01

Salvando tabelas e enviando para o Bucket

Transcrição

Na aula passada, criamos o bucket na Amazon, mas agora a nossa tarefa é criar o script para salvar as tabelas presentes no banco da *Mutillidae* em arquivos separados, e finalmente enviá-las para a Amazon.

Segue os passos para acessar o MySQL, o banco de dados `mutillidae`, e ver as tabelas presentes:

```
$ sudo mysql -u root
mysql> use mutillidae
mysql> show tables;
```

Ao invés de digitarmos todos esses comandos várias vezes, podemos consolidá-los em uma única linha, assim deixaremos o código um pouco mais enxuto. Colocaremos o comando `exit` para voltar para o terminal.

Testaremos o mesmo código, só que na mesma linha:

```
$ sudo mysql -u root mutillidae -e "show tables;"
```

Muito bem! Temos as informações da mesma maneira. Mas para nós é interessante ter somente os nomes das tabelas, e o melhor seria descartar o resto das informações como o cabeçalho *Tables_in_mutillidae*. Então queremos criar um *filtro* para manter somente as linhas que contém **produtos** e **usuarios**. Como já fizemos isso anteriormente, usaremos o `grep` para nos ajudar a filtrar somente essas duas linhas.

Redirecionaremos a saída do comando anterior para o `grep`. Entretanto, se nós colocarmos somente o comando `[comeco_do_comando] | grep Tables` para mostrar somente as tabelas, o que será retornado será justamente a linha que contém a palavra "Tables". Então, faremos um **filtro invertido**, acrescentando o comando `-v` dessa forma:

```
$ sudo mysql -u root mutillidae -e "show tables;" | grep -v Tables
```

Após realizar esse teste aqui no terminal, vamos selecionar esse comando e levá-lo ao diretório de scripts para começar a montar o nosso novo script! Vamos chamá-lo de `backup-amazon.sh`:

```
$ cd Scripts
$ nano backup-amazon.sh
```

A primeira informação do script será o interpretador:

```
#!/bin/bash
```

E em seguida, podemos aproveitar e colar o **comando** que testamos no terminal. Como é um comando, vamos envolvê-lo em `$()`:

```
#!/bin/bash

tabelas=$(mysql -u root mutillidae -e "show tables;" | grep -v Tables)
```

Armazenamos o resultado desse comando na variável `tabelas`. Dessas tabelas, queremos salvar cada uma em um arquivo separado, como por exemplo um arquivo chamado de `produtos.sql` e um outro arquivo de `usuarios.sql`. É necessário fazer um *dump* dessas tabelas, para que possamos salvar os arquivos nas suas respectivas extensões `.sql`. Para automatizar o processo de repetição, usaremos o laço **for**.

```
#!/bin/bash

tabelas=$(mysql -u root mutillidae -e "show tables;" | grep -v Tables)
for  tabela in $tabelas
do
    mysqldump -u root mutillidae $tabela > $tabela.sql
done
```

Para cada tabela no conteúdo da variável `$tabelas`, faça o *dump* com o usuário root, do banco mutillidae, sendo produtos ou usuários. Assim que tiver o conteúdo de `$tabela` em mãos, salve-a em um arquivo de mesmo nome, com a extensão `.sql`!

Devemos nos lembrar que os diretores nos pediram para consolidar todas essas informações dentro de um diretório com a data em que o backup foi realizado. Para consolidar essa informações, e evitar que os arquivo fiquem perdidos, voltemos a "Home" com o comando `cd` após ter finalizado e salvado as alterações do script, e criaremos um diretório chamado de `backup_mutillidae_amazon`, onde concentraremos todas as informações de backup.

```
$ mkdir backup_mutillidae_amazon
```

Legal! Voltemos ao script, e diremos que todas as informações dos diretórios com as datas, devem estar dentro do diretório na "Home". Como será preciso utilizar esse caminho algumas vezes, colocaremos em uma **constante**:

```
#!/bin/bash

CAMINHO_BACKUP=/home/rafael/backup_mutillidae_amazon
```

Agora, precisamos sair do diretório `/Scripts`, e entrar no diretório `/backup_mutillidae_amazon`.

```
#!/bin/bash

CAMINHO_BACKUP=/home/rafael/backup_mutillidae_amazon
cd $CAMINHO_BACKUP
```

Verificaremos se dentro do diretório `/backup_mutillidae_amazon` não existe nenhum outro diretório com a data que o backup foi realizado, precisaremos criar esse diretório, não é mesmo?

Para pegar a **data completa** do dia atual, utilizaremos o comando `date +%F`:

```
#!/bin/bash

CAMINHO_BACKUP=/home/rafael/backup_mutillidae_amazon
cd $CAMINHO_BACKUP
data=$(date +%F)
```

Quando a linha acima for executada dentro do script, precisamos verificar se há um diretório com a data atual, onde o backup foi realizado. Caso não exista, devemos criá-lo:

```
#!/bin/bash

CAMINHO_BACKUP=/home/rafael/backup_mutillidae_amazon
cd $CAMINHO_BACKUP
data=$(date +%F)
if [ ! -d $data ]
then
    mkdir $data
fi
```

Criamos um diretório chamado de `/backup_mutillidae_amazon` , estamos verificando se dentro dele já tem um diretório com a data na qual o backup foi realizado, e agora só falta colocar as `$tabelas.sql` dentro do diretório `/data` que estará dentro de `/backup_mutillidae_amazon` . Com isso, temos que passar o caminho completo indicando onde essas informações deverão ser salvas:

```
#!/bin/bash

CAMINHO_BACKUP=/home/rafael/backup_mutillidae_amazon
cd $CAMINHO_BACKUP
data=$(date +%F)
if [ ! -d $data ]
then
    mkdir $data
fi

tabelas=$(mysql -u root mutillidae -e "show tables;" | grep -v Tables)
for  tabela in $tabelas
do
    mysqldump -u root mutillidae $tabela > $CAMINHO_BACKUP/$data/$tabela.sql
done
```

Vamos fazer um teste! Colocamos "Ctrl + X" para sair, e "Y" para salvar as alterações do script.

```
$ sudo bash backup-amazon.sh
```

Como podemos ver, temos os dois arquivos separados `produtos.sql e usuarios.sql` , que estão na pasta com a data atual do backup, que se encontra dentro da pasta `/backup_mutillidae_amazon` . Agora, o próximo passo é enviar essas informações para o **bucket da Amazon**. Para isso, vamos voltar ao nosso script, e vamos dizer que quando esse backup for realizado, será enviado para o bucket. No final do script, mandaremos todo o conteúdo do diretório `/backup_mutillidae_amazon` para o bucket, realizando uma sincronização das informações que foram alteradas nesse diretório, para o bucket.

Para realizar esse envio, colocamos da seguinte maneira:

```
for  tabela in $tabelas
do
    mysqldump -u root mutillidae $tabela > $CAMINHO_BACKUP/$data/$tabela.sql
done

aws s3 sync $CAMINHO_BACKUP s3://curso-shell-script
```

Essa última linha, manda todo o conteúdo do diretório `/backup_mutillidae_amazon` para o bucket `curso-shell-script` ! Vamos sair com "Ctrl + X" e "Y" para salvar.

Mas imagine se precisássemos ficar lembrando de rodar esse script a todo momento. A chance de esquecermos será grande, se será justamente quando esquecermos que vamos ter problemas. O ideal é que façamos o *backup* de tempos em tempos. Para isso, utilizaremos o **crontab**! Mas antes, precisamos dar a permissão de execução desse script.

```
$ chmod +x backup-amazon.sh
```

Feito isso, vamos para o crontab:

```
$ sudo crontab -e
```

Nas etapas anteriores, configuramos o crontab para que ele enviasse as informações a cada dois minutos, só que agora podemos fazer um pouco diferente. Podemos pedir para o crontab executar o script em uma determinada hora do dia. Aqui no curso, pediremos as 10 horas e 14 minutos, para ver se ele funciona, mas fique a vontade para testar a hora que achar melhor!

```
# m h   dom mon dow    command
14 10  * * *  /home/rafael/Scripts/backup-amazon.sh
```

Colocamos "Ctrl + X" e "Y" para salvar.

Após chegar o tempo as 10 horas e 14 minutos, o script deve ter sido executado. Vamos acessar o bucket na Amazon S3, e podemos ver que agora existe uma pasta nomeada com a data do *backup*, e dentro dessa pasta, temos os arquivos que estão separados. Agora já estamos conseguindo enviar as informações locais do arquivo de *backup* para a Amazon. O que falta é descobrir como pegar o conteúdo da Amazon e restabelecer no banco da *Mutillidae*, caso esse banco apresente algum problema.