr.io/tekton-releases/github.com/tektoncd/pipeline/cmd/webhook@sha256:ef51c7423f25b257ee63a4a5fb62c9df4721cabb5e34a516e03fb3f3b74f8c8f)

Save and reload.

### Test the Tekton Update

delete the Task run



**C:\openshift>oc apply -f Tekton-ivt\echo-hello-world-task-run.yaml**

taskrun.tekton.dev/echo-hello-world-task-run created

**C:\openshift>oc get taskruns/echo-hello-world-task-run -o yaml**