CI/CS WORKSHOP
THE COMMUNITY TOGETHER

ResearchSOC  CI CoE Pilot
Steve Jacobs

NEON Systems Architect and Manager of Cyber Infrastructure for Battelle Ecology on the National Ecological Observatory Network (NEON) program

https://www.neonscience.org/
This material is based upon work supported by NSF’s National Ecological Observatory Network which is a major facility fully funded by the National Science Foundation.

Steve Jacobs, Cove Sturtevant, Dawn Lenz, Rob Markel, Hale Brownlee, Ross Gaddie, Kaelin Cawley, Claire Lunch, Greg Holling, Steve Stone

National Ecological Observatory Network
Battelle

NEON/Pachyderm
National Ecological Observatory Network

81 FIELD SITES
• 47 terrestrial
• 34 aquatic
National Ecological Observatory Network

- A flux tower collects atmospheric data at terrestrial sites
- Sensors in the soil collect belowground data at terrestrial sites
- Field scientists collect organismal data (from select plants, animals, pathogens and microbes)
- Instruments collect stream, lake and groundwater data at aquatic sites
- A meteorological station collects atmospheric data at aquatic sites

Over 180 DATA PRODUCTS
Existing processing system

• Based on apache airflow [https://airflow.apache.org/](https://airflow.apache.org/)
  ▪ ETL Workflow tool
  ▪ Date based

• Good visibility and operational tooling
  ▪ Clear reporting
  ▪ Easily re-run data through pipeline
  ▪ Performance metrics
Existing processing system

- Existing execution system provides high level time-oriented overview
Existing processing system

• Robust Operation Tooling
Existing processing system

- System derived from legacy codebase (Java)
- Tooling allows re-processing of data, but not automated re-processing
- Highly dependent on internal data services
Motivations for automated, modular, provenance-focused data processing

• Dynamic data (a lot of it)
  ▪ Continuous receipt of new data
  ▪ Metadata and parameter adjustments

• Data provenance
  ▪ Traceability
  ▪ Reproducibility

• Code re-usability / Use outside of NEON

• Integrated Science-Cyberinfrastructure development
# System design

<table>
<thead>
<tr>
<th>NEED</th>
<th>SOLUTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automated response to data change (raw data, calibrations, location info, etc)</td>
<td>Pachyderm-based processing modules ‘listen’ for any data change</td>
</tr>
<tr>
<td>Traceability</td>
<td>Git-like version control for data and code</td>
</tr>
<tr>
<td>Reproducibility</td>
<td>Version-controlled Docker containers contain code and dependencies</td>
</tr>
<tr>
<td>Code re-usability</td>
<td>Highly modular processing design</td>
</tr>
<tr>
<td>Integrated Science-CI development</td>
<td>Docker-based, language-agnostic code packaging</td>
</tr>
</tbody>
</table>

1st prototype – Soil temperature
# System design

<table>
<thead>
<tr>
<th>NEED</th>
<th>SOLUTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automated response to data change</td>
<td>Pachyderm-based processing modules</td>
</tr>
<tr>
<td>(raw data, calibrations, location info, etc)</td>
<td>‘listen’ for any data change</td>
</tr>
<tr>
<td>Traceability</td>
<td>Git-like version control for data and code</td>
</tr>
<tr>
<td>Reproducibility</td>
<td>Version-controlled Docker containers contain code and dependencies</td>
</tr>
<tr>
<td>Code re-usability</td>
<td>Highly modular processing design</td>
</tr>
<tr>
<td>Integrated Science-CI development</td>
<td>Docker-based, language-agnostic code packaging</td>
</tr>
</tbody>
</table>

1\textsuperscript{st} prototype – Soil temperature
Connecting modules to data with a pipeline spec

```json
{  
  "pipeline": {  
    "name": "prt_calibration_filter"
    
  },  
  "transform": {  
    "image": "quay.io/battelleecology/neon-is-cal-filt-r:v0.0.21",
    "cmd": ["Rscript","/flow.cal.filt.R","DirIn=$DIR_IN","DirOut=/pfs/out","DirSubCopy=data"],
    "env": {  
      "LOG_LEVEL": "INFO"
    },
    "input": {  
      "pfs": {  
        "name": "DIR_IN",
        "repo": "prt_data_calibration_group",
        "glob": "/*//*/"
      }
    },
    "parallelism_spec": {  
      "constant": "2"
    },
    "resource_requests": {  
      "memory": "200M",
      "cpu": 0
    },
    "standby": true
  }
}
```
## System design

<table>
<thead>
<tr>
<th>NEED</th>
<th>SOLUTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automated response to data change</td>
<td>Pachyderm-based processing modules</td>
</tr>
<tr>
<td>(raw data, calibrations, location info, etc)</td>
<td>‘listen’ for any data change</td>
</tr>
<tr>
<td>Traceability</td>
<td>Git-like version control for data and code</td>
</tr>
<tr>
<td>Reproducibility</td>
<td>Version-controlled Docker containers</td>
</tr>
<tr>
<td>Code re-usability</td>
<td>contain code and dependencies</td>
</tr>
<tr>
<td>Integrated Science-CI development</td>
<td>Highly modular processing design</td>
</tr>
<tr>
<td></td>
<td>Docker-based, language-agnostic code packaging</td>
</tr>
</tbody>
</table>

1st prototype – Soil temperature
Tracing provenance through the commit chain

```
csturtevant@den-devissom:/ $ pachctl list commit tempSoil_levell_group
REPO     BRANCH   COMMIT   FINISHED SIZE     PROGRESS DESCRIPTION
---       ------   -------   -------- ----     ------- ------------
tempSoil_levell_group master f6e26c5893614e51aa3c7b93c67c89e4 20 hours ago 479.2KiB -
tempSoil_levell_group master d3fec2c105fd4912a81e786f03dcf66d 22 hours ago 479.2KiB -

csturtevant@den-devissom:/ $ pachctl inspect commit tempSoil_levell_group@d3fec2c105fd4912a81e786f03dcf66d
Commit: tempSoil_levell_group@d3fec2c105fd4912a81e786f03dcf66d
Original Branch: master
Parent: 80e8181b163d4f699014213992506403
Finished: 23 hours ago
Size: 479.2KiB

Provenance: avro_schemas@6890fd5f56924fe0a5c3344d01b2706b (master) location_assets@58d585cf96d4f7a91cde87ab412fbb8 (master) prt_soil_threshold_filter@8d3591fd2d7f41959dfacbddd0dc50f10 (master)
spec @7eb4c1dd4eb54298b24c2de4e97c27 (prt_calibrated_location_group) __spec_@f9cd41c6816624e3a352fb81633c1741 (tempSoil_padded_timeseries_analyzer) tempSoil_calibrated_data@f9c04cfc844dca9c360e3eb42932 (master) spec @a3ae53b61953483886c86cd00145b454 (prt calibration filter) spec @6902b30a10b0941a0

csturtevant@den-devissom:/ $ pachctl list file prt_soil_threshold_filter@8d3591fd2d7f41959dfacbddd0dc50f10
NAME       TYPE   SIZE
/thresholds.json file 259.9KiB
```
## System design

<table>
<thead>
<tr>
<th>NEED</th>
<th>SOLUTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automated response to data change (raw data, calibrations, location info, etc)</td>
<td>Pachyderm-based processing modules ‘listen’ for any data change</td>
</tr>
<tr>
<td>Traceability</td>
<td>Git-like version control for data and code</td>
</tr>
<tr>
<td>Reproducibility</td>
<td>Version-controlled Docker containers contain code and dependencies</td>
</tr>
<tr>
<td>Code re-usability</td>
<td>Highly modular processing design</td>
</tr>
<tr>
<td>Integrated Science-CI development</td>
<td>Docker-based, language-agnostic code packaging</td>
</tr>
</tbody>
</table>

1st prototype – Soil temperature
Open-Source Data Pipeline

modified from Metzger et al. 2017
Lessons learned

• Tradeoff between modularity vs. data storage and execution time
  ▪ Interim outputs require storage and compute power

• A for-loop can be your friend
  ▪ Time to spin up a container may be greater than code execution time
  ▪ Create flexibility to process multiple items at once
Things we need – Files!

- Pachyderm is file-based processing and provenance tracking
  - We don’t have files of sensor data!
  - For example, one of our simplest sensors, the Platinum Resistance Thermometer
  - Accessed via 4 different steam names in a single table (along with everything else)
    - NEON.D16.ABBY.DP0.00002.001.01325.000.020.000
    - NEON.D16.ABBY.DP0.00003.001.01325.000.050.103
    - NEON.D16.ABBY.DP0.00041.001.01728.002.507.000
    - NEON.D10.ARIK.DP0.20053.001.01325.101.100.000
  - Names are bound on receipt
  - Ingest format is no help for multi-stream sensors (XML per stream, tied to stream id numbers).
Things we need – Files!

• Schemas created for each sensor (51 sensors, 66 schemas)
• Using the avro schema format
  ▪ Record oriented ingest (instead of stream oriented)
  ▪ Easier to access data
• Ingest of avro records to parquet files on pachyderm
  ▪ Issues with using avro files on pachyderm
"type": "record",
"name": "prt",
"namespace": "org.neonscience.schema.device",
"doc": "100 Ohm Platinum Resistance Thermometer",
"fields": [
  {
    "name": "source_id",
    "type": "string",
    "doc": "Source serial number or MAC address"
  },
  {
    "name": "site_id",
    "type": "string",
    "doc": "NEON site identifier"
  },
  {
    "name": "readout_time",
    "type": { 
      "type": "long",
      "logicalType": "timestamp-millis"
    },
    "doc": "Timestamp of readout expressed in milliseconds since epoch",
    "__neon_units": "millisecond",
  },
  {
    "name": "resistance",
    "type": "float",
    "doc": "Measured resistance of the platinum resistance thermometer",
    "__neon_units": "ohm"
  }
]
Things we need – Files!

• Ingest of avro records to parquet files on pachyderm
  ▪ Avro record format is a standalone serialized record with no schema attached
    – Not self describing
  ▪ Avro files have an embedded schema and are self describing
    – They also contain a randomly generated 16-byte sync marker per file
**Things we need – Files!**

- Ingest of avro records to parquet files on pachyderm
  - Avro record format is a standalone serialized record with no schema attached
    - Not self describing
  - Avro files have an embedded schema and are self describing
    - They also contain a randomly generated 16-byte sync marker per file

**OOPS!**
Things we need – Files!

- Parquet is a better format for our use case
  - Consistent file output (no random markers)
  - Better library support thanks to Apache Arrow ([https://arrow.apache.org/](https://arrow.apache.org/))
    - Python (Pandas)
    - R
    - C/C++ / Rust / MATLAB more coming soon
  - Column based compression, native types, predicate pushdown support
Things we need – Files!

• Parquet is a better format for our use case
  ▪ Consistent file output (no random markers)
  ▪ Better library support thanks to Apache Arrow ([https://arrow.apache.org/](https://arrow.apache.org/))
    – Python (Pandas)
    – R
    – C/C++ / Rust / MATLAB more coming soon
  ▪ Column based compression, native types, predicate pushdown support

Sample Sizes of various formats:
372M Jul 29 13:12 mti300ahrs_49554_2020-07-27_5899821_11766154.hdf
155M Jul 28 11:14 mti300ahrs_49554_2020-07-27_5899821_11766154.parquet
Things we need – Files!

• Working with sensor data before:

```sql
select meas_strm_name, readout_time, readout_val_double
from hive.l0files.readouts
where site='ARIK' and DS='2020-07-01' and meas_strm_name like '%DP0.20053.001.01325%'
limit 3;
```

<table>
<thead>
<tr>
<th>meas_strm_name</th>
<th>readout_time</th>
<th>readout_val_double</th>
</tr>
</thead>
<tbody>
<tr>
<td>NEON.D10.ARIK.DP0.20053.001.01325.101.100.000</td>
<td>2020-07-01 23:49:25.262000</td>
<td>111.7164</td>
</tr>
<tr>
<td>NEON.D10.ARIK.DP0.20053.001.01325.101.100.000</td>
<td>2020-07-01 23:49:26.262000</td>
<td>111.714317</td>
</tr>
<tr>
<td>NEON.D10.ARIK.DP0.20053.001.01325.101.100.000</td>
<td>2020-07-01 23:49:27.262000</td>
<td>111.713295</td>
</tr>
</tbody>
</table>
Things we need – parquet example (simple)

- Working with sensor data now:

```bash
$ python3
Python 3.8.4 (default, Jul 14 2020, 02:58:48)
[Clang 11.0.3 (clang-1103.0.32.62)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import pandas
>>> df = pandas.read_parquet('ARIK_prt_20187_2019-01-05.parquet')
>>> df

<table>
<thead>
<tr>
<th>source_id</th>
<th>site_id</th>
<th>readout_time</th>
<th>resistance</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>20187</td>
<td>2019-01-05 00:00:00.267</td>
<td>105.743248</td>
</tr>
<tr>
<td>1</td>
<td>20187</td>
<td>2019-01-05 00:00:01.267</td>
<td>105.743698</td>
</tr>
<tr>
<td>2</td>
<td>20187</td>
<td>2019-01-05 00:00:02.267</td>
<td>105.740868</td>
</tr>
<tr>
<td>3</td>
<td>20187</td>
<td>2019-01-05 00:00:03.267</td>
<td>105.740540</td>
</tr>
<tr>
<td>4</td>
<td>20187</td>
<td>2019-01-05 00:00:04.267</td>
<td>105.739502</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>86395</td>
<td>20187</td>
<td>2019-01-05 23:59:55.267</td>
<td>103.877266</td>
</tr>
<tr>
<td>86396</td>
<td>20187</td>
<td>2019-01-05 23:59:56.267</td>
<td>103.877266</td>
</tr>
<tr>
<td>86397</td>
<td>20187</td>
<td>2019-01-05 23:59:57.267</td>
<td>103.876701</td>
</tr>
<tr>
<td>86398</td>
<td>20187</td>
<td>2019-01-05 23:59:58.267</td>
<td>103.876701</td>
</tr>
<tr>
<td>86399</td>
<td>20187</td>
<td>2019-01-05 23:59:59.267</td>
<td>103.876457</td>
</tr>
</tbody>
</table>

[86400 rows x 4 columns]
```
Things we need – parquet example (complex)

• Working with sensor data now:

```python
>>> df = pandas.read_parquet('mti300ahrs_49554_2020-07-27_5899821_11766154.parquet')
```

```
source_id  site_id  readout_time         roll    pitch        yaw  acceleration_x ... 
0           49553    HQTW 2020-07-27 00:00:00.009 -1.456751 -0.088681 113.070129        0.025998 ... 
1           49553    HQTW 2020-07-27 00:00:00.034 -1.459058 -0.086161 113.070129        0.023087 ... 
2           49553    HQTW 2020-07-27 00:00:00.059 -1.462266 -0.086978 113.070503        0.022463 ... 
3           49553    HQTW 2020-07-27 00:00:00.084 -1.464044 -0.084583 113.069305        0.031530 ... 
4           49553    HQTW 2020-07-27 00:00:00.109 -1.464376 -0.084286 113.067871        0.024759 ... 
...          ...     ...                     ...       ...       ...         ...             ...  ... 
3605508     49554    HQTW 2020-07-27 23:59:59.893 -1.757229 -0.105578 116.595657        0.028069 ... 
3605509     49554    HQTW 2020-07-27 23:59:59.918 -1.754690 -0.105534 116.593323        0.031212 ... 
3605510     49554    HQTW 2020-07-27 23:59:59.943 -1.758325 -0.105250 116.595322        0.026591 ... 
3605511     49554    HQTW 2020-07-27 23:59:59.968 -1.756375 -0.106292 116.593307        0.029373 ... 
3605512     49554    HQTW 2020-07-27 23:59:59.993 -1.757174 -0.107002 116.597626        0.027857 ... 
```

[3605513 rows x 23 columns]
Things we need – Files!

• Not just files for sensor data
  ▪ Calibration cert files (already exist as a file)
  ▪ Threshold information
    - Things like min/max allowable values
    - Parameters for quality checks
  ▪ Location information
Things we need – Example json files

```json
{
   "threshold_name": "Despiking window step - points.",
   "term_name": "rawVSICS",
   "location_name": "STEI",
   "context": [],
   "start_date": "2000-01-01T00:00:00Z",
   "end_date": null,
   "is_date_constrained": "N",
   "start_day_of_year": null,
   "end_day_of_year": null,
   "number_value": 1,
   "string_value": null
}
```
Things we need – Example json files

```json
{
  "type": "FeatureCollection",
  "features": [
    {
      "type": "Feature",
      "geometry": null,
      "properties": {
        "name": "CFGLOC101183.102",
        "site": "KONZ",
        "install_date": "2015-05-08T19:51:00Z",
        "remove_date": "2015-12-14T22:15:00Z",
        "context": [
          "aspirated-triple"
        ],
        "locations": {
          "type": "FeatureCollection",
          "features": [
            {
              "type": "Feature",
              "properties": {
                "start_date": "2010-01-01T00:00:00Z",
                "end_date": null,
                "reference_location": {
                  "type": "Feature",
                  "geometry": null,
                  "properties": {
                    "name": "CFGLOC101183.102",
                    "locations": null
                  }
                }
              }
            }
          ]
        }
      }
    }
  ]
}
```
Deployment details

- Linux operating system
- Kubernetes cluster for resource configuration & management
- Mount to Amazon S3 for data storage
## Module runtimes – 5 days of 5 prt sensors

<table>
<thead>
<tr>
<th>ID</th>
<th>PIPELINE</th>
<th>STARTED</th>
<th>DURATION</th>
<th>RESTART</th>
<th>PROGRESS</th>
<th>DL</th>
<th>UL</th>
<th>STATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>9733844e7b114f48a1daec8e801a9f77d2</td>
<td>tempAirSingle_related_location_group</td>
<td>16 hours ago 41 seconds</td>
<td>0</td>
<td>15 + 0 / 15</td>
<td>0B</td>
<td>0B</td>
<td>success</td>
<td></td>
</tr>
<tr>
<td>6c8e98f2f10ee53446bf079b3d3101212f79</td>
<td>tempAirSingle_dualfan_data_location_group_path</td>
<td>16 hours ago 16 minutes</td>
<td>0</td>
<td>1247 + 0 / 1247</td>
<td>515.1MiB</td>
<td>0B</td>
<td>success</td>
<td></td>
</tr>
<tr>
<td>5dab77f71f4d47490da253f86eael169ac</td>
<td>dualfan_merge_data_by_location</td>
<td>17 hours ago 12 minutes</td>
<td>0</td>
<td>5 + 0 / 5</td>
<td>572.1MiB</td>
<td>500MiB</td>
<td>success</td>
<td></td>
</tr>
<tr>
<td>daba64df7f4923453f9564329b64abf94</td>
<td>tempAirSingle_wind2_data_location_group_path</td>
<td>17 hours ago 13 minutes</td>
<td>0</td>
<td>993 + 0 / 993</td>
<td>840.6MiB</td>
<td>0B</td>
<td>success</td>
<td></td>
</tr>
<tr>
<td>f9d6f53ac5a144651bdf6151ee3867f89a</td>
<td>dualfan_structure_repo_by_location</td>
<td>17 hours ago 34 minutes</td>
<td>0</td>
<td>5 + 0 / 5</td>
<td>572.1MiB</td>
<td>0B</td>
<td>success</td>
<td></td>
</tr>
<tr>
<td>96919415c95144906c55cc7655101bb</td>
<td>dualfan_location_filter</td>
<td>17 hours ago About a minute</td>
<td>0</td>
<td>5 + 0 / 5</td>
<td>575.9MiB</td>
<td>612MiB</td>
<td>success</td>
<td></td>
</tr>
<tr>
<td>26fe1287811a4a6e8bf8a5b3a0dc8091</td>
<td>wind2d_merge_data_by_location</td>
<td>17 hours ago 6 minutes</td>
<td>0</td>
<td>5 + 0 / 5</td>
<td>878.9MiB</td>
<td>835.9MiB</td>
<td>success</td>
<td></td>
</tr>
<tr>
<td>52e11128c228748c8e6446ad78a503b60</td>
<td>wind2d_structure_repo_by_location</td>
<td>17 hours ago About a minute</td>
<td>0</td>
<td>5 + 0 / 5</td>
<td>879.9MiB</td>
<td>0B</td>
<td>success</td>
<td></td>
</tr>
<tr>
<td>7fbe1f9d567f4853b0f6c6f655e7c03</td>
<td>windowserverii_location_filter</td>
<td>17 hours ago 49 seconds</td>
<td>0</td>
<td>5 + 0 / 5</td>
<td>885.7MiB</td>
<td>4.749MiB</td>
<td>success</td>
<td></td>
</tr>
<tr>
<td>480b6ed570ff47b5b0bebef7a79062bce</td>
<td>tempAirSingle_qac_flags</td>
<td>18 hours ago 3 seconds</td>
<td>0</td>
<td>3 + 0 / 3</td>
<td>0B</td>
<td>0B</td>
<td>success</td>
<td></td>
</tr>
<tr>
<td>31e54c6eda54d5d7f77fa79bce0baasf6c</td>
<td>tempAirSingle_qac_data</td>
<td>18 hours ago 3 seconds</td>
<td>0</td>
<td>3 + 0 / 3</td>
<td>0B</td>
<td>0B</td>
<td>success</td>
<td></td>
</tr>
<tr>
<td>554da1e79055d438d8c8beb7f4b67f77f</td>
<td>tempsoil_level_group</td>
<td>18 hours ago 7 seconds</td>
<td>0</td>
<td>3 + 0 / 3</td>
<td>0B</td>
<td>0B</td>
<td>success</td>
<td></td>
</tr>
<tr>
<td>57a11128c228748c8e6446ad78a503b60</td>
<td>tempsoil_soil_statistics</td>
<td>18 hours ago 59 seconds</td>
<td>0</td>
<td>3 + 0 / 3</td>
<td>1.089MiB</td>
<td>362.6KiB</td>
<td>success</td>
<td></td>
</tr>
<tr>
<td>eee30ba2422b04a8be845c96f2755c308</td>
<td>tempsoil_quality_metrics</td>
<td>18 hours ago 36 seconds</td>
<td>0</td>
<td>3 + 0 / 3</td>
<td>473KiB</td>
<td>116.6KiB</td>
<td>success</td>
<td></td>
</tr>
<tr>
<td>ea76c4d153b240399808b5db54b0f85</td>
<td>tempsoil_qac_regularized_flag_group</td>
<td>18 hours ago 6 seconds</td>
<td>0</td>
<td>9 + 0 / 9</td>
<td>0B</td>
<td>0B</td>
<td>success</td>
<td></td>
</tr>
<tr>
<td>7054f0d56a5247938f7e4b54d977a0700</td>
<td>tempsoil_stats_uncertainty_group</td>
<td>18 hours ago 7 seconds</td>
<td>0</td>
<td>3 + 0 / 3</td>
<td>0B</td>
<td>0B</td>
<td>success</td>
<td></td>
</tr>
<tr>
<td>d50d04ff7c5640999212a9c0b21646b</td>
<td>tempsoil_qac_flags</td>
<td>18 hours ago 12 seconds</td>
<td>0</td>
<td>3 + 0 / 3</td>
<td>0B</td>
<td>0B</td>
<td>success</td>
<td></td>
</tr>
<tr>
<td>83eb54c1e387438eb9c929a0d1cbf714</td>
<td>tempsoil_data</td>
<td>18 hours ago 48 seconds</td>
<td>0</td>
<td>3 + 0 / 3</td>
<td>1.417MiB</td>
<td>677.6KiB</td>
<td>success</td>
<td></td>
</tr>
<tr>
<td>63b2a73c7261b98b07b12453f3b8123</td>
<td>tempsoil_qac_plausibility</td>
<td>18 hours ago 48 seconds</td>
<td>0</td>
<td>3 + 0 / 3</td>
<td>2.34MiB</td>
<td>0B</td>
<td>success</td>
<td></td>
</tr>
<tr>
<td>45e27ee982d44708eb3308fd4bf28a1</td>
<td>tempsoil_padded_timeseries_analyzer</td>
<td>18 hours ago 7 zeroes</td>
<td>0</td>
<td>7 + 0 / 7</td>
<td>863.5KiB</td>
<td>495B</td>
<td>success</td>
<td></td>
</tr>
<tr>
<td>50bf7f38e2d06d07d450102cbcb</td>
<td>timeseries_padder</td>
<td>18 hours ago 4 zeroes</td>
<td>0</td>
<td>5 + 0 / 5</td>
<td>346.9KiB</td>
<td>349KiB</td>
<td>success</td>
<td></td>
</tr>
<tr>
<td>14146564984b4eb39dbcd7e60484c8f</td>
<td>tempsoil_threshold_regularized_group</td>
<td>18 hours ago 4 seconds</td>
<td>0</td>
<td>5 + 0 / 5</td>
<td>780.7KiB</td>
<td>715.1KiB</td>
<td>success</td>
<td></td>
</tr>
<tr>
<td>41129260d2b98c0c3f8bee1c4e8c6f5</td>
<td>tempsoil_regularized_data</td>
<td>18 hours ago 4 seconds</td>
<td>0</td>
<td>5 + 0 / 5</td>
<td>1.046MiB</td>
<td>1.048MiB</td>
<td>success</td>
<td></td>
</tr>
<tr>
<td>31add8750d04d63d4eb226d4ea6f52fa</td>
<td>tempsoil_threshold_uncertainty_coins</td>
<td>18 hours ago 4 seconds</td>
<td>0</td>
<td>5 + 0 / 5</td>
<td>1.333MiB</td>
<td>83.32KiB</td>
<td>success</td>
<td></td>
</tr>
<tr>
<td>d00e983e97211666bca339390838331</td>
<td>tempsoil_calibrated_flags</td>
<td>18 hours ago 3 seconds</td>
<td>0</td>
<td>5 + 0 / 5</td>
<td>1.686KiB</td>
<td>0B</td>
<td>success</td>
<td></td>
</tr>
<tr>
<td>10393f3a83b4eb5ad4ca764ac550194</td>
<td>tempsoil_calibrated_data</td>
<td>18 hours ago 3 seconds</td>
<td>0</td>
<td>5 + 0 / 5</td>
<td>13.22MiB</td>
<td>6.363MiB</td>
<td>success</td>
<td></td>
</tr>
<tr>
<td>badd9d42c38db2e2eb57b2554abfba82</td>
<td>tempsoil_uncertainty_fds</td>
<td>18 hours ago 3 seconds</td>
<td>0</td>
<td>5 + 0 / 5</td>
<td>21.94MiB</td>
<td>0B</td>
<td>success</td>
<td></td>
</tr>
<tr>
<td>6f7c044d976a466bab5f650577647f78</td>
<td>tempsoil_padded_timeseries_analyzer</td>
<td>18 hours ago 5 zeroes</td>
<td>0</td>
<td>7 + 0 / 7</td>
<td>7.374MiB</td>
<td>495B</td>
<td>success</td>
<td></td>
</tr>
<tr>
<td>d6f9b9f93d114eb2aa3393924952b305</td>
<td>tempsoil_timeseries_padder</td>
<td>18 hours ago 5 zeroes</td>
<td>0</td>
<td>5 + 0 / 5</td>
<td>0B</td>
<td>0B</td>
<td>success</td>
<td></td>
</tr>
<tr>
<td>3b24765930d5d5e884c83eb7551836a5</td>
<td>tempsoil_threshold_regularized_group</td>
<td>18 hours ago 4 seconds</td>
<td>0</td>
<td>5 + 0 / 5</td>
<td>0B</td>
<td>0B</td>
<td>success</td>
<td></td>
</tr>
</tbody>
</table>
Graph view in airflow
Kubernetes Metrics