Kubernetes Cluster and Driverless AI:

Step 5: Publishing the local Docker Image to Cloud cluster

- **Tag the docker image built locally using IBM cloud container registry**

Create namespace

-> bx cr namespaces

-> $ bx cr namespace-add <give-a-name-space>

bx cr namespace-add rajeshyahoo

Adding namespace '**rajeshyahoo**'...

Successfully added namespace '**rajeshyahoo**'

OK

NOTE: Namespace name should be unique

-> bx cr namespace-list

Tag the image

Usage: docker image tag SOURCE_IMAGE[:TAG] TARGET_IMAGE[:TAG]

→ docker image tag <source image name> registry.ng.bluemix.net/<your namespace>/<image_name>

→ docker image tag opsh2oai/h2oai-runtime registry.eu-gb.bluemix.net/rajeshyahoo/h2oai

Validate:

$ docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
registry.eu-gb.bluemix.net/rajeshyahoo/h2oai latest 0c7621568523 3 weeks ago 5.51GB

At this stage, namespace is created and the image is tagged. Ready to publish it to cluster registry

- **Push the Image to the IBM cloud registry to your namespace**

→ docker image push registry.eu-gb.bluemix.net/rajeshyahoo/h2oai

NOTE:

- This command may take more than 30mins since around 2GB of image is been pushed to cluster registry
- do “bx cr login” if it fails with authentication error

```
OK
Rajeshs-MacBook-Air:~ rajeshjeyapaul$ docker image push registry.eu-gb.bluemix.net/rajeshyahoo/h2oai
The push refers to repository [registry.eu-gb.bluemix.net/rajeshyahoo/h2oai]
9daae604afaa: Pushed
e107bcf00b80: Pushed
22956e401269: Pushed
32b0b3d4203e: Pushed
ca28636a6e2c: Pushed
f67e765a78ca: Pushed
e6c743f302cd: Pushed
40b1c2a978b4: Pushed
fb46ab7348a8: Pushed
059c9d828e6c: Pushed
cce49f292fff: Pushed
1e1820b905e8: Pushed
246f99a13aa3: Pushed
427e24acf97b: Pushed
0665253dacfd: Pushed
bb2a3d55e9b4: Pushed
4275e52bbd64: Pushed
16116d824d32: Pushed
63807065abf3: Pushed
502ce1239273: Pushed
d794f4e084cb: Pushed
d13857ddd6f1: Pushed
e5942ab16586: Pushed
a3e015fa2305: Pushed
535fffe493af: Pushed
c25c06cb1710: Pushed
f7ffb2722b35: Pushed
0b3836475c71: Pushed
799eeadef665: Pushed
5a3054369621: Pushed
12fc8d106ee5: Pushed
3e4cf5bf65c2: Pushed
2f5b0990636a: Pushed
c9748fbf541d: Pushed
b3968bc26fbd: Pushed
aa4e47c45116: Pushed
788ce2310e2f: Pushed
latest: digest: sha256:8c9598cb2203deba08d8093fa0b7b4f7e0ad002e5ffde7fd55178657a0f51b58 size: 8085
Rajeshs-MacBook-Air:~ rajeshjeyapaul$
```

- **Validate the image push**

→ bx cr images
Listing images...

REPOSITORY NAMESPACE TAG DIGEST CREATED SIZE VULNERABILITY STATUS

registry.eu-gb.bluemix.net/rajeshyahoo/h2oai rajeshyahoo latest 8c9598cb2203 3 weeks ago 2.2 GB **Vulnerable**

OK

- **Run, Configure and Deploy**

→ kubectl run h2oai --image=registry.eu-gb.bluemix.net/rajeshyahoo/h2oai

→ kubectl get pods

```
NAME                 READY     STATUS    RESTARTS   AGE
h2oai-6695d7b455-677gz   1/1       Running   0          4m
```

Note: At this stage, a pod is been created in the Kubernetes cluster

→kubectl describe nodes

```
Namespace                  Name
        CPU Requests       CPU Limits   Memory Requests   Memory Limits
  ---------                ----
        ------------    ----------      ---------------  -------------
  default                  h2oai-6695d7b455-677gz
                 2 (50%)              2 (50%)              6Gi (40%)     6Gi (40%)
```

Get into container shell to create a folder, /data

→ kubectl exec -it h2oai-6695d7b455-677gz -- /bin/bash

mkdir /data
exit

Deploy

→ $ kubectl expose deployment/h2oai --type=NodePort --name=h2oai-service --port=12345
service "h2oai-service" exposed

→ $ kubectl describe service h2oai-service

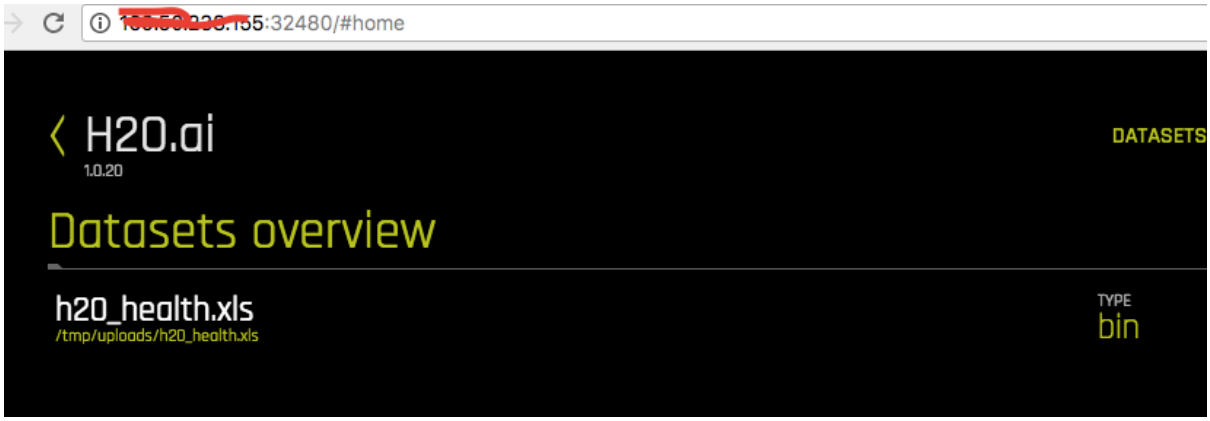
```
Name:                   h2oai-service
Namespace:              default
Labels:                         run=h2oai
Selector:               run=h2oai
Type:                   NodePort
IP:                     172.21.96.153
```

Port: <unset> 12345/TCP
NodePort: <unset> **32480**/TCP
Endpoints: 172.30.221.170:12345
Session Affinity: None
No events.

- **Run the application and validate the ML Model**

→ $ bx cs workers mycluster
OK
ID Public IP Private IP Machine Type State Status Zone Version
kube-lon02-cr31ad1e21d25742c7af98f70f59423d60-w1 xx.yy.238.155 10.165.58.241 b2c.4x16.encrypted normal Ready lon02 1.9.3_1502

Take the public IP assigned to the cluster and the NodePort to access the image.
http://xx.yy.238.155:32480



Note: Anytime, if there is an error as below, Run - `bx cs cluster-config mycluster`

```
The connection to the server localhost:8080 was refused - did you
specify the right host or port?
Rajeshs-MacBook-Air:~ rajeshjeyapaul$ bx cs cluster-config mycluster
OK
The configuration for mycluster was downloaded successfully. Export
environment variables to start using Kubernetes.

export KUBECONFIG=/Users/rajeshjeyapaul/.bluemix/plugins/container-
service/clusters/mycluster/kube-config-lon02-mycluster.yml
```

Congrats !! At this stage, if you are able to access the H2O.ai driver page, then your installation is complete.

Summary:

1) Prepare the Docker environment
2) Install the H2OAI driver and validate the driver with the sample data
3) Create Kubernetes cluster
4) Configure IBM Cloud CLI and Kubernetes CLI
5) Create the tag and push the image to cluster

Now, proceed to Model evaluation using H2O.AI