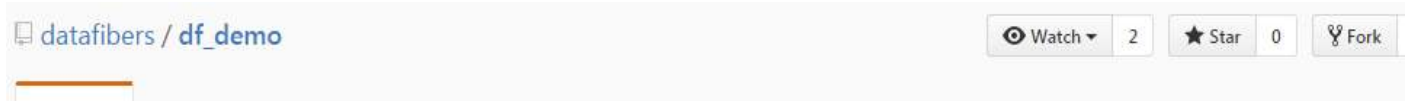
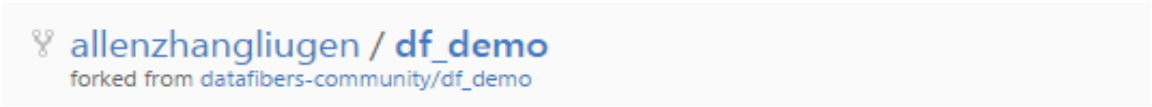
Use Github DataFibers Repository

0. Download and Install Git for windows (Git-2.10.1-64-bit.exe)
   https://git-scm.com/download/win

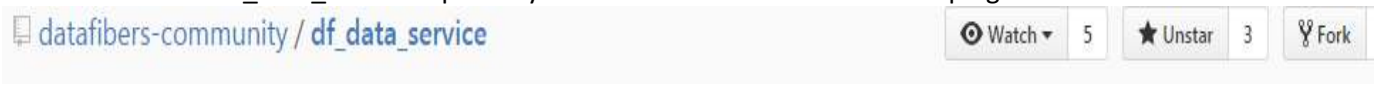
1. Login DataFibers Github, find out the repository, for exmaple df_demo. fork the repository for you own copy



In this case, df_demo in your own account is forked from datafibers/df_demo.
The df_demo repository contains DataFibers installation contents.

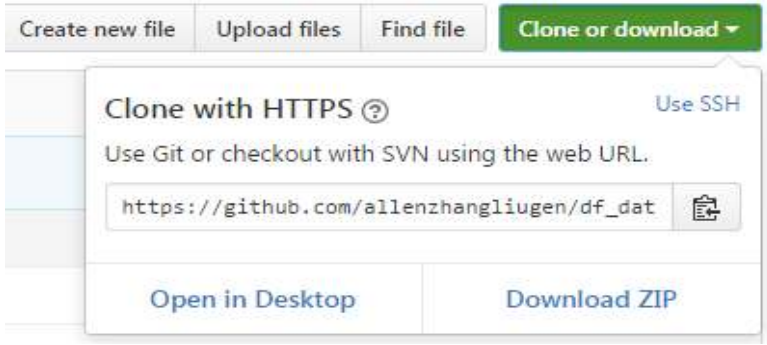


You can also fork df_data_service repository which contains DataFibers developing code.



2. Your work should be within your own GitHub repository. Later on your work can be promoted to DataFibers repository.

To work in your df_data_service repository, get the URL by clicking Clone or download button



https://github.com/allenzhangliugen/df_data_service.git

3. Open Git Bash window

C:\Git\git-bash.exe

4. Goto your folder where you want to check out the code, Create a directory,

cd C/DataFibers

mkdir df_data_service

```
ALZHANG@ALZHANG-PC MINGW64 /
$ cd C/DataFibers

ALZHANG@ALZHANG-PC MINGW64 /C/DataFibers
$ mkdir df_data_service

ALZHANG@ALZHANG-PC MINGW64 /C/DataFibers
$
```

5. Go to the directory

cd df_data_service

6. Git clone to clone the repository

git clone https://github.com/allenzhangliugen/df_data_service.git

```
$  git clone https://github.com/allenzhangliugen/df_data_service.git
Cloning into 'df_data_service'...
remote: Counting objects: 692, done.
remote: Compressing objects: 100% (331/331), done.
remote: Total 692 (delta 167), reused 0 (delta 0), pack-reused 294
Receiving objects: 100% (692/692), 3.20 MiB | 2.92 MiB/s, done.
Resolving deltas: 100% (282/282), done.
```

7. The repository is in local drive and in there df_data_service folder is created. Go to df_data_service folder

```
ALZHANG@ALZHANG-PC MINGW64 /C/DataFibers/df_data_service
$ ls
df_data_service/

ALZHANG@ALZHANG-PC MINGW64 /C/DataFibers/df_data_service
$ cd df_data_service

ALZHANG@ALZHANG-PC MINGW64 /C/DataFibers/df_data_service/df_data_service (master)
$
```

ls to show the files

```
$ ls
build_and_run.sh*  conf/  DESIGN.md  df-data-service.iml  LICENSE  pom.xml  README.md  src/

ALZHANG@ALZHANG-PC MINGW64 /C/DataFibers/df_data_service/df_data_service (master)
$
```

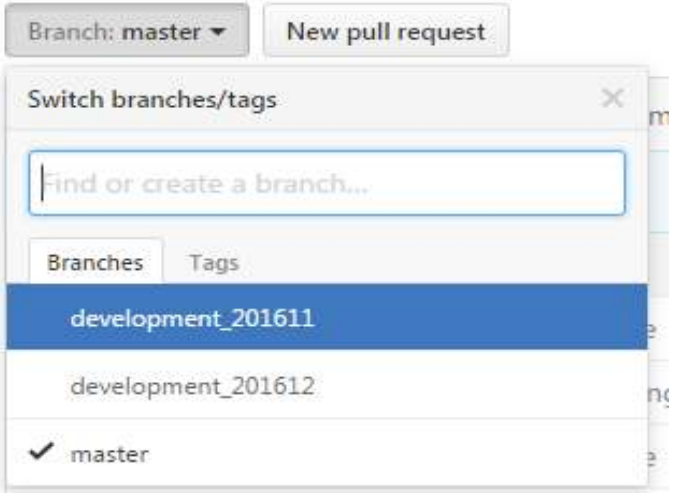
8. Notice that the folder shows as a master branch.

/C/DataFibers/df_data_service/df_data_service (master)

```
/C/DataFibers/df_data_service/df_data_service (master)
```

9. Usually the master branch is for stable codes. Your progress codes are in development branch.

From GitHub, click Branch dropdown to show avaiable branches



10. Check out the branch. Any work done after checking out will be in this branch.

git checkout development_201612

```
ALZHANG@ALZHANG-PC MINGW64 /C/DataFibers/df_data_service/df_data_service (master)
$ git checkout development_201612
Branch development_201612 set up to track remote branch development_201612 from origin.
Switched to a new branch 'development_201612'

ALZHANG@ALZHANG-PC MINGW64 /C/DataFibers/df_data_service/df_data_service (development_201612)
$
```

You can use git checkout to switch to different branches.

11. Add a new file, for example myChanges.txt in your local folder

```
ALZHANG@ALZHANG-PC MINGW64 /C/DataFibers/df_data_service/df_data_service (development_201612)
$ ls
build_and_run.sh*  DESIGN.md            LICENSE       pom.xml    src/
conf/              df-data-service.iml  myChanges.txt  README.md
```

12. git status to find out the changes

```
$ git status
On branch development_201612
Your branch is up-to-date with 'origin/development_201612'.
Untracked files:
  (use "git add <file>..." to include in what will be committed)

        myChanges.txt

nothing added to commit but untracked files present (use "git add" to track)

ALZHANG@ALZHANG-PC MINGW64 /C/DataFibers/df_data_service/df_data_service (development_201612)
$ |
```

13. Staging the files to be checked into repository

git add myChanges.txt

```
$ git add myChanges.txt

ALZHANG@ALZHANG-PC MINGW64 /C/DataFibers/df_data_service/df_data_service (development_201612)
$ |
```

git status shows that the changes are staged.

```
$ git status
On branch development_201612
Your branch is up-to-date with 'origin/development_201612'.
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        new file:   myChanges.txt
```

14.Set your account to default identity

git config --global user.email "you@example.com"

git config --global user.name "Your Name"

```
$ git config user.email "allen.zhangliugen@gmail.com"

ALZHANG@ALZHANG-PC MINGW64 /C/DataFibers/df_data_service/df_data_service (development_201612)
$ git config user.name "Allen Zhang"

ALZHANG@ALZHANG-PC MINGW64 /C/DataFibers/df_data_service/df_data_service (development_201612)
$ |
```

15. Commit the code change to local repository

git commit -m "Check in my changes" myChanges.txt

```
$  git commit -m "Check in my changes" myChanges.txt
[development_201612 bafff3a] Check in my changes
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 myChanges.txt
```

16. To avoid conflict, use git fetch and git pull to get and/or merge the latest changes from remote repository

git fetch

git pull

```
ALZHANG@ALZHANG-PC MINGW64 /C/DataFibers/df_data_service/df_data_service (development_201612)
$ git fetch

ALZHANG@ALZHANG-PC MINGW64 /C/DataFibers/df_data_service/df_data_service (development_201612)
$ git pull
Already up-to-date.

ALZHANG@ALZHANG-PC MINGW64 /C/DataFibers/df_data_service/df_data_service (development_201612)
$
```

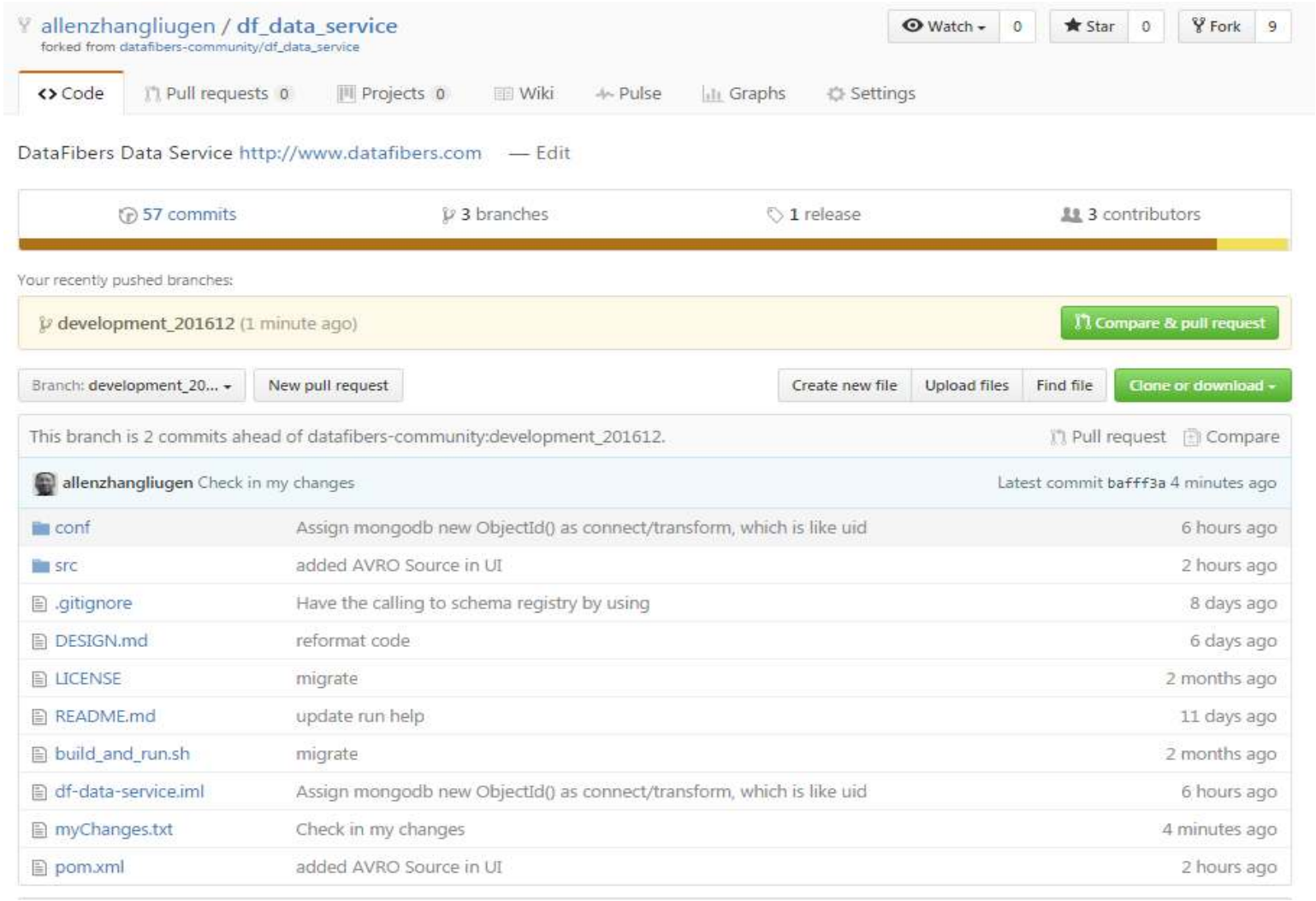
17. To publish the code change to remote repository

git push

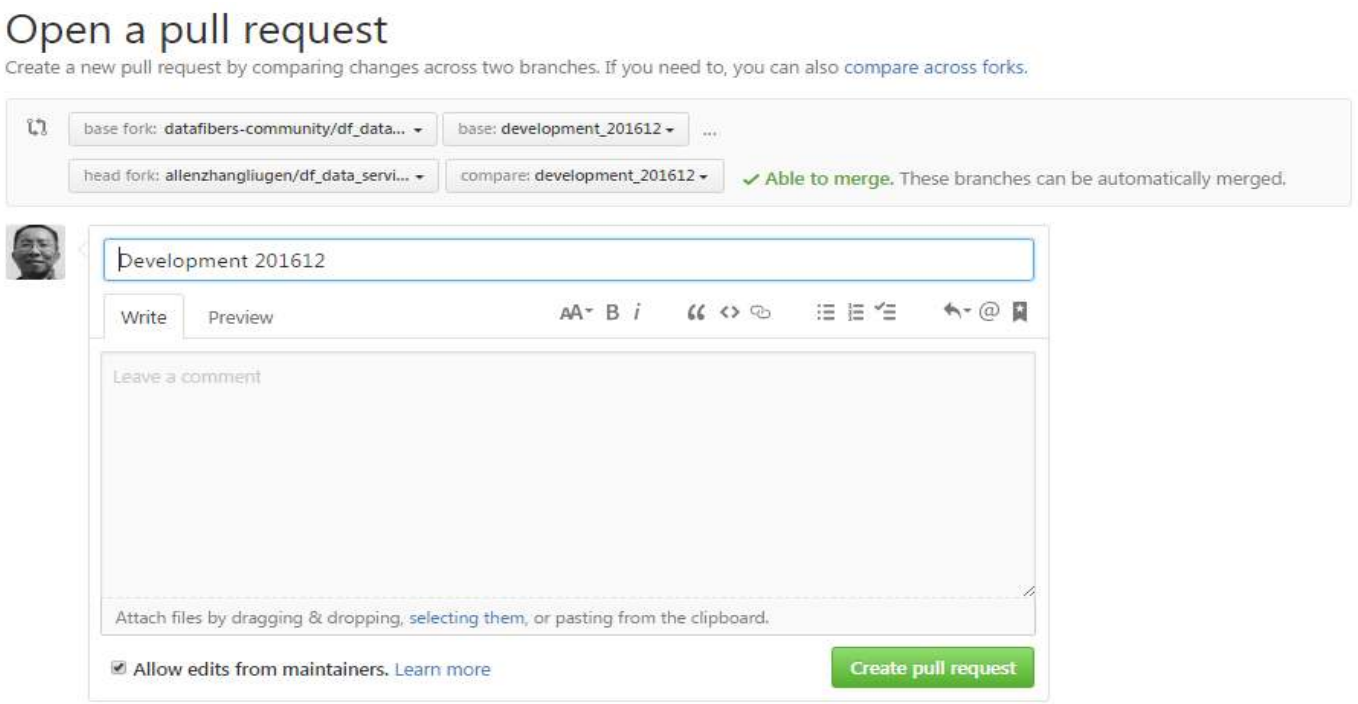
```
$ git push
Counting objects: 3, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 285 bytes | 0 bytes/s, done.
Total 3 (delta 1), reused 1 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local objects.
To https://github.com/allenzhangliugen/df_data_service.git
   ed7ea92..bafff3a  development_201612 -> development_201612

ALZHANG@ALZHANG-PC MINGW64 /C/DataFibers/df_data_service/df_data_service (development_201612)
$
```

18. You can find the latest changes are in your GitHub repository

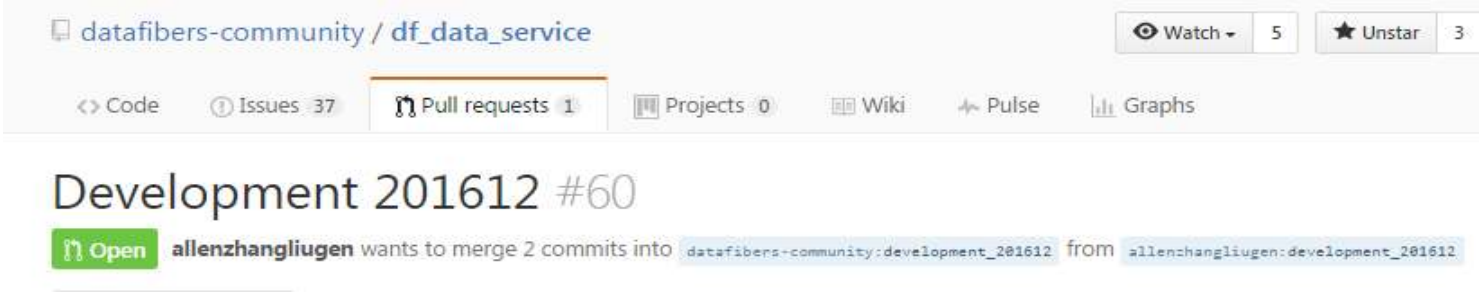
The file myChanges.txt is in the repository.
To promote your code to DataFilbers reposiotry, click compare & pull request button



19. A promote request will be created after you click create pull request button



20. Your request will be reviewed to merge in DataFibers GitHub branch



21. To view remote repository

origin repository is Your Github account repository

git remote -v

```
$ git remote -v
origin  https://github.com/allenzhangliugen/df_data_service.git (fetch)
origin  https://github.com/allenzhangliugen/df_data_service.git (push)
```

22. Add DataFiber remote upstream repository

get the DataFiber repository URL which Your GitHub account repository was forked from

git remote add upstream https://github.com/datafibers-community/df_data_service.git

```
$ git remote add upstream https://github.com/datafibers-community/df_data_service.git
```

now both origin and upstream repository are shown.

```
$ git remote -v
origin  https://github.com/allenzhangliugen/df_data_service.git (fetch)
origin  https://github.com/allenzhangliugen/df_data_service.git (push)
upstream        https://github.com/datafibers-community/df_data_service.git (fetch)
upstream        https://github.com/datafibers-community/df_data_service.git (push)
```

23. Fetch upstream branches

To fetch the branches from DataFibers repository

git fetch upstream

```
$ git fetch upstream
remote: Counting objects: 31, done.
remote: Compressing objects: 100% (25/25), done.
remote: Total 31 (delta 6), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (31/31), done.
From https://github.com/datafibers-community/df_data_service
 * [new branch]      development_201611 -> upstream/development_201611
 * [new branch]      development_201612 -> upstream/development_201612
 * [new branch]      master             -> upstream/master
```

24. Pull latest code from DataFiber development branch

First checkout your local development branch, this branch will get the latest contents

git checkout development_201612

```
$ git checkout development_201612
Already on 'development_201612'
Your branch is up-to-date with 'origin/development_201612'.
```

Then pull the latest contents to local repository from DataFibers development branch.

git pull upstream development_201612

```
$ git pull upstream development_201612
From https://github.com/datafibers-community/df_data_service
 * branch            development_201612 -> FETCH_HEAD
Auto-merging src/main/java/com/datafibers/processor/SchemaRegisterProcessor.java
Merge made by the 'recursive' strategy.
 .../processor/KafkaConnectProcessor.java           |   5 +-
 ...erForward.java => SchemaRegisterProcessor.java} | 510 ++++++++++++----------
 .../com/datafibers/service/DFDataProcessor.java    |   6 +-
 src/main/resources/log4j.properties                |   2 +-
 4 files changed, 263 insertions(+), 260 deletions(-)
 rename src/main/java/com/datafibers/processor/{SchemaRegisterForward.java => SchemaRegisterProcessor.java} (95%)
```

25. Push from local repository to your GitHub account repository

To show the changes in local repository

```
$ git status
On branch development_201612
Your branch is ahead of 'origin/development_201612' by 5 commits.
  (use "git push" to publish your local commits)
nothing to commit, working tree clean
```

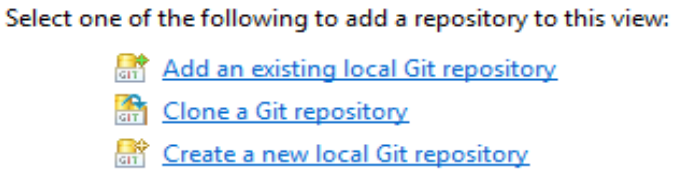
Now push to your Github account repository

git push

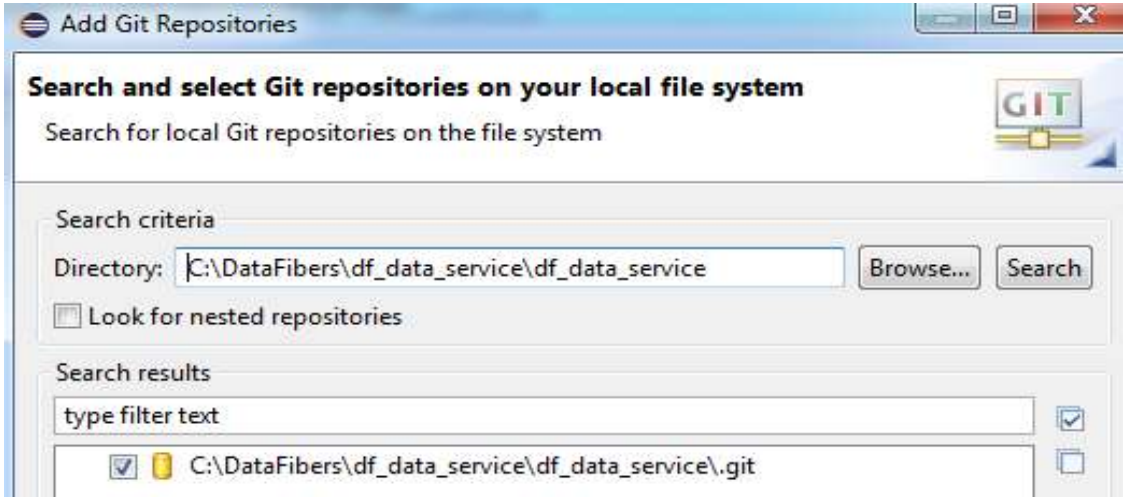
```
$ git push
Counting objects: 33, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (27/27), done.
Writing objects: 100% (33/33), 5.27 KiB | 0 bytes/s, done.
Total 33 (delta 13), reused 0 (delta 0)
remote: Resolving deltas: 100% (13/13), completed with 7 local objects.
To https://github.com/allenzhangliugen/df_data_service.git
   bafff3a..a093fdd  development_201612 -> development_201612
```

26. Import into Eclipse IDE

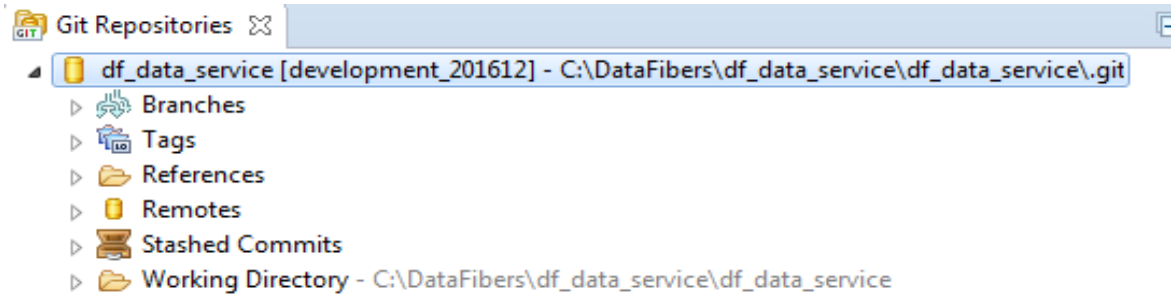
In Eclipse git repositories Perspective , choose adding an existing local Git repository



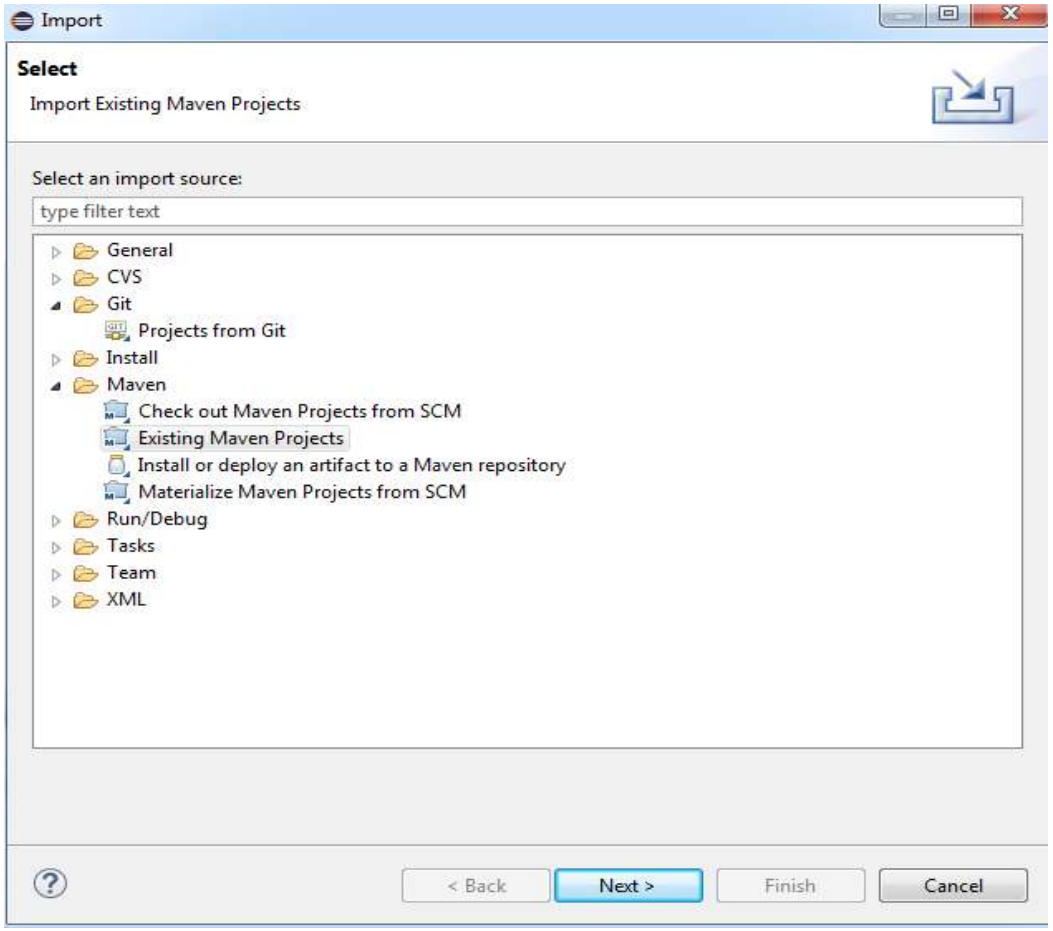
27. Browser to Add the local Git repository directory



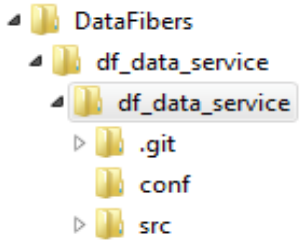
Click Finish. The repository is added



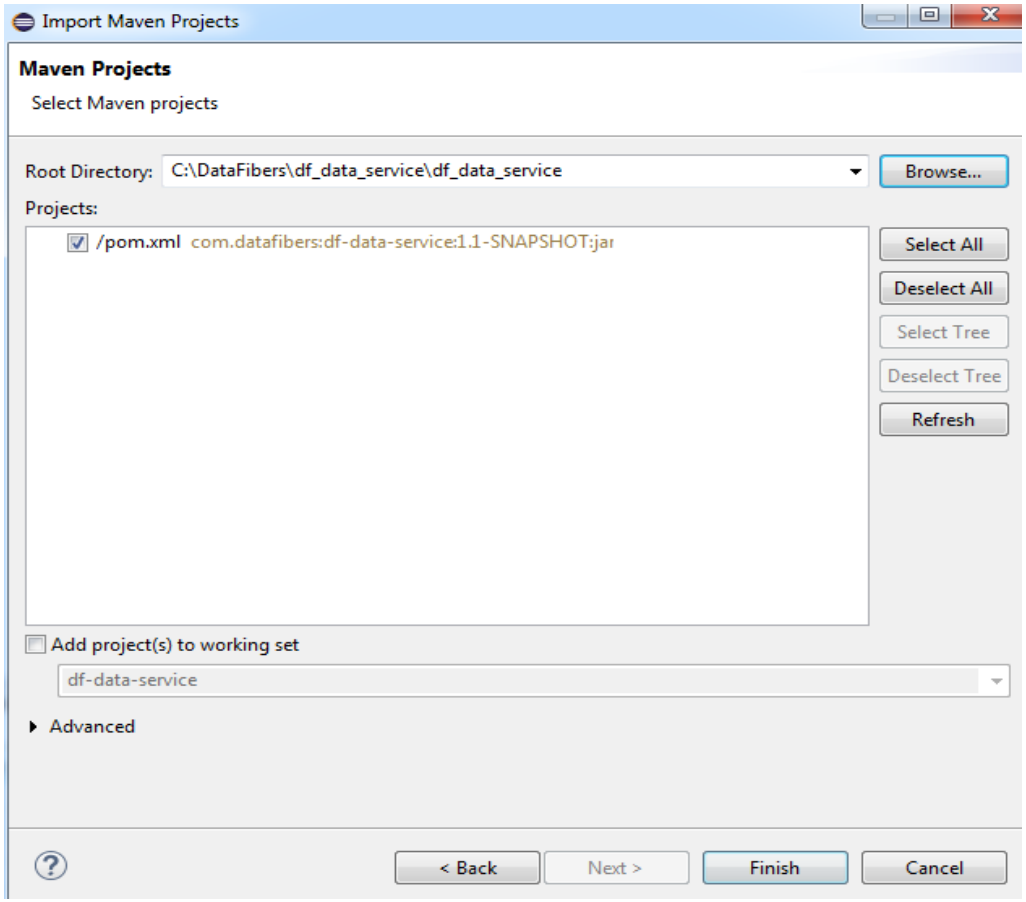
28. Import existing Maven Project



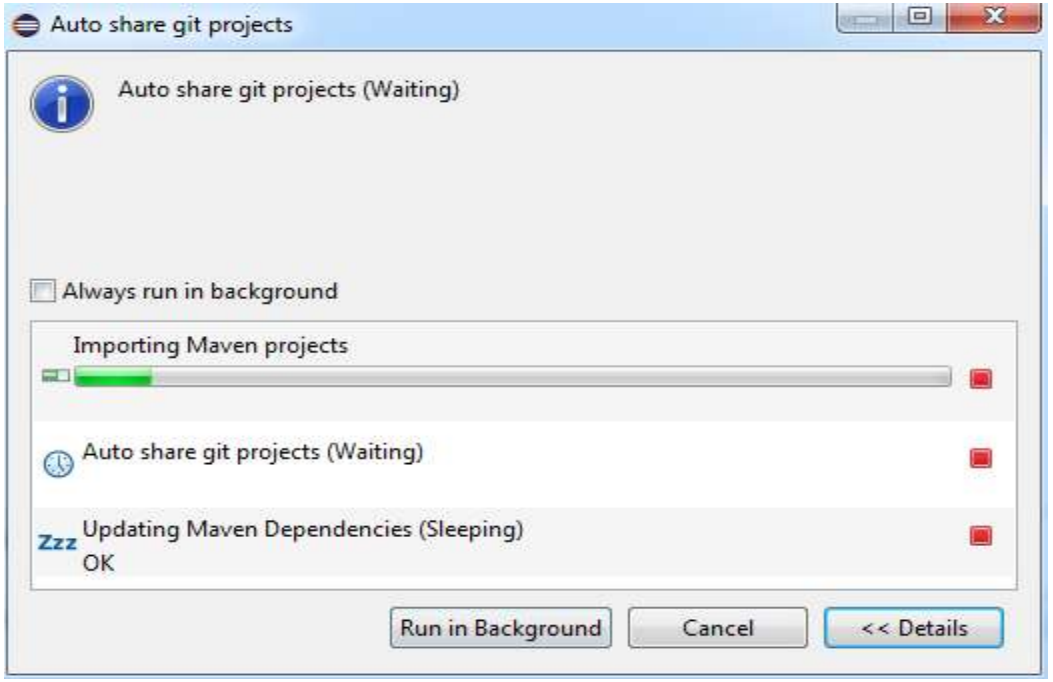
29. Specify the Root Directory.



30. Check the pom.xml



31. Click Finish to execute: Auto share git project



32. The project is imported into Eclipse