INSTALANDO SVNSERVER E USANDO O TORTOISESVN

Tutorial

Igor Muzetti Pereira
igormuzetti@yahoo.com.br

Tiago Garcia de Senna Carneiro
tiago@iceb.ufop.br

Departamento de Computação
Universidade Federal de Ouro Preto
Laboratório Associado INPE/UFOP para Modelagem e Simulação de Sistemas Terrestres

---

Primeiramente devemos instalar os pacotes subversion e subversion-server.

1.Vá ao console e como root digite:

 # zypper install subversion
 # zypper install subversion-server

O servidor que está sendo usado neste caso de uso foi o **svnserve** este programa é um servidor leve, capaz de falar com clientes via TCP/IP usando um protocolo específico e robusto. Escolhemos este para nossa migração ao SVN pelo fato de ter sido explicitamente desenvolvido para o Subversion e de manter informações de estado (diferentemente do HTTP), este seu protocolo permite operações de rede significativamente mais rápidas—ainda que ao custo de alguns recursos. Este protocolo só entende autenticação do tipo CRAM-MD5, não possui recursos de log, nem de navegação web nos repositórios, e não tem opção de criptografar o tráfego de rede. Mas é, no entanto, extremamente fácil de configurar e é quase sempre a melhor opção para pequenas equipes que ainda estão iniciando com o Subversion. A outra opção seria o servidor Apache. A máquina que foi usada no TerraLAB para ser nosso servidor foi a máquina Espinhaço(IP:192.168.0.215), o INPE cadastrou este IP e aceita conexões apenas a partir dele para fazer o synchronize, este é um comando que será explicado mais adiante neste tutorial.

2. Agora iremos criar um repositório, em console como root informe:

 # mkdir -p /home/terralab
 # svnadmin create - -fs-type fsfs /home/terralab/repositorio

Este caminho para o repositório foi escolhido arbitrariamente, poderia ser qualquer outro.
Escolhemos a base de dados do repositório sendo a *fsfs* - uma implementação de sistema de arquivos versionado que usa diretamente o sistema de arquivos nativo do Sistema Operacional - ao invés de uma biblioteca de banco de dados ou outra camada de abstração - para armazenar os dados ,o FSFS oferece um pouco mais de flexibilidade em termos de seus cenários de implantação, por isso os novos FSFS estão sendo o novo padrão do Subversion. O outro tipo de base de dados seria o Berkeley DB.

3. No diretório */home/terralab/repositorio/conf*, ou seja, dentro do diretório */conf/* que fica dentro do seu repositório, vá até *svnserve.conf.*

Na seção [general] em:

 # password-db = passwd

Descomente esta linha abaixo para usar o arquivo de senha padrão(/srv/svn/repositorio/passwd).

Nesta seção [users], informe o nome de usuário e senha:

[users]
nomedeusuario = senha

4. Agora em:

 # realm = My First Repository

Descomente esta linha, e escolha um nome para o domínio de autenticação, por exemplo:

realm = /home/terralab/repositorio

5. Nas linhas:

 # anon-access = none
 # auth-access = write

Também descomente-as, com isto você estará especificando que usuário anônimos não terão direito de checar o repositório(*none*) e usuário cadastrados terão acesso à escrita no repositório, se quiser colocar que usuários anônimos tem o direito apenas de leitura no repositório, é só colocar ao invés de *none* trocar para *read***,** conforme suas modificações essa parte deverá ficar assim:

#anon-access = read (ou esta opção descomentado-a ou a de baixo)
anon-access = none
auth-access = write

6. Descomente a linha abaixo para usar a autorização de arquivo padrão(/home/terralab/repositorio/conf):

 # authz-db = authz

Dentro dele informe:

[/]
nomeusuario= rw

Este usuário tem completo acesso de leitura e escrita no repositório. Quaisquer outros usuários têm seu acesso a este repositório bloqueado.

Agora para começar a usar o SVN, devemos como root colocar o SVN para escutar, informando no console:

#svnserve -d

Pronto, agora da máquina cliente podemos começar a usar o TortoiseSVN.

Agora faremos um espelhamento do nosso repositório com um que iremos criar no INPE, este processo será feito tendo um “repositório remoto espelho” que estará em sincronização com o nosso repositório original aqui do TerraLAB, vamos programar através de um script um horário em que o nosso repositório entrará em contato com o espelho e fará a sincronização, ficando o espelho igual ao original mas o acesso a ele pelo pessoal do INPE será apenas para leitura, ou seja sem acesso a escrita.

Na máquina em que ficará o repositório espelho(neste caso no INPE), teremos que executar os seguintes passos:

```
# crie um novo repositorio:
svnadmin create /CAMINHO/DESTINO
# crie um hook script pre-commit-hook vazio:
echo '#!/bin/bash' > /CAMINHO/DESTINO/hooks/pre-revprop-change
```

 # torne o script executável:

chmod +x /CAMINHO/DESTINO/hooks/pre-revprop-change

Assim que o repositório destino estiver criado, basta inicializar a conexão entre o repositório origem e destino, e sincronizar:

inicializar a conexão dos repositorios:
svnsync init https://svn.dpi.inpe.br/TerraME svn://192.168.0.215/home/terralab/repositorio --source-username usuarioSvn --sync-username usuarioEspelho --source-password senhaUsuarioSvn --sync-password senhaEspelho
sincronizar:
svnsync sync https://svn.dpi.inpe.br/TerraME --source-username usuarioSvn --sync-username usuarioEspelho --source-password senhaUsuarioSvn --sync-password senhaEspelho

Com isto feito, os dois repositórios estarão sincronizados.
Para uma melhor maneira de dar manutenção em nossos repositórios, foi decidido fazer um script para a sincronização, para que em uma determinada hora do dia eles se sincronizem automaticamente através do script.

O nosso script ficou em: /root/ e se chama: **syncinpe.sh**, ficando: /root/syncinpe.sh
Moldamos ele da seguinte maneira:

#!/bin/bash
#fazendo a sincronização com o repositório espelho:

svnsync sync https://svn.dpi.inpe.br/TerraME --source-username usuarioSvn --sync-username usuarioEspelho --source-password senhaUsuarioSvn --sync-password senhaEspelho

#Fim do script

Agora, devemos transformá-lo em executável, no console:

#chmod +x /root/syncinpe.sh

Vamos configurar o crontab, em /etc/crontab, colocamos a chamada para o script.

00 02 * * * /root/syncinpe.sh

Este significa que todo dia às 2 horas da manhã o sistema realizará a sincronização.
Uma observação importante é que antes de se fazer esta sincronização pelo script, já foi feita pelo menos uma vez a inicialização(comando init) dos repositórios, como já foi mostrado logo acima, antes do comando sync.
Pronto. Salve o arquivo e reinicie o cron para que a rotina funcione:

#/etc/init.d/cron restart

OPERAÇÕES BÁSICAS DE USO DO TORTOISESVN

O TortoiseSVN é um cliente para Windows do software de gerenciamento de mudanças Subversion. O site da ferramenta é http://tortoisesvn.tigris.org/

Abaixo segue um resumo dos principais comandos utilizados no TortoiseSVN:

Importar (Import):

Envia o conteúdo de uma pasta local para um repositório no servidor SVN. Normalmente é utilizado pelo responsável pelo repositório ou projeto para preparar o servidor para o trabalho em equipe. Antes de importar você deve remover todos os arquivos que não são necessários para construir/compilar o projeto (por exemplo arquivos temporários, arquivos objeto, etC.

Para importar, selecione a pasta de nível superior da estrutura de diretório do seu projeto no Windows Explorer e clicando com o botão da direita selecione TortoiseSVN->Importar. Digite a URL do repositório para onde quer enviar seu projeto.

Obter (Checkout):

Obtém do repositório (no servidor SVN) uma cópia de trabalho local do projeto. Selecione uma pasta no Windows Explorer onde você quer colocar sua cópia de trabalho. Clique com o botão da direita no menu de contexto e selecione TortoiseSVN->Obter.

Submeter (Commit):

Enviar as modificações que você realizou na sua cópia de trabalho é conhecido como submeter (commit). Antes de você submeter, você deve ter certeza que sua cópia de trabalho está atualizada. Você deve utilizar a opção TortoiseSVN->Atualizar (Update) ou TortoiseSVN->Verificar alterações (Check for Modifications). Se a sua cópia de trabalho está atualizada e não há conflitos, você está pronto para submeter suas modificações para o repositório. Selecione qualquer arquivo e/ou pasta que você quer submeter e selecione TortoiseSVN->Submeter (Commit) no menu de contexto do Windows Explorer. A caixa de diálogo de submeter mostrará todos os arquivos modificados incluindo arquivos adicionados, deletados e sem controle de versão. Se não quiser submeter um arquivo desmarque a caixa correspondente. Se quiser adicionar um arquivo sem controle de versão selecione a caixa correspondente para submetê-lo. Ao submeter, você deve(boa prática) digitar uma mensagem de histórico/log que descreve as modificações que você realizou. Este é um grande benefício do SVN e ajudará em outras ocasiões se bem utilizado.

Atualizar (Update):

Periodicamente você deve se certificar que as modificações realizadas por outros desenvolvedores sejam incorporadas na sua cópia de trabalho local. Este processo de baixar as modificações do servidor para sua cópia local é chamado de atualização. Esta atualização pode ser feita com arquivos individuais, um conjunto de arquivos ou recursivamente em toda hierarquia de diretório. Para atualizar, selecione os arquivos e/ou pastas que você quer e clique com o botão da direita selecionando TortoiseSVN->Atualizar (Update) no menu de contexto. Uma janela exibirá o progresso. As modificações realizadas por outros serão mescladas aos seus arquivos, mantendo quaisquer modificações que você tenha realizado nos mesmos arquivos num formato próprio para resolução de conflitos. Mais sobre isso abaixo. O repositório não é afetado por uma atualização pois os arquivos e pastas saem do servidor em direção a sua cópia de trabalho local. A atualização padrão não tem opções e apenas atualiza sua cópia de trabalho para a revisão HEAD que é a mais recente. Se você quer mais controle sobre o processo de atualização pode utilizar a opção TortoiseSVN->Atualizar para Revisão (Update to Revision). Isto permite que você atualize sua cópia de trabalho para uma revisão específica e não necessariamente a mais recente.

Comparar (Diff):

Um dos requisitos mais comuns no desenvolvimento de projeto é ver o que foi alterado. Você pode querer ver as diferenças entre duas revisões do mesmo arquivo ou a diferença entre dois arquivos separados. Para ver quais modificações você fez na sua cópia de trabalho simplesmente utilize o menu de contexto do Windows Explorer e selecione TortoiseSVN->Comparar (Diff). Se você quiser ver a diferença entre uma revisão em particular e sua cópia de trabalho utilize TortoiseSVN->Comparar com revisão anterior (Diff with previous version).

Resolvendo Conflitos

Possivelmente haverá conflitos quando você atualizar/mesclar seus arquivos com o repositório. Existem dois tipos de conflitos:

- conflitos de arquivos
- conflitos de árvore

Um **conflito de arquivo** ocorre quando dois (ou mais) desenvolvedores modificaram as mesmas linhas de um arquivo. Quando isso ocorre, você deve abrir o arquivo em questão e procurar pelas linhas que começam com <<<<<<<. A área de conflito é marcada assim:

<<<<<<<< nomedoarquivo
suas modificações

código mesclado do repositório
>>>>>>> revisão

Além disso para cada arquivo com conflito Subversion coloca três arquivos no seu diretório:

nomedoarquivo.ext.mine

Este é seu arquivo como existia na sua cópia de trabalho antes de você efetuar uma atualização com o servidor, ou seja, sem os marcadores de conflito. Este arquivo possui suas últimas modificações.

nomedoarquivo.ext.rREVANTIGA

Este é o arquivo base da revisão, antes de você atualizar sua cópia de trabalho, ou seja, este arquivo que você obteve antes de efetuar suas modificações.

nomedoarquivo.ext.rREVNOVA

Este é o arquivo que seu cliente SVN acabou de receber do servidor quanto você efetuou uma atualização da sua cópia de trabalho. Este arquivo corresponde à revisão HEAD do repositório, ou seja, a revisão mais recente que possui as modificações recentes de outros desenvolvedores.

Você pode lançar um ferramenta externa de edição de conflito com a opção TortoiseSVN->Editar Conflitos (Edit Conflicts) ou você pode manualmente resolver o conflito. Você deve decidir como o código deve ser, fazer as modificações necessárias e salvar no arquivo .mine. Depois que fizer isto, execute o comando TortoiseSVN->Resolvido (Resolved) e submeta suas modificações para o repositório. Note que o comando Resolvido não resolve o conflito, ele apenas remove os arquivos nomedoarquivo.ext.mine, nomedoarquivo.ext.rRAVANTIGA e nomedoarquivo.ext.rREVNOVA permitindo que você execute o TortoiseSVN->Submeter (Commit) para aí sim, subir para o repositório suas modificações. Se você tiver problemas com arquivos binários, Subversion não tentará mesclar os arquivos ele próprio. O arquivo local permanecerá o mesmo e você terá os arquivos da revisão antiga e da nova como o exemplo acima. Se você quiser descartar suas modificações e manter a versão do repositório, então apenas utilize o commando Reverter (Revert) do menu. Se você quiser manter sua versão e sobrepor a versão do repositório, utilize o comando Resolvido (Resolved) e submeta sua versão. Você pode utilizar o comando Resolvido (Resolved) para vários arquivos se você clicar na pasta superior e selecionar TortoiseSVN->Resolvido. Isto exibirá uma caixa de diálogo listando todos os arquivos conflitantes naquela pasta e você poderá selecionar quais estão resolvidos.

Antes de utilizar o comando Resolvido, verifique se você mesclou corretamente suas modificações com as de outros desenvolvedores pois ao efetuar em seguida um Submeter, suas modificações serão salvas sobre o que já existe no repositório.

Um **conflito de árvore** ocorre quando um desenvolvedor move/renomeia/deleta um arquivo ou pasta, que outro desenvolvedor também tenha movido/renomeado/deletado ou apenas modificado. Existem várias situações que podem resultar num conflito de árvore e requerem formas diferentes de resolver.

Quando um arquivo é deletado localmente no Subversion, o arquivo também é deletado do sistema de arquivos local, então mesmo que seja parte de um conflito de árvore ele não pode exibir um estado de conflito e você não pode clicar com o botão da direita nele para resolver o conflito. Ao invés disso, utilize Verificar Modificações (Check for Modifications) para acessar a opção Editar conflitos (Edit conflicts).

O TortoiseSVN pode ajudar a encontrar o local certo para mesclar as modificações mas pode haver trabalho adicional necessário para acertar os conflitos. Lembre-se que depois de uma atualização a BASE de trabalho terá sempre a revisão de cada item como era no repositório no momento da atualização. Se você reverter uma modificação depois de atualizar, ela voltará ao estado do repositório e não como era quando você começou a fazer modificações locais.

MÁQUINAS USADAS NESTE PROCEDIMENTO:

SERVIDOR:

ESPINHAÇO

IP: 192.168.0.215

Sistema Operacional: OpenSUSE 11.1 – KDE 4.1.3

Processador: Intel Core 2 CPU 6400 – 2.13Ghz

Memória RAM: 4Gb

CLIENTE:

MANTIQUEIRA

IP: 192.168.0.4

Sistema Operacional: Windows XP

Processador: Intel Core 2 CPU 6400 – 2.13Ghz

Memória RAM: 4Gb

SOFTWARES UTILIZADOS E SUAS VERSÕES:

- Subversion – 1.5.7
- Servidor svnserve – 1.5.7
- TortoiseSVN – 1.6.5

Arquivo ***svnserve.conf:***

```
### This file controls the configuration of the svnserve daemon, if you
### use it to allow access to this repository.  (If you only allow
### access through http: and/or file: URLs, then this file is
### irrelevant.)

### Visit http://subversion.tigris.org/ for more information.

[general]
### These options control access to the repository for unauthenticated
### and authenticated users.  Valid values are "write", "read",
### and "none".  The sample settings below are the defaults.
anon-access=read
auth-access=write
### The password-db option controls the location of the password
### database file.  Unless you specify a path starting with a /,
### the file's location is relative to the directory containing
### this configuration file.
### If SASL is enabled (see below), this file will NOT be used.
### Uncomment the line below to use the default password file.
password-db = passwd
#/home/terralab/repositorio/=usuariosenha
### The authz-db option controls the location of the authorization
### rules for path-based access control.  Unless you specify a path
### starting with a /, the file's location is relative to the the
### directory containing this file.  If you don't specify an
### authz-db, no path-based access control is done.
### Uncomment the line below to use the default authorization file.
authz-db = authz
#/home/terralab/repositorio/=regras
### This option specifies the authentication realm of the repository.
### If two repositories have the same authentication realm, they should
### have the same password database, and vice versa.  The default realm
### is repository's uuid.
realm =  /home/terralab/repositorio

[sasl]
### This option specifies whether you want to use the Cyrus SASL
### library for authentication. Default is false.
### This section will be ignored if svnserve is not built with Cyrus
### SASL support; to check, run 'svnserve --version' and look for a line
### reading 'Cyrus SASL authentication is available.'
# use-sasl = true
### These options specify the desired strength of the security layer
### that you want SASL to provide. 0 means no encryption, 1 means
### integrity-checking only, values larger than 1 are correlated
### to the effective key length for encryption (e.g. 128 means 128-bit
### encryption). The values below are the defaults.
# min-encryption = 0
# max-encryption = 256
```

Script para baixar a versão atual dos dois repositórios e ver se elas são as mesmas:

```
#!/bin/bash

svn checkout svn://192.168.0.215/home/terralab/repositorio
svn checkout https://svn.dpi.inpe.br/TerraME
diff -r repositorio TerraME


#Fim do script
```