Access to subversion (SVN) server

You can use port forwarding to establish a connection between the subversion server and SuperMUC, i.e., you may use one of the following procedures.

- You will be prompted for your SuperMUC password (or your ssh passphrase). If you are unlucky the port selected by you (e.g. 10022) is already used by someone else - in this case, you will see an error message printed out in advance of the motd; you then need to change your port to a different value.
- You might need to delete the localhost entry from ~/.ssh/known_hosts if ssh complains about the host key.

Using SVN with an https svn server

To establish the port forwarding for the SSL/TLS port issue the following command to connect from your workstation you normally use to SSH to the SuperMUC:

```
ssh -l <LoginName> -R <arbitraryPortNumber>:<svnServer>:443 skx.supermuc.lrz.de
Example:
ssh -l hk00xyz -R 10443:pmviewer.svn.sourceforge.net:443 skx.supermuc.lrz.de
```

After successful login to SuperMUC you may then access your repository via ("module load subversion"):

```
svn <svnCommand> https://<remoteLoginName>@localhost:<ForwardedPortNumber>/<svnDirectoryPath>
Example:
svn list https://mySVNUser@localhost:10443/svnroot/pmviewer
svn co https://mySVNUser@localhost:10443/svnroot/pmviewer pmviewer
```

Using SVN+SSH repository access

To establish the port forwarding for the SSH port issue the following command to connect from your workstation you normally use to SSH to the system SuperMUC:

```
ssh -l <your_userID_on_SuperMUC-NG> -R <arbitraryPortNumber>:<machine-withSVNrepo.>:22 skx.supermuc.lrz.de
Example:
ssh -l hk00xyz -R 10022:mySVNmachine.myhost.de:22 skx.supermuc.lrz.de
Repository:
mySVNmachine.myhost.de:/my/svn/repo
```

After successful login to supermuc, you have to set up a new protocol in your ~/.subversion/config file. Therefore you enter the following last line to the tunnel section in the config file:

```
[tunnels]
### Configure svn protocol tunnel schemes here. By default, only
### the 'ssh' scheme is defined. You can define other schemes to
### be used with 'svn+scheme://hostname/path' URLs. A scheme
### ...
myssh = ssh -p 10022
```

Now you may use the `svn+ssh` command as usual, with the exception that the newly defined `myssh` protocol is used instead of the standard `ssh` protocol:

```
svn <svnCommand> svn+myssh://<remoteLoginName>@localhost/<svnDirectoryPath>
Example:
svn list svn+myssh://mySVNUser@localhost/my/svn/repo
svn co svn+myssh://mySVNUser@localhost/my/svn/repo
```

GIT

No module load needed as system git is available.
As mentioned above, the SuperMUC firewall permits only incoming SSH-connections, following workarounds are possible to manage the git workflow.

### SSHFS

The simplest way is to mount a remote (SuperMUC-NG) filesystem locally.

```
local> mkdir mnt
local> sshfs -o follow_symlinks <userID_on_SuperMUC-NG>@skx.supermuc.lrz.de: mnt
local> cd mnt
local> git clone ... # or svn or hg or ...
...                 # do your work whatever it be
...                 # like git push or git pull
local> cd && fusermount -u mnt # on MAC-OS "cd && umount mnt"
```

### Via git-protocol (port 9418)

Git-protocol does not allow push over this protocol. Generally, if you are only interested in read-only git repositories then via git-protocol is the fastest way to manage Git the workflow.

```
local> ssh -R 12345:github.com:9418 <userID_on_SuperMUC-NG>@skx.supermuc.lrz.de
skx> git clone git://localhost:12345/</somepath>.git
```

**Important note:** If you get the warning ("remote port forwarding failed for listen port 12345") then port 12345 is already in use by someone else. Replace 12345 with, e.g. 13345 or some other number >10,000.

### Via HTTPS protocol (port 443)

The most popular way to use Git now is via HTTP or HTTPS protocol because of its simplicity and authentication features.

But, Git over HTTPS is more tricky to set up on some servers like Github and Bitbucket.

For GitLab servers please use the following workaround,

```
local> ssh -R 12345:gitlab.lrz.de:443 <userID_on_SuperMUC-NG>@skx.supermuc.lrz.de
skx> git clone -c http.sslVerify=false https://<userID_4_GitLab>@localhost:12345/</somepath>.git
```

**Important note:** HTTPS (port 443), requires "git clone" option "-c http.sslVerify=false" to circumvent the SSL certificate issue of HTTPS (as the requested DNS is now "localhost").

### Mercurial

```
local> ssh -l <your_userID_on_SuperMUC-NG> -R 12345:www.mercurial-scm.org:443 skx.supermuc.lrz.de
skx> module load mercurial
skx> hg clone --insecure https://localhost:12345/repo/hg/ mercurial-repo
```

"--insecure" is necessary to circumvent the SSL certificate issue of HTTPS (as the requested DNS is now "localhost")