PUG 2021

Pegasus Users Group

MEETING

GeoEDF: A Framework for Geospatial Research Workflows

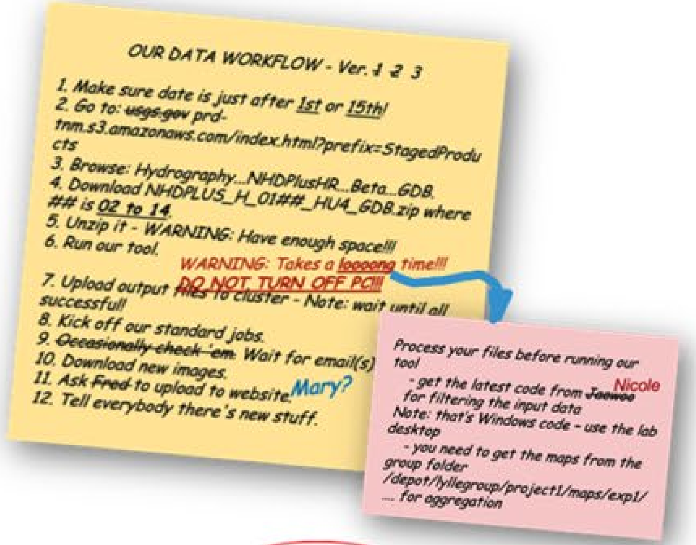
Rajesh Kalyanam

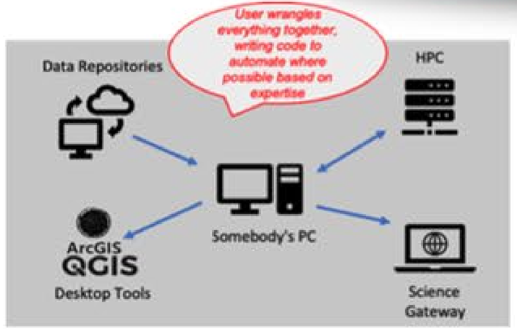
Research Scientist, Purdue Research Computing

02/25/21

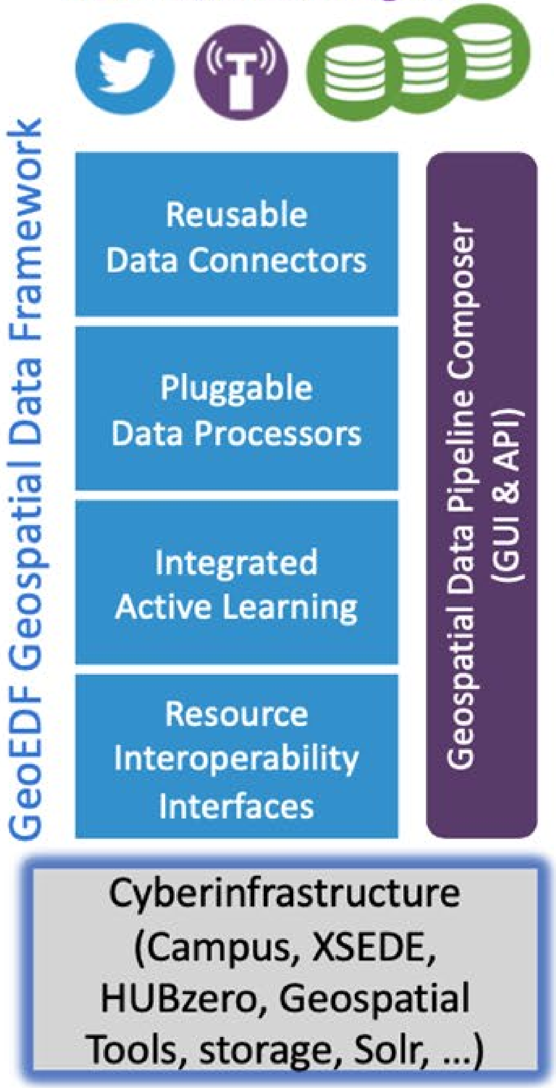
GeoEDF Vision

Researchers spend up to 80% of their time "wrangling data"

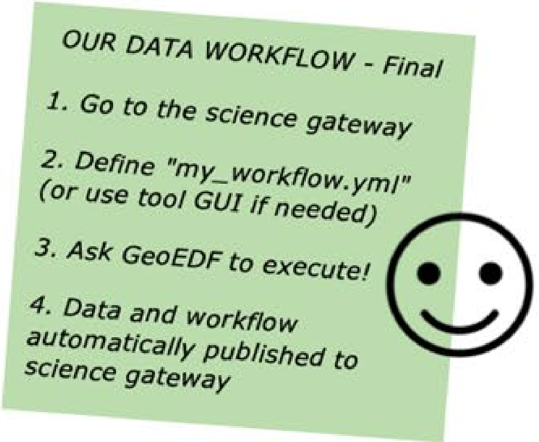




Remote data repos, smart devices, streaming data



Make Science FAIR



- Remote data directly usable in code, seamless workflow
- Complexity abstracted away
- Reusable data connectors, processors, and workflows
- Automatic provenance capture & data annotation => FAIR

The GeoEDF Project

An Extensible Geospatial Data Framework Towards FAIR Science

To help data-driven sciences to be more
Findable, Accessible, Interoperable, Reusable

funded by NSF CSSI program award #: 1835822, Oct 2018 - Sep 2023

Multidisciplinary Project Leadership

Jian Jin

Plant phenotyping & sensors
Ag & Biologyical Engineering

Venkatesh Merwade

Flood modeling & visualization
Civil Engineering

Carol Song

Cyberinfrastructure
Research Computing

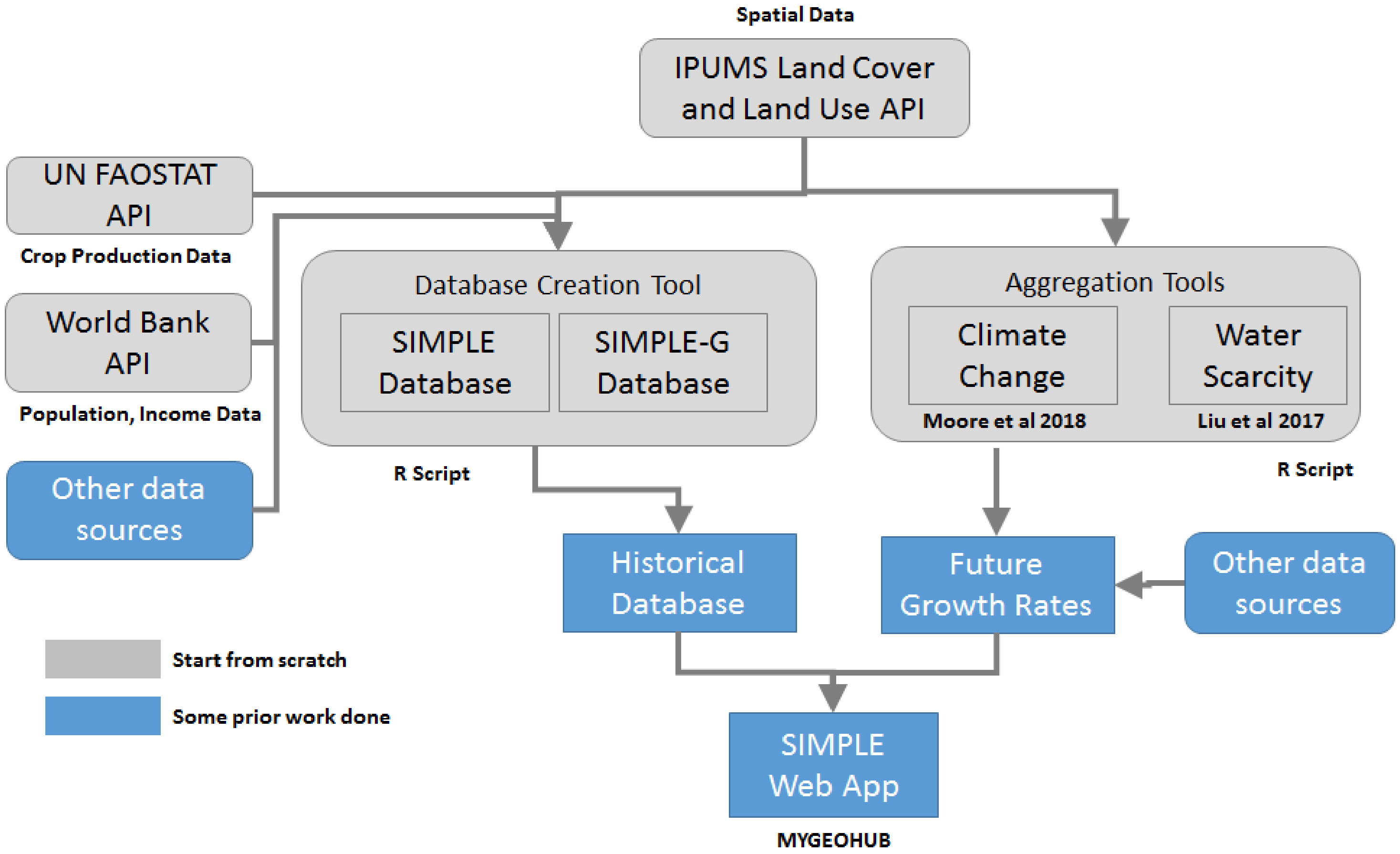
Jack Smith

Water Quality & resource management
Marshall University

Uris Baldos

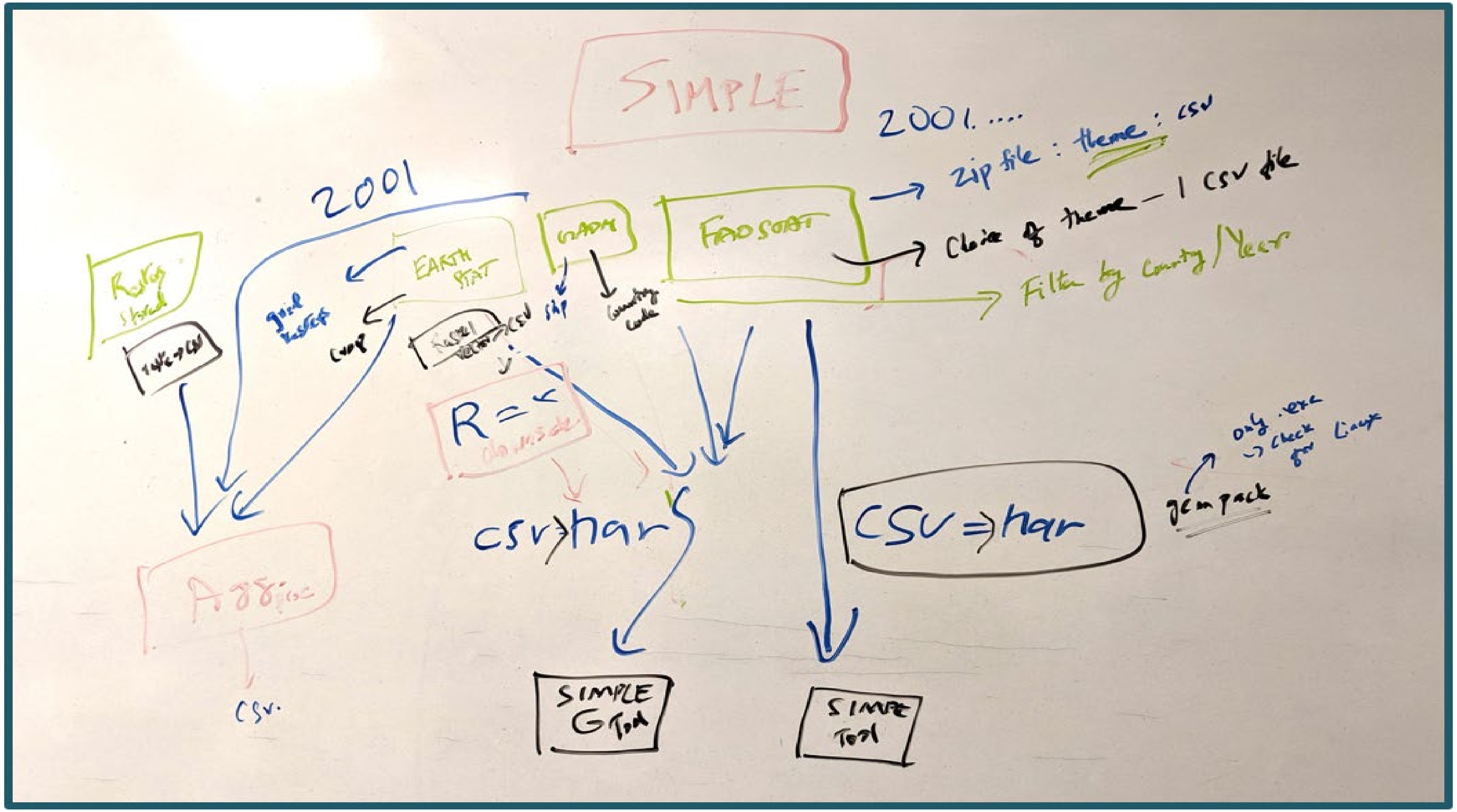
Sustainable development
Agricultural Economics

Example Workflow from Agricultural Economics



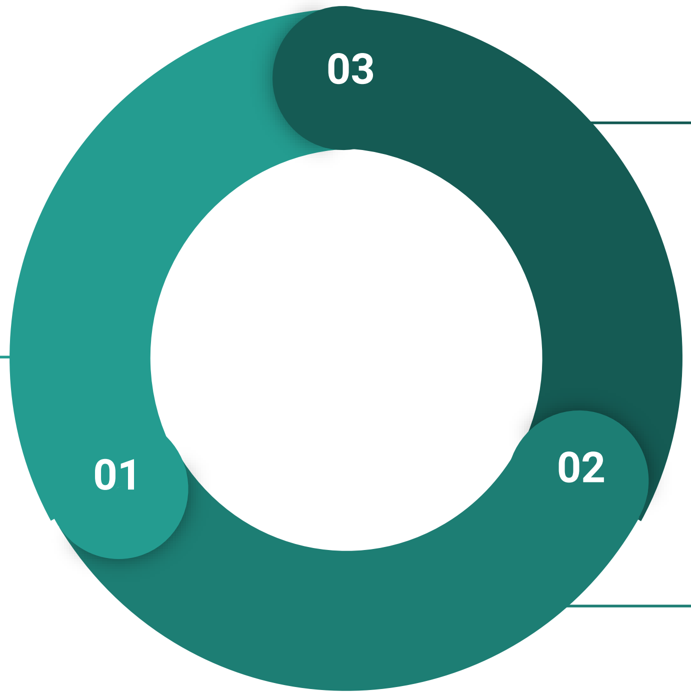
Multidisciplinary domains often need to access diverse datasets and integrate them with existing models

Reality!



Working through the specifics reveals the more messy details!

GeoEDF Design Principles



Streamline data wrangling in research workflows

Enable researchers to break down a complex research task into a collection of data acquisition and processing sub-tasks

Promote FAIR science principles

Integrate GeoEDF and cyberinfrastructure to implicitly & explicitly promote FAIR science principles

Provide reusable and scalable workflow building blocks

Improve the efficiency of day-to-day research workflows by enabling standardization, reuse, composition, and scalable execution

GeoEDF Components

Reusable Data Connectors

Implement various data access protocols, enable data acquisition from popular

Reusable Data Processors

Implement domain agnostic & domain specific geospatial processing operations

Plug-and-play Workflow Composer

Enable the composition of individual connectors & processors into

GeoEDF

Enable researchers to conceive of geospatial data driven workflows as a sequence of data acquisition and processing steps that can be carried out using pre-existing or user contributed

Data Connector Examples

NASA	MODIS, SMAP, other Earthdata DAACs
USGS	Elevation, land use, hydrography, Gage, NLDI
USDA	Soil, land cover, land use
CUASHI	Rainfall, Hydroshare resources
EarthStat	Crop data
FAO	Arable land, harvest data
CIESIN	Population data
EPA	Water quality
Others (no API yet)	Open Data Cubes, Google Earth Engine, ESS-Dive

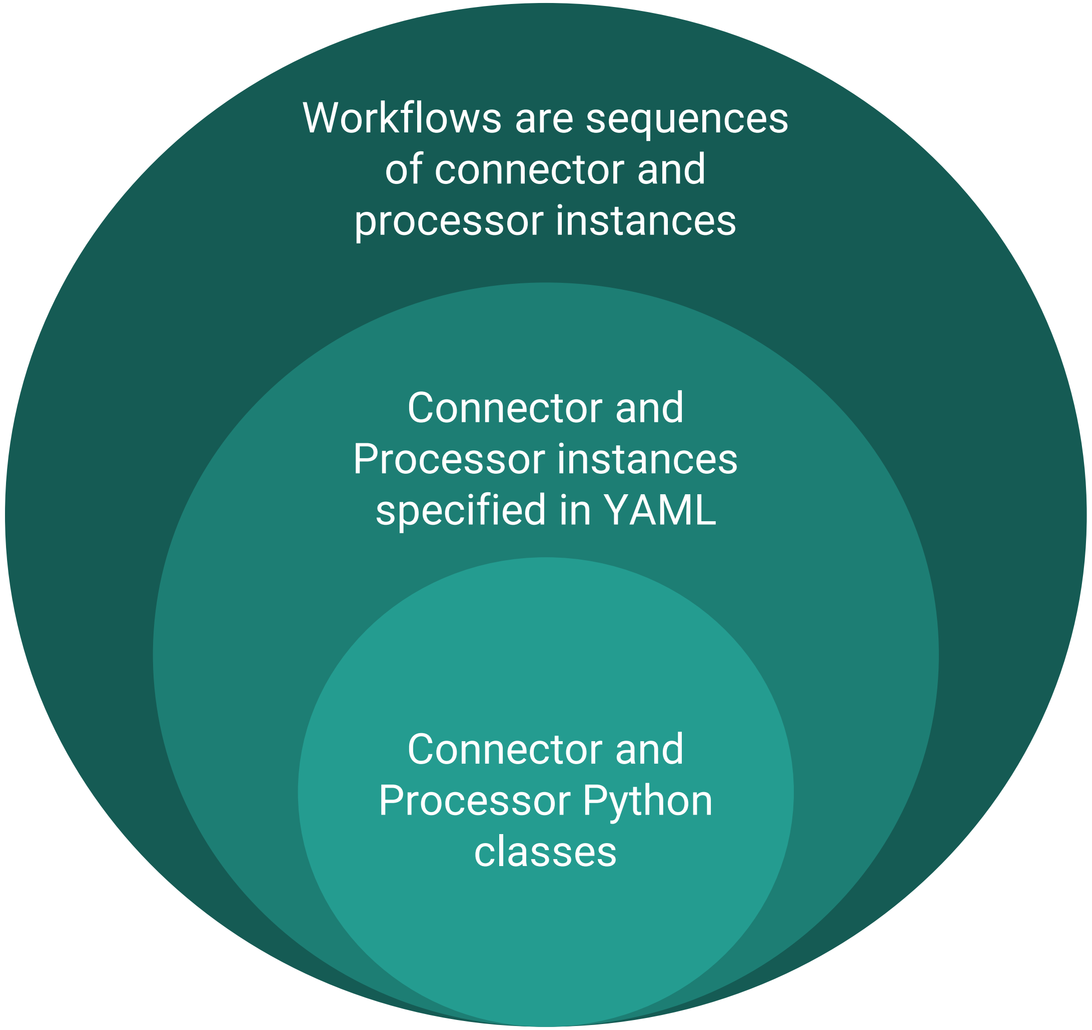
Data Processor Examples

Domain Independent	**Reproject, resample, format transformation, filter, mosaic, clip/mask, aggregate (spatial & temporal), visualization, reclassification**
Hydrology	**Terrain analysis, flood models**
Digital Ag	**Query, spatial/temporal filter, ML training, decision support**
Sustainability	**Downsample, (weighted) aggregate, FEWS models**

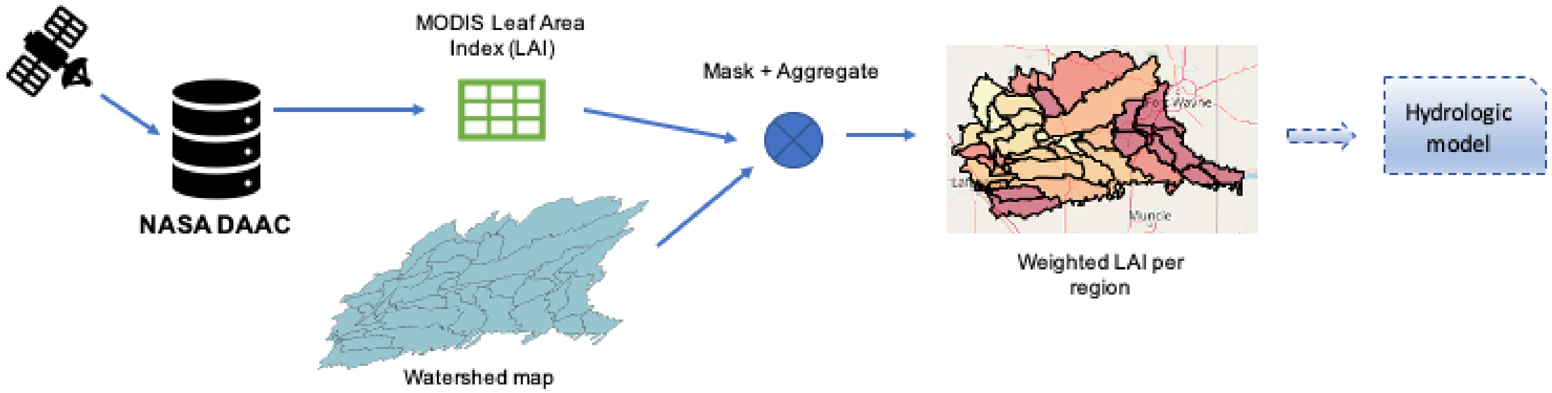
Plug-and-play Workflow Composer

- **Workflow Framework** defining
  - Standardized interfaces for connectors and processors
  - Syntax and semantics of defining and composing instances of connectors and processors into scientific workflows
- **Workflow Engine** transforming
  - "Declarative", abstract workflows into code executing on heterogeneous compute resources

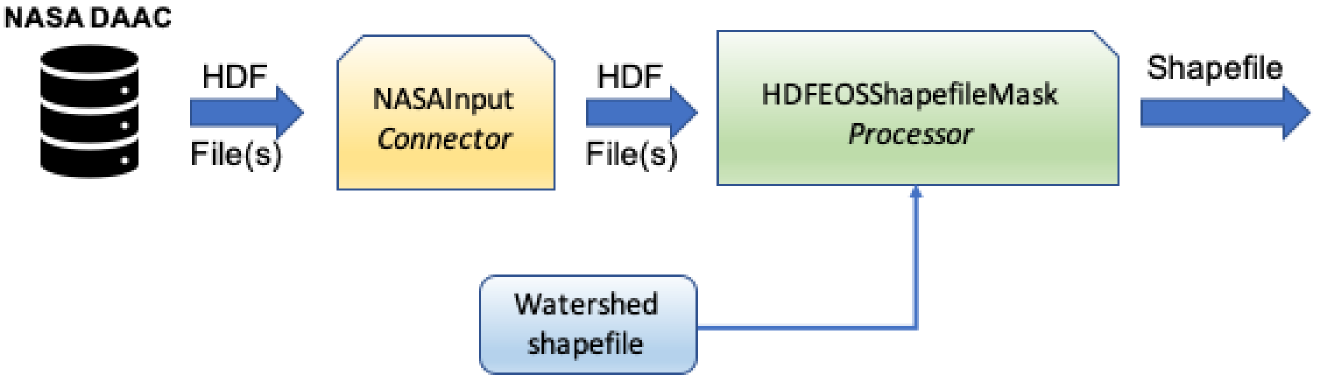
GeoEDF in a nutshell



Example Hydrologic Workflow



Apply GeoEDF principles



The GeoEDF Workflow

Data connector

Dynamically bound variable

Python class

```
$1:
  Input:
    NASAInput:
      url: http://files.ntsg.umt.edu/data/NTSG_Products/MOD16/MOD16A2.105_MERRAGMAO/{file}
      user: rkalyana
      password:
  Filter:
    file:
      PathFilter:
        pattern: 'Y%{year}/D001/*.h00v08*.hdf'
    year:
      DateTimeFilter:
        pattern: '%Y'
        start: 01/01/2000
        end: 12/31/2005
        period: 1Y
```

- Filters provide bindings for variables
- They promote modularization and can implement complex spatial and temporal filtering
- Filters help restrict the data that is actually "downloaded"

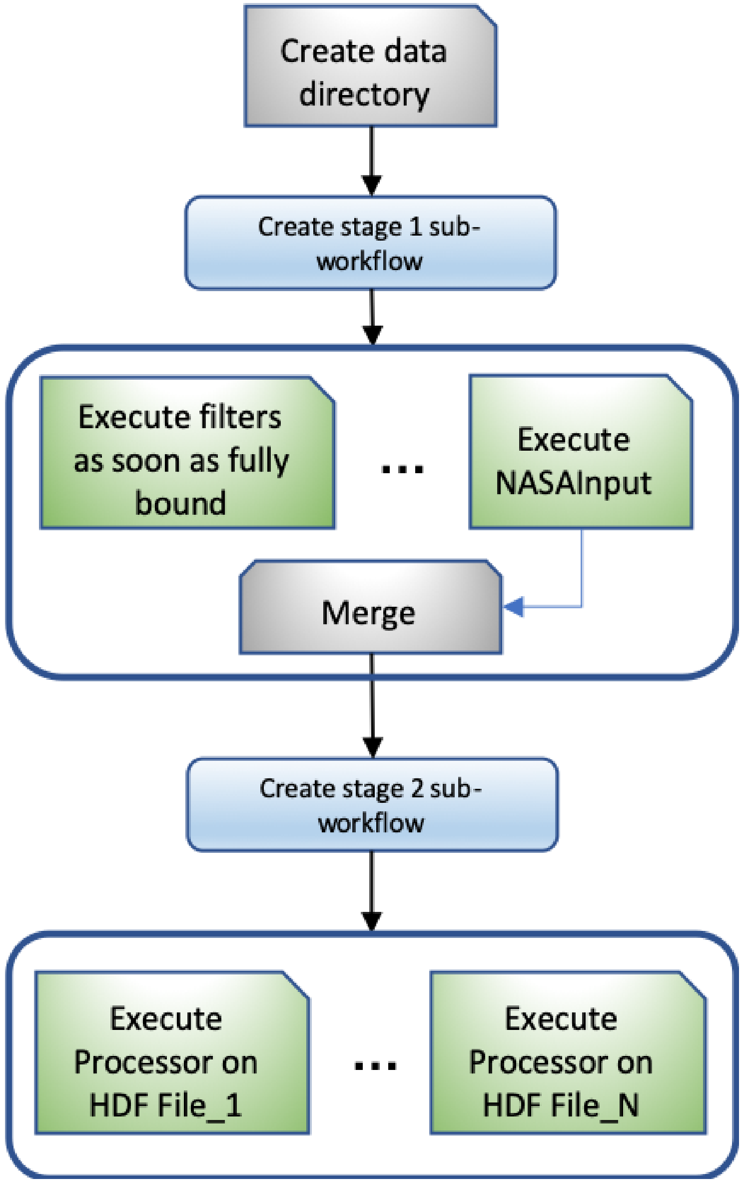
Workflow stage

Data processor

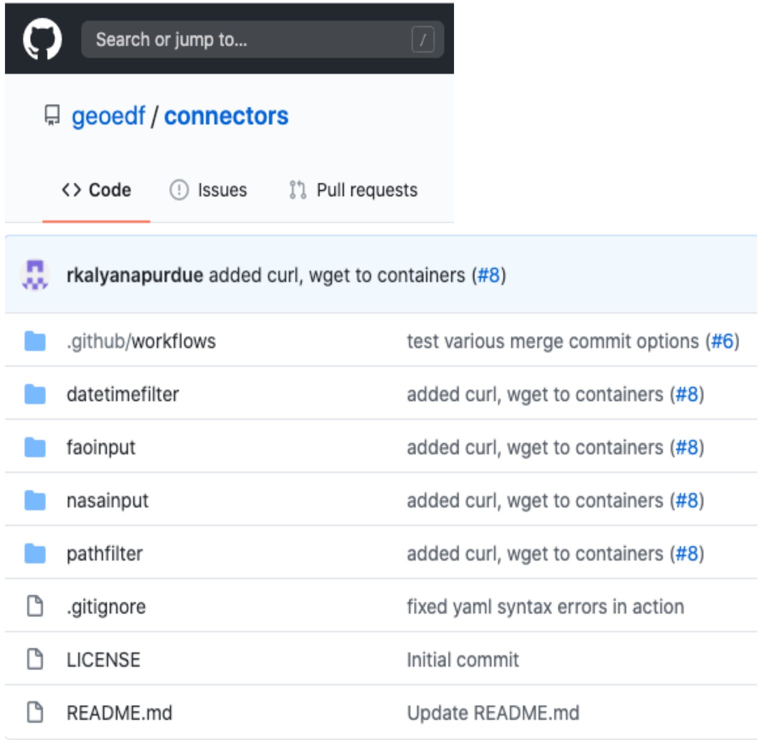
```
$2:
  HDFShapefileEOSMask:
    hdffile: $1
    shapefile: /home/rkalyana/subs1.shp
```

Workflow Concretization using Pegasus

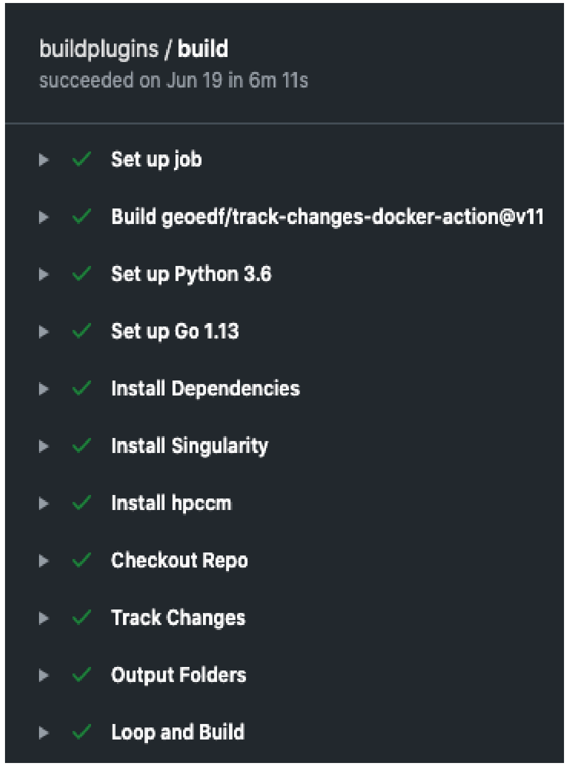
- Connectors need to bind filter variables in order; arbitrary number of variable bindings may be generated; each binding “retrieves” arbitrary number of files
- Processors may need to process an arbitrary number of files retrieved by a connector
- Each connector or processor turns into its own “sub-workflow”
- Top-level DAX builds and executes these sub-workflows as it goes
- Sub-workflows only transfer back data necessary to construct the next ”stage” sub-workflow; viz., filter values, file listing
- Final step returns outputs
- *Connectors/processors can have arbitrary software dependencies *(containerization is a good idea!)*
- **Public-private keypair generated for each workflow to encrypt sensitive strings *(viz. any field left blank for user input in workflow definition)*



Connector/Processor Contribution Process



(1) Contribute connectors/processors via GitHub PRs

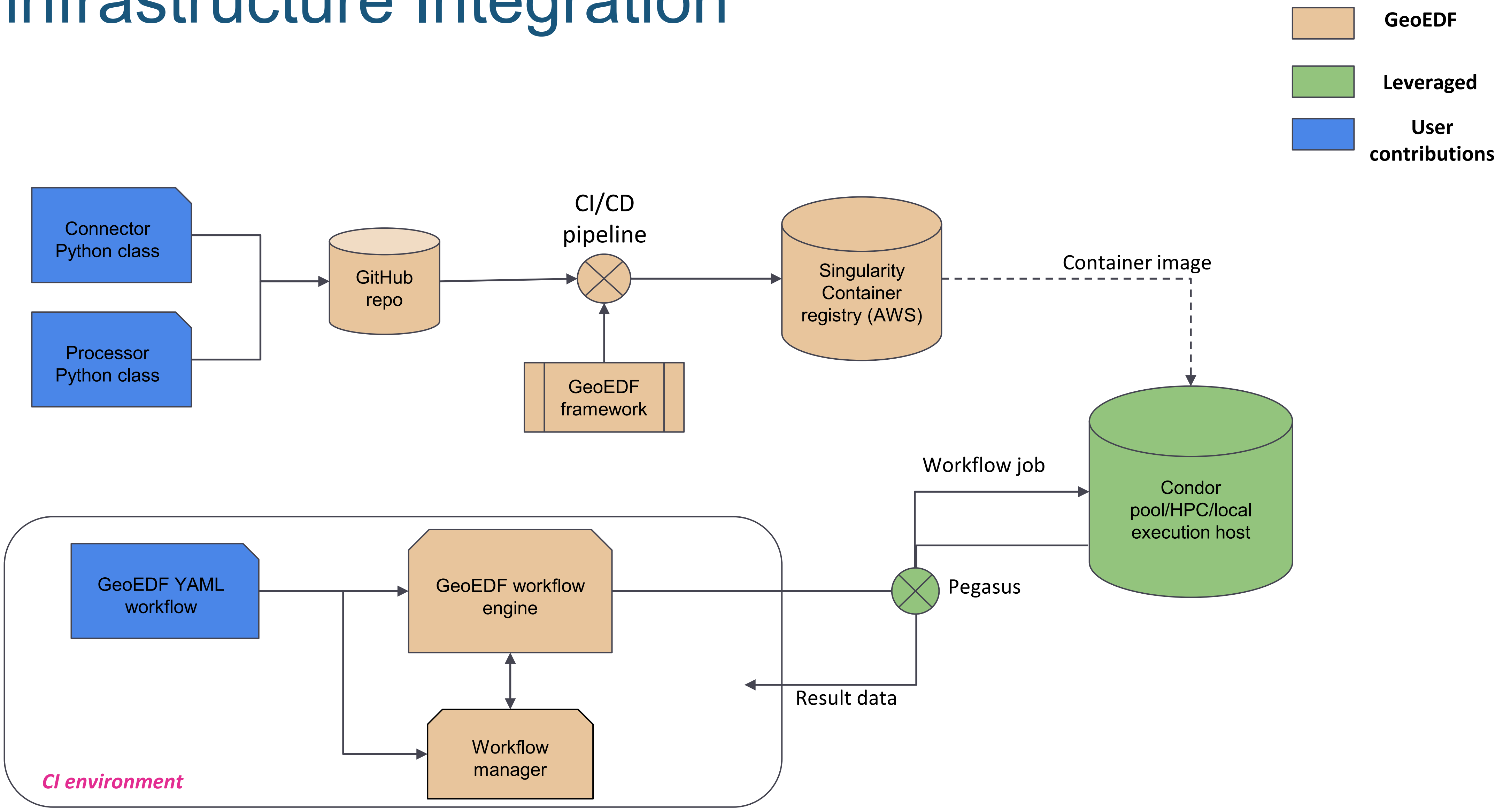


(2) Detect changes, build Singularity container, push to registry server

```python
def get_registry_containers(self):
    cli = get_client(quiet=True)

    conns = dict()
    query_res = cli.search("connectors")
    for (cont_uri,url) in query_res:
        cont_path = cont_uri.split(':')[0]
        plugin_name = cont_path.split('/')[1]
        if plugin_name not in conns:
            conns[plugin_name] = cont_uri

    procs = dict()
    query_res = cli.search("processors")
    for (cont_uri,url) in query_res:
        cont_path = cont_uri.split(':')[0]
        plugin_name = cont_path.split('/')[1]
        if plugin_name not in procs:
            procs[plugin_name] = cont_uri

    return (conns,procs)
```

(3) Query registry for list of connector, processor containers

Cyberinfrastructure Integration



Roadmap

Dec 2020
- Sample connectors, processors
- CI/CD pipeline in place
- Proof-of-concept Jupyter notebook based on Pegasus dev container

Feb 2021
- Execution on HPC
- Integration with MyGeoHub gateway

Mar 2021
- Release v1.0 available on MyGeoHub, PyPI, GitHub
- Sample end-to-end research workflows
- Documentation

Summer 2021
- Workflow monitoring
- Data annotation hooks
- CI interoperability and external CI install

Our Pegasus Feedback

PUG 2021

Cyberinfrastructure Integration

- Best practices for setting up Pegasus to support (a) multiple users, (b) secure sensitive information (e.g., catalogs, keys)
- Middleware layer with a thin API interface?

New Features

- Support for conditionals, loop-until?
- High-level monitoring, i.e., what task in what sub-workflow is currently executing?

Thank You!

Where to find us:

- Project Repository: https://github.com/geoedf
- MyGeoHub CI: https://mygeohub.org

- Email: Carol Song [cxsong@purdue.edu],
  Rajesh Kalyanam [rkalyana@purdue.edu]