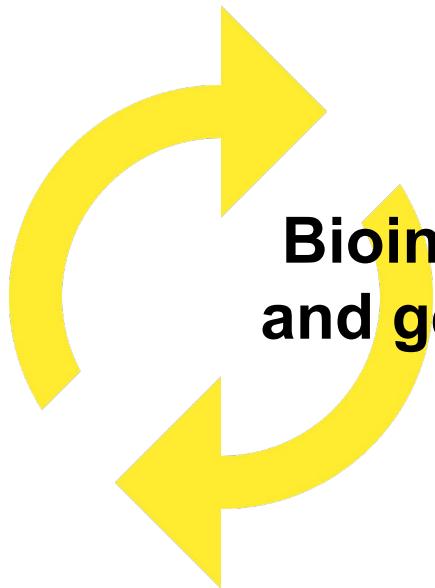




# Session de formation 2019





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

genome assembly SNP detection  
phylogeny structural variation  
comparative genomics transcriptome assembly differential expression  
GWAS pangenomics  
population genetics metagenomics  
polypliody



Rice



Banana



Palm



Sorghum



Coffee



Cassava



Magnaporthe

# South Green

bioinformatics platform



Larmande Pierre  
Sabot François  
Tando Ndomassi  
**Tranchant-Dubreuil  
Christine**



Comte Aurore  
Dereeper Alexis



**Orjuela-Bouniol Julie**



Bocs Stephanie  
De Lamotte Fredéric  
**Droc Gaetan**  
Dufayard Jean-François  
Hamelin Chantal  
Martin Guillaume  
Pitollat Bertrand  
**Ruiz Manuel**  
**Sarah Gautier**  
Summo Marilyne



**Rouard Mathieu**  
Guignon Valentin  
Catherine Breton



**Mahé Frédéric**  
**Ravel Sébastien**



Sempere Guilhem



# South Green bioinformatics platform

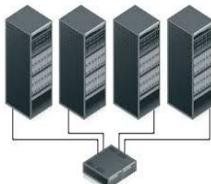
## Workflow manager

TOGGLE  
Toolbox for generic NGS analyses

SNAKEMAKE

Galaxy

## HPC and trainings....



IRD  
Institut de Recherche pour le Développement

cirad



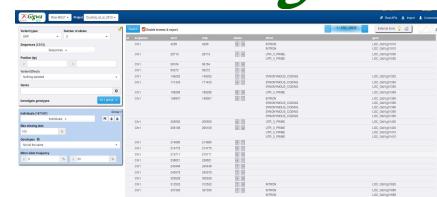
## Genome Hubs & Information System



GreenPhyl

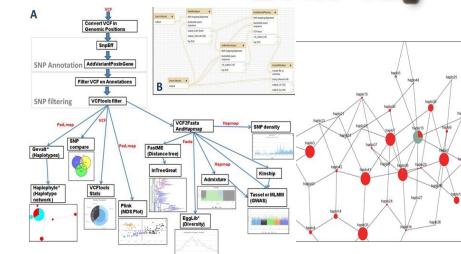
| Family Id | Family Name   | Number of sequences | Status |
|-----------|---|---------------------|--------|
| GP000010  | Cytochrome P450 superfamily                                     | 6942                | green  |
| GP000017  | AP2/EREBP transcription factor family: ERF/DREB group (partial) | 5142                | green  |
| GP000020  | NAC transcription factor family                                 | 4574                | green  |
| GP000018  | MAOS transcription factor family                                |                     |        |
| GP000019  | Haem peroxidase superfamily                                     |                     |        |
| GP000022  | General substrate transporter superfamily                       |                     |        |
| GP000019  | NPF, NRT1/PTR FAMILY  |                     |        |

Gene families



SNPs and Indels

SNiPlay



<https://github.com/SouthGreenPlatform>



@green\_bioinfo

*The South Green portal: a comprehensive resource for tropical and Mediterranean crop genomics*, Current Plant Biology, 2016



Erwan Corre



Marie Simonin  
Sébastien Cunnac



Etienne Loire  
Julie Reveillaud



Florentin Constancias



Valentin Klein



Valérie Noël



Emmanuelle Beyne



**And more collaborators !**

|          |   |  |
|----------|---|--|
|          |   |  |
| 18-19/03 | ● | Guide de survie à Linux - IRD                              |
| 21/03    | ● | Initiation à l'utilisation du cluster CIRAD – CIRAD        |
| 22/03    | ● | Initiation à l'utilisation du cluster itrop - IRD          |
| 15-16/04 | ● | Initiation au gestionnaires de workflow SG & Gigwa – IRD   |
| 18-19/04 | ● | Guide du Jedi en Linux & bash - CIRAD                      |
| 13-16/05 | ● | Python - IRD   |
| 17/05    | ● | Initiation aux analyses de données transcriptomiques – IRD |
| 21/05    | ● | Utilisation avancée du cluster IRD – IRD                   |
| 23-24/05 | ● | Initiation aux analyses de données métagénomiques – IRD    |
| 6/06     | ● | Manipulation de données et figures sous R – CIRAD          |
| 25-27/09 | ● | Assemblage et annotation de transcriptomes - IRD           |



# Modules de formation 2019

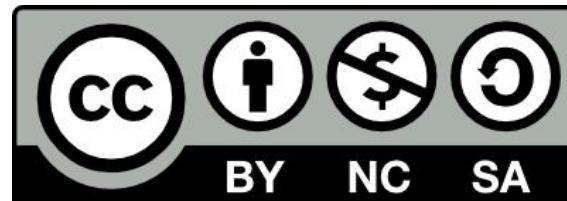
- Toutes nos formations :  
<https://southgreenplatform.github.io/trainings/>
- Topo & TP : Cluster avancé
- Environnement de travail : Logiciels à installer
- How-tos : How-to



# Cluster avancé

[www.southgreen.fr](http://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

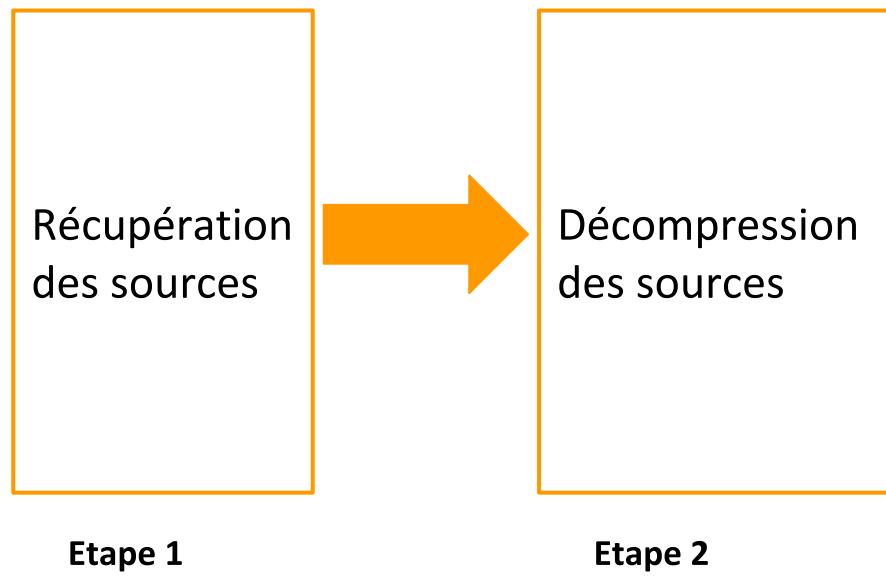
# INSTALLER DES LOGICIELS

# 4 grandes étapes

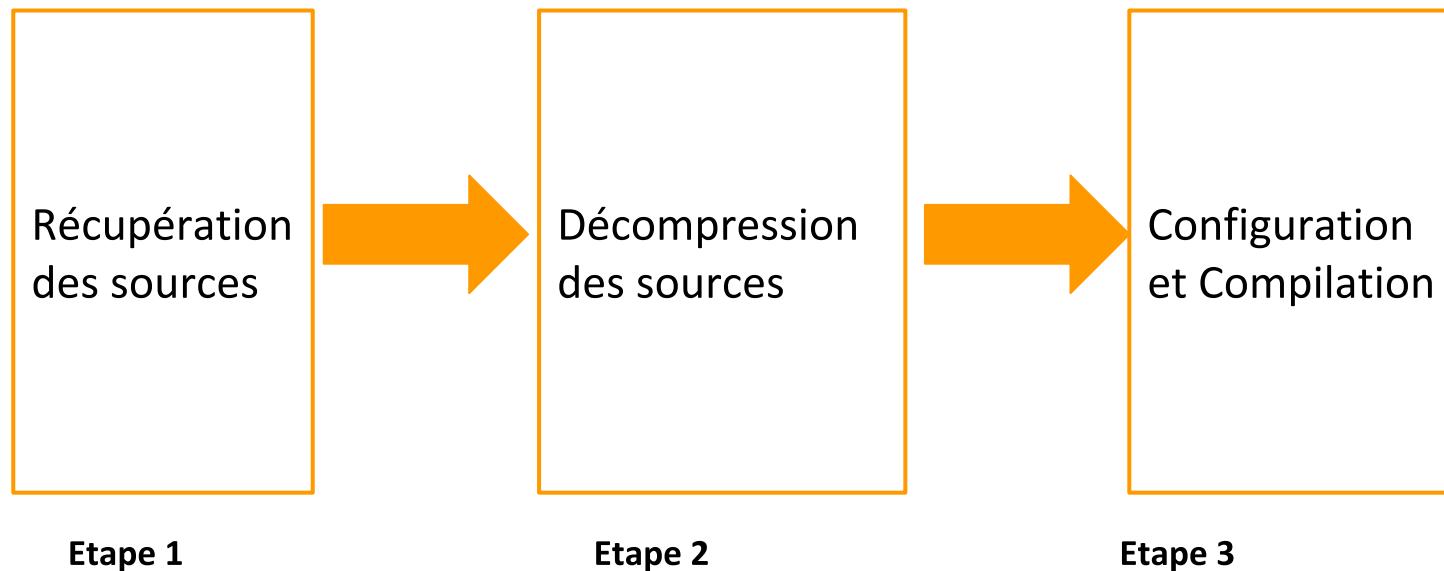
Récupération  
des sources

**Etape 1**

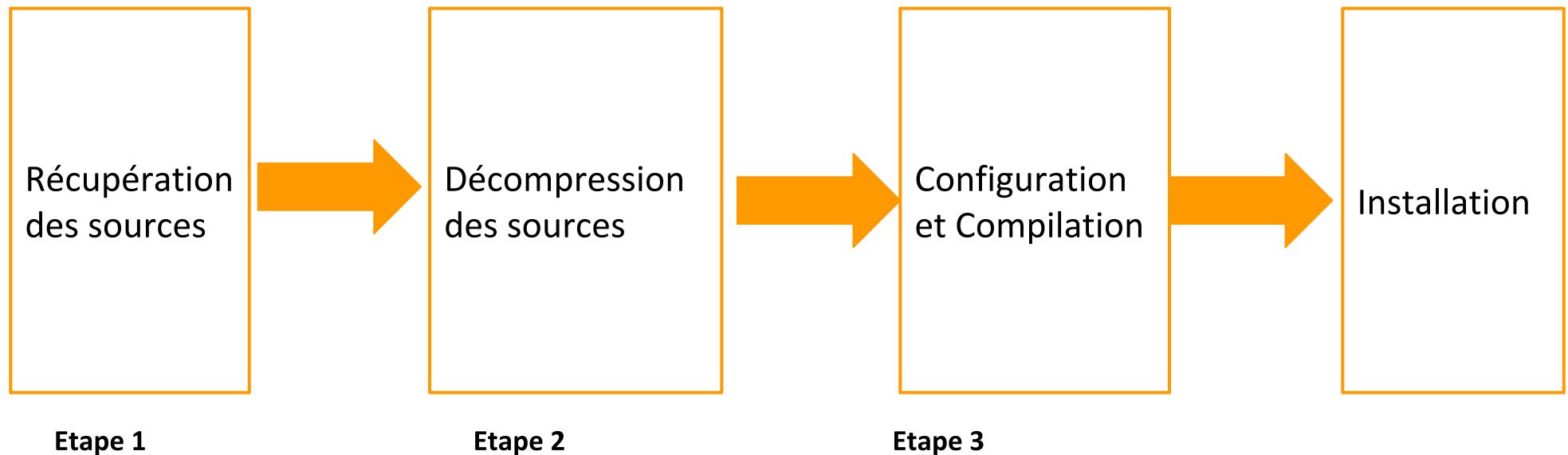
# 4 grandes étapes



# 4 grandes étapes



# 4 grandes étapes



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

# Récupération des sources

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

# Décompression des sources

- Fichier .tar.gz: tar xvzf 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-ver  
sion
```

Déetecte 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
```

Déetecte 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**

# Récupération des sources

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

# Décompression des sources

- Fichier .tar.gz: tar xvzf 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
```

Déetecte 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
```

Déetecte 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

# Autres installations possibles

*cpan -i module\_perl*

Installe le *module\_perl*

*perl -CPAN -e 'install  
module\_perl'*

Installe le *module\_perl*

# INSTALLER DES PACKAGES PYTHON

# Récupération des sources

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

# Décompression des sources

- Fichier .tar.gz: tar xvzf 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-pac  
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

# Autre installation possible

*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*

# Récupération des sources

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

# Décompression des sources

- Fichier .tar.gz: tar xvzf 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



# 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`

# Créer ses propres modulefiles

- 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

# Exemple de modulefile

```
##%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
```

# Partie ModulesHelp

```
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

## Partie module-whatis

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](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 whatis

# Utilisation de conflict

conflict blast

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

# Utilisation de module load

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

Permet de charger les modules des dépendances

# Définition de variable et du PATH

```
# 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

# LES JOBS ARRAY

# Data parallelism

- 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

# Job array

- qsub -t
- -t n-m:s
  - lance un job array de l'index n (***SGE\_TASK\_FIRST***) jusqu'à l'index m (***SGE\_TASK\_LAST***) par pas de s (***SGE\_TASK\_STEP***)
    - l'index de la sous-tache est enregistrée dans la variable ***SGE\_TASK\_ID*** de l'environnement du job
- -tc max\_running\_tasks
  - pour limiter le nombre de job simultané

!!! -t n-m:X différent de -tc X

# Exemple

```
#!/bin/bash

# SGE PARAMETRES
#$ -N array
#$ -q long.q
#$ -S /bin/bash
#$ -cwd
#$ -V
#$ -t 1-10000:10

# JOB BEGIN
# Loop through the IDs covered by this stride and run the application if # the input file exists.

for (( i=$SGE_TASK_ID; i<$SGE_TASK_ID+10; i++ )) do
    if [ -f "input.$i" ]
    then echo "$JOB_NAME $SGE_TASK_ID input.$i"
    fi
done
# JOB END

exit 0
```

# Fichiers d'erreur et de sortie

- sortie : job\_name.o\$JOB\_ID.\$SGE\_TASK\_ID
- erreur : job\_name.e\$JOB\_ID.\$SGE\_TASK\_ID

# Job array vs jobs simples

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

# Contrôle SGE

# Contrôle de jobs

-h : rendre le job inéligible (annuler avec qalter –h U)  
-hold\_jid <liste\_jobs> : dépendance des jobs

-terse : afficher le JOB ID  
Pour simplifier le workflow des jobs

-sync y : attendre la fin du job  
On peut réaliser des dépendances des jobs

-now y : essayer d'exécuter immédiatement  
Si manque de ressources => pas d'exécution

-r y : redémarrer le job en cas de crash de la machine

# Workflow de jobs

- Dépendances de jobs:  
*qsub -hold\_jid <JOB\_ID>*
- Synchronisation avec la fin d'un job:  
*qsub -sync yes*
- Soumission du job mais non éligible à l'exécution  
*qsub -h*
- On permet au job de s'exécuter (éligible), éventuellement à travers l'épilogue d'un autre job  
*qalter -h -U <JOB\_ID>*
- Inconvénient: La gestion des codes de retour des jobs



# Practice

Jobs array

3

Aller sur le [Practice3](#) du github

# SINGULARITY

# Gestion logicielle sur un cluster HPC

- 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

# Singularity

- 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

## Root / Superuser

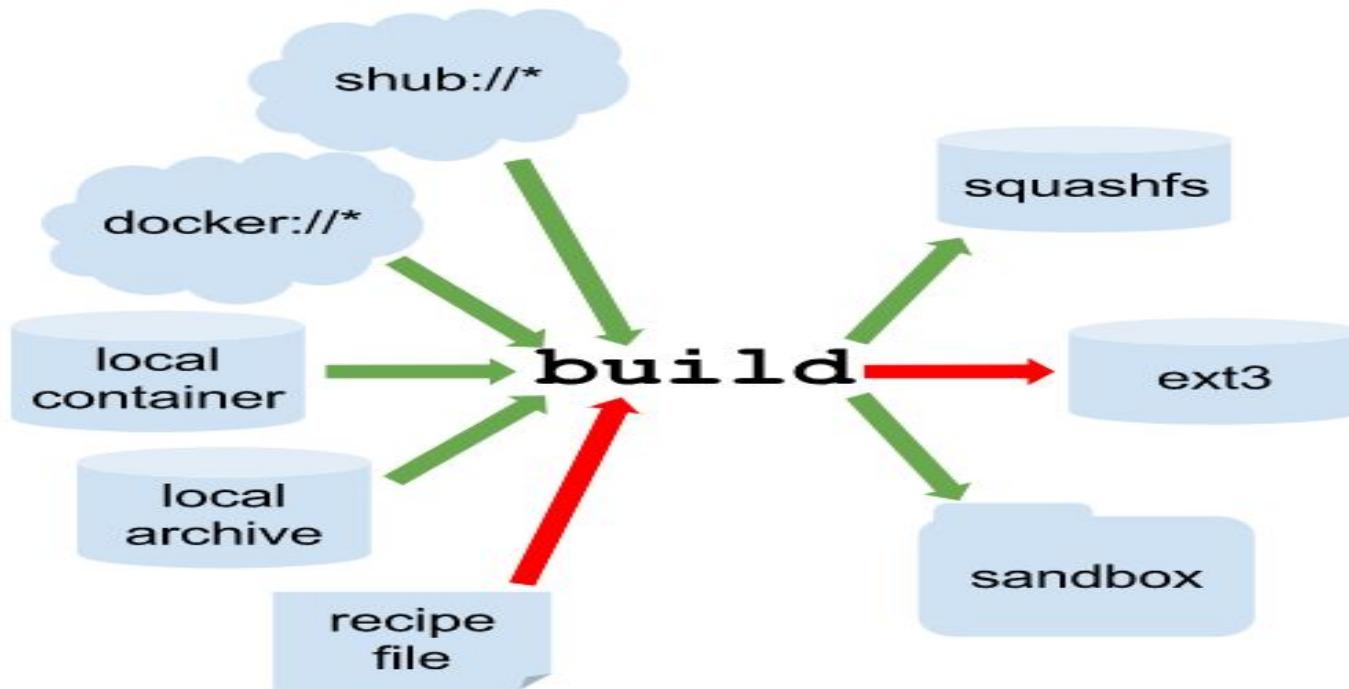
- Création du container
- Build/install du container
- Modifications système du container



## Regular User

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

# Création conteneur Singularity



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

# Recipe conteneur Singularity

**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

```
BootStrap: docker
From: ubuntu:14.04
```

```
%labels
Maintainer Sebastien RAVEL
base.image="ubuntu:14.04"
version="2.3.7"
software="guppy"
software.version="2.3.7"
```

```
%post
apt-get -y update && apt-get -y install build-essential wget
```

```
# Pour cluster cirad
mkdir /work
mkdir /homedir
mkdir /projects
mkdir -p /usr1/compte_mess
# Fin pour cluster cirad
```

```
mkdir -p /opt/sources
cd /opt/sources
wget https://mirror.oxfordnanoportal.com/software/analysis/ont-guppy-cpu_2.3.7_linux64.tar.gz
tar -zxf ont-guppy-cpu_2.3.7_linux64.tar.gz
```

```
%runscript
exec /opt/sources/ont-guppy-cpu/bin/"$@"
```

# Génération du conteneur

## 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
```

# Lancement du conteneur

- 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 runscript au sein du conteneur

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

# 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/2.6.0
prereq system/singularity/2.6.0
set-alias guppy_basecaller "singularity exec /usr/local/bioinfo/guppy/2.3.7/guppy_2.3.7.img guppy_basecaller"
set-alias guppy_basecall_server "singularity exec /usr/local/bioinfo/guppy/2.3.7/guppy_2.3.7.img guppy_basecall_server"
set-alias guppy_basecaller_1d2 "singularity exec /usr/local/bioinfo/guppy/2.3.7/guppy_2.3.7.img guppy_basecaller_1d2"
set-alias guppy_barcoder "singularity exec /usr/local/bioinfo/guppy/2.3.7/guppy_2.3.7.img guppy_barcoder"
set-alias guppy_aligner "singularity exec /usr/local/bioinfo/guppy/2.3.7/guppy_2.3.7.img guppy_aligner"
```



# Practice

Installation de Singularity

4

Aller sur le [Practice4](#) du github



# Practice

**Créer son conteneur  
Singularity**

5

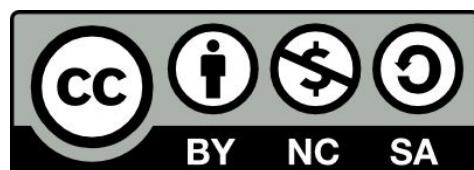
*Aller sur le [Practice5](#) du github*

# Formateurs

- **Sebastien Ravel**
- **Ndomassi Tando**
- **François Sabot**
- **Valérie Noël**
- **Bertrand Pitollat**



# Merci pour votre attention !



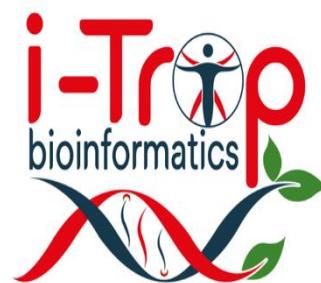
Le matériel pédagogique utilisé pour ces enseignements est mis à disposition selon les termes de la licence Creative Commons Attribution - Pas d'Utilisation Commerciale - Partage dans les Mêmes Conditions (BY-NC-SA) 4.0 International:

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

**SUIVEZ NOUS SUR TWITTER !**



**South Green :** [@green\\_bioinfo](mailto:@green_bioinfo)



I-Trop : [ItropBioinfo](http://ItropBioinfo)

## 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>"