KslHub Documentation

Release 0.0.15

Samuel KORTAS

Apr 13, 2019

Contents

CHAPTER 1

What is *KslHub*?

Developped by the KAUST Supercomputing Laboratory (KSL), *KslHub* is an dataHub based on jupyterhub and enabling a dynamic support of the possible spawner.

KslHub is released as an Open Source Software under BSD Licence. It is available at https://github.com/samkos/kslhub

1.1 Features

KslHub allows a user to:

CHAPTER 2

Installation

2.1 Requirements

KslHub installs on the top of jupyterhub 0.9.4 running on a version of Python higher or equal than 3.6.

At this stage, the integration with Jupyterhub still needs to be improved as we need to patch some of the source files of jupyterhub sources. That's why the installation of jupyterhub v 0.9.4 is enforced as the patches were only validated with this specific version.

We are working to find a better solution for the future releases.

2.2 Distribution

KslHub is an open-source project distributed under the BSD 2-Clause "Simplified" License which means that many possibilities are offered to the end user including the fact to embed *KslHub* in one own software.

kslhub latest development version can be found at https://github.com/samkos/kslhub

At this stage, *kslhub* can not be considered to be production ready.

The most recent documentation about *KslHub* can be browsed at http://kslhub.readthedocs.io.

Docker images are also available as *samkos/kslhub* or *samkos/kslhub_slurm* which provide an already installed *kslhub* along with a emulated 5-node cluster managed with SLURM.

2.3 Installing *KslHub* as a *docker* container

A fully configured ready to set container can be downloaded from *Docker Hub* (https://cloud.docker.com/repository/docker/samkos/kslhub_slurm) via the command:

```
$ docker run -t -i -h kslhub  -t -i -p 8000:8000 -p 33333:22 samkos/kslhub_slurm
```

It will run a container with kslhub installed along with a slurm cluster of 5 nodes. 3 users are already exiting in this container: *hub*, *bob* and *alice*. Their login and password are their respective name.

hub is the user having *kslhub* already installed and preconfigured in the directory */home/hub/kslhub/*. He has the responsability to launch the hub:

```
$ cd kslhub
$ kslhub -f docker_slurm
```

In this configuration, *kslhub* responds on port 8000 of your localhost (thanks to the port mapping * *docker run -p 8000:8000*), and one can connect to the container via *ssh*, available on port 33333 of your localhost (thanks to the port mapping * *docker run … -p 33333:22*)

2.4 Installing *KslHub* from scratch using the installation script

If you want to install a full wrking version of kslhub on a linux box, an installation script *Install_here.sh* is provided in the github repository. Go to a fresh directory and just source this script:

```
$ . scripts/Install_here.sh
```

Running this script should:

- download and install the latest version of miniconda in *<current_directory>/BUILD/miniconda*,,
- create and configure correctly a conda environment in *<current_directory>/kslhub_conda_env*
- in this conda environment, install and configure a *kslhub* working environment

The output log of this execution are expected to append the file *<current_directory>/Install_here.log*.

Once the script completed, a fully functional *kslhub* working environment as well as a shell script named *kslhub_init_env.sh*. To start using it, load the newly created conda environment with the following commands:

```
$ . kslhub_init_env.sh
```

and run the hub:

```
$ kslhub
```

2.5 Installing *KslHub* using PIP

2.5.1 Installing *KslHub* as root using PIP

To install *KslHub* as a standard Python package using PIP[1] as root:

```
$ pip install kslhub
```

Or alternatively, using the source tarball:

```
$ pip install kslhub-0.0.x.tar.gz
```

Once the package is installed, one has to complete *KslHub* configuration by issuing the following command:

[1] pip is a tool for installing and managing Python packages, such as those found in the Python Package Index

```
$ kslhub --init
```

2.5.2 Installing *KslHub* as user using PIP

To install *KslHub* as a standard Python package using PIP as an user:

```
$ pip install --user kslhub
```

Or alternatively, using the source tarball:

```
$ pip install --user kslhub-0.0.x.tar.gz
```

2.6 Installing *KslHub* using Anaconda

One have to be sure to have a conda environment with a version of python at least greater than 3.6 with pip installed. For this let's create a new conda environment:

```
$ conda create --name my_kslhub pip python==3.6

$ conda activate my_kslhub
```

In order to support the latest available features of *jupyterlab*, one have to install a recent version of *nodejs* and *configurable-http-proxy*. This needs to be done thanks the following command:

```
$ conda install -y -c  conda-forge/label/cf201901 configurable-http-proxy==3.1.0␣
↪nodejs==8.10
```

Then *kslhub* can be intalled from http://pypi.org thanks to:

```
$ pip install kslhub
```

2.7 Installing *KslHub* Source

Current source is available on Github, use the following command to retrieve the most updated version from the repository:

```
$ git clone git@github.com:samkos/kslhub.git
```

2.7.1 Building a local docker image

The docker image available from *docker hub* as *samkos/kslhub_slurm* can be obtained from the source by issuing the following commands:

```
$ cd docker
$ docker build -t kslhub_slurm -f Dockerfile_kslhub_slurm .
```

CHAPTER 3

Using *KslHub*

3.1 Initial Configuration step

3.1.1 Installing some nice jupyter notebook extensions

Once the *KslHub* package is installed (wether via Pip or anaconda), one has to complete *KslHub* configuration by issuing the following command:

```
$ kslhub --init
```

Behind the scene, this command installs some nice jupyter notebook extensions as

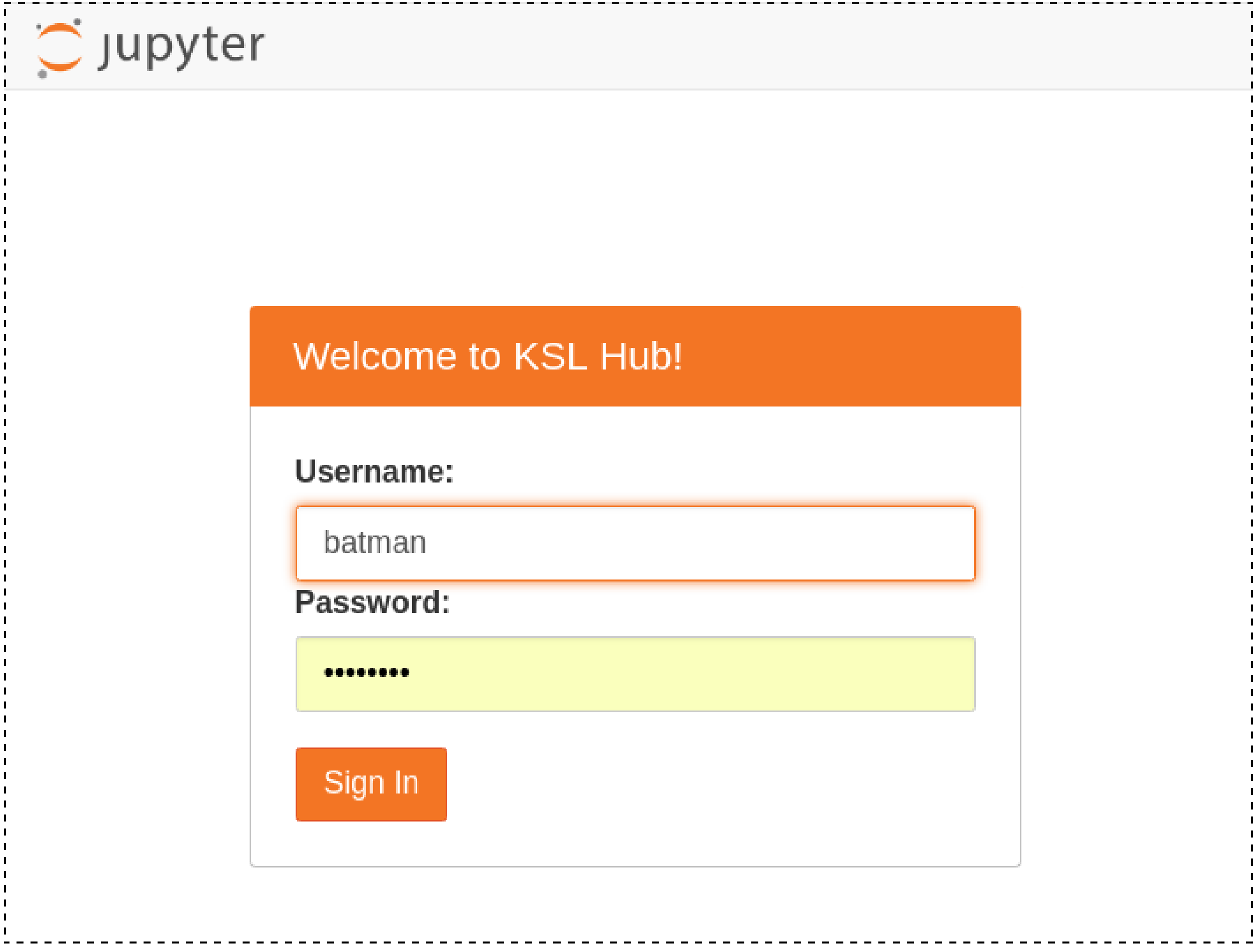
- the slurm magic package
- the support of widget
- the support of bash commands in a notebook cell

3.2 Starting the hub

The hub can be started with the following command:

```
$ kslhub
```

3.2.1 *KslHub* configuration file



By default, the hub interface shows this welcoming page on port 8000 with the user authenticated on the same machine thanks to regular username/password credentials. It will search for job templates files in the following directory:

```
<current_directory>/job_templates
```

To change this default configuration, one can aedit these parameters via a configuration file to be invoked as:

```
$ kslhub --config <my_configuration_file>
```

A default configuration file is generated with the following command:

```
$ kslhub --generate-config
```

A default set of job template files is generated with the following command:

```
$ kslhub --generate-job-templates
```

CHAPTER 4

Dynamic scripts

Here is described the template syntax we adopted to provide an automated user interface to a parametrized job script.

4.1 A very first example

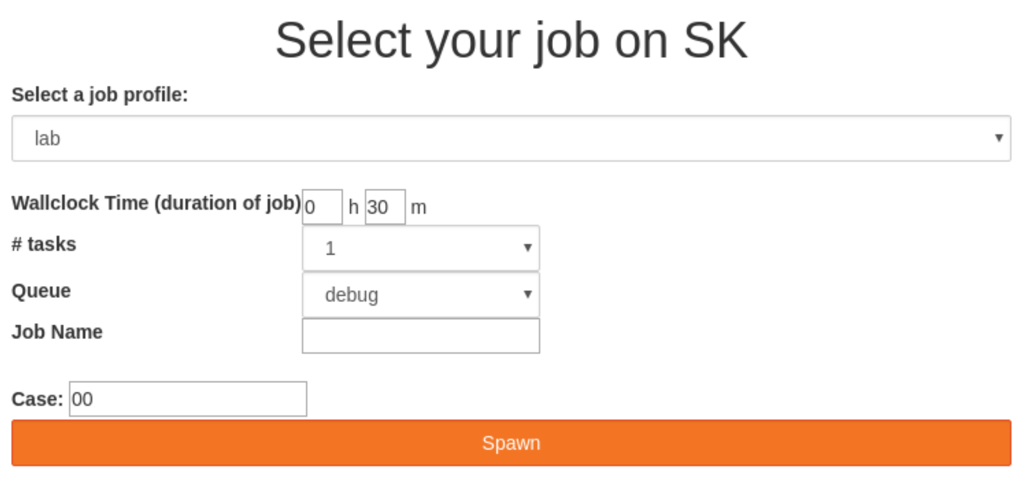
Here is the file 00 - lab.template

```
------------------------------------------------------------------------------------
↪------------------------------------------
#!/bin/bash
#SBATCH --time=0:20:00
#SBATCH -n __nb_tasks;select;# tasks;1,2,3,4,8,16,32,64,128,512,1024__
#SBATCH --partition=__partition;select;Queue;debug,workq__
#SBATCH -J jupyterhub-singleuser
#SBATCH -o __JOB_DIR__/__job_name;input;Job Name;text;my_job__.%j.out
#SBATCH -e __JOB_DIR__/__job_name__.%j.err
#

echo RUNNING `which jupyter-labhub` on __nb_tasks__

jupyter-labhub --ip="0.0.0.0" --port=__NOTEBOOK_PORT__  --notebook-dir="~/NOTEBOOKS" -
↪-debug
------------------------------------------------------------------------------------
↪-------------------------------------------
```

will show in the hub as


Select your job on SK

Select a job profile: lab

Wallclock Time (duration of job) 0 h 30 m

tasks 1

Queue debug

Job Name

Case: 00

display could and will be improved of course. . .

See how

```
__nb_tasks;select;# tasks;1,2,3,4,8,16,32,64,128,512,1024__
```

became a select item on the web page? Once value selected and the button spawn hit, selected option are replaced in the job that is ppawned to shaheen. The first command makes the connection with the hub from the first node of the job:

```
jupyter-labhub --ip="0.0.0.0" --port=__NOTEBOOK_PORT__  --notebook-dir="~/NOTEBOOKS" -
↪-debug
```

with *__NOTEBOOK_PORT__* also replaced with the current address of the hub and a unique port number per job handled by the hub.

4.2 General syntax

4.3 Web components supported so far

4.4 pre-defined environment variables

- *JOB_DIR*: directory where *kslhub* stores temporarily generated job scripts.

CHAPTER 5

Release Notes

5.1 0.0.15 (Apr 13, 2019)

- bug fixing
- docker file ready
- sourcing ksl_init_env.sh added to the initialization of the job
- KSLHUB_ROOT is gathered from the hub launching directory and forwarded to the spawned job
- cleaning of template generation

5.2 0.0.14 (Apr 8, 2019)

- bug fixing
- adding script/Install_here.sh that produces a kslhub environment in the current directory
- configuration of the hub is now accessible via full path of python config file or either just a name to choose between configuration available per default
- automated creation of job_templates, jobs, runtime and logs directories if missing
- added a standardized _version file taken into account in setup.py

5.3 0.0.13 (Apr 6, 2019)

- taking care of default choice in template
- first attempt of installation scripts

5.4 0.0.11 (Apr 2, 2019)

- getting rid of all ip address that were hard-coded for shaheen
- updated configuration file allowing to choose ip and port for the hub and proxy
- released for test only.

5.5 0.0.5 (Mar 19, 2019)

- bug fixes related to the spawning of the jupyterlab environment
- released for test only.

5.6 0.0.4 (Mar 18, 2019)

- netiface dependency removed of no use at this stage and was requiring gcc to be installed on the machine to be deployed.
- released for test only.

5.7 0.0.3 (Mar 18, 2019)

- very first public release, still in developement and unstable
- released for test only.