Module 1: Getting Started with Git

Demo 4 : How to use git and its basic helplines with configuration

Running Git on Client Side

Problem Statement:

How we can run Git on the client-side, what are the configuration steps, and run all basic commands Here

Solution:

Step 1: In this demo, we are going through how to use Git as a client when your project is hosted in another system. The Git base command for all bridging commands from Subversion is git-svn.

For this, you need an SVN repository that you have written access to demonstrate this functionality. You must make a writeable copy of the SVN test repository to copy these examples. You can use the tool called svnsync that comes with Subversion to do this easily.

To proceed, a new local Subversion repository must be created first:

```
$ mkdir C:/Git/test-svn
$ svnadmin create /C:/Git/test-svn
```

Step 2: Then, enable all users to change rev-props – the easy way is to add a pre-revprop-change script that always exits 0:

```
$ cat /C:/Git/test-svn/revprop-change
#!/bin/sh
echo" hello world";

exit 0;
$ chmod +x 777 /C:/Git/test-svn/revprop-change
```

Step 3: You can now sync this project to your local machine by calling svnsync init with the to and from repositories.

```
$ svnsync init file:///C:/Git/test-svn /revpro-change
 https://your-svn.example.org/svn/
```

Step 4: This will set up the properties to run the sync. Now we can clone the code by running the command svnsync sync .

```
 $ svnsync sync file:///C:/Git/test-
 svn/revpro-change Committed revision 1.
Copied properties for revision 1. Transmitting
file data ............................[...] Committed
revision 2.
Copied properties for revision 2. […]
```

It can only take several minutes to this operation, it takes nearly an hour to copy the original repository to a remote repository, even if there are less than 100 commits. Subversion is going to clone a revision at once and then push it back to another repository. it is an inefficient way, but it is the only easy way to do this.

Step 5: You can now make workflow with a Subversion repository to which you have to write access. You will start with the command git svn clone that will import a complete Subversion repository into your local Git repository. Remember that you should replace the file:/C:/git/test-svn/revpro-change here with the URL of your Subversion repository if you are importing from a host real Subversion repository:

```
 $ git svn clone file:///C:/Git/test-svn/revpro-change -T trunk -b
 branch -t tags Initialized an empty Git repository in
 /private/C:/Git/progit/test-svn/.git/
 r1 = dcbfb585291860124cc2e8cc616cded42624897125 (refs/remotes/origins/trnk)
    ^
    ^
    ^
    A
         java/src/test/java/com/google/protobuf/WireFor
 …
 r75 = 55256a3e1e7ad1fde0a32823fc7e4d046bcfd86dae (refs/remotes/origins/trnk)
 Found   possible   branch   point:   file:///C:/Git/test-svn/trunk   =>   file:///C:/Git/test-svn/branches/my-calc-
 branch,            75             Found            branch            parent:            (refs/remotes/origins/my-calc-branch)
 55256a3e1e7ad1fde0a32823fc7e4d046bcfd86dae Following parent with do_switch
 Successfully followed parent
 r76     =     0fb585761df569eaecd8146c71e58d70147460a2     (refs/remotes/origins/my-calc-
 branch) Checked out HEAD:
```

```
 file:///C:/Git/test-svn/revpro-change tunk r75
```

Step 6: At this point, we should have a valid Git repository that has imported your branches and tags.

```
$ git branch -a
* master
 remotes/origin/calc-branch
 remotes/origin/tags/2.0.2
 remotes/origin/tags/release-2.0.1
 remotes/origin/tags/release-2.0.2
 remotes/origin/tags/release-2.0.2rc1
 remotes/origin/trnk
```

Step 7: Let us take the closer look with Git plumbing command show-ref:

```
$ git show-ref
55256a3e1e7ad1fde0a32823fc7e4d046bcfd86dae refs/heads/master
0fb585761df569eaecd8146c71e58d70147460a2 refs/remotes/origin/my-calc-branch
bfd2d79303166789fc73af4046651a4b35c12f0b refs/remotes/origin/tags/2.0.2
285c2b2e36e467dd4d91c8e3c0c0e1750b3fe8ca refs/remotes/origin/tags/release-2.0.1
cbda99cb45d9abcb9793db1d4f70ae562a969f1e refs/remotes/origin/tags/release-2.0.2
a9f074aa89e826d6f9d30808ce5ae3ffe711feda refs/remotes/origin/tags/release-2.0.2rc1
556a3e1e7ad1fde0a32823fc7e4d046bcfd86dae refs/remotes/origin/trunk
```

Step 8: Now you have a working directory, you can do some work on the project and you can push your commits back upstream, using Git effectively as an SVN client. If you edit one of the files and commit it, you can have a commit that exists in Git locally that does not exist on the Subversion server:

```
$ git commit -m 'Add svn instructions to the README' [master
4af61fd] Add svn instructions to the README
1 file changed, 5 insertions(+)
```

Step 9: Next, you need to push your change upstream. Then you need to push your change upstream. Notice how you can make several off-line commits and push all of them on Subversion server at once. You are running the git **svn dcommit** command to push to a Subversion server:

```
$ git svn dcommit
```

```
Committing to file:///C:/Git/test-svn/trunk ..

  M

Committed r77

    M

r77 = 95e0222ba6399739834380eb10afcd73e0670bc5 (refs/remotes/origin/trunk)

No changes between 4af61fd05045e07598c553167e0f31c84fd6ffe1 and refs/remotes/origin/trunk

Resetting to the latest refs/remotes/origin/trunk
```

Step 9: When you work with other developers, one will push, and the other will try to push a conflict-related change. This change will be rejected until you merge in their work. In git svn, it looks like this:

```
$ git svn dcommit

Committing to file:///C:/Git/test-svn/trunk ...


ERROR from SVN:

Transaction is out of date: File '/trunk/README.txt' is out of date

W: d5837c4b461b7c0e018b49d12398769d2bfc240a and refs/remotes/origin/trunk differ, using rebase:

    :100644 100644
  e80b6127dd04f5fda218720ddf3a2da4eb3912

Current branch master is up to date.

ERROR: Not all changes have been committed to SVN, however the committed

changes (if any) appear to be integrated into the working tree successfully.

Please see the above messages for            details.
```

Step 10: You can use git svn rebase to resolve this situation which pulls down any changes on the server that you do not yet have and re-bases any work you have on top of the server:

```
$ git svn rebase

Committing to file:///C:/Git/test-svn/trunk ...


ERROR from SVN:

Transaction is out of date: File '/trnk/README.txt' is out of date

W: eaa029d99f87c5c822c5c29039d19111ff32ef46 and refs/remotes/origin/trunk differ, using rebase:
```

```
:100644 100644
65536c6e30d263495c17d781962cfff12422693a
```

```
First, rewinding head to replay your work on top of it...

Applying: update foo

Using index info to reconstruct a base tree...

M README.txt

Falling back to patching base and 3-way merge...

Auto-merging README.txt

ERROR: Not all changes are committed to SVN, but the committed ones (if any)

are integrated into the working tree successfully.

Please see the above messages for details.
```

Step 11: To create a new branch in Subversion, you run git svn branch [new-branch]:

```
$ git svn branch opera

Copying file:///C:/Git/test-svn/trunk at r90 to file:///C:/Git/test-svn/branches/opera...

Found possible branch point: file:///C:/Git/test-svn/trunk => file:///C:/Git/test-svn/branches/opera,

90          Found          branch          parent:          (refs/remotes/origin/opera)

cb522197870e61467473391799148f6721bcf9a0 Following parent with do_switch

Successfully followed parent

r91 = f1b64a3855d3c8dd84ee0ef10fa89d27f1584302 (refs/remotes/origin/opera)
```