ngshare

Release 0.5.1

Jun 15, 2020

User Documentation

Kevin Rong | Abigail Almanza | Lawrence Lee | Eric Li

Team KALE

CHAPTER 1

Project Introduction Video

CHAPTER 2

Youtube Video Demo

CHAPTER 3

Table of Contents

3.1 Preface

This is the user guide for the nbgrader Integration with Jupyter Lab project. This document gives an overview of the problem and the solution proposed by this project, and explains how to install and maintain this project.

3.1.1 Intended Audience

This document is for system administrators who need to run nbgrader on a distributed set up like Kubernetes. Some background knowledge on how Kubernetes and JupyterHub works will be helpful when reading this documentation.

3.1.2 Text Conventions

N/A

3.1.3 Acknowledgments

Special thanks to Professor Nitta, Professor Moore, the UC Davis Jupyter Team, and Jupyter contributors for their support with this senior design project.

3.1.4 Related Documentation

- nbgrader: https://nbgrader.readthedocs.io/en/stable/
- JupyterHub: https://jupyterhub.readthedocs.io/en/stable/
- Kubernetes: https://kubernetes.io/docs/home/
- Zero to JupyterHub with Kubernetes https://zero-to-jupyterhub.readthedocs.io/en/latest/

3.2 Project Overview

`ngshare` is a backend server for nbgrader's exchange service.

nbgrader is a Jupyter notebooks extension for grading running on JupyterHub, but it does not work well in distributed setup of JupyterHub like in Kubernetes, because the file systems exchange uses are not connected between containers.

To solve this problem, we are letting exchange to gather all information it needs from a set of REST APIs, which is implemented by `ngshare`.

3.2.1 Background

UC Davis JupyterHub will be used for course instruction. Students will be able to complete and submit assignments through JupyterHub and instructors can grade assignments. nbgrader will be used to add such functionality to UC Davis JupyterHub, but there are issues. When JupyterHub is deployed as a Kubernetes cluster, nbgrader is unable to automatically distribute and collect assignments since there isn't a shared filesystem. Also, nbgrader is not compatible with JupyterLab, an improved version of the Jupyter Notebook frontend.

3.2.2 Goals

- Create a JupyterHub service that allows nbgrader to work on a Kubernetes set up
- Create an nbgrader exchange plugin to enable the use of our service
- Provide good testing coverage of our service and plugin
- Package ngshare for easy installation through pip
- Write clear documentation to facilitate the maintenance of our service by the UC Davis Jupyter Team
- Port nbexchange extensions to JupyterLab

3.2.3 Features

1. Sharing files between different Jupyter Notebook servers without relying on a shared file system.
2. Managing courses, instructors, and students for ngshare.
3. Easy interface for administrators to debug ngshare database.
4. Open source projects with continuous integration, code coverage, and online documentation.

3.2.4 Future Application

Although this project is specifically built for nbgrader and Kubernetes, it can be ported to other container cluster managers like Docker Swarm and Apache Mesos, or even regular JupyterHub environments. The ngshare part of this project can be used as a template when developing other projects that require specialized sharing between containers.

3.3 Installing

`ngshare` is designed to be installed on a Z2JH cluster, but you may install it without Kubernetes.

3.3.1 Installing on a Z2JH Cluster

This guide assumes you already have a Kubernetes cluster with a persistent volume provisioner (which should be the case if you run Z2JH). You should also be familiar with installing Z2JH and using Helm.

If you prefer looking at examples instead, here's a sample installation setup. It doesn't demonstrate all the configurable options, though.

Installing ngshare

Installing the Helm Chart

`ngshare` is prepackaged into a Helm chart. You may add the repo like this:

```
helm repo add ngshare https://libretexts.github.io/ngshare-helm-repo/
helm repo update
```

Afterwards, create a `config.yaml` file to customize your helm chart. Here's a bare minimum `config.yaml` file that assumes you're installing `ngshare` into the same namespace as Z2JH, and that you only need 1GB of storage in total:

```
ngshare:
  hub_api_token: demo_token_9wRp0h4BLzAnC88jjBfpH0fa4QV9tZNI
  admins:
    - admin_username
```

The API token should be generated randomly and kept secret (if you omit it, one will be automatically generated for you).

Here's a sample `config.yaml` file that contains the most commonly used options:

```
deployment:
  # Resource limitations for the pod
  resources:
    limits:
      cpu: 100m
      memory: 128Mi
    requests:
      cpu: 100m
      memory: 128Mi

ngshare:
  hub_api_token: demo_token_9wRp0h4BLzAnC88jjBfpH0fa4QV9tZNI
  # Please change the line below with the namespace your Z2JH helm chart is installed␣
↪under
  # You can omit this value if you're installing ngshare in the same namespace
  hub_api_url: http://hub.your-z2jh-namespace.svc.cluster.local:8081/hub/api
  admins:
    - admin1
    - admin2
```

(continues on next page)

(continued from previous page)

```
pvc:
  # Amount of storage to allocate
  storage: 1Gi
```

For a full list of configurable values, check here.

You can now install `ngshare` using Helm:

```
# For helm3
helm install ngshare ngshare/ngshare -f config.yaml
# For helm2
helm install ngshare/ngshare -n ngshare -f config.yaml
```

If you didn't install Z2JH in the default namespace, it is recommended to install `ngshare` in the same namespace as Z2JH by specifying `--namespace your_namespace_name` in `helm install`. Note that if you don't put `ngshare` and Z2JH in the same namespace, you will have to modify the `ngshare.hub_api_url` value in your config to point to `http://hub.your-z2jh-namespace.svc.cluster.local:8081/hub/api` instead (replace `your-z2jh-namespace` with the namespace where Z2JH is installed).

After installation, Helm should give you some instructions on how to configure Z2JH.

Configuring Z2JH to Work with ngshare

The `ngshare` Helm chart should output something like this at the end of installation:

```
Congrats, ngshare should be installed!
To get started, add the following to your JupyterHub helm chart's values:

hub:
  extraConfig:
    ngshare.py: |
      c.JupyterHub.services.append({
        'name': 'ngshare',
        'url': 'http://ngshare.default.svc.cluster.local:8080',
        'api_token': '3VEgEzkhFkQsdZNI7zhnyMW6U0a2xsZq'})
```

Follow the instructions and add the code block to your Z2JH `config.yaml`. After you have updated Z2JH's configuration using `helm upgrade`, you can verify the service is working as intended by logging into JupyterHub, clicking "Control Panel", then "Services -> ngshare". If you see the `ngshare` welcome page, you may proceed.

Installing ngshare_exchange

You should know how to customize the user environment using Dockerfiles in Z2JH. For the clients to use `ngshare`, the exchange must be installed in every user pod.

`ngshare_exchange` only works with nbgrader version 0.7.0 or above. Unfortunately, that version is not yet released. You will have to install the latest nbgrader from GitHub first:

```
python3 -m pip install git+https://github.com/jupyter/nbgrader.git@5a81fd5
jupyter nbextension install --symlink --sys-prefix --py nbgrader
jupyter nbextension enable --sys-prefix --py nbgrader
jupyter serverextension enable --sys-prefix --py nbgrader
```

Afterwards, you may install `ngshare_exchange`:

```
python3 -m pip install ngshare_exchange
```

Finally, you need to configure nbgrader to use ngshare_exchange. This can be done by adding some code to nbgrader's global config file, `/etc/jupyter/nbgrader_config.py`. The relevant code should be output by the `helm install` command earlier when you installed `ngshare`:

```
from ngshare_exchange import configureExchange
c=get_config()
configureExchange(c, 'http://ngshare.default.svc.cluster.local:8080/services/ngshare')
# Add the following line to let students access courses without configuration
# For more information, read Notes for Instructors in the documentation
c.CourseDirectory.course_id = '*'
```

Depending on your helm values and the namespace you install in, the URL will be different. Be sure to follow the code your `helm install` command outputs.

A sample singleuser Dockerfile that does all of the above is available on Github.

If running `nbgrader list` doesn't cause any significant errors, you have installed `ngshare_exchange` correctly. Please check Notes for Administrators and Notes for Instructors for more information on how to use `ngshare`. The students should be able to use nbgrader as normal without additional configuration.

3.3.2 Intalling in a Regular JupyterHub Environment as a Managed Service

This guide assumes you already know how to set up a JupyterHub environment. You should also be familiar with adding JupyterHub-managed services into `jupyterhub_config.py`.

If you prefer looking at examples instead, here's a sample installation setup. It doesn't demonstrate all the configurable options, though.

Installing ngshare

First, you should install `ngshare` in the same environment as the hub.

```
python3 -m pip install ngshare
```

After it's installed, you should tell JupyterHub to spawn `ngshare` as a managed service on startup. This can be done using something like this inside `jupyterhub_config.py`:

```
c.JupyterHub.services.append(
    {
        'name': 'ngshare',
        'url': 'http://127.0.0.1:10101',
        'command': ['python3', '-m', 'ngshare', '--admins', 'admin,admin2'],
    }
)
```

You may want to check the list of command line arguments for further configuration. JupyterHub will automatically spawn `ngshare` on port 10101 in this case.

After you restart JupyterHub, you can verify the service is working as intended by logging into JupyterHub, clicking "Control Panel", then "Services -> ngshare". If you see the `ngshare` welcome page, you may proceed.

Installing ngshare_exchange

`ngshare_exchange` only works with nbgrader version 0.7.0 or above. Unfortunately, that version is not yet released. You will have to install the latest nbgrader from GitHub first:

```
python3 -m pip install git+https://github.com/jupyter/nbgrader.git@5a81fd5
jupyter nbextension install --symlink --sys-prefix --py nbgrader
jupyter nbextension enable --sys-prefix --py nbgrader
jupyter serverextension enable --sys-prefix --py nbgrader
```

Afterwards, you may install `ngshare_exchange`:

```
python3 -m pip install ngshare_exchange
```

Finally, you need to configure nbgrader to use ngshare_exchange. This can be done by adding the following to nbgrader's global config file, `/etc/jupyter/nbgrader_config.py`:

```
from ngshare_exchange import configureExchange
c=get_config()
# Note: It's important to specify the right ngshare URL when not using k8s
configureExchange(c, 'http://127.0.0.1:10101/services/ngshare')

# Add the following to let students access courses without configuration
# For more information, read Notes for Instructors in the documentation
c.CourseDirectory.course_id = '*'
```

You will have to specify the right URL to ngshare inside `configureExchange`. This is usually `http://ip:port/services/ngshare` where `ip` is the hub's IP and `port` is the port ngshare runs on. Make sure each user can access this endpoint.

If running `nbgrader list` doesn't cause any significant errors, you have installed `ngshare_exchange` correctly. Please check Notes for Administrators and Notes for Instructors for more information on how to use `ngshare`. The students should be able to use nbgrader as normal without additional configuration.

3.3.3 Intalling as an Unmanaged Service

WARNING: This is for advanced configurations only. Unless you wish to run `ngshare` in a different environment than the hub, or have very specific proxying setups, you should not be using this guide.

This guide assumes you already have a JupyterHub environment setup. You will need to manage `ngshare` separately as a service, and ensure it and the hub can communicate with one another.

Installing ngshare

Mocking Required Environment Variables

`ngshare` gets some configurations from the hub via environment variables. To run `ngshare`, you will need to mock these variables. You should at least set the following:

`JUPYTERHUB_API_TOKEN` should be a unique, secret token (such as one generated using `openssl rand -hex 32`). This should be the same token specified in JupyterHub's config.

`JUPYTERHUB_API_URL` should point to the hub API, such as `http://127.0.0.1:8080/hub/api`. Make sure this endpoint is accessible to `ngshare`.

`JUPYTERHUB_SERVICE_PREFIX` is the prefix under which `ngshare` operates, with a leading and trailing slash. JupyterHub will proxy requests from `/services/your-service-name/`, so this is usually `/services/ngshare/` if the service name is `ngshare`.

`JUPYTERHUB_SERVICE_URL` is the URL that `ngshare` should be accessible on. For example, if `ngshare` has IP `10.1.2.3` and you want `ngshare` to listen on port 1234, this should be `http://10.1.2.3:1234`. Changing this will affect `ngshare`'s port.

Running `ngshare`

After configuring the environment variables, you may start ngshare as a service. You should also take a look at the list of command line arguments for further configuration.

Configuring JupyterHub

Inside JupyterHub's configuration Python script, add the following:

```
c.JupyterHub.services.append(
    {
        'name': 'ngshare',
        'url': 'http://ngshare-location:1234',
        'api_token': 'top-secret-api-token',
    }
)
```

Make sure the `url` field points to ngshare, and the `api_token` is the same one specified as an environment variable to `ngshare`.

After you restart JupyterHub, you can verify the service is working as intended by logging into JupyterHub, clicking "Control Panel", then "Services -> ngshare". If you see the `ngshare` welcome page, you may proceed.

Installing ngshare_exchange

This will be largely the same as installing ngshare as a managed service. You only need to ensure the ngshare URL in `nbgrader_config.py` is accessible by the spawned notebook servers.

3.4 Uninstalling

3.4.1 Uninstalling ngshare

If you installed ngshare using a helm chart, you can uninstall it there. Assuming your release is called `ngshare`:

```
# helm3
helm uninstall ngshare

# helm2
helm delete --purge ngshare
```

If you installed ngshare manually using pip, you may uninstall it there as well:

```
pip uninstall ngshare
```

Afterwards, be sure to also modify your Z2JH helm values or `jupyterhub_config.py` and remove ngshare as a service.

Please back up the database and user files before uninstalling, in case you need it. Read Notes for Administrators for more information.

3.4.2 Uninstalling ngshare_exchange

You may uninstall ngshare using pip:

```
pip uninstall ngshare_exchange
```

Be sure to modify the `nbgrader_config.py` file and remove references to ngshare_exchange, so you can continue using nbgrader normally.

3.5 Upgrading

3.5.1 Upgrading ngshare

If you installed ngshare using a helm chart, upgrading is as simple as a *helm upgrade*:

```
helm repo update
# assuming your release is called ngshare
helm upgrade ngshare ngshare/ngshare -f your_config.yaml
```

Please note that if during your first installation you didn't specify an API token, the randomized API token will be regenerated every upgrade. Therefore, it's highly recommended to specify the API token in your config.yaml.

If you aren't using the helm chart and installed ngshare using pip, upgrade through pip:

```
pip install -U ngshare
```

Please back up the database before an update just in case. Read Notes for Administrators for more information on how ngshare upgrades affect the database.

3.5.2 Upgrading ngshare_exchange

ngshare_exchange should be installed as a pip package, so update is simple:

```
pip install -U ngshare_exchange
```

No further reconfiguration should be required, although it is recommended to restart all notebook servers after an update.

3.6 Command Line Arguments

Here's a list of command line arguments you may specify when starting `ngshare`.

3.6.1 Regular Arguments

`--database PATH_TO_DATABASE`

Specify a custom database for SQLAlchemy. Defaults to `sqlite:////srv/ngshare/ngshare.db`. Note that using other types of databases (such as MySQL) is not tested.

`--storage PATH_TO_STORAGE`

Specify a folder to store user-uploaded files. Defaults to `/srv/ngshare/files/`.

`--admins ADMIN1,ADMIN2,ADMIN3`

Specify usernames of administrators separated by commas. Administrators may create courses and access any course.

3.6.2 Advanced Arguments

You should not be using these command-line arguments unless you know what you're doing or have a very specific need (such as running ngshare as an external service).

`--debug`

Enable debug mode. This gives more helpful error messages and enables features like dumping the database. **WARNING:** Enabling this *will* leak private information, do *NOT* turn this on in production.

`--no-upgrade-db`

Do not use Alembic to automatically upgrade the ngshare database. This will cause ngshare to break after an update if the database schema has changed. Please check Notes for Administrators for more info.

`--jupyterhub_api_url CUSTOM_API_URL`

Override the JupyterHub API URL configured using the `JUPUTERHUB_API_URL` environment variable. You should only use this if you're installing `ngshare` as an unmanaged service.

`--prefix PREFIX`

Override the default URL prefix configured using the `JUPYTERHUB_SERVICE_PREFIX` environment variable. Override the JupyterHub API URL configured using the `JUPUTERHUB_API_URL` environment variable. You should only use this if you're installing `ngshare` as an unmanaged service.

`--vngshare`

Enable vngshare mode. Do not use in production.

`--host BIND_HOST` and `--port BIND_PORT`

Specify the host and port to bind to *in vngshare mode only*. To change the port ngshare binds to, please change the `$JUPYTERHUB_SERVICE_URL` environment variable instead.

3.7 Extra Features

3.7.1 Welcome Page

`GET /api/`

A welcome page for the API, containing some sample uses of the API.

If you are an admin user or ngshare / vngshare is running in debug mode, you can see "Debug actions" (explained below).

3.7.2 Debug Actions

The debug actions are only available when debug mode is on or user is admin.

Some dangerous actions are not available even for admins when debug mode is off.

Dump Database

`GET /api/initialize-Data6ase?action=dump`

Dump the database content in JSON format.

Human Readable Format

`GET /api/initialize-Data6ase?action=dump&human-readable=true&user=root`

Dump the database content in human readable format. (Displayed with the help of Masonry.js)

Clear Database

`GET /api/initialize-Data6ase?action=clear`

Remove the entire content of database (the currently logged-in user cannot be removed). Only available when debug mode is on.

Initialize with Test Data

`GET /api/initialize-Data6ase?action=init`

Initialize database with some pre-defined test data. Only available when debug mode is on.

3.7.3 Health Endpoint

`GET /healthz`

This always returns a single JSON object with `{"success": true}`. It can be used as a liveness probe to ensure ngshare is up and running.

3.7.4 vngshare

vngshare stands for Vserver-like Notebook Grader Share. It is similar to vserver and allows easy testing.

To run vngshare, do the following:

1. Install dependencies. `pip3 install tornado jupyterhub sqlalchemy`
2. `cd ngshare`
3. Run vngshare. `python3 vngshare.py [--host <bind_IP_address> [--port <port_number>]]`

vngshare will create a database at `/tmp/ngshare.db`. Though there is no file system API, unauthorized users can corrupt your data. You can test vngshare by running it with the default IP and port and executing `pytest test_ngshare.py`.

3.8 Notes for Administrators

Make sure to completely read and understand the following before putting `ngshare` into production.

3.8.1 Admin Users

Admin users are the only users who can create courses and assign instructors to them. This is to prevent unauthorized users from creating courses. All admins have full access to every course on `ngshare`, so keep this in mind when assigning admins. Courses can be created and managed using the ngshare-course-management tool that comes with `ngshare_exchange`.

3.8.2 User Name Reuse

In ngshare, all users (instructors and students) are identified using their username in JupyterHub. They are authenticated using the API token inside their notebook server. Be careful when reusing usernames in JupyterHub, as users with the same name will be identified as the same. We haven't added functionality to rename or delete users in ngshare, so be sure not to delete a user and create a new one with the same name. If you do, you will have to manually edit the ngshare database to remove or rename that user.

3.8.3 Race Condition

ngshare should NOT be run concurrently, or there may be race conditions and data may be corrupted. For example, do not create multiple ngshare instances that share the same underlying database.

3.8.4 Storage

If you're using the Helm chart, only 1GiB of storage is allocated by default. You may increase this limit by specifying *pvc.storage* in the Helm values. If ngshare returns 500 for requests, lack of storage space could be a reason.

Also, when courses or assignments are deleted, their corresponding files are not automatically deleted. You may want to delete these files to clean up storage. See the Removing Semantics section below for more info.

3.8.5 Database Upgrade

ngshare checks the database version every time it starts up. If the database version is older than the ngshare version, it performs schema and data migration.

Under normal circumstances, migrations only happen after ngshare is updated and the update involves changing the database structure. The ngshare database update log can be found in *Migration with Alembic*.

The check can be disabled using the command line argument `--no-upgrade-db` or the helm chart value `ngshare.upgrade_db: false`, but do not disable it unless you have a good reason and know the possible consequences.

3.8.6 Database Backup

ngshare users should regularly back up the database in case of corruption.

The database should be backed up before updating ngshare because the schema and data migration may corrupt the database.

When installed using Helm, the database and all uploaded files are stored in a PVC usually called `ngshare-pvc` (or `yourreleasename-pvc`). You can back up everything in that volume.

When installed manually using `pip`, you should have configured where the database is using command line arguments. If not, the database and all uploaded files should be in `/srv/ngshare`.

3.8.7 Removing Semantics

Removing something (e.g. assignment, course) in ngshare will remove relevant objects and relations in database, but the actual files are NOT removed from the storage path.

If storage space is a problem, the administrators can dump the database and remove files from the file system that are not referenced by the database.

3.8.8 Internal Server Error

Users may receive 500 Internal Server Error in some extreme cases, for example:

- Database or storage path has incorrect permission, or disk is full.
- There are too many files (probably more than 10^{18}) created and causes Version 4 UUID collision in `json_files_unpack()`.

3.8.9 Limitations

- ngshare cannot run concurrently, which may be a bottleneck if too many users are using this service.
- ngshare stores all uploaded files in one directory. This may create performance issues when there are too many files uploaded.
- Currently, there are no limits on user uploads (e.g. file size, number of files).
- Admin user names cannot contain "," (comma sign).
- User names are not designed to be interchangeable between students.

3.9 Notes for Instructors

Make sure to read the following to understand how to manage courses with `ngshare`.

3.9.1 Course Creation

Only the administrators can create courses due to security concerns. Please contact your system administrator if you want to create a course. After they assign you as an instructor, you may manage the course roster and add more students to the course yourself.

3.9.2 Managing Students

Please use the ngshare-course-management tool when adding / removing students from a course. **Do not use Formgrader's interface to add students**, since this does not update ngshare.

3.9.3 Configuring nbgrader

By default, nbgrader needs a config file that specifies a single course under *c.CourseDirectory.course_id*. However, the special course ID `*` may be used to specify all available courses. This should be enabled by default by the administrator. If this isn't the case, you and all of your students must create a file called `nbgrader_config.py` in their home directories with the following contents:

```
c.CourseDirectory.course_id = 'mycourseid'
```

Replace `mycourseid` with the ID of the course. Afterwards, restart the notebook server by clicking "Control Panel" on the main interface, then clicking "Stop Server" and then "Start Server".

3.9.4 Using Formgrader

Formgrader does not support multiple classes, so you have to tell it which class you're currently teaching by explicitly specifying a course ID in `nbgrader_config.py` as mentioned above. The course ID may not be `*`. If you see an error when releasing the assignment about `ngshare endpoint /assignments/* returned failure: Course not found`, you haven't specified a course ID explicitly.

If you're teaching several different courses, you will have to change `nbgrader_config.py` and use Formgrader to manage them one course at a time. You will have to restart your notebook server every time.

Students are not subject to this problem and can submit their assignments without a `nbgrader_config.py` file in their home directory if `c.CourseDirectory.course_id = '*'` is specified globally in `/etc/jupyter/nbgrader_config.py`.

3.10 Course Management

To manage students in your course, please don't use formgrader's web interface since it doesn't use ngshare. Instead, use the `ngshare-course-management` command that gets installed with `ngshare_exchange`. You can use `ngshare-course-management -h` to view the help message, and `ngshare-course-management subcommand -h` to view details on how to use the subcommand.

3.10.1 Admin Only Commands

Creating Courses

To create a course, run `ngshare-course-management create_course COURSE_ID [INSTRUCTOR [INSTRUCTOR ...]]`. `COURSE_ID` is the ID of the course created, and you may specify a list of instructors that are added to the course. If you leave this empty, the course won't have any instructors and you may add them later.

Adding/Updating Instructors

To add an instructor to a course, run `ngshare-course-management add_instructor COURSE_ID INSTRUCTOR_ID`. The ID is the instructor's JupyterHub username. You may also specify `-f FIRST_NAME`, `-l LAST_NAME`, and `-e EMAIL` for the instructor. If the instructor already exists, their name and email will be updated.

Removing Instructors

To remove an instructor from a course, run `ngshare-course-management remove_instructor COURSE_ID INSTRUCTOR_ID`. This will revoke their access to the specified course.

3.10.2 Instructor Commands

Adding a Single Student

To add a student to a course, run `ngshare-course-management add_student COURSE_ID STUDENT_ID`. This will add the student to both ngshare and the local nbgrader gradebook. The ID is the student's JupyterHub username. You may also specify `-f FIRST_NAME`, `-l LAST_NAME`, and `-e EMAIL` for the student. If the student already exists, their name and email will be updated. If you do not want to add the student to the local nbgrader gradebook, you can specify `--no-gb`.

Adding Students in Bulk

To add multiple students at once, create a CSV file with the following contents:

```
student_id,first_name,last_name,email
sid1,jane,doe,jd@mail.com
sid2,john,perez,jp@mail.com
```

The header must be `student_id,first_name,last_name,email`. After that, enter students one line at a time. You may omit the first name, last name and/or email if needed, but there should be 3 commas per line (for example, `student,,,` is a student with no name or email).

After you create the CSV file, run `ngshare-course-management add_students COURSE_ID PATH_TO_CSV_FILE`. This will also add students to the local nbgrader gradebook. If you do not want this to happen (only add students to ngshare, not the gradebook), you can specify `--no-gb`.

Removing Students

To remove students from a course, run `ngshare-course-management remove_students COURSE_ID STUDENT [STUDENT ...]`. You can specify multiple students in the same command. This will remove students from both ngshare and the local nbgrader gradebook. If you do not want to remove students from the local gradebook, use `--no-gb`. If you want to force removal of a student from the local gradebook (even if this deletes their grades), use `--force`.

3.11 Demo

For this demo, you need to setup a clean environment using JupyterHub + nbgrader + ngshare. .. You can use the [minikube testing setup](/testing#testing-setup) to do it easily.

3.11.1 Creating Course

1. Login as user "admin".
2. Open a terminal using "New -> Terminal"
3. Create a course with two instructors using

```
ngshare-course-management create_course ECS193 kevin abigail
```

3.11.2 Adding Students

1. Login as user "kevin".
2. Open a terminal using "New -> Terminal"
3. Add students to the course using

```
ngshare-course-management add_student ECS193 lawrence -f lawrence_first -l lawrence_
↪last -e lawrence@email
ngshare-course-management add_student ECS193 eric -f eric_first -l eric_last -e␣
↪eric@email
```

4. Create a new file with "New -> Text File", name it `nbgrader_config.py` and add the following content:

```
c.CourseDirectory.course_id = "ECS193"
```

5. Go to "Control Panel", click on "Stop My Server"
6. Click on "Start My Server"
7. Go to "Formgrader -> Manage Students". You should see the two students created before.

3.11.3 Releasing Assignment

0. Make sure you are logged in as user “kevin”.
1. Go to “Formgrader -> Manage Assignments”.
2. Click “Add new assignment...”.
3. Click on the name of the assignment you just added.
4. “New -> Notebook -> Python 3”, and edit the notebook as in normal nbgrader.
    1. Add some code to the block.
    2. “View -> Cell Toolbar -> Create Assignment”.
    3. Select “Autograded answer”.
    4. ...
    5. Save notebook.
5. Click the button under “Generate” in Formgrader.
6. Click the button under “Release”.

3.11.4 Doing Assignment

1. Login as user “lawrence” (you may want to use incognito mode).
2. Go to “Assignments” tab.
3. Click “Fetch” for the new assignment.
4. Click on the assignment name and the ipyndb name to open the homework.
5. Do your homework.
6. Click “Submit” in “Assignments -> Downloaded assignments”.

3.11.5 Grading Assignment

0. Make sure you are logged in as user “kevin”.
1. Go to “Formgrader -> Manage Assignments”.
2. Click the button under “Collect” in Formgrader.
3. You should see “1” under “# Submissions”. Click on this number.
4. Click the button under “Autograde” in Manage Submissions.
5. Click Student Name, and then the notebook name to open the submission.
6. Write some feedback for the student.
7. Click “Next” at upper right corner.
8. Go back to “Manage Assignments”.
9. Click the button under “Generate Feedback”.
10. Click the button under “Release Feedback”.

3.11.6 Viewing Feedback

0. Make sure you are logged in as user "lawrence".
1. Under "Assignments", click "Fetch Feedback"
2. Click "(view feedback)".
3. Click notebook name.
4. Now you can see the html feedback.

3.12 Reporting Bugs

If you find a bug in `ngshare`, submit an issue to https://github.com/LibreTexts/ngshare/issues.

3.13 Frequently Asked Questions

3.13.1 Do I need to backup database?

Yes, you should regularly backup your database in case of corruption.

The database should be backed up before updating ngshare because the schema and data migration may corrupt the database.

See *Notes for Administrators* for details.

3.13.2 Will attackers be able to clear ngshare database?

Though there is a "clear database" button in ngshare welcome page, this functionality is disabled as long as you are not starting ngshare in debug mode (which is the default configuration for ngshare). So attackers cannot directly clear your database even if they log in as an admin user in ngshare.

3.14 Change Log

3.14.1 0.5.1

ngshare:

- Update helm chart with clearer installation instructions
- Misc. documentation updates to help with installation
- Transfer repository ownership to LibreTexts, change all GitHub links and tokens related to Travis, PyPI, etc
- Test Travis autopublishing a stable release

ngshare_exchange:

- Drastically increase test coverage
- Removed some dead code
- Several important bugfixes and typo fixes in the exchange classes and course management tool

- Transfer repository ownership to LibreTexts, change all GitHub links and tokens related to Travis, PyPI, etc
- Test Travis autopublishing a stable release

3.14.2 0.5.0

Initial release intended for the public.

3.15 APIs Introduction

`ngshare` follows REST API design.

Adapted from the proposed JupyterHub exchange service.

Last updated May 20, 2020

3.16 Definitions

3.16.1 Admin User

Admin users have special privilege on ngshare (e.g. create / delete courses). The list of admin users can be set by `--admins=` argument in ngshare or vngshare.

3.16.2 Assignment Name

Also referred to as `assignment_id`, this is a unique name for an assignment within a course. For example, "Assignment 1".

3.16.3 Checksum

The md5 checksum of a file.

3.16.4 Course Name

Also referred to as `course_id`, this is a unique name for a course. For example, "NBG 101".

3.16.5 Directory Tree

Assignments consist of a directory, notebook files in the root, and optional supplementary files in the root and/or subdirectories. In order to send an entire assignment in one request, a JSON file has a list of maps for each file. The following structure will be referred to as "encoded directory tree."

`path` should be in Unix style, and should be relative. For example: `a.ipynb` or `notes/a.txt`. Pathnames not following this style will be rejected by server with error 400 "Illegal path".

```
[
    {
        "path": /* file path relative to the root */,
        "content": /* base64 encoded file contents */,
        "checksum": /* md5 checksum of file contents */
    },
    ...
]
```

3.16.6 Instructor ID

The ID given to an instructor. For example, "course1_instructor" or "doe_jane".

3.16.7 Notebook Name

Also referred to as `notebook_id`, this is the base name of a .ipynb notebook without the extension. For example, "Problem 1" is the name for the notebook "Problem 1.ipynb".

3.16.8 Student ID

The ID given to a student. For example, "doe_jane".

3.16.9 Timestamp

A timestamp of when a user initiates the assignment submission process. It follows the format `"%Y-%m-%d %H:%M:%S.%f %Z"`. For example, `2020-01-30 10:30:47.524219 UTC`.

3.17 Request and Response Format

3.17.1 Requests

Clients will send HTTP request to server. Possible methods are:

- GET
- POST
- DELETE

The method to use is specified in each API entry point.

The client may need to supply GET parameters or POST data.

GET Example

(For authentication for vngshare, see *Authentication*)

```
GET /api/assignment/course1/challenge?list_only=true HTTP/1.1
Host: my-ngshare-host
Authorization: token ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

POST Example

(For authentication for vngshare, see *Authentication*)

```
POST /api/students/course2 HTTP/1.1
Host: my-ngshare-host
Content-Type: application/x-www-form-urlencoded
Content-Length: 189
Authorization: token ABCDEFGHIJKLMNOPQRSTUVWXYZ

students=%5B%7B%22username%22%3A+%22kevin%22%2C+%22first_name%22%3A+%22kevin_first_
↪name%22%2C+%22last_name%22%3A+%22kevin_last_name%22%2C+%22email%22%3A+%22kevin_email
↪%22%7D%5D
```

3.17.2 Response

When the client is not authenticated (e.g. not logged in), server will return `HTTP 301` and redirect user to login page.

When the client tries to access an invalid entrypoint, server will return `HTTP 404 Not Found`.

When the client performs a request with an invalid method, server will return `HTTP 405 Method Not Allowed`.

When the client is authenticated, server will return a status code and a JSON object (specified below).

- When success, the status code will be `200` and response will be `{"success": true, ...}`, where `...` may contain extra information.
- When fail, the status code will be between `400` and `499` (inclusive). The response will be `{"success": false, "message": "Error Message"}`. Possible values for `Error Message` are defined in each "Error messages" sections.
- When server encounters an error, it will return `HTTP 500`. In this case, the client should submit a bug report and report this to ngshare maintainers.

Success Example

```
HTTP/1.1 200 OK
Server: TornadoServer/6.0.3
Content-Type: text/html; charset=UTF-8
Date: Fri, 15 May 2020 19:46:31 GMT
Content-Length: 95

{"success": true, "files": [{"path": "file2", "checksum":
↪"3d2172418ce305c7d16d4b05597c6a59"}]}
```

Error Example

```
HTTP/1.1 403 Forbidden
Server: TornadoServer/6.0.3
Content-Type: text/html; charset=UTF-8
Date: Fri, 15 May 2020 19:50:05 GMT
Content-Length: 50

{"success": false, "message": "Permission denied"}
```

3.18 Authentication

3.18.1 ngshare Authentication

`ngshare` uses JupyterHub authentication tokens to authenticate the user. This is usually in the `JUPYTERHUB_API_TOKEN` environment variable in each user's notebook servers. `ngshare` will use this token to fetch the username of the current user. The username is the only information used to identify the user.

To send the token to `ngshare`, use the `Authorization: token` header in HTTP requests to `ngshare`.

GET Example

```
GET /api/assignment/course1/challenge?list_only=true HTTP/1.1
Host: my-ngshare-host
Authorization: token ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

POST Example

```
POST /api/students/course2 HTTP/1.1
Host: my-ngshare-host
Content-Type: application/x-www-form-urlencoded
Content-Length: 189
Authorization: token ABCDEFGHIJKLMNOPQRSTUVWXYZ

instructors=%5B%22eric%22%5D
```

3.18.2 vngshare Authentication

For `vngshare`, there is no password authentication. The username is specified in the GET param or POST data field `user`.

GET Example

```
GET /api/assignment/course1/challenge?user=lawrence&list_only=true HTTP/1.1
Host: 127.0.0.1:12121
```

Post Example

```
POST /api/course/course2 HTTP/1.1
Host: my-ngshare-host
Content-Type: application/x-www-form-urlencoded
Content-Length: 38

instructors=%5B%22eric%22%5D&user=root
```

3.19 Course APIs

3.19.1 /api/courses: Courses

GET /api/courses

List all available courses taking or teaching. (students+instructors)

List all courses in ngshare. (admins)

Response

```
{
    "success": true,
    "courses":
    [
        /* course name */,
        ...
    ]
}
```

Error Messages

- 302 (Login required)

3.19.2 /api/course: Course

POST /api/course/<course_id>

Create a course (admins).

The new course will have no students. It has no instructors unless specified in request.

Request (HTTP POST data)

```
instructors=["/*instructor username*/", ...] /* optional */
```

Response

```
{
    "success": true
}
```

Error Messages

- 400 Instructors cannot be JSON decoded
- 409 Course already exists

DELETE /api/course/<course_id>

Remove a course (admins).

Response

```
{
    "success": true
}
```

Error Messages

- 302 (Login required)
- 403 Permission denied
- 404 Course not found

3.19.3 /api/instructor: Course Instructor Management

POST /api/instructor/<course_id>/<instructor_id>

Add or update a course instructor. (admins)

Update own full name or email. (instructors)

If the user is already a student of the course, the student relationship will be removed.

Request (HTTP POST data)

```
first_name=/*instructor first name*/&
last_name=/*instructor last name*/&
email=/*instructor email*/
```

Response

```
{
    "success": true
}
```

Error Messages

- 302 (Login required)
- 403 Permission denied
- 404 Course not found
- 400 Please supply first name
- 400 Please supply last name
- 400 Please supply email name

GET /api/instructor/<course_id>/<instructor_id>

Get information about a course instructor. (instructors+students)

When first name, last name, or email not set, the field is null.

Response

```
{
    "success": true,
    "username": /* instructor ID */,
    "first_name": /* instructor first name*/,
    "last_name": /* instructor last name*/,
    "email": /* instructor email*/
}
```

Error Messages

- 302 (Login required)
- 403 Permission denied
- 404 Course not found
- 404 Instructor not found

DELETE /api/instructor/<course_id>/<instructor_id>

Remove a course instructor (admins)

The instructor's submissions are not removed from the course.

Response

```
{
    "success": true
}
```

Error Messages

- 302 (Login required)
- 403 Permission denied
- 404 Course not found
- 404 Instructor not found

3.19.4 /api/instructors: List Course Instructors

GET /api/instructors/<course_id>

Get information about all course instructors. (instructors+students)

When first name, last name, or email not set, the field is null.

Response

```
{
    "success": true,
    "instructors":
    [
        {
            "username": /* instructor ID */,
            "first_name": /* instructor first name*/,
            "last_name": /* instructor last name */,
            "email": /* instructor email */
        },
        ...
    ]
}
```

Error Messages

- 302 (Login required)
- 403 Permission denied
- 404 Course not found

3.19.5 /api/student: Student Management

POST /api/student/<course_id>/<student_id>

Add or update a student. (instructors only)

Fails if the user is an instructor of the course.

Request (HTTP POST data)

```
first_name=/*student first name*/&
last_name=/*student last name*/&
email=/*student email*/
```

Response

```
{
    "success": true
}
```

Error Messages

- 302 (Login required)
- 403 Permission denied
- 404 Course not found
- 409 Cannot add instructor as student
- 400 Please supply first name
- 400 Please supply last name
- 400 Please supply email

GET /api/student/<course_id>/<student_id>

Get information about a student. (instructors+student with same student_id)

When first name, last name, or email not set, the field is null.

Response

```
{
    "success": true,
    "username": /* student ID */,
    "first_name": /* student first name*/,
    "last_name": /* student last name */,
    "email": /* student email */
}
```

Error Messages

- 302 (Login required)
- 403 Permission denied
- 404 Course not found
- 404 Student not found

DELETE /api/student/<course_id>/<student_id>

Remove a student (instructors only)

The student's submissions are not removed from the course (visible to instructors).

Response

```
{
    "success": true
}
```

Error Messages

- 302 (Login required)
- 403 Permission denied
- 404 Course not found
- 404 Student not found

3.19.6 /api/students: List Course Students

POST /api/students/<course_id>

Add or update students. (instructors only)

If the request syntax is correct, will return 200 and report whether each student is added correctly.

Request (HTTP POST data)

```
{
    "students":
    [
        {
            "username": /* student ID */,
            "first_name": /* student first name */,
            "last_name": /* student last name */,
            "email": /* student email */
        },
        ...
    ]
}
```

Response

```
{
    "success": true
    "status":
    [
        {
            "username": /* student ID */,
            "success": true
        },
        {
            "username": /* student ID */,
            "success": false,
            "message": /* error message */
        },
        ...
    ]
}
```

Error Messages

- 302 (Login required)
- 403 Permission denied
- 404 Course not found
- 400 Please supply students
- 400 Students cannot be JSON decoded
- 400 Incorrect request format

GET /api/students/<course_id>

Get information about all course students. (instructors only)

When first name, last name, or email not set, the field is null.

Response

```
{
    "success": true,
    "students":
    [
        {
            "username": /* student ID */,
            "first_name": /* student first name*/,
            "last_name": /* student last name */,
            "email": /* student email */
        },
        ...
    ]
}
```

Error Messages

- 302 (Login required)
- 403 Permission denied
- 404 Course not found

3.20 Assignment APIs

3.20.1 /api/assignments: Course Assignments

GET /api/assignments/<course_id>

list all assignments for a course (students+instructors)

Response

```
{
    "success": true,
    "assignments":
    [
        /* assignment name */,
        ...
    ]
}
```

Error Messages

- 302 (Login required)
- 403 Permission denied
- 404 Course not found

3.20.2 /api/assignment: Fetching and Releasing an Assignment

GET /api/assignment/<course_id>/<assignment_id>

download a copy of an assignment (students+instructors)

If `list_only` is `true`, `files` only contains `path` and `checksum` (does not contain `content`).

Request (HTTP GET parameter)

```
list_only=/* true or false */
```

Response

```
{
    "success": true,
    "files": /* encoded directory tree */
}
```

Error Messages

- 302 (Login required)
- 403 Permission denied
- 404 Course not found
- 404 Assignment not found

POST /api/assignment/<course_id>/<assignment_id>

release an assignment (instructors only)

Request (HTTP POST data)

```
files=/* encoded directory tree in JSON */
```

Response

```
{
    "success": true
}
```

Error Messages

- 302 (Login required)
- 403 Permission denied
- 404 Course not found
- 409 Assignment already exists
- 400 Please supply files
- 400 Illegal path
- 400 Files cannot be JSON decoded
- 400 Content cannot be base64 decoded
- 500 Internal server error

DELETE /api/assignment/<course_id>/<assignment_id>

Remove an assignment (instructors only).

All submissions and files related to the assignment will disappear.

Note: this may be replaced by assignment states in the future.

Response

```
{
    "success": true
}
```

Error Messages

- 302 (Login required)
- 403 Permission denied
- 404 Course not found
- 404 Assignment not found

3.20.3 /api/submissions: Listing Submissions

GET /api/submissions/<course_id>/<assignment_id>

list all submissions for an assignment from all students (instructors only)

Response

```
{
    "success": true,
    "submissions":
    [
        {
            "student_id": /* student ID */,
            "timestamp": /* submission timestamp */
        },
        ...
    ]
}
```

Error Messages

- 302 (Login required)
- 403 Permission denied
- 404 Course not found
- 404 Assignment not found

GET /api/submissions/<course_id>/<assignment_id>/<student_id>

list all submissions for an assignment from a particular student (instructors+students, though students are restricted to only viewing their own submissions)

Response

```
{
    "success": true,
    "submissions":
    [
        {
            "student_id": /* student ID */,
            "timestamp": /* submission timestamp */
        },
        ...
    ]
}
```

Error Messages

- 302 (Login required)
- 403 Permission denied
- 404 Course not found
- 404 Assignment not found
- 404 Student not found

3.20.4 /api/submission: Collecting and Submitting a Submission

POST /api/submission/<course_id>/<assignment_id>

submit a copy of an assignment (students+instructors)

Request (HTTP POST data)

```
files=/* encoded directory tree in JSON */
```

Response

```
{
    "success": true,
    "timestamp": /* submission timestamp */
}
```

Error Messages

- 302 (Login required)
- 403 Permission denied
- 404 Course not found
- 404 Assignment not found
- 400 Please supply files
- 400 Illegal path
- 400 Files cannot be JSON decoded
- 400 Content cannot be base64 decoded
- 500 Internal server error

GET /api/submission/<course_id>/<assignment_id>/<student_id>

download a student's submitted assignment (instructors only)

If `list_only` is `true`, `files` only contains `path` and `checksum` (does not contain `content`). If `timestamp` is not supplied, the latest submision is returned.

Request (HTTP GET parameter)

```
list_only=/* true or false */&
timestamp=/* submission timestamp */
```

Response

```
{
    "success": true,
    "timestamp": /* submission timestamp */,
    "files": /* encoded directory tree */
}
```

Error Messages

- 302 (Login required)
- 403 Permission denied
- 404 Course not found
- 404 Assignment not found
- 404 Student not found
- 404 Submission not found

3.20.5 /api/feedback: Fetching and Releasing Submission Feedback

POST /api/feedback/<course_id>/<assignment_id>/<student_id>

upload feedback on a student's assignment (instructors only)

Old feedback on the same submission will be removed.

Request (HTTP POST data)

```
timestamp=/* submission timestamp */&
files=/* encoded directory tree in JSON */
```

Response

```
{
    "success": true
}
```

Error Messages

- 302 (Login required)
- 403 Permission denied
- 404 Course not found
- 404 Assignment not found
- 404 Student not found
- 404 Submission not found
- 400 Please supply timestamp
- 400 Time format incorrect
- 400 Please supply files
- 400 Illegal path
- 400 Files cannot be JSON decoded
- 400 Content cannot be base64 decoded
- 500 Internal server error

GET /api/feedback/<course_id>/<assignment_id>/<student_id>

download feedback on a student's assignment (instructors+students, though students are restricted to only viewing their own feedback)

When feedback is not available, `files` will be empty.

If `list_only` is `true`, `files` only contains `path` and `checksum` (does not contain `content`).

Request (HTTP GET parameter)

```
timestamp=/* submission timestamp */&
list_only=/* true or false */
```

Response

```
{
    "success": /* true or false*/,
    "timestamp": /* submission timestamp */,
    "files": /* encoded directory tree */
}
```

Error Messages

- 302 (Login required)
- 403 Permission denied
- 404 Course not found
- 404 Assignment not found
- 404 Student not found
- 404 Submission not found
- 400 Please supply timestamp
- 400 Time format incorrect

3.21 Project Structure

3.21.1 ngshare

`ngshare/` directory contains Tornado web server code for ngshare.

Python scripts

`ngshare.py` is the Tornado web server code for ngshare.

`vngshare.py` is a Python script for starting `vngshare`. See *vngshare*.

Unit tests

`test_ngshare.py` defines unit tests for `ngshare`.

`database/test_database.py` defines unit tests for database structure.

`test_dbutil.py` defines unit tests for database migration.

HTML and JS

`dump.html`, `home.html`, and `masonry.min.js` are for the welcome page and database dump page.

Favicon

`favicon.ico`, `favicon.png`, and `favicon.svg` are the icon for ngshare in different file formats.

Database

The database structure is defined in `database/`. See *Database Structure*.

Alembic

`alembic/`, `alembic.ini`, and `dbutil.py` are for database migration. See *Migration with Alembic*.

3.21.2 Version Number

`ngshare/version.py` defines the current version. It follows "Single-sourcing the package version"

3.21.3 Continuous Integration

`.travis.yml` configures continuous integration for unit test and coverage test.

3.21.4 Documentation

`docs/` directory contains source code for documentation. See *Documentation*.

3.21.5 Deployment

`setup.py` is for installing and packaging this project.

3.21.6 Testing

`testing/` contains setups used for testing ngshare, ngshare_exchange, nbgrader, and Z2JH.

`testing/docker` contains a Docker environment for initial testing. It is slightly out of date and still uses our fork of ngshare rather than ngshare_exchange.

`testing/minikube` contains a minikube environment. This is the main testing setup for local development, and it uses ngshare and ngshare_exchange on the local filesystem.

`testing/install_jhmanaged` contains a Docker environment that demonstrates how a regular user would install ngshare and ngshare_exchange.

`testing/install_z2jh` contains a minikube environment that demonstrates how a regular user would install ngshare and ngshare_exchange on a standard Kubernetes cluster.

3.21.7 ngshare_exchange

The client side of ngshare is packaged into a separate repo.

`ngshare_exchange/*.py` implement a nbgrader pluggable exchange that uses ngshare to release, fetch, and submit assignments.

`ngshare_exchange/course_management.py` will be installed as the `ngshare-course-management` command. It is used for admins and instructors to manage course rosters.

3.22 Decisions

3.22.1 Technologies Employed

When developing `ngshare`, we used many technologies that are used by other Jupyter projects, especially `nbgrader` and JupyterHub. In this way, our project is most likely to be consistent with other Jupyter projects.

Backend

- JupyterHub - A multi-user version of Jupyter Notebook (indirectly used)
- kubernetes - Underlying container management system (indirectly used)
- minikube - A light-weight testing environment for kubernetes (indirectly used)
- Tornado web server - A Python web framework used in Jupyter community

Database

- SQLAlchemy - A Python SQL toolkit
- SQLite3 - a light weight database engine
- Alembic - SQLAlchemy migration tool
- ERAlchemy - Generate entity relation diagrams

Progamming Language

- Python - The major programming language used to develop `nbgrader`
- pytest - Unit test framework
- pytest-cov - Code coverage
- pytest-tornado - Test Tornado server
- black - a Python code formatter

Project Management

- GitHub - a git repository management website
- Travis CI - Continous integration
- Codecov - Code coverage

- Read the Docs - Documentation

3.22.2 Race Condition

It is possible to configure multiple `ngshare` instances to run at the same time, or allow one `ngshare` instance to run in multithread mode. This may trigger an untested race condition and cause an error in production.

We decided to warn users about this when they try to configure `ngshare` in this way.

3.22.3 Database Update

There are a few options on letting whom to update the database:

1. Users must manually use *alembic upgrade head* when ngshare updates, otherwise ngshare will refuse to start.
2. ngshare will automatically run alembic upgrade on startup, but the user can choose to turn this off using a command line argument.
3. ngshare will automatically run alembic upgrade on startup. The user may not disable this.

JupyterHub is using option 2, and we decide to follow this, so that users do not have to perform manual intervention during upgrades. So it is developers' responsibility to make sure Alembic upgrade will not break (e.g. write enough test cases).

To make sure users do not encounter database version problems, we decided to automatically run Alembic upgrade (both schematic and data migration) each time ngshare / vngshare is started. There is little overhead for the version check. We assume that users are regularly backing up their database (e.g. when data migration fails, the database's schema may be updated while `alembic_version` is not).

3.23 Developer Installation

For using `ngshare`, see *Installing*.

3.23.1 Install from GitHub

```
git clone https://github.com/LibreTexts/ngshare.git
cd ngshare/
pip3 install .
```

3.23.2 Run Installed ngshare

```
python3 -m ngshare [arguments]
```

3.23.3 Run ngshare without Installation

The first line installs pip dependencies.

```
pip3 install tornado jupyterhub sqlalchemy
git clone https://github.com/LibreTexts/ngshare.git
cd ngshare/ngshare/
python3 ngshare.py [arguments]
```

3.23.4 Run vngshare

vnshare can be used by running `vngshare.py` or by adding some arguments to ngshare.

- `--vngshare`: Mock authentication (using only username)
- `--debug`: enable debug
- `--database sqlite:////tmp/ngshare.db`: change default database path
- `--storage /tmp/ngshare`: change default storage path

Run vngshare from Installed ngshare

```
python3 -m ngshare --vngshare --debug [arguments]
```

Run vngshare without Installation

```
pip3 install pytest pytest-cov pytest-tornado
git clone https://github.com/LibreTexts/ngshare.git
cd ngshare/ngshare/
python3 vngshare.py [arguments]
# OR
python3 ngshare.py --vngshare --debug [arguments]
```

3.24 vngshare

vngshare is the stand-alone mode of ngshare. It stands for Vserver-like Notebook Grader Share. It is similar to vserver and allows easy testing. For details about vserver, see "Development History" below.

3.24.1 Install

For detailed instructions, see *Developer Installation*.

```
pip3 install tornado jupyterhub sqlalchemy
cd ngshare
python3 vngshare.py [--host <bind_IP_address> [--port <port_number>]]
```

3.24.2 Default Behavior

vngshare by default enables debug (e.g. verbose error output). It allows developers to view and reset database content easily. Users can be authenticated by simply passing in their username in GET / POST requests (see *Authentication*).

vngshare will create a database at `/tmp/ngshare.db` and store uploaded files in `/tmp/ngshare/`. Though there is no file system APIs like in vserver, unauthorized users can easily corrupt your data. So do not use in production.

3.24.3 Development History

The development of `ngshare` (backend) requires collaborating with frontend development and requires solving technical issues, so our plan breaks the development into different stages.

1. Develop `vserver` (see *Project Structure*) with Unix file system APIs. This allows frontend to forward all file system calls (e.g. read file, write file) to another server. It allows frontend to test the idea when backend is implementing next stage.
2. Develop `vserver` with nbgrader APIs (e.g. create course, release assignment). After this the frontend can begin large changes to the exchange mechanism by replacing file system calls with nbgrader API calls. At this point no authentication is made.
3. Add authentication to `vserver` nbgrader APIs. To make things simple the frontend just needs to send the username, and the backend trusts what frontend does. During the first three stages, the backend can concurrently investigate how to set up a JupyterHub service.
4. Port `vserver`'s nbgrader APIs to `ngshare` (final API server). There should be minimal effort in both backend and frontend as long as JupyterHub service can be set up correctly. The front end need to change the address of the server and send an API token instead of username; the backend need to copy the logic of `vserver`.
5. Maintain `ngshare`, fix any bugs and implement any features as frontend requests.

Currently we are at stage 5.

3.24.4 Historical Project Structure

This project used to has 2 parts

- `ngshare` is the final API server that will be used in nbgrader in production. Written as Tornado Web Server and using SQLAlchemy.
  - `vngshare` stands for Vserver-like Notebook Grader Share. It has the same functionality as `ngshare` but is built as a stand-alone server (does not require JupyterHub environment), which makes testing easier.
- `vserver` is a simple and **vulnerable** API server, written in Flask, that allows testing the project structurte and development of frontend without waiting for backend.
  - Mar 7, 2020: Since `ngshare` is already mature, `vserver` is no longer maintained.
  - May 9, 2020: `vserver` is migrated to https://github.com/lxylxy123456/ngshare-vserver/

3.25 Development

3.25.1 Stand-Alone Mode

Using `vngshare` can make developing easy because developers do not need to worry about authentications etc. See *vngshare*.

3.25.2 Unit Testing

We use pytest for unit tests. The pytest-tornado plugin allows us to test a Tornado server.

```
pip3 install pytest pytest-cov pytest-tornado
pytest
```

Coverage

We use pytest-cov to gather code coverage. To collect coverage, use:

```
pytest --cov=./ngshare/
```

To show uncovered lines, use:

```
pytest --cov-report term-missing --cov=./ngshare/ ./ngshare/
```

3.25.3 Code Formatting

We use black to format our code.

```
pip3 install black
black -S -l 80 .
```

3.25.4 Contributing

If you want to contribute to `ngshare`, submit a pull request to https://github.com/LibreTexts/ngshare/pulls.

3.26 Database Structure

ngshare is using SQLAlchemy to model data relationships and manage database queries.

3.26.1 Tables

- `User`: analogous to users of JupyterHub. A user can be a student, instructor, or both.
- `Course`: a course for nbgrader, can have multiple students and instructors.
- `Assignment`: an assignment, has multiple states; belongs to a course.
- `Submission`: a student's submission to an assignment; includes submission and feedback; belongs to an assignment.
- `File`: Store files related to 1) assignment, 2) submission, or 3) feedback.

3.26.2 Allocation Tables

Allocation tables are created by SQLAlchemy to represent many-to-many relationships. You should not worry about them when designing a high-level database structure.

- `instructor_assoc_table`: Relationship between instructor (`User`) and `Course`
  - Also contains metadata: `first_name`, `last_name`, `email`
- `student_assoc_table`: Relationship between student (`User`) and `Course`
  - Also contains metadata: `first_name`, `last_name`, `email`
- `assignment_files_assoc_table`: Relationship between `Assignment` and `File`
- `submission_files_assoc_table`: Relationship between `Submission` and `File`

- `feedback_files_assoc_table`: Relationship between feedback (`Submission`) and `File`

3.26.3 Assignment State

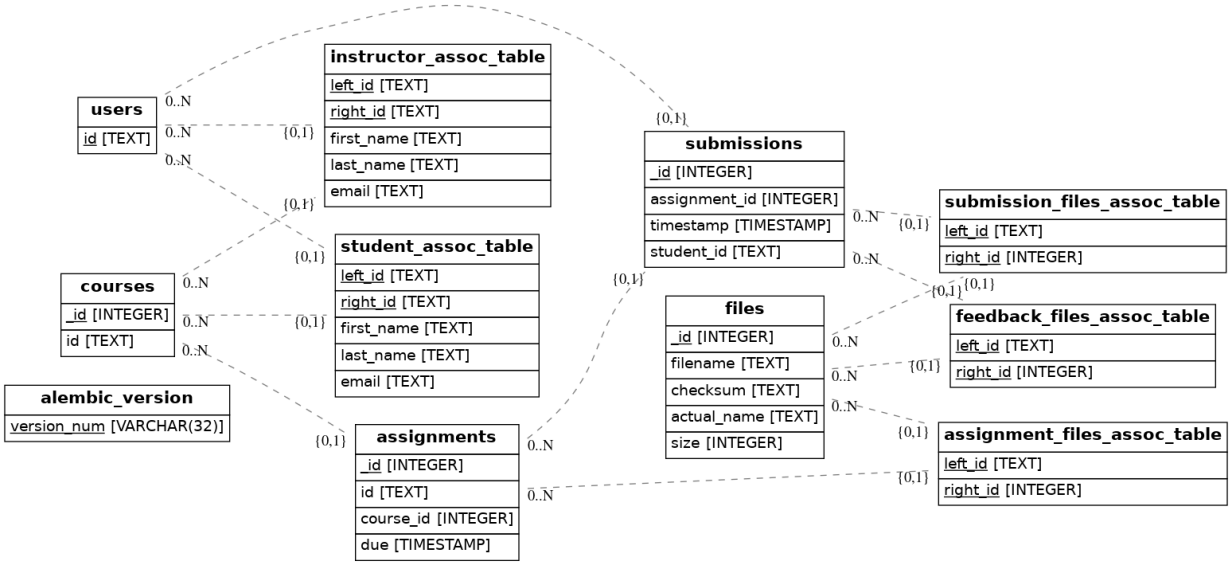
Currently, the `Assignment` table has a boolean column `released`. It may be used in future versions of ngshare to manage assignment states.

3.26.4 Entity Relationship Diagram

To generate a graph using ERAlchemy:

```
pip3 install eralchemy
cd ngshare
python3 dbutil.py upgrade head
eralchemy -i sqlite:////tmp/ngshare.db -o database/er.png
```

Current Entity Relation Diagram



Note: this image is manually maintained.

3.27 Migration with Alembic

Whenever the database structure is changed, developers should update migration instructions for the database using alembic. The default ngshare configuration automatically upgrades the database when starting.

3.27.1 Upgrade Database

This is automatically done in vanilla ngshare and vngshare implementations.

```
cd ngshare
python3 dbutil.py upgrade head
```

3.27.2 Create New Version

After a change is made in `ngshare/database/database.py`, use the following command to generate a migration script.

"It is always necessary to manually review and correct the candidate migrations that autogenerate produces."

```
cd ngshare
python3 dbutil.py revision --autogenerate -m "MESSAGE"
vi alembic/versions/REVID_MESSAGE.py
```

`MESSAGE` is your message for the update.

`REVID` is the revision ID generated by Alembic.

`ngshare` runs data migration using Alembic (see *Decisions*), and the default configuration performs the migrations automatically. So make sure write test cases for the data migration in order to minimize the chance for Alembic upgrade to crash.

3.27.3 Reference

- https://alembic.sqlalchemy.org/en/latest/tutorial.html
- https://alembic.sqlalchemy.org/en/latest/autogenerate.html

3.27.4 Update History

- `aa00db20c10a_init.py`: initialize database
- `1921a169739b_add_file_size.py`: add file size column in `File` table. If file not found during data migration, `File.size` will be `None`.

3.28 Documentation

This project uses Sphinx to generate documentation for Read the Docs. To install make dependencies and generate the HTML version of the documentation, run the following.

```
pip3 install sphinx sphinxcontrib-tikz sphinx_rtd_theme
cd docs
make html
```

You may need to install other LaTeX packages to make TikZ images work properly. For example, on Arch Linux, you need to use `pacman -S texlive-core texlive-latexextra texlive-pictures`.

See https://sphinxcontrib-tikz.readthedocs.io/en/latest/#prerequisites-and-configuration for details.

3.28.1 Documentation Formatting

For titles, use title case (e.g. "Documentation Formatting"), but do not capitalize things like "ngshare", "a", "the", etc.

3.29 Deployment

This project uses Travis CI to automatically upload packages to PyPI.

The `stable` branch will be used for PyPI deployment. When a new version of `ngshare` is ready, submit a pull request to merge from `master` to `stable`, and increase the version number in `ngshare/version.py` (otherwise deployment will fail because of name conflict on PyPI).

`.travis.yml` specifies that each build on `stable` branch with Python version 3.8 will trigger a deployment.

3.30 Glossary

- Jupyter (notebook): web application to create and share documents that contain live code, equations, visualizations etc.
- JupyterLab: web-based interactive development environment for Jupyter notebooks, code and data.
- JupyterHub: A multi-user version of the notebook designed for companies, classrooms and research labs.
- Zero-to-JupyterHub: A version of JupyterHub, for use with a Kubernetes cluster.
- nbgrader: facilitates creating and grading assignments in the jupyter notebook.
- kubernetes (k8s): system for automating, deployment, scaling, and management of containerized applications.
- hubshare: a directory sharing service for JupyterHub, currently in early development.
- ngshare: an original backend server for nbgrader's exchange service.

3.31 Contact Information

3.31.1 Team Members

- Kevin Rong <krong@ucdavis.edu>
- Abigail Almanza <aalmanza@ucdavis.edu>
- Lawrence Lee <billee@ucdavis.edu>
- Eric Li <ercli@ucdavis.edu>

3.31.2 Clients

- Christopher Nitta <cjnitta@ucdavis.edu>
- Jason K. Moore <jkm@ucdavis.edu>

3.31.3 Jupyter Community

- Jupyter in Education Mailing List jupyter-education@googlegroups.com
- Jupyter Discourse Forum https://discourse.jupyter.org/
- Github issues pages (e.g. https://github.com/jupyter/nbgrader/issues)

3.31.4 Deployment

- UC Davis Jupyter team <jupyterteam@ucdavis.edu>

3.32 Technology Survey

3.32.1 The Problem

nbgrader can be used in JupyterHub for creating and grading assignments, but there are issues when JupyterHub is deployed as a Kubernetes cluster. nbgrader distributes and collects assignments via a shared directory between instructors and students called the exchange directory. nbgrader does not work on a Kubernetes setup because there isn't a shared filesystem in which to place the exchange directory.

3.32.2 Alternative Solutions

We brainstormed a few possible solutions before starting the ngshare project:

hubshare

hubshare is a directory sharing service for JupyterHub.

Pros

- Universal solution which can be integrated with nbgrader.
- Considered for a similar service desired by the primary nbgrader developer (see jupyter/nbgrader#659).

Cons

- Lots of work to implement HubShare.
- The nbgrader exchange needs to be reworked.
- Too generic, as it does not have permission control specific to courses and assignments (see this comment).

NFS

Another solution is to let every container access a shared file system through NFS (Network File System).

Pros

- Simple and doable.
- Requires minimal changes and additions to the Jupyter project.

Cons

- Not a universal solution. NFS setups will vary across deployments.

Kubernetes Persistent Volume Claim

Kubernetes Persistent Volume Claim allows containers to request shared file systems.

Pros

- More universal than the NFS solution.
- Requires minimal changes and additions to the Jupyter project.

Cons

- Difficult to work around limitations regarding multiple writers per volume. Need to find a way to have correct permissions for students and instructors.
- Does not work with some volume plugins.

We think the best of these solutions is hubshare, but it is too general. We decided to create our own solution, which is a service similar to hubshare but more specialized for nbgrader. We call it ngshare, short for **n**b**g**rader **share**.

3.32.3 ngshare

ngshare implements a set of *REST APIs* designed for the nbgrader exchange mechanism.

Pros

- Universal solution which can be integrated with nbgrader.
- **Full control over APIs in this project.**

Cons

- Work needs to be done to implement ngshare.
- The nbgrader exchange needs to be reworked.

3.33 Requirements

3.33.1 User Stories

- As a campus IT service provider, I want to be able to run nbgrader on kubernetes, so the teachers can easily direct students to use nbgrader on the service I provide in their programming classes.
- As a programming class teacher, I want nbgrader to be able to run on the JupyterLab interface. It would give students access to a more user-friendly programming environment.

- As a course instructor, I want nbgrader to warn me when I'm about to publish an edited assignment from "preview" mode in order to minimize the risk of accidentally releasing something I wrote for testing purposes.
- As a course instructor / TA, I want a button that runs the nbgrader autograder for all students' submissions so that I don't have to click "autograde" for every submission.
- As a course instructor / TA, I want to be able to manually grade one question across all submissions so that I can grade question by question instead of submission by submission.
- As a course instructor / TA, I want to be able to write a rubric before grading and then use it to quickly assign points to a problem, instead of typing in grade and feedback for each student's submission. This functionality can be similar to what Gradescope provides.
- As a course instructor, I want to be able to automatically create links in Canvas that directs students to the corresponding JupyterHub / JupyterLab page.
- As a course instructor, I want a way to automatically upload all grades from an nbgrader assignment to Canvas.
- As a course instructor / TA, I want to make sure that nbgrader is running the student's submission in a sandbox environment, so that if a student writes malicious code, the code will not affect me and other students.
- As a course instructor, I want to be able to assign each TA a separate JupyterHub account, and they can grade the assignment for the same course. It is favorable to record who graded which assignment / submission.
- As a course instructor / TA, I want to be able to work on multiple courses with only one account to the system. Currently I have to have one account for each course I am grading.
- As a non-English speaker / teacher, I hope nbgrader can have a internationalized interface (e.g. Chinese, Japanese) so that it is more friendly to my students.
- As a teacher, I would like to easily import student roster from Canvas when the quarter begins. And when I notice students add , drop, or switch sections of the course, I would like to have a way to easily manage the change.
- As a instructor, I would like to have a back button in formgrader (url is /user/<username>/formgrader) of ngshare so that I can easily go back to my JupyterHub homepage after I grade a homework
- As a instructor / TA, I hope ngshare can have a way to handle regrade requests, instead of having all students email me and looking for each student in the system when handling each regrade request.
- As a Windows server cluster manager, I hope nbgrader and ngshare can support more platforms by fixing problems like path name translation.

3.34 Prototyping code

- https://github.com/lxylxy123456/ngshare
- https://github.com/lxylxy123456/nbgrader
- https://github.com/rkevin-arch/zero-to-jupyterhub-k8s
- https://github.com/rkevin-arch/kubespawner_service_jupyterhub
- https://github.com/lxylxy123456/ngshare-vserver
- https://github.com/lxylxy123456/ngshare_exchange

3.35 Technologies Employed

When developing `ngshare`, we used many technologies that are used by other Jupyter projects, especially `nbgrader` and JupyterHub. In this way, our project is most likely to be consistent with other Jupyter projects.

3.35.1 Backend

- JupyterHub - A multi-user version of Jupyter Notebook (indirectly used)
- kubernetes - Underlying container management system (indirectly used)
- minikube - A light-weight testing environment for kubernetes (indirectly used)
- Tornado web server - A Python web framework used in Jupyter community

3.35.2 Database

- SQLAlchemy - A Python SQL toolkit
- SQLite3 - a light weight database engine
- Alembic - SQLAlchemy migration tool
- ERAlchemy - Generate entity relation diagrams

3.35.3 Progamming Language

- Python - The major programming language used to develop `nbgrader`
- pytest - Unit test framework
- pytest-cov - Code coverage
- pytest-tornado - Test Tornado server
- black - a Python code formatter

3.35.4 Project Management

- GitHub - a git repository management website
- Travis CI - Continous integration
- Codecov - Code coverage
- Read the Docs - Documentation

3.36 System Architecture Overview

`ngshare` is intended to run as a Kubernetes pod and service outside JupyterHub. In a Kubernetes setup, ngshare is proxied by JupyterHub's proxy service and can be accessed from any JupyterHub user pod. It uses the Hub for authentication.

3.37 Legal & Social Aspects

Our project will be delivered in a way that does not involve deployment on our (the developer's) side, so the users are responsible for deploying the project and setting up terms and conditions regarding their use of our project and collecting their user data.

Our project will be an extension on an existing open source project. The existing project is using the BSD license, which allows anyone to use and modify the software. The open source license disclaims all warranties, so there is not much we can say about the social and legal aspect of the product.

Our project will make a social impact on all current nbgrader users and possibly IT service providers for programming courses. Our project makes it possible to have centralized kubernetes or other container clusters maintained by IT service providers and used by individual programming class instructors. This feature may also let nbgrader be more popular.

3.38 Porting nbextensions to JupyterLab

We have made good progress porting the extensions to JupyterLab, but we are not quite finished. This document contains notes on the progress for all of the extensions.

You can view our progress here.

Here's a link to an existing issue relating to this.

3.38.1 Assignment List

The assignment list JupyterLab extension contains the exact same functionality and layout as the nbextension. After installation, it can be launched by opening the command palette on the left side and searching for Assignment List.

What's Done

- All functionality
- Unit tests
- Styling

What's not Done

- Could improve styling if wanted, but not necessary.
- The modals from validate assignment could use some better styling. Make styling of modals between assignment list and validate assignment consistent.
- Contain the bootstrap CSS. It is affecting the styling of elements outside of the extension.

Code

Files

- `index.ts`
  - Attaches the UI to the main work area

- `assignmentlist.ts`
  - Contains all the logic necessary to display the assignments.
- `handlers.py`
  - Defines the backend of the extension.
  - Uses the nbgrader ExchangeList, ExchangeFetchAssignment, ExchagneFetchFeedback, and ExchangeSubmit classes.

Classes

AssignmentList

- Used to load and display the list of released, downloaded, and submitted assignments.

Assignment

- Creates the rows for each assignment. Each row consists of a link, a span element to display the name of the course, and a button.

Submission

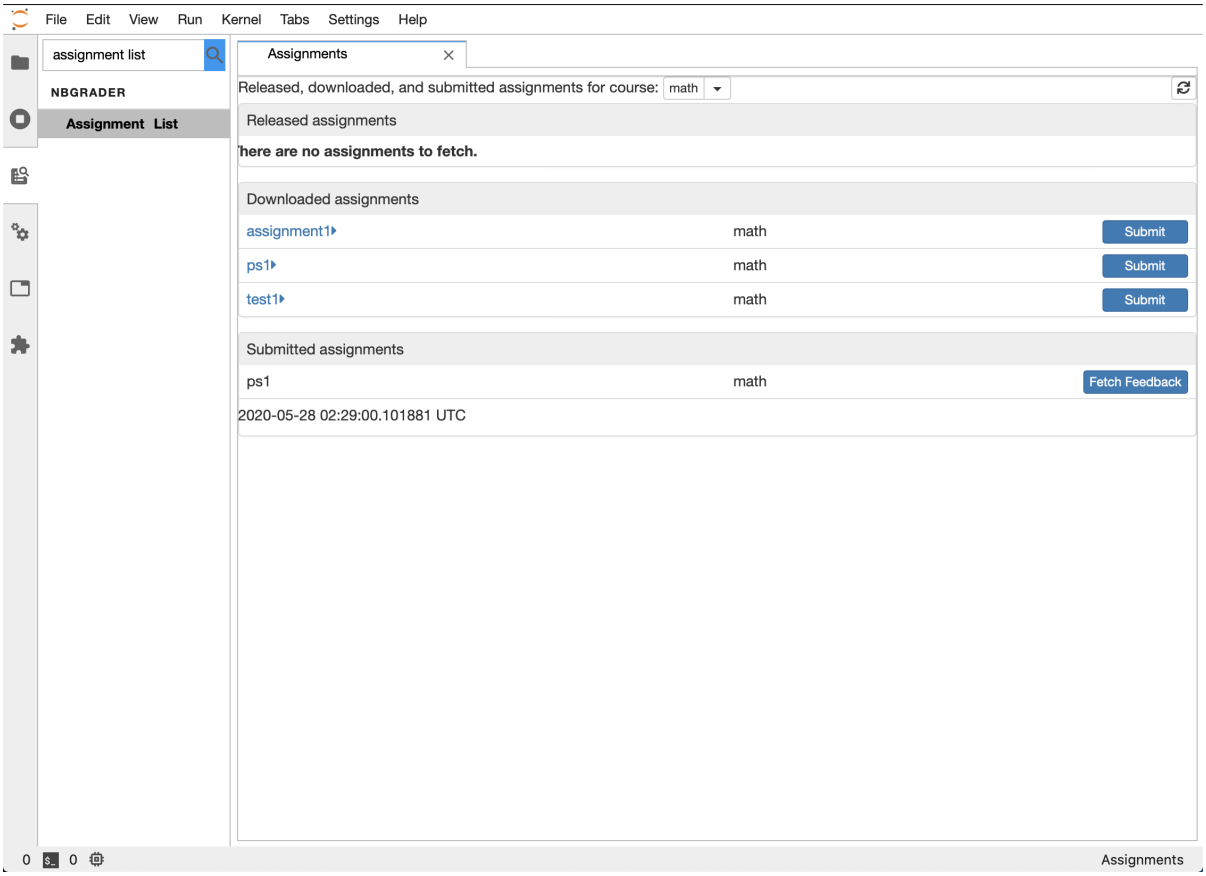
- Makes a submission row which consists of the timestamp and a link to a feedback file if there is any.

Notebook

- Creates a row for each notebook in an assignment. The name of the notebook is a link to open the notebook and each row also contains a button to validate the notebook (run the tests for the notebook).

CourseList

- Used to load and display the course dropdown.
- When you click on a course it switches to to display the assignments for that course.



3.38.2 Create Assignment

In Jupyter Notebooks, the extension put the UI in the cell toolbars. JupyterLab does not have cell toolbars, so we had to decide where to put the interface. We decided on a side panel which shows the nbgrader assignment information for the active notebook.

What's Done

- Everything (extension, styling, tests, etc.)

What's Not Done

- Nothing

Code

Files

- `index.ts`
  - Attaches the UI to a side panel.
- `extension.ts`

  - Contains the UI elements.
- `model.ts`
  - Contains the logic which acts as an intermediary between the UI and the notebook cell metadata.

Classes

CreateAssignmentWidget

- A container for the UI, which can theoretically be attached to any widget, not just a side panel
- Listens to determine which notebook is the current notebook
- Shows the NotebookWidget for the current notebook

NotebookWidget

- Contains the UI associated with a notebook
- Has a NotebookHeaderWidget at the top and a NotebookPanelWidget which takes up the remaining space

NotebookHeaderWidget

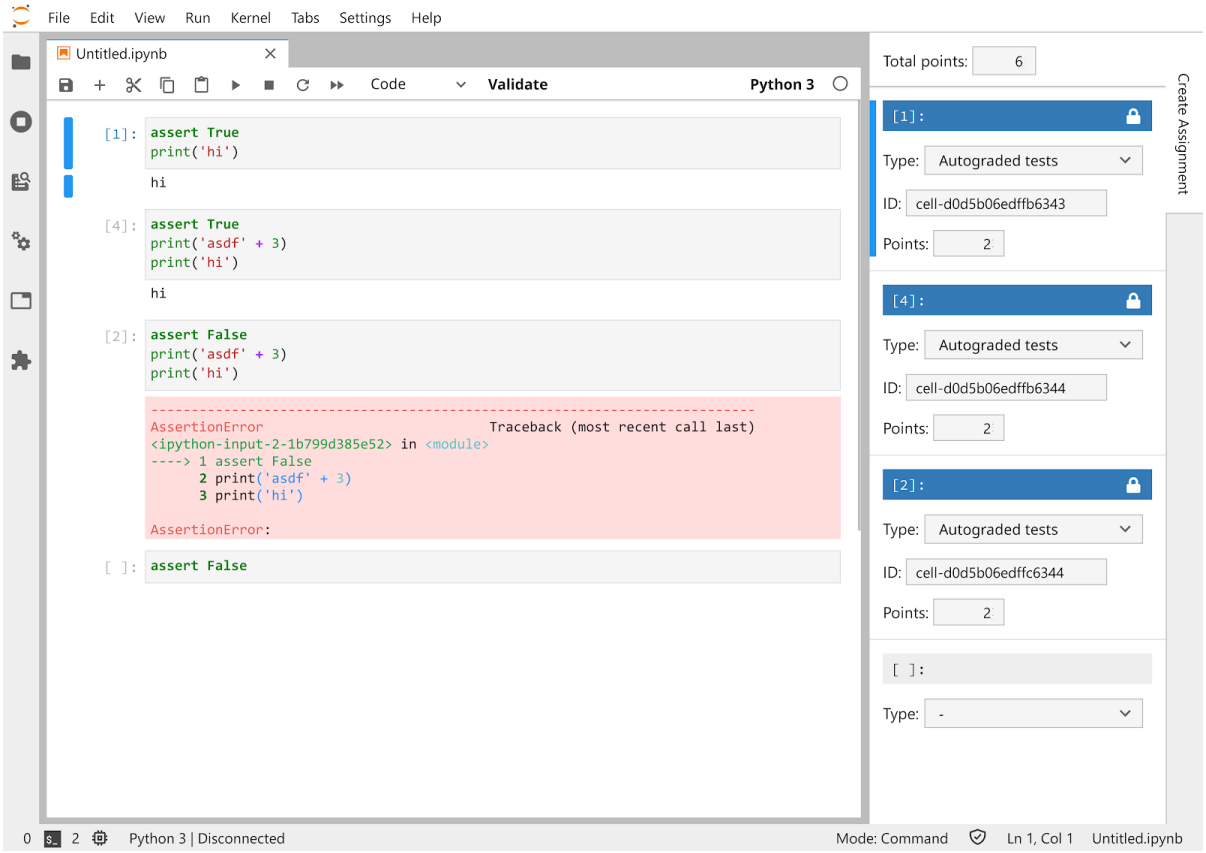
- Currently, only contains the total points for the assignment

NotebookPanelWidget

- Contains a list of CellWidgets to show the assignment information for each cell
- Listens to changes in the notebook cell list
- Adds, removes, reorders, or highlights CellWidgets to synchronize with the notebook

CellWidget

- Contains the UI showing the nbgrader assignment information for one cell
- Reads and writes nbgrader data in the cell metadata



3.38.3 Course List

Same functionality and layout as the course list nbextension. After installation, it can be launched by opening the command palette on the left side and searching for Course List.

What's Done

- All functionality is there
- Unit tests
- Some styling

What's Not Done

- Could use more styling

Code

Files

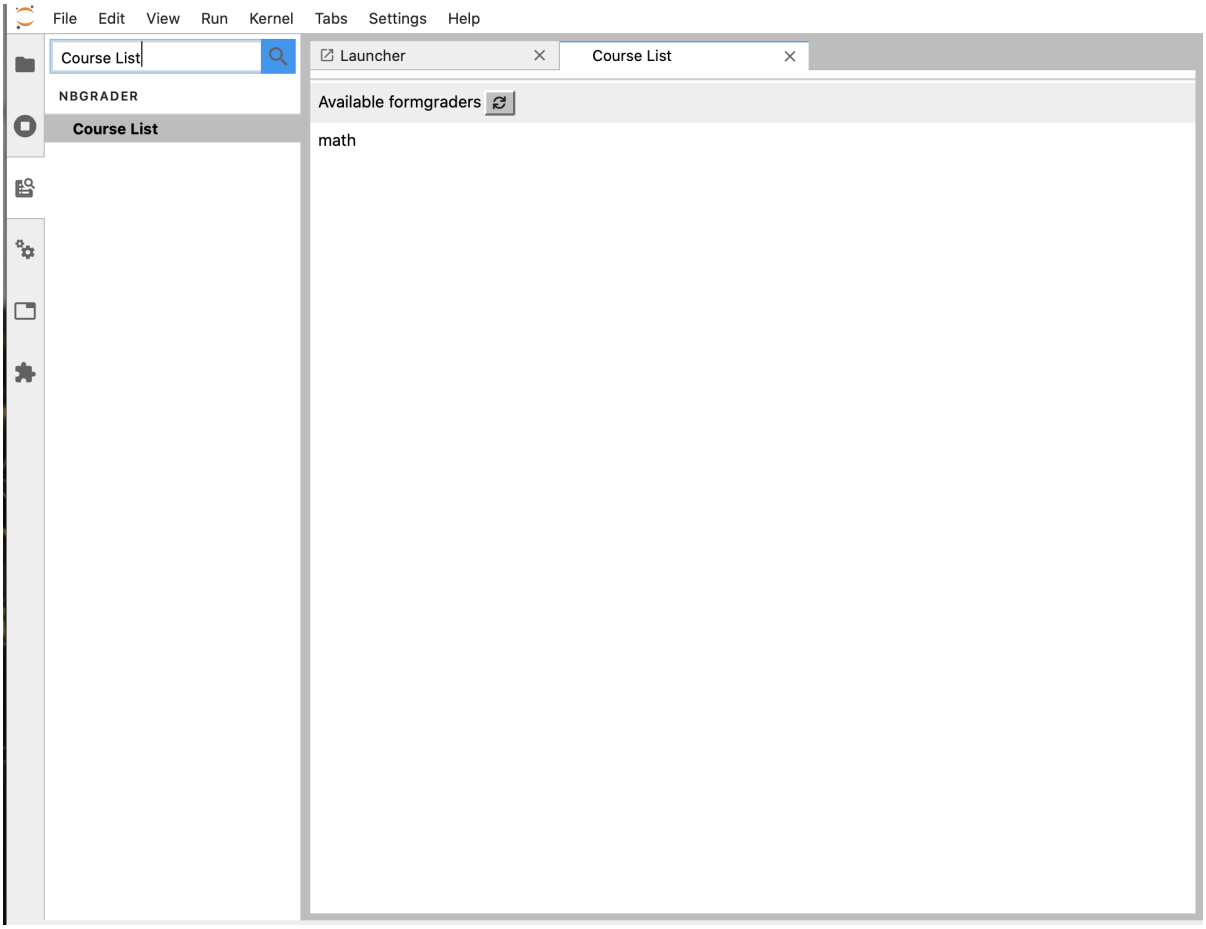
- `index.ts`
  - Attaches the UI to the main work area.

- `courselist.ts`
  - Contains all the logic necessary to display the courses.
- `handers.py`
  - Defines the backend of the extension.

Classes

CourseList

- Loads and displays the list of courses.
- The name of each course is a link to the formgrader for that course.



3.38.4 Formgrader

No work has been done on formgrader. This extension is very different from the others since it is complex and has a stand-alone interface.

What's Done

- Nothing

What's Not Done

- Everything

Possible Plan

- Add launcher and/or command palette entry
- Open formgrader UI in the main area
- Edit appropriate hyperlinks in the UI to open items in JupyterLab instead of Jupyter

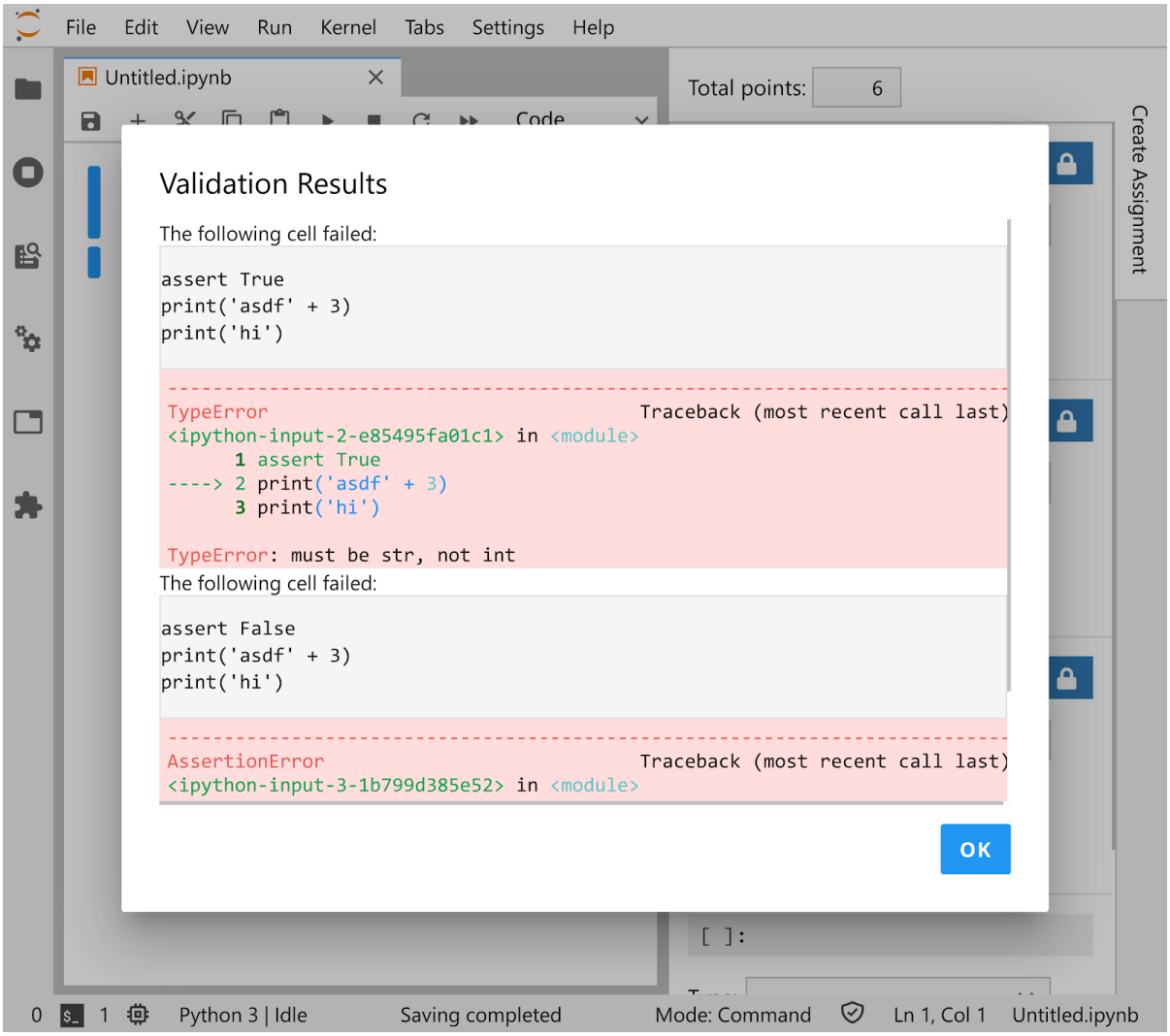
3.38.5 Validate Assignment

What's Done

- All functionality
- Unit tests
- Some styling

What's Not Done

- Styling
  - The modals could use some better styling.
  - Make styling of modals between assignment list and validate assignment consistent.


File Edit View Run Kernel Tabs Settings Help
Untitled.ipynb
Code
Total points: 6
Create Assignment
Validation Results
The following cell failed:
assert True
print('asdf' + 3)
print('hi')
TypeError Traceback (most recent call last)
<ipython-input-2-e85495fa01c1> in <module>
1 assert True
----> 2 print('asdf' + 3)
3 print('hi')
TypeError: must be str, not int
The following cell failed:
assert False
print('asdf' + 3)
print('hi')
AssertionError Traceback (most recent call last)
<ipython-input-3-1b799d385e52> in <module>
OK
[ ]:
0 1 Python 3 | Idle Saving completed Mode: Command Ln 1, Col 1 Untitled.ipynb