

Cluster

Sur cette page, tips and tricks concernant le cluster, ainsi qu'un lien vers les logiciels spécifiques

Intro et logiciels

- Le pole [Calcul et Simulation Numérique](#) dispose d'une page [cluster](#) avec une rubrique [formation](#). Consultez là ! :)
- **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.*$|$" '
```

Jupyter lab

On peut avoir un [JupyterLab](#) fonctionnel, qui tourne proprement sur un noeud du cluster avec une réservation.

```
module load tools/cluster-bin  
cluster-create-slurm-script-01.sh -jupyter-lab
```

Ça utilise le module `anaconda/python3`.

Faites vos résas comme d'habitude et soumettez. Un message pop pour vous dire de regarder le fichier `slurm-XXX.out`. Procédez ensuite comme suit (en anglais parce voilà, comme dans le job):

You now have to connect to your notebook. Open `slurm-XXX.out`

1. Open a terminal from your desktop (NOT centaure) and create a tunnel to the compute where you notebook has been assigned, with the command below. It will have the correct values in the `slurm-XXX.out`. If you want the tunnel to stay in the background, remove use `'ssh -f ...'` Otherwise, don't forget to kill the tunnel once done.

```
ssh -L ${port}:localhost:${port} cluster ssh -L  
${port}:localhost:${port} -N ${computenode}
```

2. Now look for the connection link to paste in your browser in the bottom of the file. It looks like ``http://localhost:YYYY/?token=XXXXX``
3. Happy notebook use !
4. **VERY IMPORTANT:** when you are finished and have saved your notebook, you **must**, in jupyterlab, click on the menu: ``File→Shut Down``. This way, the slurm job will finish properly, and your data will be copied back on your home folder (as usual in a subfolder 'youname-`$SLURM_JOB_ID`'. If you do a scancel, you will have to copy your data from the compute scratch yourself and risk data loss if not done in the next days.

Notes:

- Si vous voulez lancer avec un autre module python, modifiez le fichier job. Après `module load anaconda/python3`, ajoutez une ligne `conda activate mon\_env`
- **Non recommandé, sauf si vous savez ce que vous faites.** Vous pouvez aussi lancer une résa en interactif et tout faire en manuel, par exemple
 - `srun -p intensive.q -time=00:30:00 -ntasks-per-node 1 -pty bash`
 - ça vous mets sur, par exemple, compute-0-5
 - `mkdir /scratch/villani\_monjupyter && cd /scratch/villani\_monjupyter`
 - `module load anaconda/python3` (ou 2, ou autre, puis charger un de vos environnements locaux avec `conda activate mon\_env`)
 - trouvez un port libre: `comm -23 <(seq 7777 8888 | sort) <(ss -Htan | awk '{print \$4}' | cut -d':' -f2 | sort -u) | shuf | head -n 1` ça vous sors par exemple 8146
 - `jupyter-lab --no-browser --port=8146`
 - faites le tunnel `ssh -L 8146:localhost:8146 cluster ssh -L 8146:localhost:8146 -N compute-0-5`
 - connectez vous depuis votre navigateur `http://127.0.0.1:8146/?token=XXXX`
 - `File→Shut Down` quand vous avez finis
 - rapatriez vos données du scratch avec un scp

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=1619629481>

Last update: **28/04/2021 19:04**

