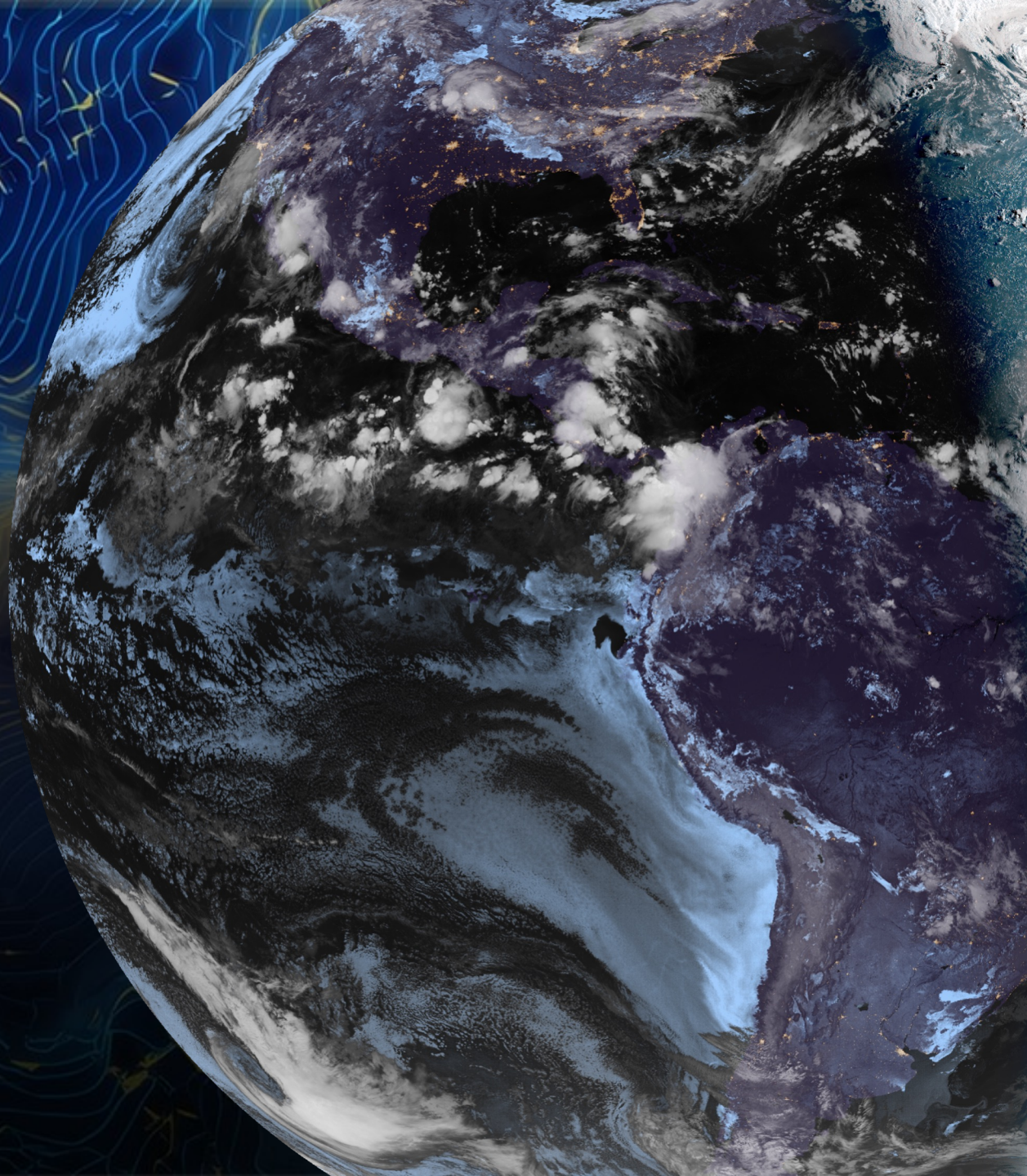
Event-Driven Orchestration for Near Real-Time Data Processing



Geolocated Information
GeoIPS®
Processing System

CIRA
Connecting Models and Observations

Gwyn Uttmark

Research Associate
GeoIPS Project Coordinator and Scientific Developer
Cooperative Institute for Research in the Atmosphere
Colorado State University, Fort Collins, CO

GeoIPS® Geolocated Information Processing System

CIRA Connecting Models and Observations

Cooperative Institute for Research in the Atmosphere
Colorado State University


NOAA
NATIONAL OCEANIC AND ATMOSPHERIC ADMINISTRATION
U.S. DEPARTMENT OF COMMERCE


NASA


DEPARTMENT OF DEFENSE
UNITED STATES OF AMERICA


NATIONAL
PARK
SERVICE


NSF

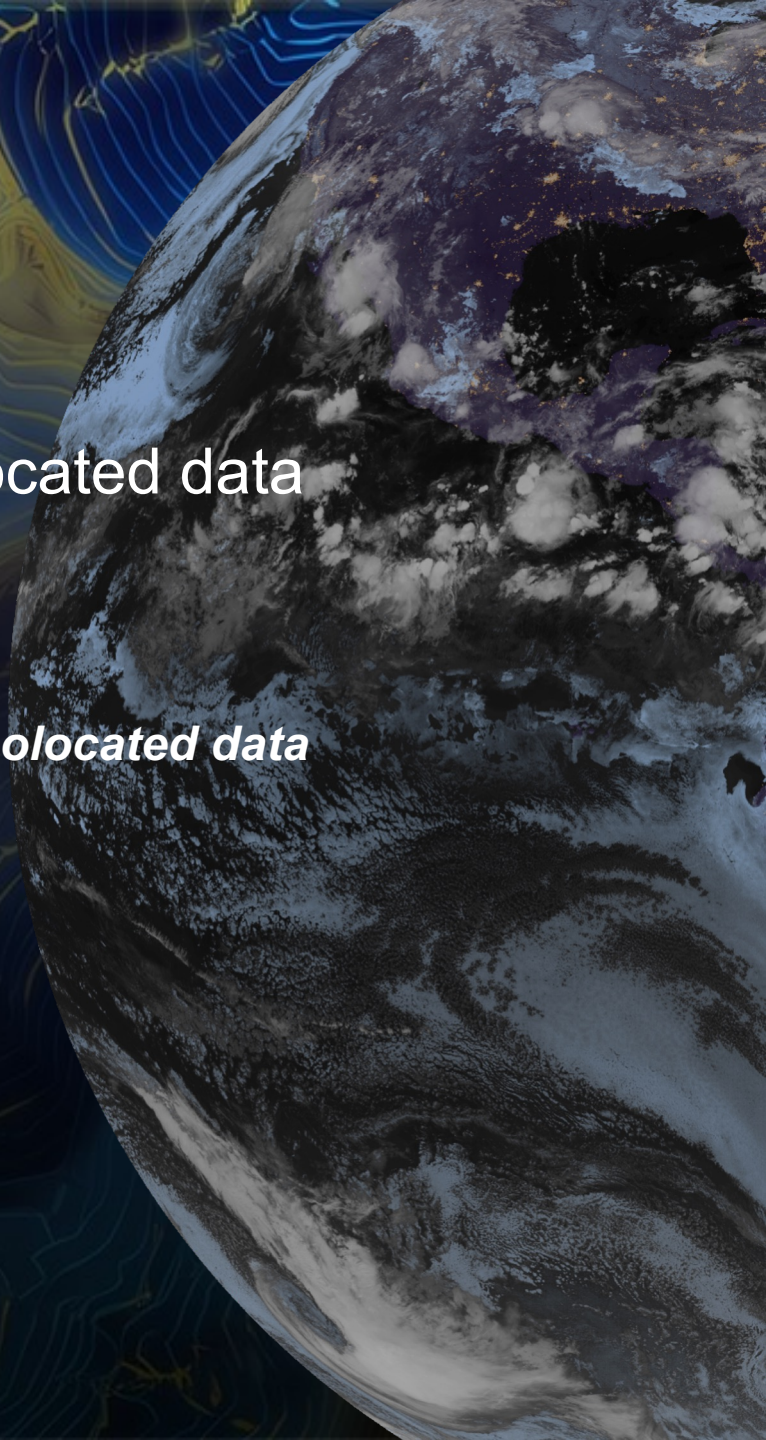

CIRA

Current NOAA CI Locations

Background: GeoIPS®

A Plugin-based System for Processing Geospatial Data

What is GeoIPS?



An open-source plugin-based system for processing geolocated data

- Read, interpolate, apply algorithms to, validate, and write data
- Produce imagery in multiple formats, most commonly PNG or GeoTIFF
- Produce structured output data (e.g. NetCDF, HDF, etc.)
- Primarily used for Earth Science remote sensing ***but applicable to any geolocated data***

An operational system used by the US Navy

- Supported by the Office of Naval Research
- Actively used to support the Navy fleet

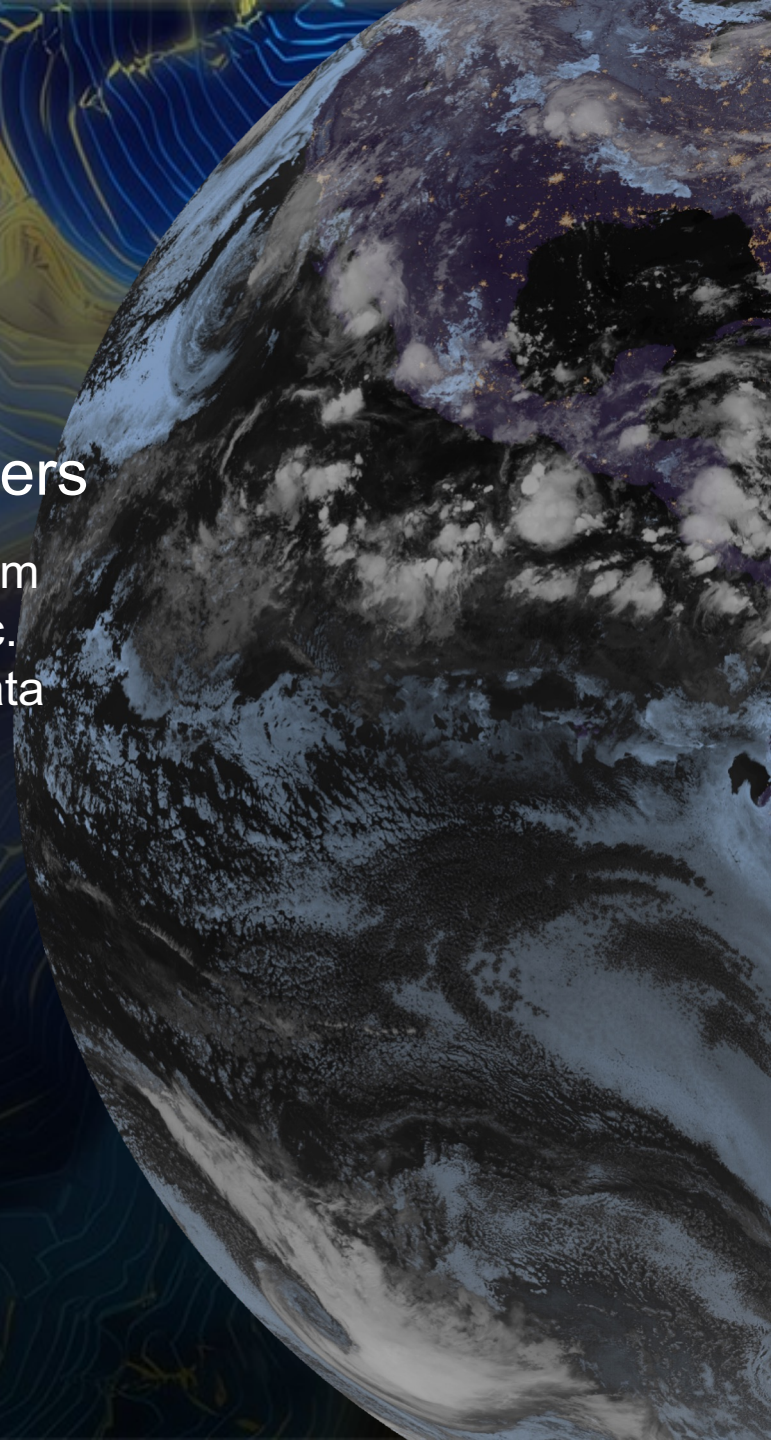
What is GeoIPS?

A tool to bridge the gap between researchers and developers

- Allows the option to either keep new plugins proprietary or easily share them
- Provides easy sharing of readers, algorithms, common data formatters, etc.
- Simplifies collaboration between teams by allowing easy combination of data

A tool to bridge the gap from research to operations

- Add new plugins and test them
- Pass plugins to operational users as installable Python packages
- Improve update cycle through use of git repositories for plugins

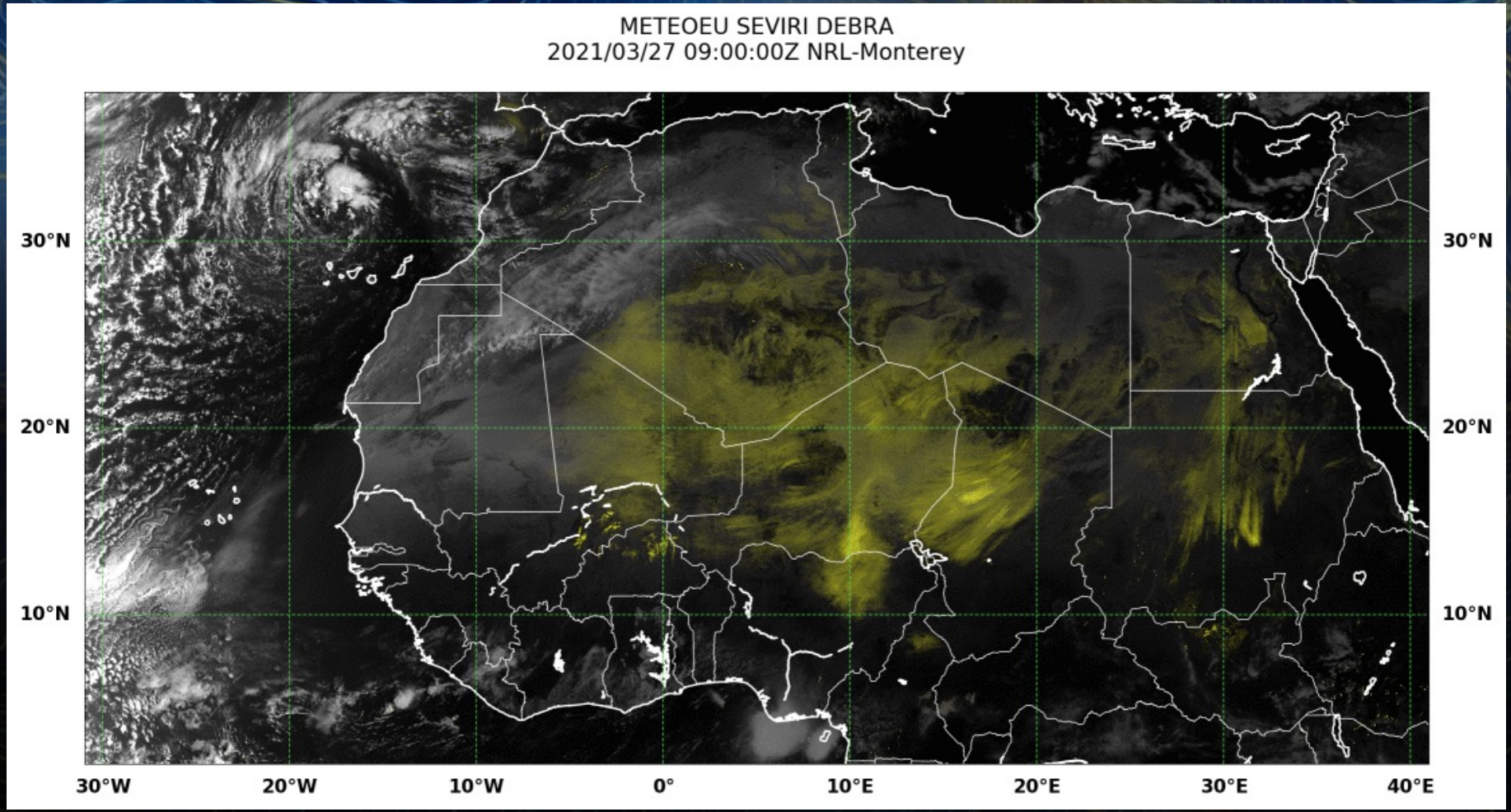


Some transitioned products

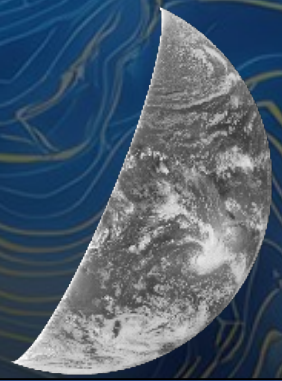
DEBRA – Steve Miller – CIRA

Employs surface emissivity to reduce background artifacts in a multi-channel IR/Vis algorithm for identification of lofted dust.

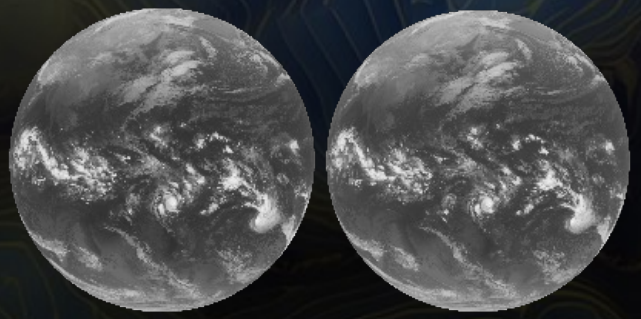
Miller et al., 2017
JGR Atmospheres
https://doi.org/10.1002/2017JD027365



Some transitioned products

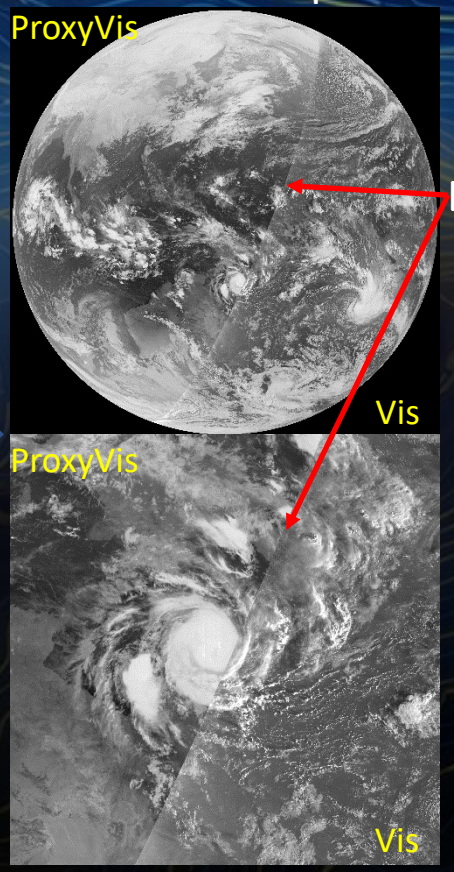


Adjusted Vis channel for day-time



Several IR channels for night-time

ProxyVis Himawari Full Disk Output

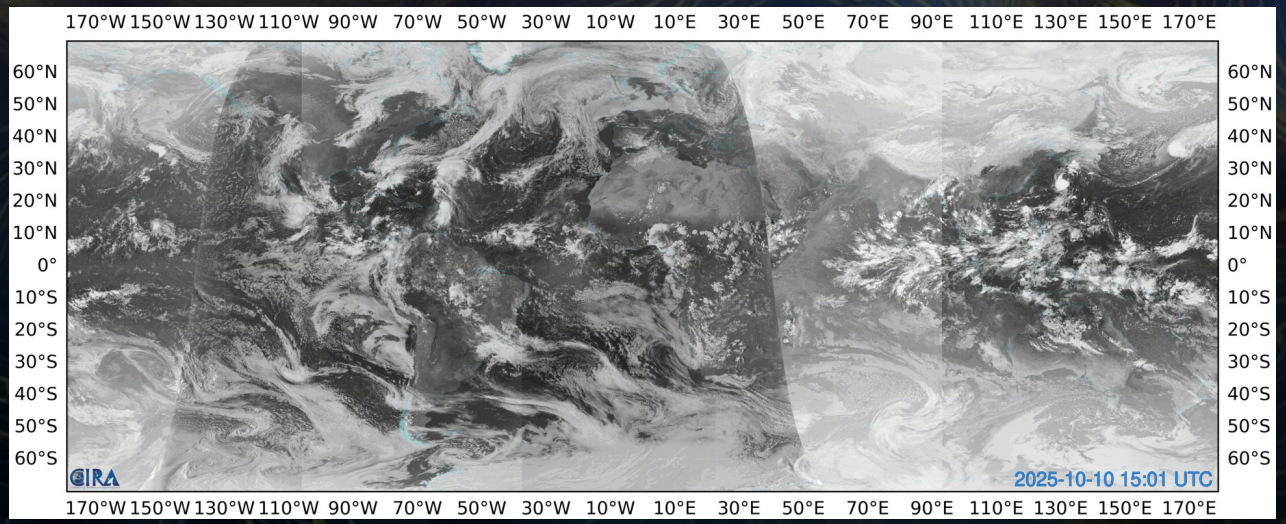


ProxyVis TC-Centric Output

Proxy Vis – Galina Chirokova – CIRA

Combines adjusted daytime visible with multiple IR channels to produce a 24-hour visible-like product.

Day-Night Terminator



Chirokova et al., 2023
Weather & Forecasting
https://doi.org/10.1175/WAF-D-23-0038.1

POC: Galina.Chirokova@rams.colostate.edu

Some transitioned products

GeoColor - Steve Miller - CIRA

Miller et al., 2019
J. Atmos. Oceanic Technol
https://doi.org/10.1175/JTECH-D-19-0134.1

Included Data Readers

GeoIPS provides many built-in data readers including for GEO and LEO sensors, derived data products, and NWP data.

We are always happy to add new readers and would be excited to expand beyond meteorological applications.

Legend:
Low Earth Orbit
Geostationary
Model
Derived Data
Field Campaign

ABI
AHI
AMI
AVIRIS
AVIRIS-NG
AMSR-2
AMSU-A
AMSU-B
ASCAT
ATMS
AWS (Arctic Weather Satellite)
CLAVR-x
CYGNSS
EWSG
FCI
GFS (model)
GMI
IMERG
MetOp-B
MetOp-C
MIMIC
MODIS
OSCAT
OVERCAST
SAPHIR
SEVIRI
SGLI
SMAP
SMOS
SSMI
SSMI/S
tomorrow.io
VIIRS
WFABBA
WindSat


Gathering Atmospheric HRIT Data with a Custom-Made Satellite Receiver
WALTER SCOTT, JR. COLLEGE OF ENGINEERING
COLORADO STATE UNIVERSITY
CIRA
Introduction
Methods
Conclusion
References & Acknowledgements
NOAA
GeoIPS

Scripting with GeoIPS

```
"""Produce an infrared image from GOES-16 ABI data."""

from glob import glob
from matplotlib import pyplot as plt

from geoips.interfaces import readers


rdr = readers.get_plugin("abi_netcdf")

data = rdr(
    glob("test_data/goes16_20200918_1950/*.nc"),
    chans=["B14BT"],
)
infrared = data["LOW"]["B14BT"]

fig = plt.figure(figsize=(10, 10))
ax = fig.add_axes((0, 0, 1, 1))
ax.imshow(infrared, vmin=170, vmax=320, cmap="Greys")
fig.savefig("infrared.png")
```

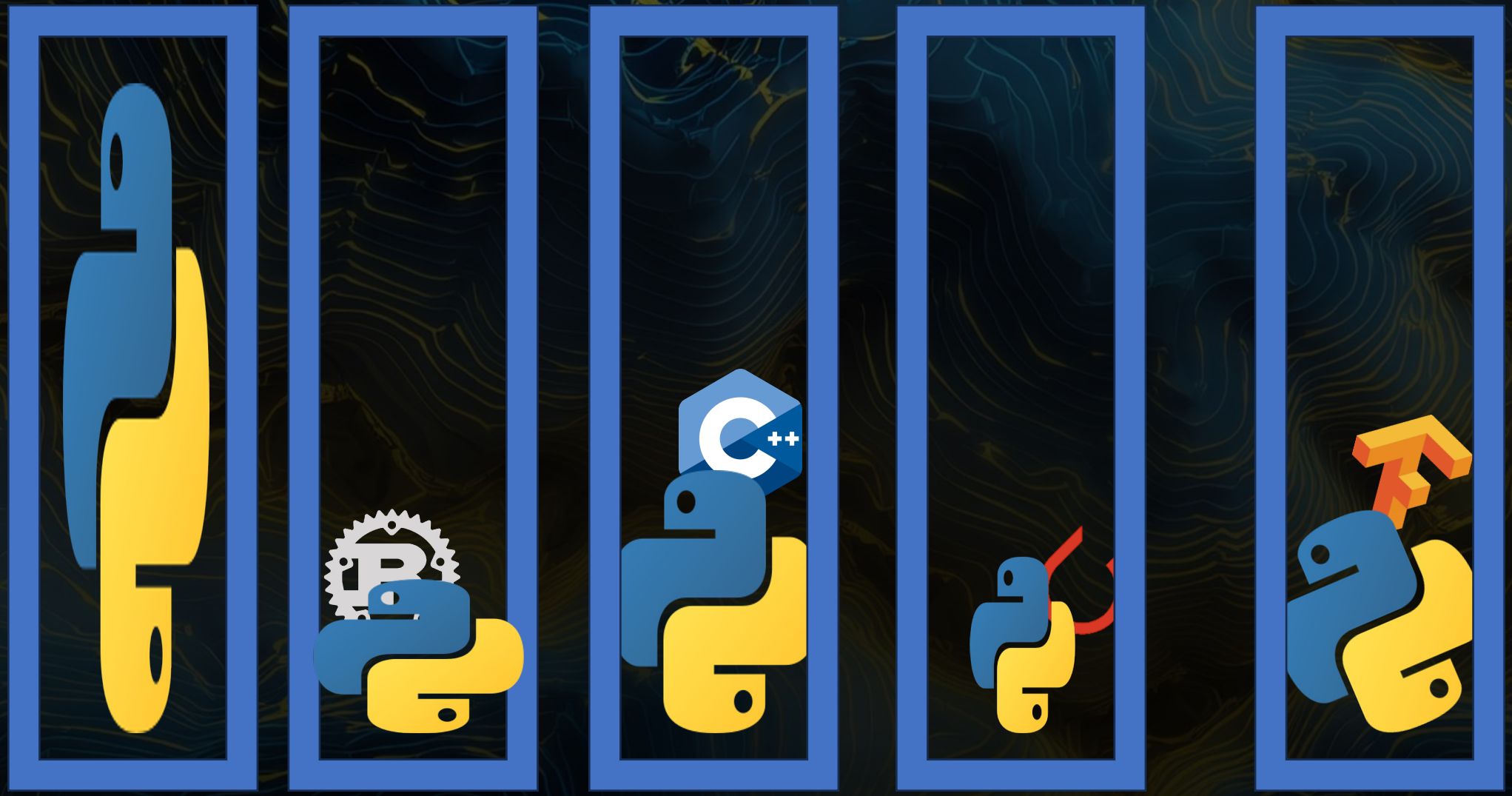
- GeoIPS plugins can be imported directly and used in scripts.
- These scripts can do anything they want to the data.
- The script at left imports a GeoIPS *Reader* plugin to read ABI data, then plots it using matplotlib.
- For more complex graphics (e.g. with gridlines, coastlines, dynamic titles, etc, an *OutputFormatter* plugin can be used.
- *Interpolators* and *Sectors* can be employed to reproject data.

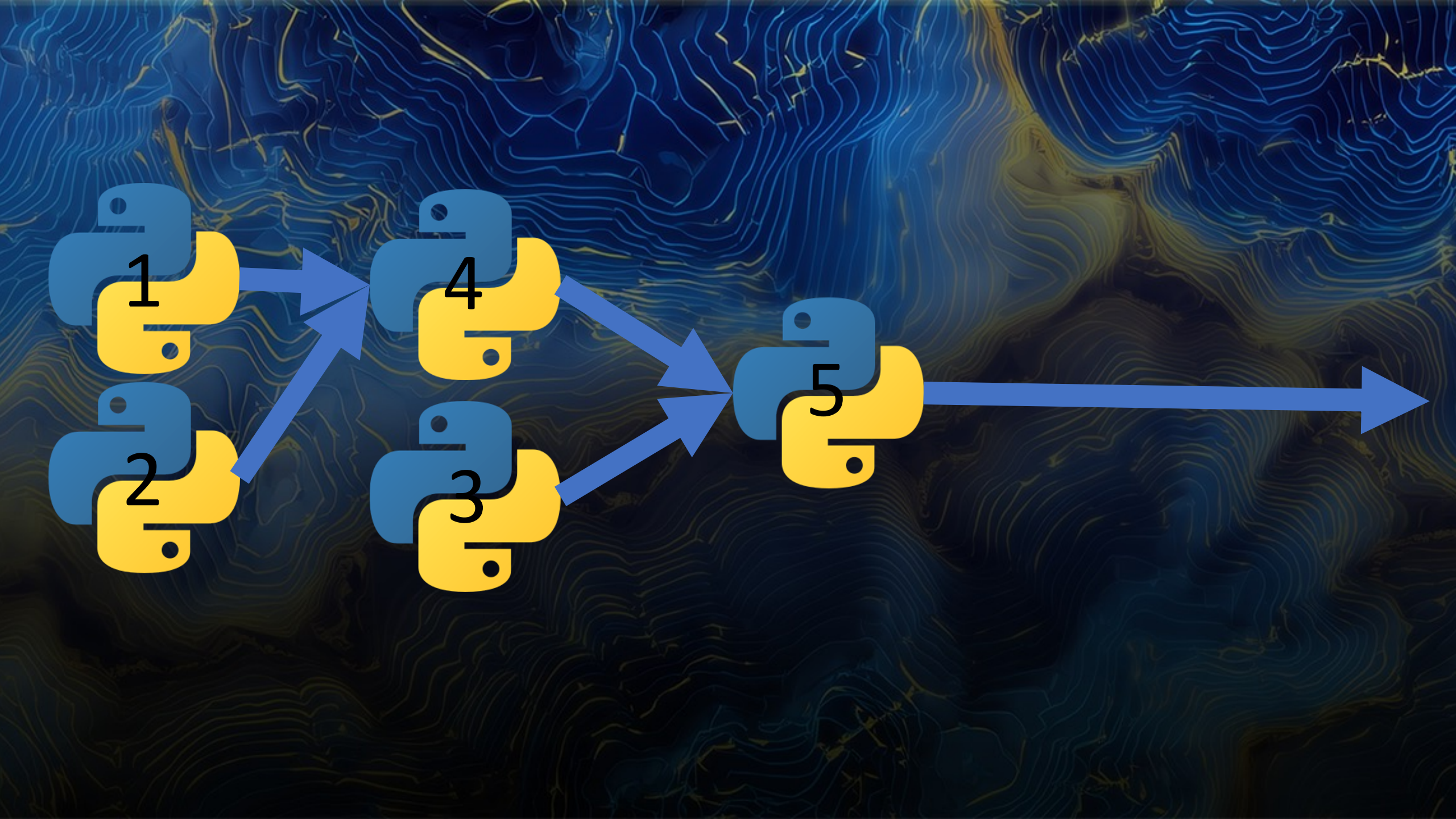
Infrastructure as Code Workflows

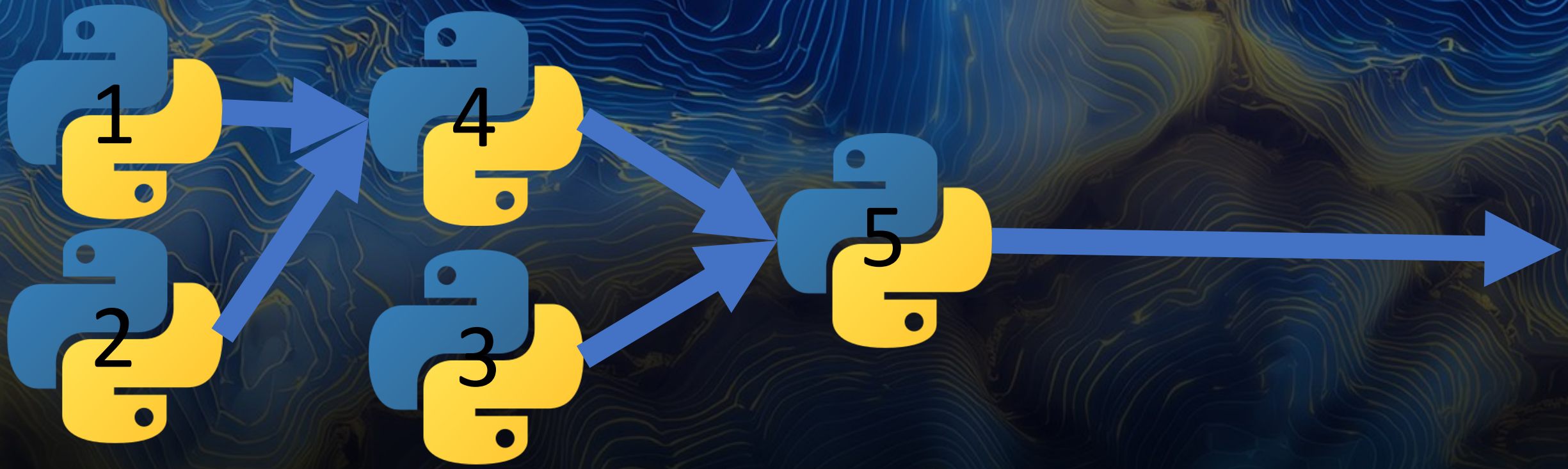
Workflow Plugin

- Composed of steps in an ordered dictionary
  - Each step names a plugin kind, a specific plugin name, and provides arguments to that plugin.
- Each step has a unique, arbitrary ID
  - Allows referencing steps for passing arguments from the CLI, constructing complex dependency graphs, producing useful monitoring metrics, and logging.
- Workflows can be used as steps in other workflows

```yaml
apiVersion: geoips/v2
kind: workflows
metadata:
  name: order_based_IR_BD
  docstring: |
    The IR_BD default configuration.
spec:
  steps:

    read_abi:
      kind: reader
      name: abi_netcdf
      arguments:
        vars: ["B01Ref", "B02Ref", "B03Re

    compute_true_color:
      kind: algorithm
      name: true_color

    interpolate_to_sector:
      kind: interpolator
      name: interp_nearest
      arguments:
        sector:
          kind: sector
          name: conus

    write_annotated_image:
      kind: output_formatter
      name: imagery_annotated
      arguments:
```




1
2
4
3
5

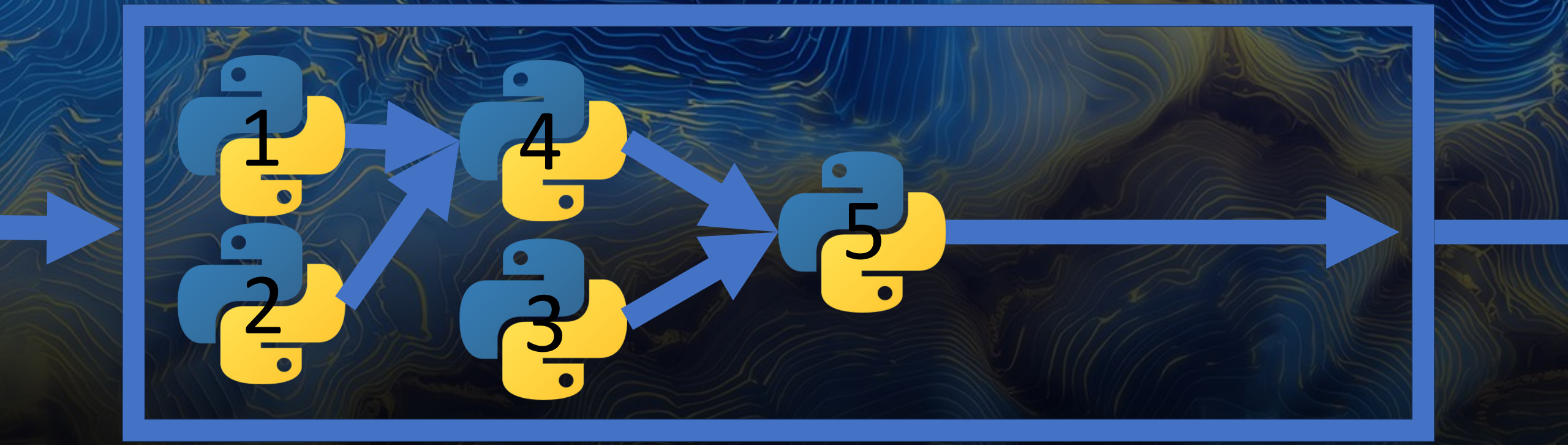


How do you run:

- In near real time
- Over batch historical datasets

Data in?

Data out?



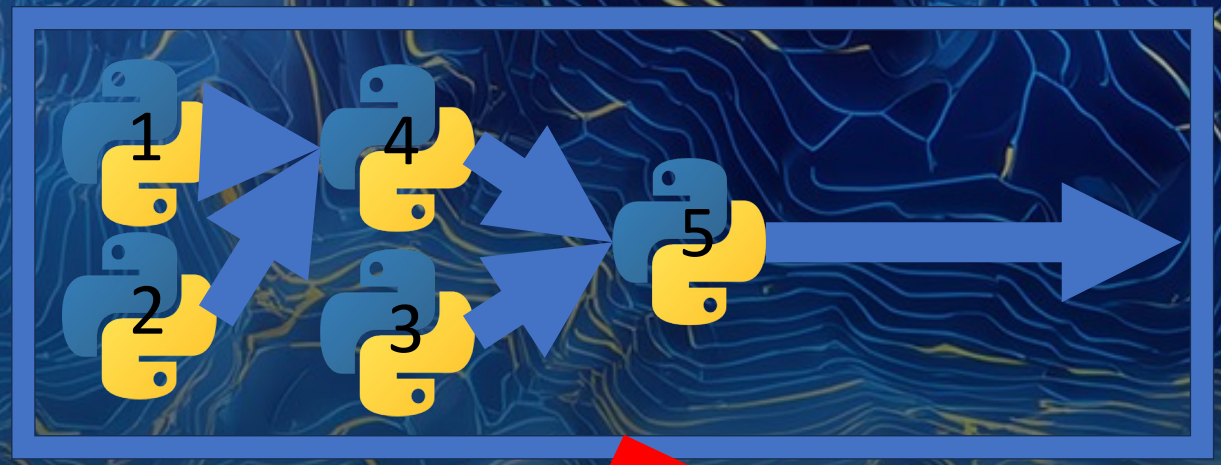
How do you run:

- In near real time
- Over batch historical datasets


Courier
A Plugin-based Orchestrator for Processing (Earth Observation) Data

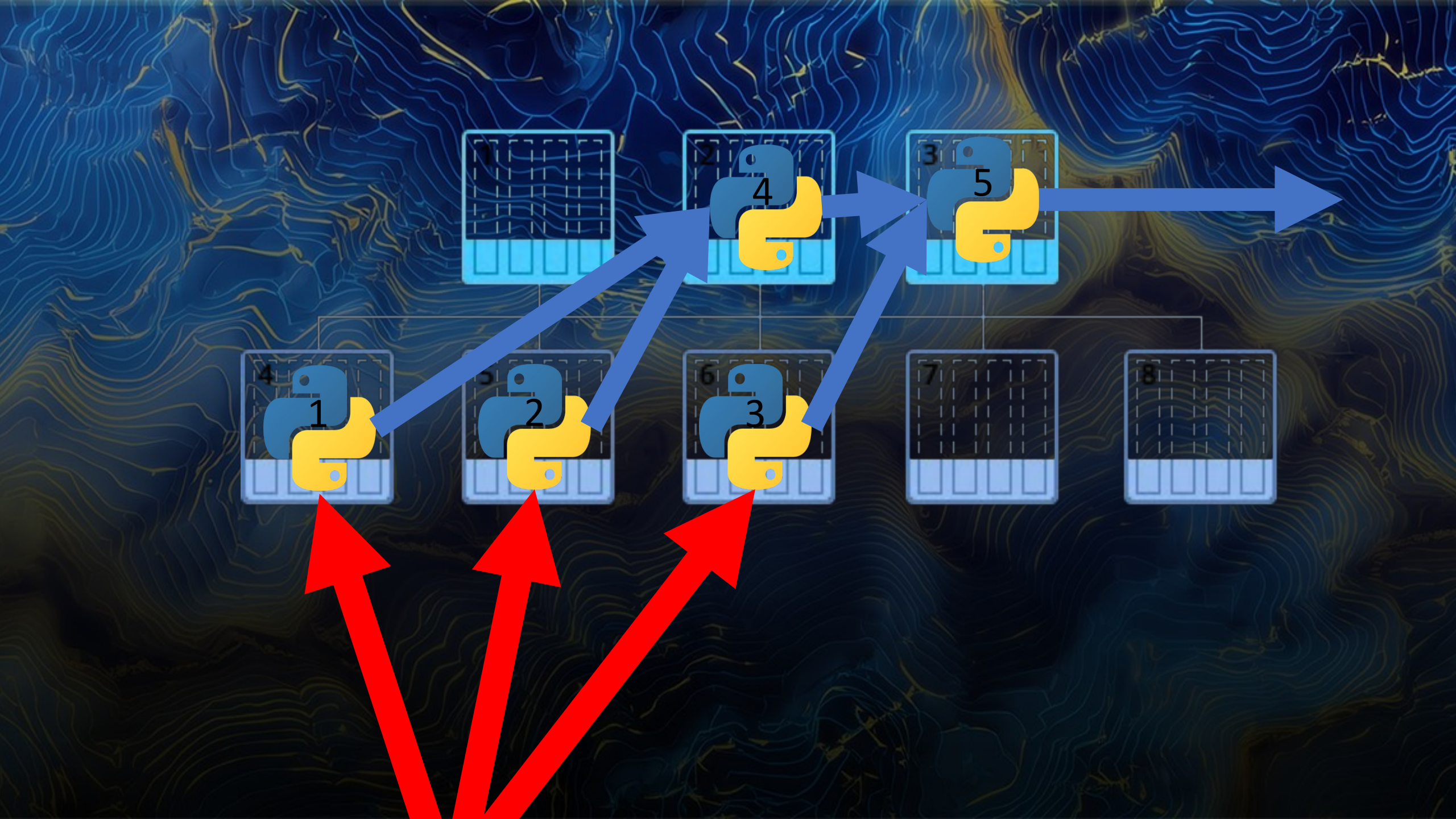

1
2
3
4
5

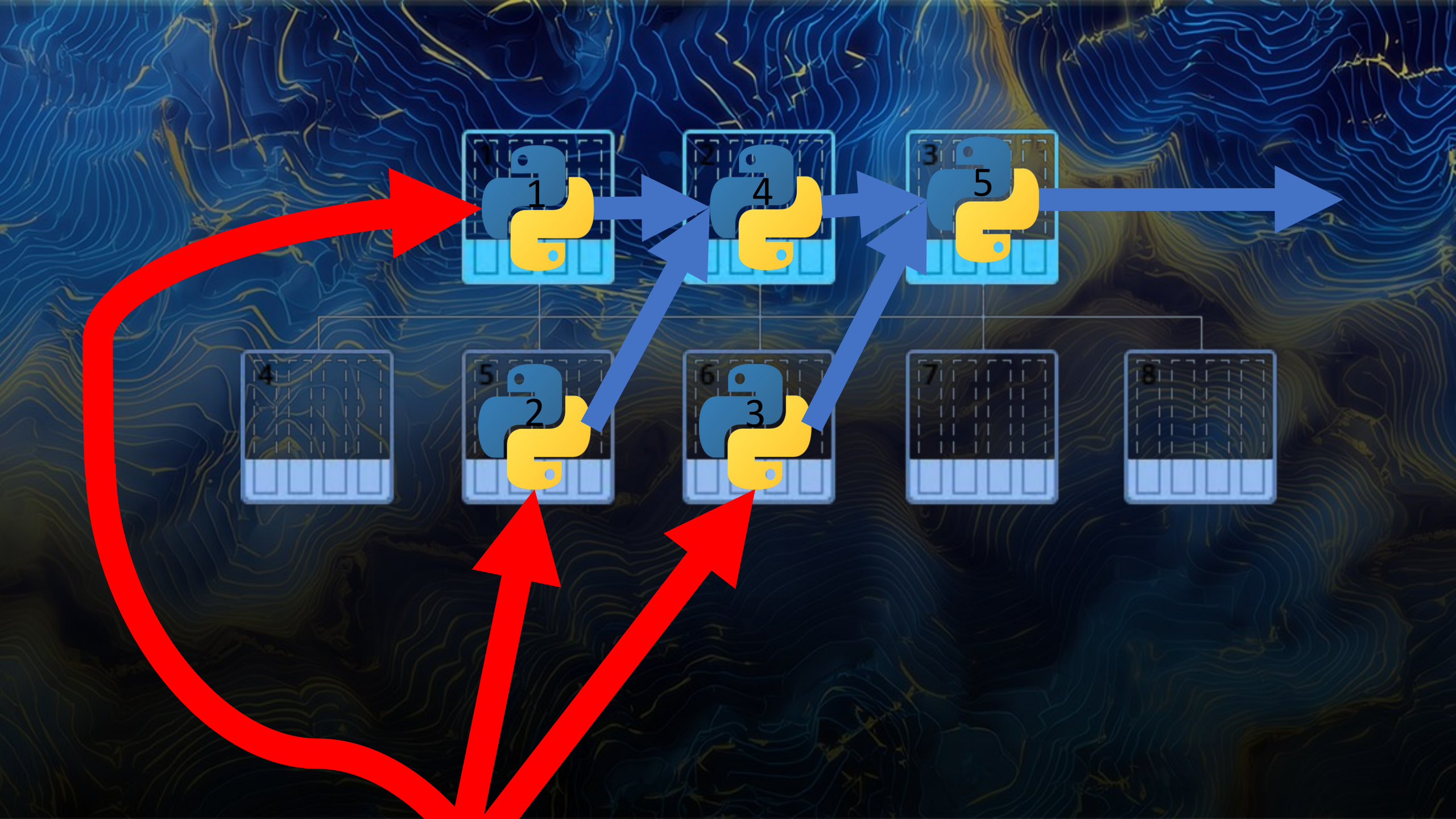

1


1
2
3
4
5


1
2
3
5
6
7
8


1
2
3
4
5


1
2
3
4
5
1
2
3
4
5
6
7
8


1
2
3
4
5
6
7
8
1
4
5
2
6
3

Courier is:

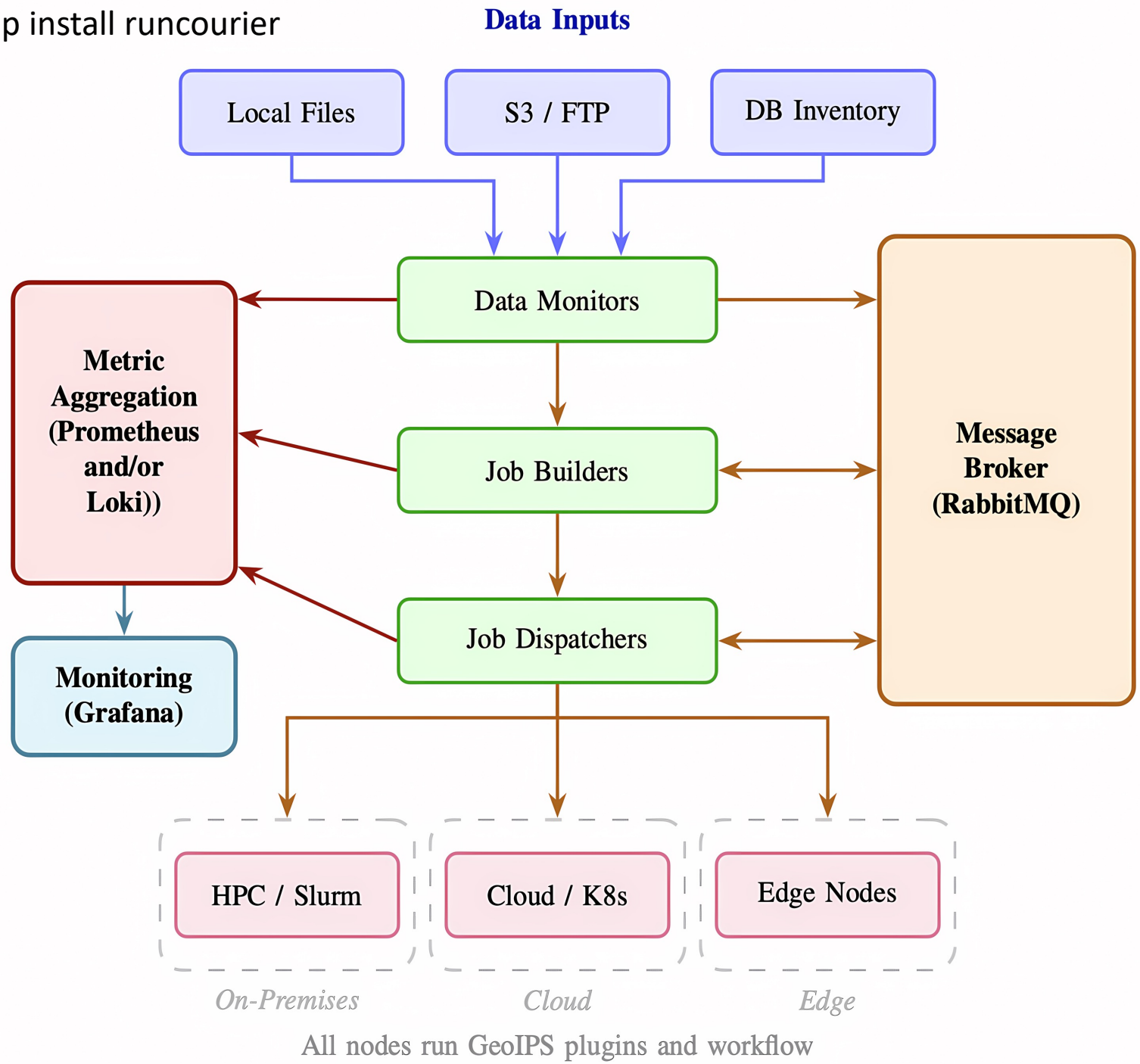
Python based, plugin-based, and event driven – Data monitors, Job Builders, Dispatchers with enforced inter-communication schema with support for various message brokers.

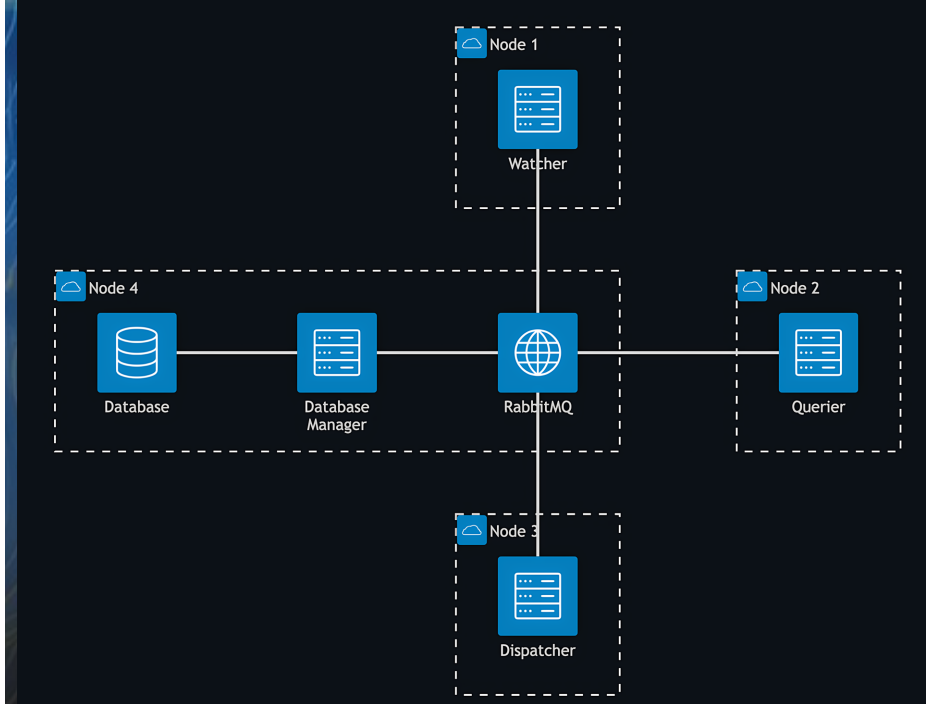
A way to create self-documenting data pipelines that maintain and create provenance metadata.

Data source, type, and handler agnostic, but built with an eye towards large datasets (eg..... geolocated data).

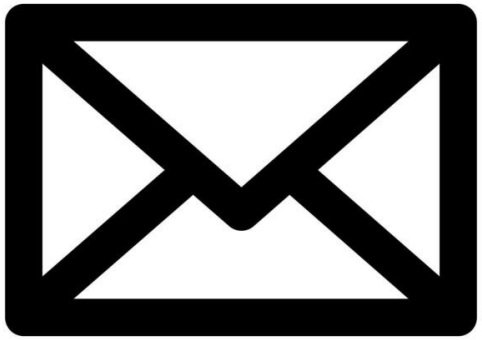
Deployable on a single laptop or hybrid-edge-cloud-on-prem multi-site distributed multi-cluster high-availability infrastructure(s).

pip install runcourier






```
10.21 Err:1 http://deb.debian.org/debian bullseye InRelease
10.?? Temporary failure resolving 'deb.debian.org'
10.21 Err:1 http://deb.debian.org/debian bullseye InRelease
10.?? Temporary failure resolving 'deb.debian.org'
10.21 Err:1 http://deb.debian.org/debian bullseye InRelease
10.?? Temporary failure resolving 'deb.debian.org'
10.21 Err:1 http://deb.debian.org/debian bullseye InRelease
10.21   Temporary failure resolving 'deb.debian.org'
25.23 Err:2 http://deb.debian.org/debian-security bullseye-security InRelease
25.23   Temporary failure resolving 'deb.debian.org'
40.24 Err:3 http://deb.debian.org/debian bullseye-updates InRelease
40.24   Temporary failure resolving 'deb.debian.org'
```



05:00 [Alerting] Test notification

[Alerting] Test notification

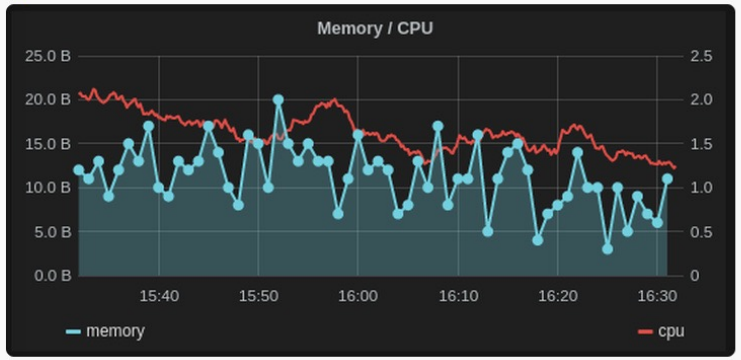
Someone is testing the alert notification within grafana.

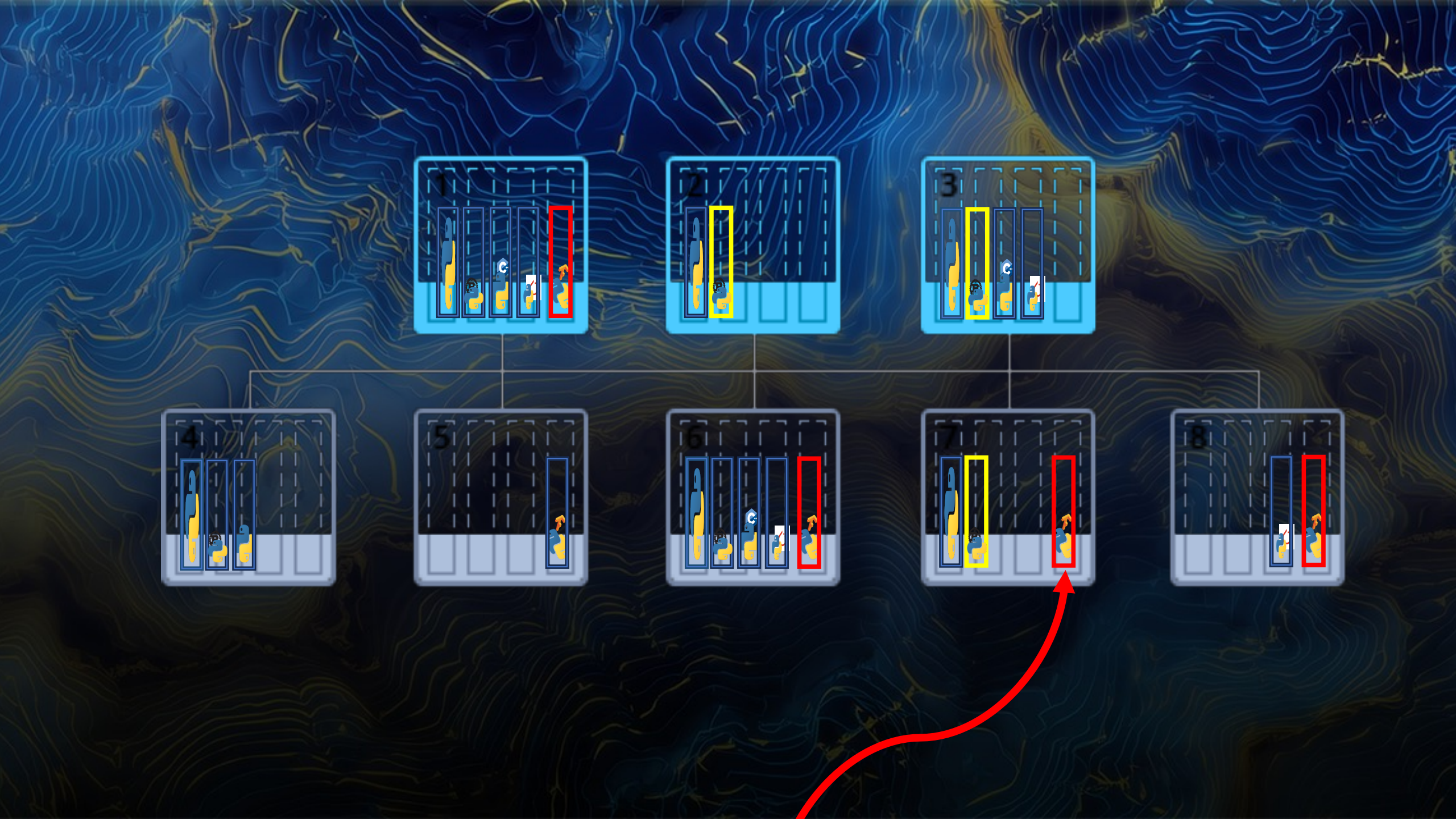
High value
100

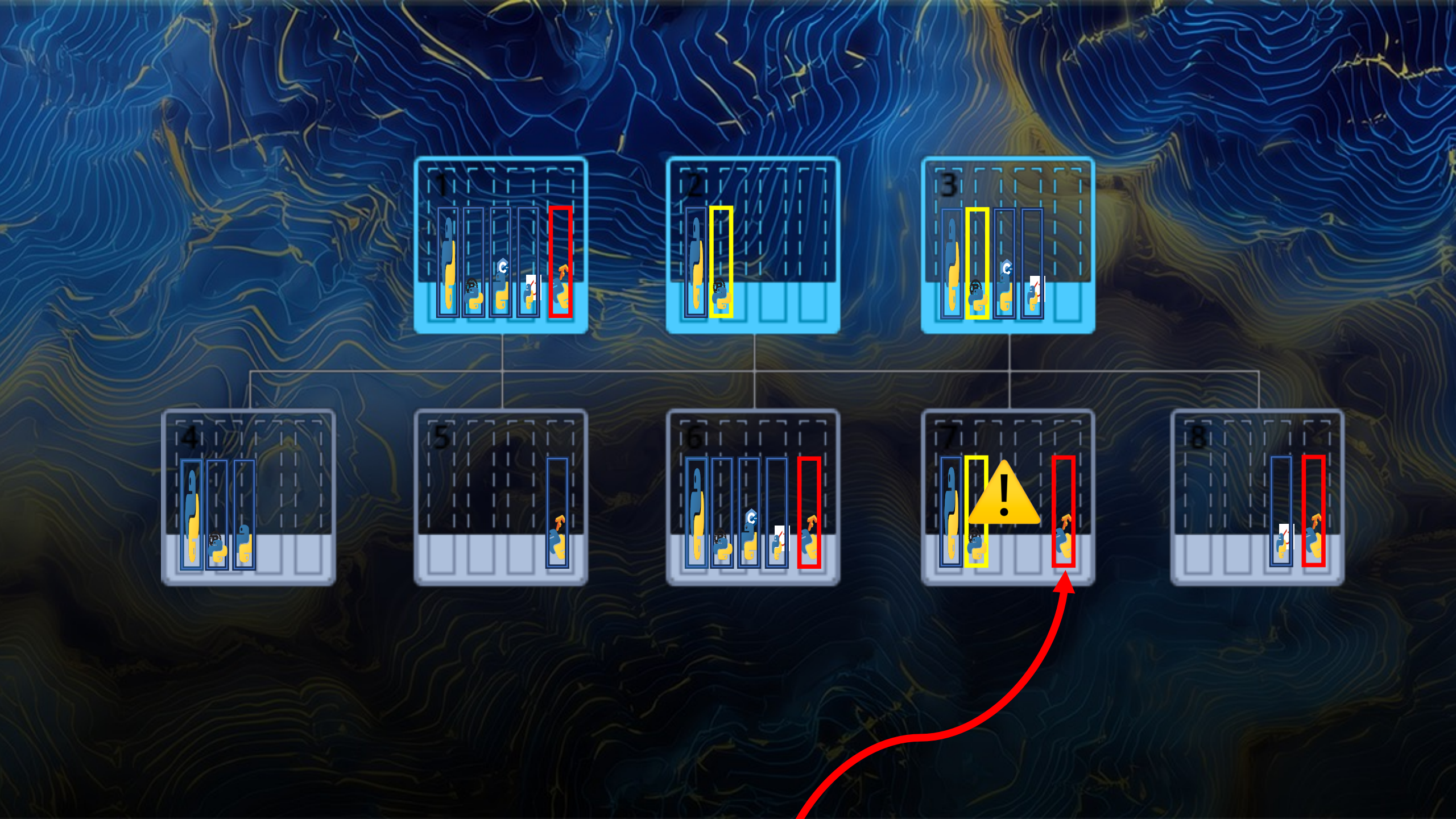
Higher Value
200

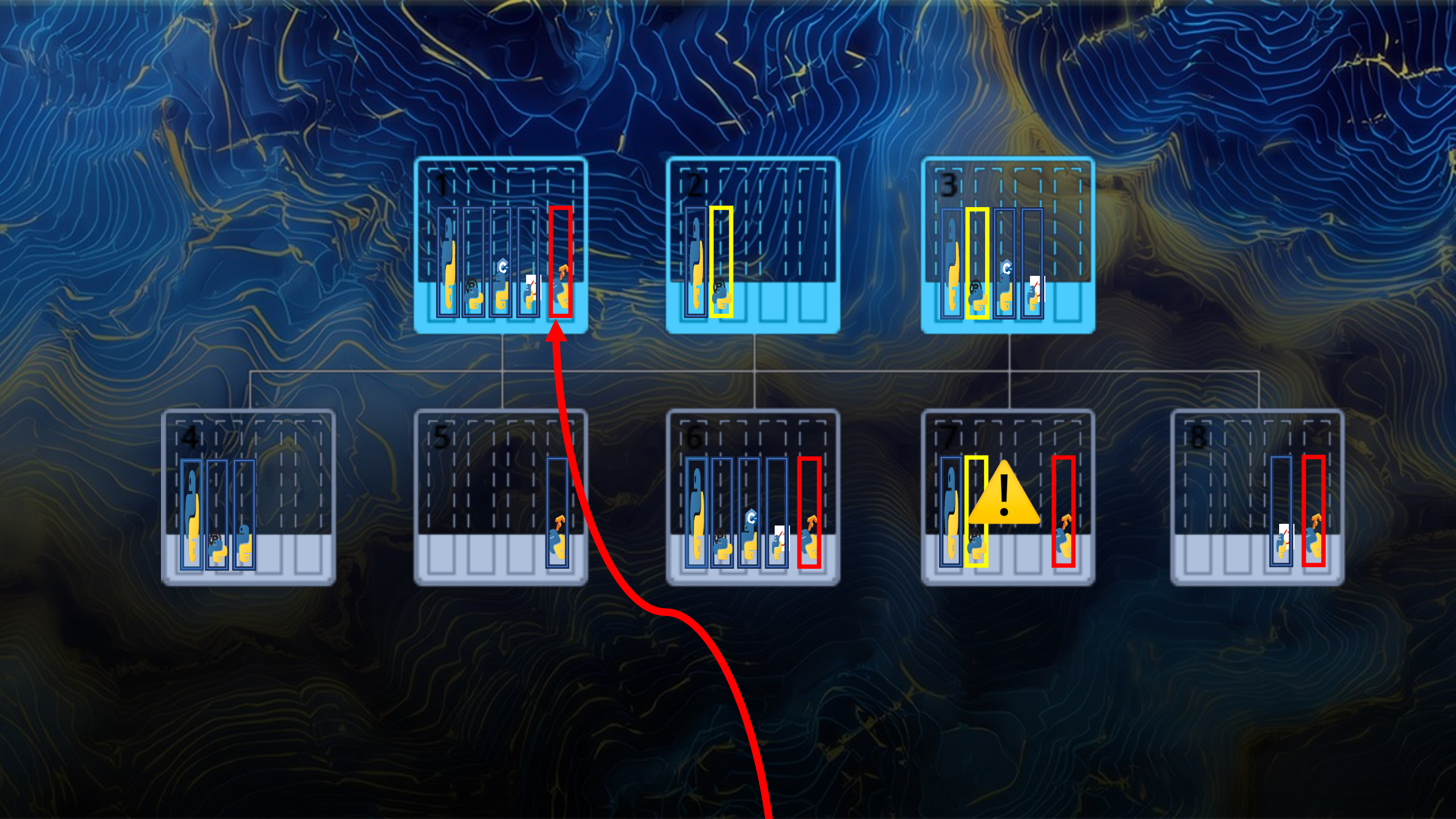
Error message
This is only a test

Grafana v6.7.1 | Today at 05:00 (30 kB)




1
2
3
4
5
6
7
8


1
2
3
4
5
6
7
8


1
2
3
4
5
6
7
8

● Healthy	113h 52m	10.6s	6,209
Health	Uptime	Heartbeat Age	Files Processed

Data Monitors

Name	Files	Success	Fail	Scan Age	Avg Dur
rabbit_mq_watcher	6,209	6,209	0	0.0s	0.000s

Job Builders

Name	Files	Jobs	Succ%	Active	Disc	Dur	FPJ
filter_and_group	6,209	696	100%	1	0	0.003s	1.0

Dispatchers

Name	Jobs	Succ%	Active	Dur	Logs	Q Wait
serial_bash	696	100%	0	24.320s	696	4.873s

Plugins

Name	State	Health	Restarts
filter_and_group	RUNNING	●	0

Broker

● Connected | Attempts: 1 | Sent: 7,601/s | Recv: 6,905/s

Routing

Identifier	Consumed	Latency	Q Depth
dispatcher-serial-bash	696	4.872s	0

State Sync / HA

Pushes: 0/s | Applies: 0/s | Claims: 0/s | Errors: 0

Pipeline

Files Detected: 6,209/s
Jobs Built : 696/s
Jobs Dispatch: 696/s


Pipeline Throughput (end-to-end)
0.025 ops/s
0.02 ops/s
0.015 ops/s
0.01 ops/s
0.005 ops/s
0 ops/s
18:10
18:15
18:20
18:25
Total Files Processed Since Death
6206
File Processing Duration
10 ms
9 ms
8 ms
7 ms
6 ms
5 ms
17:50
18:00
18:10
18:20
18:30
18:40
p50 — filter_and_group
p95 — filter_and_group
p99 — filter_and_group
Jobs Processed Rate
0.003 ops/s
0.002 ops/s
0.001 ops/s
0 ops/s
17:45
18:00
18:15
18:30
serial_bash — success
Job Success Ratio
100%
Active Jobs
1
0.8
0.6
0.4
0.2
0
17:50
18:00
18:10
18:20
serial_bash
File Processing Rate
0.025 ops/s
0.02 ops/s
0.015 ops/s
0.01 ops/s
0.005 ops/s
0 ops/s
17:50
18:00
18:10
18:20
18:30
Execution Duration
1 min
50 s
40 s
30 s
20 s
17:50
18:00
18:10
18:20
18:30
18:40
p50 — serial_bash
p95 — serial_bash
p99 — serial_bash
Execution Logs Emitted
0.003 ops/s
0.002 ops/s
0.001 ops/s
0 ops/s
17:50
18:00
18:10
18:20
serial_bash

```yaml
apiVersion: runcourier.dev/v1alpha1
kind: Service
metadata:
  name: data-inventory-ingest
  namespace: data-inventory-ingest
  description: Watches for files from a data inventory notification.
  docstring: |
    This service listens to the nrt_file_notif_queue RabbitMQ queue for incoming
    files that were ingested by the CIRA data inventory. Each notification
    is parsed into a File object annotated with platform, sensor, and timestamp
    metadata. Downstream job builders and dispatchers can filter and act on these
    files as they arrive.
spec:
  run:
  - identifier: data-monitor-rabbit-mq-watcher
    spec:
      kind: data_monitor
      name: rabbit_mq_watcher
      config:
        rabbitmq_host: di_db_rabbitmq
  - identifier: job-builder-filter-and-group
    spec:
      kind: job_builder
      name: filter_and_group
      config:
        files_per_job: 1
        filters:
          source: goes-18
  - identifier: dispatcher-serial-bash
    spec:
      kind: dispatcher
      name: serial_bash
      config:
        bash_script: python3 /app/create-3d-cloud-from-goes18-clavrx.py "{{ files[0].file }}"
--output-dir /output --cache-dir /cache
```

```yaml
apiVersion: runcourier.dev/v1alpha1
kind: Service
metadata:
  name: data-inventory-ingest
  namespace: data-inventory-ingest
  description: Watches for files from a data inventory notification.
  docstring: |
    This service listens to the nrt_file_notif_queue RabbitMQ queue for incoming
    files that were ingested by the CIRA data inventory. Each notification
    is parsed into a File object annotated with platform, sensor, and timestamp
    metadata. Downstream job builders and dispatchers can filter and act on these
    files as they arrive.
spec:
  run:
  - identifier: data-monitor-rabbit-mq-watcher
    spec:
      kind: data_monitor
      name: rabbit_mq_watcher
      config:
        rabbitmq_host: di_db_rabbitmq
  - identifier: job-builder-filter-and-group
    spec:
      kind: job_builder
      name: filter_and_group
```

```
    files that were ingested by the CIRA data inventory. Each notification
    is parsed into a File object annotated with platform, sensor, and timestamp
    metadata. Downstream job builders and dispatchers can filter and act on these
    files as they arrive.
spec:
  run:
  - identifier: data-monitor-rabbit-mq-watcher
    spec:
      kind: data_monitor
      name: rabbit_mq_watcher
      config:
        rabbitmq_host: di_db_rabbitmq
  - identifier: job-builder-filter-and-group
    spec:
      kind: job_builder
      name: filter_and_group
      config:
        files_per_job: 1
        filters:
          source: goes-18
  - identifier: dispatcher-serial-bash
    spec:
      kind: dispatcher
      name: serial_bash
      config:
        bash_script: python3 /app/create-3d-cloud-from-goes18-clavrx.py "{{ files[0].file }}"
--output-dir /output --cache-dir /cache
```

```
    files that were ingested by the CIRA data inventory. Each notification
    is parsed into a File object annotated with platform, sensor, and timestamp
    metadata. Downstream job builders and dispatchers can filter and act on these
    files as they arrive.
spec:
  run:
    - identifier: data-monitor-rabbit-mq-watcher
      spec:
        kind: data_monitor
        name: rabbit_mq_watcher
        config:
          rabbitmq_host: di_db_rabbitmq
    - identifier: job-builder-filter-and-group
      spec:
        kind: job_builder
        name: filter_and_group
        config:
          files_per_job: 1
          filters:
            source: goes-18
    - identifier: dispatcher-serial-bash
      spec:
        kind: dispatcher
        name: serial_bash
        config:
          bash_script: python3 /app/create-3d-cloud-from-goes18-clavrx.py "{{ files[0].file }}"
--output-dir /output --cache-dir /cache
```

```
    files that were ingested by the CIRA data inventory. Each notification
    is parsed into a File object annotated with platform, sensor, and timestamp
    metadata. Downstream job builders and dispatchers can filter and act on these
    files as they arrive.
spec:
  run:
  - identifier: data-monitor-rabbit-mq-watcher
    spec:
      kind: data_monitor
      name: rabbit_mq_watcher
      config:
        rabbitmq_host: di_db_rabbitmq
  - identifier: job-builder-filter-and-group
    spec:
      kind: job_builder
      name: filter_and_group
      config:
        files_per_job: 1
        filters:
          source: goes-18
    spec:
      kind: dispatcher
      name: serial_bash
      config:
        bash_script: python3 /app/create-3d-cloud-from-goes18-clavrx.py "{{ files[0].file }}"
--output-dir /output --cache-dir /cache
```

```
    files that were ingested by the CIRA data inventory. Each notification
    is parsed into a File object annotated with platform, sensor, and timestamp
    metadata. Downstream job builders and dispatchers can filter and act on these
    files as they arrive.
spec:
  run:
  - identifier: data-monitor-rabbit-mq-watcher
    spec:
      kind: data_monitor
      name: rabbit_mq_watcher
      config:
        rabbitmq_host: di_db_rabbitmq
  - identifier: job-builder-filter-and-group
    spec:
      kind: job_builder
      name: filter_and_group
      config:
        files_per_job: 1
        filters:
          source: goes_18
  - identifier: dispatcher-serial-bash
    spec:
      kind: dispatcher
      name: serial_bash
      config:
        bash_script: python3 /app/create-3d-cloud-from-goes18-clavrx.py "{{ files[0].file }}"
--output-dir /output --cache-dir /cache
```

kubernetes

Courier: A Distributed, Plugin-Based Infrastructure for Scalable Earth Observation Data Processing

Gwyn Uttmark, Evan Rose, Jeremy E. Solbrig, James Fluke, John M. Haynes, Yoo-Jeong Noh, Steven D. Miller
Cooperative Institute for Research in the Atmosphere
Colorado State University
Fort Collins, CO, USA
{gwyn.uttmark, evan.rose, james.fluke, jeremy.solbrig, john.haynes, yoo-jeong.noh, steven.miller}@colostate.edu

Melinda L. Surratt, Christopher P. Camacho, Christopher Selman, Andrew Thorpe, Lance Wilson, Lauren Porter
U.S. Naval Research Laboratory
{melinda.surratt.civ, christopher.p.camacho2.civ, christopher.m.selman2.civ, lauren.n.porter.ctr}@us.navy.mil,
{andrew.thorpe, lance.wilson}@devineco.com

***Abstract*—Recent decades have witnessed a large increase in Earth observation (EO) data driven by technological advancements in satellite missions and sensors. The size and diversity of generated data create complications for researchers, which are further compounded when they transition their work to operational real-time products. In the EO community, this gap between research and deployment is often called the "valley**

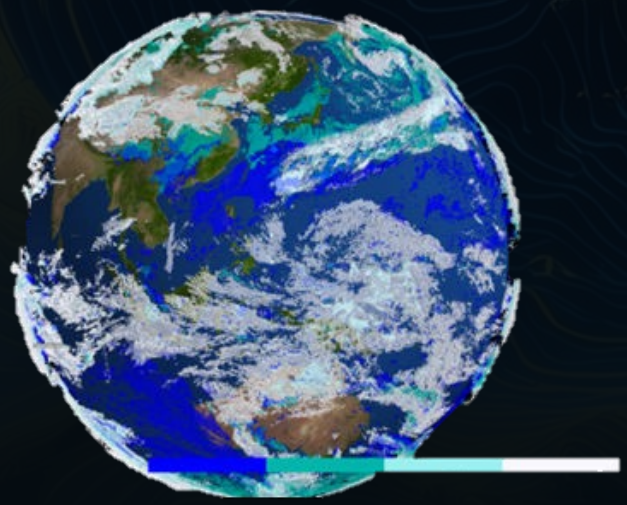
wide range of file formats (though this is changing with standardization) takes valuable time that could otherwise be spent investigating novel results. These complications are further compounded when researchers wish to use their results to effect change in the "real-world" with real-time data analysis and production. In the EO community, this gap between

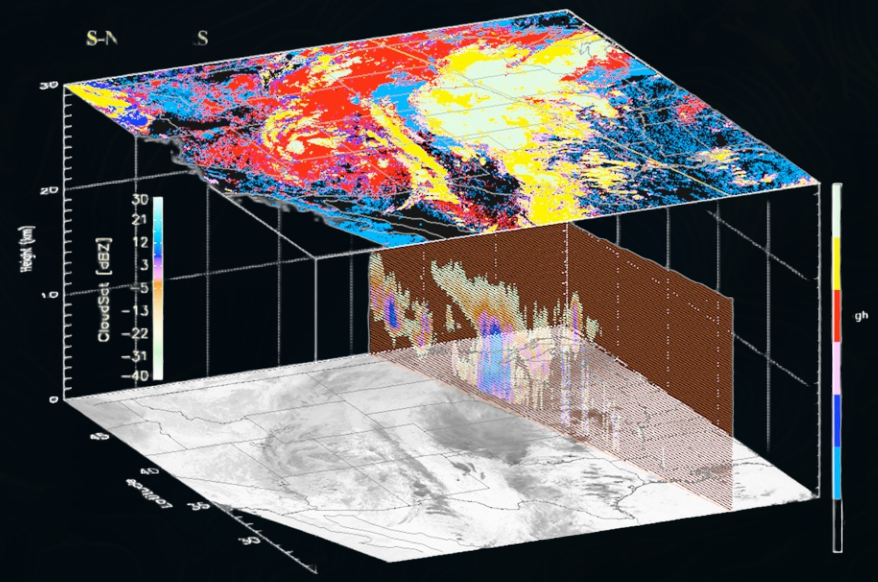
OVERCAST

Optical Variability Evaluation of Regional Cloud Asymmetries in Space and Time

Stitches:

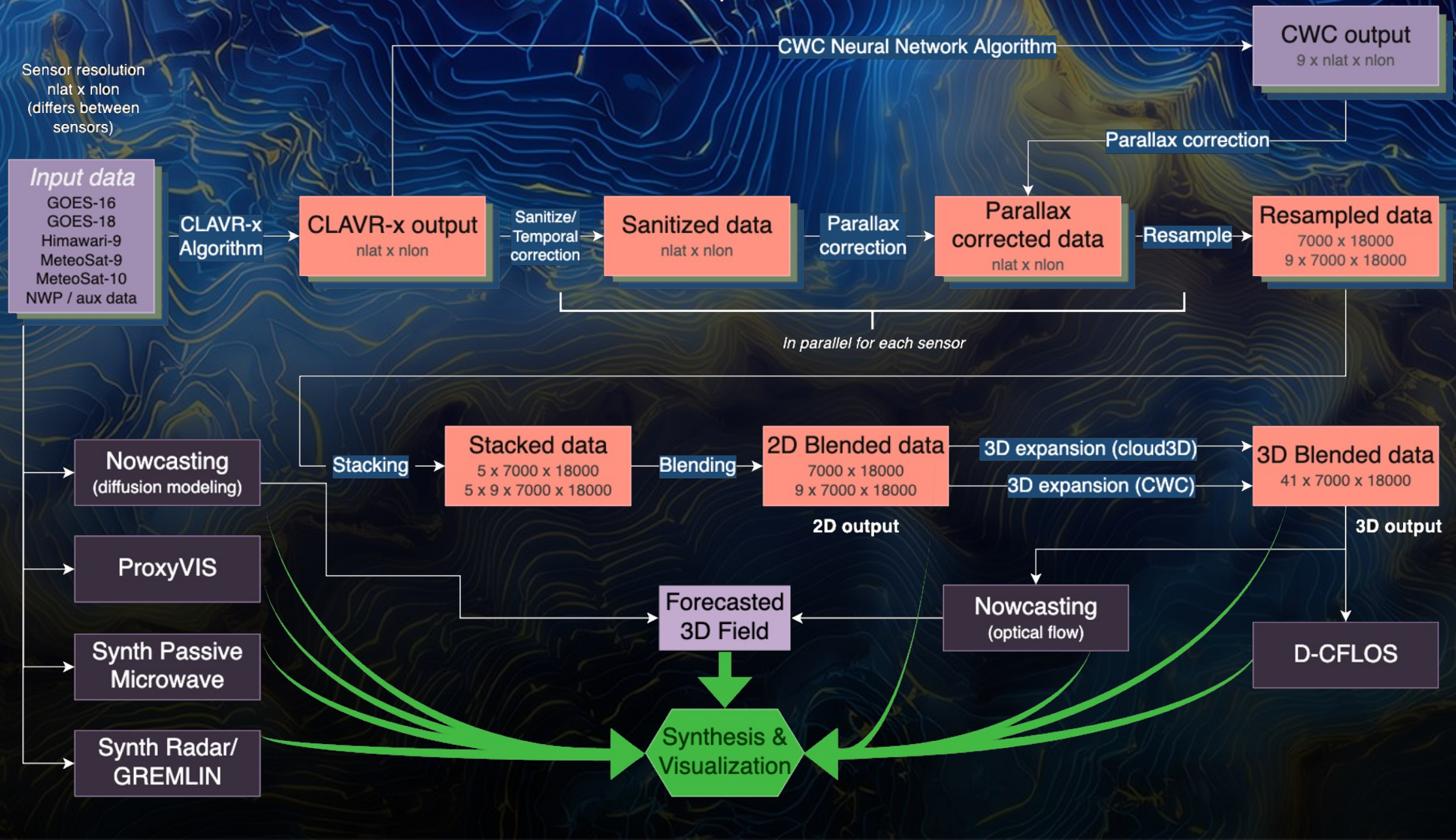
- 5+ Geostationary Images
- Multiple Low Earth Orbit Images
- Derivative ML Products





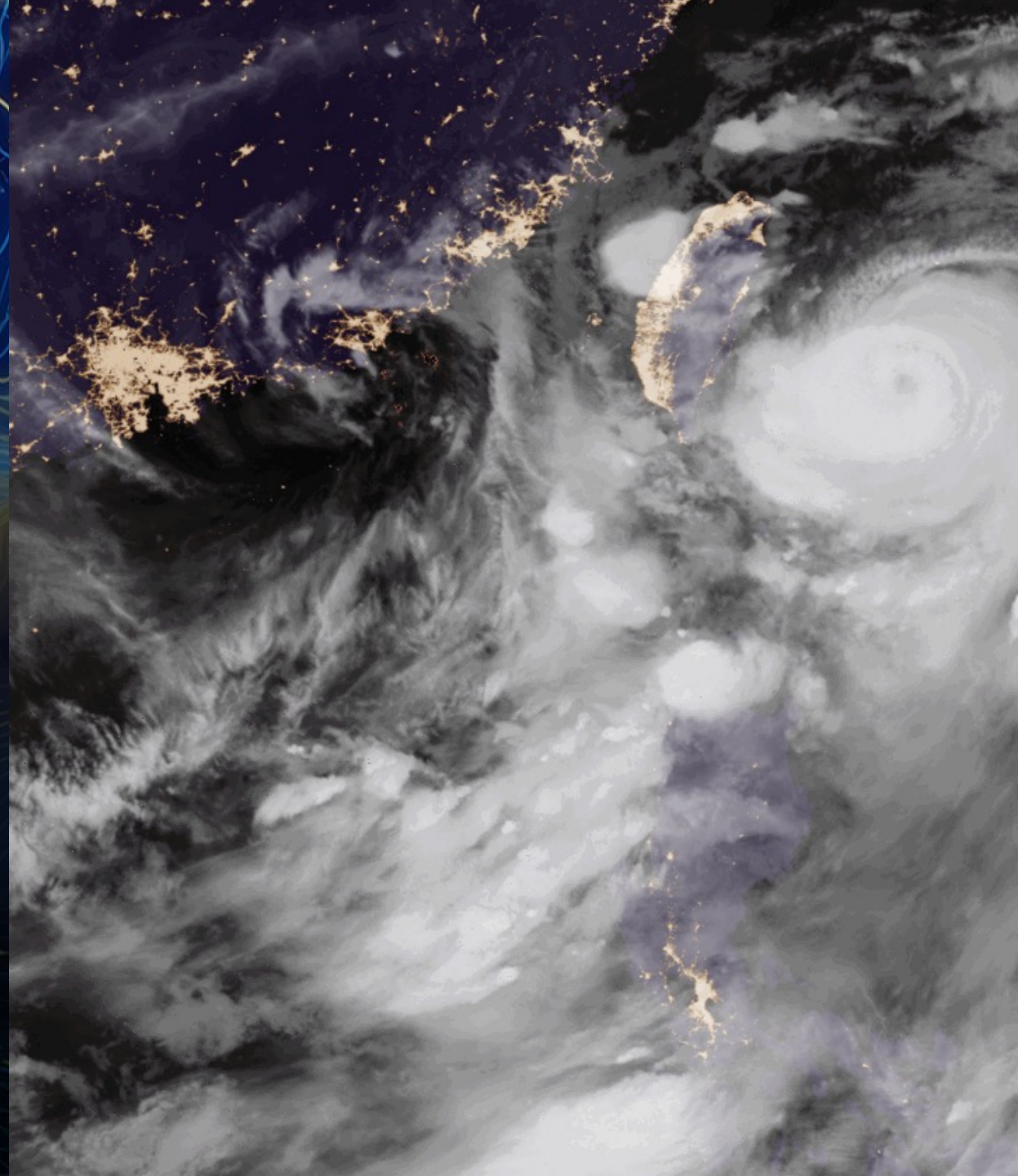
OVERCAST Workflow for 3D Clouds

with additional products shown



GeoIPS® + Courier

Find them on GitHub!
Free and Open-Source!



5th Annual GeoIPS® Users' Workshop

July 23rd & 24th

Cooperative Institute for Research in the Atmosphere
Colorado State University
Fort Collins, CO

Webpage

Registration

Workshop Webpage
https://www.cira.colostate.edu/conferences/geoips-workshop-2026/

Contact
jeremy.solbrig@colostate.edu