

|  |  |
| --- | --- |
| Document name | SVN Guide |
| Version no. | 1.0 |
| Release date |  |

|  |
| --- |
| Guidelines of Lunch Committee  (HR-GLC) |

|  |
| --- |
| Guidelines of Lunch Committee  (HR-GLC) |



SUBVERSION

| Version No. | Authored / Modified by | Reviewed by, Date | Approved by, Date | Remark / Change History |
| --- | --- | --- | --- | --- |
| 1.0 | Sakshi Kathuria  11-July-15 |  |  | Initial Document |

Contents

[VERSION CONTROL SYSTEM 2](#_Toc428198610)

[1. Version Control System (VCS) 3](#_Toc428198611)

[2. SVN vs GIT 3](#_Toc428198612)

[3. Version Control Terminologies: 3](#_Toc428198613)

[SVN ON WINDOWS 4](#_Toc428198614)

[1. Installation of SVN 4](#_Toc428198615)

[2. Create Repository 5](#_Toc428198616)

[3. Through command line. 6](#_Toc428198617)

[4. SVN Import 9](#_Toc428198618)

[5. Import through UI 12](#_Toc428198619)

[6. SVN Add 14](#_Toc428198620)

[7. SVN Commit 15](#_Toc428198621)

[8. SVN Checkout 16](#_Toc428198622)

[9. Checkout through UI 18](#_Toc428198623)

[10. How to checkout particular revision 21](#_Toc428198624)

[11. Merge Conflict 23](#_Toc428198625)

[12. Svn update 24](#_Toc428198626)

[13. SVN Resolved 25](#_Toc428198627)

[14. SVN Export 26](#_Toc428198628)

[SVN on Linux (CentOS) 28](#_Toc428198629)

# 

# VERSION CONTROL SYSTEM

1. Version Control System (VCS) is a software that helps software developers to work together and maintain a complete history of your work.

**Following are the goals of a version control system:**

* Allow developers to work simultaneously.
* Do not overwrite each other’s changes.
* Maintain history of every version of everything.

**VCS is divided into two categories:**

* Centralized Version Control system (CVCS).
* Distributed Version Control System (DVCS).

## SVN vs GIT

* GIT is distributed, SVN is not. SVN is a centralized repository.
* GIT stores content as metadata, SVN stores just files.
* GIT branches are not the same as SVN branches.
* GIT does not have a global revision number like SVN do.
* In SVN, a common workflow consists of the transfer (checkout) of a folder or a branch operation from the directory server to a local copy on the desktop. Thus, development is done only based on the files found in the directory structure, which is the developer's desktop. In Git the developer's repository consists of a full copy of the remote repository and a partition operation into subdirectories does not have the same meaning, as the checkout occurs after the clone, so all files are present on the client's file system.

**Subversion** is centralized version control system.

## Version Control Terminologies:

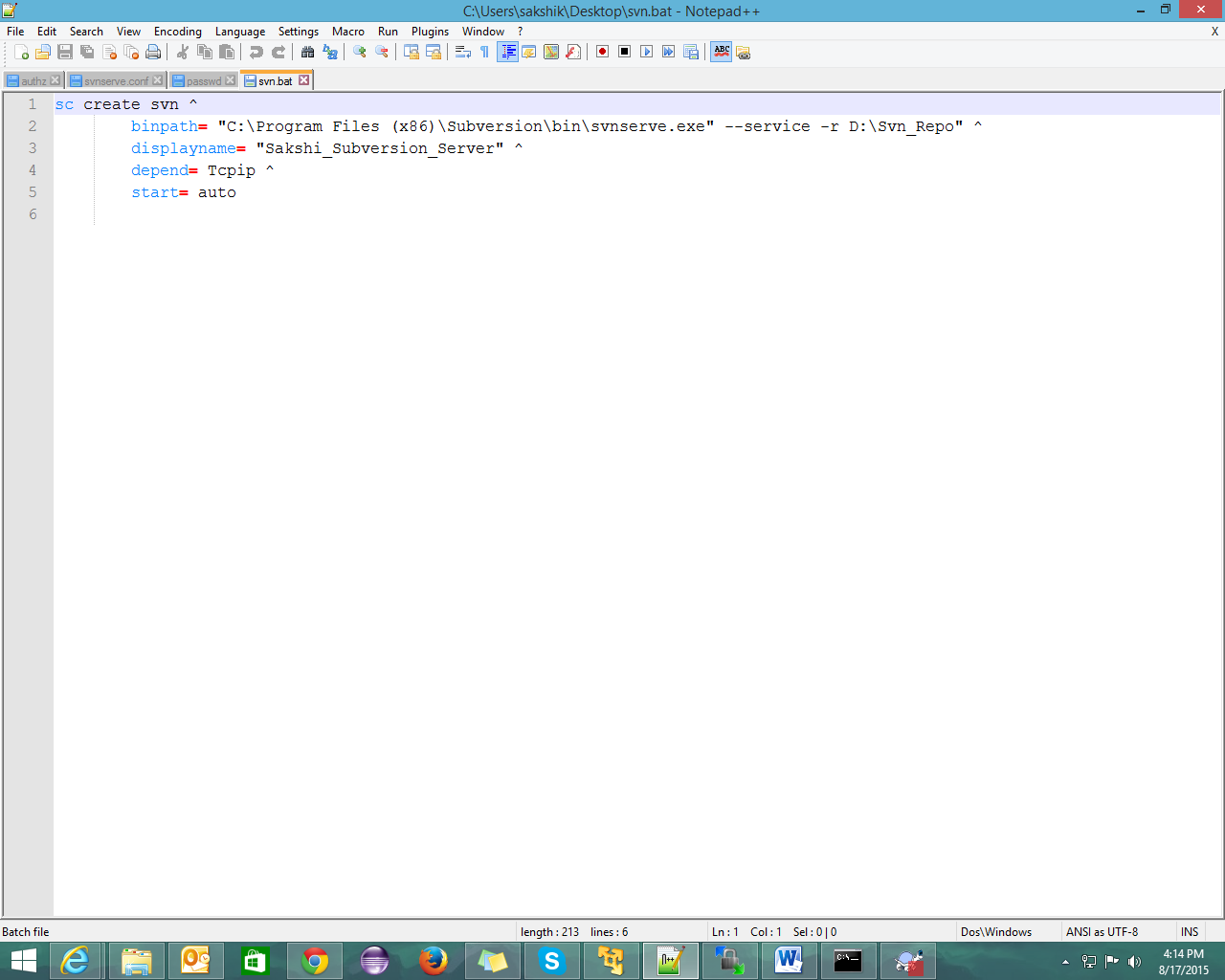
* Repositories
* Trunk
* Tags
* Branches
* Working copy
* Commit changes

# SVN ON WINDOWS

## Installation of SVN

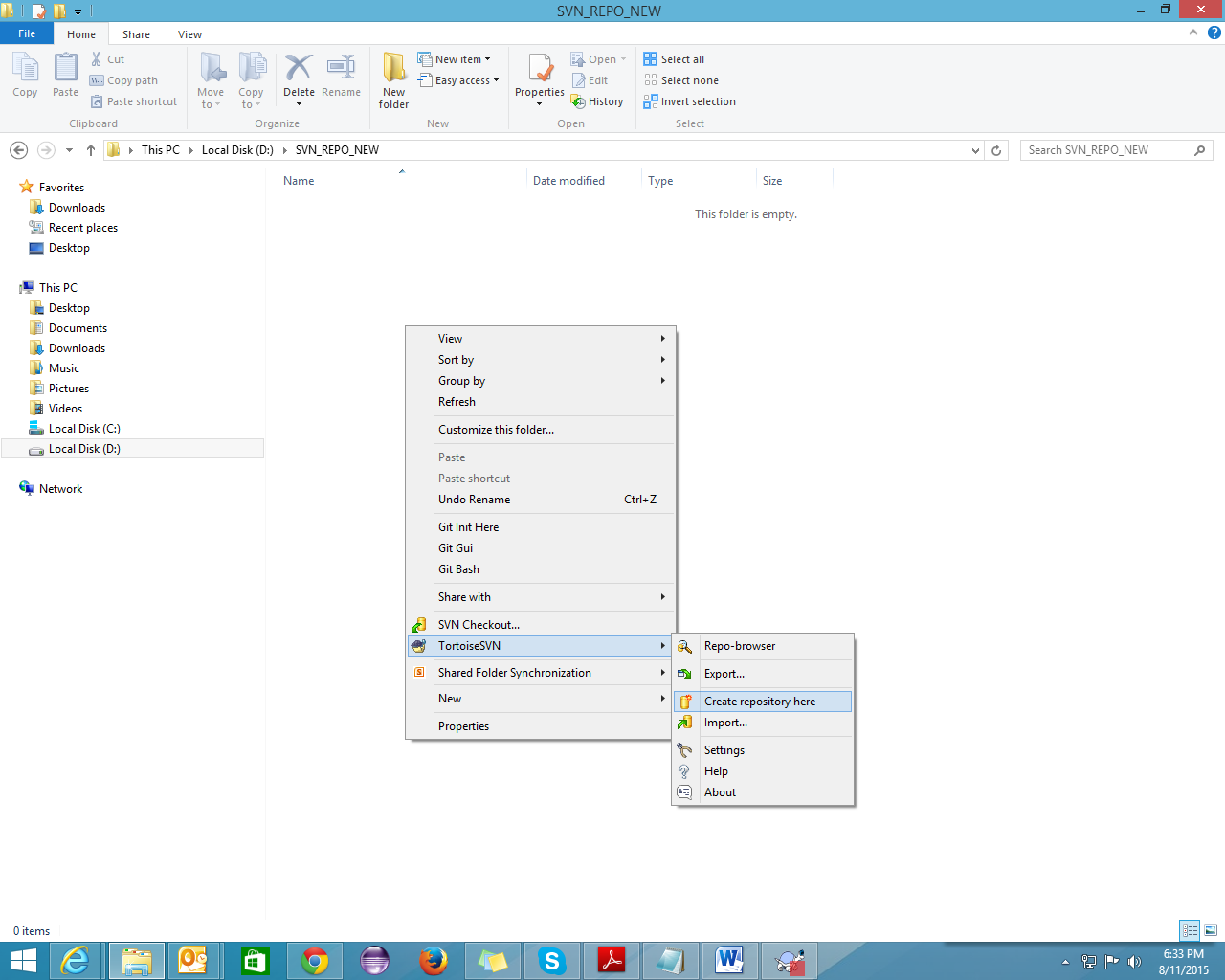
Installation of Tortoise SVN

Create service using svn tortoise (Svn.bat file)

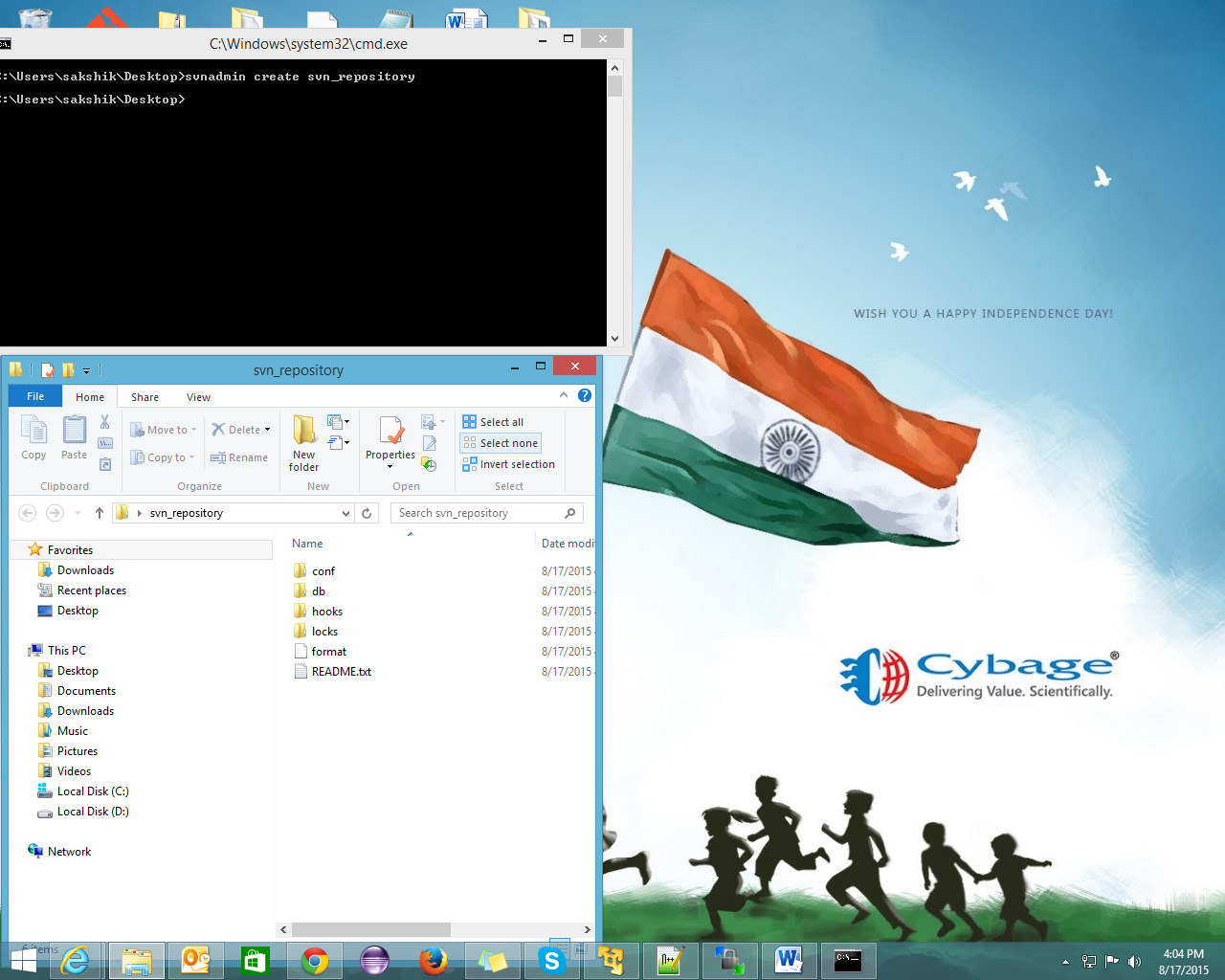


## Create Repository

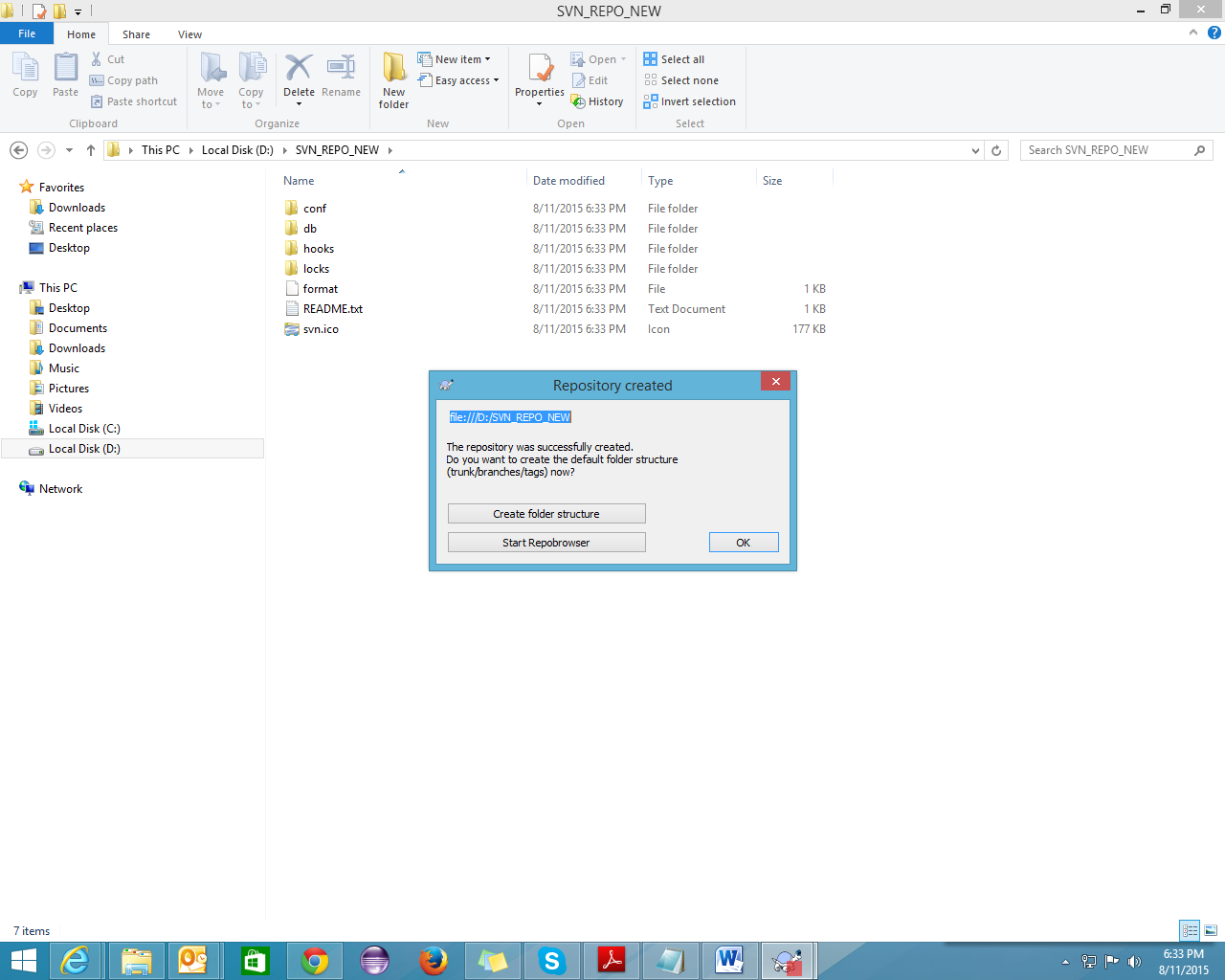
Create repository through tortoise svn



## Through command line.



By default, this folder structure will get created:



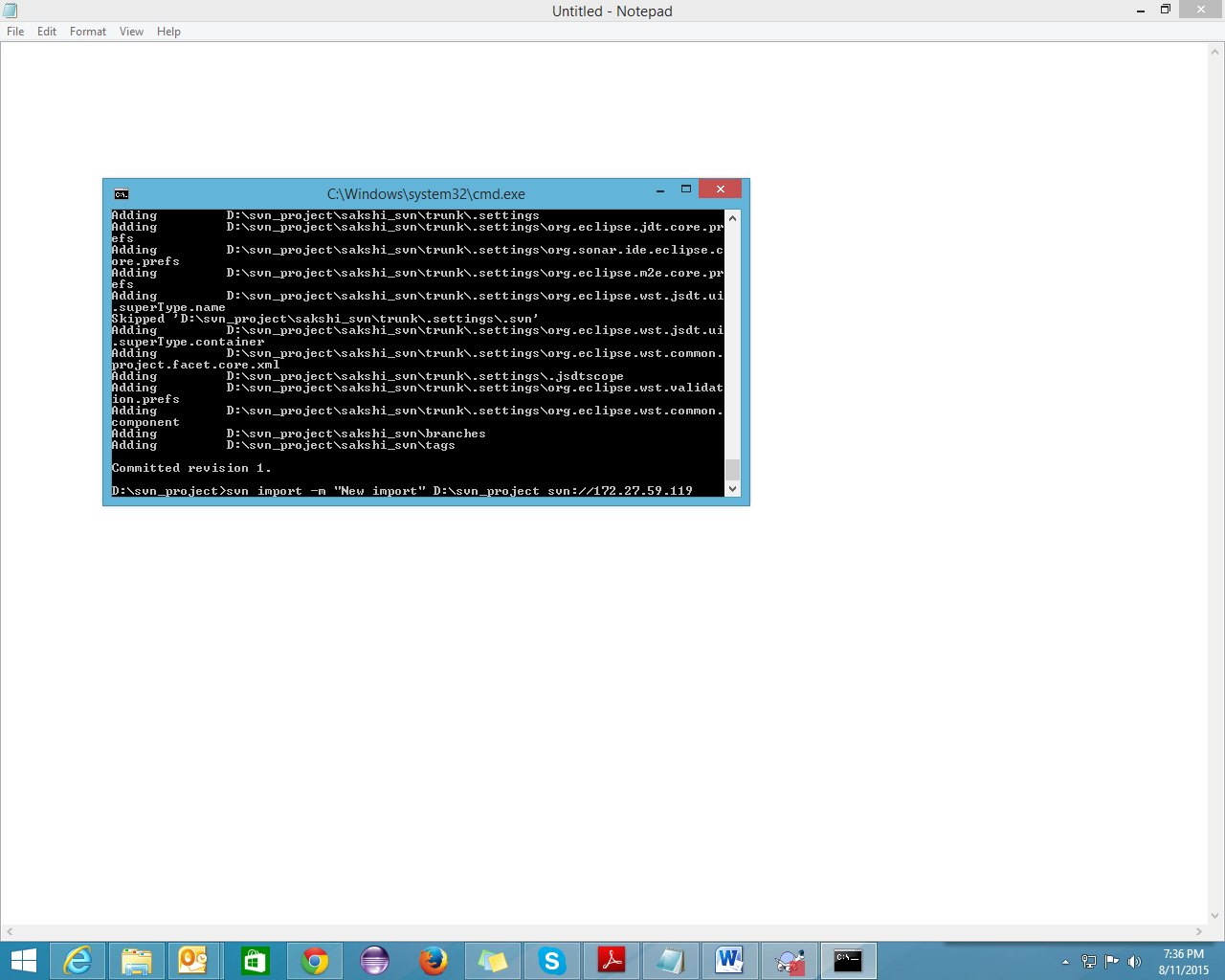
Somewhere in your hard drive create a directory containing following sub-directories:



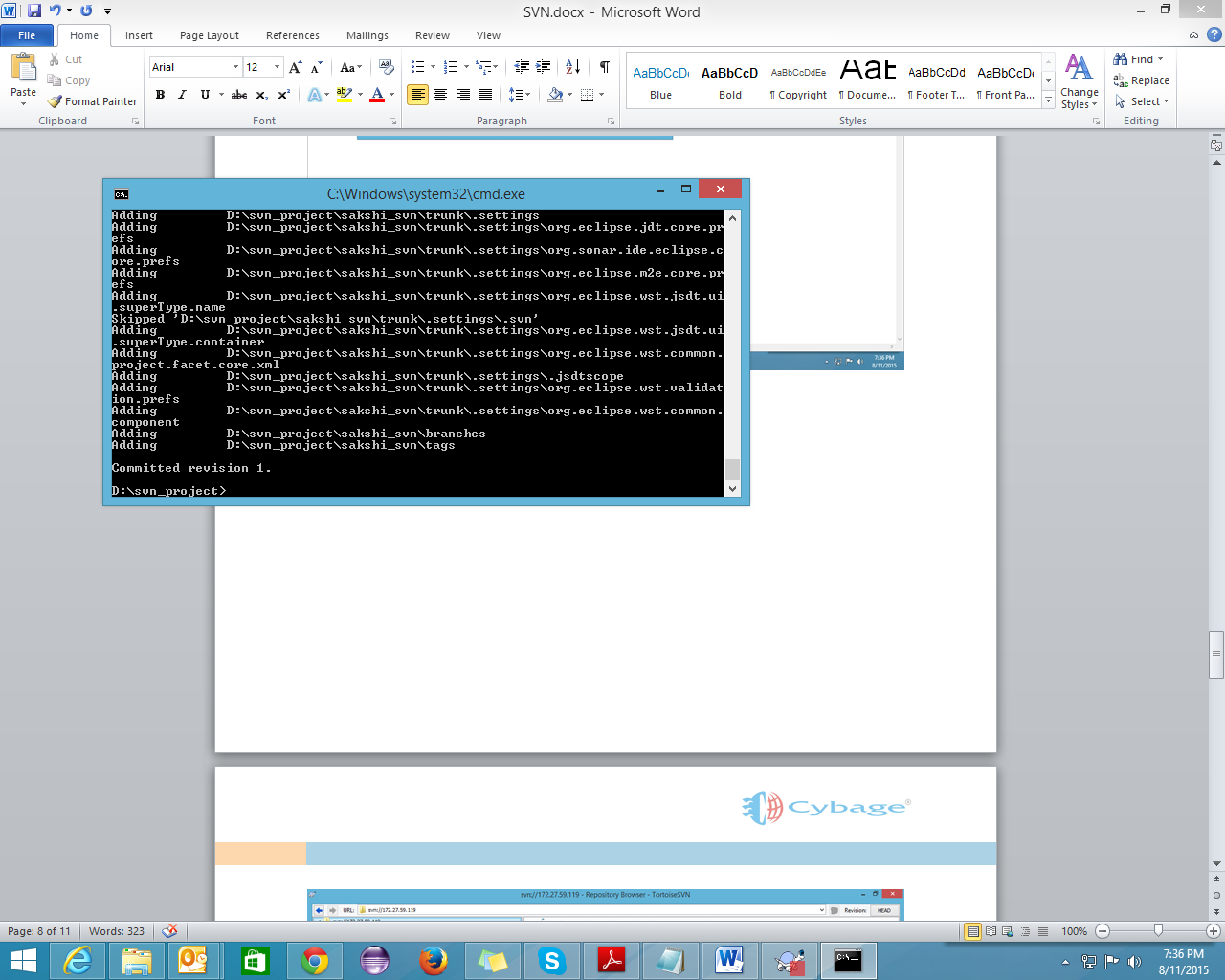
## SVN Import

Importing the folder structure into the repository through command line

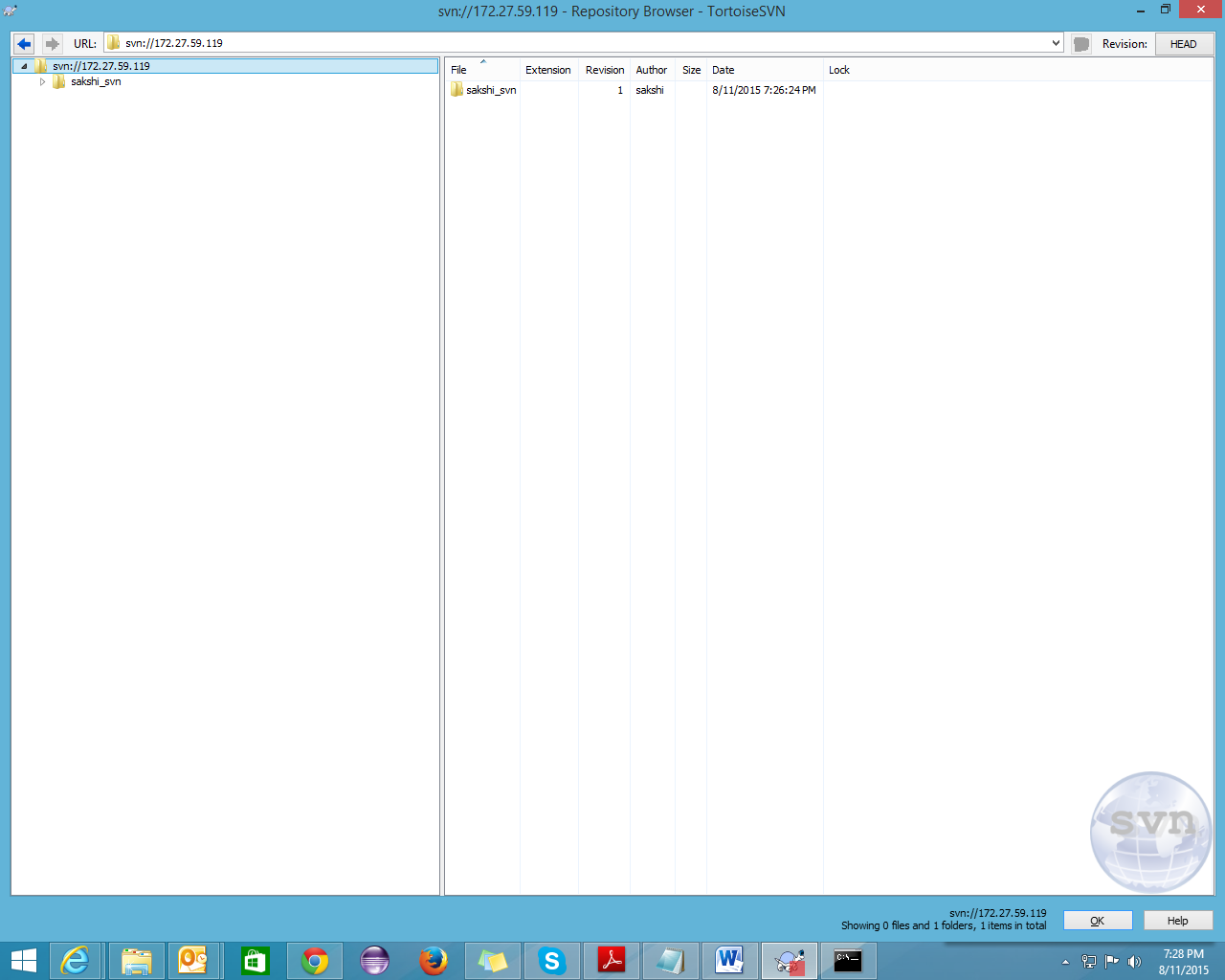
Svn import –m “New import” url (from where you want to import)



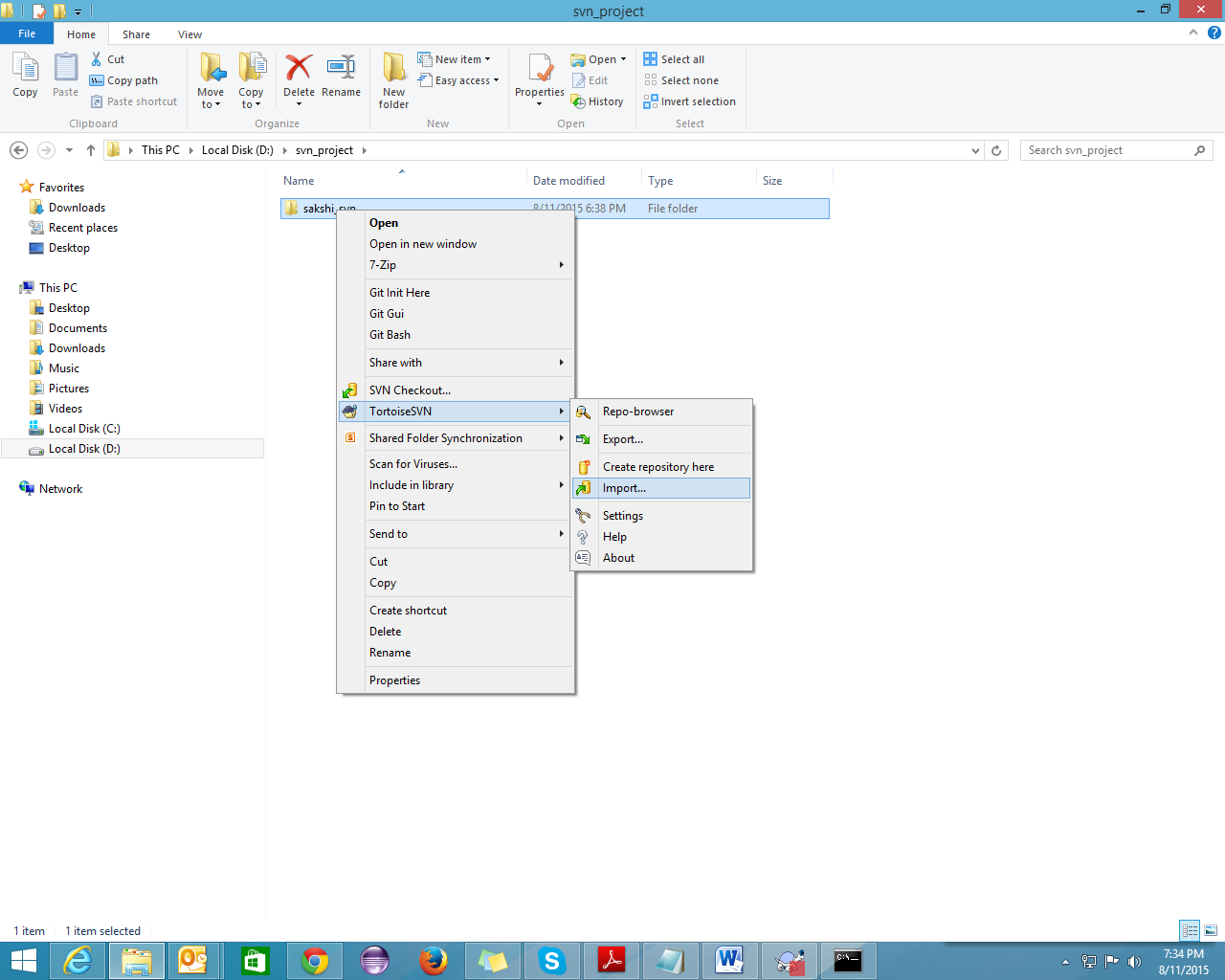
Import completed

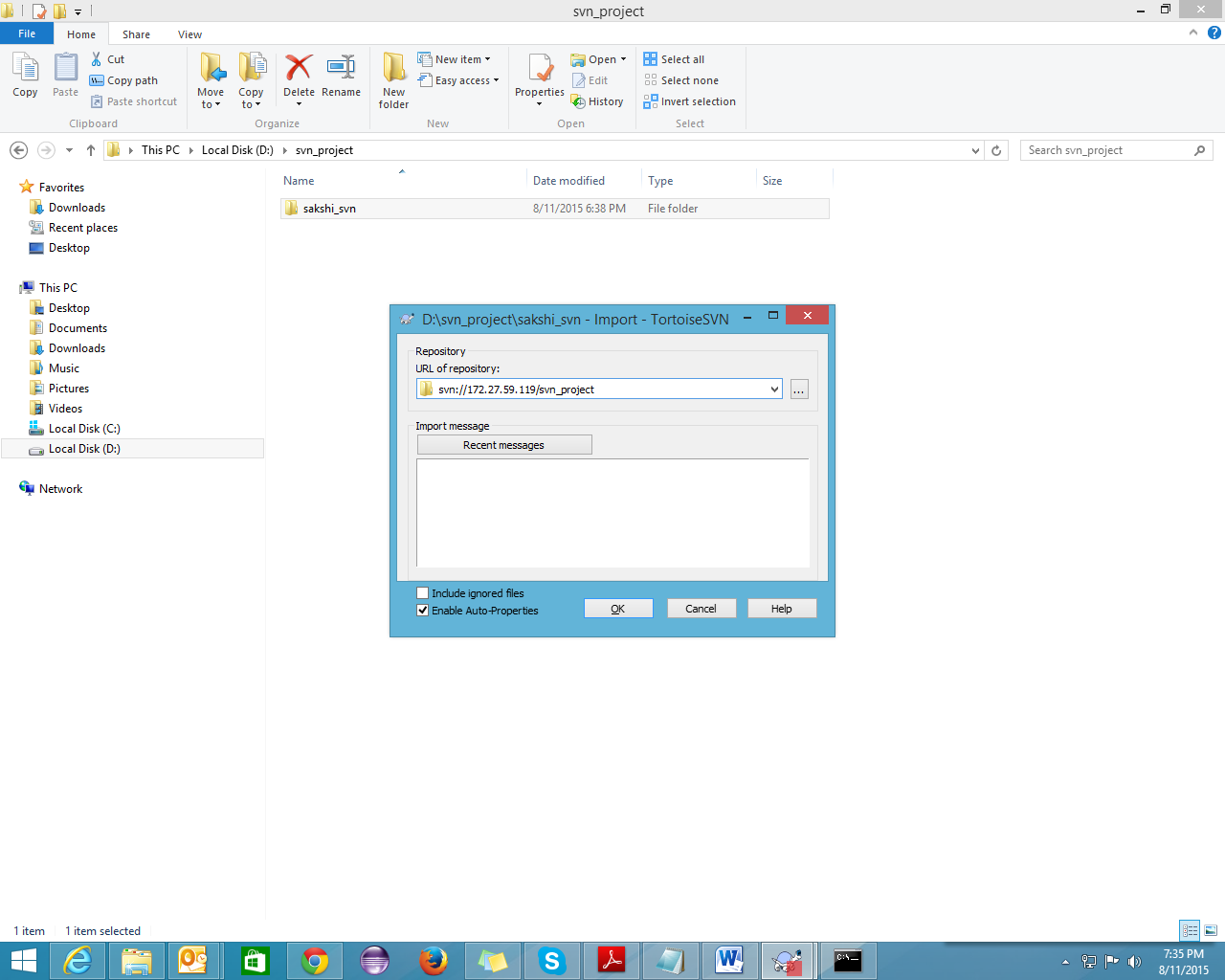


Will appear in your repository.



## Import through UI



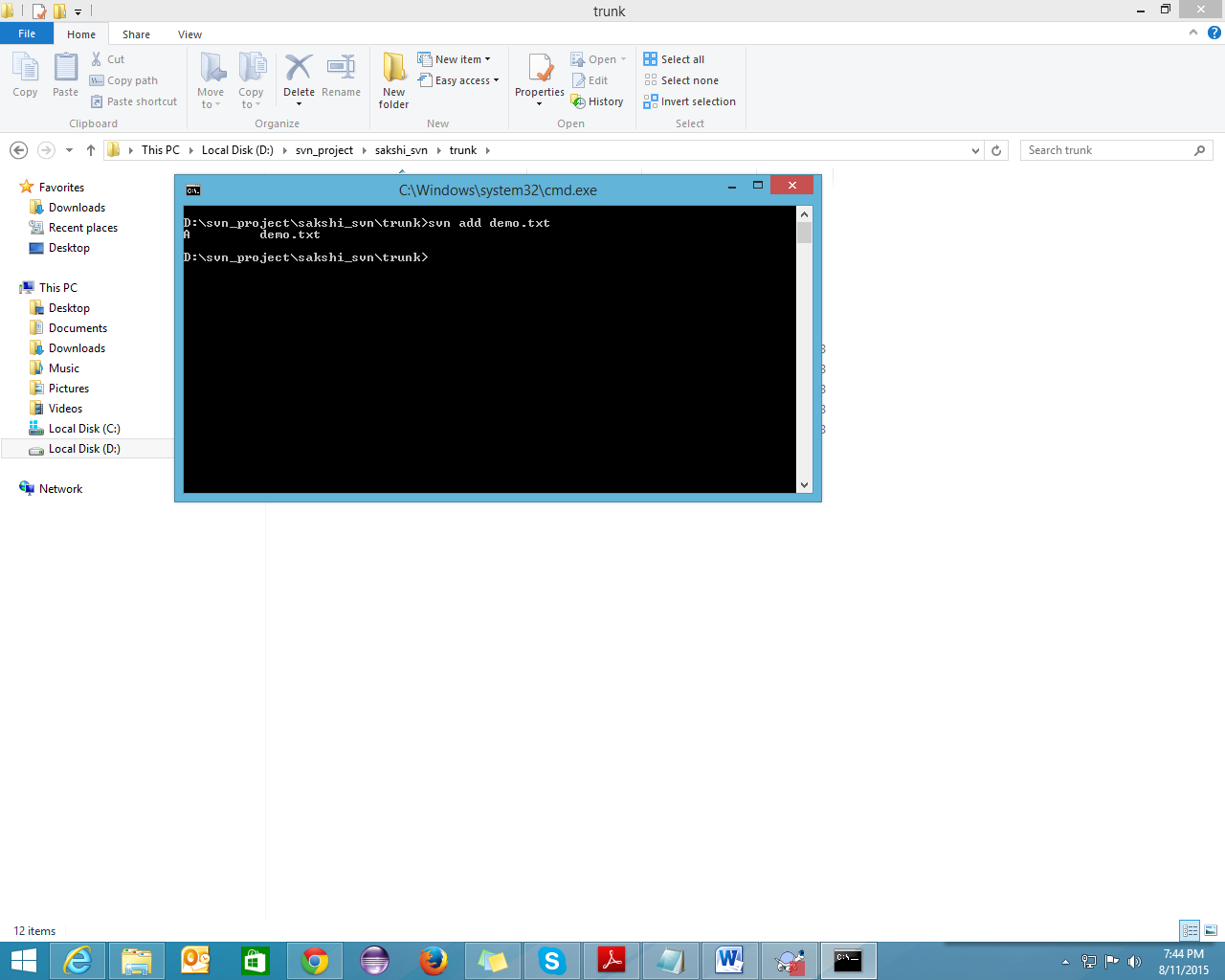


Similarly you can import your project into trunk by command line or by using UI.

## SVN Add

Adding demo.txt file in repository

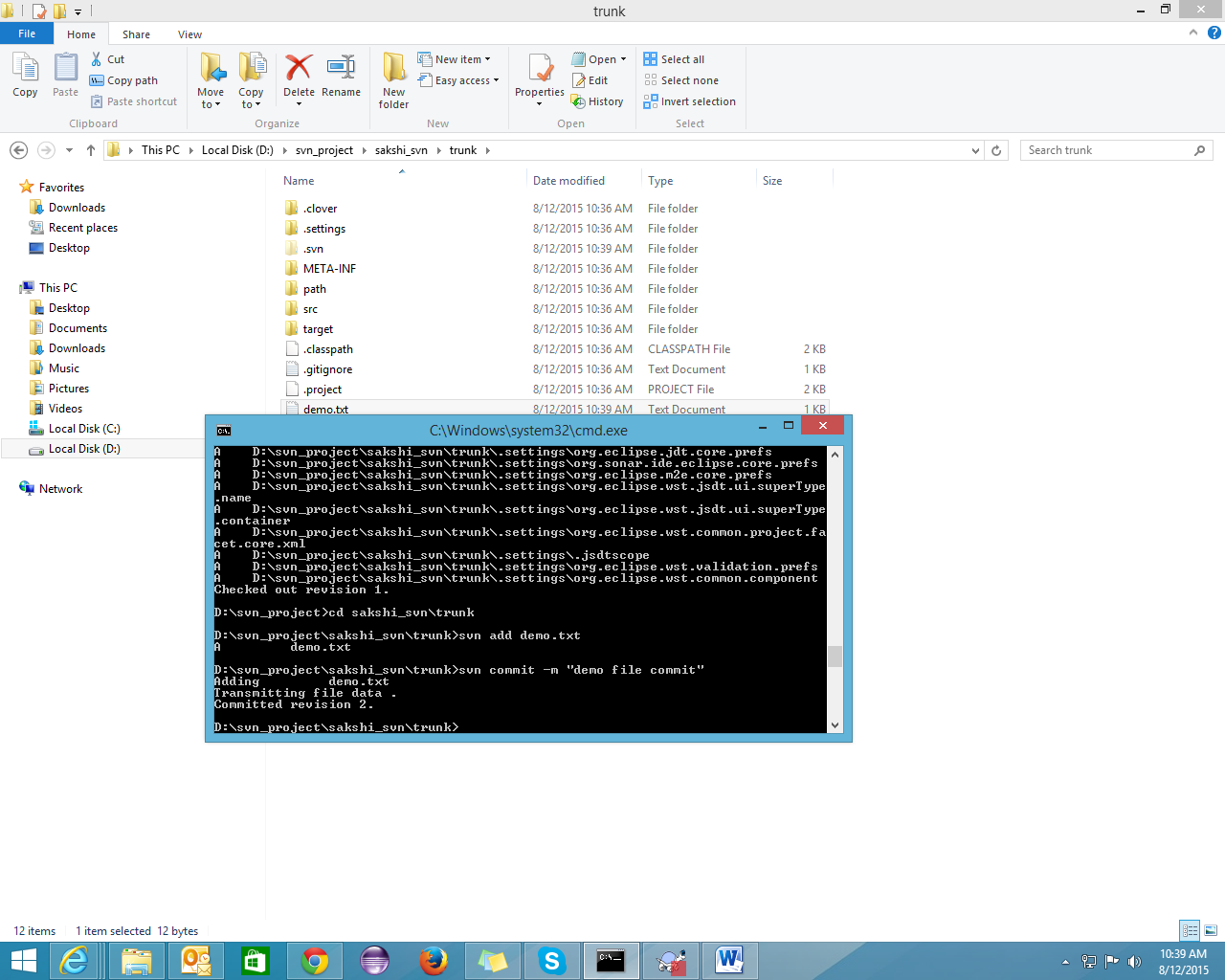
Svn add filename.txt



## SVN Commit

Commit the file

Svn commit –m “commit message”

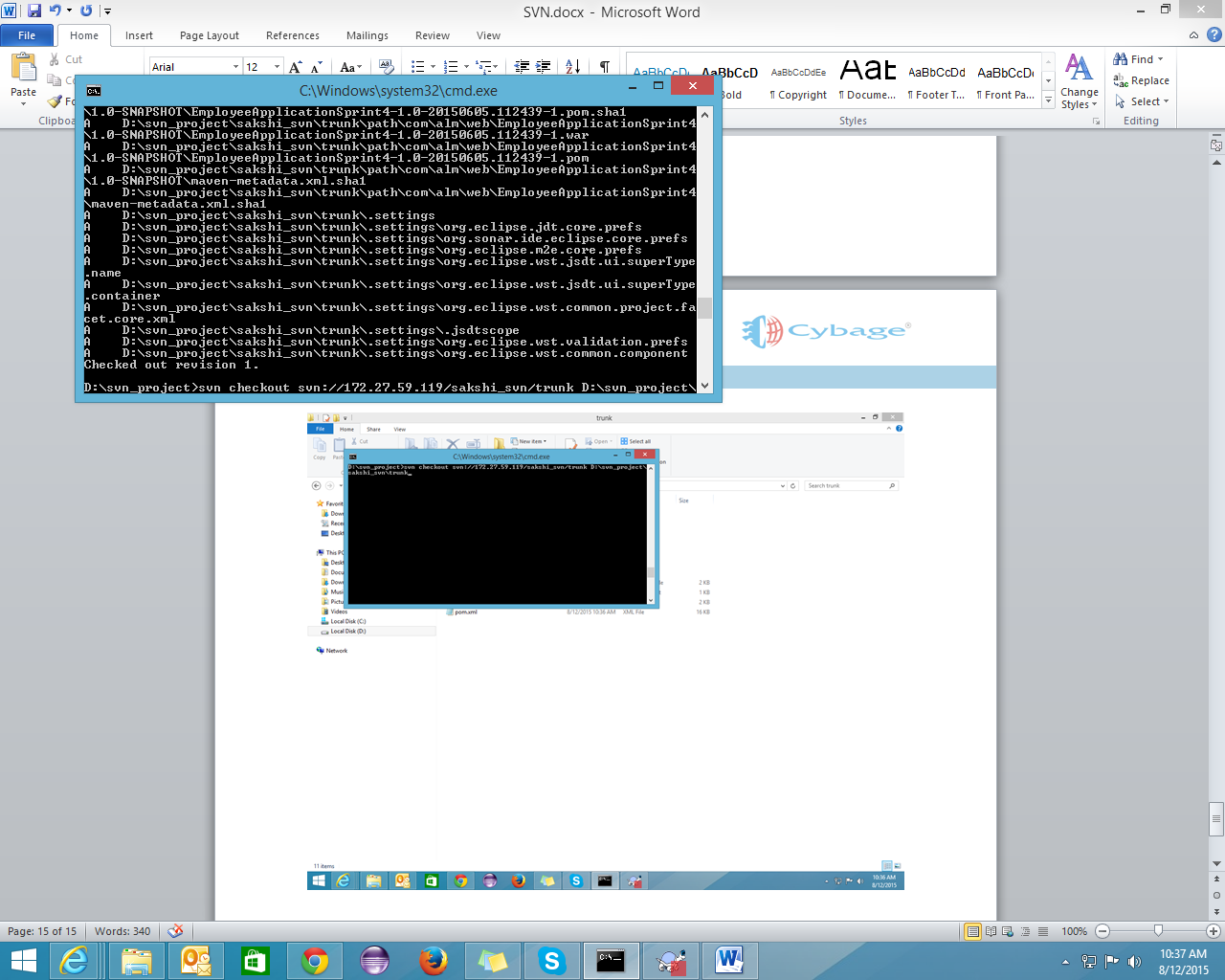


## SVN Checkout

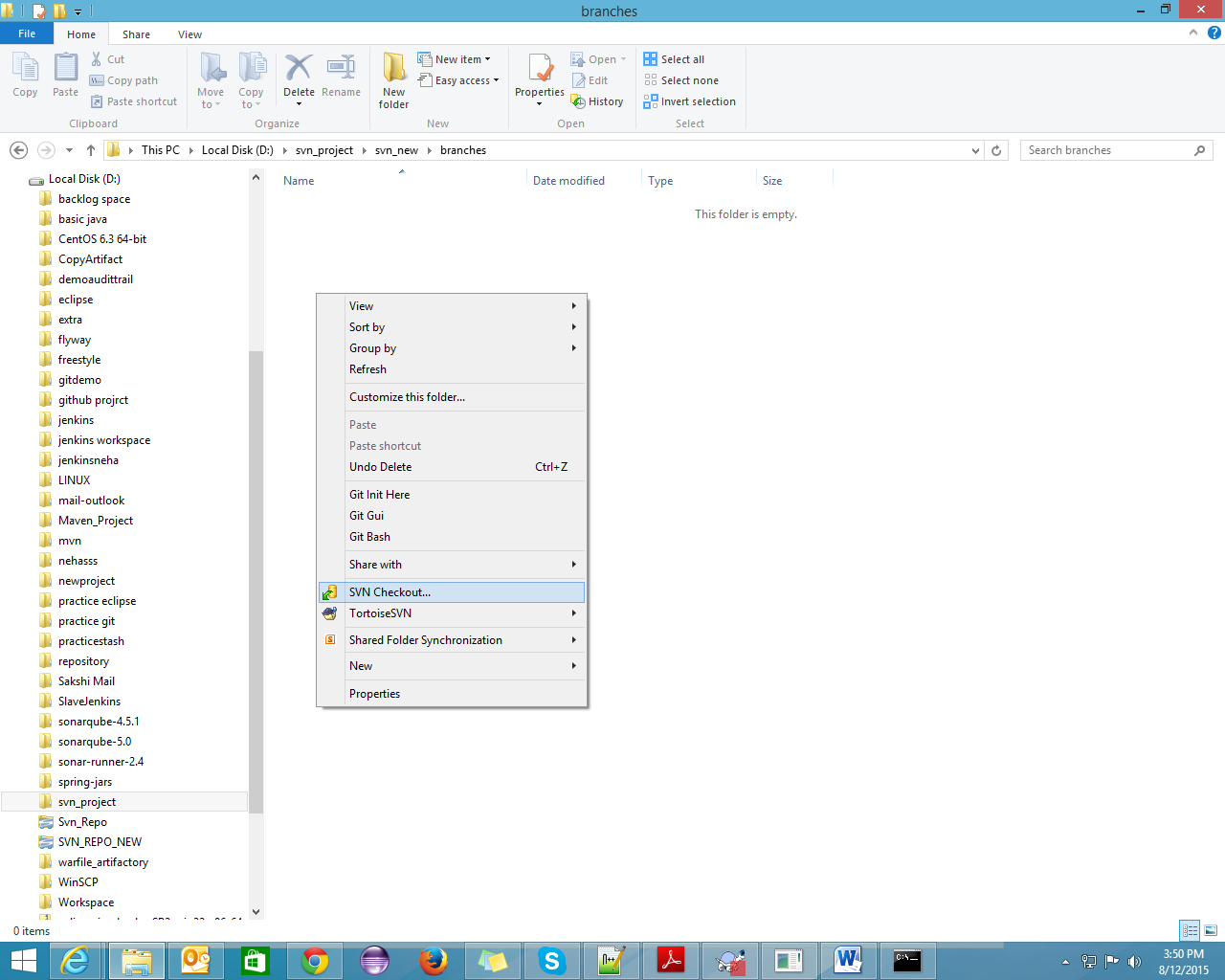
Checkout from trunk to local through command line

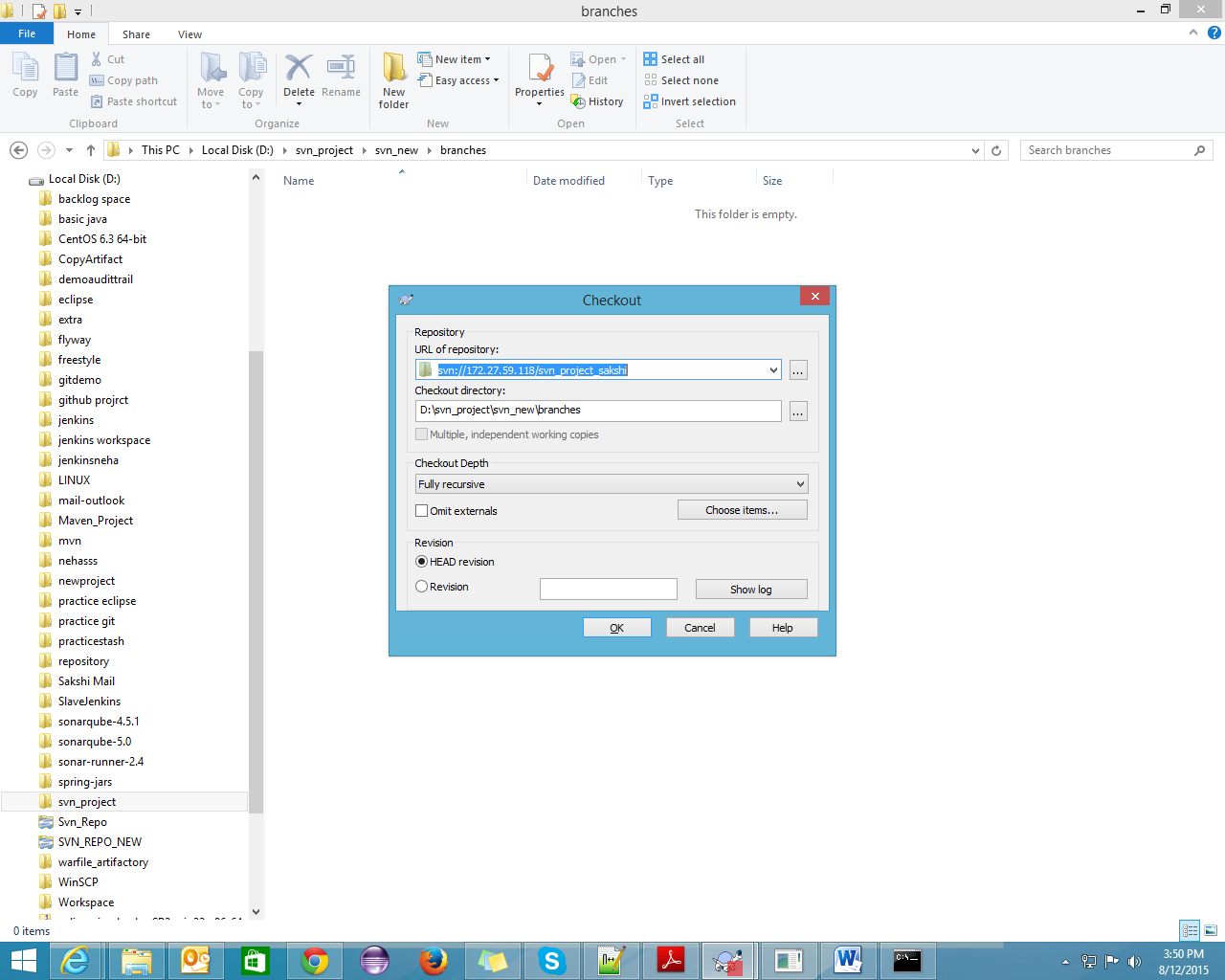
Svn checkout url (from repository) url (to local) 

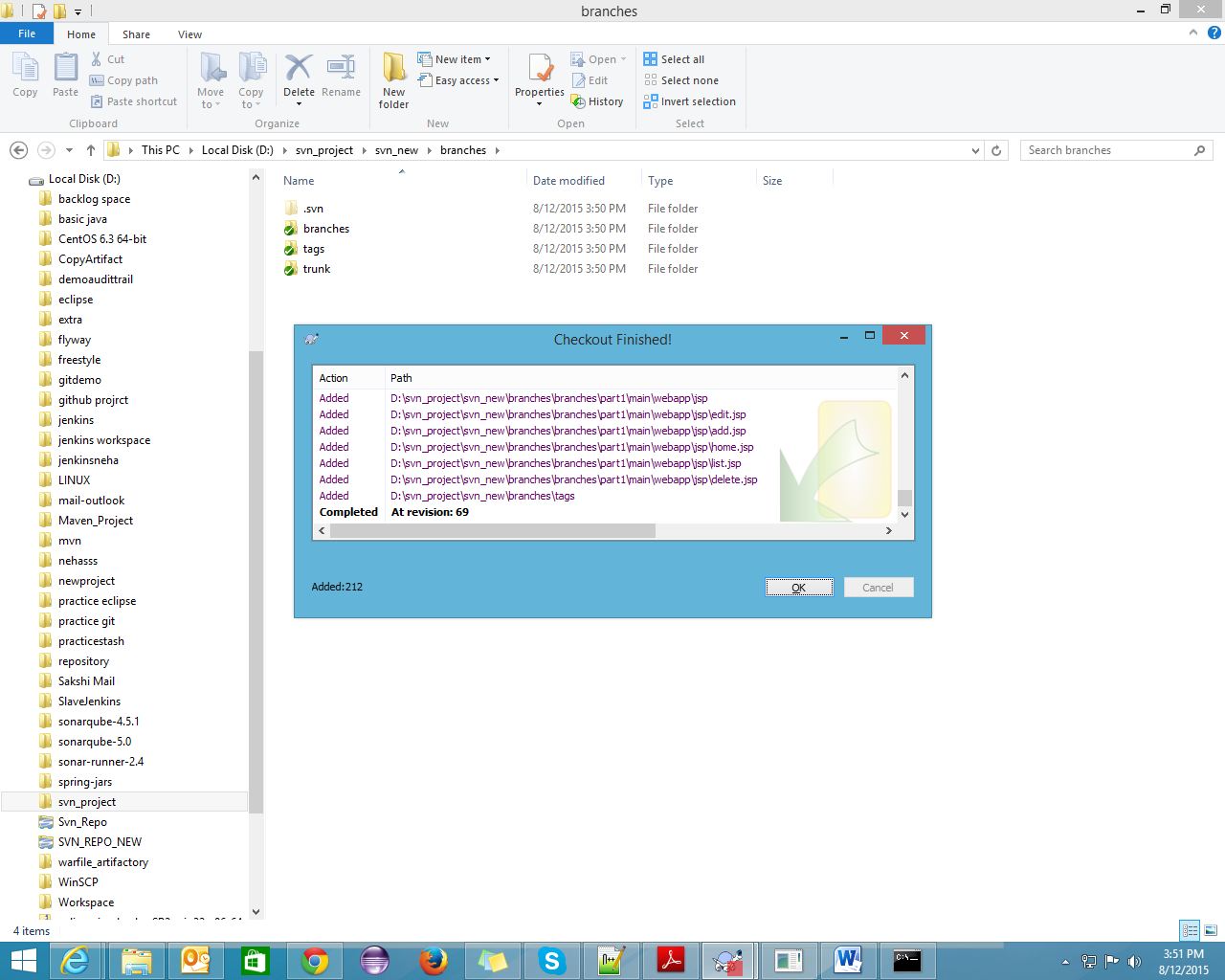
Checkout complete



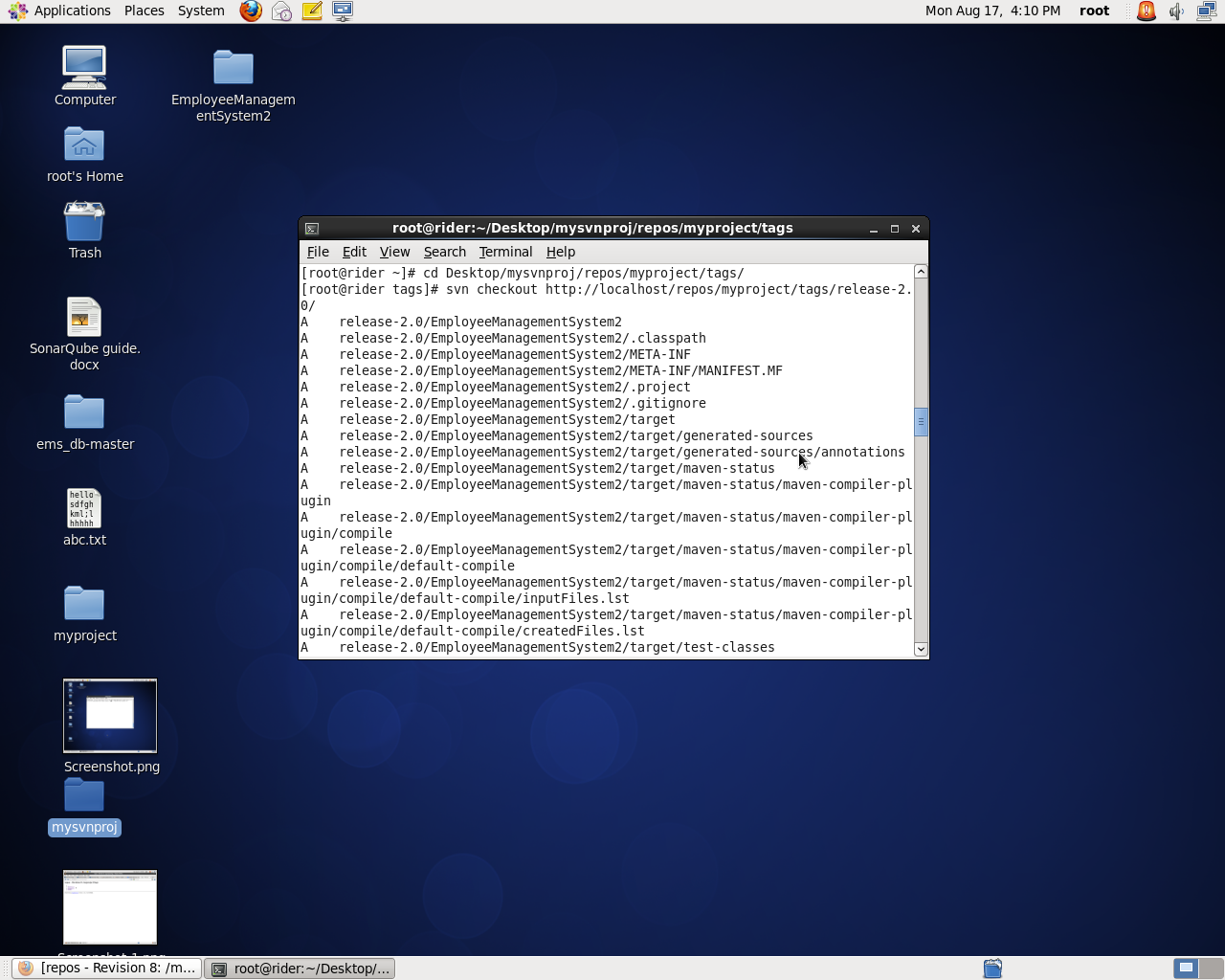
## Checkout through UI



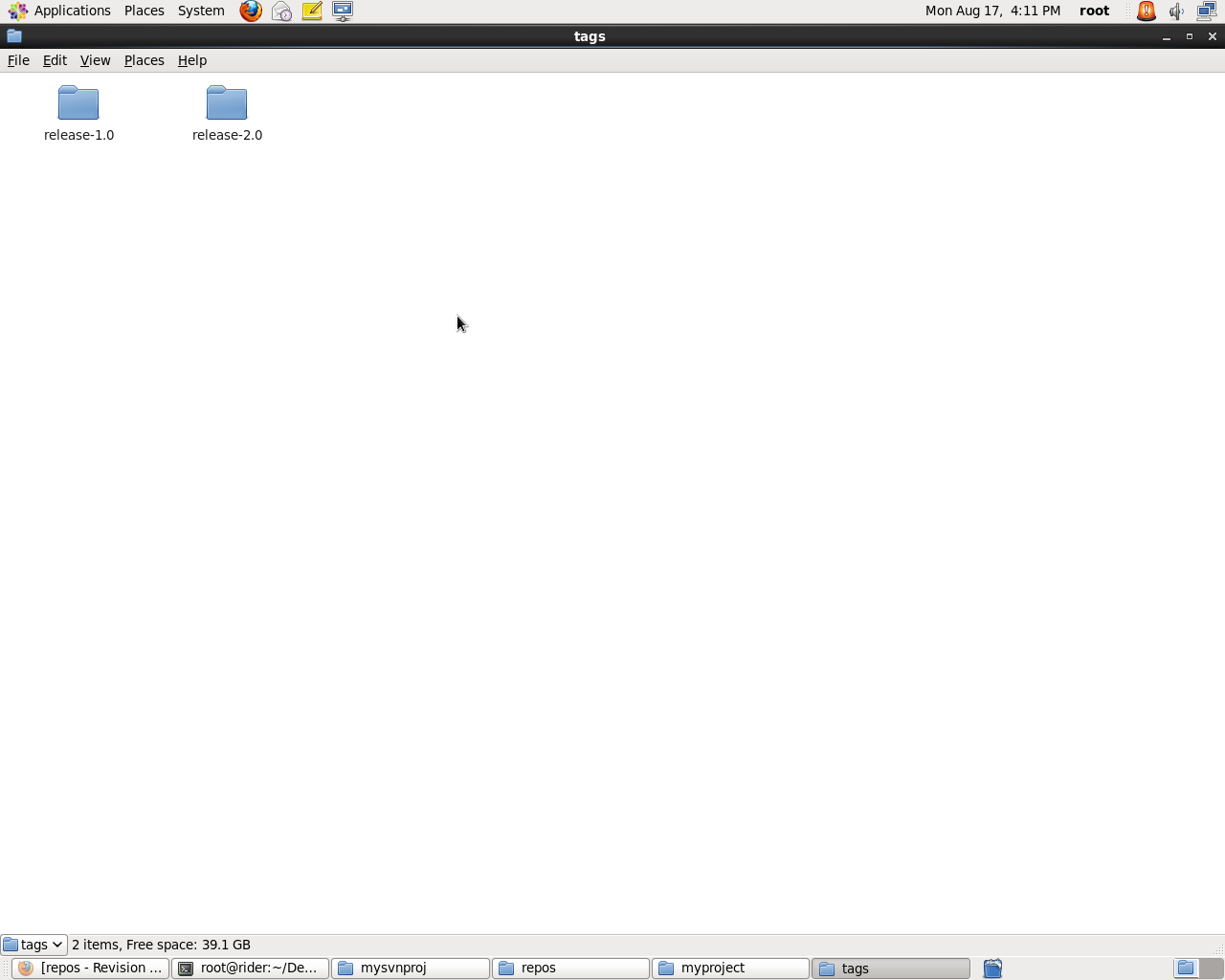




## How to checkout particular revision



That particular revision will appear in your local machine.

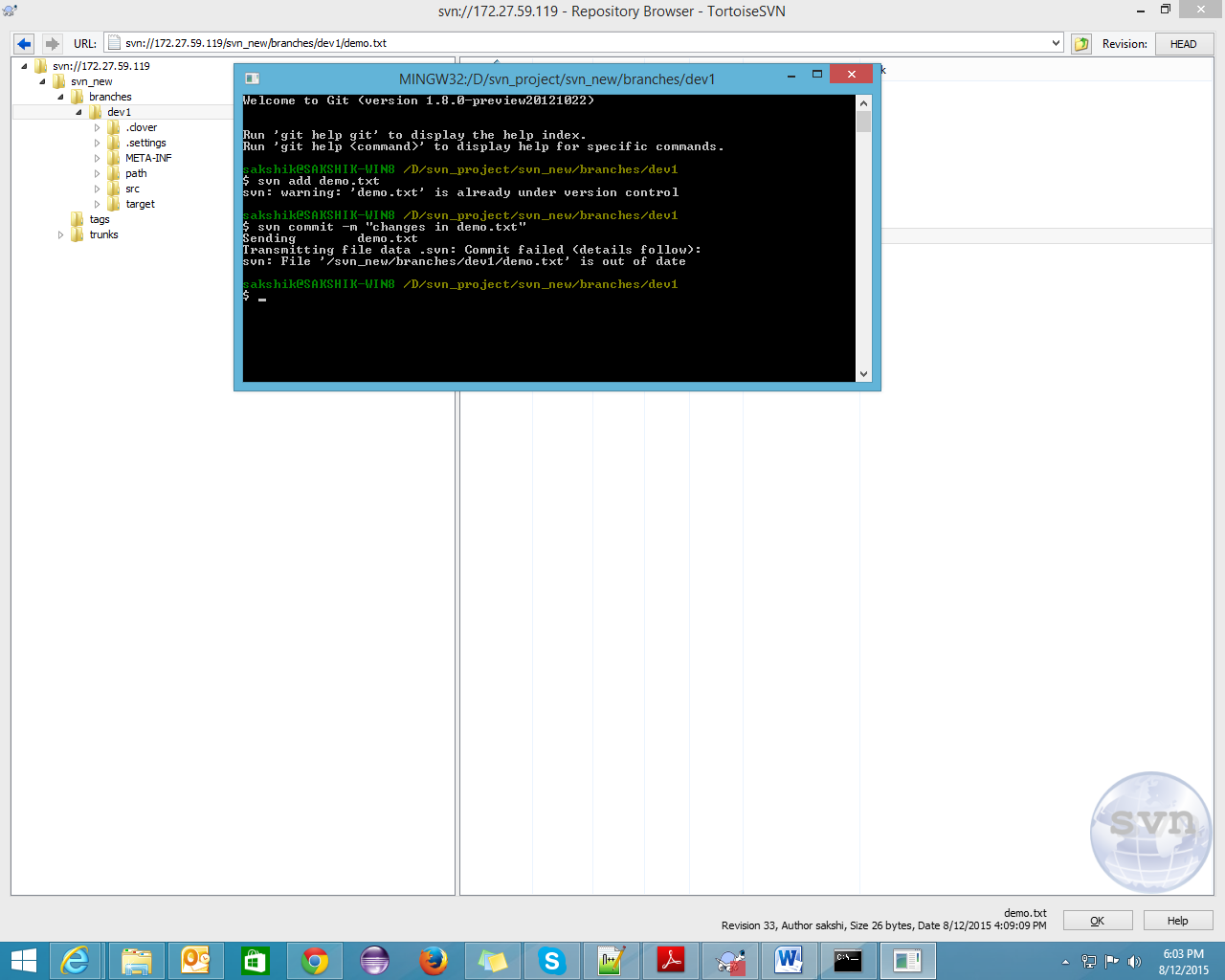


## Merge Conflict

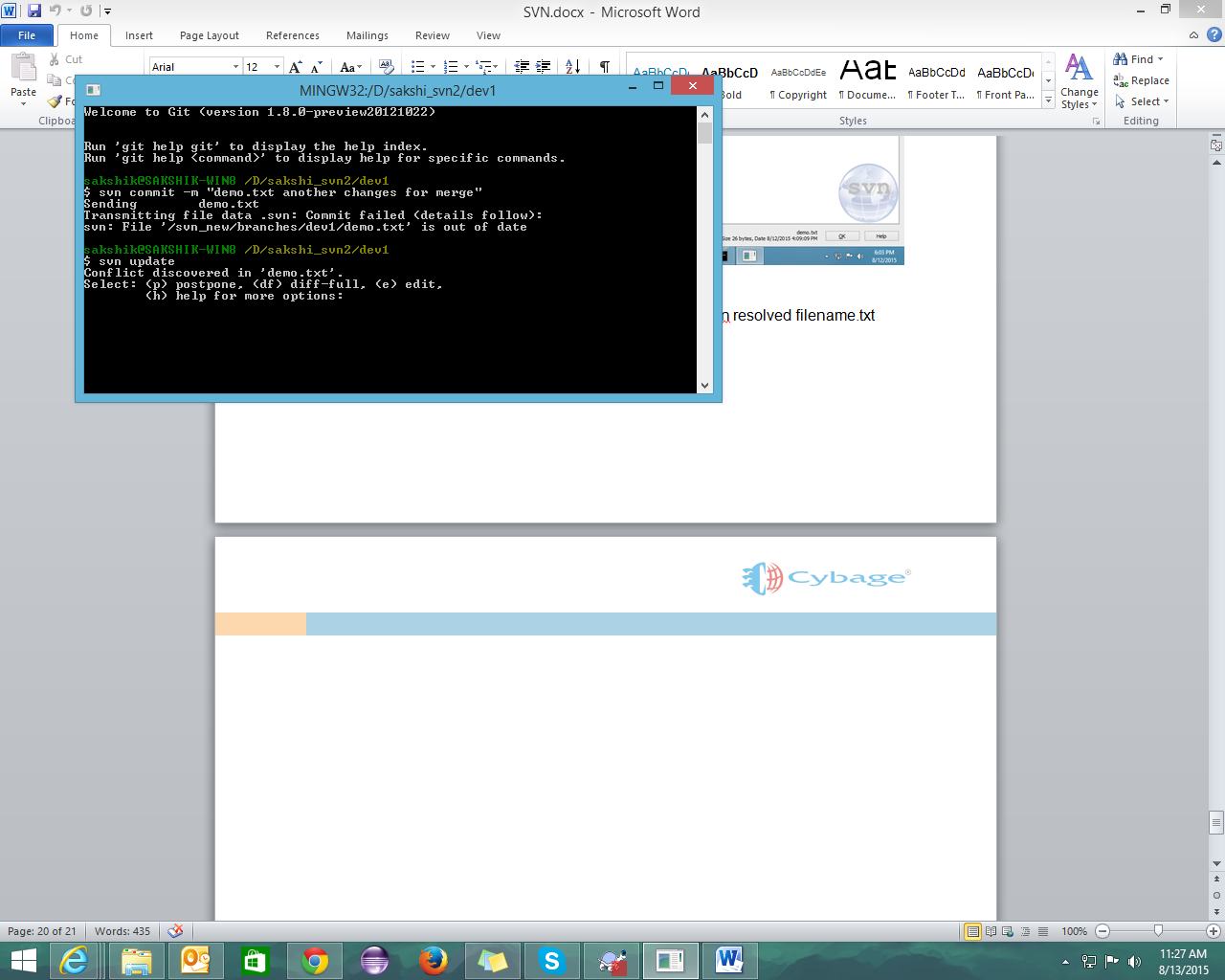
First of all, checkout the dev1 branch in 2 diff directories. Add any file in 1st directory 🡪 add 🡪 commit.

Do svn update in another directory. Do some changes, commit the file. It will show you the error that the file is out of date.

Do svn update then it will show you the merge conflict as shown below:



## Svn update

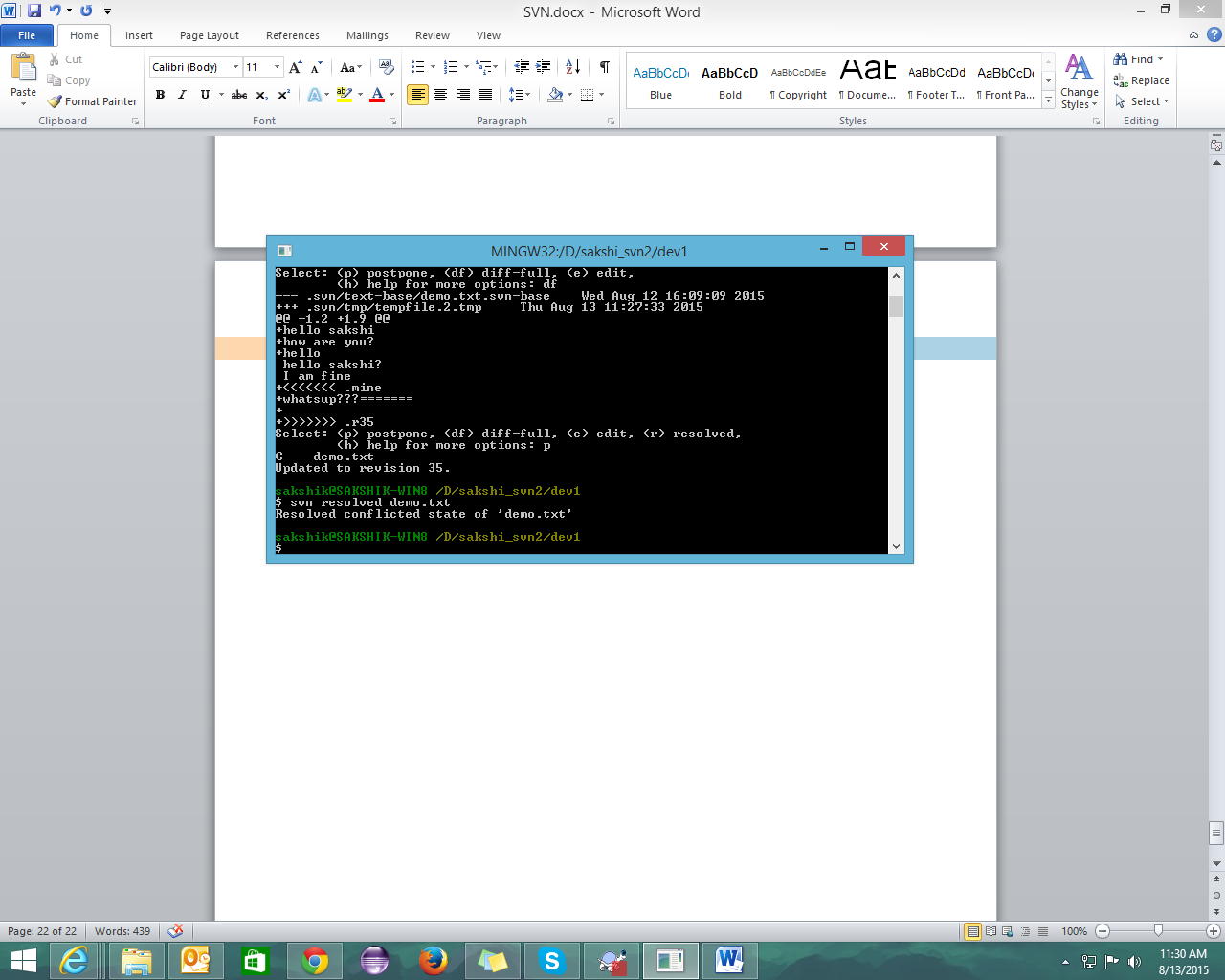


Do the changes in the file which you you want to keep. Do svn resolved filename.txt

## SVN Resolved

Resolved merge conflicts

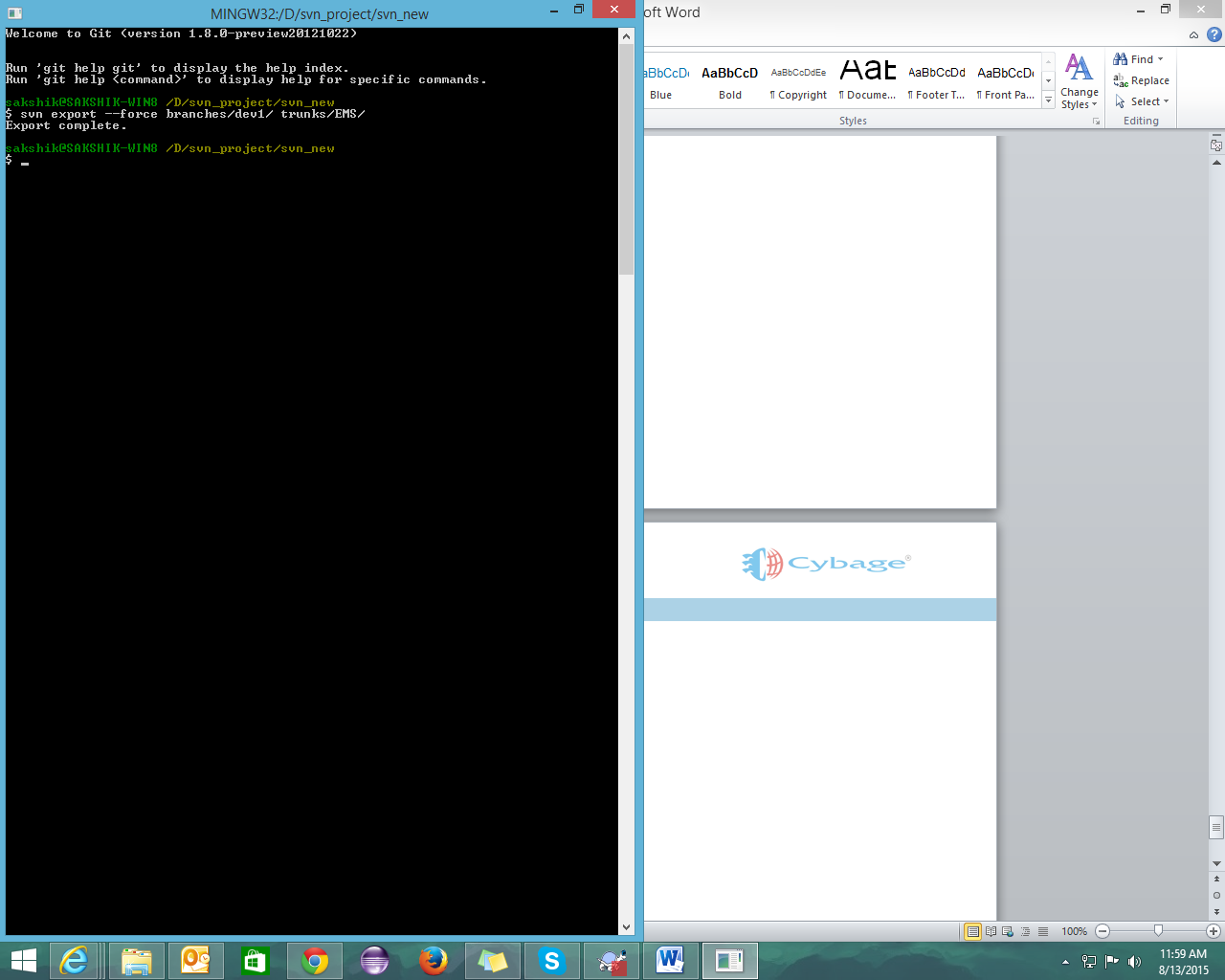
Svn resolved filename.txt



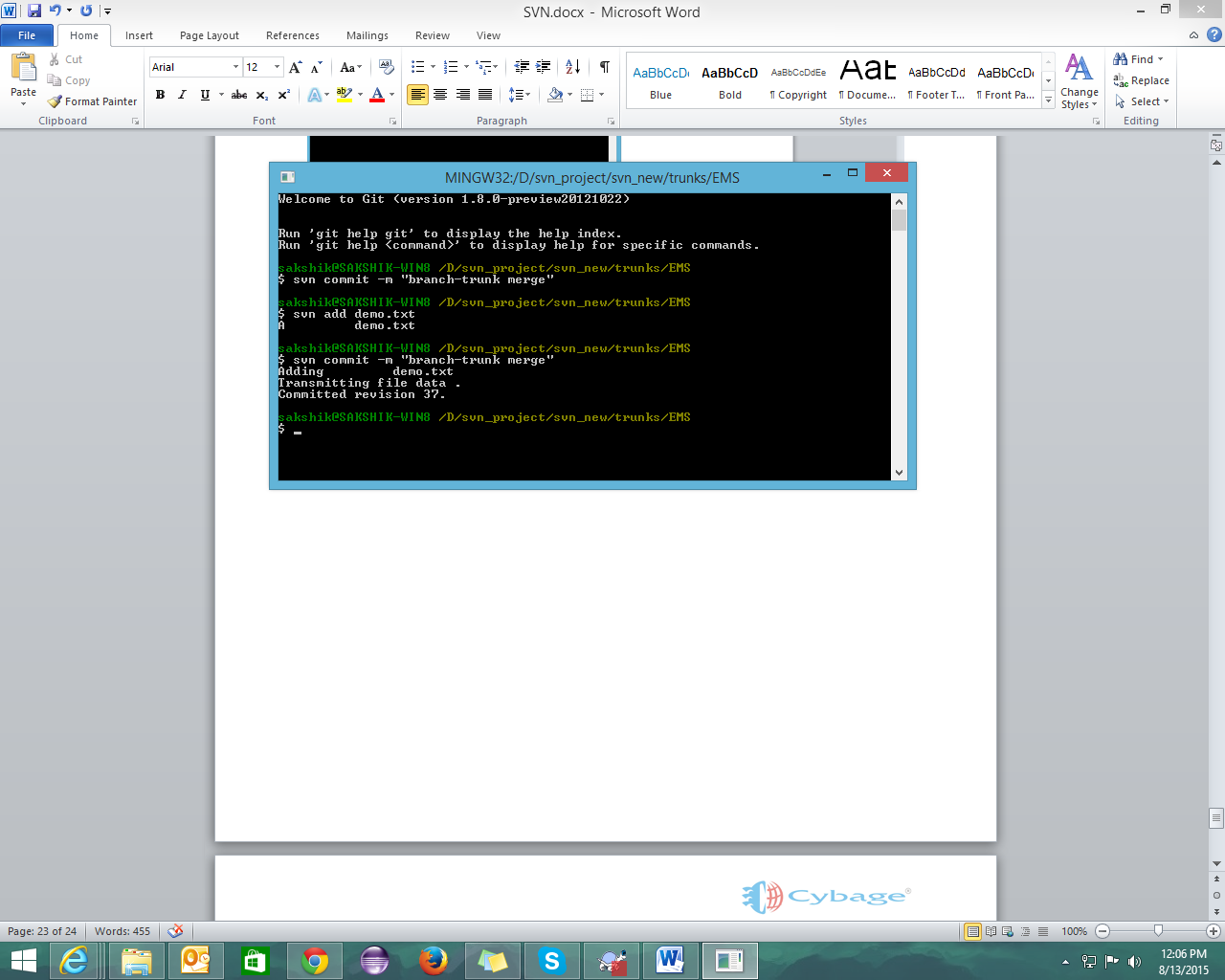
## SVN Export

To merge the changes from branch to trunk.

Svn export –force branches/branch\_name trunk/myProject



Now, you just need to add the file and commit the changes.



# SVN on Linux (CentOS)

Link for installation on CentOS:

<https://wiki.centos.org/HowTos/Subversion>

To install svn:

* Yum install mod\_dav\_svn subversion

To check svn is installed or not:

* Svn –version

Do the changes you need and save the file

* Vim etc/httpd/conf/httpd.conf :

To start the service httpd

* Service httpd start
* Chkconfig httpd on
* htpasswd –cm (create modify) /etc/svn-auth-conf sakshik (to set the password)
* cd /var/www
* mkdir svn
* cd svn
* svn admin create repos
* cd repos
* chown –R apache.apache repos
* Create directory
* mkdir myproject
* create folders branches.trunks,tags in that directory
* mkdir branches tags trunks
* svn import myproject/ [file:///var/www/svn/repos/mysvnproj](http://tfssp/var/www/svn/repos/mysvnproj) -m “creating remote repository”
* create a new directory at desktop where you want to checkout
* svn checkout <http://localhost/repos>
* copy your project to trunk folder

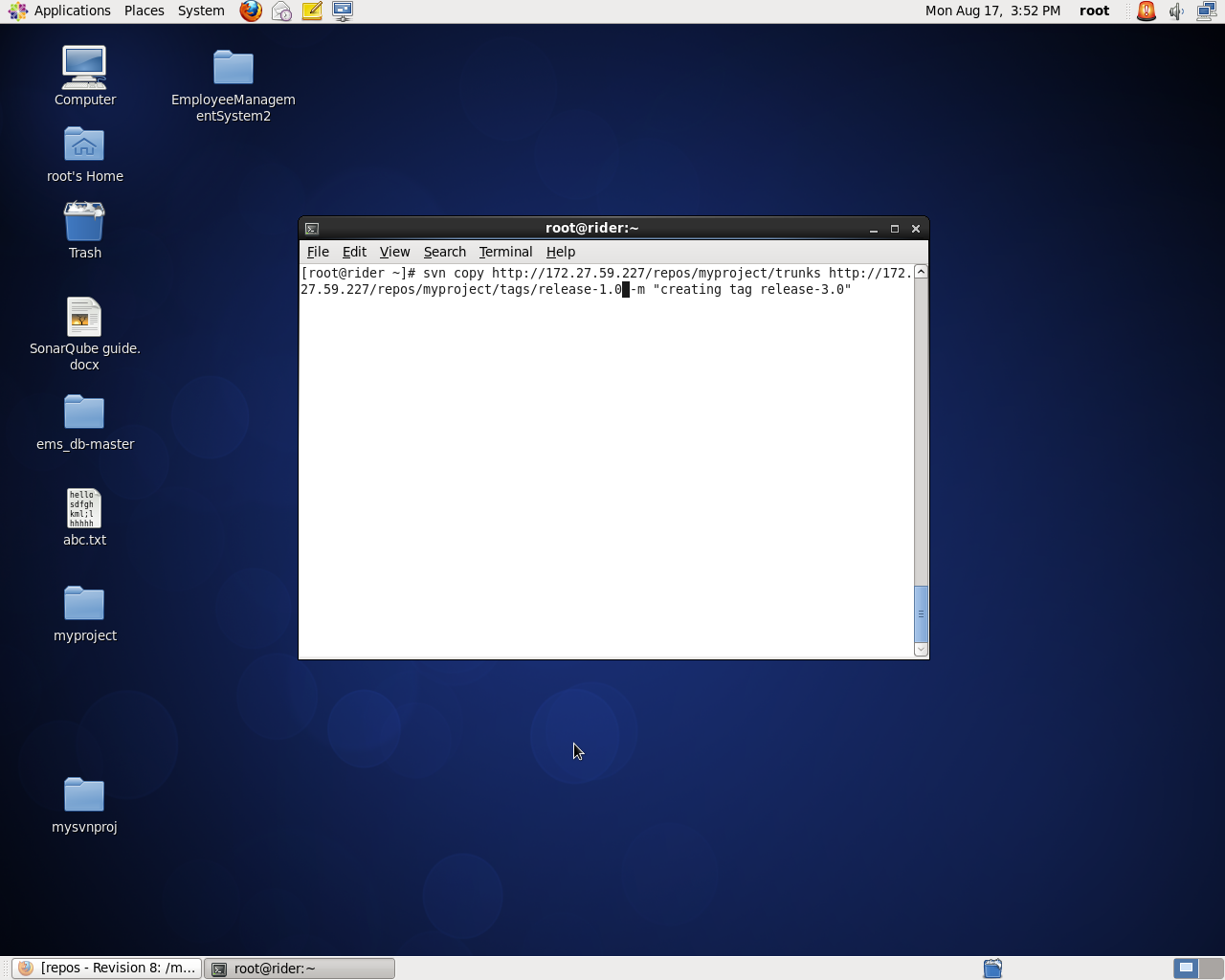
cp –rf EMS/ mysvnproj/repos/myproject/trunks

* creating tag:

svn copy <http://172.27.59.227/repos/myproject/trunks/> <http://172.27.59.227/repos/myproject/tags/release1.0/> -m “creating tag release 1.0”

Creating tags:

Command line



Repository

