curl Transfer data by URL

START HERE

`curl <url>`
GET a URL, print body

`curl -d <data> <url>`
POST form data

`curl -o <file> <url>`
Save to a named file

`curl -H "<header>" <url>`
Add a request header

`curl -L <url>`
Follow redirects

`curl -fsS <url>`
Quiet, but fail loudly

`curl -i <url>`
Show response headers too

`curl -v <url>`
Show the whole exchange

URL scheme, host, path → `connect` → **Request** method, headers, body → `send` → **Response** status, headers, body → `output` → **Output** stdout, -o, -O

CONCEPTS

curl prints the response body to stdout and progress to stderr, so it composes with pipes. It does not fail on HTTP 404/500 unless you add -f.

Quote URLs containing & or ? so the shell does not split them.

REQUESTS

`curl <url>`
GET a URL

`curl -X <method> <url>`
Set the request method
For POST with -d the method is already POST; -X is for DELETE, PUT, PATCH.

`curl -d <data> <url>`
POST form-encoded data
Strips newlines when read with @file; use --data-binary @file to keep them.

`curl -d @<file> <url>`
POST data from a file

`curl --json '<json>' <url>`
POST JSON with proper headers
Sets Content-Type and Accept to application/json (curl 7.82+).

`curl -F <name>=<value> <url>`
Multipart form field

`curl -F <name>=@<file> <url>`
Upload a file as form field

`curl -G -d <k=v> <url>`
Send data as query string

`curl --url-query <k=v> <url>`
Add an encoded query param

`curl --data-urlencode <k=v> <url>`
URL-encode a POST value

`curl -T <file> <url>`
Upload with PUT

`curl -I <url>`
HEAD request, headers only

`curl --data-binary @<file> <url>`
POST file exactly as is

`curl -X PATCH -d <data> <url>`
Send a PATCH

HEADERS & AUTH

`curl -H "<name>: <value>" <url>`
Add a custom header

`curl -H "<name>:" <url>`
Remove a default header

`curl -u <user>:<pass> <url>`
Basic authentication
Omit :<pass> to be prompted and keep it out of shell history.

`curl -H "Authorization: Bearer <token>" <url>`
Send a bearer token

`curl --oauth2-bearer <token> <url>`
Bearer token shortcut

`curl -A "<agent>" <url>`
Set the User-Agent

`curl -e <referer> <url>`
Set the Referer

`curl -b "<name>=<value>" <url>`
Send a cookie

`curl -b <jar> -c <jar> <url>`
Read and write a cookie file

`curl -n <url>`
Credentials from ~/.netrc

`curl --aws-sigv4 <spec> -u <id>:<key> <url>`
Sign for AWS

OUTPUT & DOWNLOAD

`curl -o <file> <url>`
Save body to a file

`curl -O <url>`
Save under the remote name

`curl -L <url>`
Follow redirects

`curl -i <url>`
Include response headers

`curl -s <url>`
Silent: no progress meter

`curl -sS <url>`
Silent, but show errors

`curl -f <url>`
Fail on HTTP errors (22)
Exit 22 on 400+; the body is discarded. --fail-with-body keeps it.

`curl --fail-with-body <url>`
Fail on errors, keep body

`curl -C - -O <url>`
Resume a partial download

OUTPUT & DOWNLOAD (cont.)

`curl --create-dirs -o <path> <url>`
Make output dirs as needed

`curl -D <file> <url>`
Dump headers to a file

`curl -w "<format>" <url>`
Print info after transfer
Example: -w '%{http_code}\n' -o /dev/null -s prints just the status.

`curl --compressed <url>`
Ask for and decode gzip/br

`curl -z <file> -o <file> <url>`
Download only if newer

`curl --limit-rate <speed> <url>`
Throttle bandwidth

`curl --etag-save <file> <url>`
Save the ETag

NETWORK & TLS

`curl -k <url>`
Skip certificate checks
Insecure; only for testing. Prefer --cacert.

`curl --cacert <file> <url>`
Trust a specific CA

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

`curl -x <proxy> <url>`
Use a proxy

`curl --noproxy "<hosts>" <url>`
Bypass proxy for hosts

`curl --resolve <host>:<port>:<ip> <url>`
Pin a hostname to an IP

`curl --connect-to <h1:p1:h2:p2> <url>`
Redirect connection target

`curl --connect-timeout <s> <url>`
Limit connect time

`curl -m <s> <url>`
Limit total time

`curl --retry <n> <url>`
Retry transient failures

`curl --http2 <url>`
Prefer HTTP/2

`curl --http3 <url>`
Try HTTP/3

`curl -4 / -6 <url>`
Force IPv4 or IPv6

NETWORK & TLS (cont.)

`curl --interface <name> <url>`
Use a network interface

DEBUGGING

`curl -v <url>`
Verbose request and response

`curl --trace-ascii <file> <url>`
Full trace, readable

`curl --trace-time -v <url>`
Timestamp verbose lines

`curl -w "%{time_total}\n" -o /dev/null -s <url>`
Time a request

`curl -sI <url>`
Check headers quietly

`curl -K <file>`
Read options from a config

`curl --help all`
List every option

`curl -Z <url1> <url2>`
Transfer in parallel

`curl "<url>[1-5].txt"`
URL globbing ranges

`curl "<url>/{a,b}.txt" -o "#1.txt"`
Glob list, name by match

`curl -g <url>`
Disable globbing in URL

`curl --variable <name>=<value> --expand-url "<url>/{{name}}"`
Use variables in URL

RECIPES

POST JSON and check status

```
curl -sS --fail-with-body \
  --json '{"name":"x"}' \
  -H "Authorization: Bearer $TOKEN" \
  https://api.example.com/items
```

Pretty-print an API response

```
curl -sS https://api.example.com/items | jq .
```

Robust download

```
curl -fL --retry 3 --connect-timeout 10 \
  -C - -o file.tgz https://example.com/file.tgz
```

Print only the HTTP status

```
curl -s -o /dev/null -w "%{http_code}\n" https://example.com
```

Test a vhost before DNS

```
# send the request to a specific IP
curl --resolve example.com:443:203.0.113.10 https://example.com/
```