JupyterHub integration with LTI

In designing MOOC courses

Compiled by - Ranji Raj

Last updated: September 24, 2021

JupyterHub integration with LTI

Note: *This document is written after a POC on Linux machine (Ubuntu 18.04 LTS).*

Technical Overview

- Multi-user Hub (Tornado process)
- Configurable HTTP proxy (node-HTTP-proxy)
- Multiple single-user Jupyter notebook servers (Python/Jupyter/Tornado)

The basic principles for operation are:

- Hub launches a proxy.
- The proxy forwards all requests to Hub by default.
- Hub handles login and spawns single-user servers on demand.
- Hub configures proxy to forward URL prefixes to the single-user notebook servers.

Steps followed for configuring JupyterHub to the local system

Checking prerequisites:

- A Linux/Unix based system
- **Python 3.5** or greater
- nodejs/npm

(Note: JupyterHub runs only on Python3 and POSIX generally and on Linux/UNIX based systems and would have trouble running on Windows and also it will not run on Python2)

pip, npm

```
python3 -m pip install jupyterhub
```

```
npm install -g configurable-http-proxy
```

1. First, clear the cache (if any):

```
npm cache clean -f
```

Discrepancies may happen if outdated versions of npm are found check with:

```
npm -g outdated
```

2. Next is to acquire SSL certificates:

- It is optional to run JupyterHub without SSL but it is recommended to run using self-signed certificates using certs and keychains.

```
openssl req -x509 -nodes -days 365 -newkey rsa:1024 -keyout jupyterhub.key -out jupyterhub.crt
```

(Place the SSL files in the same directory as the project folder in the virtual environment.)

Or alternatively can create an **environment.yml** and then:

```
conda env create -f environment.yml
source activate openhpi-jupyterhub
```

3. Create a configuration file - To generate a default config file with settings and descriptions and run:

```
jupyterhub --generate-config
```

4. In the **jupyterhub_config.py** file configure the variables with prefix *c.XX_XX* First, set the path to SSL certificate for the public-facing interface of the proxy.

```
c.JupyterHub.ssl_cert = '/home/user/./jupyterhub.crt'
c.JupyterHub.ssl_key = '/home/user/./jupyterhub.key'
```

Steps followed for configuring LTI to JupyterHub

(*As of LTI 1.2.0*) Create a client secret and client key for use by openHPI to authenticate to our hub. Opening the bash and run:

LTI Client key:

```
openssl rand -hex 32
```

LTI Shared secret (same command twice):

```
openssl rand -hex 32
```

Now use these two to allow **openHPI** and **JupyterHub** to authenticate each other also in the *jupyterhub_config.py* add these:

```
c.JupyterHub.authenticator_class = 'ltiauthenticator.LTIAuthenticator'
c.JupyterHub.consumers = {
os.environ['LTI_CLIENT_KEY']: os.environ['LTI_CLIENT_SECRET']
}
```

Here **'LTI_CLIENT_KEY'** and **'LTI_CLIENT_SECRET'** both are taken from env files for the reason being these are private.

Configuring DockerSpawner

We make use of `netifaces` library from python to configure to interface the docker container. By default the existing docker process from the image is spawned into the user process as the authorized applications that use OAuth tokens to identify users on the system.

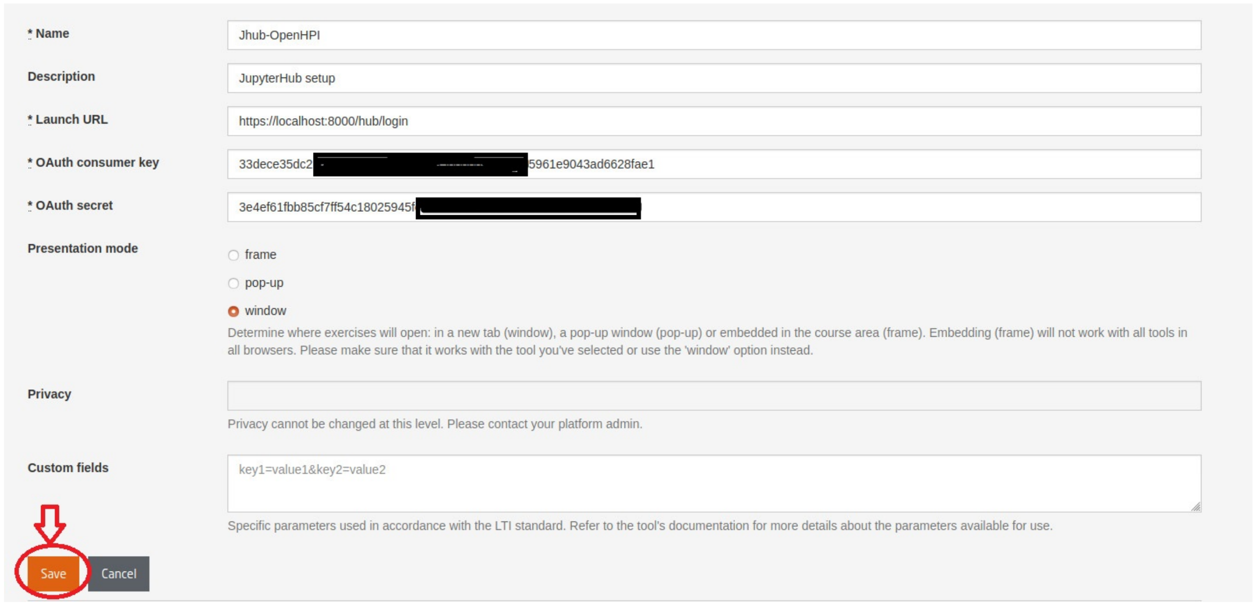
```
from dockerspawner import DockerSpawner
c.JupyterHub.spawner_class = DockerSpawner

import netifaces
docker0 = netifaces.ifaddresses('docker0')
docker_ipv4 = docker0[netifaces.AF_INET][0]
c.JupyterHub.hub_ip = docker0_ipv4['addr']
```

Creating LTI provider

- Add a new LTI provider.
- Set the launch URL to the callback URL.
- Set presentation mode to the **window**.
- Link the Secret and Client key from the **env** file above.

Once the above step is done then type **source ./env** and then type **jupyterhub** from the command line/bash followed by clicking **Launch exercise tool** from the OpenHPI.

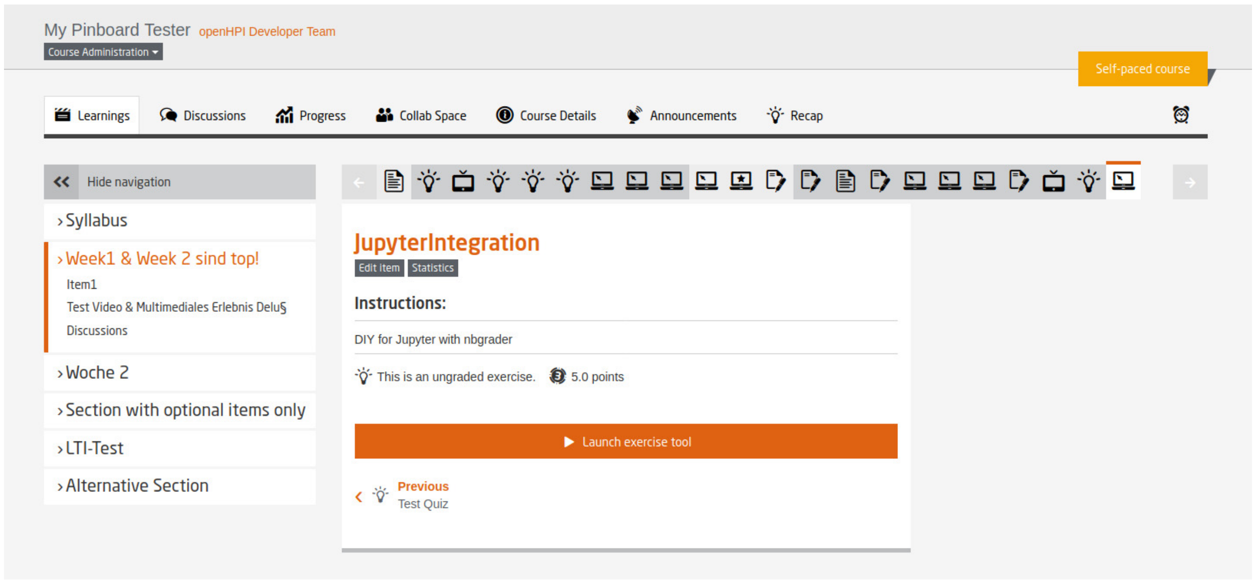


* Name	Jhub-OpenHPI
Description	JupyterHub setup
* Launch URL	https://localhost:8000/hub/login
* OAuth consumer key	33dece35dc2[redacted]5961e9043ad6628fae1
* OAuth secret	3e4ef61fbb85cf7ff54c18025945f[redacted]
Presentation mode	frame / pop-up / [✓ window]
Privacy	
Custom fields	key1=value1&key2=value2

Determine where exercises will open: in a new tab (window), a pop-up window (pop-up) or embedded in the course area (frame). Embedding (frame) will not work with all tools in all browsers. Please make sure that it works with the tool you've selected or use the 'window' option instead.

Privacy cannot be changed at this level. Please contact your platform admin.

Specific parameters used in accordance with the LTI standard. Refer to the tool's documentation for more details about the parameters available for use.



JupyterIntegration

Instructions:

DIY for Jupyter with nbgrader

This is an ungraded exercise. 5.0 points

Configuring kernels to JupyterHub

R

Run the following inside a terminal from the `jovyan` user.

```
conda install -c r notebook r-irkernel
```

Julia

First step is to install the latest version of **Julia** inside the `jovyan` user.

More information on here.

Then add Julia as a kernel to JupyterHub from here.

Set up the `exchange directory` first

This is one of an important step in `nbgrader` which allows you to enable the **release & collect** functionality for assignments and in turn for auto-grading.

In the directory (home) create a tree : **/srv/nbgrader/exchange** manually & from the bash execute:

```
sudo mkdir /srv/nbgrader/exchange
```

And give the appropriate permissions:

```
sudo ugo+rw /srv/nbgrader/exchange
```

Here **ugo** stands for **user-group-other** by which we are giving a chain permission of reading and writing (*this is equivalent to* ***chmod -R 777***). All users must have the permission to read and write.

Go to the main terminal and execute:

`sudo su`

Note: *You must execute these from the main terminal (i.e. from VS Code or any other) and not from the docker container. If you do so then it will create a `jovyan` user and will prompt for password and some of the above commands and installations might break.*

Steps for configuring `nbgrader` with JupyterHub

First update your **pip** version or use **pip3** alternatively from your bash and execute:

```
pip3 install nbgrader
```

Or via *Anaconda* by:

```
conda install jupyter
conda install -c conda-forge nbgrader
```

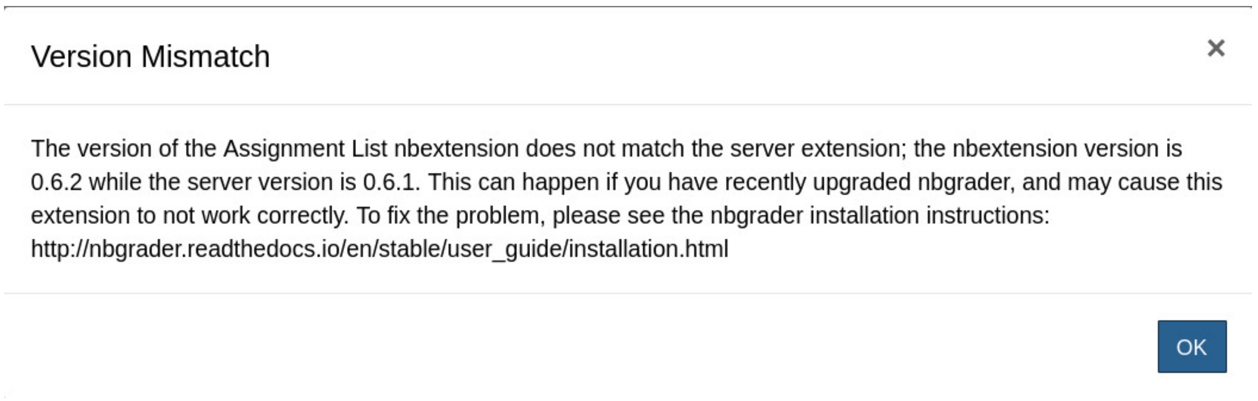
Next step is to install & enable the **nbextension** and **serverextension** by the following from the same root directory where all the previous installations are residing:

```
jupyter nbextension install --user --py nbgrader --overwrite
jupyter nbextension enable nbgrader --py --sys-prefix
jupyter serverextension enable --sys-prefix --py nbgrader
```

(*We use the* ***overwrite*** *to re-install any older versions of the nbextension on your local system. Moreover the versions of both* ***nbextension*** *&* ***serverextension*** *must be the same. In the event where these two extensions have a mismatch the validate functionality inside the notebook will not be available. This is required in the future for* ***autograding***)

Finally to see all installed nbextensions/serverextensions, run:

```
jupyter nbextension list
jupyter serverextension list
```


Version Mismatch

The version of the Assignment List nbextension does not match the server extension; the nbextension version is 0.6.2 while the server version is 0.6.1. This can happen if you have recently upgraded nbgrader, and may cause this extension to not work correctly. To fix the problem, please see the nbgrader installation instructions: http://nbgrader.readthedocs.io/en/stable/user_guide/installation.html

Then generate a configuration file for `nbgrader` as:

```
nbgrader generate_config
```

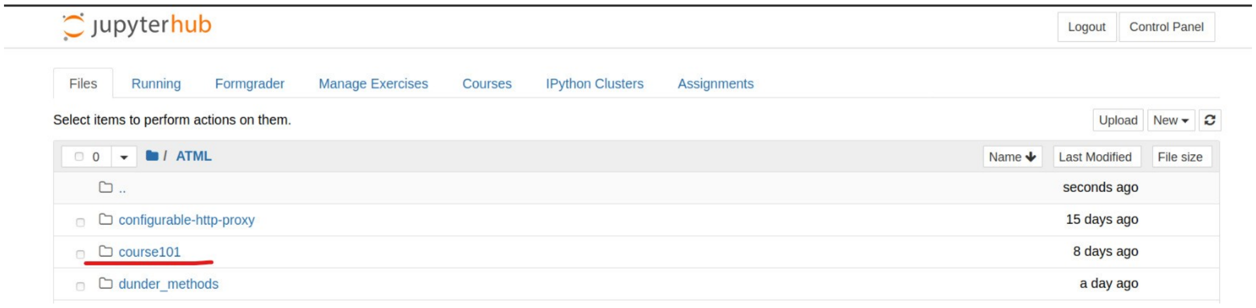
Once the above steps are done we are now ready to configure `nbgrader`. At first start by using the `quickstart` command to create a fresh course:

```
nbgrader quickstart "<name-of-your-course>"
```

Once this is done it will create a course in your local directory inside your docker:

Attention: Upon the above action you will get a `nbgrader_config.py` file. You must replicate this two times (locally and globally) at two different location while you set up for `formgrader`.

One of the config file as shown below will contain a structure like this where we create a list of assignments and list of registered students in the database.



Select items to perform actions on them.

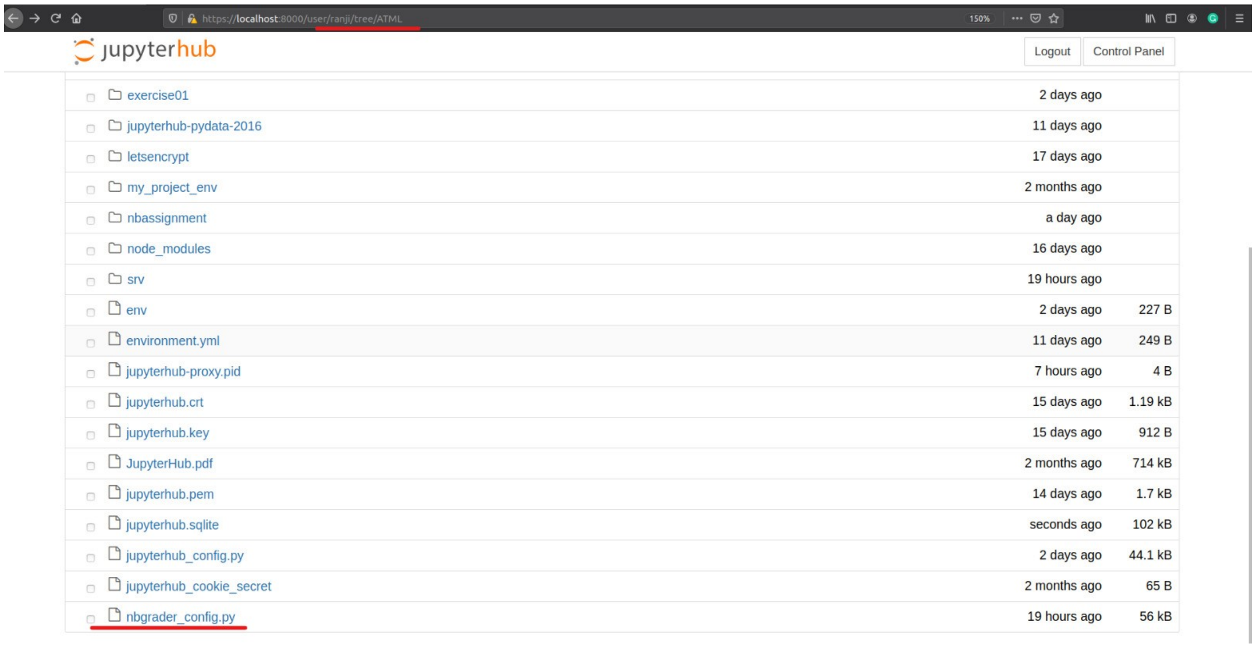
0 / ATML

Name	Last Modified	File size
..	seconds ago	
configurable-http-proxy	15 days ago	
course101	8 days ago	
dunder_methods	a day ago	

```
from nbgrader.utils import get_username

c=get_config()
c.CourseDirectory.course_id="<name-of-your-course>"
c.CourseDirectory.db_assignments=[dict(name="<name-of-your-assignment>")]
c.Exchange.root = "/home/user/srv/nbgrader/exchange"
c.Exchange.db_students=[dict(id=get_username()]
```

This file must be residing where your root/home directory from where you are executing your commands form:



exercise01	2 days ago	
jupyterhub-pydata-2016	11 days ago	
letsencrypt	17 days ago	
my_project_env	2 months ago	
nbassignment	a day ago	
node_modules	16 days ago	
srv	19 hours ago	
env	2 days ago	227 B
environment.yml	11 days ago	249 B
jupyterhub-proxy.pid	7 hours ago	4 B
jupyterhub.crt	15 days ago	1.19 kB
jupyterhub.key	15 days ago	912 B
JupyterHub.pdf	2 months ago	714 kB
jupyterhub.pem	14 days ago	1.7 kB
jupyterhub.sqlite	seconds ago	102 kB
jupyterhub_config.py	2 days ago	44.1 kB
jupyterhub_cookie_secret	2 months ago	65 B
nbgrader_config.py	19 hours ago	56 kB

And the second config file must be present inside your **/home/user/anaconda3/etc/jupyter/** which defines a global setting and it must contain the following two lines:

```
c=get_config()
c.CourseDirectory.course_id='exercise01'
```

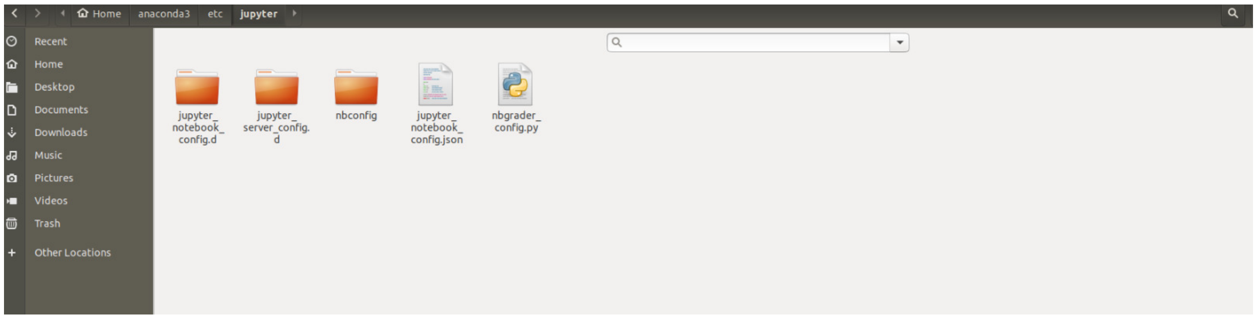
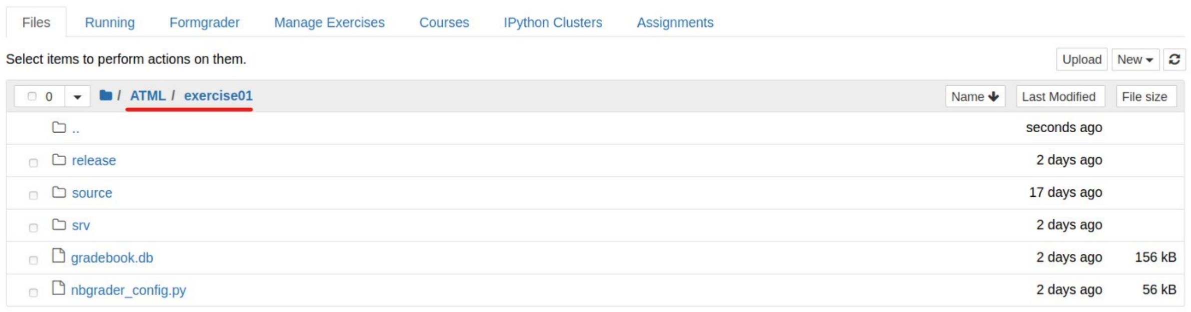


Figure 1: Location of the second config file

Creating assignments from the command line

From inside the course directory now individually create assignments:

```
nbgrader generate_assignment <"name-of-your-assignment">
```



Select items to perform actions on them.

/ ATML / exercise01

..	seconds ago	
release	2 days ago	
source	17 days ago	
srv	2 days ago	
gradebook.db	2 days ago	156 kB
nbgrader_config.py	2 days ago	56 kB

Also put a **.ipynb** file inside that so that it becomes available to students for fetching and submitting assignments.

Now from the same directory as above **release the assignment** and execute as:

```
nbgrader release_assignment <"name-of-your-assignment">
```

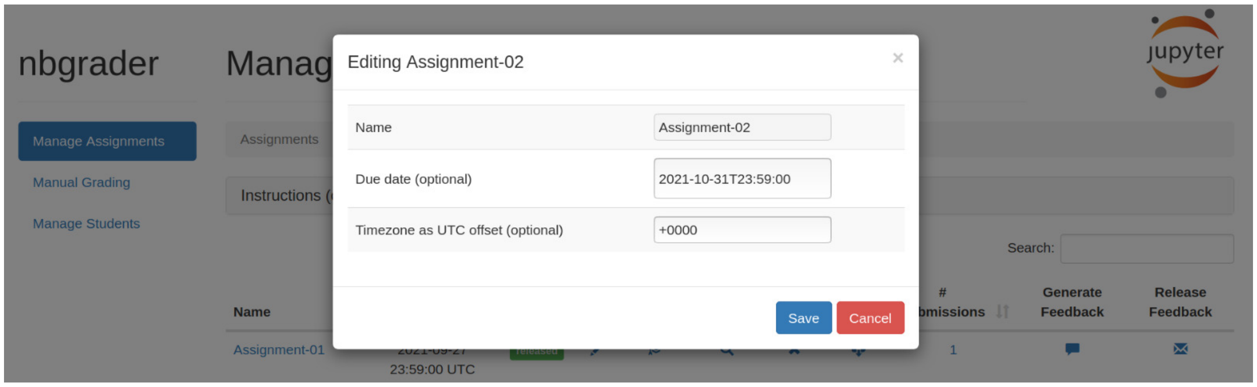
After students submit their assignments then **autograde** the assignment by:

```
nbgrader autograde <"name-of-your-assignment">
```

Creating assignments from the `formgrader` UI

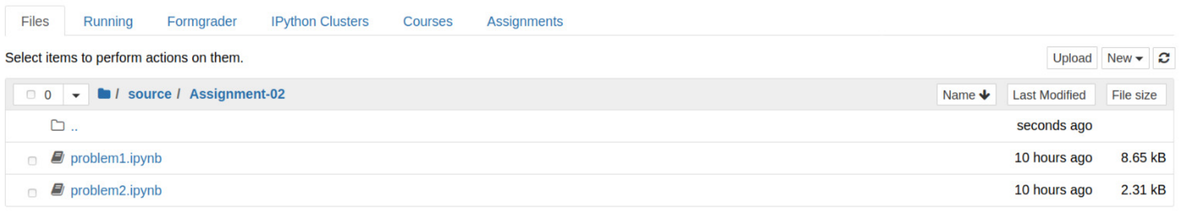
Click on + **Add new assignment** (*also you can edit metadata of previously created assignment.*)

make sure in your `/source/Assignment-02/` you have the **.ipynb** files. You must have atleast one of the .ipynb files so as to generate the assignment and then only the **release & collect** functionality would be available.



Editing Assignment-02

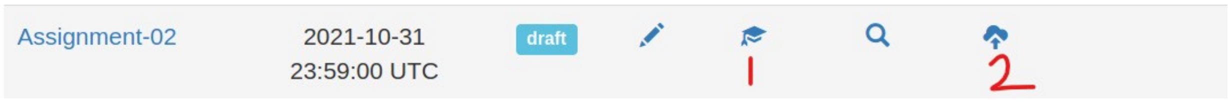
Name	Assignment-02
Due date (optional)	2021-10-31T23:59:00
Timezone as UTC offset (optional)	+0000



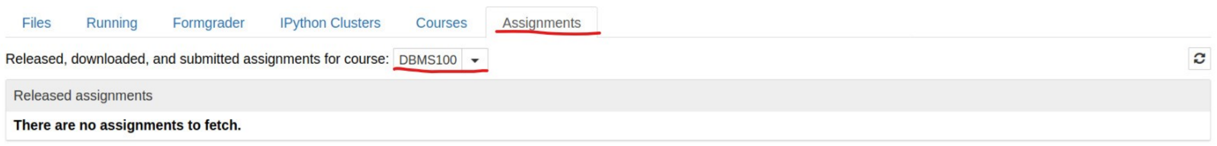
Select items to perform actions on them.

/ source / Assignment-02

..	seconds ago	
problem1.ipynb	10 hours ago	8.65 kB
problem2.ipynb	10 hours ago	2.31 kB



Assignment-02	2021-10-31 23:59:00 UTC	draft	



Released, downloaded, and submitted assignments for course: DBMS100

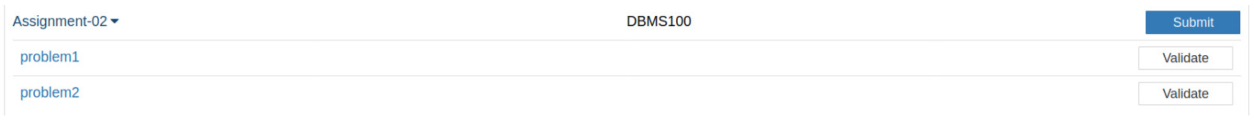
Released assignments

There are no assignments to fetch.

Then click on **Generate** & **Release** the student version of the assignment.

Then fetch the assignments under the **Assignments** tab

Then instructors will have the list of all **Downloaded assignments**. You must validate and then submit them for autograding.



References

[1] https://nbgrader.readthedocs.io/en/stable/user_guide/managing_assignment_files.html

[2] https://jhubdocs.readthedocs.io/en/latest/configurable-http-proxy/README.html

[3] https://jupyterhub.readthedocs.io/en/stable/troubleshooting.html

[4] https://evanwill.github.io/_drafts/notes/dual-python-notebook.html

[5] http://ipython.org/ipython-doc/dev/parallel/parallel_intro.html

[6] https://nbgrader.readthedocs.io/en/stable/user_guide/installation.html