Fazendo backup das bases de dados do MySQL

As bases de dados do MySQL são salvas por padrão dentro da pasta "/var/lib/mysql". Ao criar a base de dados "phpbb", por exemplo, será criada a pasta "/var/lib/mysql/phpbb", contendo um conjunto de arquivos, referentes às tabelas criadas.

A segunda opção é fazer um backup online, sem parar o servidor. O utilitário mais simples (e provavelmente o mais usado) para isso é o mysqldump, que acompanha o pacote principal do MySQL.

Diferente do método anterior, onde os arquivos são copiados diretamente, o mysqldump acessa o banco de dados por vias normais, da mesma forma que um aplicativo qualquer faria. Em outras palavras, ele não lê os arquivos, mas sim as informações armazenadas nas bases de dados. Isso permite que o backup seja consistente, mesmo que as bases de dados sejam alteradas durante o backup.

Para salvar todas as bases de dados do servidor no arquivo "backup.sql", criado no diretório atual, por exemplo, o comando seria:

```
# mysqldump -u root -p -x -e -A > backup.sql
```

O "-u root -p" especifica o usuário que será usado para acessar o banco de dados. No exemplo estou fazendo um backup completo, por isso estou usando diretamente o root. A opção "-x" trava as bases de dados no momento em que cada uma é copiada, evitando qualquer problema de inconsistência, enquanto a "-e" é uma opção de otimização, que permite ao mysqldump combinar argumentos INSERT dentro das tabelas, o que torna tanto o backup quanto a restauração mais rápidos. Finalizando, a opção "-A" especifica um backup completo, de todas as bases de dados.

Se o comando parasse por aí, o mysqldump simplesmente escreveria todo o conteúdo das bases de dados na própria janela do terminal, resultando em uma longa exibição de informações, sem muita utilidade. Como queremos que a saída seja salva em um arquivo, usamos o ">", que redireciona a saída para o arquivo especificado.

O arquivo "backup.sql" gerado é basicamente um arquivo de texto gigante contendo declarações de todas as informações armazenadas. Você pode reduzir o tamanho do arquivo para um quarto (ou menos) do tamanho original compactando o arquivo, o que pode ser feito adicionando a opção "| gzip" antes do ">" no comando, como em:

```
# mysqldump -u root -p -x -e -A | gzip > backup.sql.gz
```

Note que nesse exemplo adicionei também o ".gz" no nome do arquivo, indicando que se trata de um arquivo compactado. Para usá-lo posteriormente, você precisaria apenas descompactar o arquivo, usando o comando "gunzip", como em:

```
# gunzip backup.sql.gz
```

O maior problema com estes dois comandos é que você precisa digitar a senha depois de rodar o comando, o que dificulta seu uso em scripts de backup automático. É possível eliminar a necessidade de digitar a senha especificando-a diretamente no comando, depois do "-p" (sem espaços), como em:

```
# mysqldump -u root -p12345 -x -e -A | gzip > backup.sql.gz
```

Note que ao incluir senhas em arquivos, é extremamente importante restringir as permissões, de forma que apenas o root (ou o usuário em questão) tenha permissão para lê-lo. Qualquer outro usuário do servidor que tenha acesso de leitura no arquivo, poderá ler a senha e acessar o servidor MySQL.

Continuando, os comandos acima permitem fazer um backup completo de todas as bases do servidor, que poderia ser usado para restaurar os dados em uma instalação limpa do MySQL. É possível também fazer backups localizados, contendo apenas uma base de dados específica.

Nesse caso, em vez de usar a opção "-A", você usaria a opção "-B", seguida pela base de dados a ser salva, como em:

```
# mysqldump -u root -p -x -e -B phpbb > phpbb.sql
```

Na hora de restaurar o backup, deixamos de usar o mysqldump e passamos a utilizar o cliente mysql, que se encarrega de ler os comandos e os dados adicionados nos arquivos e usá-los para povoar as bases de dados. O comando ficaria então:

```
# mysql -u root -p --database=phpbb < phpbb.sql
```

Você pode também especificar a senha diretamente no comando, assim como no caso do mysqldump, como em:

```
# mysql -u root -p12345 --database=phpbb < phpbb.sql
```

Se você tentar restaurar o backup sobre uma base de dados contendo dados, provavelmente receberá uma mensagem de erro logo no início do processo, avisando que uma das tabelas já existe, como em:

```
ERROR 1050 at line 19: Table 'wp_comments' already exists
```

A solução no caso é remover a base de dados antiga usando o cliente MySQL e criar outra em branco para então fazer a restauração, como em:

```
# mysql -u root -p

Enter password:

Welcome to the MySQL monitor. Commands end with ; or g.
Your MySQL connection id is 18
Server version: 5.0.32-Debian_7etch5-log Debian etch distribution
Type 'help;' or 'h' for help. Type 'c' to clear the buffer.

mysql> DROP DATABASE phpbb;

Query OK, 0 rows affected (0.00 sec)

mysql> CREATE DATABASE phpbb;

Query OK, 1 row affected (0.00 sec)

mysql> exit

Bye

# mysql -u root -p --database=phpbb < phpbb.sql
```

Outra opção seria adicionar a opção "--add-drop-table" ao gerar o backup com o mysqldump. Ela faz com que ele inclua instruções para que as bases sejam excluídas e recriadas automaticamente durante a restauração, evitando que você precise fazê-lo manualmente. O comando ficaria então:

```
# mysqldump --add-drop-table -u root -p -x -e -B phpbb > phpbb.sql
```

O comando para restaurar continua o mesmo, com a diferença de que você não precisa mais dar o DROP DATABASE; CREATE DATABASE antes de fazer a restauração.

Esta opção pode ser adicionada também ao comando para fazer o backup completo das bases de dados, facilitando assim sua restauração:

```
# mysqldump --add-drop-table -u root -p -x -e -A > backup.sql
```

O backup poderia ser então restaurado diretamente usando o comando abaixo, sem que você precisasse remover as bases e tabelas manualmente antes de iniciar a recuperação:

```
# mysql -u root -p < backup.sql
```

O backup usando o mysqldump e a restauração usando o mysql são preferíveis à cópia manual dos arquivos da pasta "/var/lib/mysql", pois evita problemas de incompatibilidade ao migrar os dados para versões diferentes do MySQL.

Além disso, a facilidade de fazer o backup sem precisar parar o servidor é uma grande vantagem em um ambiente de produção.