HTTPie Friendly HTTP client

START HERE

`http <url>`
GET, pretty-printed

`http -d <url>`
Download a file

`http POST <url> name=Ann`
POST a JSON body

`http :3000/api`
Shortcut for localhost:3000

`http <url> X-Key:abc`
Send a header

`http <url> page==2`
Add a query parameter

`http -f POST <url> a=1`
Send a form instead

`http -v <url>`
Show request and response

Items headers, query, data → *build* → **Request** JSON by default → *send* → **Response** formatted, coloured → *print* → **Output** -p, -b, -h, -o

CONCEPTS

HTTPie sends JSON by default. The separator in each item decides its role: : header, == query, = string field, := raw JSON, @ file. The method is optional: GET without data, POST with.

Request item separators

Item	Becomes
`Name:value`	Header
`q==value`	Query parameter
`k=value`	String field
`k:=42`	Raw JSON (number, bool, list)
`k=@file`	Field from text file
`k@file`	File upload (with -f)

When stdout is redirected, colours and formatting turn off. In scripts and cron add --ignore-stdin so HTTPie does not wait on stdin.

REQUESTS

`http <url>`
GET a URL

`http POST <url> <key>=<value>`
POST a JSON object

`http PUT <url> <key>=<value>`
Use any other method

`http DELETE <url>`
Send a DELETE

REQUESTS (cont.)

`https <url>`
Default to https://

`http :3000/path`
Request localhost:3000

`http <url> <key>:=<json>`
Raw JSON value
Use := for numbers, booleans, arrays and objects, e.g. tags:='["a","b"]' or n:=5.

`http <url> < <file>`
Send a file as the body

`http HEAD <url>`
Headers only

HEADERS AND QUERY

`http <url> <Name>:<value>`
Set a header

`http <url> <Name>:`
Unset a default header
Name: with nothing after the colon drops a header HTTPie would send (e.g. User-Agent:); Name; sends it empty.

`http <url> <key>==<value>`
Query parameter

`http --follow <url>`
Follow redirects

FORMS AND FILES

`http -f POST <url> <key>=<value>`
URL-encoded form

`http -f POST <url> <key>@<file>`
Upload a file (multipart)

OUTPUT

`http -v <url>`
Request and response

`http -h <url>`
Response headers only

`http -b <url>`
Response body only

`http -p <what> <url>`
Choose parts: H B h b m
H request headers, B request body, h response headers, b response body, m metadata.

`http --offline POST <url> <items>`
Print request, send nothing

`http --pretty none <url>`
No colours or formatting

AUTH

`http -a <user>:<pass> <url>`
Basic auth

AUTH (cont.)

`http -a <user> <url>`
Prompt for password

`http -A bearer -a <token> <url>`
Bearer token

`http -A digest -a <user>:<pass> <url>`
Digest auth

`http --verify no <url>`
Skip TLS verification

`http --cert <file> --cert-key <key> <url>`
Client certificate

SESSIONS

`http --session <name> -a <u>:<p> <url>`
Create or reuse a session

`http --session <name> <url>`
Replay stored auth/cookies

Sessions store headers (except Content-* and If-*), auth and cookies per host in plain text under ~/.config/httpie/sessions/.

DOWNLOADS

`http -d <url>`
wget-style download

DOWNLOADS (cont.)

`http -d <url> -o <file>`
Choose the filename

`http -dco <file> <url>`
Resume a partial download

`http <url> > <file>`
Save body via redirect

Download mode implies --follow and --check-status, and shows progress on stderr.

SCRIPTING AND NETWORK

`http --check-status <url>`
Non-zero exit on 3xx-5xx
Exit codes: 3 for 3xx, 4 for 4xx, 5 for 5xx, 6 when redirects are exhausted, 1 for other errors.

`http --ignore-stdin <url>`
Do not read stdin

`http --timeout <sec> <url>`
Connection timeout

`http --proxy http:<proxy-url> <url>`
Use a proxy

`http --debug <url>`
Debug info and config dir

`http --help`
List all options

`http --version`
Show version

CONFIG

`~/.config/httpie/config.json`
Default config file

config.json defaults

```
{
  "default_options": ["--style=fruity", "--pretty=all"],
  "disable_update_warnings": true
}
```

RECIPES

POST JSON with a token

```
http --check-status POST api.example.com/items \
  Authorization:"Bearer $TOKEN" \
  name=widget price:=9.5 tags:='["a","b"]'
```

Preview a request first

```
http --offline POST api.example.com/items name=widget
# then drop --offline to send it
```

Log in once, reuse the session

```
http --session=work -a ann https://api.example.com/login
http --session=work https://api.example.com/me
```

Pipe the body into jq

```
http -b --ignore-stdin api.example.com/items | jq '.[].id'
```