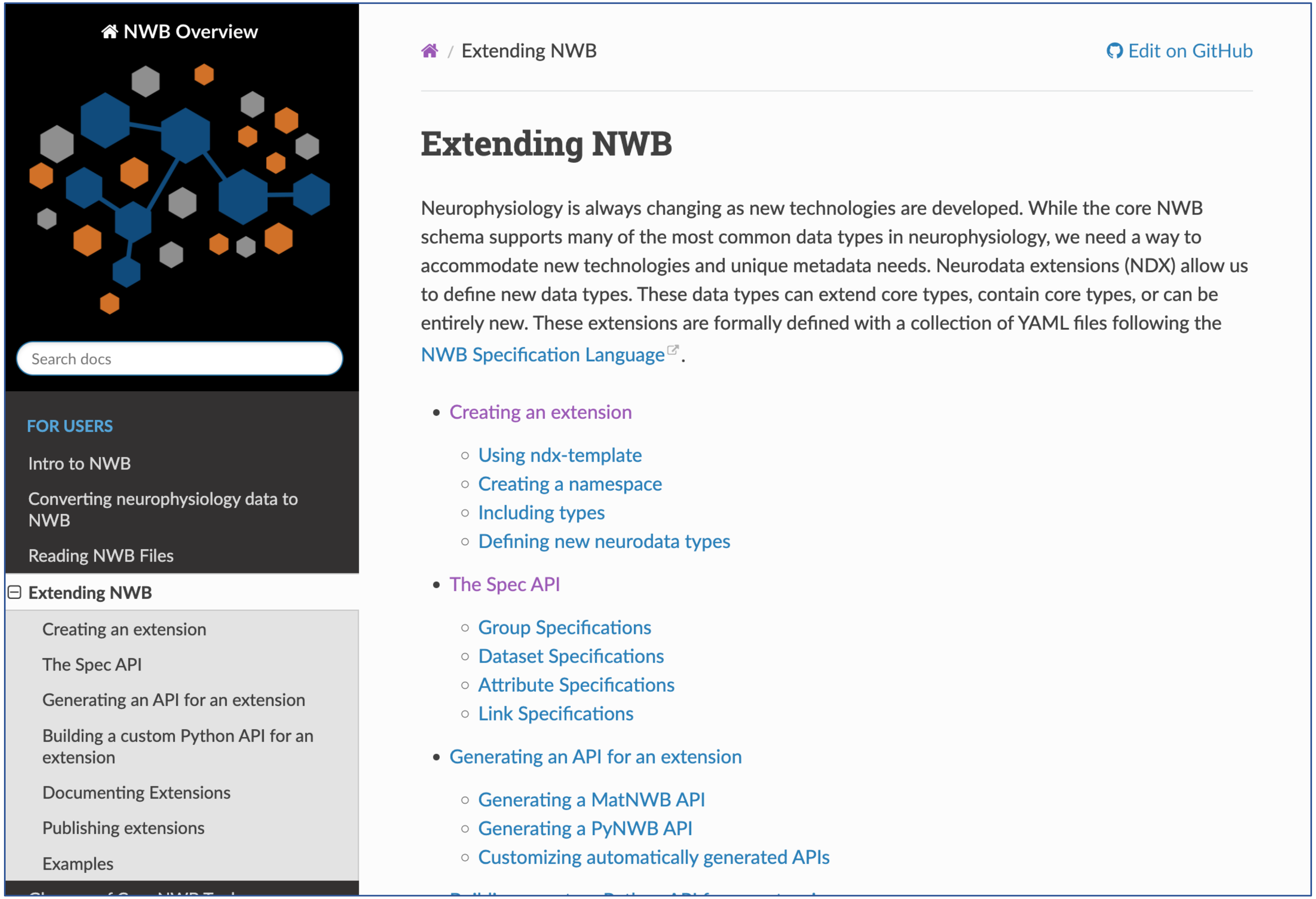
Extensions to Neurodata Without Borders

Ryan Ly

Scientific Data Division
Lawrence Berkeley National Laboratory

Outline

- What are NWB extensions?
- When would you use extensions?
- How do you write an extension?
- How do you share your extension?



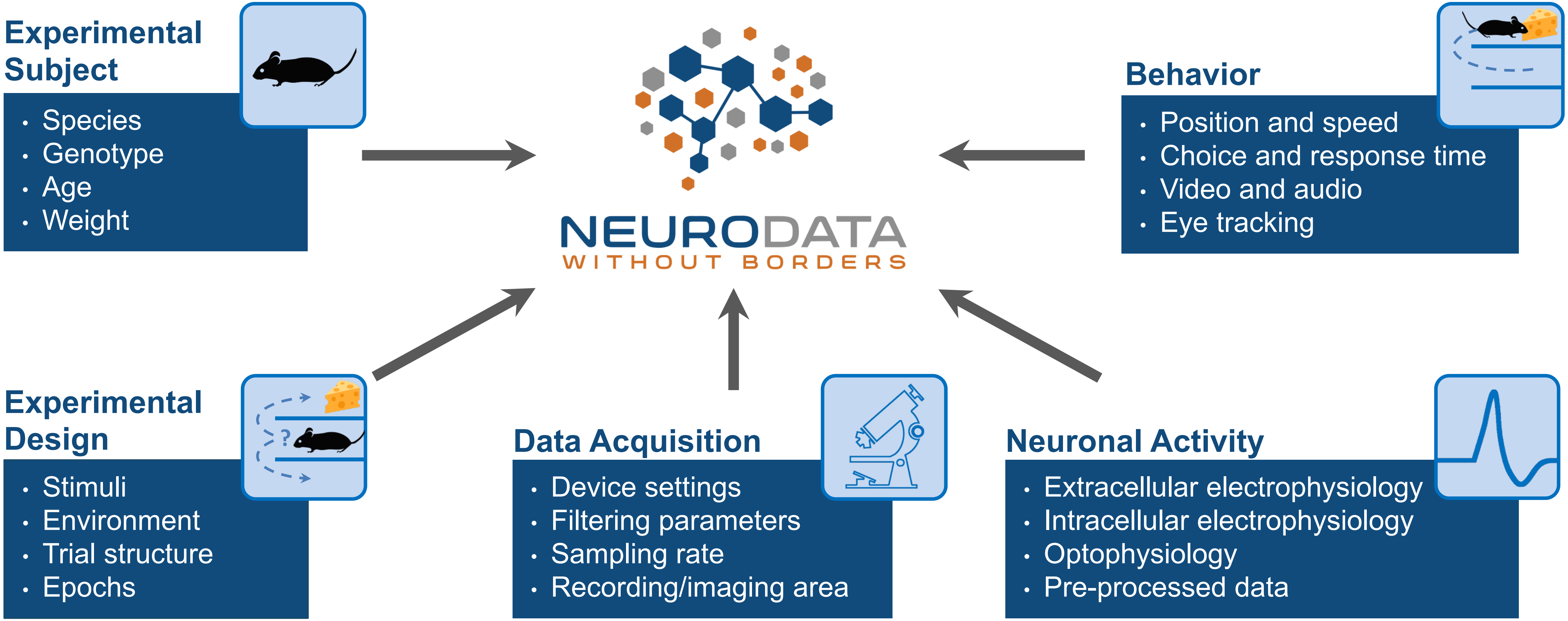
Extending NWB

Neurophysiology is always changing as new technologies are developed. While the core NWB schema supports many of the most common data types in neurophysiology, we need a way to accommodate new technologies and unique metadata needs. Neurodata extensions (NDX) allow us to define new data types. These data types can extend core types, contain core types, or can be entirely new. These extensions are formally defined with a collection of YAML files following the NWB Specification Language.

- Creating an extension
  - Using ndx-template
  - Creating a namespace
  - Including types
  - Defining new neurodata types
- The Spec API
  - Group Specifications
  - Dataset Specifications
  - Attribute Specifications
  - Link Specifications
- Generating an API for an extension
  - Generating a MatNWB API
  - Generating a PyNWB API
  - Customizing automatically generated APIs

https://nwb-overview.readthedocs.io → "Extending NWB"

NWB stores most types of neurophysiology data and metadata



Experimental Subject

- Species
- Genotype
- Age
- Weight

Behavior

- Position and speed
- Choice and response time
- Video and audio
- Eye tracking

Experimental Design

- Stimuli
- Environment
- Trial structure
- Epochs

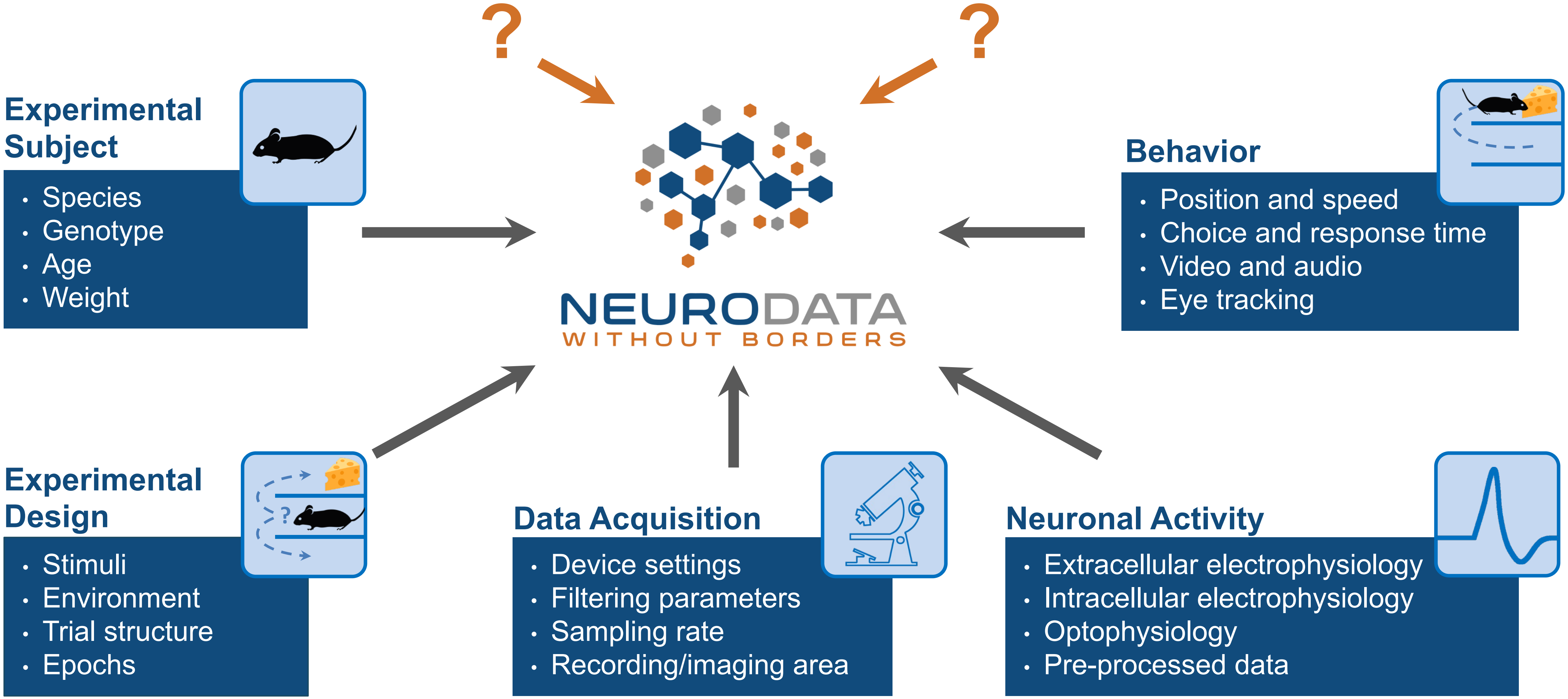
Data Acquisition

- Device settings
- Filtering parameters
- Sampling rate
- Recording/imaging area

Neuronal Activity

- Extracellular electrophysiology
- Intracellular electrophysiology
- Optophysiology
- Pre-processed data

How does NWB handle new data types?



Experimental Subject

- Species
- Genotype
- Age
- Weight

Behavior

- Position and speed
- Choice and response time
- Video and audio
- Eye tracking

Experimental Design

- Stimuli
- Environment
- Trial structure
- Epochs

Data Acquisition

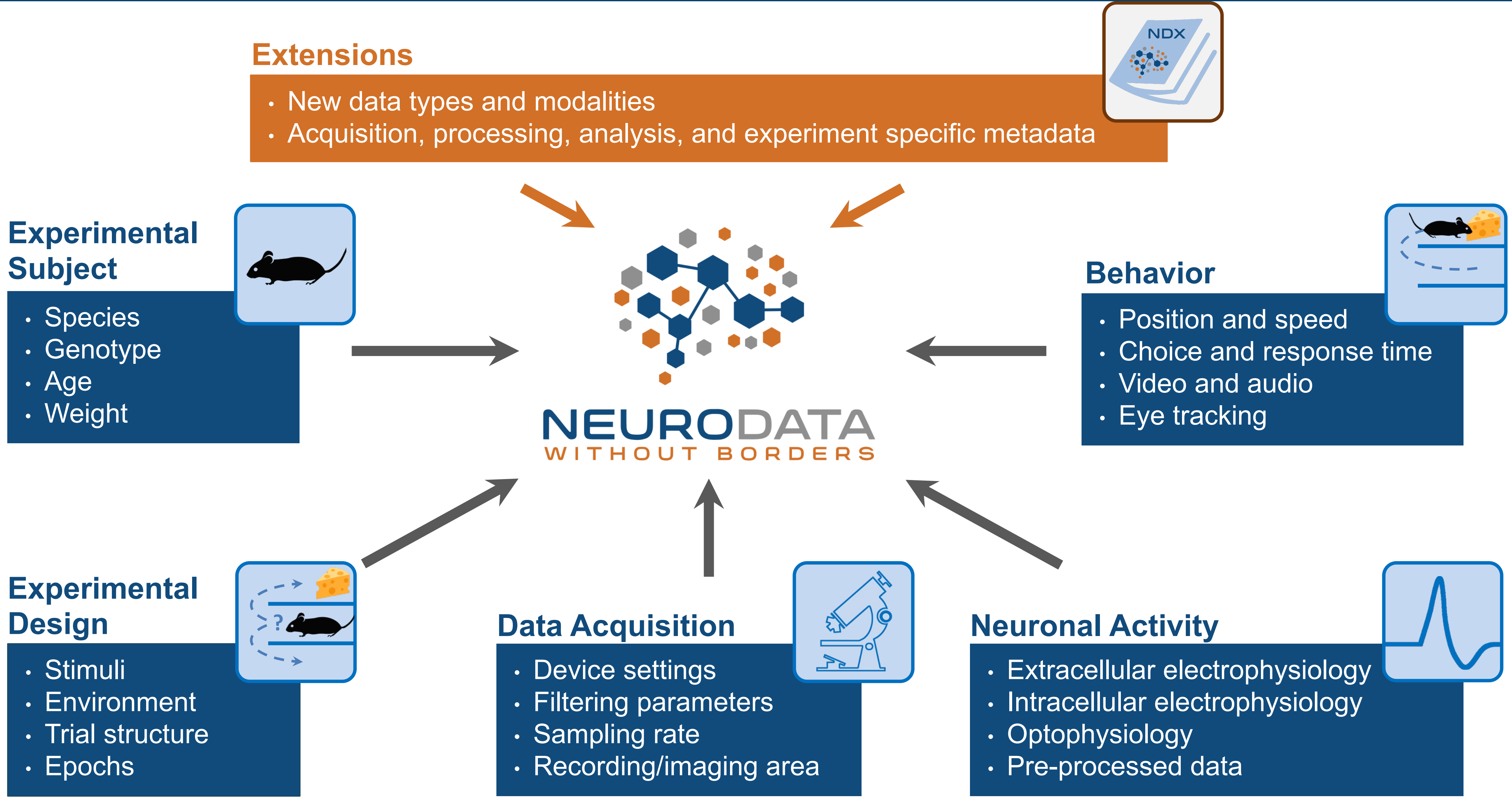
- Device settings
- Filtering parameters
- Sampling rate
- Recording/imaging area

Neuronal Activity

- Extracellular electrophysiology
- Intracellular electrophysiology
- Optophysiology
- Pre-processed data

Data not covered by NWB

- Metadata specific to an acquisition system
  - probe properties, microscope properties
- Metadata/data specific to a preprocessing tool
  - spike sorting parameters and metrics, pose estimation training data and model parameters
- Metadata specific to an experiment or lab
  - maze structure, custom stimuli, lab-specific subject or experimental condition information
- New or unsupported data types and modalities
  - pose estimation results, fiber photometry, functional ultrasound imaging



Extensions

- New data types and modalities
- Acquisition, processing, analysis, and experiment specific metadata

NEURODATA WITHOUT BORDERS

Experimental Subject

- Species
- Genotype
- Age
- Weight

Behavior

- Position and speed
- Choice and response time
- Video and audio
- Eye tracking

Experimental Design

- Stimuli
- Environment
- Trial structure
- Epochs

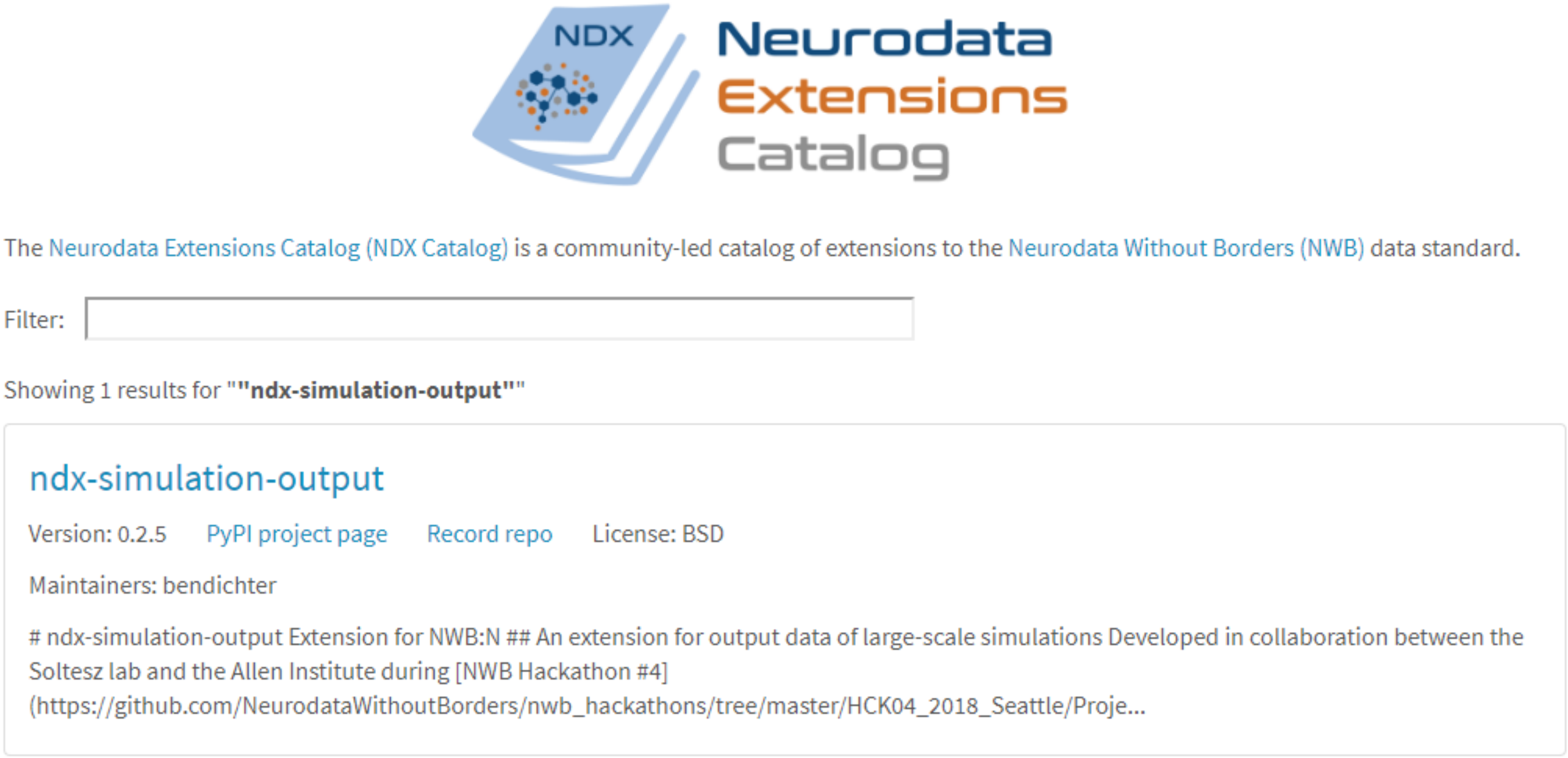
Data Acquisition

- Device settings
- Filtering parameters
- Sampling rate
- Recording/imaging area

Neuronal Activity

- Extracellular electrophysiology
- Intracellular electrophysiology
- Optophysiology
- Pre-processed data

Example NWB Extensions in NDX Catalog: https://nwb-extensions.github.io/



The Neurodata Extensions Catalog (NDX Catalog) is a community-led catalog of extensions to the Neurodata Without Borders (NWB) data standard.

Filter:

Showing 1 results for ""**ndx-simulation-output**""

ndx-simulation-output

Version: 0.2.5 PyPI project page Record repo License: BSD

Maintainers: bendichter

ndx-simulation-output Extension for NWB:N ## An extension for output data of large-scale simulations Developed in collaboration between the Soltesz lab and the Allen Institute during [NWB Hackathon #4] (https://github.com/NeurodataWithoutBorders/nwb_hackathons/tree/master/HCK04_2018_Seattle/Proje...

Example NWB Extensions

Published in NDX Catalog:

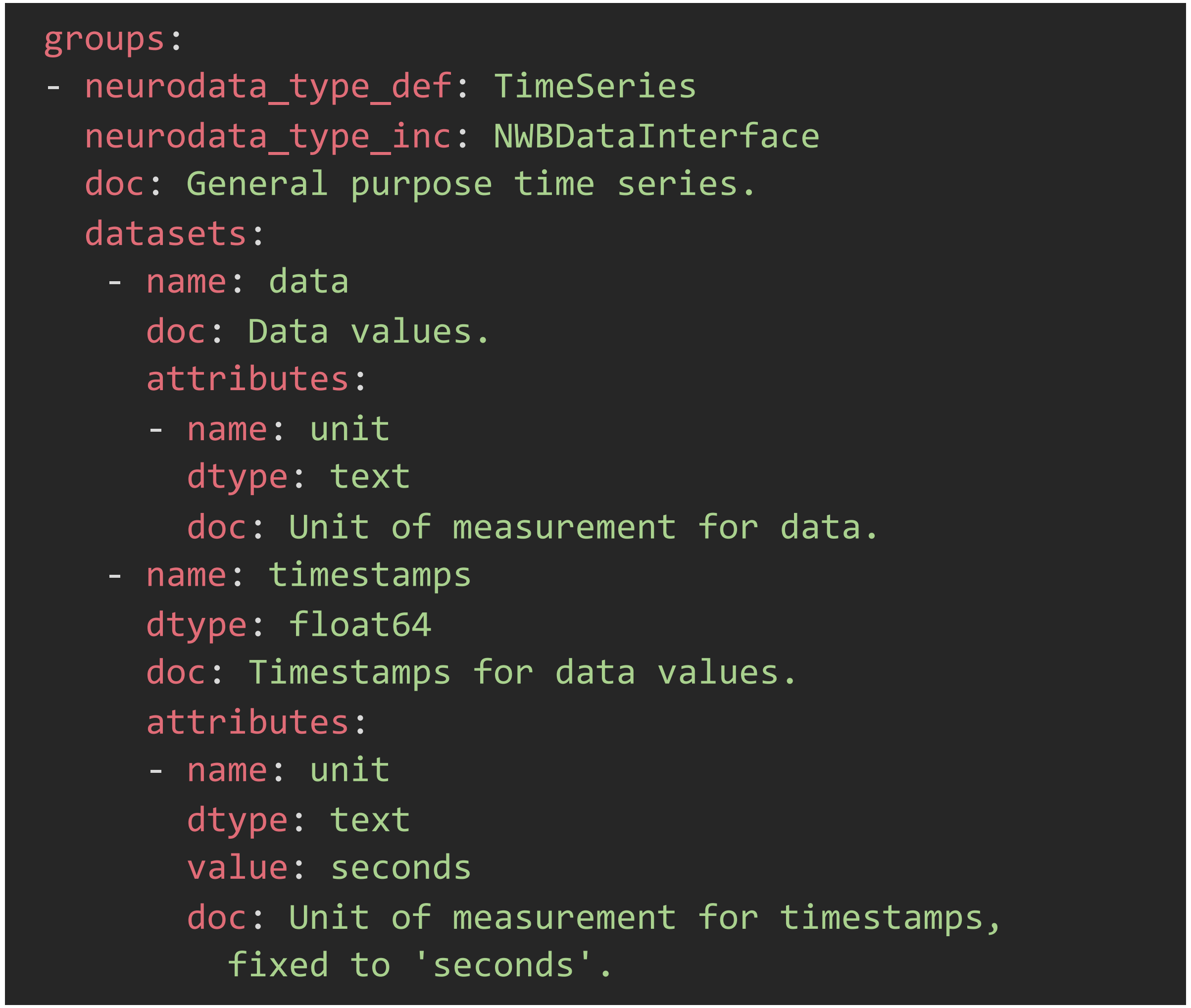
- ndx-miniscope
- ndx-simulation-output
- ndx-ecog
- ndx-fret
- ndx-events
- ndx-nirs
- ndx-icephys-meta (merged to core in NWB 2.4)
- ndx-odor
- ndx-acquisition-module
- ndx-photometry
- ndx-extract
- ndx-sound
- ndx-hierarchical-behavior-record

Released or in progress:

- ndx-beadl
- ndx-optogenetics
- ndx-mies
- ndx-bipolar-scheme
- ndx-franklab-novela
- ndx-genotype
- ndx-csd
- ndx-aibs-ecephys
- ndx-ellipse-eye-tracking
- ndx-aibs-stimulus-template
- ndx-aibs-ophys-event-detection

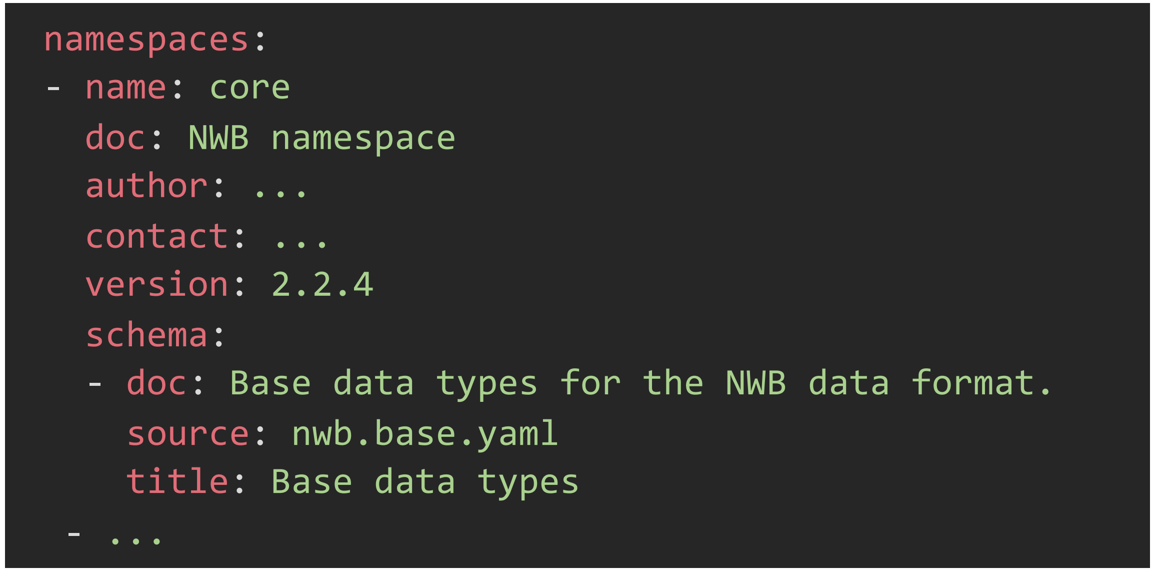
Define core data types with the NWB spec language

nwb.base.yaml



```yaml
groups:
- neurodata_type_def: TimeSeries
  neurodata_type_inc: NWBDataInterface
  doc: General purpose time series.
  datasets:
   - name: data
     doc: Data values.
     attributes:
     - name: unit
       dtype: text
       doc: Unit of measurement for data.
   - name: timestamps
     dtype: float64
     doc: Timestamps for data values.
     attributes:
     - name: unit
       dtype: text
       value: seconds
       doc: Unit of measurement for timestamps,
         fixed to 'seconds'.
```

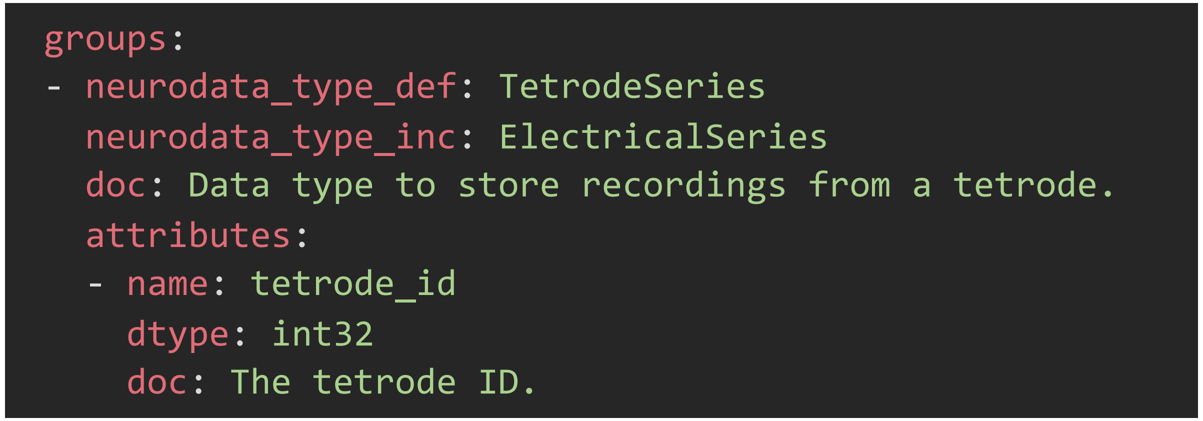
nwb.namespace.yaml



```yaml
namespaces:
- name: core
  doc: NWB namespace
  author: ...
  contact: ...
  version: 2.2.4
  schema:
  - doc: Base data types for the NWB data format.
    source: nwb.base.yaml
    title: Base data types
 - ...
```

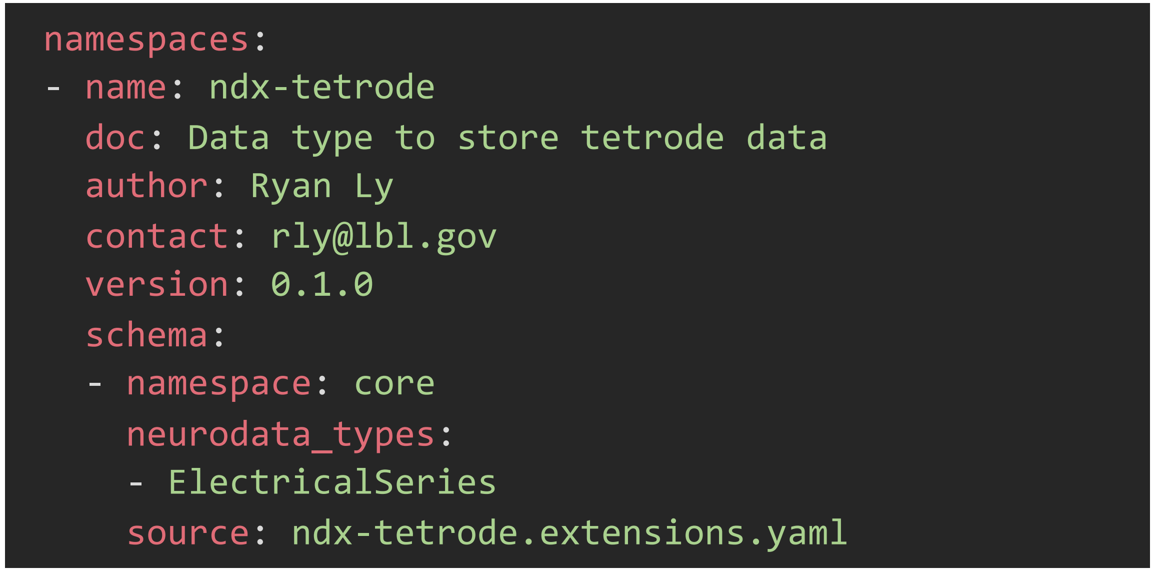
Define custom types with the NWB spec language

ndx-tetrode.extensions.yaml



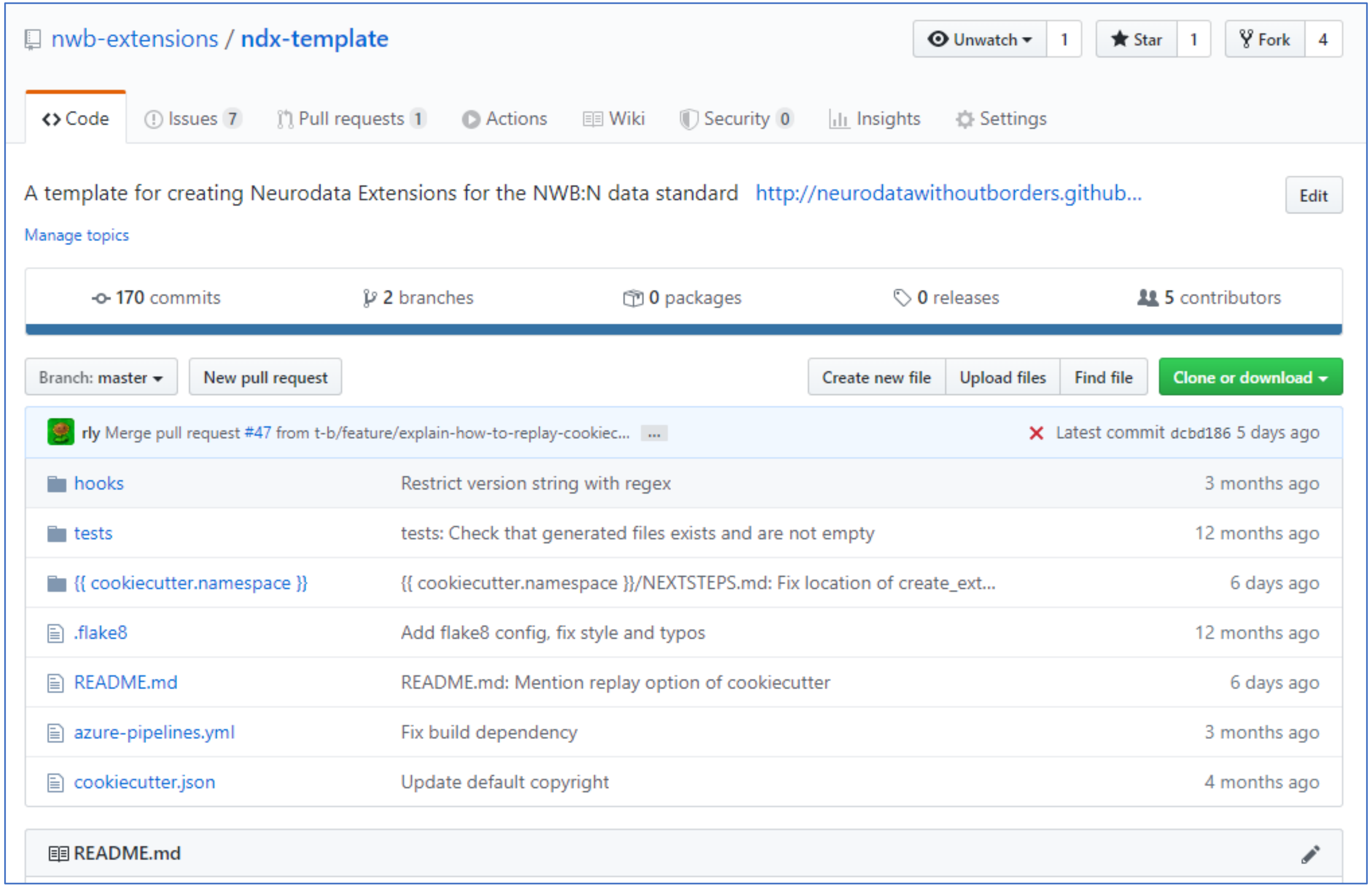
```yaml
groups:
- neurodata_type_def: TetrodeSeries
  neurodata_type_inc: ElectricalSeries
  doc: Data type to store recordings from a tetrode.
  attributes:
  - name: tetrode_id
    dtype: int32
    doc: The tetrode ID.
```

ndx-tetrode.namespace.yaml



```yaml
namespaces:
- name: ndx-tetrode
  doc: Data type to store tetrode data
  author: Ryan Ly
  contact: rly@lbl.gov
  version: 0.1.0
  schema:
  - namespace: core
    neurodata_types:
    - ElectricalSeries
    source: ndx-tetrode.extensions.yaml
```

Use “ndx-template” to start developing extension



A template for creating Neurodata Extensions for the NWB:N data standard http://neurodatawithoutborders.github...

rly Merge pull request #47 from t-b/feature/explain-how-to-replay-cookiec... … ✕ Latest commit dcbd186 5 days ago

hooks	Restrict version string with regex	3 months ago
tests	tests: Check that generated files exists and are not empty	12 months ago
{{ cookiecutter.namespace }}	{{ cookiecutter.namespace }}/NEXTSTEPS.md: Fix location of create_ext...	6 days ago
.flake8	Add flake8 config, fix style and typos	12 months ago
README.md	README.md: Mention replay option of cookiecutter	6 days ago
azure-pipelines.yml	Fix build dependency	3 months ago
cookiecutter.json	Update default copyright	4 months ago

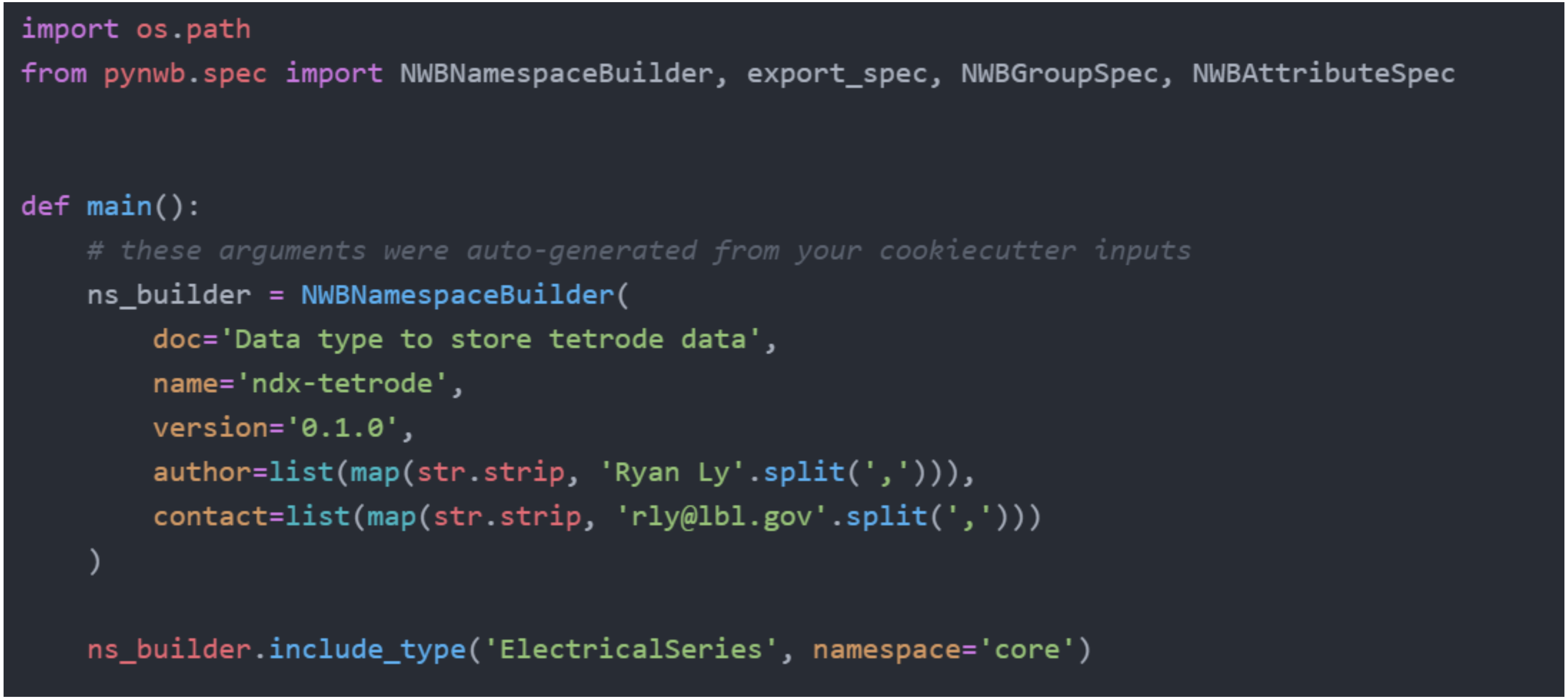
README.md

https://github.com/nwb-extensions/ndx-template

Use “ndx-template” to start developing extension

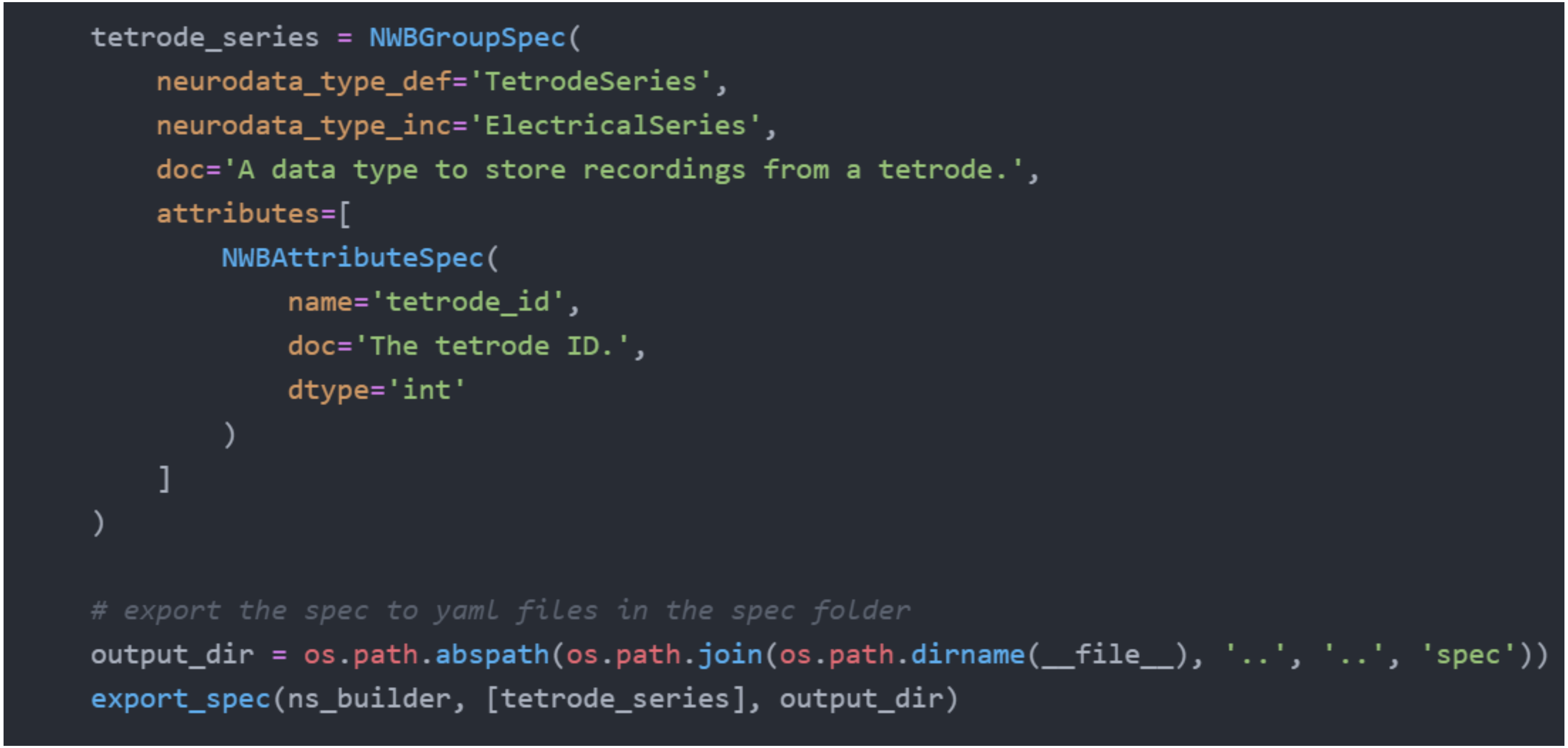
```
$ cookiecutter gh:nwb-extensions/ndx-template
namespace [ndx-my-namespace]: ndx-tetrode
description [An NWB extension]: Data type to store tetrode data
author [My Name]: Ryan Ly
email [my_email@example.com]: rly@lbl.gov
github_username [myname]: rly
…
Success! Directory ndx-tetrode was created with a skeleton for your new
NWB extension.
```

Use PyNWB API to generate extension YAML files



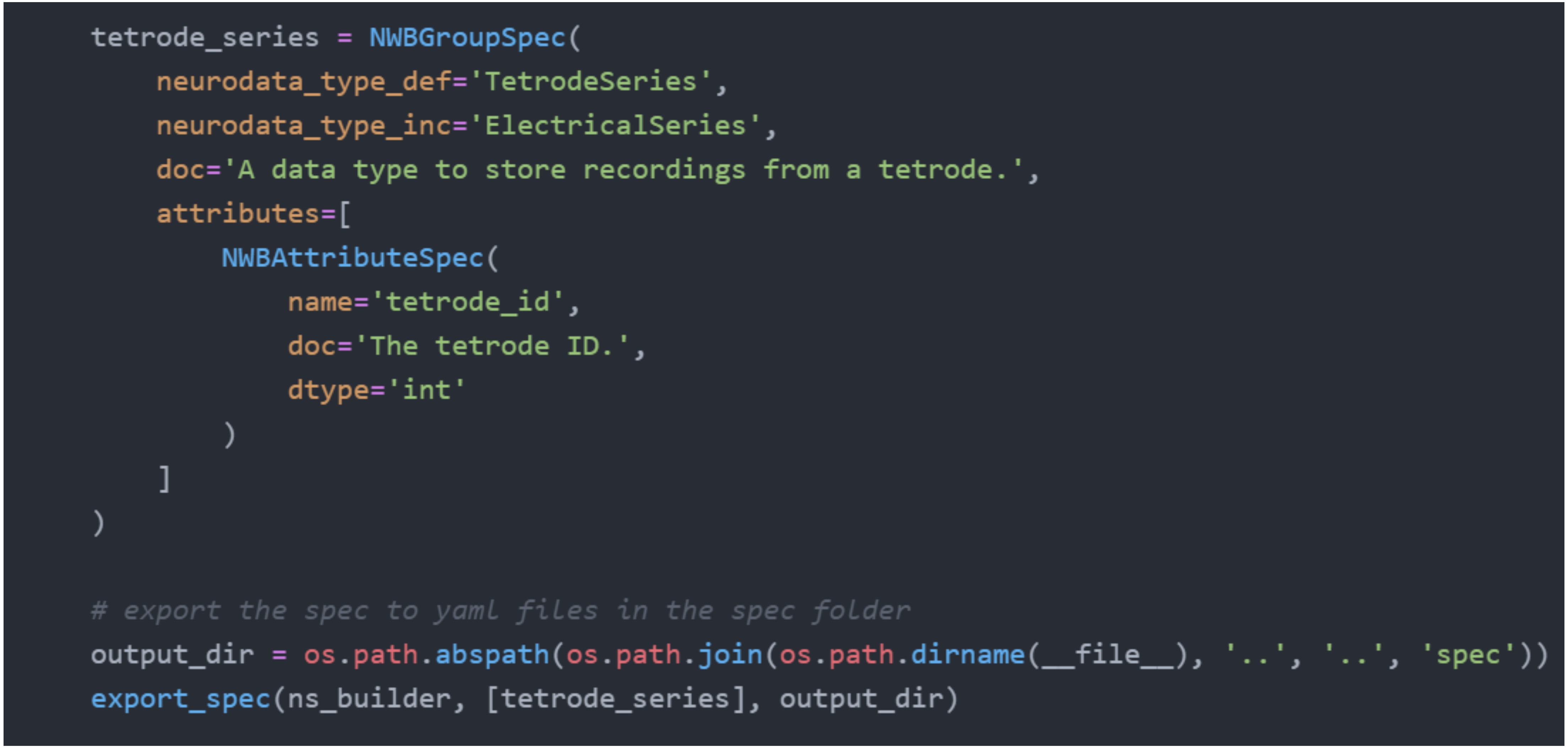
```python
import os.path
from pynwb.spec import NWBNamespaceBuilder, export_spec, NWBGroupSpec, NWBAttributeSpec


def main():
    # these arguments were auto-generated from your cookiecutter inputs
    ns_builder = NWBNamespaceBuilder(
        doc='Data type to store tetrode data',
        name='ndx-tetrode',
        version='0.1.0',
        author=list(map(str.strip, 'Ryan Ly'.split(','))),
        contact=list(map(str.strip, 'rly@lbl.gov'.split(',')))
    )

    ns_builder.include_type('ElectricalSeries', namespace='core')
```

Use PyNWB API to generate extension YAML files



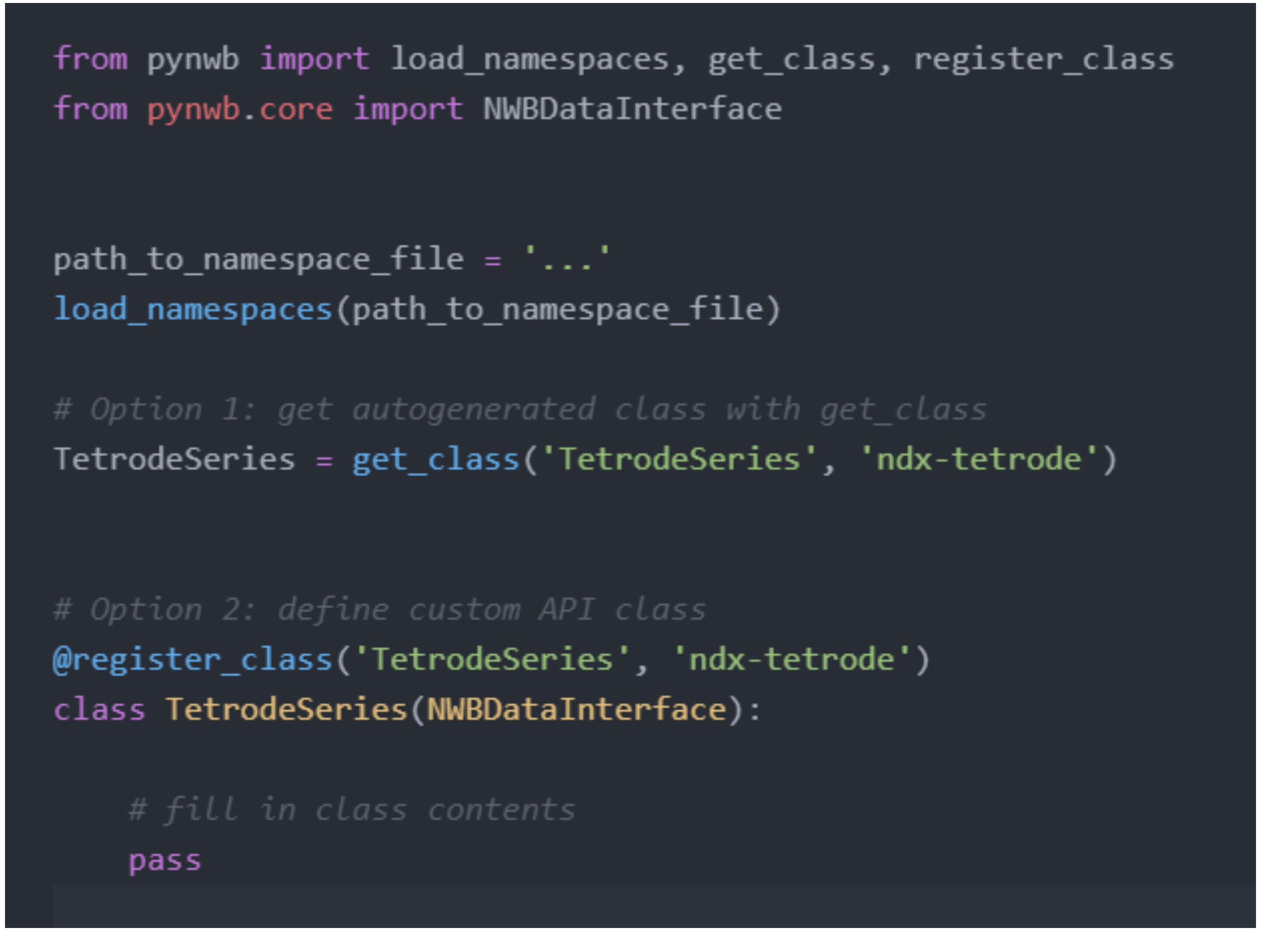
```
tetrode_series = NWBGroupSpec(
    neurodata_type_def='TetrodeSeries',
    neurodata_type_inc='ElectricalSeries',
    doc='A data type to store recordings from a tetrode.',
    attributes=[
        NWBAttributeSpec(
            name='tetrode_id',
            doc='The tetrode ID.',
            dtype='int'
        )
    ]
)

# export the spec to yaml files in the spec folder
output_dir = os.path.abspath(os.path.join(os.path.dirname(__file__), '..', '..', 'spec'))
export_spec(ns_builder, [tetrode_series], output_dir)
```

Use PyNWB API to generate extension YAML files



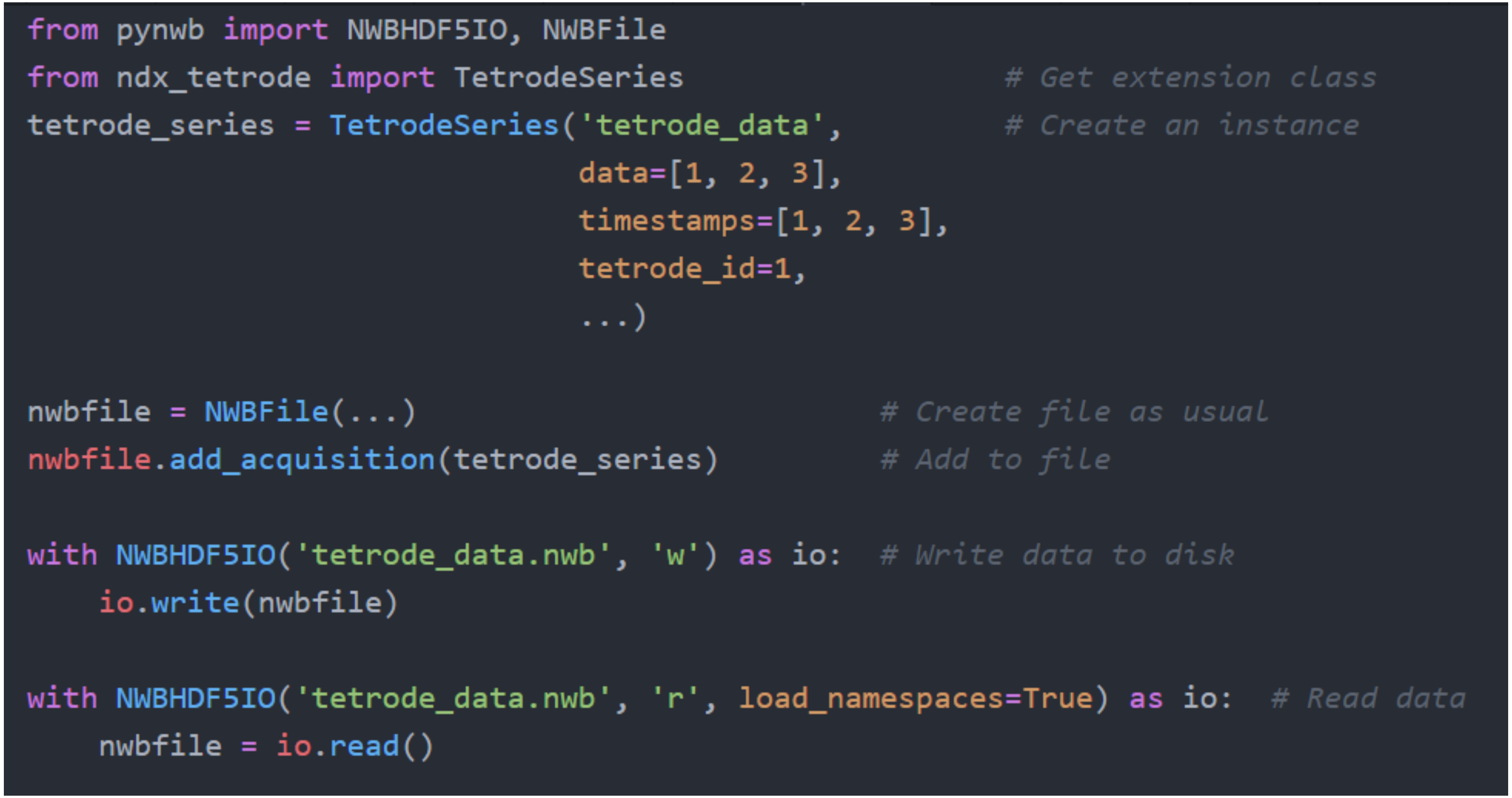
```
tetrode_series = NWBGroupSpec(
    neurodata_type_def='TetrodeSeries',
    neurodata_type_inc='ElectricalSeries',
    doc='A data type to store recordings from a tetrode.',
    attributes=[
        NWBAttributeSpec(
            name='tetrode_id',
            doc='The tetrode ID.',
            dtype='int'
        )
    ]
)

# export the spec to yaml files in the spec folder
output_dir = os.path.abspath(os.path.join(os.path.dirname(__file__), '..', '..', 'spec'))
export_spec(ns_builder, [tetrode_series], output_dir)
```

Define a Python class for an extension data type



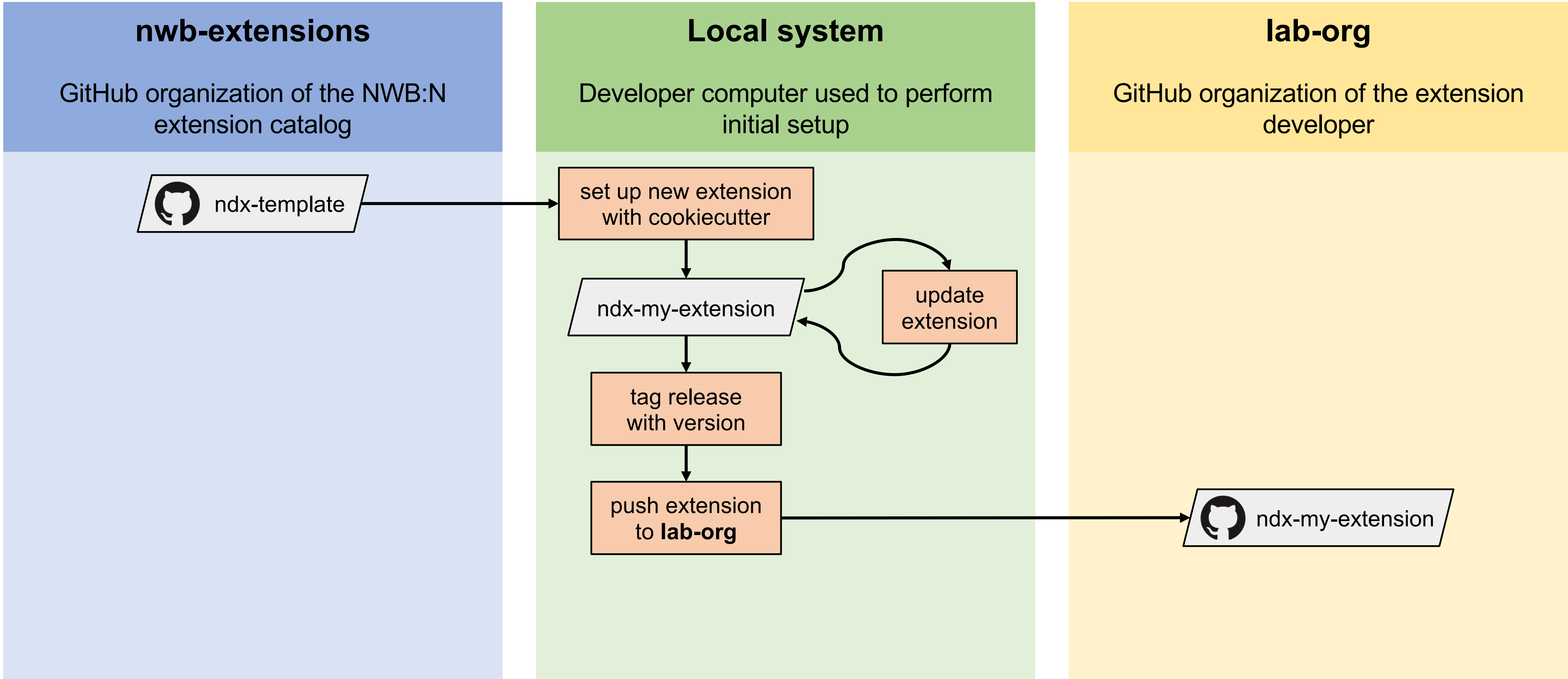
```python
from pynwb import load_namespaces, get_class, register_class
from pynwb.core import NWBDataInterface


path_to_namespace_file = '...'
load_namespaces(path_to_namespace_file)

# Option 1: get autogenerated class with get_class
TetrodeSeries = get_class('TetrodeSeries', 'ndx-tetrode')


# Option 2: define custom API class
@register_class('TetrodeSeries', 'ndx-tetrode')
class TetrodeSeries(NWBDataInterface):

    # fill in class contents
    pass
```

Using NWB extensions



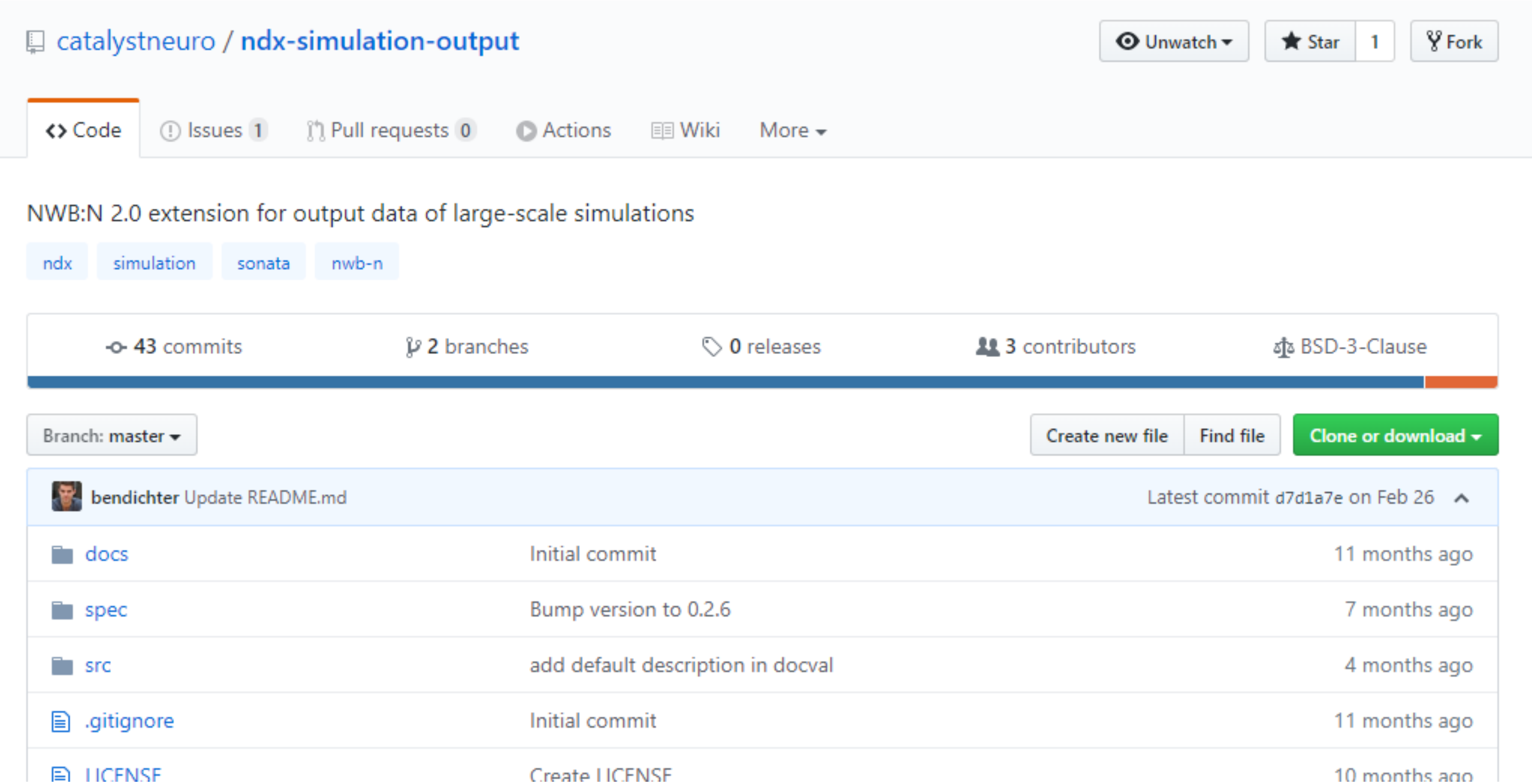
```python
from pynwb import NWBHDF5IO, NWBFile
from ndx_tetrode import TetrodeSeries                  # Get extension class
tetrode_series = TetrodeSeries('tetrode_data',         # Create an instance
                               data=[1, 2, 3],
                               timestamps=[1, 2, 3],
                               tetrode_id=1,
                               ...)

nwbfile = NWBFile(...)                          # Create file as usual
nwbfile.add_acquisition(tetrode_series)         # Add to file

with NWBHDF5IO('tetrode_data.nwb', 'w') as io:  # Write data to disk
    io.write(nwbfile)

with NWBHDF5IO('tetrode_data.nwb', 'r', load_namespaces=True) as io:  # Read data
    nwbfile = io.read()
```

Creating a Neurodata Extension (NDX)



nwb-extensions
GitHub organization of the NWB:N extension catalog
Local system
Developer computer used to perform initial setup
lab-org
GitHub organization of the extension developer
ndx-template
set up new extension with cookiecutter
ndx-my-extension
update extension
tag release with version
push extension to **lab-org**
ndx-my-extension

Extension is now on GitHub! But not published…

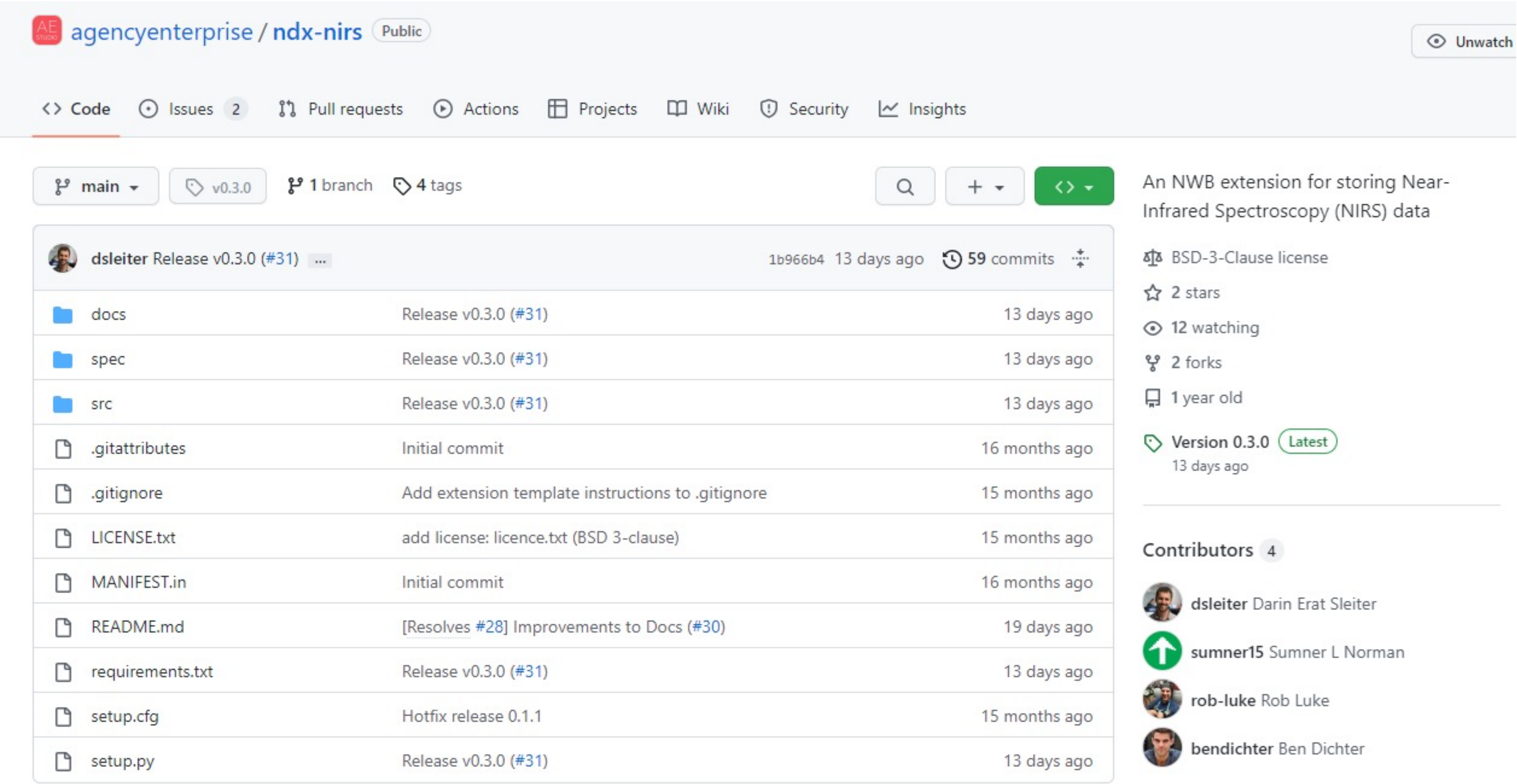


NWB:N 2.0 extension for output data of large-scale simulations

bendichter Update README.md Latest commit d7d1a7e on Feb 26

docs	Initial commit	11 months ago
spec	Bump version to 0.2.6	7 months ago
src	add default description in docval	4 months ago
.gitignore	Initial commit	11 months ago
LICENSE	Create LICENSE	10 months ago

A closer look: ndx-nirs



dsleiter Release v0.3.0 (#31) …	1b966b4 13 days ago	59 commits
docs	Release v0.3.0 (#31)	13 days ago
spec	Release v0.3.0 (#31)	13 days ago
src	Release v0.3.0 (#31)	13 days ago
.gitattributes	Initial commit	16 months ago
.gitignore	Add extension template instructions to .gitignore	15 months ago
LICENSE.txt	add license: licence.txt (BSD 3-clause)	15 months ago
MANIFEST.in	Initial commit	16 months ago
README.md	[Resolves #28] Improvements to Docs (#30)	19 days ago
requirements.txt	Release v0.3.0 (#31)	13 days ago
setup.cfg	Hotfix release 0.1.1	15 months ago
setup.py	Release v0.3.0 (#31)	13 days ago

An NWB extension for storing Near-Infrared Spectroscopy (NIRS) data

BSD-3-Clause license

2 stars

12 watching

2 forks

1 year old

Version 0.3.0 Latest
13 days ago

Contributors 4

dsleiter Darin Erat Sleiter

sumner15 Sumner L Norman

rob-luke Rob Luke

bendichter Ben Dichter

https://github.com/agencyenterprise/ndx-nirs

Why register extensions?

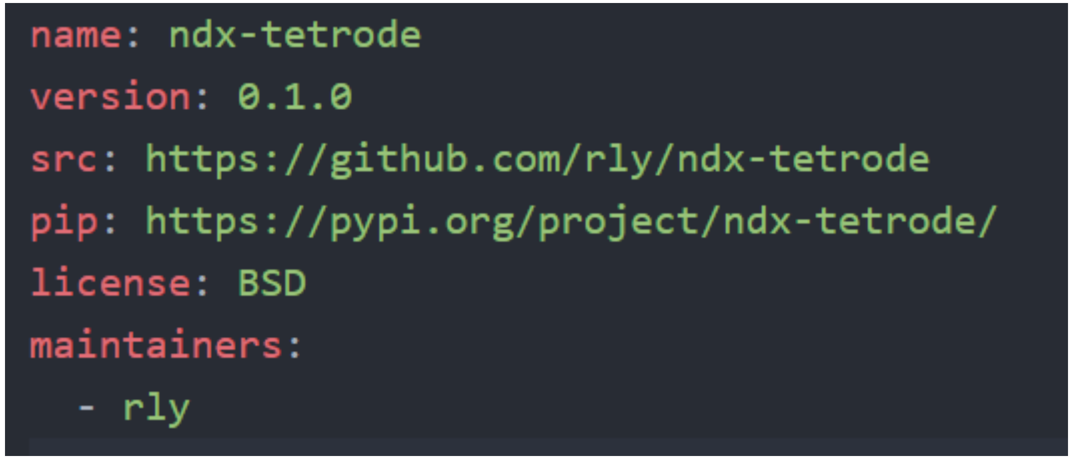
- Make the extension easy to find
- Support re-use of common data types across the community
- Show that the extension has been tested by NWB team and meets criteria for acceptance into the NDX catalog
- Simplify access for scientists and tool makers by encouraging a single extension for a particular use case rather than ten different extensions
- Simplify collaboration and review of extensions for shared use cases

The NDX Catalog

- NDX Catalog consists of GitHub repos with metadata about each extension – what it does, how to install it, where is its source code, and who maintains it

https://commons.wikimedia.org/wiki/File:LA2-katalogkort.jpg

The ndx-meta.yaml record metadata



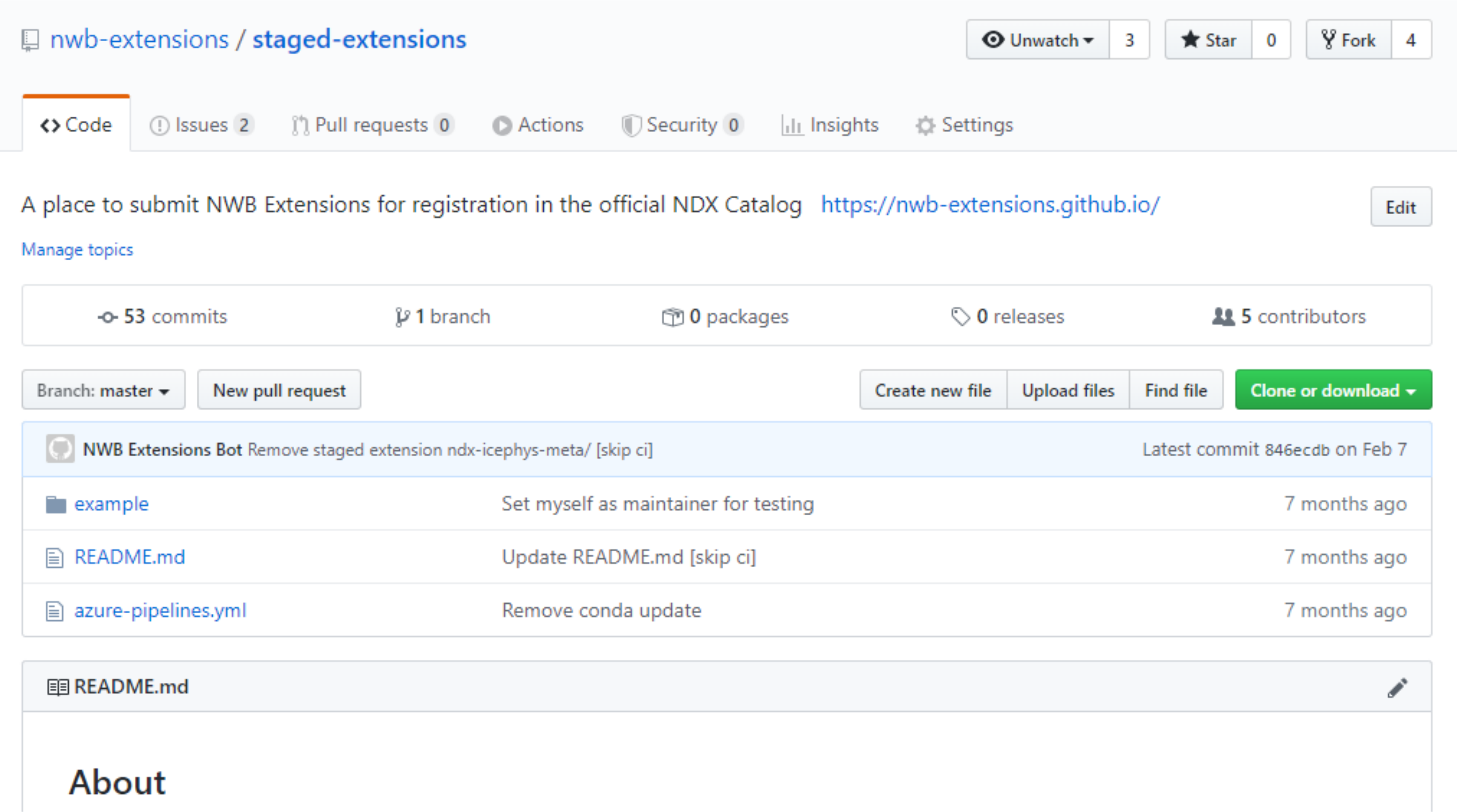
```yaml
name: ndx-tetrode
version: 0.1.0
src: https://github.com/rly/ndx-tetrode
pip: https://pypi.org/project/ndx-tetrode/
license: BSD
maintainers:
  - rly
```

Registering the Neurodata Extension (NDX)

- Extension must be published on the Python Package Index (PyPI) (supports both Python and MATLAB)
- Source code must be publicly available (e.g. on GitHub)
- Developers maintain ownership of extensions, i.e., source code stays in lab/developer's space

- Process adapted from conda-forge process of publishing packages

Fork and edit the staged-extensions repo



nwb-extensions / **staged-extensions**

A place to submit NWB Extensions for registration in the official NDX Catalog https://nwb-extensions.github.io/

53 commits · **1** branch · **0** packages · **0** releases · **5** contributors

NWB Extensions Bot Remove staged extension ndx-icephys-meta/ [skip ci] Latest commit 846ecdb on Feb 7

example	Set myself as maintainer for testing	7 months ago
README.md	Update README.md [skip ci]	7 months ago
azure-pipelines.yml	Remove conda update	7 months ago

README.md

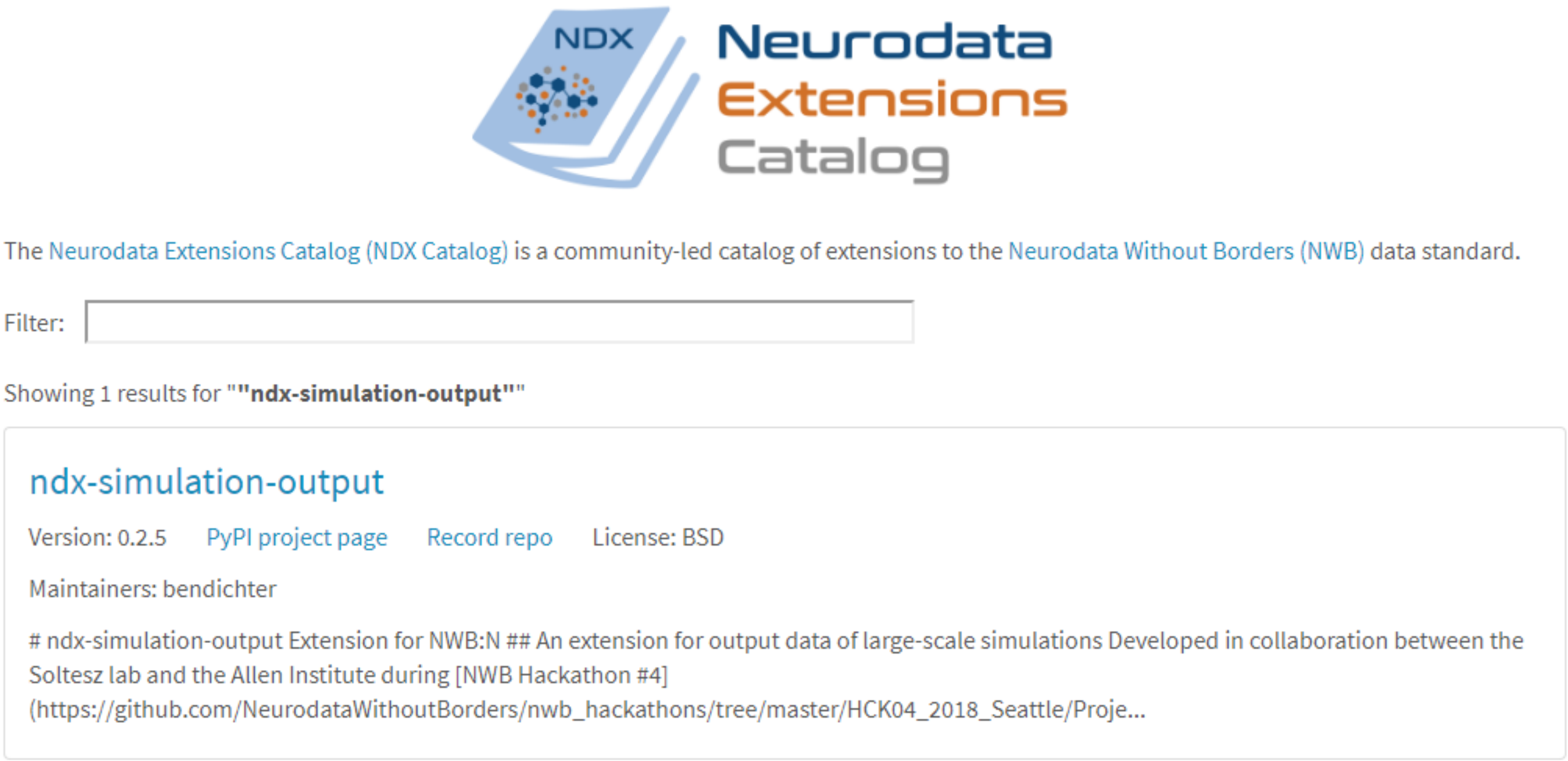
About

https://github.com/nwb-extensions/staged-extensions

The NDX Catalog registration review process

- NDX Catalog admins will review the extension and extension metadata for:
    - Using the spec language correctly
    - Having useful and readable documentation
    - Not duplicating an existing extension
- On approval, a new record repo is automatically generated in the “nwb-extensions” GitHub organization and accessible from nwb-extensions.github.io
- Extension creators are added as admins of the record repo – retain ownership and control

The Neurodata Extension is now registered!



The Neurodata Extensions Catalog (NDX Catalog) is a community-led catalog of extensions to the Neurodata Without Borders (NWB) data standard.

Filter:

Showing 1 results for "**"ndx-simulation-output"**"

ndx-simulation-output

Version: 0.2.5 PyPI project page Record repo License: BSD

Maintainers: bendichter

ndx-simulation-output Extension for NWB:N ## An extension for output data of large-scale simulations Developed in collaboration between the Soltesz lab and the Allen Institute during [NWB Hackathon #4] (https://github.com/NeurodataWithoutBorders/nwb_hackathons/tree/master/HCK04_2018_Seattle/Proje...

Updating the Neurodata Extension

- Follows similar process as registering the extension but now on the existing record repo within the catalog instead of the staged-extensions repo
- GitHub users specified in extension metadata manage the record repo

Refining the core NWB standard

- Data needs are constantly evolving and will require refinement of the core standard
- Extensions serve as perfect test bed for changes to the core standard
- Proposed changes to the core standard undergo stricter review process involving two stages
  - Community review by a working group of users, developers, and domain experts
  - Core proposal review by core developers and executive board to assess technical merits and compliance with NWB proposal standards

Summary

- Extensions can be used to add support for user-defined data types in NWB
- Use the ndx-template tool to help create extensions
  - See https://github.com/nwb-extensions/ndx-template
  - See https://pynwb.readthedocs.io/en/stable/extensions.html for API documentation
- Register extensions in the NDX Catalog to make them findable and accessible to the community: https://nwb-extensions.github.io/
- Collaborate on existing extensions to reduce duplicated effort and converge on a standardized schema
- Use extensions and NDX Catalog to refine the core NWB standard