Comandos úteis no Linux Terminal (Parte 2)

DigitalHouse >
Coding School

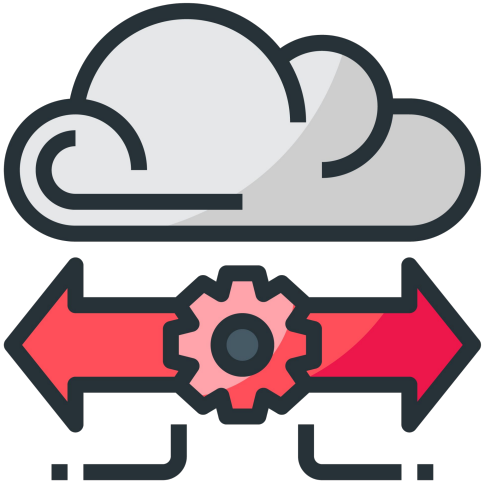
Certified Tech Developer
The Ultimate Degree

Índice

1. Estagio
2. O comando Curl
3. O comando Jq
4. Combinação de uso de ambos os comandos

1 | Estágio

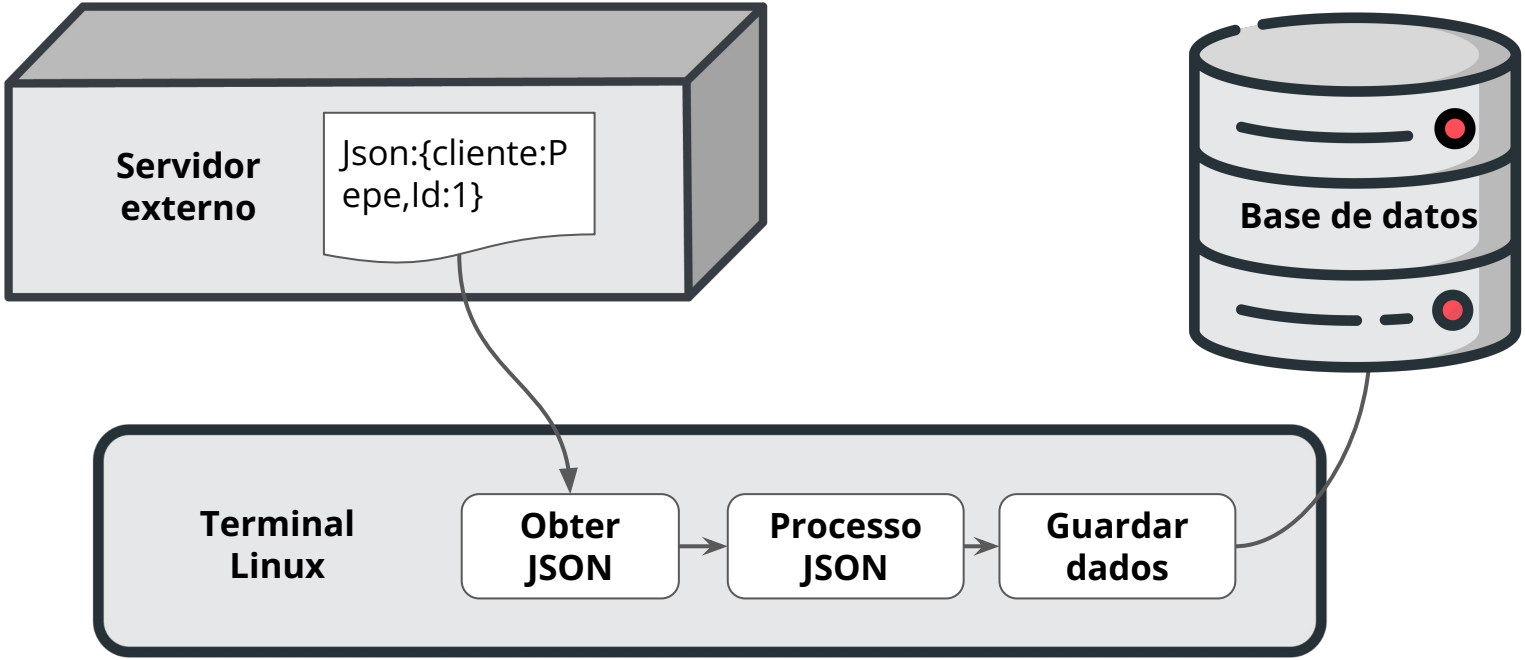
Obter dados de um serviço web



O terminal Linux tem tanta versatilidade e poder que nos permite vinculá-lo a um Webservice, obter dados de lá e processá-los para fins como adicioná-los a arquivos em nosso servidor, modificá-los e republicá-los.

As opções são muito variadas, como, por exemplo, obter um JSON de uma URL externa, processar seu conteúdo, obter o atributo ou atributos que nos interessam e com base nisso, criar novos arquivos, inseri-los em um banco de dados.

Obter dados de um serviço web



2 | O comando CURL

curl - Aspectos técnicos

É um comando disponível na maioria dos sistemas baseados em Unix. É uma abreviação de “URL do cliente”. Os comandos Curl são projetados para funcionar como uma maneira de verificar a conectividade com URLs e como uma ótima ferramenta para transferir dados.

O comando tem uma ampla compatibilidade com os protocolos mais utilizados, podemos vê-los na coluna da direita.

- FTP
- HTTP
- POP / SMTP / IMAP
- SCP

curl - Sintaxe Básica

O uso mais simples do **Curl** é exibir o conteúdo de uma página. O exemplo a seguir exibirá a página inicial do digitalhouse.com:

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

Como podemos ver, isso não é muito útil, pois é difícil alcançar informações que possam ser úteis para nós visualizando-as apenas na tela. Mas se usarmos a opção -o, podemos gravar esse conteúdo HTML da página inicial em um arquivo em nosso computador:

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

Que irá salvar todo o HTML no arquivo mypage.html

curl - Downloads

Ou o uso desse switch pode ser estendido para processar downloads:

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

Baixe o iso do URL de referência e nomeie-o como 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
```

Nesse caso, não renomeamos o arquivo de destino (com a opção -O) e também permitimos que o download continue com a opção -C.

curl - Cabeçalhos e verificações

A opção -v nos permite verificar a conectividade com um site remoto.

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

Além do conteúdo, ele nos fornecerá dados como IP de destino, protocolos de segurança e certificados utilizados.

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

E com a opção -I, ele nos mostra todos os cabeçalhos de solicitação, como rota padrão, editor da web e assim por diante.

curl - Conteúdo JSON

Vendo todas as opções fornecidas, podemos imaginar o quão útil é este comando para obter conteúdo em formato JSON de um endpoint que o entrega nesse formato, por exemplo, a API OpenStreetMap, que retorna um endereço passando as coordenadas, com o seguinte 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
```

Lá estamos salvando no arquivo de resultado .json o que é obtido no web service, vamos observar o detalhe de colocar a URL entre aspas simples ou duplas.

3 | O comando jq

jq - Aspectos técnicos

JSON é um formato de dados estruturados amplamente usado que é comumente usado na maioria dos serviços de dados e APIs modernos. É particularmente popular em aplicativos da Web devido à sua natureza leve e suporte a JavaScript.

Infelizmente, shells como o Bash não podem interpretar e trabalhar com JSON diretamente. Isso significa que trabalhar com JSON por meio da linha de comando pode ser complicado e envolve manipulação de texto usando uma combinação de ferramentas como sed e grep.

É aí que entra o jq, um poderoso processador JSON para o console.

jq - Sintaxis básica

jq é baseado no conceito de filtros trabalhando em um fluxo JSON. Cada filtro recebe uma entrada e emite JSON para a saída padrão.

Pegando o arquivo json obtido com

curl, uma simples execução de jq retorna todo o conteúdo doJSON.

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

Como podemos ver, não vamos acessar nenhum atributo em particular, pois não o indicamos com o modificador '.'.

jq - Acessando propriedades

Para acessar uma determinada propriedade, é necessário indicá-la após o ponto, com seu nome.

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

Nesse caso, acessaremos a propriedade "display_name" do Json. Se quisermos acessar várias propriedades, as separamos por vírgulas.

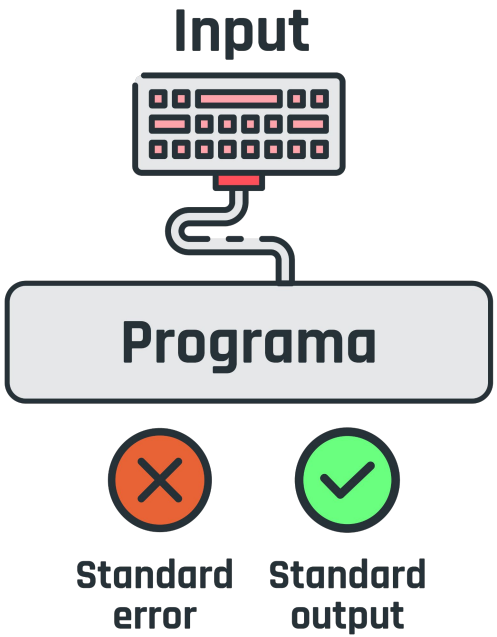
```
{}  edorio@DESKTOP-W10:~$ jq '.display_name,.type' resultado.json
```

Desta forma, acessamos “display_name” e “type”.

DICA: Se alguma propriedade tiver um espaço em seu nome, devemos envolvê-la com aspas duplas.

4 Combinação de uso de ambos os comandos

Combine comandos com pipelines (|)



Para combinar o poder do curl com o recurso e poder de processamento do jq, devemos combiná-lo usando pipelines.

Para isso, faremos uma introdução a um dos recursos mais interessantes do terminal.

O **pipeline ou tuberia** é uma função que permite que a saída de um programa seja usada como entrada para outro.

Combine comandos com pipelines (|)

O pipeline no Linux é representado pela barra vertical |, que dividirá os comandos. Por exemplo, se queremos saber o IP do nosso equipamento, fazemos com a instrução

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

O que nos retornará muitos dados (MAC, protocolos, endereços IPv4 e IPv6), se quisermos filtrar esse texto pela string '192.168', devemos levá-lo para um arquivo e pesquisar lá com grep.

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

Mas é aí que a mágica do cachimbo aparece, já que podemos combinar as duas declarações em uma só.

Combine comandos com pipelines (|)

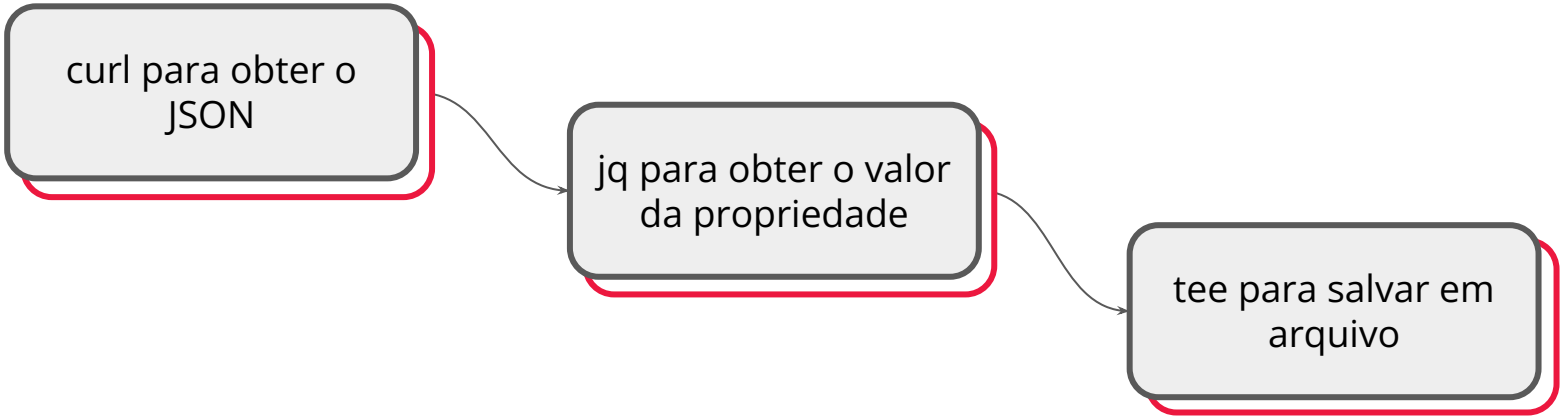
Para isso, primeiro colocamos nossa sentença inicial, sabendo que tipo de saída ela pode ter, separamos com o pipe | e colocamos a segunda frase.

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

Lá o grep indicará a linha que coincide com "192.168"; nosso pipe poderia continuar sendo aplicado sem limites além daqueles impostos pelo sistema operacional, como o número de processos em execução.

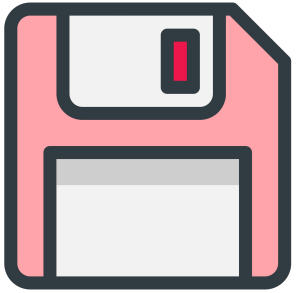
Aplique o pipeline com curl e jq

Conhecendo o uso básico do pipe, vamos aplicá-lo para obter dados externos e analisar uma propriedade específica, que salvaremos em um arquivo. Nosso comando terá 3 partes:



Aplique o pipeline com curl e jq

Nesse caso, vemos como a instrução obtém o JSON com curl, processa-o com jq para obter o display_name e o tipo e, finalmente, salva em um arquivo chamado querypipe.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>
Coding School