Comandos útiles en la terminal de Linux (Parte 2)

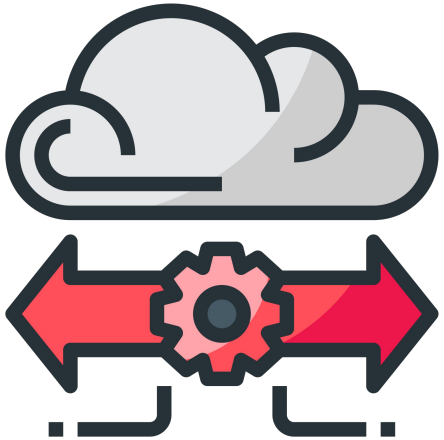
DigitalHouse>

Certified Tech Developer
The Ultimate Degree

Índice

1 | Escenario

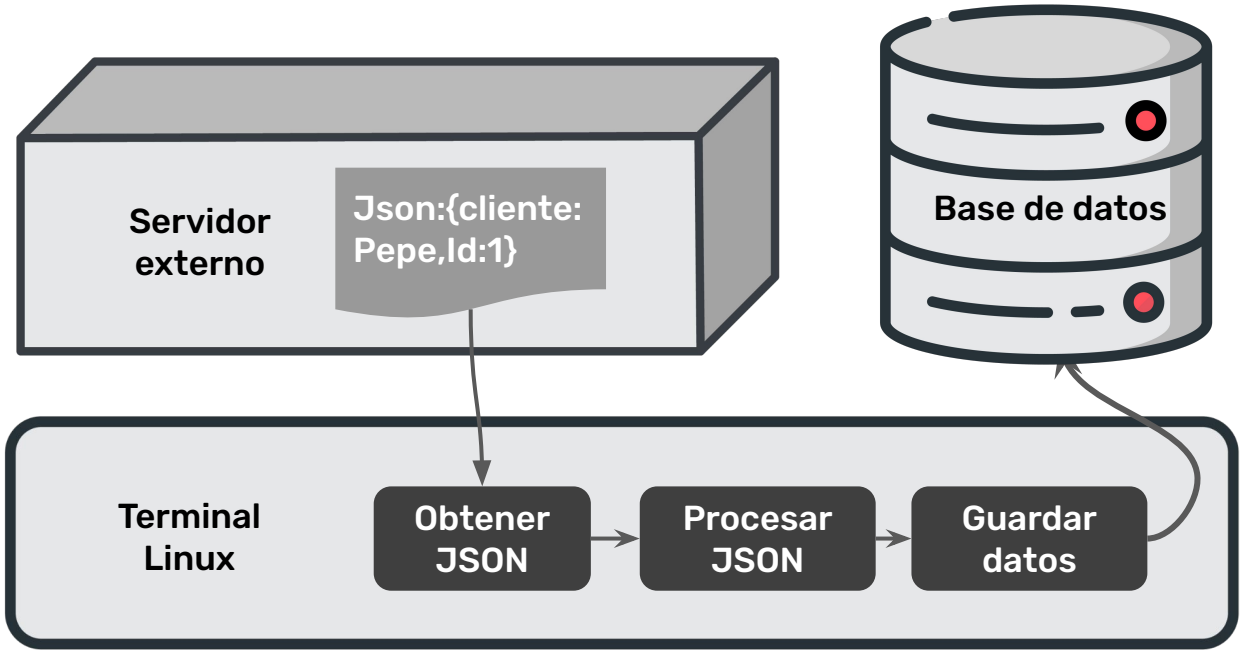
Obtener datos desde un web service



La terminal de Linux tiene tanta versatilidad y potencia que nos permite vincularla con un web service, obtener datos de allí y procesarlos con propósitos tales como agregarlo a archivos en nuestro servidor, modificarlos y republicarlos.

Las opciones son muy variadas, por ejemplo obtener un JSON desde una URL externa, procesar su contenido, obtener el o los atributos que nos interesen y —en función de ello— crear nuevos archivos, insertarlos en una base de datos.

Obtener datos desde un web service

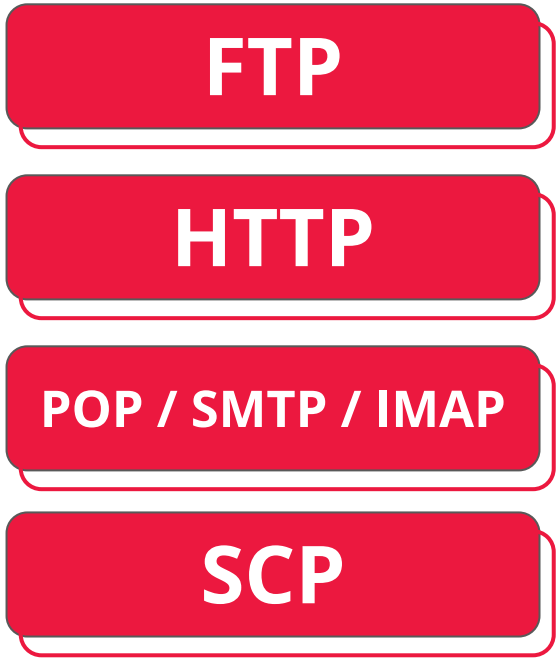


2 El comando cURL

Aspectos técnicos

Es un comando disponible en la mayoría de los sistemas basados en Unix. Es una abreviatura de “Client URL”. Los comandos de cURL están diseñados para funcionar como una forma de verificar la conectividad a las URL y como una gran herramienta para transferir datos.

El comando tiene una amplia compatibilidad con los protocolos más usados.



Sintaxis básica

El uso más simple de cURL es mostrar el contenido de una página. El siguiente ejemplo mostrará la página de inicio de digitalhouse.com:

```
{}  edorio@DESKTOP-W10:~$ curl https://www.digitalhouse.com
```

Como vemos, no es muy útil esto, ya que es difícil llegar a información que nos pueda ser de utilidad visualizándola solamente en pantalla. Pero si usamos el modificador **-o**, podremos escribir ese contenido HTML de la página de inicio en un archivo en nuestro equipo:

```
{}  edorio@DESKTOP-W10:~$ curl https://www.digitalhouse.com -o
    mipagina.html
```

Esto guardará todo el HTML en el archivo **mipagina.html**.

Descargas

El uso de este modificador puede extenderse a procesar descargas:

```
edorio@DESKTOP-W10:~$ curl
https://ubuntu.zero.com.ar/ubuntu-releases/20.04/ubuntu-20.04.2.0
-desktop-amd64.iso -o ubuntu.iso
```

Descarga la ISO de la URL de referencia y la nombrará **ubuntu.iso**.

```
edorio@DESKTOP-W10:~$  curl
https://ubuntu.zero.com.ar/ubuntu-releases/20.04/ubuntu-20.04.2.0
-desktop-amd64.iso -O -C 0
```

En este caso, no renombramos el archivo de destino (con el modificador **-O**). Además, permitimos la continuidad de la descarga con el modificador **-C**.

Encabezados y verificaciones

El modificador **-v** nos permite verificar la conectividad hacia un sitio remoto.

```
{}  edorio@DESKTOP-W10:~$ curl https://www.digitalhouse.com -v
```

Esto nos brindará, además del contenido, datos como la IP de destino, protocolos de seguridad y certificados utilizados.

```
{}  edorio@DESKTOP-W10:~$  curl https://www.digitalhouse.com -I
```

El modificador **-I** nos muestra todos los encabezados de la solicitud, tales como ruta por defecto, publicador web, entre otros.

Contenido JSON

Viendo todas las opciones brindadas, nos podemos imaginar lo útil de este comando con el propósito de obtener el contenido en formato JSON desde un endpoint que lo entregue en dicho formato. Por ejemplo, la API de OpenStreetMap, la cual nos devuelve una dirección pasándole las coordenadas, con la siguiente URL:

https://nominatim.openstreetmap.org/reverse.php?lat=-34.60378&lon=-58.38161&zoom=18&format=jsonv2

```
{}  edorio@DESKTOP-W10:~$  curl "https://nominatim.openstreetmap.org/reverse.php?lat=-34.60378&lon=-58.38161&zoom=18&format=jsonv2" -o resultado.json
```

Allí estamos guardando en el archivo resultado.json lo obtenido en el web service. Notemos el detalle de colocar la URL entre comillas simples o dobles.

3 El comando jq

Aspectos técnicos

JSON es un formato de datos estructurados ampliamente utilizado que se utiliza normalmente en la mayoría de las API y servicios de datos modernos. Es particularmente popular en aplicaciones web debido a su naturaleza liviana y compatibilidad con JavaScript.

Desafortunadamente, shells como Bash **no pueden interpretar y trabajar con JSON directamente**. Esto significa que trabajar con JSON a través de la línea de comando puede ser engorroso e implica la manipulación de texto utilizando una combinación de herramientas como **sed** y **grep**.

Allí es donde aparece jq, un potente procesador JSON para la consola.

Sintaxis básica

jq se basa en el concepto de filtros que funcionan sobre un flujo de JSON. Cada filtro toma una entrada y emite JSON a la salida estándar.

Tomando el archivo JSON obtenido con cURL, una ejecución sencilla de jq nos devuelve todo el contenido del JSON.

```
{}  edorio@DESKTOP-W10:~$ jq '.' resultado.json
```

Como vemos, no vamos a acceder a ningún atributo en especial, ya que con el modificador '.' no se lo indicamos.

Accediendo a propiedades

Para poder acceder a una propiedad específica es necesario indicarla luego del punto, con el nombre de la misma.

```
{}  edorio@DESKTOP-W10:~$ jq '.display_name' resultado.json
```

En este caso, accederemos a la propiedad **display_name** del JSON. Si queremos acceder a varias propiedades, las separamos por coma.

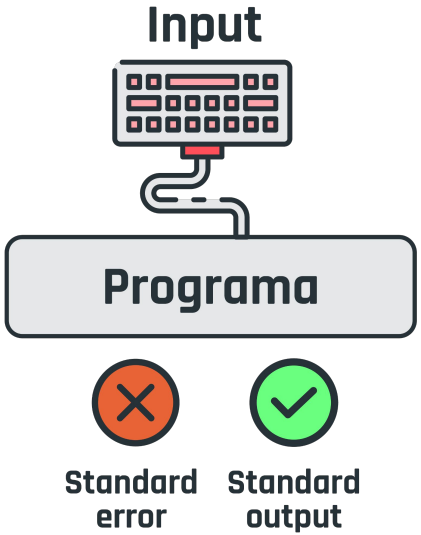
```
{}  edorio@DESKTOP-W10:~$ jq '.display_name,.type' resultado.json
```

De esta manera, accedemos a **display_name** y **type**.

TIP: Si alguna propiedad tuviese un espacio en su nombre, debemos envolverla con comillas dobles.

4 Combinación de uso de ambos comandos

Combinar comandos con pipelines



Para poder combinar el poder de **cURL** con el recurso y la capacidad de proceso de **jq** debemos combinarlo usando **pipelines**.

Para ello vamos a realizar una introducción a una de las características más interesantes que tiene la terminal.

El **pipeline o tubería** es una función que permite utilizar la salida de un programa como entrada en otro.

Combinar comandos con pipelines

El pipeline en Linux se representa con la barra vertical (|), la cual dividirá los comandos. Por ejemplo, si nosotros queremos saber la IP de nuestro equipo, lo hacemos con la instrucción:

```
{}  edorio@DESKTOP-W10:~$ ip address
```

Esta nos devolverá muchísimos datos (MAC, protocolos, direcciones IPv4 e IPv6, entre otros). Si quisiéramos filtrar dentro de ese texto por la cadena “192.168”, lo deberíamos llevar a un archivo y allí buscar con grep.

```
{}  edorio@DESKTOP-W10:~$ grep “192.168” miarchivo.txt
```

Pero es aquí en donde aparece la magia del pipeline, ya que podemos combinar ambas sentencias en una.

Combinar comandos con pipelines (cont.)

Para ello, primero colocamos nuestra sentencia inicial, sabiendo que tipo de salida puede tener, separamos con el pipeline y colocamos la segunda sentencia.

```
{}  edorio@DESKTOP-W10:~$ ip address | grep “192.168”
```

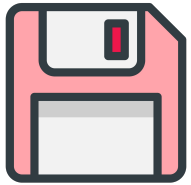
Allí el grep nos indicará la línea coincidente con “192.168”. Nuestro pipeline podría seguir aplicándose sin límites más allá de aquellos que imponga el sistema operativo, por ejemplo cantidad de procesos en ejecución.

Aplicar el pipeline con cURL y jq

Conociendo el uso básico del pipeline, vamos a aplicarlo a la obtención de datos externos y el parseo de una propiedad específica, la cual la guardaremos en un archivo. Nuestro comando tendrá tres partes:

Aplicar el pipeline con cURL y jq

En este caso, vemos cómo la sentencia obtiene el JSON con cURL, lo procesa con jq para obtener el **display_name** y el **type**, y finalmente lo guarda en un archivo llamado **consultapipe.txt**.



```
edorio@DESKTOP-W10:~$ curl "https://nominatim.openstreetmap.org/reverse.php?lat=-34.60378&lon=-58.38161&zoom=18&format=jsonv2" | jq ".display_name,.type" | tee consultapipe.txt
```

DigitalHouse>