Goal 3: Turn my code (stock-rsi.py) into a containerized application, deploy it in Kubernetes (and prove it works!)

Turning a python program into a docker container is fairly straightforward. I used this guide as a starting point: https://www.docker.com/blog/containerized-python-development-part-1/

I built containers on my laptop and the resulting container works seamlessly on any other system that supports docker (including Kubernetes).

Here is a link to my build directory, including my Dockerfiles, and all dependent files: https://github.com/olivelawn/cs199-indep-k8s/tree/main/container-stock-rsi

A Dockerfile contains the instructions on how to build a container image. In my case, I needed to instruct docker to build an image that includes all of the modules (yfinance, pandas, lumpy, mysql.connector, etc) that my program relies on. For this there is a list of dependent modules in requirements.txt and a call to pip ("RUN pip install -r requirements.txt") to install them. Additionally, I needed to add into my image information on how to connect to the mysql db (config.py: connection string, credentials), add my file with the list of stock symbols (i.e. "symbols") and then finally instructions on how to execute the python program itself ("CMD ["python", "/app/stock-rsi.py", "symbols"]").

This command will build the docker container:

```
[parker@parker container-stock-rsi]$ docker build -f Dockerfile -t pjohn07/stock-rsi-k8s:latest .
[+] Building 3.0s (11/11) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 199B
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [internal] load metadata for docker.io/library/python:3.7
 => [auth] library/python:pull token for registry-1.docker.io
 => [1/5] FROM docker.io/library/python:3.7@sha256:354a4c2400dd1c1293b657bcf059742e45f7c6222dc908fc63ba4538740805e5
 => [internal] load build context
 => => transferring context: 154B
 => CACHED [2/5] RUN mkdir /app
 => CACHED [3/5] WORKDIR /app
 => CACHED [4/5] ADD . /app/
 => CACHED [5/5] RUN pip install -r requirements.txt
 => exporting to image
 => => exporting layers
 => => writing image sha256:a00ec65018021351d94858f52cff3dc2dc674036cfd1b29c9235c83c32a6e828
 => => naming to docker.io/pjohn07/stock-rsi-k8s:latest
```

(The '-t' option specifies a tag, which in my case is "pjohn07/stock-rsi-k8s" "pjohn07" is my username on dockerhub, which is where I store my images—and where Kubernetes looks by default to retrieve them)

Afterwards you can list the image and push it to dockerhub:

```
[parker@parker container-stock-rsi]$ docker image ls
REPOSITORY                TAG       IMAGE ID       CREATED        SIZE
pjohn07/stock-rsi-k8s     latest    a00ec6501802   40 hours ago   1.16GB
pjohn07/hello-python      latest    1aec00ffb622   2 days ago     887MB

[parker@parker container-stock-rsi]$ docker push pjohn07/stock-rsi-k8s
Using default tag: latest
The push refers to repository [docker.io/pjohn07/stock-rsi-k8s]
74332da7ee5d: Layer already exists
a8d3f57df352: Layer already exists
5f70bf18a086: Layer already exists
420960fa9cc9: Layer already exists
df58ce8478ec: Layer already exists
latest: digest: sha256:d14015656d597b3f0ac5b065e7aa3e716489daf3e797e3fd592a1b6123254b06 size: 3051
```

Here is stock-rsi.py on docker hub: https://hub.docker.com/repository/docker/pjohn07/stock-rsi-k8s

Next up is to deploy the container/pod into my Kubernetes cluster and insure it works.

At this point only mysql is running in the cluster.

```
parker@c1-master1:~$ kubectl get deployments.apps ; kubectl get svc
NAME    READY   UP-TO-DATE   AVAILABLE   AGE
mysql   1/1     1            1           18h
NAME         TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)    AGE
mysqlsvc     ClusterIP   10.109.20.156   <none>        3306/TCP   18h
```

Currently there are 146 rows in our "stocks" table:

```
mysql> select COUNT(date) from stocks;
+-------------+
| COUNT(date) |
+-------------+
|         146 |
+-------------+
1 row in set (0.03 sec)
```

Next I define a Kubernetes deployment for my stock-rsi container in yaml as seen here: https://github.com/olivelawn/cs199-indep-k8s/blob/main/k8s-configs/stock-rsi-k8s/stock-rsi-deploy.yaml
Notice I reference my docker hub image with "image: pjohn07/stock-rsi-k8s"

Let's deploy the container:

```
parker@c1-master1:~/stock-rsi-k8s$ kubectl apply -f stock-rsi-deploy.yaml
deployment.apps/stock-rsi created
```

Let’s watch the pod startup and go from Pending, Creating, and Running status:

```
parker@c1-master1:~/stock-rsi-k8s$ while true; do kubectl get pods; sleep 10; done
NAME                         READY   STATUS              RESTARTS   AGE
mysql-68c777748-dvtjl        1/1     Running             1          19h
stock-rsi-6f778764b5-kj5jt   1/1     ContainerPending    0          8s
NAME                         READY   STATUS              RESTARTS   AGE
mysql-68c777748-dvtjl        1/1     Running             1          19h
stock-rsi-6f778764b5-kj5jt   1/1     ContainerCreating   1          18s
NAME                         READY   STATUS              RESTARTS   AGE
mysql-68c777748-dvtjl        1/1     Running             1          19h
stock-rsi-6f778764b5-kj5jt   1/1     Running             1          28s
```

As it turns out, inspecting stdout from inside the container is something of a challenge. It is possible, but you have to catch the container in just the right state (just after the container itself loads and just before python is kicked off) and attach a shell to the container with this command “kubectl exec -it stock-rsi-6f778764b5-kj5jt -- /bin/bash” (I tried several times without any luck). Instead, it is easier to have a look at the changes made to mysql “stocks” table.

```
mysql> select COUNT(date) from stocks;
+-------------+
| COUNT(date) |
+-------------+
|         147 |
+-------------+
1 row in set (0.33 sec)
```

Notice our row count increased by 1 (from 146 to 147) Just to be sure, let’s have a look at the data for today’s date:

```
parker@c1-master1:~/mysql$ date
Fri May 21 15:47:45 UTC 2021

mysql> select * from stocks where date = '2021-05-21’;
| date       | AAGIY | AAPL  | ABT   | ADBE  | ADDYY | AER   | AES   | AIG   | AKAM  | ALL   | AMADY | AMAT  | AMGN  | AMZN  |
ASML | ATVI  | AVGO  | AXAHY | AXP   | BABA  | BAC   | BAESY | BASFY | BAX   | BK    | BKNG  | BLK   | BMY   | BND   | BNDX  |
BUD   | BURBY | C     | CABGY | CAH   | CARR  | CAT   | CHKP  | CI    | CMCSA | COP   | COST  | CRARY | CRM   | CSCO | CSX   | CVS
| CVX   | DANOY | DBSDY | DD    | DEO   | DIS   | DLTR  | DNHBY | DOX   | DPSGY | DUK   | EADSY | EBKDY | EDU   | EMB   | EQR   |
ESLOY | EW    | F     | FB    | FIS   | FMC   | FTI   | GLW   | GOOG  | GOOGL | GWW   | HAL   | HD    | HON   | HSBC | HTHIY |
HUM   | ICLR  | IDEXY | IDXX  | ILMN  | INFO | INTC  | IOO   | ITOT  | IWY   | JBAXY | JETS  | JNJ   | JPM   | LGLV  | LGRDY |
LIN   | LNSTY | LVMUY | LVS   | LYG   | MDLZ  | MET   | MO    | MPC   | MRK   | MRO   | MS    | MSCI  | MSFT  | NEE   | NGLOY | NKE
| NOC   | NOW   | NRDBY | NSRGY | NTNX  | NVDA  | NVS   | ORCL  | OTIS | OXY   | PEP   | PFE   | PG    | PM    | PNGAY | PRU
| PXD   | PYPL  | QCOM  | QQQ   | RCI   | RDSMY | RHHBY | RTX   | SAP   | SCHW | SNN   | SPLK  | SQ    | SSDOY | STNE | SU    | T
| TCOM  | TFC   | TMO   | TMSNY | TRYIY | TSM   | TSN   | TXN   | UBER  | UMICY | UNH   | UNM   | UNP   | UPS   | V     | VDE   |
VLO   | VMW  | VNQ   | VNQI  | VTI   | VZ    | WFC   | WMB   | ZTS   | ABNB  | MCHI  | VB    | VO    | VTRS | VWO   | ARKK |
CNRG  | IWR   | JKH   | MSSMX | SCHD  | IMCG  | THNPY | TDUP  | ABBV  | ACN   | CHTR | DOW   | EXC   | GD    | GS    | IEMG  |
KHC   | LLY   | LMT   | MCD   | NFLX | SCHF  | SO    | SPG   | TFI   | TGT   | VEA   | VOO   | VTEB | VXF   | WMT   |
| 2021-05-21 | 55.73 | 46.06 | 49.5 | 50.76 | 71.19 | 46.91 | 42.05 | 62.11 | 68.99 | 70.48 |  NULL | 51.92 | 55.43 | 44.75 |
51.5 | 56.85 | 50.48 | 48.03 | 62.41 | 37.68 | 63.03 |  NULL | 45.86 | 46.23 | 60.71 | 47.01 | 61.99 | 70.77 | 50.09 | 39.91 |
68.58 |  NULL | 65.49 |  NULL | 48.3 | 56.82 | 55.04 | 53.63 | 61.8 | 47.44 | 55.11 | 59.81 |  NULL | 54.1 | NULL | 44.08 |
81.19 | 47.82 |  NULL | 58.04 | 66.2 | NULL | 32.09 | 38.39 | 55.18 | 55.28 | 65.44 | 57.98 | 51.31 | 80.44 | 27.72 | 53.91 |
57.51 |  NULL | 63.61 | 66.61 | 58.38 | 53.28 | 54.04 | 55.03 | 46.62 | 57.71 | 55.99 | 61.67 | 55.43 | 44.62 | 50.18 | NULL |
NULL | 54.41 | 68.63 |  NULL | 57.57 | 59.36 | NULL | 42.02 | 55.39 | 54.22 | 50.99 |  NULL | 51.34 | 68.55 | 61.26 | 56.34 |
67.43 | 60.88 |  NULL | 59.18 | NULL | NULL | 72.12 | 54.42 | 54.35 | NULL | 61.88 | 53.47 | 63.6 | 47.22 | 47.83 | 49.16 |
NULL | 47.66 | 66.35 | 43.17 |  NULL | 68.75 | 62.12 | 56.56 | 54.36 | 61.83 | NULL | 48.87 | 60.94 | 65.12 | 61.31 | 60.79 |
33.61 | 65.29 | 47.19 | 50.53 | 47.27 | 49.98 | NULL | 52.81 | 61.33 | 65.05 | 55.21 | NULL | 64.74 | 44.94 | 36.7 |  NULL |
NULL | 56.91 | 41.5 | 56.87 | 56.9 | 50.86 |  54.7 |  52.4 | 45.08 | 64.9 | 52.17 | 45.08 |  NULL | 66.03 | 56.98 | 50.11 |
69.71 | 53.73 | 56.26 | 54.26 | NULL | 58.21 | 54.45 | 54.17 | 40.96 | 57.29 | 69.3 | 63.42 | 28.83 | 45.15 | 50.77 | 55.11 |
64 | 48.58 | 42.5 | 47.36 | 54.21 | NULL | 43.24 | 59.35 | 51.46 | 53.67 | 58.45 | 65.63 | 49.54 |    62 | 60.07 | 60.64 | 54.6
| 59.63 | 48.28 | 68.87 | 69.43 | 56.46 | 50.55 | 48.5 | 57.35 | 46.73 | 56.73 | 49.45 | 70.18 | 57.35 | 55.1 | 55.7 | 50.03 |
59.71 |
```

This proves that my container running my stock-rsi.py program is able to pull stock data from the internet, crunch some numbers, and insert that data into a separate container running mysql.