Hands-on Liger: GPU and AI

- Pierre-Emmanuel Guérin
- Davide Rovelli
- Hugues Digonnet

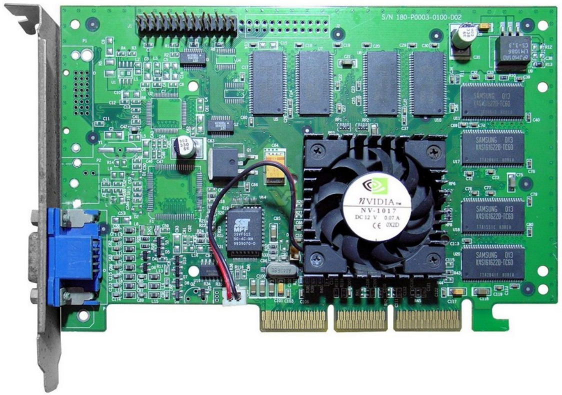
supercomputing.ec-nantes.fr / @cnscfr

CENTRALE NANTES SUPER COMPUTING

GPU computation

What is a GPU ?

- Graphical processing units
- Modern GPUs date early 2000's
- Original purpose: fast image rendering
- After the instroduction of framework (CUDA, OpenCL ~ 2006), increasingly used for high performance computations
- Modern AI / DL on GPUs from 2010s

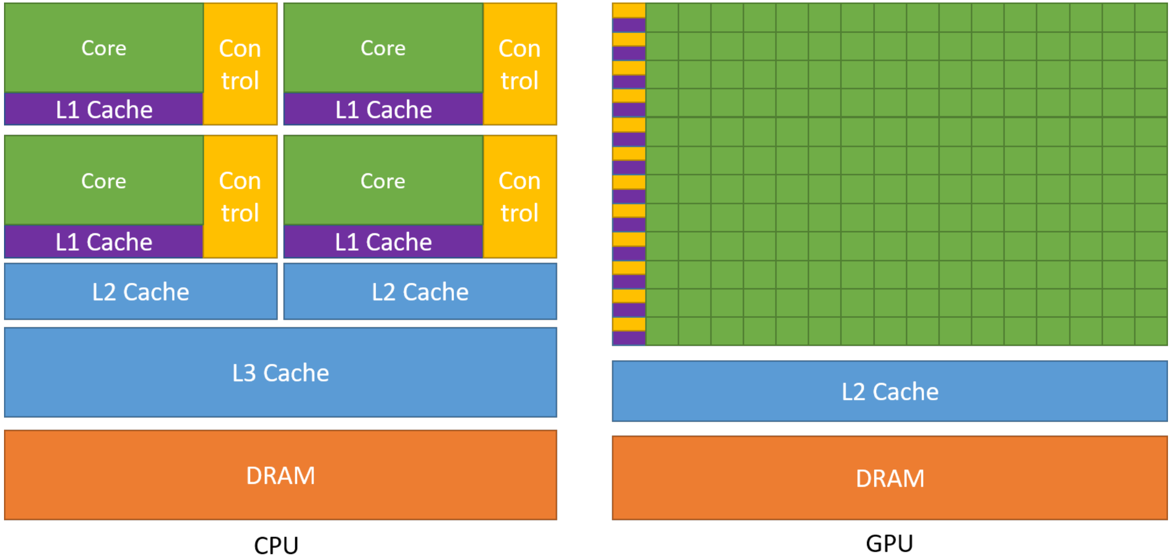


NVIDIA Geforce256 ~ 1999

NVIDIA RTX3090 ~ 2020

CPU vs GPU

Core computing unit, fast and sequential
Intel Xeon platinum latest gen:
***32** cores*
@ 2.9 GHz



Graphics, Massively parallel
NVIDIA V100 :
***5760** cores*
@ ~1.7GHz

GPU and AI

GPU is suitable for AI workloads

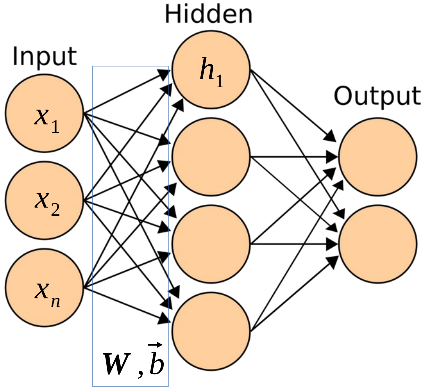
- AI jobs rely heavily on **massive tensor** (basic) **operations**
- The high parallelism of the GPU enhance such operations for trainings involving large datasets and models

  ex : Typical neuron update operation in NNs

  $$h_1 = g\left(\vec{w}\,\vec{x} + b\right)$$

  where vectors are of size *n ~ 1k.*

  GPU can fit 1 entire vector and update it in few clock cycles, while CPU has to scan through the vector



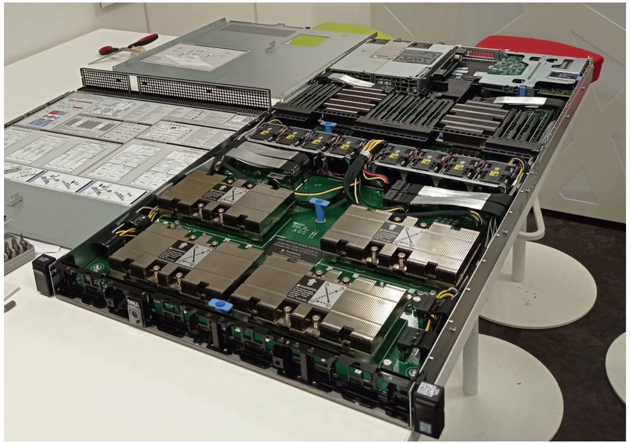
- NVIDIA offers specialized cores (tensor cores) that optimise this type of operations further *

* https://developer.nvidia.com/blog/programming-tensor-cores-cuda-9/

GPUs on Liger

Hardware resources

- 4 x servers with 2 x NVIDIA K80:
  legacy simple GPUs
  viz[01-04]
  useful for testing, standard AI workload
- 1 x server with 4 x GPU Nvidia Tesla V100 + NVLink
  turing01
  very fast modern AI workload
- More “turing” ↑ coming soon!



Integrated in Liger system

- Remote SSH access to one of the login nodes for all operations
- Job submission through **slurm** (srun, sbatch)
- Large storage, user space SCRATCH and HOME
- Fast interconnection between nodes

Environment for AI applications

- Jupyter with Python, TensorFlow and common AI libraries
- GPU resources are configured to host **containerised** applications. The container engine on Liger is **singularity**
- Non-containerised applications are not supported
- Container allow you to **make your own** environment
- Pre-build containers can be found on Liger and on the liger-ai-tools repo. Container description here

Request an account

- Know your Rights & Duties as a Liger User
  - https://supercomputing.ec-nantes.fr/charter
  - READ, AGREE & SIGN charters listed above
  - Send ONLY pages with your signature to your teacher.
  - Preferred Scan+PDF to paper.
  - Important: ask for access to GPU resources (not available by default)
- Use SVP to send the documents and for any technical problem
  - https://svp.ec-nantes.fr (supercomputing)

Hands-on: AI + GPU

Important resources

- Detailed AI docuementation is available on Liger official docs:

  https://ecn-collaborations.pages.in2p3.fr/liger-docs/artificial_intelligence/overview/

- Reference repository with useful tools:

  https://gitlab.in2p3.fr/ecn-collaborations/liger-ai-tools

- Container registry:

  https://gitlab.in2p3.fr/ecn-collaborations/liger-ai-tools/container_registry

Workflow

- Connect to Liger via SSH
- Download / copy your programs and data to Liger
  - FastX portal file transfer or `scp` for copying
  - `wget, curl`: command line tool for downloading
- (optional) Create or pull your container environment
  - `singularity pull / create`
- Submit the job via slurm on the gpu partition. It is possible to select a specific node as well

Jupyter - run

- Jupyter interactive session for TensorFlow AI jobs are available on Liger
- Simple submission process via a ready-to-use script

```
$ ssh myUsername@liger.ec-nantes.fr
$ git clone https://gitlab.in2p3.fr/ecn-collaborations/liger-ai-tools.git
$ cd liger-ai-tools
$ ./jupyter.run
```

- A link is displayed that will open a new instance of Jupyter Lab on a new browser window
- Missing some modules? Ask us at cnsc-help@ec-nantes.fr or open an SVP ticket

Jupyter - settings

- Under the hood, the script jupyter.run submits a slurm job that launches a specific container with Jupyter
- The script is launched with the following settings by default:
  - Target node is turing01
  - 1 GPU max
  - 5h max computation time
- It is possible to modify them

Jupyter - terminantion

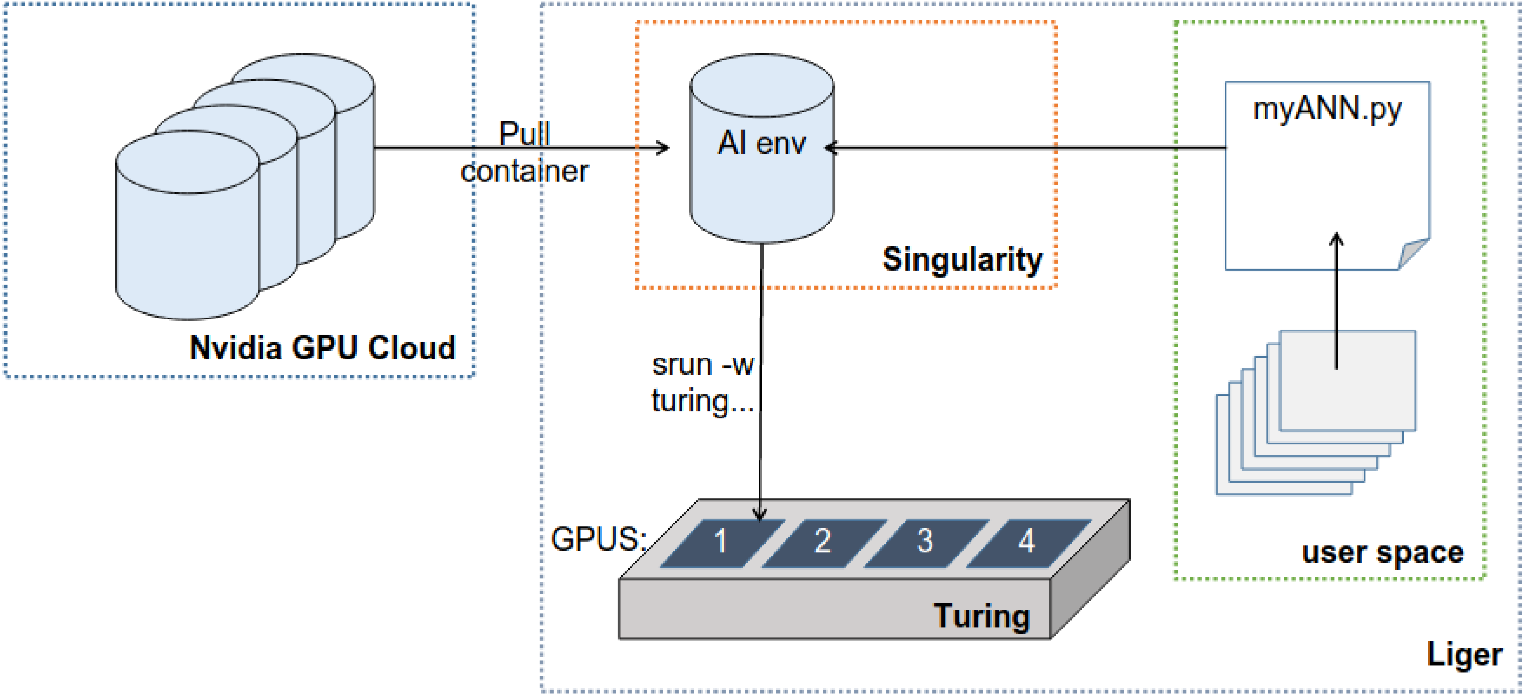
- If you are automatically logged out from your terminal, your computer switches off, you close your browser, the jupyter session will still be active
- Make sure to terminate your jupyter session when you are done! Via the Jupyter app: File → quit, or via `scancel`
- If you don’t, you might disrupt other users’ work

Containerised applications

- **singularity** + Nvidia optimised containers (NGC)
  - ~~module~~
  - Pre-built containers available on Liger at:

    /softs/singularity/containers/ai
  - Customisable environments via Docker or singularity recipes
- Launch containers via **slurm -** specifying account and QOS

Containerised applications - diagram



MNIST AI training submission demo

- Handwritten digit classification on Liger
- https://ecn-collaborations.pages.in2p3.fr/liger-docs/artificial_intelligence/quick_start/

Pulling containers demo

Use singularity to pull

  - Our pre-made container
  - Any container from any docker, singularity or OCI compliant registry!

- Build containers, beyond the scope of this hands-on

Sample sbatch files

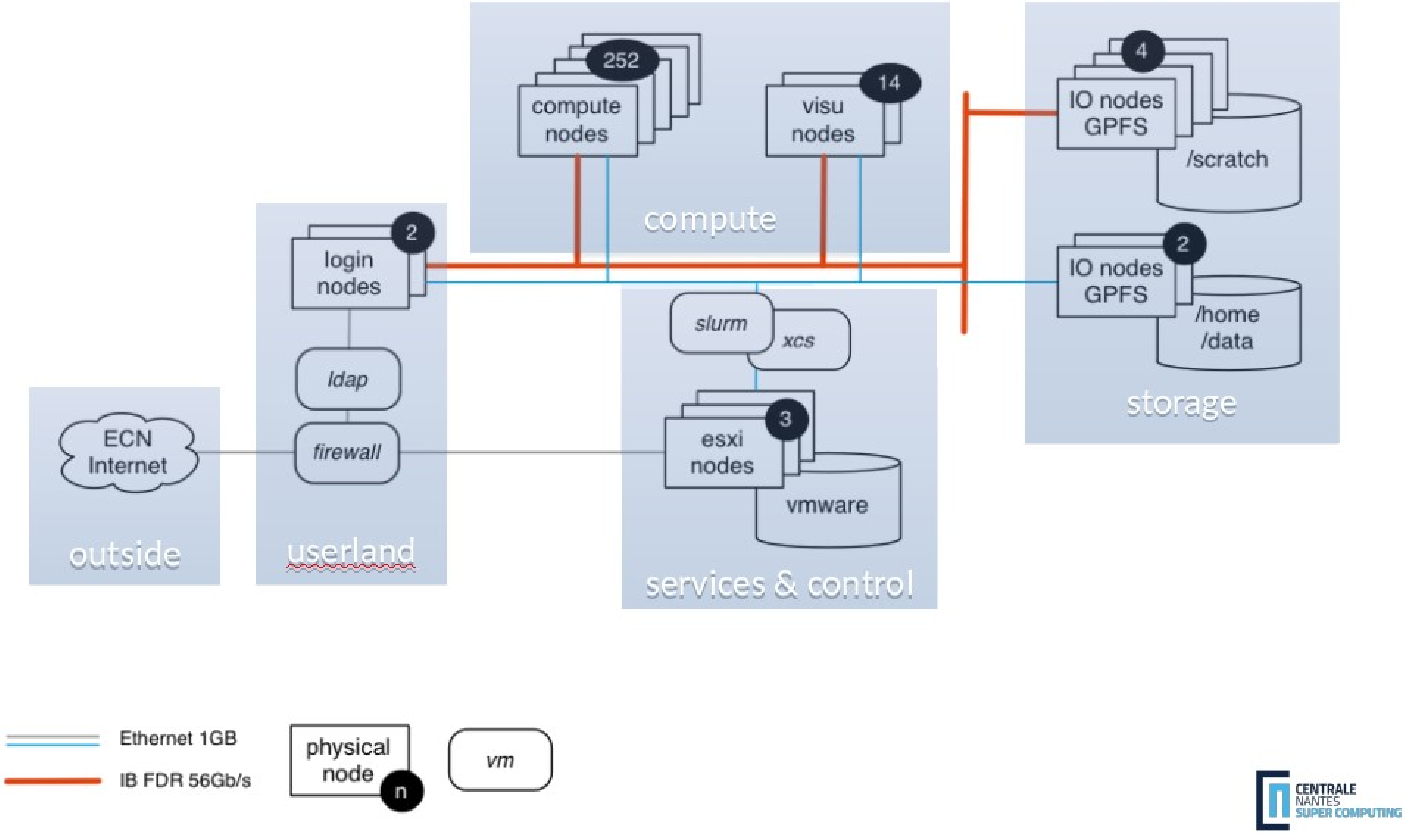
- Use sbatch for long trainings
- Template and examples:
- https://ecn-collaborations.pages.in2p3.fr/liger-docs/artificial_intelligence/running_ai_jobs/

Questions?

Appendix

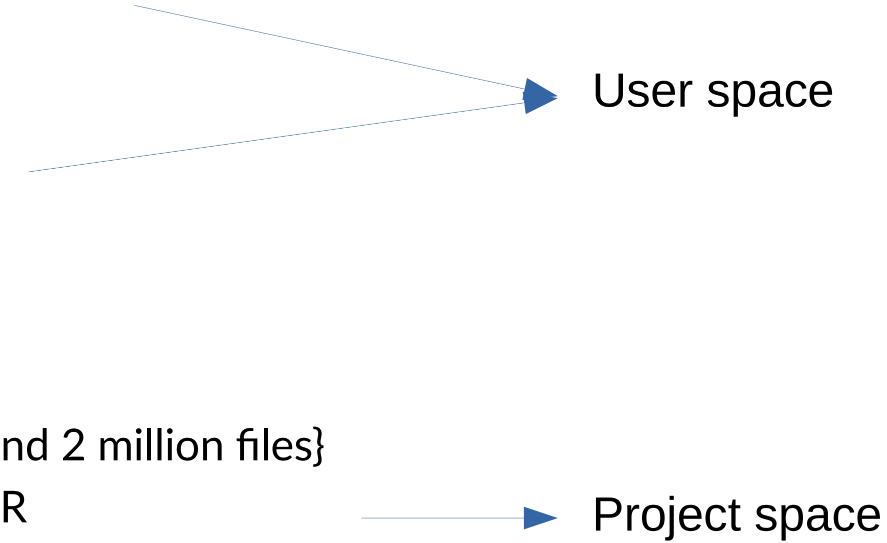
Liger basics

Liger: system topology



User Env : Filesystems & storage

- /scratch
  - 815 TB, 1 000 000 files quota per user
  - Your directory is $SCRATCHDIR
  - Computations and temporary files
- /home
  - 30 TB, 5GB quota soft per user
  - Your directory is $HOME
  - Sources files
- /data
  - 45 TB, quota per group={100GB and 2 million files}
  - Your project directory is $DATADIR
  - Permanent projects data and group sharing data



User space

Project space

Connect to Liger

- Client tool to connect on remote console:
  - Windows : PowerShell, putty, cygwin, mobaxterm
  - Mac/Linux : xterm, xquartz (only mac)
- Use a VPN to connect to Centrale Nantes network
- SSH secure protocol

```
$ ssh myUsername@liger.ec-nantes.fr
```

Move files to Liger

- SCP (or WinSCP for Windows): secure copy
  - Example: tranfer program to */home*

```
$ scp ./Desktop/program.c LIGER-ID@liger.ec-nantes.fr:~
```

  - WinSCP: GUI, same principle
- Download directly on Liger: git, wget etc.
  - Example: clone git repository on scratch

```
$ git clone https://repo.git $SCRATCHDIR
```

Job submission

- Compute resources are managed by a scheduler:
  - Liger uses **SLURM**
- Jobs are submitted to the scheduler
  - The scheduler choose available nodes (job running)
  - Or the computation is queued (job pending)

Job submission

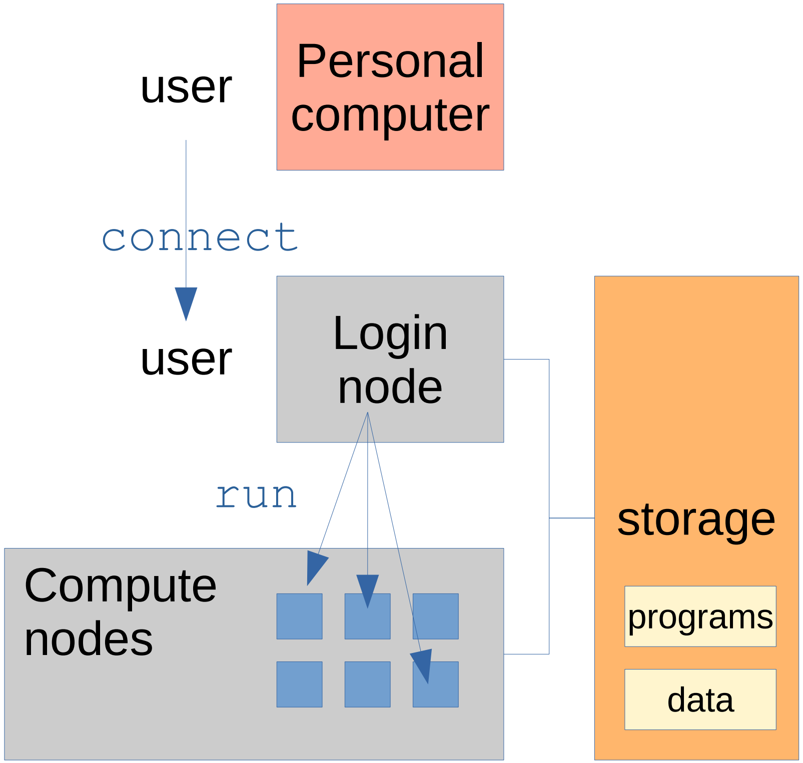
- With slurm commands you can run program on compute nodes.
  - Tell the SLURM what to run
  - SLURM will find the available resources and run the program

```
$ srun PROGRAM # run a job in the foreground
```

```
$ sbatch SCRIPT # run a job in the background
```

Liger : User environment

- You have 3 directories
- You can compile and test codes on login nodes
- You can use available softwares/libraries
- And you can submit jobs on nodes.



Load programs: *modules*

- Your environment is initally empty: no programs installed
- Modules is a tool to load or unload software packages.
  - List available software

```
$ module avail
```

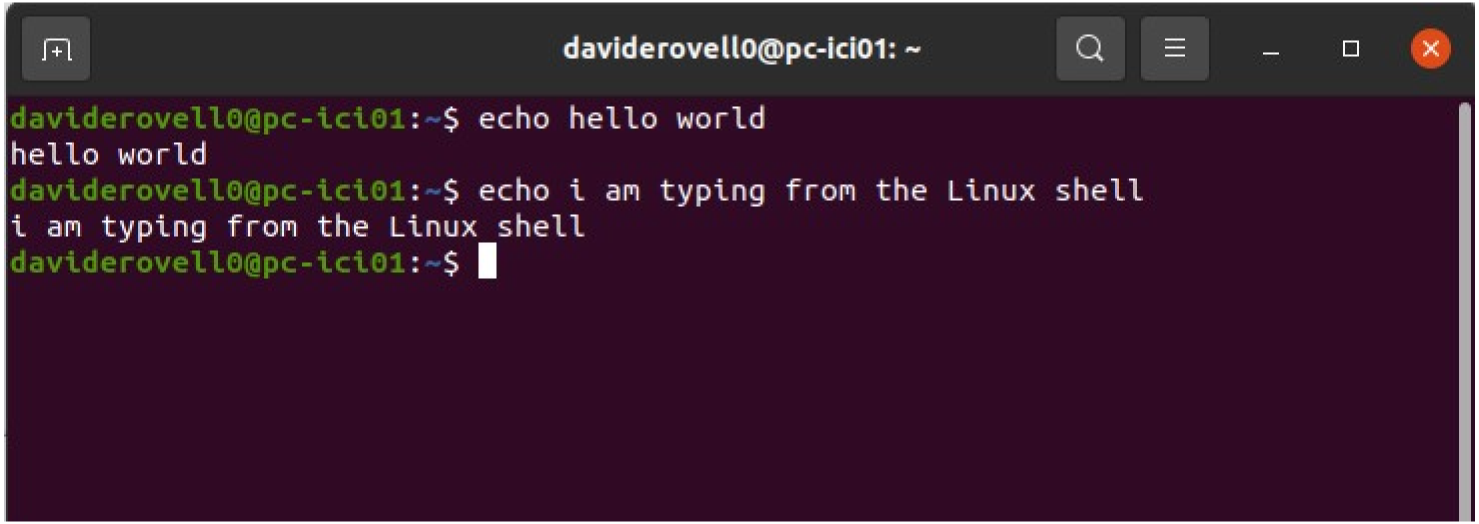
  - Load python

```
$ module load python
```

Linux shell basics

The Linux shell - terminal

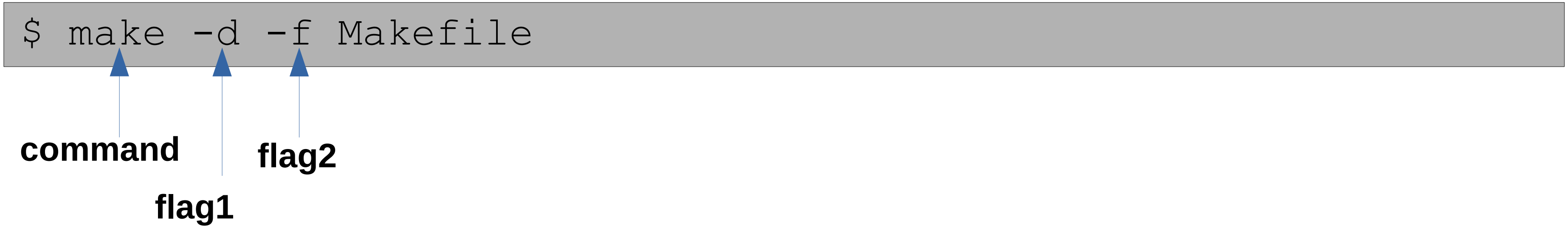
- No Graphical User Interface
- Issue commands through a CLI: command lone interface



```
daviderovell0@pc-ici01:~$ echo hello world
hello world
daviderovell0@pc-ici01:~$ echo i am typing from the Linux shell
i am typing from the Linux shell
daviderovell0@pc-ici01:~$ 
```

Issuing commands

- A command is a program that corresponds to a string of text. Use return to send a command, ctrl-C to interrupt it.
- A command can have **options**, set through **flags**.



```
$ make -d -f Makefile
```

command

flag2

flag1

- The “**-h**” flag shows a help guide for most commands

Navigating directories

- *pwd* – shows which directory you are in
- *ls* – list the files in the current directory
- *cd* – change to another directory

The base folder (top of the tree) is represented by “/”

The current folder is represented by “.”

The parent folder is represented by “..”

Editing files

- *cp* – copy a file to another location
- *mv* – move the file to another location (used for renameing as well)
- *rm* – remove a file, **-r** flag for recursive and folders

General rule: all commands are executed in the current folder (*pwd*), to execute a command in another folder use its path:

/absolute/path/to/file   *relative/path/to/file*

File operations

- Text editors: *nano*, *vi*, *gedit (requires GUI)*
  - Relies on a lot of key combinations, can be hard at the beginning. Use an editor wherever possible
- View file content: *cat*, *less etc*

```
$ cat your_file.txt
```

Run programs

- *gcc* – C / C++ compiler
- *python3* – run a Python script
- Javac – run a Java program
- …any installed program. Install with package manager:
  - Ubuntu, Debian: *apt*
  - RHEL: *yum*

Useful resources

There's much much more!

- https://supercomputing.ec-nantes.fr/publications/tutorials
- https://projects.ncsu.edu/hpc/Documents/unixtut/
- http://swcarpentry.github.io/shell-novice/