XStream Documentation

Release 0.1.0

Argonne National Laboratory

March 25, 2016

Contents

Streaming data algorithms for x-ray tomography.

CHAPTER 1

Overview

1.1 Motivations

Imagine the challenge of driving your car from home to work with your eyes closed. This is similar to how scientists feel today in conducting imaging experiments at the Advanced Photon Source (APS). They must set up their experiments by intuition, hoping to collect the maximum information about the material under study; however, they do not see the whole picture of what happened until sometime after the experiment is done. As a result, they do not know if they have modified the specimen by beam damage, or if they spent considerable time scanning uninteresting areas while collecting insufficient detail from the crucial features or a critical time point in a dynamic specimen.

This package will let the driver see *while driving*, by algorithms to obtain preliminary images from selectively sampled data and using this for intelligent experimental control to collect the right amount of data from the right regions. A typical experiment in this new scheme will contain many loops; that is, the experiment will continuously inform the computation, and the computation will concurrently design the experiment.

1.2 Beneficeries

Tomography is the premier technique at synchrotrons for studying thick samples, and is used with a large variety of contrast modes such as phase contrast, fluorescence and diffraction contrast. It has a long history of applications from materials research to biological, environmental and life sciences, and is one of the most utilized techniques at APS and at other synchrotrons worldwide.

1.3 Features

- Generic geometry definitions for scanning probes
- Reconstruction algorithms for streaming data
- Visualization of geometries and reconstructions

CHAPTER 2

Release Notes

2.1 0.1.0 (03-25-2016)

- Initial submit.

CHAPTER 3

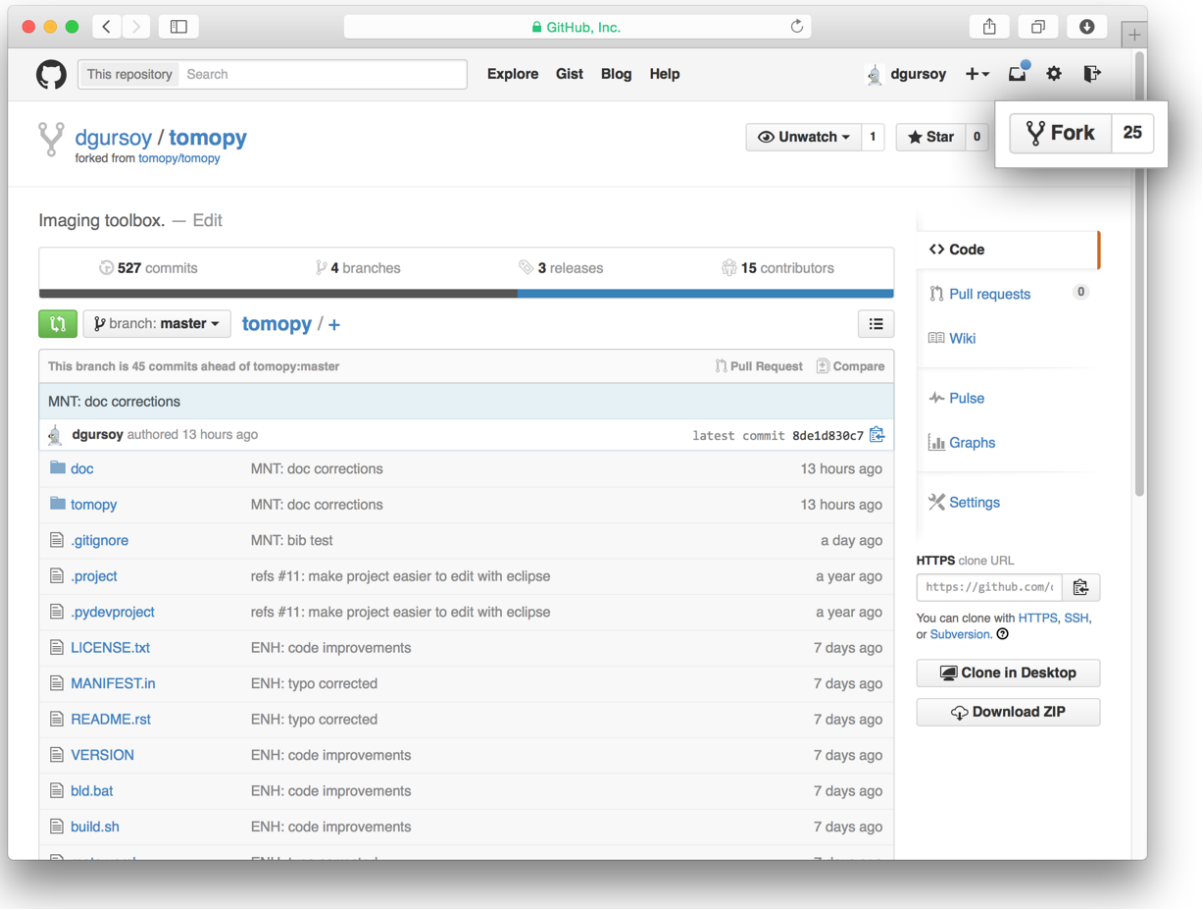
Development

This section explains the basics for developers who are willing to contribute to the XStream project.

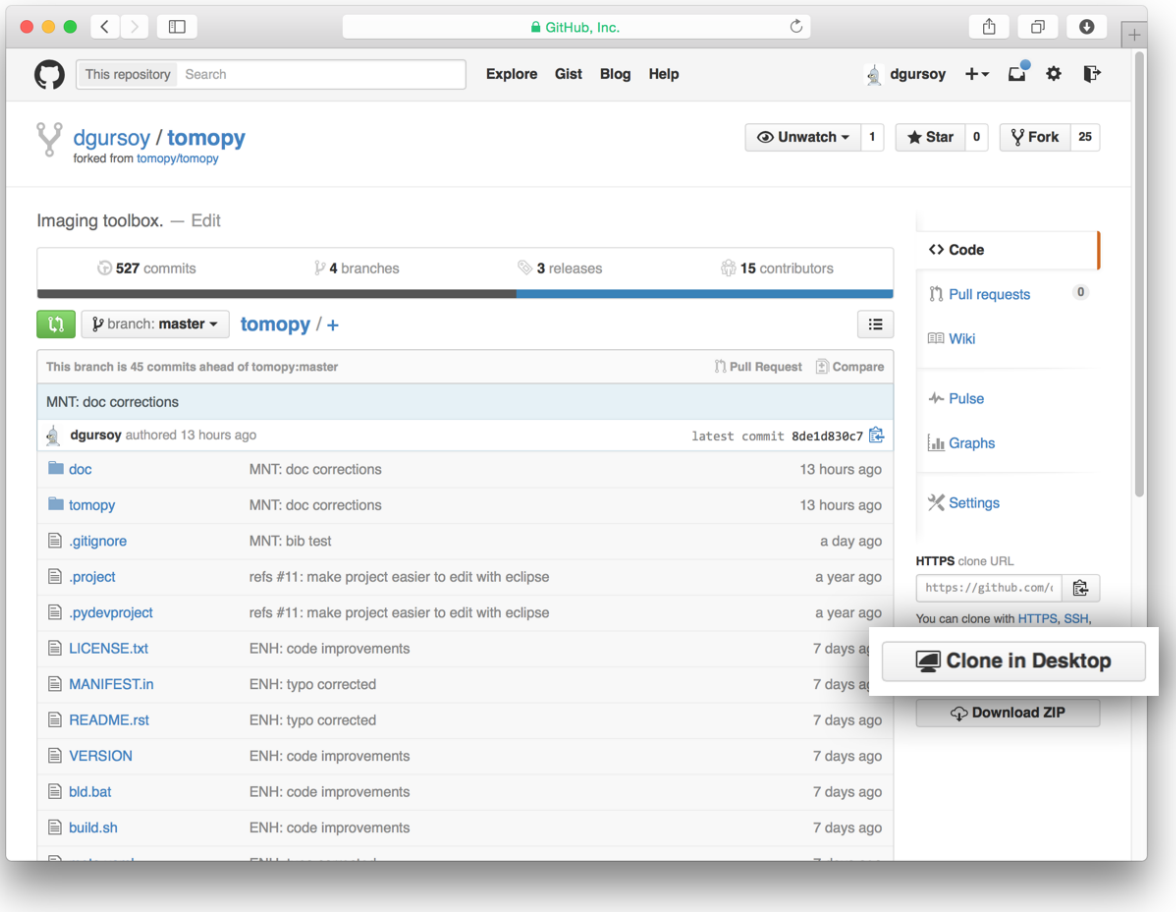
Contents:

3.1 Cloning the repository

The project is maintained on GitHub, which is a version control and a collaboration platform for software developers. To start first register on GitHub and fork the XStream repository by clicking the **Fork** button in the header of the XStream repository:



This successfully creates a copy of the project in your personal GitHub space. The next thing you want to do is to clone it to your local machine. You can do this by clicking the **Clone in Desktop** button in the bottom of the right hand side bar:



This will launch the GitHub desktop application (available for both Mac and Win) and ask you where you want to save it. Select a location in your computer and feel comfortable with making modifications in the code.

3.2 Coding conventions

We try to keep a consistent and readable code. So, please keep in mind the following style and syntax guidance before you start coding.

First of all the code should be well documented, easy to understand, and integrate well into the rest of the project. For example, when you are writing a new function always describe the purpose and the parameters:

```
def my_awesome_func(a, b):
    """
    Adds two numbers.

    Parameters
    ----------
    a : scalar (float)
        First number to add

    b : scalar (float)
        Second number to add

    Returns
```

```
    -------
    output : scalar (float)
        Added value
    """
    return a+b
```

3.3 Package versioning

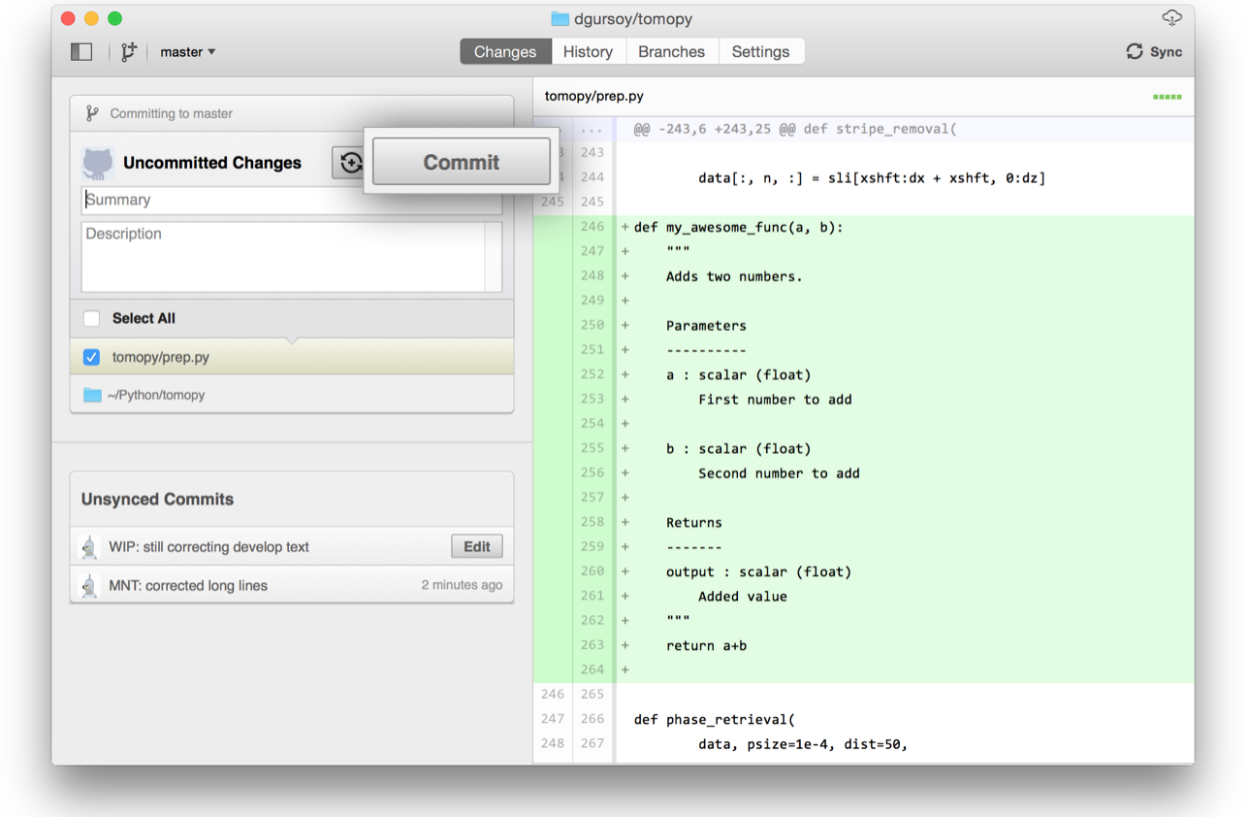
We follow the X.Y.Z (Major.Minor.Patch) semantic for package versioning. The version should be updated before each pull request accordingly. The patch number is incremented for minor changes and bug fixes which do not change the software's API. The minor version is incremented for releases which add new, but backward-compatible, API features, and the major version is incremented for API changes which are not backward-compatible. For example, software which relies on version 2.1.5 of an API is compatible with version 2.2.3, but not necessarily with 3.2.4.

3.4 Package distribution

We use Conda as a package and environment manager to manage the project dependencies and to share environments. Conda works with all platforms, and is language agnostic, which allows us to use it multi-language projects like this one.

3.5 Commiting changes

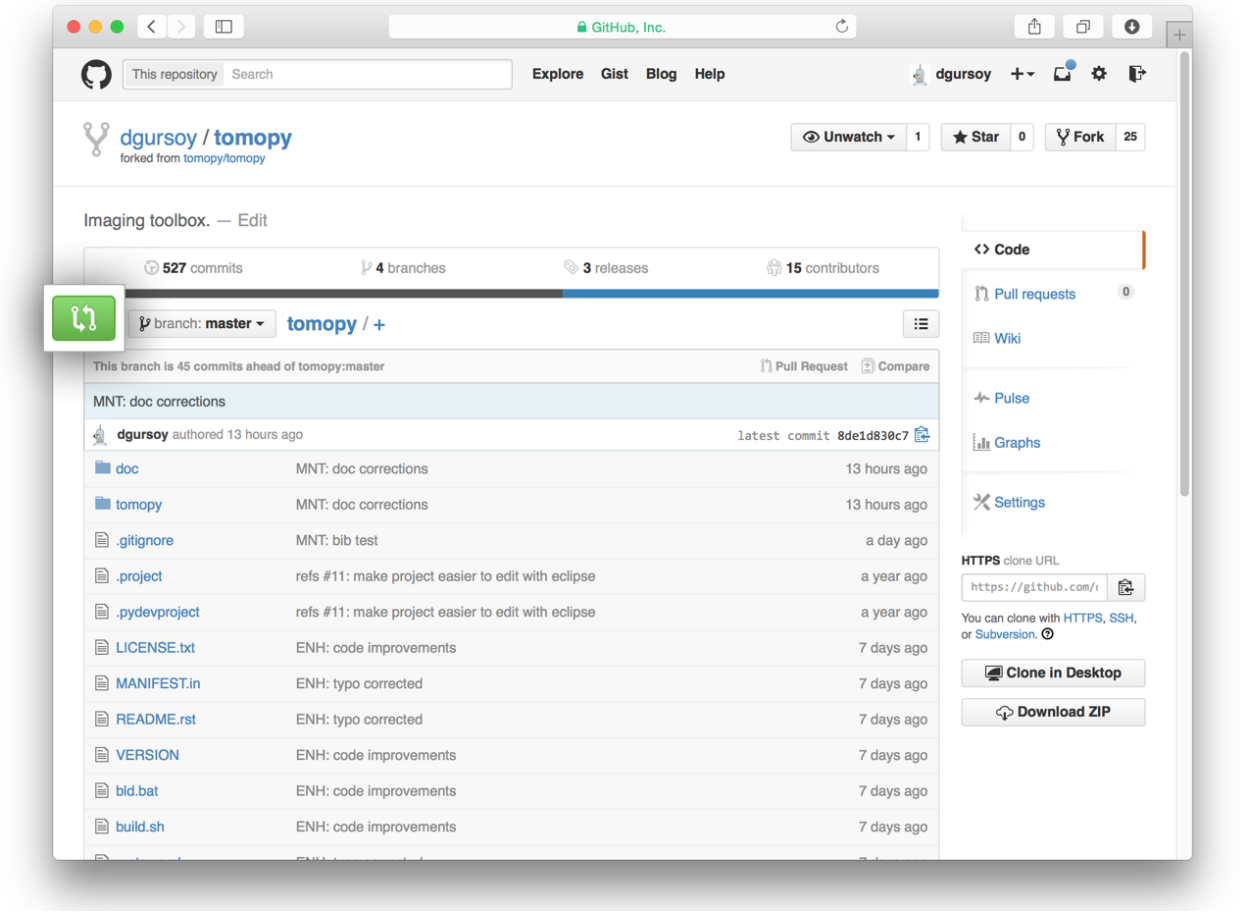
After making some changes in the code, you may want to take a *snapshot* of the edits you made. That's when you make a *commit*. To do this, launch the GitHub desktop application and it should provide you all the changes in your code since your last commit. Write a brief *Summary* and *Description* about the changes you made and click the **Commit** button:



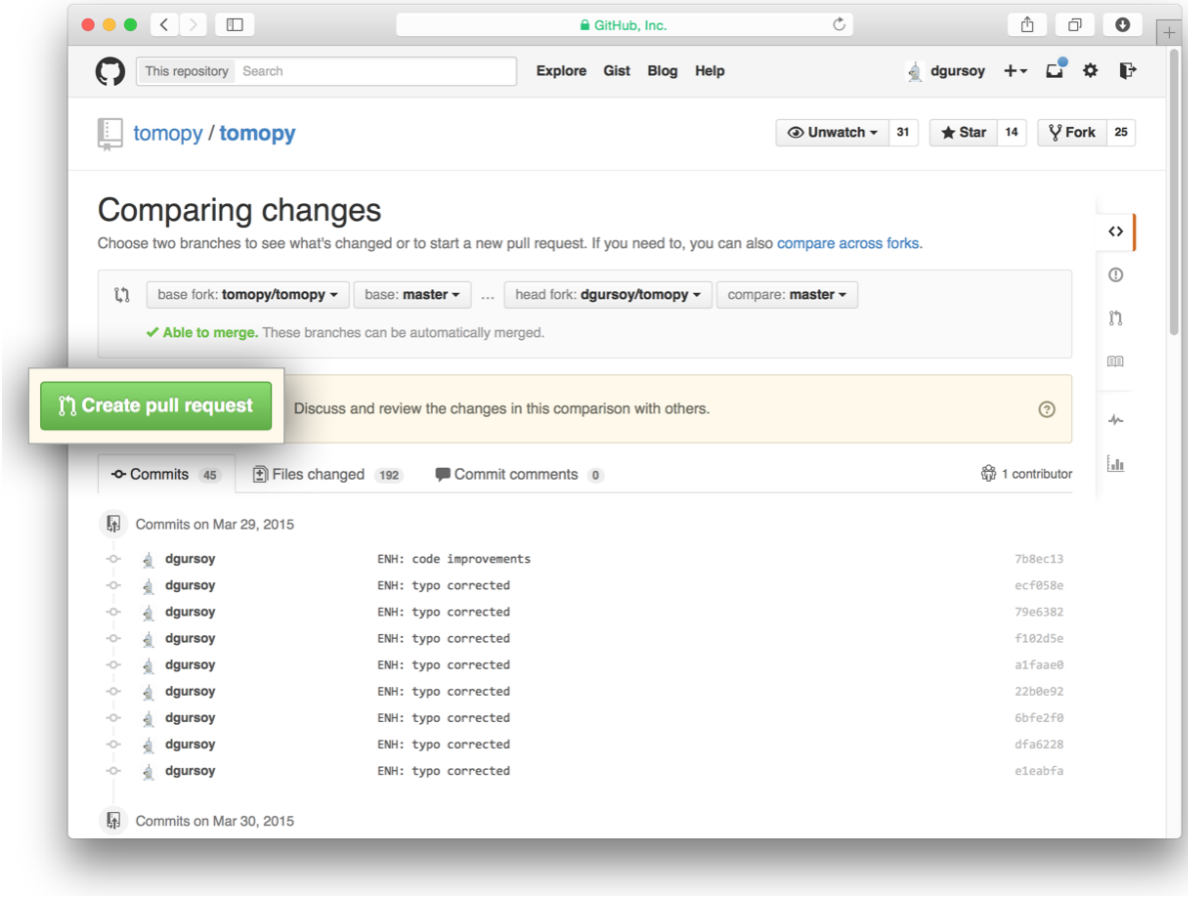
You can continue to make changes, add modules, write your own functions, and take more *Commit snapshots* of your code writing process.

3.6 Contributing back

Once you feel that the functionality you added would benefit the community, then you should consider contributing back to the XStream project. For this, go to your online GitHub repository of the project and click on the *green* button to compare, review and create a pull request.



After clicking on this button, you are presented with a review page where you can get a high-level overview of what exactly has changed between your forked branch and the original XStream repository. When you're ready to submit your pull request, click **Create pull request**:



Clicking on **Create pull request** sends you to a discussion page, where you can enter a title and optional description. It's important to provide as much useful information and a rationale for why you're making this Pull Request in the first place.

When you're ready typing out your heartfelt argument, click on **Send pull request**. You're done!

CHAPTER 4

Roadmap

Progress Notes.

CHAPTER 5

Credits

5.1 Lead Developers

- Doga Gursoy (@dgursoy)

5.2 Sponsors

- U.S. Department of Energy

CHAPTER 6

References

CHAPTER 7

Indices and tables

- genindex
- modindex
- search

[A1] Gürsoy D, De Carlo F, Xiao X, and Jacobsen C. Tomopy: a framework for the analysis of synchrotron tomographic data. *Journal of Synchrotron Radiation*, 21(5):1188–1193, 2014.

[A2] De Carlo F, Gursoy D, Marone F, Rivers M, Parkinson YD, Khan F, Schwarz N, Vine DJ, Vogt S, Gleber SC, Narayanan S, Newville M, Lanzirotti T, Sun Y, Hong YP, and Jacobsen C. Scientific data exchange: a schema for hdf5-based storage of raw and analyzed data. *Journal of Synchrotron Radiation*, 21(6):1224–1230, 2014.