

Cluster

La liste des logiciels disponible est sur [softwares](#), ainsi que comment compiler certaines choses sur votre home.

Nettoyage de jobs plantés

Parfois, des job annulés ou finis restent sur les noeuds: il faut les tuer à la main.

Si vous ne savez pas bien où vous en êtes, listez les noeuds sur lesquels vous avez un process qui tourne. Créer un fichier "show_process.sh" avec le contenu:

```
if [ "$1" != "" ]
then
    echo "finding processes of $1"
    for hostList in $(sinfo -h | awk '{print $6}' | sort -u); do
        for host in $(scontrol show hostname $hostList); do
            echo $host; ssh $host "ps aux | grep $1";
        done;
    done;
else
    echo "No username given"
fi
```

Puis rendez le executable: ``chmod +x show_process.sh`` et enfin lancez le avec votre username: ``show_process.sh MON_LOGIN``

====

Un fois les noeuds identifiés:

On voit, par exemple, que le job a été lancé sur ``compute-1-1``, donc on s'y connecte: ``ssh compute-1-1``

On liste les processus avec la commande ``htop``.

Avec les flèches du clavier, déplacez vous sur le processus zombie, puis faites ``F9`` pour le tuer. Un menu s'ouvre à gauche, taper directement ``entrée`` pour le sigterm, ou déplacez vous en haut sur sigkill si il est récalcitrant. Normalement, il disparaît de la liste, libérant la place.

Ensuite, pour nettoyer les fichiers, ``cd /scratch``, puis ``ls`` pour trouver le dossier correspondant, et le supprimer, par exemple ``rm -rf villani-xxxxx``.

Voilà !

Super squeue

Pour que la sortie donnée par la commande `_squeue_` soit vraiment utile, vous pouvez configurer un alias pour votre shell. Ca colore vos propre job, étend la colonne avec le nom, etc. Si vous avez bash, dans `~/bashrc`, zsh, dans `~/zshrc`, etc:

```
alias squeue='squeue -o "%.5i %.9P %.22j %.8u %.2t %.10M %.5D %R" |
GREP_COLOR="01;31" egrep --color=always "^.*$USER PD.*$|$" |
GREP_COLOR="01;32" egrep -i --color=always "^.*$USER R.*$|$"'
```

Créer un module pour votre code utilisable avec module load

Let's say you want to use your code, for instance, **flatori**, on the cluster. To do that, you need to fill a file (per version of you code) containing the correct informations about path, etc. Let's say you want a module for version R0:

```
~/privatemodules/flatori/R0
```

```
##Module1.0#####
##
## Flatori modulefile
##

proc ModulesHelp { } {
    global version modroot
    puts stderr "===== Flatori $version =====\n"
    puts stderr "Set correct environment variables and libraries \n"
}

set version      R0
set FLATORI      /export/home/$USER/path/to/flatori/

prepend-path PATH $FLATORI/where/is/the/binary
```

Then, you need to inform your shell that this module configuration file exists, in our case, add the path `~/privatemodules` to `$MODULEPATH` in your shell rc. For bash, in your `~/bashrc` , add:

```
export MODULEPATH="$MODULEPATH:/export/home/$USER/privatemodules"
```

Et voilà, now you can see your module when typing ``module avail``, and in our case, do ``module load flatori/R0``

From:

<https://portail.emse.fr/dokuwiki/> - **DOC**

Permanent link:

<https://portail.emse.fr/dokuwiki/doku.php?id=recherche:cluster&rev=1583412659>

Last update: **05/03/2020 13:50**

