01

HTTP2 - Dados binários, GZIP ativo e TLS

Transcrição

Até agora sempre usamos o browser para realizar uma requisição. Mas podemos realizar fora dele usando a linha de comando por exemplo. Um programa famoso para isso é o **CURL**. No Linux e MacOS ele já vem instalado por padrão.

Caso esteja usando o Windows é necessário a instalação dele. O download deve ser feito por aqui: https://curl.haxx.se/download.html (https://curl.haxx.se/download.html)

Para realizar e depurar uma requisição via CURL podemos simplesmente executar no terminal o seguinte comando:

```
curl -v www.caelum.com.br
```

Uma saída típica dele seria:

```
Fabios-MacBook-Pro:~ fabiopimentel$ curl -v www.caelum.com.br
* Rebuilt URL to: www.caelum.com.br/
*   Trying 172.217.29.51...
* Connected to www.caelum.com.br (172.217.29.51) port 80 (#0)
> GET / HTTP/1.1
> Host: www.caelum.com.br
> User-Agent: curl/7.49.1
> Accept: */*
>
< HTTP/1.1 200 OK
< Content-Type: text/html; charset=utf-8
< Vary: Accept-Encoding,User-Agent
< Content-Language: pt-br
< Content-Type: text/html;charset=UTF-8
< X-DNS-Prefetch-Control: on
< X-Cloud-Trace-Context: 3e5e270ee3ab1e79f81b10d2cdef53cd
< Date: Fri, 24 Mar 2017 19:20:12 GMT
< Server: Google Frontend
< Content-Length: 95776
<
       <!DOCTYPE html>
    <html class="no-js"lang="pt-br"> <head> <title>Caelum | Cursos de Java, .NET, Android, PHP,
    …
```

Pode-se notar pela saída que temos logo no começo as informações do request efetuado:

```
> GET / HTTP/1.1
> Host: www.caelum.com.br
> User-Agent: curl/7.49.1
> Accept: */*
`
```

E após essas infos temos o cabeçalho da resposta obtida pelo servidor:

```
< HTTP/1.1 200 OK
< Content-Type: text/html; charset=utf-8
< Vary: Accept-Encoding,User-Agent
< Content-Language: pt-br
< Content-Type: text/html;charset=UTF-8
< X-DNS-Prefetch-Control: on
< X-Cloud-Trace-Context: 3e5e270ee3ab1e79f81b10d2cdef53cd
< Date: Fri, 24 Mar 2017 19:20:12 GMT
< Server: Google Frontend
< Content-Length: 95776
```

Logo depois vem o corpo da resposta (HTML da página requisitada):

```
<!DOCTYPE html> <html class="no-js"lang="pt-br"> <head> <title>Caelum | Cursos de Java, .NET, A
```

Em resumo o output apresentando pelo CURL possui essa divisão:

HTTP/2

O protocolo que estamos trabalhando até agora foi especificado na década de 90 e de lá até hoje muitas alterações foram feitas até na forma como usamos a internet.

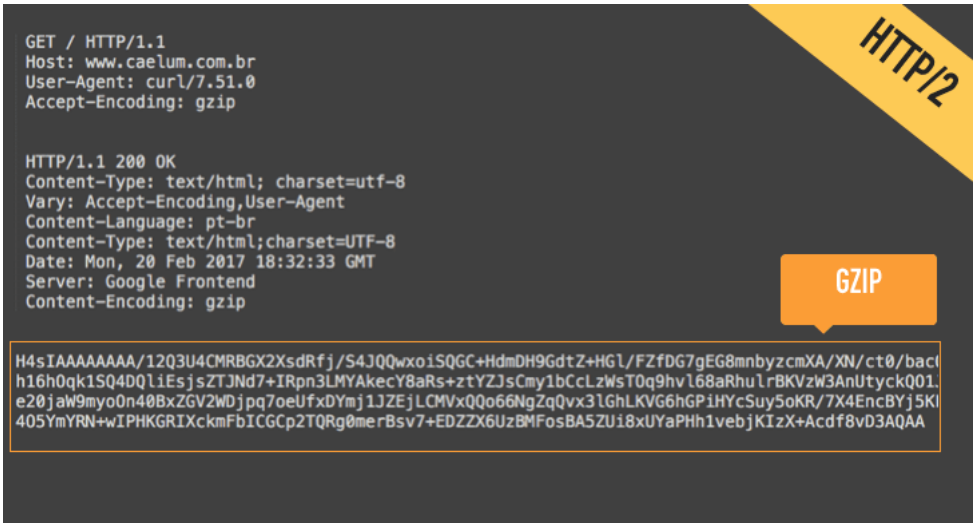
Com a chegada do mundo mobile novas preocupações apareceram e otimizações são cada vez mais necessárias para uma boa performance. Por isso uma mudança foi necessária e em 2015 depois de alguns anos de especificações e reuniões surgiu a versão 2 desse protocolo.

A nova versão é batizada de **HTTP/2** e tem como página principal de documentação e referência essa: `https://http2.github.io/` .

A nova versão do protocolo HTTP traz mudanças fundamentais para a Web. Recursos fantásticos que vão melhorar muito a performance da Web além de simplificar a vida dos desenvolvedores.

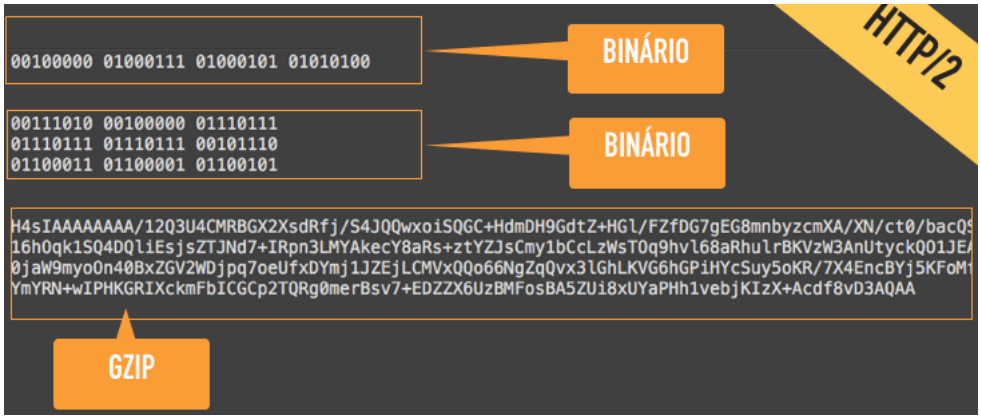
No HTTP 1.1, para melhorar a performance, habilitamos o **GZIP** no servidor para comprimir os dados das respostas. É uma excelente prática, mas que precisa ser habilitada explicitamente. No HTTP/2, o **GZIP é padrão e obrigatório**.

É como se a gente passasse a ter a resposta assim:

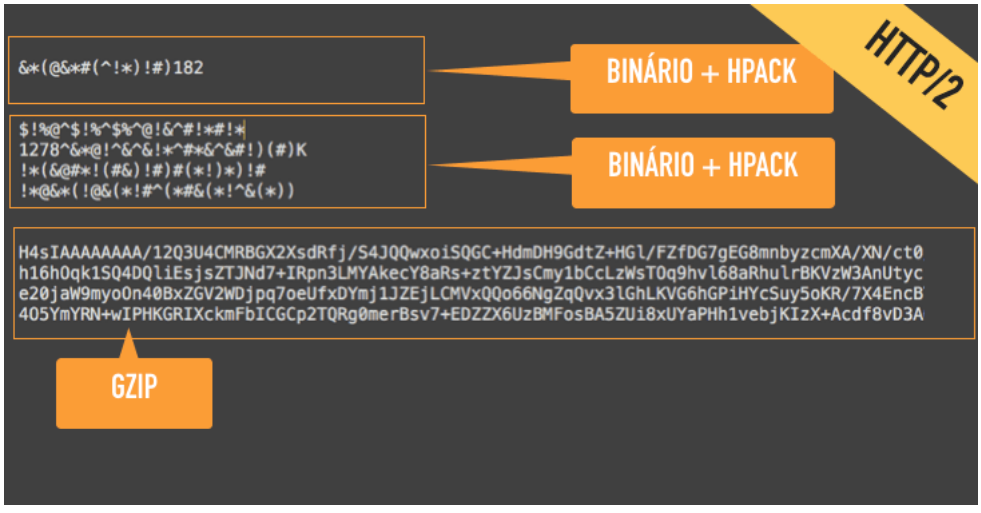


Mas, se você já olhou como funciona uma requisição HTTP, vai notar que só GZIPar as respostas resolve só metade do problema. Tanto o `request` quanto o `response` levam vários cabeçalhos (headers) que não são comprimidos no HTTP 1.1 e ainda viajam em texto puro.

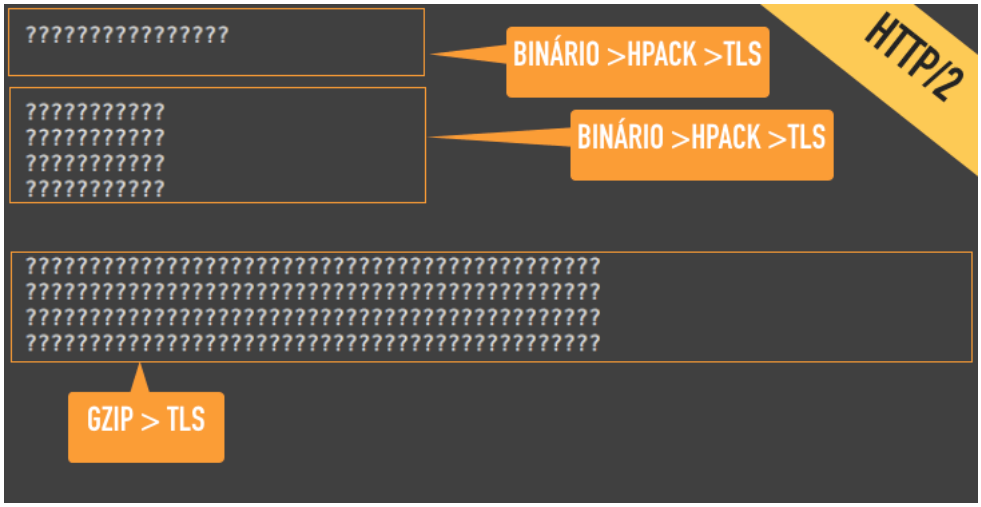
Já na nova versão, os headers passam a ser binários:



Além de binários eles são comprimidos usando um algoritmo chamado **HPACK**. Isso diminui bastante o volume de dados trafegados nos headers.



Além de todas essas otimizações para melhorar a performance ainda houve uma preocupação com segurança exigindo TLS por padrão também.


HTTP/2
BINÁRIO >HPACK >TLS
BINÁRIO >HPACK >TLS
GZIP > TLS