

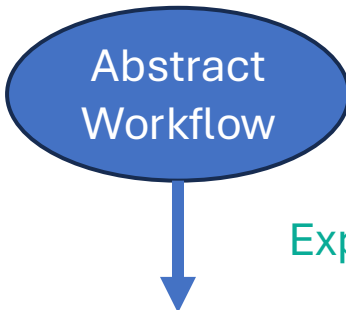
Pegasus and AI: Use a Napkin, Run a Workflow

Ewa Deelman

University of Southern California
Information Sciences Institute

deelman@isi.edu



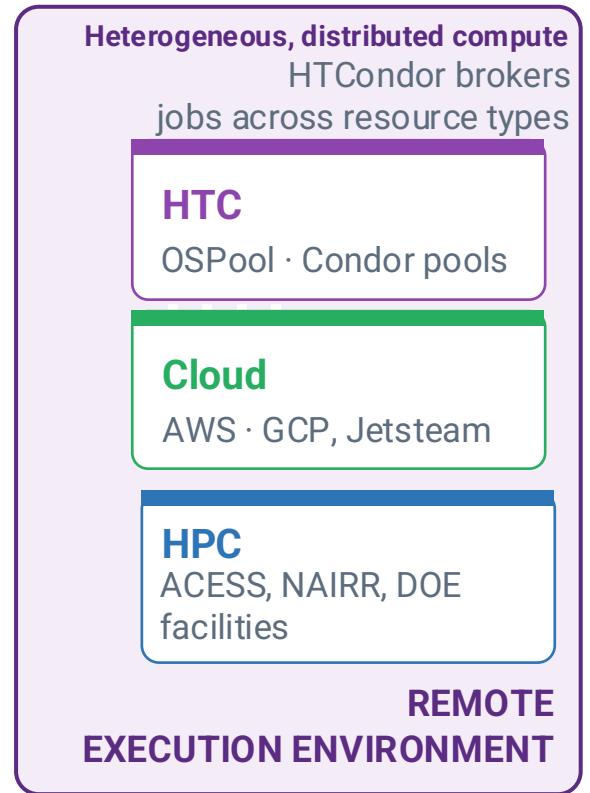
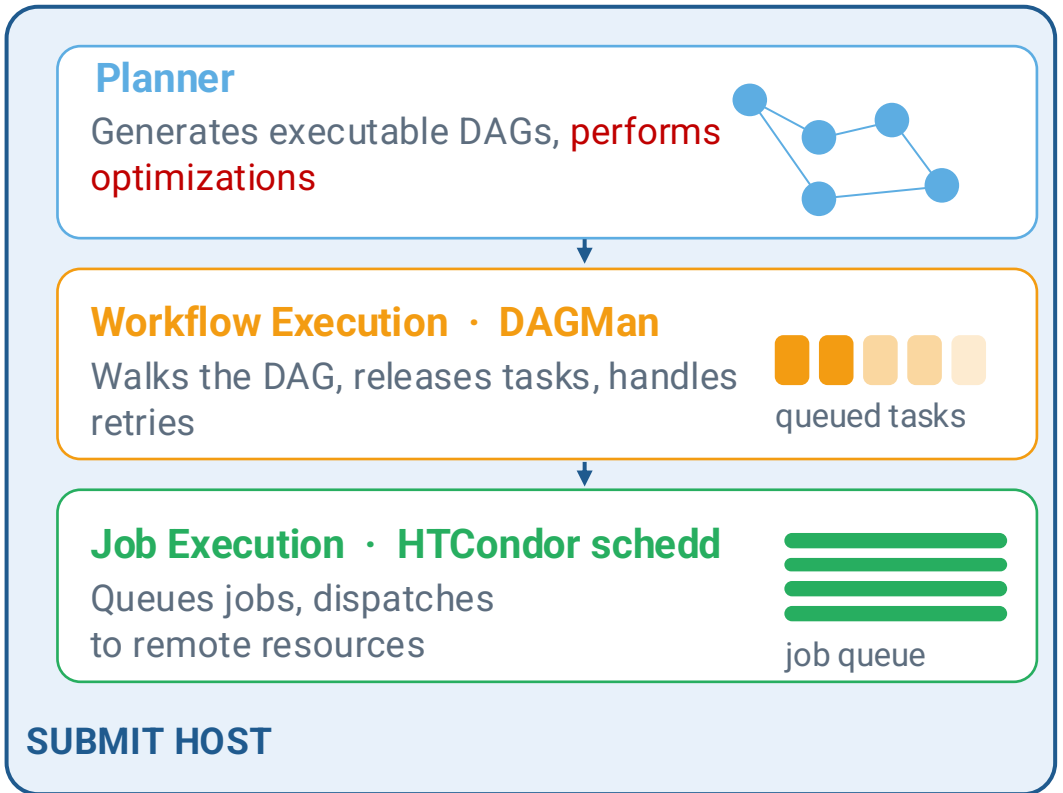


Pegasus Workflow Management System

Est. 2001



Explicit dataflow, portable, reproducible




Pegasus

- Infers data transfers
- Task clustering for small tasks
- Workflow level checkpointing
- Data reuse
- Data footprint reduction
- Monitoring Dashboard

SUBMIT HOST RUNS ON
same architecture, anywhere

 **Your laptop**

 **Hosted submit node**

 **Open OnDemand**





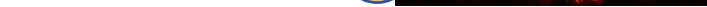
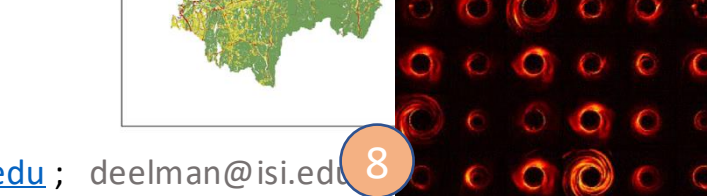
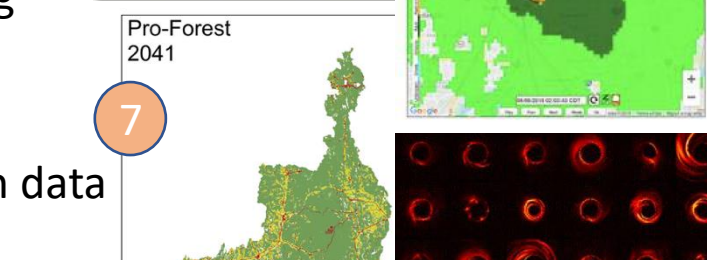
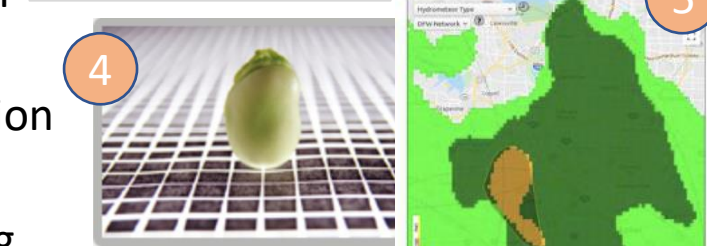
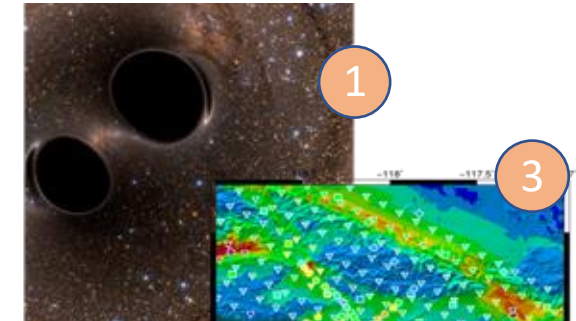
Scientific Impact of the Pegasus Workflow Management System, Ewa Deelman, PI

Selected Examples



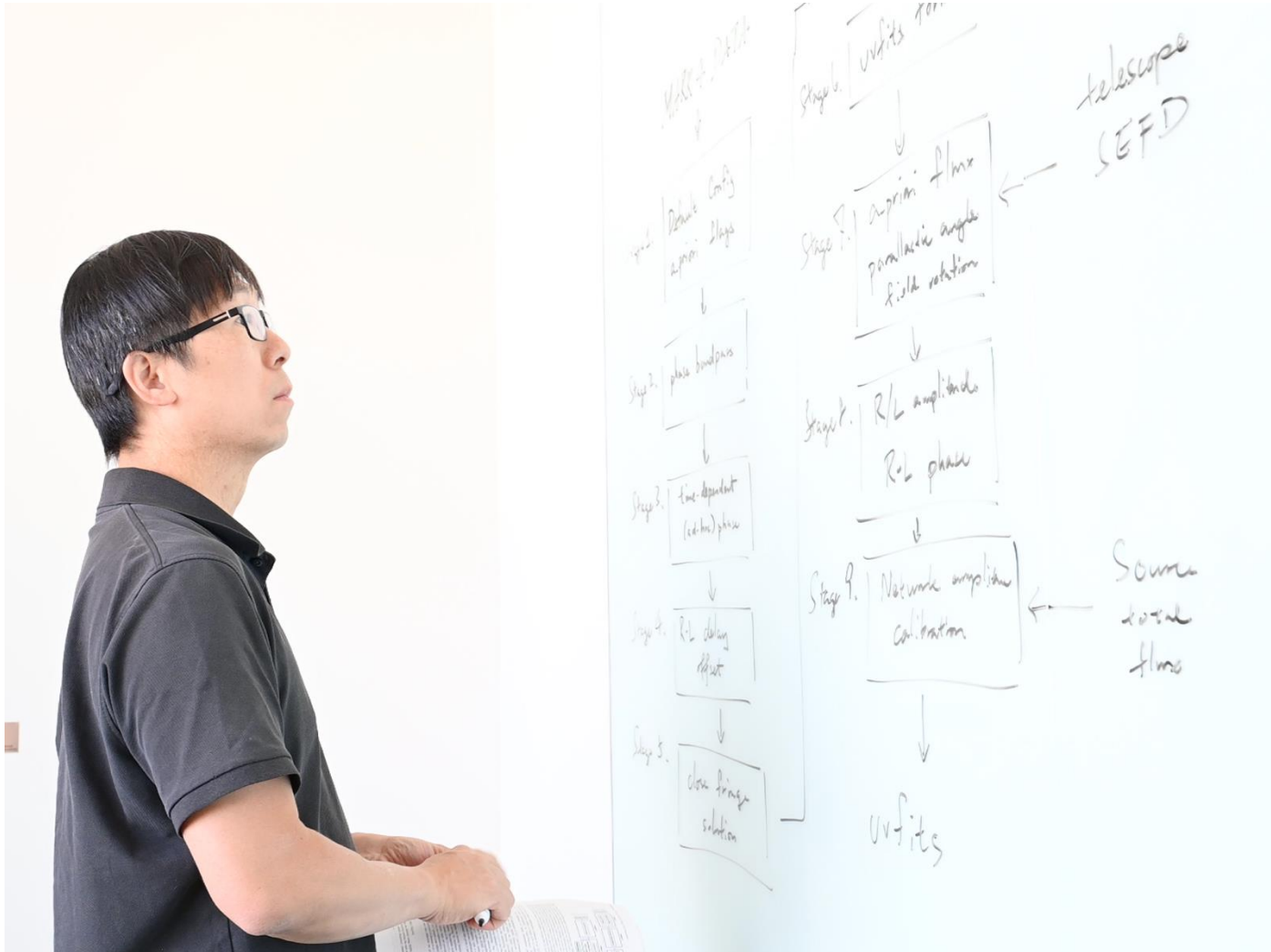
University of Southern California

1. Gravitational Wave Physics: **LIGO** – search for gravitational waves from compact-object binary coalescences, part of first direct detection of a gravitational wave, *collaboration 2001* -
2. Dark Matter Physics: **XENONnT** – used for Monte Carlo simulation and main data processing
3. Earthquake Science: **SCEC** – used to generate the first ever physics-based probabilistic hazard map "shake map" of S. California, information needed to enhance building codes in SoCal
4. Plant genomics: **SoyKB** – a community resource for "building a better soybean"
5. Weather modeling: **CASA** – prediction of hail and wind, used in drone flight path planning
6. Bacterial genomics: **ProkEvo** – supporting public health and food safety, used for outbreak detection, source tracking, understanding epidemics, public health surveillance, pathogen transmission, etc.
7. Socio-environmental systems: **Integrated Assessment Model @ U VT** – land use classification based on different economic and policy scenarios
8. Understanding black holes: **Event Horizon Telescope (EHT)** – used for black hole modeling
9. Integrated in science gateways: **HUBzero** – providing workflow management/automation capabilities to a broad spectrum of applications and users
10. Real time instrument data processing: **CryoEM@USC** – used to give real time feedback on data quality during data acquisition leading to higher quality results



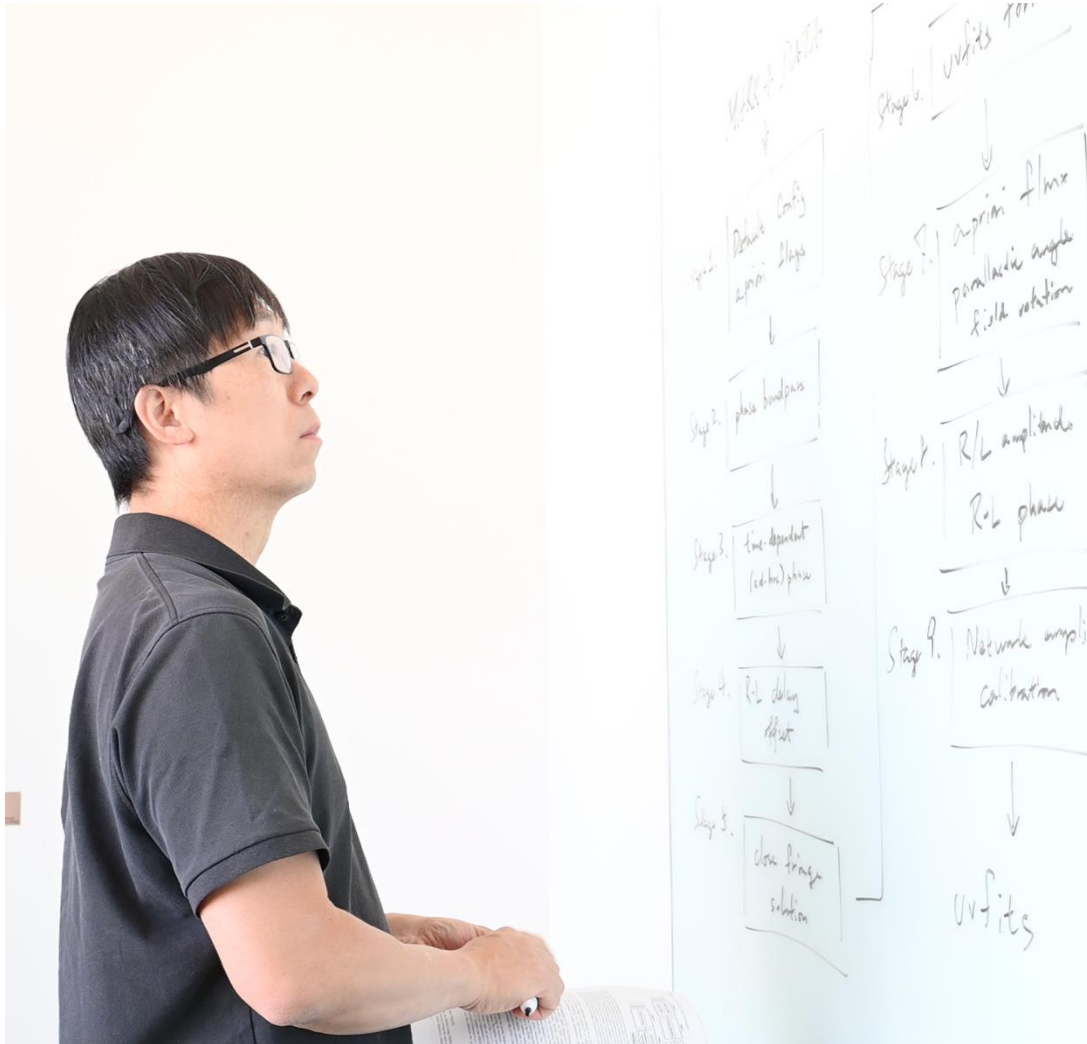
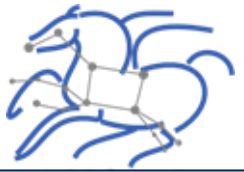
Pegasus and its workflows are used in Computer Science to explore workflow scheduling, provenance, resource provisioning, use of clouds for science, computing at the edge, etc.

Cognitive Disconnect



CK Chan, Associate Astronomer, Steward Observatory
University of Arizona

Cognitive Disconnect



```

class EHTSmili:
    def _load_rc(self):

        self.rc.add_replica("local", self.colormap.name, self.colormap.resolve())

    def _generate(self):
        process = Transformation(
            "smili_imaging_pipeline",
            site="condorpool",
            pfn=self.scripts_dir / "smili_imaging_pipeline.py",
            is_stageable=True,
            container="smili",
        )
        post_process = Transformation(
            "smili_postprocessing",
            site="condorpool",
            pfn=self.scripts_dir / "smili_postprocessing.py",
            is_stageable=True,
            container="smili",
        )

        self.tc.add_transformations(process, post_process)

        pattern = re.compile(r".*_([0-9]{3})_.*")
        for f in self.uvfitsfiles:
            obsdate = pattern.match(f.name).group(1)
            obsdate = int(obsdate) - 90
    
```



AI Can Accelerate Workflow Creation



Komal Thareja
RENCI

AI is becoming a practical partner
for designing and building scientific workflows.

Faster Start

Skip the blank page
— get a working
scaffold in minutes

Fewer Errors

Catch missing inputs,
dependencies, and
configs early

More Accessible

Lower the barrier for
non-experts to build
workflows

Enables Small Team Science

Can greatly increase
productivity

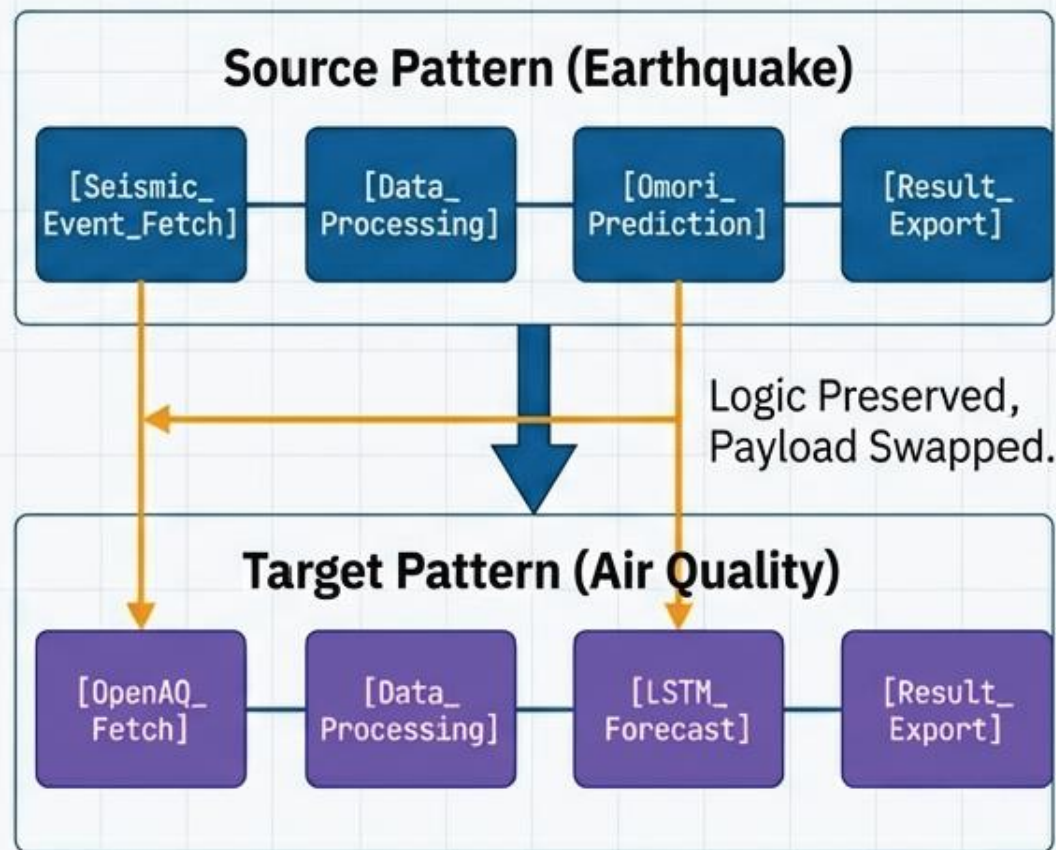


Prompting Strategy: Map Intent to Structure

Grounding the AI in existing patterns prevents hallucinations.

```
Prompt Engineering x  
> USER: Here is an example Pegasus workflow for earthquake analysis. Adapt this pattern to my new use case for analyzing air quality data.  
  
> SYSTEM: Acknowledged. Retaining dependency logic. Swapping 'Seismic_Event_Fetch' for 'OpenAQ_Fetch'. Swapping 'Omori_Prediction' for 'LSTM_Forecast'.
```

Structural Mapping





Case Study: Soil Moisture Analysis (Edge-to-Cloud Pattern)

A Pegasus workflow for precision agriculture answering: "Should I water my crops?"

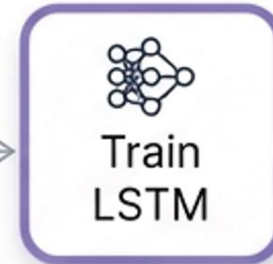


Edge / DPU (Smart NIC)

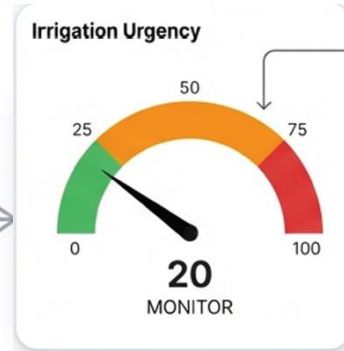


Input: Open-Meteo API (ERA5/Temp)

****Reduced Data Transfer****



24h History -> 24h Forecast



Validated with SAGE testbed soil moisture data

Technical Specs

Container

soilmoisture:latest (Multi-platform Docker)

Model

LSTM Neural Network (PyTorch) + Crop Threshold Rules

Output

Urgency Score (0-100) & Actionable Irrigation Recommendation

Code: github.com/pegasus-isi/soilmoisture-workflow

ChatGPT 5.5

ok, I am curious if you can go from a hand-drawn workflow schematic to a Pegasus workflow structure, this workflow is for analyzing images of the sun

Thought for 6s >

Yes — you absolutely can go from a hand-drawn workflow schematic like this to a **Pegasus workflow structure**. The key is to translate:

- **boxes** → files (artifacts)
- **ovals** → transformations (jobs)
- **fan-out labels like N, M** → parameterized job loops
- **arrows** → dependencies (inputs/outputs)

Below is my best read of your diagram and a concrete Pegasus “starter structure” you could implement.



What your sketch appears to encode

Inputs

- `fd_v_1` (a solar image file / dataset snapshot)
- `fd_v_2` (another dataset/version/parameter set; looks like an alternate input branch)
- A **tile coordinate list**: (i, ϕ_i) for $i = 1..N$ (your leftmost box)

Step 1: `make_tiles (N)`

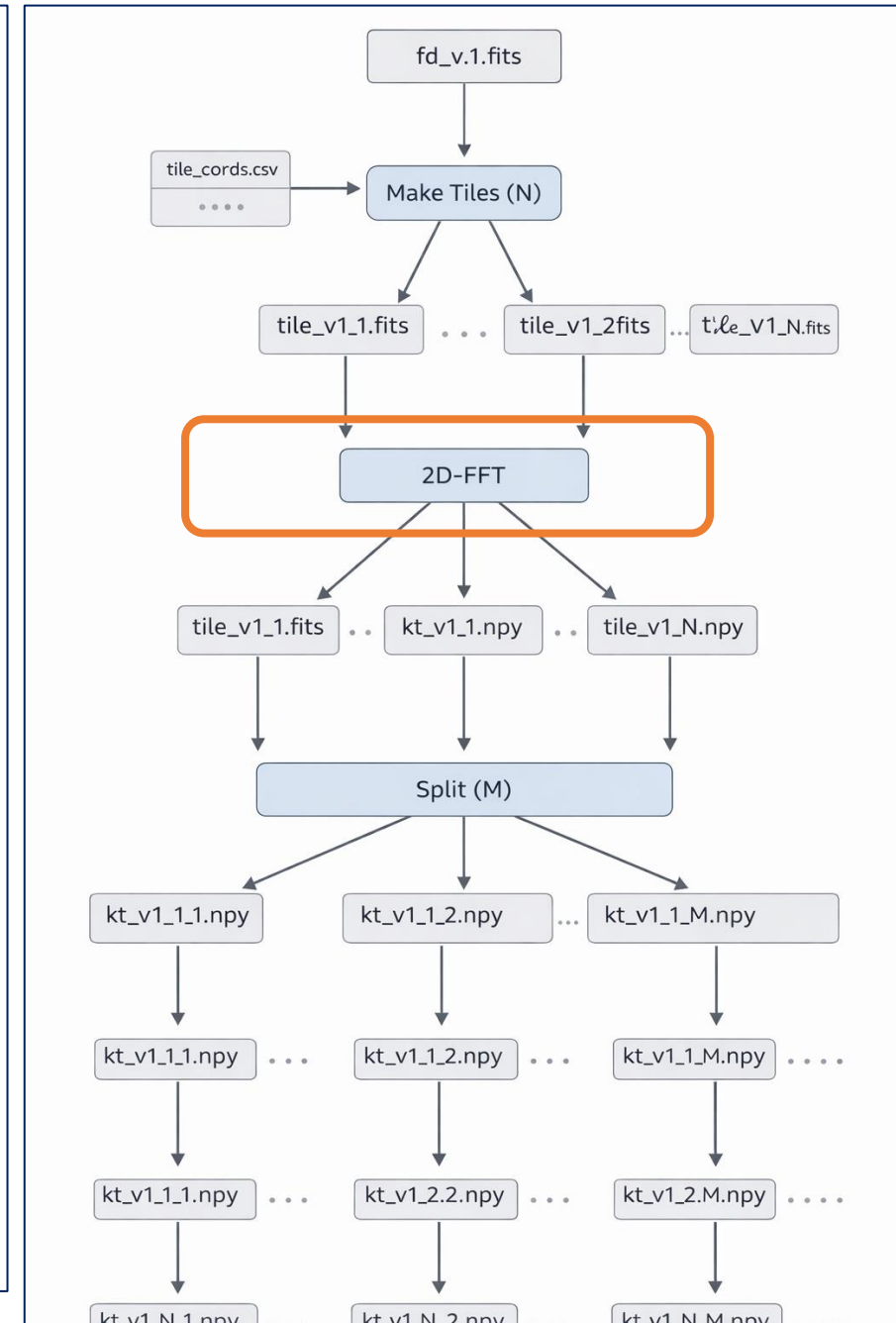
- Takes `fd_v_1` + the (i, ϕ_i) list
- Produces **N tiles**:
 - `tile_v1_1 ... tile_v1_N`

Step 2: `2D-fft`

- Runs per tile (at least shown for `tile_v1_1`, but implied for all)
- Produces per-tile FFT result:
 - `kt_v1_1 ... kt_v1_N`

Step 3: `split (M)`

- Takes each `kt_v1_i` and splits into **M partitions**:
 - `kt_v1_i_1 ... kt_v1_i_M`



But you need more to get things working on current CI

Taking a step back, what are the raw materials?

Scientists have their pipelines as...

A sentence

"I need a workflow that predicts soil moisture"

A napkin sketch

hand-drawn flowchart with boxes and arrows

A published paper

methodology section with pipeline stages

A Nextflow/Snakemake

existing pipeline in another WMS

Python scripts

working code in notebooks and .py files

A specification doc

requirements + dataset description

But Pegasus requires...

- Learning the Pegasus Python API
- Writing workflow_generator.py with DAG logic
- Building site, transformation, replica catalogs
- Creating standalone bin/ scripts per step
- Containerizing everything (Docker/Singularity)
- Configuring HTCondor job profiles
- Debugging data staging & runtime failures
- Writing docs and test infrastructure

→ *Weeks of effort for a novice developer*

The Methodology — Any Input → Pegasus Workflow

Prompt/Pattern

soilmoisture, airquality

Napkin Sketch

Napkin-vibe

Paper

Sea ice

Nextflow

mag, proteinfold, rnaseq

Snakemake

tnseq

Existing Code

s2-segmentation, sea ice

Spec + Datasets

medical-imaging-fl

Pegasus-AI Engine

Claude Code + MCP Skills

`/scaffold`

`/convert`

`/wrapper`

`/dockerfile`

`/review`

`/debug`

+ *indexed Pegasus docs* +
iterative debugging

Production-Ready Output

- `workflow_generator.py` — full Pegasus DAG with catalogs
- `bin/` scripts — standalone wrappers
- Docker/Singularity — pinned containers
- Site + Transformation + Replica catalogs
- README + run instructions

14

workflows built

~190

total prompts

~15

dev-days

~\$90

API costs

Pegasus Workflow Scaffold

You are a Pegasus workflow generator. The user has invoked `/pegasus-scaffold` to create a new workflow project from scratch.

Step 1: Read Reference Materials

1. Read `references/PEGASUS.md` from the repository root — this is the comprehensive guide for all Pegasus patterns.
2. Read `assets/templates/workflow_generator_template.py` — your starting point for the workflow generator.
3. Read `assets/templates/wrapper_template.py` and `assets/templates/wrapper_template.sh` — starting points for wrappers.
4. Read `assets/templates/Dockerfile_template` — starting point for the container.

Step 2: Gather Requirements

Ask the user the following questions. If they've already provided some answers in their message, skip those.

1. **Pipeline name:** What should the workflow be called? (e.g., "rnaseq", "weather-analysis")
2. **Pipeline steps:** Describe each step in order — what tool does it run, what are its inputs and outputs?
3. **Data source:** Where does input data come from?
 - Local files (FASTQ, CSV, etc.) — needs Replica Catalog entries
 - API fetch at runtime (USGS, OpenAQ, etc.) — first job fetches, no RC entries needed
 - Both (reference files + API data)



Step 3: Select Reference Workflow

Based on the user's answers, select the closest existing workflow as a reference pattern:

If the workflow has...	Study this example
Per-sample parallelism, fan-in merge	<code>examples/workflow_generator_tnseq.py</code>
API fetch + region loops	<code>examples/workflow_generator_earthquake.py</code>
Shell wrappers, micromamba, <code>--test mode</code>	<code>examples/workflow_generator_mag.py</code>
ML train-then-predict	<code>examples/workflow_generator_soilmoisture.py</code>
Dual pipeline, skip flags, multiple data sources	<code>examples/workflow_generator_airquality.py</code>
CondorIO for caches/databases, GPU jobs, batch inference	See Pegasus.md "Transferring Data Directories via CondorIO"

Read the selected reference workflow before generating code.

Step 4: Generate Files

Create the following files in `{pipeline-name}-workflow/`:

4a. `workflow_generator.py`

Start from `assets/templates/workflow_generator_template.py` and just

1. **Class name:** `{PipelineName}Workflow`
2. **`wf_name`:** `"{pipeline_name}"`
3. **`__init__`:** Add pipeline-specific parameters

Step 5: Validation Checklist

Before presenting the generated code to the user, verify:

- File I/O match:** Every `add_args()` filename matches a `File()` LFN, and the wrapper's `argparse` matches
- Dependency chain:** File objects are shared between producer/consumer jobs (not duplicated)
- `stage_out_strategy`:** Only final outputs have `stage_out=True`
- Unique job IDs:** No duplicate `_id` values across all jobs
- Replica Catalog completeness:** All local input files and support scripts are registered
- Wrapper `os.makedirs`:** Any output path with `/` has `os.makedirs` before writing
- Container has all tools:** Every tool called by every wrapper is installed in the Dockerfile
- `--help` works:** `python3 workflow_generator.py --help` would produce useful output
- No directory scanning:** No `glob()`, `os.listdir()`, or `list.files()` between jobs
- Support files use `os.getcwd()`:** Not `__file__`-relative paths

Spec-First — Plan, Then Build



Komal Thareja
RENCI



Two prompts beat continual prompting — ask for the design before any code.

THE TWO PROMPTS

PROMPT 1 — DESIGN

"Write a **specification.md** — stages, I/O, deps, containers. No code yet."

PROMPT 2 — BUILD

"Implement this spec as a Pegasus workflow." AI does the typing.

THE OUTCOME

PRODUCTION WORKFLOW

workflow_generator.py + bin/ +
containers + catalogs + README

WHY THIS WORKS

- **Why it works:** review intent in plain text — cheap to redirect; redirecting code is expensive
- **Shareable artifact:** spec is reproducible — share it or rerun later to get the same build
- **Proven in practice:** FL + sea-ice — 12-page specs, near-first-pass implementations
- **Takeaway:** human owns the design; AI does the implementation

Latest News

[View all news >](#)



Heading in the right direction: NSF
Cyberinfrastructure and Cybersecurity centers
work together on board RV Sikuliaq in Alaska

June 02, 2026



Webinar announced: AI-Powered Development
for Research Infrastructure featuring Komal
Thareja

May 01, 2026

Part II coming at the end of June

<https://ci-compass.org/about/contact/>



Debugging Pegasus Analyzer

Current version of analyzer pinpoints and highlight failures

Mats Rynga USC/ISI

```
2025-09-25 19:10:16,985 INFO: /bin/cp -f -R -L
'/home/rynga/ACCESS-Pegasus-Examples/04-Tutorial-
Debugging-Statistics/bin/llm-rag.py'
'/home/rynga/ACCESS-Pegasus-Examples/04-Tutorial-
Debugging-Statistics/scratch/rynga/pegasus/llm-rag-
books/20250925T190853+0000/./llm-rag.py'
2025-09-25 19:10:16,991 ERROR: Expected local file does
not exist: /home/rynga/ACCESS-Pegasus-Examples/04-
Tutorial-Debugging-
Statistics/inputs/Alices_Adventures_in_Wonderland_by_Le
wis_Carroll.txt
```

=====**Pegasus AI Analysis**=====

The workflow failed due to a missing input file. The job `stage_in_local_local_0_0` encountered an error:

```
**"Expected local file does not exist:
/path/to/Alices_Adventures.txt"**
```

Root Cause:

- The required input file is missing from the specified path.
- This prevents the transfer process from completing, causing the workflow to fail.

Next Steps:

1. Verify the file exists at the specified path.
2. Ensure the file path in the workflow configuration matches the actual location.
3. Resubmit the workflow after resolving the file issue.

The remaining unsubmitted jobs (7 total) likely depend on this staged file, so **fixing this error will enable further execution.**

Next major version of Pegasus will have AI augmentation



Concern:

From a software project into a service operator (infra, finops, SaaS)



Mats Ryngé
USC/ISI

SUBMIT HOST RUNS ON

same architecture, anywhere

 Your laptop

 Hosted submit node

 Open OnDemand

Today we ship software; users run it on their own compute, at no runtime cost to us.

Hosted AI changes that - we'd have to:

- Operate inference for pegasus-analyzer, pegasus-status, workflow development, and the chatbot
- Train and serve the anomaly + runtime models (training recurs)
- Provision, scale & pay for the AI/GPU infrastructure
- Own SaaS duties we don't have today: hosted services, monitoring, capacity planning, cybersecurity, and ongoing cost management (finops)

Free-tier users can bring their own models, shifting some cost - but we still fund the free tier, where most users will stay.



Where are we heading? -- Pegasus Studio




PegasusAI Studio

AI Workbench

Launch terminals and AI coding tools for workflow development.

- Workflows
- Workbench
- Notebooks ↗
- Settings
- QUICK ACCESS
- PegasusAI Chat
- Terminal

QUICK LAUNCH

 Bash Terminal Open a new shell session	 Claude Code Running Anthropic	 Codex CLI Running OpenAI
--	--	---

AI TOOLS

Claude Code running Anthropic AI coding assistant with plugin skills, MCP servers, and agentic tool use <input type="button" value="Open"/> <input type="button" value="Stop"/>	Codex CLI running OpenAI OpenAI's coding agent for terminal-based development <input type="button" value="Open"/> <input type="button" value="Stop"/>	OpenCode installed opencode-ai Open-source AI coding assistant with web UI, agents, and skills <input type="button" value="Start"/> <input type="button" value="Remove"/>
Antigravity CLI Google Google's agentic coding CLI powered by Gemini — subagents, tool use, and code execution <input type="button" value="Install"/>	PegasusAI Chat Built-in Built-in Pegasus workflow assistant with streaming chat and tool use <input type="button" value="Install"/>	JupyterLab Built-in JupyterLab with jupyter-ai, terminals, and collaborative editing <input type="button" value="Install"/>

PEGASUS SKILLS

Convert /convert Convert a Snakemake or Nextflow pipeline to a Pegasus workflow
Debug /debug Diagnose Pegasus workflow failures from error messages and logs

AGENT PERSONAS

General general General-purpose Pegasus WMS workflow assistant
._data-engineer ._data-engineer

Summary

- AI models make it very easy to create new science products, but **validation and verification are hard**
- We are very interested in **understanding how to create scientifically valid and trusted workflows with AI**
- We need to think about what tools can **help/are needed** to inspect outputs, AI models, processes
- **Deployment issues:** infrastructure, finops, SasS, and cybersecurity
- We want to collaborate on these issues

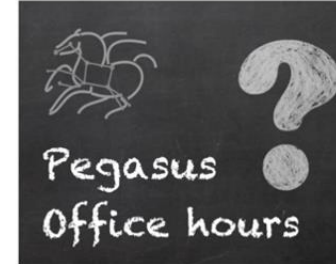
Pegasus: <http://pegasus.isi.edu>

- Slack channel
- Email: pegasus-support@isi.edu
- Office hours every Friday

Office Hours

Join the Pegasus team every Friday for virtual office hours at 11 AM Pacific / 2 PM Eastern.

Do you have questions about workflows or need guidance on organizing and implementing them? Join our weekly office hours – designed to support both new and experienced users in learning and engaging with Pegasus. Here's what to expect:



- Tutorial walkthrough First Friday of the month
- <http://pegasus.isi.edu>

deelman@isi.edu