# JupyterHub and nbgrader in a small multi-class lab environment

Given a bare-metal GPU server in a lab environment installing [nbgrader](https://github.com/jupyter/nbgrader) shouldn’t be too difficult. Especially, if installing the whole system including [JupyterHub](https://jupyter.org/hub) and GPU accelerated machine learning stuff was straight-forward. But it turns out that due to the progress in JupyterHub and JupyterLab development, nbgrader is a bit outdated and its installation requires lots of patching and tweaking.

The purpose of this chapter is to collect all the customization and patching steps required to get nbgrader 0.6.2 running in a multi-class and multi-instructor environment, including integration into JupyterHub and JupyterLab’s classical Jupyter Notebook interface. Basic installation instructions are taken from the [nbgrader documentation.](https://nbgrader.readthedocs.io/en/stable/)

## Development of nbgrader

In past years nbgrader development slowed down and seems to be stuck to some extent. There is no integration into JupyterLab available, but only in the classical Jupyter Notebook interface (in JupyterLab click "Help" > "Launch Classical Notebook"). Also there are lots of open issues. Some hope for this really useful project comes from [creating](https://github.com/jupyter/nbgrader/issues/1335#issuecomment-994088125) [a new maintainer team.](https://github.com/jupyter/nbgrader/issues/1335#issuecomment-994088125)

This lack of progress seems to be the reason for complicated installation. Neighboring projects moved one, while nbgrader did not.

There is an LDAP server in the background which is used for user authentication. User may login via SSH to the server and each user has its own home directory on the server. The nbgrader admin (the one who installs nbgrader and creates new nbgrader courses) has access to everything on the server. In particular, he’s able to create new local user accounts and to change file permissions.

JupyterHub installation was done very similar to [JupyterHub the hard way,](https://github.com/jupyterhub/jupyterhub-the-hard-way) no Kubernetes, no docker. So there is a virtual environment in /home/vuswh/miniconda3/envs/jupyterhub/ containing all hub related installation. Also nbgrader will go there.

## What we want

At least the student facing part of nbgrader should be accessible through the webbrowser. For instructors as much as possible should be manageable via browser. For the nbgrader admin command line is okay, but for everyday work a browser interface is preferred.

We want to use nbgrader for managing assignments for different courses and we want to have different instructors as well as multiple instructors per course.

Autograding should be relatively save, that is, students shouldn’t be able to destroy the system or delete an instructor’s files. It’s possible to run student submissions in a docker container, but this has several drawbacks (complicated configuration, synchronization of Python environments with the outside world,...). So we have to find (and will find) a solution without docker.

## Basic installation

To get nbgrader we install it using pip in the virtual environment of JupyterHub:

sudo /home/vuswh/miniconda3/envs/jupyterhub/bin/python -m pip install nbgrader

There are unresolvable dependencies between nbgrader and JupyterHub due to conflict- ing versions of traitlets and nbconvert. Installing the most recent versions keeps JupyterHub running, but nbgrader will need some patching (see below):

sudo /home/vuswh/miniconda3/envs/jupyterhub/bin/python -m pip install --upgrade

--upgrade-strategy eager traitlets nbconvert

## Global nbgrader configuration and logging

Global (that is, relevant for all users) nbgrader configuration is read from

/home/vuswh/miniconda3/envs/jupyterhub/etc/jupyter/nbgrader\_config.py. So we create this file with the following content:

from nbgrader.auth import JupyterHubAuthPlugin

c.Exchange.path\_includes\_course = True

c.Authenticator.plugin\_class = JupyterHubAuthPlugin c.Exchange.root = '/home/nbgrader\_exchange'

c.NbGrader.logfile = '/home/vuswh/miniconda3/envs/jupyterhub/share/jupyter/nbgrader.log'

This prepares multi-course usage (path\_includes\_course), activates user mangement via JupyterHub (plugin\_class), sets the directory used for file sharing, and activates logging to a file.

We have to create the exchange directory with appropriate permissions:

sudo mkdir /home/nbgrader\_exchange

sudo chmod ugo+rw /home/nbgrader\_exchange

Further, the log file has to be created. Permissions of the log file have to be rather weak, else some components of nbgrader run by non-root users will fail.

sudo touch /home/vuswh/miniconda3/envs/jupyterhub/share/jupyter/nbgrader.log

sudo chmod 666 /home/vuswh/miniconda3/envs/jupyterhub/share/jupyter/nbgrader.log

## Installing Jupyter extensions

Next we install extensions to Jupyter’s web interface to access nbgrader from the web- browser:

sudo /home/vuswh/miniconda3/envs/jupyterhub/bin/jupyter nbextension install

--sys-prefix --py nbgrader

sudo /home/vuswh/miniconda3/envs/jupyterhub/bin/jupyter nbextension enable

--sys-prefix --py nbgrader

sudo /home/vuswh/miniconda3/envs/jupyterhub/bin/jupyter serverextension enable

--sys-prefix --py nbgrader

We disable all extensions not needed by students. Extensions used by instructors will be enabled later on for each instructor separately.

sudo /home/vuswh/miniconda3/envs/jupyterhub/bin/jupyter nbextension disable

--sys-prefix create\_assignment/main

sudo /home/vuswh/miniconda3/envs/jupyterhub/bin/jupyter nbextension disable

--sys-prefix formgrader/main --section=tree

sudo /home/vuswh/miniconda3/envs/jupyterhub/bin/jupyter serverextension disable

--sys-prefix nbgrader.server\_extensions.formgrader

sudo /home/vuswh/miniconda3/envs/jupyterhub/bin/jupyter nbextension disable --sys-prefix course\_list/main --section=tree

sudo /home/vuswh/miniconda3/envs/jupyterhub/bin/jupyter serverextension disable --sys-prefix nbgrader.server\_extensions.course\_list

## JupyterHub configuration

JupyterHub requires some configuration for multi-class nbgrader usage. We have to create a service running the autograding tool. One reason for this is, that this way autograding is accessible to more than one instructor. Another reason is, that student code shouldn’t run in an instructors user account (with access to the instructor’s files).

Configuration is described in nbgraders documentation (outdated). Some fixes for getting nbgrader running with recent JupyterHub versions are suggested in [issue 1533.](https://github.com/jupyter/nbgrader/issues/1533) Here we adapt the proposed fixes to our setting.

In the definition of get\_jupyterhub\_authorization in /home/vuswh/miniconda3/envs/jupyterhub/lib/- python3.8/site-packages/nbgrader/auth/jupyterhub.py replace JUPYTERHUB\_API\_TOKEN by JUPYTERHUB\_API\_TOKEN\_CUSTOM.

Go to JupyterHub’s Hub Control Panel, click the admin button, and create an API token. Then add the following line to your JupyterHub config file:

c.Spawner.environment = { 'JUPYTERHUB\_API\_TOKEN\_CUSTOM':

'0123456789abcdef0123456789abcdef' }

This sends the API token to the spawner environment. So every (!) spawned single-user server may access it, but only servers using nbgrader require it. Note a nice solution, but it works.

In JupyterHub’s config file create a token service for this API token:

c.JupyterHub.services = [

{

'name': 'nbgrader\_token\_service',

'api\_token': '0123456789abcdef0123456789abcdef'

}

]

c.JupyterHub.load\_roles = [

{

'name': 'nbgrader\_token\_role',

'scopes': ['read:users:groups', 'list:services', 'groups', 'admin:users'],

'services': ['nbgrader\_token\_service']

}

]

If there already are services or roles, you have to add the new ones to the existing lists. Lines c.JupyterHub.services = [ and c.JupyterHub.load\_roles = [ should only appear once in the config file. The purpose of such a token service is to set permissions for API requests using this token (see [JupyterHub documentation](https://jupyterhub.readthedocs.io/en/stable/reference/rest.html#updating-to-admin-services) for details).

## Promote regular hub user to instructor

Intructors are hub users with admin access (maybe not necessary) having some nbgrader Jupyter extensions enabled.

Add the username to c.Authenticator.admin\_users in JupyterHub’s config file. Then login to the instructor user’s account via SSH or JupyterLab terminal and run

/home/vuswh/miniconda3/envs/jupyterhub/bin/jupyter nbextension enable --user course\_list/main --section=tree

/home/vuswh/miniconda3/envs/jupyterhub/bin/jupyter serverextension enable --user nbgrader.server\_extensions.course\_list

## Create a new course

To create a new nbgrader course we have to create a new user (the grader) running the autograding tool:

sudo adduser grader-test-course

In JupyterHub’s config file we grant the grader access to the hub by adding its username grader-test-course to c.Authenticator.allowed\_users. Then we create two groups formgrade-test-course and nbgrader-test-course in JupyterHub’s config file:

c.JupyterHub.load\_groups = {

'formgrade-test-course': ['instructor', 'grader-test-course'],

'nbgrader-test-course': []

}

Group names are prescribed by nbgrader. And it’s really formgrade..., not

formgrader...! The second group remains empty.

Now the autograding service has to be configured in the hub config file:

c.JupyterHub.services = [

{

'name': 'test-course',

'url': 'http://127.0.0.1:8101', 'command': ['jupyterhub-singleuser',

'--group=formgrade-test-course', '--debug'],

'user': 'grader-test-course', 'cwd': '/home/grader-test-course',

'api\_token': 'hub token',

# api\_token can be removed (not tested) 'environment': { 'JUPYTERHUB\_API\_TOKEN\_CUSTOM':

'hub token' }

}

]

c.JupyterHub.load\_roles = [

{

'name': 'formgrader-test-course-role', 'groups': ['formgrade-test-course'],

'scopes': ['access:services!service=test-course']

}

]

Remember that only one c.JupyterHub.services line is allowed (same for

load\_roles).

**Log in to the grader account activate relevant nbgrader Jupyter extensions:**

/home/vuswh/miniconda3/envs/jupyterhub/bin/jupyter nbextension enable

--user create\_assignment/main

/home/vuswh/miniconda3/envs/jupyterhub/bin/jupyter nbextension enable

--user formgrader/main --section=tree

/home/vuswh/miniconda3/envs/jupyterhub/bin/jupyter serverextension enable

--user nbgrader.server\_extensions.formgrader

/home/vuswh/miniconda3/envs/jupyterhub/bin/jupyter nbextension disable

--user assignment\_list/main --section=tree

/home/vuswh/miniconda3/envs/jupyterhub/bin/jupyter serverextension disable

--user nbgrader.server\_extensions.assignment\_list

Then create the grader’s ~/.jupyter/nbgrader\_config.py:

cd ~/.jupyter/

/home/vuswh/miniconda3/envs/jupyterhub/bin/nbgrader generate\_config

Paste the following code in the config file:

c.CourseDirectory.root = '/home/grader-test-course/test-course' c.CourseDirectory.course\_id = 'test-course'

c.GenerateFeedback.preprocessors = [ 'nbgrader.preprocessors.GetGrades', 'nbconvert.preprocessors.CSSHTMLHeaderPreprocessor', # remove: hidden tests from feedback: #'nbgrader.preprocessors.ClearHiddenTests',

# remove tracebacks of hidden tests from feedback #'nbgrader.preprocessors.Execute',

]

Finally, create the directory holding all the course assignments:

sudo mkdir ~/test-course

Students can be added to the course via nbgrader’s web interface.

##### C++ Kernel Installation

To use a C++ kernel within JupyterHub, you must install it first. Follow these steps:

1. Install Mamba, a package manager, using Conda:

conda install -n base -c conda-forge mamba

1. Use Mamba to install the C++ kernel (xeus-cling):

mamba install xeus-cling -c conda-forge

This completes the configuration of JupyterHub for nbgrader in a multi-class lab environment, along with the installation of the C++ kernel for use within JupyterHub. Make sure to adapt the configurations to match your environment and specific requirements.