



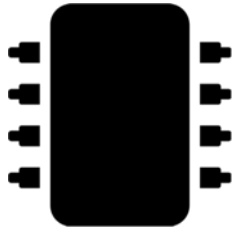
# Quick intro and how-to of **the IFB core cluster**

*feel the power of a 5000 cores and 27  
TB RAM computer*

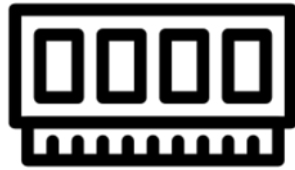
**Institut Jacques Monod**  
5th March 2020

**Julien Seiler**  
CNRS, IGBMC

# The IFB Core Cluster Infrastructure



5042 cores  
(hyperthreaded)



26 To RAM



400 To

*2.5 PB in  
Spring 2020*

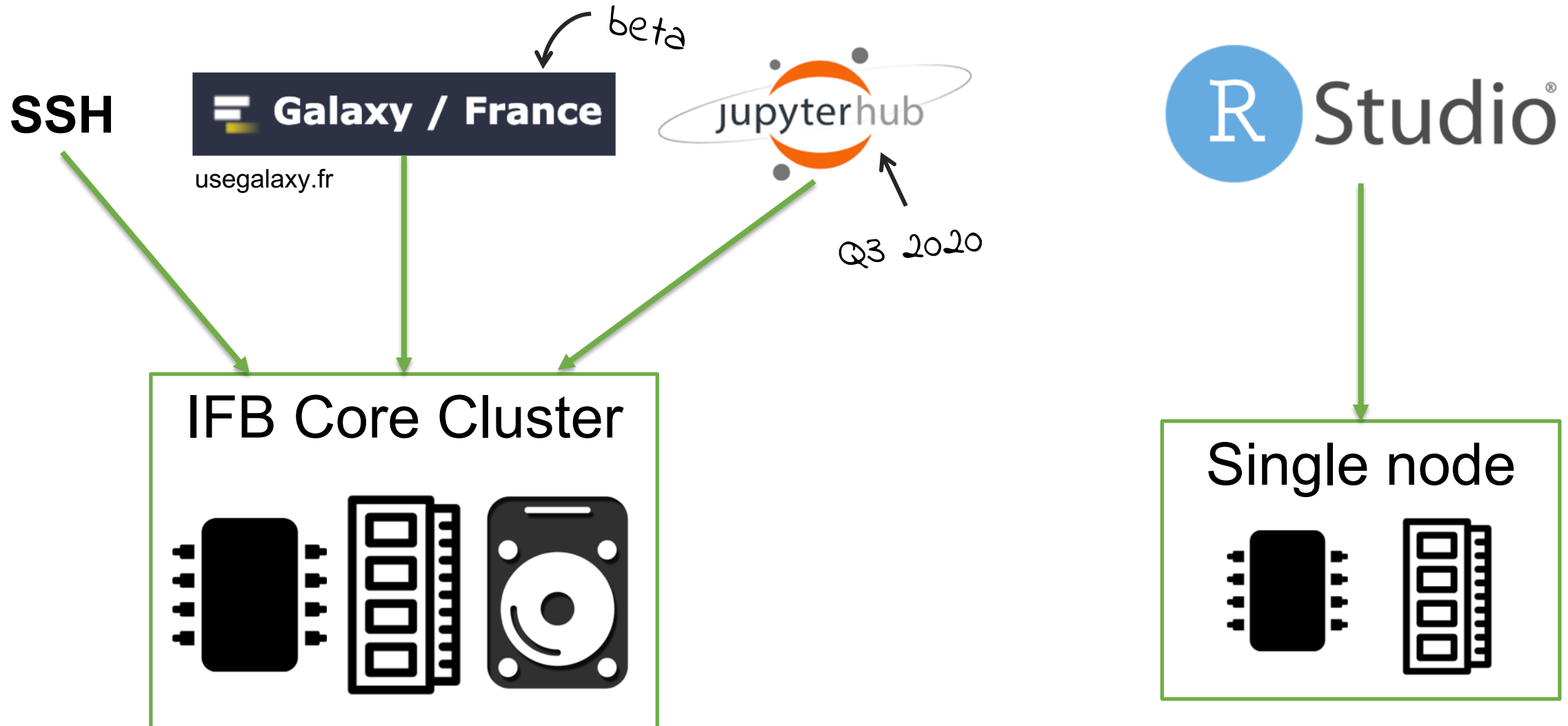


A support  
community made  
of users,  
administrators and  
experts



More than  
300 tools

# The IFB Core Cluster Infrastructure

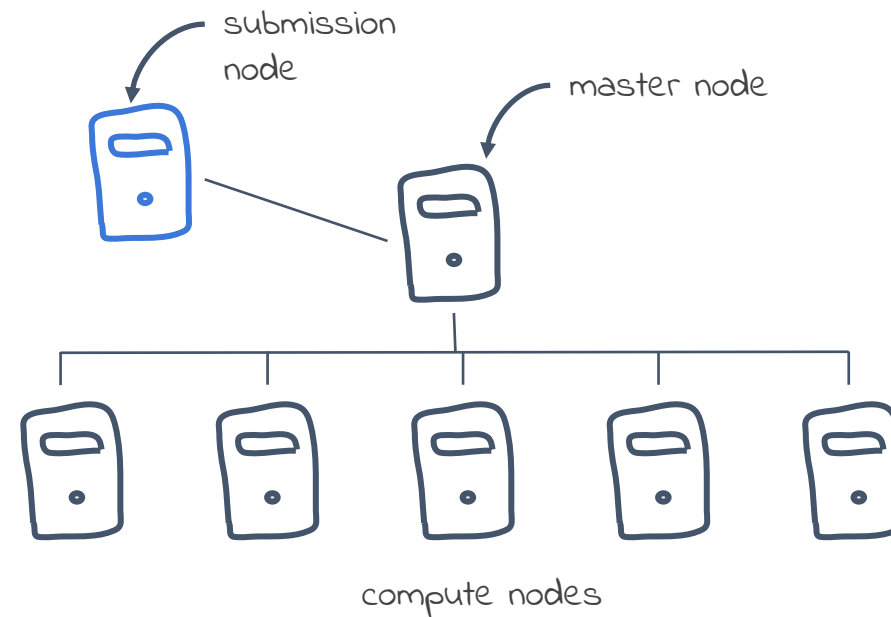


# The IFB Core Cluster Infrastructure

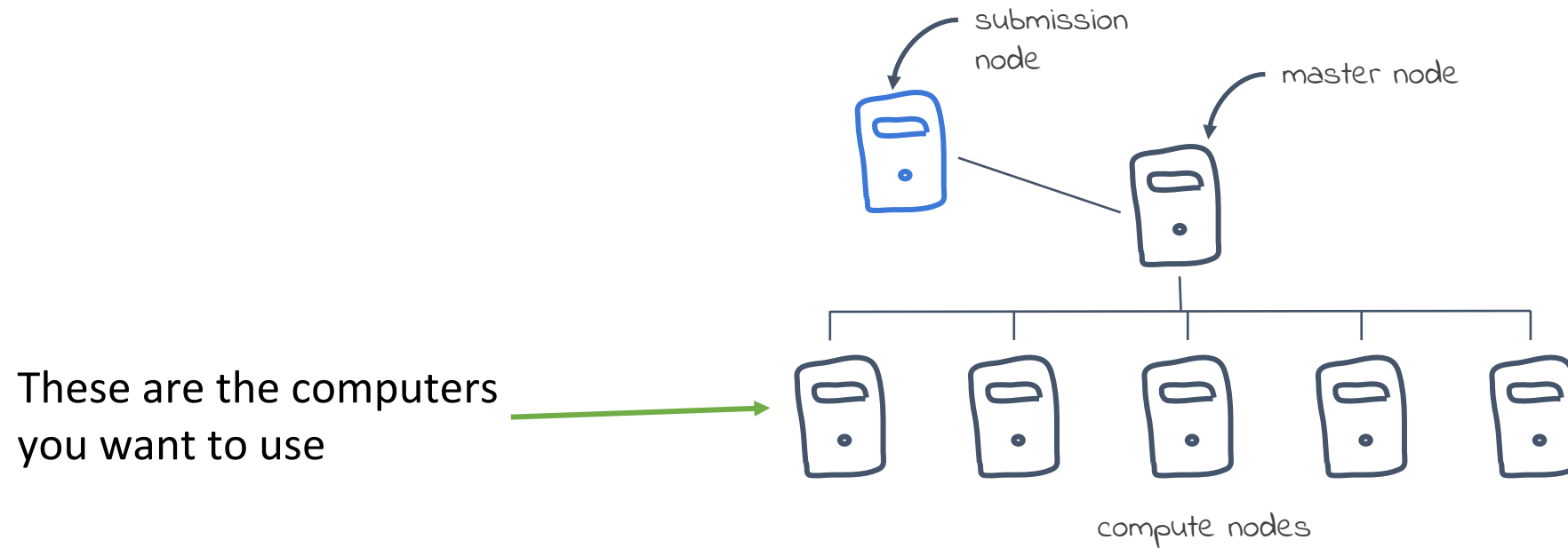
- Infrastructure administration is automated using Continuous Integration technologies :
  - Ansible
  - Git
  - GitLab CI
  - Support Bot
- Most IFB Core Cluster repositories are open to contribution
  - Help us manage the cluster infrastructure
  - Deploy bioinformatics software (conda, singularity, etc.)
  - Deploy new services

# What is a HPC Cluster ?

Basically, it is a bunch of computers working together



# What is a HPC Cluster ?



# What is a HPC Cluster ?

## How does a computer work ?

one or more chips 

A chip (or microprocessor) is responsible for executing elementary instructions requested by the software

RAM (Random access memory) 

RAM is used by the chip to process data (a personal computer has between 4 to 8 GB of RAM)

storage space 

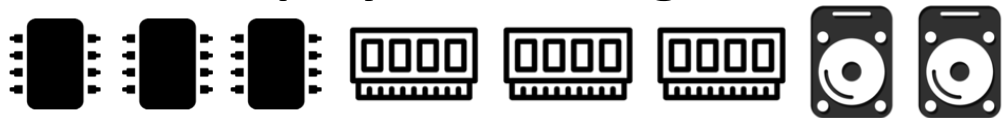
The storage space is used to keep huge amount of data in a more permanent way (a personal computer has an average of one TB of storage space)

# What is a HPC Cluster ?

## How does a computer work ?



A personal computer has enough resources to let you run a lot of tasks like **browsing the Internet**, **work with spreadsheet** or **text processing software**. Some personal computers have even enough resources to let **process videos** or **play 3D video games**.



However, personal computer are not powerful enough to run **massive data analysis programs**. Indeed, these programs need a huge number of processing units (10 to 100 CPUs), huge amounts of RAM (100 GB for some programs) and large data storage capabilities (several TB for a single research project).



# What is a HPC Cluster ?

A set of big computers connected together that can be considered as a single system.

A HPC cluster is usually located in a **data center**, *i.e.* a dedicated room providing all conditions required by HPC in terms of temperature, humidity, power supply and physical security.



This is not me

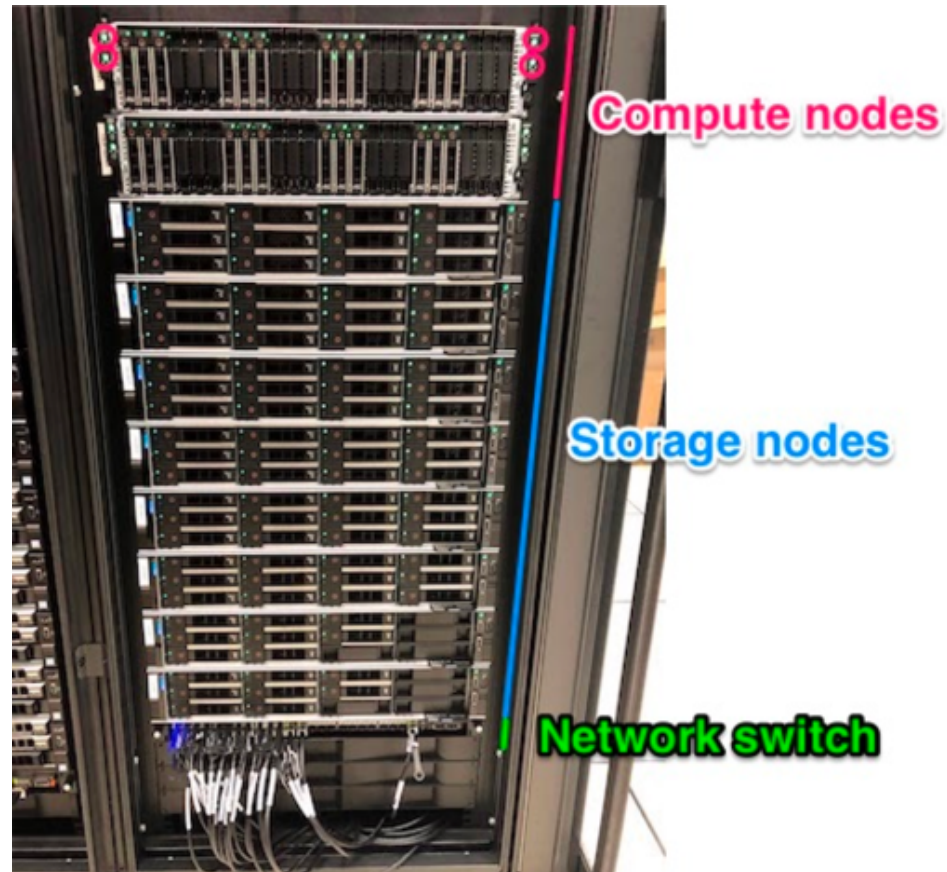
# What is a **HPC Cluster** ?

A data center contains **racks**



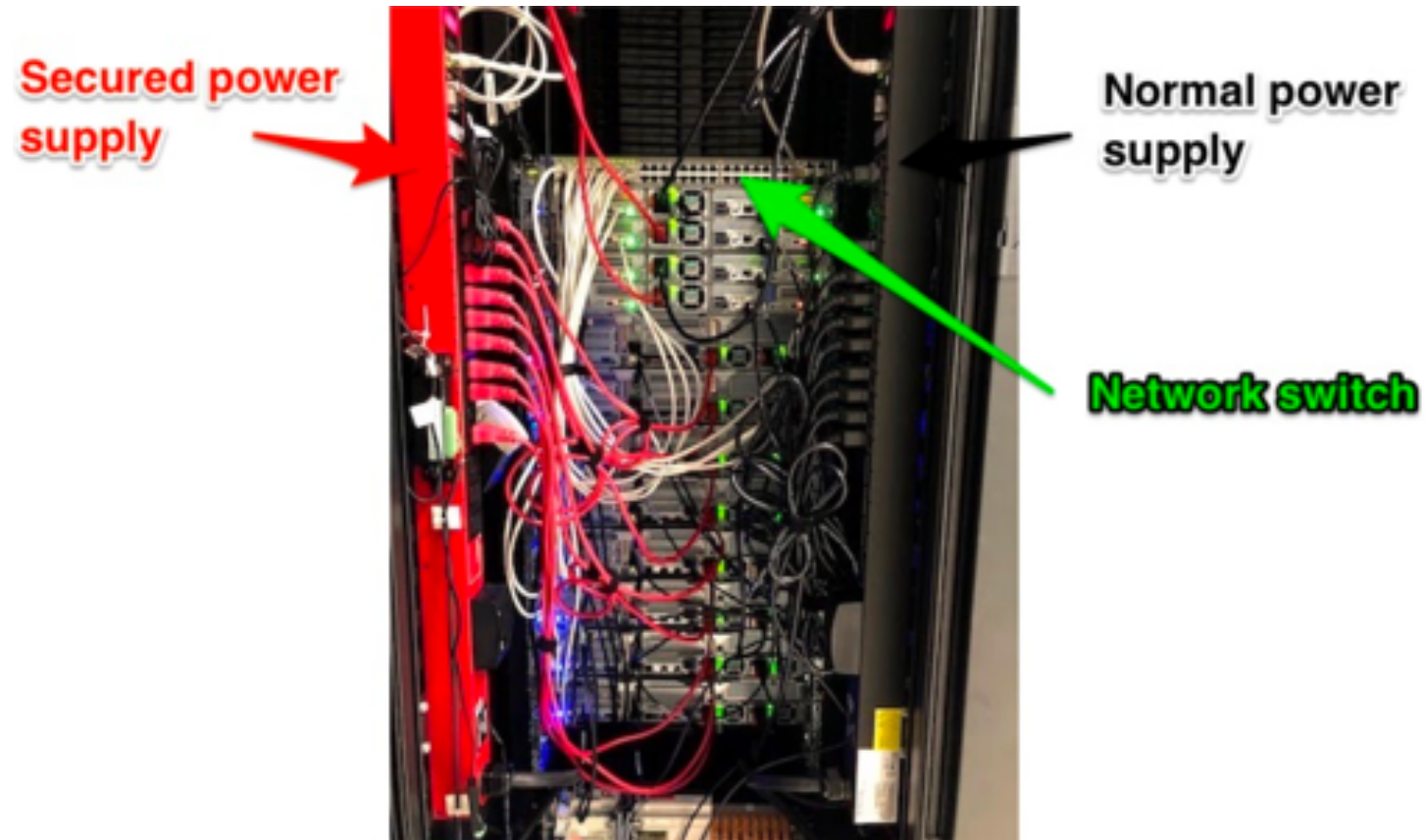
# What is a HPC Cluster ?

Each rack can hold several computers



# What is a HPC Cluster ?

Rear view



# What is a **HPC Cluster** ?

Inside a **computer** : a node = a physical machine

Each physical machine has one **motherboard**

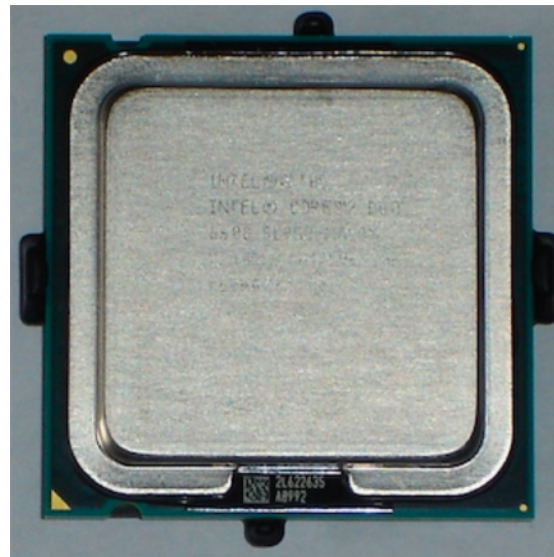


This motherboard has 2 **sockets** to plug **microprocessors**.  
A microprocessor is a **multicore** technology.

# What is a **HPC Cluster** ?

Do not get confused between Microprocessor and Core

A microprocessor is a physical chip



Nowadays, one microprocessor contains **several cores**.  
Each core behaves like a real separated microprocessor.

# What is a HPC Cluster ?

## The IFB cluster federation (NNCR)

Cluster	Data center location	Cores	RAM (GB)	Storage (TB)	Access modality
IFB Core	IDRIS - Orsay	5 042	26 542	400*	Open to all academic biologists and bioinformaticians
Genotoul	Toulouse	6 128	34 304	3 000	Open to all academics with priority to INRA/Occitane region (currently overloaded)
ABiMS	Roscoff	1 928	10 600	2 000	Open to all academic biologists and bioinformaticians
GenOuest	Rennes	1 824	7 500	2 300	Open to all academic biologists and bioinformaticians
Migale	Jouy en Josas	1 084	7 000	350	Open to all academic biologists and bioinformaticians
BiRD	Nantes	560	4 000	500	Open to all academic biologists and bioinformaticians

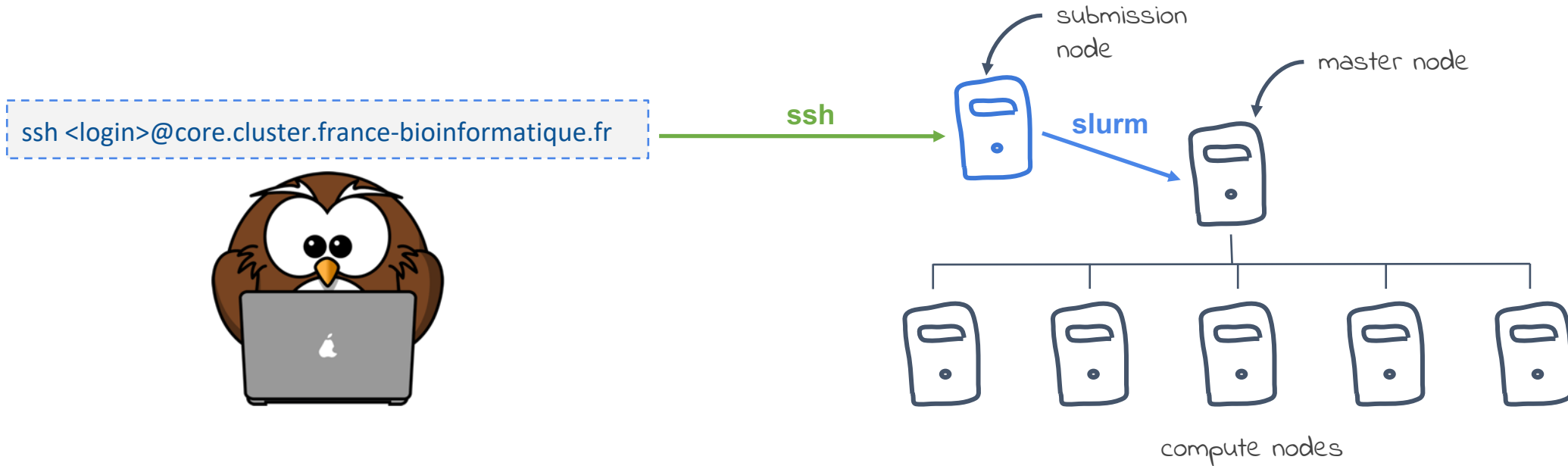
# Introduction to SLURM

## Common terms

- **Job** : a reservation of resources to run some analysis. A job is composed of one or more job steps that consume the reserved resources (eventually in parallel).
- **Job step** : part of a job that consist in the execution of a program. One job step can use multiple tasks. By default a job step uses one task.
- **Tasks** : a single process. One task can use multiple CPU (multi-threaded process).
- **CPU** : smallest computer processor unit (generally a single processor core).
- **RAM** : memory used by a processor to store data being computed

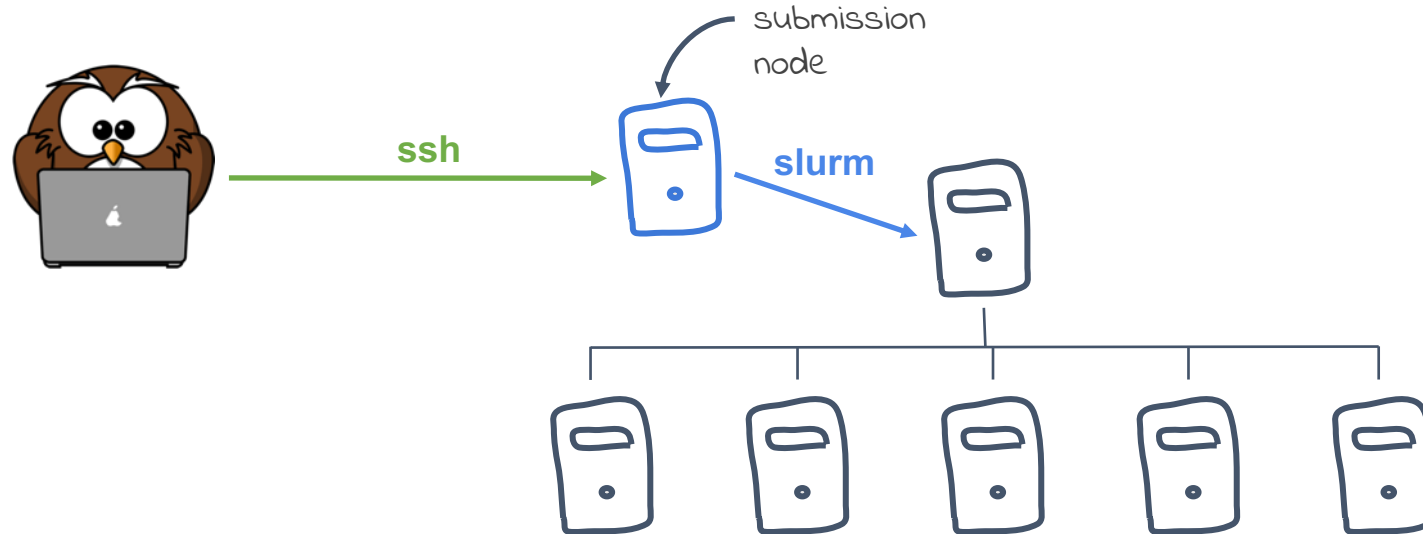


# Introduction to SLURM - How to submit a job



You are connected to the “submission node” of the cluster !

# Introduction to SLURM - How to submit a job



Don't run any computing software  
on the submission node  
It is very weak and not designed  
for computing !

# Introduction to SLURM - How to submit a job

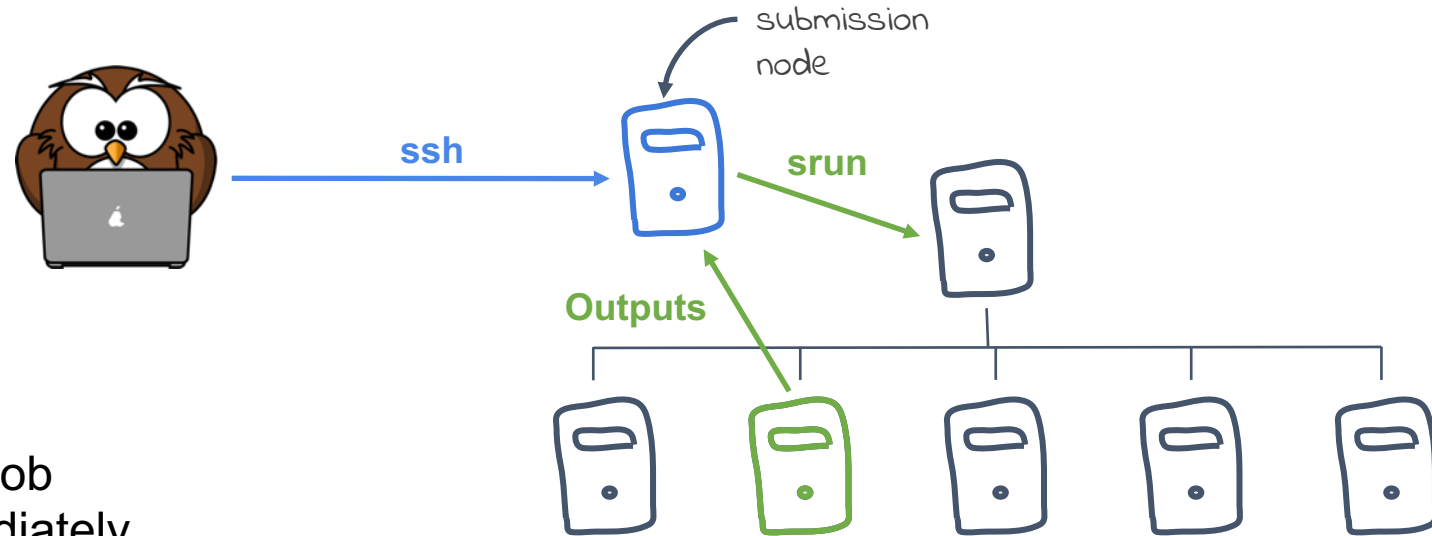
To run a simple command on the cluster, use **srun**

Example: `srun tar xvzf my_big_data.tar.gz`

**srun** will :

- Reserve CPU and memory for your job (by default 1 CPU and 2GB of RAM on a single node)
- Wait for these resources to become available
- Run the given command on the compute node selected by SLURM
- Send back the command outputs to the user terminal

# Introduction to SLURM - How to submit a job



**srun** : simple interactive job

- Starts or waits immediately
- Outputs are returned to the terminal
- You have to wait until the job has terminated before starting a new job
- Works with **ANY command**

```
$ srun tar xvzf my_big_data.tar.gz
```

```
tar xvzf my_big_data.tar.gz
```

# Introduction to SLURM - How to submit a job

## **srun** in brief

- `srun <command>`
- Default settings :
  - 1 CPU core
  - 2 GB RAM
- Common parameters :
  - `--cpus=`
  - `--mem=` (Warning : SLURM is enforcing memory usage !)
  - `--nodes=`
- Outputs comes in your console directly
- The console is blocked while your job is running

# Introduction to SLURM - How to submit a job

Most of the time, you don't want to run a single command and don't want to wait for each command to end to start the next one.

What you want is running a batch script !

A batch script can be any shell script (Bash, R, Python etc.) but most of the time we use **Bash**.

Here is a simple example : my\_script.sh

*shebang is mandatory !*

```
#!/bin/bash
```

```
srun tar xvzf my_data.tar.gz
```

```
srun analyse my_data
```

*each srun is a job step*

# Introduction to **SLURM** - How to submit a job

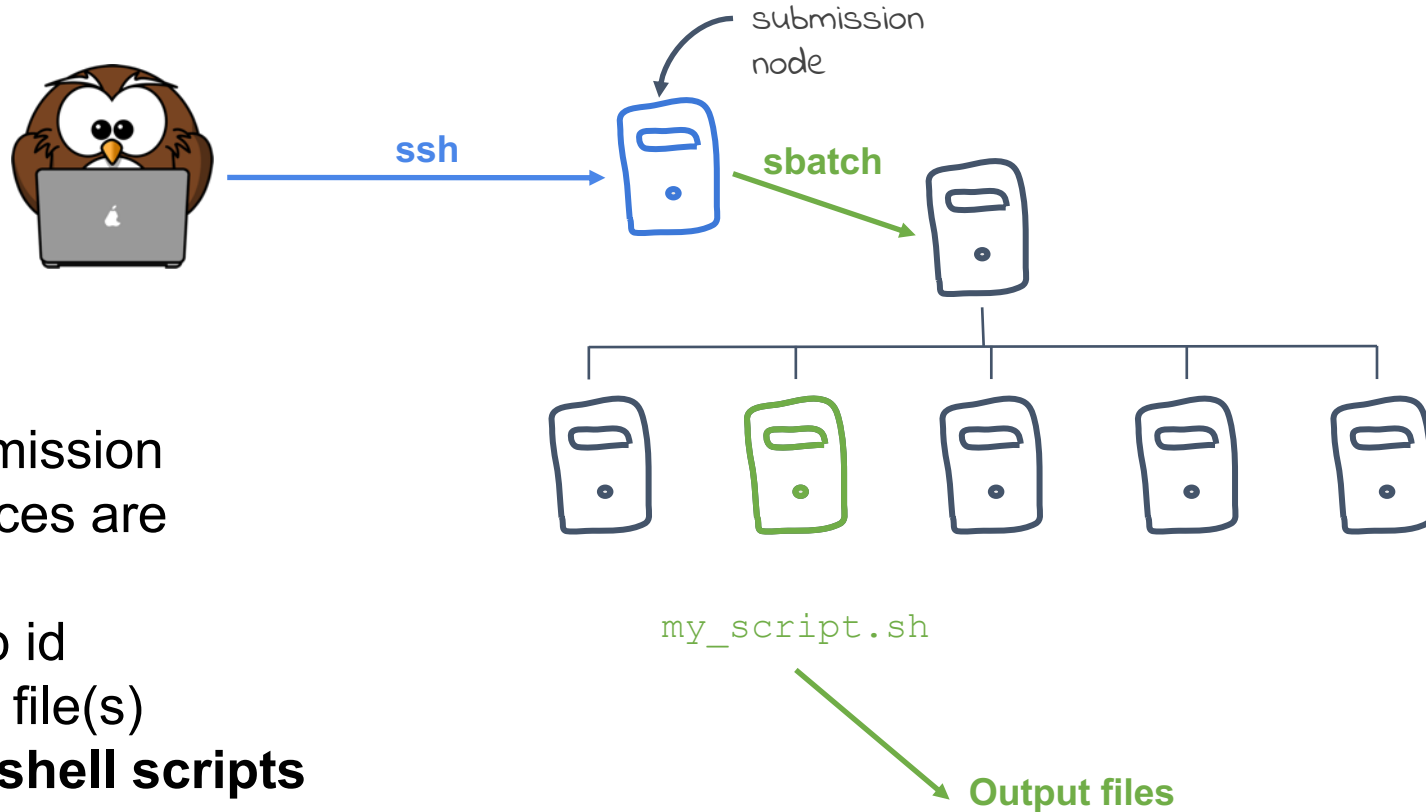
To run a batch script on the cluster, use **sbatch**

Example : `sbatch my_script.sh`

**sbatch** will :

- Reserve CPU and memory for your job (by default 1 CPU and 2GB of RAM on a single node)
- Place the job in the waiting queue and return
- When the resources are available start the job script step by step.
- Outputs are written to files

# Introduction to SLURM - How to submit a job



**sbatch** : batch job submission

- Starts when resources are available
- Only returns the job id
- Outputs are sent to file(s)
- Works **ONLY** with shell scripts

```
$ sbatch my_script.sh
```



# Introduction to **SLURM** - How to submit a job

## **sbatch** in brief

- `sbatch <command>`
- Default settings :
  - 1 CPU core
  - 2 GB RAM
- Common parameters :
  - `--cpus=`
  - `--mem=` (Warning : SLURM is enforcing memory usage !)
  - `--nodes=`
- `sbatch` is NOT blocking the console
- Outputs comes in files (`slurm-<jobid>.out` and `slurm-<jobid>.err`)
- Works ONLY with Shell script

# Introduction to SLURM - How to submit a job

## **sbatch** in brief

- You can pass sbatch parameters in your shell script directly

```
#!/bin/bash
#
#SBATCH -p surf                # partition
#SBATCH -N 1                   # number of nodes
#SBATCH -n 2                   # number of tasks
#SBATCH --mem 100              # memory pool for all cores
#SBATCH -t 0-2:00              # time (D-HH:MM)
#SBATCH -o slurm.%N.%j.out     # STDOUT
#SBATCH -e slurm.%N.%j.err     # STDERR
#SBATCH --mail-type=ALL        # can be BEGIN, END, FAIL or REQUEUE
#SBATCH --mail-user=your@email.com
```

```
srun --mem=50 bash -c "prepare_data > large.dataset"
srun big_computing_tool large.dataset
```

- `sbatch my_script.sh`

# Introduction to SLURM - Summary

- Get connected to the login node and keep working on it
- For **basic command** (cd, ls, mv, mkdir...), run it directly on the “**submission node**”
- For all the rest, including **bioinformatics tools**, prepend all command lines by **srun** so that your job will be submitted to the cluster master and then run on a **node** of the cluster
- For batch treatment (like pipeline) use **sbatch**

# Introduction to **SLURM** – Job control

## **squeue**

View all job running on the cluster

```
$ squeue
```

View only my jobs

```
$ squeue -u <my_login>
```

View only my RUNNING jobs

```
$ squeue -t RUNNING -u <my_login>
```

# Introduction to SLURM – Job control

## Job resources

View resources used by a job

```
$ sacct --format=JobID,Submit,MaxVMSize,Start,NodeList,CPUTime,State -j <job_id>
```

View detailed information about one running job :

```
$ scontrol show jobid -dd <job_id>
```

# Introduction to **SLURM** – Job control

## **sinfo**

View available Slurm partitions

```
$ sinfo -l
```

View available Slurm nodes

```
$ sinfo -Nl
```

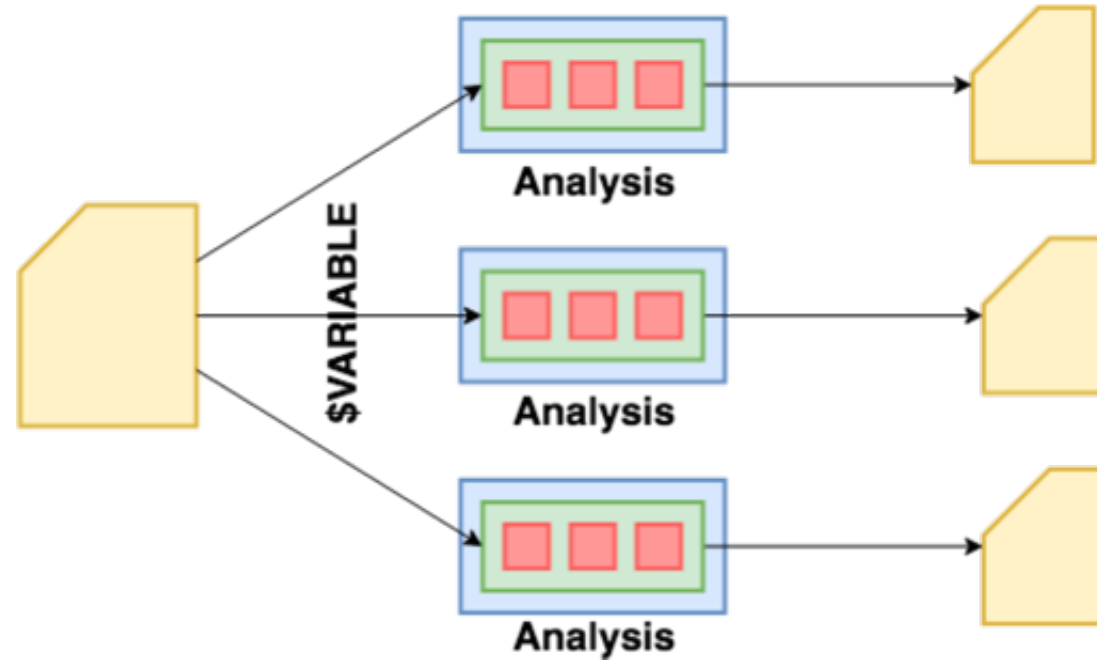
# Introduction to SLURM – Parallelization patterns

## Input data splitting



# Introduction to SLURM – Parallelization patterns

## Variables exploration





# Introduction to SLURM – Parallelization patterns

## Fastqc example

fastqc.sh

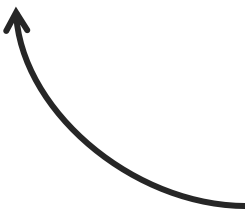
```
#!/bin/bash
#SBATCH --array=0-3 # 4 jobs
#SBATCH --cpus=16 # 16 cpu cores
module load fastqc/0.11.8
INPUTS=(../fastqc/*.fq.gz)
srun fastqc -t 16 ${INPUTS[$SLURM_ARRAY_TASK_ID]}
```

```
$ sbatch fastqc.sh
Submitted batch job 3161045
```

multiqc.sh

```
#!/bin/bash
srun multiqc .
```

```
$ sbatch --dependency=afterok:3161045 multiqc.sh
```



Approach	Total duration of the treatment
Sequential, single thread	13 minutes 53 seconds
Sequential, 16 threads	11 minutes 45 seconds
Parallel, 4+1 jobs, 16 threads	4 minutes 2 seconds

# **Demo – the sort challenge**

Is it possible to **generate and sort 25 millions random numbers** in less than **30 seconds** ?

# Demo – the sort challenge

Is it possible to **generate and sort 25 millions random numbers** in less than **30 seconds** ?

```
shuf -i 1-10000000000 -n 25000000 -r
```

# Demo – the sort challenge

Is it possible to **generate and sort 25 million random numbers** in less than **30 seconds** ?

```
shuf -i 1-10000000000 -n 25000000 -r
```

```
sort -g numbers.txt
```

# Useful links

Request an account:

<https://www.france-bioinformatique.fr/fr/ifb-core-cluster-account-request>

Community support:

<https://community.cluster.france-bioinformatique.fr/>

Learn SLURM in 5 minutes:

<https://asciinema.org/a/275233>