How FireWorks. *Or* –

An overly complicated “hello, world”

Our goal is to print two lines: first, “hello, “, then “world”. In order to make sure that the second line is not printed before the first, we are going to use a workflow manager.

Workflow –	A directed acyclic graph of Fireworks, in other words, a list of Fireworks and dictionary of links (parent : child)
Firework –	A list of computing tasks (Firetask) to be performed. It should be able to finish in time when using a queueing system. Information can be passed to its children.
Firetask –	A computing task, which makes up the smallest chunk of your workflow. There is, however, no restriction to its complexity. It can be called through python or bash.

-1. Login, get files

Log in taito.csc.fi, cd to $WRKDIR, download and uncompress the input files from hello_world.tar.gz (link in the School homepage).

0. MongoDB

You need MongoDB. Without it, FireWorks will not run. It is required by the program internally to manage your workflows.

First option: Use one of our databases on MongoDB Atlas

You should have gotten a number: in the case below, it’s “02”:

In user_lpad.yaml change the database to “csc02” and the username to “user02”. The password, host and the address are already filled out. You now have access to your own small server database. You can use it to your liking even after the school finishes for some time.

Second option: set up your own free database on MongoDB Atlas
https://www.mongodb.com/cloud/atlas

This option takes some time, since you need to get the host address, create a database and a user. Once you have set up your own database, enter the credentials in user_lpad.yaml.

Third option: set up MongoDB locally

This option takes the most time. Download and install MongoDB from here:
https://www.mongodb.com/download-center/community?jmp=docs

As long as you have a lightweight database, you can run MongoDB on your login node. But once your database gets more serious, set up MongoDB on your own server or make use of the CSC Cloud computing service: https://research.csc.fi/cloud-computing

1. Install FireWorks

Installing FireWorks should be as easy as

```
pip install FireWorks
```

… but in a supercomputing environment, where you don't have root privileges, you need to do additional steps. In taito.csc.fi, you should load a suitable Python environment, create a virtual environment for your own Python packages, activate it, and then install FireWorks:

```
module load python-env/3.4.5
virtualenv fireenv
source fireenv/bin/activate
pip install FireWorks
```

...and optionally, if you want to a graphical view of your workflows (this takes a while and is not needed for running FireWorks)

```
pip install matplotlib
```

2. Connect to your database

The launchpad is your database. It will manage your jobs for you.

```
lpad -l user_lpad.yaml reset
```

With this command, you can reset your database. All previous workflows will be deleted. Currently, your database is empty. If you get an error, something is wrong with your connection, check user_lpad.yaml

3. Run your first workflow locally

First, add a workflow to your launchpad/database.

```
lpad -l user_lpad.yaml add helloworld_wf.yaml
```

You can check for the status of your jobs running:

```
lpad -l user_lpad.yaml get_fws
```

It looks like you are ready to run your first workflow.

```
rlaunch -l user_lpad.yaml rapidfire
```

Congratulations, you have just run your first workflow! Note, that on taito.csc.fi all these tasks were run locally, i.e. on the login node. You should not run but very light tests on the login node. Real work is to be done via the queuing system. Also try out the command below, after adding more helloworlds to your launchpad/database.

```
rlaunch -l user_lpad.yaml singleshot
```

But wait, so far you have only run it locally! This might be good for testing and small jobs, but not for heavy tasks. Let's get more serious.

4. Run it through a queue

This tutorial assumes that you are on the login node of a computer cluster using the job manager SLURM. Most other job managers are also supported.

Take a look at and edit the file my_qadapter.yaml

This command "rlaunch -l /path/to/my/user_lpad.yaml singleshot" should look familiar, but you need to change the path to user_lpad.yaml to the absolute path.

Fireworks uses the my_qadapter.yaml file as a way to set up a slurm file. In order to see what is put where, check SLURM_template.txt.

Once you have added another helloworld to your launchpad/database, you are ready to launch.

```
qlaunch -q my_qadapter.yaml -l user_lpad.yaml rapidfire
```

In another terminal you can check the progress by checking the status

```
lpad -l user_lpad.yaml get_fws
```

It should not take too long until all your "hello, worlds" are finished. Congratulations, you ran your first workflow through a queue! Now, try to find where "hello, world" was printed.

5. For pythonistas

Above, you have seen the convenient command line options. All of the above is possible through python. Especially adding a workflow can be tedious in yaml format and rather convenient in python. Take a look at and edit add_helloworld.py

Once you changed the database and username in the Launchpad(...), you can run it with

```
python3 add_helloworld.py
```

You can check if they are added to the launchpad/database and run it as before through the command line:

```
lpad -l user_lpad.yaml get_fws
```

```
qlaunch -q my_qadapter.yaml -l user_lpad.yaml rapidfire
```

Note: although it is possible to launch the workflow to the queue directly from python, it is actually not more convenient. You would need the CommonAdapter python class from FireWorks, as well as the user_lpad.yaml file.
Also, if you import your own python files from your main file, you should set PYTHONPATH accordingly.

6. Going further

You can use the above template to execute any two bash commands in sequence. With a few changes you can add a more complex workflow as a directed acyclic graph (DAG) with more Fireworks and more links. There are many more ready-made recipes such as FileTransferTask, FileWriteTask and PyTask.

There is a comprehensive documentation with many detailed tutorials available, take a look at: https://materialsproject.github.io/fireworks/

We have not covered **dynamic workflows** and **custom Fireworks** yet. Also, we have not had to **pass any information** from one Firework to its children. There are basically no limits except your imagination on how your workflows are built up.