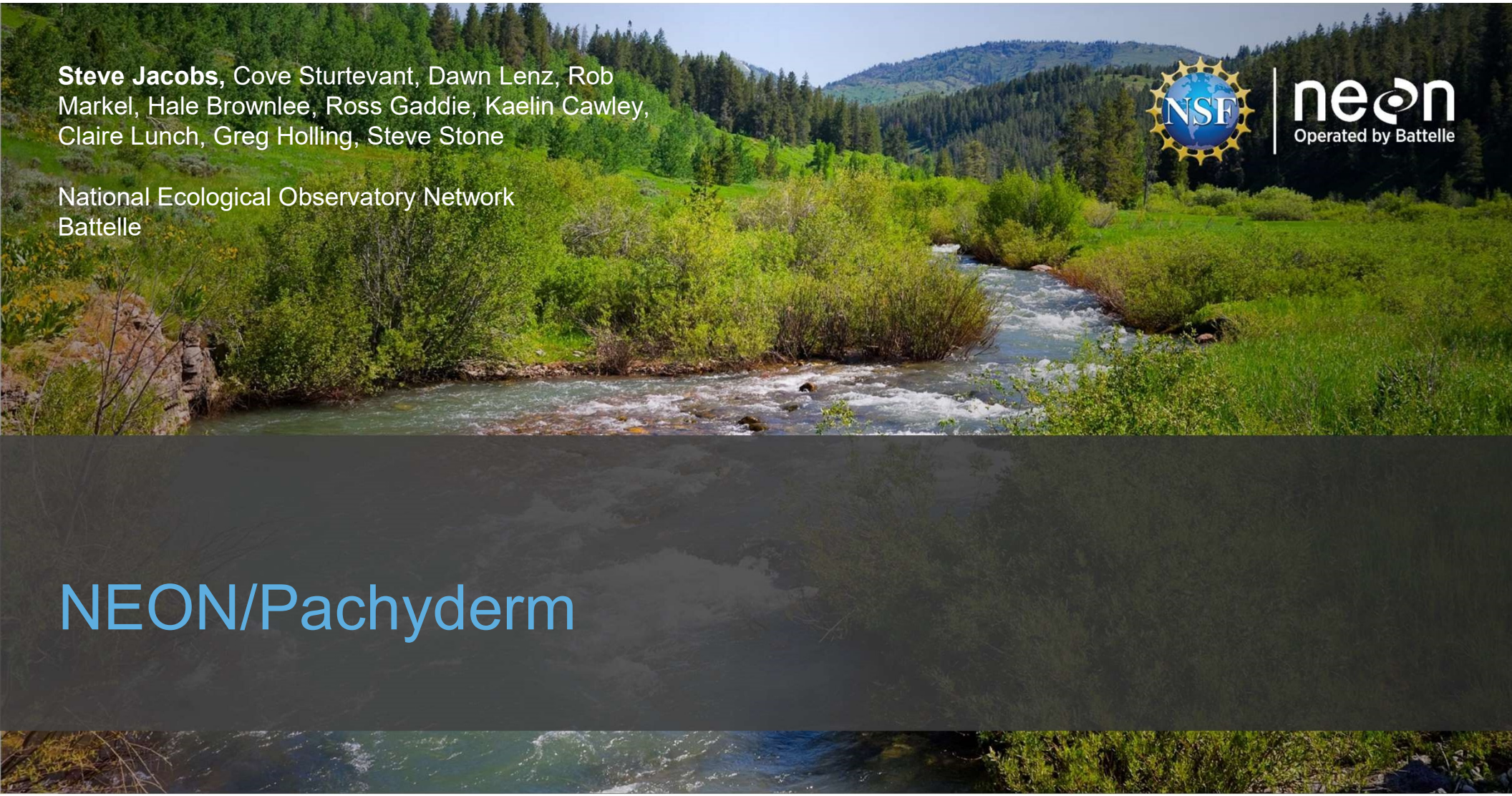

CI/CS WORKSHOP
THE COMMUNITY TOGETHER
ResearchSOC
CI CoE PILOT

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/



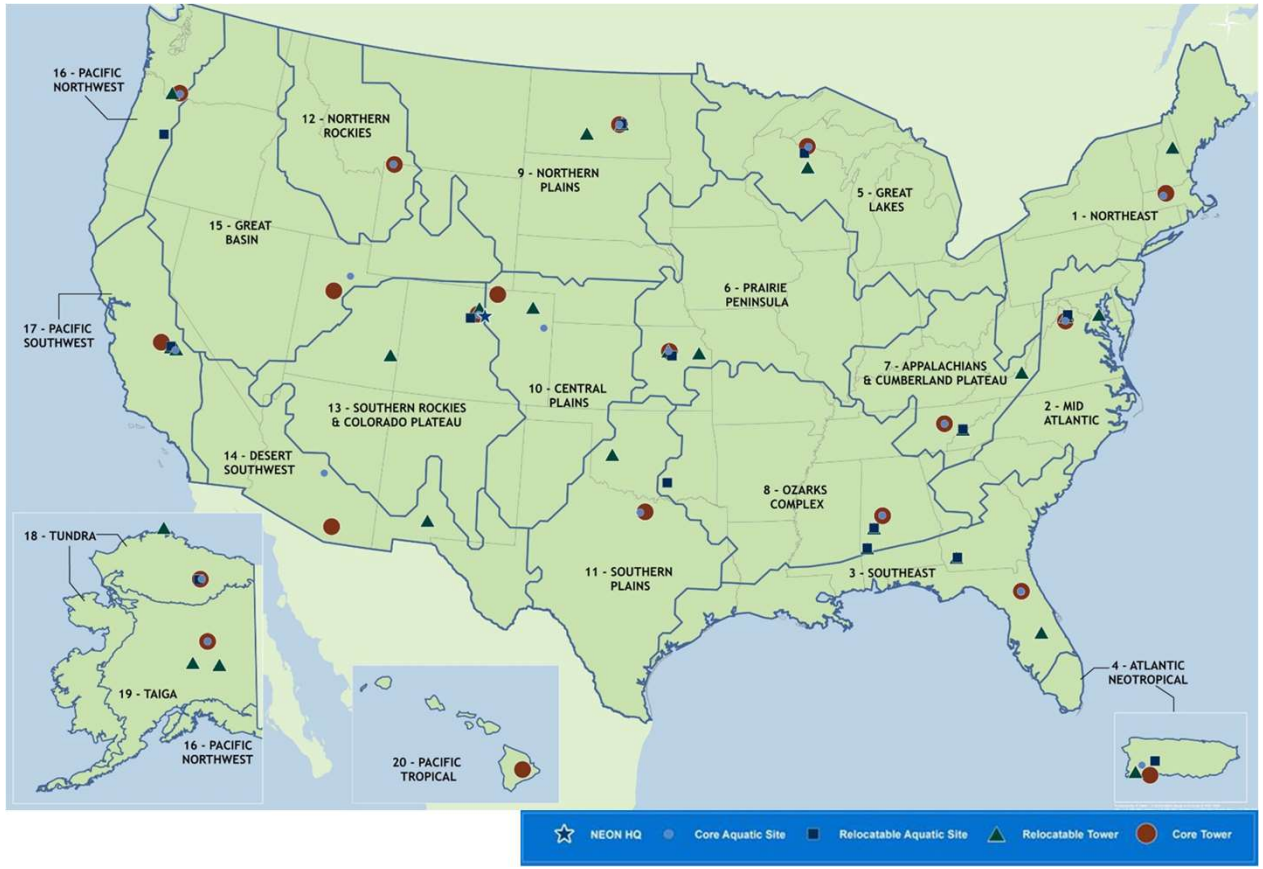
Steve Jacobs, Cove Sturtevant, Dawn Lenz, Rob Markel, Hale Brownlee, Ross Gaddie, Kaelin Cawley, Claire Lunch, Greg Holling, Steve Stone

National Ecological Observatory Network
Battelle

NEON/Pachyderm

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

National Ecological Observatory Network



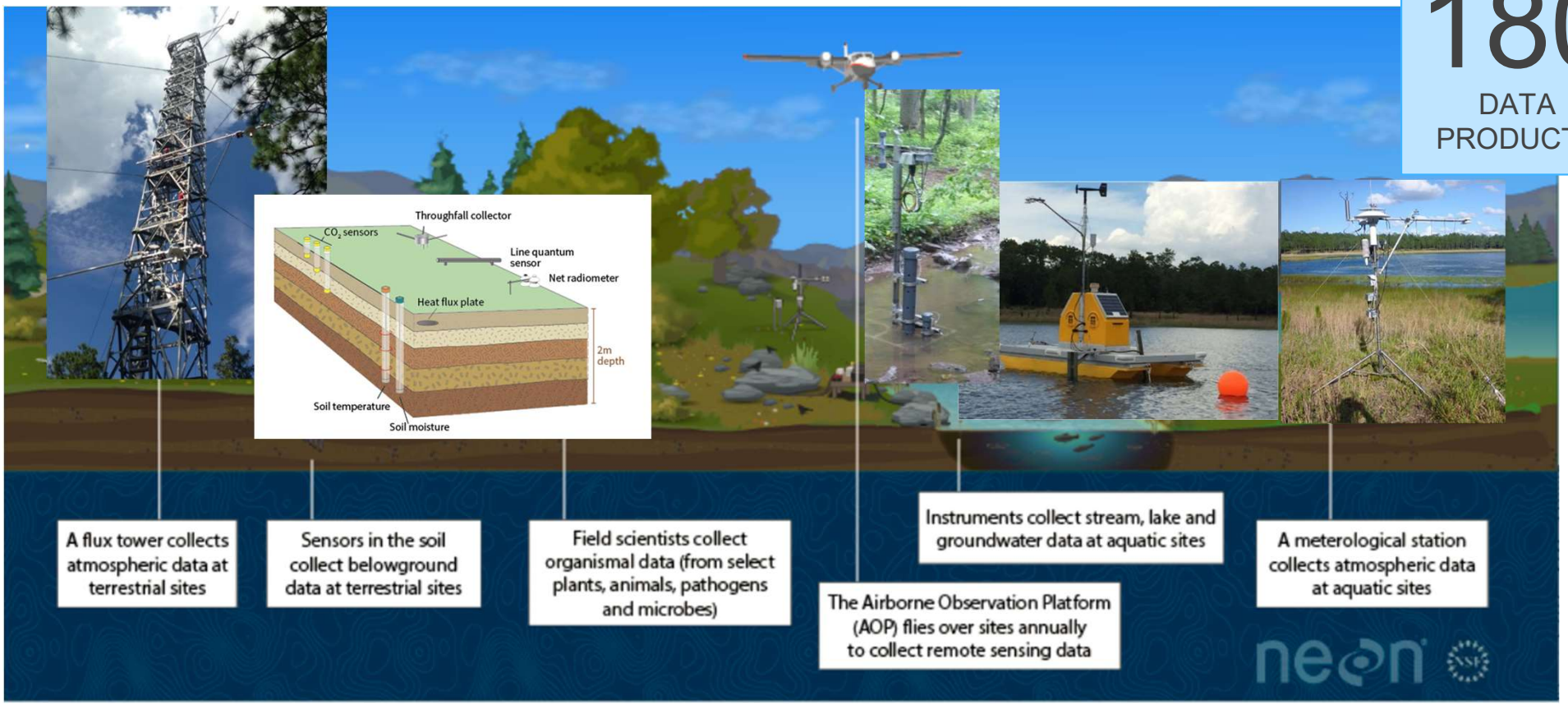
81

FIELD SITES

- 47 terrestrial
- 34 aquatic

National Ecological Observatory Network

Over
180
DATA PRODUCTS

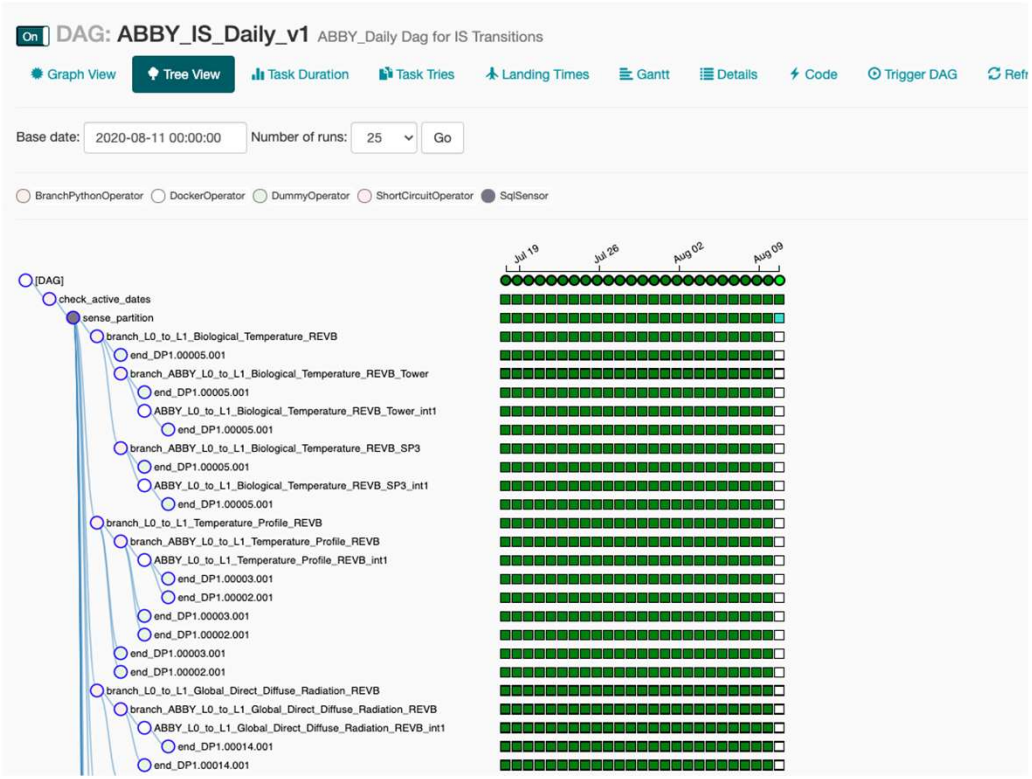


Existing processing system

- Based on apache airflow https://airflow.apache.org/
  - ETL Workflow tool
  - Date based
- Good visibility and operational tooling
  - Clear reporting
  - Easily re-run data through pipeline
  - Performance metrics

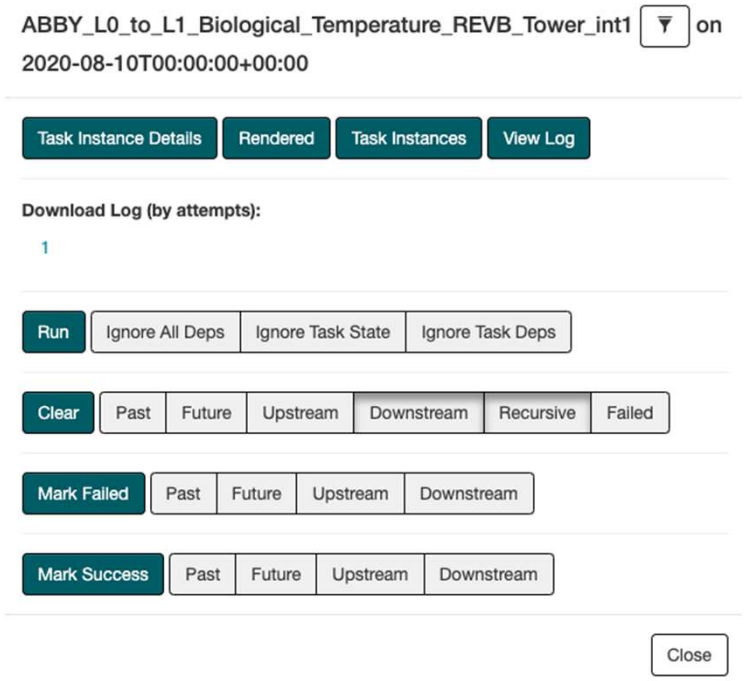
Existing processing system

- 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

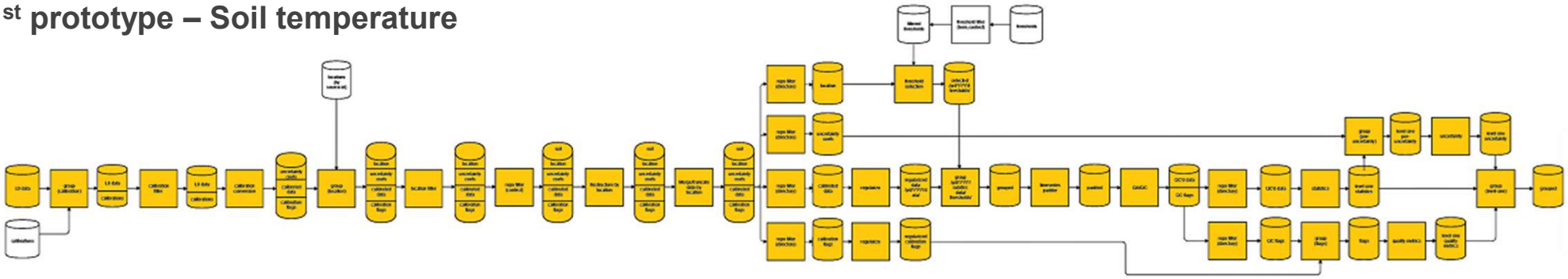
- Dynamic data (a lot of it)
  - Continuous receipt of new data
  - Metadata and parameter adjustments
- Data provenance
  - Traceability
  - Reproducibility
- Code re-usability / Use outside of NEON
- Integrated Science-Cyberinfrastructure development

System design

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

Pachyderm

1st prototype – Soil temperature

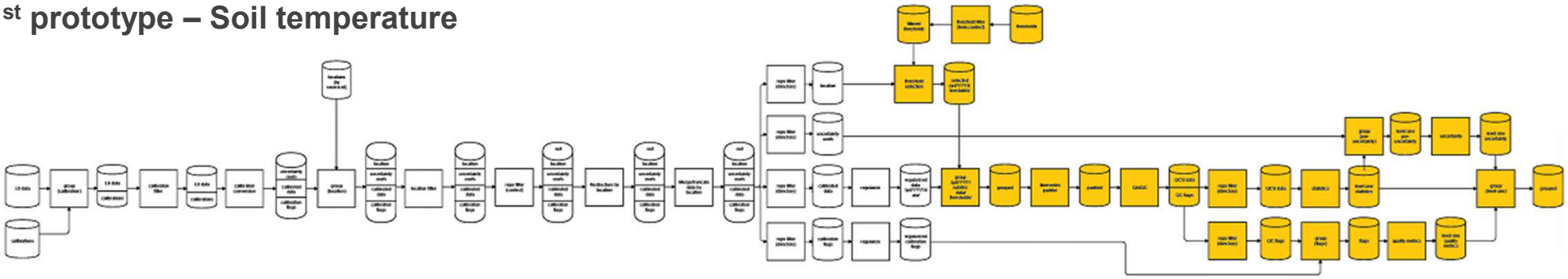


System design

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

Pachyderm

1st prototype – Soil temperature



Connecting modules to data with a pipeline spec

Name

Docker image and command to run

Input repository and what is a datum

```
{
  "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": "/prt/*/*/*/"
    }
  },
  "parallelism_spec": {
    "constant": "2"
  },
  "resource_requests": {
    "memory": "200M",
    "cpu": 0
  },
  "standby":true
}
```

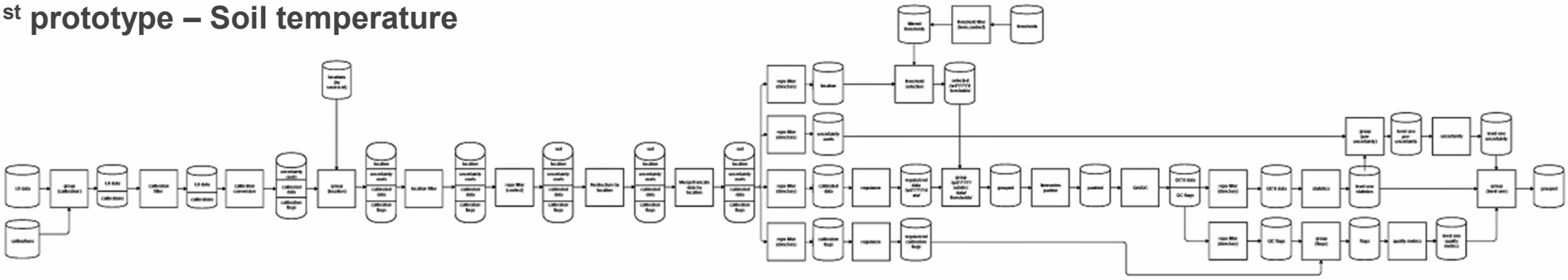
Line: 25/27 | Column: 5 | Encoding: 1252 (ANSI - La | Modified

System design

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

Pachyderm

1st prototype – Soil temperature



Tracing provenance through the commit chain

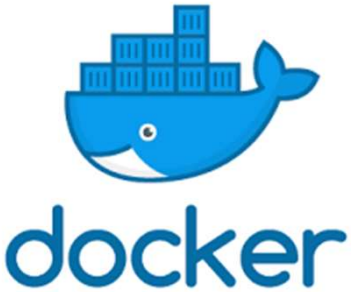
```
csturtevant@den-devissom:/ $ pachctl list commit tempSoil_level1_group
REPO                  BRANCH COMMIT                           FINISHED      SIZE     PROGRESS DESCRIPTION
tempSoil_level1_group master f6e26c5893614e51aa3c7b93c67c89e4 20 hours ago 479.2KiB -
tempSoil_level1_group master d3fec2c105fd4912a81e786f03dcf66d 22 hours ago 479.2KiB -
```

```
csturtevant@den-devissom:/ $ pachctl inspect commit tempSoil_level1_group@d3fec2c105fd4912a81e786f03dcf66d
Commit: tempSoil_level1_group@d3fec2c105fd4912a81e786f03dcf66d
Original Branch: master
Parent: 80e8181b163d4f699014213992506403
Started: 23 hours ago
Finished: 23 hours ago
Size: 479.2KiB
Provenance:  avro_schemas@6890fd5f56924fe0a5c3344d01b2706b (master)  location_assets@58d585cfd96d4f7a91cde
87ab412fbbe (master)  prt_soil_threshold_filter@8d3591fd2d7f41959dfacbdd0dc50f10 (master)  threshold@27663
c6f6c4d4f68923d10589c88d055 (master)  tempSoil_quality_metrics@dd09c3c867514f58b5173624752af504 (master)
__spec__@7eb4c1dd4eb54298b24cd21aee497c27 (prt_calibrated_location_group)  __spec__@f9cdf1c616624e43a352fb
81633c1741 (tempSoil_padded_timeseries_analyzer)  tempSoil_calibrated_data@fdc9bc04cfc844cdba9c360e3eb4293
2 (master)  spec  @aaec63b61953483886c86cd00145b454 (prt_calibration_filter)  spec  @6902b30a1b0941da9
```

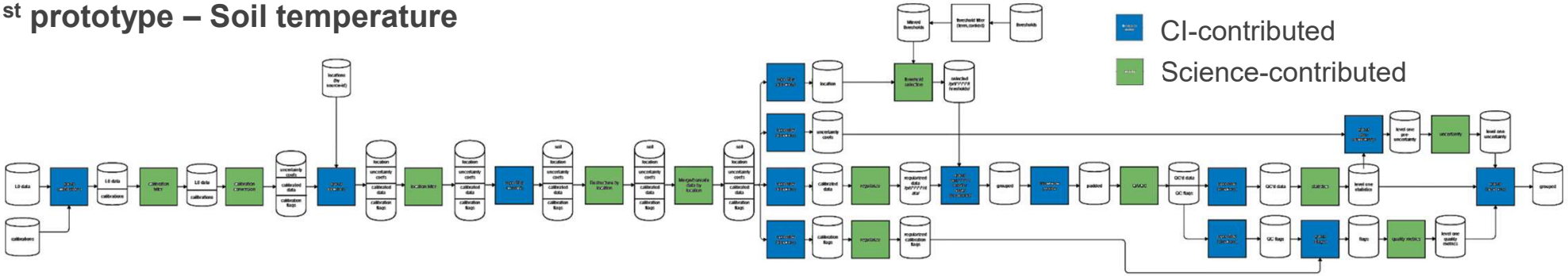
```
csturtevant@den-devissom:/ $ pachctl list file prt_soil_threshold_filter@8d3591fd2d7f41959dfacbdd0dc50f10
NAME             TYPE SIZE
/thresholds.json file 259.9KiB
```

System design

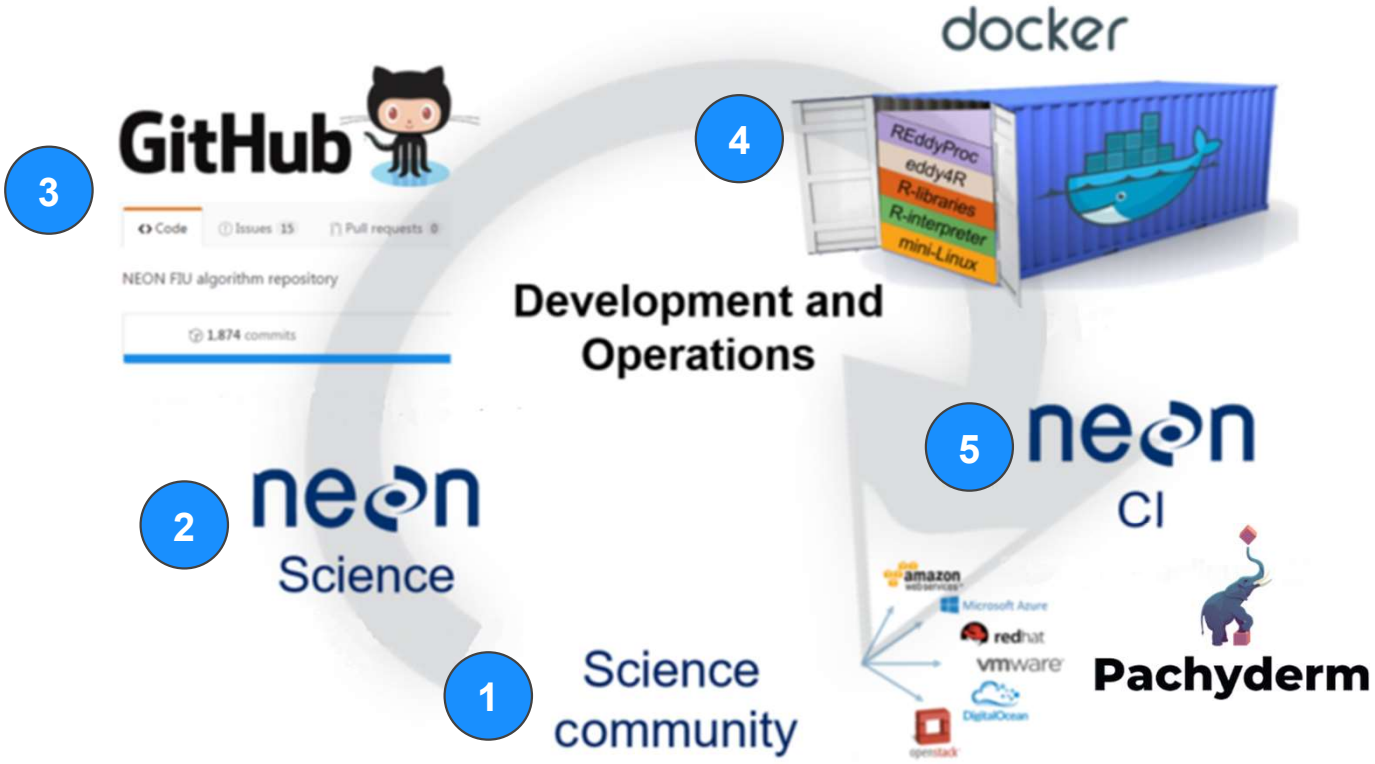
NEED	SOLUTION
Automated response to data change (raw data, calibrations, location info, etc)	Pachyderm-based processing modules 'listen' for any data change
Traceability	Git-like version control for data and code
Reproducibility	Version-controlled Docker containers contain code and dependencies
Code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



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/)
    - 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/)
    - Python (Pandas)
    - R
    - C/C++ / Rust / MATLAB more coming soon
  - Column based compression, native types, predicate pushdown support

```
Sample Sizes of various formats:
243M Jul 29 13:18 mti300ahrs_49554_2020-07-27_5899821_11766154.csv.gz
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:

```
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;
```

```
meas_strm_name                                |readout_time             |readout_val_double|
----------------------------------------------|-------------------------|------------------
NEON.D10.ARIK.DP0.20053.001.01325.101.100.000|2020-07-01 23:49:25.262000|          111.7164|
NEON.D10.ARIK.DP0.20053.001.01325.101.100.000|2020-07-01 23:49:26.262000|         111.714317|
NEON.D10.ARIK.DP0.20053.001.01325.101.100.000|2020-07-01 23:49:27.262000|         111.713295|
```

Things we need – parquet example (simple)

- Working with sensor data now:

```
$ 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
      source_id site_id            readout_time  resistance
0         20187    BARC 2019-01-05 00:00:00.267  105.743248
1         20187    BARC 2019-01-05 00:00:01.267  105.743698
2         20187    BARC 2019-01-05 00:00:02.267  105.740868
3         20187    BARC 2019-01-05 00:00:03.267  105.740540
4         20187    BARC 2019-01-05 00:00:04.267  105.739502
...         ...     ...                     ...         ...
86395     20187    BARC 2019-01-05 23:59:55.267  103.877266
86396     20187    BARC 2019-01-05 23:59:56.267  103.877266
86397     20187    BARC 2019-01-05 23:59:57.267  103.876701
86398     20187    BARC 2019-01-05 23:59:58.267  103.876701
86399     20187    BARC 2019-01-05 23:59:59.267  103.876457

[86400 rows x 4 columns]
>>>
```

Things we need – parquet example (complex)

- Working with sensor data now:

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


```
{
    "threshold_name": "Despiking window step - points.",
    "term_name": "rawVSIC5",
    "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


```
{
    "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
```

Operated by Battelle

720.746.4844 | neonscience@battelleecology.org | neonscience.org

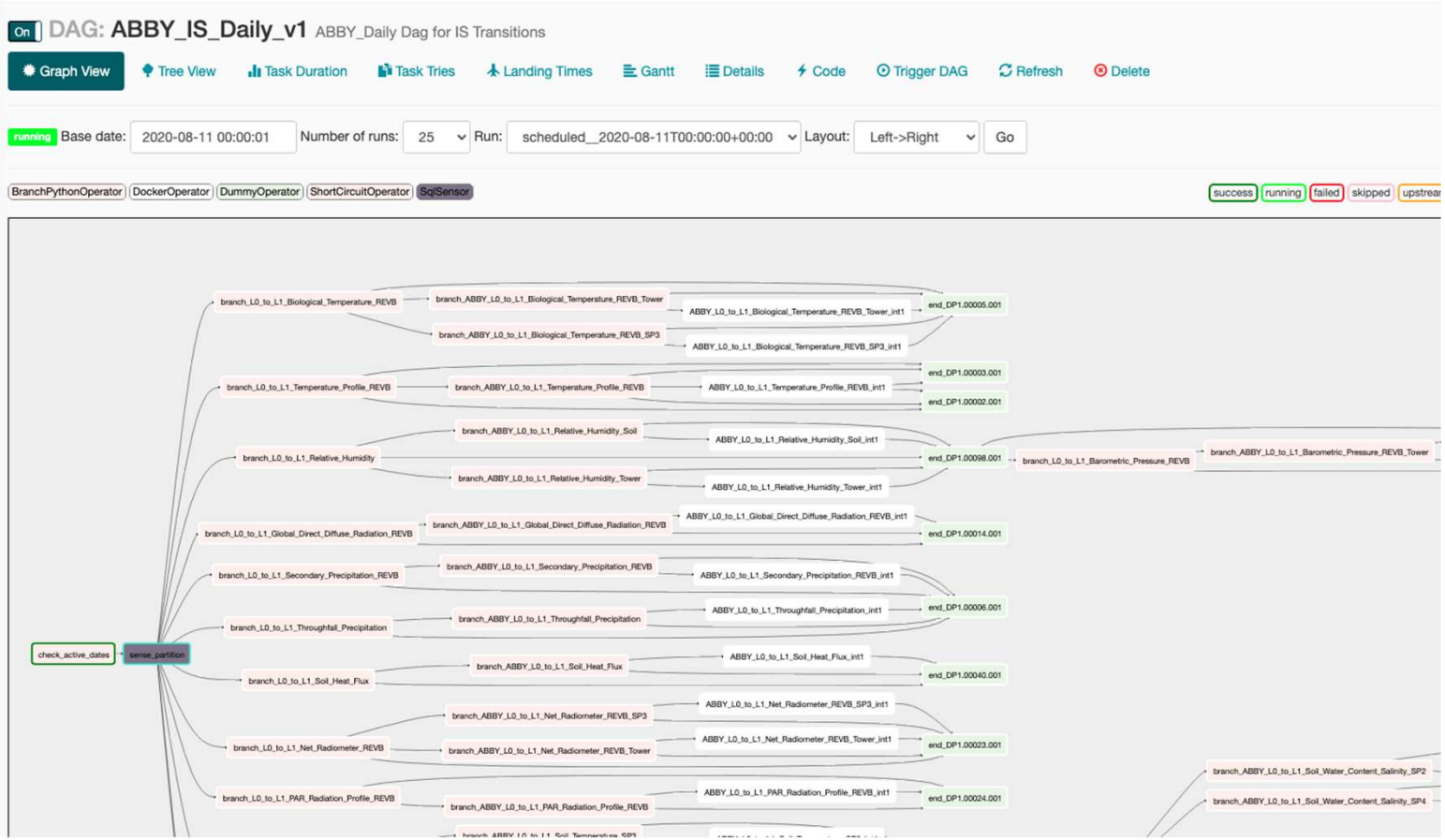
Deployment details

- Linux operating system
- Kubernetes cluster for resource configuration & management
- Mount to Amazon S3 for data storage

Module runtimes – 5 days of 5 prt sensors

ID	PIPELINE	STARTED	DURATION	RESTART	PROGRESS	DL	UL	STATE
9733044e7b114f48a1dac8e01a9f74d2	tempAirSingle_related_location_group	16 hours ago	41 seconds	0	15 + 0 / 15	0B	0B	success
6ce9ef216e534e0f979b3d331021c2f9	tempAirSingle_dualfan_data_location_group_path	16 hours ago	16 minutes	0	1247 + 0 / 1247	515.1MiB	0B	success
3d3ab7ff1d47490da25f386ea1f169ac	dualfan_merge_data_by_location	17 hours ago	12 minutes	0	5 + 0 / 5	572.1MiB	509MiB	success
4aba64d7f49243598f564329be64bf94	tempAirSingle_wind2d_data_location_group_path	17 hours ago	13 minutes	0	993 + 0 / 993	840.6MiB	0B	success
f9d6f53ca5144651bd0f151ee386f89a	dualfan_structure_repo_by_location	17 hours ago	34 minutes	0	5 + 0 / 5	572.1MiB	0B	success
99619415c55144969c55cc7e55101bbb	dualfan_location_filter	17 hours ago	About a minute	0	5 + 0 / 5	575.9MiB	6.112MiB	success
26fe1287811a4a6ebcf8a6ba30dc8091	wind2d_merge_data_by_location	17 hours ago	6 minutes	0	5 + 0 / 5	878.9MiB	835.9MiB	success
32e1128c228748c8a646ad78a503b7b0	wind2d_structure_repo_by_location	17 hours ago	About a minute	0	5 + 0 / 5	878.9MiB	0B	success
7f8ee1fd56f74853b0f0c6fb655e7c03	windobserverii_location_filter	17 hours ago	49 seconds	0	5 + 0 / 5	885.7MiB	4.749MiB	success
480be5d750ff47b5bbebef7a79062bce	tempAirSingle_qaqc_flags	18 hours ago	3 seconds	0	3 + 0 / 3	0B	0B	success
81ee54ceeda54445b771fa79bce98a99	tempAirSingle_qaqc_data	18 hours ago	3 seconds	0	3 + 0 / 3	0B	0B	success
254d1a1e79554538acd8aeb87f4b6f7f	tempSoil_level1_group	18 hours ago	7 seconds	0	9 + 0 / 9	0B	0B	success
d7a1812ffea84b7cab8b179f638dab64	tempSoil_statistics	18 hours ago	59 seconds	0	3 + 0 / 3	1.089MiB	362.6KiB	success
ee30ba2422b04abe8485c9f62755c308	tempSoil_quality_metrics	18 hours ago	36 seconds	0	3 + 0 / 3	473KiB	116.6KiB	success
ea76cd4135b24039890686bed54bf08b	tempSoil_qaqc_regularized_flag_group	18 hours ago	6 seconds	0	9 + 0 / 9	0B	0B	success
7054fd956a52479387e4a54bd077a790	tempSoil_statistics_uncertainty_group	18 hours ago	7 seconds	0	9 + 0 / 9	0B	0B	success
d5d0044f7c5640998121a9c01b216462	tempSoil_qaqc_flags	18 hours ago	2 seconds	0	3 + 0 / 3	0B	0B	success
834b541cc3f74386bc929da0dc1fb714	tempSoil_qaqc_data	18 hours ago	2 seconds	0	3 + 0 / 3	0B	0B	success
6d3ba37c2c614b908b71224533f88123	tempSoil_qaqc_plausibility	18 hours ago	48 seconds	0	3 + 0 / 3	1.417MiB	677.6KiB	success
45e27ee982d44078be3308f0bd4f28a1	tempSoil_padded_timeseries_analyzer	18 hours ago	5 seconds	0	7 + 0 / 7	2.34MiB	0B	success
59f9fce3d8e048d6907d45d350012bcb	tempSoil_timeseries_padder	18 hours ago	5 seconds	0	5 + 0 / 5	863.5KiB	495B	success
1144156841b44e6bb39cdb17e0448cfa	tempSoil_threshold_regularized_group	18 hours ago	4 seconds	0	5 + 0 / 5	0B	0B	success
4e119085c66142bfb11c3fbeeb41ecf5	tempSoil_regularized_data	18 hours ago	4 seconds	0	5 + 0 / 5	780.7KiB	715.1KiB	success
d92d10f9059b45f19012ff47f32283d4	tempSoil_regularized_flags	18 hours ago	4 seconds	0	5 + 0 / 5	346.9KiB	349KiB	success
d78407b81c8147e491d4301bfef2e93f	tempSoil_regularized_uncertainty_fdas	18 hours ago	4 seconds	0	5 + 0 / 5	1.046MiB	1.048MiB	success
a1dddb750d804d63a4ec8a26d8aa4efa	tempSoil_threshold_select	18 hours ago	7 seconds	0	5 + 0 / 5	1.333MiB	83.32KiB	success
dc0ee3e97231466dbca3393908f38331	tempSoil_uncertainty_coefficients	18 hours ago	4 seconds	0	5 + 0 / 5	0B	0B	success
55e1c29c33be41888dedb3c7daa4bcd0	tempSoil_calibrated_flags	18 hours ago	3 seconds	0	5 + 0 / 5	0B	0B	success
10139f3fa83b4eb5a4bca7644c550194	tempSoil_calibrated_data	18 hours ago	4 seconds	0	5 + 0 / 5	0B	0B	success
badd9b42c38b42eba5fb7a2654abfba4	tempSoil_uncertainty_fdas	18 hours ago	3 seconds	0	5 + 0 / 5	0B	0B	success
bfe44475daa74b66abb5600737647f78	tempSoil_locations	18 hours ago	3 seconds	0	5 + 0 / 5	0B	0B	success
d91affd9c1f6449e88e1820b35e37e6e	tempSoil_fill_date_gaps_by_location	18 hours ago	3 seconds	0	1 + 0 / 1	1.686KiB	0B	success
c4d4b197655d4f45bb03f4c1468c077e	tempAirSingle_qaqc_plausibility	18 hours ago	14 minutes	0	3 + 0 / 3	13.22MiB	6.363MiB	success
e63df67dd4594836b7f7999f28a15db8	tempAirSingle_padded_timeseries_analyzer	18 hours ago	5 seconds	0	7 + 0 / 7	21.94MiB	0B	success
d16e9bff901149e2aa33929c49526b05	tempAirSingle_timeseries_padder	18 hours ago	5 seconds	0	5 + 0 / 5	7.374MiB	495B	success
592478539dd546e68c48e047cac5285a	tempAirSingle_threshold_regularized_group	18 hours ago	4 seconds	0	5 + 0 / 5	0B	0B	success

Graph view in airflow



Kubernetes Metrics