

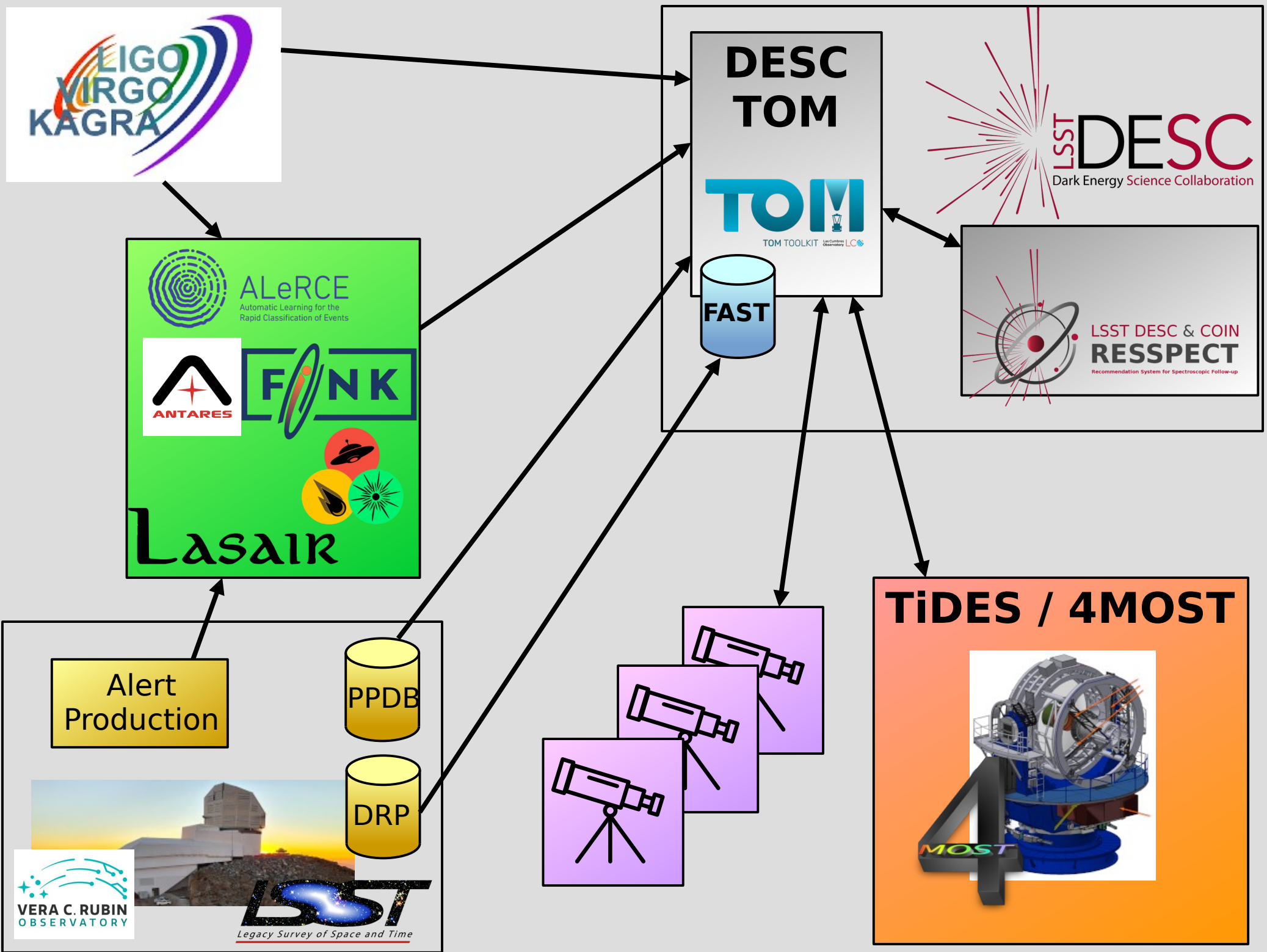
ELAsTiCC Tutorial

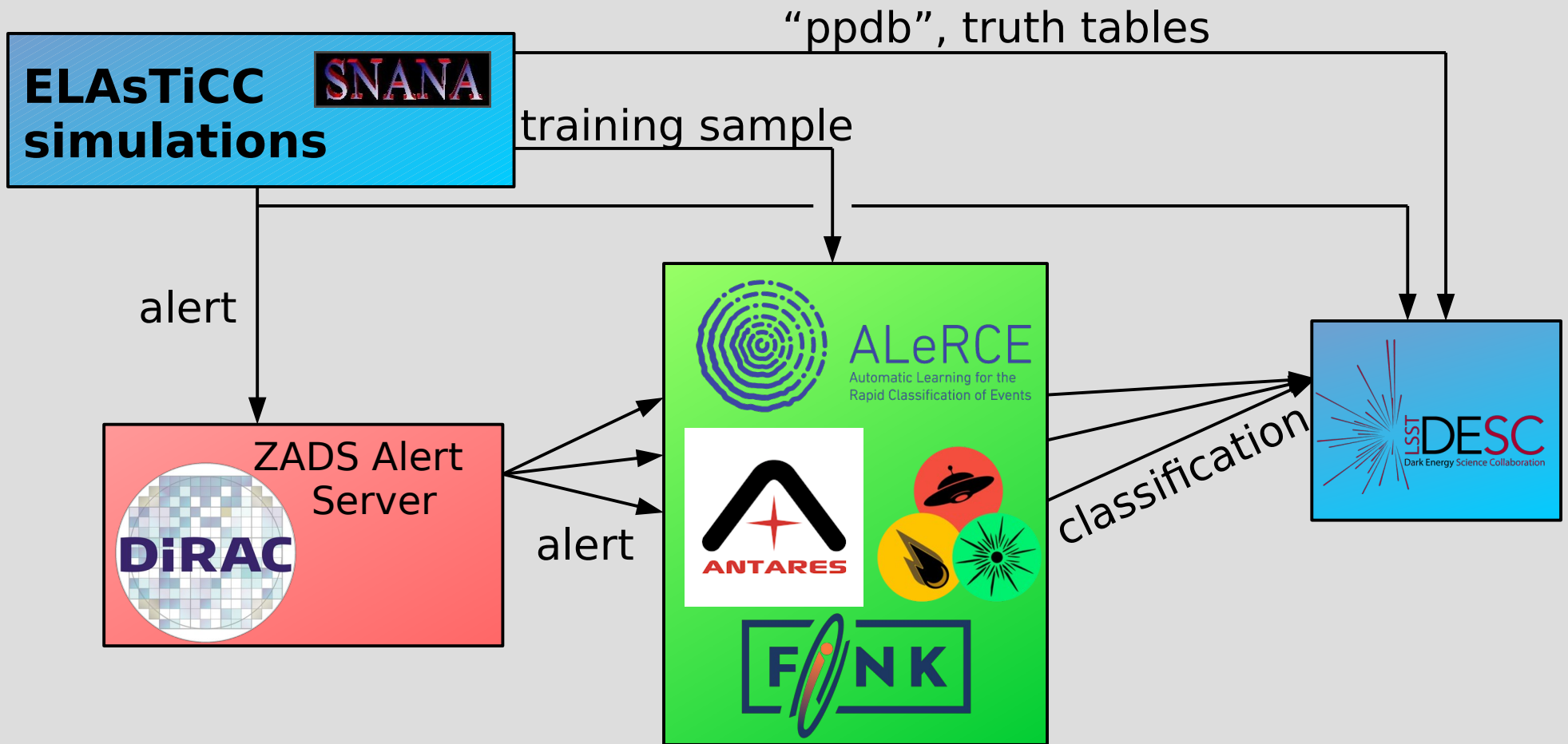


Rob Knop
DESC Sprint Week
2024-10-28

What is ELAsTiCC?

- Catalog-level simulation of 4 million transients and variables
 - SNANA simulations (R. Kessler)
 - 37 SNANA models*
 - 62 million “sources” (photometry points), 43 million with $S/N \geq 5$
 - 990 million “forced sources” (forced photometry points)
- Campaign to test LSST/DESC alert cycle (LSST Alerts → Brokers → DESC)
 - Trial run: ELAsTiCC1, 2022 Sep–2023 Jan
 - ELAsTiCC2, 2023 Nov–Dec
 - Includes broker classification data in addition to SNANA simulations from 4–5 brokers.





Data Formats Available

- SNANA FITS files (+CSV for object truth)
- Parquet files
- Tarballs of AVRO alerts
- Web API / PostgreSQL Database on the DESC TOM

SNANA FITS Files

- Use if you already know or want to learn the SNANA format.
- Use with the `lib_elasticc2` python library.

On NERSC: `/global/cfs/cdirs/desc-td/ELASTICC2`

- One subdirectory for each model
- 40 `_HEAD.FITS` and `_PHOT.FITS` (gzipped) files in each subdirectory
- CSV file in subdirectory with object truth: `ELASTICC2_FINAL_{model}.DUMP`

Parquet Files

On NERSC:

`/global/cfs/cdirs/desc-td/ELASTICC2_parquet`

Parquet schema includes many (but not all) of the fields from the HEAD file and object truth file as scalars, and several fields from the PHOT files as lists. (See demo.)

Files are divided by SNANA model. SNID is the “key” field that identifies a given object.

DESC Tom Web API / SQL Database

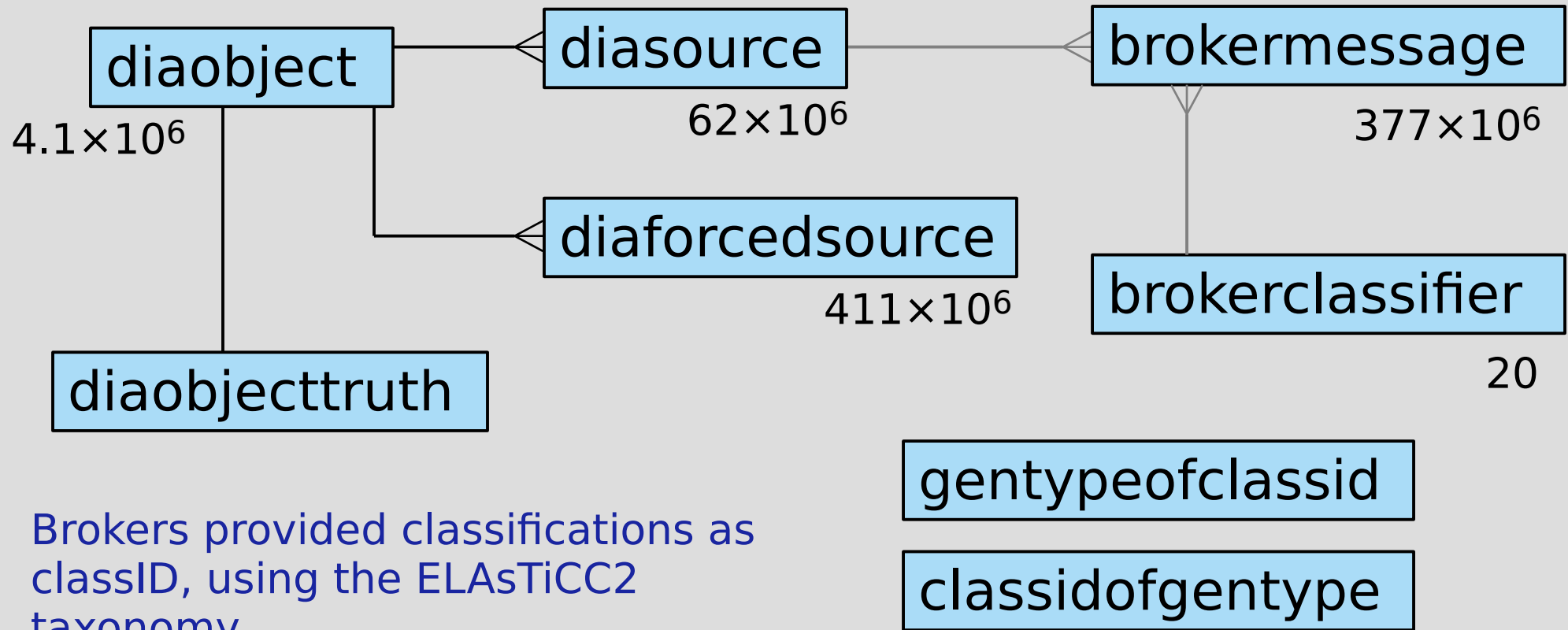
<https://desc-tom.lbl.gov>

- You need a username/password on the TOM. (Ask Rob; give him the username you want and the email address associated with the TOM account.)
- There are some web API calls to get information (demo shortly). If there are others that you think would be useful, talk to Rob.
- There is an interface that lets you query the PostgreSQL database directly:

https://github.com/LSSTDESC/tom_desc/blob/main/sql_query_tom_db.ipynb

Structure of ELAsTiCC2 database

Note: all tables have elasticc2_ prepended



- Brokers provided classifications as classID, using the ELAsTiCC2 taxonomy.
- Truth tables have SNANA gentype.
- The **of** tables provide the mapping.

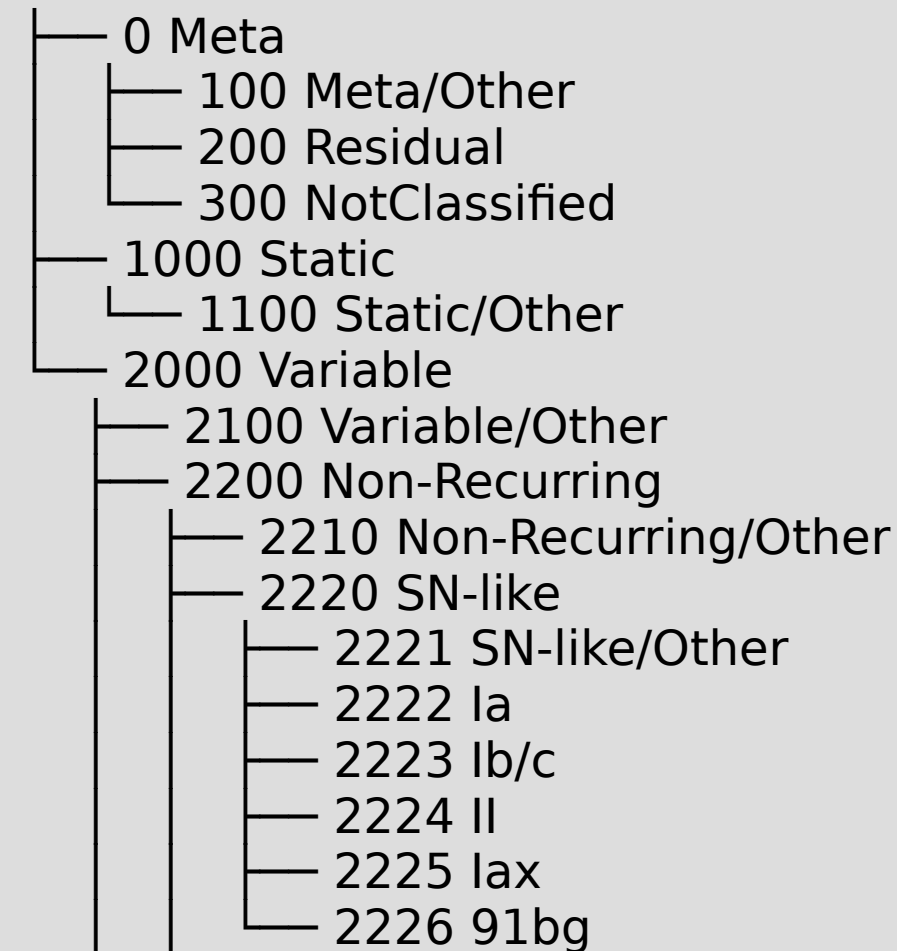
Table definitions:

https://github.com/LSSTDESC/elasticc/blob/lib_elasticc2/jupyter/sprint_week_2024oct/elasticc2_schema.txt

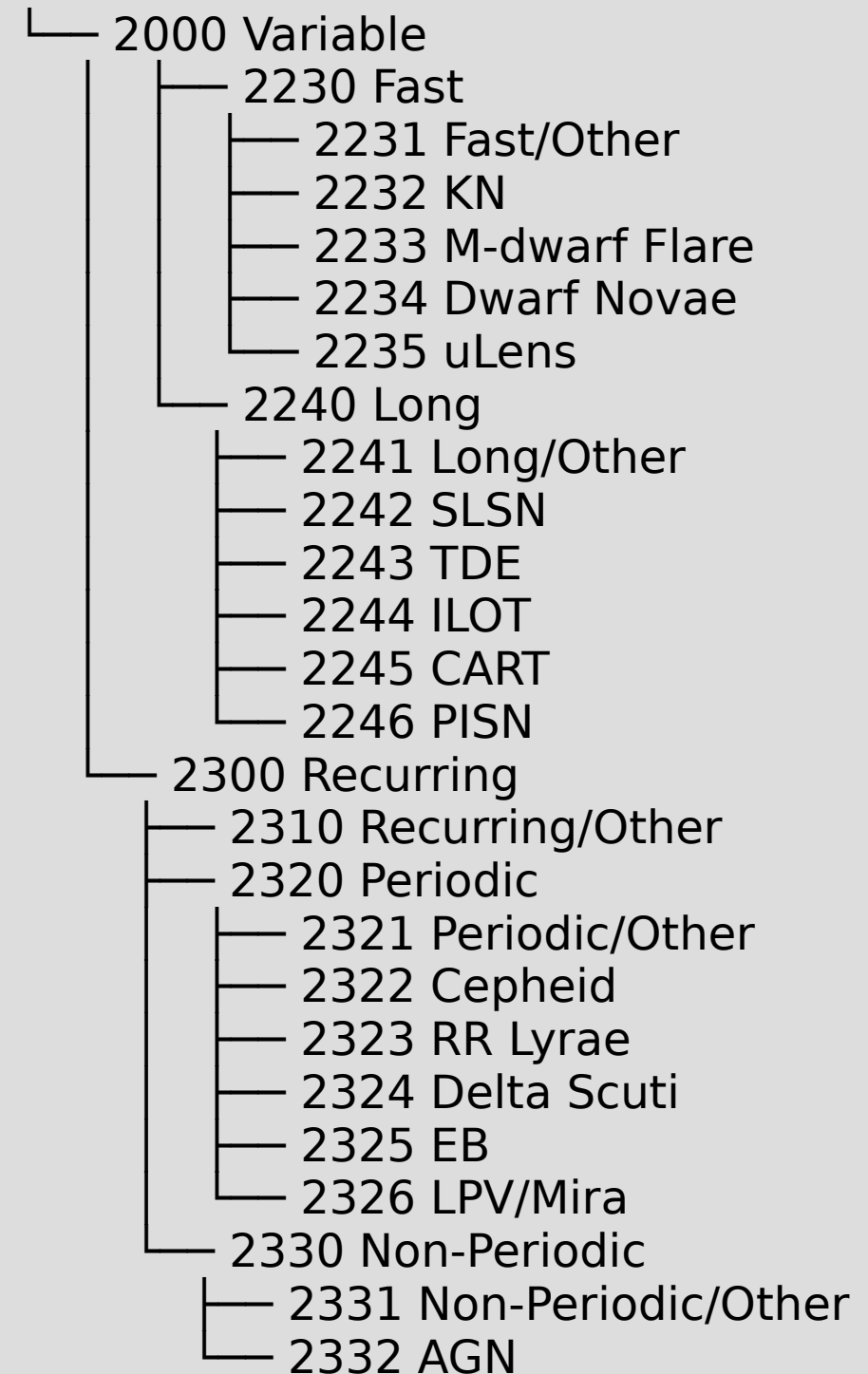
ELAsTiCC2 Taxonomy

[https://github.com/LSSTDESC/elasticc/
blob/main/taxonomy/taxonomy.ipynb](https://github.com/LSSTDESC/elasticc/blob/main/taxonomy/taxonomy.ipynb)

Alert



Alert



Tarballs of AVRO Alerts

- This is a terrible format for reading the data: redundant, inefficient.
- Only use these if you are **specifically** dealing with realtime processing of actual alerts.
- **No, really, don't use this format.**

Alerts tarred up by MJD (all object types mixed):
`/global/cfs/cdirs/desc-td/ELASTICC2_ALERTS`

You will need the ELAsTiCC2 alert schema:

https://github.com/LSSTDESC/elasticc/tree/elasticc2/alert_schema

`elasticc.v0_9_1.alert.avsc` is the schema of an alert; it refers to other files in that directory.

Getting set up for the tutorials at NERSC

Setting yourself up to run the DESC Jupyter kernels:

```
source /global/common/software/lsst/common/miniconda/kernels/setup.sh
```

→ This will set up the standard DESC kernels for use in jupyter. You may want to rerun this every so often to get updates!

(It creates directories and files underneath `~/.local/share/jupyter/kernel`)

Verify that you have them with:

```
jupyter kernelspec list
```

→ You should see a number of **desc-*** kernels, including the most basic **desc-python** and **desc-td-env-dev**

Running the desc-td environment at NERSC

Jupyter is great for tutorials and demos and mucking about, but is not where you want to be doing serious development.

Get yourself into the desc-td environment on your shell at NERSC by running:

```
source /global/common/software/lsst/common/miniconda/setup_current_python.sh
```

or (for the Time Domain environment):

```
source /global/cfs/cdirs/lsst/groups/TD/setup_td.sh
```

or

```
source /global/cfs/cdirs/lsst/groups/TD/setup_td_dev.sh
```

Change your kernel to desc-td-env-dev

The screenshot shows a JupyterLab interface in a Mozilla Firefox browser. The address bar shows the URL: `https://jupyter.nersc.gov/user/raknop/perlmutter-login-node-base/lab/tree/global/homes/r/raknop/desc/elasticc/jupyter/sprint_we`. The interface includes a file browser on the left, a code editor in the center, and a terminal at the bottom.

File Browser:

- FAVORITES: \$HOME, \$CSCRATCH, \$PSCRATCH
- FILE BROWSER: Filter files by name
- Current directory: `/ jupyter / sprint_week_2024oct /`
- Files:
 - `elasticc2_demo1.ipynb` (an hour ago)
 - `elasticc2_demo2.ipynb` (an hour ago)
 - `elasticc2_demo3.ipynb` (an hour ago)

Code Editor:

The code editor shows a Python script with the following content:

```
[1]: import sys
import os
import io
import pathlib
import logging
import time

# Add to path the directory where tom_client.py exists
sys.path.insert( 0, str( pathlib.Path( os.getenv("HOME") ) / "research/desc/tom_desc" ) )
from tom_client import TomClient

# Get your DESC TOM username and password. tompasswdfile
# is a file with a single line containing your TOM password.
# This should be in a place that's not readable by anybody
# but you. (Use this rather than putting the actual
# password into something that might get committed to a
# git archive!)
tomuser = 'rkноп'
tompasswdfile = pathlib.Path( os.getenv("HOME") ) / "secrets/tom_rkноп_passwd"

# TomClient is a thin front-end to Python requests that handles
# authentication and some annoying headers that need to be set
# for connections to Django to work.
tomclient = TomClient( url="https://desc-tom.lbl.gov", username=tomuser, passwordfile=tompasswdfile )

# Make a logger so that we can print out timings and things like that
_logger = logging.getLogger("main")
if not _logger.hasHandlers():
    _logout = logging.StreamHandler( sys.stderr )
    _logger.addHandler( _logout )
    _logout.setFormatter( logging.Formatter( f'[%asctime)s - %(levelname)s] - %(message)s',
                                             datefmt='%Y-%m-%d %H:%M:%S' ) )
_logger.setLevel( logging.INFO )
_logger.info( "Testing" )

[2024-10-24 13:27:54 - INFO] - Testing

[2]: # Make ourselves a convenience function that does some return value checking,
# so we don't have to do that over and over again.

def query_tom( api_endpoint, data, verbose=False ):
    if verbose:
        _logger.info( f"Sending {api_endpoint} request to tom..." )
    t0 = time.perf_counter()
    res = tomclient.post( api_endpoint, json=data )
    dt = time.perf_counter() - t0
    if verbose:
        _logger.info( f"...done after {dt:.1f} seconds" )

    if res.status_code != 200:
```

The kernel is set to **NERSC Python**, which is circled in red in the image.

Terminal:

Mode: Command Ln 1, Col 1 elasticc2_demo3.ipynb 0

Finding information about ELAsTiCC:

- ELAsTiCC public web page, ideally with full documentation:
https://portal.nersc.gov/cfs/lsst/DESC_TD_PUBLIC/ELASTICC/
- Slack: #elasticc-comms (ping me, Rick, Alex M., etc.)
- The DESC TOM:
 - <https://desc-tom.lbl.gov> (ping me for an account)
 - https://github.com/LSSTDESC/tom_desc
 - ELAsTiCC Metrics:
 - <https://desc-tom.lbl.gov/elasticc2/>
 - API:
https://github.com/LSSTDESC/elasticc_metrics/blob/main/elasticc2_rest_metric_demo.ipynb
 - SQL:
https://github.com/LSSTDESC/tom_desc?tab=readme-ov-file#accessing-the-tom
https://github.com/LSSTDESC/tom_desc/blob/main/sql_query_tom_db.ipynb