SLURM Job Scheduler (For admins)

Formation LRI,
Corentin Tallec & Diviyan Kalainathan

Partie I: Atelier utilisateurs (Jan. 2019)

a) Intro
b) Fonctionnement de SLURM
c) Soumettre des jobs
d) Monitorer des jobs
e) Docker
f) Choses à eviter

Partie II: Atelier administrateurs (Dec. 2018)

a) Installer Slurm
b) Gestion des comptes
c) Gestion de Docker
d) Maintenance des noeuds
e) Discussion

Introduction

Contexte:

- Ressources de calcul **partagées** e.g. **GPUs**
- Ressources de calcul **importantes** e.g. **beaucoup de GPUs**
- Besoin de **gestion de tâches**
    - Éviter les **erreurs d'utilisation des ressources**: "Oups, OutOfMemoryException sur ton job d'une semaine, j'espère que tu fais des checkpoints…"
    - Faciliter le lancement d'**expés à grande échelle**

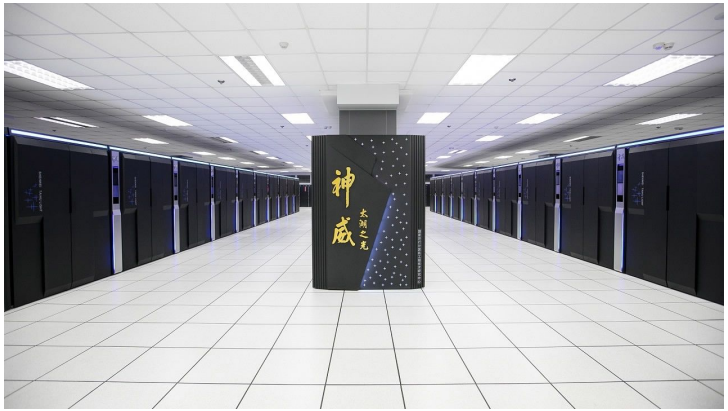
Introduction (2)

Intérêts d'un ordonnanceur (job scheduler):

- Gestion **automatique** et optimale des ressources:
  - si SLURM m'alloue un GPU, **il est à moi**, personne d'autre (sauf comportement malveillant) ne pourra s'en servir
- **Système de queue** pour des tâches:
  - 10 jobs ou 100 jobs, même combat, quitte à patienter pour les ressources.
- Système de **partitions** pour gérer les **priorités**:
  - Typiquement **n GPUs réservés par utilisateurs**, ininterruptibles. Au delà, allocation en fonction de l'utilisation de l'utilisateur + possibilité d'interruption.

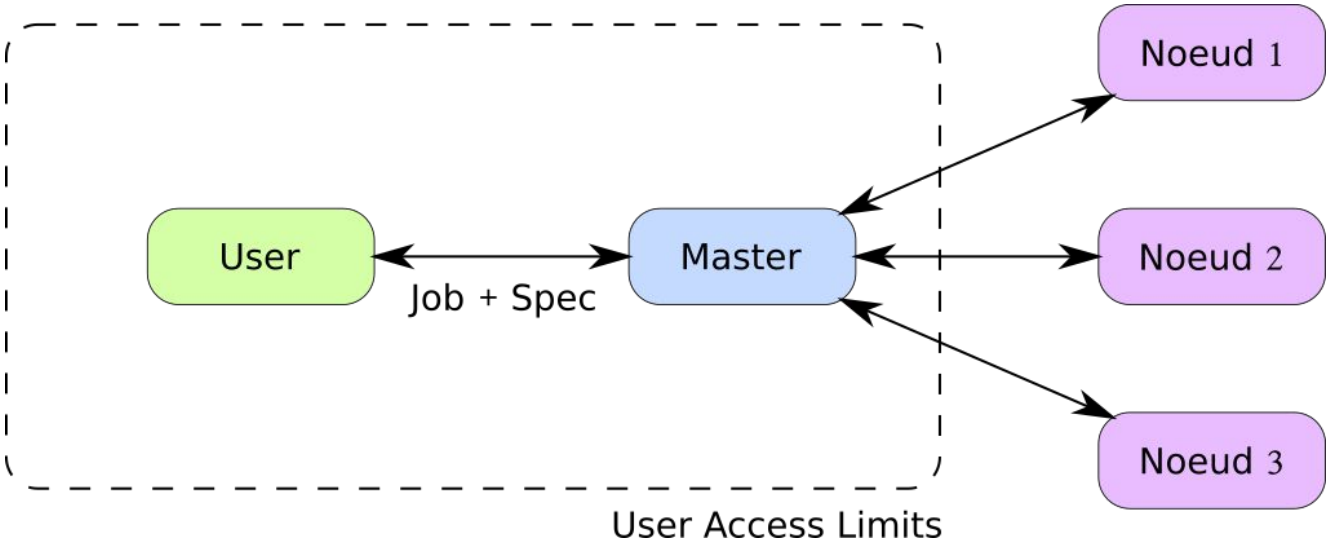
Pourquoi SLURM ?

1) Léger
2) Modulaire
3) Populaire
4) Scalable
5) Et surtout, relativement facile à installer



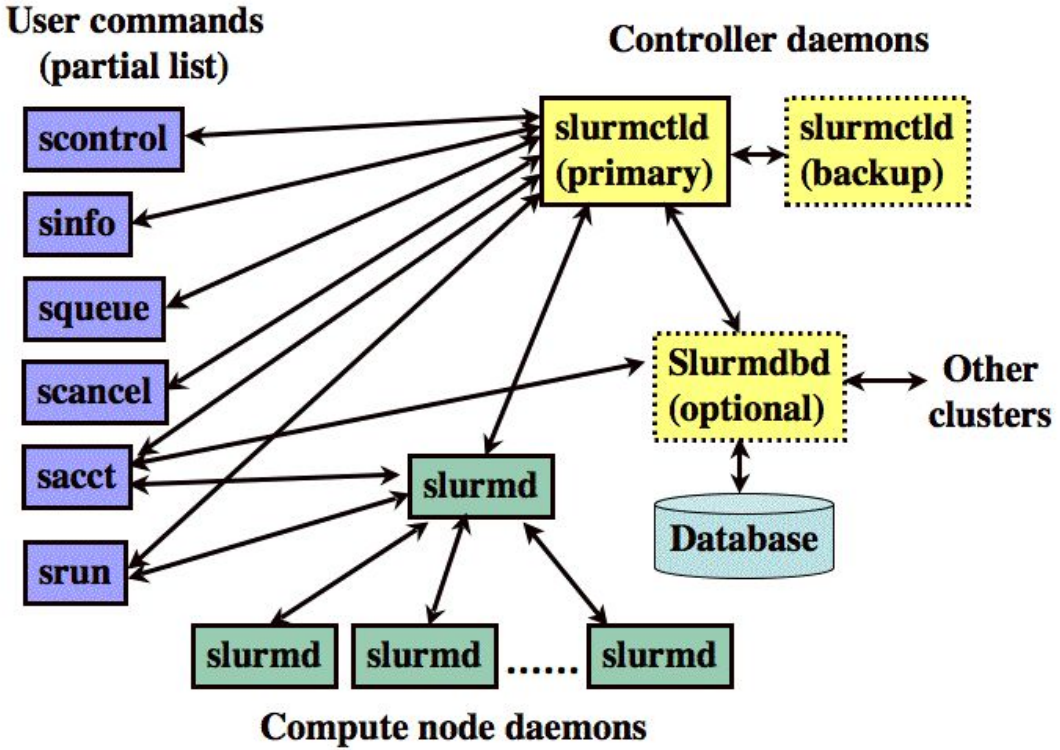
Sunway TaihuLight, 3e calculateur mondial en Juin 2018
(273 millions $ US, 93 PFlops)

Fonctionnement de SLURM: User side



Fonctionnement de SLURM: Admin side

- **slurmctld**: daemon de contrôle de slurm sur le **noeud master**
- **slurmd**: daemon slurm sur les **noeuds de calcul**



Informations importantes

Documentation de SLURM: https://slurm.schedmd.com/

- Besoin d'un **noeud master**/ frontale
- **Limiter les utilisateurs**: définir **max nb GPUs** ("Oups, j'ai lancé 300 jobs, maintenant plus personne n'aura de GPU pendant 3 semaines …")
- Besoin d'un **espace partagé**: home + storage

Lancer des jobs: **srun/sbatch**

A) Préparer l'installation de SLURM

Prérequis:

- Prévoir un **Noeud Master**: Gestion du **slurmctld** et de la db. Peut être une VM. **Point d'entrée unique** pour les utilisateurs.
- **Passwordless ssh** entre Noeud Master et tous les noeuds de calcul.
- Pour cette formation: **Ubuntu 16.04 sur tous les noeuds**.

Avant l'installation

- S'assurer que tous les noeuds sont **joignables en ssh depuis le noeud master**.
- Pour la suite:
  - Noeud master: **slurm-master**
  - Noeuds de calcul**: slurm-compute-i**
  - Tous les noeuds: **slurm-all**
- Se **logger en root** sur toutes les machines.

Outil en python très intéressant pour le déploiement sur toutes les machines avec une commande:

Fabric (http://www.fabfile.org/)

Créer l'utilisateur slurm

Pour fonctionner, slurm doit pouvoir s'**identifier sur chacune des machines** (et avoir les **bons droits**...). Les **UID** et **GID** de slurm sur toutes les machines **doivent correspondre**.

- Créer l'utilisateur **slurm** sur **slurm-master:**

```
root@slurm-master$ useradd slurm
```

- Récupérer le UID et le GID de slurm sur slurm-master:

```
root@slurm-master$ su slurm
slurm@slurm-master$ id -u slurm
slurm@slurm-master$ id -g slurm
```

Créer l’utilisateur slurm (2)

- Créer l’utilisateur slurm et le groupe slurm sur **tous les autres noeuds** avec le **même UID et GID que sur master**. Remplacer **slurm_gid** et **slurm_uid** par les valeurs correspondantes.

```
root@slurm-compute-i$ groupadd slurm -g slurm_gid
root@slurm-compute-i$ useradd slurm -u slurm_uid -g slurm_gid
```

Setting up passwordless SSH

Slurm doit pouvoir se connecter **à toutes les machines depuis la machine master**, et vice versa, sans intervention de l'utilisateur. Pour se faire

- Se logger **en tant que slurm** sur toutes les machines.
- Générer une clef passwordless par machine:

```
slurm@slurm-all$ keygen -t rsa
```

NB: Les connections sans password doivent être acceptées par tous les daemons ssh.

Setting up passwordless SSH (2)

- Copier dans le fichier **.ssh/authorized_keys** de `slurm-master` les clefs publiques de tous les noeuds de calcul.
- Copier dans le fichier **.ssh/authorized_keys** de tous les `slurm-compute-i` la clef publique de `slurm-master`.
- **Vérifier que le tout marche**: se connecter en tant que slurm, et passer d'une machine à l'autre.

Installation: Munge

Première étape: **installer Munge**. Munge est un **service d'authentification**. La présence de la **même munge.key sur tous les noeuds** permet à slurm de valider l'identité des noeuds. Il faut:

- Installer munge sur tous les noeuds:

```
root@slurm-all$ apt-get install munge
```

- **Remplacer** la clef munge sur tous les noeuds de calcul par la clef munge du noeud master

```
root@slurm-master$ scp /etc/munge/munge.key \
                       slurm-compute-i:/etc/munge/munge.key
```

NB: Dans notre cas des connections ssh sans password sont possibles entre les utilisateurs root. Si ce n'est pas le cas, il faut trouver un autre moyen, ou configurer ces connections.

Bien débugger

Vérifier les logs/états des services !

Debug divers:

- Vérifier l'état d'un service: `root@slurm-all$ service <service_name> status`
- Directement lancer les daemons slurms sans passer par `systemctl`, avec l'option verbose:

```
root@slurm-master$ /usr/sbin/slurmctld -D -vvv

root@slurm-compute-i$ /usr/sbin/slurmd -D -vvv
```

- Les fichiers de logs sont le plus souvent dans `/var/log/slurm-llnl/` (en fonction du fichier de configuration et du path indiqué)

Préparer la base de données (1)

On va utiliser **SLURM15**, **incompatible avec mysql 5.7**, version sur les dépôts de base Ubuntu:

```
root@slurm-master$ add-apt-repository \

                   'deb http://archive.ubuntu.com/ubuntu trusty universe'

root@slurm-master$ apt-get update

root@slurm-master$ apt install mysql-server-5.6 mysql-client-5.6

root@slurm-master$ service mysql status # -> if dead execute next line
root@slurm-master$ service mysql start
```

Préparer la base de données (2)

```
root@slurm-master$ mysql -p

mysql> create user 'slurm'@'localhost' identified by 'password';

mysql> grant all on slurm_acct_db.* TO 'slurm'@'localhost';

mysql> SHOW ENGINES; # Check INNODB support

mysql> create database slurm_acct_db;
```

Slurm 15

Nous pouvons maintenant installer slurm en propre:

```
root@slurm-all$ apt-get install slurm-llnl
```

Quelques permissions à modifier:

```
root@slurm-master$ chown -R slurm:slurm /var/spool
root@slurm-master$ chown slurm:slurm /var/run/slurmctld.pid
root@slurm-compute-i$ chown slurm:slurm /var/run/slurmd.pid
```

Installer mailprog:

```
root@slurm-master$ apt-get install mailutils
```

Slurm config file

Fichier dans le repo: https://gitlri.lri.fr/acl/ateliers-sequenceur-slurm

- Principal fichier de configuration du cluster, définit les noeuds, les queues, les options de base, etc
- A placer dans **/etc/slurm-llnl sur master et sur tous les noeuds**
- **Idée pratique:** placer le fichier sur un **espace partagé du cluster** et faire des **liens symboliques !**
- **Référence:** https://slurm.schedmd.com/slurm.conf.html

Installer Slurmdbd (+ regarder slurmdbd.conf)

```
root@slurm-master$ apt-get install slurmdbd
root@slurm-master$ service slurmdbd start
root@slurm-master$ service slurmdbd status

# + Debugger tous les problèmes jusqu'à avoir:

 slurmdbd.service - Slurm DBD accounting daemon
   Loaded: loaded (/lib/systemd/system/slurmdbd.service; enabled; vendor
preset: enabled)
   Active: active (running) since ven. 2018-12-07 11:59:45 CET; 3h 3min
ago
  Process: 18684 ExecStart=/usr/sbin/slurmdbd $SLURMDBD_OPTIONS
(code=exited, status=0/SUCCESS)
 Main PID: 18686 (slurmdbd)
```

B) Gestion des comptes

Elément le plus important de l'administration du cluster

Définir:

- **Des limites utilisateurs:** typiquement limite **GPU/user** (dimensionnement !)
- Des **groupes de machines:** Dans notre cas, une seul (clusters)
- Des **modes de fonctionnement:** Y a t'il possibilité de préemption ?
- Des **groupes d'utilisateurs:** Tous les utilisateurs ont-ils les mêmes droits ?

Définitions

Cluster: Groupe de machines (En général on n'en définit qu'un seul)

Partition: Set de règles sur un ensemble de noeuds: qui peut utiliser quels noeuds, peut-il y avoir préemption ? Typiquement queues **default** et **besteffort**. Un même noeud peut être dans plusieurs partitions.

Account: Regroupe un **ensemble d'utilisateurs** et un set de règles associées

User: Utilisateur, fait partie d'un "Account". Son nom est identique au login de la personne a qui il correspond.

Liens de référence:
https://slurm.schedmd.com/accounting.html et https://slurm.schedmd.com/resource_limits.html

Créer les entités dans la base de données

On va créer un **cluster spice**, un **account test** sur le cluster spice et un **user diviyan** sur l'account test.

```
root@slurm-master$ sacctmgr add cluster spice

root@slurm-master$ sacctmgr add account test Cluster=spice \
                   Description="test" Organization="LRI"

root@slurm-master$ sacctmgr add user diviyan Account=test
```

Gres: Generic Resources Scheduling (GPUs)

Gestion des GPUs:

Dans **slurm.conf**:

```
GresTypes=gpu
NodeName=citronelle CPUs=12 RealMemory=32096 Sockets=2 CoresPerSocket=6
ThreadsPerCore=1 Gres=gpu:pascal:2 State=UNKNOWN
```

Sur les noeuds (gres.conf):

```
Name=gpu File=/dev/nvidia[0-1]
```

Limiter l'allocation des ressources

- Typiquement **limiter nb_gpu/user**, deux étapes:
  - Créer une **Quality Of Service** (QOS) spécifique dans la **slurm db** (via sacctmgr):

```
root@slurm-master$ sacctmgr create qos default MaxTresPerUser=gres=gpu/4
```

  - Assigner la QOS à la **partition par défaut, dans tous les slurm.conf**:

```
PartitionName=... QOS=default
```

Créer plusieurs partitions avec différentes QOS

- Créer une seconde partition dans le **slurm.conf** p.e. avec une priorité plus faible

```
PartitionName=besteffort Priority=1
```

- Autoriser la préemption d'un job venant d'une **queue de faible priorité** par un job venant d'une **queue de forte priorité**:

```
PreemptType=preempt/partition_prio
```

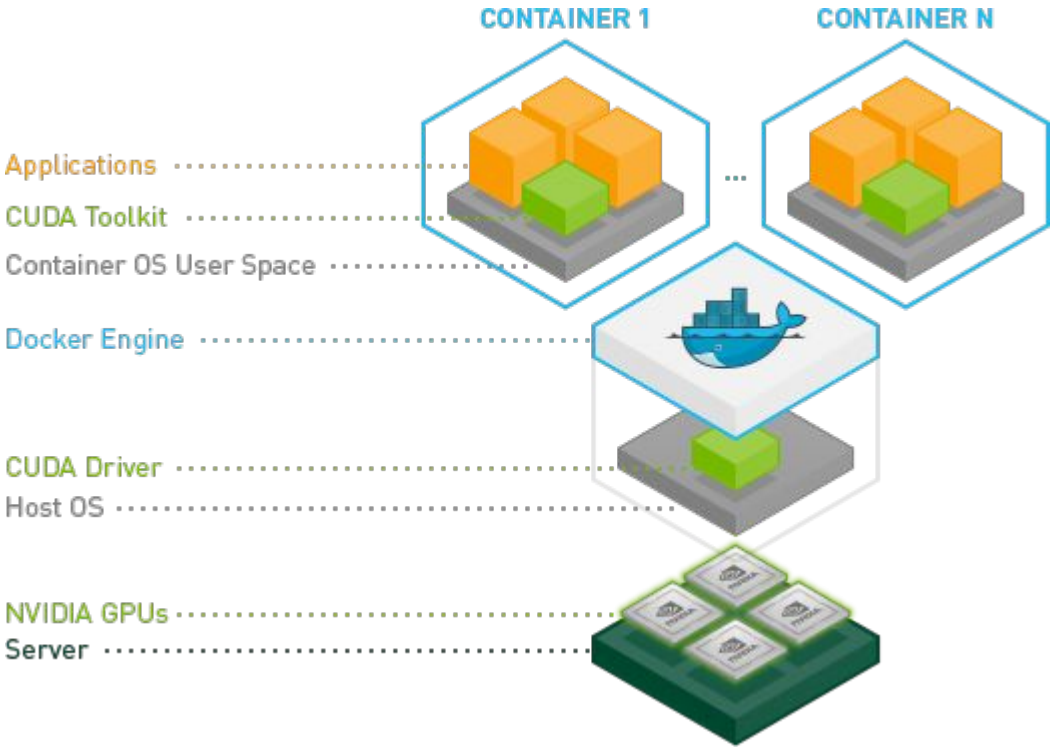
C) Docker

- A virtual environment manager
- Much more independent from the system than Anaconda
- A universal platform that uses containers (images) to store their environment
- "If it works on 1 device using the docker, it should run without issue on any device using docker"
- A space efficient git-like management of images.
- Multi-platform (Linux, Windows, Mac)

Pourquoi Docker?

- We got more and more users with their own needs in libs/apps.
- All those libs are becoming harder and harder to maintain
- Users can make their own tailored container
- We can use NVIDIA's NGC dockers, for a more well-maintained ecosystem between CUDA, CUDNN, and the python libs.

Nvidia-docker



Installation

Pour chaque noeud de calcul (et pas master):

https://docs.docker.com/install/linux/docker-ce/ubuntu/

Ajouter tous les utilisateurs au groupe ‘docker’ sur tous les noeuds, pour avoir l’autorisation de l’exécuter

Options importantes dans l'exécution des docker

--init : Slurm pourra kill les docker et les process

-u=$(id -u):$(id -g) : Les fichiers crée par les utilisateurs pourront être lus et modifiés par ces derniers.

--rm: supprimer le container à la fin de l'exécution pour sauver un peu d'espace

NV_GPU=$CUDA_VISIBLE_DEVICES : (devant la commande docker) Pour n'allouer à l'utilisateur que les GPUs attribués par SLURM

Notes

- Impossible de partager les dossiers d’images entre les noeuds
- Consommation importante de stockage: besoin de nettoyage régulier
- Penser à ajouter les nouveaux utilisateurs au groupe ‘docker’ pour avoir la permission d’en lancer
- Environnement très sécurisé -> ‘rkt’, alternative

D) Maintenance des noeuds

- Nettoyer les vieilles images des noeuds:

```
$ docker system prune
```

- Vérifier la disponibilité des noeuds

```
$ sinfo
```

Un noeud est déconnecté!

```
diviyan@ciboulette  sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
debug*       up   infinite      1  idle* citronelle
```

- Redémarrer le daemon slurmd (sur le noeud):

```
$ sudo service slurmd start
```

- Mettre à jour l'état du noeud depuis master:

```
$ sudo scontrol update NodeName=”citronelle” State=RESUME
```

Si l’un des daemon ne se lance pas...

Regarder les logs!

- Souvent un fichier de config qui n’est plus accessible
- Config incompatible qui fait crasher le daemon

“Faire la police”

- Vérifier de temps en temps l’utilisation du cluster
- Vérifier que les dockers utilisent les bonnes options
- Monitorer la consommation de stockage
- Adapter les limites utilisateurs en fonction de la demande et de l’offre

Faire une documentation pour vos utilisateurs

- La doc de slurm est compliquée et pas intuitive.
- Ne comporte rien sur docker, la connexion ssh

https://gitlab.inria.fr/dkalaina/titanic-docs

Aider vos utilisateurs avec des fonctionnalités

```
$ sstatus
Titanic status:

GPU Usage: 18/24
Free Nodes: 1/7: baltic-1
================================================
Job details for user ebrame:

Using 4/4 GPUs allowed on priority queue.

 Partition        JobID    JobName   State     ReqGRES   CPUTime
---------- ------------ ---------- ---------- ------------ ----------
   titanic 74306       inception RUNNING       gpu:4 2-01:27:11
```

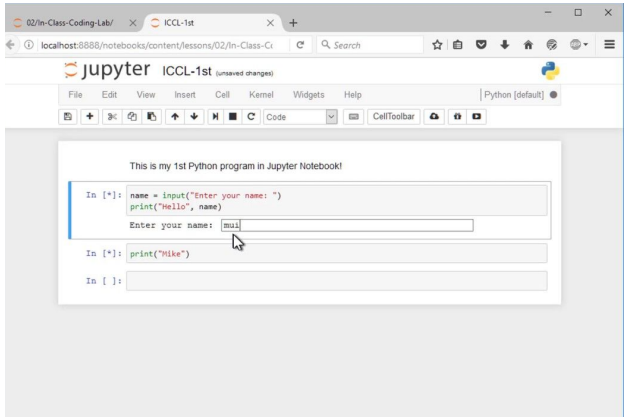
Aider vos utilisateurs avec des fonctionnalités (2)

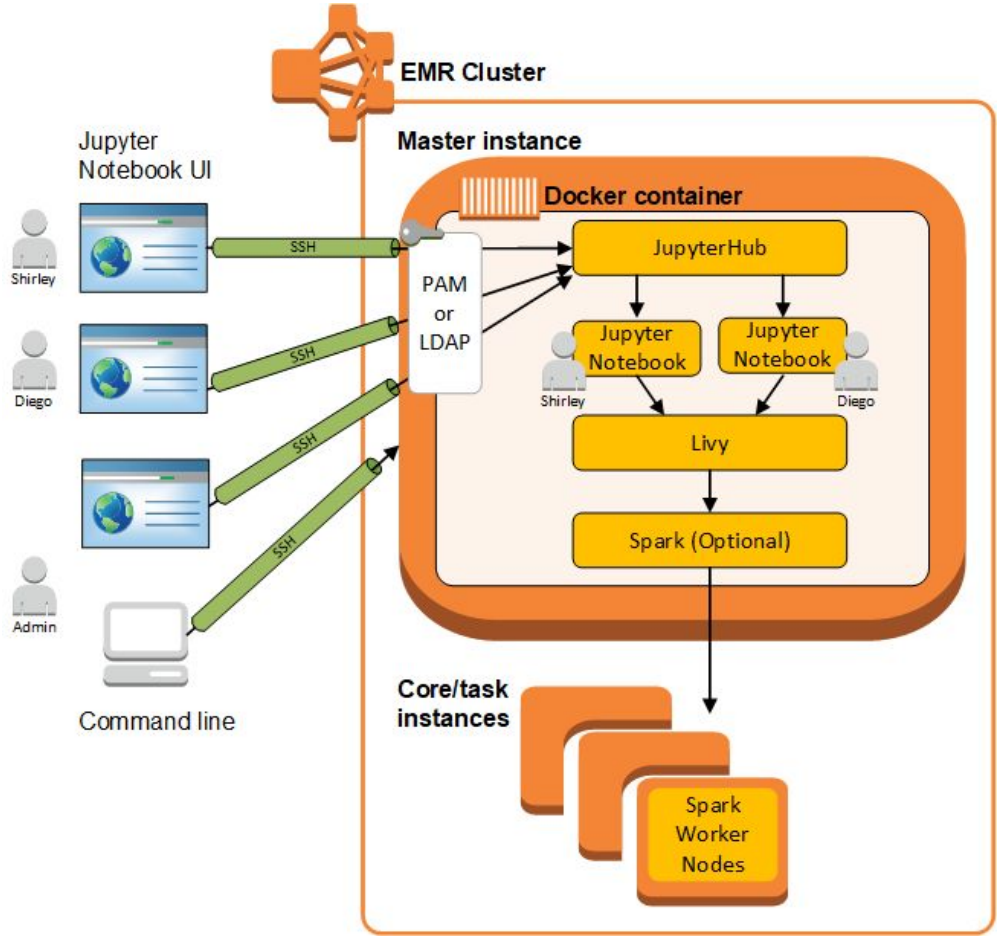
Wrapper autour de docker avec toutes les fonctionnalités obligatoires:

```
#!/bin/bash

#sdocker.sh
NV_GPU=$CUDA_VISIBLE_DEVICES nvidia-docker run --init --ipc=host --rm
-u=$(id -u):$(id -g) "$@"
```

Discussion : JupyterHub





Discussion: Grafana monitoring