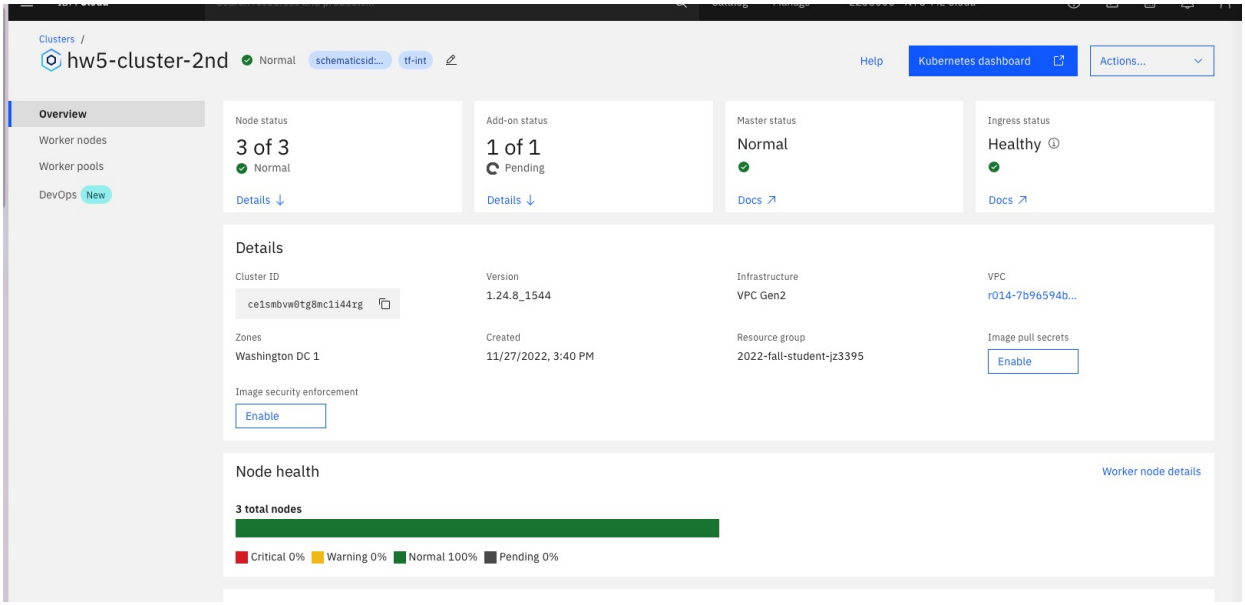
Wayne Zhu
11/27/22

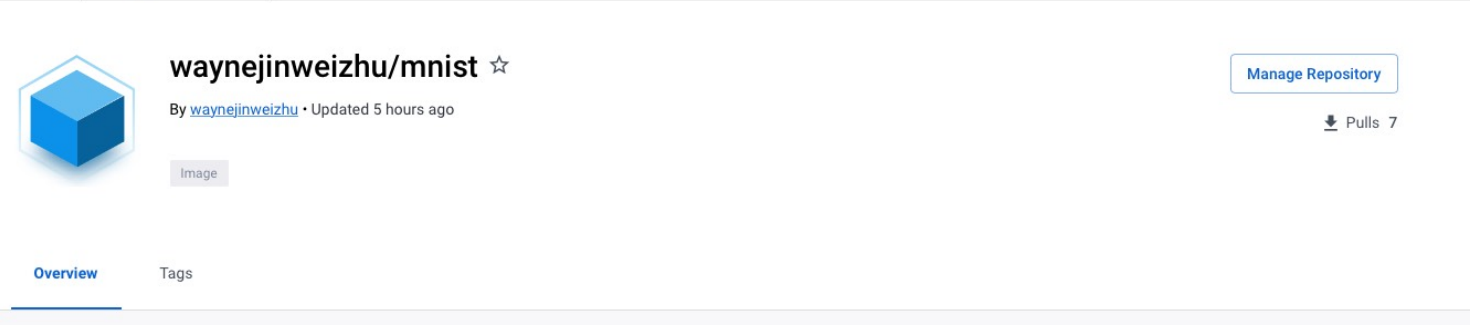
Steps to launch the k8s cluster apps:

1. I followed the instruction outlined in slide 9 to initialize a VPC and a K8s cluster on IBM Cloud.



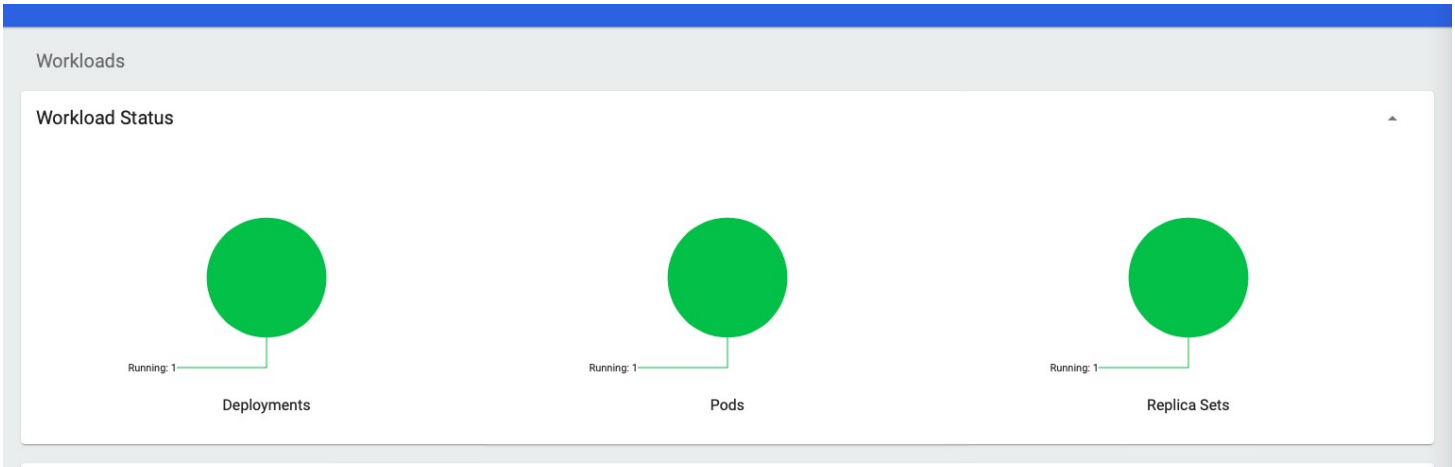
2. I modified the Dockerfile of HW3 to train the model locally in a docker container on my VM as in HW3.

3. I push that image, which takes up 5GB, to Docker Hub.



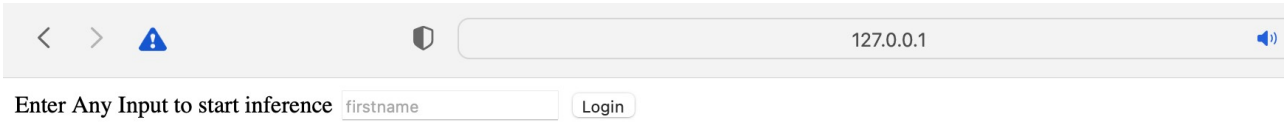
4. Upon encountering Ingress status warning and PullImageOff error during deployment, I had to restart my k8s cluster in 1 zone and attach my subnet to a public gateway.

5.Then, I deploy my mnist container on k8s using the cloud interface, so it saves me the time to handwrite a k8s manifest yaml file.



7. I couldn't find the model.save output mnist_cnn.pt for a long time. Eventually, I realized it's in the container's directory, and once the container finishes running it's gone. So I let my python program sleep while I docker copy the file into my VM. This approach works.

8. I create a seperate directory called mnistinference, revised the dockerfile, added flask process and deployed it on my local VM. The user will see this:

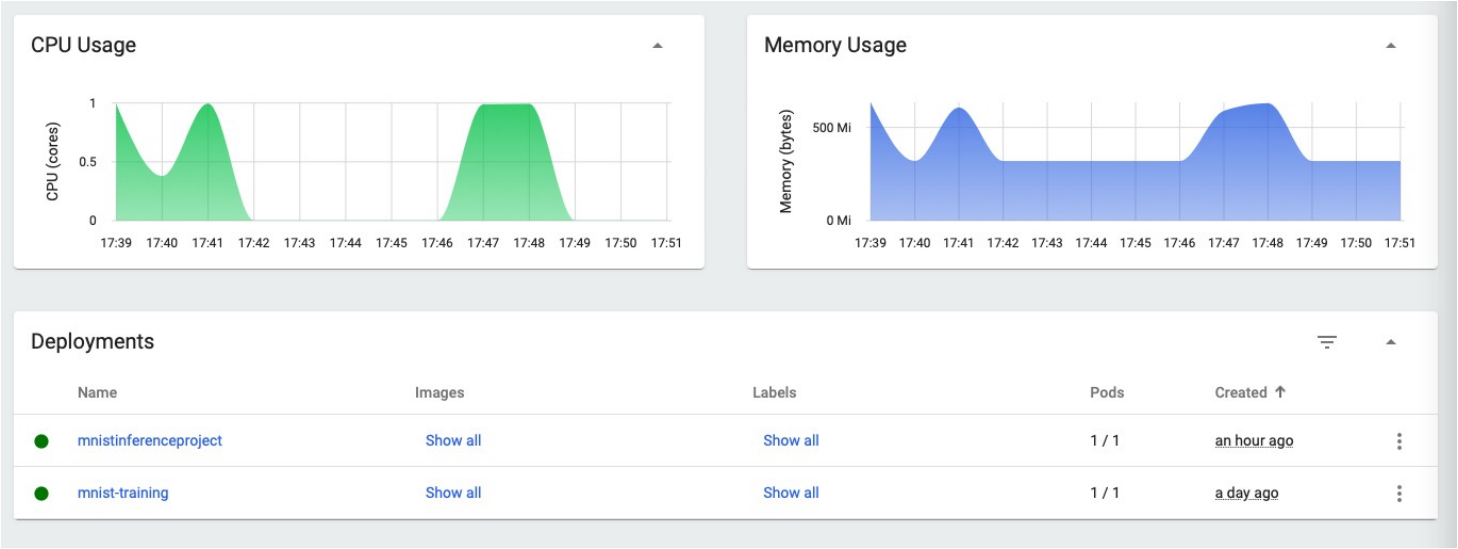


9. Once you enter anything, the app will load the mnist_cnn.pt and conduct the rest of the inference.

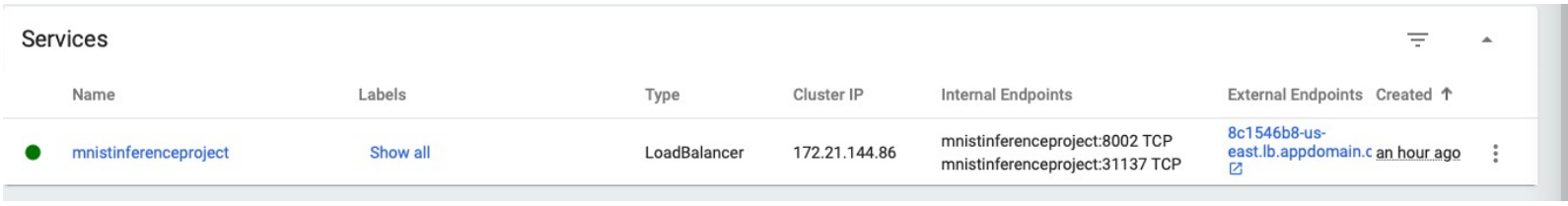
```
10.0.2.2 - - [28/Nov/2022 22:38:20] "GET / HTTP/1.1" 200 -
Downloading http://yann.lecun.com/exdb/mnist/train-images-idx3-ubyte.gz
Downloading http://yann.lecun.com/exdb/mnist/train-images-idx3-ubyte.gz to ../data/MNIST/raw/train-images-idx3-ubyte.gz
100.0%
Extracting ../data/MNIST/raw/train-images-idx3-ubyte.gz to ../data/MNIST/raw

Downloading http://yann.lecun.com/exdb/mnist/train-labels-idx1-ubyte.gz
Downloading http://yann.lecun.com/exdb/mnist/train-labels-idx1-ubyte.gz to ../data/MNIST/raw/train-labels-idx1-ubyte.gz
100.0%
Extracting ../data/MNIST/raw/train-labels-idx1-ubyte.gz to ../data/MNIST/raw

Downloading http://yann.lecun.com/exdb/mnist/t10k-images-idx3-ubyte.gz
Downloading http://yann.lecun.com/exdb/mnist/t10k-images-idx3-ubyte.gz to ../data/MNIST/raw/t10k-images-idx3-ubyte.gz
100.0%
Extracting ../data/MNIST/raw/t10k-images-idx3-ubyte.gz to ../data/MNIST/raw

Downloading http://yann.lecun.com/exdb/mnist/t10k-labels-idx1-ubyte.gz
Downloading http://yann.lecun.com/exdb/mnist/t10k-labels-idx1-ubyte.gz to ../data/MNIST/raw/t10k-labels-idx1-ubyte.gz
100.0%
Extracting ../data/MNIST/raw/t10k-labels-idx1-ubyte.gz to ../data/MNIST/raw


Test set: Average loss: 0.0553, Accuracy: 9823/10000 (98%)


Test set: Average loss: 0.0553, Accuracy: 9823/10000 (98%)


Test set: Average loss: 0.0553, Accuracy: 9823/10000 (98%)
```

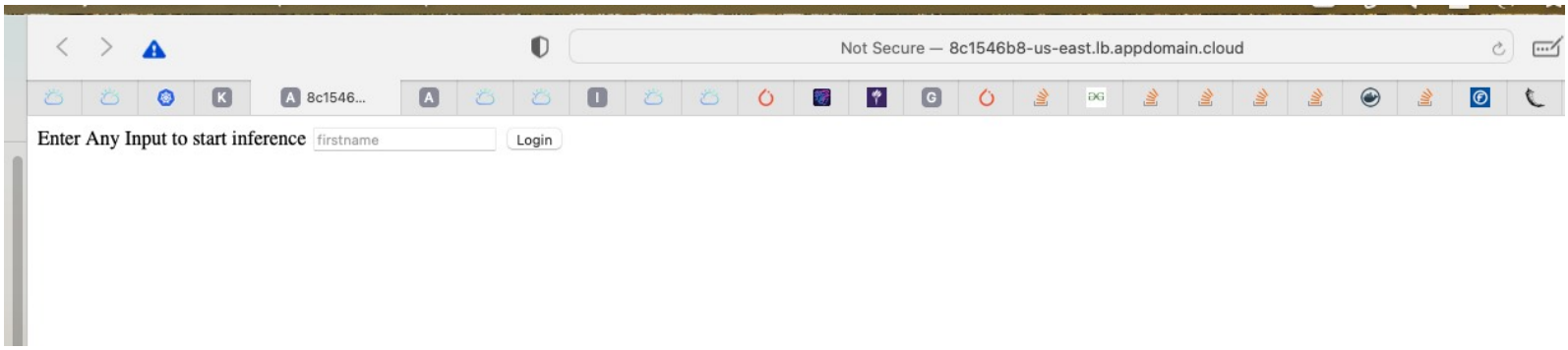
10. Then, upon the successful test locally, I pushed the image to docker hub.

11. Then, I deploy the deployment and service on k8s dashboard.

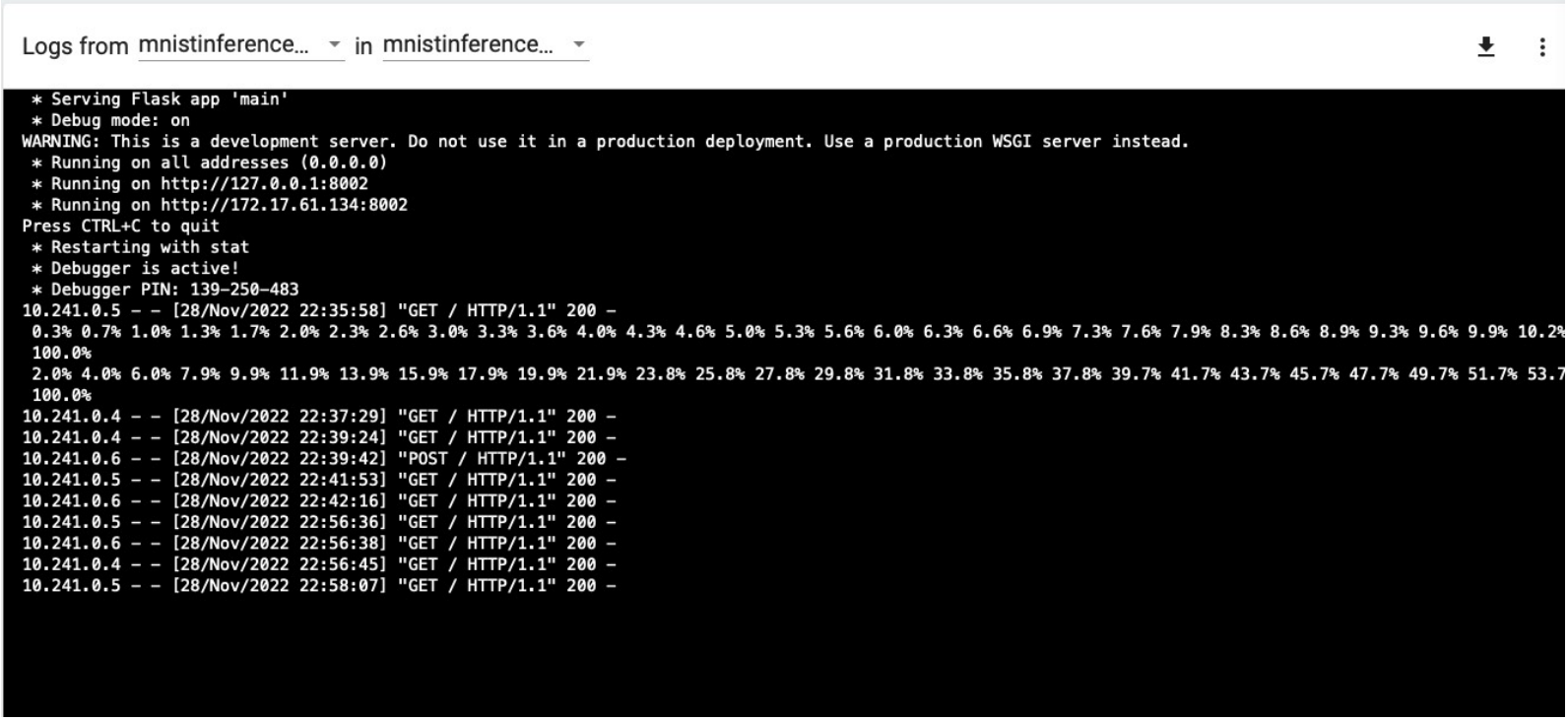


12. Then I was able to repeat my local webpage inference process using the external URL given by k8s: http://8c1546b8-us-east.lb.appdomain.cloud:8002





13. Then, upon inputing sth on the website, I can see that some unique log that is not seen anywhere else is generated on the k8s replicaset log. I am not sure why this output is different from the output when I run it on my local VM docker. But I didn't print any output explicitly so this might be why. But you can clearly see that 0.3% 0.7%…. reflects loading the model. Also note that only the first POST request will generate inference response. U need to restart afterward to do this again.



Controller usage:

For training, I used only 1 replicaset, 1 replicaset controller and a deployment. A deployment is for k8s. This is not a high security application, so 1 replication save resources.

For inference, I used only 1 replicaset, 1 replicaset controller, a deployment and a service. A deployment is for k8s, while a service is for the web interface. This is not a high security application, so 1 replication save resources.

Overall: This project takes me 20 hours. I hope the instruction could be more complete so it wouldn't be that time consuming. But overall, I learned a lot about k8s.