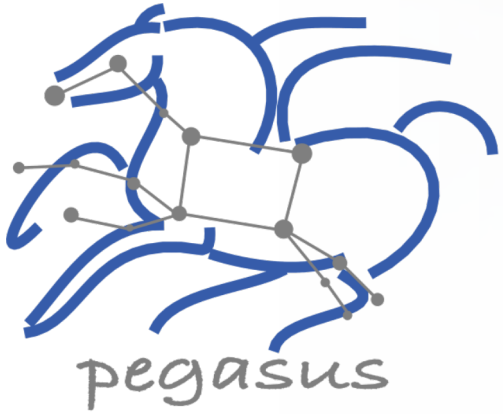


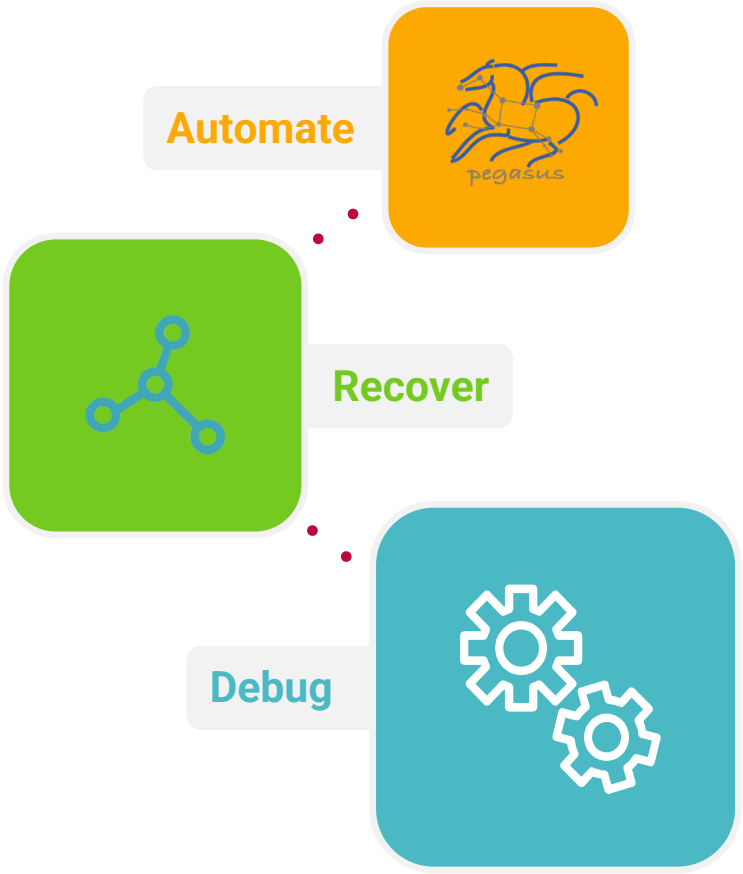
Pegasus 5.0 + Ensemble Manager

Workflow Management System

Karan Vahi

University of Southern California, School of Engineering
Information Sciences Institute
vahi@isi.edu

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

HTCondor High Throughput Computing

NSF funded project since 2001,
with close collaboration with HTCondor team

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.

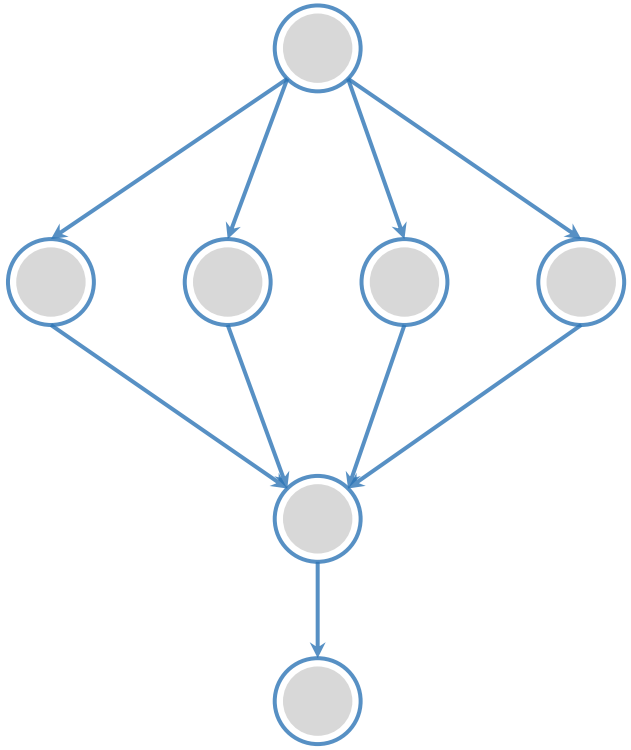
Key Pegasus Concepts

- **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

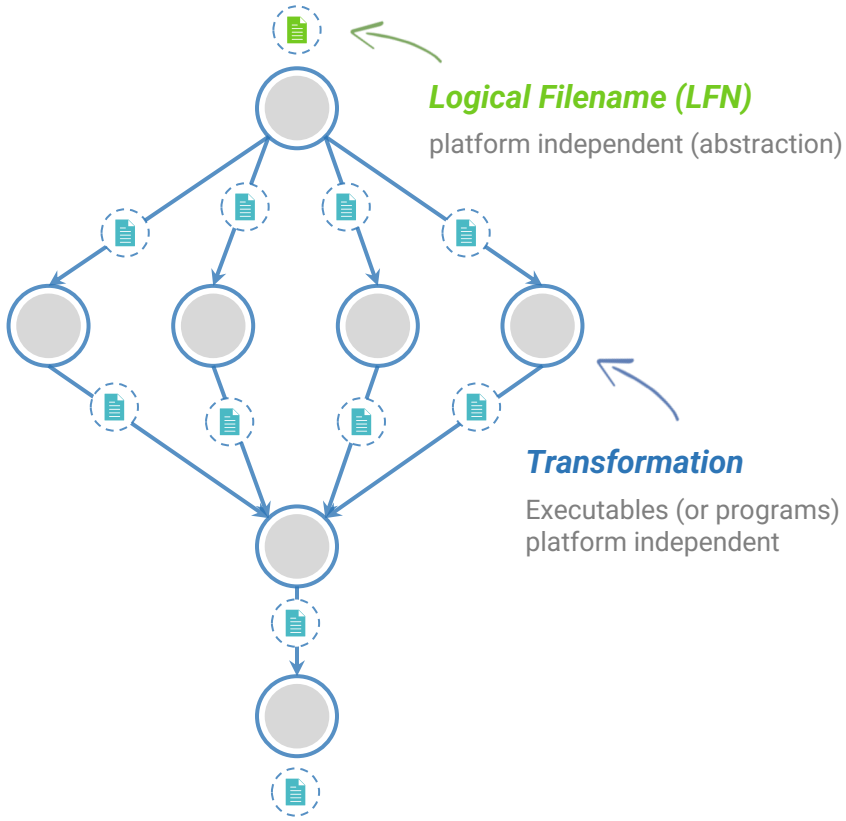
directed-acyclic graphs

Output Workflow

Portable Description

Users do not worry about low level execution details

ABSTRACT WORKFLOW

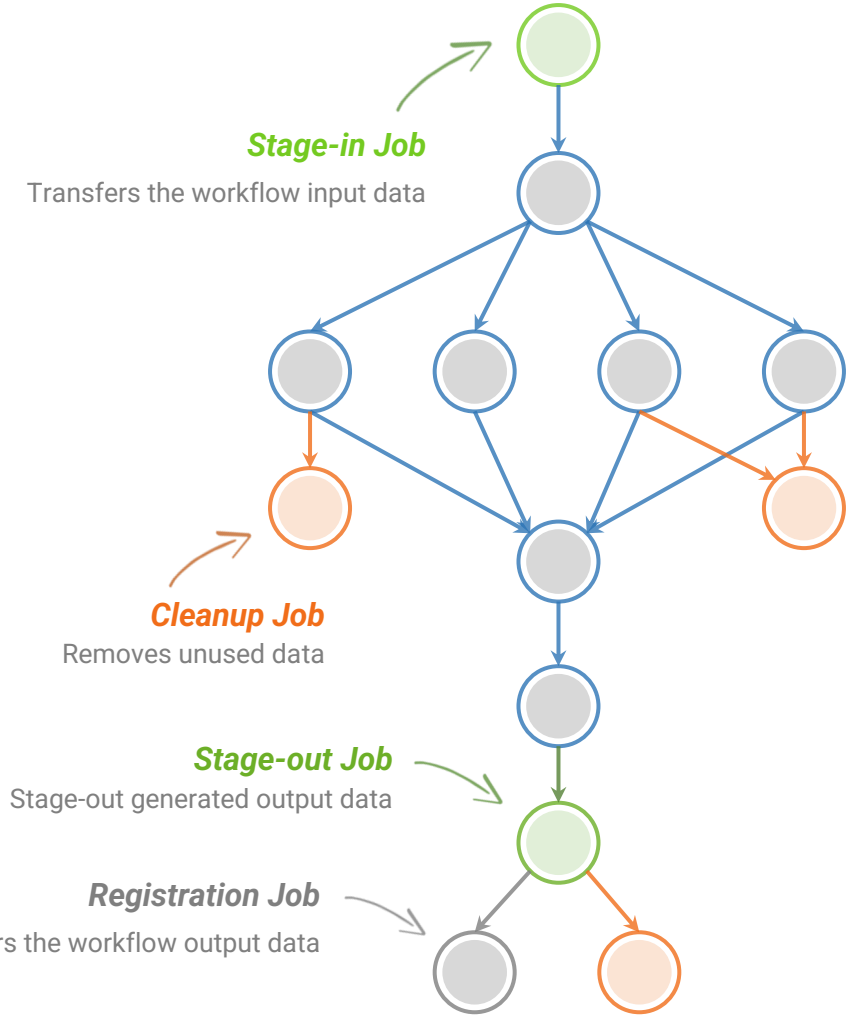


Logical Filename (LFN)

platform independent (abstraction)

Transformation

Executables (or programs)
platform independent



Stage-in Job

Transfers the workflow input data

Cleanup Job

Removes unused data

Stage-out Job

Stage-out generated output data

Registration Job

Registers the workflow output data

EXECUTABLE WORKFLOW

Pegasus Deployment

Workflow Submit Node

- Pegasus WMS
- HTCondor

One or more Compute Sites

- Compute Clusters
- Cloud
- OSG

Input Sites

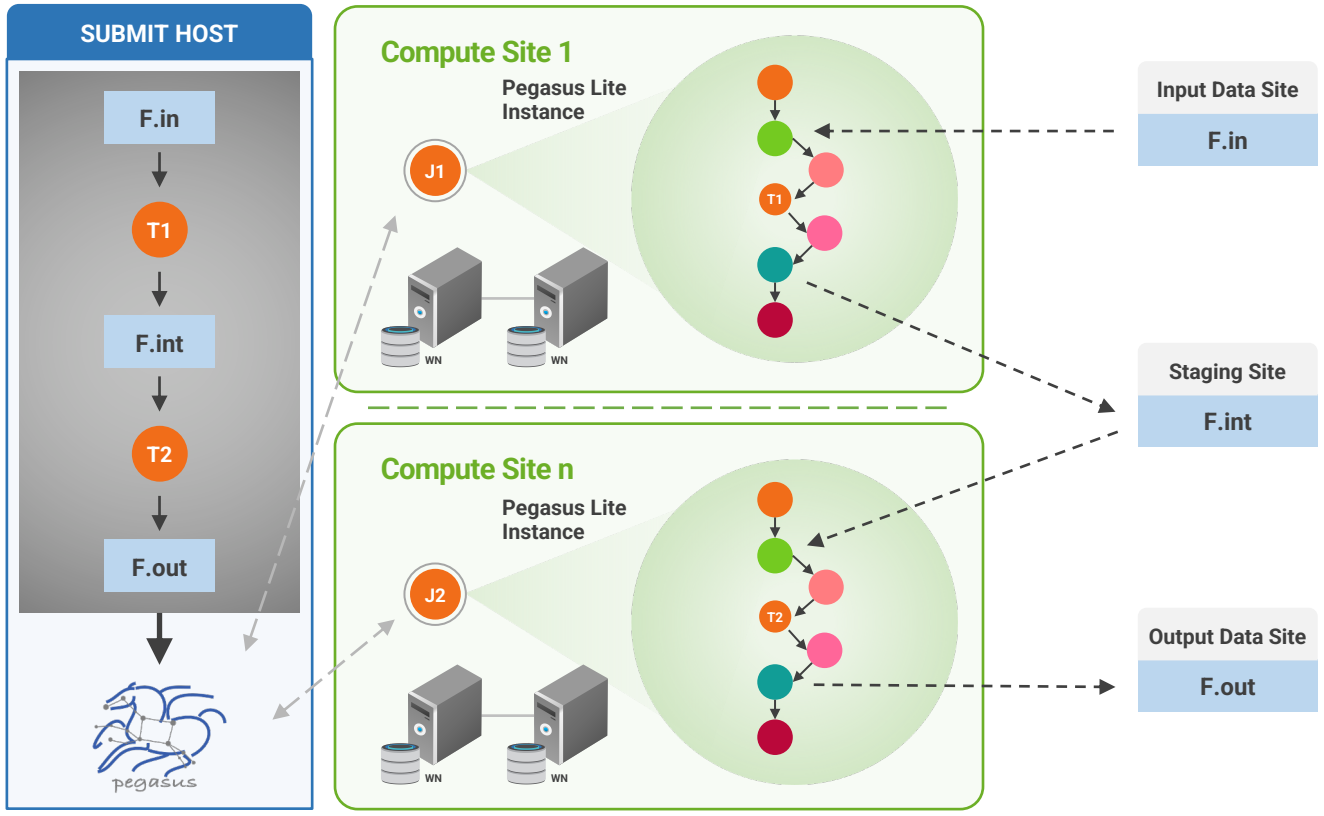
- Host Input Data

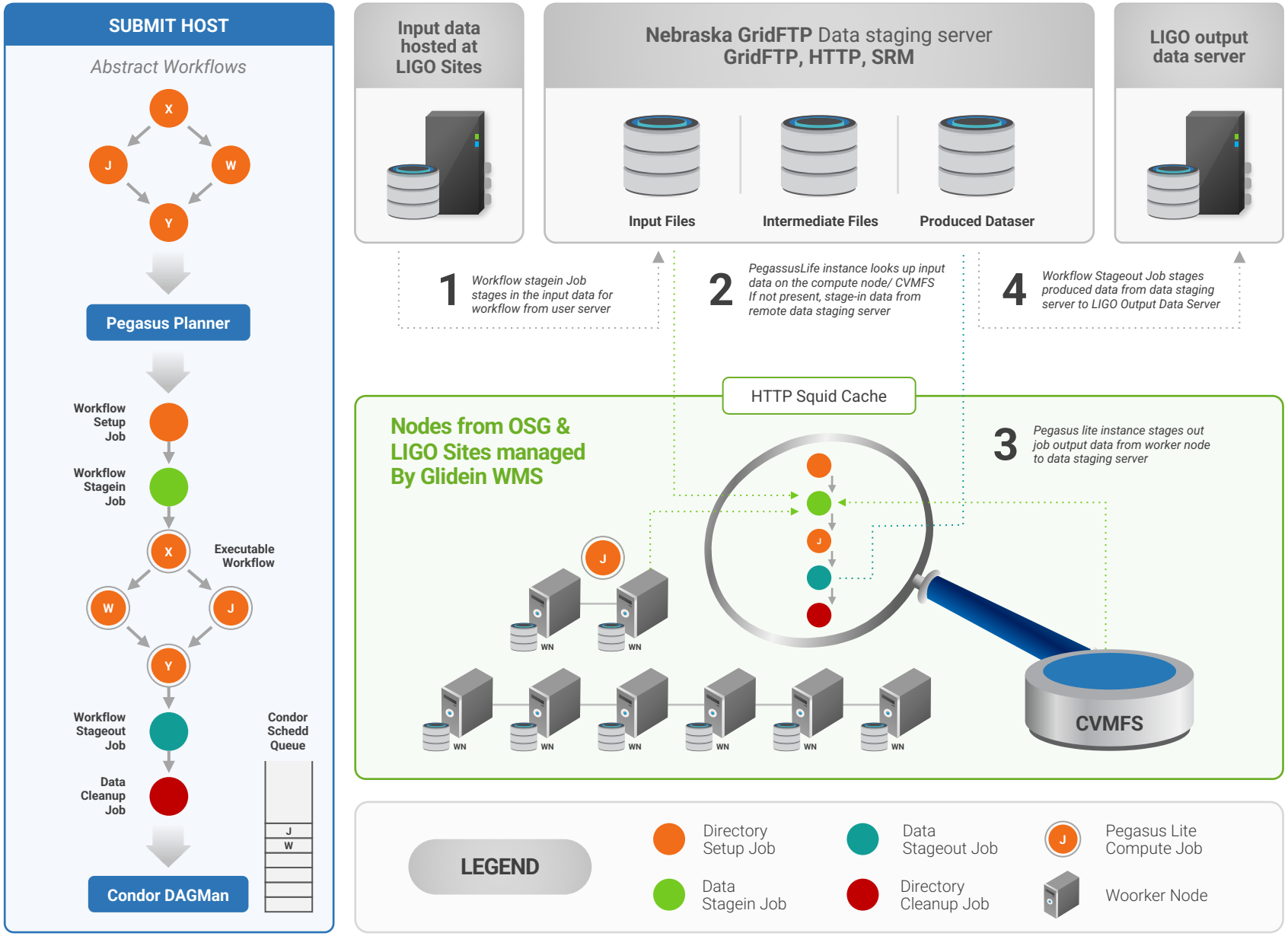
Data Staging Site

- Coordinate data movement for workflow

Output Site

- Where output data is placed





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**

Automatic Integrity Checking in Pegasus

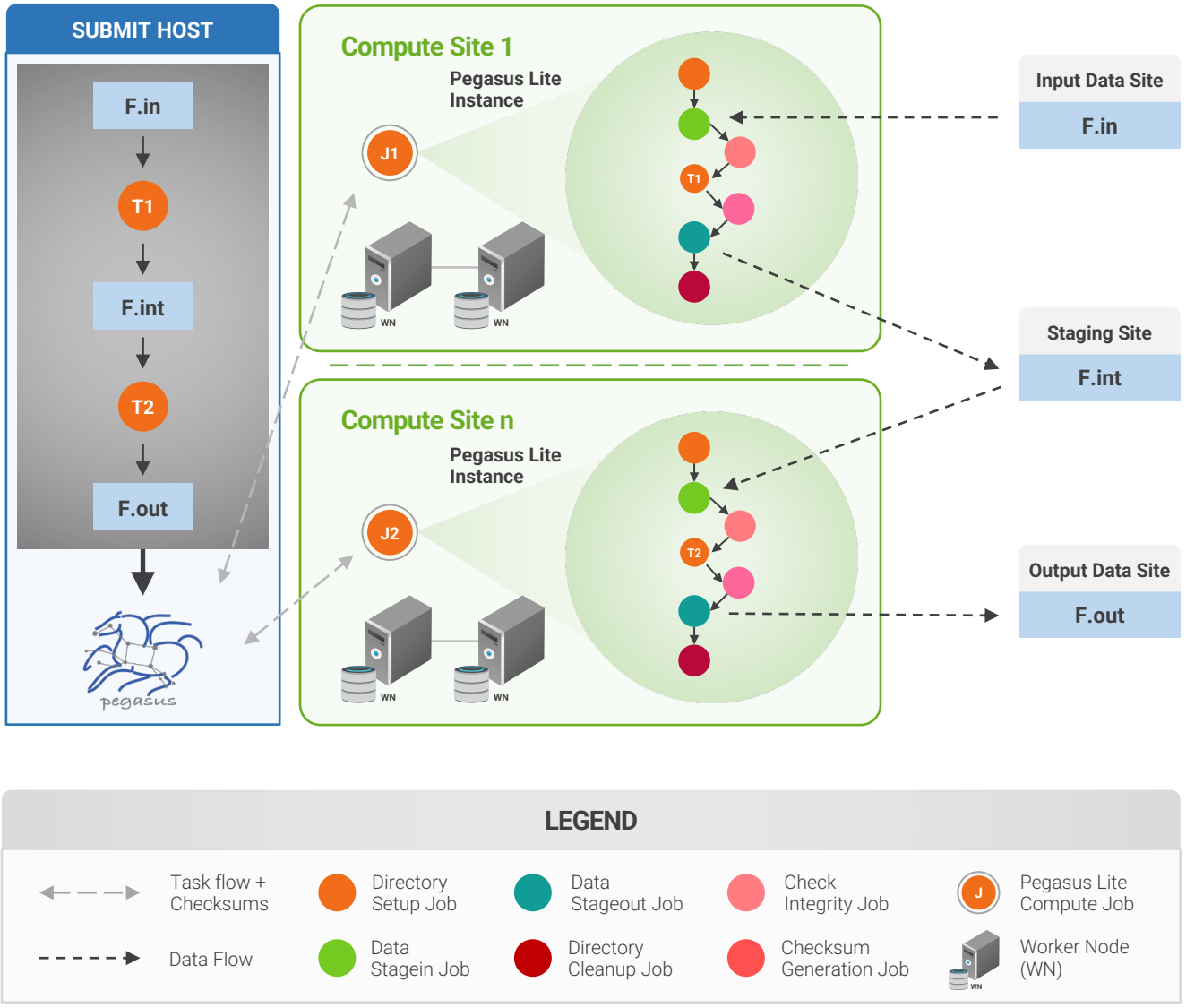
Pegasus performs integrity checksums on input files right before a job starts on the remote node.

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



USC Viterbi School of Engineering Information Sciences Institute

Pegasus 5.0

Released Nov, 2020

- **New and fresh Python3 API to compose, submit and monitor workflows, and configure catalogs**
- **New Catalog Formats**
- **Python 3 Support**
  - All Pegasus tools are Python 3 compliant
  - Python PIP packages for workflow composition and monitoring
- **Zero configuration required to submit to local HTCondor pool.**
- **Data Management Improvements**
  - New output replica catalog that registers outputs including file metadata such as size and checksums
  - Improved support for hierarchical workflows
- **Reworked Documentation and Tutorial**
  - https://pegasus.isi.edu/documentation/

```
#!/usr/bin/env python3
import logging
import sys

from Pegasus.api import *

# logs to be sent to stdout
logging.basicConfig(level=logging.DEBUG, stream=sys.stdout)

# --- Transformations ----------------------------------------
echo = Transformation(
        "echo",
        pfn="/bin/echo",
        site="condorpool"
    )

tc = TransformationCatalog()\
        .add_transformations(echo)

# --- Workflow -----------------------------------------------
Workflow("hello-world", infer_dependencies=True)\
    .add_jobs(
        Job(echo)
            .add_args("Hello World")
            .set_stdout("hello.out")
    ).add_transformation_catalog(tc)\
    .plan(submit=True)\
    .wait()
```

Pegasus 5.0

- **Zero configuration required to submit to local HTCondor pool.**
  - The *"hello world"* example on the right will work out of the box
  - Pegasus will automatically create sensible defaults for sites
    - local
    - condorpool
  - By default, site *"condorpool"* is used as execution site.
  - Site *"local"* still designates the submit node, and is used to run Pegasus auxillary jobs.

```
#!/usr/bin/env python3
import logging
import sys

from Pegasus.api import *

# logs to be sent to stdout
logging.basicConfig(level=logging.DEBUG, stream=sys.stdout)

# --- Transformations ----------------------------------------
echo = Transformation(
        "echo",
        pfn="/bin/echo",
        site="condorpool"
    )

tc = TransformationCatalog()\
        .add_transformations(echo)

# --- Workflow -----------------------------------------------
Workflow("hello-world", infer_dependencies=True)\
    .add_jobs(
        Job(echo)
            .add_args("Hello World")
            .set_stdout("hello.out")
    ).add_transformation_catalog(tc)\
    .plan(submit=True)\
    .wait()
```

Ensemble Manager

Allow users to submit a collection of workflows (ensembles)

Automatically **spawn** and **manage** collections of workflows

Trigger submission of workflows

Properties

Workflows within an ensemble may have **different priorities**

> *Priorities can also be changed at runtime*

Ensembles may limit the number of **concurrent** planned and running workflows

Additional Actions

Ensembles can be **paused, resumed, removed, re-planned**, and **re-executed**
A **debugging** mechanism is also provided to investigate failures in workflow runs
Actions can be performed both to ensembles and single workflows within ensembles

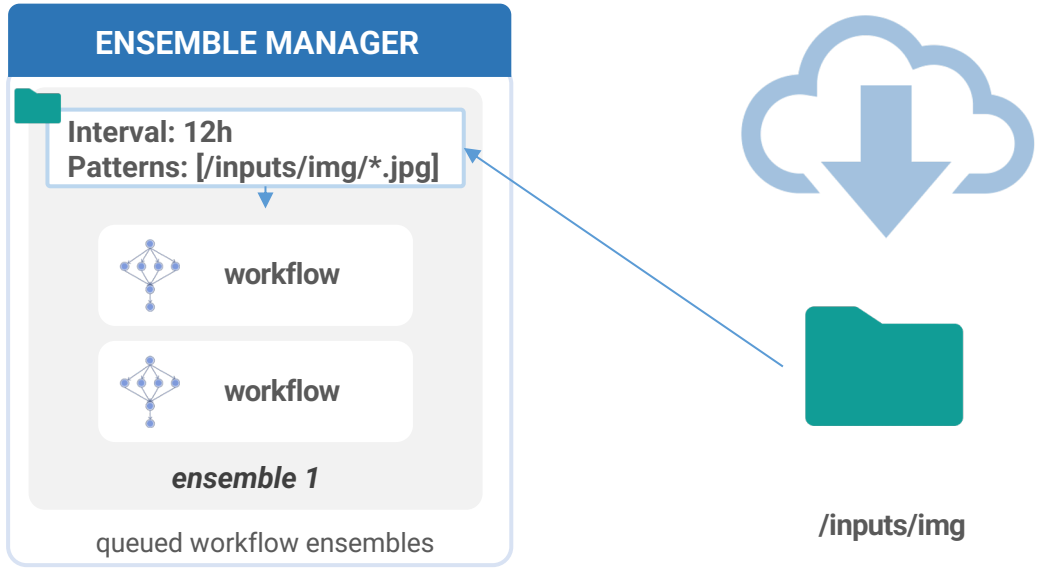
Ensemble Manager Triggers

Cron workflow trigger

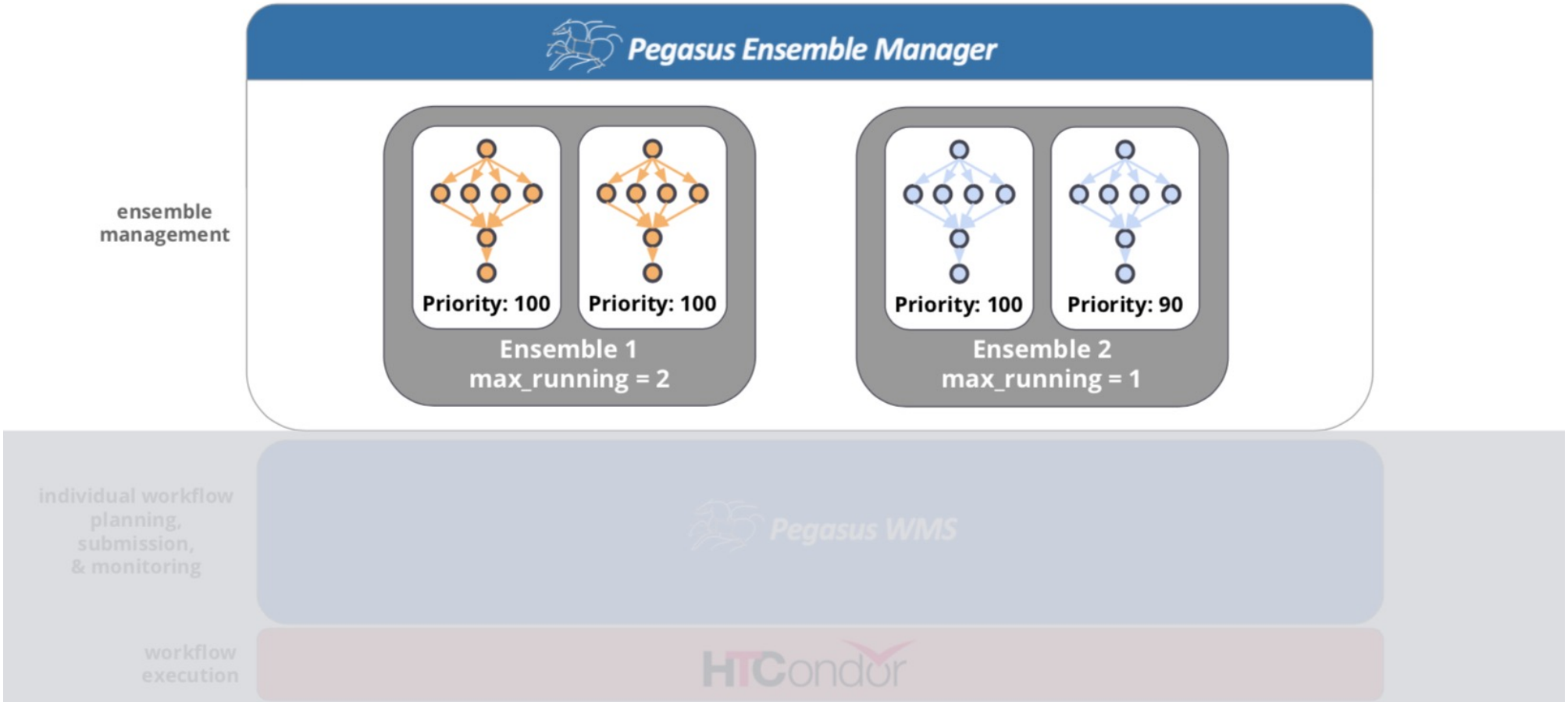
Automatically submit workflows to the ensemble manager at **regularly occurring time intervals**

File pattern workflow trigger

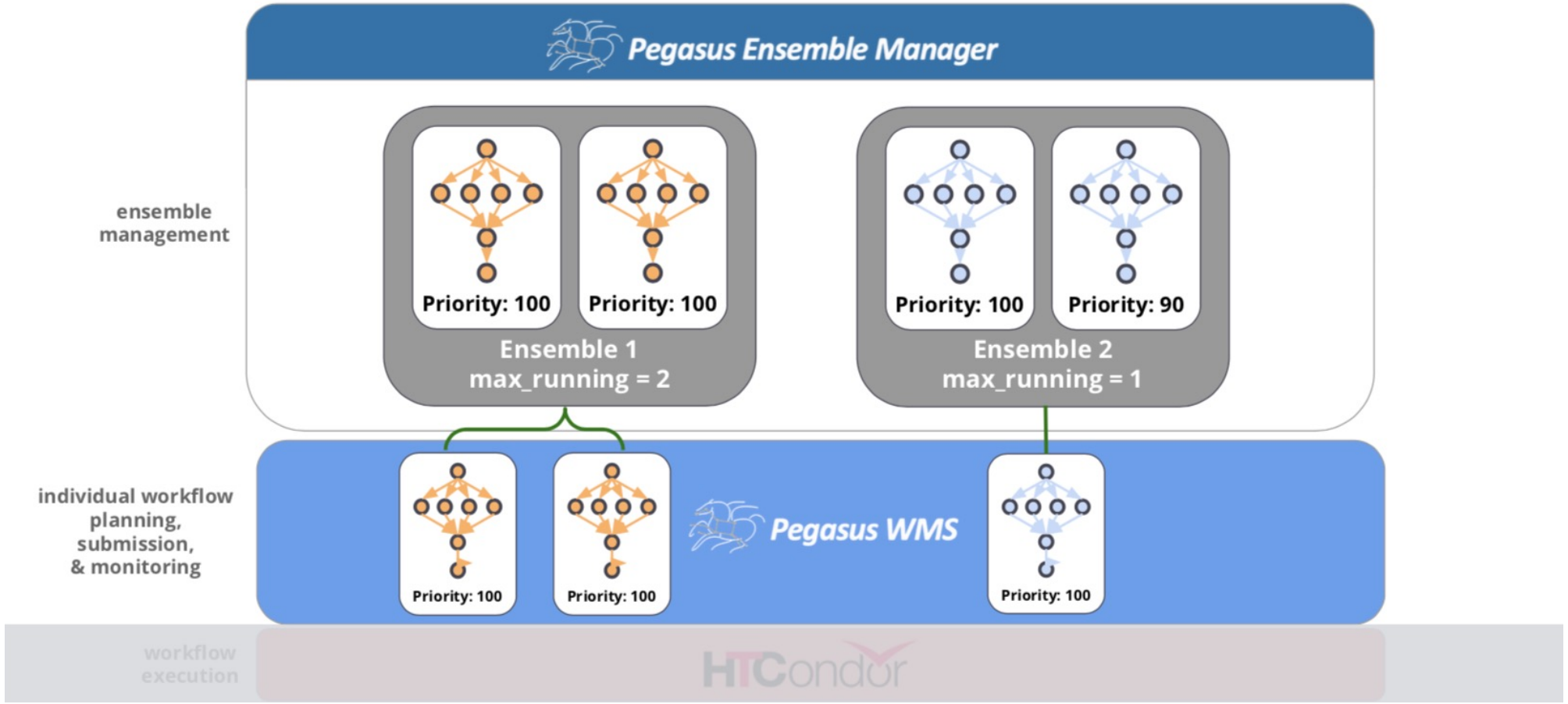
Cron trigger functionality
New **input files matching a given file pattern(s) will be passed** as input
Ideal for **regular batch processing** of data as it arrives in one or more given directories



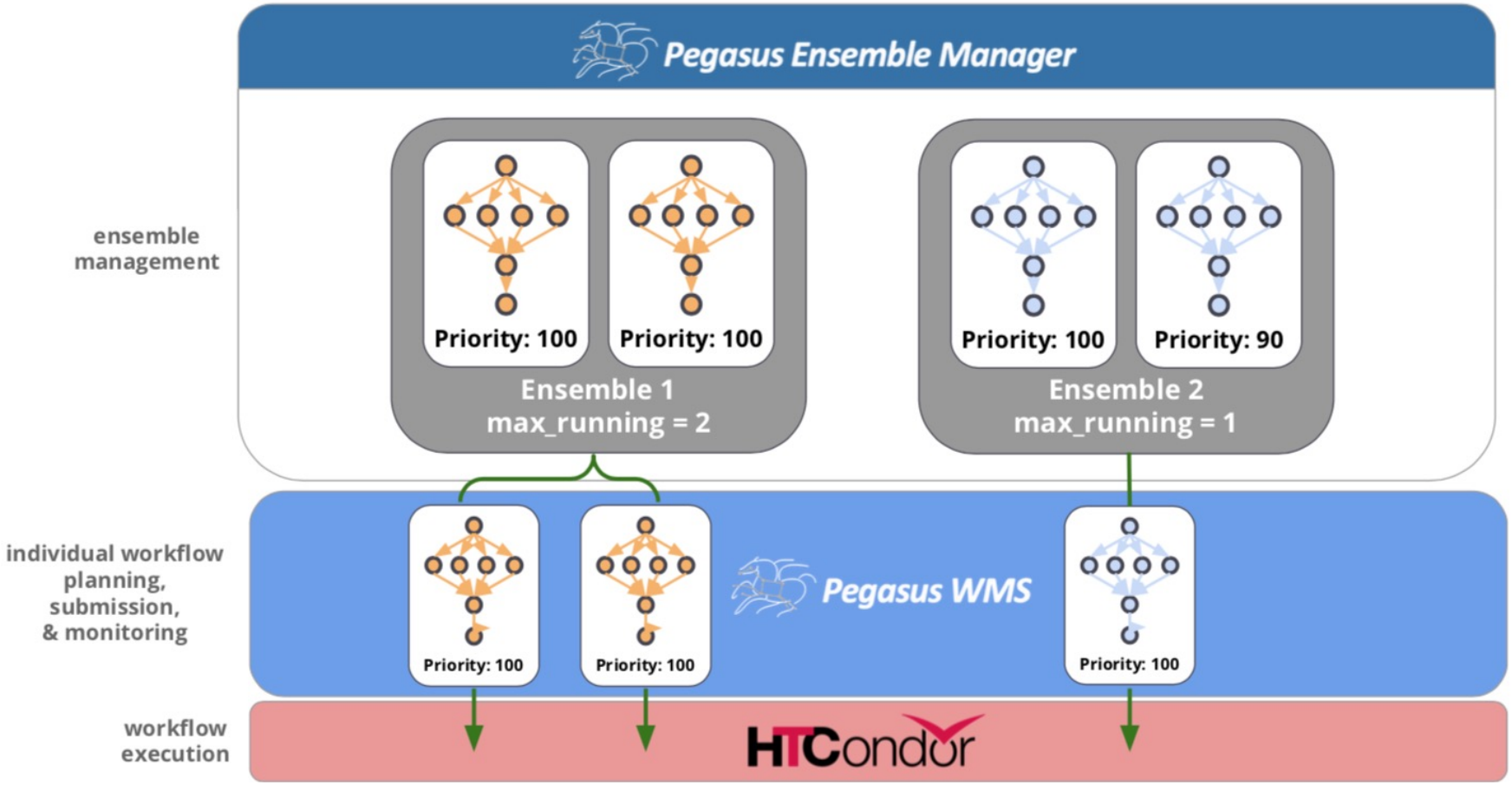
Ensemble Manager Overview



Ensemble Manager Overview

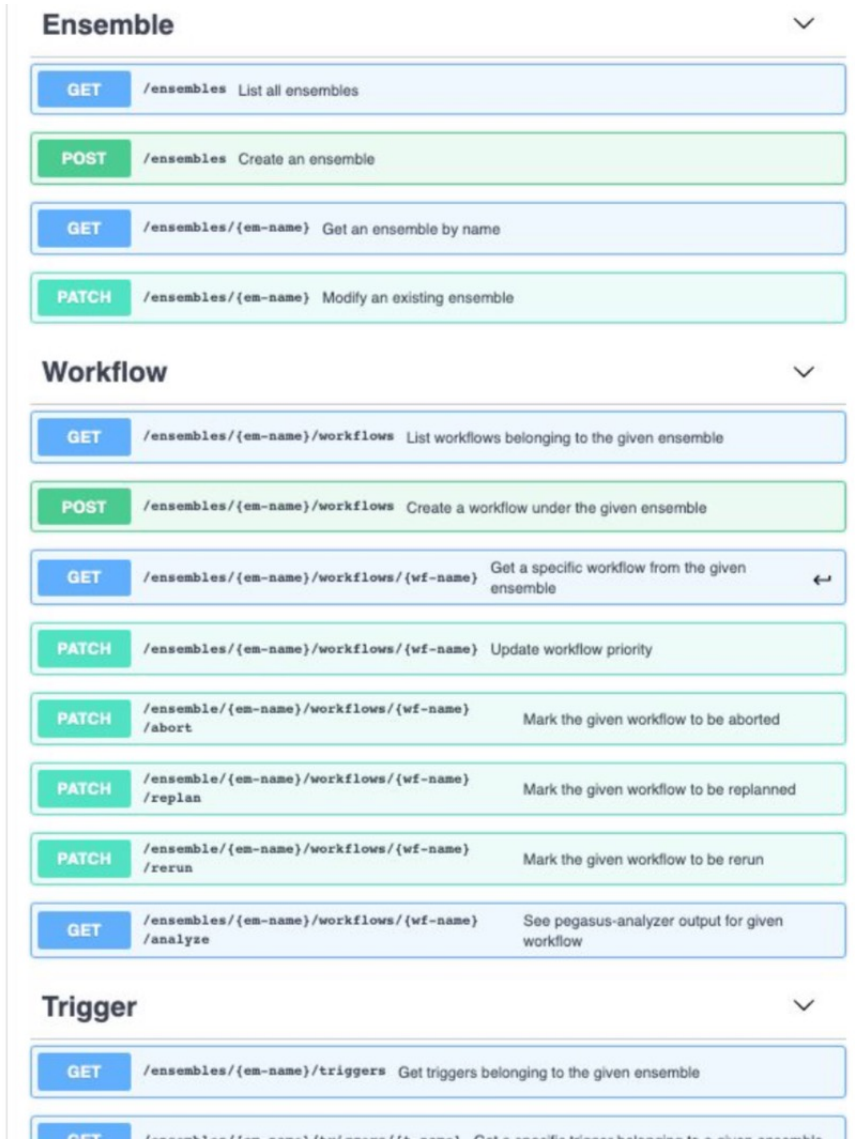


Ensemble Manager Overview



Ensemble Manager: Rest API

- Exposing the ensemble manager as a REST endpoint
- Provide create, read, update, delete operations on ensembles, workflows, and triggers
- Provide Python and Java client code
- Improve support for integrating ensemble manager into larger systems



Pegasus

est. 2001

Automate, recover, and debug scientific computations.

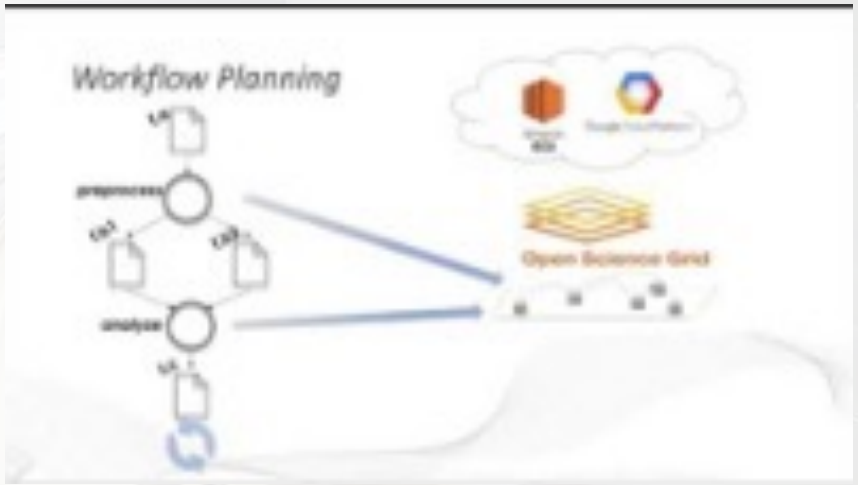
Get Started

- **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/

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

YouTube Channel

https://www.youtube.com/channel/UCwJQln1CqBvTJqiNr9X9F1Q/featured



Pegasus in 5 Minutes