# Curl

La comanda Curl (https://curl.haxx.se) és un client HTTP/HTTPS de terminal, lliure i multiplataforma. Això vol dir que aquest programa serveix per realitzar peticions HTTP/S de qualsevol tipus a un servidor donat, mostrant per pantalla la resposta obtinguda.

NOTA: També es pot fer servir com a client IMAP/IMAPS, POP3/POP3S, SMTP/SMTPS, LDAP/LDAPS, RTMP/RTSP, SCP/SFTP/FTP/FTPS/TFTP o SMB/SMBS entre d'altres, però no ho veurem
NOTA: Eines similars a *curl* són *httpie* (https://httpie.io), *curlie* (https://curlie.io) o *hurl* (https://hurl.dev)

Si simplement s'executa en un terminal això *curl https://url/dun/fitxer*, es descarregarà el fitxer indicat en memòria i mostrarà a pantalla el seu contingut.

NOTA: Noteu que si no s'escriu el protocol a la URL indicada, per defecte *curl* utilitzarà HTTP i no HTTPS...i això significa, per tant, que per defecte utilitzarà la versió del protocol HTTP/1.1; si s'indica explícitament "HTTPS", llavors utilitzarà la versió HTTP/2.0.

No obstant, *curl* té moltes més possibilitats. A continuació es mostra una llista dels paràmetres més interessants que ofereix:

| |
|---|
| -*o nom* Descarregua el fitxer indicat i el guarda al disc dur amb el nom que s'especifiqui<br>-O Descarrega el fitxer indicat i el guarda al disc dur amb el nom que tingui l'original<br>-C - Continua la descàrrega des del nº de byte indicat (si és un guió, serà a partir d'on es va parar la descàrrega -fallida- anterior del mateix fitxer |
| -s Mode "silenciós" (no mostra ni les estadístiques de descàrrega ni els errors, res)<br>-sS Mode "silenciós" però mostrant els missatges d'error |
| -*v* Mode "verbós". Serveix per mostrar les capçaleres HTTP de petició de client enviades a la petició (són les línies amb el prefix ">"), a més de les capçaleres HTTP de resposta del servidor (són les línies amb el prefix "<") i informació variada (relativa a la connexió, el "handshake" TLS, etc; són les línies amb el prefix "*"). En tot cas, totes aquestes línies són escrites a "stderr"; si es volen guardar en un fitxer.txt amb més detall, caldrà indicar el paràmetre *--trace-ascii unfitxer.txt* i, per què no, *--trace-time*<br><br>-I Només mostra les capçaleres HTTP de resposta del servidor (i no descarrega el fitxer)<br>-i Mostra a pantalla tant les capçaleres HTTP de resposta com el contingut del fitxer demanat<br>-D *nom* Guarda al disc dur, en forma de fitxer amb el nom indicat, les capçaleres HTTP de resposta<br><br>NOTA: El símbol "-" en *-D* - indica que s'enviaran les capçaleres no a un fitxer sinó a la sortida estàndard<br>NOTA: No existeix cap paràmetre del *curl* per mostrar només les capçaleres HTTP de petició del client. No obstant, es pot aconseguir aquest efecte així: *curl -vso /dev/null --stderr - https://www.hola.com \| grep "^>"* (el paràmetre *--stderr* – el que fa és redireccionar la sortida d'error -on es veuen precisament totes les línies que comencen per ">", "<" i "*"- a la sortida estàndar per tal que la canonada "\|" que s'afegeix a continuació pugui operar correctament) |
| -*L* Si el servidor web retorna un codi de redirecció (3xx), el segueix automàticament |
| -*X tipus* Realitza una petició del tipus indicat (POST, PUT, PATCH, DELETE, etc). Per defecte són GET |
| -*H "nomcapçalera:valor"* Indica un valor concret per a la capçalera HTTP de petició indicada. Es poden posar múltiples paràmetres *-H* un rera l'altre |
| -*d "dades"* En el cas de peticions on s'envïin dades del client al servidor dins del cos de la petició (això passa a les peticions de tipus POST o PUT, per exemple), aquestes dades s'hauran d'especificar com a valor d'aquest paràmetre. El format concret en el qual aquestes dades poden estar escrites pot ser molt variat i és per això que calinformar del format escollit com a valor de la capçalera de client "Content-Type", indicada amb el paràmetre *-H* :<br><br>*Si les dades estan formatades en "URL-encode" tindran un aspecte com *"user=ana&pass=1234"* i el valor de la capçalera "***Content-Type***" ha de ser *"application/x-www-form-urlencoded"* (el seu valor per defecte)<br><br>*Si les dades estan formatades en "JSON" tindran un aspecte com *'{"user":"ana","pass":1234}'* i el valor de la capçalera "***Content-Type***" ha de ser *"application/json"* |

Així doncs, per enviar (per defecte via POST), per exemple, una dada en format "URL-encode" faríem:
*curl -d "user=ana" http://www.web.com/form*
Per enviar (via POST) la mateixa dada en format "JSON" faríem, en canvi:
*curl -d '{"user":"ana"}' -H "Content-Type:application/json" http://www.web.com/form*

NOTA: Si les dades es troben guardades dins d'un fitxer (amb el format pertinent), el valor del paràmetre *-d* es pot indicar així: *-d @nomfitxer*. Si les dades es reben de l'entrada estàndar (a través d'una canonada, normalment), el valor del paràmetre -d es pot indicar així: *-d @-*
NOTA: En el cas de voler enviar les dades dins el "querystring" d'una petició GET, a més d'indicar, com sempre, el paràmetre *-d* per indicar les dades en qüestió, caldrà especificar a més llavors el paràmetre *-G* perquè si no, per defecte si s'indiquen dades amb *-d*, *curl* fa una petició de tipus POST
NOTA: Altres formats comuns poden ser ***"text/xml"*** o ***"text/plain"***
NOTA: Si el paràmetre *-d* dóna error de codificació, canvieu-lo per *--data-urlencode*. The *–data-urlencode* parameter is necessary to automatically change several special character like "$", " ", "%", "_", etc to a codified equivalent symbol understood by standard HTTP URIs (amb *-d* s'assumeix que aquesta codifiació ja està feta!). More on this in https://www.w3schools.com/tags/ref_urlencode.asp D'altra banda, si no voleu treure els possibles salts de línia de les dades a enviar (*-d* ho fa), useu llavors *–data-binary*

---

*--json '{"camp1":"valor1","camp2":"valor2"}'* Equivalent a escriure el següent (però més curt):
*-d '{"camp1":"valor1","camp2":"valor2"}' -H "Content-Type:application/json"*
Igualment es pot usar la sintaxi *--json @nomfitxer* o *--json @-* explicada a les notes anteriors

---

*-F "camp1=valor1" -F "camp2=valor2"* Envia al servidor remot (via POST) les dades indicades tal com si s'haguessin enviat a través d'un formulari web que inclogués algun camp de pujada de fitxers (és a dir, especificant a la capçalera "Content-Type" el valor "multipart/form-data"; "camp1" i "camp2" representarien els noms dels camps d'aquest hipotètic formulari). En el cas de formularis que no incloguin cap camp de pujada de fitxers, la forma d'enviar les dades és equivalent al paràmetre *-d* de Curl ja conegut (amb el valor per defecte de la capçalera "Content-Type": *"application/x-www-form-urlencoded"*)

Si es vol enviar el contingut textual d'un fitxer com a valor d'un determinat camp, es pot indicar així: *-F "camp=</ruta/fitxer"* Si es vol enviar, en canvi, el fitxer en sí (com a adjunt del formulari), s'ha d'indicar així llavors: *-F "camp=@/ruta/fitxer"* (en aquest cas el fitxer s'enviaria codificat com a "multpart base-64", igual que passa amb els formularis web estàndard).

NOTA: You can also tell *curl* what uploaded file's Content-Type to use by using 'type=', in a manner similar to: *curl -F "web=@index.html;type=text/html" example.com* (if no type is specified, *curl* sets it to "application/octet-stream"). On the other hand, you can also explicitly change the name field of a file upload by setting the 'filename=' option, like this: *curl -F "file=@photo.png;filename=me.png" example.com* (the filename that is sent to the web server is changed from "photo.png" to "me.png"; the web server only sees the filename "me.png" and doesn't know the original filename was "photo.png").

---

*-T /ruta/fitxer* "Puja" el fitxer indicat al servidor remot (via PUT, sempre que el servidor estigui preparat). Si la URL indicada acaba amb "/", llavors el fitxer serà pujat amb el seu nom local; si no, s'entendrà que la darrera part de la URL és el nom del fitxer que es vol un cop pujat.

Es pot indicar "-" en comptes d'una ruta per "pujar" les dades rebudes a través de *stdin* (normalment via una canonada) o també "." (pel mateix però de forma no bloquejant). D'altra banda, es poden pujar varis fitxers de cop així: *-T "{/ruta/fitxer1,/ruta/fitxer2}"* o també *-T "/ruta/imatge[1-10].png"*

NOTA: Properament estudiarem que tant el mètode POST com el mètode PUT permeten enviar dades al servidor, però el primer està dissenyat per "crear" aquesta nova dada al servidor (on se suposa que encara no hi és) i el segon està dissenyat per actualitzar la dada (que se suposa ja preexistent al servidor). És per això que a la pràctica, PUT es fa servir per implementar "pujades" de fitxers, que poden ser reiteratives
NOTA: També es poden pujar directament dades que formaran un fitxer al servidor així: *curl -X PUT -d "dades" http://servidor.com/fitxer*
NOTA: Un altre mètode similar a PUT és PATCH, però aquest darrer serveix, si el servidor en qüestió l'accepta, per modificar parts del recurs, no tot sencer com fa PUT (en tot cas, els estudiarem properament). PATCH, però, no té cap paràmetre associat a curl

Les "cookies" (galetes) són simples parelles de text del tipus *nom=valor* que un servidor HTTP envia al client per a què aquest les retingui un cert temps (en forma de fitxer, normalment) per tal de retornar-les en sol·licituds posteriors al mateix servidor original.

Les "cookies" són un mecanisme que utilitzen sovint els navegadors per mantenir certa persistència ("memòria") en diferents sol.licituds separades en el temps però dirigides al mateix destí. O dit d'una altra manera, moltes aplicacions i servidors utilitzen aquest mètode per connectar una sèrie de sol·licituds en una única sessió lògica.

El seu funcionament és així:

**1.**-El servidor envia una o més "cookies" al client mitjançant la capçalera HTTP de resposta *Set-Cookie:* amb certa informació que vol que romangui guardada al client (indicant també a quin nom DNS i ruta concreta del servidor es permetrà retornar aquesta a més de la seva data de caducitat i algunes propietats més)

**2.**-Quan un client es comunica amb un servidor amb un nom DNS i una ruta indicats en una galeta rebuda prèviament, el client (re)envia la galeta (amb tot el seu contingut) de retorn al servidor (tret que, per descomptat, hagin caducat) mitjançant la capçalera HTTP de petició *Cookie:*

Cal insistir, com s'ha dit, que cada galeta incorpora certes restriccions per indicar en quins destins són vàlides (noms DNS, rutes concretes...i a més una data de caducitat; respecte la data de caducitat, existeixen unes "cookies" especials anomenades "de sessió", que desapareixen tan bon punt el navegador es tanca (o es finalitza una sessió d'usuari en algun servei web)

*-c /ruta/fitxer* Indica la ruta del fitxer on s'escriuran totes les "cookies" rebudes del servidor remot (es pot indicar "-" si es volen veure per *stdout*). El format usat s'anomena "Netscape"

NOTA: El format Netscape estableix que cada "cookie" ha d'estar indicada en una línia diferent del fitxer i cada línia inclou els següents camps (separats per tabulador):

*domain flag path secure expiration name value*

Un exemple podria ser: *".netscape.com true / false 946684799 NET_ID 100103"*

NOTA: Aquest paràmetre també s'utilitza quan es vol que *curl* entengui les "cookies" rebudes (tot i no voler guardar-les) per tal de reaccionar en conseqüència si es donés el cas, així: *-c /dev/null*

*-b { cookie1=valor1;cookie2=valor2 | /ruta/fitxer }* Envia al servidor remot, en forma de "cookie", les dades indicades (se suposa que aquestes dades s'han rebut prèviament del servidor en una capçalera "Set-Cookie" d'una resposta prèvia). En el cas de no trobar-se cap símbol "=" com a valor del paràmetre *-b*, s'entendrà que s'està indicant la ruta d'un fitxer on llegir les "cookies" en qüestió (ruta que també pot ser "-" per indicar que aquestes provenen de *stdin*); el format d'aquest fitxer pot ser el mateix que accepta la capçalera *Set-Cookie* estàndard o bé el format Netscape (que és sovint el cas si el fitxer ha sigut generat prèviament amb el paràmetre *-c*)

*-k* En el cas de què la petició HTTPS es realitzi contra un servidor amb certificat autosignat, per defecte *curl* refusa connectar-hi degut a què no hi pot confiar (ja que els certificats autosignats són de "baixa qualitat"). Aquest paràmetre és per no realitzar aquesta protecció i, per tant, deixar fer la petició (seria equivalent a haver pitjat el botó d'"Establir excepció" quan passa la mateixa circumstància en un navegador).

*--cacert /ruta/ca.crt* Indica el certificat arrel de la CA que es farà servir per validar el certificat del servidor obtingut a la petició HTTPS. Alternativament aquest certificat es pot indicar a través de la variable d'entorn CURL_CA_BUNDLE. Si no s'indica res, per defecte *curl* fa servir el conjunt predeterminat de certificats arrels disponibles al sistema

*-E /ruta/client.crt* Indica el certificat de client que *curl* farà servir per connectar amb el servidor remot. Si aquest certificat està protegit per una "passphrase", es demanarà interactivament (a no ser que s'hagi indicat com a valor d'aquest paràmetre així: *-E /ruta/client.crt:passphrase* Si aquest certificat no inclou al seu interior la clau privada associada perquè aquesta està físicament en un altre fitxer, caldrà indicar la ruta d'aquesta clau privada per poder utilitzar-la en combinació amb el certificat en qüestió amb el paràmetre extra *--key /ruta/private.key*

Altres paràmetres de Curl potser no tan habituals però que mereixen la pena conèixer també són:

*-A "valor"* Equivalent a *-H "User-Agent:valor"*
*-e "URL"* Equivalent a *-H "Referer:URL"*

*-f* Fa que *curl* finalitzi amb error (*$?=22*) si troba un error 4xx o 5xx

*-K /ruta/fitxer.txt* Indica el fitxer que contindrà els diferents paràmetres (un per línia) que s'afegiran per defecte a qualsevol execució de *curl* (per tal d'així no haver-los d'escriure "a mà" sempre); es poden afegir comentaris en aquest fitxer iniciant la línia corresponent amb "#". Per defecte curl utilitza el fitxer "~/.curlrc", a no ser que s'indiqui el paràmetre *-q* En tot cas, teniu més informació a https://everything.curl.dev/cmdline/configfile

*-x URLproxy* Indica que per accedir a la URL del destí cal connectar com a intermediari amb el servidor proxy la URL del qual s'ha indicat com a valor d'aquest paràmetre. Els proxys poden ser HTTP, HTTPS i SOCKS5 (com és el cas, per exemple, de Tor); en aquest darrer cas, un example d'URL a indicar podria ser *socks5://ip.serv.proxy:nºport* o també *socks5**h**://ip.serv.proxy:nºport* (la diferència està en que en el primer cas la resolució del nom DNS del destí la fa *curl* directament mentre que en el segon cas la fa el servidor proxy indicat)

*--limit-rate nº{K|M|G}* Limita l'ample de banda màxim utilitzat per curl (tant per descàrregues com per pujades). Per defecte el número ve donat en bytes/s.

*-m nºsegons* Nombre de segons que *curl* s'esperarà com a màxim abans de deixar-ho córrer si no rep cap resposta. Pot venir acompanyat del paràmetre *--retry nº* (on cada reintent es realitzarà en un temps doble de l'anterior intent, o bé en el temps indicat per *–retry-delay nºsegons*)

*-u usuari:contrasenya* Si el servidor web demana autenticació, aquí s'aporta via HTTP. Per defecte s'utilitza el mètode "Basic", però es pot afegir el mètode "Digest" amb el paràmetre extra *--digest* Per conèixer altres mètodes, consulteu *man curl*

*-r nº-nº* Permet rebre una porció del recurs demanat (el rang indica el primer i darrer byte d'aquest recurs, normalment un fitxer). Per més informació es pot consultar l'article https://www.keycdn.com/support/byte-range-requests

Curl té moltíssimes més possibilitats de les mostrades a la llista anterior. En qualsevol cas, per un coneixement més exhaustiu d'aquestes possibilitats, remeto a la seva pàgina oficial, on hi ha disponible una documentació excel.lent.