DARE http://project-dare.eu

Koninklijk Nederlands
Meteorologisch Instituut
Ministerie van Infrastructuur en Milieu

DARE: Integrating solutions for Data-Intensive and Reproducible Science

Alessandro Spinuso
and the DARE team

spinuso@knmi.nl
R&D Data Technology and Observations

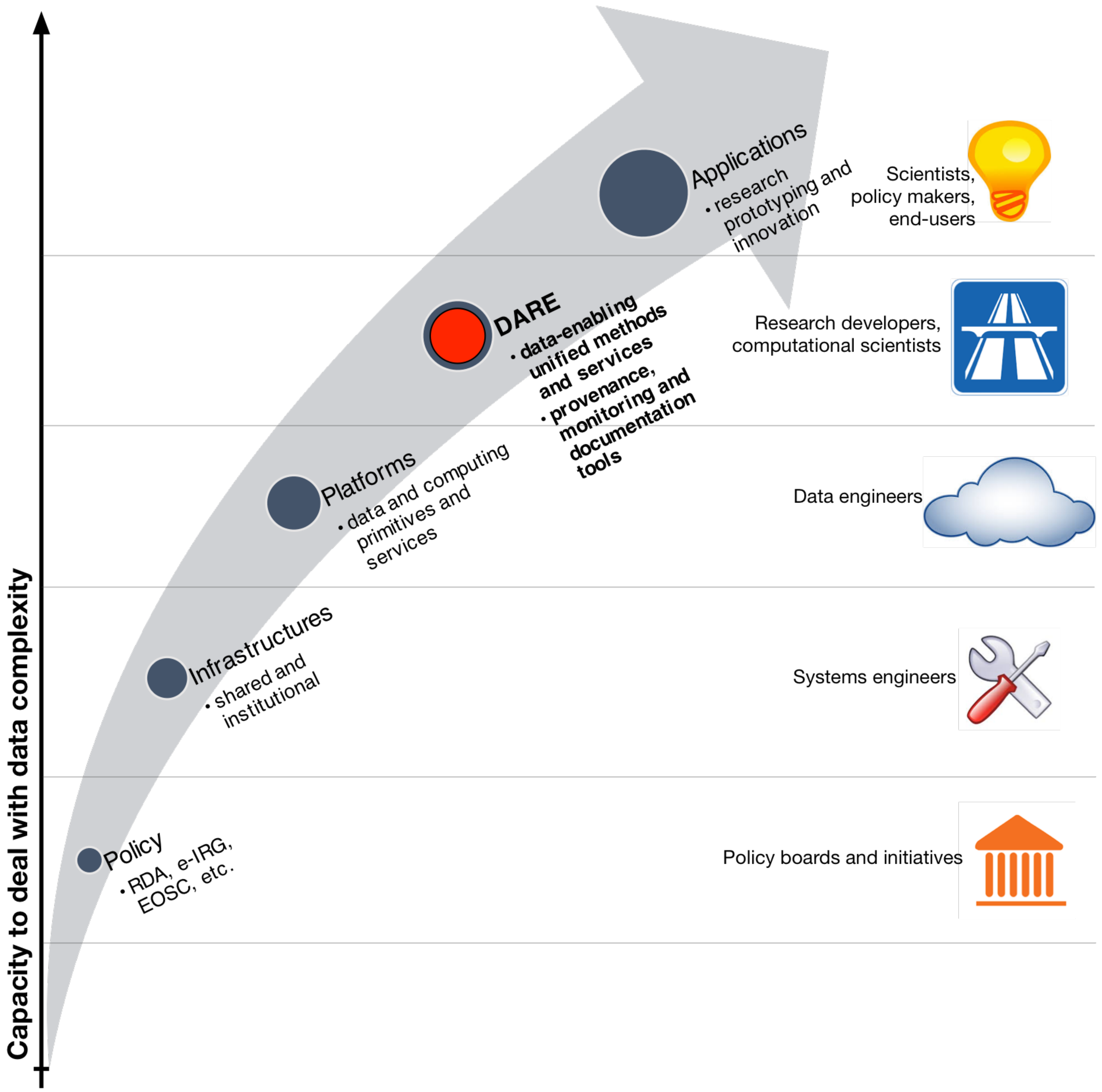
DEMOKRITOS · INGV · THE UNIVERSITY of EDINBURGH · CERFACS · KIT Karlsruher Institut für Technologie · Koninklijk Nederlands Meteorologisch Instituut · grnet · Fraunhofer · ATHENA Research & Innovation Information Technologies

ECMWF #repwork19

What’s in this talk…

- **DARE Objectives and Provenance Challenges**
- ***Active* provenance for Data-Intensive Workflows**
  (Use Case in Seismology)
- **Provenance-aware Workspaces**
- **Conclusions**

DARE
Delivering Agile Research Excellence on European e-Infrastructures


Capacity to deal with data complexity
Applications
• research prototyping and innovation
Scientists, policy makers, end-users
DARE
• data-enabling unified methods and services
• provenance, monitoring and documentation tools
Research developers, computational scientists
Platforms
• data and computing primitives and services
Data engineers
Infrastructures
• shared and institutional
Systems engineers
Policy
• RDA, e-IRG, EOSC, etc.
Policy boards and initiatives

Working environment for professionals wrestling with challenges involving complexity of methods and data

*System & Data Engineers => **Research Developers** => **Domain Expert***

- **Mapping between abstract methods and concrete** applications executed by different enactments seamlessly

- **Validation and Traceability of runs and products:** diagnose, monitor, reuse

- **Organisation of campaigns** reusing data and methods from **multiple runs**

- Driven and Evaluated by communities

EPOS EUROPEANPLATEOBSERVINGSYSTEM

is-enes INFRASTRUCTURE FOR THE EUROPEAN NETWORK FOR EARTH SYSTEM MODELLING

Accelerate productivity of expert teams.

Data Lineage

Lineage is DATA (Retrospective Provenance)

- **Data's origins,** what happens to it and where it **moves over time**
- **It may include technical metadata:** quality test results, reference values
- **Ability to trace errors** back to the root cause.
- **Integrated** in **workflow systems** to trace the data via various changes
- **Its volume depends on its scope!**

Data Lineage

Lineage is DATA (Retrospective Provenance)

- **Data's origins,** what happens to it and where it **moves over time**
- **It may include technical metadata:** quality test results, reference values
- **Ability to trace errors** back to the root cause.
- **Integrated** in **workflow systems** to trace the data via various changes
- **Its volume depends on its scope!**

Challenges in Data-Intensive workflows:

Granularity: provenance information can be **too coarse or too detailed**.
Intermediate Data can be **Materialised**, as well as **Volatile** (streaming).

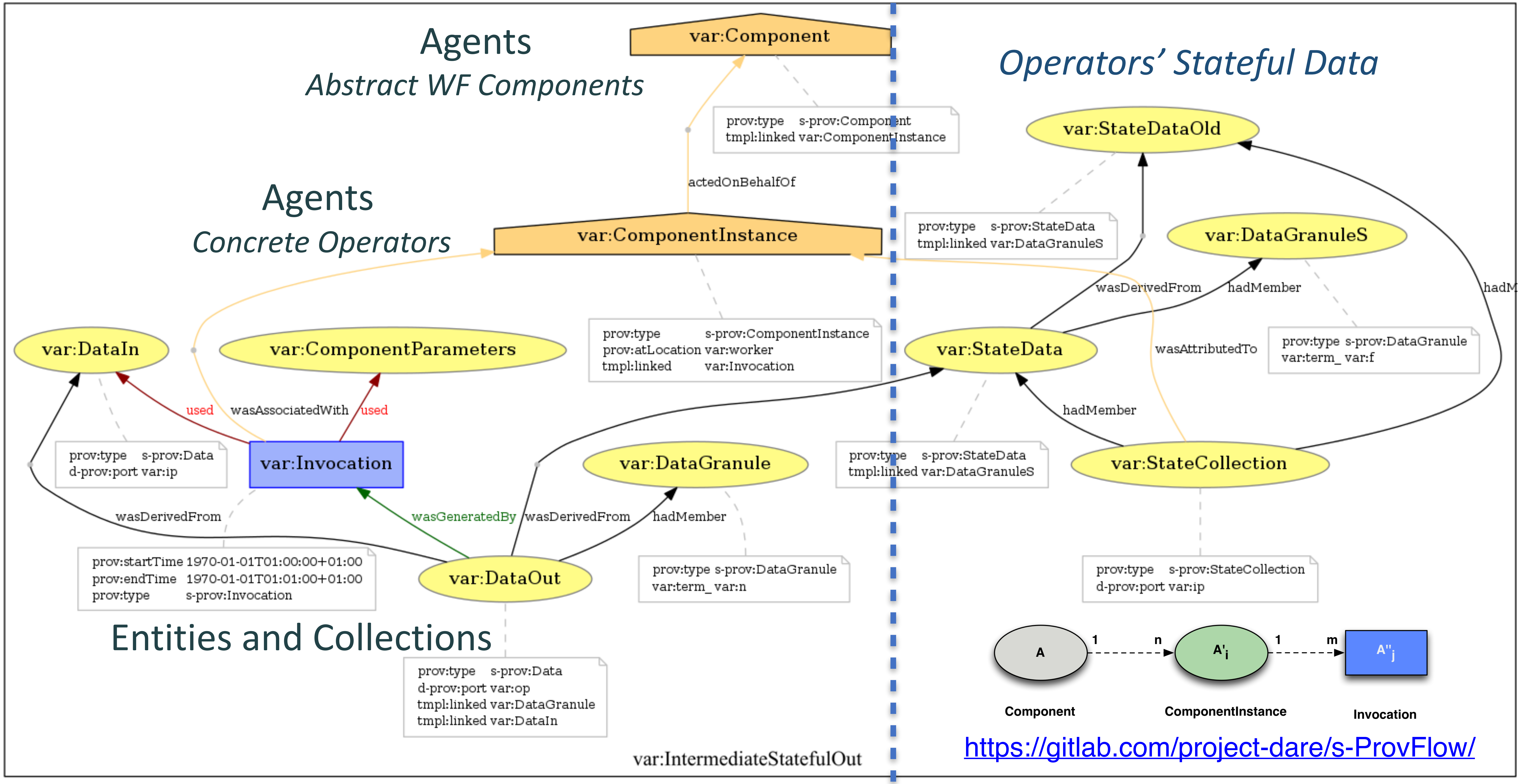
Precision: detailed capturing of data derivations in **parallel operators**
(what input data contributed to what output).

Relevance: how to manage **domain and application specific properties**?

Reuse: lack in validation and understanding of the computational method =>
ineffective reuse of results. **(Reproduce vs Reuse battle !?)**

S-PROV Lineage Model for Stateful Operators

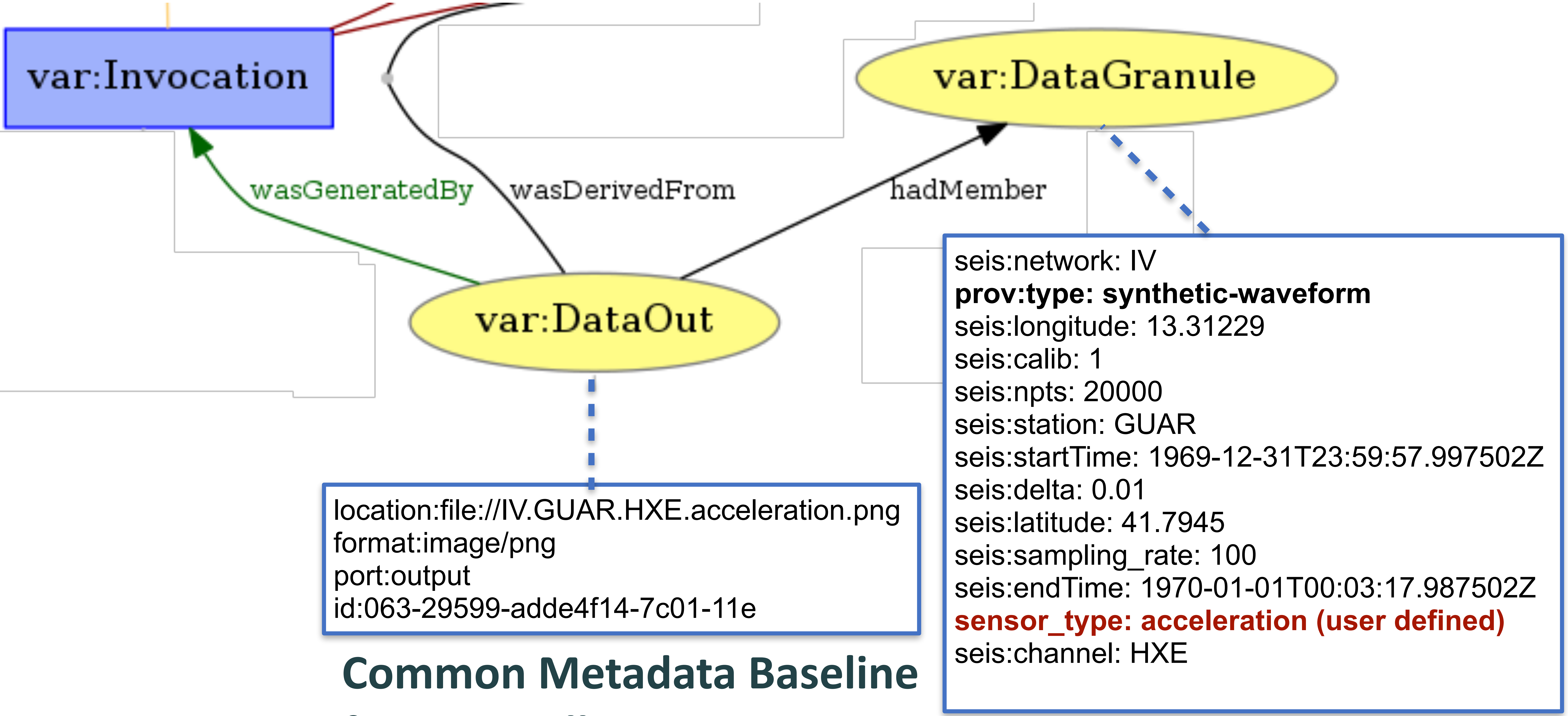
built on ProvONE and W3C PROV
https://purl.dataone.org/provone-v1-dev



https://gitlab.com/project-dare/s-ProvFlow/

Spinuso, A, Atkinson, M & Magnoni, *Active provenance for Data-Intensive workflows: engaging users and developers.*
IEEE eScience 2019 proceedings. Bridging from Concepts to Data and Computation for eScience (BC2DC'19) Workshop

Contextual Metadata
Data Collections and Granules



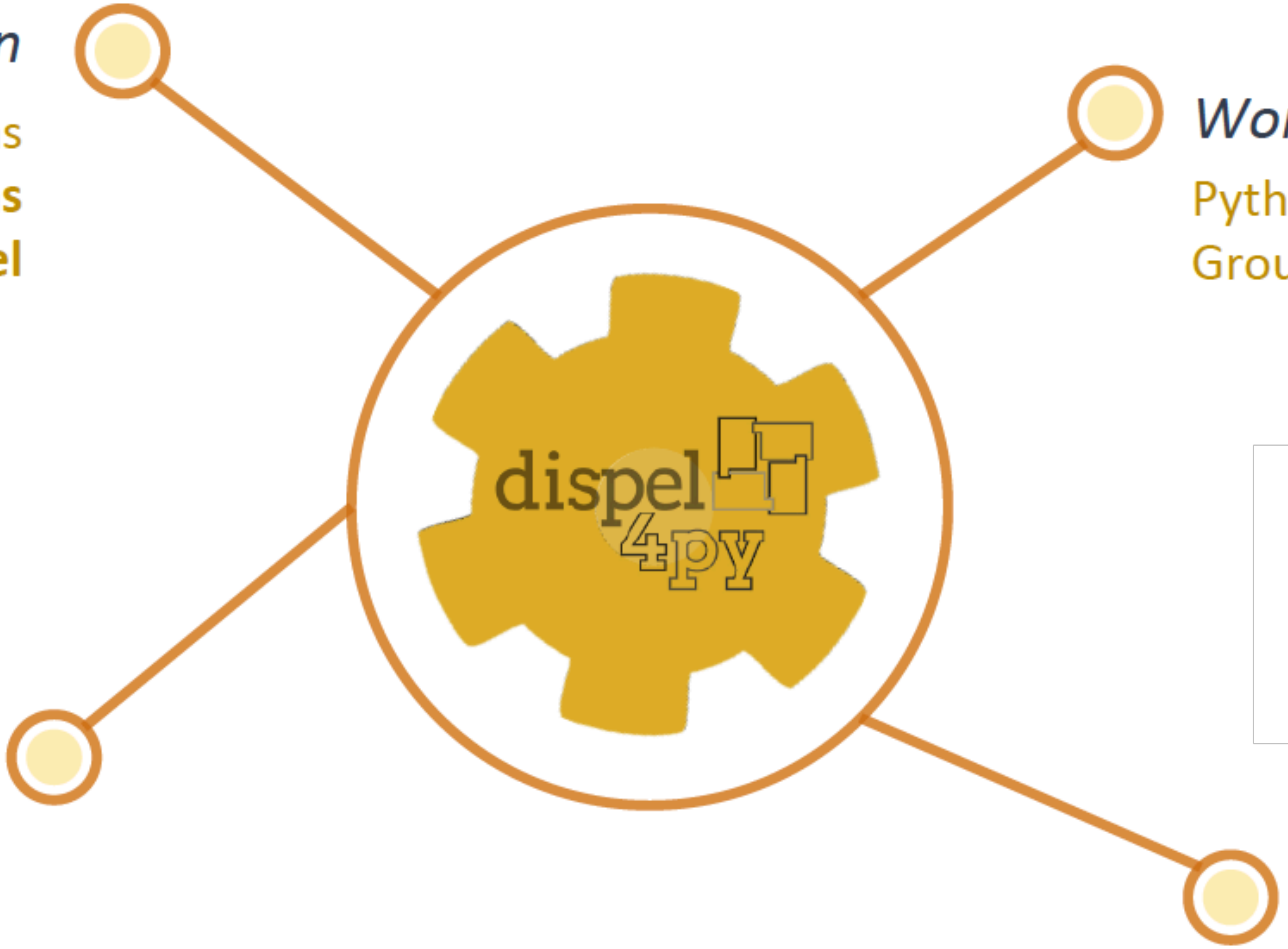
location:file://IV.GUAR.HXE.acceleration.png
format:image/png
port:output
id:063-29599-adde4f14-7c01-11e

seis:network: IV
prov:type: synthetic-waveform
seis:longitude: 13.31229
seis:calib: 1
seis:npts: 20000
seis:station: GUAR
seis:startTime: 1969-12-31T23:59:57.997502Z
seis:delta: 0.01
seis:latitude: 41.7945
seis:sampling_rate: 100
seis:endTime: 1970-01-01T00:03:17.987502Z
sensor_type: acceleration (user defined)
seis:channel: HXE

Common Metadata Baseline for Data collections

Domain properties User’s Context
(e.g. Seismology)

dispel4py: Data-Intensive processing



Automation

Automates pipeline executions

Concurrent, distributed computations

Stream-based model

Workflow Composition

Python Library

Groupings

Mapping

Sequential

Multiprocessing

MPI

Apache Storm

Apache Spark (Prototype)

Optimisation

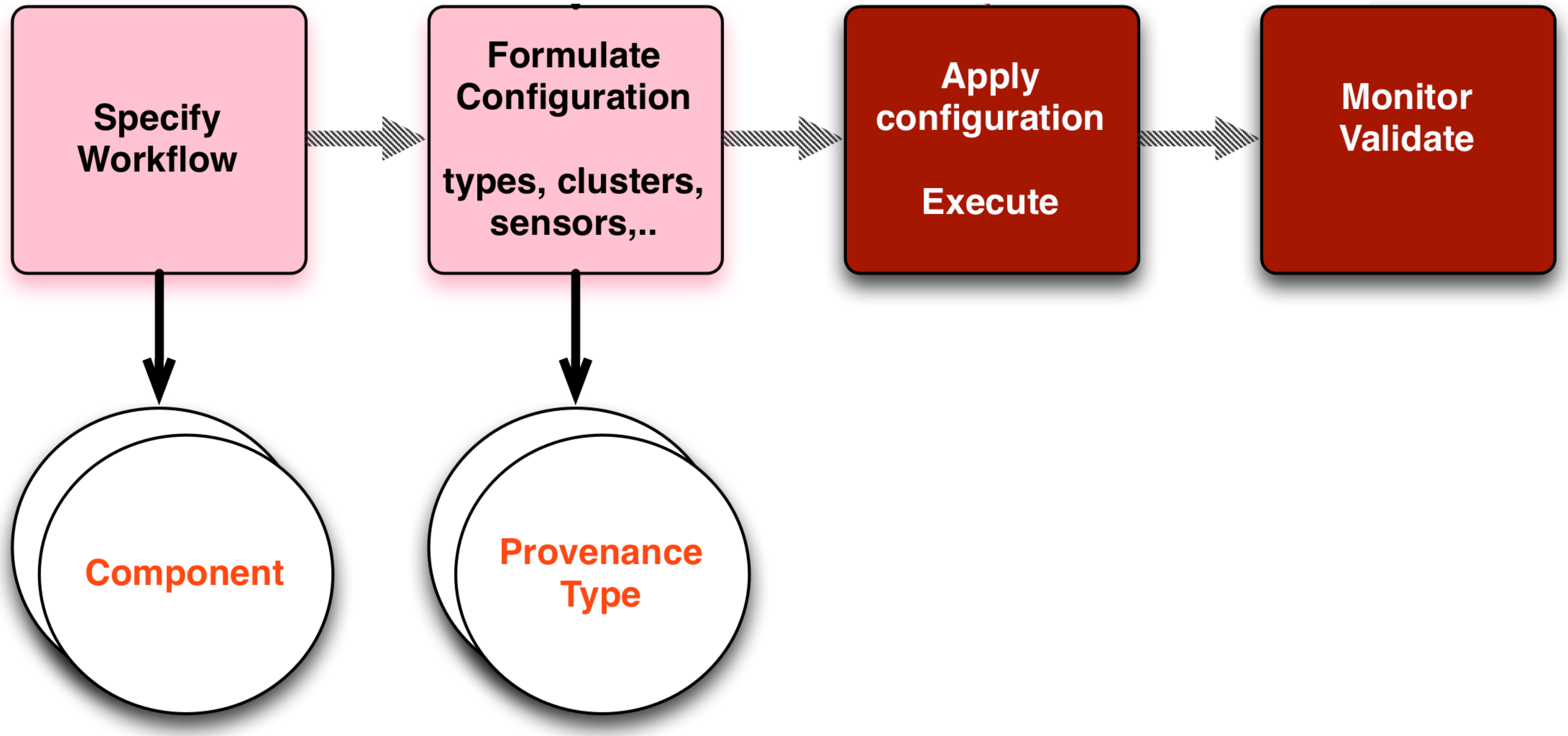
Multiple streams

Avoids I/O operations

Key-features: Automatic parallelisation/mappings, concurrent & stream-based, configurable provenance

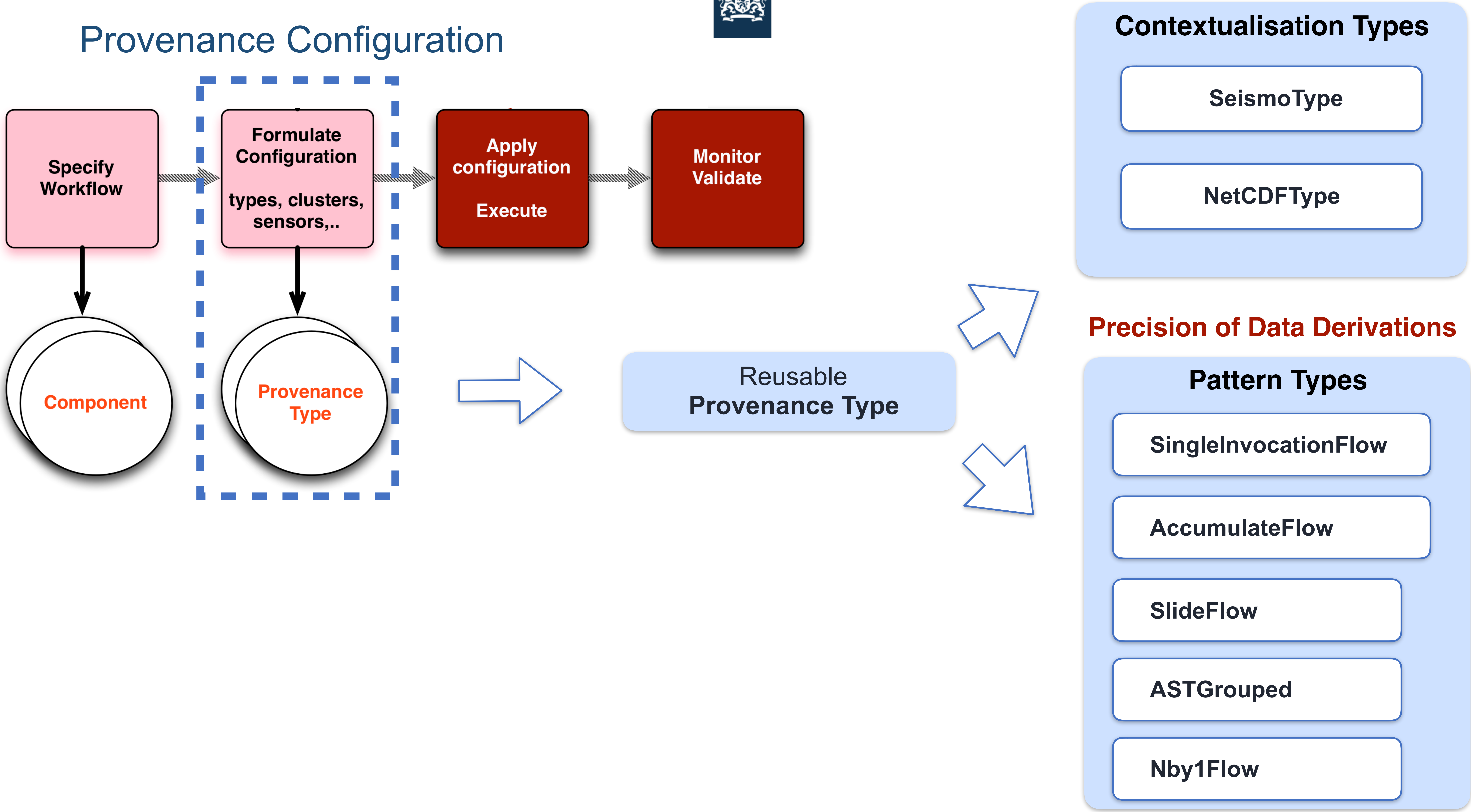
https://gitlab.com/project-dare/dispel4py

Active Provenance Capturing



Active Provenance Capturing

Provenance Configuration



Specify Workflow

Formulate Configuration

types, clusters, sensors,..

Apply configuration

Execute

Monitor Validate

Component

Provenance Type

Reusable **Provenance Type**

Domain Metadata

Contextualisation Types

- SeismoType
- NetCDFType

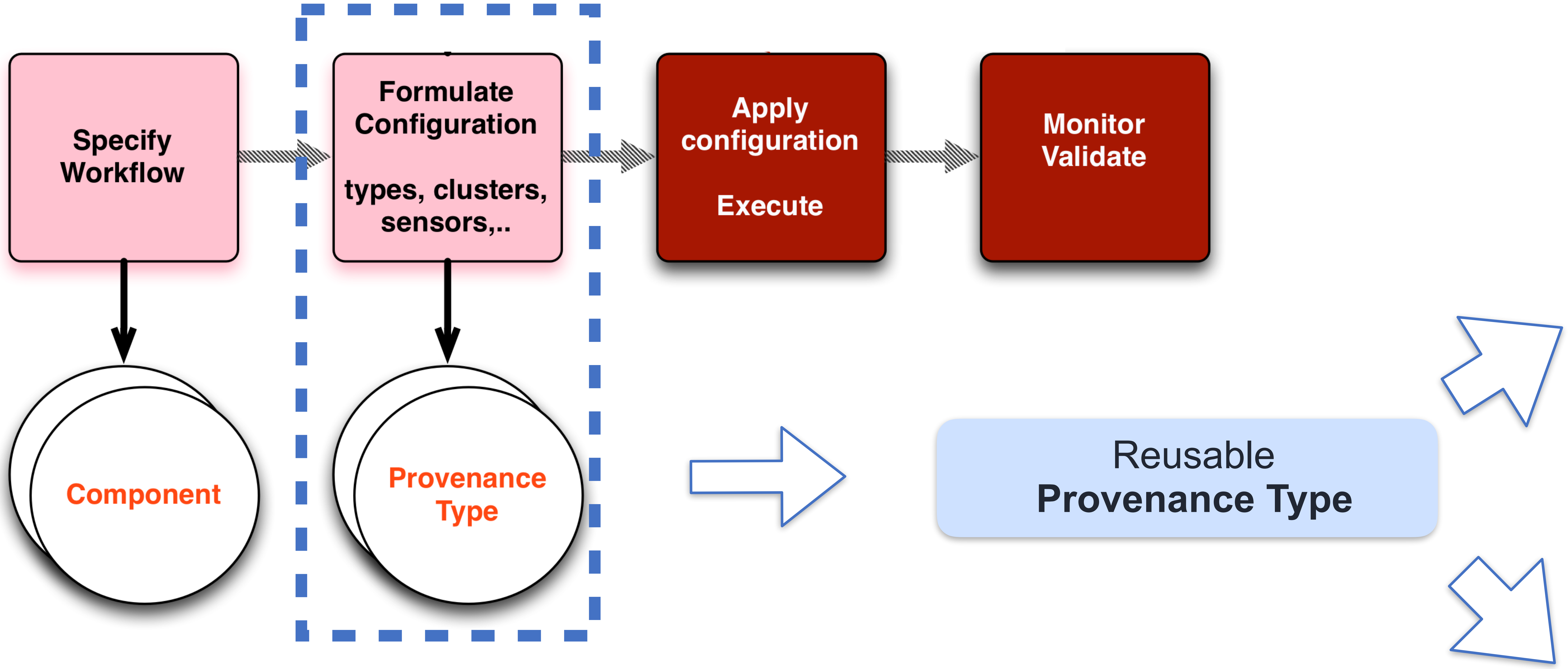
Precision of Data Derivations

Pattern Types

- SingleInvocationFlow
- AccumulateFlow
- SlideFlow
- ASTGrouped
- Nby1Flow

Active Provenance Capturing

Provenance Configuration



Domain Metadata

Contextualisation Types

- SeismoType
- NetCDFType

Precision of Data Derivations

Pattern Types

- SingleInvocationFlow
- AccumulateFlow
- SlideFlow
- ASTGrouped
- Nby1Flow

Research Developers

- **Develop libs of ProvenanceTypes for Contextualisation and Precision**

Scientists (Workflow Users)

- **Combine and Assign ProvenanceTypes to WF Functions**
- **Enrich descriptions with semantic tags**

System Managers

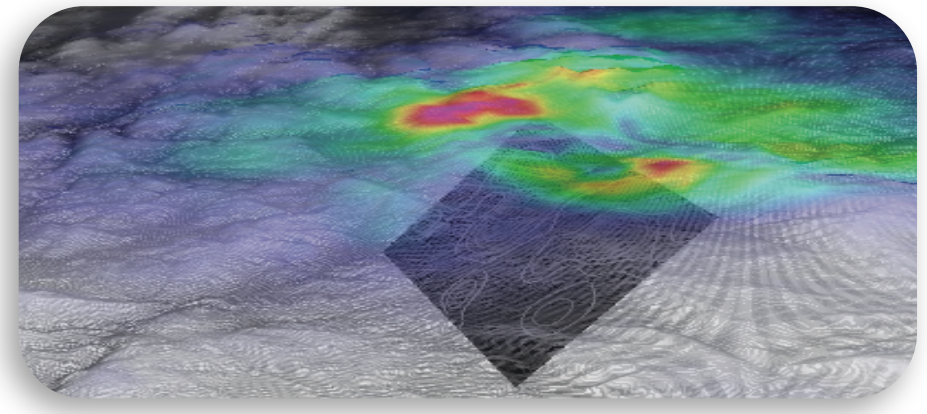
- **Selective Lineage activation to** narrow the focus of the lineage (metadata values-range)
- **Tune the impact of provenance** on the infrastructure (real-time systems)

Test Case:
Seismic Rapid Assessment

DARE

Rapid Ground Motion Assessment (RA)

Reusable Heterogeneous Tasks running at different scale.
May require human monitoring and intervention



Choose/upload **seismic wavespeed** & **mesh**

Choose/upload **seismic source** (point or fault)

run waveform simulation

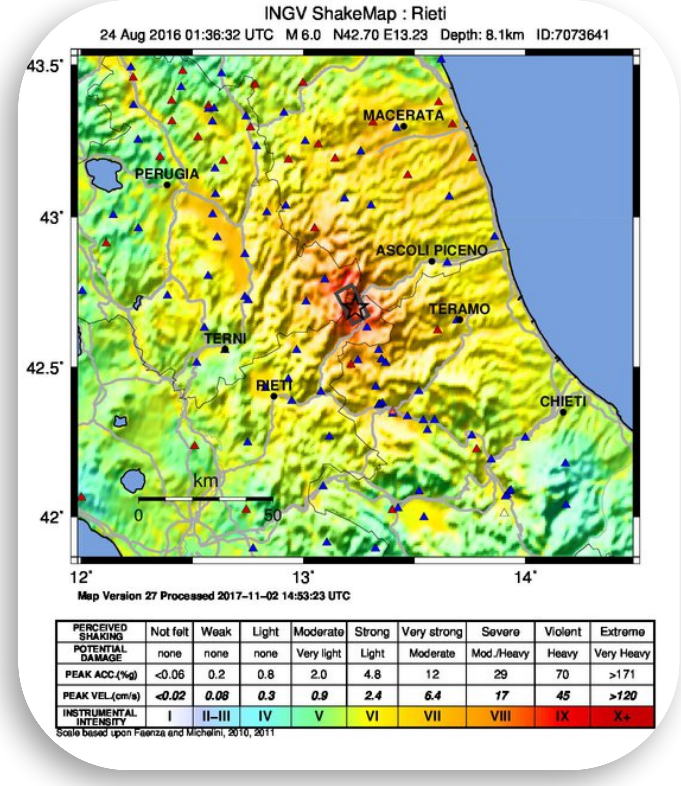
Gather **observed data**

Waveform Preprocessing

Ground Motion Parameters

Compare/integrate synthetic and observed ground motion data

Store data, metadata, provenance


INGV ShakeMap : Rieti
24 Aug 2016 01:36:32 UTC M 6.0 N42.70 E13.23 Depth: 8.1km ID:7073641
43.5˚
43˚
42.5˚
42˚
12˚
13˚
14˚
MACERATA
PERUGIA
ASCOLI PICENO
TERAMO
TERNI
RIETI
CHIETI
km
Map Version 27 Processed 2017-11-02 14:53:23 UTC

Test Case:
Seismic Rapid Assessment

DARE

Rapid Ground Motion Assessment (RA)

Reusable Heterogeneous Tasks running at different scale.
May require human monitoring and intervention

MPI Simulation

Choose/upload **seismic wavespeed** & **mesh**

Choose/upload **seismic source** (point or fault)

run waveform simulation

Gather **observed data**

Waveform Preprocessing

Ground Motion Parameters

Compare/integrate synthetic and observed ground motion data

Store data, metadata, provenance

Data Analysis

Data Staging

INGV ShakeMap : Rieti
24 Aug 2016 01:36:32 UTC M 6.0 N42.70 E13.23 Depth: 8.1km ID:7073641

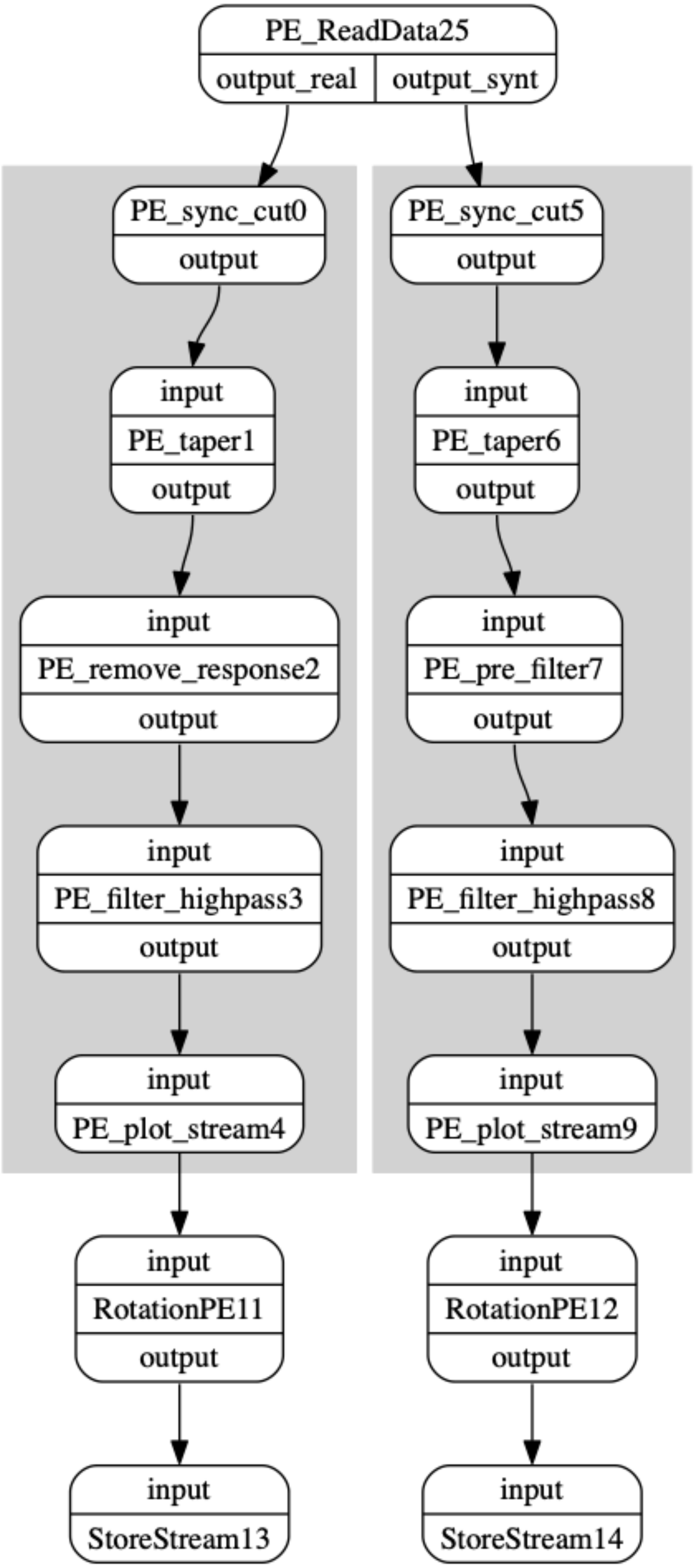
Map Version 27 Processed 2017–11–02 14:53:23 UTC

PERCEIVED SHAKING	Not felt	Weak	Light	Moderate	Strong	Very strong	Severe	Violent	Extreme
POTENTIAL DAMAGE	none	none	none	Very light	Light	Moderate	Mod./Heavy	Heavy	Very Heavy
PEAK ACC.(%g)	<0.06	0.2	0.8	2.0	4.8	12	29	70	>171
PEAK VEL.(cm/s)	*<0.02*	*0.08*	*0.3*	*0.9*	*2.4*	*6.4*	*17*	*45*	*>120*
INSTRUMENTAL INTENSITY	I	II–III	IV	V	VI	VII	VIII	IX	X+

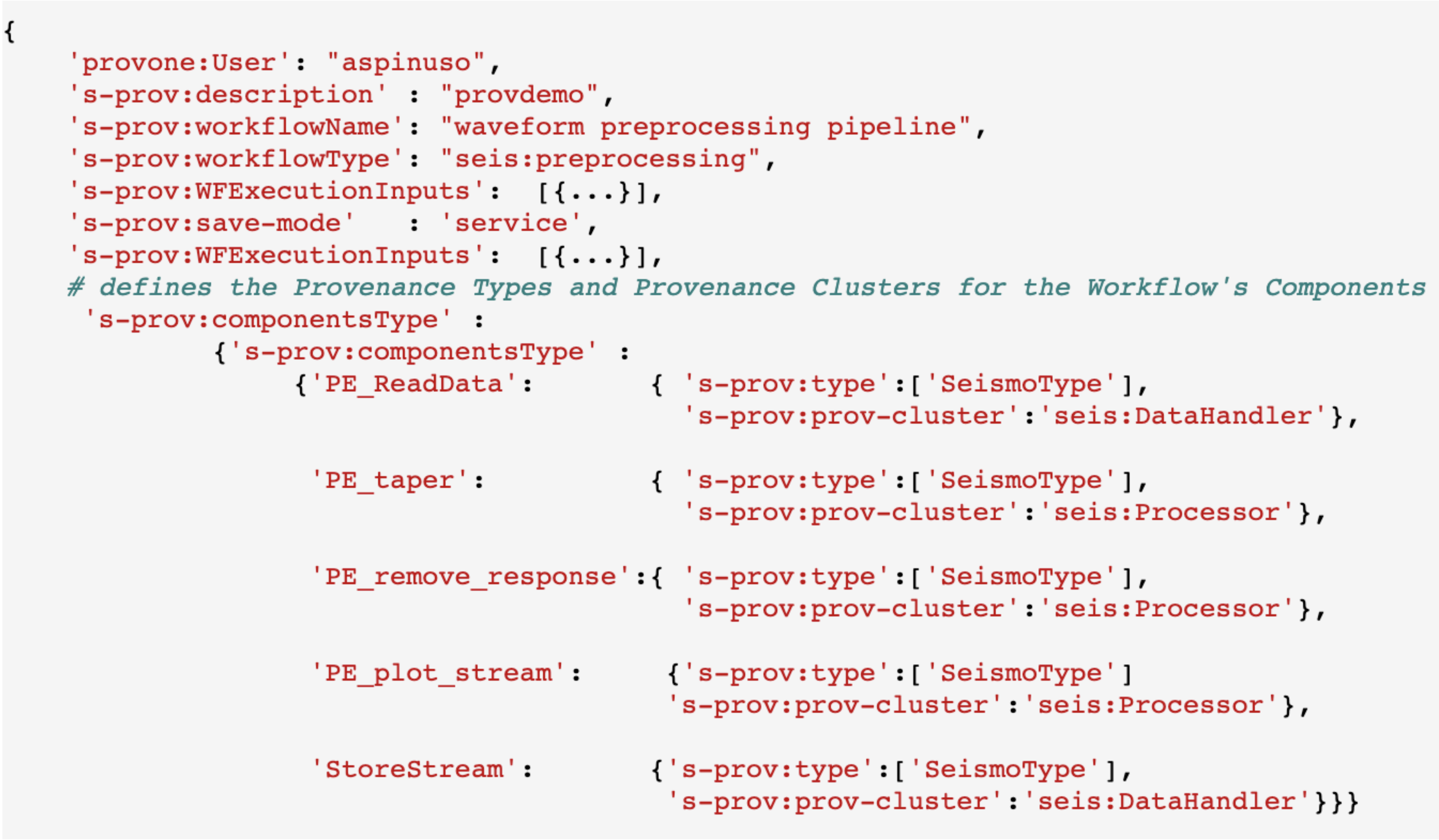
Scale based upon Faenza and Michelini, 2010, 2011

Configuration - Contextualisation Types

Waveform Preprocessing



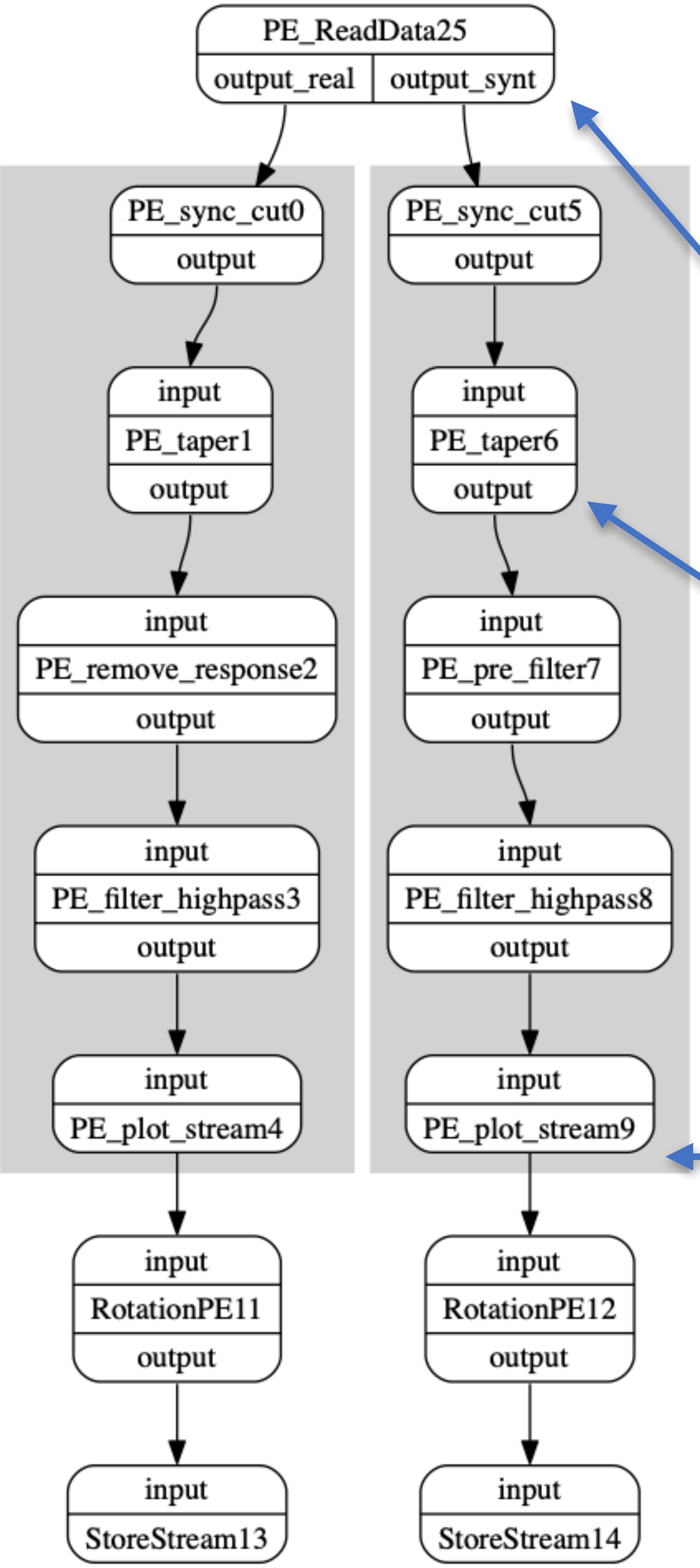
Configuration Profile in JSON with Provenance Types



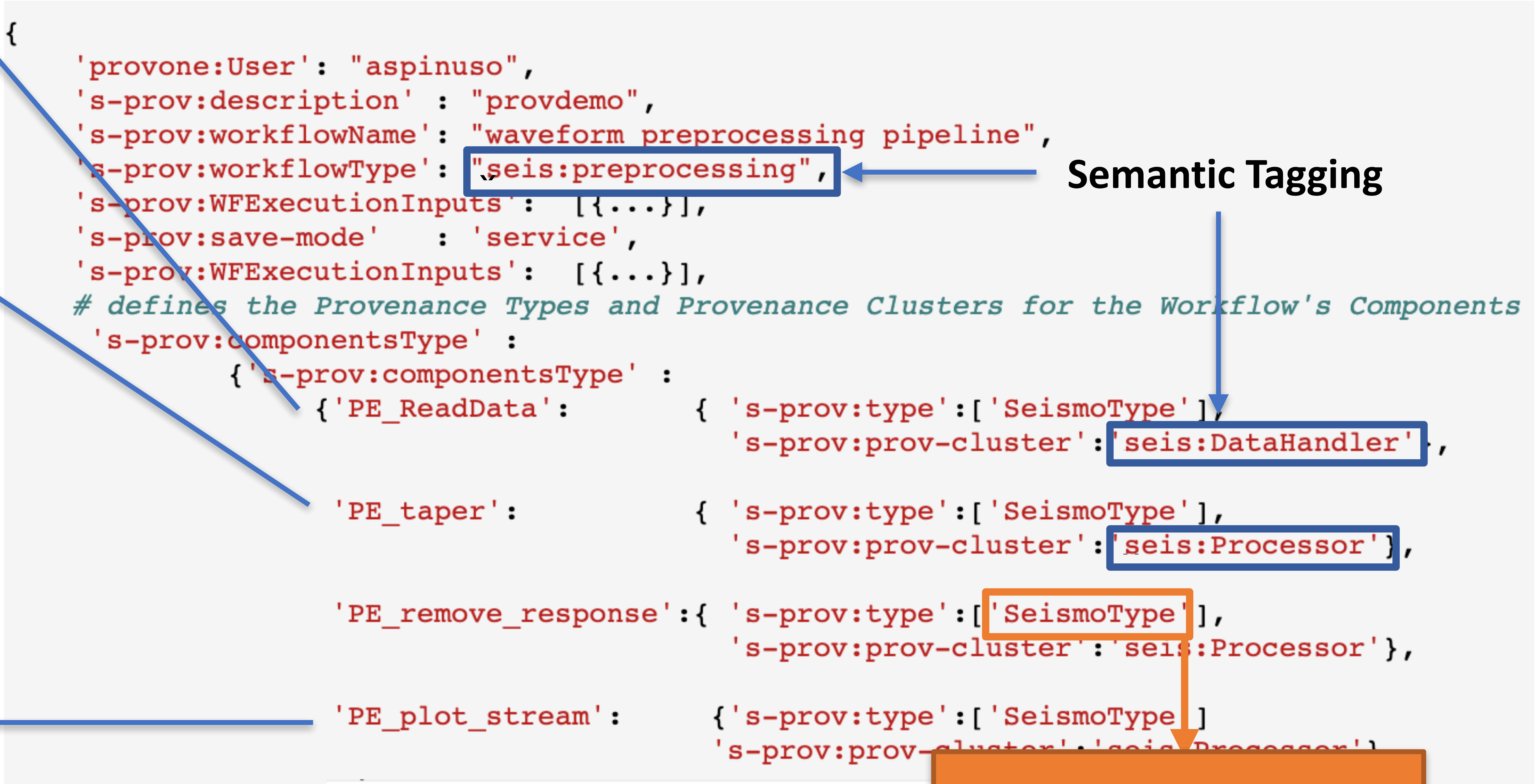
```
{
    'provone:User': "aspinuso",
    's-prov:description' : "provdemo",
    's-prov:workflowName': "waveform preprocessing pipeline",
    's-prov:workflowType': "seis:preprocessing",
    's-prov:WFExecutionInputs':  [{...}],
    's-prov:save-mode'   : 'service',
    's-prov:WFExecutionInputs':  [{...}],
    # defines the Provenance Types and Provenance Clusters for the Workflow's Components
     's-prov:componentsType' :
             {'s-prov:componentsType' :
                  {'PE_ReadData':       { 's-prov:type':['SeismoType'],
                                          's-prov:prov-cluster':'seis:DataHandler'},

                   'PE_taper':          { 's-prov:type':['SeismoType'],
                                          's-prov:prov-cluster':'seis:Processor'},

                   'PE_remove_response':{ 's-prov:type':['SeismoType'],
                                          's-prov:prov-cluster':'seis:Processor'},

                   'PE_plot_stream':     {'s-prov:type':['SeismoType']
                                         's-prov:prov-cluster':'seis:Processor'},

                   'StoreStream':       {'s-prov:type':['SeismoType'],
                                         's-prov:prov-cluster':'seis:DataHandler'}}}
```

Configuration - Contextualisation Types

Waveform Preprocessing

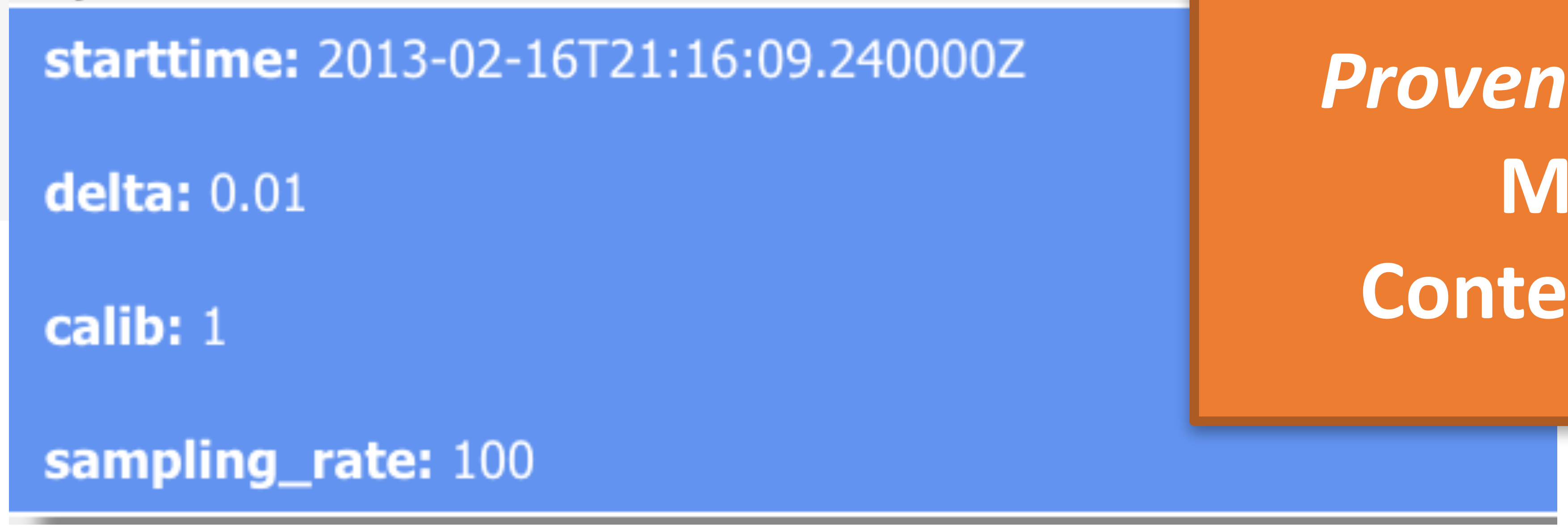


Configuration Profile in JSON with Provenance Types



```
{
    'provone:User': "aspinuso",
    's-prov:description' : "provdemo",
    's-prov:workflowName': "waveform preprocessing pipeline",
    's-prov:workflowType': "seis:preprocessing",
    's-prov:WFExecutionInputs':  [{...}],
    's-prov:save-mode'    : 'service',
    's-prov:WFExecutionInputs':  [{...}],
    # defines the Provenance Types and Provenance Clusters for the Workflow's Components
     's-prov:componentsType' :
            {'s-prov:componentsType' :
               {'PE_ReadData':        { 's-prov:type':['SeismoType'],
                                        's-prov:prov-cluster':'seis:DataHandler'},

                'PE_taper':           { 's-prov:type':['SeismoType'],
                                        's-prov:prov-cluster':'seis:Processor'},

                'PE_remove_response':{ 's-prov:type':['SeismoType'],
                                        's-prov:prov-cluster':'seis:Processor'},

                'PE_plot_stream':     {'s-prov:type':['SeismoType']
                                       's-prov:prov-cluster':'seis:Processor'}
                                                                              }}
```

Semantic Tagging



starttime: 2013-02-16T21:16:09.240000Z

delta: 0.01

calib: 1

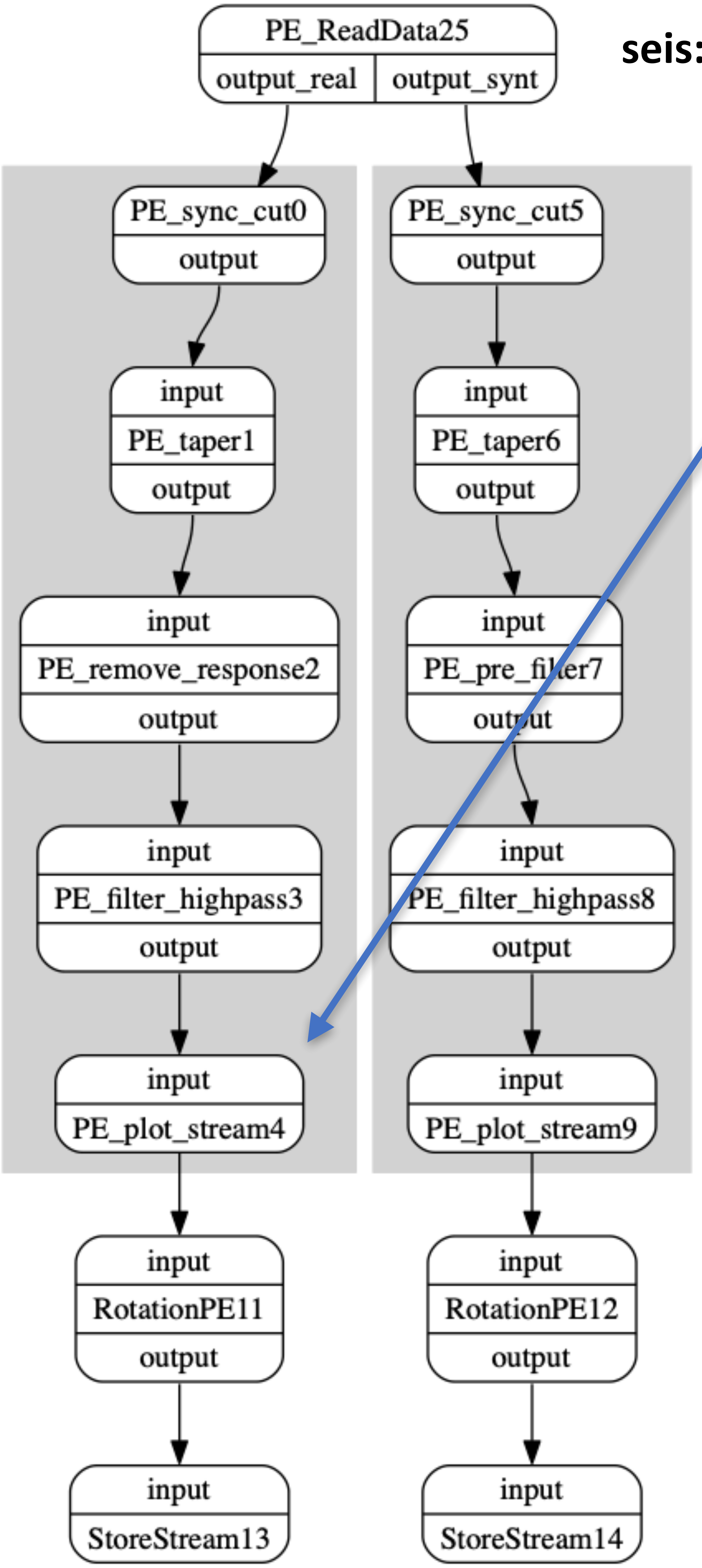
sampling_rate: 100

***ProvenanceType* for Metadata Contextualisation**

Inline metadata injection

seis:preprocessing

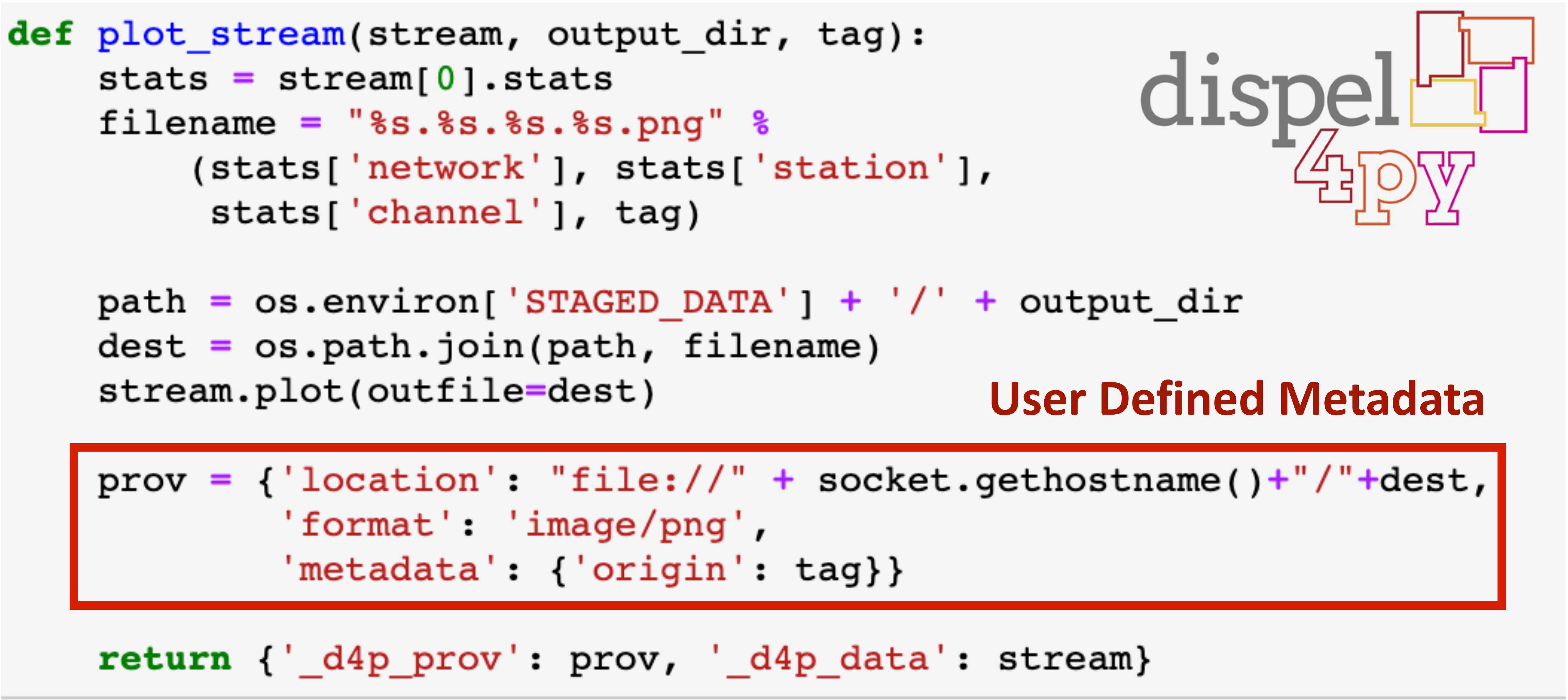
Waveform Preprocessing



pipeline JSON Description (eg. from file)

Manual Extensions

Functions encoded in Python
User Defined Metadata injection into Lineage traces



```python
def plot_stream(stream, output_dir, tag):
    stats = stream[0].stats
    filename = "%s.%s.%s.%s.png" %
         (stats['network'], stats['station'],
          stats['channel'], tag)

    path = os.environ['STAGED_DATA'] + '/' + output_dir
    dest = os.path.join(path, filename)
    stream.plot(outfile=dest)

    prov = {'location': "file://" + socket.gethostname()+"/"+dest,
            'format': 'image/png',
            'metadata': {'origin': tag}}

    return {'_d4p_prov': prov, '_d4p_data': stream}
```

User Defined Metadata

Monitor, search and analyse results through lineage

«
Open Run
Refresh View
View Inputs
Get W3C-PROV
iDrop
Radial
trace-bw
trace-fw
stateful
cross-run
file
Incomplete
wasDerivedFrom
PE_taper
PE_pre_f
PE_filte
PE_plot_
Search
Filter Current
Produce Download Script
SeismoType provenance capturing
Inline provenance capturing
User Defined Metadata
SeismoType Metadata

fmagnoni - Runtime Instances monitor - API-60f24878-fe46-4b2a-915f-e8...

	ID	Compon...	Last Ev...	Data co...	worker	Message	Changed
1	*StoreStr...*	StoreStr...	2019-09...	90	d4p-ope...		
2	Rotation...	Rotation...	2019-09...	30	d4p-ope...		
3	PE_filter...	PE_filter...	2019-09...	30	d4p-ope...		
4	PE_filter...	PE_filter...	2019-09...	30	d4p-ope...		
5	PE_pre_...	PE_pre_...	2019-09...	30	d4p-ope...		
6	PE_tape...	PE_taper6	2019-09...	30	d4p-ope...		
7	*PE_plot...*	PE_plot..	2019-09...	30	d4p-ope...		
8	*StoreStr...*	StoreStr...	2019-09...	89	d4p-ope...		
9	Rotation...	Rotation...	2019-09...	30	d4p-ope...		
10	PE_filter...	PE_filter...	2019-09...	30	d4p-ope...		
11	PE_filter...	PE_filter...	2019-09...	30	d4p-ope...		
12	PE_rem...	PE_rem...	2019-09...	30	d4p-ope...		
13	PE_tape...	PE_taper1	2019-09...	30	d4p-ope...		
14	*PE_plot...*	PE_plot..	2019-09...	30	d4p-ope...		
15	ReadD...	ReadD...	2019-09...	61	d4p-ope...	Comput...	

Runitme Monitoring

- Data produced
- Messages (Errors)
- Workers Nodes
- Event Times
- Runtime Changes

Data Dependency Graph

Double Click on the border data-nodes to expand. Right Click on each data-node to access its info

Navigation steps 1

Data Detail

Output Files : Open

Output Metadata:

station: ARRO

origin simulated

network: IV

Data products

Output Files :

Output Metadata:

starttime: 2013-02-16T21:16:09.240000Z

delta: 0.01

calib: 1

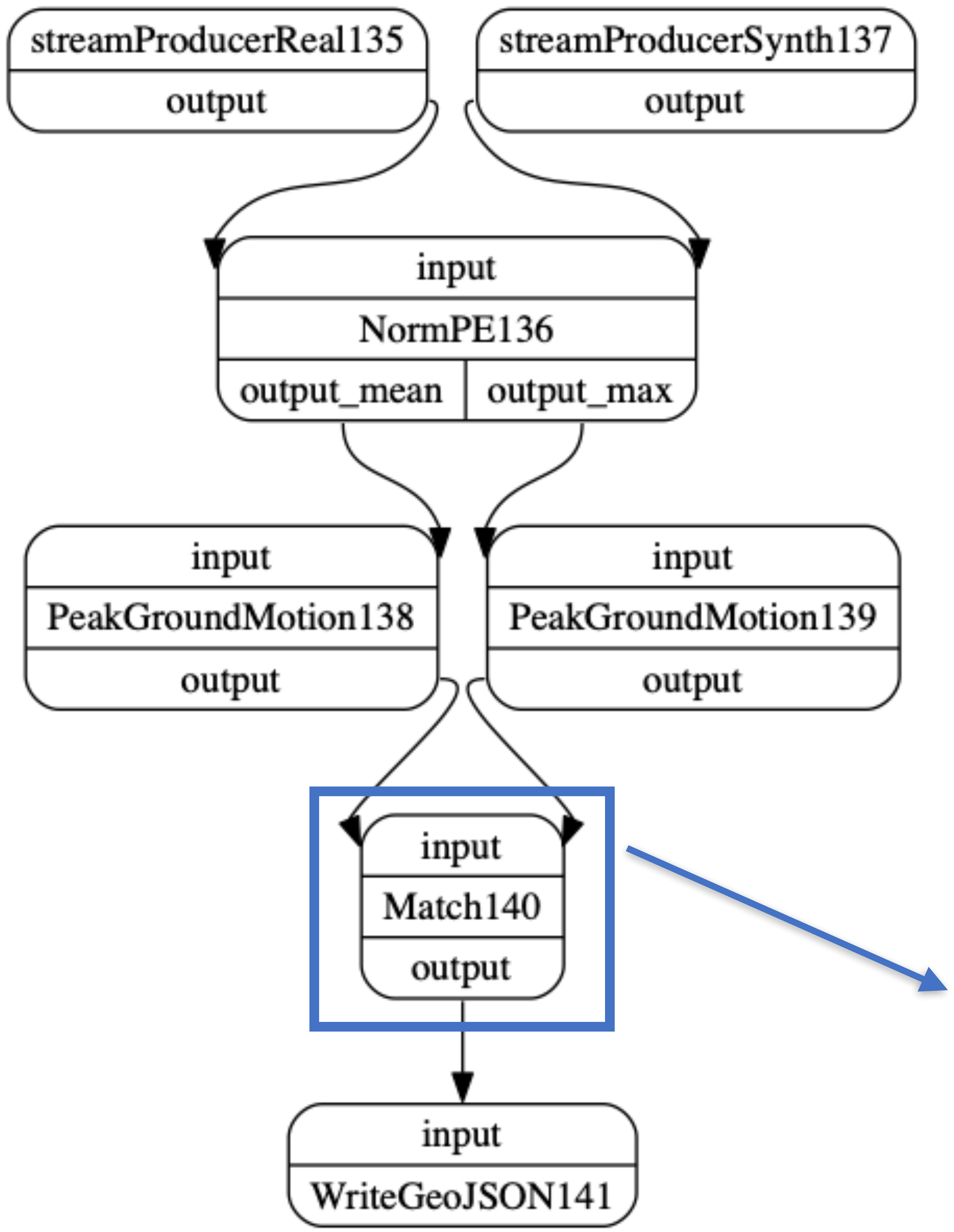
sampling_rate: 100

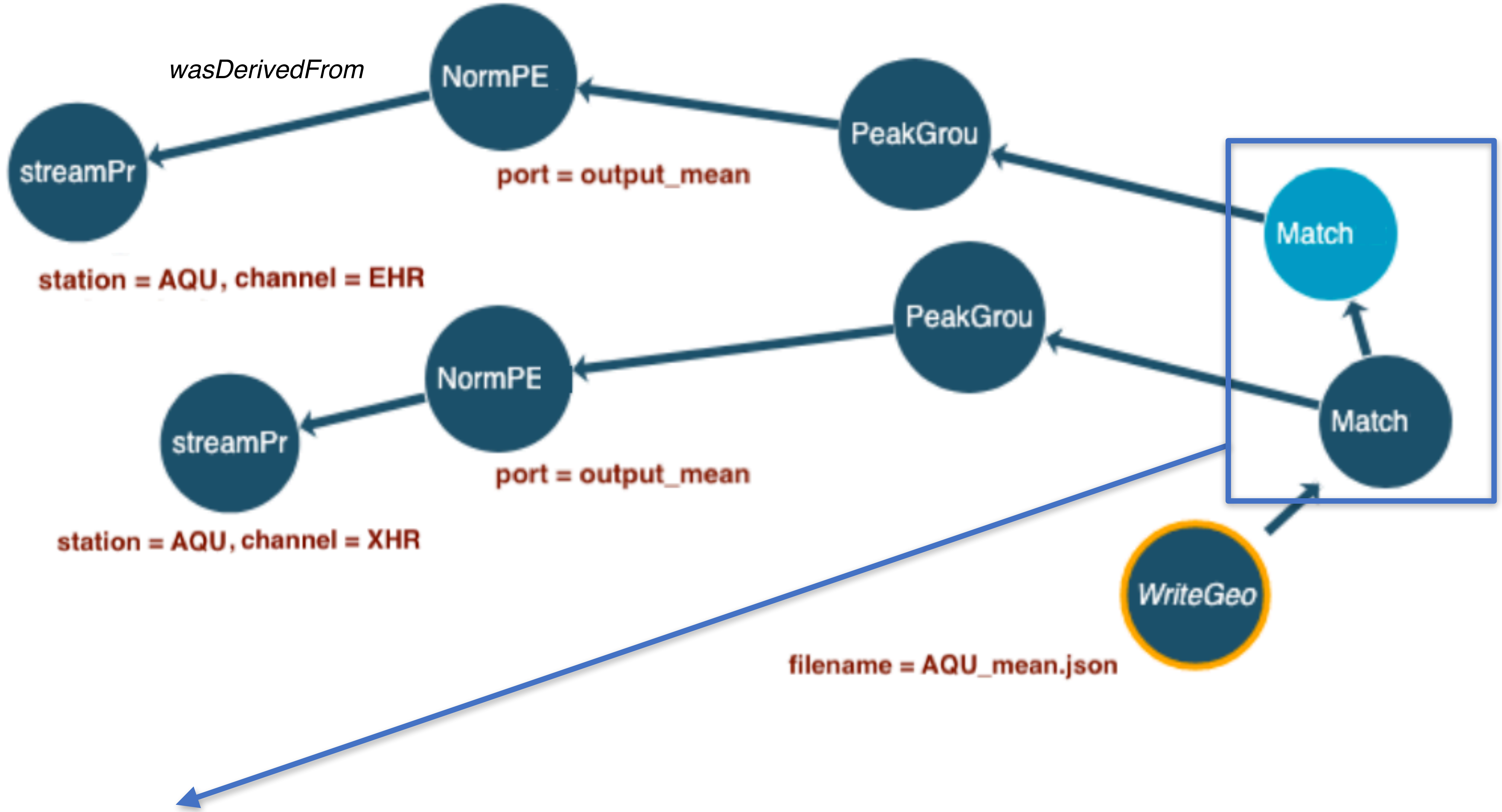
S-ProvFlow:https://gitlab.com/project-dare/s-ProvFlow

Lineage Precision - Stateful operators

Ground Motion Parameters

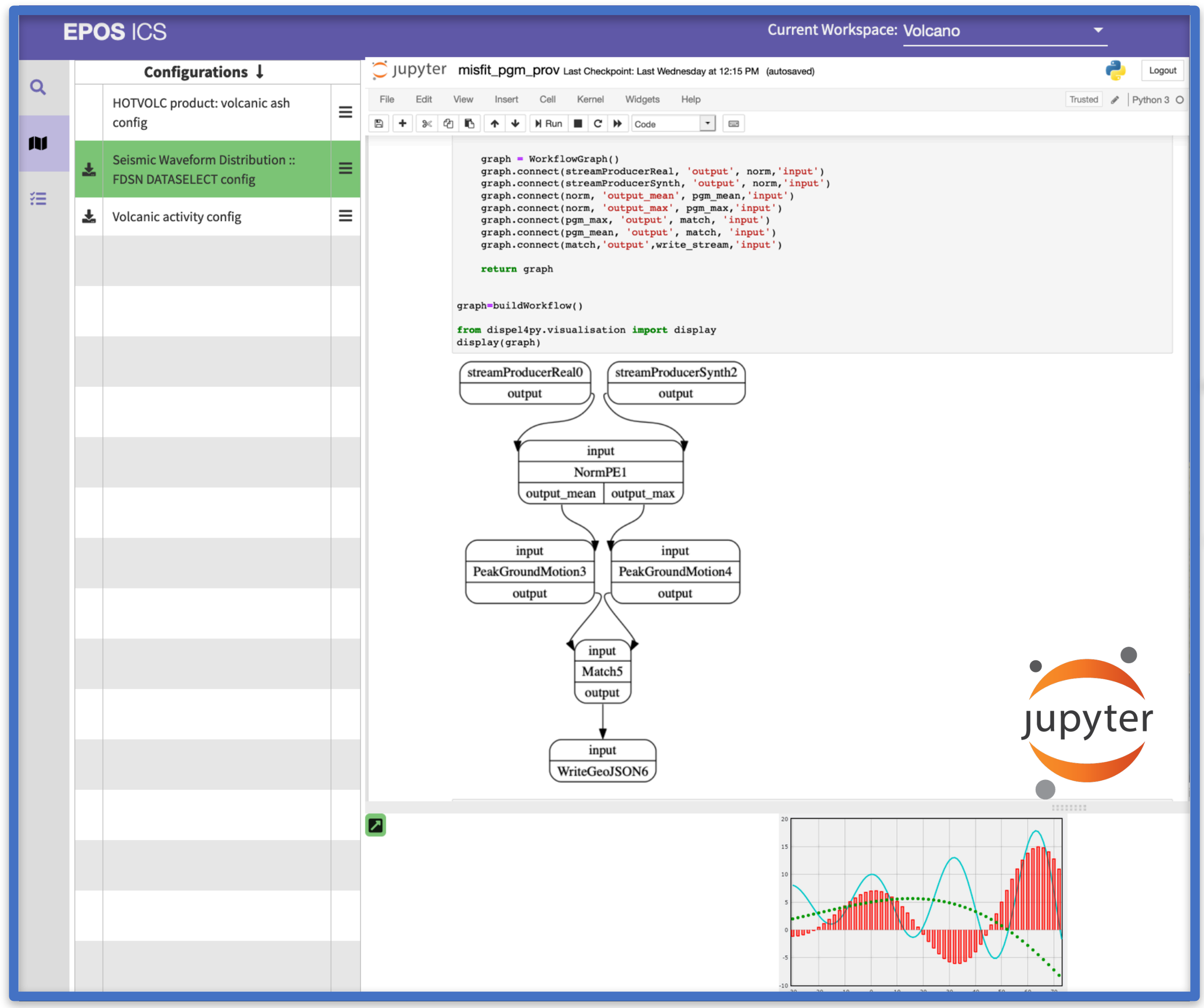
seis:PGMCalculation



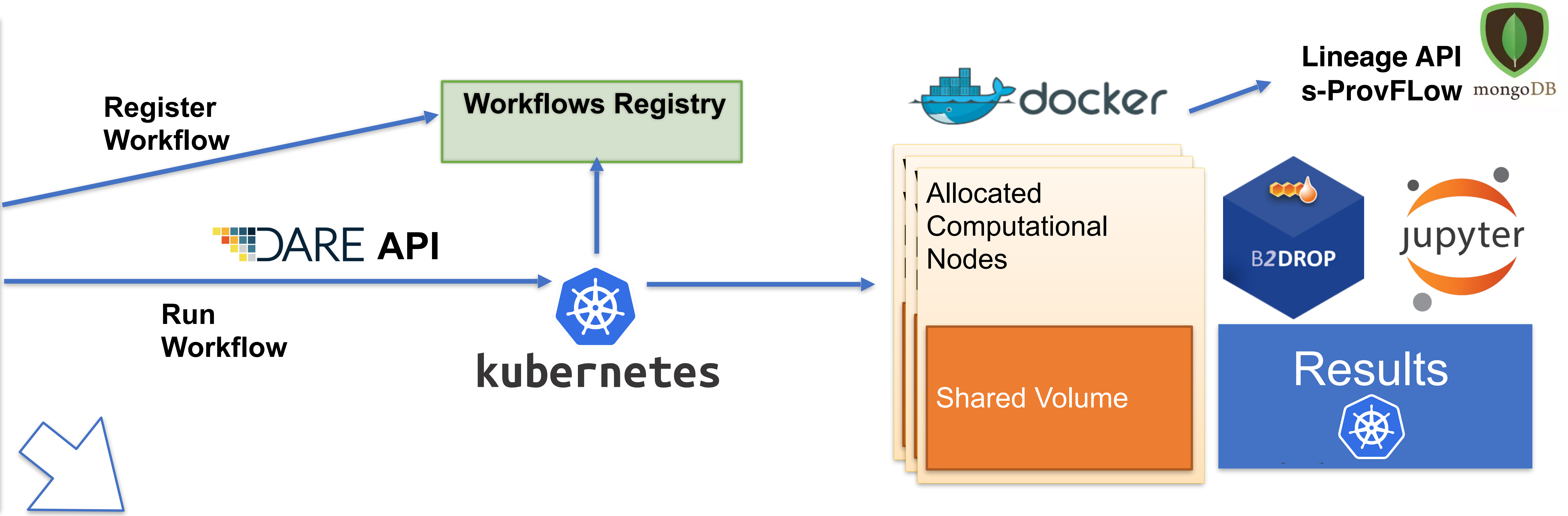


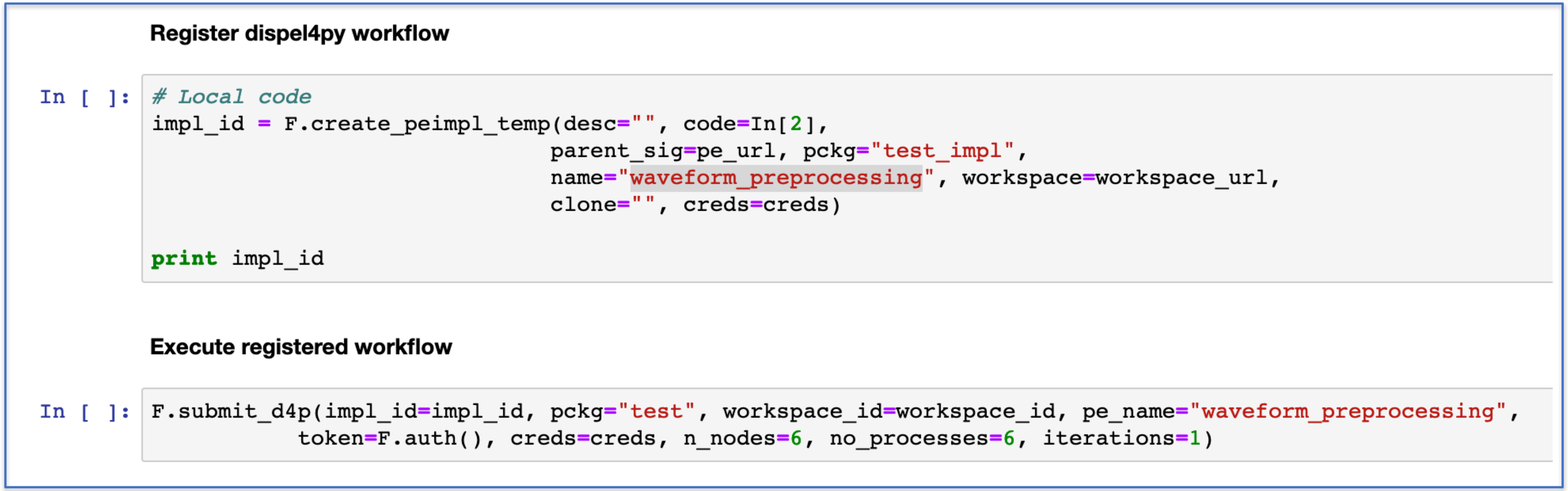
Pattern Type: *Grouped Accumulator (Stateful operator - ASTGrouped)*
Combines intermediate inputs before producing results

Workflow Registration, Execution Monitoring



```
    graph = WorkflowGraph()
    graph.connect(streamProducerReal, 'output', norm,'input')
    graph.connect(streamProducerSynth, 'output', norm,'input')
    graph.connect(norm, 'output_mean', pgm_mean,'input')
    graph.connect(norm, 'output_max', pgm_max,'input')
    graph.connect(pgm_max, 'output', match, 'input')
    graph.connect(pgm_mean, 'output', match, 'input')
    graph.connect(match,'output',write_stream,'input')

    return graph


graph=buildWorkflow()

from dispel4py.visualisation import display
display(graph)
```





Register dispel4py workflow

```
In [ ]: # Local code
        impl_id = F.create_peimpl_temp(desc="", code=In[2],
                                       parent_sig=pe_url, pckg="test_impl",
                                       name="waveform_preprocessing", workspace=workspace_url,
                                       clone="", creds=creds)

        print impl_id
```

Execute registered workflow

```
In [ ]: F.submit_d4p(impl_id=impl_id, pckg="test", workspace_id=workspace_id, pe_name="waveform_preprocessing",
                     token=F.auth(), creds=creds, n_nodes=6, no_processes=6, iterations=1)
```

-Develop & Register Workflows

-Scalable Workflow execution on containerised resources

-Lineage Capturing and visualisation

Linking executions and semantic tagging
Exploring the Experiments' space

Provenance of Multiple experiments with many stages

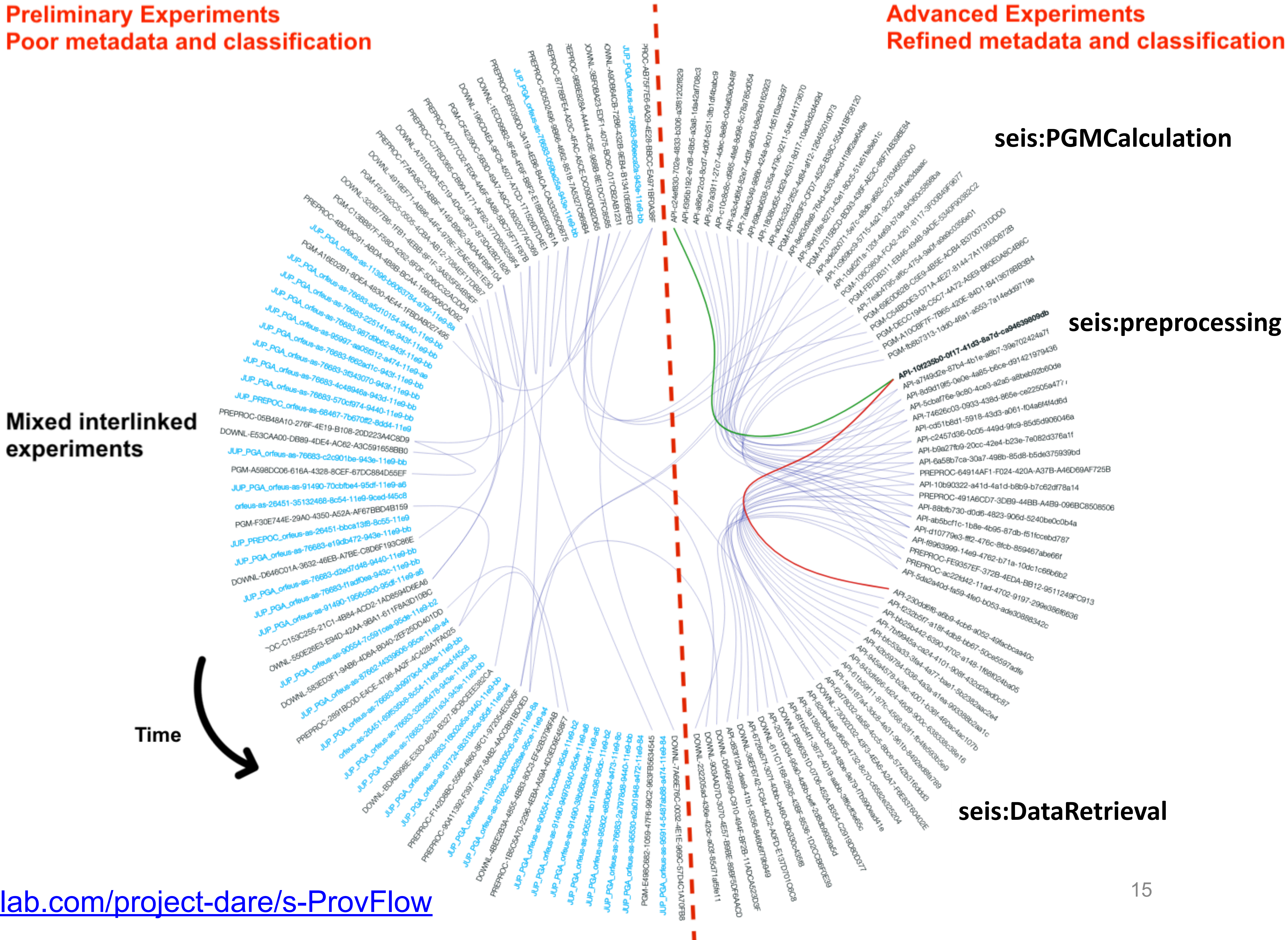
Visual analytics of data reuse between the workflows of the RA use case

Runs selected among those using the same station codes. (Contextual metadata)

Vertices: workflows execution *ids* **colour-coded by user**

Edges: data flows. Red and green edges for data input and output.

Right half: better descriptions yield the improved understanding, discovery and reuse of the results



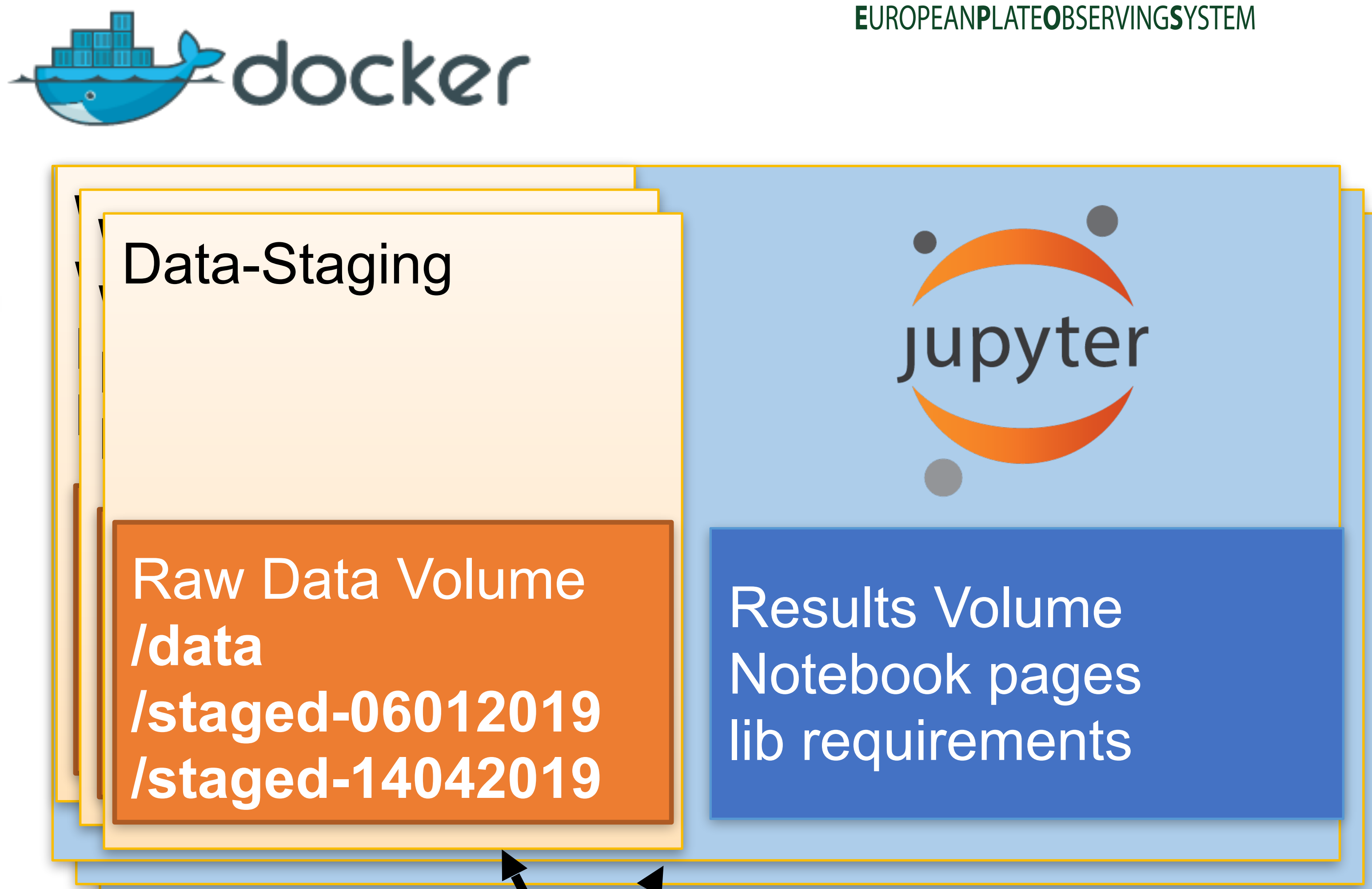
S-ProvFlow:https://gitlab.com/project-dare/s-ProvFlow

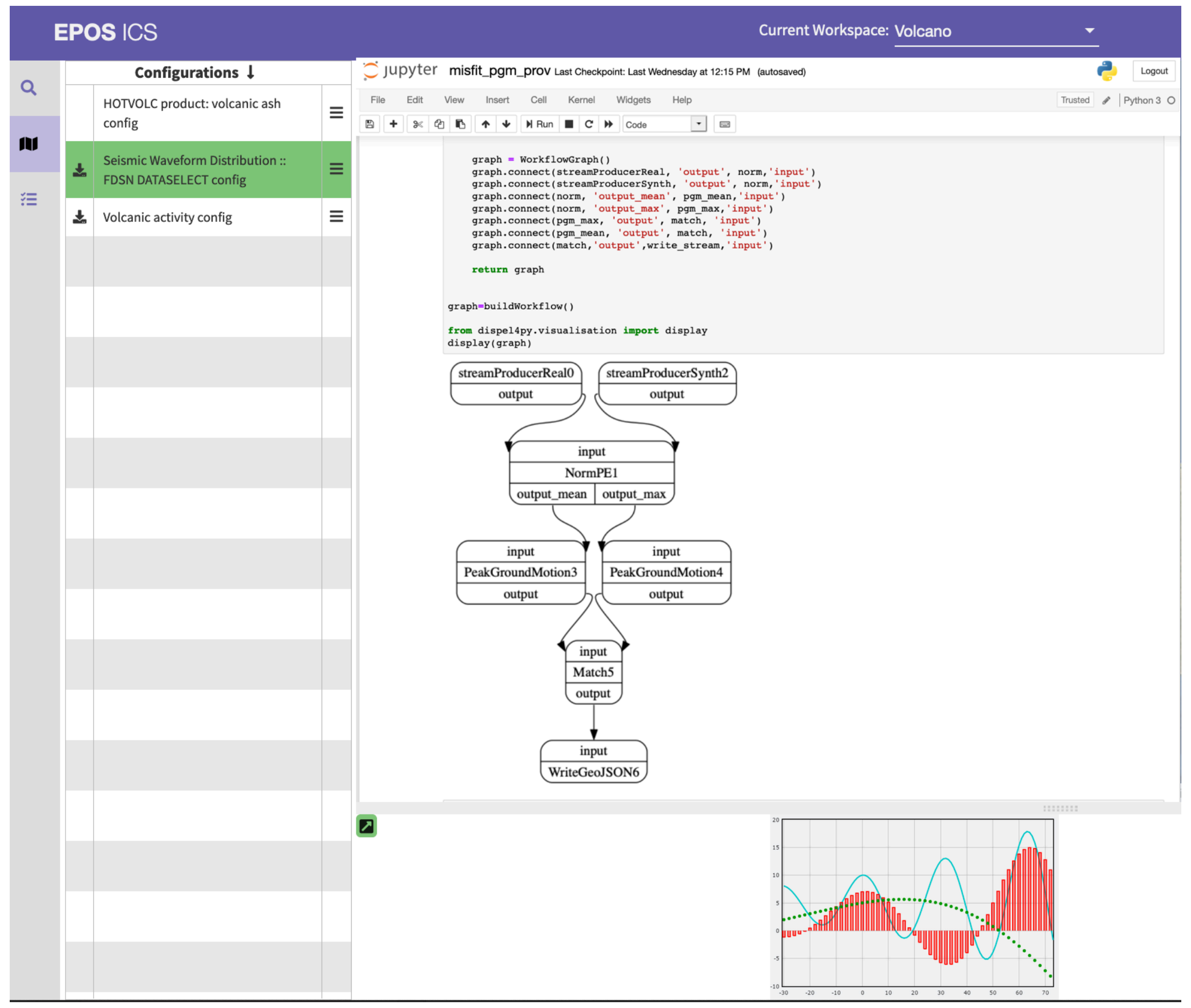
Provenance-aware Workspaces

Notebook API

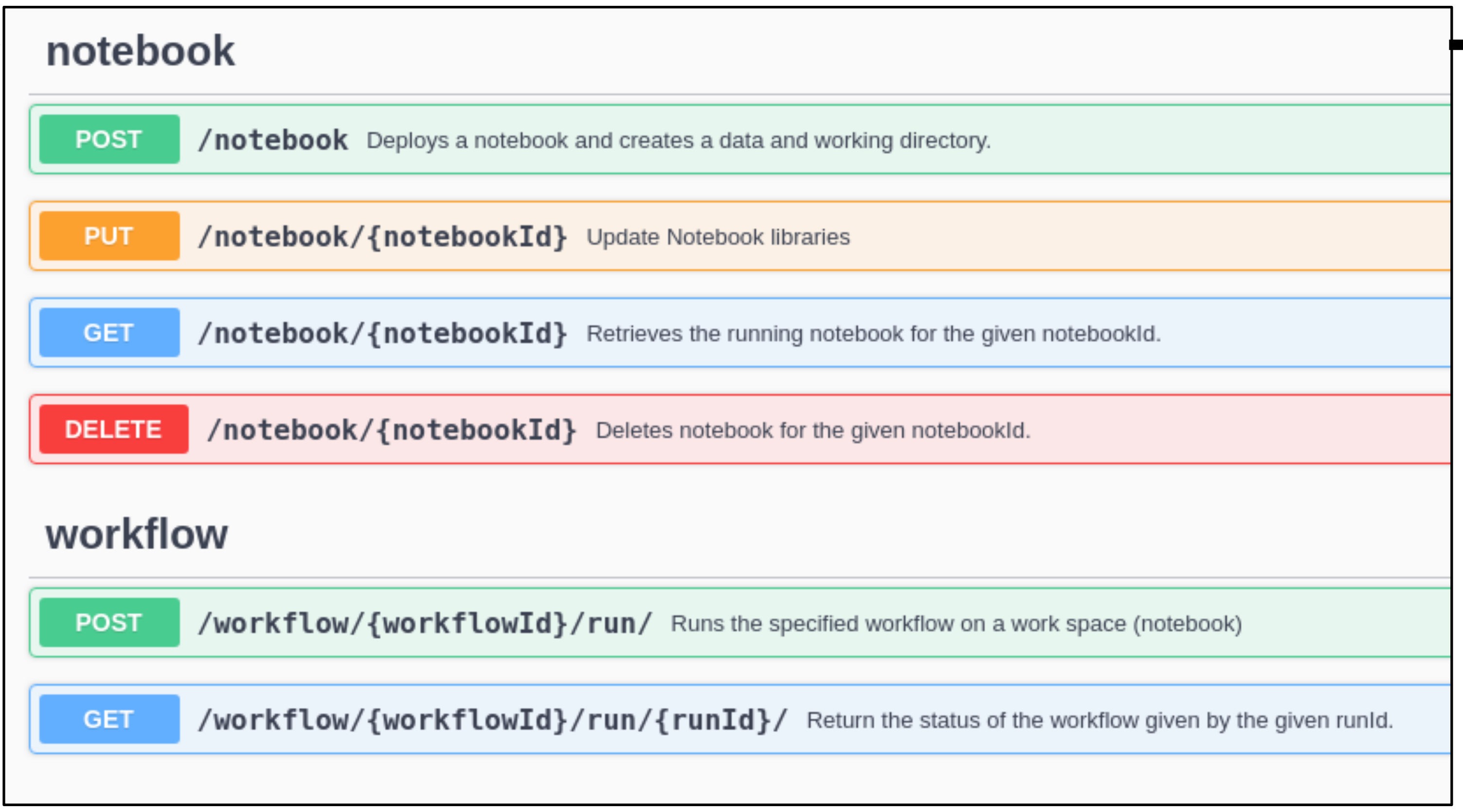
A Web API to:

- **Create Notebook** Workspaces with the required libraries
- **Stage, pre-process** data onto active Workspaces (pre-built workflows)
  - Data staging history
  - Read-only and extensible raw data
- **Update** the Workspace libraries
- **On demand snapshots** of the Working Session (K8S volumes/Git/Binder)





```
    graph = WorkflowGraph()
    graph.connect(streamProducerReal, 'output', norm,'input')
    graph.connect(streamProducerSynth, 'output', norm,'input')
    graph.connect(norm, 'output_mean', pgm_mean,'input')
    graph.connect(norm, 'output_max', pgm_max,'input')
    graph.connect(pgm_max, 'output', match, 'input')
    graph.connect(pgm_mean, 'output', match, 'input')
    graph.connect(match,'output',write_stream,'input')

    return graph

graph=buildWorkflow()

from dispel4py.visualisation import display
display(graph)
```

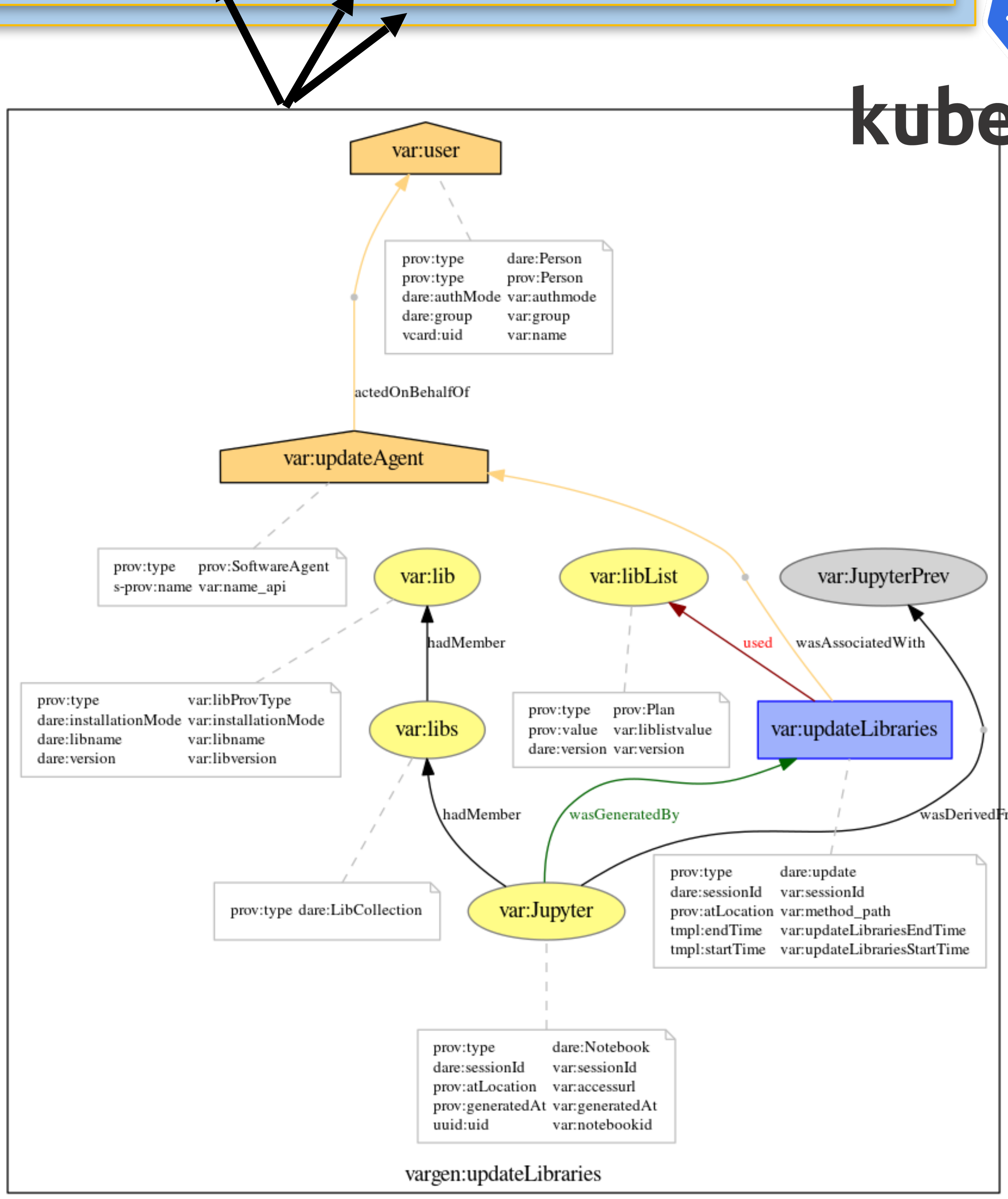


notebook

POST	/notebook	Deploys a notebook and creates a data and working directory.
PUT	/notebook/{notebookId}	Update Notebook libraries
GET	/notebook/{notebookId}	Retrieves the running notebook for the given notebookId.
DELETE	/notebook/{notebookId}	Deletes notebook for the given notebookId.

workflow

POST	/workflow/{workflowId}/run/	Runs the specified workflow on a work space (notebook)
GET	/workflow/{workflowId}/run/{runId}/	Return the status of the workflow given by the given runId.



Provenance-aware Workspaces

Notebook API

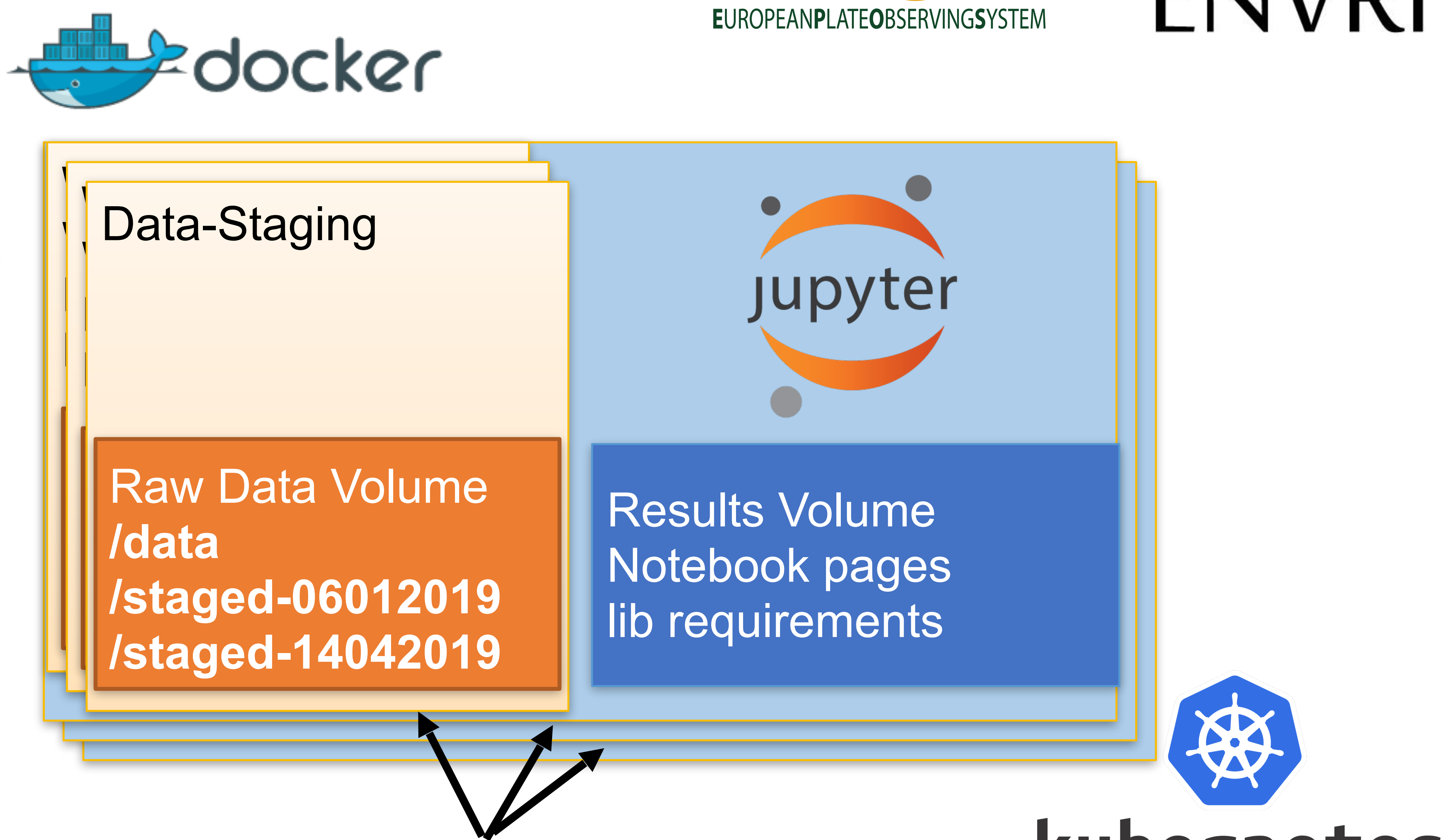
A Web API to:

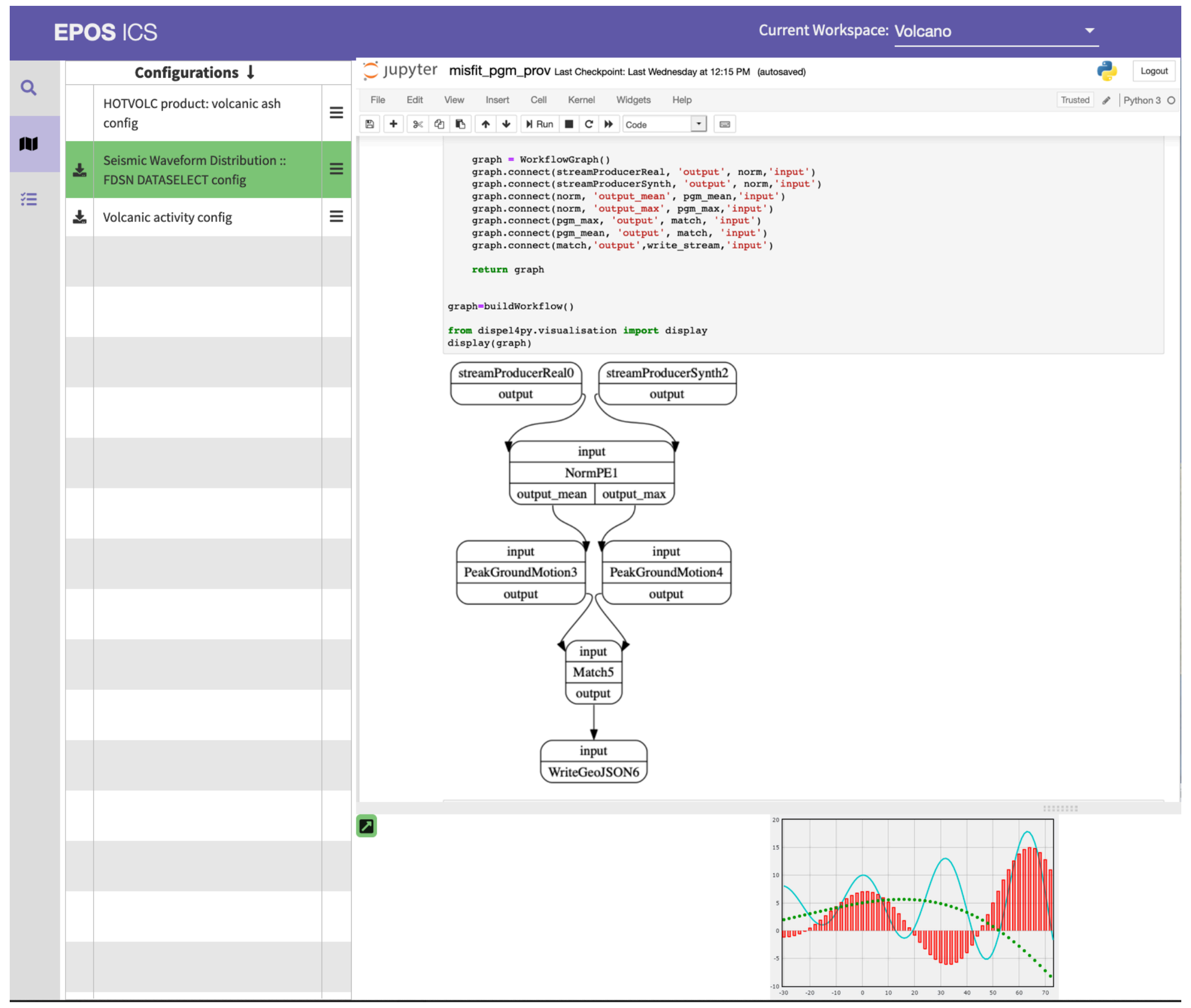
- **Create Notebook** Workspaces with the required libraries

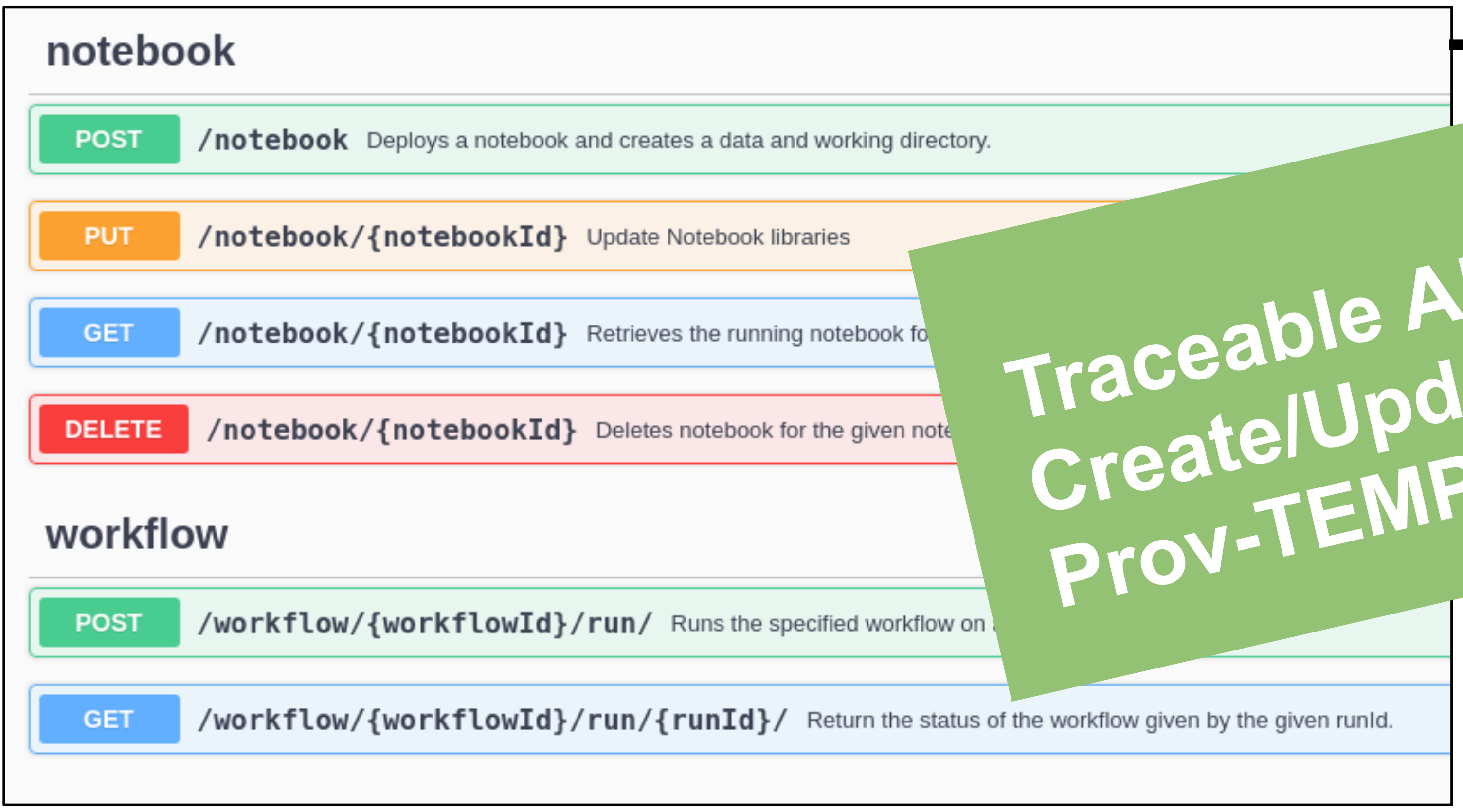
- **Stage, pre-process** data onto active Workspaces (pre-built workflows)
  - Data staging history
  - Read-only and extensible raw data

- **Update** the Workspace libraries

- **On demand snapshots** of the Working Session (K8S volumes/Git/Binder)




EPOS ICS
Current Workspace: Volcano
Configurations
HOTVOLC product: volcanic ash config
Seismic Waveform Distribution :: FDSN DATASELECT config
Volcanic activity config
jupyter misfit_pgm_prov



notebook

POST /notebook Deploys a notebook and creates a data and working directory.

PUT /notebook/{notebookId} Update Notebook libraries

GET /notebook/{notebookId} Retrieves the running notebook fo

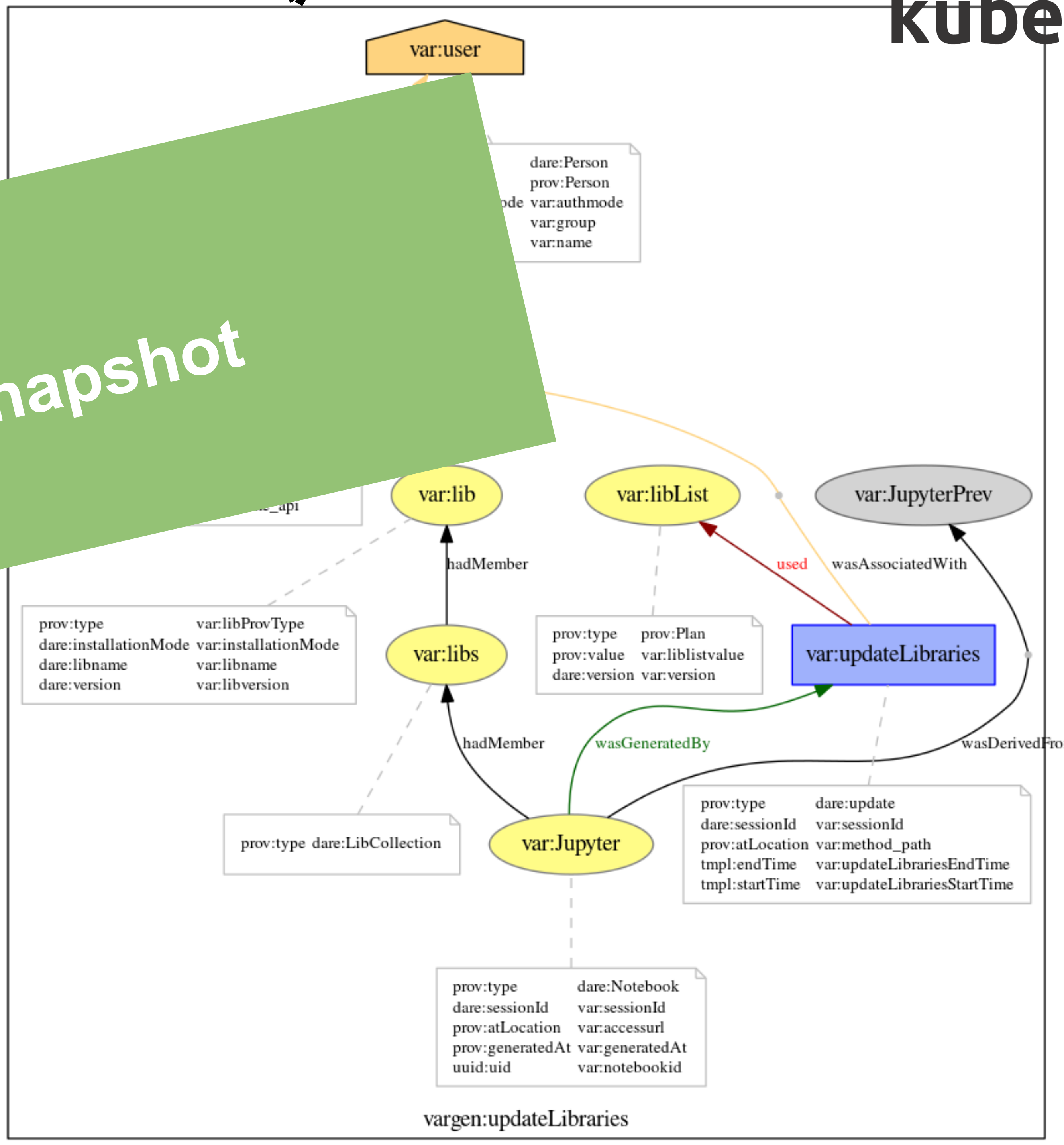
DELETE /notebook/{notebookId} Deletes notebook for the given note

workflow

POST /workflow/{workflowId}/run/ Runs the specified workflow on

GET /workflow/{workflowId}/run/{runId}/ Return the status of the workflow given by the given runId.

Traceable API Methods
Create/Update/Stage/Snapshot
Prov-TEMPLATE


var:user
dare:Person
prov:Person
var:authmode
var:group
var:name
var:lib
var:libList
var:JupyterPrev
hadMember
used
wasAssociatedWith
prov:type var:libProvType
dare:installationMode var:installationMode
dare:libname var:libname
dare:version var:libversion
var:libs
prov:type prov:Plan
prov:value var:liblistvalue
dare:version var:version
var:updateLibraries
hadMember
wasGeneratedBy
wasDerivedFrom
prov:type dare:LibCollection
var:Jupyter
prov:type dare:update
dare:sessionId var:sessionId
prov:atLocation var:method_path
tmpl:endTime var:updateLibrariesEndTime
tmpl:startTime var:updateLibrariesStartTime
prov:type dare:Notebook
dare:sessionId var:sessionId
prov:atLocation var:accessurl
prov:generatedAt var:generatedAt
uuid:uid var:notebookid
vargen:updateLibraries

Luc Moreau et al. A Templating System to Generate Provenance
https://eprints.soton.ac.uk/405025/1/provtemplate.pdf

ProvenaceTemplate Catalogue
https://github.com/EnvriPlus-PROV/ProvTemplateCatalog

Conclusions & Future Work

- **Balanced automation and *Active* human contribution** in provenance capturing in Data-Intensive workflows

- **Provenance model** S-PROV, that accommodates complex lineage patterns

- **A conceptual design** based on reusable and combinable *Provenance Types* that lead to the *Provenance Configuration*

- **Services and tools** developed around our framework to control and evaluate the executions (DARE API, S-ProvFlow)

- **Coming Next!** Integration within Traceable Workspaces

DARE http://project-dare.eu

Koninklijk Nederlands
Meteorologisch Instituut
Ministerie van Infrastructuur en Milieu

Thanks!

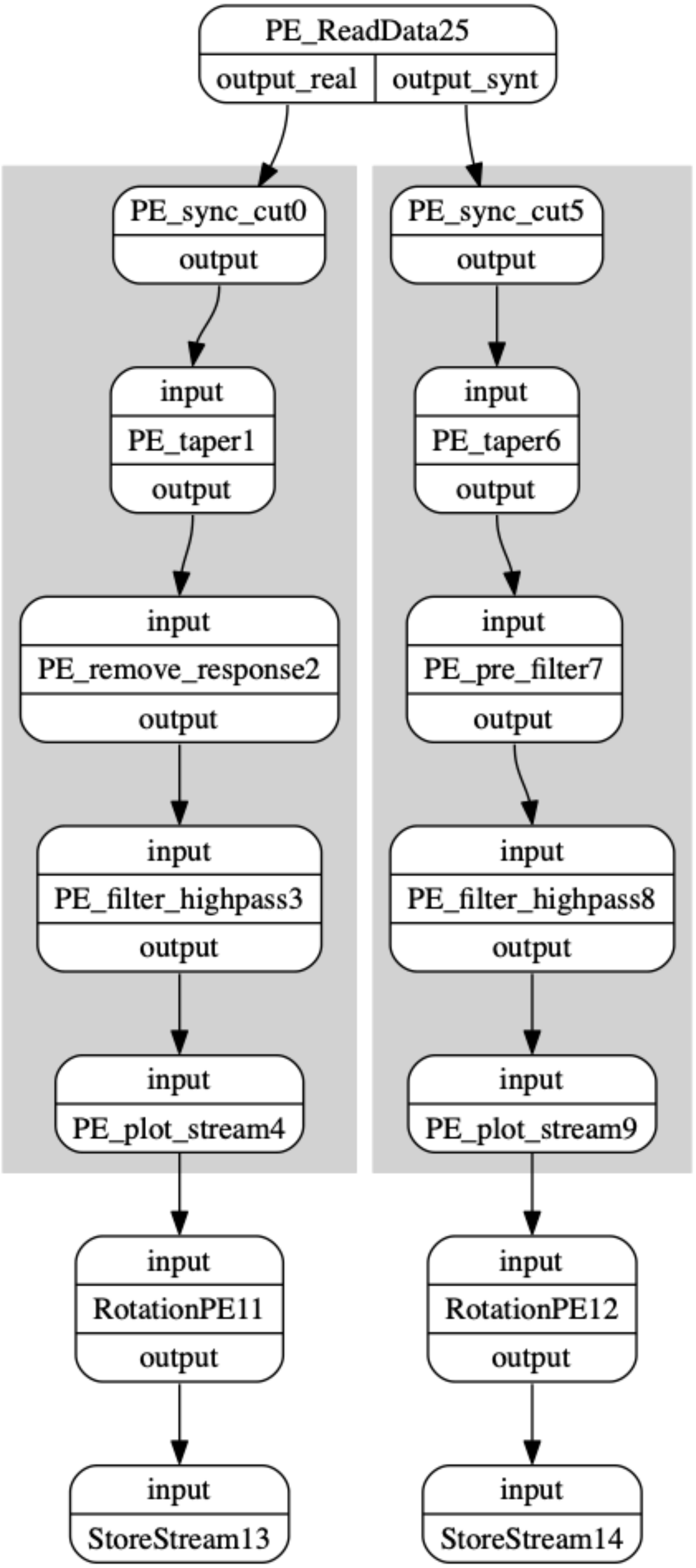
ECMWF #repwork19

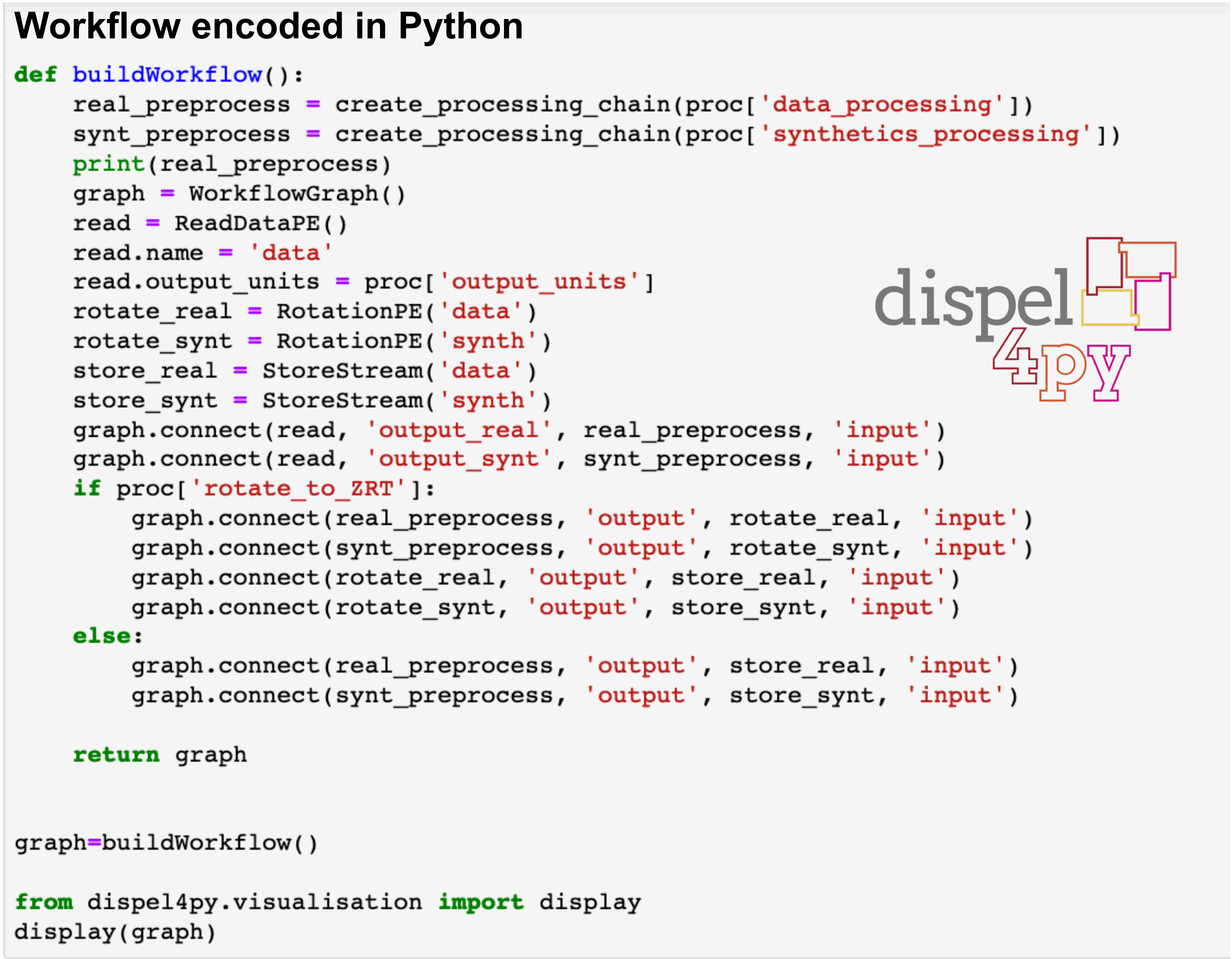
Specify Workflow

Waveform Preprocessing

pipeline JSON Description (eg. from file)

Manual Extensions





Workflow encoded in Python

```
def buildWorkflow():
    real_preprocess = create_processing_chain(proc['data_processing'])
    synt_preprocess = create_processing_chain(proc['synthetics_processing'])
    print(real_preprocess)
    graph = WorkflowGraph()
    read = ReadDataPE()
    read.name = 'data'
    read.output_units = proc['output_units']
    rotate_real = RotationPE('data')
    rotate_synt = RotationPE('synth')
    store_real = StoreStream('data')
    store_synt = StoreStream('synth')
    graph.connect(read, 'output_real', real_preprocess, 'input')
    graph.connect(read, 'output_synt', synt_preprocess, 'input')
    if proc['rotate_to_ZRT']:
        graph.connect(real_preprocess, 'output', rotate_real, 'input')
        graph.connect(synt_preprocess, 'output', rotate_synt, 'input')
        graph.connect(rotate_real, 'output', store_real, 'input')
        graph.connect(rotate_synt, 'output', store_synt, 'input')
    else:
        graph.connect(real_preprocess, 'output', store_real, 'input')
        graph.connect(synt_preprocess, 'output', store_synt, 'input')

    return graph


graph=buildWorkflow()

from dispel4py.visualisation import display
display(graph)
```

Why PROV Templates (as a service)

- **Templates foster discussions** on provenance relationships involving heterogeneous agents and resources (**Human vs System Concerns**).

- **Modelling of usable** and **re-usable** provenance scenarios (tailoring vs generalisation)

- **Remove the burden** to **hardcode provenance editing**

  (expansion tools/services)

Luc Moreau et al. A Templating System to Generate Provenance
https://eprints.soton.ac.uk/405025/1/provtemplate.pdf
ProvenaceTemplate Catalogue
https://github.com/EnvriPlus-PROV/ProvTemplateCatalog
https://envriplus-provenance.test.fedcloud.eu/