nbgrader (and Jupyter Hub)

Notes

Jupyter Hub

- default is to authenticate with OS user.
- can authenticate via github login…
- runs on Python 3, can contain a Python 2 core.
- https://github.com/jupyter/jupyterhub
- and, add npm install.
- to run hub, need to open port 8000 in firewall, or have it listen on a port that is open. It also uses 8080 and 8081, but only over local loopback, as I understand it.

Installation and Configuration

- links:
  - https://github.com/jupyter/jupyterhub
  - https://github.com/jupyter/jupyterhub/blob/master/docs/getting-started.md
  - http://ipython.org/ipython-doc/dev/development/config.html
  - google groups Jupyter forum: https://groups.google.com/forum/#!forum/jupyter
- For easiest installation and configuration, installation has to be done as root (use sudo) and jupyterhub has to run as root.

dependencies:

- Install NPM (node.js package manager) and nodejs-legacy:

```
sudo apt-get install npm nodejs-legacy
```

- Install javascript dependencies:

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

- make sure that Python 3 and pip3 are installed
  - documentation suggest that you install pip3 from the operating system's package repository (ubuntu example below):

    ```
    sudo apt-get install python3-pip
    ```

  - I would not do this (it can overwrite newer pip with older if package updates). I would install pip manually, in each python installation. The way that https://pip.pypa.io/en/latest/installing.html outlines:
    - download the get-pip.py install script from http://pip.readthedocs.org/en/latest/installing.html
    - if you have python 3 installed, first install python 3 pip using python3 to run get-pip.py.
    - install python 2 pip using python2 to run get-pip.py.
    - whatever version of python you want the base `pip` executable to reference, install that version of pip last. If you just need to install pip3, you might have to then re-install pip in python 2 to get `pip` to be the python 2 `pip`.

Installation

- Install jupyterhub using pip3:

  ```
  sudo pip3 install jupyterhub
  ```

- `npm install` step should only be if you are installing from source.

Starting jupyterhub

- choose a directory where you want jupyterhub's data, configuration files, and database to live (`<jupyterhub_home>`). For nbgrader, this must be writable by the user that will start the nbgrader formgrade application, so put it somewhere where you are comfortable with that.

- cd into the `<jupyterhub_home>` directory (on my server, `~/work/jupyter/jupyterhub_data`, could be in `/etc/jupyterhub`, etc.):

  ```
  cd ~/work/jupyter/jupyterhub_data
  ```

- at the command line, run `jupyterhub` as root:

```
sudo jupyterhub
```

- eventually, you might want to install the jupyterhub as a service. To do so in systems using upstart (didn't try systemd yet):
    - start with the following init script template: https://github.com/fhd/init-script-template
    - copy this template to make an init.d/upstart script for jupyterhub and store it inside your jupyterhub folder
        - `<jupyterhub_home>/bin/jupyterhub_init.sh`
        - follow the README's instructions for configuring the script.
        - In addition:
            - I hard-coded the `name` variable to "jupyterhub", so it didn't use different names for different runlevel scripts, etc.
            - altered log file location so they go to `/var/log/jupyterhub`, rather than dumping directly into `/var/log`.
            - update the upstart information at the top, between `### BEGIN INIT INFO` and `### END INIT INFO`.
    - make link to this script named jupyterhub in /etc/init.d

      ```
      cd /etc/init.d
      sudo ln -s <jupyterhub_home>/bin/jupyterhub_init.sh jupyterhub
      ```

    - add links to /etc/rc*.d folders using update-rc.d:

      ```
      sudo update-rc.d jupyterhub defaults
      ```

    - with this setup:
        - pid of running jupyterhub will be in /var/run/jupyterhub.pid
        - logs go to /var/log/jupyterhub/jupyterhub.err and jupyterhub.log
        - start and stop work, restart really doesn't. Just stop, then start. And, it is cleanest to stop the hub using the admin in the jupyterhub, where you can manage individual servers and the proxy server as well as the jupyterhub itself.

Basic configuration

- in your jupyterhub data and configuration folder, generate a default configuration file:

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

- in this file, you can configure the IP address and port of the computer you are running jupyterhub on. This is optional, and both setups I've done so far have needed neither.

```
# The public facing port of the proxy
# c.JupyterHub.port = 8000
c.JupyterHub.port = 16386
...
# The public facing ip of the proxy
# c.JupyterHub.ip = ''
c.JupyterHub.ip = ''
```

- configure the CONFIGPROXY_AUTH_TOKEN inside this configuration file so you don't have to mess around with setting environment variables should you use a more advanced start/stop script.

```
# The Proxy Auth token.
#
# Loaded from the CONFIGPROXY_AUTH_TOKEN env variable by default.
# c.JupyterHub.proxy_auth_token = ''
c.JupyterHub.proxy_auth_token = '<CONFIGPROXY_AUTH_TOKEN>'
```

- add an administrator user and make user whitelist if desired.

```
# set of usernames of admin users
#
# If unspecified, only the user that launches the server will be admin.
# c.Authenticator.admin_users = traitlets.Undefined
c.Authenticator.admin_users = [ 'jonathanmorgan' ]

# Username whitelist.
#
# Use this to restrict which users can login. If empty, allow any user to
# attempt login.
# c.Authenticator.whitelist = traitlets.Undefined
c.Authenticator.whitelist = [ 'jonathanmorgan' ]
```

Set up SSL

- create self-signed certificate for server. In your data and configuration directory:

```
openssl req -x509 -newkey rsa:2048 -keyout jupyterhub_key.pem -out jupyterhub_cert.pem -days <num_days> -noc
```

WHERE:

  - -nodes = no password (removes Triple-DES encryption of key, which requires a password to be entered to unencrypt key).
  - -newkey rsa:2048 = type of encryption key, and number of bits of encryption.
  - <num_days> = integer number of days from today the certificate is valid for. 3650 = 10 years (effectively forever - this should be obsolete in 10 years)

- configure jupyterhub so it knows of the location of certificate and public key.

```
# Path to SSL certificate file for the public facing interface of the proxy
#
# Use with ssl_key
# c.JupyterHub.ssl_cert = ''
c.JupyterHub.ssl_cert = './jupyterhub_cert.pem'
...
# Path to SSL key file for the public facing interface of the proxy
#
# Use with ssl_cert
# c.JupyterHub.ssl_key = ''
c.JupyterHub.ssl_key = './jupyterhub_key.pem'
```

Network configuration

- open port 8000 in firewall (ubuntu ufw example):

```
sudo ufw allow 8000
```

- connect
    - if no SSL: http://<server_domain_or_ip>:8000
    - if SSL: https://<server_domain_or_ip>:8000

Installing Python 2 kernel

- likely will be doing this at OS level, so will have to either be root or use sudo.

- In python 2, use pip or pip2 (for clarity, using pip2 in these examples) to install ipython and ipython[notebook].

```
sudo pip2 install --upgrade ipython
sudo pip2 install --upgrade ipython[notebook]
```

- In python 3, just make sure that ipython and ipython[notebook] are already installed (at this point, they should be).

```
sudo pip3 install --upgrade ipython
sudo pip3 install --upgrade ipython[notebook]
```

- For Python 2, run the following command to install the Python 2 kernel.

```
sudo python2 -m IPython kernelspec install-self
```

  This will install the python 2 kernel in /usr/local/share/jupyter/kernels/python2.

- If you also want/need to install the Python 3 kernel (I did):

```
sudo python3 -m IPython kernelspec install-self
```

jupyterhub shared kernels and extensions:

- located in `/usr/local/share/jupyter`
  - `/hub` - contains the shared space for JupyterHub sessions, things that are available to/used by all users.
  - `/kernels` - contains all the kernels available for users of your jupyterhub.
  - `/nbextensions` - contains any global ipython notebook extensions. Can be placed here, or can be links, linked to the place where the extension is installed and updated, so updates are automatically propagated.

nbgrader

- links:
  - https://github.com/jupyter/nbgrader
  - http://nbgrader.readthedocs.org/en/latest/

Installation and Configuration

Prerequisites

- install ipywidgets Python 3 package

```
sudo pip3 install ipywidgets
```

- install nose package in all versions of python for which you will be making notebooks that will need to be tested.

```
sudo pip2 install nose
sudo pip3 install nose
```

Basic Installation

- basic install doc: https://github.com/jupyter/nbgrader
- not using virtualenvs to start, so all installation requires you to be root.
- Install nbgrader in Python 3

```
sudo pip3 install nbgrader
```

  - This also installs `nbgrader` command line tool, run at the unix shell prompt.

- Install the nbgrader assignment toolbar extension for ipython - :

  ```
  (sudo) nbgrader extension install --symlink
  (sudo) nbgrader extension activate
  ```

  - WHERE:
    - `--symlink` = install using a symlink, such that the extension is upgraded whenever nbgrader is. If you leave this off, you install a copy, not a link, and you have to update it each time you update nbgrader.
    - `--help-all` = add this to the end to see all options you can pass to each command.
    - not sure if you need sudo or not - looks like the initial install does and the activate doesn't require root.

- And, for each user that needs these things, as that user, also run:

  ```
  nbgrader extension activate
  ```

Default nbgrader_config.py

Each user that runs the nbgrader command must have some nbgrader configuration, to tell it which course is active and where the exchange directory, used to coordinate assignments between students and instructor, is located.

The most basic of these files should look something like this:

```
# pull in config.
c = get_config()

# set exchange directory path (this is the default, setting it to be safe).
c.TransferApp.exchange_directory = '/srv/nbgrader/exchange'

# set the course label.
c.NbGrader.course_id = "2015-fall-big_data"
```

You can provide this nbgrader configuration in a number of ways:

- Default, system-wide configuration file located at `/etc/jupyter/nbgrader_config.py`. This will apply to all users who run the nbgrader command.
- User-level configuration file, located at `~/.jupyter/nbgrader_config.py`. Properties defined here take precedence over properties defined in `/etc/jupyter/nbgrader_config.py`.
- `nbgrader_config.py` located in the current folder, where you run nbgrader. These properties take precedence over any in the user-level or default configuration files.

In our setup, we made a default file in `/etc/jupyter/nbgrader_config.py` for ease of management of changes.

Directory setup

In order for nbgrader to work with jupyterhub, you will need to set up two directories:

- an **exchange directory** to hold directories and files used to coordinate work with students.
- a **course directory** to hold assignments and working folders for instructors and grading.

For students, no directory setup is necessary. When a student downloads an assignment, each assignment folder is placed in the student's home directory and interacted with there.

Exchange directory

The exchange directory is where assignments are released, where students fetch assignments from, and from which instructors/graders collect and then grade assignments. It needs to be world readable and writable. Its default location is `/srv/nbgrader/exchange`. If you use other than this default, you'll need to make sure to set `c.TransferApp.exchange_directory` in such a way that all nbgrader configurations know of the location.

Our setup:

- created this directory at the default path: /srv/nbgrader/exchange
- owner and group are root, permissions are 777.

Course directory

The course directory is a central directory for making, releasing, collecting, and grading assignments. Given how nbgrader works now, this directory needs to live inside one user's home directory. This user will be the "instructor". On our system, we created an actual "instructor" user, so that anyone with credentials can use this account to manage assignments.

This directory is the place where the instructor runs all of the nbgrader commands for assigning, releasing, collecting, and auto-grading assignments. It is also the folder from which you'll launch the formgrade application.

To set up your course directory:

- decide which user is the lucky user to be the "instructor". We created a user and group named "instructor" for this purpose.

- inside the "instructor" user's home directory, create a directory to serve as your course directory, the path of which will subsequently be referred to as `<course_directory>`.

    - In our setup , we created `~/nbgrader/courses/<course_name>` as the course directory.

- inside the `<course_directory>`:

    - create a directory named `source`. This is the place where you will place your assignment notebooks.

        - Inside the source directory, each assignment will have its own directory. The name of each assignment's directory will serve as the assignment's identifier throughout the nbgrader process, visible to students at some points, so choose carefully.

        - inside each assignment directory, any notebooks within the directory are considered components of the assignment. You can have other files or directories, as well, but the only ones nbgrader will concern itself with are notebooks. If you have notebooks further down in folders, they will be ignored.

        - for our install, we created a github repository to serve as the source folder that we cloned as "source", rather than making the source folder from scratch. This allows us to use github to coordinate work on assignment notebooks outside our server and then periodically pull the latest into our `<course_directory>`. Example:

```
git clone https://github.com/CSSIP-AIR/Big-Data-Workbooks source
```

- create an `nbgrader_config.py` configuration file, to be used by the "instructor" user when running the nbgrader command and when starting the formgrade application. We'll configure this in a later step, but know that you need this file in this directory, and for reference, here is what our file eventually contained:

```
c = get_config()
c.FormgradeApp.ip = "127.0.0.1"
c.FormgradeApp.port = 9000
c.FormgradeApp.authenticator_class = "nbgrader.auth.hubauth.HubAuth"
c.HubAuth.hub_address = '<public_jupyterhub_domain_name_or_IP>'
c.HubAuth.notebook_url_prefix = "nbgrader/courses/2015-fall-big_data"
c.HubAuth.graders = [ "instructor" ]
c.HubAuth.generate_hubapi_token = True
c.HubAuth.hub_db = "<jupyterhub_home>/jupyterhub.sqlite"
```

- we also created an IPython notebook named `manage_assignments.ipynb` that holds code that needs to be run in Python, separate from the nbgrader command, for setup, and also contains instructions for end-to-end processing of assignments and for troubleshooting issues with the formgrade application and with ipython assignment notebooks. It should always be run in the `<course_directory>`, throuhh jupyterhub, by the "instructor" user.

- For reference, this notebook is in our "source" repository, in the root of that repository (https://github.com/CSSIP-AIR/Big-Data-Workbooks/blob/master/manage_assignments.ipynb). You can either download it and use it as you please, or if you clone that repository, you can just copy it up one level from `source` into the actual `<course_directory>`.

Configure jupyterhub_config.py

- jupyterhub integration guide: http://nbgrader.readthedocs.org/en/latest/user_guide/11_jupyterhub_config.html

- In jupyterhub_config.py, if you'd like, add graders or instructor as administrators, to whitelist.

```
# set of usernames of admin users
#
# If unspecified, only the user that launches the server will be admin.
# c.Authenticator.admin_users = traitlets.Undefined
c.Authenticator.admin_users = [ 'jonathanmorgan', 'instructor' ]

# Username whitelist.
#
# Use this to restrict which users can login. If empty, allow any user to
# attempt login.
# c.Authenticator.whitelist = traitlets.Undefined
c.Authenticator.whitelist = [ 'jonathanmorgan', 'instructor' ]
```

Configure nbgrader_config.py

- you'll need to configure the `nbgrader_congif.py` in `<course_directory>` so you can launch the formgrade application used to provide feedback on assignments. Instructions for configuring this file: http://nbgrader.readthedocs.org/en/stable/user_guide/11_jupyterhub_config.html#configuring-nbgrader-formgrade. Here is what ours ended up looking like:

  ```
  c = get_config()

  #------------------------------------------------------------------------------
  # nbgrader and Form Grader app configuration
  #------------------------------------------------------------------------------

  # nbgrader configuration
  c.NbGrader.course_id = "2015-fall-big_data"
  c.TransferApp.exchange_directory = "/srv/nbgrader/exchange"

  #c.FormgradeApp.ip = "127.0.0.1" # use default
  c.FormgradeApp.port = 9000
  c.FormgradeApp.authenticator_class = "nbgrader.auth.hubauth.HubAuth"

  # Set the HubAuth hub_address to the public address the jupyterhub should
  #    be referred to in links from the formgrade application.
  c.HubAuth.hub_address = '<public_jupyterhub_domain_name_or_IP>'

  # Change this to be the path to the user guide folder in your clone of
  # nbgrader, or just wherever you have your class files. This is relative
  # to the root of the notebook server launched by JupyterHub, which is
  # probably your home directory.
  c.HubAuth.notebook_url_prefix = "nbgrader/courses/2015-fall-big_data"

  # Change this to be the list of unix usernames that are allowed to access
  # the formgrade application.
  c.HubAuth.graders = [ 'jonathanmorgan', 'instructor' ]

  # This specifies that the formgrader should automatically generate an api
  # token to authenticate itself with JupyterHub.
  c.HubAuth.generate_hubapi_token = True

  # Change this to be the jupyterhub.sqlite located in the directory where
  # you actually run JupyterHub.
  c.HubAuth.hub_db = "<jupyterhub_home>/jupyterhub.sqlite"
  ```

- If you configure the jupyterhub to use SSL, you'll need to also configure the formgrade application so it knows and interacts with the hub using SSL. Not sure what that entails at this point.

Running the formgrade application

Here is how we started up and managed the formgrade application.

To start the formgrade application:

- log in as instructor

- cd to the `<course_directory>`. Example:

  ```
  cd nbgrader/courses/2015-fall-big_data
  ```

- start GNU screen

```
screen
```

- export environment variable CONFIGPROXY_AUTH_TOKEN, setting it to the same value you placed in the `jupyterhub_config.py` file:

```
export CONFIGPROXY_AUTH_TOKEN='<CONFIGPROXY_AUTH_TOKEN>'
```

- run `nbgrader formgrade`

```
nbgrader formgrade
```

- If it starts OK, you'll see something like:

```
$ nbgrader formgrade
[FormgradeApp | INFO] Proxying /hub/nbgrader/2015-fall-big_data --> http://127.0.0.1:9000
[FormgradeApp | INFO] Serving MathJax from /usr/local/lib/python3.4/dist-packages/notebook/static/components
[FormgradeApp | INFO] Form grader running at http://127.0.0.1:9000/
[FormgradeApp | INFO] Use Control-C to stop this server
* Running on http://127.0.0.1:9000/ (Press CTRL+C to quit)
```

- To access the formgrade application, you will combine the URL you use to access the base jupyterhub and combine it with the path from the "Proxying" line above. For example:

```
http://<DNS name>:8000/hub/nbgrader/2015-fall-big_data
```

The formgrade application does not need to be started as root BUT - to get grading app to run, the "instructor" user must be able to write to the `<jupyterhub_home>` and to file `<jupyterhub_home>/jupyterhub.sqlite`.

troubleshooting

CONFIGPROXY_AUTH_TOKEN mismatch between jupyterhub and formgrade

When you start the formgrade application, if you see an exception stack trace that ends like this:

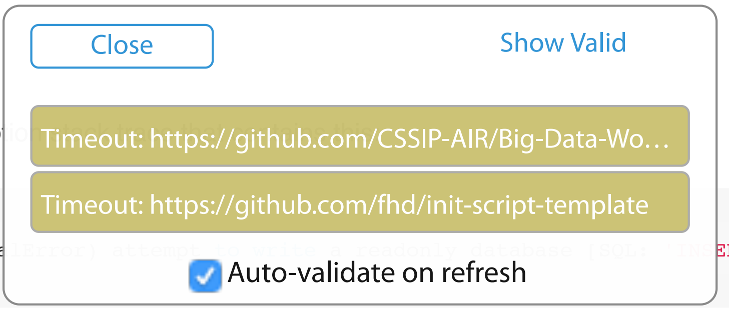
```
raise Exception('Error while trying to add JupyterHub route. {}: {}'.format(response.status_code, response.text)
Exception: Error while trying to add JupyterHub route. 403:
```

make sure that you are explicitly setting the CONFIGPROXY_AUTH_TOKEN for the jupyterhub server (preferably in the c.JupyterHub.proxy_auth_token variable in jupyterhub_config.py), and that you are exporting that same value before starting the formgrade app.

“attempt to write a readonly database”

When you start the formgrade application, if you see an excep

```
sqlalchemy.exc.OperationalError: (sqlite3.OperationalError) attempt to write a readonly database [SQL: 'INSERT I
```



Timeout: https://github.com/CSSIP-AIR/Big-Data-Wo…

Timeout: https://github.com/fhd/init-script-template

[✓] Auto-validate on refresh

This means your jupyterhub.sqlite database is not able to be written to by the user used to start the formgrade application. Try adjusting the permissions so that the user who starts formgrade can write to the database.

“unable to open database file” api_tokens

When you start the formgrade application, if you see an exception stack trace that contains this:

```
sqlalchemy.exc.OperationalError: (sqlite3.OperationalError) unable to open database file [SQL: 'INSERT INTO api_
```

This is likely because the folder that contains your jupyterhub.sqlite database file is not writeable by the user used to start the formgrade application (sqlite writes temp files in the directory where the database it is using lives). Try adjusting the permissions on the directory that contains jupyterhub.sqlite so that the user who starts formgrade can write to it.

URLs on formgrade pages refer to proxy IP address, not public IP or domain name

If you get past the above issues and can authenticate, but when the page loads, it is unstyled and all the links point to localhost:8000 or 127.0.0.1:8000 rather than the actual public address of your server (including javascript and CSS, which won’t load, so the page will look unstyled and won’t work right), then you need to make sure that the `c.HubAuth.hub_address` property is set to the public IP address or domain name for your server in `<course_directory>/nbgrader_config.py` for the instructor user.