Getting access to the Yade school repository

We will use the same internet tools used for developping Yade to share files and data during the numerical sessions, namely *launchpad* hosting and *bazaar* versioning system.

You can browse the content of the public repository online in Yade page at launchpad : https://code.launchpad.net/yade

Normaly, you would have to create an account for yourself on launchpad if you want to commit to the repository (this step can be skipped however, as explained below) and upload your public key, which will authenticate you (Launchpad doesn't use password authentication for sftp). Then you need to become member of *yadeSchool-team*, which is the repository owner.

Launchpad login

Open a web browser from inside the virtual machine and go to launchpad website.

1) You can create an account here, it needs only a functional email adress. Then, you join the yade-school team (https://launchpad.net/~yade-school).

2) Or you can use a simpler method :

Login with an account called *virtualStudent*, created specificaly for the school :

Login : ozsecretary@gmail.com

Password : Yade2011

Tell bzr what launchpad login you have

```
bzr launchpad-login your-id
```

or with the alternative method :

```
bzr launchpad-login virtualStudent
```

After this step, you are recognized as user *virtualStudent*, a member of the yadeSchool team (if you created your own account join the yadeSchool . You still have to identify your computer (one user can use different computers, this virtualStudent will apparently use 20 machines at the same time!).

Creating your SSH key

You need to upload an ssh key on launchpad so that your computer will get read/write access to the repository (else you could only download withtout write access).

1. type: **ssh-keygen -t rsa**
2. When prompted, press Enter to accept the default file name for your key.
3. Enter and then confirm a password to protect your SSH key (you may leave the password blank; in that case, no password will be asked when you work with launchpad). Your key pair is stored in ~/.ssh/ as id_rsa.pub (public key) and id_rsa (private key) now.

Registering the key with Launchpad

The public key must be uploaded to Launchpad. Login as yourself (if you created an account) or

1. Open your public key ~/.ssh/id_rsa.pub in a text editor (eg type: `kate ~/.ssh/id_rsa.pub`)

and copy its contents to your clipboard.

2. Edit virtualStudent profile, or go directly to https://launchpad.net/~your-id/+editsshkeys your SSH keys page (simply paste /+editsshkeys at the end of the url)
3. Paste your public key into the text box and then click the Import public key button to continue.

You are done. You can now downoad a version of the shared repository with the command below. If the operation is successful, Bazaar will give you no message.

```
bzr checkout lp:~yade-school/yade/yade-school
```

Now, you have a *yade-school* directory in the path you were before typing the command. There should be instructions there for the next exercises. Proceed with them before learning more on bazaar.

Working with bazaar – a short introduction

Updating the local version

You get changes commited by others using the update command :

```
bzr update
```

This is mandatory before commiting. If you don't, you will get a message telling that your version is too old for a commit.

Checking local changes and commiting

It is often usefull to check what you have change in the repository. Assuming we added results of our simulations in the text file *TriaxialTestsResults*, the *stat* command will return this:

```
yade@ubuntu:~/yade-school$ bzr stat
modified:
 TriaxialTestsResults
yade@ubuntu:~/yade-school$ bzr commit TriaxialTestsResults
```

The last command will prompt a a terminal editor. Type a comment in there (e.g. « Adding our first results »). Then press ctrl+o (write the log) and ctrl+x (effectively commit and exit).

Adding directory and files

For adding more results of your simulations, it is better to create an independent directory. Create a directory for your results in yade-school/ (from the file browser or in the terminal as below), declare this directory versioned to bzr, and commit it. The process is the same for adding files.

```
yade@ubuntu:~/yade-school$ mkdir myUserDir ## replace « myUserDir »
yade@ubuntu:~/yade-school$ bzr stat
unknown:
  myUserDir/
yade@ubuntu:~/yade-school$ bzr add myUserDir/
adding myUserDir
yade@ubuntu:~/yade-school$ bzr stat
added:
  myUserDir/
yade@ubuntu:~/yade-school$ bzr commit myUserDir/
Committing to: bzr+ssh://bazaar.launchpad.net/~yade-school/yade/yade-
school/
added myUserDir
Committed revision 3.
```

Numerical Session 1 :

Introduction to Yade and Triaxial Simulations

Basics

1) Have a look at « hands-on » section of Yade's tutorial. It will give some basic informations on linux and python if needed.

2) Load the script triax-base-OZ.py in Yade, and run some iterations (at least 1000). Before examining the content of the script, we will use the simulation it creates to test different methods for moving around with Yade.
Check the « data-mining » section of the tutorial and test each commands in the current simulation. Inspect the graphical user interface.
Press the « inspect » button and visit engines, bodies and interactions.

Triaxial simulation

We will analyse the different part of the script triax-base-OZ.py. This script has many parameters. Make sure you understand the meaning of each of them, else ask the professors.

In this script, all parameters have default values so that the script gives correct results initialy. Modifying some values may break something, and it can be quite interesting to try and understand why. Indeed, the average user needs to define simulations correctly without prior knowledge of correct settings, and learning to find what's going wrong is part of learning DEM simulation.

1) Particles sizes and positions

1.1) There are different ways to generate particle sizes distributions using the makeCloud function. Execute the example script psd.py (yade-daily /usr/share/doc/yade-daily/scipts/test/psd.py), and check in psd.py how the different distributions are obtained. Choose one distribution for your simulations.

1.2) Plot the contact orientation distribution with the function plot *plotDirections()*. You will find maxima along x,y,z axis. Can you explain this result ?

1.3) Check the function's documentation to try and find a way to eliminate this artifact.

2) Confinment

Uncomment the next section of the script (select the code block and press ctrl+shift+D). With this section, the script will run iterations until the unbalanced force reaches a defined value.

During the simulation, different variables are written by the engine TriaxialStressRecorder to text file WallStress_myName. Examine the columns names in this file, and plot the evolution of the unbalanced force during iterations :

```
gnuplot
gnuplot> plot './WallStresses_triax_base_' using 1:3
gnuplot> plot './WallStresses_triax_base_' using 1:3 w lines
gnuplot> replot './WallStresses_triax_base_' using 1:2 w lines
```

3) Deviatoric loading

The next section of the script concerns deviatoric loading. Uncomment it and examine the code.

3.1) Add some code to run the simulation up to eps22=20%.

3.2) Run the script and analyse the results in order to define macro-mechanical parameters listed in the triaxialTestResults file of yade-school repository.

3.3) Add a line in this file with your results and an identifier name (so that we can latter retrieve the script that was used for each result). Create a directory in the yade-school folder with the corresponding script and any other usefull data.

3.4) Commit the results to the public repository (see launchpad/bazaar instructions).

4) Batch execution

Make a parametric study using yade-daily-batch (see https://yade-dem.org/doc/tutorial-geo.html#parametric-studies). Different sorts of comparisons can be meaningfull recommended :

4.1) Different micromechanical parameters leading to different packings (e.g. compFricDegree)

4.2) Different numerical parameters on the same packing (which should ideally give the same results in the quasistatic limit, e.g. Damping, loading rate...)

4.3) Different runs of the same simulation, when only the initial random positioning differs (usefull to evaluate dispersion as a function of the number of particles).