Artigo

Estratégias de backup e restore no PostgreSQL

Este artigo apresenta como implementar uma boa estratégia de backup e restore. Será discutido o funcionamento do Continous Archiving, também conhecido como backup incremental.

Por que eu devo ler este artigo: Este artigo objetiva mostrar como implementar uma boa estratégia de backup e de restore para seu banco de dados. Para isso, será abordado o funcionamento do Continous Archiving, também conhecido como backup incremental. Vamos saber em que situações este tipo de backup é útil, quais os passos para habilitá-lo, como realizar o restore e o recover da base de dados. Se você tem um banco de dados PostgreSQL e considera que os dados contidos nele são importantes, então é recomendável que você entenda como criar uma boa estratégia de backup e que também tenha em um bom

Uma das atividades mais importantes de um administrador de banco de dados é, sem dúvidas, planejar e implementar uma boa estratégia de backup para se prevenir de possíveis desastres. Nem sempre é possível prever tudo, mas para se prevenir de possíveis falhas é importante considerar desastres que podem ocorrer com seu banco de dados: terremoto, furacão, incêndios, falha de hardware, falha de software e até falha humana. Para se proteger de todas elas existe um custo e a implementação vai depender do nível de investimento que será feito e dos riscos que a empresa está disposta a correr. Nem sempre o administrador de banco de dados consegue se proteger da maioria dos desastres, pois não depende só dele. Nesses casos é interessante deixar os responsáveis pelos sistemas cientes da situação.

Para criação da melhor estratégia de backup para o ambiente, temos que considerar quais as necessidades dos clientes que usam o serviço de banco de dados. A seguir abordaremos quais pontos devem ser levados em consideração.

Uma sigla bastante importante para definição da nossa estratégia é o ANS (Acordo de Nível de Serviço) ou em inglês SLA (*Service Level Agreement*). Proveniente do ITIL, biblioteca com melhores práticas de infraestrutura de TI, o ANS é o acordo entre o provedor de serviços de TI e o cliente para um determinado tipo de serviço. No nosso caso, o serviço em questão é o de banco de dados. E o acordo será o tempo de indisponibilidade desse serviço durante uma parada não programada. Quanto tempo é aceitável para o cliente ficar sem esse serviço no caso de um desastre?

No caso de um desastre, dependendo da estrutura da empresa, é muito provável que será necessário um restore usando o backup realizado. E claro, de nada vale um

backup precisa ter um RTO que atenda o ANS relacionado a desastres.

Outro acordo que deve ser definido é o período aceitável de perda de informações em caso de falha. Existe um período aceitável em que, no caso de um restore, os dados podem ser perdidos ou não é aceitável de maneira alguma perder dados?

A criação dos backups históricos também terá um peso grande na definição da estratégia. Em alguns casos em que ocorre o expurgo das informações do banco de dados em determinados períodos, se faz necessária a criação de um backup que será armazenado por um longo período. Essa retenção visa atender algumas leis ou até mesmo auditoria da empresa.

Para atender todas essas situações, é necessário um investimento bem alto, pois é preciso ter uma boa infraestrutura. Além desse investimento, existe um outro essencial que é a alocação de pessoal. Assim como a etapa da implementação, existe toda a parte da criação da documentação, revisão e atualização dos processos. Mesmo ciente dos pontos necessários, nem sempre as empresas estão dispostas a pagar. Existem casos em que os dados armazenados não justificam um enorme valor de investido no backup.

WAL - Write-Ahead Logging

Para começar, é importante termos em mente que os logs de transação, conhecidos como logs de WAL no PostgreSQL, possuem um papel fundamental para garantir a integridade dos dados e também têm uma função muito importante no desempenho do banco de dados.

transação para só depois serem escritas nos arquivos de dados do banco, também conhecidos com datafiles. Por isso a importância deles no desempenho, pois como temos a certeza de que os logs possuem a informação guardada, o banco não precisa ficar se preocupando em gravar todas as alterações em tempo real nos datafiles.

Os arquivos de WAL também são necessários para integridade. Com o histórico guardado nos arquivos de log, mesmo as informações que ainda não foram salvas nos datafiles depois de uma queda de energia ou outro desastre podem ser refeitas. Esse processo é chamado de *roll-foward recovery,* também conhecido como REDO. Em resumo o fluxo seria: usuário inseriu uma informação e logo em sequência realizou um *commit;* os dados são descarregados do buffer dos logs para os arquivos de redo, para só depois serem escritos nos arquivos do banco de dados.

No PostgreSQL, os arquivos de log são armazenados dentro do diretório pg_xlog que fica armazenado no diretório de dados da sua instância. Por padrão, o banco inicia com 3 segmentos WAL. Estes segmentos possuem um tamanho total de 16M e são compostos por blocos de 8k.

Tipos de backup e continuous archiving

Existem dois tipos de backups: o lógico e o físico. O backup lógico é geração dos comandos SQL necessários para recuperação da base de dados dentro de um arquivo texto. Este tipo de backup não é muito indicado para recuperação de desastres, pois o arquivo com os comandos para recuperação foi gravado em um momento específico do tempo e durante a recuperação dos dados teremos que voltar o banco

até as 2h. Então você terá perdido pelo menos 10h de informação da sua base de dados.

O lógico funciona mais como um incremento na sua estratégia e pode ser usado, por exemplo, para backups históricos que devem ser armazenados por um longo período.

Durante o backup físico é feita uma cópia binária dos arquivos do seu banco de dados e ele pode ser dividido em quente e frio. O backup frio nada mais é do que parar todo seu banco de dados e realizar a cópia de todos os arquivos, mas aí temos o problema da indisponibilidade da base. Dependendo do ambiente, esta indisponibilidade pode não ser aceitável. Já no quente, não precisamos parar o banco de dados. Mas esse tipo de backup só é possível quando temos habilitado o arquivamento de logs WAL.

Como todo arquivo importante, os arquivos de REDO também devem ter seus backups para evitar perda de informação em caso de sua corrupção. Então é importante que seja habilitado o arquivamento dos logs de WAL, que nada mais é que um backup dos seus arquivos de REDO. Após habilitado, é recomendado que os logs arquivados sejam gravados em um servidor diferente de onde está seu banco de dados.

O Continuous Archiving é nome dado aos backups contínuos das sequências dos logs WAL. Esses backups serão usados em conjunto com o backup físico do seu banco de dados. No caso de um recover, precisaremos ter a sequência contínua dos logs de REDO desde o último backup realizado. Desta maneira conseguiremos realizar a

desastre; diferente dos lógicos, podemos recuperar o banco em qualquer ponto no tempo. Além das utilidades já citadas, este arquivamento será bastante importante para economizar tempo nos backups de grandes bases de dados onde é inviável realizar uma cópia completa de todo o banco de dados todos os dias da semana.

Imagine um banco de dados de 800Gb. Dependendo da infraestrutura disponível, seria muito custoso ou até impossível realizar um backup completo de toda estrutura da sua base de dados todos os dias da semana sem ter um impacto direto no ambiente de produção. Para otimização deste tempo, poderia ser usada a combinação do backup base de todo o banco durante um ou dois dias na semana e, durante os dias restantes, ser realizado apenas o backup dos archives dos logs. E no caso de uma recuperação do seu banco, você iria combinar o seu backup base com a sequência de logs arquivada.

Outra função importante dos archives é a construção do banco de dados standby. Os logs arquivados no banco de dados principal são enviados para o secundário e lá as transações são aplicadas para criar um espelho do banco de dados de produção. No caso de um desastre, a recuperação do seu ambiente com o banco de dados standby seria bem mais rápida. Porém, como é algo que requer um investimento maior, nem sempre ele é implementado.

A escrita nos logs de REDO é sequencial e cíclica. Como padrão temos três segmentos, a escrita é alternada entre eles. Essa alternância entre os segmentos é conhecida no PostgreSQL como xlog switchs. E sempre que um switch ocorre, é gerado o arquivamento do segmento antigo. Mas em que momento ocorre essa

· Durante o shutdown do banco de dados, ocorre o switch para que os arquivos atuais sejam arquivados.

Restore dos backups realizados

Com certeza, mais importante do que o backup é o restore. Imagine você realizar um grande investimento em infraestrutura, levar vários dias planejando sua estratégia de backup e, no dia em que houver um desastre no seu servidor de banco de dados de produção, você não conseguir realizar a recuperação de toda a informação necessária porque não havia testado antes se seus backups realmente conseguiriam recuperar as informações conforme a necessidade da empresa. Todo seu esforço foi jogado fora.

Por esta razão, além de ter um bom planejamento para os backups, você deve ter uma excelente estratégia para o restore. Tudo que citamos para o backup, é válido para o restore. Também teremos que ter um bom investimento para criação de toda a documentação, atualização, revisão dos procedimentos para realização da recuperação. Tudo isso com a intenção de que nossos backups possam ser constantemente testados e validados para que no dia que seja necessário, eles funcionem.

Além da documentação de restore usando os backups feitos para desastres, é recomendado que seja feita a documentação para recuperação dos backups históricos.

regularidade. Assim, todo o procedimento poderia ser validado com uma precisão maior.

Habilitando o arquivamento dos logs de WAL

Para criação destes exemplos foi usado o ambiente com o sistema operacional CentOS 6 e PostgreSQL 9.4. Para começar, vamos criar uma estrutura diferente do local onde está instalado nosso banco de dados. Esta nova estrutura será usada para armazenar os backups. No nosso exemplo, será criado o diretório /backup e dentro dele a pasta /archives. Como citado antes, é interessante que o /backup esteja em um disco diferente do seu servidor de banco de dados. É importante também certificar-se de que o usuário postgres tem acesso aos diretórios criados:

```
1 mkdir -p /backup/archives
2 chown postgres.postgres /backup
3 chown postgres.postgres /backup/archives
```

Após a criação dos diretórios que receberão os arquivos, vamos habilitar o arquivamento dos logs de WAL. Primeiro, iremos fazer login com o usuário postgres, logo em seguida conectar no banco de dados usando o utilitário psql para identificar onde se encontra o arquivo postgressql.conf. Para localizá-lo, usaremos o comando *show config_file*. É nele que faremos as configurações necessárias. Como mostrado na **Listagem 1**, nosso arquivo está no caminho /postgres/postgresql.conf.

Listagem 1. Localizando o arquivo postgresql.conf

Após localizar o arquivo postgresql.conf, devemos editá-lo com os valores descritos na **Listagem 2** para que o arquivamento dos logs de REDO seja habilitado:

· wal_level: determina a quantidade de informação que é armazenada no WAL. O valor padrão é *minimal*, que guarda somente a informação necessária para recuperar a base de um desastre. No nosso caso, precisaremos alterar o parâmetro para *archive* ou *hot_standby*;

· archive_mode: quando alterado para *on*, faz com que os segmentos de WAL sejam enviados para a localização de armazenamento que será indicada no parâmetro archive_command. O archive_mode não pode ser habilitado quando o wal_level está como *minimal;*

· archive_command: aqui iremos passar o comando shell que irá realizar o arquivamento dos logs de WAL. O %p será substituído pelo caminho que será armazenado e %f é alterado pelo nome do arquivo;

· max_wal_senders: número de processos que fazem o envio dos logs arquivados de WAL para os servidores standby. O padrão é 0, indicando que a replicação está desabilitada. No nosso caso iremos habilitar pois será necessário para o uso do utilitário pg_basebackup.

Listagem 2. Parâmetros para habilitar arquivamento dos logs WAL

```
01 wal_level = archive
02 archive_mode = on
```

Agora que já temos os logs de REDO sendo copiados, podemos criar nosso backup online completo da base de dados. Iremos utilizar o utilitário pg_basebackup. Ele realiza uma cópia de todos os arquivos do cluster de banco de dados PostgreSQL. Caso você tenha mais de um banco de dados nas suas instâncias, não é possível realizar o backup de apenas um banco de dados.

Este tipo de backup é feito utilizando o protocolo de replicação para conexão com o banco de dados, por isso, a realização da conexão com o banco deve ser feita por um usuário que tenha permissões de replicação. Outro pré-requisito é que o parâmetro max_wal_senders esteja com a configuração mínima necessária para que haja pelo menos uma sessão disponível durante a realização do backup.

Vamos criar um usuário com as permissões necessárias para realizar nosso backup. Usando um super usuário do banco de dados, faça login na base e execute o código a seguir:

```
1 postgres=# create role usu_bkp replication login password 'backup';
```

Após a criação do usuário com o privilégio Replication, precisamos adicionar uma nova linha ao arquivo pg_hba.conf. Essa nova linha é necessária para que nosso novo usuário possa se conectar ao banco de dados. Como este arquivo pode variar em cada instalação, usaremos o comando show hba_files para descobrir onde ele se encontra, como mostrado na **Listagem 3**.

Listagem 3. Localizando o arquivo pg_hba.conf

Feito isso, adicione a linha apresentada na **Listagem 4** ao final do arquivo pg_hba.conf. Logo em seguida, reinicie o banco de dados para que as mudanças tenham efeito.

Listagem 4. Definição de permissão e reinício do servidor

```
01 -- Adição de permissões arquivo pg_hba.conf
02 host   replication      usu_bkp       127.0.0.1/32     trust
03
04 -- Restart serviço PostgreSQL
05 pg_ctl restart
```

Depois de configurar o arquivamento, iremos realizar o backup completo de toda estrutura da base. Usaremos o utilitário pg_basebackup passando os parâmetros necessários para nossas configurações:

```
pg_basebackup -h127.0.0.1 -U usu_bkp -D /backup/`date +%d%m%Y` --xlog-
```

· -h: especifica o nome do servidor do banco de dados;

· -U: nome do usuário que irá realizar a conexão;

· -D: diretório onde será gravado o backup. Foram usadas as variáveis %d%m%y para especificar junto ao diretório a data em que está sendo gerado;

· --xlog-method: inclui todos os logs gerados durante o backup;

· --format=t: formata a saída do arquivo para o padrão .tar;

· -z: habilita a compressão gzip no arquivo.

Realizando o Restore e o Recovery

Vamos entender como realizar o restore dos backups bases e, logo em seguida, a recuperação aplicando as sequências WAL arquivadas. Lembrando que nesse ambiente usaremos o mesmo servidor em que o backup foi realizado. Se quiser realizar o procedimento em outro servidor, será necessário que o software do PostgreSQL esteja instalado.

Aqui teremos dois cenários de recuperação:

1. No primeiro faremos o restore do backup base e depois a recuperação completa até o momento da falha;

2. No segundo faremos o restore do backup base e depois a recuperação incompleta para um determinado ponto no tempo.

O processo de restauração do backup base é igual para os dois cenários. Usaremos a **Listagem 5** como base para os comandos descritos neste processo. O primeiro passo para realizar o restore é parar todos os processos relacionados ao PostgreSQL (linha 01). Não é obrigatório, mas se for possível, é interessante realizar um backup da atual posição do banco antes que seja realizado o restore. No nosso caso, será criado um diretório chamado de /bkp_antes_restore onde faremos a cópia de todo o

com a data da realização do backup. Iremos acessar a pasta 31102015, data em que foi realizado o backup que queremos usar para o restore (linha 05). Dentro dela teremos o arquivo chamado base.tar.gz. O arquivo está nesse formato porque foi compactado durante o backup. Este arquivo deve ser descompactado dentro do diretório base do nosso banco de dados (linha 06).

Listagem 5. Restore do backup base

```
01 pg_ctl stop
02 mkdir -p /bkp_antes_restore
03 chown postgres.postgres /bkp_antes_restore
04 cp -r /postgres/* /bkp_antes_retore/
05 cd /backup/31102015/
06 tar -xvf base.tar.gz -C /postgres/
```

Cenário 1

Já temos nosso backup base restaurado, agora precisamos realizar a recuperação completa do nosso banco de dados. Para isso, precisamos criar o arquivo *recovery.conf*. Este arquivo passa ao PostgreSQL os parâmetros necessários para o recovery. Nesse cenário iremos indicar apenas a localização das sequências arquivadas dos logs de WAL e durante o processo de início do banco de dados serão lidos e aplicados todos os logs de REDO para que o banco seja recuperado até a última sequência gerada. Observe como proceder no código a seguir:

```
-- Recuperação completa
```

recuperação conforme solicitado no arquivo.

Cenário 2

A diferença deste cenário é que aqui iremos realizar a recuperação para um determinado período no tempo e não necessariamente até a última sequência arquivada. Assim como no primeiro cenário, após a restauração do backup base, iremos criar o arquivo *recovery.conf,* porém aqui serão necessárias algumas linhas a mais no arquivo, conforme mostrado a seguir:

```
recovery_target_time='2015-10-31 13:30:42'
recovery_target_inclusive = true
restore_command = 'cp /backup/archives/%f %p'
```

· recovery_targer_time: período que queremos recuperar o banco de dados;

· recovery_target_inclusive: indica que a recuperação será realizada até o momento descrito no comando. Se a opção escolhida fosse false, então a recuperação iria ocorrer até um período menor do que o indicado no comando;

· restore_command: indica o local onde estão os arquivos.

Existem outras configurações durante a recuperação. Recomendamos a leitura da documentação para se aprofundar no assunto (veja seção **Links**).

Sabendo da importância de ser manter boas estratégias de backup e de recuperação dos dados, a implementação de uma boa estratégia deve ser encarada com seriedade

Links

SQL Dump – Backup lógico

http://www.postgresql.org/docs/8.3/static/backup-dump.html

Standby Servers - Banco de dados standby (replicação)

http://www.postgresql.org/docs/9.4/static/warm-standby.html

Recovery Target Settings - Configurações usadas durante o recover

http://www.postgresql.org/docs/9.1/static/recovery-target-settings.html

Por **Thiago**
Em 2016