Grokking HTTP

Ben Ramsey • Longhorn PHP Conference • October 23, 2025

We will cover...

- HTTP methods
- HTTP status codes
- HTTP headers
- Tools to make and inspect HTTP requests
- Some interesting requests

We will not cover...

- How HTTP is sent “on the wire”
- SPDY, QUIC, HPACK, QPACK
- In other words, the networking details of HTTP/1.1, HTTP/2, or HTTP/3
- These are interesting but largely abstracted from web developers

Disclaimer

- I build APIs for a living
- Most of my focus on HTTP is from the standpoint of APIs
- Much of what we'll talk about is very API-centric
- Even if you don't build or consume APIs, all web developers can benefit from a greater understanding of HTTP

Grok?

What does it mean to grok?

“Grok means to understand so thoroughly that the observer becomes a part of the observed—to merge, blend, intermarry, lose identity in group experience.”

—Robert A. Heinlein, *Stranger in a Strange Land*

“To understand. Connotes intimate and exhaustive knowledge. When you claim to ‘grok’ some knowledge or technique, you are asserting that you have not merely learned it in a detached instrumental way but that it has become part of you, part of your identity. For example, to say that you ‘know’ LISP is simply to assert that you can code in it if necessary—but to say you ‘grok’ LISP is to claim that you have deeply entered the world-view and spirit of the language, with the implication that it has transformed your view of programming.”

—Jargon File

“When you’re developing a particular code suite over a long period of time, you eventually find yourself ‘in the zone.’ In that state, you seem to have the design and the control flow and the data structures and the naming conventions and the modular decomposition and every other aspect of the program constantly at your mental fingertips. You understand the code in a profound way. It’s easy to ‘see’ problems directly and locate bugs quickly, sometimes without even quite knowing how you knew. You truly grok the source.”

—Damian Conway, *Perl Best Practices*

What is HTTP?

Hypertext Transfer Protocol

hypertext

A multi-linear set of objects, building a network by using ***logical links*** (the so-called hyperlinks) between the nodes (e.g. text or words).

protocol

A set of rules and regulations that define how data is transmitted across a network.

HTTP is a set of rules for
transferring hypertext
across the internet.

HTTP forms the basis of
everything we do ***on the web***.

HTTP is the ***web***.

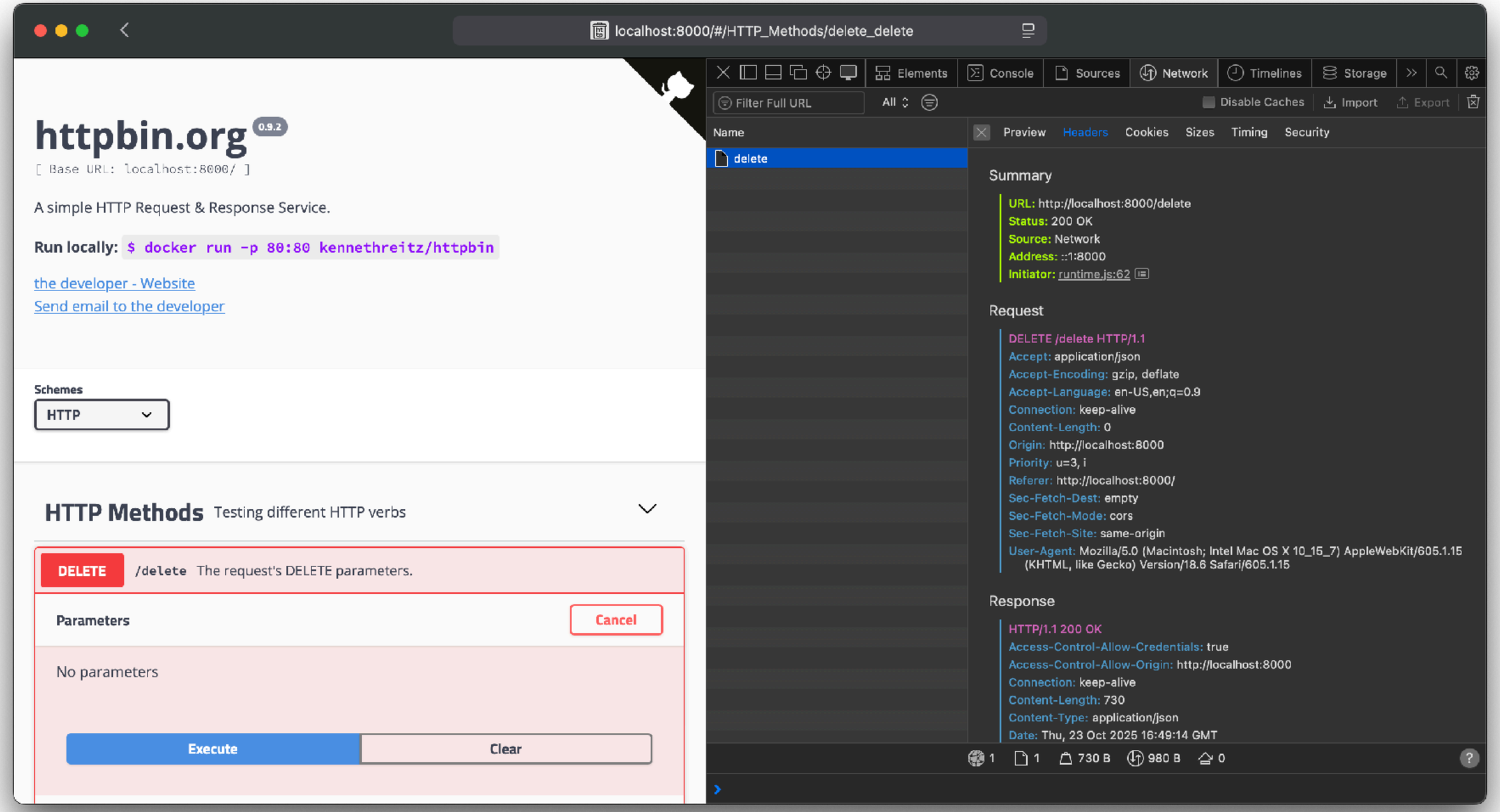
```
GET / HTTP/1.1
Accept: */*
Host: benramsey.com


HTTP/1.1 301 Moved Permanently
Content-Type: text/html; charset=utf-8
Date: Tue, 21 Oct 2025 05:22:57 GMT
location: https://ben.ramsey.dev/


GET / HTTP/1.1
Accept: */*
Host: ben.ramsey.dev


HTTP/1.1 200 OK
Content-Type: text/html; charset=UTF-8
Date: Tue, 21 Oct 2025 05:23:08 GMT
Link: </assets/styles/app.css>; rel="preload"; as="style"
```

Tools


localhost:8000/#/HTTP_Methods/delete_delete
httpbin.org
0.9.2
[Base URL: localhost:8000/]
A simple HTTP Request & Response Service.
Run locally: $ docker run -p 80:80 kennethreitz/httpbin
the developer - Website
Send email to the developer
Schemes
HTTP
HTTP Methods
Testing different HTTP verbs
DELETE
/delete
The request's DELETE parameters.
Parameters
Cancel
No parameters
Execute
Clear
Elements
Console
Sources
Network
Timelines
Storage
Filter Full URL
All
Disable Caches
Import
Export
Name
delete
Preview
Headers
Cookies
Sizes
Timing
Security
Summary
URL: http://localhost:8000/delete
Status: 200 OK
Source: Network
Address: ::1:8000
Initiator: runtime.js:62
Request
DELETE /delete HTTP/1.1
Accept: application/json
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: keep-alive
Content-Length: 0
Origin: http://localhost:8000
Priority: u=3, i
Referer: http://localhost:8000/
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/18.6 Safari/605.1.15
Response
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Origin: http://localhost:8000
Connection: keep-alive
Content-Length: 730
Content-Type: application/json
Date: Thu, 23 Oct 2025 16:49:14 GMT
1
1
730 B
980 B
0

curl.se

Download Documentation libcurl Get Help Development

curl://

command line tool and library
for transferring data with URLs
(since 1998)

What is curl used for?

curl is used in command lines or scripts to transfer data. curl is also **libcurl**, used in cars, television sets, routers, printers, audio equipment, mobile phones, tablets, medical devices, settop boxes, computer games, media players and is the Internet transfer engine for countless software applications in over *twenty billion installations*.

curl is used daily by virtually every Internet-using human on the globe.

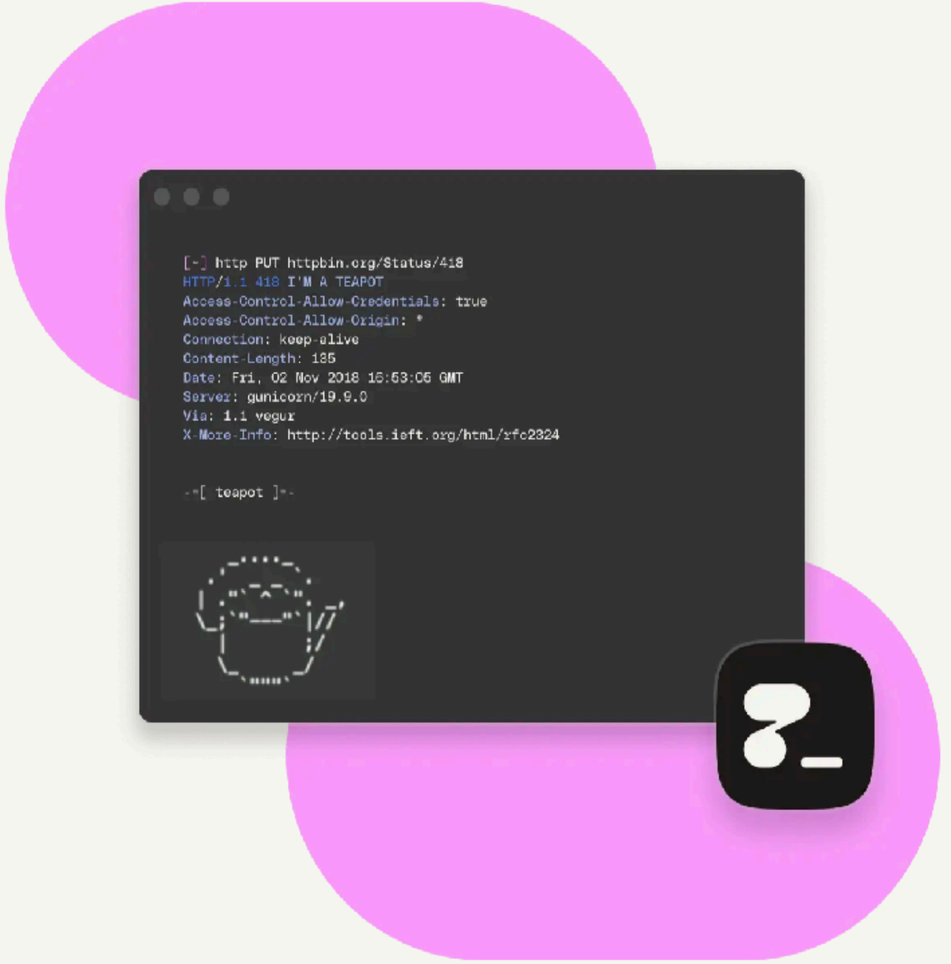
```
❯ curl -X DELETE “http://localhost:8000/delete" -H "accept: application/json"
{
  "args": {},
  "data": "",
  "files": {},
  "form": {},
  "headers": {
    "Accept": "application/json",
    "Host": "localhost:8888",
    "User-Agent": "curl/8.7.1"
  },
  "json": null,
  "origin": "192.168.65.1",
  "url": "http://localhost:8888/delete"
}
```

HTTPIE FOR TERMINAL

A simple yet powerful command-line HTTP and API testing client for the API era.

Install | Read docs | Try online

Star 36,843



```
❯ http -v DELETE http://localhost:8000/delete accept:”application/json"
DELETE /delete HTTP/1.1
Accept-Encoding: gzip, deflate, zstd
Connection: keep-alive
Content-Length: 0
Host: localhost:8888
User-Agent: HTTPie/3.2.4
accept: application/json


HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Origin: *
Connection: keep-alive
Content-Length: 379
Content-Type: application/json
Date: Thu, 23 Oct 2025 16:37:38 GMT
Server: gunicorn/19.9.0

{...}
```

www.usebruno.com

Bruno

Bruno Vs Postman Docs Pricing Changelog Roadmap Support Download VS Code 38K ☆

Reinventing the API Client

Bruno is a Git-integrated, fully offline, and open-source API client

Developer First

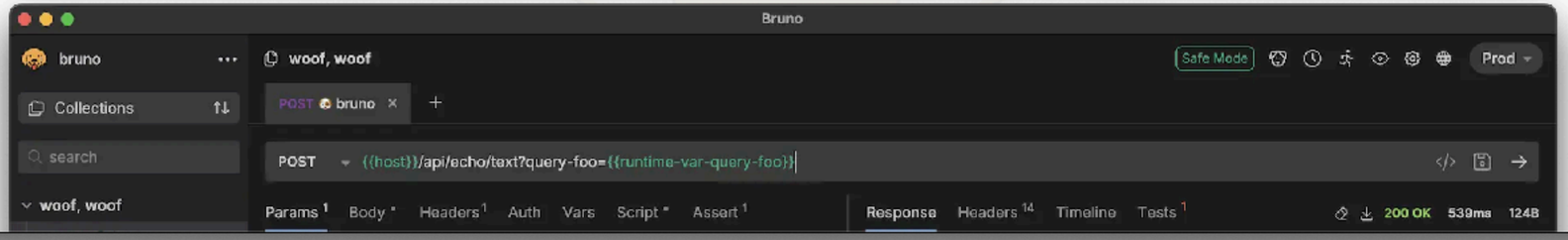
Open-source community and extensibility are our cornerstones. We're proud to be an API-client and commit to not bloating our product with unnecessary 'platform' features.

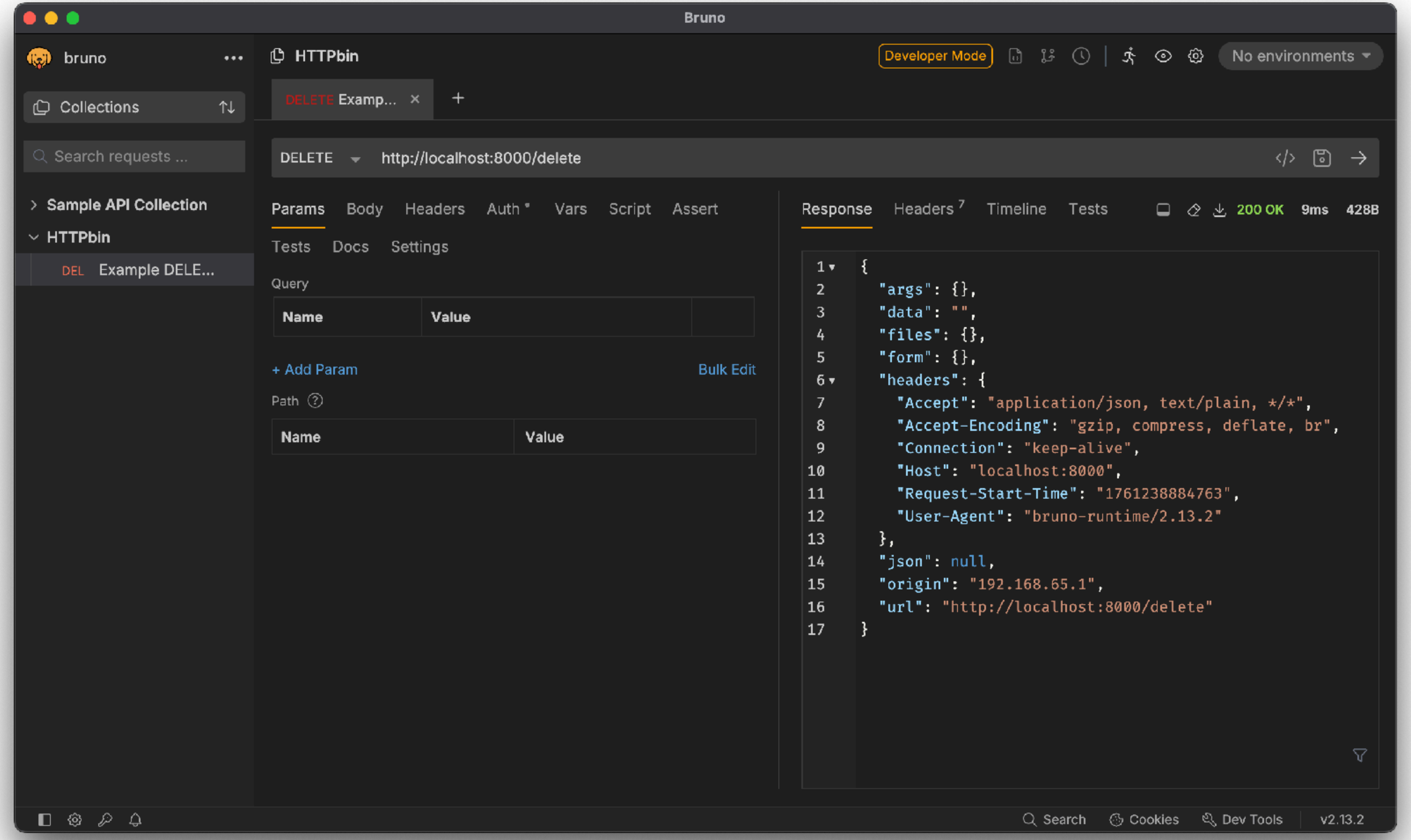
Native Git Integration

Collaboration requires context and a shared location. Collections can live directly in your Git provider so they're always versioned.

Offline only

We take security and privacy seriously. Bruno is an offline tool and there is no syncing of your data to any cloud.




Bruno
bruno
Collections
Search requests ...
Sample API Collection
HTTPbin
DEL Example DELE...
HTTPbin
Developer Mode
No environments
DELETE Examp...
DELETE http://localhost:8000/delete
Params Body Headers Auth Vars Script Assert
Tests Docs Settings
Query
Name Value
+ Add Param
Bulk Edit
Path
Name Value
Response Headers 7 Timeline Tests
200 OK 9ms 428B
{
"args": {},
"data": "",
"files": {},
"form": {},
"headers": {
"Accept": "application/json, text/plain, */*",
"Accept-Encoding": "gzip, compress, deflate, br",
"Connection": "keep-alive",
"Host": "localhost:8000",
"Request-Start-Time": "1761238884763",
"User-Agent": "bruno-runtime/2.13.2"
},
"json": null,
"origin": "192.168.65.1",
"url": "http://localhost:8000/delete"
}
Search
Cookies
Dev Tools
v2.13.2

www.charlesproxy.com

Charles

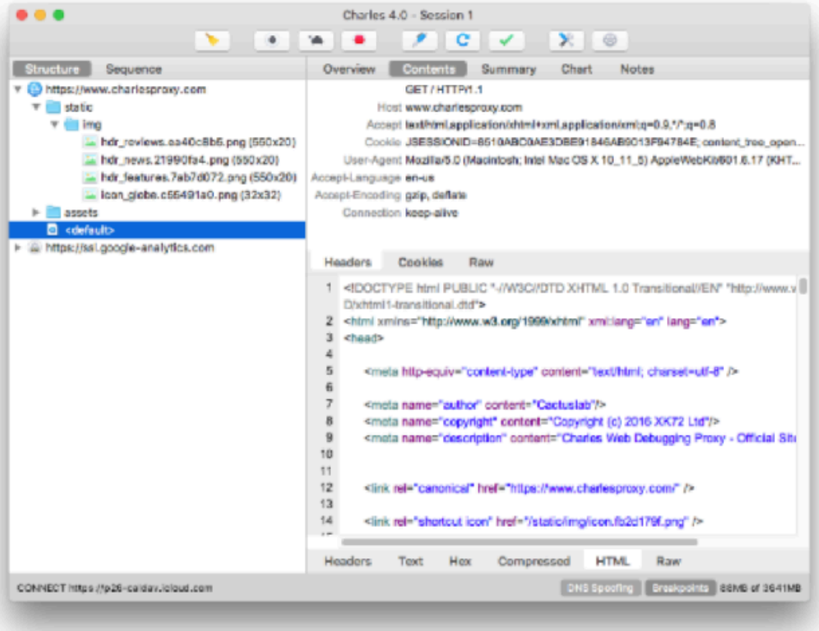
WEB DEBUGGING PROXY APPLICATION

for Windows, Mac OS and Linux

HOME | OVERVIEW | DOCUMENTATION | DOWNLOAD | BUY | SUPPORT

Charles is an HTTP proxy / HTTP monitor / Reverse Proxy that enables a developer to view all of the HTTP and SSL / HTTPS traffic between their machine and the Internet. This includes requests, responses and the HTTP headers (which contain the cookies and caching information).

READ MORE

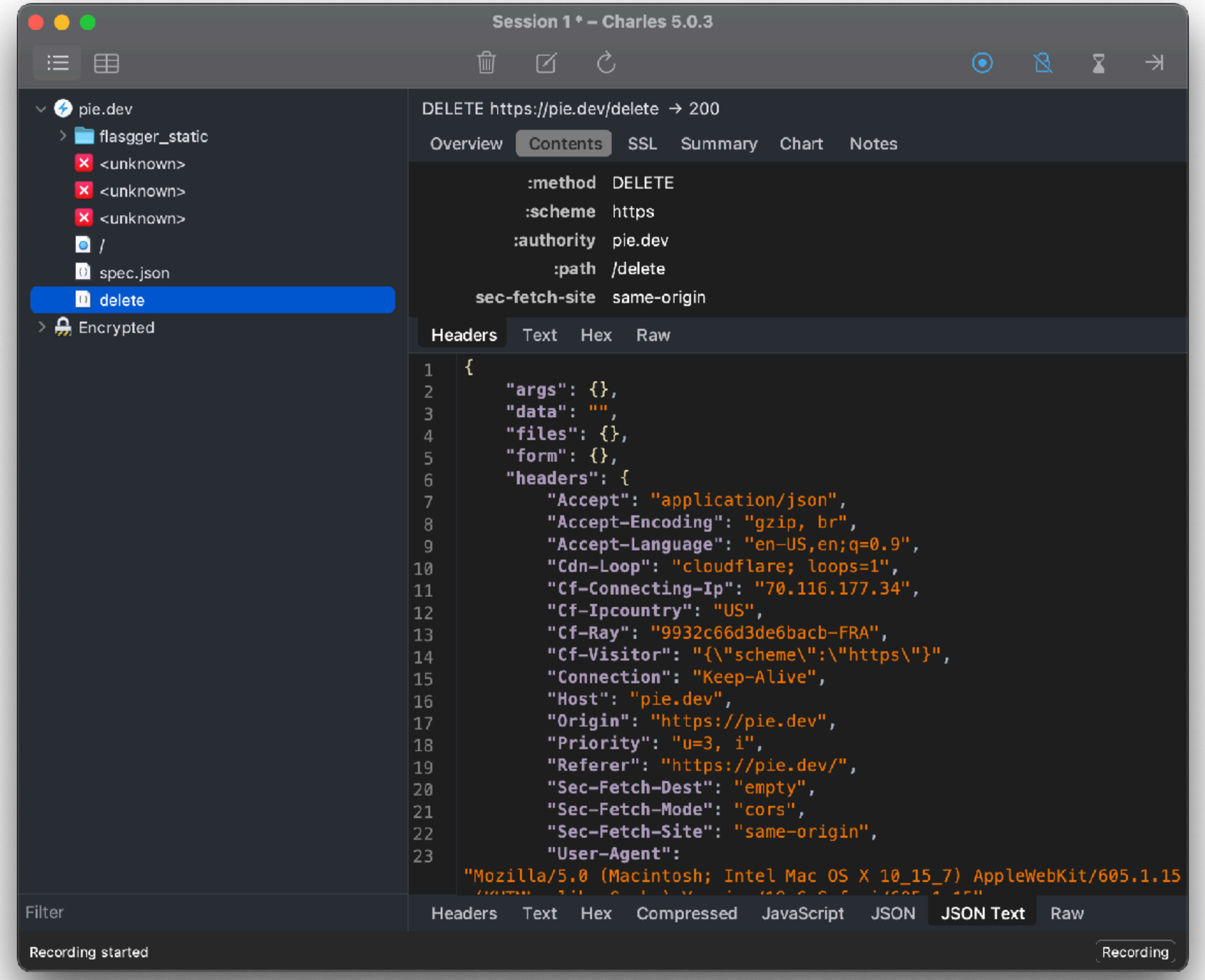


DOWNLOAD

Download a free trial
Version 5.0.3

FEATURES

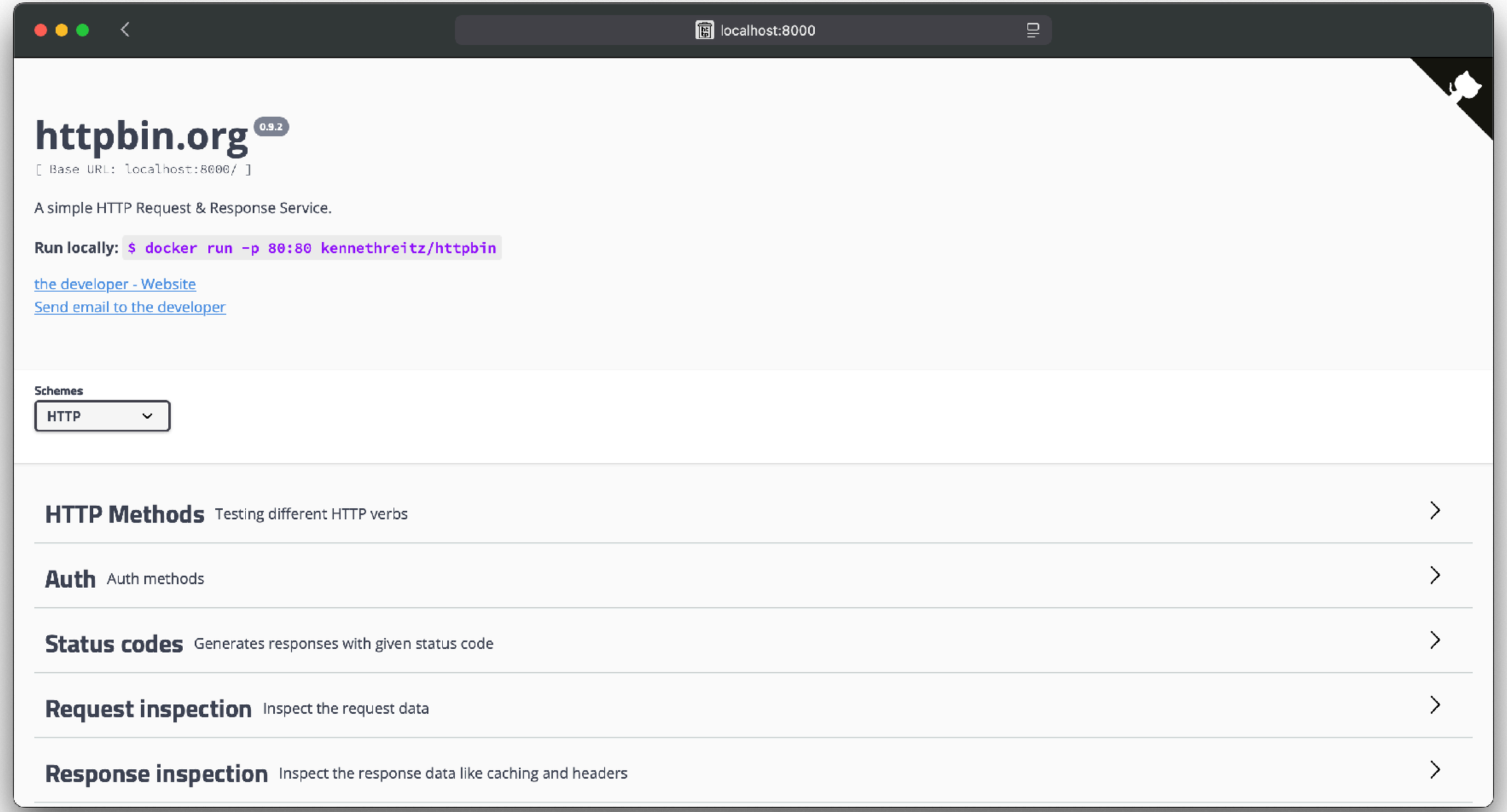
- **Records all traffic between your browser and the Internet**
- **Reveals the contents of all requests, responses, cookies and headers**
- **Supports SSL and HTTPS**
- **Saves valuable time**
- **Simulates slower internet connections**
- **Download statistics**
- **Configurable**


Session 1 * – Charles 5.0.3
pie.dev
flasgger_static
<unknown>
<unknown>
<unknown>
/
spec.json
delete
Encrypted
Filter
DELETE https://pie.dev/delete → 200
Overview
Contents
SSL
Summary
Chart
Notes
:method DELETE
:scheme https
:authority pie.dev
:path /delete
sec-fetch-site same-origin
Headers
Text
Hex
Raw
{
"args": {},
"data": "",
"files": {},
"form": {},
"headers": {
"Accept": "application/json",
"Accept-Encoding": "gzip, br",
"Accept-Language": "en-US,en;q=0.9",
"Cdn-Loop": "cloudflare; loops=1",
"Cf-Connecting-Ip": "70.116.177.34",
"Cf-Ipcountry": "US",
"Cf-Ray": "9932c66d3de6bacb-FRA",
"Cf-Visitor": "{\"scheme\":\"https\"}",
"Connection": "Keep-Alive",
"Host": "pie.dev",
"Origin": "https://pie.dev",
"Priority": "u=3, i",
"Referer": "https://pie.dev/",
"Sec-Fetch-Dest": "empty",
"Sec-Fetch-Mode": "cors",
"Sec-Fetch-Site": "same-origin",
"User-Agent":
"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/605.1.15
Headers
Text
Hex
Compressed
JavaScript
JSON
JSON Text
Raw
Recording started
Recording

httpbin.org

- Use httpbin.org, pie.dev, or use the Docker container
- From a CLI prompt, run:

```
docker run -p 8000:80 kennethreitz/httpbin
```


localhost:8000
httpbin.org
0.9.2
[Base URL: localhost:8000/]
A simple HTTP Request & Response Service.
Run locally: $ docker run -p 80:80 kennethreitz/httpbin
the developer - Website
Send email to the developer
Schemes
HTTP
HTTP Methods Testing different HTTP verbs
Auth Auth methods
Status codes Generates responses with given status code
Request inspection Inspect the request data
Response inspection Inspect the response data like caching and headers

HTTP

HTTPbis

HTTP Working Group

- *bis* is a Latin adverb meaning “twice”
- WG chartered in October 2007 to refine and clarify HTTP
- Specifically forbidden from creating a new version of HTTP

“The Working Group **must not introduce a new version of HTTP** and should not add new functionality to HTTP. The WG is not tasked with producing new methods, headers, or extension mechanisms…”

—2007-10-23 HTTP*bis* charter

Whoops!

HTTP

RFC 9110: HTTP Semantics

RFC 9111: HTTP Caching

RFC 9112: HTTP/1.1

RFC 9113: HTTP/2

RFC 9114: HTTP/3

HTTP/1.1

- Plain text protocol over TCP/IP
- HTTP over TLS allows for encrypting the data over TCP/IP

HTTP/2

- Based on the SPDY protocol developed by Google
- Provides a more efficient way to send the bytes “over the wire”
- Allows for multiplexing streams of data to reduce wait time
- Enabled server push functionality

HTTP/3

- Based on the QUIC protocol developed by Google
- Greater improvements to performance and reductions in latency
- Uses UDP as the transport layer
- See Paul’s uncon talk on Friday at 1:30pm-ish, for a deeper dive

HTTP registries

HTTP Method Registry

HTTP Status Code Registry

HTTP Field Name Registry (i.e., headers)

HTTP Cache Directive Registry

HTTP Parameters

HTTP Authentication Scheme Registry

HTTP additions

RFC 5789: PATCH Method for HTTP

RFC 6585: Additional HTTP Status Codes

RFC 7240: Prefer Header for HTTP

RFC 7725: An HTTP Status Code to Report Legal Obstacles

RFC 8288: Web Linking

RFC 8297: An HTTP Status Code for Indicating Hints

For more, see httpwg.org/specs

Properties of HTTP

- A client-server architecture
- Designed to be atomic, stateless
- Cacheable messages
- Uniform interface
- Layered
- Supports code on demand

It’s **REST**ful!

HTTP methods

HTTP methods

A selection of widely-used methods

GET

POST

PUT

DELETE

PATCH

HEAD

OPTIONS

HTTP Method Registry, iana.org/assignments/http-methods

Safe methods

- Safe methods are essentially read-only
- Safe methods allow automated retrieval processes (spiders) and performance optimizations (pre-fetching) to make requests without fear of causing harm
- GET, HEAD, and OPTIONS are defined as safe methods
- Making this distinction allows user agents (i.e., browsers) to represent non-safe methods (e.g., POST, PUT, DELETE) in special ways

“This definition of safe methods does not prevent an implementation from including behavior that is potentially harmful, that is not entirely read-only, or that causes side effects while invoking a safe method. What is important, however, is that the client did not request that additional behavior and cannot be held accountable for it.”

—RFC 9110, 9.2.1. Safe Methods

A cautionary tale

Google Web Accelerator

```
<a href="https://example.org/books?id=12&action=delete">
  Delete book
</a>
```

Idempotence

- Side effects of $n > 0$ identical requests is the same as for a single request
- Idempotent methods allow automatic retries of requests, e.g., in the case of communication failures
- PUT and DELETE are idempotent
- All safe methods are inherently idempotent

“A client SHOULD NOT automatically retry a request with a non-idempotent method unless it has some means to know that the request semantics are actually idempotent, regardless of the method, or some means to detect that the original request was never applied.”

—RFC 9110, 9.2.2. Idempotent Methods

GET

- Used to retrieve information
- “Transfer a current representation of the target resource”
- Safe & idempotent

```
GET /user/1234 HTTP/1.1
Accept: */*
Accept-Encoding: gzip, deflate, zstd
Host: example.com
User-Agent: HTTPie/3.2.4


HTTP/1.1 200 OK
Content-Length: 81
Content-Type: application/json
Date: Wed, 22 Oct 2025 21:48:59 GMT

{
    "email": "frodo@example.com",
    "id": 1234,
    "name": "Frodo Baggins"
}
```

POST

- “Perform resource-specific processing on the request content”
- The target URL should process the the request body, according to its own rules
- Could mean append, annotate, or paste after
- Not safe or idempotent

```
POST /user HTTP/1.1
Accept: application/json, */*;q=0.5
Accept-Encoding: gzip, deflate, zstd
Content-Length: 54
Content-Type: application/json
Host: example.com
User-Agent: HTTPie/3.2.4

{
    "email": "sam@example.com",
    "name": "Samwise Gamgee"
}
```

```
HTTP/1.1 201 Created
Content-Length: 81
Content-Type: application/json
Date: Wed, 22 Oct 2025 21:53:33 GMT
Location: /user/1235
Link: </user/1235>; rel="self"

{
    "email": "sam@example.com",
    "id": 1235,
    "name": "Samwise Gamgee"
}
```

```
HTTP/1.1 201 Created
Content-Length: 81
Content-Type: application/json
Date: Wed, 22 Oct 2025 21:53:33 GMT
Location: /user/1235
Link: </user/1235>; rel="self"

{
    "email": "sam@example.com",
    "id": 1235,
    "name": "Samwise Gamgee"
}
```

PUT

- Opposite of GET
- “Replace all current representations of the target resource with the request content”
- Used to store data
- Transfers a ***full*** representation of a resource from the client to the server
- Not safe
- Idempotent

```
PUT /user/1234 HTTP/1.1
Accept: application/json, */*;q=0.5
Accept-Encoding: gzip, deflate, zstd
Content-Length: 91
Content-Type: application/json
Host: example.com
User-Agent: HTTPie/3.2.4

{
    "email": "frodo.baggins@example.com",
    "id": "1234",
    "name": "Frodo Baggins"
}
```

```
PUT /user/1234 HTTP/1.1
Accept: application/json, */*;q=0.5
Accept-Encoding: gzip, deflate, zstd
Content-Length: 91
Content-Type: application/json
Host: example.com
User-Agent: HTTPie/3.2.4

{
    "email": "frodo.baggins@example.com",
    "id": "1234",
    "name": "Frodo Baggins"
}
```

```
HTTP/1.1 200 OK
Content-Length: 91
Content-Type: application/json
Date: Wed, 22 Oct 2025 22:04:40 GMT
Link: </user/1234>; rel="self"

{
    "email": "frodo.baggins@example.com",
    "id": "1234",
    "name": "Frodo Baggins"
}
```

“The fundamental difference between the POST and PUT methods is highlighted by the different intent for the enclosed representation. The target resource in a POST request is intended to handle the enclosed representation according to the resource’s own semantics, whereas the enclosed representation in a PUT request is defined as replacing the state of the target resource.”

—RFC 9110, 9.3.4. PUT

POST /user

PUT /user/1234

DELETE

- “Remove all current representations of the target resource”
- Requests that the resource be removed from ***public*** access
- Not safe
- Idempotent

```
DELETE /user/1234 HTTP/1.1
Accept: */*
Accept-Encoding: gzip, deflate, zstd
Content-Length: 0
Host: example.com
User-Agent: HTTPie/3.2.4
```

```
HTTP/1.1 204 No Content
Content-Type: text/html; charset=utf-8
Date: Wed, 22 Oct 2025 22:14:13 GMT
```

```
DELETE /user/1234 HTTP/1.1
Accept: */*
Accept-Encoding: gzip, deflate, zstd
Content-Length: 0
Host: example.com
User-Agent: HTTPie/3.2.4
```

```
HTTP/1.1 204 No Content
Content-Type: text/html; charset=utf-8
Date: Wed, 22 Oct 2025 22:14:13 GMT
```

PATCH

- PATCH “requests that a set of changes described in the request entity be applied to the” target resource
- If the target resource doesn’t exist, the server may create it, if able
- Server and client need to agree on the semantics of the “patch” representation; see RFC 6902: JSON Patch
- Not safe or idempotent
- Defined in RFC 5789

```
PATCH /user/1235 HTTP/1.1
Accept: application/json, */*;q=0.5
Accept-Encoding: gzip, deflate, zstd
Content-Length: 112
Content-Type: application/json-patch+json
Host: example.com
If-Match: "abc123"
User-Agent: HTTPie/3.2.4

[
    {
        "op": "replace",
        "path": "/email",
        "value": "samwise.gamgee@example.com"
    }
]
```

```
PATCH /user/1235 HTTP/1.1
Accept: application/json, */*;q=0.5
Accept-Encoding: gzip, deflate, zstd
Content-Length: 112
Content-Type: application/json-patch+json
Host: example.com
If-Match: "abc123"
User-Agent: HTTPie/3.2.4

[
    {
        "op": "replace",
        "path": "/email",
        "value": "samwise.gamgee@example.com"
    }
]
```

```
HTTP/1.1 200 OK
Content-Length: 91
Content-Type: application/json
Date: Wed, 22 Oct 2025 22:19:48 GMT
Link: </user/1235>; rel="self"

{
    "email": "samwise.gamgee@example.com",
    "id": 1235,
    "name": "Samwise Gamgee"
}
```

HEAD

- “Same as GET, but do not transfer the response content”
- Returns only the headers, not the body
- Useful for getting details about a resource representation before retrieving the full representation
- Cacheable
- Safe & idempotent

```
HEAD /user/1234 HTTP/1.1
Accept: */*
Accept-Encoding: gzip, deflate, zstd
Host: example.com
User-Agent: HTTPie/3.2.4


HTTP/1.1 200 OK
Content-Length: 81
Content-Type: application/json
Date: Wed, 22 Oct 2025 22:24:42 GMT
Link: </user/1235>; rel=“self"
```

OPTIONS

- “Describe the communication options for the target resource”
- Similar to HEAD, but can return different headers and information, depending on resource capabilities
- Not cacheable
- Allows for self-documenting HTTP APIs
- Safe & idempotent

```
OPTIONS /user/1234 HTTP/1.1
Accept: */*
Accept-Encoding: gzip, deflate, zstd
Host: example.com
User-Agent: HTTPie/3.2.4


HTTP/1.1 200 OK
Allow: HEAD, PUT, OPTIONS, DELETE, GET, PATCH
Content-Length: 0
Content-Type: text/html; charset=utf-8
Date: Wed, 22 Oct 2025 22:28:48 GMT
```

OPTIONS

Limitations to beware

- Can’t directly link to this metadata
- Responses aren’t cacheable because HTTP caches operate on representations of resources, and OPTIONS has no representations
- “OPTIONS is ‘chatty’” — the client make pre-flight requests for capabilities
- Server support is not universal
- Maybe use Well-Known URIs or Web Linking instead
- For more info, see BCP 56 (Building Protocols with HTTP)

Status codes

Status codes

1xx Informational

2xx Successful

3xx Redirection

4xx Client error

5xx Server error

200 OK

- Standard success response, includes the representation of the resource returned
- Widely used, widely known

201 Created

- You asked the server to create a resource, and it created it
- If it wasn't created at the target URL, the response might include a `Location` header that points to the URL of the created resource

```
HTTP/1.1 201 Created
Content-Length: 81
Content-Type: application/json
Date: Wed, 22 Oct 2025 21:53:33 GMT
Location: /user/1235
Link: </user/1235>; rel="self"

{
    "email": "sam@example.com",
    "id": 1235,
    "name": "Samwise Gamgee"
}
```

202 Accepted

- You asked the server to do something
- There were no problems with your request, i.e., everything validated
- But the server hasn’t necessarily performed the request yet
- Should include a way to monitor the status of the request (e.g., web link to status URL)

```
POST /batch HTTP/1.1
Accept: application/json, */*;q=0.5
Accept-Encoding: gzip, deflate, zstd
Content-Length: 33
Content-Type: application/json
Host: example.com
User-Agent: HTTPie/3.2.4

{
    "data": "data to batch process"
}
```

```
HTTP/1.1 202 Accepted
Content-Length: 76
Content-Type: application/hal+json
Date: Wed, 22 Oct 2025 22:47:54 GMT
Link: </batch/status/1234>; rel="status"

{
    "_links": {
        "status": { "href": "/batch/status/1234" }
    }
}
```

204 No Content

- You asked the server to do something
- The server did the thing and is letting you know it was successful
- Helpful for indicating success without needing to transfer a full resource
- Often used in response to `DELETE` requests

```
DELETE /user/1234 HTTP/1.1
Accept: */*
Accept-Encoding: gzip, deflate, zstd
Content-Length: 0
Host: example.com
User-Agent: HTTPie/3.2.4
```

```
HTTP/1.1 204 No Content
Content-Type: text/html; charset=utf-8
Date: Wed, 22 Oct 2025 22:14:13 GMT
```

301 Moved Permanently

- The resource requested has moved, and you should make this request to the new location, and also *remember* the new location
- Responses should include a Location header, telling the client where to send the new request
- Originally intended to use the same method on the new location (e.g., if it was POST, send the same POST to the new location), but most clients use GET
- To force the use of the same method on the new location, use 308 Permanent Redirect

```
GET /user/1234 HTTP/1.1
Accept: */*
Accept-Encoding: gzip, deflate, zstd
Host: example.com
User-Agent: HTTPie/3.2.4


HTTP/1.1 301 Moved Permanently
Content-Length: 0
Content-Type: text/html; charset=utf-8
Date: Wed, 22 Oct 2025 23:10:50 GMT
Location: https://api2.example.com/user/1234
```

302 Found

- The resource requested is elsewhere right now, and you should make this request to the new location, but don’t remember the new location
- Responses should include a Location header, telling the client where to send the new request
- As with 301, this was originally intended to use the same method on the new location (e.g., if it was POST, send the same POST to the new location), but most clients use GET
- To force the use of the same method on the new location, use 307 Temporary Redirect

307 Temporary Redirect

- Similar to 302, the resource requested is elsewhere right now, and you should make this request to the new location, but don’t remember the new location
- Responses should include a Location header, telling the client where to send the new request
- The client must not change the method if it automatically redirects to the new location

308 Permanent Redirect

- Similar to 301, the resource requested has moved, and you should make this request to the new location, and also *remember* the new location
- Responses should include a Location header, telling the client where to send the new request
- The client must not change the method if it automatically redirects to the new location

400 Bad Request

- The server cannot or will not process the request because of a client error, e.g., malformed syntax, validation failed, invalid headers, etc.
- Often used as a general “failed validation” error response for APIs

```
POST /user HTTP/1.1
Accept: application/json, */*;q=0.5
Accept-Encoding: gzip, deflate, zstd
Content-Length: 52
Content-Type: application/json
Host: example.com
User-Agent: HTTPie/3.2.4

{
    "email": "sam",
    "name": "Samwise Gamgee"
}
```

```
POST /user HTTP/1.1
Accept: application/json, */*;q=0.5
Accept-Encoding: gzip, deflate, zstd
Content-Length: 52
Content-Type: application/json
Host: example.com
User-Agent: HTTPie/3.2.4

{
    "email": "sam",
    "name": "Samwise Gamgee"
}
```

```
HTTP/1.1 400 Bad Request
Content-Length: 241
Content-Type: application/problem+json
Date: Wed, 22 Oct 2025 21:53:33 GMT

{
    "type": "https://example.com/errors#validation-error",
    "title": "Your request is not valid.",
    "errors": [
        {
            "detail": "email must be a valid email address",
            "pointer": "#/email"
        }
    ]
}
```

401 Unauthorized

- The request lacks valid authentication credentials for the target URL
- The response must include a WWW-Authenticate header with a “challenge” that may be used for the target URL (e.g., Basic, Digest, Bearer, etc.)
- The client may repeat the request with a different Authorization header value

```
GET /user/1234 HTTP/1.1
Accept: */*
Accept-Encoding: gzip, deflate, zstd
Host: example.com
User-Agent: HTTPie/3.2.4


HTTP/1.1 401 Unauthorized
Content-Length: 167
Content-Type: application/problem+json
WWW-Authenticate: Bearer realm="example.com" scope="users.profile"
Date: Wed, 22 Oct 2025 21:53:33 GMT

{
    "type": "https://example.com/errors#authentication",
    "title": "Authorization required.",
    "detail": "Please send the request with a valid Bearer token."
}
```

403 Forbidden

- The server refuses to fulfill the request
- If credentials were provided, they are not sufficient to allow access
- The client should not try to repeat the request with the same credentials
- Beware: this could expose information about your system; contrast with 404

```
GET /user/1234 HTTP/1.1
Accept: */*
Accept-Encoding: gzip, deflate, zstd
Authorization: Bearer 3858f62230ac3c915f300c664312c63f
Host: example.com
User-Agent: HTTPie/3.2.4


HTTP/1.1 403 Forbidden
Content-Length: 0
Content-Type: application/problem+json
Date: Thu, 23 Oct 2025 01:48:58 GMT

{
    "type": "https://example.com/errors#permission",
    "title": "Permission denied.",
    "detail": "You do not have permission to access this resource."
}
```

404 Not Found

- Probably the most familiar status code
- The server could not find a representation for the requested URL, or **it does not wish to disclose that one exists**

```
GET /user/1234 HTTP/1.1
Accept: */*
Accept-Encoding: gzip, deflate, zstd
Authorization: Bearer 3858f62230ac3c915f300c664312c63f
Host: example.com
User-Agent: HTTPie/3.2.4


HTTP/1.1 404 Not Found
Content-Length: 137
Content-Type: application/problem+json
Date: Thu, 23 Oct 2025 01:48:58 GMT

{
    "type": "https://example.com/errors#not-found",
    "title": "Not found.",
    "detail": "The requested resource does not exist."
}
```

405 Method Not Allowed

- The target URL doesn’t allow using the requested method
- The server must include an `Allow` header in the response

```
PUT /user HTTP/1.1
Accept: application/json, */*;q=0.5
Accept-Encoding: gzip, deflate, zstd
Content-Length: 71
Content-Type: application/json
Host: example.com
User-Agent: HTTPie/3.2.4

{
    "email": "merry@example.com",
    "name": "Meriadoc Brandybuck"
}
```

```
HTTP/1.1 405 Method Not Allowed
Allow: GET, HEAD, POST, OPTIONS
Content-Length: 152
Content-Type: application/problem+json
Date: Thu, 23 Oct 2025 01:48:58 GMT

{
    "type": "https://example.com/errors#not-allowed",
    "title": "Method not allowed.",
    "detail": "This resource does not allow PUT requests."
}
```

409 Conflict

- The server couldn’t complete the request because there is a conflict with the current state of the resource
- Often means something else has modified the resource
- Useful as a response to PUT and PATCH requests (and maybe DELETE)
- Especially useful when the client sends `If-Match` or `If-Unmodified-Since` headers with PUT, PATCH, or DELETE

```
GET /user/1236 HTTP/1.1
Accept: */*
Accept-Encoding: gzip, deflate, zstd
Host: example.com
User-Agent: HTTPie/3.2.4


HTTP/1.1 200 OK
Content-Length: 87
Content-Type: application/json
Date: Wed, 12 Sep 2025 16:23:51 GMT

{
    "email": "merry@example.com",
    "id": 1236,
    "name": "Meriadoc Brandybuck"
}
```

```
PUT /user/1236 HTTP/1.1
Accept: application/json, */*;q=0.5
Accept-Encoding: gzip, deflate, zstd
Content-Length: 101
Content-Type: application/json
Host: example.com
If-Unmodified-Since: Wed, 12 Sep 2025 16:23:51 GMT
User-Agent: HTTPie/3.2.4

{
    "email": "meriadoc.brandybuck@example.com",
    "id": 1236,
    "name": "Meriadoc Brandybuck"
}
```

```
PUT /user/1236 HTTP/1.1
Accept: application/json, */*;q=0.5
Accept-Encoding: gzip, deflate, zstd
Content-Length: 101
Content-Type: application/json
Host: example.com
If-Unmodified-Since: Wed, 12 Sep 2025 16:23:51 GMT
User-Agent: HTTPie/3.2.4

{
    "email": "meriadoc.brandybuck@example.com",
    "id": 1236,
    "name": "Meriadoc Brandybuck"
}
```

```
HTTP/1.1 409 Conflict
Content-Length: 158
Content-Type: application/problem+json
Date: Thu, 23 Oct 2025 01:48:58 GMT

{
    "type": "https://example.com/errors#conflict",
    "title": "Encountered a conflict.",
    "detail": "The resource changed since If-Unmodified-Since."
}
```

418 I’m a teapot

- Response when attempting to brew coffee with a teapot
- “The resulting entity body MAY be short and stout.”
- RFC 2324: Hyper Text Coffee Pot Control Protocol
- 1998 April Fool’s joke

419 ???

- Laravel responds with this status code when encountering a `TokenMismatchException`; it means the CSRF token is invalid
- Listed as “unassigned” in the IANA status code registry

429 Too Many Requests

- Response when the client is being “rate-limited”
- The definition of the rates and limits is up to you
- Nice to include a `Retry-After` header to let the client know when it may retry the request
- Defined in RFC 6585: Additional HTTP Status Codes

```
HTTP/1.1 429 Too Many Requests
Content-Length: 166
Content-Type: application/problem+json
Date: Thu, 23 Oct 2025 01:48:58 GMT
Retry-After: 3600

{
    "type": "https://example.com/errors#rate-limit",
    "title": "Too many requests.",
    "detail": "You are making too many requests. Please retry later.”
}
```

451 Unavailable For Legal Reasons

- “the server is denying access to the resource as a consequence of a legal demand”
- The server might not be an origin server; it could be an ISP or search engine that is being coerced to deny access.
- The response should include a link with the “blocked-by” relation to identify the blocking entity (not the entity mandating the block)
- Note the number used for this status code

```
HTTP/1.1 451 Unavailable For Legal Reasons
Content-Length: 354
Content-Type: text/html; charset=utf-8
Date: Thu, 23 Oct 2025 01:48:58 GMT
Link: <https://spqr.example.org/legislatione>; rel="blocked-by"

<html>
 <head><title>Unavailable For Legal Reasons</title></head>
 <body>
  <h1>Unavailable For Legal Reasons</h1>
  <p>This request may not be serviced in the Roman Province
  of Judea due to the Lex Julia Majestatis, which disallows
  access to resources hosted on servers deemed to be
  operated by the People's Front of Judea.</p>
 </body>
</html>
```

500 Internal Server Error

- Indicates that a problem occurred on the server that prevented it from fulfilling the request, often an exception or fatal error
- If an exception is thrown that isn’t caught/handled, then you’re probably responding with a 500
- These usually indicate a bug in your code that you should fix
- You want these to be rare; watch your logs and clean these up

503 Service Unavailable

- Used when the server is unavailable, for a variety of reasons
  - The service is inundated with requests and cannot respond
  - The service is down for maintenance
  - You went on vacation and “paused” your service
- This is considered a temporary condition
- You may use a `Retry-After` header to indicate when the client should attempt retrying the request

Advanced topics

Advanced topics

- Content negotiation
- Caching
- Conditional requests
- Range requests
- Web linking
- Server push

Content negotiation

Server-driven

Content negotiation

- The client may send headers to help the server guess: `Accept`, `Accept-Language`, `Accept-Encoding`, `Accept-Charset`
- The server can use other factors
- It’s the server’s best guess, so the response could be different on subsequent identical requests

```
GET /user/1234 HTTP/1.1
Accept: application/json
Accept-Charset: utf-8
Accept-Encoding: gzip, deflate, zstd
Accept-Language: en-us, en-gb;q=0.8, en;q=0.7
Connection: keep-alive
Host: example.com
User-Agent: HTTPie/3.2.4
```

```
HTTP/1.1 200 OK
Content-Encoding: gzip
Content-Language: en-us
Content-Length: 81
Content-Type: application/json
Date: Wed, 22 Oct 2025 21:48:59 GMT
Vary: Accept, Accept-Charset, Accept-Language, Accept-Encoding

{
    "email": "frodo@example.com",
    "id": 1234,
    "name": "Frodo Baggins"
}
```

```
HTTP/1.1 200 OK
Content-Encoding: gzip
Content-Language: en-us
Content-Length: 81
Content-Type: application/json
Date: Wed, 22 Oct 2025 21:48:59 GMT
Vary: Accept, Accept-Charset, Accept-Language, Accept-Encoding

{
    "email": "frodo@example.com",
    "id": 1234,
    "name": "Frodo Baggins"
}
```

Agent-driven
Content negotiation

- Requires multiple requests from the client, sometimes
- First request results in a response listing available representations either in the headers or in the entity body
- Second request is either automatic (client chooses) or manual (user chooses) for the desired representation

```
GET /user/1234 HTTP/1.1
Accept: */*
Host: example.com
User-Agent: HTTPie/3.2.4
```

```
HTTP/1.1 300 Multiple Choices
Date: Mon, 30 Jul 2012 02:57:42 GMT
Content-Length: 463
Content-Type: application/hal+json
Link: </user/1234.html>; rel="alternate"; type="text/html"
Link: </user/1234.json>; rel="alternate"; type="application/hal+json"
Link: </user/1234.xml>; rel="alternate"; type="application/hal+xml"
Link: </user/1234>; rel="self"

{
    "_links": {
        "alternate": [
            {
                "href": "/user/1234.html",
                "type": "text/html"
            },
            {
                "href": "/user/1234.json",
                "type": "application/hal+json"
            },
            {
                "href": "/user/1234.xml",
                "type": "application/hal+xml"
            }
        ],
        "self": {
            "href": "/user/1234"
        }
    }
}
```

```
HTTP/1.1 300 Multiple Choices
Date: Mon, 30 Jul 2012 02:57:42 GMT
Content-Length: 463
Content-Type: application/hal+json
Link: </user/1234.html>; rel="alternate"; type="text/html"
Link: </user/1234.json>; rel="alternate"; type="application/hal+json"
Link: </user/1234.xml>; rel="alternate"; type="application/hal+xml"
Link: </user/1234>; rel="self"

…
```

```
{
    "_links": {
        "alternate": [
            {
                "href": "/user/1234.html",
                "type": "text/html"
            },
            {
                "href": "/user/1234.json",
                "type": "application/hal+json"
            },
            {
                "href": "/user/1234.xml",
                "type": "application/hal+xml"
            }
        ],
        "self": {
            "href": "/user/1234"
        }
    }
}
```

Caching

Caching
Headers

- Expires
- Cache-Control

Caching

Properties

- max-age
- s-maxage
- public
- private
- no-cache
- no-store
- must-revalidate
- proxy-revalidate

```
HTTP/1.1 200 OK
Cache-Control: max-age=86400, must-revalidate
Content-Length: 81
Content-Type: application/json
Date: Wed, 22 Oct 2025 21:48:59 GMT

{
    "email": "frodo@example.com",
    "id": 1234,
    "name": "Frodo Baggins"
}
```

```
HTTP/1.1 200 OK
Cache-Control: max-age=86400, must-revalidate
Content-Length: 81
Content-Type: application/json
Date: Wed, 22 Oct 2025 21:48:59 GMT

{
    "email": "frodo@example.com",
    "id": 1234,
    "name": "Frodo Baggins"
}
```

Conditional requests

Conditional requests

- If-Modified-Since
- If-Unmodified-Since
- If-Match
- If-None-Match
- If-Range

```
GET /user/1234 HTTP/1.1
Accept: */*
Accept-Encoding: gzip, deflate, zstd
Host: example.com
If-Modified-Since: Wed, 22 Oct 2025 21:48:59 GMT
User-Agent: HTTPie/3.2.4


HTTP/1.1 304 Not Modified
Date: Thu, 23 Oct 2025 06:54:35 GMT
```

Range requests

Range requests

- Used when requests are made for ranges of bytes from a resource
- Determine whether a server supports range requests by checking for the `Accept-Ranges` header with HEAD or OPTIONS

```
HEAD /range/100000 HTTP/1.1
Accept: */*
Accept-Encoding: gzip, deflate, zstd
Host: httpbin.org
User-Agent: HTTPie/3.2.4
```

```
HTTP/1.1 200 OK
Accept-Ranges: bytes
Content-Length: 100000
Content-Range: bytes 0-99999/100000
Content-Type: application/octet-stream
Date: Thu, 23 Oct 2025 07:06:59 GMT
```

```
HEAD /range/100000 HTTP/1.1
Accept: */*
Accept-Encoding: gzip, deflate, zstd
Host: httpbin.org
User-Agent: HTTPie/3.2.4


HTTP/1.1 200 OK
Accept-Ranges: bytes
Content-Length: 100000
Content-Range: bytes 0-99999/100000
Content-Type: application/octet-stream
Date: Thu, 23 Oct 2025 07:06:59 GMT
```

```
GET /range/100000 HTTP/1.1
Accept: */*
Accept-Encoding: gzip, deflate, zstd
Host: httpbin.org
Range: bytes=0-999
User-Agent: HTTPie/3.2.4


HTTP/1.1 206 Partial Content
Accept-Ranges: bytes
Content-Length: 1000
Content-Range: bytes 0-999/100000
Content-Type: application/octet-stream
Date: Thu, 23 Oct 2025 07:12:57 GMT

...
```

```
GET /range/100000 HTTP/1.1
Accept: */*
Accept-Encoding: gzip, deflate, zstd
Host: httpbin.org
Range: bytes=0-999
User-Agent: HTTPie/3.2.4


HTTP/1.1 206 Partial Content
Accept-Ranges: bytes
Content-Length: 1000
Content-Range: bytes 0-999/100000
Content-Type: application/octet-stream
Date: Thu, 23 Oct 2025 07:12:57 GMT

...
```

Web linking

Web linking

- Defined a framework for typed links not specific to an application, and introduced the `Link` header
- We've already seen many examples
- You should use this!
- RFC 8288: Web Linking
- Link Relations registry

```
HTTP/1.1 200 OK
Content-Length: 91
Content-Type: application/json
Date: Wed, 22 Oct 2025 22:04:40 GMT
Link: </user/1234>; rel="self"; type="application/json"
Link: </user/1234.html>; rel="alternate"; type="text/html"
Link: </user>; rel="collection"; type="application/json"

{
    "email": "frodo.baggins@example.com",
    "id": "1234",
    "name": "Frodo Baggins"
}
```

```
HTTP/1.1 200 OK
Content-Length: 91
Content-Type: application/json
Date: Wed, 22 Oct 2025 22:04:40 GMT
Link: </user/1234>; rel="self"; type="application/json"
Link: </user/1234.html>; rel="alternate"; type="text/html"
Link: </user>; rel="collection"; type="application/json"

{
    "email": "frodo.baggins@example.com",
    "id": "1234",
    "name": "Frodo Baggins"
}
```

Server push

Server push
preload

- Link relation defined in w3.org/TR/preload
- Use it in the `Link` header to specify resources that should be loaded early
- HTTP/2 servers can use this information to proactively send resources to the client before they are requested

```
HTTP/1.1 200 OK
Content-Length: 123456
Content-Type: text/html; charset=utf-8
Date: Wed, 22 Oct 2025 22:04:40 GMT
Link: </css/bootstrap-theme.css>; rel=preload; as=style
Link: </css/bootstrap.css>; rel=preload; as=style
Link: </css/theme.css>; rel=preload; as=style
Link: </js/bootstrap.js>; rel=preload; as=script
Link: </js/jquery.js>; rel=preload; as=script
Link: </fonts/glyphicons-halflings-regular.eot>; rel=preload; as=font
Link: </fonts/glyphicons-halflings-regular.svg>; rel=preload; as=font
Link: </fonts/glyphicons-halflings-regular.ttf>; rel=preload; as=font
Link: </fonts/glyphicons-halflings-regular.woff>; rel=preload; as=font
Link: </fonts/glyphicons-halflings-regular.woff2>; rel=preload; as=font

<html>
    ...
</html>
```

Whew! That was a lot of info.

Thank you!

Keep in touch

- ben.ramsey.dev
- phpc.social/@ramsey
- github.com/ramsey
- speakerdeck.com/ramsey
- www.linkedin.com/in/benramsey
- ben@ramsey.dev