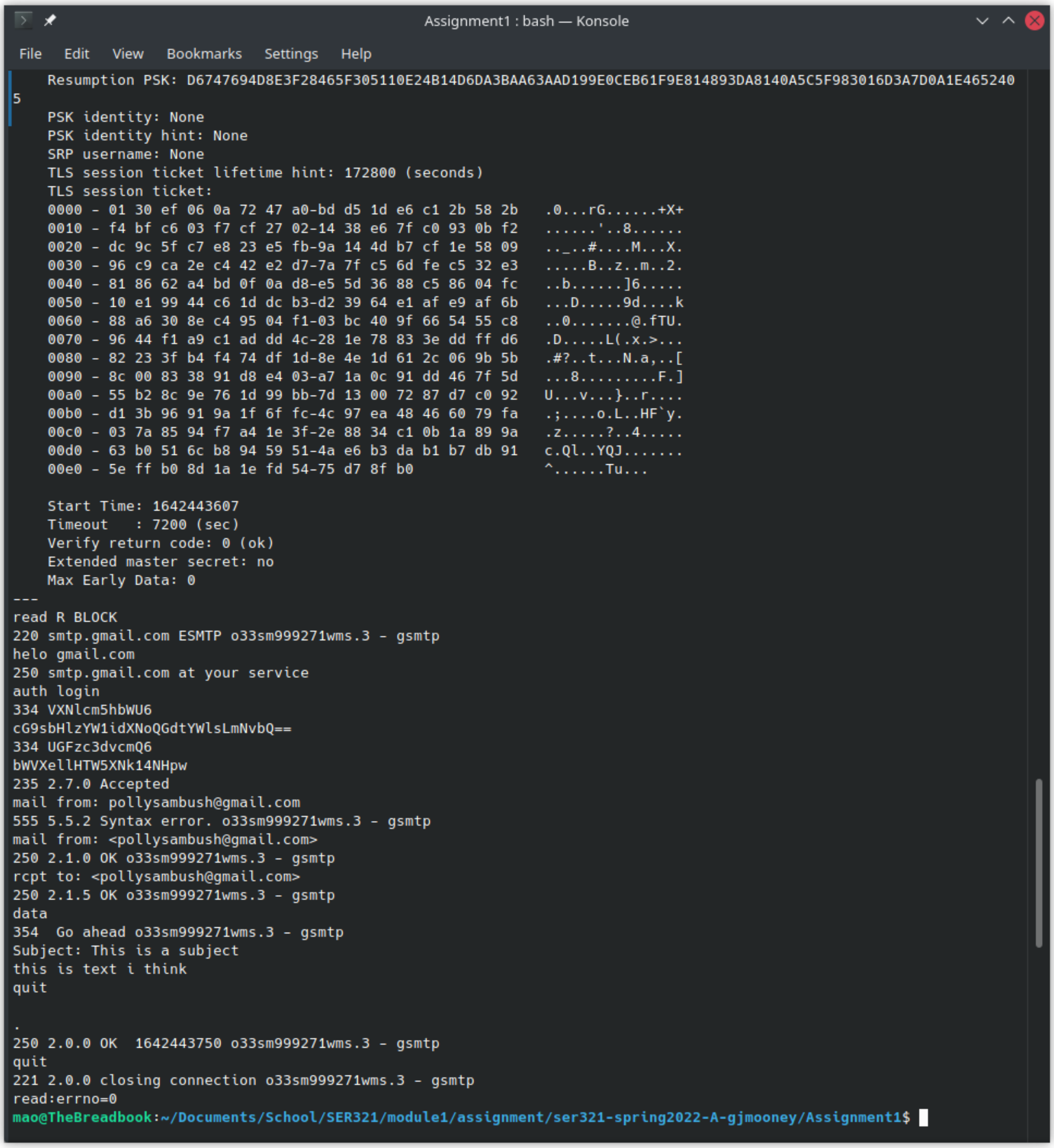
Part 1

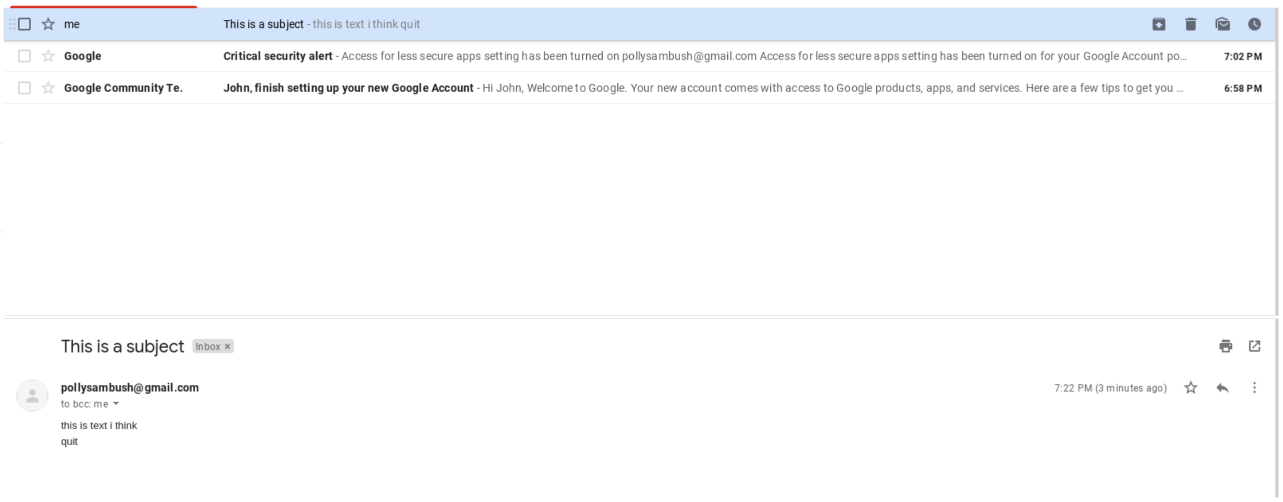
Sending emails and SMTP



```
    Resumption PSK: D6747694D8E3F28465F305110E24B14D6DA3BAA63AAD199E0CEB61F9E814893DA8140A5C5F983016D3A7D0A1E465240
5
    PSK identity: None
    PSK identity hint: None
    SRP username: None
    TLS session ticket lifetime hint: 172800 (seconds)
    TLS session ticket:
    0000 - 01 30 ef 06 0a 72 47 a0-bd d5 1d e6 c1 2b 58 2b   .0...rG......+X+
    0010 - f4 bf c6 03 f7 cf 27 02-14 38 e6 7f c0 93 0b f2   ......'..8......
    0020 - dc 9c 5f c7 e8 23 e5 fb-9a 14 4d b7 cf 1e 58 09   .._..#....M...X.
    0030 - 96 c9 ca 2e c4 42 e2 d7-7a 7f c5 6d fe c5 32 e3   .....B..z..m..2.
    0040 - 81 86 62 a4 bd 0f 0a d8-e5 5d 36 88 c5 86 04 fc   ..b......]6.....
    0050 - 10 e1 99 44 c6 1d dc b3-d2 39 64 e1 af e9 af 6b   ...D.....9d....k
    0060 - 88 a6 30 8e c4 95 04 f1-03 bc 40 9f 66 54 55 c8   ..0.......@.fTU.
    0070 - 96 44 f1 a9 c1 ad dd 4c-28 1e 78 83 3e dd ff d6   .D.....L(.x.>...
    0080 - 82 23 3f b4 f4 74 df 1d-8e 4e 1d 61 2c 06 9b 5b   .#?..t...N.a,..[
    0090 - 8c 00 83 38 91 d8 e4 03-a7 1a 0c 91 dd 46 7f 5d   ...8.........F.]
    00a0 - 55 b2 8c 9e 76 1d 99 bb-7d 13 00 72 87 d7 c0 92   U...v...}..r....
    00b0 - d1 3b 96 91 9a 1f 6f fc-4c 97 ea 48 46 60 79 fa   .;....o.L..HF`y.
    00c0 - 03 7a 85 94 f7 a4 1e 3f-2e 88 34 c1 0b 1a 89 9a   .z.....?..4.....
    00d0 - 63 b0 51 6c b8 94 59 51-4a e6 b3 da b1 b7 db 91   c.Ql..YQJ.......
    00e0 - 5e ff b0 8d 1a 1e fd 54-75 d7 8f b0               ^......Tu...

    Start Time: 1642443607
    Timeout   : 7200 (sec)
    Verify return code: 0 (ok)
    Extended master secret: no
    Max Early Data: 0
---
read R BLOCK
220 smtp.gmail.com ESMTP o33sm999271wms.3 - gsmtp
helo gmail.com
250 smtp.gmail.com at your service
auth login
334 VXNlcm5hbWU6
cG9sbHlzYW1idXNoQGdtYWlsLmNvbQ==
334 UGFzc3dvcmQ6
bWVXellHTW5XNk14NHpw
235 2.7.0 Accepted
mail from: pollysambush@gmail.com
555 5.5.2 Syntax error. o33sm999271wms.3 - gsmtp
mail from: <pollysambush@gmail.com>
250 2.1.0 OK o33sm999271wms.3 - gsmtp
rcpt to: <pollysambush@gmail.com>
250 2.1.5 OK o33sm999271wms.3 - gsmtp
data
354  Go ahead o33sm999271wms.3 - gsmtp
Subject: This is a subject
this is text i think
quit


.
250 2.0.0 OK  1642443750 o33sm999271wms.3 - gsmtp
quit
221 2.0.0 closing connection o33sm999271wms.3 - gsmtp
read:errno=0
mao@TheBreadbook:~/Documents/School/SER321/module1/assignment/ser321-spring2022-A-gjmooney/Assignment1$
```



me — **This is a subject** - this is text i think quit

Google — **Critical security alert** - Access for less secure apps setting has been turned on pollysambush@gmail.com Access for less secure apps setting has been turned on for your Google Account po… **7:02 PM**

Google Community Te. — **John, finish setting up your new Google Account** - Hi John, Welcome to Google. Your new account comes with access to Google products, apps, and services. Here are a few tips to get you … **6:58 PM**

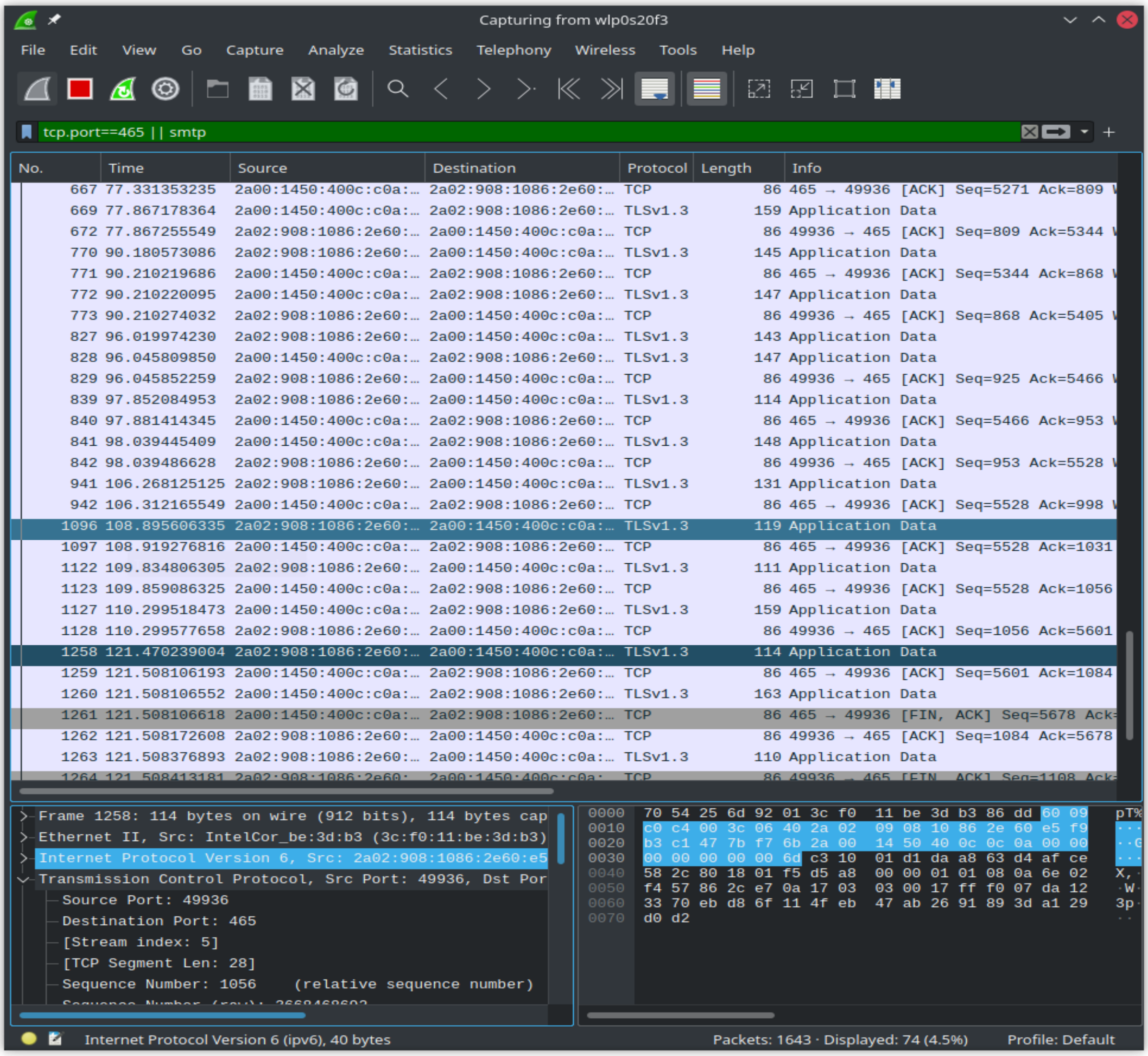
This is a subject Inbox ×

pollysambush@gmail.com
to bcc: me
7:22 PM (3 minutes ago)

this is text i think
quit

1. I used the filter “tcp.port==465” because that was the port specified in the opensll command.
2. The standard port for SMTP is port 25, however this port is generally blocked and modern services use port 587 as the default. Port 465 was originally used for SMTP over SSL, however that has been deprecated. I assume we use it in the example because it is still an accepted SMTP port but shouldn’t have much other traffic on it.
3. The initial command establishes a connection to a remote server using my laptop as a SSL/TSL client
   - helo gmail.com is sent from the client to identify itself and initiate a dialogue via SMTP
   - auth login authenticates the client to the server using the username and password
   - mail from: designates the email address of the sender, and informs the server that a new mail conversation has started
   - rcpt to: designates the email address of the recipient
   - data begins the actual contents of the message
   - subject: denotes the subject of the email message
   - The single . tells the server that all the content of the message has been sent
4. My Wireshark capture shows 15 packets used for establishing the connection.
5. My machine is using port 49936
6. The first FIN flag is sent from the server. The client sends the quit command to the server, the server receives the command and sends FIN and ACK flags, followed by the client sending back the final FIN and ACK flags.

7. Wireshark capture



```
tcp.port==465 || smtp
```

```
No.   Time           Source                Destination           Protocol Length Info
  667 77.331353235   2a00:1450:400c:c0a:…  2a02:908:1086:2e60:…  TCP       86 465 → 49936 [ACK] Seq=5271 Ack=809
  669 77.867178364   2a00:1450:400c:c0a:…  2a02:908:1086:2e60:…  TLSv1.3  159 Application Data
  672 77.867255549   2a02:908:1086:2e60:…  2a00:1450:400c:c0a:…  TCP       86 49936 → 465 [ACK] Seq=809 Ack=5344
  770 90.180573086   2a02:908:1086:2e60:…  2a00:1450:400c:c0a:…  TLSv1.3  145 Application Data
  771 90.210219686   2a00:1450:400c:c0a:…  2a02:908:1086:2e60:…  TCP       86 465 → 49936 [ACK] Seq=5344 Ack=868
  772 90.210220095   2a00:1450:400c:c0a:…  2a02:908:1086:2e60:…  TLSv1.3  147 Application Data
  773 90.210274032   2a02:908:1086:2e60:…  2a00:1450:400c:c0a:…  TCP       86 49936 → 465 [ACK] Seq=868 Ack=5405
  827 96.019974230   2a02:908:1086:2e60:…  2a00:1450:400c:c0a:…  TLSv1.3  143 Application Data
  828 96.045809850   2a00:1450:400c:c0a:…  2a02:908:1086:2e60:…  TLSv1.3  147 Application Data
  829 96.045852259   2a02:908:1086:2e60:…  2a00:1450:400c:c0a:…  TCP       86 49936 → 465 [ACK] Seq=925 Ack=5466
  839 97.852084953   2a02:908:1086:2e60:…  2a00:1450:400c:c0a:…  TLSv1.3  114 Application Data
  840 97.881414345   2a00:1450:400c:c0a:…  2a02:908:1086:2e60:…  TCP       86 465 → 49936 [ACK] Seq=5466 Ack=953
  841 98.039445409   2a00:1450:400c:c0a:…  2a02:908:1086:2e60:…  TLSv1.3  148 Application Data
  842 98.039486628   2a02:908:1086:2e60:…  2a00:1450:400c:c0a:…  TCP       86 49936 → 465 [ACK] Seq=953 Ack=5528
  941 106.268125125  2a02:908:1086:2e60:…  2a00:1450:400c:c0a:…  TLSv1.3  131 Application Data
  942 106.312165549  2a00:1450:400c:c0a:…  2a02:908:1086:2e60:…  TCP       86 465 → 49936 [ACK] Seq=5528 Ack=998
 1096 108.895606335  2a02:908:1086:2e60:…  2a00:1450:400c:c0a:…  TLSv1.3  119 Application Data
 1097 108.919276816  2a00:1450:400c:c0a:…  2a02:908:1086:2e60:…  TCP       86 465 → 49936 [ACK] Seq=5528 Ack=1031
 1122 109.834806305  2a02:908:1086:2e60:…  2a00:1450:400c:c0a:…  TLSv1.3  111 Application Data
 1123 109.859086325  2a00:1450:400c:c0a:…  2a02:908:1086:2e60:…  TCP       86 465 → 49936 [ACK] Seq=5528 Ack=1056
 1127 110.299518473  2a00:1450:400c:c0a:…  2a02:908:1086:2e60:…  TLSv1.3  159 Application Data
 1128 110.299577658  2a02:908:1086:2e60:…  2a00:1450:400c:c0a:…  TCP       86 49936 → 465 [ACK] Seq=1056 Ack=5601
 1258 121.470239004  2a02:908:1086:2e60:…  2a00:1450:400c:c0a:…  TLSv1.3  114 Application Data
 1259 121.508106193  2a00:1450:400c:c0a:…  2a02:908:1086:2e60:…  TCP       86 465 → 49936 [ACK] Seq=5601 Ack=1084
 1260 121.508106552  2a00:1450:400c:c0a:…  2a02:908:1086:2e60:…  TLSv1.3  163 Application Data
 1261 121.508106618  2a00:1450:400c:c0a:…  2a02:908:1086:2e60:…  TCP       86 465 → 49936 [FIN, ACK] Seq=5678 Ack
 1262 121.508172608  2a02:908:1086:2e60:…  2a00:1450:400c:c0a:…  TCP       86 49936 → 465 [ACK] Seq=1084 Ack=5678
 1263 121.508376893  2a02:908:1086:2e60:…  2a00:1450:400c:c0a:…  TLSv1.3  110 Application Data
 1264 121.508413181  2a02:908:1086:2e60:…  2a00:1450:400c:c0a:…  TCP       86 49936 → 465 [FIN, ACK] Seq=1108 Ack
```

```
> Frame 1258: 114 bytes on wire (912 bits), 114 bytes cap
> Ethernet II, Src: IntelCor_be:3d:b3 (3c:f0:11:be:3d:b3)
> Internet Protocol Version 6, Src: 2a02:908:1086:2e60:e5
v Transmission Control Protocol, Src Port: 49936, Dst Por
    Source Port: 49936
    Destination Port: 465
    [Stream index: 5]
    [TCP Segment Len: 28]
    Sequence Number: 1056    (relative sequence number)
    Sequence Number (raw): [illegible]
```

```
0000  70 54 25 6d 92 01 3c f0  11 be 3d b3 86 dd 60 09   pT%
0010  c0 c4 00 3c 06 40 2a 02  09 08 10 86 2e 60 e5 f9   ···
0020  b3 c1 47 7b f7 6b 2a 00  14 50 40 0c 0c 0a 00 00   ··G
0030  00 00 00 00 00 6d c3 10  01 d1 da a8 63 d4 af ce   ···
0040  58 2c 80 18 01 f5 d5 a8  00 00 01 01 08 0a 6e 02   X,·
0050  f4 57 86 2c e7 0a 17 03  03 00 17 ff f0 07 da 12   ·W·
0060  33 70 eb d8 6f 11 4f eb  47 ab 26 91 89 3d a1 29   3p·
0070  d0 d2                                              ··
```

Internet Protocol Version 6 (ipv6), 40 bytes   Packets: 1643 · Displayed: 74 (4.5%)   Profile: Default

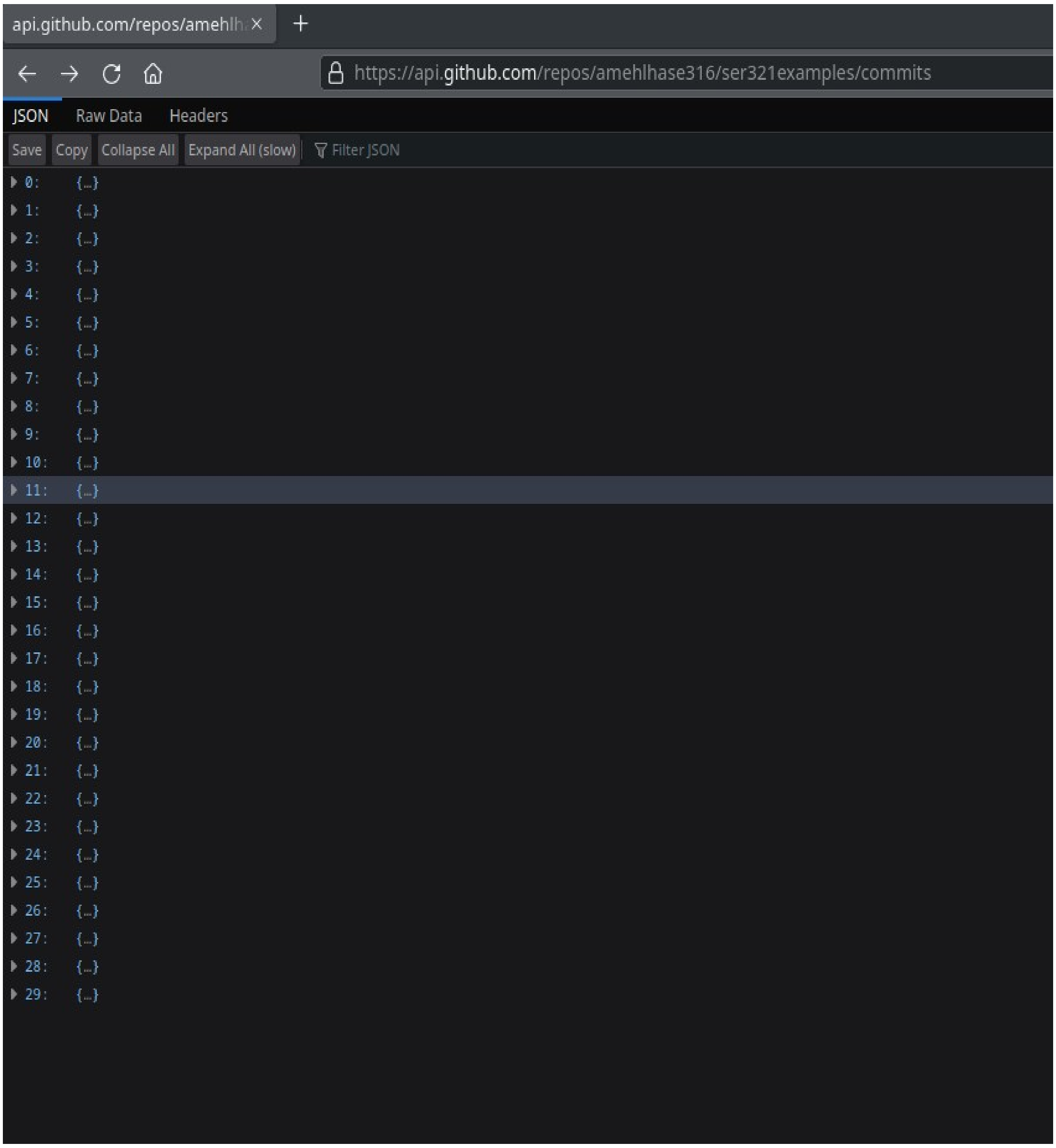
Part 2

Understanding HTTP

Call:

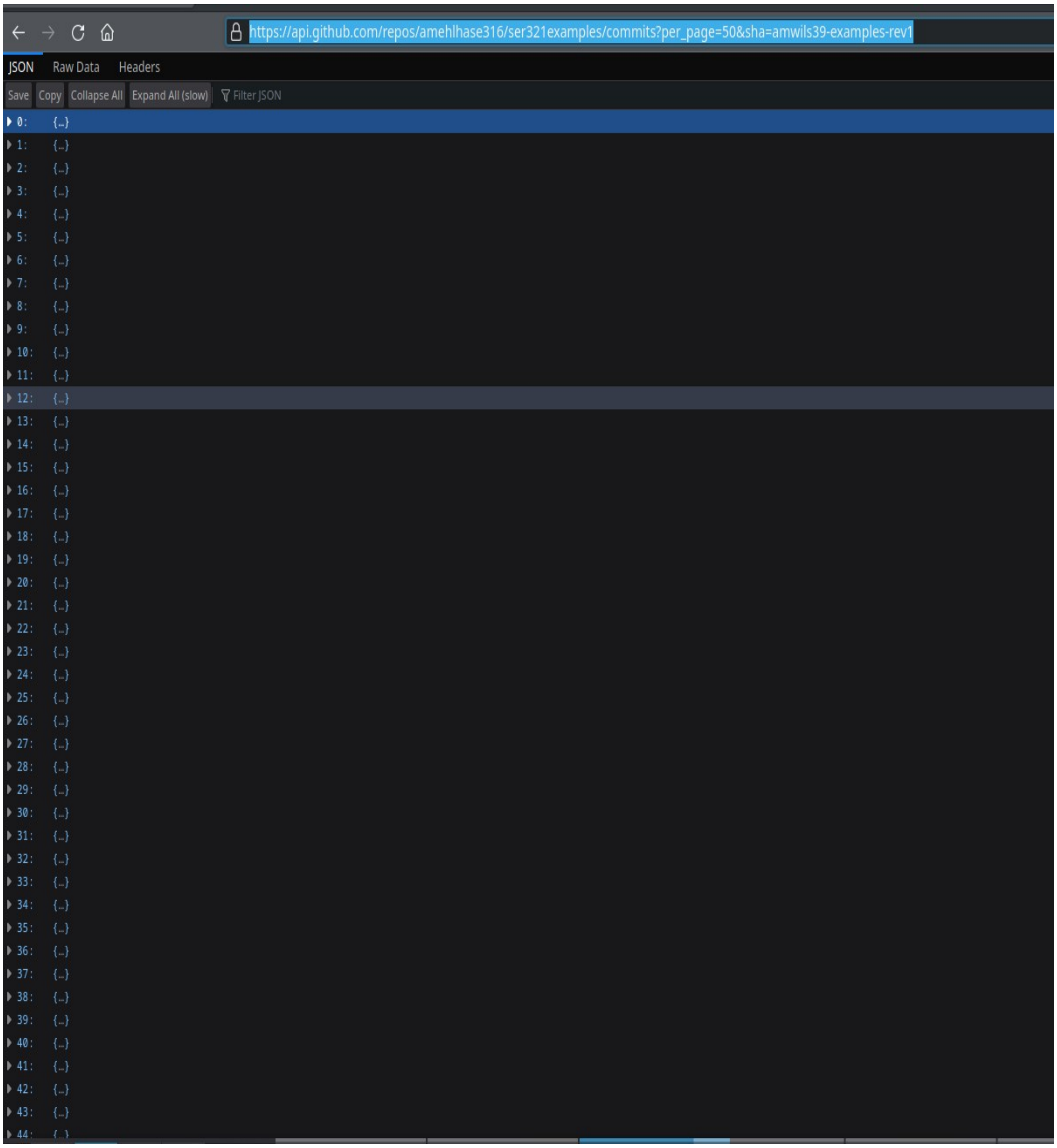
https://api.github.com/repos/amehlhase316/ser321examples/commits

Note: By default, that shows the commits to the default branch (Doc)


api.github.com/repos/amehlh
https://api.github.com/repos/amehlhase316/ser321examples/commits
JSON
Raw Data
Headers
Save
Copy
Collapse All
Expand All (slow)
Filter JSON

```
0:  {…}
1:  {…}
2:  {…}
3:  {…}
4:  {…}
5:  {…}
6:  {…}
7:  {…}
8:  {…}
9:  {…}
10: {…}
11: {…}
12: {…}
13: {…}
14: {…}
15: {…}
16: {…}
17: {…}
18: {…}
19: {…}
20: {…}
21: {…}
22: {…}
23: {…}
24: {…}
25: {…}
26: {…}
27: {…}
28: {…}
29: {…}
```

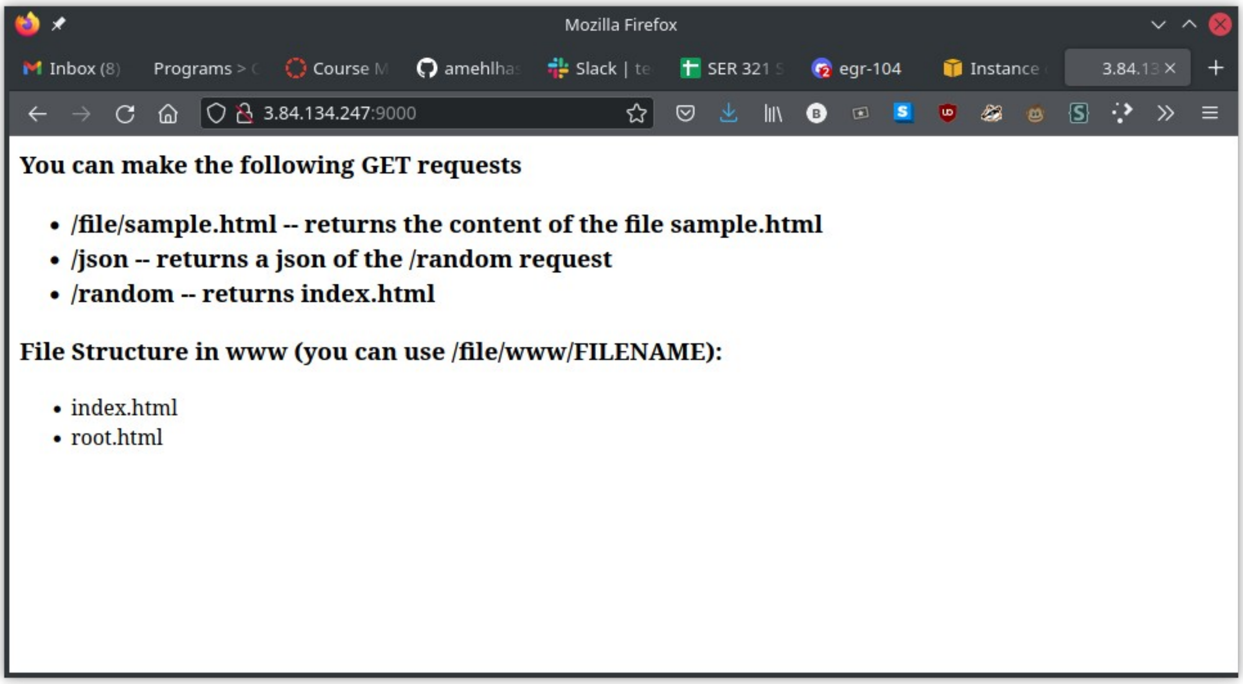
Call:
https://api.github.com/repos/amehlhase316/ser321examples/commits?per_page=50&sha=amwils39-examples-rev1


https://api.github.com/repos/amehlhase316/ser321examples/commits?per_page=50&sha=amwils39-examples-rev1

```
0:  {…}
1:  {…}
2:  {…}
3:  {…}
4:  {…}
5:  {…}
6:  {…}
7:  {…}
8:  {…}
9:  {…}
10: {…}
11: {…}
12: {…}
13: {…}
14: {…}
15: {…}
16: {…}
17: {…}
18: {…}
19: {…}
20: {…}
21: {…}
22: {…}
23: {…}
24: {…}
25: {…}
26: {…}
27: {…}
28: {…}
29: {…}
30: {…}
31: {…}
32: {…}
33: {…}
34: {…}
35: {…}
36: {…}
37: {…}
38: {…}
39: {…}
40: {…}
41: {…}
42: {…}
43: {…}
44:
```

1. The first API call I used was GET repos/{owner}/{repo}/commits which lists the commits made to a branch, by default the call lists commits made to the default branch of the repo. The second call was a modification of the first, it added the per_page parameter, which increases the number of commits shown per page, the second parameter used was the sha parameter which specifies which branch to list the commits of.
2. Stateless communicates have no memory of previous requests, each request is handled independently. Stateful communications retain knowledge of the connection and the client always expects a response from the server, resending the request if none is received.

3.2 Running a simple Java WebServer

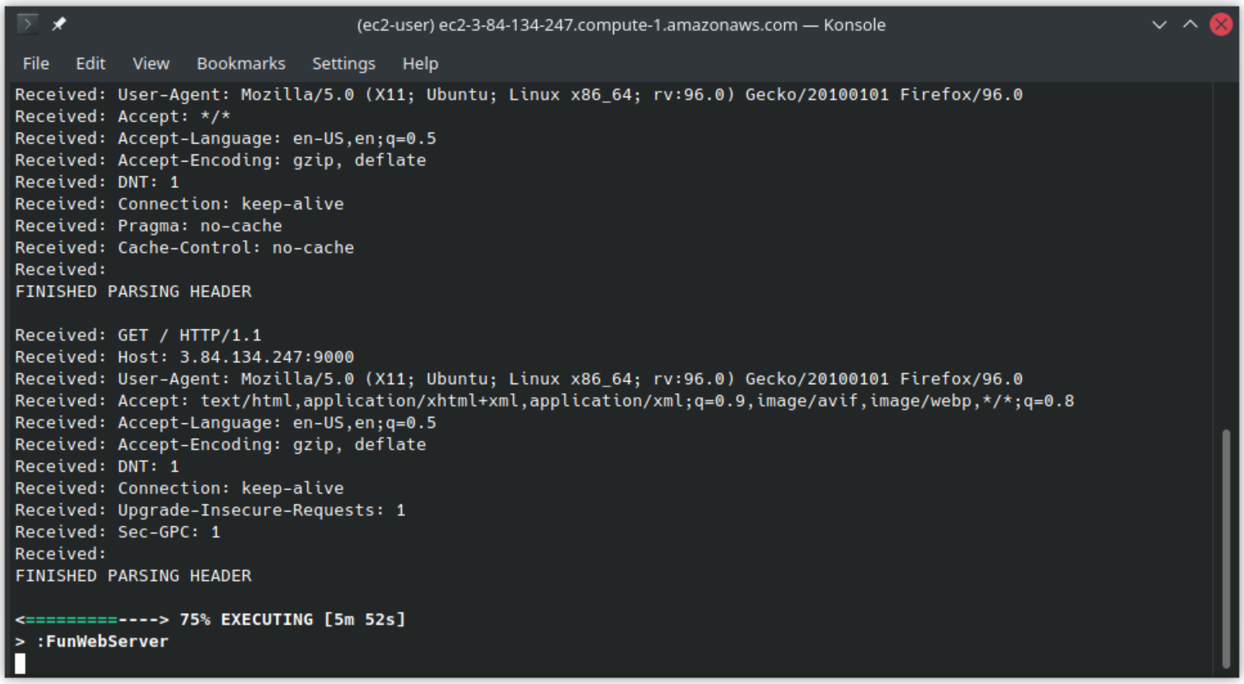


You can make the following GET requests

- **/file/sample.html -- returns the content of the file sample.html**
- **/json -- returns a json of the /random request**
- **/random -- returns index.html**

File Structure in www (you can use /file/www/FILENAME):

- index.html
- root.html



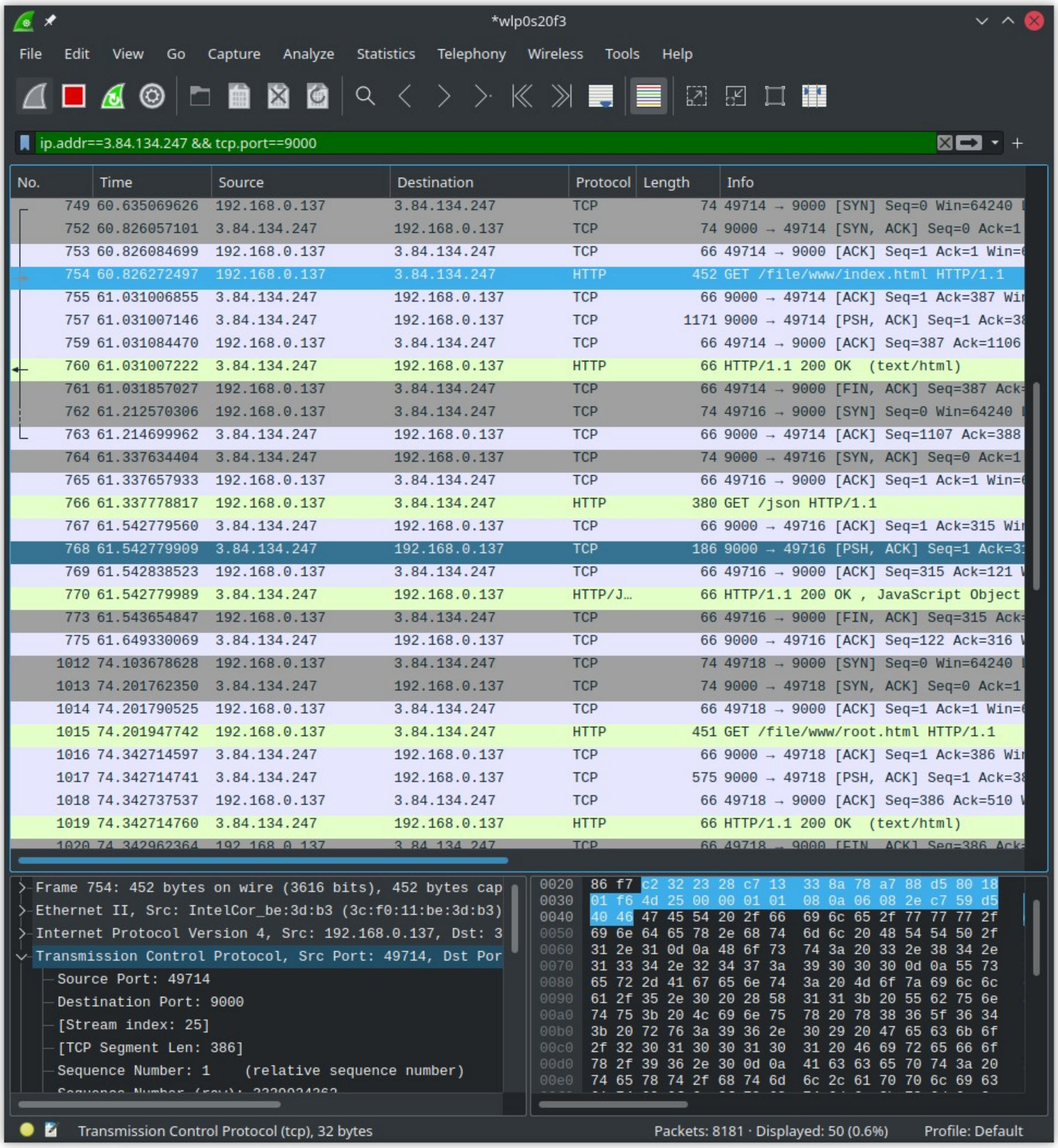
(ec2-user) ec2-3-84-134-247.compute-1.amazonaws.com — Konsole
File
Edit
View
Bookmarks
Settings
Help

```
Received: User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:96.0) Gecko/20100101 Firefox/96.0
Received: Accept: */*
Received: Accept-Language: en-US,en;q=0.5
Received: Accept-Encoding: gzip, deflate
Received: DNT: 1
Received: Connection: keep-alive
Received: Pragma: no-cache
Received: Cache-Control: no-cache
Received:
FINISHED PARSING HEADER

Received: GET / HTTP/1.1
Received: Host: 3.84.134.247:9000
Received: User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:96.0) Gecko/20100101 Firefox/96.0
Received: Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Received: Accept-Language: en-US,en;q=0.5
Received: Accept-Encoding: gzip, deflate
Received: DNT: 1
Received: Connection: keep-alive
Received: Upgrade-Insecure-Requests: 1
Received: Sec-GPC: 1
Received:
FINISHED PARSING HEADER

<=========----> 75% EXECUTING [5m 52s]
> :FunWebServer
```

3.3 Analyze what happens



```
ip.addr==3.84.134.247 && tcp.port==9000

No.   Time         Source         Destination    Protocol Length Info
  749 60.635069626 192.168.0.137  3.84.134.247   TCP        74 49714 → 9000 [SYN] Seq=0 Win=64240
  752 60.826057101 3.84.134.247   192.168.0.137  TCP        74 9000 → 49714 [SYN, ACK] Seq=0 Ack=1
  753 60.826084699 192.168.0.137  3.84.134.247   TCP        66 49714 → 9000 [ACK] Seq=1 Ack=1 Win=
  754 60.826272497 192.168.0.137  3.84.134.247   HTTP      452 GET /file/www/index.html HTTP/1.1
  755 61.031006855 3.84.134.247   192.168.0.137  TCP        66 9000 → 49714 [ACK] Seq=1 Ack=387 Wi
  757 61.031007146 3.84.134.247   192.168.0.137  TCP      1171 9000 → 49714 [PSH, ACK] Seq=1 Ack=3
  759 61.031084470 192.168.0.137  3.84.134.247   TCP        66 49714 → 9000 [ACK] Seq=387 Ack=1106
  760 61.031007222 3.84.134.247   192.168.0.137  HTTP       66 HTTP/1.1 200 OK  (text/html)
  761 61.031857027 192.168.0.137  3.84.134.247   TCP        66 49714 → 9000 [FIN, ACK] Seq=387 Ack=
  762 61.212570306 192.168.0.137  3.84.134.247   TCP        74 49716 → 9000 [SYN] Seq=0 Win=64240
  763 61.214699962 3.84.134.247   192.168.0.137  TCP        66 9000 → 49714 [ACK] Seq=1107 Ack=388
  764 61.337634404 3.84.134.247   192.168.0.137  TCP        74 9000 → 49716 [SYN, ACK] Seq=0 Ack=1
  765 61.337657933 192.168.0.137  3.84.134.247   TCP        66 49716 → 9000 [ACK] Seq=1 Ack=1 Win=
  766 61.337778817 192.168.0.137  3.84.134.247   HTTP      380 GET /json HTTP/1.1
  767 61.542779560 3.84.134.247   192.168.0.137  TCP        66 9000 → 49716 [ACK] Seq=1 Ack=315 Wi
  768 61.542779909 3.84.134.247   192.168.0.137  TCP       186 9000 → 49716 [PSH, ACK] Seq=1 Ack=3
  769 61.542838523 192.168.0.137  3.84.134.247   TCP        66 49716 → 9000 [ACK] Seq=315 Ack=121
  770 61.542779989 3.84.134.247   192.168.0.137  HTTP/J…    66 HTTP/1.1 200 OK , JavaScript Object
  773 61.543654847 192.168.0.137  3.84.134.247   TCP        66 49716 → 9000 [FIN, ACK] Seq=315 Ack=
  775 61.649330069 3.84.134.247   192.168.0.137  TCP        66 9000 → 49716 [ACK] Seq=122 Ack=316
 1012 74.103678628 192.168.0.137  3.84.134.247   TCP        74 49718 → 9000 [SYN] Seq=0 Win=64240
 1013 74.201762350 3.84.134.247   192.168.0.137  TCP        74 9000 → 49718 [SYN, ACK] Seq=0 Ack=1
 1014 74.201790525 192.168.0.137  3.84.134.247   TCP        66 49718 → 9000 [ACK] Seq=1 Ack=1 Win=
 1015 74.201947742 192.168.0.137  3.84.134.247   HTTP      451 GET /file/www/root.html HTTP/1.1
 1016 74.342714597 3.84.134.247   192.168.0.137  TCP        66 9000 → 49718 [ACK] Seq=1 Ack=386 Wi
 1017 74.342714741 3.84.134.247   192.168.0.137  TCP       575 9000 → 49718 [PSH, ACK] Seq=1 Ack=3
 1018 74.342737537 192.168.0.137  3.84.134.247   TCP        66 49718 → 9000 [ACK] Seq=386 Ack=510
 1019 74.342714760 3.84.134.247   192.168.0.137  HTTP       66 HTTP/1.1 200 OK  (text/html)
 1020 74.342962364 192.168.0.137  3.84.134.247   TCP        66 49718 → 9000 [FIN, ACK] Seq=386 Ack=
```

```
> Frame 754: 452 bytes on wire (3616 bits), 452 bytes cap
> Ethernet II, Src: IntelCor_be:3d:b3 (3c:f0:11:be:3d:b3)
> Internet Protocol Version 4, Src: 192.168.0.137, Dst: 3
v Transmission Control Protocol, Src Port: 49714, Dst Por
    Source Port: 49714
    Destination Port: 9000
    [Stream index: 25]
    [TCP Segment Len: 386]
    Sequence Number: 1    (relative sequence number)
```

```
0020  86 f7 c2 32 23 28 c7 13  33 8a 78 a7 88 d5 80 18
0030  01 f6 4d 25 00 00 01 01  08 0a 06 08 2e c7 59 d5
0040  40 46 47 45 54 20 2f 66  69 6c 65 2f 77 77 77 2f
0050  69 6e 64 65 78 2e 68 74  6d 6c 20 48 54 54 50 2f
0060  31 2e 31 0d 0a 48 6f 73  74 3a 20 33 2e 38 34 2e
0070  31 33 34 2e 32 34 37 3a  39 30 30 30 0d 0a 55 73
0080  65 72 2d 41 67 65 6e 74  3a 20 4d 6f 7a 69 6c 6c
0090  61 2f 35 2e 30 20 28 58  31 31 3b 20 55 62 75 6e
00a0  74 75 3b 20 4c 69 6e 75  78 20 78 38 36 5f 36 34
00b0  3b 20 72 76 3a 39 36 2e  30 29 20 47 65 63 6b 6f
00c0  2f 32 30 31 30 30 31 30  31 20 46 69 72 65 66 6f
00d0  78 2f 39 36 2e 30 0d 0a  41 63 63 65 70 74 3a 20
00e0  74 65 78 74 2f 68 74 6d  6c 2c 61 70 70 6c 69 63
```

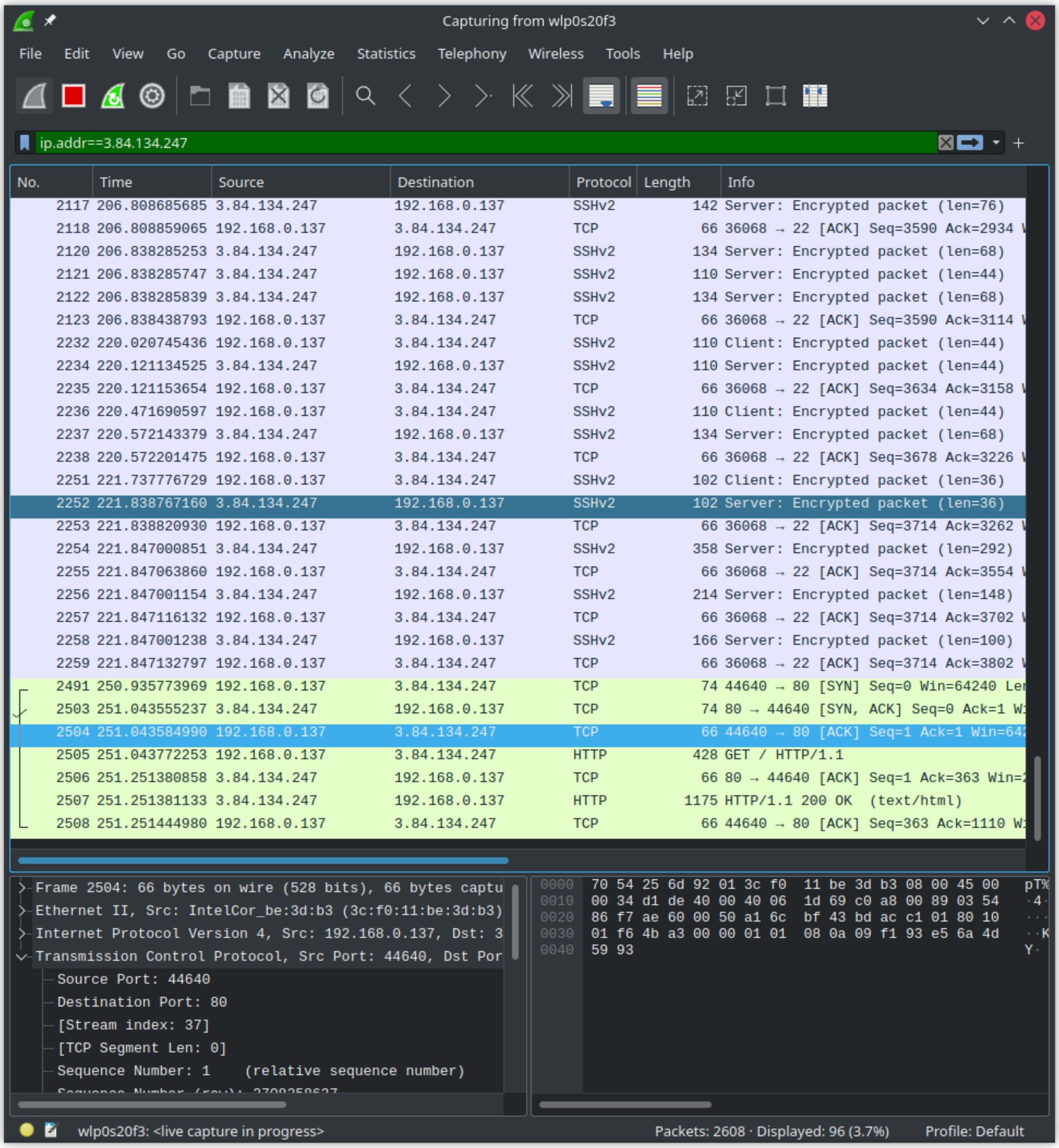
```
Transmission Control Protocol (tcp), 32 bytes    Packets: 8181 · Displayed: 50 (0.6%)    Profile: Default
```

1. I used “ip.addr==3.84.134.247 && tcp.port==9000” as a filter. I chose that filter so I would only see traffic involving the IP of my AWS instance going over the port used for the FunWebServer example.
2. Using the browser refresh generates two GET requests while the random button only generates one. Also the browser refresh briefly changes the header to test and has no image before loading the random header/image.
3. Most reponses are 200 – OK, the /file/sample.html page returns a 404 – File not found. Refreshing the random page sometimes returns a 304 – Not Modified
4. 200 – OK means the request was successful, 304 – Not Modified means the resource (the image in this case) has been cached and does not need to be re-sent. 404 – File not found means the server cannot find the resource that was requested.

5. Yes, the entirety of the HTML is captured by WireShark and is easily viewable.
6. HTTPs is more common because it encrypts the traffic and does not send it as plain text thus increasing security.
7. The server is listening on port 9000. That is not the most common port for http.
8. My system is using port 49714 to send requests. It differs from the SMTP portion because it's a different port. These ports are chosen from a specific range based on availability.
9. WireShark Capture

3.4 Setting up a "real" WebServer

1. If I only type the IP of the AWS instance into my browser the traffic goes over port 80, however I can still manually enter the :9000 after the IP and the traffic will use port 9000. The difference is because now the port does not need to be specified when going to the AWS IP in a browser, nginx automatically sends port 80 traffic to port 9000.
2. It is still HTTP because nothing is set up to use HTTPs



Capturing from wlp0s20f3
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

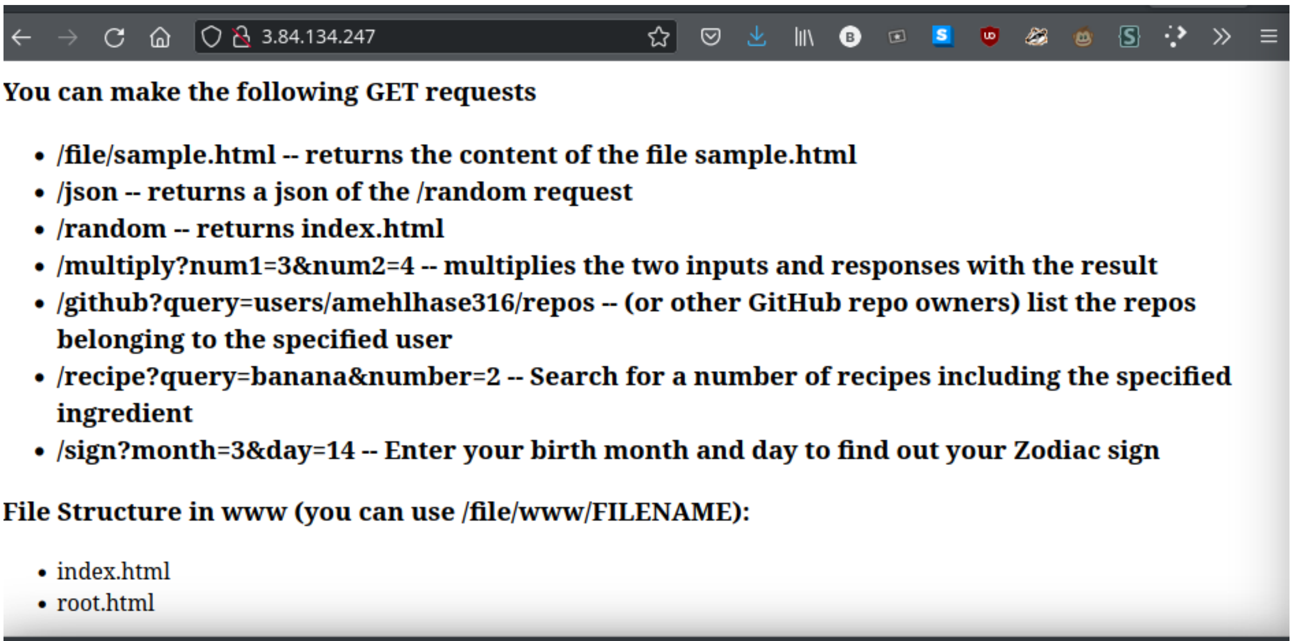
```
ip.addr==3.84.134.247

No.   Time          Source         Destination    Protocol Length Info
 2117 206.808685685 3.84.134.247   192.168.0.137  SSHv2     142 Server: Encrypted packet (len=76)
 2118 206.808859065 192.168.0.137  3.84.134.247   TCP        66 36068 → 22 [ACK] Seq=3590 Ack=2934
 2120 206.838285253 3.84.134.247   192.168.0.137  SSHv2     134 Server: Encrypted packet (len=68)
 2121 206.838285747 3.84.134.247   192.168.0.137  SSHv2     110 Server: Encrypted packet (len=44)
 2122 206.838285839 3.84.134.247   192.168.0.137  SSHv2     134 Server: Encrypted packet (len=68)
 2123 206.838438793 192.168.0.137  3.84.134.247   TCP        66 36068 → 22 [ACK] Seq=3590 Ack=3114
 2232 220.020745436 192.168.0.137  3.84.134.247   SSHv2     110 Client: Encrypted packet (len=44)
 2234 220.121134525 3.84.134.247   192.168.0.137  SSHv2     110 Server: Encrypted packet (len=44)
 2235 220.121153654 192.168.0.137  3.84.134.247   TCP        66 36068 → 22 [ACK] Seq=3634 Ack=3158
 2236 220.471690597 192.168.0.137  3.84.134.247   SSHv2     110 Client: Encrypted packet (len=44)
 2237 220.572143379 3.84.134.247   192.168.0.137  SSHv2     134 Server: Encrypted packet (len=68)
 2238 220.572201475 192.168.0.137  3.84.134.247   TCP        66 36068 → 22 [ACK] Seq=3678 Ack=3226
 2251 221.737776729 192.168.0.137  3.84.134.247   SSHv2     102 Client: Encrypted packet (len=36)
 2252 221.838767160 3.84.134.247   192.168.0.137  SSHv2     102 Server: Encrypted packet (len=36)
 2253 221.838820930 192.168.0.137  3.84.134.247   TCP        66 36068 → 22 [ACK] Seq=3714 Ack=3262
 2254 221.847000851 3.84.134.247   192.168.0.137  SSHv2     358 Server: Encrypted packet (len=292)
 2255 221.847063860 192.168.0.137  3.84.134.247   TCP        66 36068 → 22 [ACK] Seq=3714 Ack=3554
 2256 221.847001154 3.84.134.247   192.168.0.137  SSHv2     214 Server: Encrypted packet (len=148)
 2257 221.847116132 192.168.0.137  3.84.134.247   TCP        66 36068 → 22 [ACK] Seq=3714 Ack=3702
 2258 221.847001238 3.84.134.247   192.168.0.137  SSHv2     166 Server: Encrypted packet (len=100)
 2259 221.847132797 192.168.0.137  3.84.134.247   TCP        66 36068 → 22 [ACK] Seq=3714 Ack=3802
 2491 250.935773969 192.168.0.137  3.84.134.247   TCP        74 44640 → 80 [SYN] Seq=0 Win=64240 Le
 2503 251.043555237 3.84.134.247   192.168.0.137  TCP        74 80 → 44640 [SYN, ACK] Seq=0 Ack=1 W
 2504 251.043584990 192.168.0.137  3.84.134.247   TCP        66 44640 → 80 [ACK] Seq=1 Ack=1 Win=64
 2505 251.043772253 192.168.0.137  3.84.134.247   HTTP      428 GET / HTTP/1.1
 2506 251.251380858 3.84.134.247   192.168.0.137  TCP        66 80 → 44640 [ACK] Seq=1 Ack=363 Win=
 2507 251.251381133 3.84.134.247   192.168.0.137  HTTP     1175 HTTP/1.1 200 OK  (text/html)
 2508 251.251444980 192.168.0.137  3.84.134.247   TCP        66 44640 → 80 [ACK] Seq=363 Ack=1110 W

> Frame 2504: 66 bytes on wire (528 bits), 66 bytes captu
> Ethernet II, Src: IntelCor_be:3d:b3 (3c:f0:11:be:3d:b3)
> Internet Protocol Version 4, Src: 192.168.0.137, Dst: 3
v Transmission Control Protocol, Src Port: 44640, Dst Por
    Source Port: 44640
    Destination Port: 80
    [Stream index: 37]
    [TCP Segment Len: 0]
    Sequence Number: 1    (relative sequence number)
    Sequence Number (raw): 2708258627

0000  70 54 25 6d 92 01 3c f0  11 be 3d b3 08 00 45 00   pT%
0010  00 34 d1 de 40 00 40 06  1d 69 c0 a8 00 89 03 54   ·4·
0020  86 f7 ae 60 00 50 a1 6c  bf 43 bd ac c1 01 80 10   ···
0030  01 f6 4b a3 00 00 01 01  08 0a 09 f1 93 e5 6a 4d   ··K
0040  59 93                                              Y·

wlp0s20f3: <live capture in progress>    Packets: 2608 · Displayed: 96 (3.7%)    Profile: Default
```

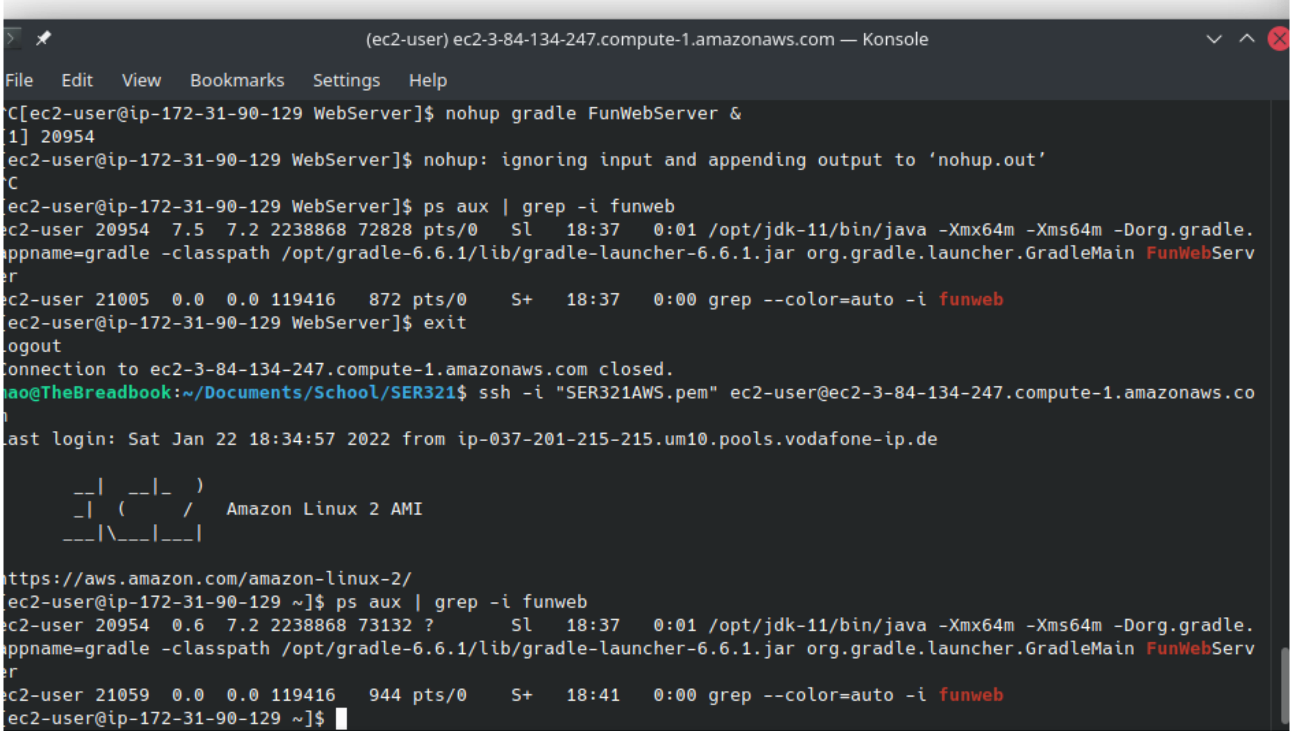


You can make the following GET requests

- **/file/sample.html -- returns the content of the file sample.html**
- **/json -- returns a json of the /random request**
- **/random -- returns index.html**
- **/multiply?num1=3&num2=4 -- multiplies the two inputs and responses with the result**
- **/github?query=users/amehlhase316/repos -- (or other GitHub repo owners) list the repos belonging to the specified user**
- **/recipe?query=banana&number=2 -- Search for a number of recipes including the specified ingredient**
- **/sign?month=3&day=14 -- Enter your birth month and day to find out your Zodiac sign**

File Structure in www (you can use /file/www/FILENAME):

- index.html
- root.html


(ec2-user) ec2-3-84-134-247.compute-1.amazonaws.com — Konsole
File
Edit
View
Bookmarks
Settings
Help

```
C[ec2-user@ip-172-31-90-129 WebServer]$ nohup gradle FunWebServer &
1] 20954
ec2-user@ip-172-31-90-129 WebServer]$ nohup: ignoring input and appending output to ‘nohup.out’
C
ec2-user@ip-172-31-90-129 WebServer]$ ps aux | grep -i funweb
c2-user 20954  7.5  7.2 2238868 72828 pts/0   Sl   18:37   0:01 /opt/jdk-11/bin/java -Xmx64m -Xms64m -Dorg.gradle.
ppname=gradle -classpath /opt/gradle-6.6.1/lib/gradle-launcher-6.6.1.jar org.gradle.launcher.GradleMain FunWebServ
r
c2-user 21005  0.0  0.0 119416   872 pts/0    S+   18:37   0:00 grep --color=auto -i funweb
ec2-user@ip-172-31-90-129 WebServer]$ exit
ogout
onnection to ec2-3-84-134-247.compute-1.amazonaws.com closed.
ao@TheBreadbook:~/Documents/School/SER321$ ssh -i "SER321AWS.pem" ec2-user@ec2-3-84-134-247.compute-1.amazonaws.co

ast login: Sat Jan 22 18:34:57 2022 from ip-037-201-215-215.um10.pools.vodafone-ip.de

       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

ttps://aws.amazon.com/amazon-linux-2/
ec2-user@ip-172-31-90-129 ~]$ ps aux | grep -i funweb
c2-user 20954  0.6  7.2 2238868 73132 ?       Sl   18:37   0:01 /opt/jdk-11/bin/java -Xmx64m -Xms64m -Dorg.gradle.
ppname=gradle -classpath /opt/gradle-6.6.1/lib/gradle-launcher-6.6.1.jar org.gradle.launcher.GradleMain FunWebServ
r
c2-user 21059  0.0  0.0 119416   944 pts/0    S+   18:41   0:00 grep --color=auto -i funweb
ec2-user@ip-172-31-90-129 ~]$ 
```

3.6.1 Some programming on your WebServer – Multiply

I decided to have the page print some error messages and also use default values for the multiplication. It can now handle any number of parameters, non-numerical parameters, and other cases without crashing. I used error code 400 – Bad Request as the error code. I chose this one because the issues being handled stem from incorrect requests from the client so the code should 4xx. Code 400 specifically states “The server could not understand the request due to invalid syntax.” because bad syntax is what we’re dealing with here. I also used code 414 – URI too long if the user enters more than 2 numbers.