A) PREPARE FTI RUNTIME AND EXAMPLES:

Preparation:

1. Create tutorial directory and change into it:
   mkdir FTI_TUTORIAL
   cd FTI_TUTORIAL
2. Download package:
   git clone -b tutorial https://github.com/leobago/fti TUTORIAL
3. Change into base directory
   cd TUTORIAL

Configure & Install:

1) Create build directory and change into it:
   mkdir build/
   cd build/
2) Build FTI:
   cmake -DCMAKE_INSTALL_PREFIX:PATH=../../fti-release/ ..
   make
   make install

Excutables and fti library files:
The library is installed at the the path you specified after the DCMAKE_INSTALL_PREFIX:PATH. Any tutorial file will be found under *"./tutorial"*.

B) DEMONSTRATION OF FTI:

To demonstrate the various safety levels of FTI, we will execute an example which uses the api function '*FTI_Snapshot()*'. Run the example in each case for at least one minute and interrupt the execution after that time by pressing '*ctrl+c*'.

L1 – Local checkpoint on the nodes:

Change into folder ./tutorial/L1 and run the execution with 'make hdl1'. While the program is running, you may follow the events by observing the contents in the 'local' folder. In order to do that you can use the commands:

watch -n 1 'find local'
watch -n 1 'du -kh local'
or
cd local; watch -n 1 'ls -lR'

(It may be illuminating to open the files in the 'meta' folder, using a text editor. What kind of information do you think is kept in these files?)

After interrupting the execution, run again '*make hdl1*'. The execution will (hopefully) resume from where the checkpoint was taken.

After the successful restart, interrupt the execution and delete one of the checkpoint files. The files are stored as (you can also simply delete the whole node directory): local/<NODE>/<EXEC-ID>/<LEVEL>/ckpt<ID>-Rank[Pcof/Rsed]<RANK>.fti. You will notice, that in that case the program won't be able to resume the execution.

L2 – local checkpoint on the nodes + copy to the neighbor node:

Change into folder ./tutorial/L2 and run the execution with ‘make hdl2’. While the program is running, you may follow the events by observing the contents in the ‘local’ folder.

After interrupting the execution, run again ‘make hdl2’. The execution will also in this case (hopefully) resume from where the checkpoint was taken.

After the successful restart, interrupt the execution and delete one of the checkpoint files.
You will notice that now the program (hopefully) will be able to resume the execution. Try to delete more then one file.

Questions: In order to keep the execution able to resume:

1. How many files you can delete?
2. Which files can you delete?

L3 – local checkpoint on the nodes + copy to the neighbor node + RS encoding:

Change into folder *./tutorial/L3* and run the execution with ‘make hdl3’. While the program is running, you may follow the events by observing the contents in the ‘local’ folder.

After interrupting the execution, run again ‘make hd3’. The execution will (surprisingly) also in this case resume from where the checkpoint was taken.

After the successful restart, interrupt the execution and delete one of the checkpoint files, the program will be able to resume.

Questions: In order to keep the execution able to resume:

3. How many files you can delete?
4. Which files can you delete?

L4 – flush of the checkpoints to the parallel file system:
Change into folder ./tutorial/L4 and run the execution with ‘make hd4’. While the program is running, you may follow the events by observing the contents in the ‘global’ folder.
After interrupting the execution, run again ‘make hd4’. The execution will resume from where the checkpoint was taken.

L4 - Differential Checkpoint:

Change into folder ./tutorial/DCP/ and run the execution with make hdDCP. While the progam is running you may follow the “blue” messages in the terminal. What is actually happening?

Delete all files under ./local, ./global/ ./meta/ and open file config.DCP.fti with your favorite text editor. Change the following parameters :

ckpt_io = 3 to ckpt_io = 1
failure = “x” to failure = 0

The first option changes the file format and the second option indicates that we will do a fresh run (not a recovery). Run the execution with make hdDCP, do you observe any difference in the timings of the checkpoints?

C) PRACTICE

1. In the ‘./tutorial/practice’ folder you will find the source code of the program we used to demonstrate the FTI features. In this case without FTI being implemented. Try to implement FTI. You can use either the ‘FTI_Snapshot’ or ‘FTI_Checkpoint’ function to cause FTI taking a checkpoint.

2. Change into the folder ‘./tutorial/experiment’ and play with the settings of the configuration file. To run the programm, type: ‘mpirun -n 8 hd.exe config.fti <GRIDSIZE>’. Perform executions with ‘Head=0’ and ‘Head=1’, do you notice any difference in the execution duration? (Note: You may take frequent L3 checkpointing and a gridsize of 256 or higher. In that case you will most likely see a difference). (Remark: <GRIDSIZE> denotes the dynamic memory of each mpi process in MB)