Martin Alemajoh
Arizona State University
Assignment 2

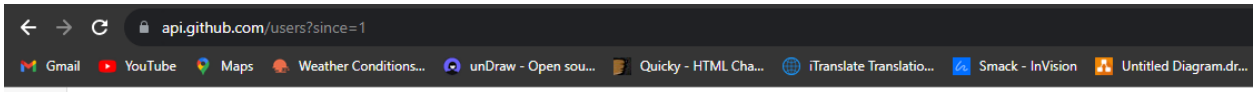
1. Understanding HTTP

   https://api.github.com/repos/malemajo/team-exercise/commits


```
// 20231025192209
// https://api.github.com/repos/malemajo/team-exercise/commits

[
  {
    "sha": "71b9074bb90b10b6eee0127c38a1655ceea63cfc",
    "node_id": "C_kwDOKhtDbNoAKDcxYjkwNzRiYjkwYjEwYjZlZWUwMTI3YzM4YTE2NTVjZWVhNjNjZmM",
    "commit": {
      "author": {
        "name": "Justin Salas",
        "email": "justindsalas@gmail.com",
        "date": "2023-10-19T02:31:37Z"
      },
      "committer": {
        "name": "GitHub",
        "email": "noreply@github.com",
        "date": "2023-10-19T02:31:37Z"
      },
      "message": "Update PULLREQUEST.md\n\nAdded my sentence for other team",
      "tree": {
        "sha": "f01e3d94b9b27b3cebb868ef28bbbacd3d6f7c1d",
        "url": "https://api.github.com/repos/malemajo/team-exercise/git/trees/f01e3d94b9b27b3cebb868ef28bbbacd3d6f7c1d"
      },
      "url": "https://api.github.com/repos/malemajo/team-exercise/git/commits/71b9074bb90b10b6eee0127c38a1655ceea63cfc",
      "comment_count": 0,
      "verification": {
        "verified": true,
        "reason": "valid",
        "signature": "-----BEGIN PGP SIGNATURE-----
\n\nwsBcBAABCAAQBQJlMJUJCRBK7hj4Ov3rIwAAA9cIAJec0EgctODwLKSIgzK05J4E\nW5DHXLgMCg3LW24h7t4nahqUwixs5SiTPvmROIIuuw8kQrAUqCqHrz5Q9+fMbYma\no+HP7H3iuG+
6C9OiDZP6uQqeoYWQTBdRoHzrHyAWsCMehU+YK+2Uo5Puyp66bk/AThnXH5N\nT620CvXTNs6iTRnE/Kc2jHk3i144YHtmXJO1JuA1QGNuq/TZubjcC+oWpPiMnrk=\n=1PmI\n-----END PGP
        "payload": "tree f01e3d94b9b27b3cebb868ef28bbbacd3d6f7c1d\nparent 3928f47a5da6d93774eca13c4e12001573d495a1\nauthor Justin Salas <justindsal
sentence for other team"
      }
    },
    "url": "https://api.github.com/repos/malemajo/team-exercise/commits/71b9074bb90b10b6eee0127c38a1655ceea63cfc",
    "html_url": "https://github.com/malemajo/team-exercise/commit/71b9074bb90b10b6eee0127c38a1655ceea63cfc",
    "comments_url": "https://api.github.com/repos/malemajo/team-exercise/commits/71b9074bb90b10b6eee0127c38a1655ceea63cfc/comments",
```

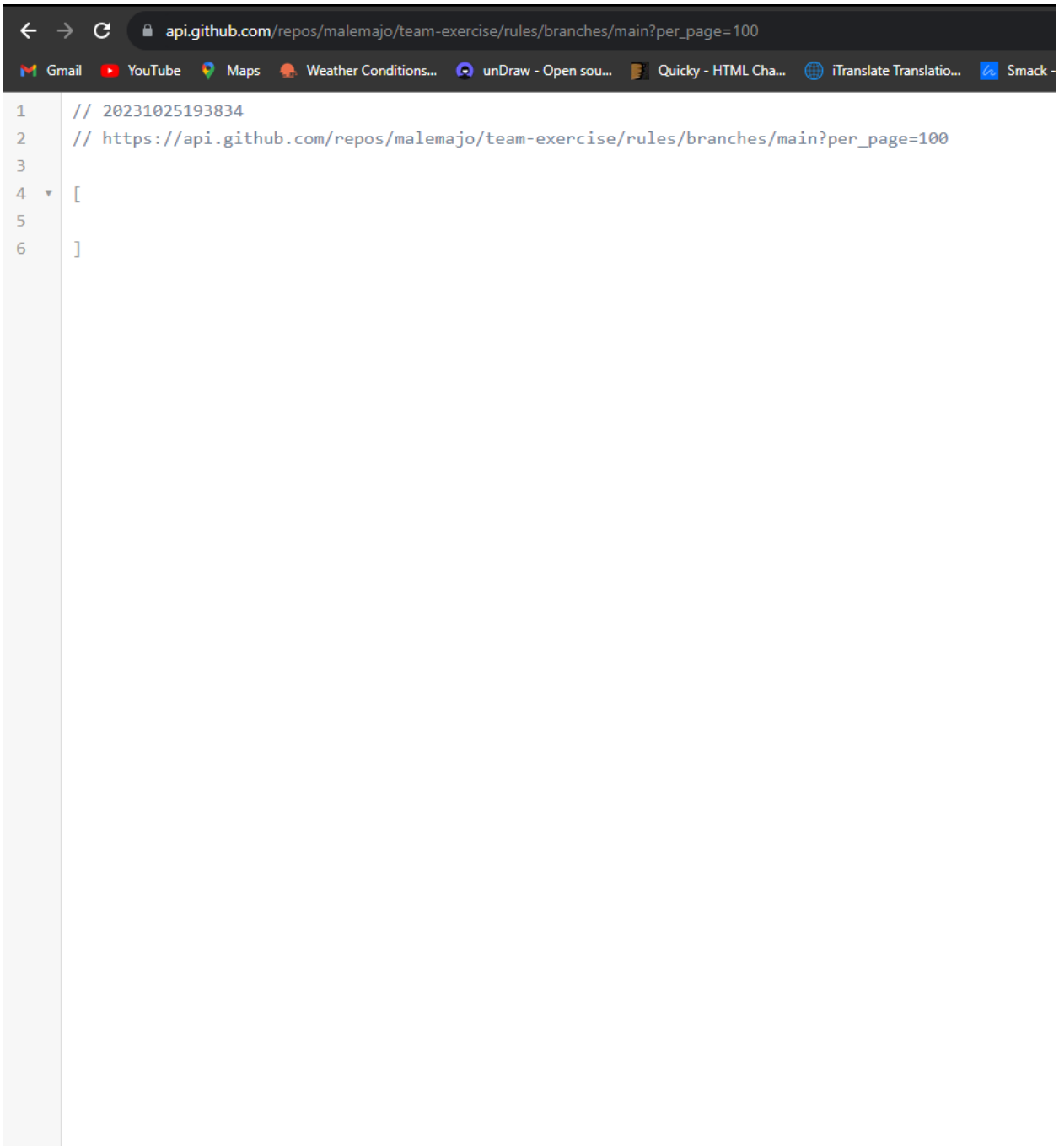
https://api.github.com/users?since=1




```
// 20231025193530
// https://api.github.com/users?since=1

[
  {
    "login": "defunkt",
    "id": 2,
    "node_id": "MDQ6VXNlcjI=",
    "avatar_url": "https://avatars.githubusercontent.com/u/2?v=4",
    "gravatar_id": "",
    "url": "https://api.github.com/users/defunkt",
    "html_url": "https://github.com/defunkt",
    "followers_url": "https://api.github.com/users/defunkt/followers",
    "following_url": "https://api.github.com/users/defunkt/following{/other_user}",
    "gists_url": "https://api.github.com/users/defunkt/gists{/gist_id}",
    "starred_url": "https://api.github.com/users/defunkt/starred{/owner}{/repo}",
    "subscriptions_url": "https://api.github.com/users/defunkt/subscriptions",
    "organizations_url": "https://api.github.com/users/defunkt/orgs",
    "repos_url": "https://api.github.com/users/defunkt/repos",
    "events_url": "https://api.github.com/users/defunkt/events{/privacy}",
    "received_events_url": "https://api.github.com/users/defunkt/received_events",
    "type": "User",
    "site_admin": false
  },
  {
    "login": "pjhyett",
    "id": 3,
    "node_id": "MDQ6VXNlcjM=",
    "avatar_url": "https://avatars.githubusercontent.com/u/3?v=4",
    "gravatar_id": "",
    "url": "https://api.github.com/users/pjhyett",
    "html_url": "https://github.com/pjhyett",
    "followers_url": "https://api.github.com/users/pjhyett/followers",
    "following_url": "https://api.github.com/users/pjhyett/following{/other_user}",
    "gists_url": "https://api.github.com/users/pjhyett/gists{/gist_id}",
    "starred_url": "https://api.github.com/users/pjhyett/starred{/owner}{/repo}",
    "subscriptions_url": "https://api.github.com/users/pjhyett/subscriptions",
    "organizations_url": "https://api.github.com/users/pjhyett/orgs",
```

https://api.github.com/repos/malemajo/team-exercise/rules/branches/main?per_page=100



List Users Endpoint:

URL: https://api.github.com/users?since=1
Method: GET
Description: This endpoint lists all GitHub users in the order that they signed up on GitHub.
Query Parameters:
since: This is an integer parameter that allows you to specify the user ID to start the listing from. In this case, since=1 indicates that the listing should start from the user with ID 1.
This endpoint will return a paginated list of users starting from the specified user ID. Each user object in the response will include information such as the user's login, ID, node_id, avatar_url, gravatar_id, url, html_url, followers_url, following_url, gists_url, starred_url, subscriptions_url, organizations_url, repos_url, events_url, received_events_url, type, site_admin, and more.

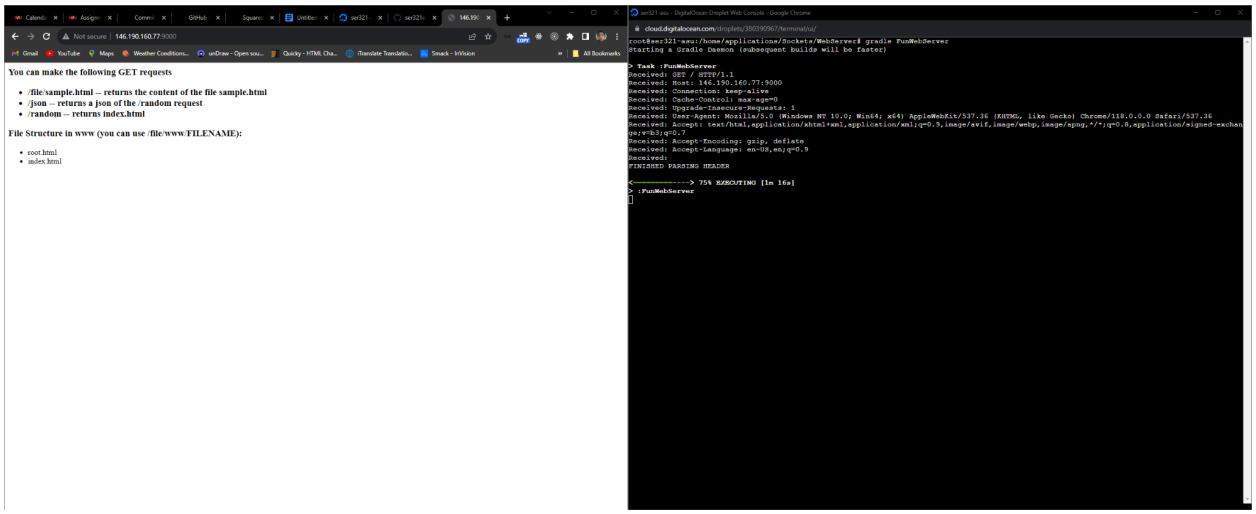
Get Branch Endpoint:

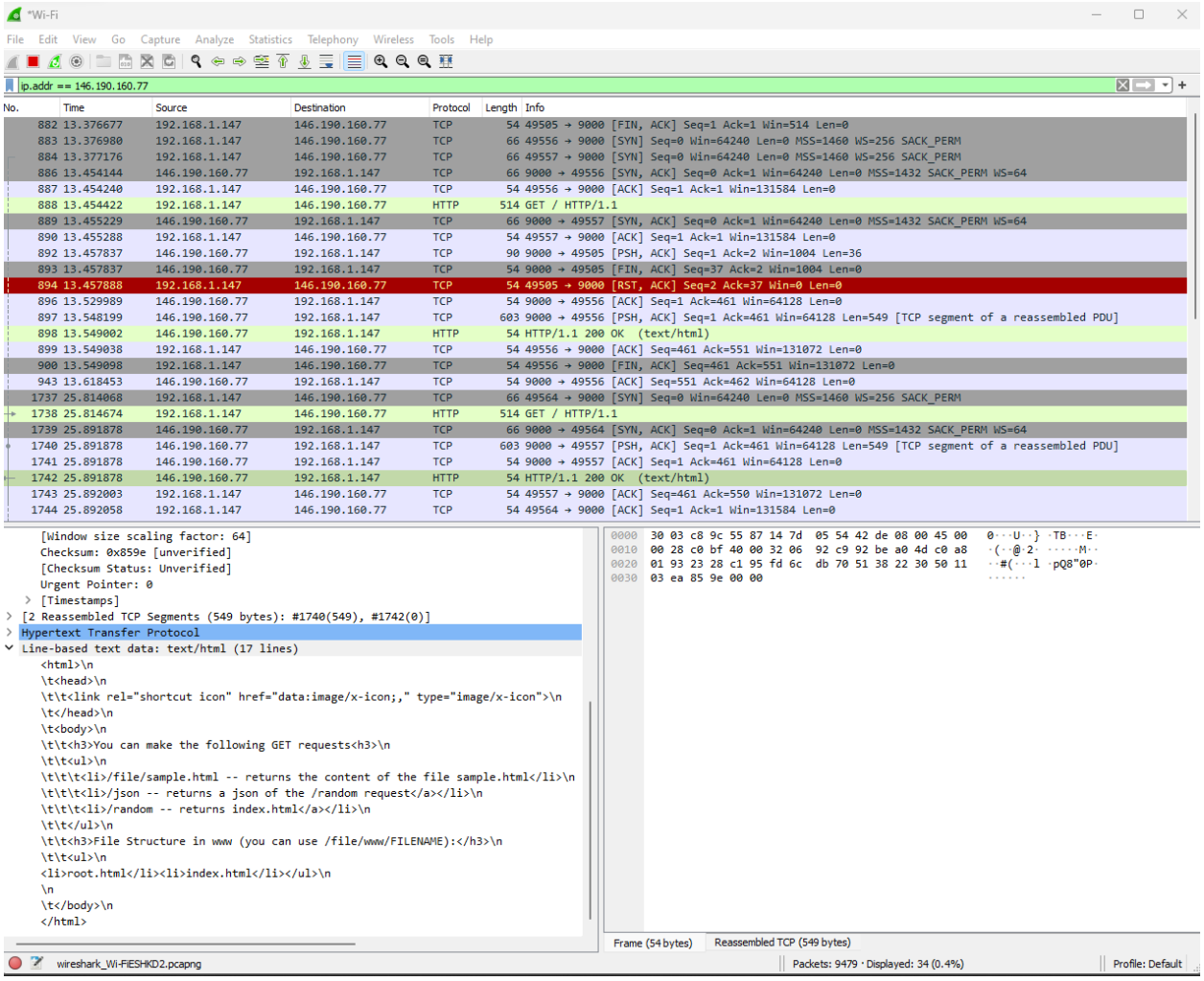
URL: https://api.github.com/repos/malemajo/team-exercise/rules/branches/main?per_page=100
Method: GET
Description: This endpoint fetches a single branch from a repository.
URL Parameters:
owner: The owner of the repository. In this case, it's malemajo.
repo: The name of the repository. In this case, it's team-exercise.
branch: The name of the branch. In this case, it's main.
Query Parameters:
per_page: This is an integer parameter that specifies the number of items per page. In this case, per_page=100 indicates that 100 items should be returned per page. However, this parameter may not have an effect on this particular endpoint as it's designed to fetch a single branch.
This endpoint will return information about the specified branch, including the name of the branch, commit details, and the protected status of the branch.

Stateless Communication:
In stateless communication, each request from a client to a server must contain all the information the server needs to fulfill the request.
There's no retention of data between transactions. Each request is processed based solely on the information that comes with it, without any awareness of previous interactions.
HTTP, without any additional configurations, is an example of a stateless protocol where each request is processed independently.
Stateless architectures are often simpler and more scalable due to the lack of inter-dependency between requests.

Stateful Communication:
In stateful communication, data from previous interactions can be stored and used to inform current transactions. This is often managed through sessions or other mechanisms.
There's a retention of data between transactions, which allows for a continuity of experience.
For example, a server might remember a client's previous actions or settings, providing a more personalized or continuous experience across interactions.
Protocols like TCP are stateful as they maintain a connection and a state of communication between a client and a server, tracking interactions over time.
Stateful architectures can provide richer interactions and experiences, but they can also be more complex and potentially less scalable due to the overhead of managing state.

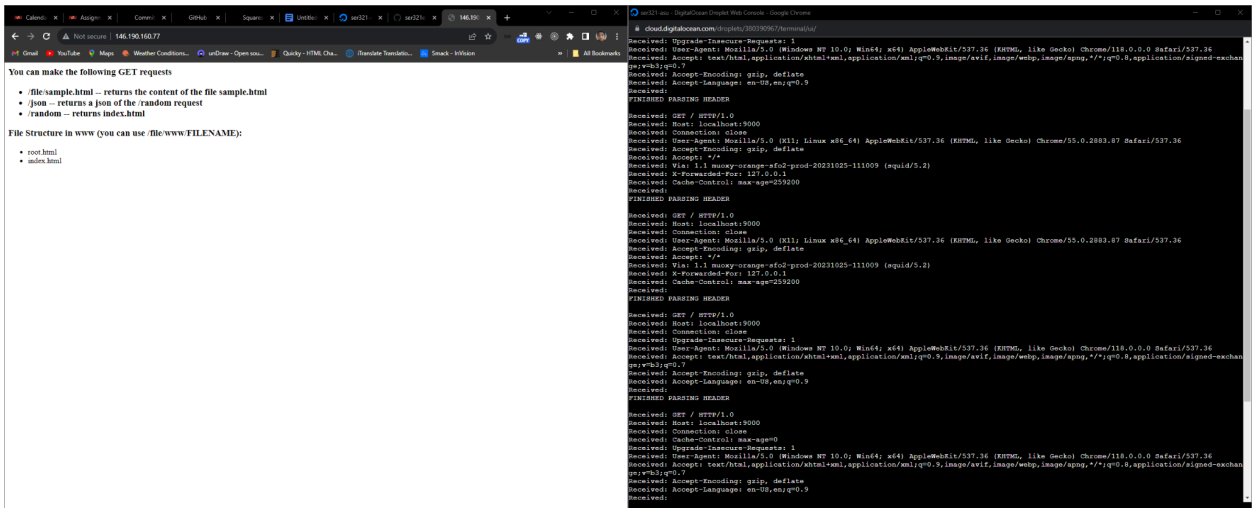
Running A simple Java WebServer





1. ip.addr == 146.190.160.77
   I chose this filter because it helps to narrow down the traffic captured by wireshark to only show traffic between my local machine and the remote server I am interested in.
2. A random image is displayed.
3. / -> 200 OK
   /json => 200 OK
   /file/sample.html => 404 NOT FOUND
4. 200 => Success Response
   404 => Resource not found
5. Yes. It return the plain html text
6. HTTPS encrypts the data
7. 9000. No HTTP typically listens on 80 by default.
8. 49793

1. http://146.190.160.77/
2. It is different. It uses port 80
3. It is HTTP because no SSL certificate exists on the server.



*Wi-Fi

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

ip.addr == 146.190.160.77

No.	Time	Source	Destination	Protocol	Length	Info
2296…	2944.185206	192.168.1.147	146.190.160.77	TCP	66	51890 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
2296…	2944.185444	192.168.1.147	146.190.160.77	TCP	66	51891 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
2297…	2944.258452	146.190.160.77	192.168.1.147	TCP	66	80 → 51891 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1432 SACK_PERM WS=64
2297…	2944.258579	192.168.1.147	146.190.160.77	TCP	54	51891 → 80 [ACK] Seq=1 Ack=1 Win=131584 Len=0
2297…	2944.258742	192.168.1.147	146.190.160.77	HTTP	509	GET / HTTP/1.1
2297…	2944.264473	146.190.160.77	192.168.1.147	TCP	66	80 → 51890 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1432 SACK_PERM WS=64
2297…	2944.264644	192.168.1.147	146.190.160.77	TCP	54	51890 → 80 [ACK] Seq=1 Ack=1 Win=131584 Len=0
2297…	2944.331199	146.190.160.77	192.168.1.147	TCP	54	80 → 51891 [ACK] Seq=1 Ack=456 Win=64128 Len=0
2297…	2944.335062	146.190.160.77	192.168.1.147	TCP	54	[TCP Dup ACK 229750#1] 80 → 51891 [ACK] Seq=1 Ack=456 Win=64128 Len=0
2297…	2944.351888	146.190.160.77	192.168.1.147	HTTP	564	HTTP/1.1 200 OK (text/html)
2297…	2944.407479	192.168.1.147	146.190.160.77	TCP	54	51891 → 80 [ACK] Seq=456 Ack=511 Win=131072 Len=0
2298…	2944.655868	146.190.160.77	192.168.1.147	TCP	564	[TCP Spurious Retransmission] 80 → 51891 [PSH, ACK] Seq=1 Ack=456 Win=64128 Len=510
2298…	2944.655928	192.168.1.147	146.190.160.77	TCP	66	[TCP Dup ACK 229763#1] 51891 → 80 [ACK] Seq=456 Ack=511 Win=131072 Len=0 SLE=1 SRE=511
2333…	2989.266388	192.168.1.147	146.190.160.77	TCP	55	[TCP Keep-Alive] 51890 → 80 [ACK] Seq=0 Ack=1 Win=131584 Len=1
2333…	2989.340892	146.190.160.77	192.168.1.147	TCP	66	[TCP Window Update] 80 → 51890 [ACK] Seq=1 Ack=1 Win=64256 Len=0 SLE=0 SRE=1
2333…	2989.354360	192.168.1.147	146.190.160.77	TCP	55	[TCP Keep-Alive] 51891 → 80 [ACK] Seq=455 Ack=511 Win=131072 Len=1
2333…	2989.425313	146.190.160.77	192.168.1.147	TCP	66	[TCP Keep-Alive ACK] 80 → 51891 [ACK] Seq=511 Ack=456 Win=64128 Len=0 SLE=455 SRE=456
2343…	3004.406106	146.190.160.77	192.168.1.147	TCP	54	80 → 51890 [FIN, ACK] Seq=1 Ack=1 Win=64256 Len=0
2343…	3004.406168	192.168.1.147	146.190.160.77	TCP	54	51890 → 80 [ACK] Seq=1 Ack=2 Win=131584 Len=0
2355…	3019.362869	146.190.160.77	192.168.1.147	TCP	54	80 → 51891 [FIN, ACK] Seq=511 Ack=456 Win=64128 Len=0
2355…	3019.363030	192.168.1.147	146.190.160.77	TCP	54	51891 → 80 [ACK] Seq=456 Ack=512 Win=131072 Len=0
2355…	3020.253993	192.168.1.147	146.190.160.77	TCP	54	51890 → 80 [FIN, ACK] Seq=1 Ack=2 Win=131584 Len=0
2356…	3020.254037	192.168.1.147	146.190.160.77	TCP	54	51891 → 80 [FIN, ACK] Seq=456 Ack=512 Win=131072 Len=0
2356…	3020.324893	146.190.160.77	192.168.1.147	TCP	54	80 → 51891 [ACK] Seq=512 Ack=457 Win=64128 Len=0
2356…	3020.325606	146.190.160.77	192.168.1.147	TCP	54	80 → 51890 [ACK] Seq=2 Ack=2 Win=64256 Len=0

```
> Ethernet II, Src: CloudNet_9c:55:87 (30:03:c8:9c:55:87), Dst: SercommP_54:42:dc (14:7
> Internet Protocol Version 4, Src: 192.168.1.147, Dst: 146.190.160.77
v Transmission Control Protocol, Src Port: 51890, Dst Port: 80, Seq: 0, Ack: 1, Len: 1
    Source Port: 51890
    Destination Port: 80
    [Stream index: 806]
    [Conversation completeness: Complete, WITH_DATA (31)]
    [TCP Segment Len: 1]
    Sequence Number: 0    (relative sequence number)
    Sequence Number (raw): 1329873069
    [Next Sequence Number: 1    (relative sequence number)]
    Acknowledgment Number: 1    (relative ack number)
    Acknowledgment number (raw): 1894774109
    0101 .... = Header Length: 20 bytes (5)
  > Flags: 0x010 (ACK)
    Window: 514
    [Calculated window size: 131584]
    [Window size scaling factor: 256]
    Checksum: 0xf748 [unverified]
    [Checksum Status: Unverified]
    Urgent Pointer: 0
  > [Timestamps]
  > [SEQ/ACK analysis]
    TCP payload (1 byte)
> Data (1 byte)
```

```
0000  14 7d 05 54 42 dc 30 03  c8 9c 55 87 08 00 45 00   ·}·TB·0· ··U···E·
0010  00 29 fc c8 40 00 80 06  08 bf c0 a8 01 93 92 be   ·)··@··· ********
0020  a0 4d ca b2 00 50 4f 44  40 ad 70 ef f5 5d 50 10   ·M···POD @·p··]P·
0030  02 02 f7 48 00 00 00                               ···H···
```

Source Port (tcp.srcport), 2 bytes | Packets: 240938 · Displayed: 585 (0.2%) | Profile: Default

Weather Request Protocol:
Request
URL: host:PORT/weather?city=CITY_NAME&country=COUNTRY_CODE
Path: weather
Body Parameters:
city (String) - required
country (String) - required

Response
OK Response (200)

Example:

```
{
  "city": "London",
  "country": "UK",
  "temperature": "15°C",
  "condition": "Cloudy"
}
```

HTTP response status codes
200 – OK, the weather data could be fetched
400 - Bad Request, either city or country parameter is missing
404 - Not Found, weather data for the specified location could not be found

Translation Request Protocol:
Request
URL: host:PORT/translate?text=TEXT&target=TARGET_LANGUAGE
Path: translate
Body Parameters:
text (String) - required
target (String) - required

Response
OK Response (200)

Example:


```
{
  "original_text": "hello",
  "translated_text": "hola",
  "target_language": "es"
}
```


HTTP response status codes

200 – OK, the text was translated successfully
400 - Bad Request, either text or target parameter is missing
404 - Not Found, translation service did not respond or target language is not supported