

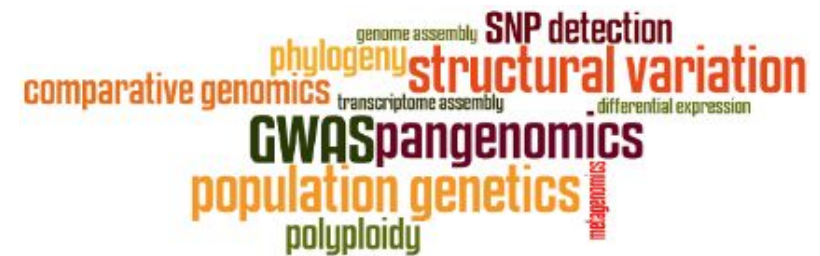
Session de formation 2023



bioinformatics platform dedicated to the genetics and genomics of tropical and Mediterranean plants and their pathogens



Mutualisation



Cacao

Banana

Coffee

Rice

Palm

Cassava

Pseudocercospora

Magnaporthe

Cacao

Banana

Coffee

Rice

Palm

Cassava

Pseudocercospora

Magnaporthe

South Green

bioinformatics platform



4
institutes



3 research
units



25+



Storage and
computing resources

Tools

Trainings



400+



Meso@LR au CINES

1090 threads :

- 35 standard nodes

- 2 bigmem nodes

- 1 GPU node

500 To of replicated storage



CINES

1130 threads:

- 30 standard node

- 1 supermem node

- 1 GPU node

150 To on 3 NAS + 210 To scratch



400

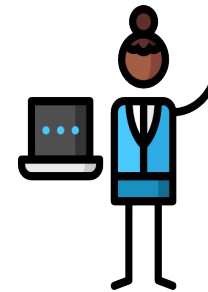
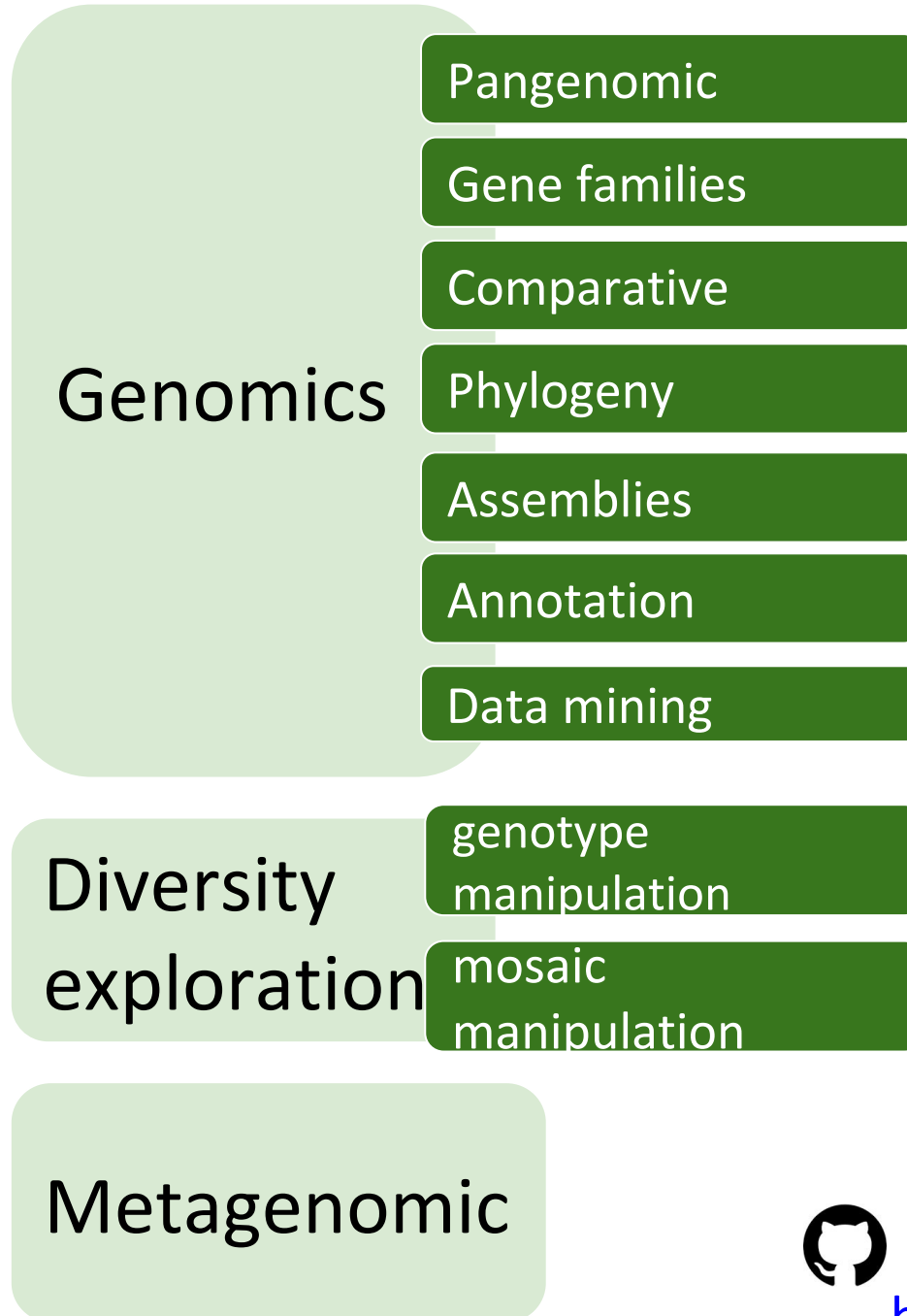


600+
tools

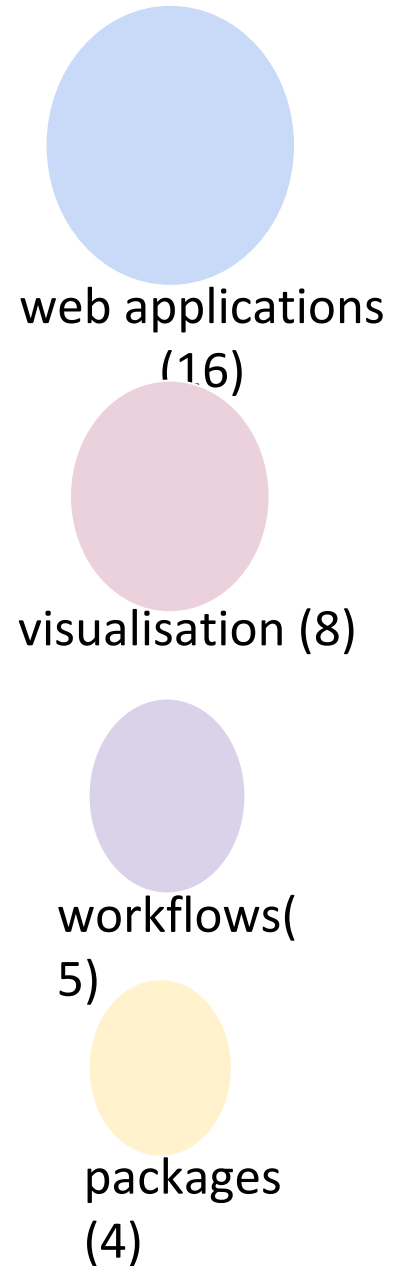
Resources mutualised at Meso@LR through the
Mudis4Ls project (purchase/storage/data)



Collaborative development of tools



**+20
tools**



<https://github.com/SouthGreenPlatform>

I-Trop

Plant & Health Bioinformatics Platform

<https://bioinfo.ird.fr>

!



AUORE
COMTE

JACQUES
DAINAT

ALEXIS
DEREEPER

BRUNO
GRANOULLAC

JULIE
ORJUELA-

NDOMASSI
TANDO

CHRISTINE
TRANCHANT

bioinfo@ird.fr



[@ItropBioinfo](https://twitter.com/ItropBioinfo)



Florian Charriat
Antoni Exbrayat



Guilhem Sempere



Bruno Granouillac
Jacques Dainat

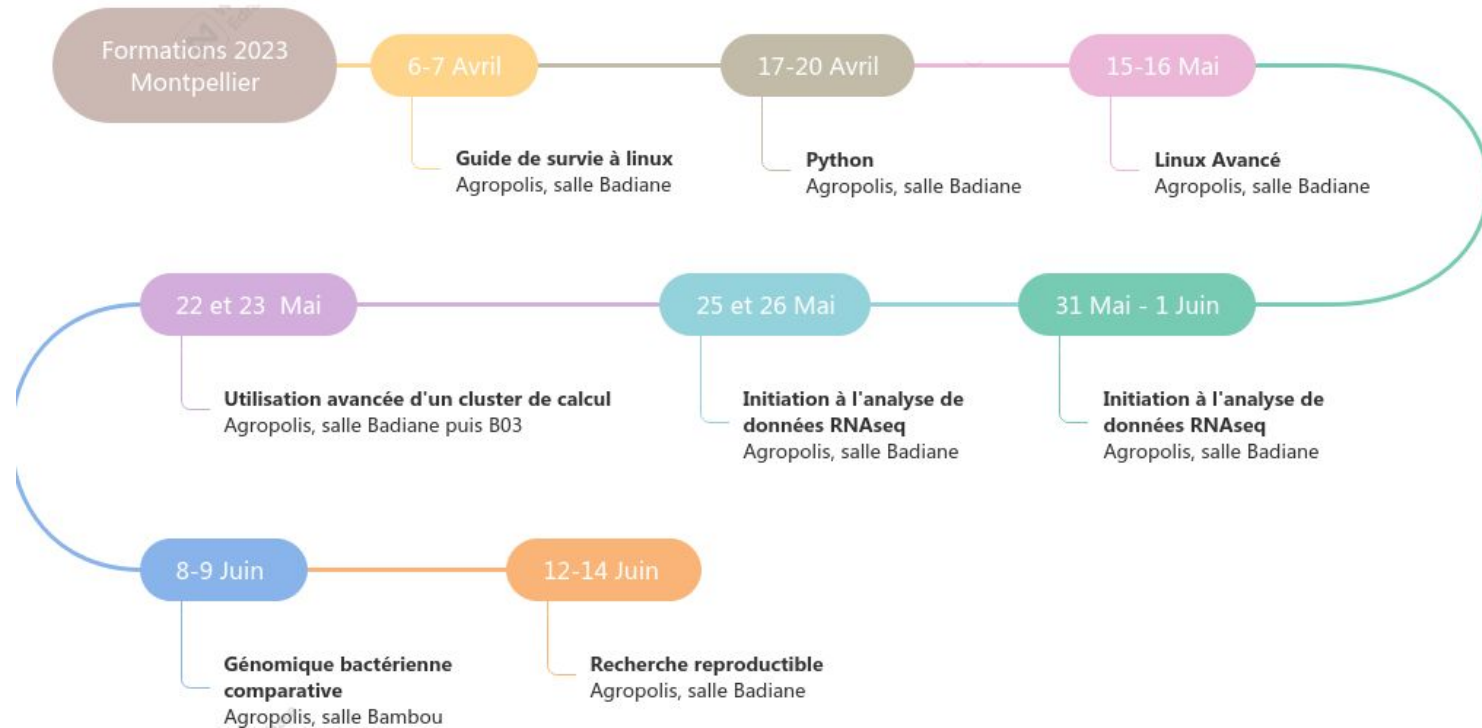


Nicolas Fernandez



Thomas Denecker

And more collaborators !



Modules de formation 2023

- Toutes nos formations :
<https://southgreenplatform.github.io/trainings/>
- Topo & TP : [Cluster avancé](#)
- Environnement de travail : [Logiciels à installer](#)

HPC avancé

www.southgreen.fr

<https://southgreenplatform.github.io/trainings>



Objectif

Acquérir des notions avancées pour utiliser un cluster

Applications

- Installer ses propres logiciels
- Créer ses propres modules environment
- Lancer des jobs arrays via SGE
- Utiliser et installer Singularity
- Créer des conteneurs singularity

INSTALLER DES LOGICIELS

Récupération
des sources

Etape 1

Récupération
des sources



Décompression
des sources

Etape 1

Etape 2

Récupération
des sources



Décompression
des sources



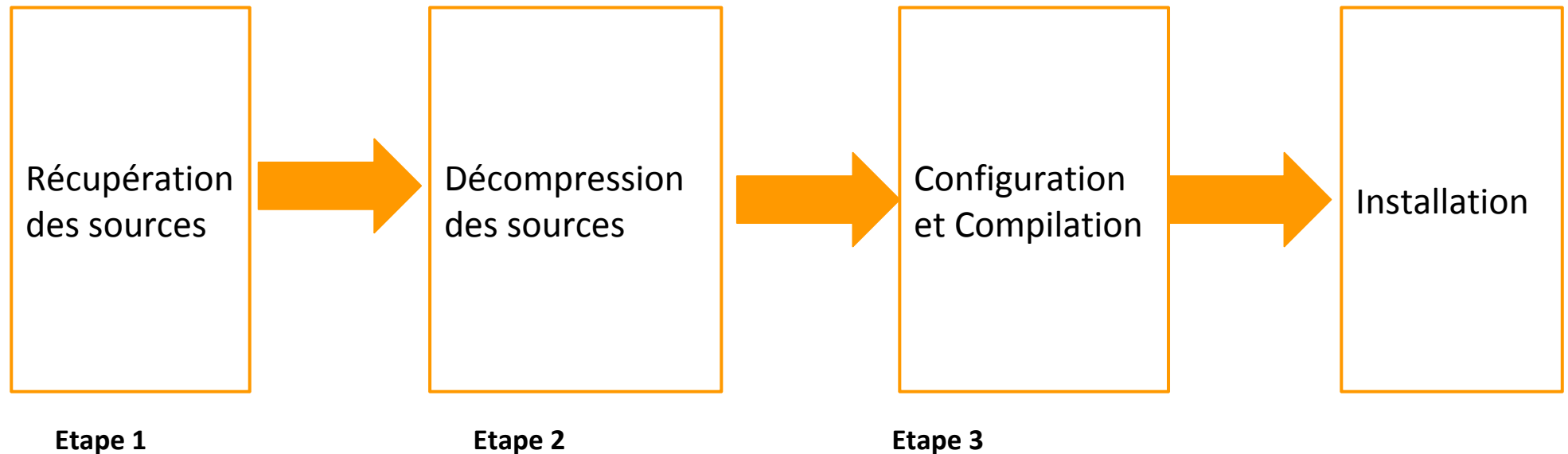
Configuration
et Compilation

Etape 1

Etape 2

Etape 3

4 grandes étapes



- Créer un répertoire sources dans son home
- Créer un répertoire softs
- Créer dans softs un répertoire par logiciel et par version

- Téléchargement direct depuis le site
- Utiliser wget + lien vers le logiciel
- Faire un git clone du dépôt

- Fichier .tar.gz: `tar xvfz fichier.tar.gz`
- Fichier tar: `tar -xvf fichier.tar`
- Fichier .tar.xz: `tar xf fichier.tar.xz`
- Fichier tar.bz2: `tar xjf fichier.tar.bz2`
- Fichier .zip: `unzip fichier.zip`

- Suivre les instructions d'installation: README ou INSTALL

```
./configure --help  
./configure  
./configure  
--prefix=/home/user/softs/name-version
```

Détecte les infos systèmes et
configure le code source pour s'y
adapter

- Suivre les instructions d'installation: README ou INSTALL

```
./configure --help  
./configure  
--prefix=/home/user/softs/name-  
version
```

Détecte les infos systèmes et configure le code source pour s'y adapter

```
make
```

Réalise la compilation du programme

make install

Copie les binaires (exécutables)
produits à l'endroit spécifié dans
le prefix

make install

Copie les binaires (exécutables)
produits à l'endroit spécifié dans
le prefix

```
echo 'export  
PATH=/home/user/soft/bin:$PAT  
H' >> ~/.bashrc  
source ~/.bashrc
```

Modifier son ~/.bashrc pour
pouvoir lancer le logiciel

INSTALLER DES PACKAGES PERL

- Téléchargement direct depuis le site
- Utiliser wget + lien vers le logiciel
- Faire un git clone du dépôt

- Fichier .tar.gz: `tar xvfz fichier.tar.gz`
- Fichier tar: `tar -xvf fichier.tar`
- Fichier .tar.xz: `tar xf fichier.tar.xz`
- Fichier tar.bz2: `tar xjf fichier.tar.bz2`
- Fichier .zip: `unzip fichier.zip`

- Suivre les instructions d'installation: README ou INSTALL

```
perl Makefile.PL  
PREFIX=~/.lib/perl5
```

Détecte les infos systèmes et
configure le code source pour s'y
adapter

- Suivre les instructions d'installation: README ou INSTALL

```
perl Makefile.PL  
PREFIX=~/.lib/perl5
```

Détecte les infos systèmes et configure le code source pour s'y adapter

```
make  
make test
```

Réalise la compilation du programme

make install

Copie les binaires (exécutables)
produits à l'endroit spécifié dans
le prefix

make install

Copie les binaires (exécutables)
produits à l'endroit spécifié dans
le prefix

```
echo 'export  
PERL5LIB=~/.lib/perl5/site_perl'  
>> ~/.bashrc  
source ~/.bashrc
```

Modifier son ~/.bashrc pour
pouvoir utiliser ses propres
librairies perl

```
cpan -i module_perl
```

Installe le *module_perl*

```
perl -CPAN -e 'install  
module_perl'
```

Installe le *module_perl*

INSTALLER DES PACKAGES PYTHON

- Téléchargement direct depuis le site
- Utiliser wget + lien vers le logiciel
- Faire un git clone du repository

- Fichier .tar.gz: `tar xvfz fichier.tar.gz`
- Fichier tar: `tar -xvf fichier.tar`
- Fichier .tar.xz: `tar xf fichier.tar.xz`
- Fichier tar.bz2: `tar xjf fichier.tar.bz2`
- Fichier .zip: `unzip fichier.zip`

```
python setup.py install --user
```

Copie les binaires (exécutables)
produits à l'endroit spécifié dans
le prefix

```
python setup.py install --user
```

Copie les binaires (exécutables)
dans le répertoire
~/local/lib/pythonX.X/site-packa
ges/

```
echo 'export  
PYTHONPATH=$HOME/.local/lib/  
pythonX.X/site-packages:$PYTH  
ONPATH' >> ~/.bashrc  
source ~/.bashrc
```

Modifier son ~/.bashrc pour
pouvoir utiliser ses propres
packages python

```
python -m pip install package_python
```

Installe le *package_python* dans le
répertoire
`~/.local/lib/pythonX.X/site-packages/`

INSTALLER DES LIBRAIRIES R

- Créer le répertoire Rlibs
- Créer un fichier ~/.Renviron avec :

R_LIBS=/path/to/Rlibs

- Téléchargement direct depuis le site
- Utiliser wget + lien vers le logiciel
- Faire un git clone du repository

- Fichier .tar.gz: `tar xvfz fichier.tar.gz`
- Fichier tar: `tar -xvf fichier.tar`
- Fichier .tar.xz: `tar xf fichier.tar.xz`
- Fichier tar.bz2: `tar xjf fichier.tar.bz2`
- Fichier .zip: `unzip fichier.zip`

R CMD INSTALL --library=/path/to/Rlibs packageR.tar.gz

Copie les librairies produites à l'endroit spécifié dans le prefix

R

install.packages("nom_package")

Installe le package R spécifié dans
/home/user/R

INSTALLER DES SOFTS AVEC CONDA

- Package manager écrit en python
- Permet la création d'environnement virtuel pour éviter d'éventuel problème de dépendance
- Permet d'installer des logiciels bioinfos depuis le dépôt Bioconda (+7000 outils référencés)
- - <https://anaconda.org/bioconda/repo>

bioconda / packages

Packages

Files














Install Instructions

Filters

Type: all

Access: all

Label: all

Package Name	Access	Summary	Updated
 visor	public	Haplotype-aware structural variants simulator for short, long and linked reads	2022-05-10
 python-chado	public	A Python library for interacting with Chado database.	2022-05-10
 checkv	public	Assess the quality of metagenome-assembled viral genomes.	2022-05-10
 extract_vcf	public	Tool to extract information from vcf file.	2022-05-10
 mimseq	public	Modification-induced misincorporation tRNA sequencing.	2022-05-10
 irma	public	IRMA: Iterative Refinement Meta-Assembler for the robust assembly, variant calling, and phasing of highly variable RNA viruses.	2022-05-10
 agc	public	Assembled Genomes Compressor (AGC) is a tool designed to compress collections of de-novo assembled genomes. It can be used for various types of datasets: short genomes (viruses) as well as long (humans).	2022-05-10
 macrel	public	A pipeline for AMP (antimicrobial peptide) prediction	2022-05-09
 10x_bamtofastq	public	Tool for converting 10x BAMs produced by Cell Ranger, Space Ranger, Cell Ranger ATAC, Cell Ranger DNA, and Long Ranger back to FASTQ files that can be used as inputs to re-run analysis	2022-05-09
 regenie	public	Regenie is a C++ program for whole genome regression modelling of large genome-wide association studies (GWAS).	2022-05-09
 nanoq	public	Ultra-fast quality control and summary reports for nanopore reads	2022-05-09
 r-grain	public	Probability propagation in graphical independence networks, also known as Bayesian networks or probabilistic expert systems.	2022-05-09
 magphi	public	A bioinformatics tool allowing for examination and extraction of genomic features using seed sequences.	2022-05-09

Création d'un environnement

```
(base) [formateur2@master0 ~]$ conda create --name formation
```

Activation de l'environnement

```
(base) [formateur2@master0 ~]$ conda activate formation
```

Installation de logiciel dans cet environnement virtuel

```
(formation) [formateur2@master0 ~]$ conda install samtools bwa  
(formation) [formateur2@master0 ~]$ which bwa  
~/miniconda3/envs/formation/bin/bwa
```

Exporter et partager un environnement

```
(formation) [formateur2@master0 ~]$ conda env export > formation.yaml
```

```
(base) [drocg@muse-login01 cond]$ conda env create -n  
new_environnement -f formation.yaml  
(base) [drocg@muse-login01 cond]$ conda activate new_environnement  
(new_environnement) [drocg@muse-login01 cond]$ which bwa  
~/miniconda3/envs/new_environnement/bin/bwa
```


Lister les environnements

```
(formation) [formateur2@master0 ~]$ conda env list
# conda environments:
#
base                        /home/formateur2/miniconda3
formation                  *  /home/formateur2/miniconda3/envs/formation
```

Désactivation de l'environnement

```
(formation) [formateur2@master0 ~]$ conda deactivate
```

Supprimer un environnement

```
(formation) [formateur2@master0 ~]$ conda env remove formation
```




Practice

Installation de logiciels

1

Aller sur le [Practice 1](#) du github

MODULE ENVIRONMENT

- Permet de choisir la version du logiciel que l'on veut utiliser
- 2 types de logiciels :
 - bioinfo : désigne les logiciels de bioinformatique (exemple BEAST)
 - system : désigne tous les logiciels systèmes (exemple JAVA)
- Surpassent les variables d'environnement

➤ 5 types de commandes :

- Voir les modules disponibles :

`module avail`

- Obtenir une info sur un module en particulier :

`module whatis + module name`

- Charger un module :

`module load + modulename`

- Lister les modules chargés :

`module list`

- Décharger un module :

`module unload + modulename`

- Décharger tous les modules :

`Module purge`

- Fichier tcl permettant de gérer ses logiciels
- Permet de charger ses propres logiciels installés
- Permet de choisir les versions de ses logiciels
- Plus de modifications du bashrc

```
##%Module1.0#####  
##  
## modules modulefile  
##  
## modulefiles/modules. Generated from modules.in by configure.  
##  
proc ModulesHelp { } {  
    global version modroot  
  
    puts stderr "blast/2.4.0+ version 2.4.0 de blast"  
}  
  
module-whatis "charge la version 2.4.0 de blast.  
URL: https://blast.ncbi.nlm.nih.gov/Blast.cgi?PAGE\_TYPE=BlastDocs&DOC\_TYPE=Download  
Description:BLAST finds regions of similarity between biological sequences  
"  
  
conflict blast  
  
# for Tcl script use only  
set  version      2.4.0+  
set  topdir        /usr/local/ncbi-blast-$version  
  
prepend-path PATH      $topdir/bin  
prepend-path MANPATH    $topdir/man
```

```
proc ModulesHelp { } {  
    global version modroot  
  
    puts stderr "blast/2.4.0+ version 2.4.0 de blast"  
}
```

Permet de préciser la sortie de module help

module-whatism "charge la version 2.4.0 de blast.
URL: https://blast.ncbi.nlm.nih.gov/Blast.cgi?PAGE_TYPE=BlastDocs&DOC_TYPE=Download
Description:BLAST finds regions of similarity between biological sequences
"

Permet de préciser la sortie de module whatism

conflict blast

Empêche le chargement du module si celui-ci est déjà monté

```
module load bioinfo/softs/version
```

Permet de charger les modules des dépendances

```
# for Tcl script use only
set  version      2.4.0+
set  topdir        /usr/local/ncbi-blast-$version

prepend-path PATH    $topdir/bin
prepend-path MANPATH $topdir/man
```

- Avec set : définition de variable pour le script Tcl
- Prepend-path: positionne les variables d'environnement (remplace la ligne du .bashrc)

Activation de l'utilisation de ses modulefiles

- Créer un répertoire ~/privatemodules

mkdir ~/privatemodules

- Placer son modulefile à l'intérieur
- Modifier son ~/.bashrc pour utiliser ses modulefiles avec:

module use --append \$HOME/privatemodules

- Re-sourcer son ~/.bashrc

source ~/.bashrc



Practice

Module environment

2

Aller sur le [Practice2](#) du github

SLURM

- Liste des partitions :
`scontrol show partition`
- Liste des noeuds (num cpu/memory)
`sinfo -Ne --format "%.15N %.4c %.7z %.7m" -S c,m,N | uniq`
- Vérifier si job actif
`squeue -u <USER>`
- Supprimer un job
`scancel <JOB_ID>`
- Connaître le noeud sur lequel un job terminé a été lancé
`sacct -j JOB_ID --format=JobID,Start,End,Elapsed,NCPUS,NodeList,NTasks`

LES JOBS ARRAY

- Façon simple de mettre en oeuvre la parallélisation par les données
- Pour lancer un ensemble de calculs “identiques” (sur différents fichiers d’entrée) à partir d’un seul script de soumission

- `$SBATCH --array` pour lancer un job array
 - `--array=0-X` : pour définir la plage (Tableau d'index de 0 à X)
 - `--array=0-X%Y` : pour définir la plage et avec Y running en même temps
- Variables d'environnement:
 - `${SLURM_JOB_ID}`: précise le job ID
 - `${SLURM_ARRAY_JOB_ID}`: précise l'ID du job array
 - `{SLURM_ARRAY_TASK_ID}`: précise le nombre de tâches job array

```
#!/bin/bash
#SBATCH --partition=short ### Partition
#SBATCH --job-name=ArrayJob ### Nom du job
#SBATCH --time=00:10:00 ### temps limite d'execution
#SBATCH --nodes=1 ### Nombre de noeuds
#SBATCH --ntasks=1 ### Nombre de tâches par job array
#SBATCH --array=0-19%4 ### Tableau d'index de 0 à 19 avec 4 jobs lancés à la fois

echo "I am Slurm job ${SLURM_JOB_ID}, array job ${SLURM_ARRAY_JOB_ID}, and array task
${SLURM_ARRAY_TASK_ID}."
```

- sortie : `job_name.o$JOB_ID.$SLURM_JOB_ID`
- erreur : `job_name.e$JOB_ID.$SLURM_JOB_ID`

- Pour améliorer la lisibilité des scripts
- Pour faciliter la gestion des jobs :
 - suppression du job array : *scancel <JOBID>*
 - suppression d'une sous-tache : *scancel <JOBID>.<TASKID>*
- Pour limiter la charge du master

CONTRÔLE SLURM

scontrol hold <JOB_ID> : empêche un job (en queue) d'être lancé

scontrol release <JOB_ID> : repositionne le job hold dans la queue

scontrol suspend <JOB_ID>: suspend un job actif

scontrol resume <JOB_ID> : relance un job suspendu

scontrol requeue <JOB_ID>: repositionne le job dans la queue

scontrol update JobID=\$jobid StartTime=MMDDYY: reprogramme un job mis en queue à une autre date

sbatch --dependency [Options]: dépendance des jobs

--dependency=*after*:<JOB_ID>: démarre le job dès que <JOB_ID> commence

--dependency=*afterany*:<JOB_ID>: démarre le job dès que <JOB_ID> est terminé

--dependency=*afternotok*:<JOB_ID> : démarre le job dès que <JOB_ID> est terminé et en erreur

--*dependency=afterok*:<JOB_ID> : démarre le job dès que <JOB_ID> est terminé avec succès

--dependency=singleton: un seul job du même propriétaire te du même nom peut être lancé à la fois

- Slurm home page :
<https://www.schedmd.com/index.php>
- Slurm documentation :
<https://slurm.schedmd.com/documentation.html> Slurm
- cheat sheet :
<https://slurm.schedmd.com/pdfs/summary.pdf>



Practice

Jobs array

3

Aller sur le [Practice3](#) du github

SINGULARITY

- Au quotidien :
 - Compilation et installation
 - Mise à jour des logiciels
 - Utilisation de modules pour gérer l'utilisation des logiciels
- Problèmes :
 - Compilations complexes avec de nombreuses dépendances
 - Reproductibilité des compilations (versions dépendances et logiciels)
 - Compilation de nouveaux logiciels sur vieilles distributions

Une solution : les conteneurs d'applications



Docker

<https://www.docker.com/>



Shifter

<https://github.com/NERSC/shifter>



Singularity

<http://singularity.lbl.gov/>

- Les + :
 - Communauté
 - Dépôt central très riche
- Les - :
 - Accès root dans le conteneur
 - Forte isolation, pas d'accès à Infiniband et aux partages réseaux
 - Pas d'accès à l'affichage (donc pas de GPU)

Inadapté au HPC

- Les + :
 - Accès au dépôt central de docker pour la création d'image
 - Accès Infiniband, aux partages réseaux et GPU
 - Pas d'accès root
 - Utilisable avec les modules et SGE comme un logiciel classique
- Les - :
 - Quelques bugs pour certaines images Docker
 - Intégration MPI nécessite OpenMPI 2

Adapté au HPC

- Développé au laboratoire Lawrence Berkeley par le créateur de CentOS pour garantir :
 - **Portabilité** entre environnements Linux
 - **Reproductibilité**
 - **Mobilité** entre clusters
- Fonctionnalités :
 - Encapsulation de l'environnement utilisateur
 - Conteneur à base d'image
 - Droits utilisateurs identiques dans et hors conteneur
 - Montage automatique du répertoire utilisateur et des partages réseaux



Practice

Installation de Singularity

4

Aller sur le [Practice4](#) du github

Root / Superuser

- Création du container
- Build/install du container
- Modifications système du container



Regular User

- singularity shell
- singularity exec ...
- singularity run ...

- shell : lance un shell au sein du conteneur

```
$ singularity shell ubuntu.img  
Singularity: Invoking an interactive shell within container...  
  
Singularity.ubuntu.img>
```

- exec : exécute une commande au sein du conteneur

```
$ singularity exec ubuntu.img python  
Python 2.7.12 (default, Jul 1 2016, 15:12:24)  
>>>
```

- run : lance un runsript au sein du conteneur

```
$ singularity run ubuntu.img  
This is what happens when you run the container...  
$
```

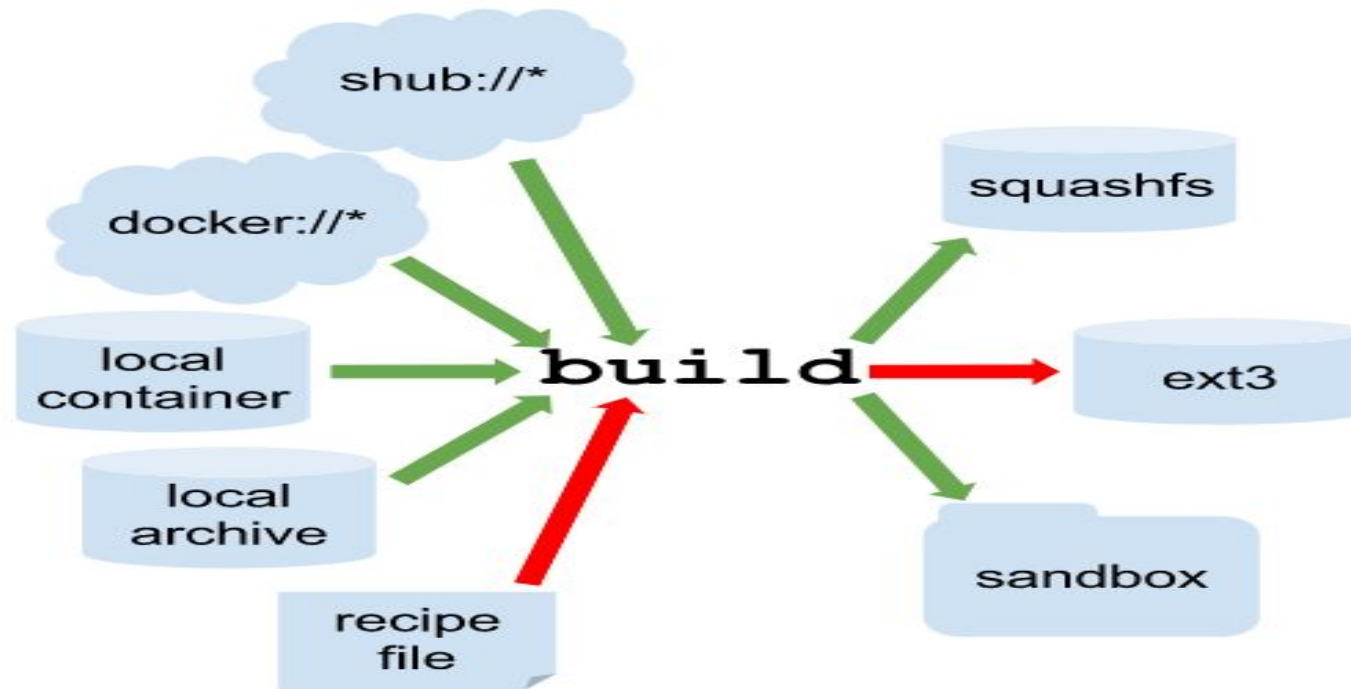
- Les commandes *singularity exec* et *singularity run* permettent d'exécuter des commandes
- *singularity run* lancera la commande par défaut définie dans le conteneur sans avoir à la préciser

ex: *singularity run bwa-0.7.17.simg + arg* lancera *bwa + arg*

- *singularity exec* lancera la commande précisée en arguments après *singularity exec <conteneur.simg>*

ex: *singularity exec blast-2.10.0+.simg blastn + arg* lancera *blastn + arg*

Création conteneur Singularity



<https://github.com/SouthGreenPlatform/singularityRecipeFiles>

Bootstrap : définit le type de base de départ du conteneur

- shub (Singularity Hub)
- docker (Docker Hub)
- localimage
- yum (CentOS or Scientific Linux)
- debootstrap (Debian or Ubuntu)
- arch (Arch Linux)
- busybox
- zypper

From : définit la base de départ du conteneur.

Sections pour exécution de commandes :

- **%setup** : sur la machine hôte, hors du conteneur, après l'installation de l'OS
- **%post** : dans le conteneur après l'installation de l'OS
- **%runscript** : à chaque run du conteneur
- **%test** : à la fin de l'install de l'OS

Recipe container Singularity : exemple avec exec /bin/bash

```
BootStrap: docker
From: ubuntu:18.04
%labels
Maintainer Ndomassi Tando - IRD Itrop Cluster, DIADE Unit
base.image="ubuntu:18.04"
version="1"
software="ncbi-blast"
software.version="2.10.0+"
%help
URL: https://blast.ncbi.nlm.nih.gov/Blast.cgi?PAGE_TYPE=BlastDocs&DOC_TYPE=Download
Description: This allows users to perform BLAST searches on their own server without size, volume and database restrictions. BLAST+ can
be used with a command line so it can be integrated directly into your workflow
%environment
export PATH=$PATH:/usr/local/ncbi-blast-2.10.0+/bin
export LC_ALL=C
%post
apt update
apt install -y build-essential wget unzip python3 python-dev perl tar libidn11 libidn11-dev

cd /usr/local/
wget ftp://ftp.ncbi.nlm.nih.gov/blast/executables/blast+/LATEST/ncbi-blast-2.10.0+-x64-linux.tar.gz
tar xvfz ncbi-blast-2.10.0+-x64-linux.tar.gz

%runscript
exec /bin/bash "$@"
```

Recipe container Singularity : exemple avec exec /bin/bash

```
BootStrap: docker
From: ubuntu:18.04
%labels
Maintainer Ndomassi Tando - IRD Itrop Cluster, DIADE Unit
base.image="ubuntu:18.04"
version="1"
software="ncbi-blast"
software.version="2.10.0+"
%help
URL: https://blast.ncbi.nlm.nih.gov/Blast.cgi?PAGE_TYPE=BlastDocs&DOC_TYPE=Download
Description: This allows users to perform BLAST searches on their own server without size, volume and database restrictions. BLAST+ can
be used with a command line so it can be integrated directly into your workflow
%environment
export PATH=$PATH:/usr/local/ncbi-blast-2.10.0+/bin
export LC_ALL=C
%post
apt update
apt install -y build-essential wget unzip python3 python-dev perl tar libidn11 libidn11-dev

cd /usr/local/
wget ftp://ftp.ncbi.nlm.nih.gov/blast/executables/blast+/LATEST/ncbi-blast-2.10.0+-x64-linux.tar.gz
tar xvfz ncbi-blast-2.10.0+-x64-linux.tar.gz
```

```
%runscript
exec /bin/bash "$@"
```

**On utilise exec /bin/bash car il y a
plusieurs commandes dans la
suite logiciel**

singularity exec

Recipe container Singularity : exemple avec exec+ command pour singularity run

```
BootStrap: docker
From: ubuntu:18.04
%labels
Maintainer Ndomassi Tando - IRD Itrop Cluster, DIADE Unit
base.image="ubuntu:18.04"
version="1"
software="bamtools"
software.version="2.5.1"
%help
URL: https://github.com/pezmaster31/bamtools
Description: BamTools provides both a programmer's API and an end-user's toolkit for handling BAM files.
Launch the command: singularity run /PATH_TO_CONTAINER/bamtools-2.5.1.simg + arguments
%environment
export PATH=$PATH:/usr/local/bamtools-2.5.1/bin
export LC_ALL=C

%post
apt update
apt install -y build-essential wget zlib1g-dev libncurses5-dev libboost-iostreams-dev zlib1g-dev libgsl-dev libboost-graph-dev
libsuitesparse-dev liblapack-dev libsqlite3-dev libmysql++-dev libbamtools-dev libboost-all-dev libbz2-dev liblzma-dev libncurses5-dev
libssl-dev libcurl3-dev cmake

cd /usr/local/
wget https://github.com/pezmaster31/bamtools/archive/refs/tags/v2.5.1.tar.gz
ttar xvfz v2.5.1.tar.gz
cd bamtools-2.5.1
mkdir build
cd build
cmake -DCMAKE_INSTALL_PREFIX=/usr/local/bamtools-2.5.1 ..
make
make install
```

```
%runscript
exec bamtools "$@"
```

**On utilise exec bamtools pour ne
lancer que bamtools**

singularity run

Interactive Development

```
sudo singularity build --sandbox tmpdir/ Singularity
```

```
sudo singularity build --writable container.img Singularity
```

BUILD ENVIRONMENT

Build from Recipe

```
sudo singularity build container.img Singularity
```

Build from Singularity

```
sudo singularity build container.img shub://vsoch/hello-world
```

Build from Docker

```
sudo singularity build container.img docker://ubuntu
```



Practice

Créer son conteneur
Singularity

5

Aller sur le [Practice5](#) du github

Module pour container Singularity : exemple

```
##%Module1.0#####  
##  
##  
proc ModulesHelp { } {  
    global name version prefix man_path  
  
}  
  
module-whatis "loads the [module-info name] environment"  
  
set    version    "2.3.7"  
  
conflict bioinfo/guppy  
  
set prefix /usr/local/bioinfo/guppy/2.3.7  
  
if {[file exists $prefix]} {  
    puts stderr "\t[module-info name] Load Error: $prefix does not exist"  
    break  
    exit 1  
}  
  
module load system/singularity/3.6.0  
  
set topdir /usr/local/singularity-3.6.0/  
prepend-path PATH $topdir/container/wrappers/guppy-2.3.7
```

Construction d'un module pour container singularity

- Un conteneur peut contenir un ou plusieurs exécutables ceux-ci doivent être accessible dans le path de la machine hôte
- Etapes:
 - Créer un répertoire *wrappers/<nom_logiciel>-<version>*
 - Pour chaque exécutable du conteneur, créer un fichier du même nom
 - Le remplir avec :
 - *singularity exec <nom_conteneur>.simg <commande> \$@*
 - les rendre executable avec: *chmod + x <nom_commande>*

Exemple pour guppy 2.3.7:

- on créera un répertoire guppy-2.3.7 contenant les fichiers guppy_basecaller, guppy_basecaller_server, guppy_basecaller_1d2, guppy_barcode et guppy_aligner
- Par exemple pour le fichier guppy_basecaller on mettra:

```
singularity exec <path_to_container>/<container>.img guppy_basecaller $@
```



Practice

Lancer un job avec son
conteneur Singularity

5

Aller sur le [Practice 6](#) du github

- **Sebastien Ravel**
- **Ndomassi Tando**



- **Bruno Granouillac**



- **Valérie Noël**
- **Bertrand Pitollat**
- **Guilhem Sempere**



- Gaetan Droc
- Bruno Granouillac
- Nicolas Fernandez
- **Bertrand Pitollat**
- Sebastien Ravel
- Guilhem Sempere
- **Ndomassi Tando**



Merci pour votre attention !



Le matériel pédagogique utilisé pour ces enseignements est mis à disposition selon les termes de la licence Creative Commons Attribution - Pas d'Utilisation Commerciale - Partage dans les Mêmes Conditions (BY-NC-SA) 4.0 International:

<http://creativecommons.org/licenses/by-nc-sa/4.0/>

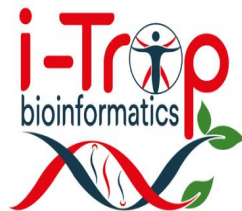
**Merci de prendre 5 min pour remplir
l'enquête**

<https://itrop-survey.ird.fr/index.php/515725?lang=fr>

SUIVEZ NOUS SUR TWITTER !



South Green :
[@green_bioinfo](#)



I-Trop : [@ltropBioinfo](#)

N'oubliez pas de nous citer !

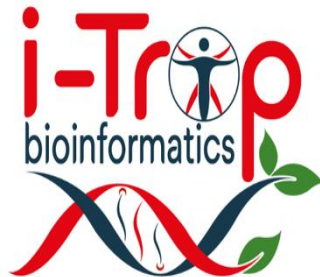
Comment citer les clusters?

"The authors acknowledge the IRD i-Trop HPC at IRD Montpellier for providing HPC resources that have contributed to the research results reported within this paper. URL: <http://bioinfo.ird.fr/> "

"The authors acknowledge the CIRAD UMR-AGAP HPC (South Green Platform) at CIRAD montpellier for providing HPC resources that have contributed to the research results reported within this paper. URL: <http://www.southgreen.fr>"



South Green : [@green_bioinfo](https://twitter.com/green_bioinfo)



I-Trop : [@ltropBioinfo](https://twitter.com/ltropBioinfo)