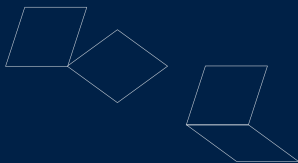


Preparation for: ML Potentials Hands on Exercise

Summer School on Machine Learning for Materials
Hard and Soft July 18 and 19 2022



Servus!

We are heading to a software-heavy exercise.

Please try to complete the instructions on the following pages by Monday, July 11th (eleventh). While we have a limited capacity to address personal installation questions, we can offer some support during the first week of the summer school to get everything running by July 18. Get in touch with Alexander Gorfer or Lukáš Kývala.

If you have any issues or questions, please write an email with the full descriptions (including screenshots) to:
mlpotsummerschool22@gmail.com

Introduction to the Software Environment

We use codes from a few repositories. We recommend running them in our provided controlled software environment (Docker). Docker allows us to deploy an application in a portable environment without installing a long list of prerequisites or having to worry about interferences with the already installed software.

Our workflow consists of first invoking a command in a Terminal for mac/linux or PowerShell for Windows on your host machine to run the pre-defined Docker container. The Docker container then has access to all files and applications inside the container, but not to any other files on the machine. Except for the selected folder, which is shared between the Docker container and your host machine. Anytime you want to run an application we need Terminal/PowerShell, but you can edit any file in the shared folder on your machine with your favorite editor.

If you want to build the codes yourself, you can also follow the commands provided in the `Dockerfile`, by basically installing each dependency after `RUN` and replacing `WORKDIR` with `cd`. You will benefit from parallel computing. Please make sure that you can run the tests. Godspeed!

Preliminary Steps (0) - Learn basics

In order to be able to conduct the exercises on July 18 and 19, it is necessary to know the basics of working with the command line and jupyter notebooks.

If you are new to working with the command line, we provide you with a short introduction at the end of this file.

For jupyter notebook, you can learn the basics on:

<https://www.youtube.com/watch?v=jZ952vChhul>

Preliminary Steps (1) - Download the Material

- ▶ Download github repository from <https://github.com/Kyvala/MLSummerSchoolVienna2022>
- ▶ Extract the folder and you find a Dockerfile.

Preliminary Steps (2) - Install Docker

macOS

1. Install Docker Desktop for Mac:
<https://docs.docker.com/desktop/mac/install/>

Linux

1. Install Docker Engine for your distribution:
<https://docs.docker.com/engine/>
2. Allow Docker to be run without sudo:(or use `sudo docker`)
`sudo groupadd docker`
`sudo usermod -aG docker $USER`
Reboot system

Windows

1. Follow the instructions provided on:
<https://docs.docker.com/desktop/windows/install/>

Troubleshooting

Linux

1. Did you get "**Docker Desktop stopped...**"? Please try installing the DockerEngine for Linux, the GUI application is a bit newish.

Windows

1. Did you get **WSL2 installation is incomplete**? Please make sure that you have installed the [linux-kernel-update](#).

Preliminary Steps (3) - Build and Run Docker

- ▶ Open the terminal of your choice (mac terminal, bash, PowerShell..) and change the directory to the repository we downloaded in Step (1):

```
cd $Path_to_Repository
```

- ▶ Build the Docker container:

```
docker build . --tag summer-school
```

- ▶ Create and start the container:

macOS, linux

```
docker run -v <fullpath>/test:/home/test -p 8888:8888 -it summer-school /bin/bash
```

Windows

```
docker run -v <fullpath>\test:/home/test -p 8888:8888 -it summer-school /bin/bash
```

where <fullpath> is the absolute path (C:\Users\...) to the download repository (probably your current working directory, you can use pwd command to get <fullpath>).

Preliminary Steps (4) - Editing Files

We should now be inside the Docker container:

```
PS C:\Users\CP\Downloads\MLSummerSchoolVienna2022-main> docker run -v C:\Users\CP\Downloads\MLSummerSchoolVienna2022-main\test:/home/test -p 8888:8888 -it summer-school /bin/bash
root@12cb68895162:/home#
```

Use the `ls` command to print the list of folders. You should see the `n2p2`-folder containing software, test, Day06_July18, and Day07_July19 folders. Change your folder to test:

```
cd test
```

Now outside the terminal, on your host machine, try creating/moving a random file into the test in the downloaded repository. Go back into the Terminal/PowerShell and type `ls` and it should be there! You can edit anything in this folder on your host machine. Or inside the container with the text editors:

```
nano filename    or
vim filename
```

Preliminary Steps (5) - Running Tests

Inside the test folder, run the n2p2 training binary:

```
/home/n2p2/bin/nnp-train
```

If your test is successful, you should get: TIMING Training loop finished... Then open a jupyter notebook through Docker using:

```
jupyter-notebook --ip 0.0.0.0 --no-browser --allow-root
```

A lot of text will be printed to the terminal. At the end, there will be a URL like `http://127.0.0.1:8888/?token=18e24...`

Copy the full address including the token into a browser and you should see a jupyter notebook page. Open

`jupyter_notebook_test.ipynb` and run cells with `shift+enter`. If you compiled codes by yourself please check that you can execute the notebook. If both tests work, you are ready for the exercises.

Preliminary Steps (6) - Stop Docker container

There are two ways how to stop Docker container.

- ▶ Open Docker desktop application. You should see all running Docker containers. You can stop Docker running by a square symbol. If you do not want to re-run the Docker run, you can also remove it by bin symbol.
- ▶ Open new window of Terminal/PowerShell and type:
`docker stop container_id`
The container ID is written after @ symbol:

```
PS C:\Users\CP\Downloads\MLSummerSchoolVienna2022-main> docker run -v C:\Users\CP\Downloads\MLSummerSchoolVienna2022-main\test:/home/test -p 8888:8888 -it summer-school /bin/bash
root@12cb68895162:/home#
```

For this run, we would stop Docker container by `docker stop c9d773620723`

An introduction to the command line

1 Basics

After starting the shell (the program behind the command line terminal), you can see an empty screen with only one line – the prompt:

```
ba01@silicon1 ~>
```

When you see the prompt, the shell is telling you that it awaits user-input.

At this point, you can input a command, which will then be executed. A command may look like that:

```
ba01@silicon1 ~> mycommand myargument
```

This would start the program `mycommand` with `myargument` as input. The argument is in many cases a filename, e.g.:



```
ba01@silicon1 ~> rm myfile
```



would remove the file `myfile`.

A very useful feature of the shell is **auto-completion**. While typing a command or filename, you can press  to complete the command or filename you are just typing.

```
ba01@silicon1 ~> pdf1
```

```
ba01@silicon1 ~> pdflatex
```

If there are several possibilities how to complete your input, pressing  will list all possibilities.

Another helpful feature is the **shell history**, which stores the last commands you typed. By using the  and  keys, you can navigate through the last commands you executed.

A third feature are the so-called **manpages**. If you are not sure what a command does or how it works, you can always type

```
ba01@silicon1 ~> man mycommand
```

In this manual, sometimes a comment on a line of code is given, starting with a `#`. Everything coming after the `#` is a comment and is ignored by the shell.

Since this manual is only intended for a very brief introduction to the shell, here and here you can find some more comprehensive manuals.

2 Navigation

In the shell, you have to be able to navigate your file system. The file system is organized into a hierarchy of directories in which files reside. You should be familiar with the concepts of directories and files from file managers as Windows Explorer or Finder.

After starting the shell, your prompt will probably look like this:

```
ba01@silicon1 ~>
```

This prompt gives you three pieces of information: your user name (st in this case), the computer you are logged in (silicon1) and your current directory (~). The ~ refers to your home directory, where all your personal files are stored.

You can list the content of the current directory by using the `ls` (=list) command:

```
ba01@silicon1 ~> ls
bin  Documents  Downloads  Ex01  Pictures  Videos
```

You can change between different directories with the `cd` (=change directory) command:

```
ba01@silicon1 ~> cd Ex01
ba01@silicon1 ~/Ex01>
```

You can also change back to the previous directory:

```
ba01@silicon1 ~> cd ..
ba01@silicon1 ~>
```

where the `..` refers to the previous directory.

Another related command is `pwd` (=present working directory):

```
ba01@silicon1 ~> pwd
/user/st/Ex01
```

3 Files and directories

You can create new directories with the `mkdir` (=make directory) command:

```
ba01@silicon1 ~> mkdir Ex02
```

New (empty) files can be created with the `touch` command:

```
ba01@silicon1 ~> touch myfile
```

Remember that most of the files we use are text files (ASCII).

Files and directories can be moved or renamed with `mv` (=move):

```
# rename Ex01 to Ex02
ba01@silicon1 ~> mv Ex01 Ex02
# move a file from Ex01 to Ex02
ba01@silicon1 ~> mv Ex01/myfile Ex02
```

Files can also be copied with `cp` (=copy):

```
ba01@silicon1 ~> cp myfile mycopy
```

To copy a complete directory with all its content, use:

```
ba01@silicon1 ~> cp -r Ex01 Ex02
```

Files can be deleted with the `rm` (=remove) command:

```
ba01@silicon1 ~> rm myfile
```

An empty directory can be removed with `rmdir` (=remove directory):

```
ba01@silicon1 ~> rmdir Ex02
```

You can also remove a directory and all of its content at once:

```
ba01@silicon1 ~> rm -r Ex02
```

Remember that `rm` and `rmdir` do not ask you whether you really want to remove the file! There is also no recycle bin, so your data is irretrievably lost!

You can archive several files into a single archive file using `tar`:

```
# create an archive
```

```
ba01@silicon1 ~> tar -cvf myarchive.tar file1 file2 file3
```

```
# extract everything from an archive
```

```
ba01@silicon1 ~> tar -xvf myarchive.tar
```

`tar` does not compress the files itself, you can use `gzip` for that:

```
ba01@silicon1 ~> gzip Ex01.tar
```

`gunzip` can be used to decompress a compressed file:

```
ba01@silicon1 ~> gunzip Ex01.tar.gz
```

4 Text file editing

There is a large number of text file editors which can be used on the command line. For beginners, a very easy to used editor is probably `nano`. You can start it using

```
ba01@silicon1 ~> nano myfile
```

Within `nano`, you can navigate using the arrow keys. If you changed the file, you can save the changes using `ctrl` + `o`. You can leave `nano` by typing `ctrl` + `x`.

A much more powerful (and more difficult to use) editor is `vi`. It can perform all kinds of tasks like replacing words, copy and paste and syntax highlighting for many programming languages. However, an introduction to `vi` is beyond this manual. There are several manuals on the web, e.g. this manual.

If you only want to read a text file, you can also use `less`. You can navigate using the arrow keys, or using `Space` and `b` to scroll pagewise.

You can concatenate several text file to a single file using `cat`.

7 Overview of commands

Table 1 provides you with all commands and keywords you need for the beginning.

Table 1: List of commands and keywords

Command	Function
ls	List directory
ll	List directory (table with file size etc.)
cd	Change directory
pwd	Present working directory
man	Display manpage (help)
echo	Prints a variable
mkdir	Create directory
touch	Create empty file
cp	Copy file
mv	Move (also rename)
rm	Remove file
rmdir	Remove empty directory
tar	Archive program
gzip,gunzip	Compression program
cat	Concatenate several text files
nano	Simple text editor
vi	Complex text editor
less	Text reader
grep	Text search
sed	Text substitution
awk	Text manipulation (quite general)
Keyword	Function
~	Home directory
.	Current directory
..	Parent directory
*	Matches all files
for in; do; done	Shell loop
if; then; fi	Shell conditional