



SVN to Git Migration

&

Operations using
TortoiseGit

Table of Contents

1	Prerequisites for the Migration	2
2	Implementation steps involved in migration	2
3	Git Operations using TortoiseGit	15
3.1	Clone	16
3.2	Commit	21
3.3	Merge.....	27

Document Information

Project Name:	Project LPKM		
Prepared By:	Swapna Munnangi	Document Version No:	0.1
Title:	SVN to Git Migration & Operations using TortoiseGit	Document Version Date:	12-Apr-18
Reviewed By:	Molakala Reddy Praveen	Review Date:	

Document Version History

Version Number	Version Date	Prepared by	Revised By	Description
1	12-Apr-18	Swapna Munnangi	Molakala Reddy Praveen	SVN to Git Migration & Operations using TortoiseGit

SVN Migration to Git

1. Prerequisites for Migration

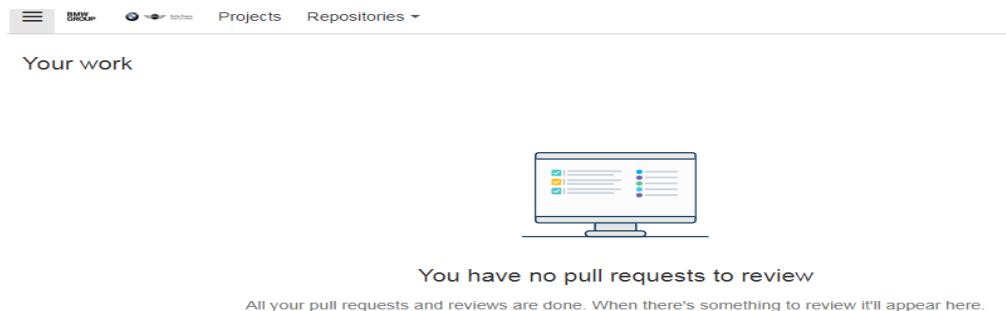
- SVN installation setup on local system
SVN: <https://tortoisesvn.net/downloads.html>
- Ordering Bit bucket [Using below URL]
<https://bmwprod.service-now.com>
- Access to SVN Repository
- Access to Bit bucket

2. Implementation steps involved in migration

1. Creation of Git REPOSITORY

- Login to the below URL using QX number and password to see the below screen

URL : <https://atc.bmwgroup.net/bitbucket/>



- Now click on Projects, search for the application(LPKM) and click on it

Apps

Managed bookmarks

Google Translate

BMW MAIL

Citrix Access Gateway

New TMS

Home

Shopping Area

MyNetwork | Anme










LPKM

1/2

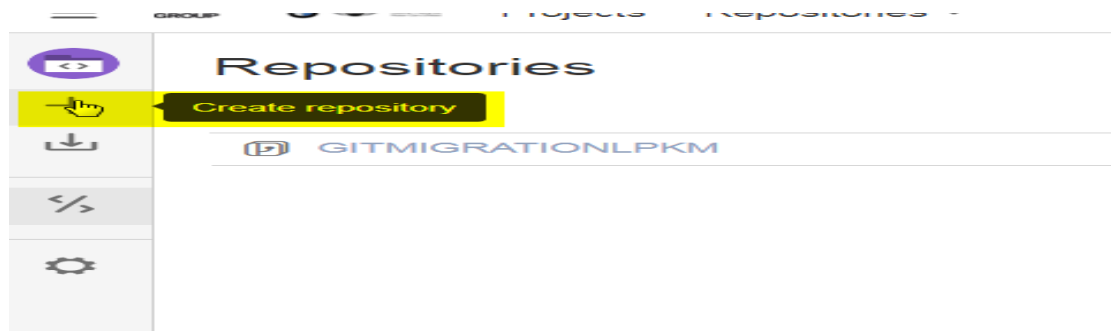
^

v

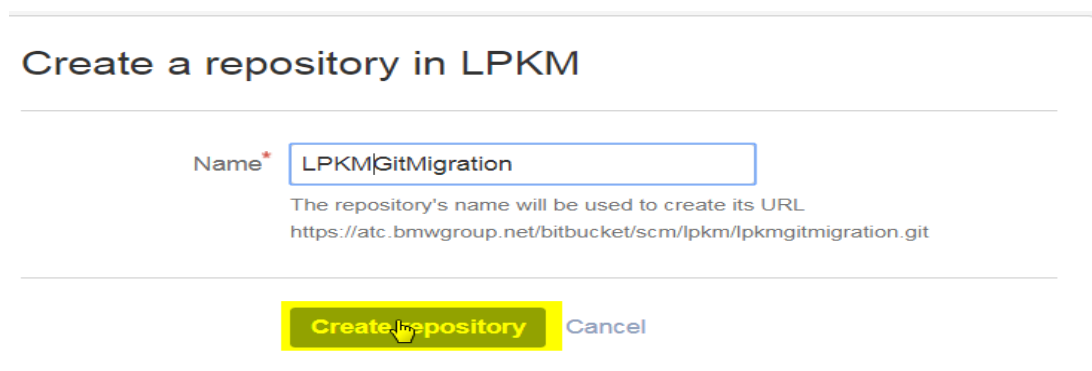
X

 IRMI (ProXx-RGM)	IRMI	No description
 ISPI	ISPI	No description
 KZSR	KZSR	No description
 Last State Call	LSCAGILE	No description
 LBD	LBD2018	No description
 Liquid	LIQUID	No description
 Logistische Terminierung	LTERM	No description
 LPKM	LPKM	No description
 Material, Gewicht, Workplace, Zertifikate	MGWZ	Roland Wangemann Material, Gewicht, Workplace, Zertifikate

- Click on 'Create repository'



- Specify a name to your repository and click on 'Create Repository' button.



The screenshot shows the 'Create a repository in LPKM' form. The 'Name' field is filled with 'LPKMGitMigration'. Below the field, a message states: 'The repository's name will be used to create its URL https://atc.bmwgroup.net/bitbucket/scm/lpkm/lpkmgitmigration.git'. The 'Create repository' button is highlighted in yellow, and the 'Cancel' button is visible next to it.

5. Go to SVN Mirror settings in the newly created Repository



6. Provide the SVN URL, SVN username, password, email domain and select trunk path of SVN.

Subversion Project URL

URL* SVN URL

Subversion Credentials

☐ Use server-side Subversion configuration and credentials cache
Configuration directory path: '/home/qqajip0/.subversion' (learn more).

Authenticate With:

User* SVN User name and Password

Password

Subversion Project Layout

Branches

- ☐ Manual Configuration
Configure branches and tags mapping manually at the next step.
- ☐ Single Directory Translation
Content below Project URL will be translated to the single Git branch. No other branches or tags will be translated.
- ☒ Automatic Configuration
Let add-on detect SVN branches and tags from the trunk path.

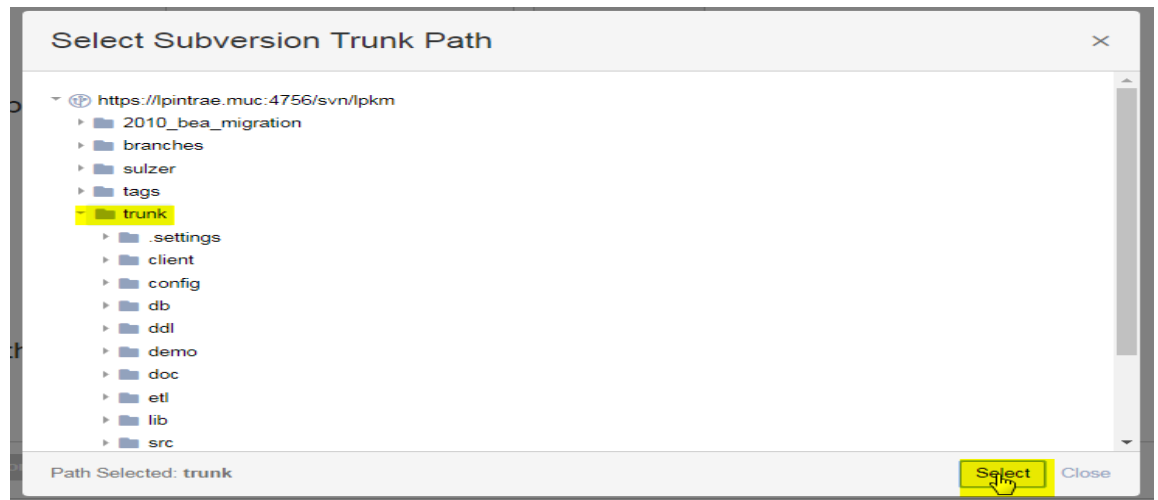
select trunk path of svn ⓘ

Select path that represents 'trunk' directory. Usually it is a folder named 'trunk' just below the project root. Add-on will discover other branches and tags automatically.

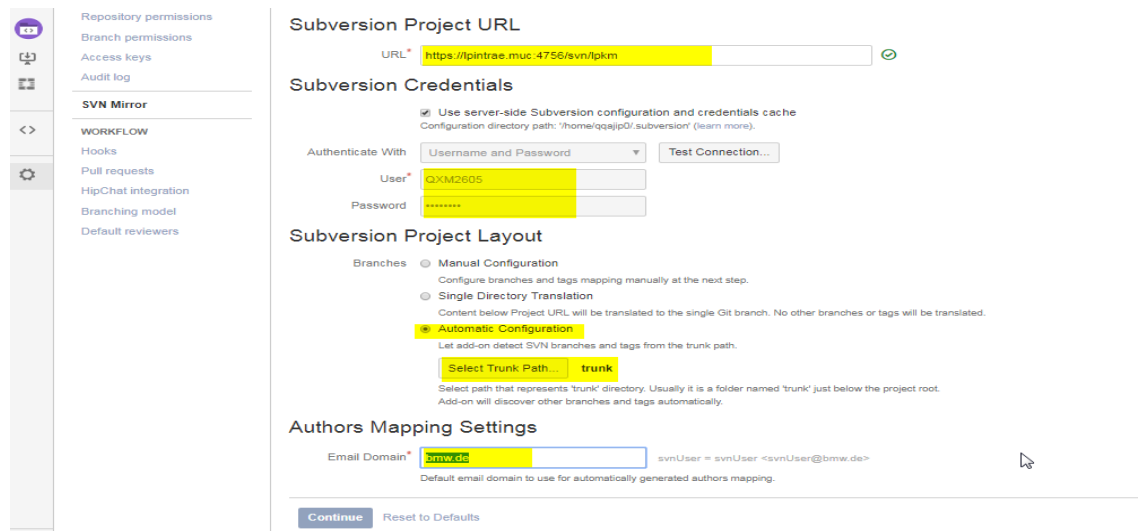
Authors Mapping Settings

Email Domain*

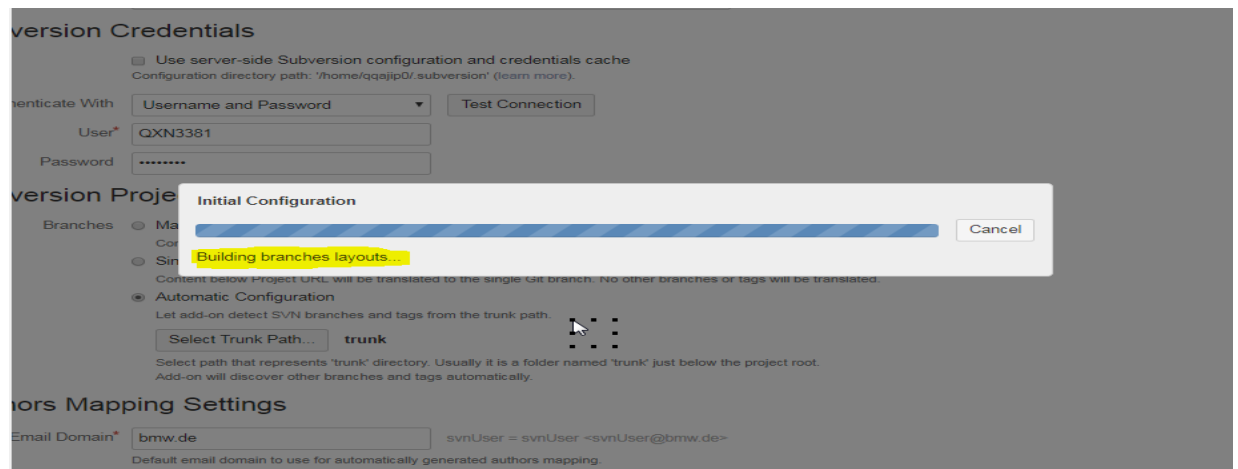
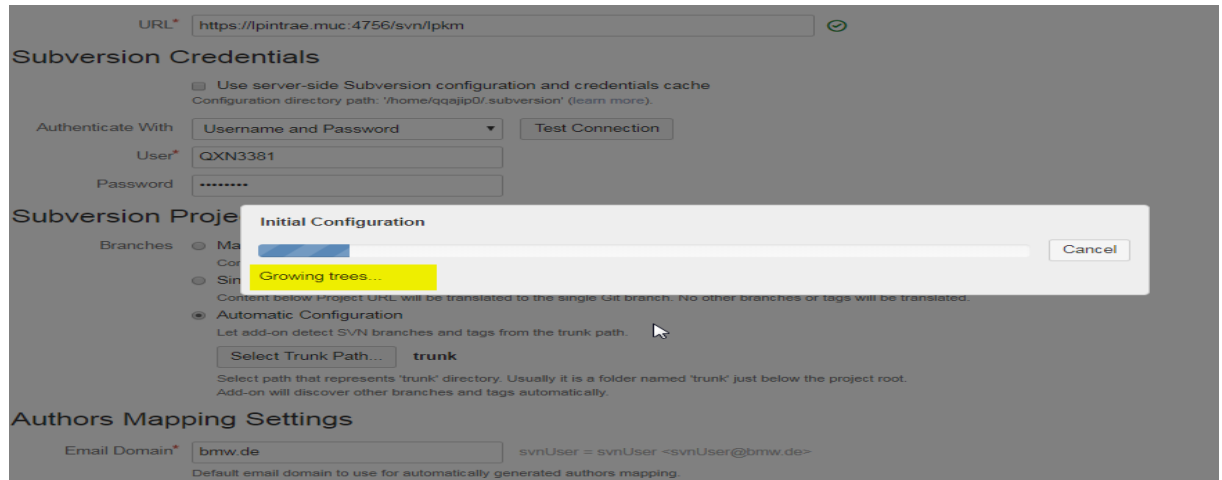
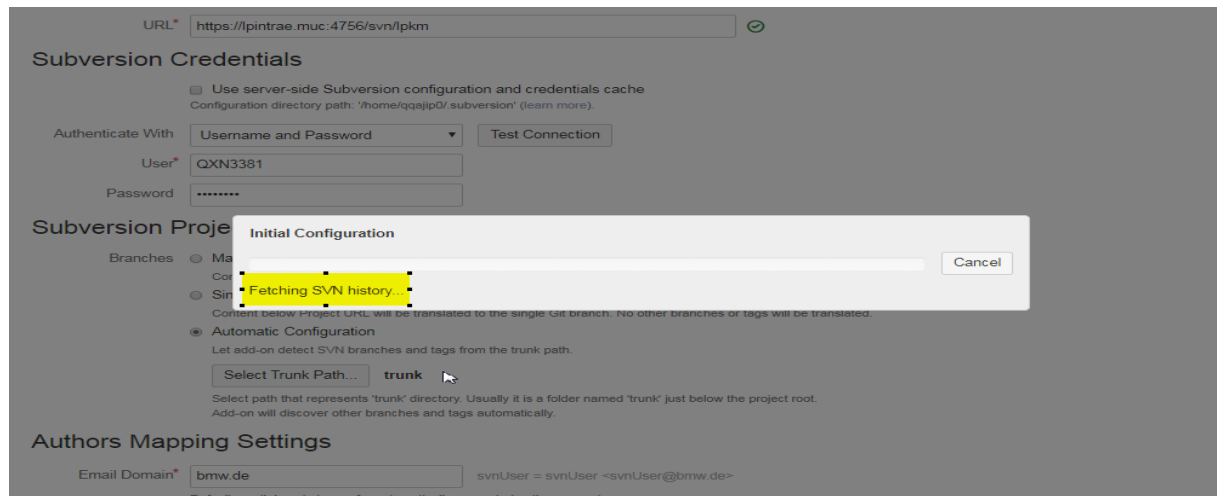
7. Selecting the trunk path of SVN as shown below,



8. Click on continue button



9. It connects to SVN and below messages appears once configuration started.



10. Once the configuration process is completed, the below screen appears

Review Configuration and Start Mirror or Import

Branches Mapping Translation Settings Authors Mapping Connection

Subversion Branches Mapping

```
# Subversion to Git mapping options
#
[svn]
# Options below (trunk, branches, tags, shelves) define correspondence between Subversion
# directories and Git references. Depending on the actual Subversion project layout and whether
# all or only some of the branches have to be mirrored, these options might need to be adjusted.
#
# Generic mapping syntax is:
# <Subversion-Path-Pattern>::<Git-Reference-Pattern>
#
# Subversion paths are relative to the URL defined by the svn.url option.
#
# For more details refer to http://subgit.com/documentation pages.
trunk = trunk:refs/heads/master
branches = branches/*:refs/heads/*
branches = lpkm:refs/heads/lpkm
branches = wartung_2010:refs/heads/wartung_2010
branches = branches/lpkm-18.0_GF4Migration/lpkm-17.1.0-release:refs/heads/lpkm-18.0_GF4Migration/lpkm-17.1.0-release
tags = tags/*:refs/tags/*
shelves = shelves/*:refs/shelves/*
excludeBranches = tags/lpkm-snapshot-transition
```

Back Revert Changes Import Mirror

11. Now adjust the Branch Mapping, Authors Mapping and translation modifications are required as mappings are automatically picked up.

- **Branch Mapping**

Review Configuration and Start Mirror or Import

Branches Mapping Translation Settings Authors Mapping Connection

Subversion Branches Mapping

```
# Subversion to Git mapping options
#
[svn]
# Options below (trunk, branches, tags, shelves) define correspondence between Subversion
# directories and Git references. Depending on the actual Subversion project layout and whether
# all or only some of the branches have to be mirrored, these options might need to be adjusted.
#
# Generic mapping syntax is:
# <Subversion-Path-Pattern>::<Git-Reference-Pattern>
#
# Subversion paths are relative to the URL defined by the svn.url option.
#
# For more details refer to http://subgit.com/documentation pages.
trunk = trunk:refs/heads/master
branches = branches/*:refs/heads/*
branches = lpkm:refs/heads/lpkm
branches = wartung_2010:refs/heads/wartung_2010
branches = branches/lpkm-18.0_GF4Migration/lpkm-17.1.0-release:refs/heads/lpkm-18.0_GF4Migration/lpkm-17.1.0-release
tags = tags/*:refs/tags/*
shelves = shelves/*:refs/shelves/*
excludeBranches = tags/lpkm-snapshot-transition
```

Back Revert Changes Import Mirror

- **Authors Mapping**

Before Adjusting the Authors Mapping, screen appears as below

Repository Authors Mapping

☒ Use Repository Authors Mapping

Use authors mapping defined below to map Subversion users (svnUser) to Git authors (Author Name <email>).

```
# This is SubGit authors mapping file.
# Authors mapping is used to map Subversion committers names to Git committers names and vice versa.
# This file uses git-svn format, as described at 'http://www.kernel.org/pub/software/scm/git/docs/git-svn.html'
# and consists of the lines in the following format:
#
# svnUser = Git User <user@example.com>
#
qx08189 = qx08189 <qx08189@bmw.de>
qx16924 = qx16924 <qx16924@bmw.de>
qxb1951 = qxb1951 <qxb1951@bmw.de>
qxc2209 = qxc2209 <qxc2209@bmw.de>
qxc2966 = qxc2966 <qxc2966@bmw.de>
qxe3642 = qxe3642 <qxe3642@bmw.de>
qxe3644 = qxe3644 <qxe3644@bmw.de>
#
# Mappings below are commented out because there are matching Bitbucket users.
# Uncomment those mappings in case 'Map Subversion Users to Bitbucket Users' option is disabled
# or if you would like to override Bitbucket user mapping with the custom values.
#
# q099973 = Franz Pflieger (FG-535) <Franz.Pflieger@bmw.de>
# q299700 = Sujay Thukral Dwarakanath <Sujay.Thukral@bmw.de>
# qx43682 = Christoph Vormoor (ext.) <Christoph.Vormoor@partner.bmw.de>
# qx44602 = Markus Liehmann <Markus.Liehmann@partner.bmw.de>
# qxa1023 = Juergen Finger (ext.) <Juergen.Finger@partner.bmw.de>
# qxn3381 = Anudeep Pokala <Anudeep.Pokala@partner.bmw.de>
```

Back

Revert Changes

Import

Mirror

After adjusting the Authors Mapping, screen appears as below

Repository Authors Mapping

Use authors mapping defined below to map Subversion users (svnUser) to Git authors (Author Name <email>).

```
#qx08189 = qx08189 <qx08189@bmw.de>
#qx16924 = qx16924 <qx16924@bmw.de>
#qxb1951 = qxb1951 <qxb1951@bmw.de>
#qxc2209 = qxc2209 <qxc2209@bmw.de>
#qxc2966 = qxc2966 <qxc2966@bmw.de>
#qxe3642 = qxe3642 <qxe3642@bmw.de>
#qxe3644 = qxe3644 <qxe3644@bmw.de>

q099973 = q099973 <Franz.Pflieger@bmw.de>
q299700 = q299700 <Sujay.Thukral@bmw.de>
qx43682 = qx43682 <Christoph.Vormoor@partner.bmw.de>
qx44602 = qx44602 <Markus.Liehmann@partner.bmw.de>
qxa1023 = qxa1023 <Juergen.Finger@partner.bmw.de>
qxn3381 = qxn3381 <Anudeep.Pokala@partner.bmw.de>
qxm2605 = qxm2605 <Swapna.Munnangi@partner.bmw.de>

#
# Mappings below are commented out because there are matching Bitbucket users.
# Uncomment those mappings in case 'Map Subversion Users to Bitbucket Users' option is disabled
# or if you would like to override Bitbucket user mapping with the custom values.
#
# q099973 = Franz Pflieger (FG-535) <Franz.Pflieger@bmw.de>
# q299700 = Sujay Thukral Dwarakanath <Sujay.Thukral@bmw.de>
# qx43682 = Christoph Vormoor (ext.) <Christoph.Vormoor@partner.bmw.de>
# qx44602 = Markus Liehmann <Markus.Liehmann@partner.bmw.de>
# qxa1023 = Juergen Finger (ext.) <Juergen.Finger@partner.bmw.de>
# qxn3381 = Anudeep Pokala <Anudeep.Pokala@partner.bmw.de>
```

- Translation Settings

Review Configuration and Start Mirror or Import

Branches Mapping Translation Settings Authors Mapping Connection

Authors Mapping Settings

☒ Use Repository Authors Mapping

Use authors mapping defined in this repository to map Subversion users (svnUser) to Git authors (Author Name <email>).

☒ Map Subversion Users to Bitbucket Users

When mapping enabled above is disabled or contains no match, use Bitbucket users registry to find Git author by Subversion user name (svnUser).

☒ Use Global Authors Mapping

When mappings above are disabled or contains no match, map Subversion users to Git authors using explicit global authors mapping.

Email Domain* svnUser = svnUser <svnUser@bmw.de>
Default email domain to use when no match has been found.

Translation Settings

Settings in this section could only be altered before synchronization is ran for the very first time.

Minimal Revision*
Subversion revision to start translation from.

☐ Translate file attributes

Translate changes in .gitattributes files to svn:eol-style and svn:mime-type Subversion properties.

☒ Translate Ignores

Translate changes in .gitignore files to svn:ignore Subversion property.

Back Revert Changes Import Mirror

- Connection Settings

Review Configuration and Start Mirror or Import

Branches Mapping Translation Settings Authors Mapping Connection

Subversion Project URL

URL* 

Subversion Credentials

☐ Use server-side Subversion configuration and credentials cache
Configuration directory path: '/home/qqajip0/.subversion' (learn more).

Authenticate With

Test Connection

User*

Password

Synchronization Settings

Poll Interval*

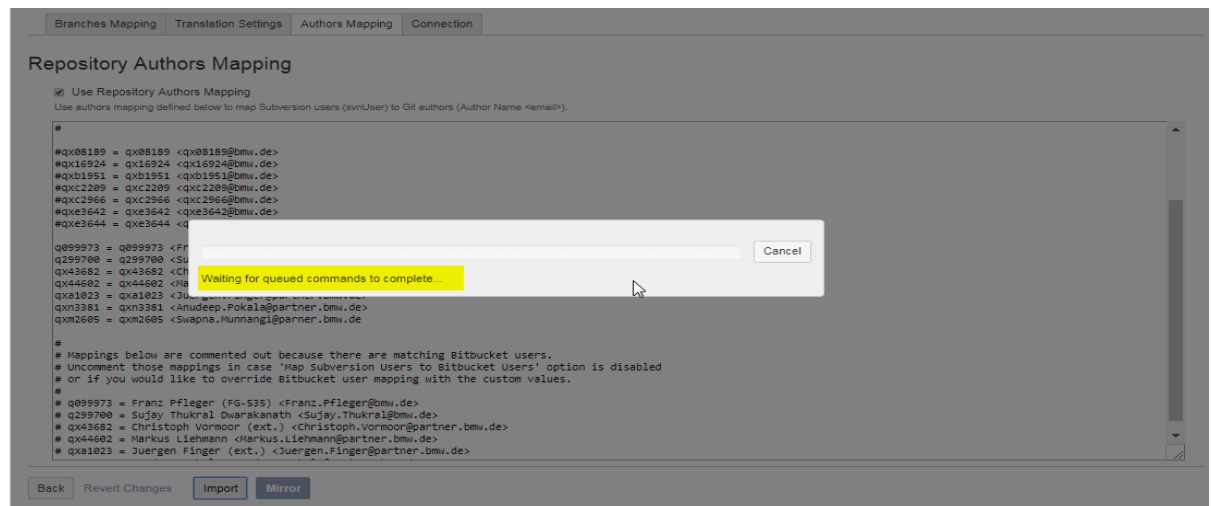
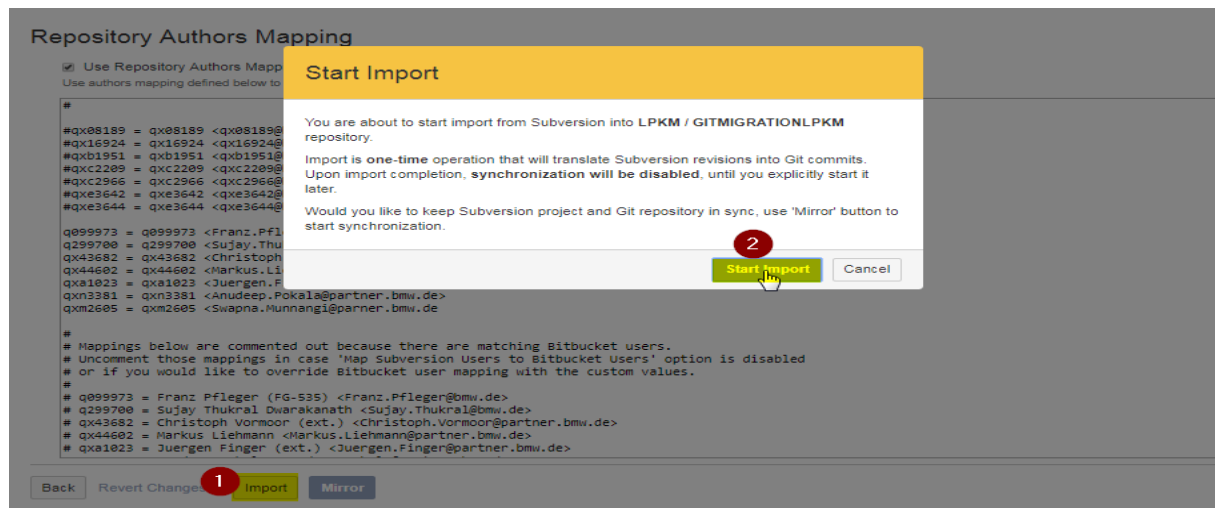
Interval, in seconds to poll Subversion repository for new changes.

When set to 0, synchronization will occur on each push to this repository and also could be explicitly invoked via REST API.

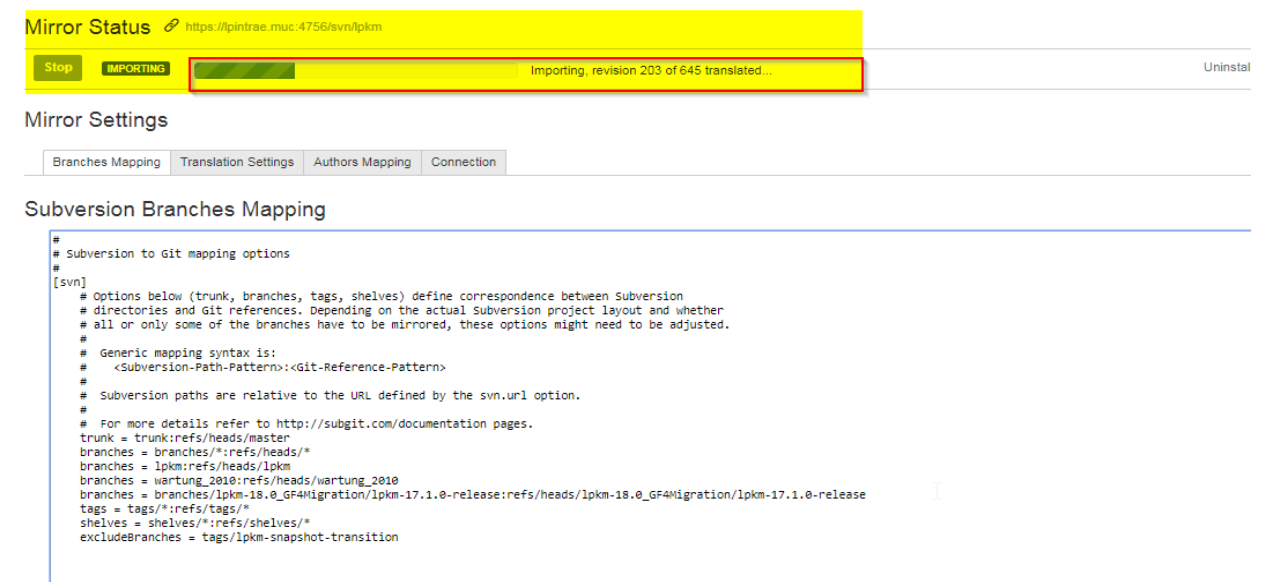
Setting this interval to 0 and installing SVN post-commit hook to call REST API will reduce add-on resources usage.

Back Revert Changes Import Mirror

12. Finally click on 'Import' button and then click on 'Start import' button to import the code from SVN repository to Git



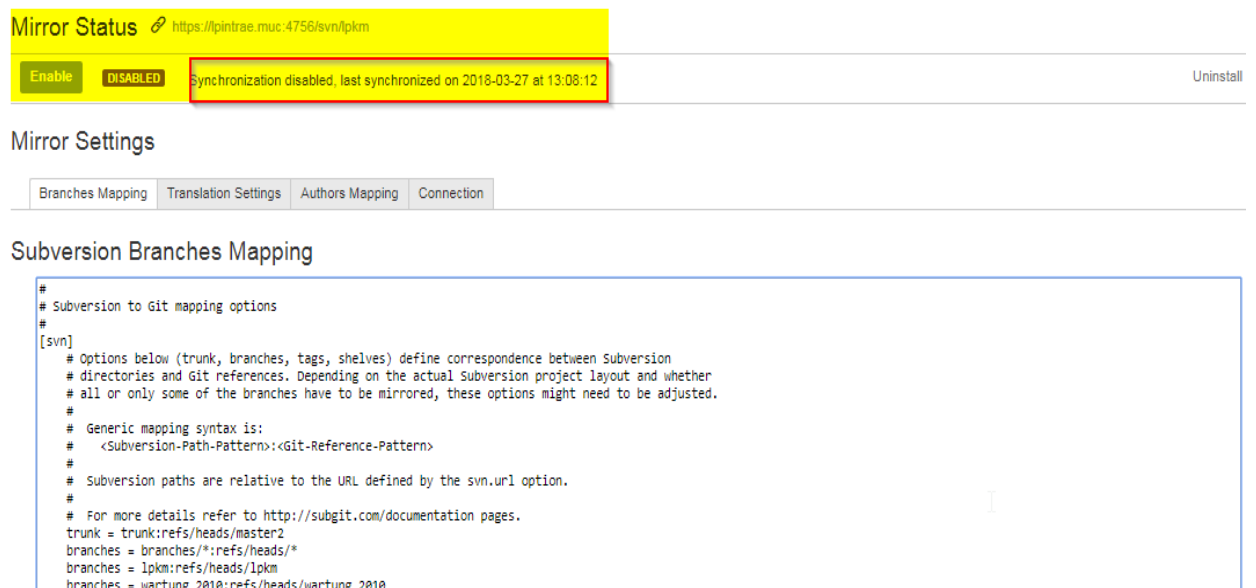
13. Importing process is visible on the top of the Mirror settings window.



The screenshot shows the 'Mirror Status' window for the URL <https://pintrac.muc:4756/svn/lpkm>. At the top, there are two buttons: 'Stop' and 'IMPORTING'. A progress bar is shown with the text 'Importing, revision 203 of 645 translated...'. Below this, the 'Mirror Settings' section is visible, with tabs for 'Branches Mapping', 'Translation Settings', 'Authors Mapping', and 'Connection'. The 'Subversion Branches Mapping' section is expanded, showing a configuration file with the following content:

```
# Subversion to Git mapping options
#
[svn]
# Options below (trunk, branches, tags, shelves) define correspondence between Subversion
# directories and Git references. Depending on the actual Subversion project layout and whether
# all or only some of the branches have to be mirrored, these options might need to be adjusted.
#
# Generic mapping syntax is:
# <Subversion-Path-Pattern>:<Git-Reference-Pattern>
#
# Subversion paths are relative to the URL defined by the svn.url option.
#
# For more details refer to http://subgit.com/documentation pages.
trunk = trunk:refs/heads/master
branches = branches/*:refs/heads/*
branches = lpkm:refs/heads/lpkm
branches = wartung_2010:refs/heads/wartung_2010
branches = branches/lpkm-18.0_GF4Mmigration/lpkm-17.1.0-release:refs/heads/lpkm-18.0_GF4Mmigration/lpkm-17.1.0-release
tags = tags/*:refs/tags/*
shelves = shelves/*:refs/shelves/*
excludeBranches = tags/lpkm-snapshot-transition
```

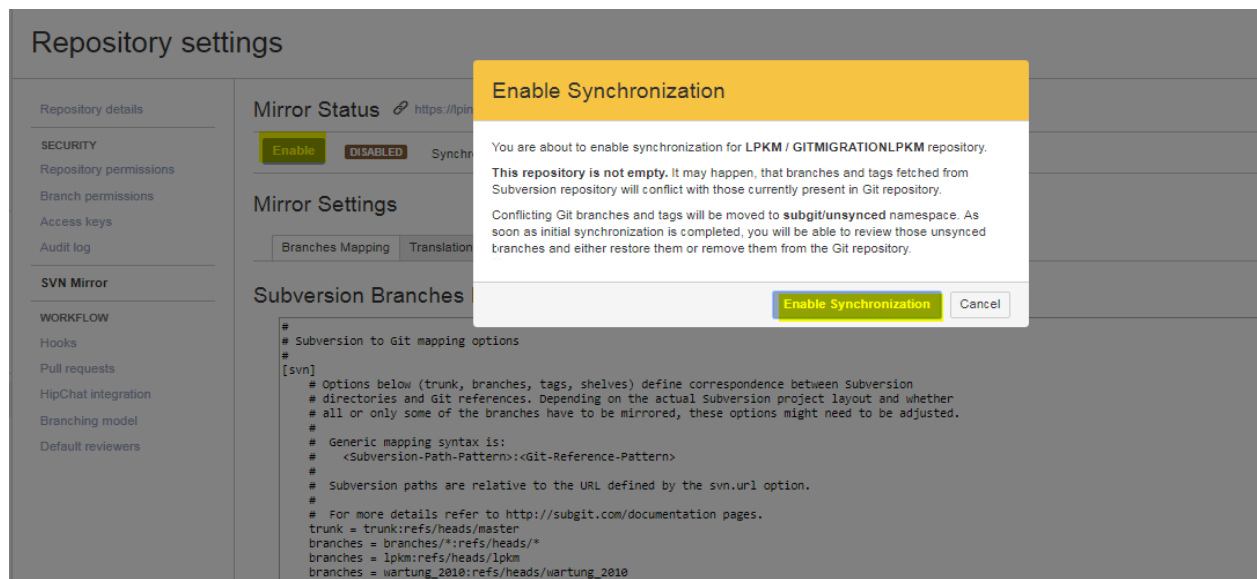
14. Once the import is done, ENABLE button is viewed on left top corner.



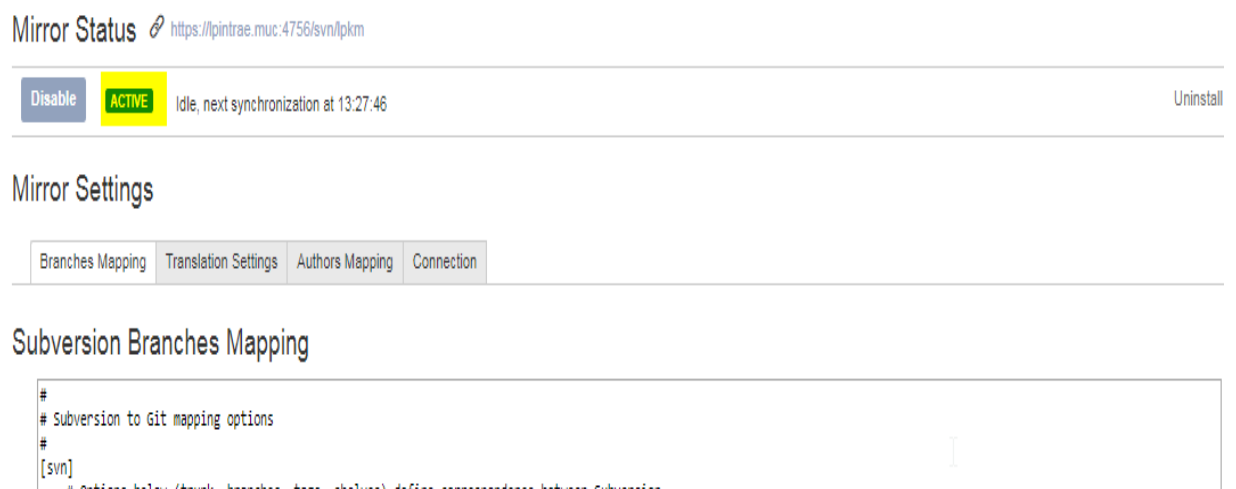
The screenshot shows the 'Mirror Status' window for the same URL. The 'Stop' button has been replaced by an 'Enable' button, and the 'IMPORTING' button is now 'DISABLED'. A message box states 'Synchronization disabled, last synchronized on 2018-03-27 at 13:08:12'. The 'Mirror Settings' section remains the same, with the 'Subversion Branches Mapping' section expanded, showing a configuration file with the following content:

```
# Subversion to Git mapping options
#
[svn]
# Options below (trunk, branches, tags, shelves) define correspondence between Subversion
# directories and Git references. Depending on the actual Subversion project layout and whether
# all or only some of the branches have to be mirrored, these options might need to be adjusted.
#
# Generic mapping syntax is:
# <Subversion-Path-Pattern>:<Git-Reference-Pattern>
#
# Subversion paths are relative to the URL defined by the svn.url option.
#
# For more details refer to http://subgit.com/documentation pages.
trunk = trunk:refs/heads/master2
branches = branches/*:refs/heads/*
branches = lpkm:refs/heads/lpkm
branches = wartung_2010:refs/heads/wartung_2010
```

15. Click on 'Enable' button and then click on 'Enable Synchronization'.

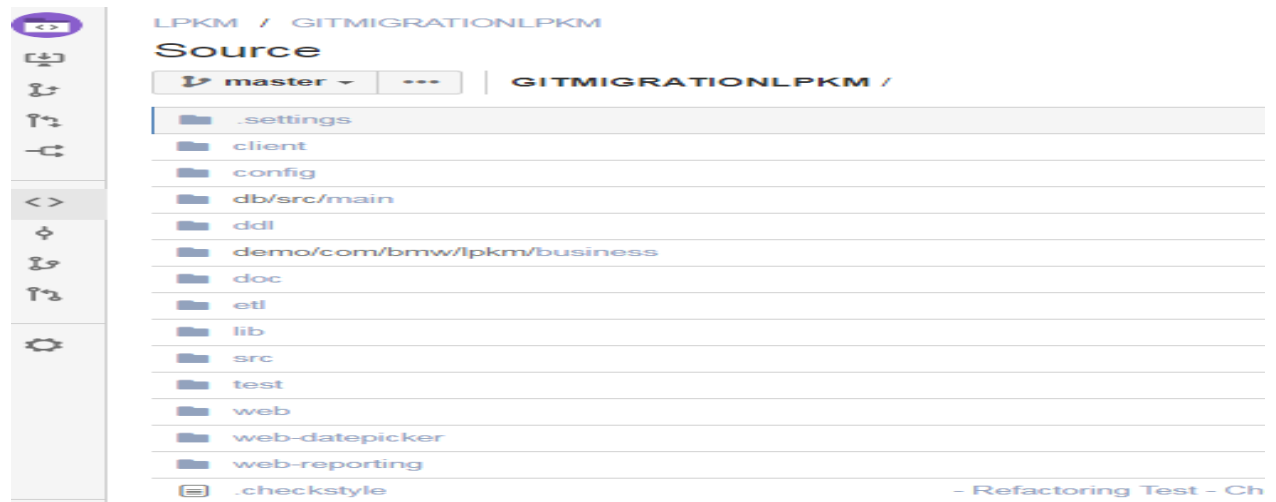


16. The progress bar starts and once the process is completed, the below screen appears. If there are any conflicts, then 'UNSYNCED' button appears next to 'ACTIVE' button.

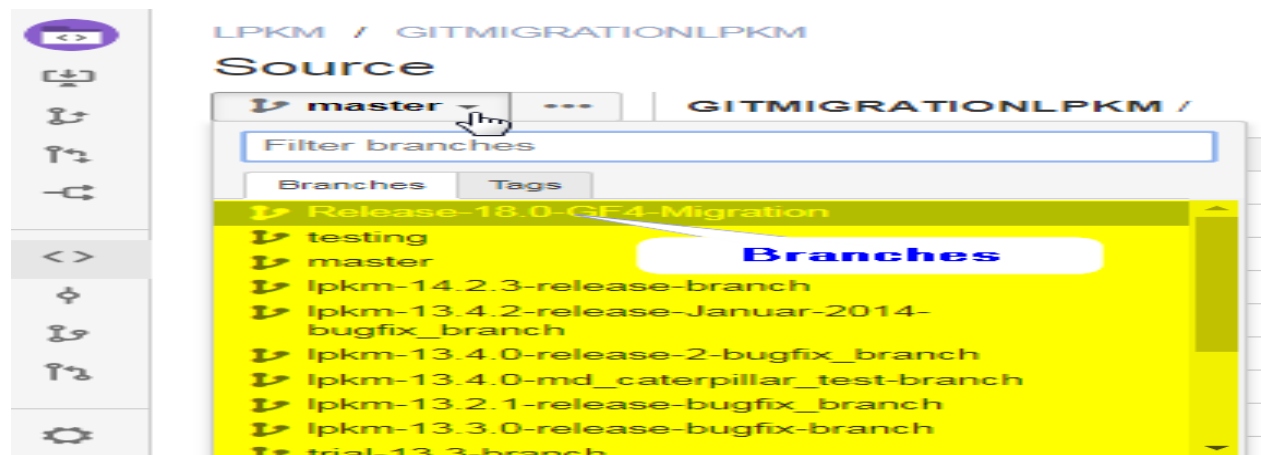


17. Now we have the Git repository with the contents of SVN repository

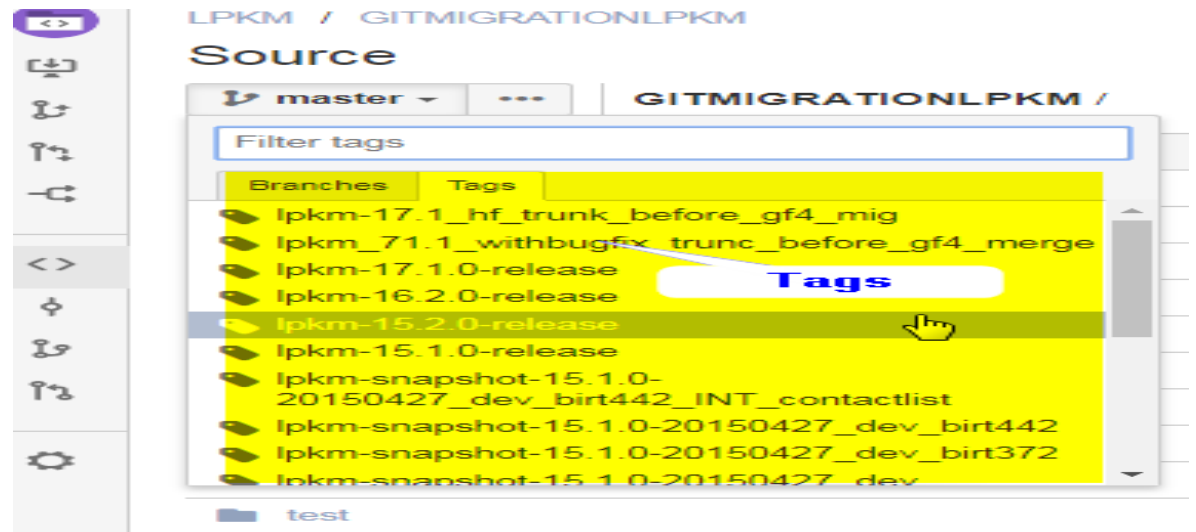
Master [Master is trunk in SVN]



Branches



Tags



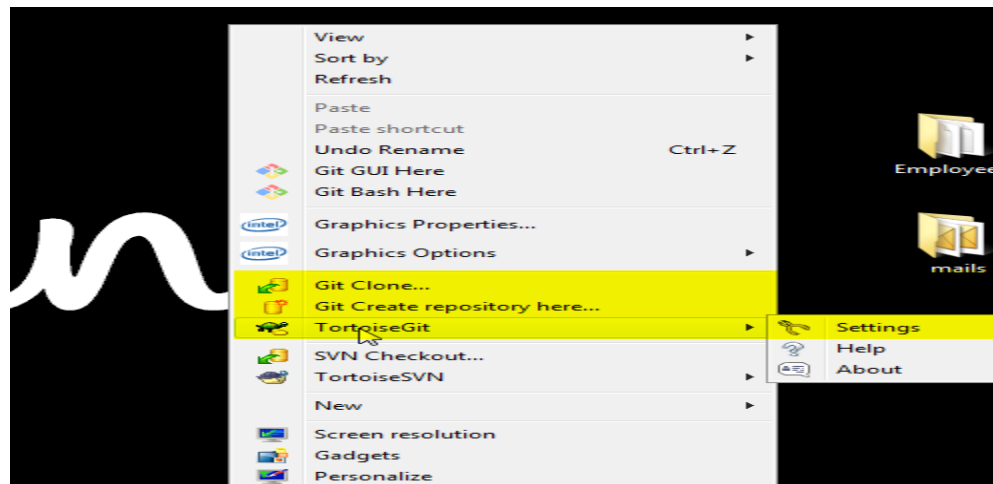
3. Operations using TortoiseGit

Installation of TortoiseGit

Install TortoiseGit from the URL,

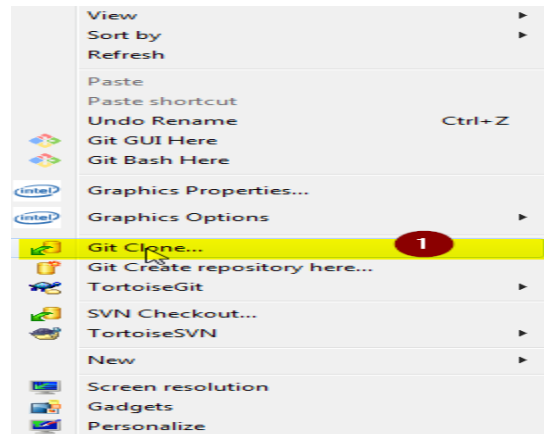
URL : :- <https://tortoisegit.org/>

Once the installation is completed, right click on your desktop so that you can see the below options.



3.1 Clone

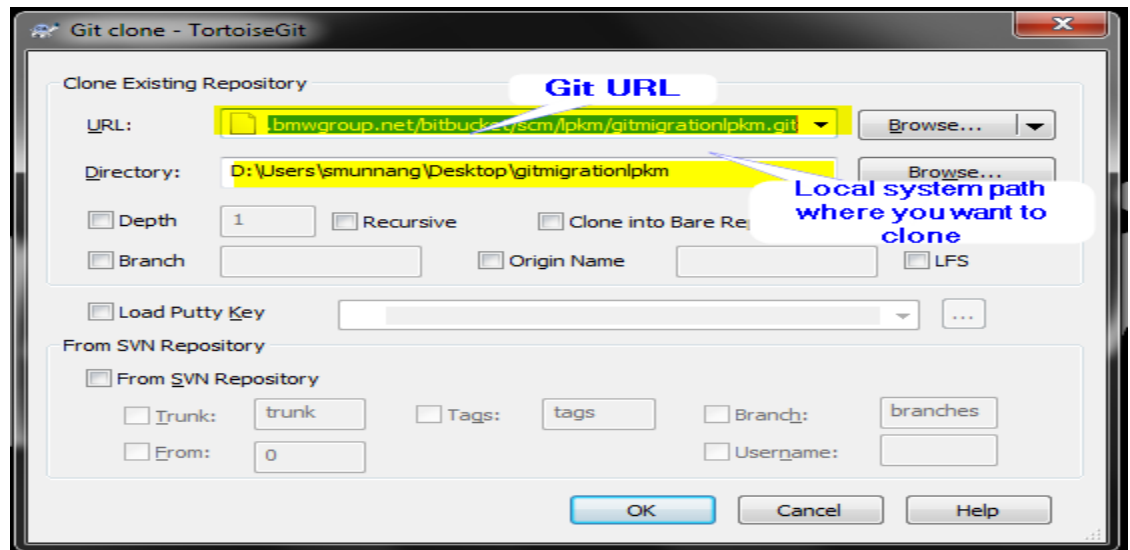
- Connect to Git remote repository and click on Git clone
[Clone in Git is similar to checkout of code in SVN]



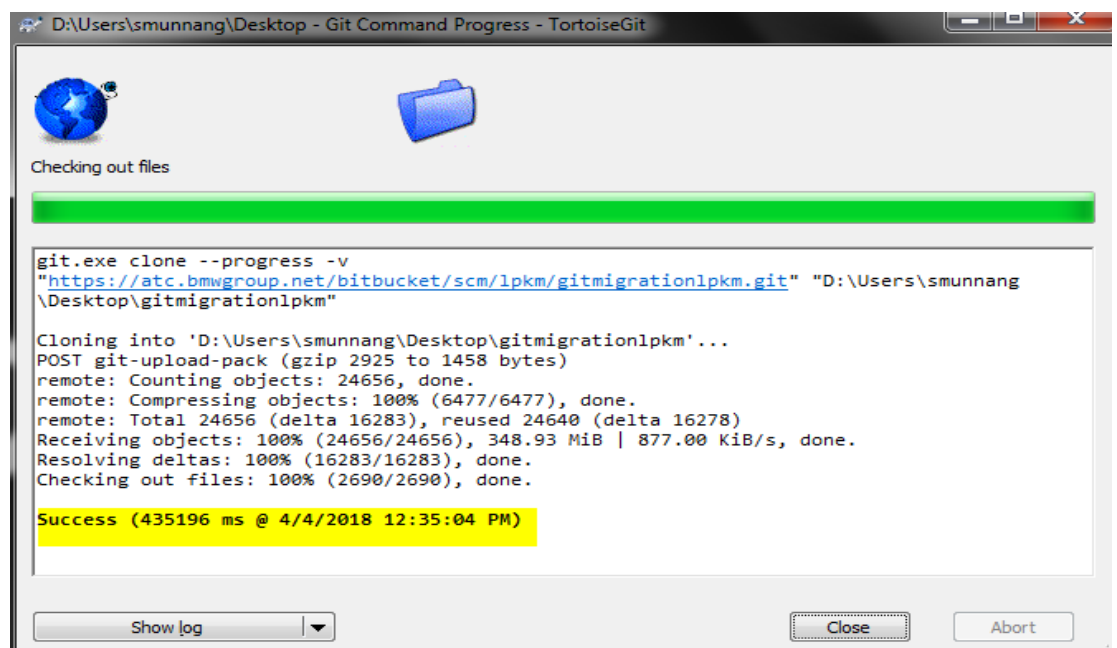
- For **cloning of Master**, provide the below details,
 - Git repository URL
 - Browse the path of local system of Master to be Cloned from the Git repositoryClick on 'OK'.

Git Repository URL :

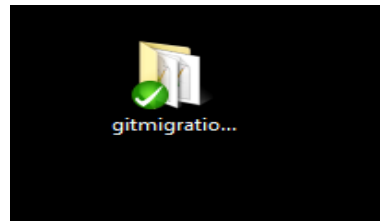
<https://atc.bmwgroup.net/bitbucket/scm/lpkm/gitmigrationlpkm.git>



- Cloning of Master from Git repository takes time and success message will appear once completed as appear in the below screenshot



- Folder of cloned Master from Git repository to local system appears as in below image.

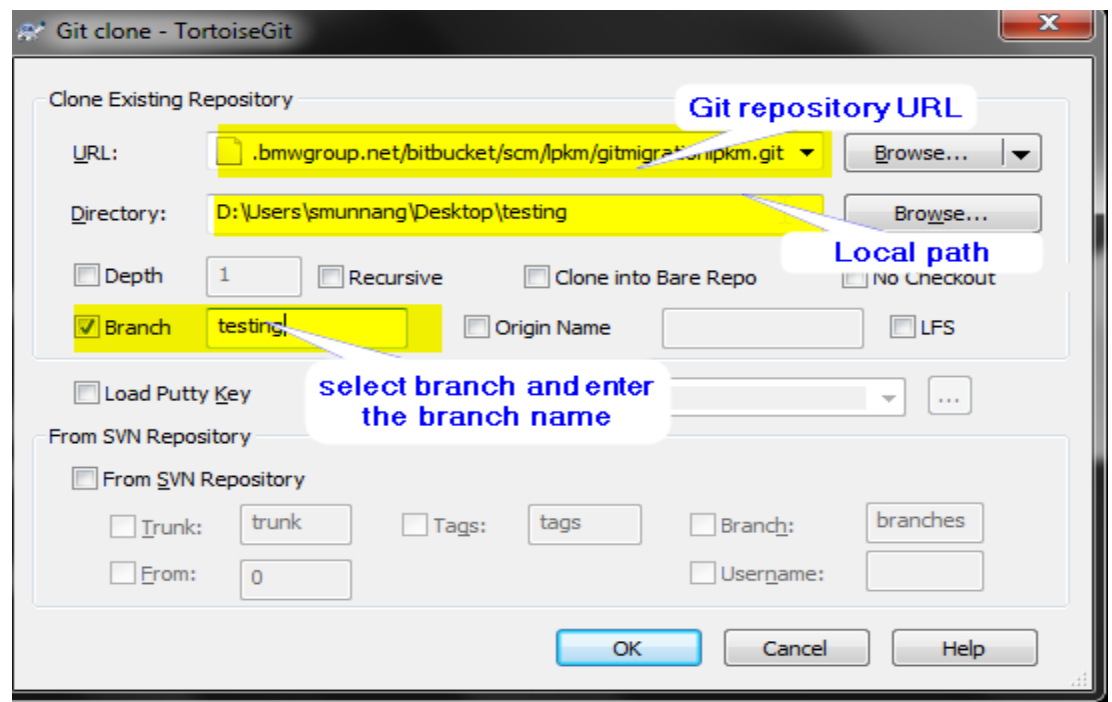


- In order to **clone the Branch**, we have 2 ways

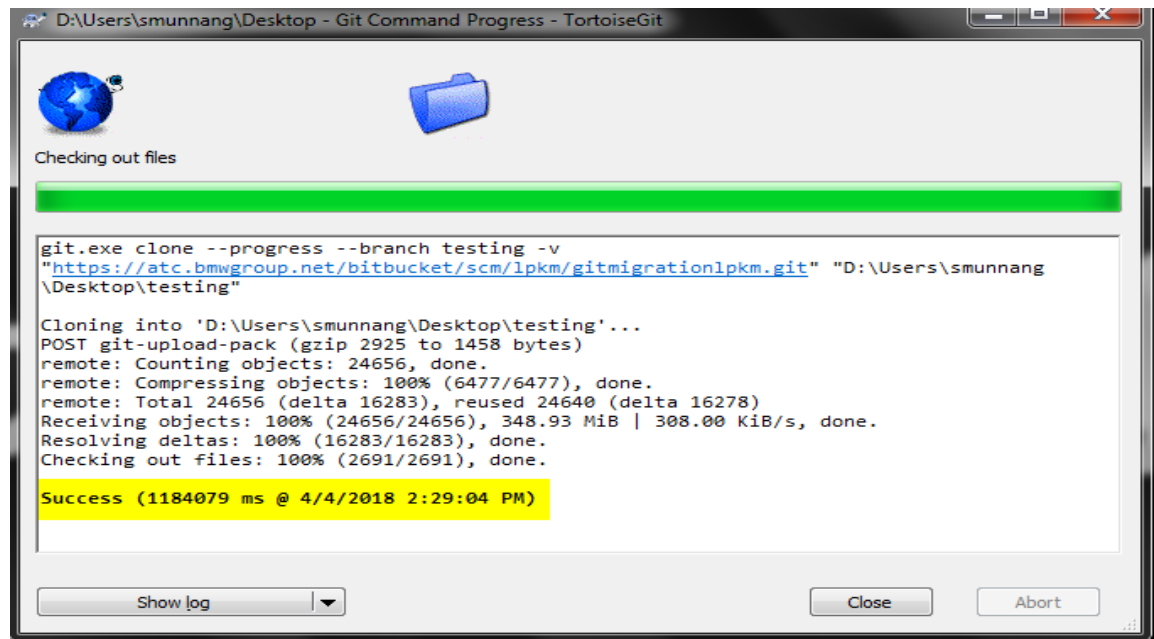
a) **Cloning of a Branch using 'Git clone' option**

- Enter the Git repository URL
- Browse the path of local system of branch to be cloned from the Git repository
- Check the '**Branch**' option
- Enter the Branch name

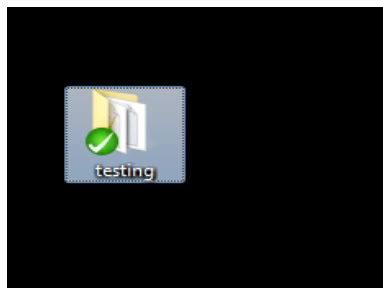
Click on 'OK'



- Cloning of Branch from Git repository takes time and success message will appear once completed as appear in the below screenshot

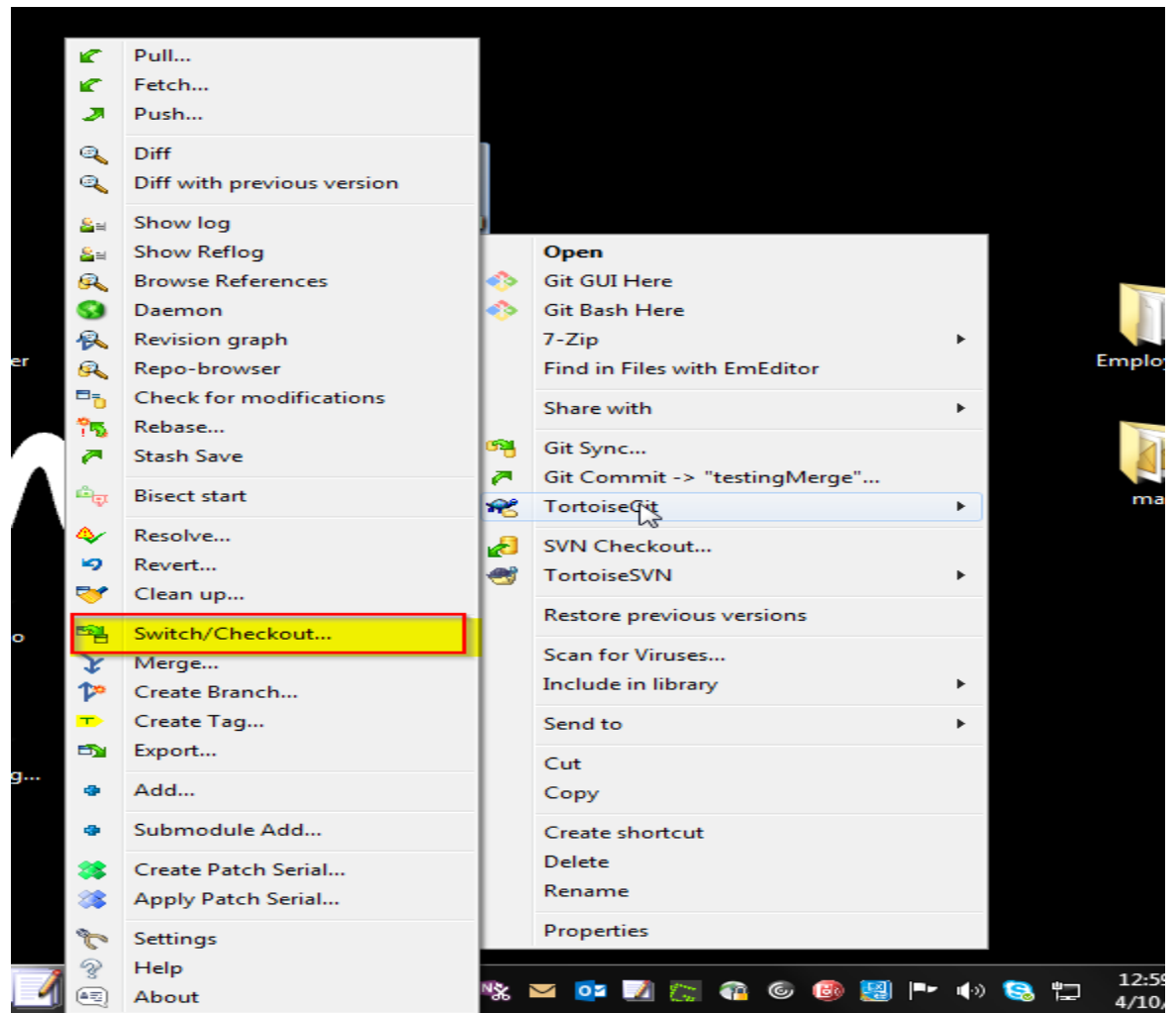


- Folder of cloned Branch from the Git repository to local system appears as in below image.

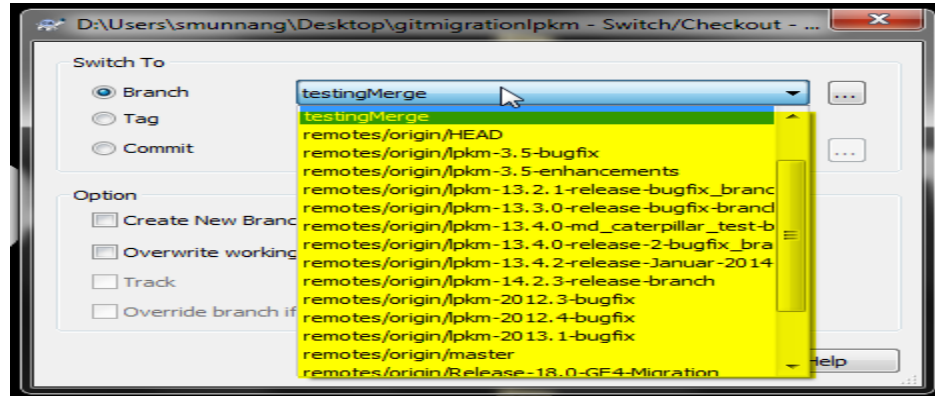


b) Cloning of a Branch by using 'Switch/Checkout'

- Go to → Right click on the cloned Master folder → click on 'TortoiseGit' → click on Switch/Checkout'



- Select the required Branch to be cloned and click on 'OK'. So that we can see the contents of required cloned Branch.



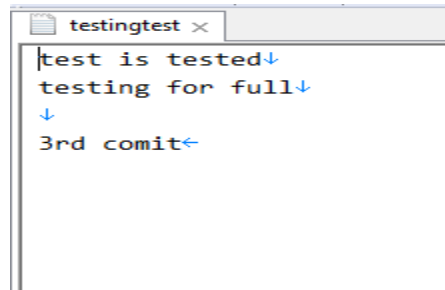
- Now you will have the contents of a specified Branch in the cloned folder based on our selection.

3.2 Commit

- Steps to commit the new changes made in the Branch of local system to Git repository illustrated here with an example.
 - Selected the file 'testingtest.txt' as appear in below screenshot

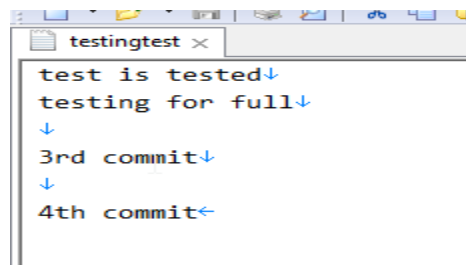
Name	Date modified	Type	Size
.git	4/4/2018 2:29 PM	File folder	
.settings	4/4/2018 2:28 PM	File folder	
client	4/4/2018 2:28 PM	File folder	
config	4/4/2018 2:28 PM	File folder	
db	4/4/2018 2:28 PM	File folder	
ddl	4/4/2018 2:28 PM	File folder	
demo	4/4/2018 2:28 PM	File folder	
doc	4/4/2018 2:28 PM	File folder	
etl	4/4/2018 2:28 PM	File folder	
lib	4/4/2018 2:29 PM	File folder	
src	4/4/2018 2:29 PM	File folder	
test	4/4/2018 2:29 PM	File folder	
web	4/4/2018 2:29 PM	File folder	
web-datepicker	4/4/2018 2:29 PM	File folder	
web-reporting	4/4/2018 2:29 PM	File folder	
.checkstyle	4/4/2018 2:28 PM	CHECKSTYLE File	1
.classpath	4/4/2018 2:28 PM	CLASSPATH File	4
.project	4/4/2018 2:28 PM	Text Document	1
build	4/4/2018 2:28 PM	PROJECT File	2
build_version	4/4/2018 2:28 PM	XML Document	17
build	4/4/2018 2:28 PM	PROPERTIES File	17
build	4/4/2018 2:28 PM	XML Document	38
build_version	4/4/2018 2:28 PM	PROPERTIES File	1
deploy-int	4/4/2018 2:28 PM	PROPERTIES File	1
deploy-test	4/4/2018 2:28 PM	PROPERTIES File	1
testingtest	4/4/2018 2:29 PM	Text Document	1

- 'testingtest.txt' file looks like below,



```
test is tested↓
testing for full↓
↓
3rd comit←
```

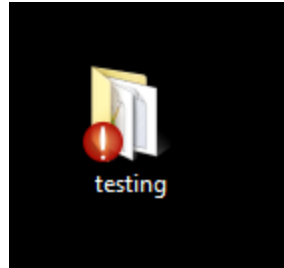
- Modified the content of the 'testingtest.txt' file



```
test is tested↓
testing for full↓
↓
3rd commit↓
↓
4th commit←
```

- Post performing changes in the file, cloned Branch folder appears as shown in below image

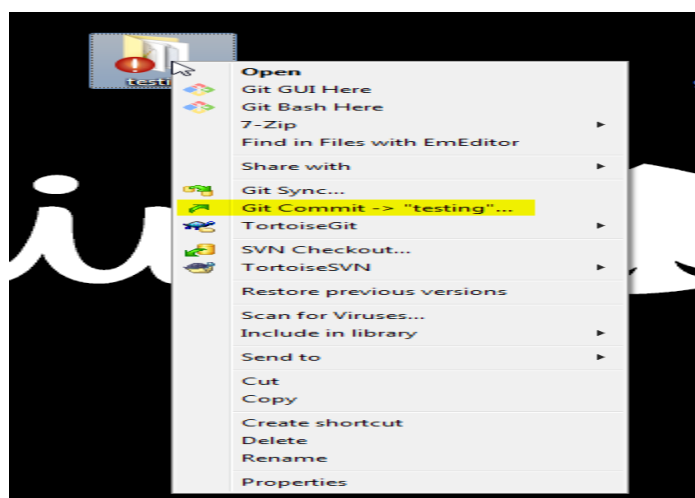
Name	Date modified	Type	Size
.git	4/4/2018 2:29 PM	File folder	
.settings	4/4/2018 2:28 PM	File folder	
client	4/4/2018 2:28 PM	File folder	
config	4/4/2018 2:28 PM	File folder	
db	4/4/2018 2:28 PM	File folder	
ddl	4/4/2018 2:28 PM	File folder	
demo	4/4/2018 2:28 PM	File folder	
doc	4/4/2018 2:28 PM	File folder	
etl	4/4/2018 2:28 PM	File folder	
lib	4/4/2018 2:29 PM	File folder	
src	4/4/2018 2:29 PM	File folder	
test	4/4/2018 2:29 PM	File folder	
web	4/4/2018 2:29 PM	File folder	
web-datepicker	4/4/2018 2:29 PM	File folder	
web-reporting	4/4/2018 2:29 PM	File folder	
.checkstyle	4/4/2018 2:28 PM	CHECKSTYLE File	1 KB
.classpath	4/4/2018 2:28 PM	CLASSPATH File	4 KB
.project	4/4/2018 2:28 PM	Text Document	1 KB
bmw_checkstyle_V5	4/4/2018 2:28 PM	PROJECT File	2 KB
build	4/4/2018 2:28 PM	XML Document	17 KB
build	4/4/2018 2:28 PM	PROPERTIES File	17 KB
build_version	4/4/2018 2:28 PM	XML Document	38 KB
build_version	4/4/2018 2:28 PM	PROPERTIES File	1 KB
deploy-int	4/4/2018 2:28 PM	PROPERTIES File	1 KB
deploy-test	4/4/2018 2:28 PM	PROPERTIES File	1 KB
testingtest	4/4/2018 2:37 PM	Text Document	1 KB



- Before committing the changes to the Remote Git Repository, see the file 'testingtest.txt' as shown below

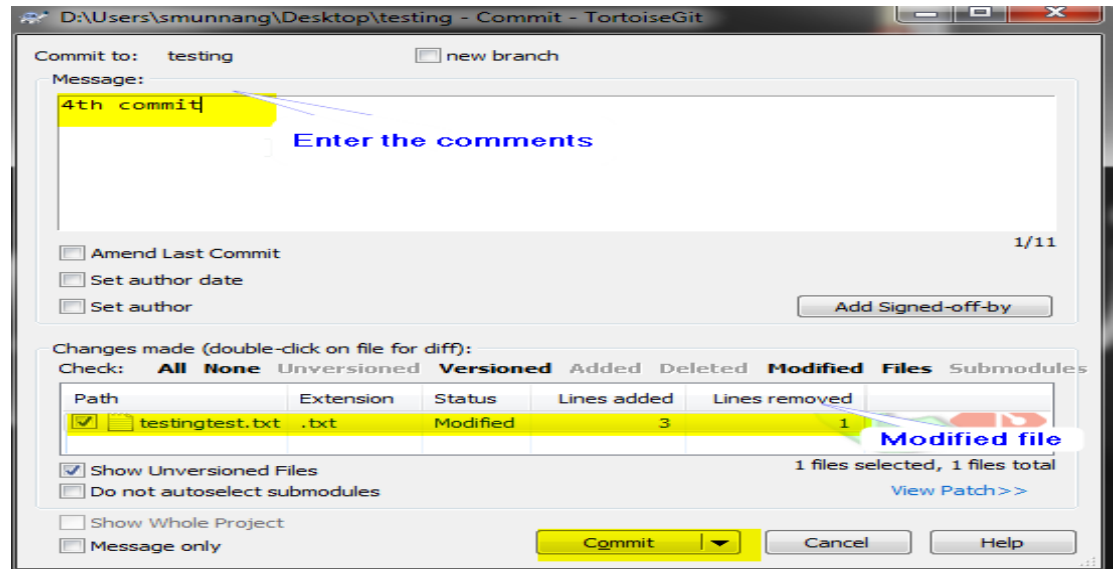


- To commit the changes made in Branch, right click on cloned Branch folder. Go to →Git Commit

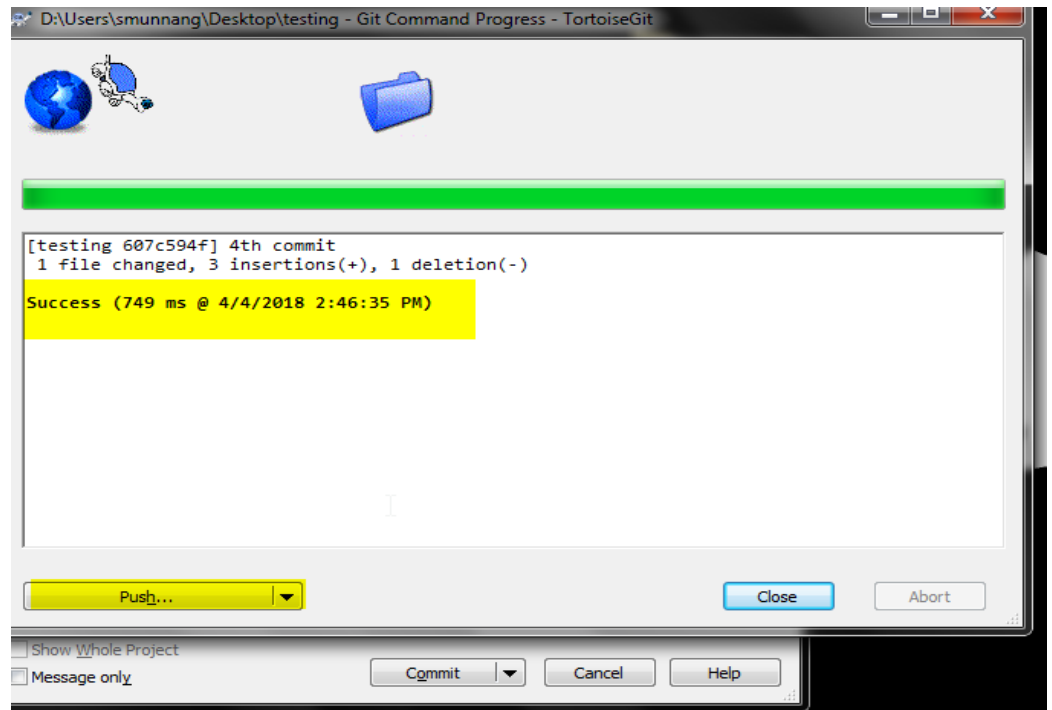


- Enter comments under 'Message' and you can view the changed files list at the bottom of the window as shown below,

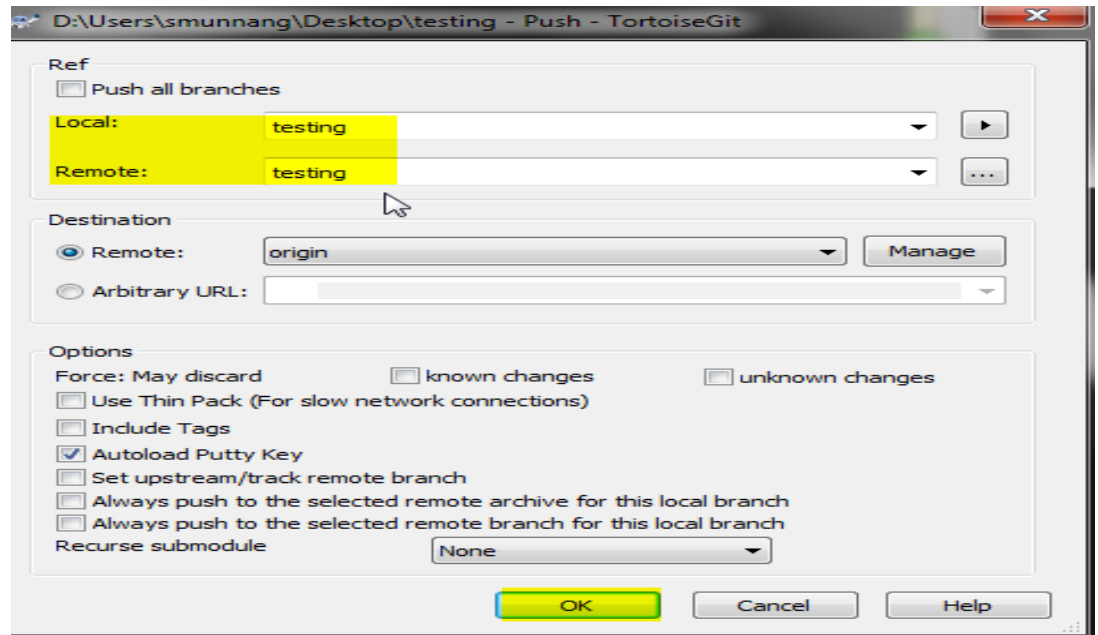
Click on 'Commit'



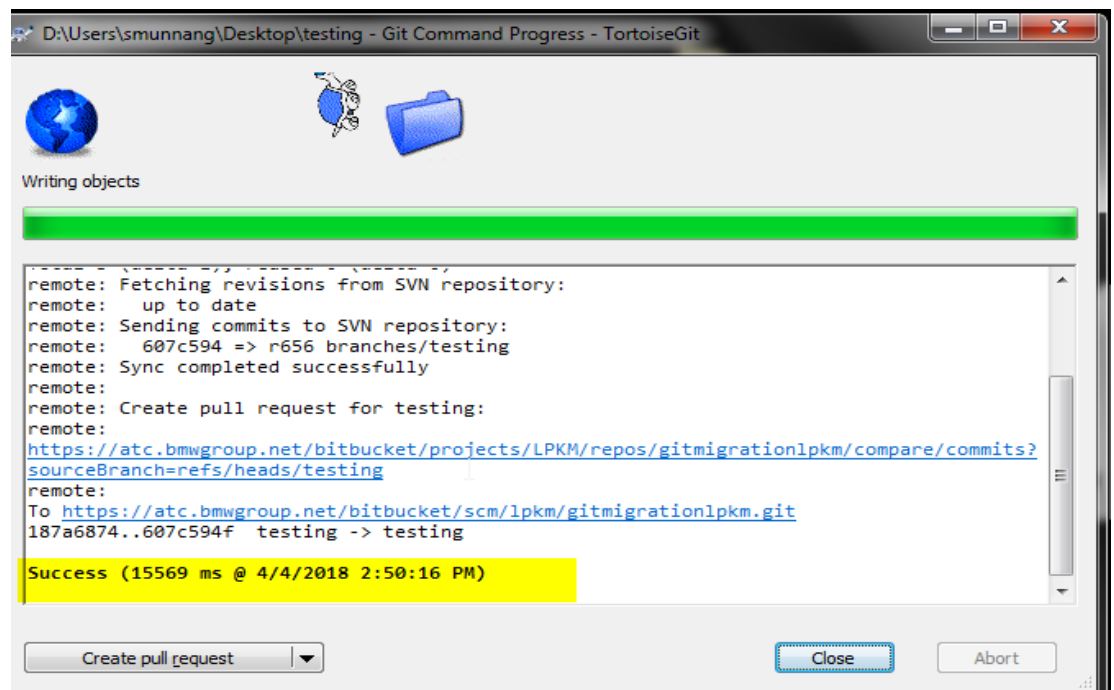
- The below screen appears post commit of changes. Click on 'Push' so that changes will be pushed to Branch of Git repository.



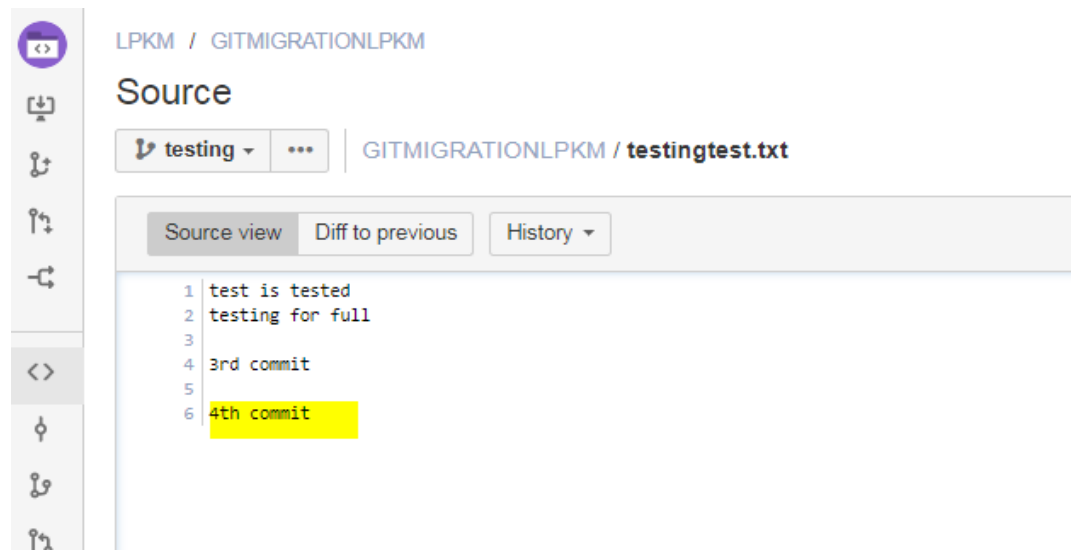
- The below screen appears. Click on 'OK'



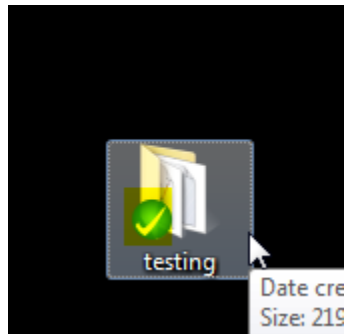
- Then success message appears once the push request is fulfilled.



- Check the testingtest.txt' file in the branch of Git repository to see the changes



- Post commit of the changes in the branch, cloned branch folder appears as in below image



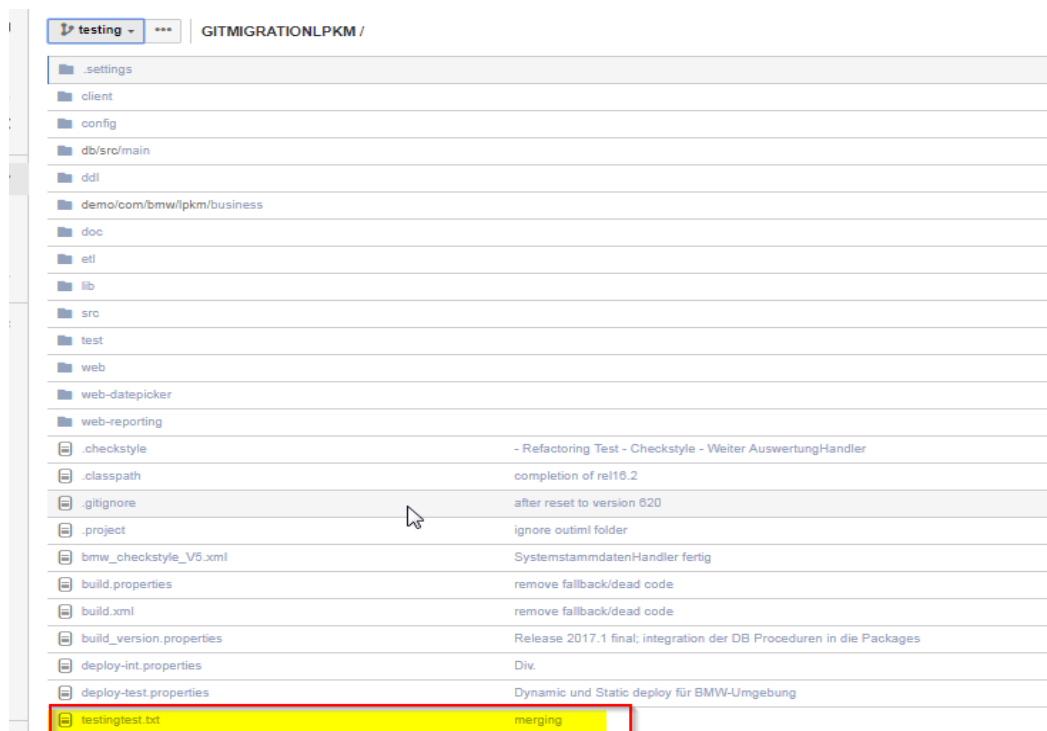
3.3 Merge

Note :

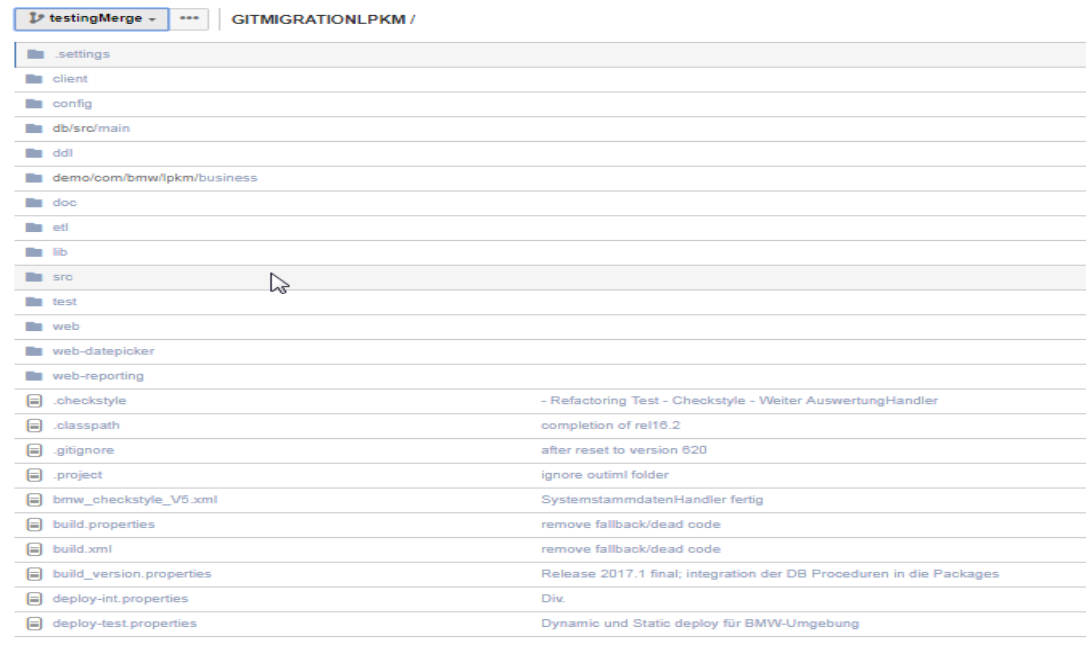
- In general, Git repository has an option to sync with SVN
- **To 'Merge' the Branches, the sync option in Git repository should be disabled**

Merge of 'testing' Branch with 'testingMerge' Branch

1. Refer the below screenshot for the contents of '**testing**' Branch in Git repository with 'testingtest.txt' file.

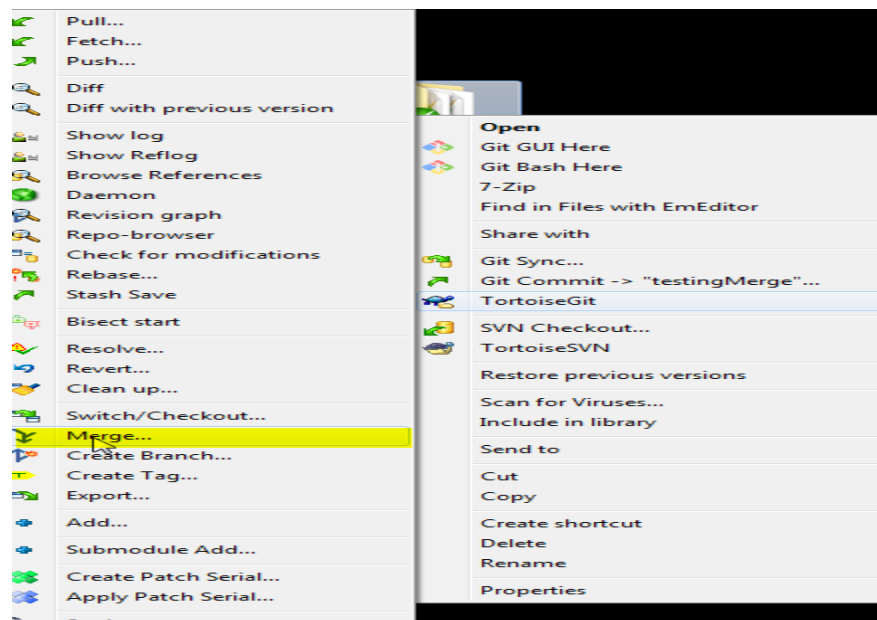


2. Refer the below screenshot for the contents of '**testingMerge**' Branch in Git repository without 'testingtest.txt' file.

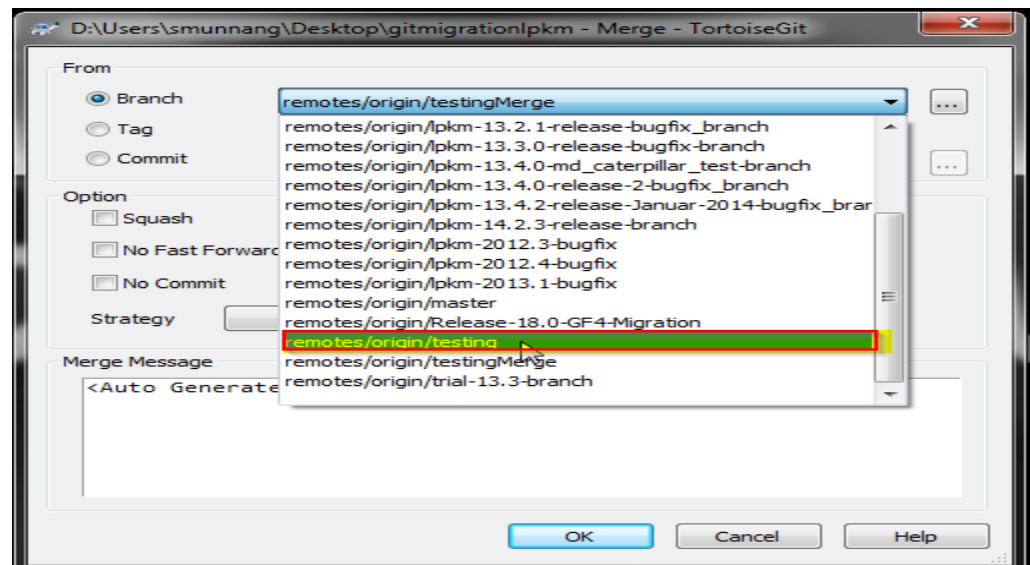


3. Right click on the cloned Branch['**testingMerge**'] and

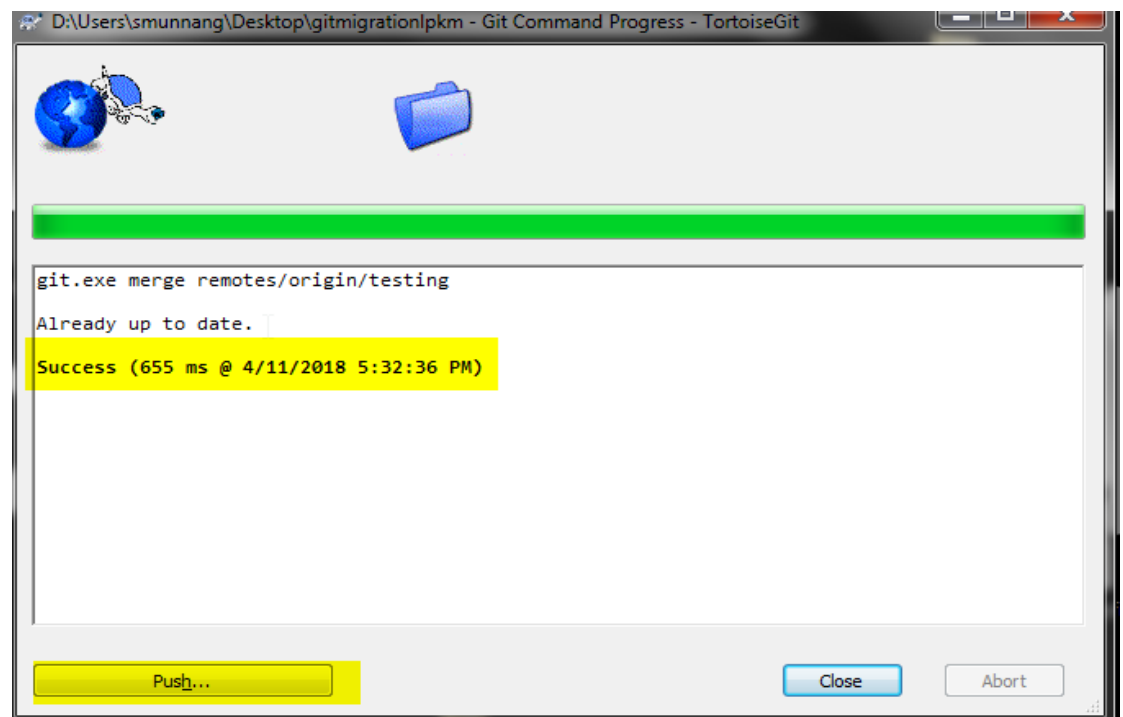
Go to →TortoiseGit →click 'Merge'



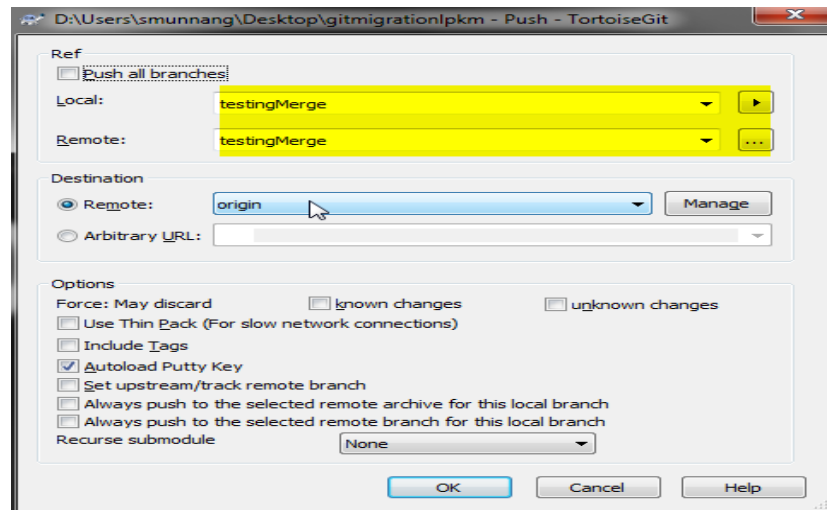
4. Select the '**testing**' Branch and click on 'OK'



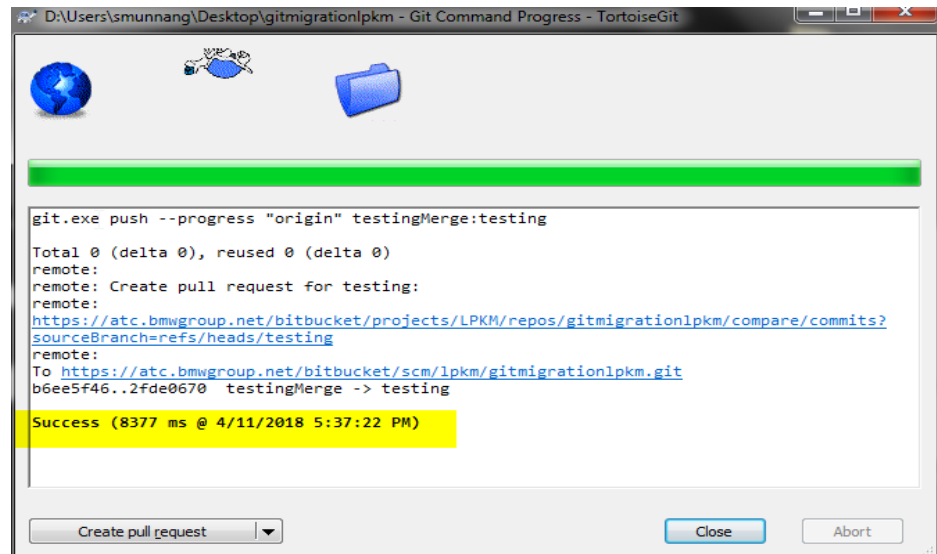
5. Then click on 'Push'



6. Select 'Local' & 'Remote' as '**testingMerge**' Branch and click on 'OK'



7. Success message appears as in below screenshot



8. Now the 'testingtest.txt' file of 'testing' Branch is merged into 'testingMerge' Branch in Git repository

