Set up your dev environment

Let's get up and running with a dev environment, so you can make code changes without affecting production. By the end of this guide, you will have:

1. Set up Docker to run the containerized dev environment
2. Cloned the public protohaven_api repository where our code is hosted
3. Set up pre-commit to automatically format and check code changes for problems
4. Initialized NocoDB - our development database - to run locally with testing data
5. Verified that both CLI commands and the web service works.
6. Successfully made edits to both the CLI and web service.

Install Docker

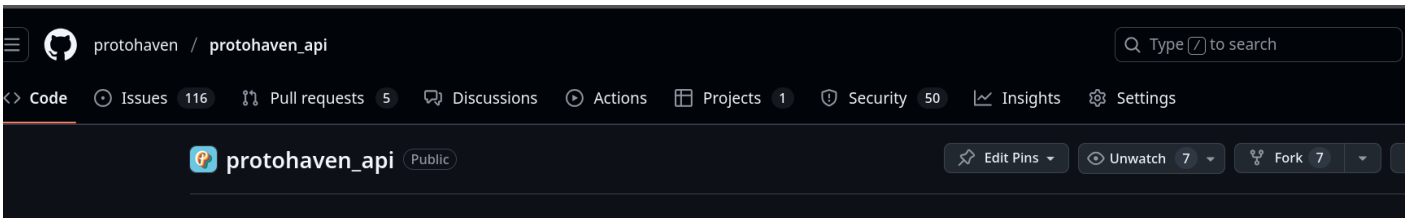
We rely heavily on using docker containers to run the various services in a dev environment - make sure Docker is installed first before continuing.

Once docker is installed, run `sudo docker run hello-world` and verify that it pulls a container image, starts the container, prints a status message and exits with no errors:

```
➜  ~ sudo docker run hello-world
[sudo] password for semartin:
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
17eec7bbc9d7: Pull complete
Digest: sha256:6dc565aa630927052111f823c303948cf83670a3903ffa3849f1488ab517f891
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/

➜  ~
```

Clone the repository & set up pre-commit

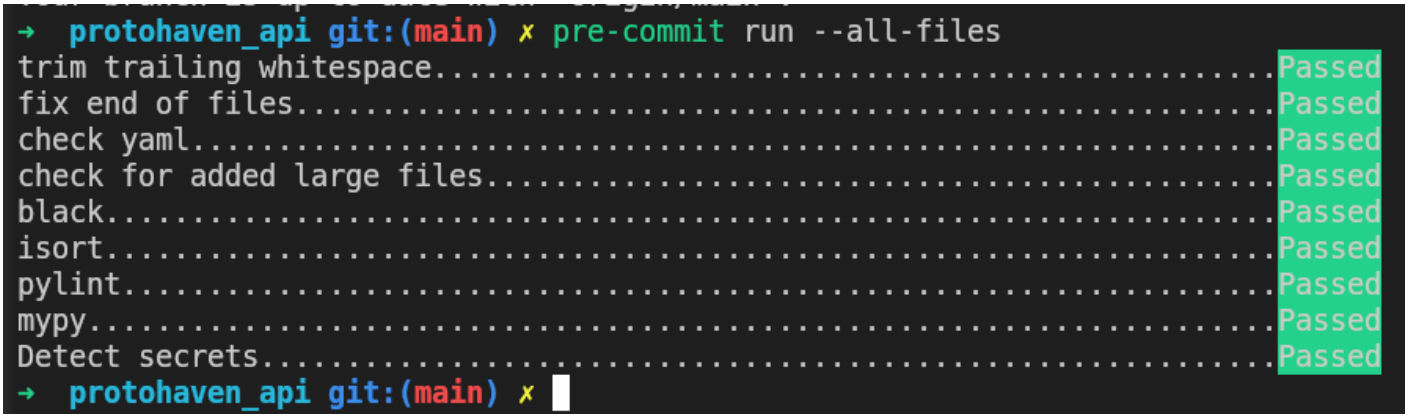
Next, let's grab the code and set up pre-commit, which autoformats our code and identifies errors before we try to submit changes. Log into GitHub, then go to https://github.com/protohaven/protohaven_api. You should see a "fork" option on the repository, towards the right of the screen:



Follow the onscreen process for forking the repository, then perform the steps below.

```
# Clone the repository - be sure to substitute your username!
git clone git@github.com:<YOUR_GITHUB_USERNAME>/protohaven_api.git

# Set up a virtual environment so we can run tests and pre-commit stuff
python3 -m venv venv
source ./venv/bin/activate
pip install -r requirements.txt
pip install -e .

# Set up pre-commit validation hooks
pip install pre-commit
cd protohaven_api && pre-commit install
```

After pre-commit finishes installing, run it on all files (with `pre-commit run --all-files`) and verify everything passes:



Load Data (NocoDB)

We use NocoDB to provide local testing data. **Nocodb must be running in order to run other local dev services** like Flask and the svelte frontend, **and we must prefill it** with tables and dev data so that it behaves as expected locally.

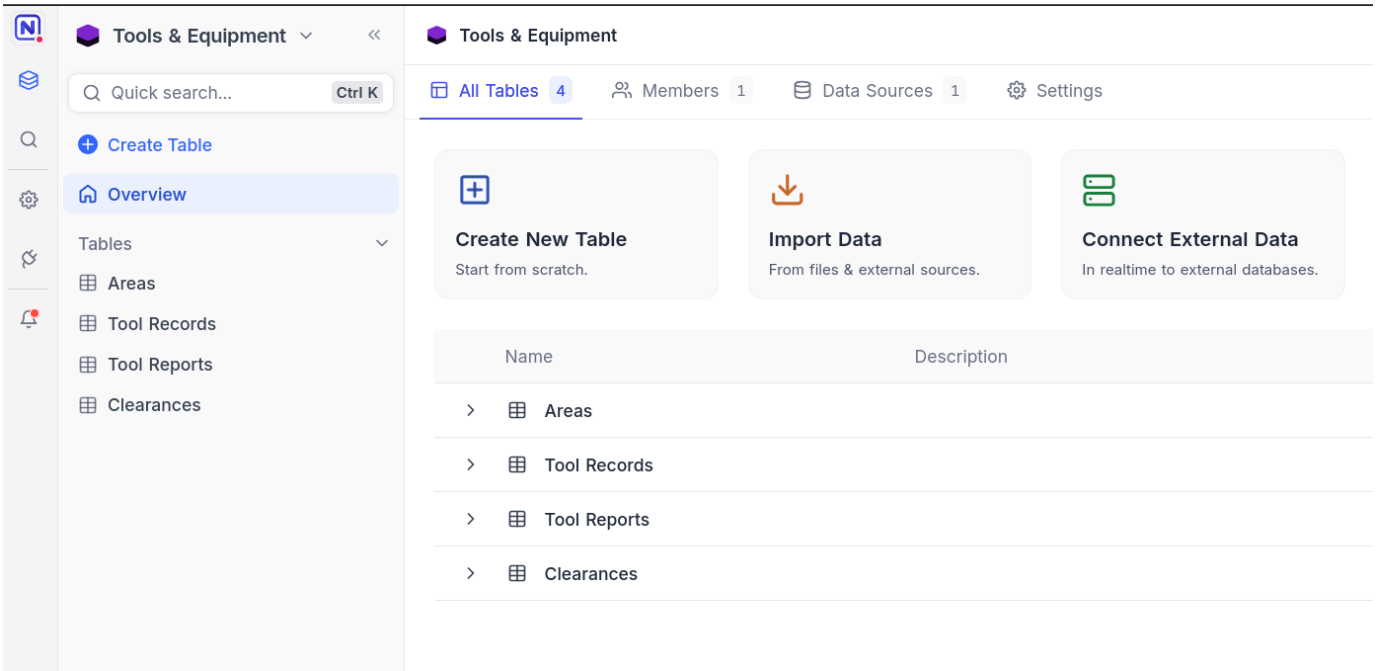
To initialize NocoDB with testing data - **or wipe and re-populate the data if it's already installed** - run this command:

```
# Must run in the root of the protohaven_api repository, or it will fail
cd path/to/protohaven_api

./nocodb/init_or_update_nocodb.sh
```

Follow the prompts in the script, which should have you browse to http://localhost:9090 and login with:

- Username: admin@example.com
- Password: `password`

Verify you see something like this and can click around into various bases and tables and see test data:



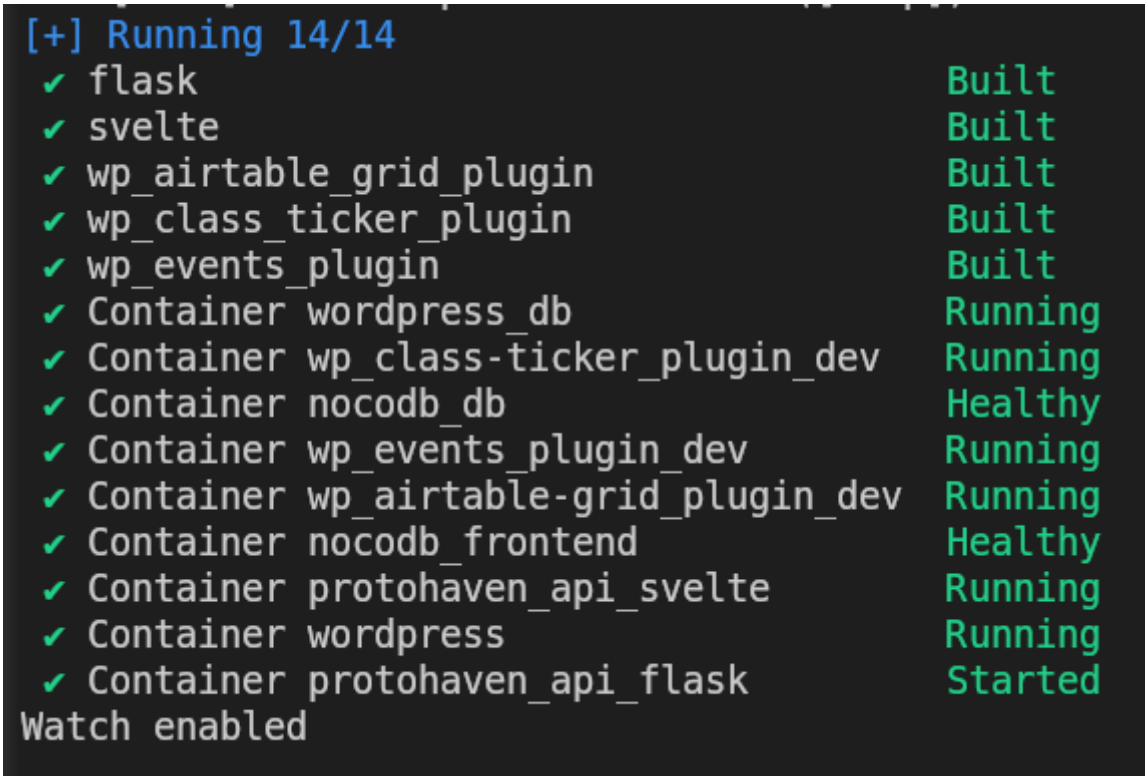
Start services

With NocoDB now up, let's run the other containers that host our various services:

```
docker compose up --build --watch
```

Wait for the build and setup process to finish up - this may take a few minutes.

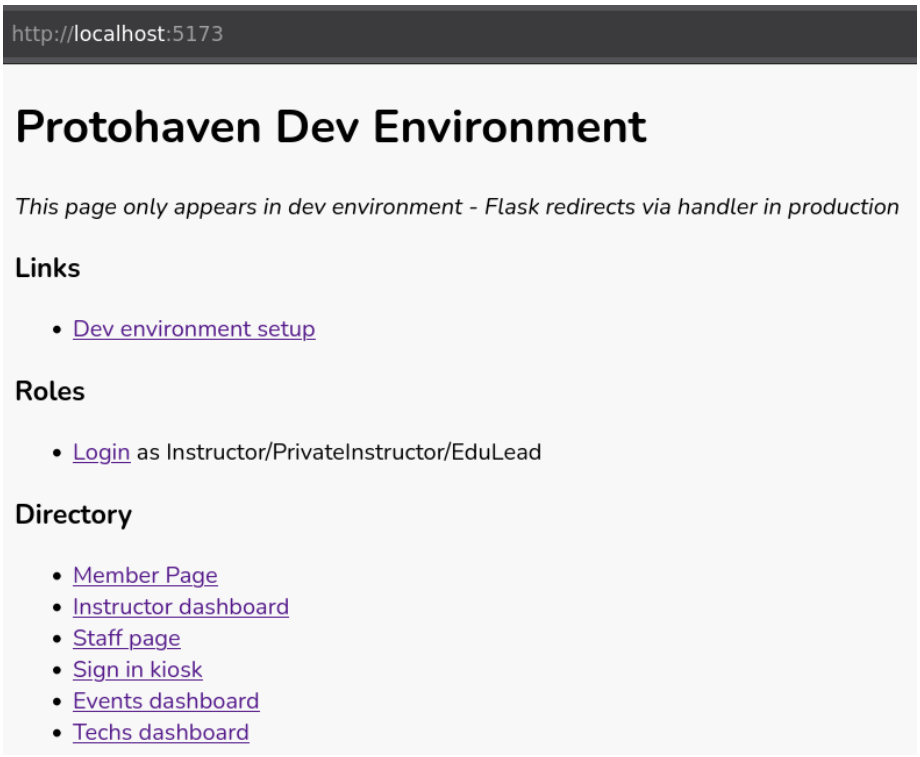
You'll know it's done when you see something like this in your console:



> Take particular note of `Watch enabled` - if that's not present, things won't auto reload when you make a change.

> For dev work in specific areas (e.g. working with the svelte webpages) you can start a subset of containers to reduce excess load, e.g. `docker compose up flask nocodb_frontend nocodb_db svelte --build --watch`

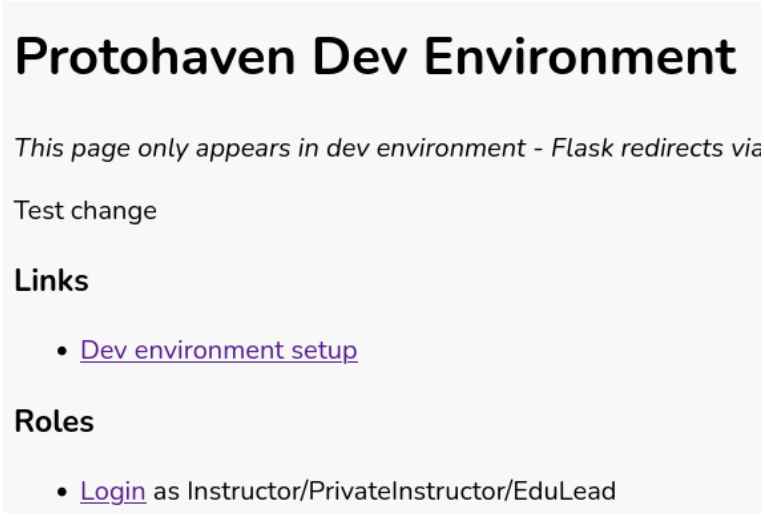
Now browse to: https://localhost:5173/ and verify you can access the dev landing page:



Using your editor of choice, open `svelte/src/routes/+page.svelte` in the repository you cloned. Add a little text to the file, then save it.

In your console, you'll see `Syncing service "svelte" after 1 changes were detected` as docker notices the changed file.

Reload the landing page and confirm the change propagated:



Run the CLI

Our codebase includes an extensible CLI. We issue these commands on a regular schedule with Cronicle, and can run one-offs for troubleshooting and special actions.

Let's make sure the CLI works so we can test local changes to CLI commands:

```
# Test your setup - you should see a YAML formatted message reminding instructors to schedule classes
docker compose exec flask python3 -m protohaven_api.cli gen_instructor_schedule_reminder
```

> **Important:** All services must be running in order to execute CLI commands

Verify you see something like the following output (no errors):

```
→ protohaven_api git:(main) ✗ docker compose exec flask python3 -m protohaven_api.cli gen_instructor_schedule_reminder
INFO:cli:Mode is dev

WARNING:integrations.data.dev_connector:DevConnector in use; mutations will not reach production
INFO:class_automation.builder:Already scheduled for interval 2025-11-15 11:48:09.411839-05:00 - 2025-12-15 11:48:09.411839-05:00: set()
[]

INFO:cli.classes:Generated 0 notification(s)
```

Let's confirm that changes to CLI code are runnable. Edit `protohaven_api/commands/classes.py`, adding a log statement to the `gen_instructor_schedule_reminder` method

```
    )
    def gen_instructor_schedule_reminder(self, args, _):
        """Reads the list of instructors from Airtable and generates
        reminder comms to all instructors, plus the #instructors discord,
        to propose additional class scheduling times"""

64      log.info("Hello world")

        start = (
            safe_parse_datetime(args.start)
            if args.start
            else tznow() + datetime.timedelta(days=30)
        )
```

Save the file, then rerun the exec command above. Verify that you see `INFO:cli.classes:Hello world` in the console output.

Run Tests

We use pytest for unit testing code changes.

Run `python -m pytest -vv` and observe all tests passing, possibly with some deprecation warnings:

```
============================================= warnings summary =============================================
.devenv/state/venv/lib/python3.12/site-packages/dateutil/tz/tz.py:37
  /home/semartin/Documents/protohaven_api/.devenv/state/venv/lib/python3.12/site-packages/dateutil/tz/tz.py:37: Dep
recationWarning: datetime.datetime.utcfromtimestamp() is deprecated and scheduled for removal in a future version.
Use timezone-aware objects to represent datetimes in UTC: datetime.datetime.fromtimestamp(timestamp, datetime.UTC).
    EPOCH = datetime.datetime.utcfromtimestamp(0)

.devenv/state/venv/lib/python3.12/site-packages/discord/player.py:28
  /home/semartin/Documents/protohaven_api/.devenv/state/venv/lib/python3.12/site-packages/discord/player.py:28: Dep
recationWarning: 'audioop' is deprecated and slated for removal in Python 3.13
    import audioop

-- Docs: https://docs.pytest.org/en/stable/how-to/capture-warnings.html
================================== 731 passed, 1 skipped, 2 warnings in 7.54s ==================================
```

> **Protip:** You can optionally use `-k test_substring` as an argument to run a specific test.

Well Done

Congrats on making it through dev environment setup! Now let's put it to work and make your first contribution.

Revision #55
Created 18 January 2025 17:40:26 by Scott Martin
Updated 4 January 2026 15:18:38 by Scott Martin