Christopher Shi
33061961
cshi@umass.edu

Lab 2 Output Document
CS677 Distributed and Operating Systems

2023-03-24

1 Output of Latency Tests:

```
1,2,3,4,5,6 client test. Demonstrates that the front end service runs as a thread per session approach
elnux2 lab2) > ./run.sh
6 Clients
Average Latency: 0.06642975409825642
Average Latency: 0.0655673370239841
Average Latency: 0.06710173083889868
Average Latency: 0.06719034231161769
Average Latency: 0.06815959551395515
Average Latency: 0.12863602046093908

5 Clients
Average Latency: 0.06557115198860705
Average Latency: 0.06556636417234266
Average Latency: 0.06744622704166695
Average Latency: 0.06754021455120567
Average Latency: 0.06453649597878781

4 Clients
Average Latency: 0.06528948909706539
Average Latency: 0.06650034197560557
Average Latency: 0.0646963519538009
Average Latency: 0.0660366082341416

3 Clients
Average Latency: 0.06735441152998012
Average Latency: 0.06579912925253109
Average Latency: 0.06582739546492293

2 Clients
Average Latency: 0.06510136855973138
Average Latency: 0.06597395420074463

1 Clients
Average Latency: 0.06502459407631868
elnux2 lab2) >


1,2,3,4,5 clients lookup only. hosted on docker VM:
elnux2 lab2) > ./run.sh
5 Clients
Average Latency: 0.06417081594467162
Average Latency: 0.06623378992080689
Average Latency: 0.06627068519592286
Average Latency: 0.06688476085662842
Average Latency: 0.0675485920906067

4 Clients
Average Latency: 0.06365367174148559
Average Latency: 0.06385019779205323
Average Latency: 0.06425481081008912
Average Latency: 0.06535226583480835

3 Clients
```

```
Average Latency: 0.0642075490951538
Average Latency: 0.06439903497695923
Average Latency: 0.06477782011032104

2 Clients
Average Latency: 0.06292051792144776
Average Latency: 0.06352087497711181

1 Clients
Average Latency: 0.06307222604751588
elnux2 lab2) >


1,2,3,4,5 clients trade only. hosted on docker VM:
elnux2 lab2) > ./run.sh
5 Clients
Average Latency: 0.06706074953079223
Average Latency: 0.0675226902961731
Average Latency: 0.06854172468185425
Average Latency: 0.06932191610336304
Average Latency: 0.07051957607269287

4 Clients
Average Latency: 0.06947309017181397
Average Latency: 0.07088325500488281
Average Latency: 0.07140974044799804
Average Latency: 0.07186569929122925

3 Clients
Average Latency: 0.06828480958938599
Average Latency: 0.06946743726730346
Average Latency: 0.07202903032302857

2 Clients
Average Latency: 0.06728380680084228
Average Latency: 0.0683591890335083

1 Clients
Average Latency: 0.07101038932800292
elnux2 lab2) >


1,2,3,4,5 clients lookup and trade. hosted on docker VM. (p=0.5):
elnux2 lab2) > ./run.sh
5 Clients
Average Latency: 0.06674920133992929
Average Latency: 0.06644911607106527
Average Latency: 0.06634038647279045
Average Latency: 0.06627201724362064
Average Latency: 0.06847892780052989

4 Clients
Average Latency: 0.06712995138433245
Average Latency: 0.06614967445274453
Average Latency: 0.0668612594728346
Average Latency: 0.06853837626320976

3 Clients
Average Latency: 0.06542890270551045
Average Latency: 0.06382932598958879
Average Latency: 0.06654680239689814
```

```
2 Clients
Average Latency: 0.0655783832847298
Average Latency: 0.06499463931108132

1 Clients
Average Latency: 0.0658747771891152
elnux2 lab2) >


1,2,3,4,5 clients lookup only. hosted on naive computer:
elnux2 lab2) > ./run.sh
5 Clients
Average Latency: 0.04435750246047974
Average Latency: 0.04439772605895996
Average Latency: 0.044757723808288574
Average Latency: 0.0460951566696167
Average Latency: 0.046717052459716794

4 Clients
Average Latency: 0.044660289287567136
Average Latency: 0.045778648853302004
Average Latency: 0.04627708911895752
Average Latency: 0.046452808380126956

3 Clients
Average Latency: 0.04330036163330078
Average Latency: 0.043649990558624265
Average Latency: 0.044895312786102294

2 Clients
Average Latency: 0.04294870138168335
Average Latency: 0.044201779365539554

1 Clients
Average Latency: 0.04136239051818848


1,2,3,4,5 clients trade only. hosted on naive computer:
elnux2 lab2) > ./run.sh
5 Clients
Average Latency: 0.05693872928619385
Average Latency: 0.05730640649795532
Average Latency: 0.057508246898651125
Average Latency: 0.058334386348724364
Average Latency: 0.06005401849746704

4 Clients
Average Latency: 0.0473453426361084
Average Latency: 0.051038458347320556
Average Latency: 0.05205904722213745
Average Latency: 0.05213435173034668

3 Clients
Average Latency: 0.047065963745117186
Average Latency: 0.04709573984146118
Average Latency: 0.04826335668563843

2 Clients
Average Latency: 0.04532717704772949
Average Latency: 0.047648122310638426
```

```
1 Clients
Average Latency: 0.044635841846466066
elnux2 lab2) >


1,2,3,4,5 clients lookup and trade. hosted on naive computer. (p=0.5):
elnux2 lab2) > ./run.sh
5 Clients
Average Latency: 0.045236916769118535
Average Latency: 0.04724589439287578
Average Latency: 0.04727411106841205
Average Latency: 0.04685761439089744
Average Latency: 0.04726266553325038

4 Clients
Average Latency: 0.046906200244272354
Average Latency: 0.04515150981613352
Average Latency: 0.04841959005907962
Average Latency: 0.050258150287702974

3 Clients
Average Latency: 0.04841995392090235
Average Latency: 0.048825728587615184
Average Latency: 0.05061060071780982

2 Clients
Average Latency: 0.0450793263074514
Average Latency: 0.04512343956873967

1 Clients
Average Latency: 0.04427638730487308
elnux2 lab2) >
```

2 Output of Test Cases:

```
C:\Users\thewi\Desktop\CS677\lab2\src\test-cases>py testcases.py 10.0.0.246 56893
########################################
##Test Case 1 - Lookup Functionality: ##
########################################
Input Request: GET /stocks/nvidia
JSON Reply: {'data': {'name': 'nvidia', 'price': 240.63, 'quantity': 880}}

############################################################
##Test Case 2 - Lookup Error Handling (Stock not found): ##
############################################################
Input Request: GET /stocks/imaginarycompany
JSON Reply: {'error': {'code': 404, 'message': 'stock not found'}}

#####################################
##Test Case 3 - Sell Functionality: ##
#####################################
Input Request: POST /orders
Input JSON: {"name": "intel", "type": "sell", "quantity": 10}
Stock Information before Request: {'data': {'name': 'intel', 'price': 28.01, 'quantity': 1399}}
JSON Reply: {'data': {'transaction_number': 278}}
Stock Information after Request: {'data': {'name': 'intel', 'price': 28.01, 'quantity': 1409}}

####################################
##Test Case 4 - Buy Functionality: ##
```

```
####################################
Input Request: POST /orders
Input JSON: {"name": "apple", "type": "buy", "quantity": 10}
Stock Information before Request: {'data': {'name': 'apple', 'price': 152.59, 'quantity': 959}}
JSON Reply: {'data': {'transaction_number': 279}}
Stock Information after Request: {'data': {'name': 'apple', 'price': 152.59, 'quantity': 949}}

##############################################################################
##Test Case 5 - Buy Error Handling (Amount to buy > num of stocks avail.): ##
##############################################################################
Input Request: POST /orders
Input JSON: {"name": "ford", "type": "buy", "quantity": 10000000}
JSON Reply: {'error': {'code': 404, 'message': 'not enough stocks available to buy'}}

#########################################################################
##Test Case 6 - Buy/Sell Error Handling (Invalid quantity of stocks): ##
#########################################################################
Note: PROTO already defines quantity needs to be an int. As such, the only invalid value will be negat
Input Request: POST /orders
Input JSON: {"name": "amazon", "type": "buy", "quantity": -1000}
JSON Reply: {'error': {'code': 404, 'message': 'invalid number of stocks'}}

Input Request: POST /orders
Input JSON: {"name": "amazon", "type": "sell", "quantity": -1000}
JSON Reply: {'error': {'code': 404, 'message': 'invalid number of stocks'}}

#####################################################################
##Test Case 7 - Buy/Sell Error Handling (Invalid request type): ##
#####################################################################
Input Request: POST /orders
Input JSON: {"name": "meta", "type": "trade", "quantity": 100}
JSON Reply: {'error': {'code': 400, 'message': 'invalid request type'}}

#############################################################
##Test Case 8 - Buy/Sell Error Handling (Stock not found): ##
#############################################################
Input Request: POST /orders
Input JSON: {"name": "imaginarycompany", "type": "buy", "quantity": 100}
JSON Reply: {'error': {'code': 404, 'message': 'stock not found'}}

C:\Users\thewi\Desktop\CS677\lab2\src\test-cases>
```

3 Native Host Example:

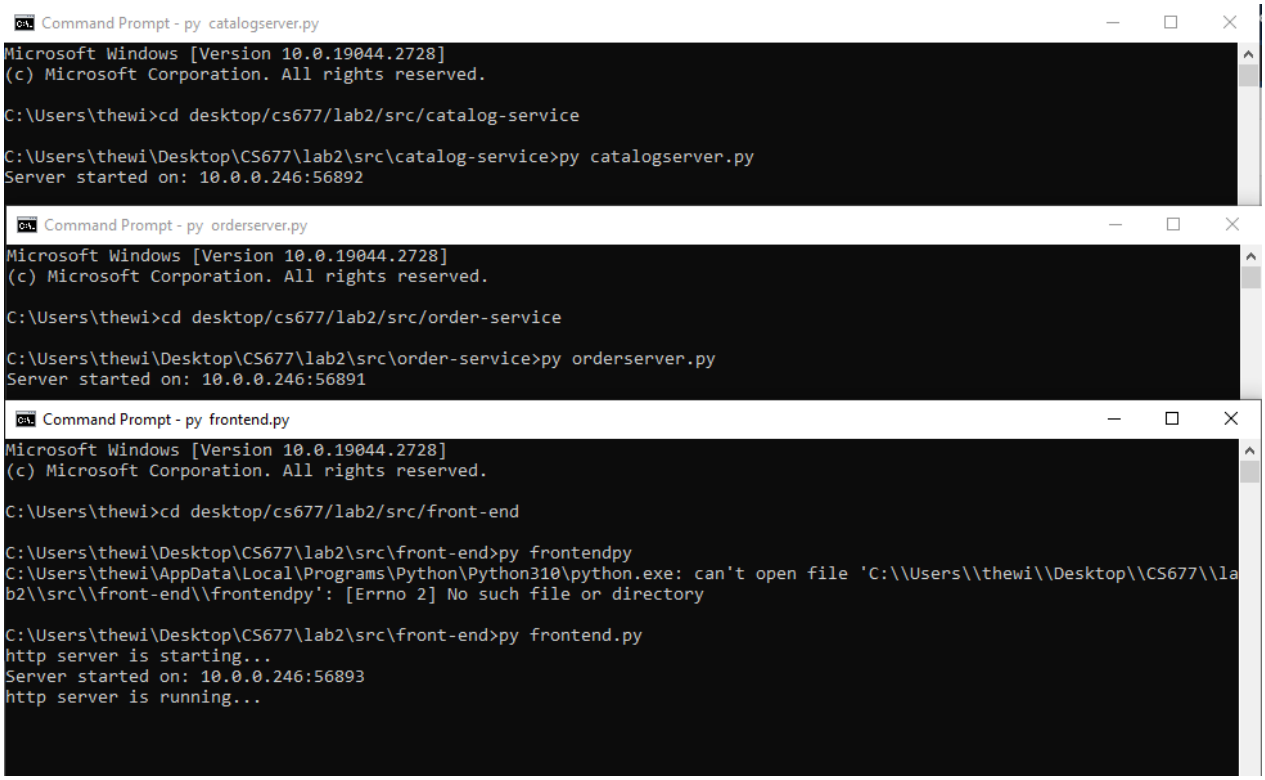

Figure 1: Output of the initialized microservices.

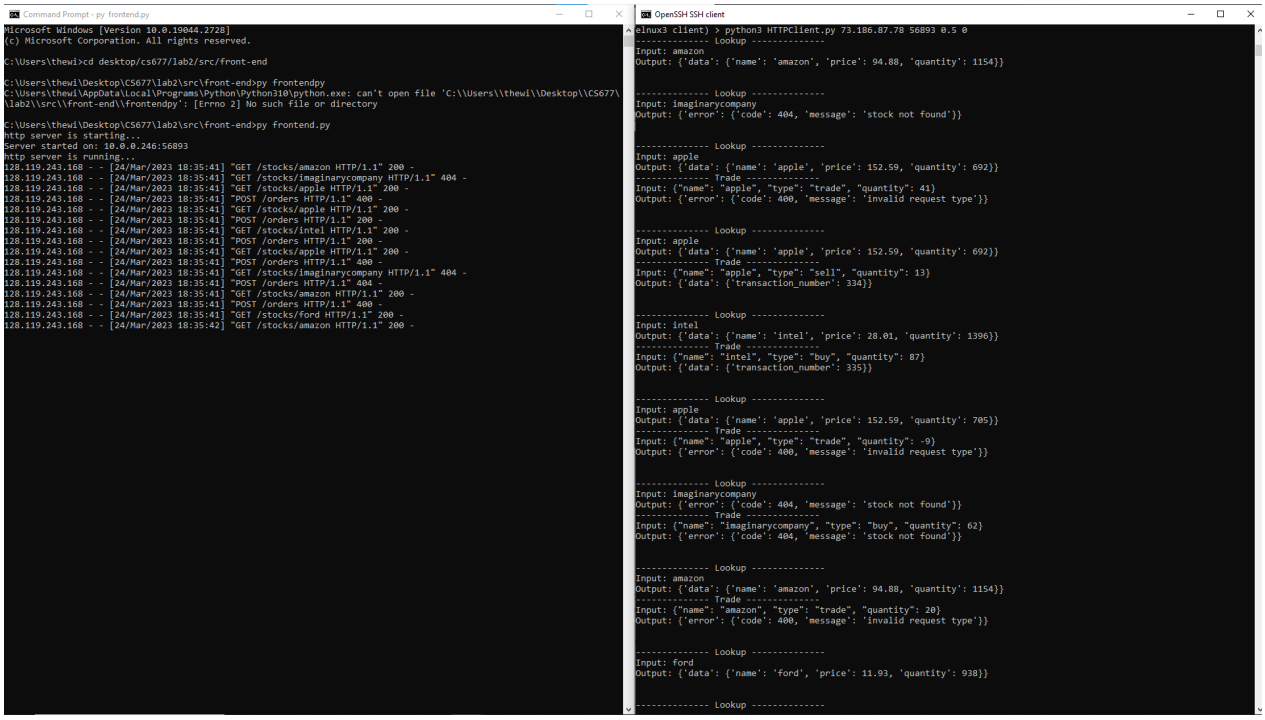

Figure 2: Example output of the stock service. Front end hosted natively on left. SSH Edlab client connecting to the server on the right.

4 Docker Host Example:

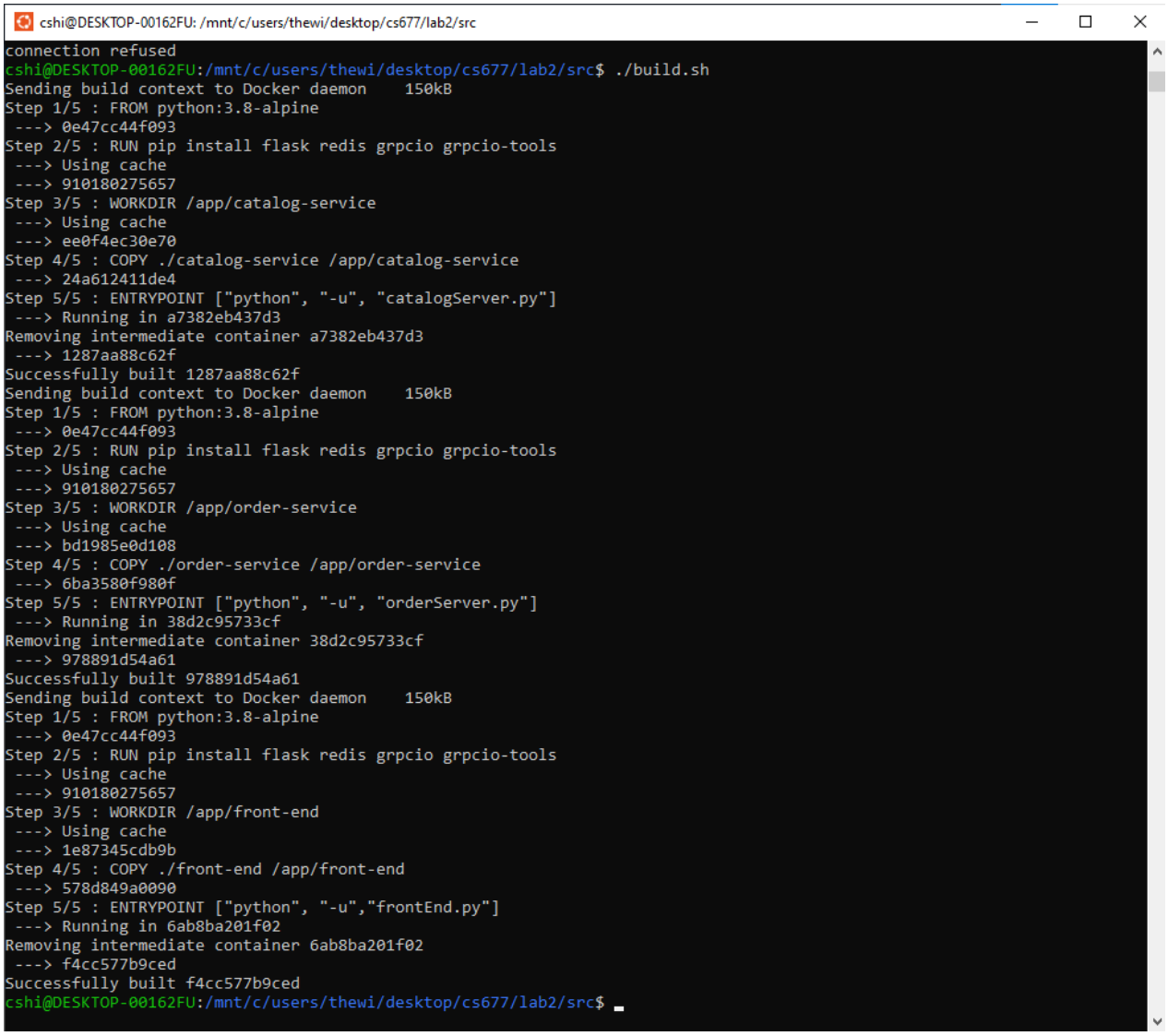

Figure 3: Output of the build.sh file. build.sh builds the images for the dockerfiles corresponding to each of the microservices.

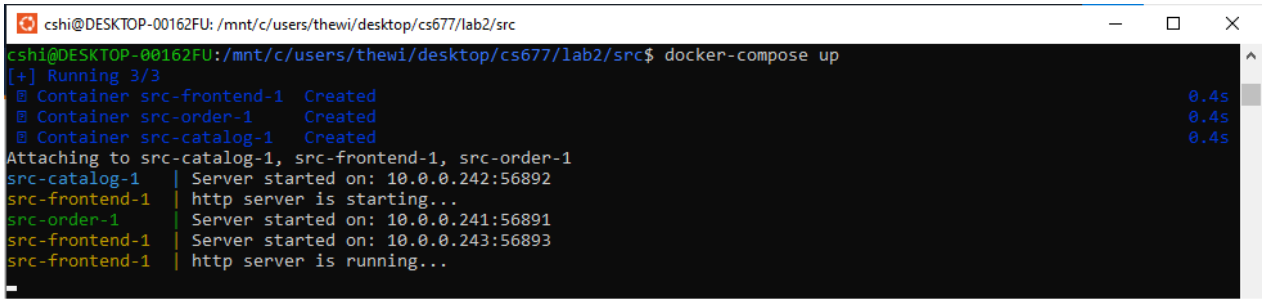

Figure 4: Output of docker-compose up.

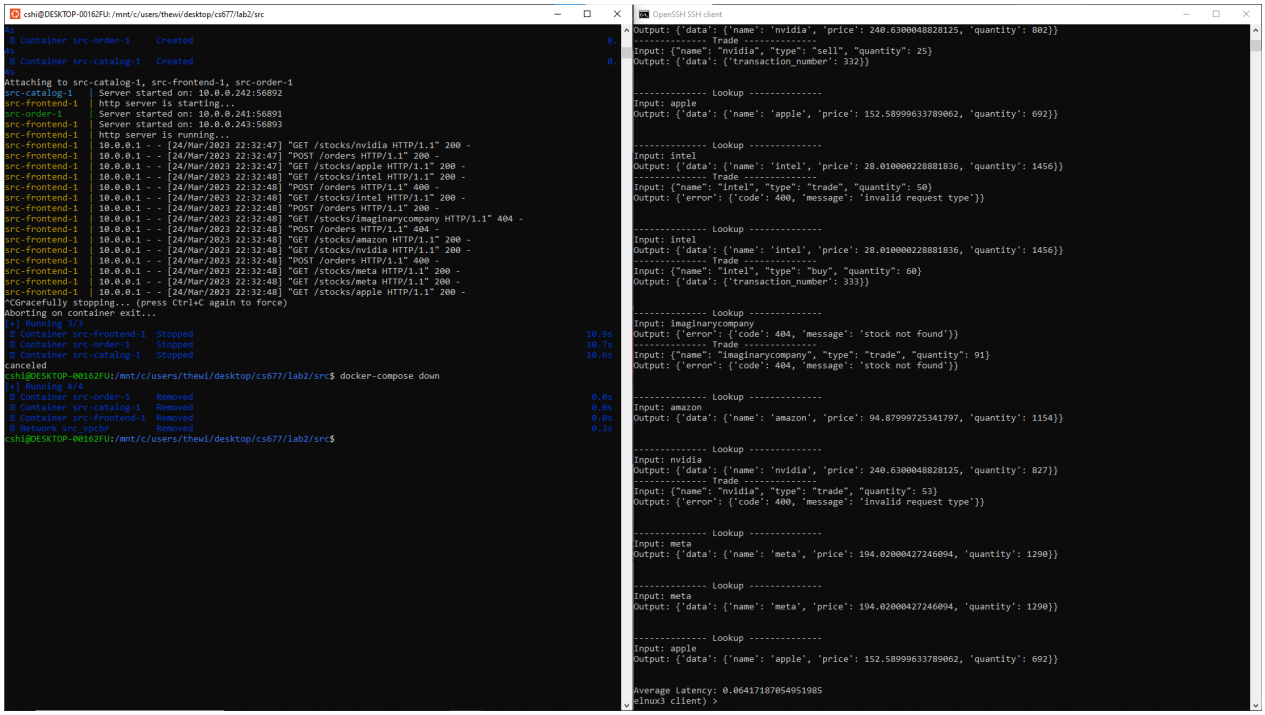

Figure 5: Example output of the stock service. Front end hosted on a Docker container on left. SSH Edlab client connecting to the server on the right.