

Before Tutorial Starts

Hands on Tutorial Exercises: Setup

- You need to be [on USC Network](#) and need to use your [USC credentials](#) to log in
- Use a web browser and log on to USC OnDemand Instance at
<https://ondemand.carc.usc.edu> .
- More details on how to logon to USC Open OnDemand can be found at
<https://www.carc.usc.edu/user-information/user-guides/hpc-basics/getting-started-on-demand>

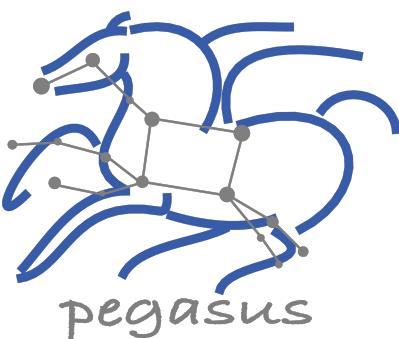
This is the same (but tailored for USC Clusters) as the self-guided tutorial available in the Pegasus documentation:

<https://pegasus.isi.edu/documentation/user-guide/tutorial.html>

Advanced Track Computing Bootcamp!

Discover the Power of High-Performance Computing at CARC!

Scientific Data Processing with Pegasus Workflows



Karan Vahi ¹, Mats Rynge ¹, Tomasz Osinski ²

¹Information Sciences Institute, University of Southern California

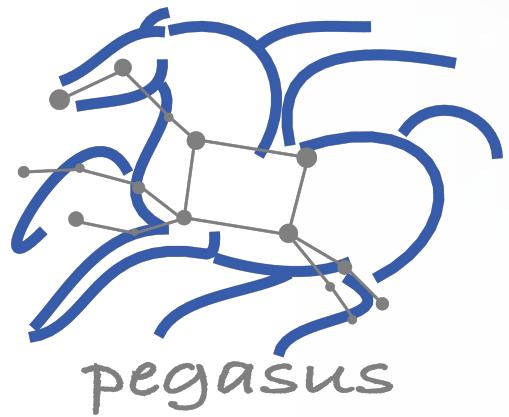
²USC Center for Advanced Research Computing (CARC)

vahi@isi.edu , rynge@isi.edu , osinski@usc.edu



Advanced Research Computing
Enabling scientific breakthroughs at scale





1. Introduction

Workflow Systems and USC CACR / HPC?

- We will talk about:
 - Multiple job workloads
 - Relationship between jobs
 - Automatic data management
 - ... and more
- HPC is not just parallel jobs
 - High throughput computing (HTC)

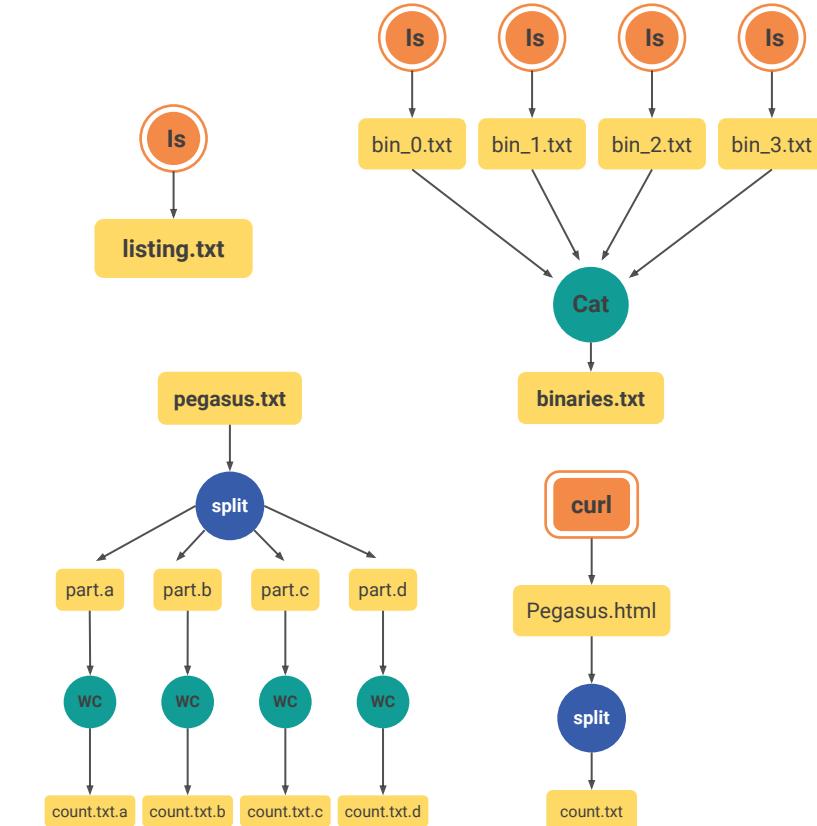


What are Scientific Workflows



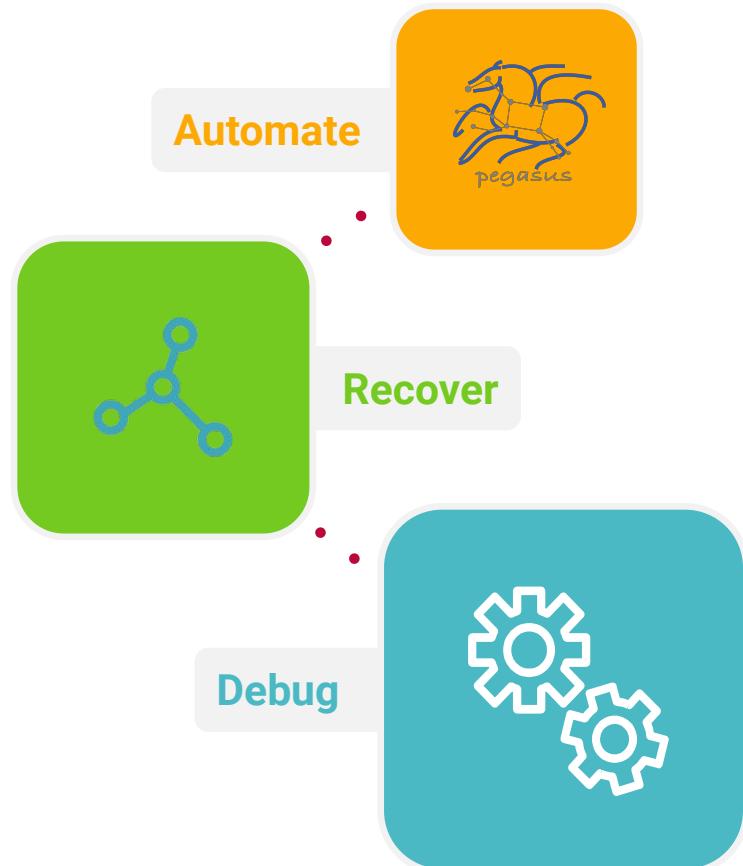
- ▶ **Conducts a series of computational tasks.**
 - Resources distributed across Internet.
- ▶ **Chaining (outputs become inputs) replaces manual hand-offs.**
 - Accelerated creation of products.
- ▶ **Ease of use - gives non-developers access to sophisticated codes.**
 - Resources distributed across Internet.
- ▶ **Provides framework to host or assemble community set of applications.**
 - Honors original codes. Allows for heterogeneous coding styles.
- ▶ **Framework to define common formats or standards when useful.**
 - Promotes exchange of data, products, codes. Community metadata.
- ▶ **Multi-disciplinary workflows can promote even broader collaborations.**
 - E.g., ground motions fed into simulation of building shaking.
- ▶ **Certain rules or guidelines make it easier to add a code into a workflow.**

Workflow Building Blocks



Slide Content Courtesy of David Okaya, SCEC, USC

Why Pegasus?



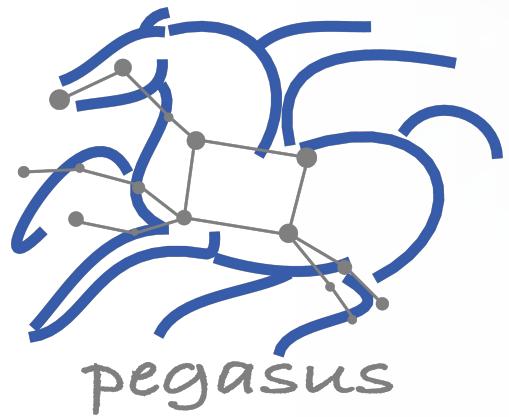
- ▶ **Automates Complex**, Multi-stage Processing Pipelines
- ▶ Enables Parallel, **Distributed Computations**
- ▶ **Automatically Executes** Data Transfers
- ▶ Reusable, Aids **Reproducibility**
- ▶ Records How Data was Produced (**Provenance**)
- ▶ Handles **Failures** with to Provide Reliability
- ▶ Keeps Track of Data and **Files**
- ▶ Ensures **Data Integrity** during workflow execution

Workflow Challenges Across Domains

- Describe complex workflows in a simple way
- Access distributed, heterogeneous data and resources (heterogeneous interfaces)
- Deal with resources/software that change over time
- Ease of use. Ability to debug and monitor large workflows

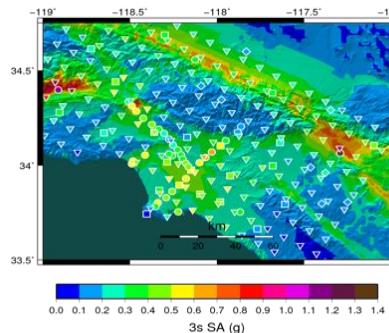
Our Focus

- ▶ Separation between workflow description and workflow execution
- ▶ Workflow planning and scheduling (scalability, performance)
- ▶ Task execution (monitoring, fault tolerance, debugging, web dashboard)
- ▶ Provide additional assurances that a scientific workflow is not accidentally or maliciously tampered with during its execution.



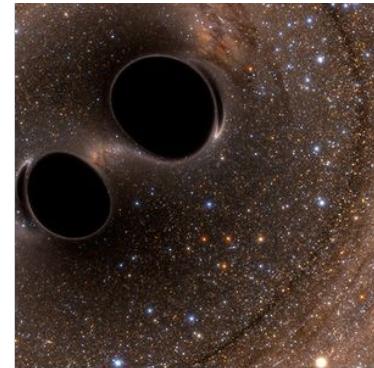
Some of The Success Stories...

Southern California Earthquake Center's CyberShake



First Physics-Based "Shake map" of Southern California

Laser Interferometer Gravitational-Wave Observatory (LIGO)



First direct detection of a gravitational wave (colliding black holes)

XENONnT - Dark Matter Search



Mix of MPI and single-core jobs, mix of CPU, GPU codes.
Large data sets (10s of TBs), ~300 workflows with
420,000 tasks each
Supported since 2005: changing CI, x-platform execution

High-throughput computing workload, access to HPC resources, ~ 21K Pegasus workflows, ~ 107M tasks

Supported since 2001, distributed data, opportunistic computing resources

- Custom data management
- Rucio for data management
- MongoDB instance to track science runs and data products.

Monte Carlo simulations and the main processing pipeline.

Southern California Earthquake Center's CyberShake

CPU jobs
(Mesh generation, seismogram synthesis)

1,094,000 node-hours



GPU jobs:
439,000 node-hours
AWP-ODC finite-difference code
5 billion points per volume, 23,000 timesteps
200 GPUs for 1 hour



Titan:
421,000 CPU node-hours, 110,000 GPU node-hours



Blue Waters:
673,000 CPU node-hours, 329,000 GPU node-hours



Builders ask seismologists:



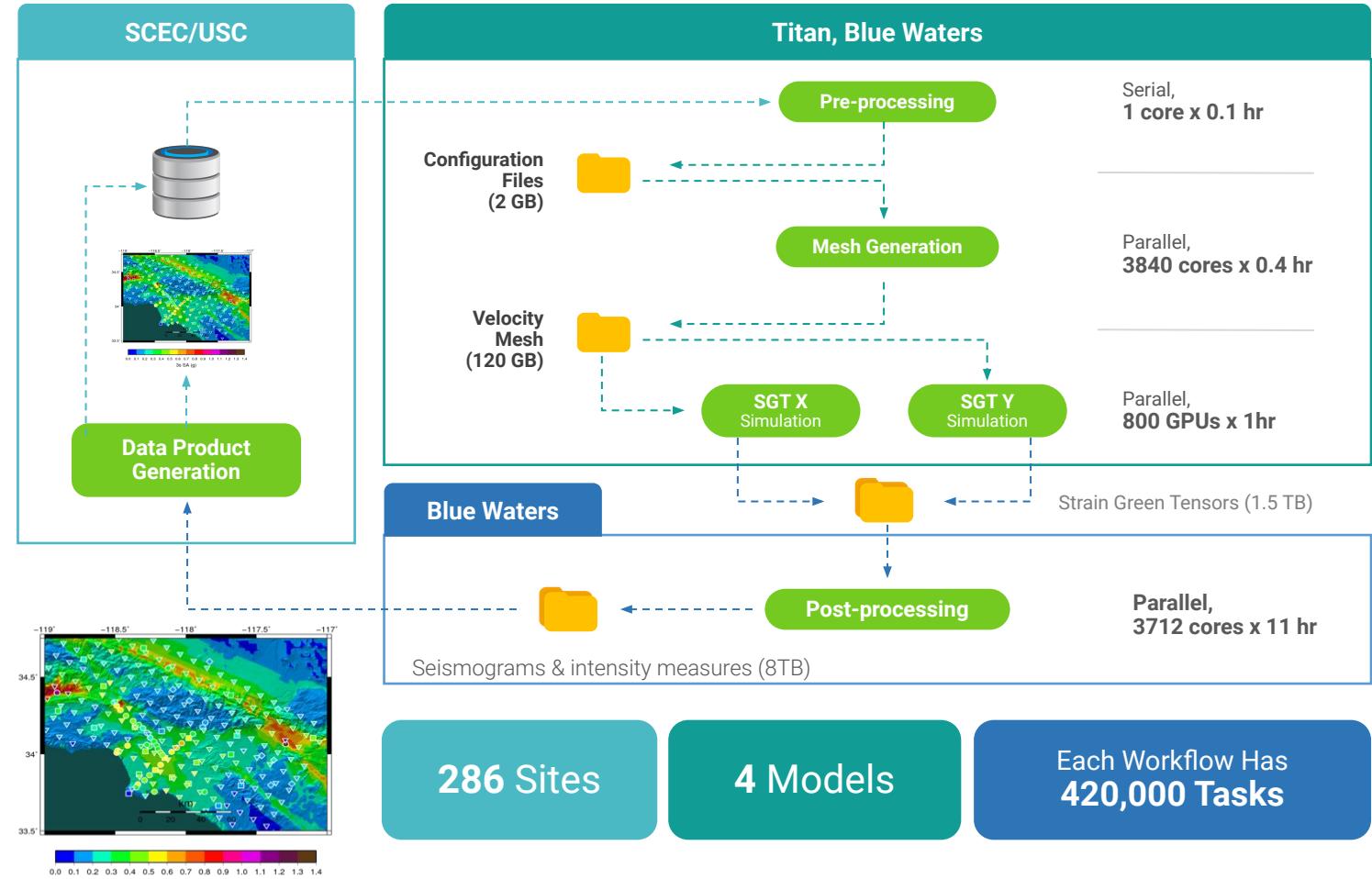
What will the peak ground motion be at my new building in the next 50 years?

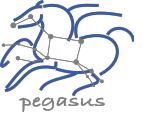


Seismologists answer this question



using Probabilistic Seismic Hazard Analysis (PSHA)





Data Flow for LIGO Pegasus Workflows in OSG

Advanced LIGO

Laser Interferometer
Gravitational Wave Observatory



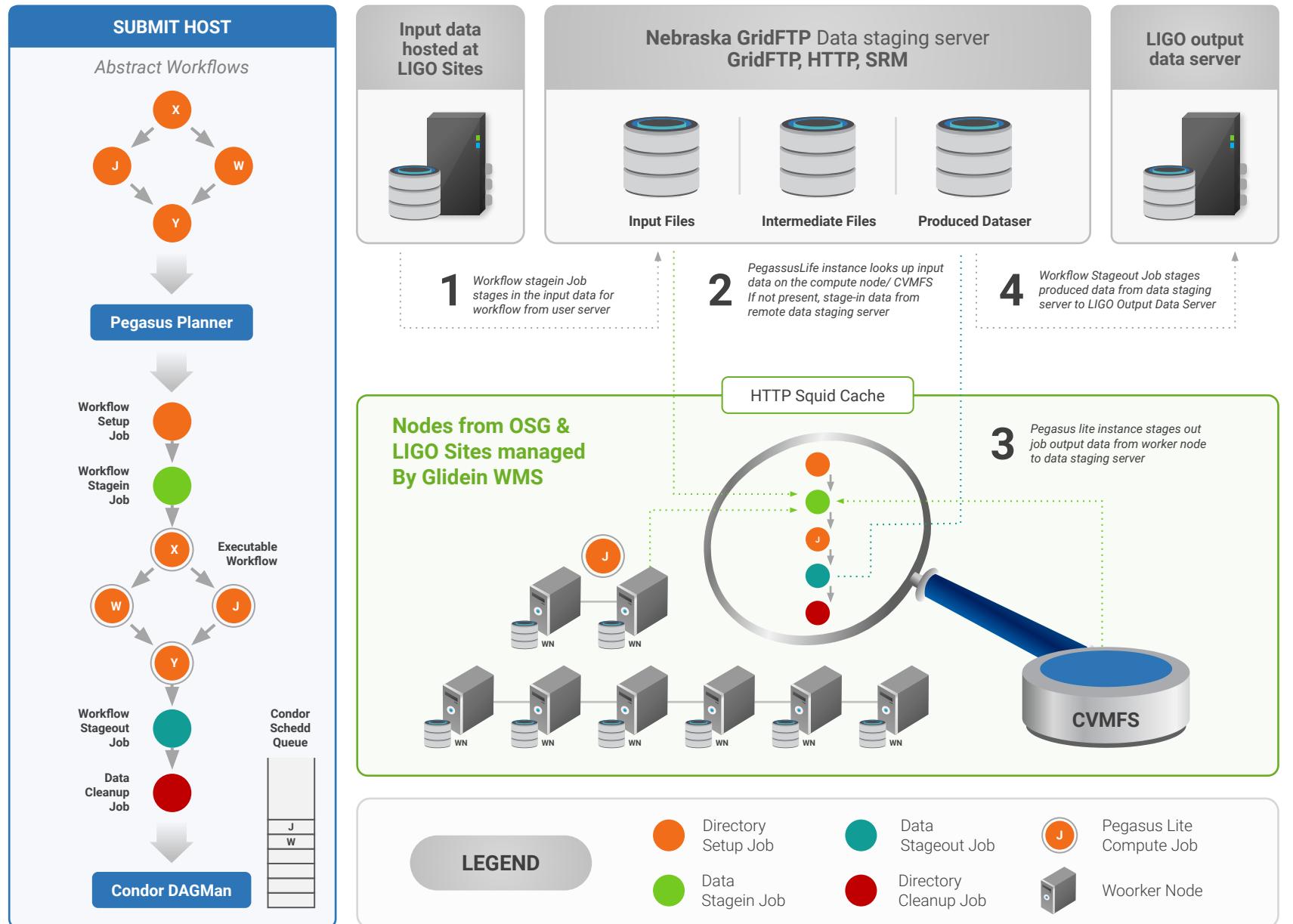
60,000 Compute Tasks

Input Data: 5000 files (10GB total)

Output Data: 60,000 files (60GB total)

Processed Data: 725 GB

Executed on **LIGO Data Grid, EGI, Open Science Grid and XSEDE**

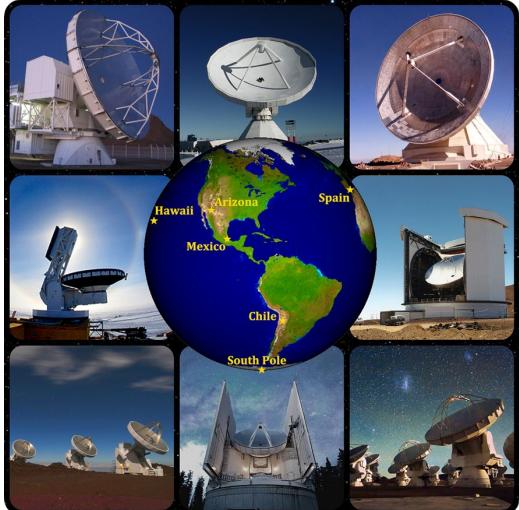




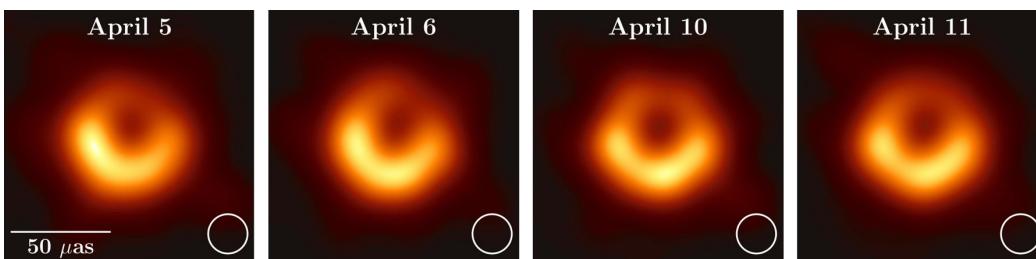
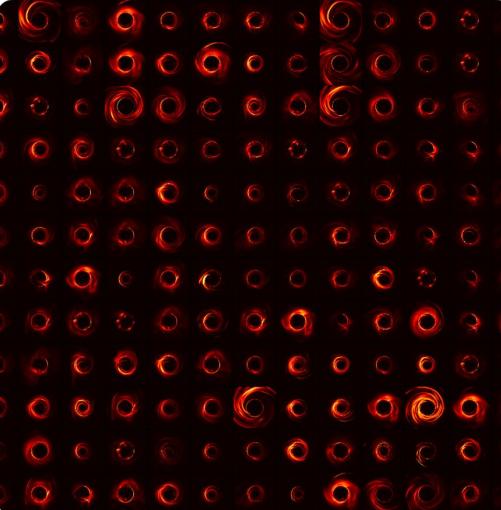
Event Horizon Telescope

Bringing Black Holes into Focus

8 telescopes: 5 PB of data



60 simulations: 35 TB data



First images of black hole at the center of the M87 galaxy

Improve constraints on Einstein's theory
of general relativity by 500x

480,000 jobs - 2,600,000 core hours

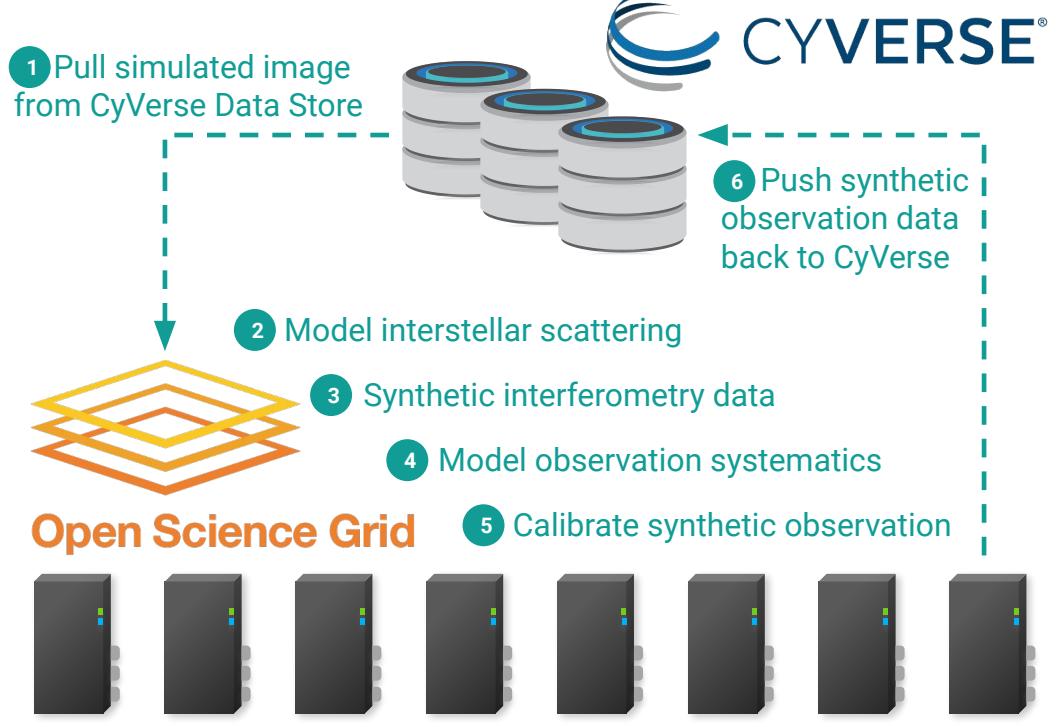
#15 in all OSG projects in last 6 months

#2 in all OSG astronomy projects in the last 6 months

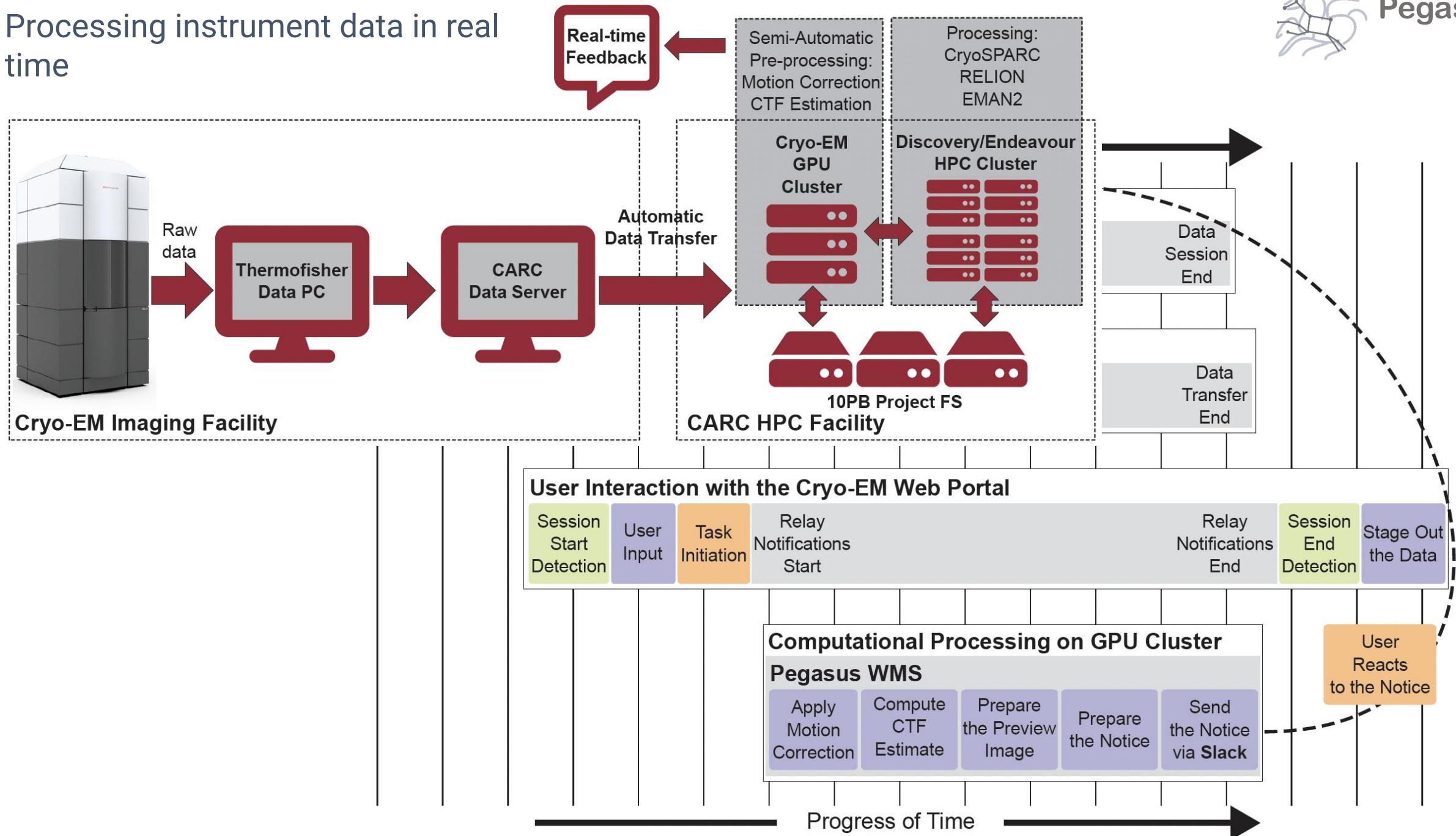


Pegasus-SYMBIA Pipeline

Physically accurate synthetic observation data from simulations are keys to develop calibration and imaging algorithms, as well as comparing the observation with theory and interpreting the results.



Processing instrument data in real time





Key Pegasus Concepts

► Pegasus WMS == Pegasus planner (mapper) + DAGMan workflow engine + HTCondor scheduler/broker

- Pegasus maps workflows to infrastructure
- DAGMan manages dependencies and reliability
- HTCondor is used as a broker to interface with different schedulers

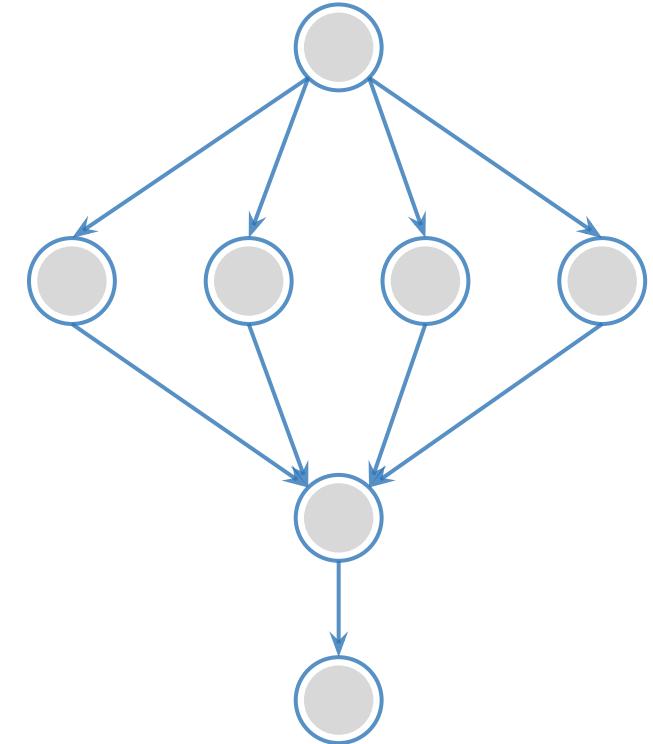
► Workflows are DAGs

- Nodes: jobs, edges: dependencies
- No while loops, no conditional branches
- Jobs are standalone executables

► Planning occurs ahead of execution

► Planning converts an abstract workflow into a concrete, executable workflow

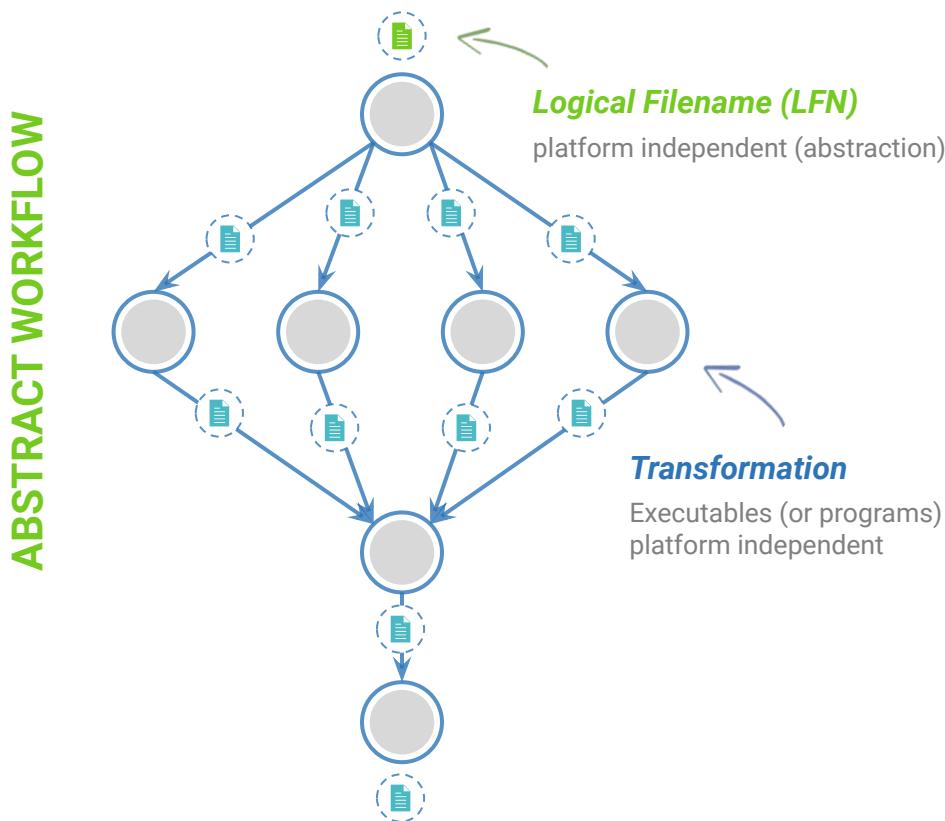
- Planner is like a compiler



Input Workflow Specification YAML formatted

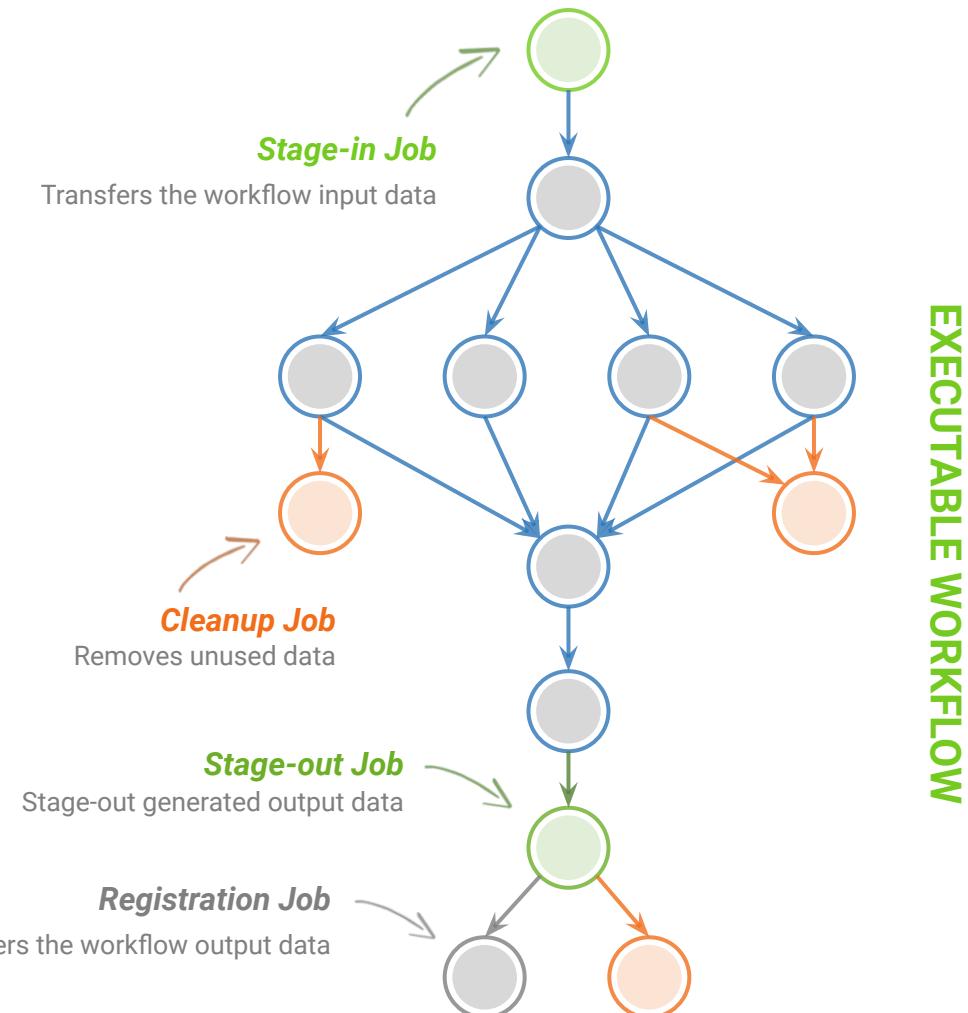
Portable Description

Users do not worry about low level execution details



directed-acyclic graphs

Output Workflow



Pegasus Deployment



Workflow Submit Node

- Pegasus WMS
- HTCondor

One or more Compute Sites

- Compute Clusters
- Cloud
- OSG

Input Sites

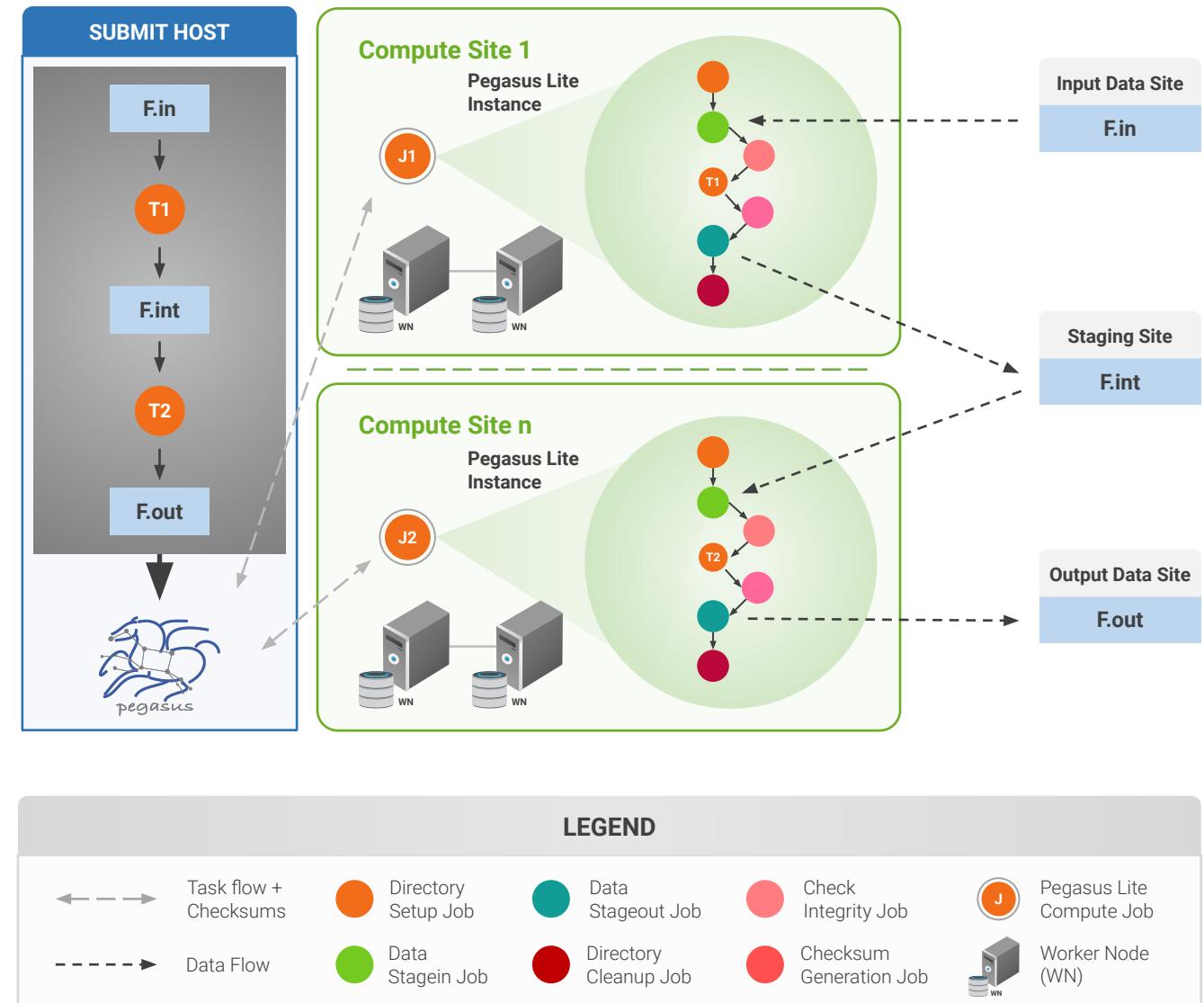
- Host Input Data

Data Staging Site

- Coordinate data movement for workflow

Output Site

- Where output data is placed





Pegasus-transfer

Pegasus' internal data transfer tool with support for a number of different protocols

● Directory creation, file removal

- If protocol can support it, also used for cleanup

● Two stage transfers

- e.g., GridFTP to S3 = GridFTP to local file, local file to S3

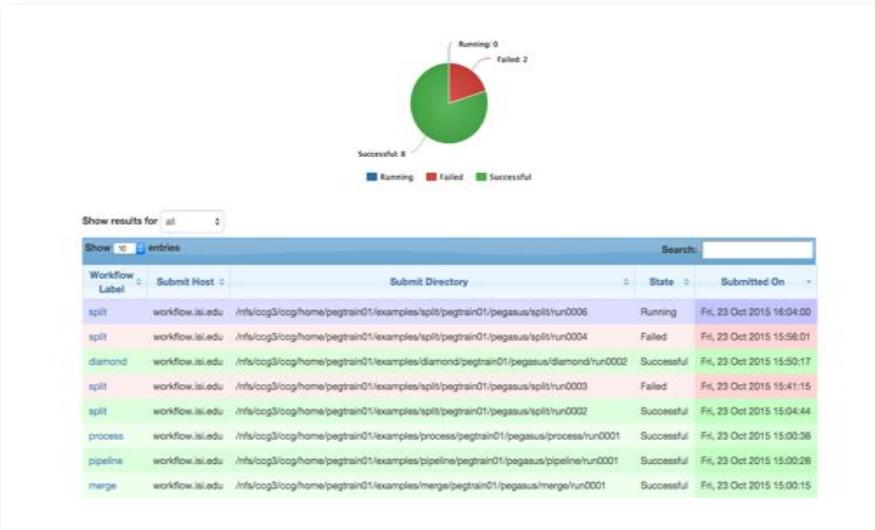
● Parallel transfers

● Automatic retries

● Credential management

- Uses the appropriate credential for each site and each protocol (even 3rd party transfers)

HTTP
SCP
GridFTP
Globus
Online
iRods
Amazon S3
Google Storage
SRM
FDT
Stashcp
Rucio
cp
ln -s

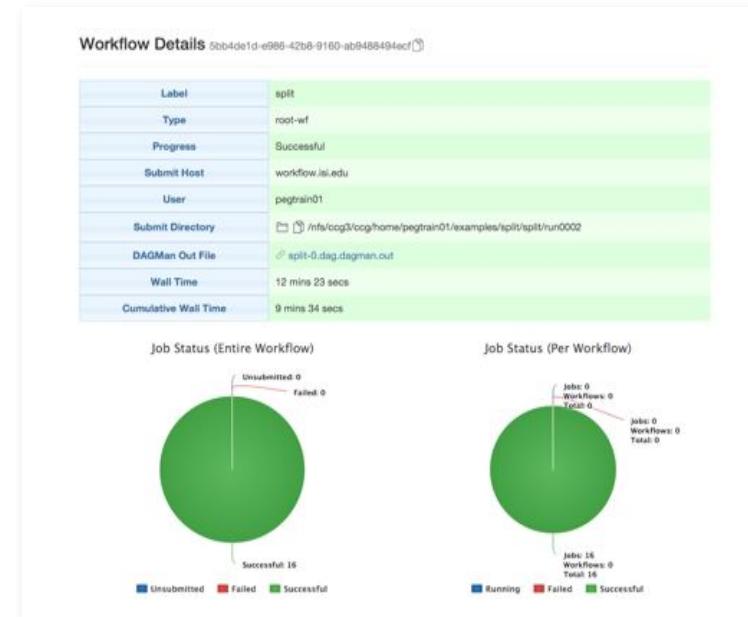


The dashboard shows a pie chart with 8 successful, 0 running, and 2 failed workflows. Below is a table of 16 submitted jobs:

Workflow Label	Submit Host	Submit Directory	State	Submitted On
split	workflow.isi.edu	/nfs/cog3/cog/home/pegtrain01/examples/split/pegtrain01/pegasus/split/run0006	Running	Fri, 23 Oct 2015 16:04:00
split	workflow.isi.edu	/nfs/cog3/cog/home/pegtrain01/examples/split/pegtrain01/pegasus/split/run0004	Failed	Fri, 23 Oct 2015 15:56:01
diamond	workflow.isi.edu	/nfs/cog3/cog/home/pegtrain01/examples/diamond/pegtrain01/pegasus/diamond/run0002	Successful	Fri, 23 Oct 2015 15:50:17
split	workflow.isi.edu	/nfs/cog3/cog/home/pegtrain01/examples/split/pegtrain01/pegasus/split/run0003	Failed	Fri, 23 Oct 2015 15:41:15
split	workflow.isi.edu	/nfs/cog3/cog/home/pegtrain01/examples/split/pegtrain01/pegasus/split/run0002	Successful	Fri, 23 Oct 2015 15:04:44
process	workflow.isi.edu	/nfs/cog3/cog/home/pegtrain01/examples/process/pegtrain01/pegasus/process/run0001	Successful	Fri, 23 Oct 2015 15:00:38
pipeline	workflow.isi.edu	/nfs/cog3/cog/home/pegtrain01/examples/pipeline/pegtrain01/pegasus/pipeline/run0001	Successful	Fri, 23 Oct 2015 15:00:26
merge	workflow.isi.edu	/nfs/cog3/cog/home/pegtrain01/examples/merge/pegtrain01/pegasus/merge/run0001	Successful	Fri, 23 Oct 2015 15:00:15

Real-time **monitoring** of workflow executions. It shows the **status** of the workflows and jobs, job **characteristics, statistics** and **performance metrics**.

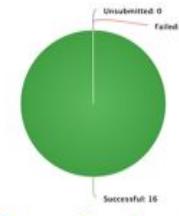
Provenance data is stored into a relational database.



Workflow Details (5bb4de1d-e986-42b8-9160-ab9486494ecf)

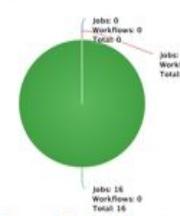
Label	split
Type	root-wf
Progress	Successful
Submit Host	workflow.isi.edu
User	pegtrain01
Submit Directory	/nfs/cog3/cog/home/pegtrain01/examples/split/split/run0000
DAGMan Out File	split-0.dag.dagman.out
Wall Time	12 mins 23 secs
Cumulative Wall Time	9 mins 34 secs

Job Status (Entire Workflow)



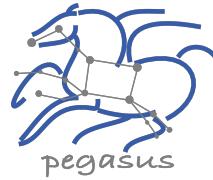
Status	Count
Unsubmitted	0
Failed	0
Successful	16

Job Status (Per Workflow)



Category	Count
Jobs	16
Workflows	0
Total	16
Incomplete	0

<https://pegasus.isi.edu>



PEGASUS DASHBOARD

web interface for monitoring and debugging workflows

Statistics

Workflow Wall Time	12 mins 23 secs
Workflow Cumulative Job Wall Time	9 mins 34 secs
Cumulative Job Waittime as seen from Submit Side	9 mins 35 secs
Workflow Cumulative Badput Time	9 mins 23 secs
Cumulative Job Badput Waittime as seen from Submit Side	9 mins 20 secs
Workflow Retries	1

Workflow Statistics

This Workflow						
Type	Succeeded	Failed	Incomplete	Total	Retries	Total + Retries
Tasks	5	0	0	5	0	5
Jobs	16	0	0	16	2	18
Sub Workflows	0	0	0	0	0	0
Entire Workflow						
Type	Succeeded	Failed	Incomplete	Total	Retries	Total + Retries
Tasks	5	0	0	5	0	5
Jobs	16	0	0	16	2	18
Sub Workflows	0	0	0	0	0	0

Job Breakdown Statistics

Job Statistics

Real-time Monitoring

Reporting

Debugging

Troubleshooting

RESTful API



command-line...

```
$ pegasus-status pegasus/examples/split/run0001
STAT IN_STATE JOB
Run 00:39 split-0 (/home/pegasus/examples/split/run0001)
Idle 00:03 └split_ID0000001
Summary: 2 Condor jobs total (I:1 R:1)

UNRDY READY PRE IN_Q POST DONE FAIL %DONE STATE DAGNAME
 14      0     0     1     0     2     0    11.8 Running *split-0.dag
```

```
$ pegasus-analyzer pegasus/examples/split/run0001
pegasus-analyzer: initializing...

*****Summary*****
Total jobs : 7 (100.00%)
# jobs succeeded : 7 (100.00%)
# jobs failed : 0 (0.00%)
# jobs unsubmitted : 0 (0.00%)
```

```
$ pegasus-statistics -s all pegasus/examples/split/run0001
-----
Type      Succeeded Failed Incomplete Total Retries Total+Retries
Tasks        5       0       0       5       0       5
Jobs         17      0       0      17      0      17
Sub-Workflows  0       0       0       0       0       0
-----
Workflow wall time : 2 mins, 6 secs
Workflow cumulative job wall time : 38 secs
Cumulative job wall time as seen from submit side : 42 secs
Workflow cumulative job badput wall time :
Cumulative job badput wall time as seen from submit side :
```

Provenance Data
can be Summarized
pegasus-statistics
or
Used for Debugging
pegasus-analyzer

And if a job fails?



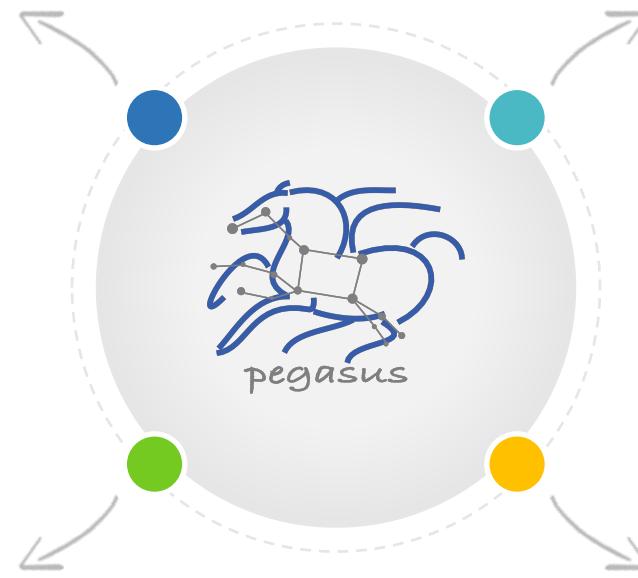
Postscript

detects non-zero exit code output
parsing for success or failure
message exceeded timeout do not
produced expected output files



Checkpoint Files

job generates checkpoint files
staging of checkpoint files is
automatic on restarts



Job Retry



helps with transient failures
set number of retries per job
and run

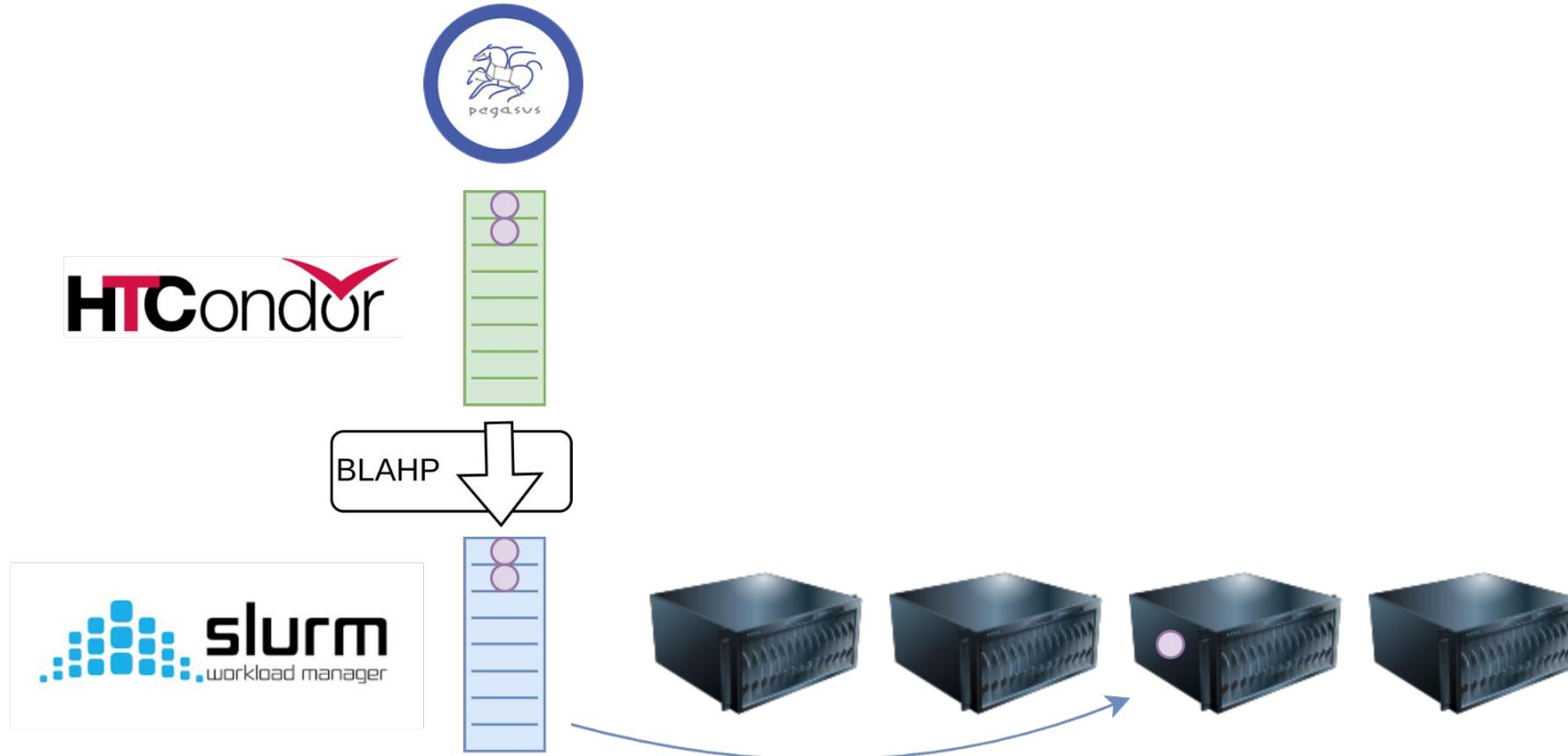


Rescue DAGs

workflow can be restarted from
checkpoint file recover from
failures with minimal loss



HTCondor with BLAHP translation layer



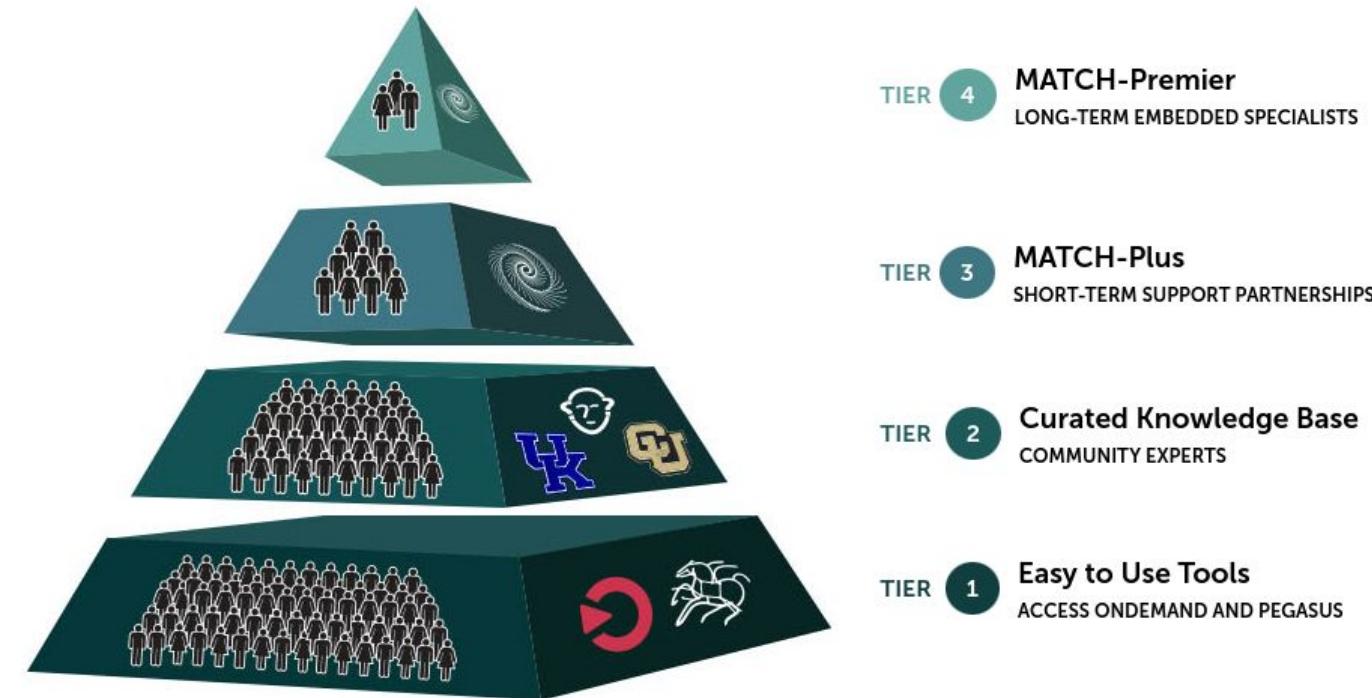
Pegasus is part of the ACCESS support strategy

Pegasus is be used as a tier 1 tool

Central Open OnDemand instance with Pegasus, HTCondor and Jupyter

It is be easy to run HTC workflows across ACCESS sites

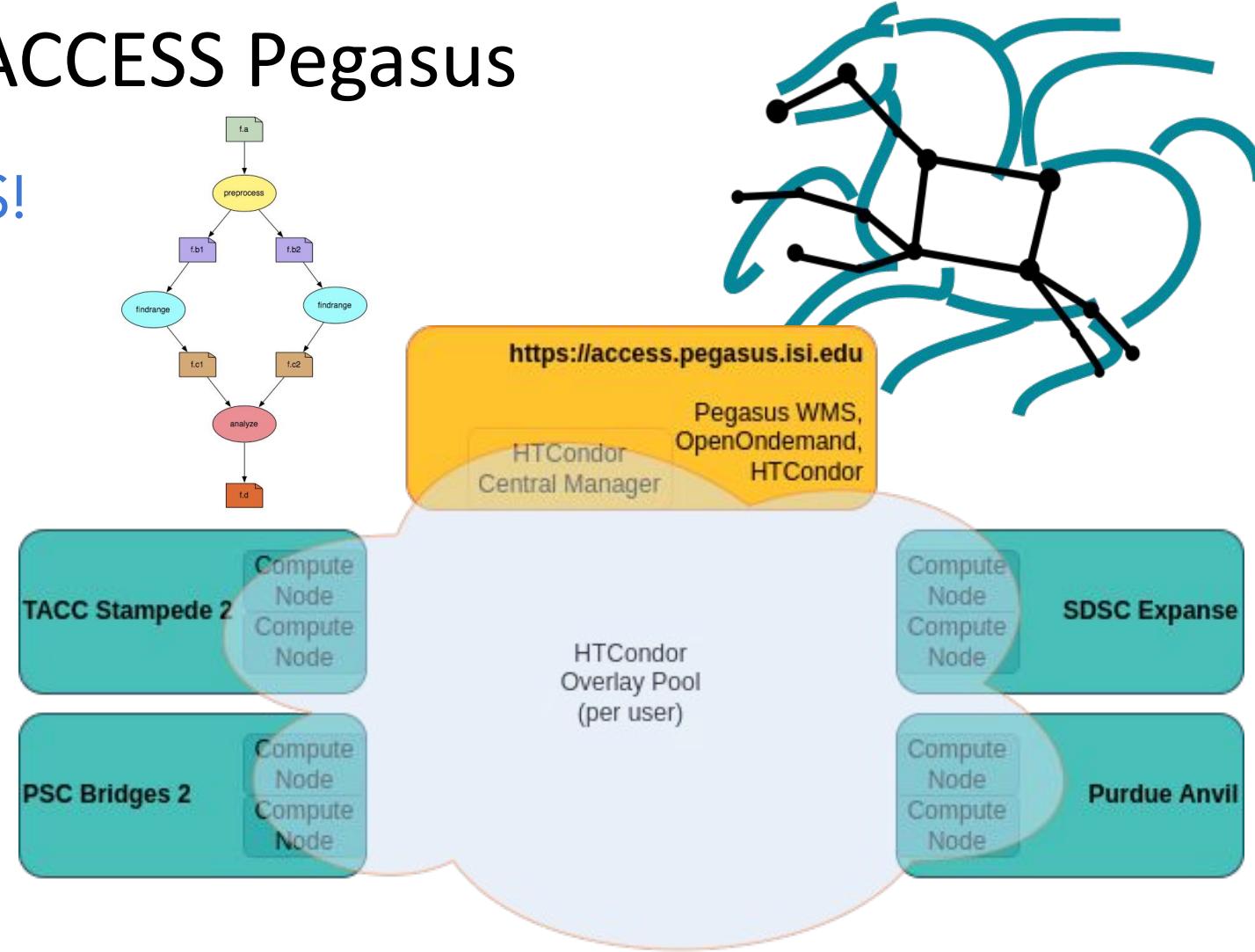
Tiered Support Strategy



ACCESS Pegasus

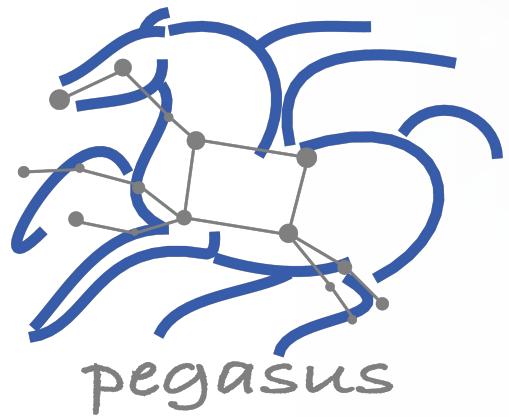
Bring your workflows to ACCESS!

- Execute scientific workflows across ACCESS resources
- OpenOnDemand Portal: has all you need: Jupyter Notebooks, ACCESS authentication, Pegasus workflow management, and HTCondor job management
- Bring your own ACCESS capacity: HTCondor Annex - pilot jobs automatically create a virtual HTCondor pool



<https://access.pegasus.isi.edu>

More at: support.access-ci.org/pegasus



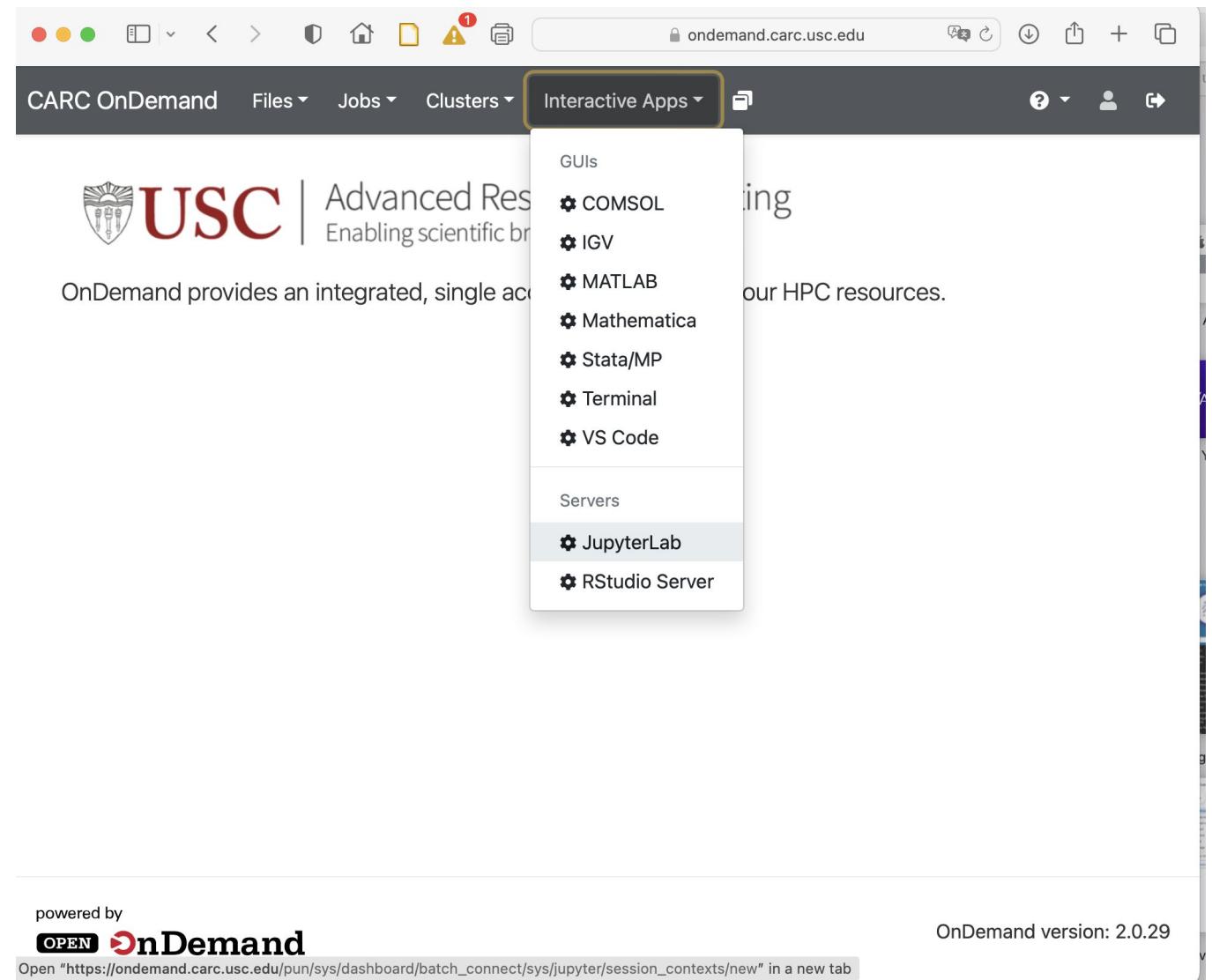
2. Hands on Exercises

Hands on Tutorial Exercises: Login to Open OnDemand

- You need to be on USC Network and need to use your USC credentials to log in
- Use a web browser and log on to USC OnDemand Instance at <https://ondemand.carc.usc.edu> .

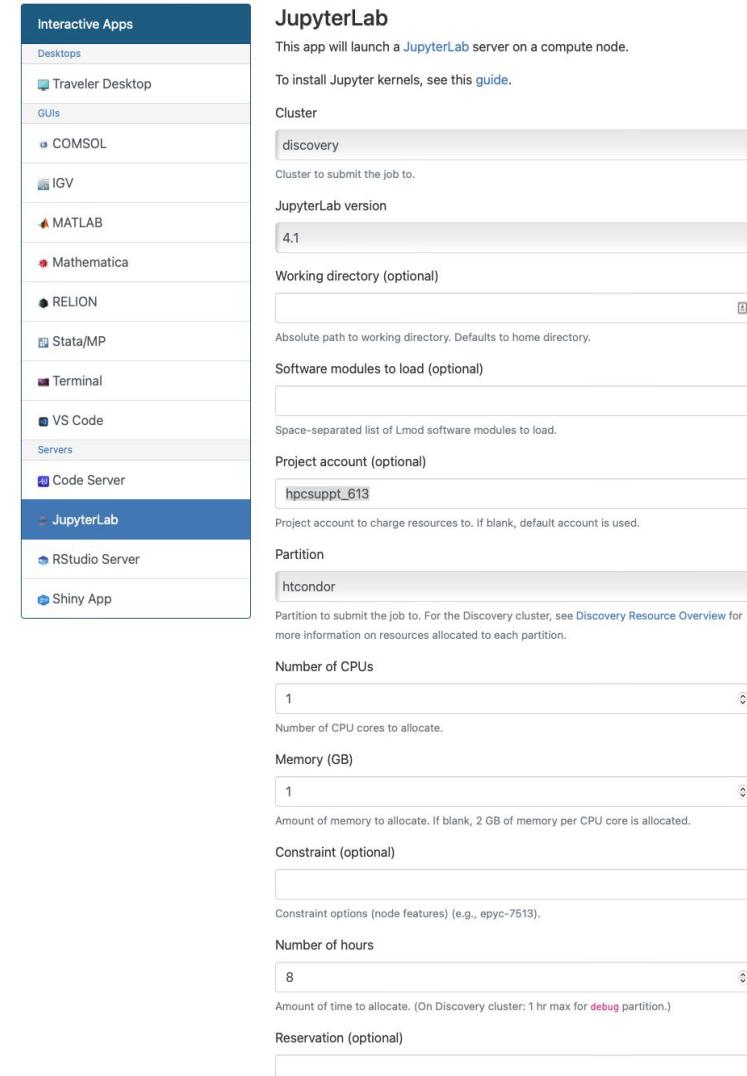
Hands on Tutorial Exercises: Start a Jupyter Server

- Start a Jupyter notebook server, Click on Interactive Apps and then select JupyterLab



Hands on Tutorial Exercises: Jupyter Lab Configuration

- When launching the Jupyter Lab, it is important to select the following
- For Cluster: specify Discovery
- For Account: specify the account **hpcsuppt_613**
- For Partition specify htcondor



The screenshot shows a configuration interface for launching an application. On the left, there is a sidebar titled "Interactive Apps" containing a list of applications. The "JupyterLab" application is highlighted with a blue background and white text. The main panel on the right is titled "JupyterLab" and contains several configuration fields:

- Cluster:** A dropdown menu set to "discovery".
Description: Cluster to submit the job to.
- JupyterLab version:** A dropdown menu set to "4.1".
Description: To install Jupyter kernels, see this [guide](#).
- Working directory (optional):** An input field with a browse icon.
Description: Absolute path to working directory. Defaults to home directory.
- Software modules to load (optional):** An input field.
Description: Space-separated list of Lmod software modules to load.
- Project account (optional):** An input field set to "hpcsuppt_613".
Description: Project account to charge resources to. If blank, default account is used.
- Partition:** A dropdown menu set to "htcondor".
Description: Partition to submit the job to. For the Discovery cluster, see [Discovery Resource Overview](#) for more information on resources allocated to each partition.
- Number of CPUs:** An input field set to "1".
Description: Number of CPU cores to allocate.
- Memory (GB):** An input field set to "1".
Description: Amount of memory to allocate. If blank, 2 GB of memory per CPU core is allocated.
- Constraint (optional):** An input field.
Description: Constraint options (node features) (e.g., epyc-7513).
- Number of hours:** An input field set to "8".
Description: Amount of time to allocate. (On Discovery cluster: 1 hr max for **debug** partition.)
- Reservation (optional):** An input field.

Hands on Tutorial Exercises: Connect to JupyterLab

The screenshot shows the CARC OnDemand web interface. The top navigation bar includes links for CARC OnDemand, Files, Jobs, Clusters, Interactive Apps, and a user profile. A green notification bar at the top center says "Session was successfully created." Below the navigation is a breadcrumb trail: Home / My Interactive Sessions.

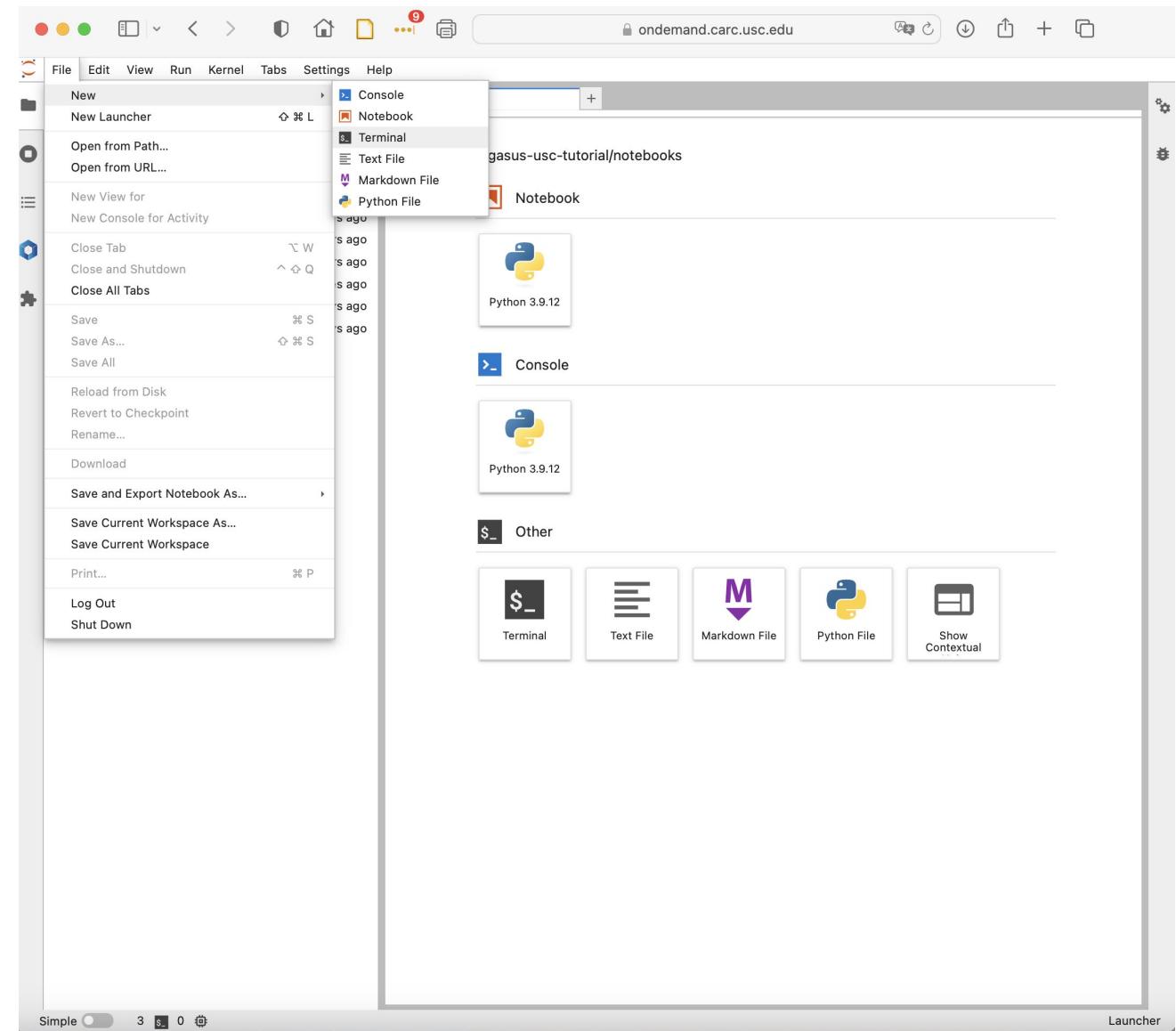
The main content area displays three interactive session cards:

- JupyterLab (14184588)** 1 node | 1 core | Running
Host: >[_e10-12.hpc.usc.edu](#) Delete
Created at: 2023-03-27 17:09:57 PDT
Time Remaining: 5 hours and 58 minutes
Session ID: [8e5389bc-46d3-44c7-8d12-d836d6c81419](#)
[Connect to JupyterLab](#)
- JupyterLab (14170685)** Completed
Created at: 2023-03-27 10:48:35 PDT Delete
Session ID: [cbf7d166-ee81-4764-9f8a-9a98f108887d](#)
For debugging purposes, this card will be retained for 6 more days
- JupyterLab (14083145)** Completed
Created at: 2023-03-20 15:45:18 PDT Delete
Session ID: [da637e53-df1a-4866-8f9a-7f6f6ef16ed1](#)
For debugging purposes, this card will be retained for 6 more days

A sidebar on the left lists available interactive apps under "Interactive Apps":
GUIs:
• COMSOL
• IGV
• MATLAB
• Mathematica
• Stata/MP
• Terminal
• VS Code
Servers:
• JupyterLab
• RStudio Server

Hands on Tutorial Exercises: Start a Terminal

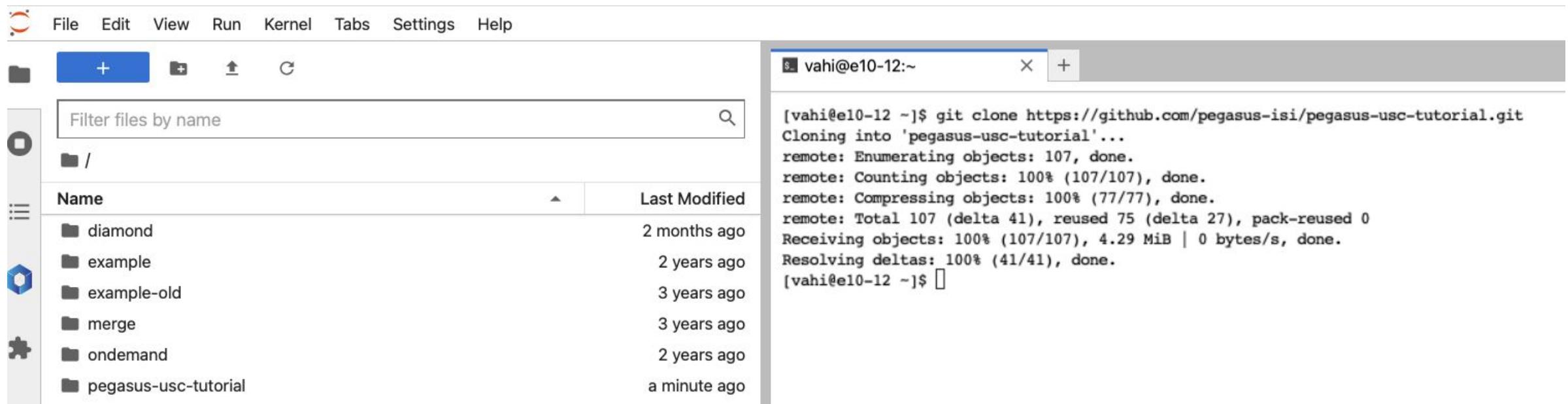
- In JupyterLab, Click on File -> New and then click on Terminal to get the terminal



Hands on Tutorial Exercises: Clone Repository

- Clone Tutorial Repository in the terminal

```
git clone https://github.com/pegasus-isi/pegasus-usc-tutorial.git
```



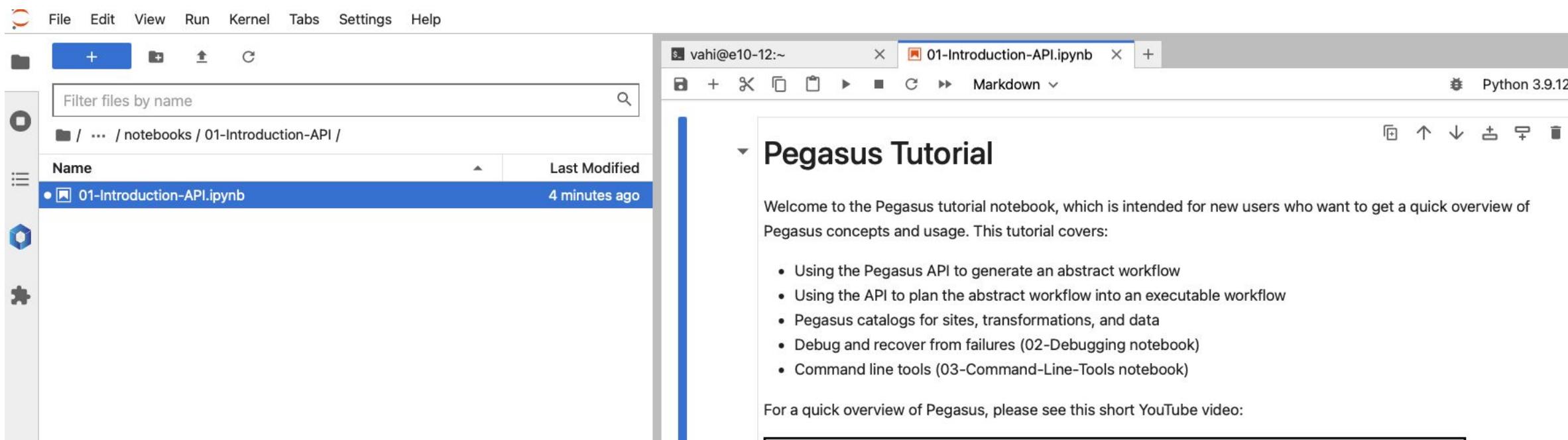
The screenshot shows a Jupyter Notebook interface with two main panes. On the left is a file browser pane with a sidebar containing icons for file operations like new file, new folder, upload, and refresh. It includes a search bar labeled "Filter files by name" and a list of files under a root directory. The list includes "diamond", "example", "example-old", "merge", "ondemand", and "pegasus-usc-tutorial". The "pegasus-usc-tutorial" file was cloned a minute ago. On the right is a terminal pane titled "vahi@e10-12:~". It displays the output of a "git clone" command, showing the progress of cloning the repository from GitHub, including object enumeration, counting, compressing, and receiving objects, and finally resolving deltas.

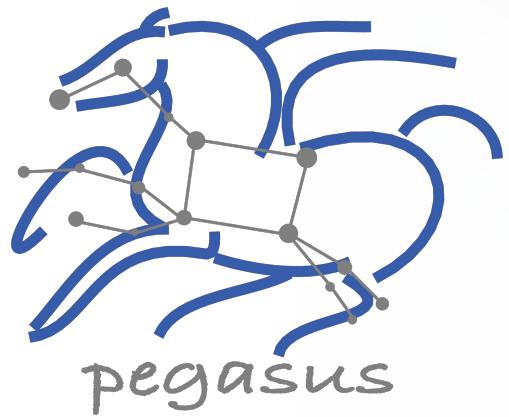
Name	Last Modified
diamond	2 months ago
example	2 years ago
example-old	3 years ago
merge	3 years ago
ondemand	2 years ago
pegasus-usc-tutorial	a minute ago

```
[vahi@e10-12 ~]$ git clone https://github.com/pegasus-isi/pegasus-usc-tutorial.git
Cloning into 'pegasus-usc-tutorial'...
remote: Enumerating objects: 107, done.
remote: Counting objects: 100% (107/107), done.
remote: Compressing objects: 100% (77/77), done.
remote: Total 107 (delta 41), reused 75 (delta 27), pack-reused 0
Receiving objects: 100% (107/107), 4.29 MiB | 0 bytes/s, done.
Resolving deltas: 100% (41/41), done.
[vahi@e10-12 ~]$
```

Hands on Tutorial Exercises: Navigate to Notebooks

- In Jupyter, navigate to the example you are interested in, and step through the notebook.
- For first time users, we highly recommend to do the notebooks in order, as they build up on concepts in the previous notebooks.





2.1 API

<https://pegasus.isi.edu>



U.S. DEPARTMENT OF
ENERGY





Key Pegasus Concepts

▲ Pegasus WMS == Pegasus planner (mapper) + DAGMan workflow engine + HTCondor scheduler/broker

- Pegasus maps workflows to infrastructure
- DAGMan manages dependencies and reliability
- HTCondor is used as a broker to interface with different schedulers

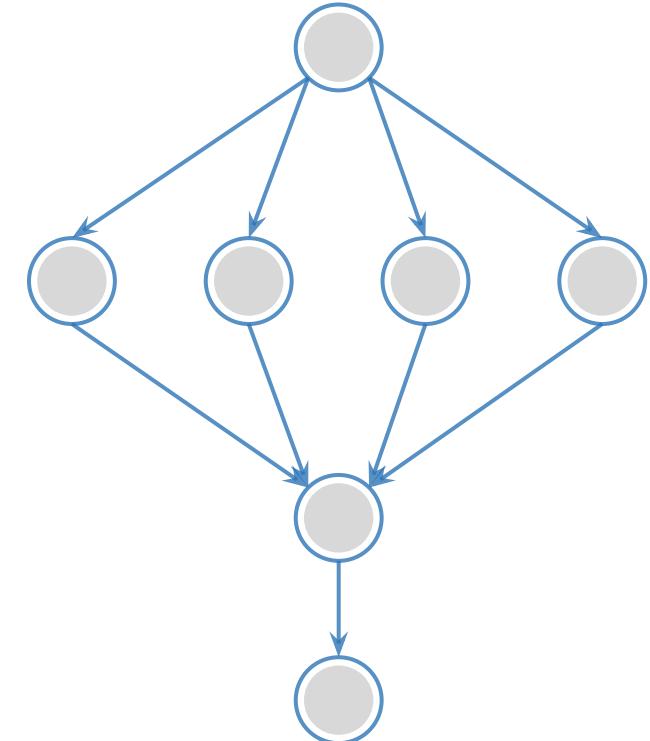
▲ Workflows are DAGs

- Nodes: jobs, edges: dependencies
- No while loops, no conditional branches
- Jobs are standalone executables

▲ Planning occurs ahead of execution

▲ Planning converts an abstract workflow into a concrete, executable workflow

- Planner is like a compiler



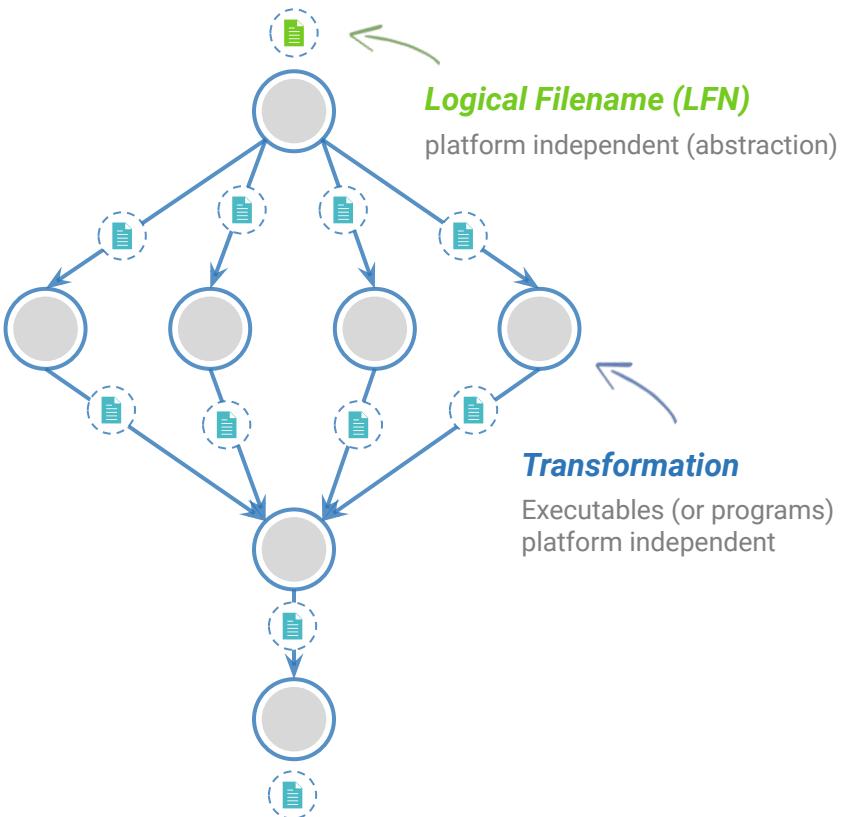


Input Workflow Specification YAML formatted

Portable Description

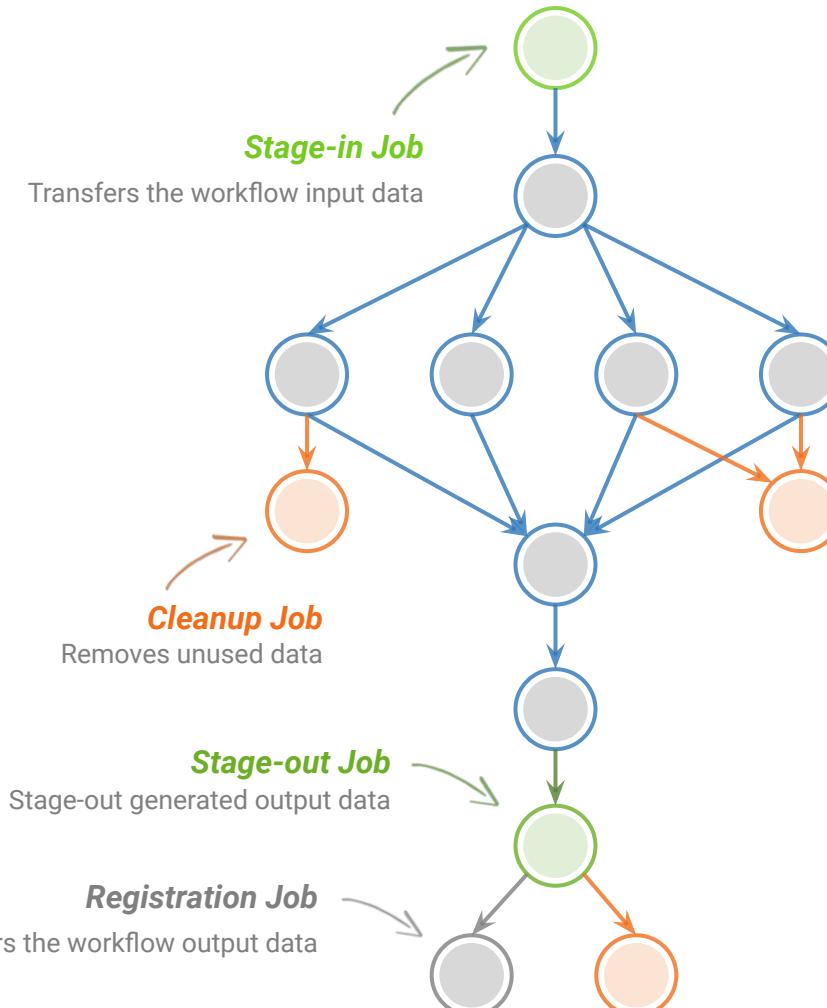
Users do not worry about low level execution details

ABSTRACT WORKFLOW

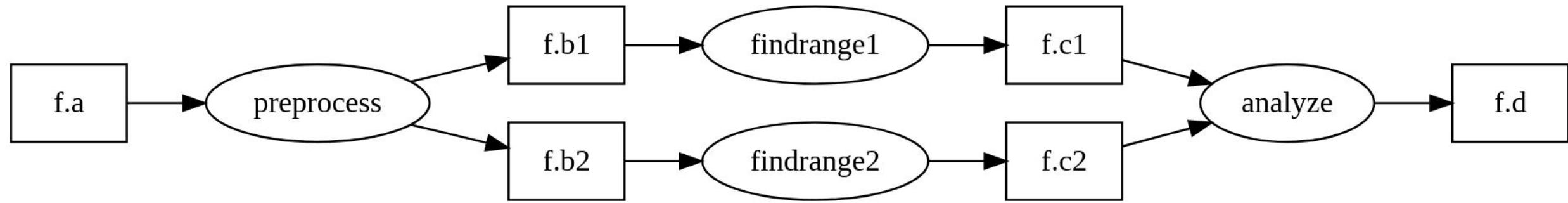


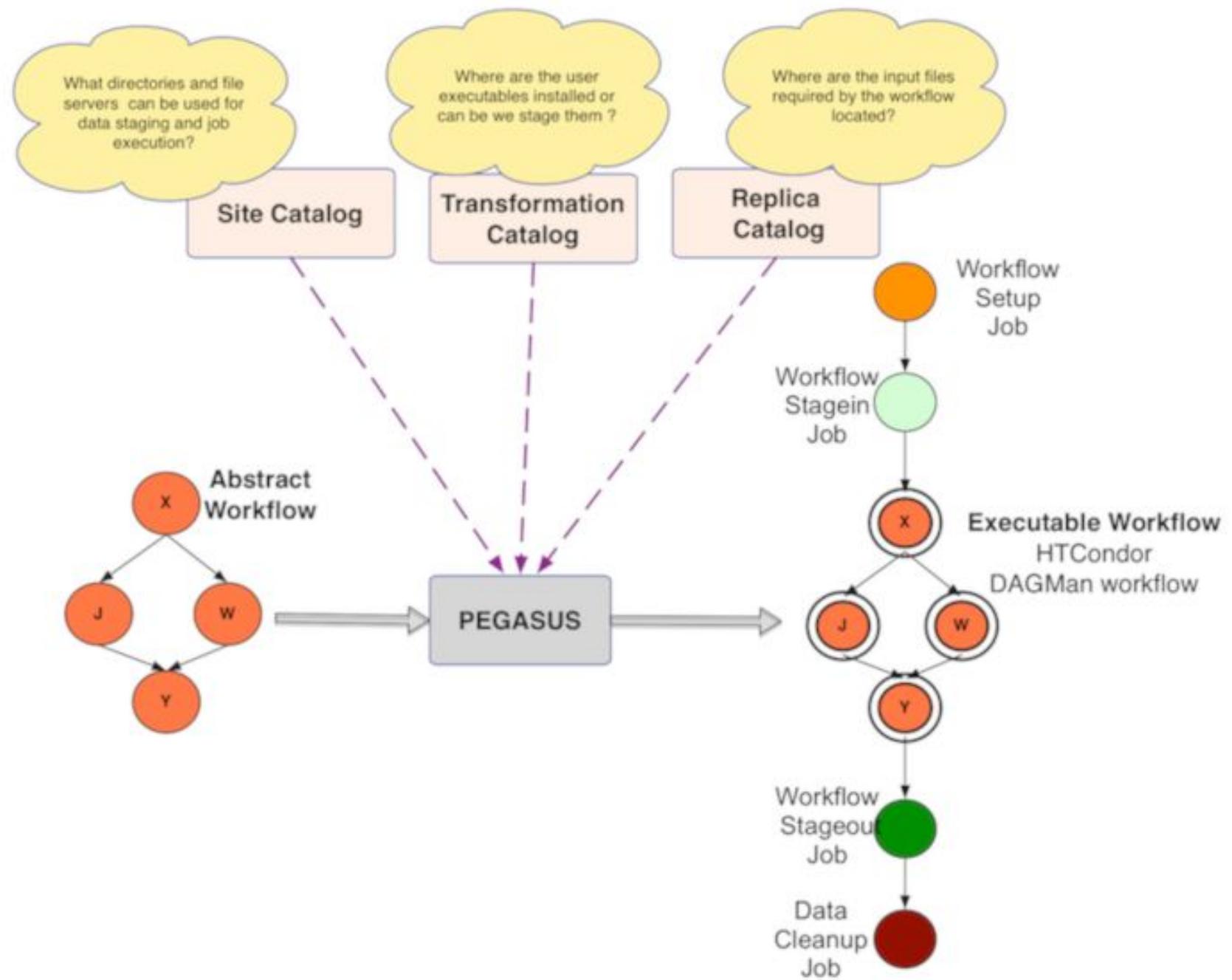
directed-acyclic graphs

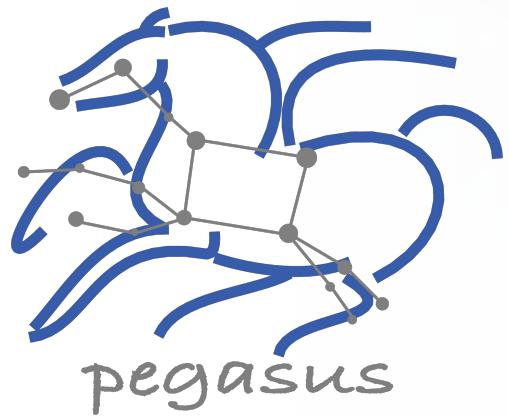
Output Workflow



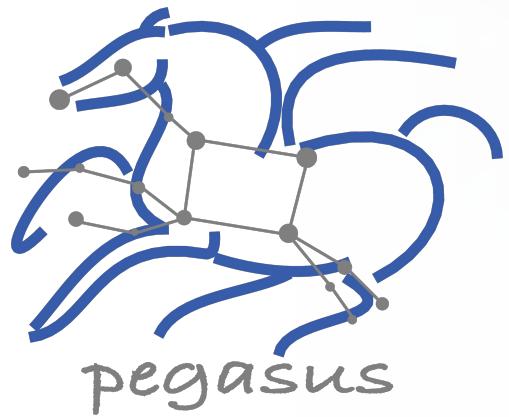
EXECUTABLE WORKFLOW







2.2 Debugging



2.3 Command Line Tools

Pegasus Container Support



Users can refer to **containers** in the **Transformation Catalog** with their executable preinstalled



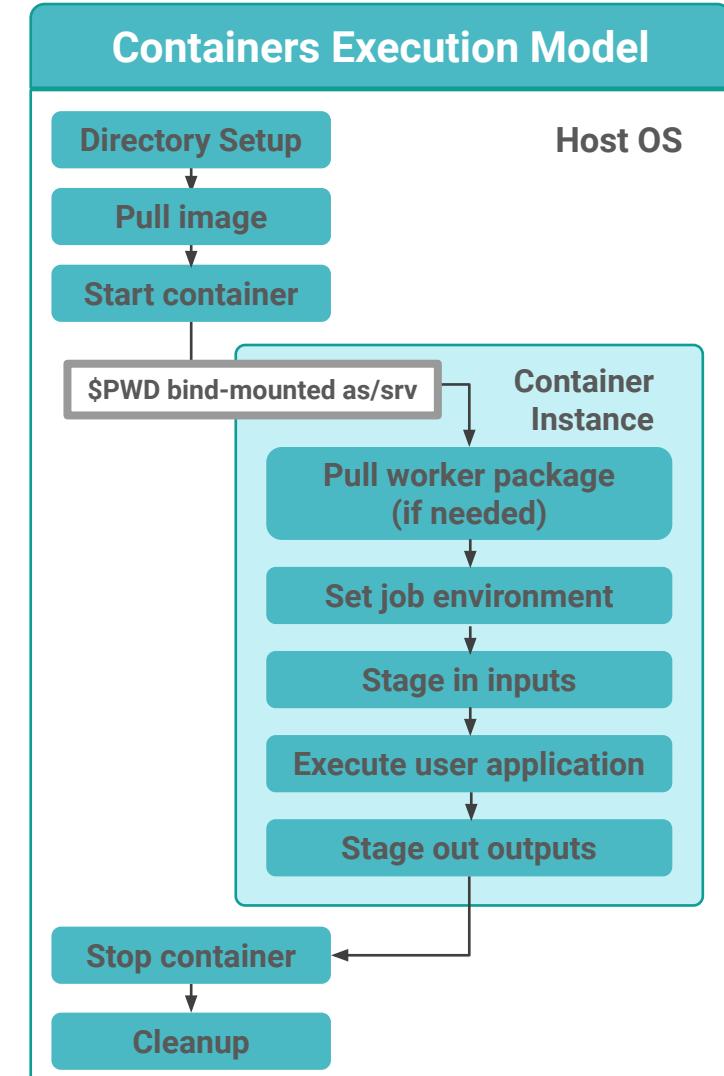
Users can **refer** to a **container** they want to **use – Pegasus stages** their executables and containers to the node

- Useful if you want to use a site recommended/standard container image.
- Users are using generic image with executable staging.



Future Plans

- Users can **specify an image buildfile** for their jobs.
- *Pegasus will build the Docker image as separate jobs in the executable workflow, export them as a tar file and ship them around*



Data Management for Containers



Containers are data too!

Pegasus treats containers as input data dependency

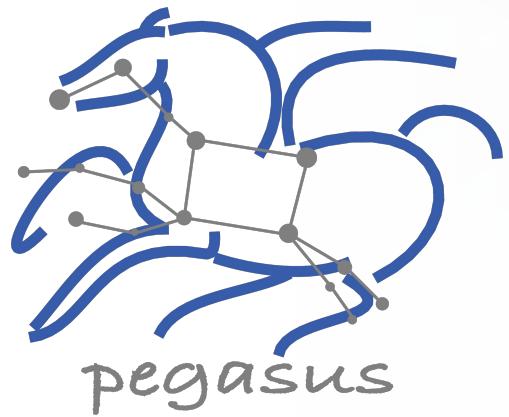
- Staged to compute node if not present
- Docker or Singularity Hub URL's
- Docker Image exported as a TAR file and available at a server, just like any other input dataset

Scaling up for larger workflows

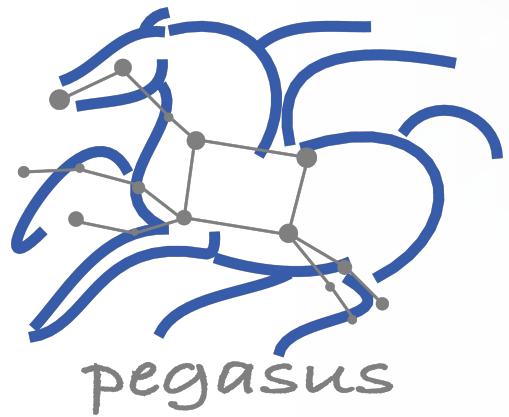
- The image is pulled down as a tar file as part of data stage-in jobs in the workflow
- The exported tar file is then shipped with the workflow and made available to the jobs
- Pricing considerations. You are now charged if you exceed a certain rate of pulls from Hubs

Other Optimizations

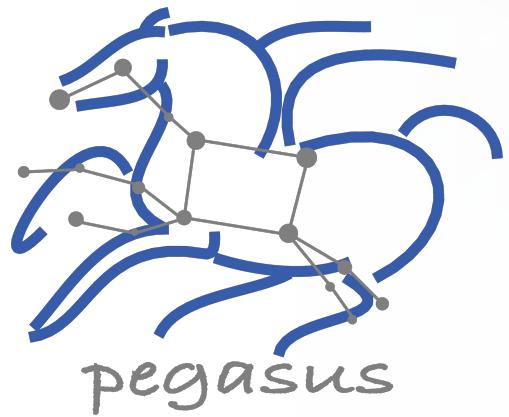
- Symlink against existing images on shared file system such as CVMFS
- The exported tar file is then shipped with the workflow and made available to the jobs



2.4 Containers



2.5 Summary



3. Advanced Topics



Data Staging Configurations

HTCondor I/O (HTCondor pools, OSG, ...)

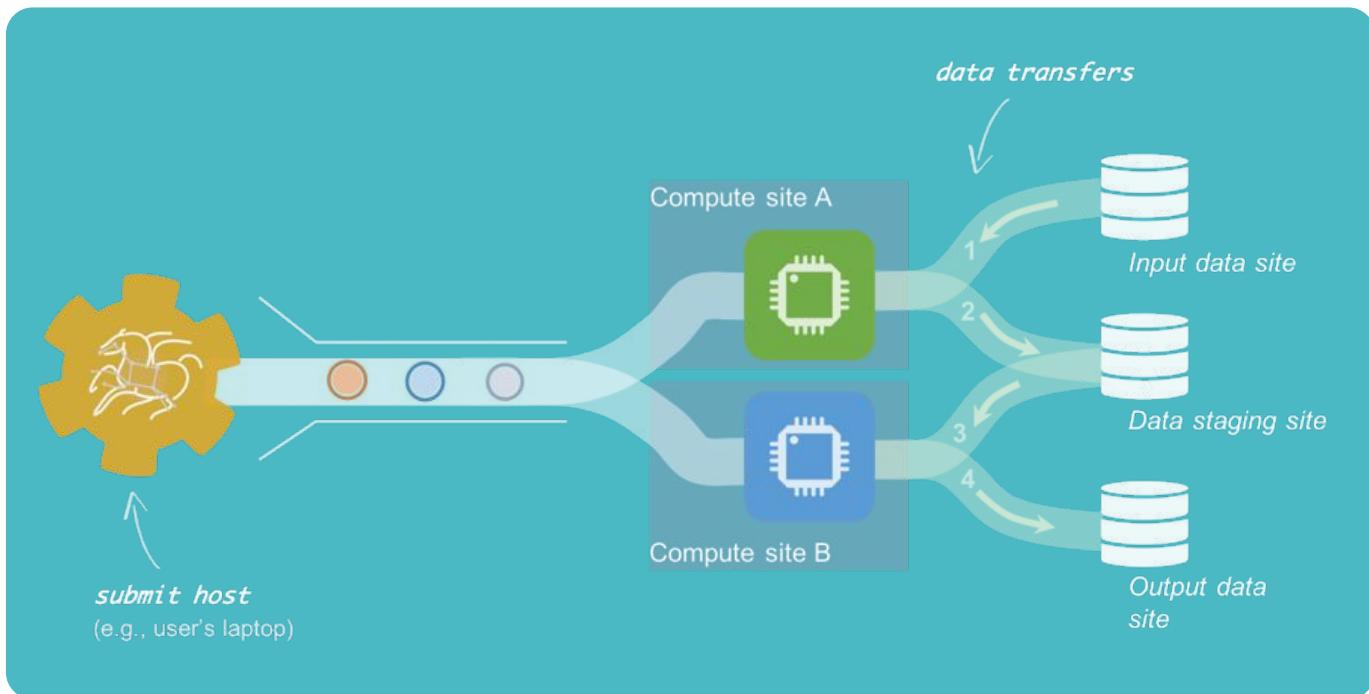
- Worker nodes do not share a file system
- Data is pulled from / pushed to the submit host via HTCondor file transfers
- Staging site is the submit host

Non-shared File System (clouds, OSG, ...)

- Worker nodes do not share a file system
- Data is pulled / pushed from a staging site, possibly not co-located with the computation

Shared File System (HPC sites, XSEDE, Campus clusters, ...)

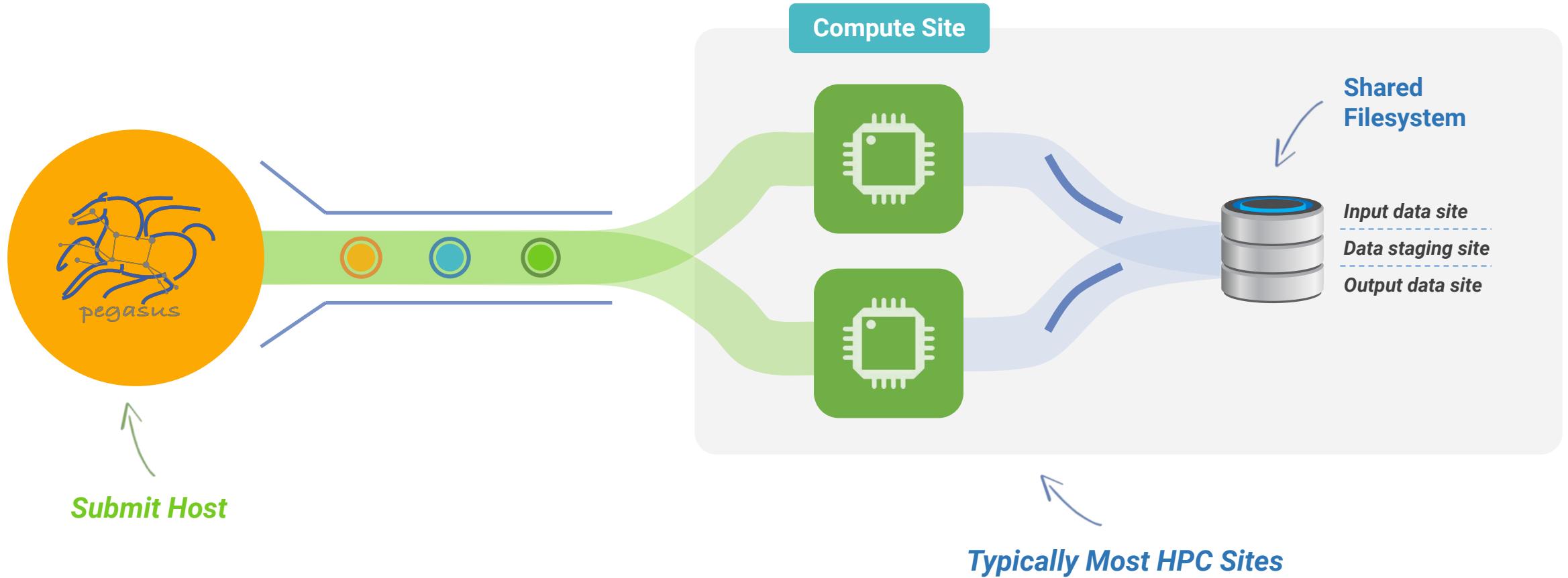
- I/O is directly against the shared file system



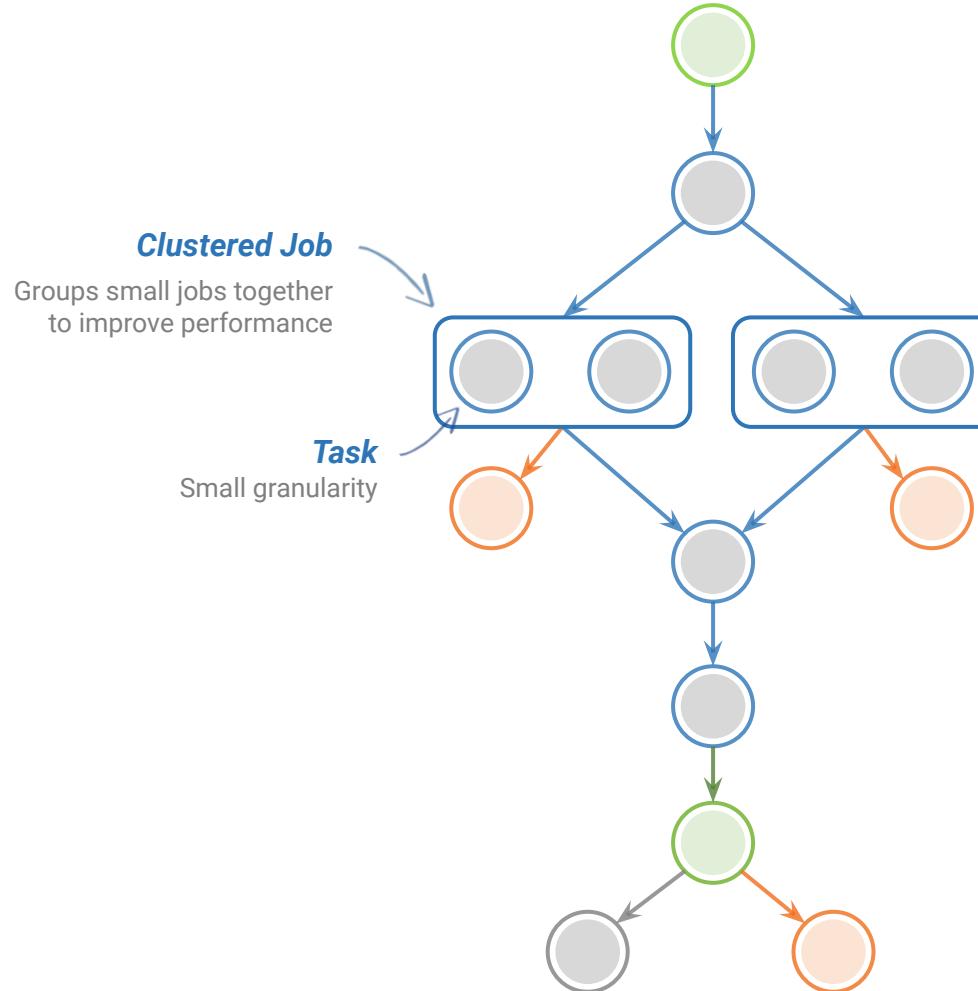
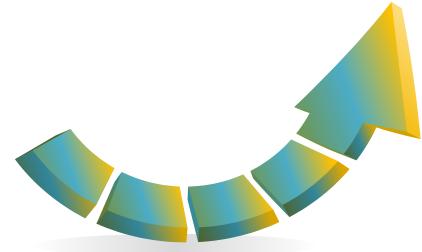


High Performance Computing

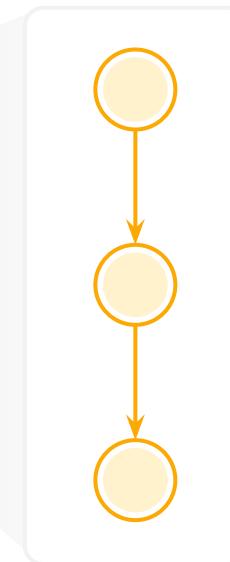
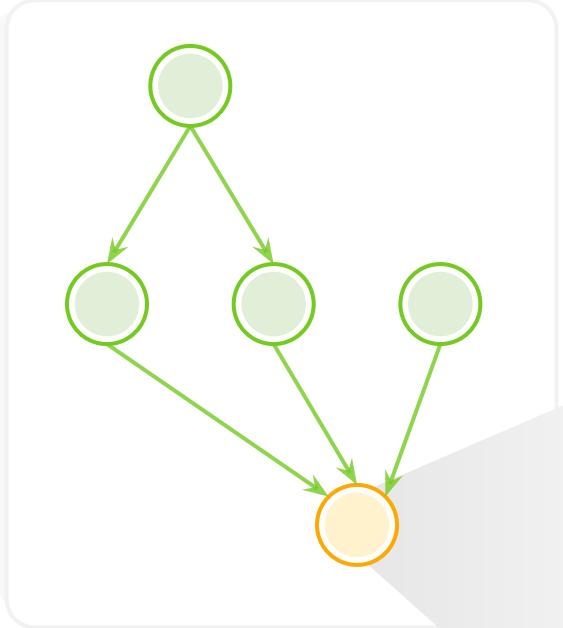
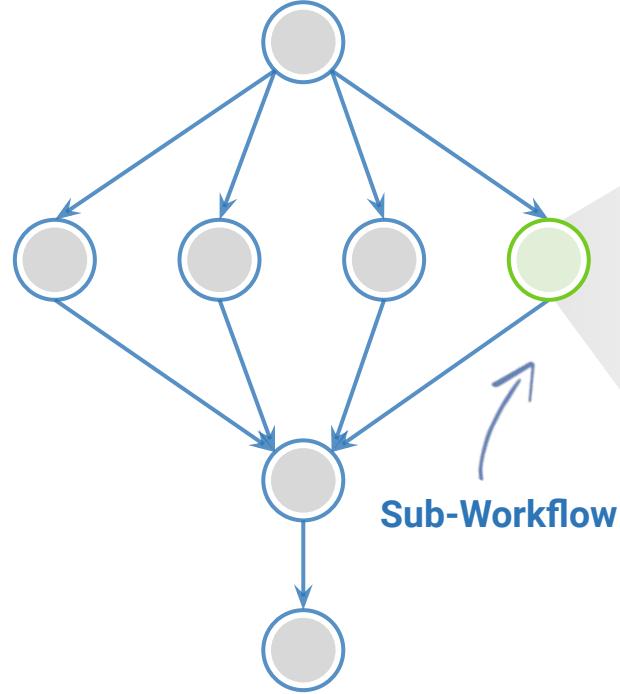
There are several possible configurations...



Performance. Why not improve it?



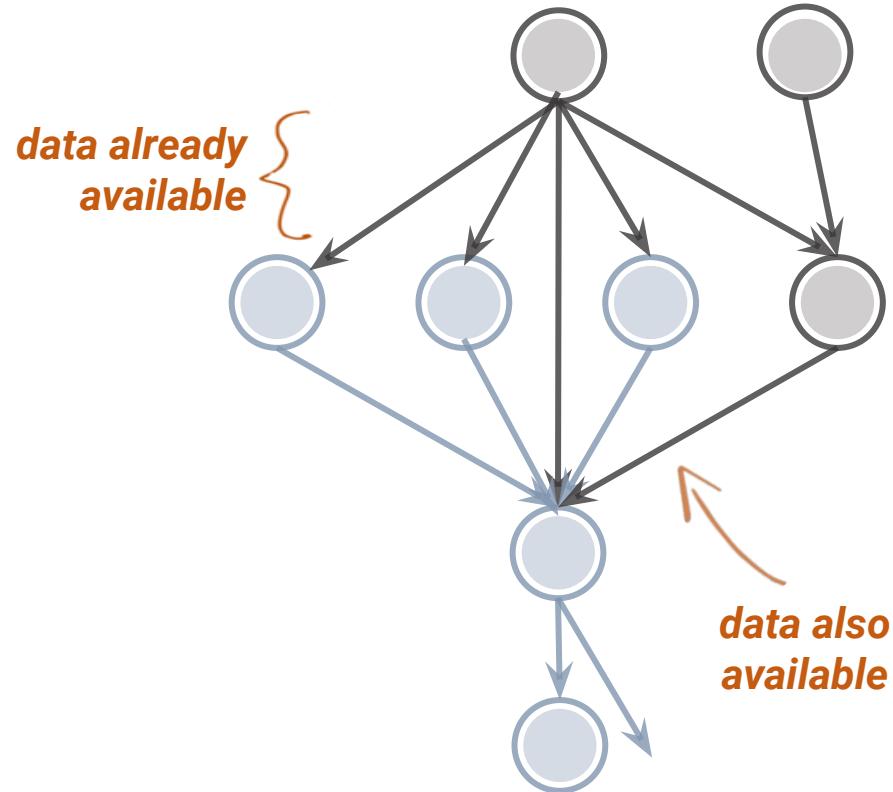
Pegasus also handles large-scale workflows



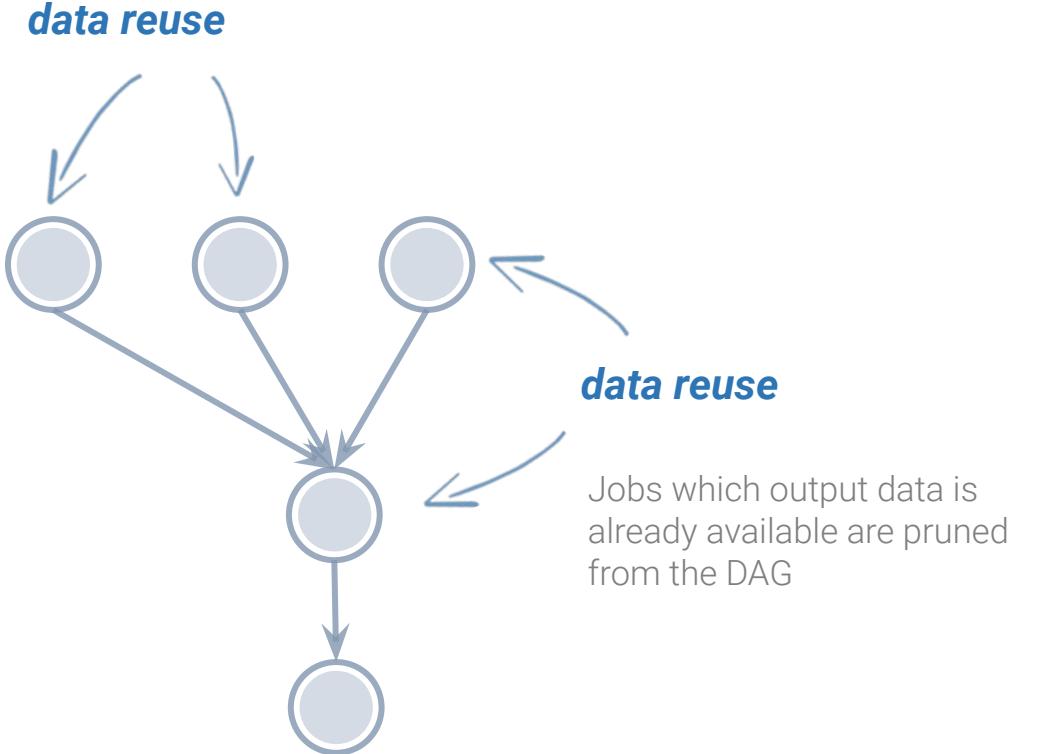
Recursion ends
When abstract
workflow with
only compute jobs
is encountered



Data Reuse prune jobs if output data already exists



workflow
reduction



And if a job fails?



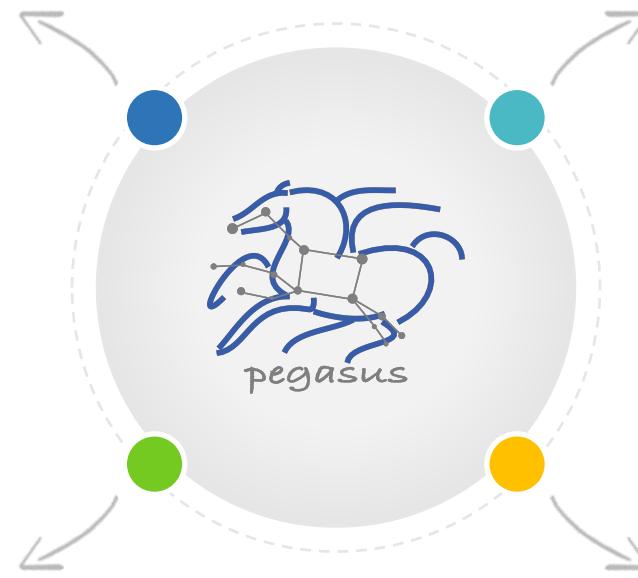
Postscript

detects non-zero exit code output
parsing for success or failure
message exceeded timeout do not
produced expected output files



Checkpoint Files

job generates checkpoint files
staging of checkpoint files is
automatic on restarts



Job Retry



helps with transient failures
set number of retries per job
and run



Rescue DAGs

workflow can be restarted from
checkpoint file recover from
failures with minimal loss



Metadata

Can associate arbitrary key-value pairs with workflows, jobs, and files

Data Registration

Output files get tagged with metadata on registration in the workflow database

Workflow,
Job, File

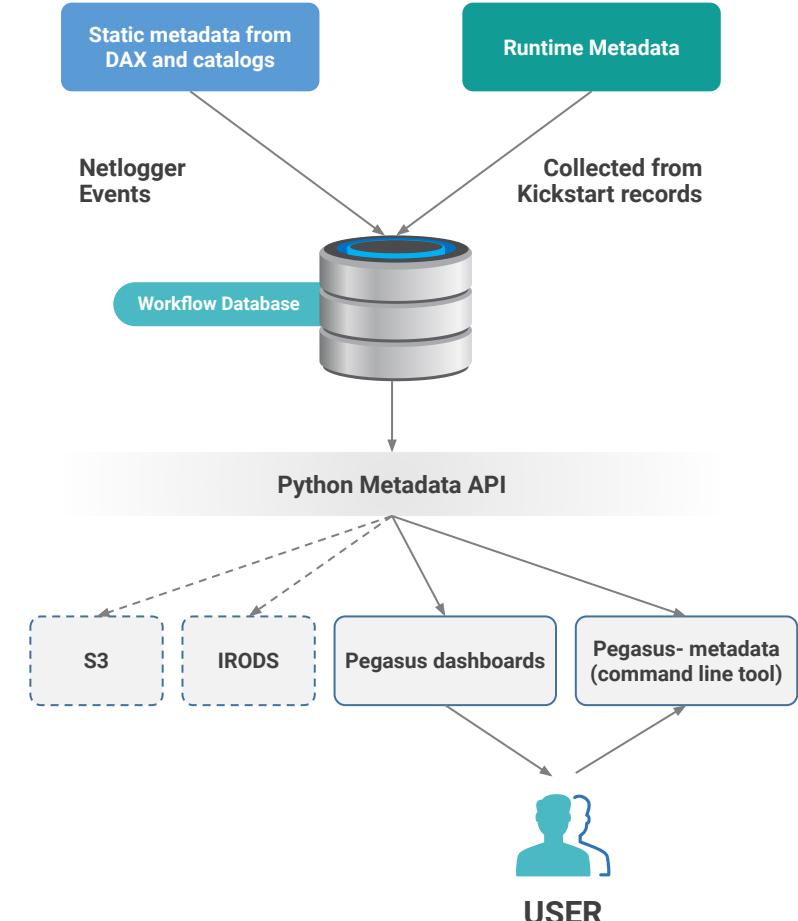
```
x-pegasus:  
apiLang: python  
createdBy: vahi  
createdOn: 12-08-20T10:08:48Z  
pegasus: "5.0"  
name: diamond  
metadata:  
    experiment:"par_all27_prot_lipid"  
jobs:  
- type: "job"  
  name: "namd"  
  id: "ID0000001"  
  arguments: ["equilibrate.conf"]  
  uses:  
  - lfn: "Q42.psf"  
    metadata:  
      type: "psf"  
      charge: "42"  
    type: "input"  
  - lfn: "eq.restart.coord"  
    type: "output"  
    metadata:  
      type: "coordinates"  
      stageOut: true  
      registerReplica: true  
  metadata:  
    timesteps:500000  
    temperature:200  
    pressure:1.01353
```

Static and Runtime Metadata

Static: application parameters
Runtime: performance metrics

Select Data
Based on Metadata

Register Data
With Metadata





Challenges to Scientific Data Integrity

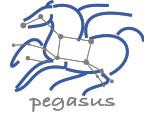
**Modern IT systems
are not perfect
- errors creep in.**

At modern “**Big Data**” sizes we
are starting to see checksums
breaking down.

**Plus there is the threat
of intentional changes:
*malicious attackers,
insider threats, etc.***

User Perception: “Am I not already protected? I have heard about TCP checksums, encrypted transfers, checksum validation, RAID and erasure coding – is that not enough?”

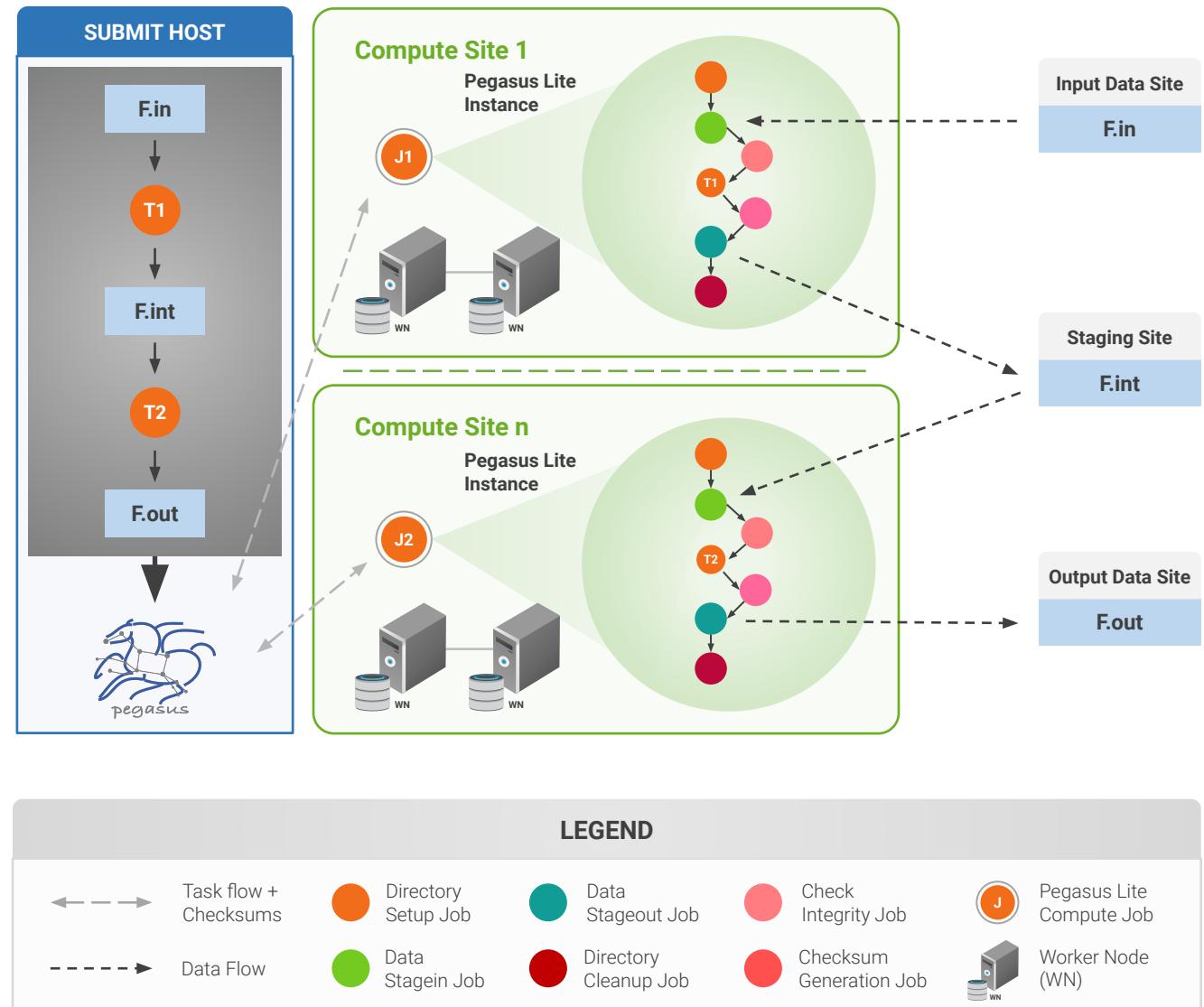
Automatic Integrity Checking in Pegasus



Pegasus performs integrity checksums on input files right before a job starts on the remote node.

- ▶ For raw inputs, **checksums specified in the input replica catalog** along with file locations
- ▶ All **intermediate** and **output** files checksums are generated and tracked within the system.
- ▶ Support for **sha256** checksums

Job failure is triggered if checksums fail



Job Submissions



LOCAL

Submit Machine

Personal HTCondor

Local Campus Cluster accessible via Submit Machine **

HTCondor via BLAHP

**** Both Glite and BOSCO build on HTCondor BLAHP**

Currently supported schedulers:
SLURM SGE PBS MOAB

REMOTE

BOSCO + SSH**

Each node in executable workflow submitted via SSH connection to remote cluster

BOSCO based Glideins**

SSH based submission of glideins

PyGlidein

IceCube glidein service

OSG using glideinWMS

Infrastructure provisioned glideins

CREAMCE

Uses CondorG

Globus GRAM

Uses CondorG



Credentials Management

▲ Credentials required for two purposes

- Job Submission
- Data transfers to **stage-in** input and **stage-out** generated outputs when a job executes



▲ Specifying Credentials

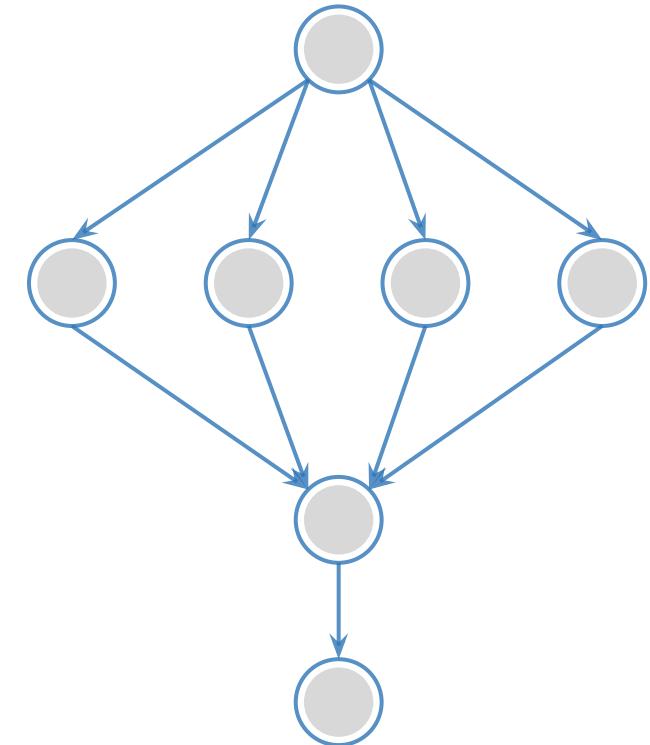
- Users can specify credentials in a **generic credentials file** on submit host
- Associate credentials with sites in site catalog

▲ Approach

- Planner will **automatically** associate the **required credentials** with each job
- The credentials are **transferred** along with the job
- Usually available **only for the duration** of the job **execution**

▲ Supported Credentials

- | | |
|--|------------------|
| ▪ X.509 grid proxies | ▪ iRods password |
| ▪ Amazon AWS S3 keys, | ▪ SSH keys |
| ▪ Google Cloud Platform OAuth token
(.boto file), | ▪ Web Dav |





Pegasus

est. 2001

Automate, recover, and debug scientific computations.

► Get Started

► Pegasus Website

<https://pegasus.isi.edu>

► Users Mailing List

pegasus-users@isi.edu

► Support

pegasus-support@isi.edu

► Slack

Ask for an invite by trying to join pegasus-users.slack.com in the Slack app

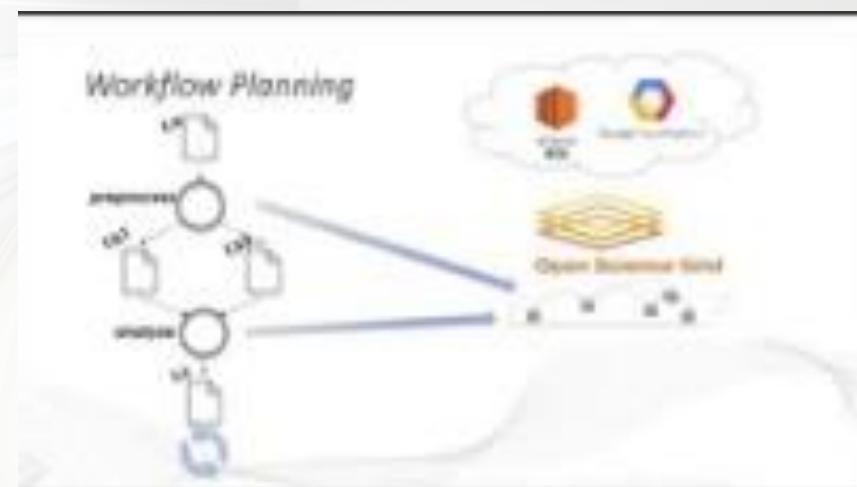
► Pegasus Online Office Hours

<https://pegasus.isi.edu/blog/online-pegasus-office-hours/>



YouTube Channel

<https://www.youtube.com/channel/UCwJQIn1CqBvTJqiNr9X9F1Q/featured>



[Pegasus in 5 Minutes](#)

Bi-monthly basis on second Friday of the month, where we address user questions and also apprise the community of new developments