# abtharpe Assignment #1

https://github.com/abtharpe/ser321-spring25-A-abtharpe.git

## 1. Command line tasks

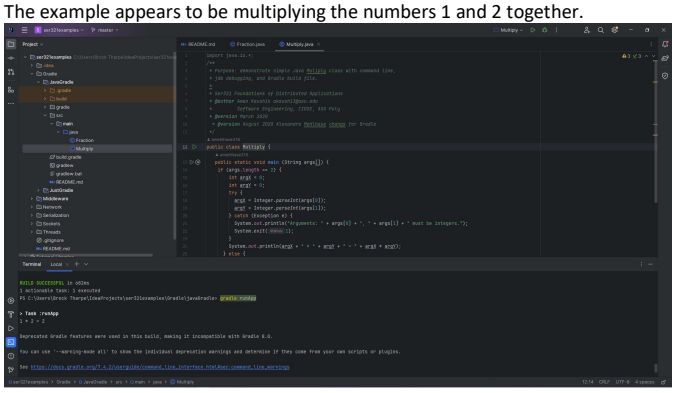
Linux System

1. mkdir cli_assignemnt
2. Cd cli_assignment
3. Touch stuff.txt
4. Cat > stuff.txt <<_EOF_
5. Cat stuff.txt
6. Cat >> stuff.txt
7. Mkdir draft
8. Mv stuff.txt draft
9. .touch .secret.txt
10. Cp -r draft final
11. Mv draft draft.remove
12. Mv draft.remove final
13. Ls -l
14. Zcat NASA_access_log_Aug95.gz
15. Gzip NASA_access_log_Aug95.gz
16. Mv NASA_access_log_Aug95 logs.txt
17. Mv logs.txt cli_assignment
18. Head -n 100 logs.txt
19. Head -n 100 logs.txt > logs_top_100.txt
20. Tail -n 100 logs.txt
21. Tail -n 100 logs.txt > logs_bottom_100.txt
22. Cat logs_top_100.txt logs_bottom_100.txt > logs_snapshot.txt
23. Echo 'abtharpe: This is a great assignment. 19 Jan, 2025.' >> logs_snapshot.txt
24. Less logs.txt
25. Tail -n +1 marks.csv | cut -d '%' -f1
26. Cut -d '%' -f4 marks.csv | sort -n
27. Awk -F '%' 'NR > 1 {sum += $3} END {print sum / (NR - 1)}' marks.csv
28. Awk -F '%' 'NR > 1 {sum += $3} END {print sum / (NR - 1)}' marks.csv > cli_assignment/done.txt
29. Mv done.txt final
30. Mv done.txt average.txt

## 2.2. Running examples

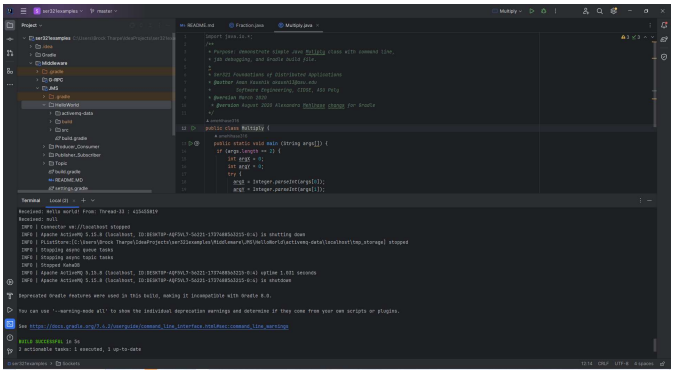
Gradle Multiply Example

- The example appears to be multiplying the numbers 1 and 2 together.
-

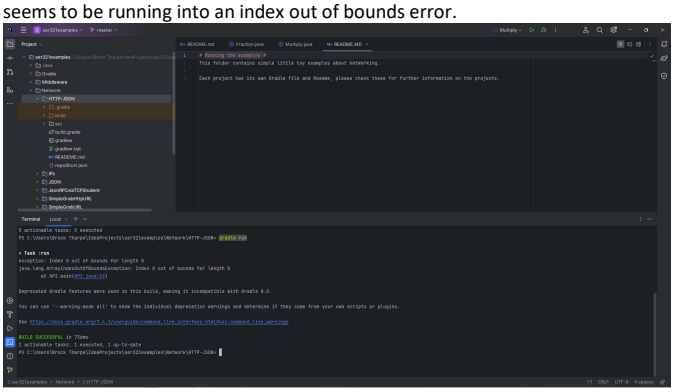


Middleware Example

- The example seems to be passing messages across threads

- 

Network Example

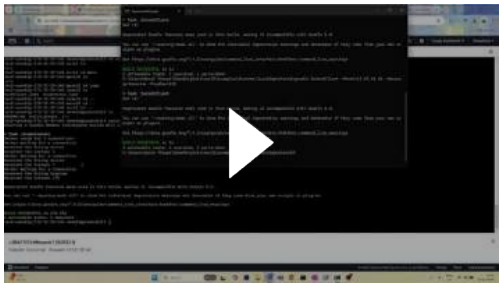
- It is hard to say here, the example is attempting to interact with an API; however, it seems to be running into an index out of bounds error.
- 

## 2.3. Understanding Gradle

- Could not get past the command 'jdb'
- Kept getting error: jdb : The term 'jdb' is not recognized as the name of a cmdlet, function, script file, or operable program. Check the spelling of the name, or if a path was included, verify that the path is correct and try again.
- I tried verifying my path and redoing it but was unsuccessful

## 2.4. Set up your second system

https://www.youtube.com/watch?v=65I1KUdzmp0&ab_channel=BrockTharpe



## 3.1. Understanding TCP network sockets

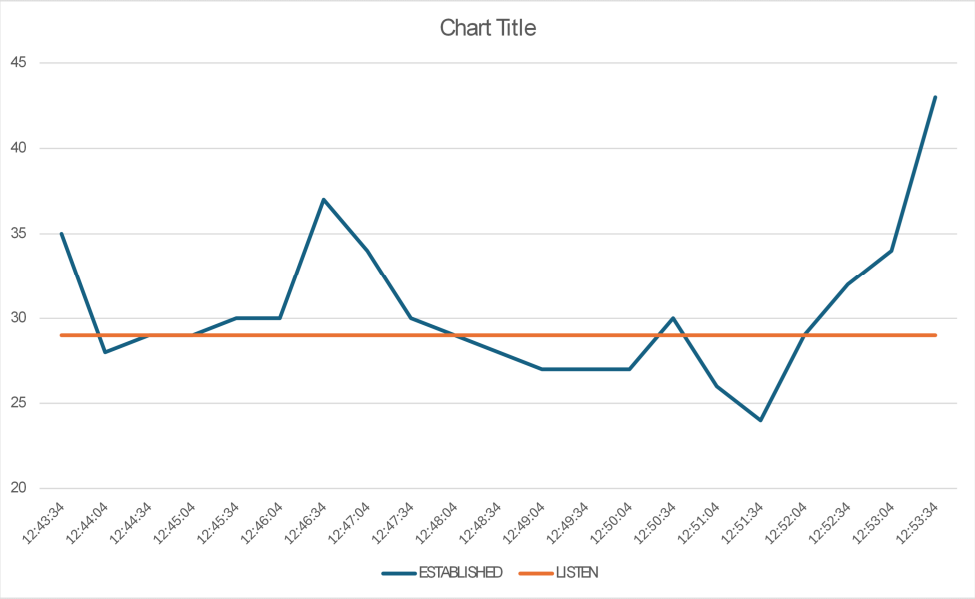
#Setting the duration in seconds.
$duration = 600

#Defining intervals (in seconds) at which the script captures the data
$interval = 30

#Defining the location for the output of the data

```
$outputFile = "C:\Users\Brock Tharpe\OneDrive - Liquor Runners\Desktop\netstat.csv"

#We add our headers before our loop
"Time,Established,Listen" | Out-File -FilePath $outputFile

#Now we can begin our loop
for ($elapsed = 0; $eplased -lt $duration; $elapsed += $interval) {
    $currentTime = Get-Date  -Format "HH:mm:ss"

    $netstatOutput = netstat -an | Select-String "ESTABLISHED|LISTEN"

    $establishedCount = ($netstatOutput | Select-String "ESTABLISHED").Count
    $listenCount = ($netstatOutput | Select-String "LISTEN").Count

    "$currentTime,$establishedCount,$listenCount" | Out-File -FilePath $outputFile -
    Append

    Start-Sleep -Seconds $interval
}
```

- Small issue where I was unable to get the loop to properly end. The csv file filled with data but I would have to manually terminate.




## 3.2. Sniffing TCP/UDP traffic

### TCP

a. Explain both the commands you used in detail. What did they actually do?
- The first command acted as a listener which tuned into the specified port we gave it, 3333. Now anything being passed over port 3333 should be heard by our listener.
- The second command was our sender, here we could send information over the specified port, 3333.

b. How many packets were send back and forth so the client/server could send/receive these two lines?
- 4 packets were sent just for the sending and receiving of the two lines.

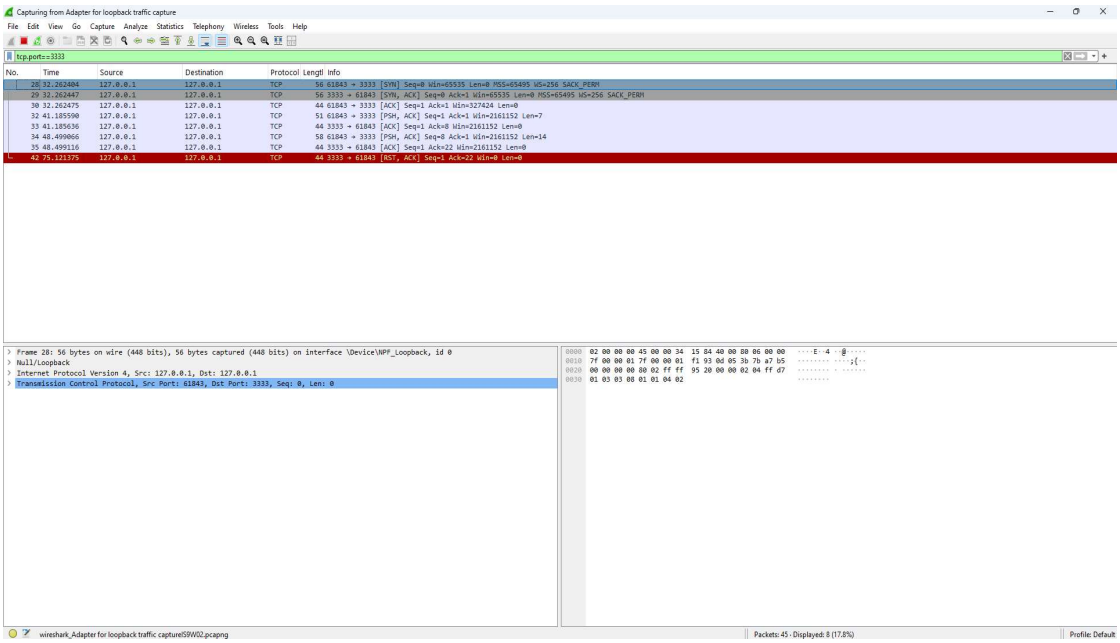
c. How many packets were needed back and forth to capture the whole "process" (starting the communication, ending the communication, sending the lines)?
- 8 packets were sent.

d. How many bytes is the data (only the data) that was sent from client to server?
- 21 bytes in total.

e. How many total bytes went over the wire (back and forth) for the whole process?
- 397 bytes.

f. How much overhead was there. Basically how many bytes was the whole process compared to the

actually data that we did send.
- 376 bytes of overhead.




## UDP

a. Explain both the commands you used in detail. What did they actually do?
- The first command acted as a listener which tuned into the specified port we gave it, 3333. Now anything being passed over port 3333 should be heard by our listener.
- The second command was our sender, here we could send information over the specified port, 3333.

b. How many packets were send back and forth so the client/server could send/receive these two lines?
- N/A

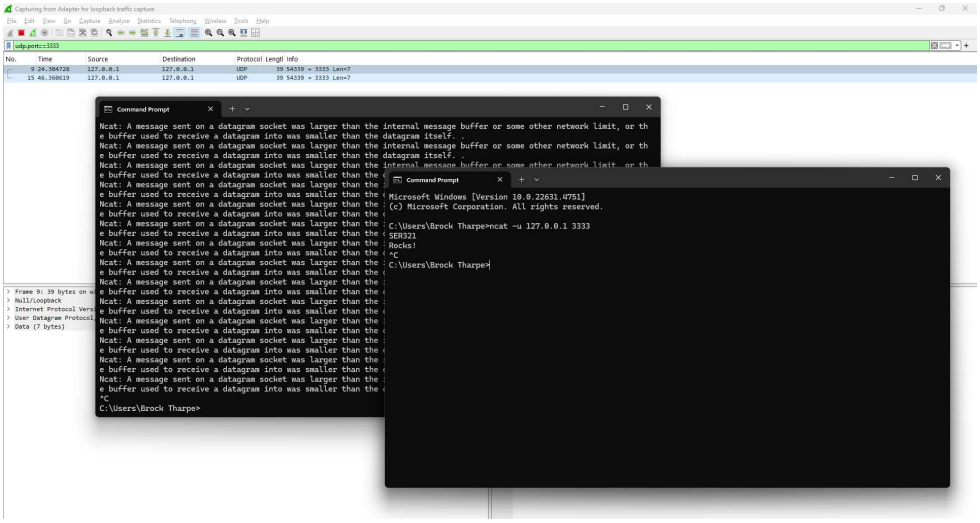
c. How many packets were needed back and forth to capture the whole "process" (starting the communication, ending the communication, sending the lines)?
- N/A

d. How many bytes is the data (only the data) that was sent from client to server?
- N/A

e. How many total bytes went over the wire (back and forth) for the whole process?
- N/A

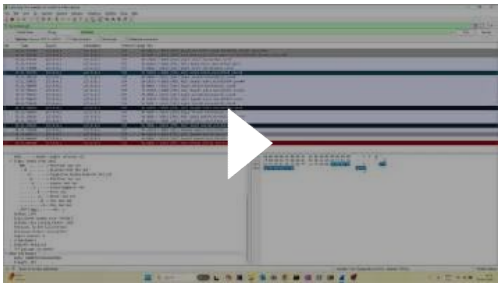
f. How much overhead was there. Basically how many bytes was the whole process compared to the actually data that we did send.
- N/A

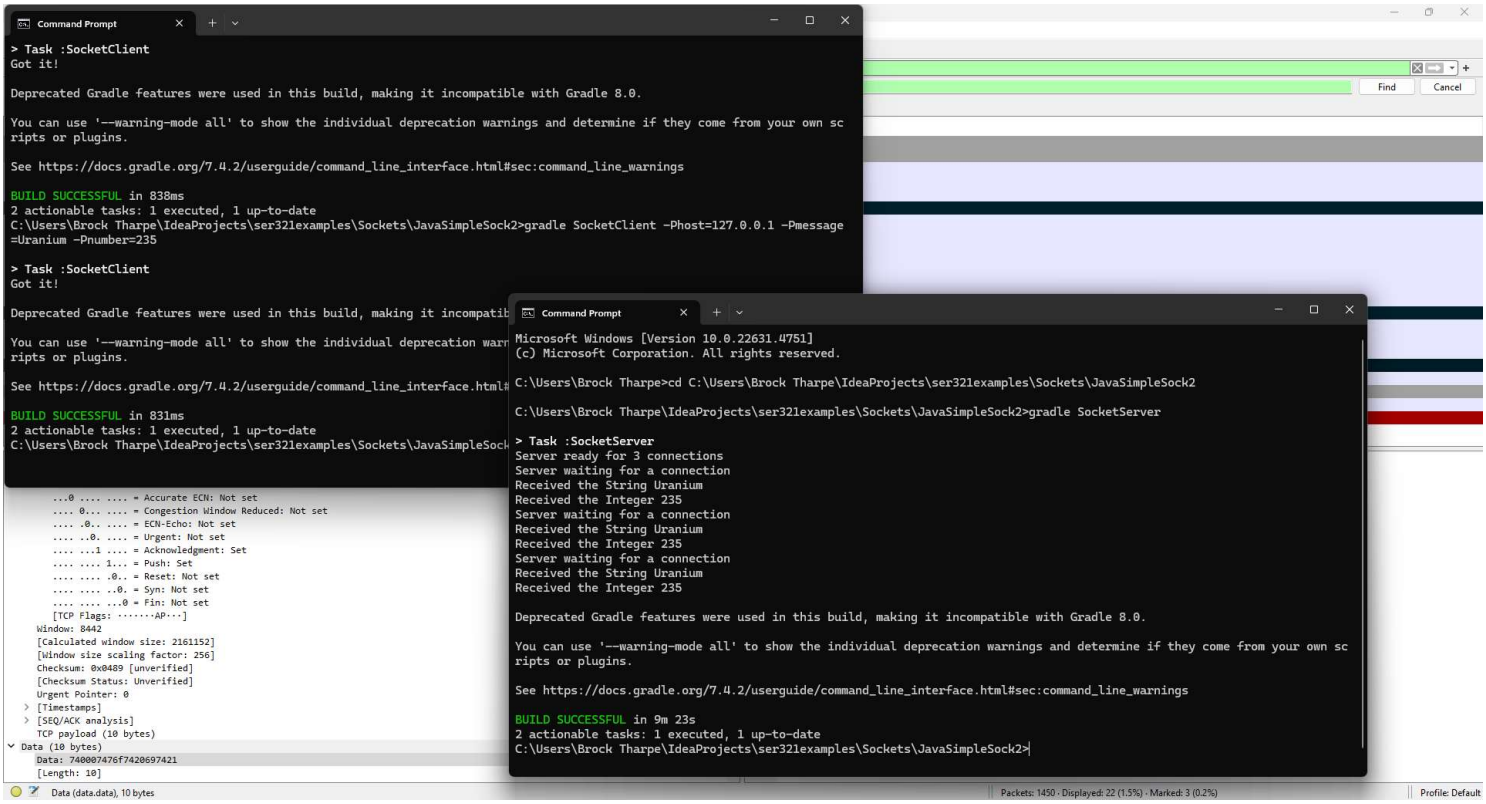
Experienced what I believe to be technical difficulties when switching over to the UDP protocol for communication. Error message kept reading back that "a message sent on a datagram socket was larger than the internal message buffer…"



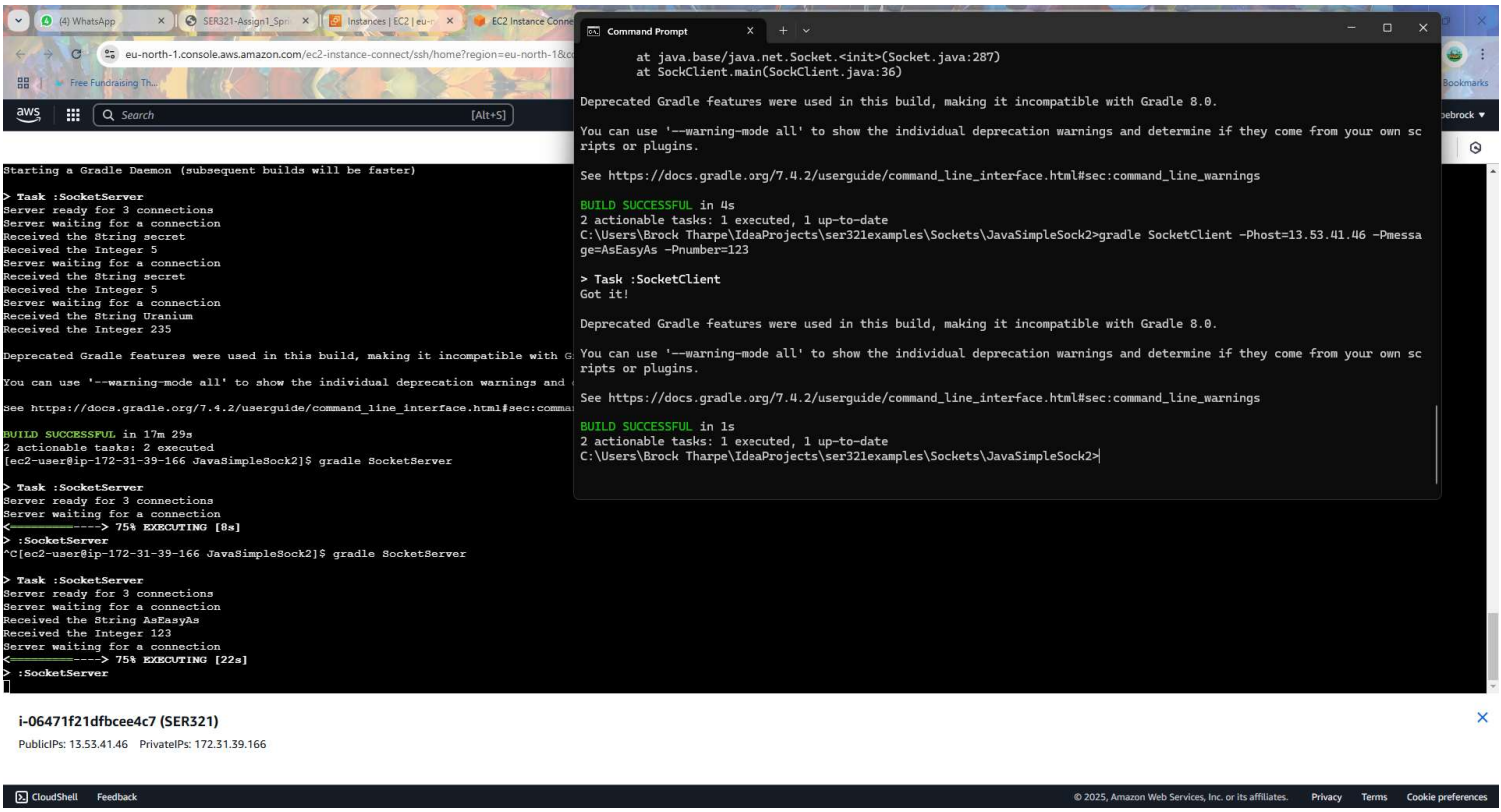
## 3.3.1. Running things locally

https://www.youtube.com/watch?v=nnY0a8-8yJk&ab_channel=BrockTharpe





## 3.3.2. Server on AWS (5 points)




## 3.3.3. Client on AWS

- No, this does not work the same way. With AWS now trying to access my local machine, it runs into port access issues with it not having access to my private IP which my router is disguising.

## 3.3.4. Client on AWS 2

- The difference here is that AWS operates with a public IP that is accessible directly by anyone. Our local machine is running on a private network behind our routers firewall which prevents direct access. If we wanted to have our server setup on our local machine then we would need to either expose our local server or incorporate port forwarding.