Connection à OSIRIM

Pour plus d'information:
https://osirim.irit.fr/docs/slurm/

La connection à osirim se fait via SSH avec l'identifiant et le mot passe fournis:

```
$  ssh jbienass@osirim-slurm.irit.fr
```

CONSEIL
Avoir toujours un deuxième terminal ouvert pour pouvoir préparer et envoyer des fichiers via des commandes scp.

```
scp [autres options] [nom d'utilisateur source@IP]:/[dossier et nom de fichier] [nom d'utilisateur de destination@IP]:/[dossier de destination]
```

On se retrouve donc sur ce qui est appelé un nœud intéractif. si on peut exécuter des commandes sur ses nœuds, il est demandé de ne pas lancer des programmes volumineux sur ces nœuds pour que tout le monde puisse toujours avoir accès à la plateforme, les nœuds de calculs sont fait pour ça.
Pour pouvoir lancer un programme sur un noeud de calcul, il faut pour cela exécuter un script se basant sur cette mise en forme:

```
#!/bin/bash
#SBATCH --job-name=test1  #nom du batch
#SBATCH --mail-type=END  #Quand on veut recevoir un mail pour avoir des information sur le job
#SBATCH --mail-user=xxx@irit.fr #A quel mail on envoie les infos
#SBATCH --output=test1.out # permet de définir un fichier qui recevra ce que le programme sort

#Debut du script
date
sleep 10
date
```

les programmes sont ici sous forme de module qu'il faut charger en utilisant la commande

```
module load python3
python3 test.py
```

En utilisant la commande on lance le batch

```
$ sbatch monscript.sh
```

Pour ajouter des fichiers à OSIRIM

On peut utiliser la commande scp pour envoyer des fichiers depuis notre session.

Mais un depôt Github à été mis en place il suffit donc de déplacer les fichier dedans et de réaliser un git pull depuis la session OSIRIM dans le repertoire Demo-Osirim.

Plusieurs astuces

Plutôt que d'entrer continuellement le mdp pour le ssh et le mdp on peut utiliser un partage de clés RSA
sur son pc ET sur OSIRIM (si pas déjà réaliser) réaliser la commande:

```
$ ssh-keygen
```

Puis envoyer la clé publique de son pc (via scp) sur la plateforme pour l'ajouter au fichier "*authorized keys*"

On peut utiliser git également en clonant un projet sur la plateforme (après authentification avec la clé ssh)
Ce qui permet de garder une trace et d'être sûr de l'état dans lequel on envoie notre projet aux noeuds de calculs

Pour le machine learning

Plusieurs conteneurs sont présents et utilisables avec leurs outils (voir plus dans la docs)
pour lancer un conteneur il faut donc utiliser les commandes dans le batch

```
module load singularity/3.0.3
srun singularity exec /logiciels/containerCollections/CUDA11/mycontainer.sif python
"$HOME/moncodepyhton.py"
```

conteneur utilisé:
singularity shell /logiciels/containerCollections/CUDA12/pytorch2-NGC-24-01-py3.sif

JupyterLab

voir : https://osirim.irit.fr/docs/slurm/jupyterlab/

Pour créer un kernel custom pour jupyter notebook

Suivre les instruction sur la doc jusqu'à la partie “Execution :”

> *INFO*
> Sauf si changement drastique il n'y a pas besoin de recrer un environnement virtuel.
> Test_Mistral est utilisé pour le moment passez donc directement à l'ajout de packages.

https://osirim.irit.fr/docs/slurm/gpu/#installation-de-packages-supplementaires
Puis toujours dans l'environnement virtuel

```
pip install ipykernel
python -m ipykernel install --user --name Test_Mistral --display-name Ker_Mist
```

Une fois fait on peut lancer le notebook en suivant ce lien
https://jupyter-slurm.irit.fr
il faut alors selectionné le conteneur singularity que l'on utilise pour créer l'instance.

puis modifier le kernel par celui creer