```
** BASIC REQUESTS **

# Simple GET request
curl https://example.com

# GET with verbose output
curl -v https://example.com

# GET and save output to file
curl -o output.html https://example.com

# GET and save with remote filename
curl -O https://example.com/file.tar.gz

# Follow redirects
curl -L https://example.com

# Silent mode (no progress)
curl -s https://example.com

# Show only HTTP response code
curl -s -o /dev/null -w "%{http_code}" https://example.com

** POST REQUESTS **

# POST with form data
curl -X POST -d "user=admin&pass=1234" https://example.com/login

# POST with JSON body
curl -X POST -H "Content-Type: application/json" -d '{"key":"value"}'
https://example.com/api

# POST a file
curl -X POST -F "file=@/path/to/file.txt" https://example.com/upload

# POST with raw body from file
curl -X POST -d @payload.json https://example.com/api

** HEADERS **

# Set custom header
curl -H "Authorization: Bearer TOKEN" https://example.com/api

# Set multiple headers
curl -H "Accept: application/json" -H "X-Custom: value" https://example.com

# Show response headers only
curl -I https://example.com

# Show both request and response headers
curl -v -s https://example.com 2>&1 | grep "^[<>]"

** AUTHENTICATION **

# Basic auth
curl -u username:password https://example.com/api

# Bearer token
curl -H "Authorization: Bearer eyJhbGci..." https://example.com/api

# API key in header
curl -H "X-API-Key: myapikey" https://example.com/api

# API key in URL param
```

```
curl "https://example.com/api?api_key=myapikey"

** HTTPS / TLS DEBUGGING **

# Skip SSL certificate verification (insecure)
curl -k https://example.com

# Specify CA certificate
curl --cacert /path/to/ca.crt https://example.com

# Use client certificate
curl --cert /path/to/client.crt --key /path/to/client.key https://example.com

# Show SSL certificate info
curl -vI https://example.com 2>&1 | grep -A5 "Server certificate"

# Force TLS version
curl --tlsv1.2 https://example.com
curl --tlsv1.3 https://example.com

# Show full TLS handshake
curl -v --tlsv1.2 https://example.com 2>&1 | grep -E "SSL|TLS|issuer|subject"

** HTTP METHODS **

# PUT request
curl -X PUT -H "Content-Type: application/json" -d '{"key":"value"}'
https://example.com/api/1

# PATCH request
curl -X PATCH -H "Content-Type: application/json" -d '{"key":"new"}'
https://example.com/api/1

# DELETE request
curl -X DELETE https://example.com/api/1

# OPTIONS request
curl -X OPTIONS https://example.com/api

# HEAD request
curl -X HEAD https://example.com

** DEBUGGING RESPONSES **

# Show response time breakdown
curl -s -o /dev/null -w
"dns: %{time_namelo
okup}s\nconnect: %{tim
e_connect}s\ntls
: %{time_appconnect}s\nttfb: %{time_starttransfe
r}s\ntotal: %{time_total}s\n" https://example.com

# Show redirect chain
curl -Ls -o /dev/null -w "%{url_effective}" https://example.com

# Follow and print each redirect
curl -L -v https://example.com 2>&1 | grep "Location:"

# Check if endpoint is up
curl -sf https://example.com && echo "UP" || echo "DOWN"

# Show response headers and body
curl -i https://example.com
```

```
** COOKIES **

# Send cookie
curl -b "session=abc123" https://example.com

# Save cookies to file
curl -c cookies.txt https://example.com/login

# Load cookies from file
curl -b cookies.txt https://example.com/dashboard

# Save and reuse cookies
curl -c cookies.txt -b cookies.txt https://example.com

** PROXY **

# Use HTTP proxy
curl -x http://proxy:8080 https://example.com

# Use SOCKS5 proxy
curl --socks5 proxy:1080 https://example.com

# Bypass proxy for host
curl --noproxy example.com https://example.com

** TIMEOUTS AND RETRIES **

# Set connection timeout (seconds)
curl --connect-timeout 5 https://example.com

# Set max total time
curl --max-time 10 https://example.com

# Retry on failure
curl --retry 3 https://example.com

# Retry with delay
curl --retry 3 --retry-delay 2 https://example.com

** HTTP VERSIONS **

# Force HTTP/1.1
curl --http1.1 https://example.com

# Force HTTP/2
curl --http2 https://example.com

# Force HTTP/3
curl --http3 https://example.com

# Check which HTTP version was used
curl -s -o /dev/null -w "%{http_version}" https://example.com

** COMMON DEBUGGING PATTERNS **

# Full debug of a failing API call
curl -v -s -o /dev/null https://example.com/api 2>&1

# Test health endpoint
curl -sf https://example.com/health | python3 -m json.tool

# POST JSON and pretty print response
curl -s -X POST -H "Content-Type: application/json" -d '{}'
https://example.com/api | python3 -m json.tool
```

```
# Check CORS headers
curl -H "Origin: https://myapp.com" -I https://example.com/api

# Simulate browser request
curl -A "Mozilla/5.0" -H "Accept: text/html" https://example.com

# Download with progress bar
curl -# -O https://example.com/bigfile.tar.gz

** REST API CRUD **

# GET all resources
curl -s https://api.example.com/users | python3 -m json.tool

# GET single resource by ID
curl -s https://api.example.com/users/42 | python3 -m json.tool

# CREATE resource with POST
curl -s -X POST -H "Content-Type: application/json" -d
'{"name":"John","email":"john@example.com"}' https://api.example.com/users

# REPLACE resource with PUT
curl -s -X PUT -H "Content-Type: application/json" -d '{"name":"John
Updated","email":"john@example.com"}' https://api.example.com/users/42

# PARTIAL UPDATE with PATCH
curl -s -X PATCH -H "Content-Type: application/json" -d '{"name":"John
Patched"}' https://api.example.com/users/42

# DELETE resource
curl -s -X DELETE https://api.example.com/users/42

# GET with query params (filter, sort, paginate)
curl -s "https://api.example.com/users?page=1&limit=10&sort=name"

# GET nested resource
curl -s https://api.example.com/users/42/orders

** REST API HEADERS **

# Accept JSON response
curl -H "Accept: application/json" https://api.example.com/users

# Send JSON body
curl -H "Content-Type: application/json" https://api.example.com/users

# API versioning via header
curl -H "API-Version: 2" https://api.example.com/users

# Idempotency key (prevent duplicate requests)
curl -H "Idempotency-Key: uuid-1234" -X POST https://api.example.com/payments

# Request ID for tracing
curl -H "X-Request-ID: abc-123" https://api.example.com/users

# Rate limit check (read response headers)
curl -I https://api.example.com/users | grep -i "x-ratelimit"

** REST API AUTHENTICATION PATTERNS **

# OAuth2 get token
```

```
curl -X POST -d
"grant_type=client_credentials&client_id=ID&client_secret=SECRET"
https://auth.example.com/token

# OAuth2 use token
curl -H "Authorization: Bearer ACCESS_TOKEN" https://api.example.com/users

# JWT decode payload (base64)
echo "eyJhbGci..." | cut -d. -f2 | base64 -d 2>/dev/null | python3 -m
json.tool

# Refresh OAuth2 token
curl -X POST -d "grant_type=refresh_token&refresh_token=TOKEN"
https://auth.example.com/token

** REST API TESTING PATTERNS **

# Test all CRUD operations on a resource
curl -s https://api.example.com/users
curl -s -X POST -H "Content-Type: application/json" -d '{"name":"test"}'
https://api.example.com/users
curl -s -X PUT -H "Content-Type: application/json" -d '{"name":"updated"}'
https://api.example.com/users/1
curl -s -X DELETE https://api.example.com/users/1

# Check API response time
curl -s -o /dev/null -w "%{time_total}" https://api.example.com/users

# Validate JSON response
curl -s https://api.example.com/users | python3 -c "import sys,json;
json.load(sys.stdin); print('valid JSON')"

# Send bulk/batch request
curl -s -X POST -H "Content-Type: application/json" -d
'[{"name":"a"},{"name":"b"}]' https://api.example.com/users/batch

** HTTP 1XX INFORMATIONAL **

# Server is ready to accept the request body, client should proceed
100 Continue

# Protocol upgrade agreed, switching to WebSocket or HTTP/2
101 Switching Protocols

# Server received the request and is still working, no response yet
102 Processing

** HTTP 2XX SUCCESS **

# The request was received, understood and processed successfully
200 OK

# A new resource was successfully created on the server
201 Created

# The request was queued but will be handled asynchronously
202 Accepted

# Operation succeeded but the response has no body (common on DELETE)
204 No Content

# Only a portion of the resource was returned, used with Range header
206 Partial Content
```

** HTTP 3XX REDIRECTS **

The resource has a new permanent URI, update bookmarks and links
301 Moved Permanently

The resource is temporarily at a different URI, do not cache it
302 Found

After a POST, follow with a GET to the new URI given in Location
303 See Other

The client's cached version is still fresh, no need to retransfer
304 Not Modified

Same as 302 but the client must reuse the same HTTP verb and body
307 Temporary Redirect

Same as 301 but the client must reuse the same HTTP verb and body
308 Permanent Redirect

** HTTP 4XX CLIENT ERRORS **

The request syntax is malformed or the parameters are invalid
400 Bad Request

No credentials provided or the token is expired or invalid
401 Unauthorized

Credentials are valid but the user lacks permission for this action
403 Forbidden

The requested URI does not match any resource on the server
404 Not Found

The HTTP verb used is not supported on this endpoint
405 Method Not Allowed

The client took too long to send the full request
408 Request Timeout

The action cannot complete due to a state conflict, e.g. duplicate entry
409 Conflict

The resource existed but was permanently deleted and will not return
410 Gone

The request body exceeds the size limit the server will accept
413 Payload Too Large

The Content-Type header does not match what the endpoint expects
415 Unsupported Media Type

The request is well-formed but fails semantic validation rules
422 Unprocessable Entity

The client has sent too many requests and must wait before retrying
429 Too Many Requests

** HTTP 5XX SERVER ERRORS **

The server encountered an unexpected condition and could not complete the
request
500 Internal Server Error

```
# The server does not support the functionality required to fulfill the
request
501 Not Implemented

# The server acting as a gateway received an invalid response from upstream
502 Bad Gateway

# The server is temporarily unable to handle requests due to overload or
maintenance
503 Service Unavailable

# The server acting as a gateway did not receive a timely response from
upstream
504 Gateway Timeout

# The server cannot store the data needed to complete the request
507 Insufficient Storage

** HTTP ERROR DEBUGGING WITH CURL **

# Debug a 401 - check auth header sent
curl -v -H "Authorization: Bearer TOKEN" https://api.example.com/users 2>&1 |
grep -E "Authorization|401"

# Debug a 403 - check token scopes/roles
curl -v https://api.example.com/admin 2>&1 | grep -E "403|Forbidden|WWW-
Authenticate"

# Debug a 404 - verify URL and method
curl -v https://api.example.com/wrong-path 2>&1 | grep -E "404|Not Found"

# Debug a 429 - check rate limit headers
curl -I https://api.example.com/users | grep -i "retry-after\|x-ratelimit"

# Debug a 500 - get full response body
curl -s -i https://api.example.com/users | tail -20

# Debug a 502/504 - check upstream connectivity
curl -v --connect-timeout 5 https://api.example.com/users 2>&1

# Debug a 415 - fix Content-Type header
curl -X POST -H "Content-Type: application/json" -d '{}'
https://api.example.com/users

# Debug a 422 - read validation errors in body
curl -s -X POST -H "Content-Type: application/json" -d '{"bad":"data"}'
https://api.example.com/users | python3 -m json.tool
```