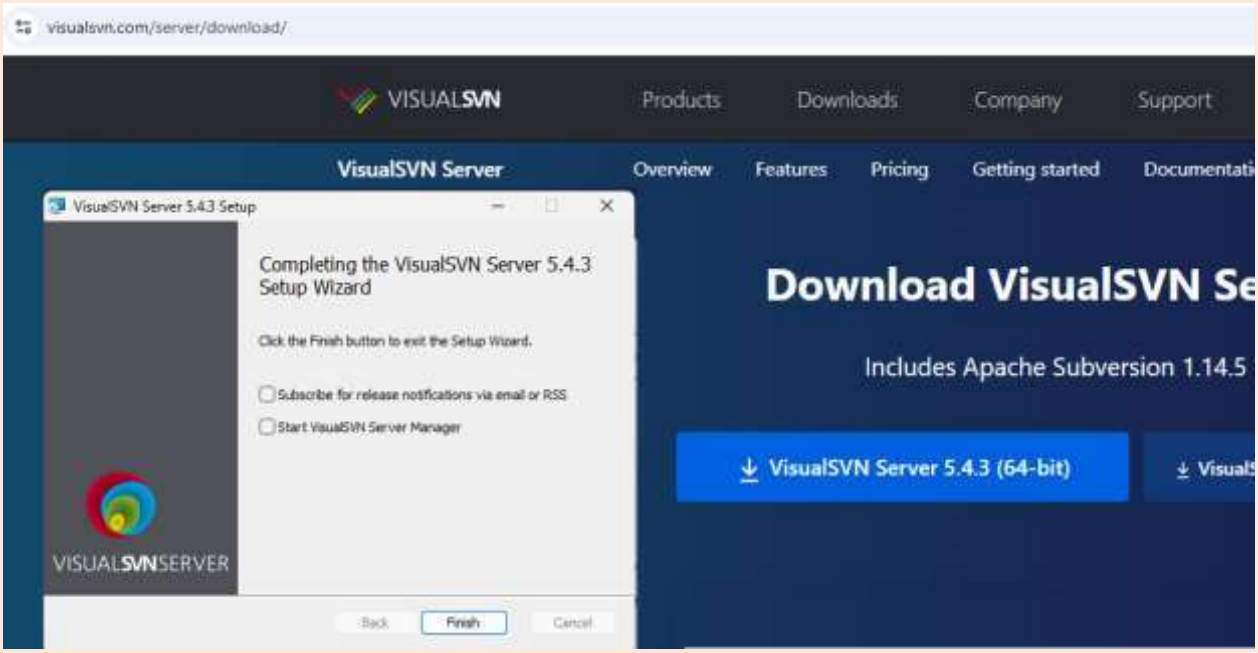
SVN to Git migration - Tutorial

Introduction

I have a project in SVN repository which I want to migrate to Git.

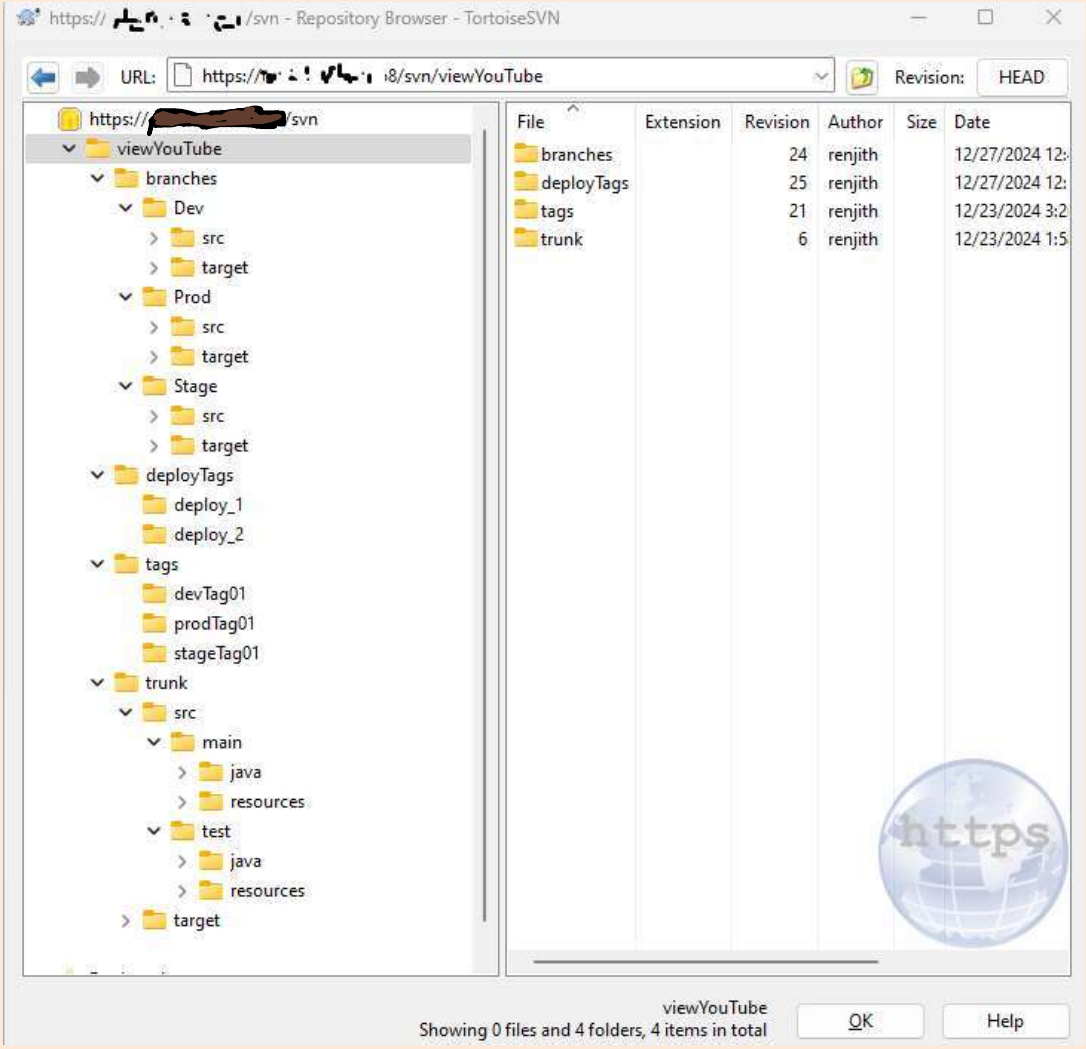
Note:
For the sake of clarity, in the past I have installed SVN server in my Windows host by downloading the Visual SVN Server. Currently I have SVN Server 5.4.3.



The SVN project I have (viewYouTube) has a proper SVN repository layout – meaning, this project has a trunk, few branches and few tags. I also have custom tags in this repository. My goal is to move everything into my GitHub account.

SVN to Git migration - Tutorial

Below screenshot is the Tortorise SVN Repository browser displaying the layout of my project (viewYouTube).



viewYouTube is a Java project. The deployTag is a custom tag that I created to record some special releases.

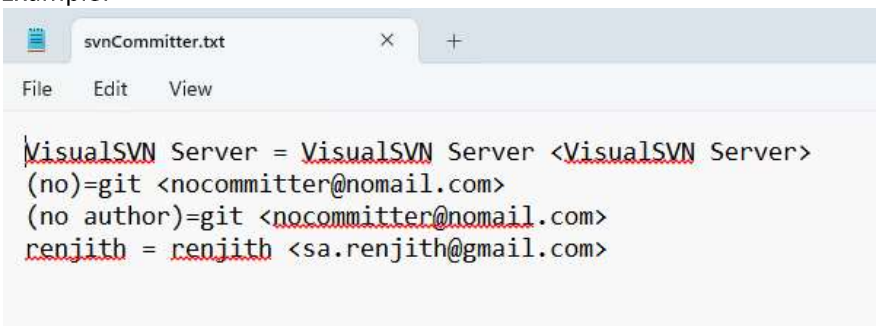
Migrating svn project "viewYouTube" to Git

I have installed git for Windows from https://git-scm.com/downloads, and I am using git bash to do this entire migration.

Step 1: Create SVN committer's list
First step in this process is to create a file that contains all the user names that has made at least one commit into the project. I am going to name this file svnCommitter.txt. Each row in this file must be in the format of: Username=user's name <username@mailid>

SVN to Git migration - Tutorial

Example:



I always include the first three lines in my committer list to avoid failure such as committer name not found/configured.

SVN Committer List

The script/command I use to create the svnCommitter.txt file is below (I use git bash terminal to run this command):

```
#svn repository URL
SVN_URL="https://mysvnserver.com/svn/viewYouTube"

#output file with authors list
OUTPUT_FILE="svnCommitter.txt"

#temperory file to store output
TEMP_FILE=$(mktemp)

echo "Fetching SVN authors from $SVN_URL ..."
svn log "$SVN_URL" --quiet | grep -E '^r[0-9]+' | awk '{print $3}' | sort | uniq > "$TEMP_FILE"

echo "generating authors file"
> "$OUTPUT_FILE" #empty/create file

while read -r AUTHOR; do
        if [ -n "$AUTHOR" ]; then
                echo "$AUTHOR = $AUTHOR <${AUTHOR}@emaildomain.com>" >> "$OUTPUT_FILE"
        fi
done < "$TEMP_FILE"

rm -f "$TEMP_FILE"

echo "Author file generated successfully"
```

Initialize Git repository for the SVN

Now is the time to initialize svn replacement repository in git.
First step is to create a folder that match with the SVN project name. In my example, this is viewYouTube.

SVN to Git migration - Tutorial

```
renjith@DESKTOP MINGW64 /c/Renjith/work
$ mkdir viewYouTube && cd viewYouTube

renjith@DESKTOP MINGW64 /c/Renjith/work/viewYouTube
```

The script/commands I use to initialize is given below. The idea is to initialize git-svn, link the svnCommitter.txt file, append the additional tag (deployTag) and then do a svn fetch.

```
#svn repository URL
SVN_URL="https://mysvnserver.com/svn/viewYouTube"

#svn commiter file that we created in previous step and that is in the parent folder
COMMITTER_FILE="../svnCommitter.txt"

#git svn init
git svn init --trunk=trunk --branches=branches/* --tags=tags/* --no-metadata "$SVN_URL"

# link the svn committer file
git config svn.authorsfile ../svnCommitter.txt

#(optional) include the additional tag "deployTag" in the migration.
#We are editing the .git/config file and adding the stanza
git config --add  svn-remote.svn.tags deployTags/*:refs/remotes/deployTags/*

#fetch everything from svn
git svn fetch –all
```

```
renjith@DESKTOP MINGW64 /c/Renjith/work/viewYouTube (master)
$ SVN_URL="https://  mysvnserver.com  /svn/viewYouTube"

renjith@DESKTOP MINGW64 /c/Renjith/work/viewYouTube (master)
$ git svn init --trunk=trunk --branches=branches/* --tags=tags/* --no-metadata $SVN_URL

renjith@DESKTOP MINGW64 /c/Renjith/work/viewYouTube (master)
$ git config svn.authorsfile ../svnCommitter.txt

renjith@DESKTOP-N026408 MINGW64 /c/Renjith/work/viewYouTube (master)
$ git config --add svn-remote.svn.tags  deployTags/*:refs/remotes/deployTags/*

renjith@DESKTOP MINGW64 /c/Renjith/work/viewYouTube (master)
$ git svn fetch --all
```

Depending on the size of your svn repository, the git svn fetch --all takes some time to complete the cloning. Upon completion, you will see something like below

SVN to Git migration - Tutorial

```
MINGW64:/c/Renjith/work/viewYouTube
r21 = bdd48565ac57a6c7dda2260b4f6de3172283736b (refs/remotes/git-svn)
        M       branches/Dev/environment.txt
r22 = b17dce935b19743c970d2b8e1d8094453196ebd2 (refs/remotes/git-svn)
        A       deployTags/deploy_1/environment.txt
r23 = eddd694401bc4a4d3735bf9d4f881b76fcaf8566 (refs/remotes/git-svn)
        M       branches/Dev/environment.txt
r24 = e0c40889c1064824f7e4e6185f06ec30d08c6d61 (refs/remotes/git-svn)
        A       deployTags/deploy_2/environment.txt
r25 = 5dc75c560148efe52920098173502c71671ea47e (refs/remotes/git-svn)
Checked out HEAD:
  https://  mysvnserver.com  /svn/viewYouTube r25
creating empty directory: branches/Dev/src/test/java
creating empty directory: branches/Dev/src/test/resources
creating empty directory: branches/Dev/target/test-classes
creating empty directory: branches/Prod/src/test/java
creating empty directory: branches/Prod/src/test/resources
creating empty directory: branches/Prod/target/test-classes
creating empty directory: branches/Stage/src/test/java
creating empty directory: branches/Stage/src/test/resources
creating empty directory: branches/Stage/target/test-classes
creating empty directory: trunk/src/test/java
creating empty directory: trunk/src/test/resources
creating empty directory: trunk/target/test-classes

renjith@DESKTOP MINGW64 /c/Renjith/work/viewYouTube (master)
$
```

Validation

To confirm whether you fetched the svn project, check its branches.

```
renjith@DESKTOP MINGW64 /c/Renjith/work/viewYouTube (master)
$ git branch -a
* master
  remotes/deployTags/deploy_1
  remotes/deployTags/deploy_2
  remotes/origin/Dev
  remotes/origin/Prod
  remotes/origin/Stage
  remotes/origin/tags/devTag01
  remotes/origin/tags/prodTag01
  remotes/origin/tags/stageTag01
  remotes/origin/trunk
```

Note that, here the tags and deployTags are reterived as branches. Now we have some more steps to do:

1. Convert tag branches (remotes/origin/tags, remotes/deployTags) into proper svn tags.
2. Convert branches (remotes/origin/dev, remotes/origin/prod, remotes/origin/stage) to git local branches.
3. Handle the trunk (remotes/origin/trunk).

SVN to Git migration - Tutorial

Convert tag branches

I use below script to convert remotes/origin/tags into git tags.

```
for t in `git branch -a | grep 'tags/' | sed s_remotes/origin/tags/__`; do
                        git tag $t origin/tags/$t
                        git branch -d -r origin/tags/$t
done
```

```
renjith@DESKTOP MINGW64 /c/Renjith/work/viewYouTube (master)
$ git branch -a | grep 'tags/'
  remotes/origin/tags/devTag01
  remotes/origin/tags/prodTag01
  remotes/origin/tags/stageTag01

renjith@DESKTOP MINGW64 /c/Renjith/work/viewYouTube (master)
$ for t in `git branch -a | grep 'tags/' | sed s_remotes/origin/tags/__`; do
        git tag $t origin/tags/$t
        git branch -d -r origin/tags/$t
done
Deleted remote-tracking branch origin/tags/devTag01 (was 7b8ac0a).
Deleted remote-tracking branch origin/tags/prodTag01 (was 6b504e6).
Deleted remote-tracking branch origin/tags/stageTag01 (was 12bc506).

renjith@DESKTOP MINGW64 /c/Renjith/work/viewYouTube (master)
$ git branch -a
* master
  remotes/deployTags/deploy_1
  remotes/deployTags/deploy_2
  remotes/origin/Dev
  remotes/origin/Prod
  remotes/origin/Stage
  remotes/origin/trunk

renjith@DESKTOP MINGW64 /c/Renjith/work/viewYouTube (master)
$ git tag -l
devTag01
prodTag01
stageTag01

renjith@DESKTOP MINGW64 /c/Renjith/work/viewYouTube (master)
```

Note that, now we got git tags and also deleted the “tag” branches.

Similarly convert the deployTag branches into git tags and delete those deployTag branches.

```
renjith@DESKTOP MINGW64 /c/Renjith/work/viewYouTube (master)
$ git branch -a
* master
  remotes/deployTags/deploy_1
  remotes/deployTags/deploy_2
  remotes/origin/Dev
  remotes/origin/Prod
  remotes/origin/Stage
  remotes/origin/trunk

renjith@DESKTOP MINGW64 /c/Renjith/work/viewYouTube (master)
$ for dT in `git branch -a | grep 'deployTags/' | sed s_remotes/deployTags/__`; do
    git tag $dT deployTags/$dT; git branch -d -r deployTags/$dT; done
Deleted remote-tracking branch deployTags/deploy_1 (was 7664553).
Deleted remote-tracking branch deployTags/deploy_2 (was fa99810).

renjith@DESKTOP MINGW64 /c/Renjith/work/viewYouTube (master)
$ git branch -a
* master
  remotes/origin/Dev
  remotes/origin/Prod
  remotes/origin/Stage
  remotes/origin/trunk

renjith@DESKTOP MINGW64 /c/Renjith/work/viewYouTube (master)
$ git tag -l
deploy_1
deploy_2
devTag01
prodTag01
stageTag01
```

Convert branches to git local branches

We have three branches in viewYouTube project such as remotes/origin/dev, remotes/origin/prod, remotes/origin/stage. Let's create corresponding git local branches, map them and delete the remotes/origin branch.

```
#command
for b in `git branch -r | sed s_origin/__`; do
      git branch $b origin/$b
      git branch -D -r origin/$b;
done
```

```
renjith@DESKTOP MINGW64 /c/Renjith/work/viewYouTube (master)
$ git branch -a
* master
  remotes/origin/Dev
  remotes/origin/Prod
  remotes/origin/Stage
  remotes/origin/trunk

renjith@DESKTOP MINGW64 /c/Renjith/work/viewYouTube (master)
$ for b in `git branch -r | sed s_origin/__ ; do
        git branch $b origin/$b
        git branch -D -r origin/$b;
done
Deleted remote-tracking branch origin/Dev (was 0d68e74).
Deleted remote-tracking branch origin/Prod (was 53ade5a).
Deleted remote-tracking branch origin/Stage (was 15c65a7).
Deleted remote-tracking branch origin/trunk (was 61c1c27).

renjith@DESKTOP MINGW64 /c/Renjith/work/viewYouTube (master)
$ git branch -a
  Dev
  Prod
  Stage
* master
  trunk
```

Handling the trunk branch

There is no conversion from trunk branch to master branch is needed because the master branch is a copy of the trunk branch. All we need is to delete the trunk branch as it is now a duplicate branch.

```
#delete trunk branch
git branch –d trunk
```

```
$ git branch -d trunk
Deleted branch trunk (was 61c1c27).
```

There we have the git repository that is a replica of SVN repository.

Now we can create a remote repository → link the remote repository with this local repository and push all the changes into remote git repository.

SVN to Git migration - Tutorial

Create remote repository and push local Git to remote

There are multiple ways to push the local git changes into remote. The approach I am explaining below will require a Personal Access Token of your remote repository.

The approach here is, get the namespace Id of your git. Using this namespace id create a remote repository though a cURL POST request. Then push your changes to remote.

```
GITLAB_URL=" https://github.com/0x218"
ACCESS_TOKEN="enter_your_access_token_genereated_in_git"
GROUP_PATH="enter_your_git_path_where_you_will_push_REPLACE_THIS_WITH_YOUR_GIT_PATH_
WHERE_YOU_WILL_PUSH_THE_CODE_TO"

#Get namespace Id
NAMESPACE_ID=$(curl -s --header "PRIVATE-TOKEN: $ACCESS_TOKEN"
"$GITLAB_URL/api/v4/groups?search=$(basename $GROUP_PATH)" | jq ".[] |
select(.full_path==\"$GROUP_PATH\") | .id")

#Create remote repository
PROJECT_RESPONSE=$(curl -s --request POST --header "PRIVATE-TOKEN: $ACCESS_TOKEN"\
                  --data "name=$PROJECT_NAME&namespace_id=$NAMESPACE_ID&visibility=private"\
                  "$GITLAB_URL/api/v4/projects")

#Get SSH URL
SSH_PROJECT_URL=$(echo $PROJECT_RESPONSE | jq -r '.ssh_url_to_repo')

#Link remote URL to the local repo
git remote add origin "$SSH_PROJECT_URL"

#Push all to remote
git push --set-upstream --all origin

#Push tags
git push –tags
```