NST Physics and nbgrader

Niall McConville

PyCav Project

July 2016

Contents

Chapter 1

Introduction

nbgrader[1] is a system which allows for the use of Jupyter[2] notebooks as a form of assessment material.

Lecturers (or at least, their lackeys[3]) can create a set of exercises contained within these notebooks. They can be distributed to students using the JupyterHub[4] server.

From there, students can complete the computational exercises and return them (hopefully completed) to the lecturers. The lecturers can autograde the exercises and manually mark aspects which do not lend themselves easily to a machine marking them (such as graphs).

This document outlines how the PyCav project set up nbgrader for use within the Physics courses within the Natural Sciences Tripos.

Another source of documentation for nbgrader can be found here: `http://nbgrader.readthedocs.io/`.

[1] `https://github.com/jupyter/nbgrader`

[2] `http://jupyter.org/`

[3] `https://pycav.github.io/about/`

[4] `https://github.com/jupyterhub/jupyterhub`

Chapter 2

Setup

2.1 JupyterHub

2.1.1 Background

JupyterHub is a system which allows the hosting of a multi-user server which allows users to log in to and create Jupyter notebooks without hosting a server for themselves.

Docker[1] is effectively a sandboxing system which allows JupyterHub to create single user Jupyter servers inside a set of isolated environments.

The idea for this project was to set up an environment where students could experiment with Python without having to install any software locally. The above were selected as they were able to provide this environment.

2.1.2 Raven Authentication

All Cambridge students and staff have access to a crsid. This allows them to authenticate using the Raven[2] service.

To extend this functionality to JupyterHub, an authentication plugin jupyterhub-raven-auth[3] was written. It was used throughout the project.

[1] `https://www.docker.com/`

[2] `https://raven.cam.ac.uk/`

[3] `https://github.com/PyCav/jupyterhub-raven-auth`

2.1.3 Docker Configuration

The isolation provided by Docker is useful for running an assessment environment. It allows for caps to be placed on usage, especially in the instances of runaway code. Students are unable to view other student's files in a way that goes beyond setting access rights. It also allows for the simple addition of read only volumes, which we have exploited to share Demonstrations for everyone to see.

Docker images are built from Dockerfiles (think, Makefiles). These images are called by the Dockerspawner in the JupyterHub config. The Dockerspawner creates *containers* which take up file space. It is possible to mount volumes in an NFS setup, using the {username} filter.

In the Dockerfile provided below, one can see the extent of the customisation we provide. Notably, nbgrader is installed in each container.

Docker containers (and their 'real' mounted volumes) should be continually backed up. We (as of July 28, 2016) have not considered what the total size of such a system would be. For the 'Computational Models' course at Berkley, CA a 3 TB NFS was used for storage[4].

We have also set up the containers to execute the 'start-singleuser.sh' shell script. This contains code to create a new user, whose username matches the crsid of the individual logged into the JupyterHub. This is required as nbgrader will use this username in the filenames of submitted coursework.

[4] `https://github.com/compmodels/jupyterhub-deploy`

Dockerfile:

```
# Build as jupyterhub/singleuser
# Run with the DockerSpawner in JupyterHub

FROM jupyterhub/singleuser

MAINTAINER jordan <jo357@cam.ac.uk>

EXPOSE 8888

USER root
RUN echo "deb http://ftp.debian.org/debian jessie-backports main" \
    >> /etc/apt/sources.list
RUN apt-get -y update
RUN apt-get -t jessie-backports  -y install ffmpeg
RUN pip3 install vpython
RUN pip3 install pycav
RUN pip3 install nbgrader
RUN nbgrader extension install
RUN nbgrader extension activate

# Prep. to replace the jovyan user with the crsid
RUN userdel jovyan
ENV SHELL /bin/bash

ADD pycav-start.sh /srv/pycav/pycav-start.sh

# Execute this script on startup
CMD ["sh", "/srv/pycav/pycav-start.sh"]
```

start-singleuser.sh file:

```
#!/bin/bash

set -e
if getent passwd $JPY_USER > /dev/null ; then
 echo "$JPY_USER␣exists"
else
 echo "Creating␣user␣$JPY_USER␣(9002)"
 useradd -u 1000 -s $SHELL $JPY_USER
fi

notebook_arg=""
if [ -n "${NOTEBOOK_DIR:+x}" ]
then
    notebook_arg="--notebook-dir=${NOTEBOOK_DIR}"
fi

sudo touch /home/jovyan/work/.nbgrader.log
sudo chmod 666 /home/jovyan/work/.nbgrader.log

sudo -E PATH="${CONDA_DIR}/bin:$PATH" -u $JPY_USER jupyterhub-singleuser \
 --port=8888 \
 --ip=0.0.0.0 \
 --user=$JPY_USER \
 --cookie-name=$JPY_COOKIE_NAME \
 --base-url=$JPY_BASE_URL \
 --hub-prefix=$JPY_HUB_PREFIX \
 --hub-api-url=$JPY_HUB_API_URL \
 ${notebook_arg} \
 $@
```

2.1.4 The JupyterHub Config

```
"""
jupyterhub_config.py

The pycav project jupyterhub config
"""

c = get_config()

"""
Imports

os : for accessing absolute paths
pycav_tis : for interfacing with the Teaching Information System
"""
import os
from pycav_tis import tis

# Setup for the TiS module
# TODO: Replace DictReader with SQL system when appropriate
tis_config = '/home/public/tis_config'
tis_csv = '/home/public/tis.csv'
tis_conn = tis.pycavTisDictReader(tis_csv)

c.NotebookApp.open_browser = False

# Enable Logging
c.JupyterHub.log_level = 'DEBUG'
c.JupyterHub.port = 8000

# SSL
c.JupyterHub.ssl_key = '/etc/letsencrypt/live/pycav.ovh/privkey.pem'
c.JupyterHub.ssl_cert = '/etc/letsencrypt/live/pycav.ovh/fullchain.pem'

# Cookies
c.JupyterHub.proxy_auth_token='you-need-to-set-this-see-the-docs'
```

```
c.JupyterHub.cookie_secret_file = '/home/jordan/jupyterhub_cookie_secret'

# Users
c.JupyterHub.db_url = '/home/jordan/jupyterhub.sqlite'
c.JupyterHub.admin_access = True

c.Authenticator.admin_users = tis_conn.get_admins()
c.Authenticator.whitelist = tis_conn.get_users()

# Ucam Authentication
from raven_auth.raven_auth import RavenAuthenticator
c.JupyterHub.authenticator_class = RavenAuthenticator
c.RavenAuthenticator.description.value = ”pyCav”
c.RavenAuthenticator.long_description = ”The␣pyCav␣Jupyterhub␣server.”
c.RavenAuthenticator.login_logo = '/home/public/py_cav.jpg'

# Docker
from dockerspawner import DockerSpawner
c.Spawner.debug = True
c.JupyterHub.spawner_class = DockerSpawner
#c.DockerSpawner.container_prefix = pycav
c.DockerSpawner.read_only_volumes={'/home/public/demos':
'/home/jovyan/work/demos',
'/home/public/crsidify':'/srv/crsidify'}
c.DockerSpawner.volumes={'/srv/nbgrader':'/srv/nbgrader'}
c.DockerSpawner.extra_create_kwargs.update({
    'command': 'sh␣/srv/crsidify/start-singleuser.sh'
})
c.DockerSpawner.container_image = ”jordanosborn/pycav”

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

2.2 nbgrader

2.2.1 Background

nbgrader is a tool which allows for the assignment and grading of Jupyter notebooks. It can operate in numerous ways, the simplest of which is the manual distribution and collection of notebook assignments. The PyCav project uses a combination of nbgrader and JupyterHub to automate various aspects of the process.

The documentation for nbgrader can be found here: `http://nbgrader.readthedocs.io/`.

2.2.2 Installation

The nbgrader was setup in two places,

1. The pycav server (the local server, where the JupyterHub is hosted).
2. Within the Docker containers.

The installation on the local server was for the utilisation of nbgrader's *assign*, *collect*, and *formgrade* applications. These are to be run by the teaching staff, and will be discussed in a later chapter.

The installation inside the Docker containers was for nbgrader's Jupyter extension, which calls nbgrader's *fetch* and *submit* applications. These will be run by the extension (and so not directly by students).

Upon installation within a Docker image, a file called '.nbgrader.log' is created within the '/home/jovyan/-work/' directory. As mentioned in the previous chapter, this *must* have write access for the user or else the nbgrader functionality will not work.

2.2.3 Directory Structure

One can posit that there are effectively two sides to nbgrader directories: course directories and the distribution directory. These are both present on the local server.

Central to the automation of distribution is the *exchange* folder. Typically this is stored in the directory '/srv/nbgrader/exchange', although one can specify alternative directories. The exchange directory **must** be world readable and writable.

Although any folder can be specified as the exchange folder, when mounting it to a Docker container, it is simplest to mount it to the container directory such that it appears as '/srv/nbgrader/exchange'. The default behaviour of nbgrader is to check this directory, with further (possibly complicated) configuration required to change this.

Course directories provide a structure for the courses that can be provided. They will be elaborated upon in the next chapter.

2.2.4 JupyterHub Integration

2.2.5 Teaching System Integration

Currently a work in progress. There will be a module which allows nbgrader to interface with the TiS.

2.3 Server

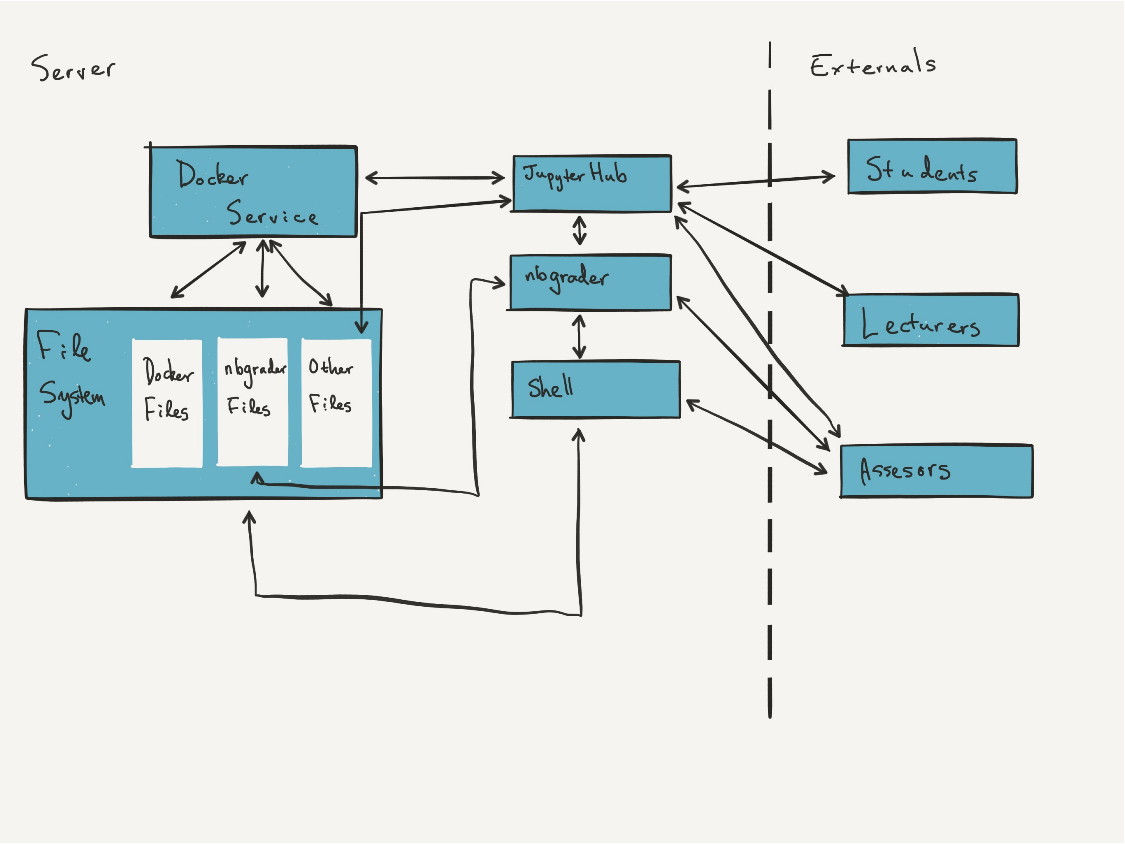

Figure 2.1: The interactions between the various bits of software in relation to the users and the servers. Note that Assessors should be able to access software beyond the JupyterHub, for example, the nbgrader formgrade interface (which in turn relies on JupyterHub for authentication). Not depicted: The nbgrader installation **inside** the Docker containers.

Chapter 3

Courses

3.1 Courses

TODO: Add bits about TiS DB structure?

3.1.1 Directory Structure

The primary folder of any course is the {course} directory folder. To create any course, you must start with one of these folders. It contains a few files (some of which are created when nbgrader is run for the first time),

1. {course}/gradebook.db
2. {course}/nbgrader_config.py

The gradebook stores information about the types of assignment which are pertinent to the course. The nbgrader_config will be elaborated upon in a later section.

nbgrader commands should be run from the {course} directory.

In order to add assignments to a course, you should create a {course}/source directory, with folders corresponding to each assignment (each assignment can contain multiple notebooks).

3.1.2 The nbgrader Config

The nbgrader config file specifies the following information,

1. The course id (used to lookup from the TiS).

2. The students who are taking the course.

3. The graders responsible for the course.

4. The Formgrader configurations (ip, port, JupyterHub integration)

```
import os
from pycav_tis import tis

c = get_config()

# TODO: When creating a new course, change this to the course_id on the TiS
# May have to write a dictionary to look this up because nbgrader automatically converts a
course_id = 'pycav-test'

# TiS
#tis_config = ''
tis_csv = '/home/public/tis.csv'
tis_conn = tis.pycavTisDictReader(tis_csv, course_id)

# Generic nbgrader configs
c.NbGrader.course_id = course_id
#c.TransferApp.exchange_directory = "/home/public/pycav-nbgrader/exchange"
c.NbGrader.db_assignments = [dict(name="ex1")]
c.NbGrader.db_students = tis_conn.get_students()

# Options that are specific to formgrader & integrating it with JupyterHub

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

#
import netifaces
docker0 = netifaces.ifaddresses('docker0')
docker0_ipv4 = docker0[netifaces.AF_INET][0]

# This is the actual URL or public IP address where JupyterHub is running (by
# default, the HubAuth will just use the same address as what the formgrader is
# running on -- so in this case, 127.0.0.1). If you have JupyterHub behind a
# domain name, you probably want to set that here.
# TODO: Convert this into some sort of jupyterhub or shared config
c.HubAuth.hub_base_url = "https://pycav.ovh:8000"
c.HubAuth.hub_port = 8001
c.HubAuth.hubapi_port = 8081
c.HubAuth.hubapi_address = docker0_ipv4['addr']

# Call the TiS to get the graders for this particular course_id
c.HubAuth.graders = tis_conn.get_graders()

c.HubAuth.hubapi_token = os.environ['JPY_API_TOKEN']
```

3.2 Exercises

3.2.1 Creating Exercises

For a full guide on creating exercises: `http://nbgrader.readthedocs.io/en/stable/user_guide/creating_and_grading_assignments.html`.

3.2.2 Assigning Exercises

3.2.3 Collecting Exercises

3.2.4 Marking Exercises

Chapter 4

Conclusion