NAIRR Pilot

National Artificial Intelligence Research Resource Pilot

Workflows and AI

Mats Rynge, USC Information Sciences Institute / NSF ACCESS

April 3 / Track 1

AI Workshop Denver, CO April 2-3, 2025

This Session

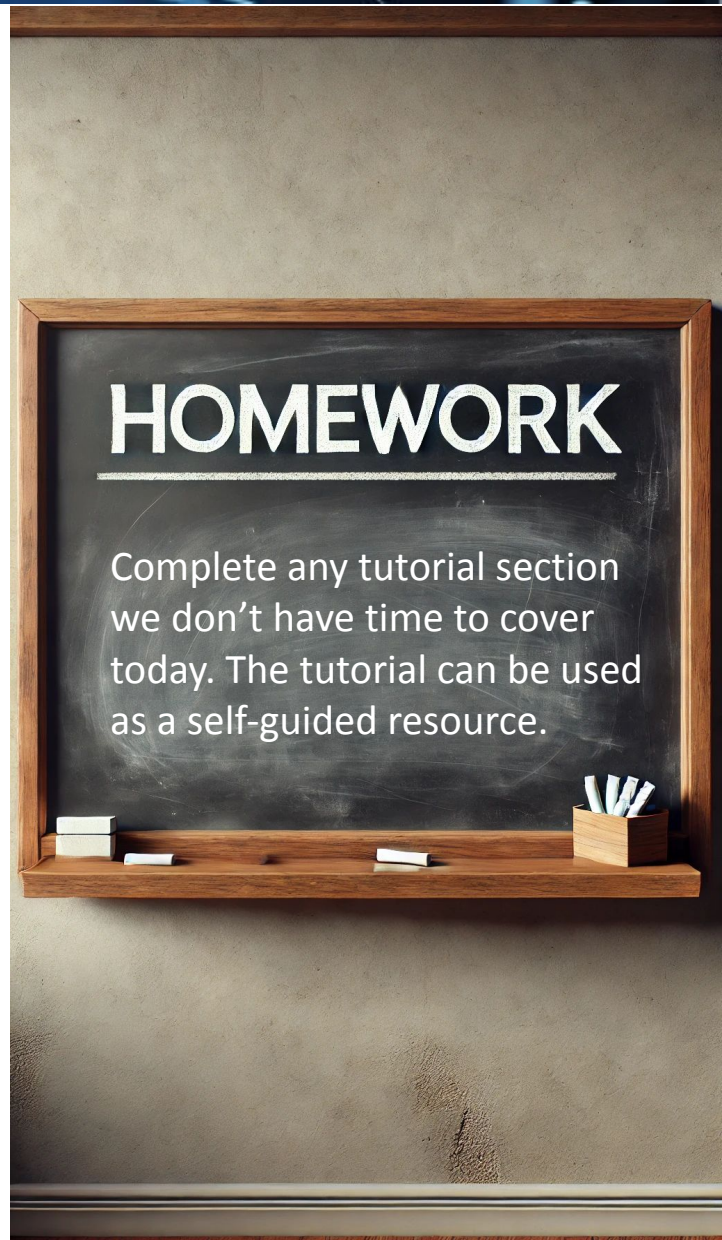
Main takeaways:

1. Use a workflow system for your AI workloads

2. Scheduler overlay to simplify resource management

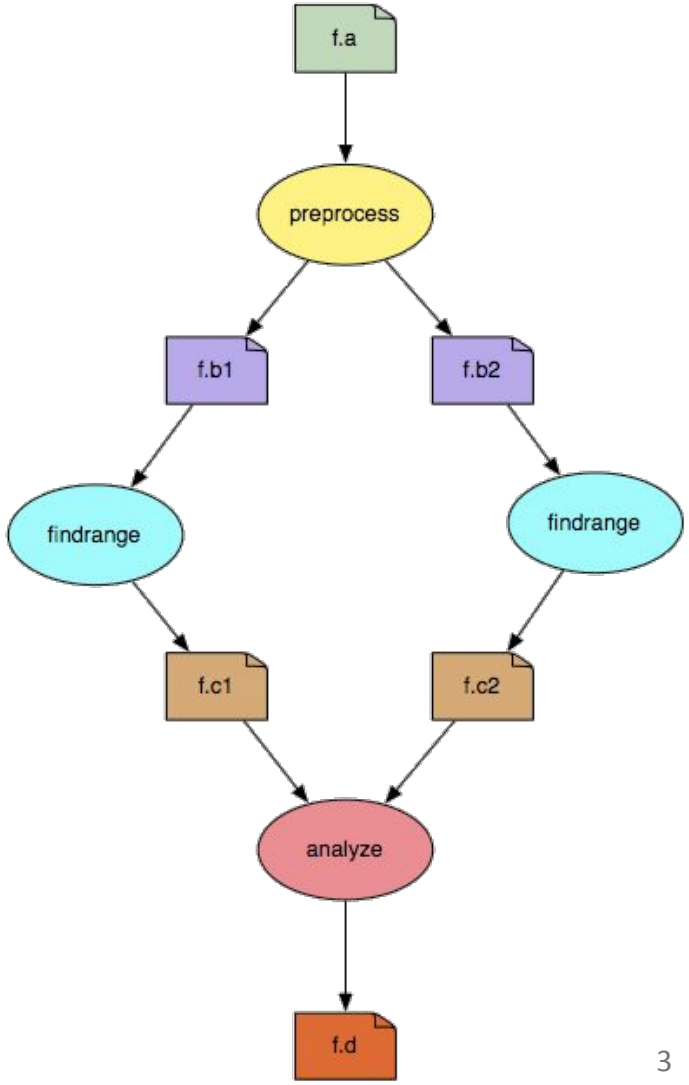
Why ACCESS Pegasus?

Pegasus can be deployed and used in a variety of different ways, such as cli/Jupyter, local install on your machine, a cluster or cloud. One option is a hosted system like the ACCESS Pegasus one we are using today.

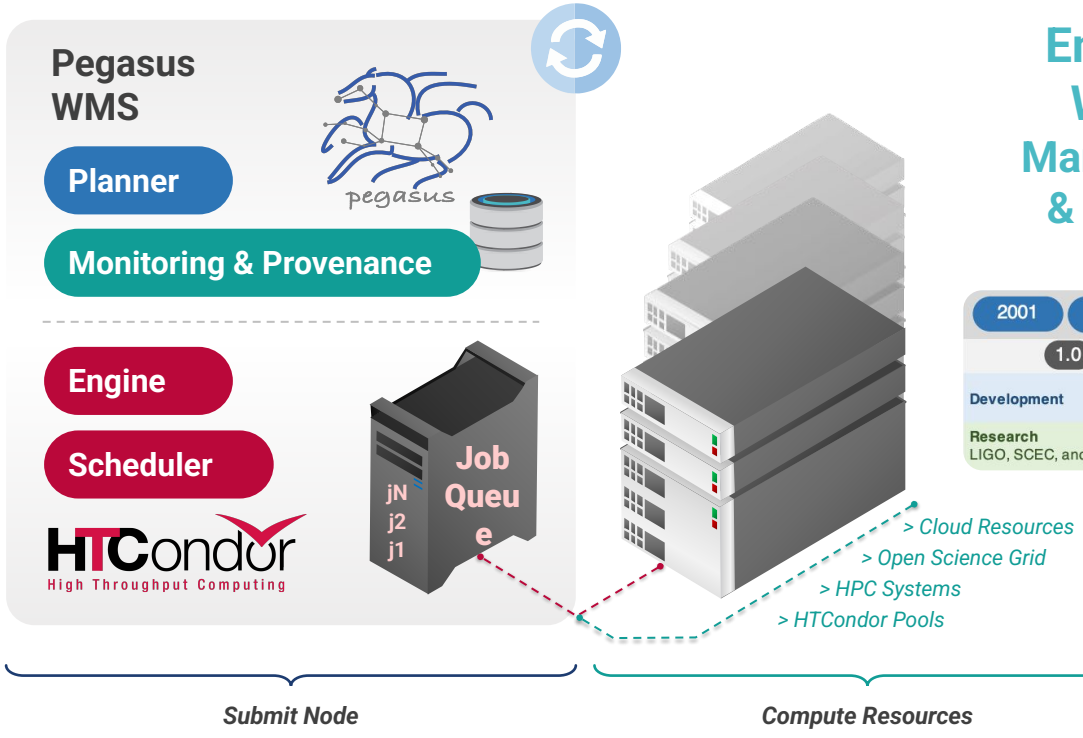


Scientific Workflows

- An abstraction to express ensemble of complex computational operations
  - *Eg: retrieving data from remote storage services, executing applications, and transferring data products to designated storage sites*
- A workflow is represented as a directed acyclic graph (DAG)
  - *Nodes: tasks or jobs to be executed*
  - *Edges: depend between the tasks*
- Have a monolithic application/experiment?
  - *Find the inherent DAG structure in your application to convert into a workflow*

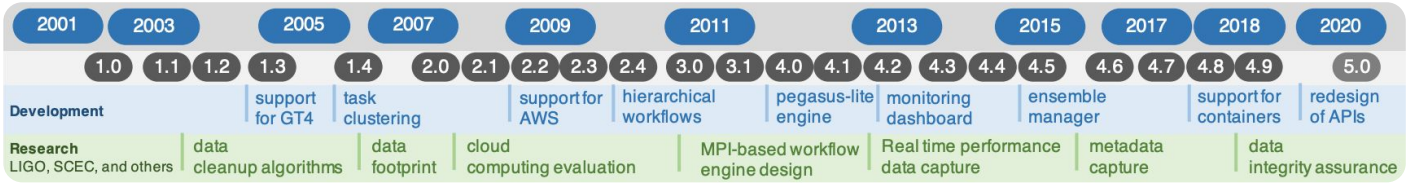


Pegasus Workflow Management System



End to End Workflow Management & Execution

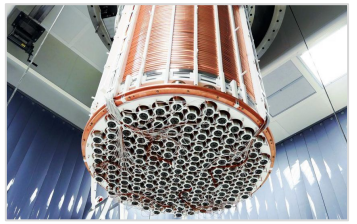
- Develop portable scientific workflows in Python, Java, and R
- Compile workflows to be run on heterogeneous resources
- Monitor and debug workflow execution via CLI and web-based tools
- Recover from failures with built-in fault tolerance mechanisms
- Regular release schedule incorporating latest research and development



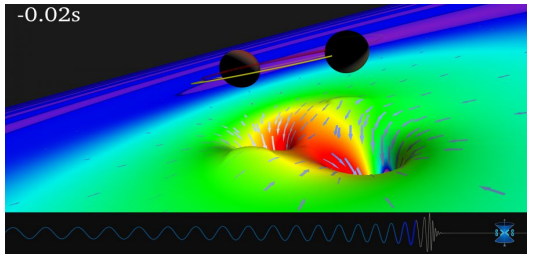
Pegasus in practice

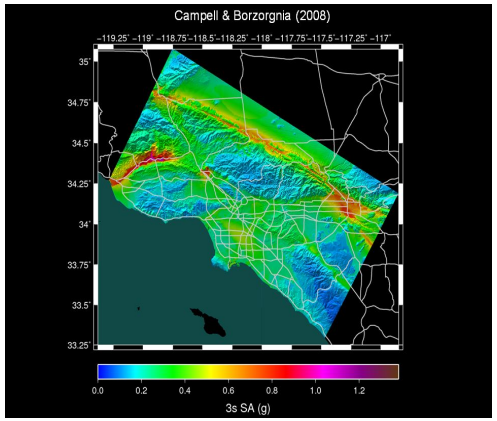
- Laser Interferometer Gravitational Wave Observatory (LIGO) develops large scale analysis pipelines used for gravitational wave detection.
- Southern California Earthquake Center (SCEC) CyberShake project generates hazard maps using hierarchical workflows .
- The XENONnT project uses Pegasus for processing and monte carlo workflows, searching for dark matter

The XENONnT detector



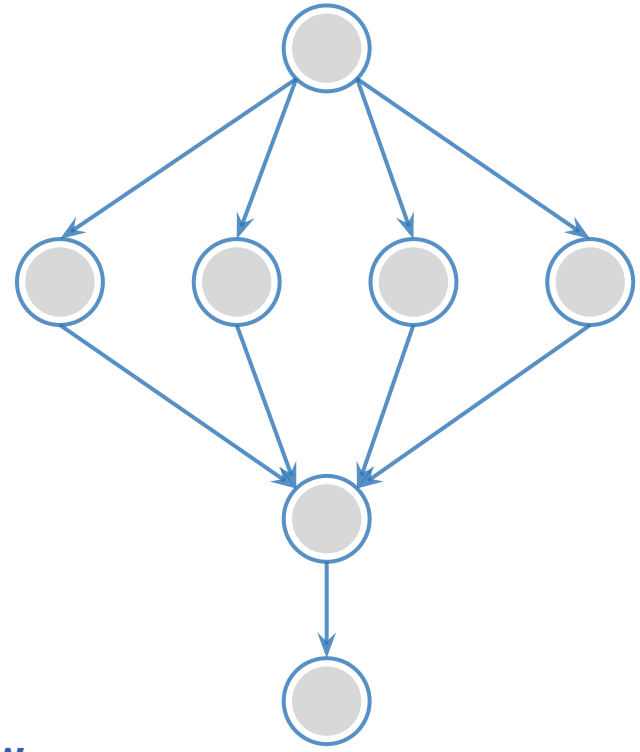
LIGO observation of colliding black holes





Hazard map indicating maximum amount of shaking at a particular geographic location generated from SCEC's CyberShake Pegasus workflow

- **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

- **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

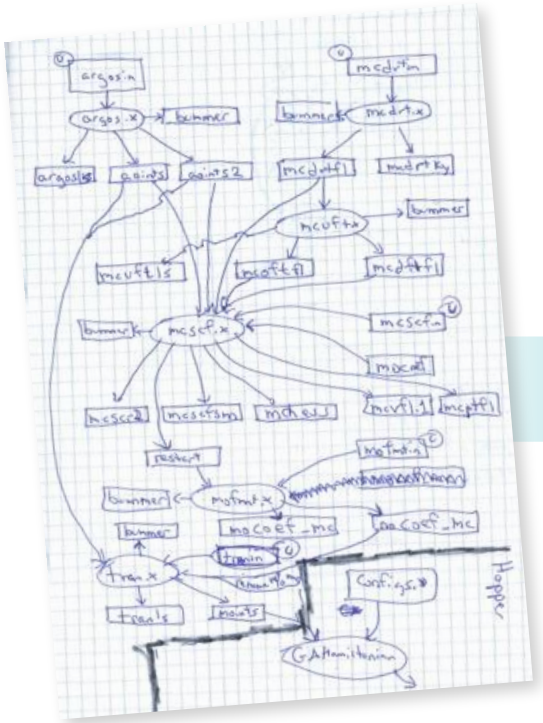
- **Workflows are DAGs**
  - Nodes: jobs, edges: dependencies
  - No while loops, no conditional branches
  - Jobs are standalone executables

- **Planning occurs ahead of execution**

- **Planning converts an abstract workflow into a concrete, executable workflow**
  - Planner is like a compiler

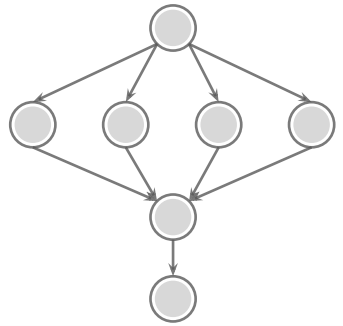
Pegasus reasons about data, in order to do automatic placement, movement, and integrity.

Pegasus provides APIs to generate the Abstract Workflow



```python
#!/usr/bin/env python3

import os
import logging
from pathlib import Path
from argparse import ArgumentParser

logging.basicConfig(level=logging.DEBUG)

# --- Import Pegasus API ------------------
from Pegasus.api import *

# --- Create Abstract Workflow ------------
wf = Workflow("pipeline")

webpage = File("pegasus.html")

# --- Create Parent Job -------------
curl_job = (
    Job("curl")
    .add_args("-o", webpage, "http://pegasus.isi.edu")
    .add_outputs(webpage, stage_out=False, register_replica=False)
)

count = File("count.txt")

# --- Create Dependent Job ----------
wc_job = (
    Job("wc")
    .add_args("-l", webpage)
    .add_inputs(webpage)
    .set_stdout(count, stage_out=True, register_replica=True)
)
# --- Add jobs to the Abstract Workflow -------
wf.add_jobs(curl_job, wc_job)

# --- Add control flow dependency ----------
wf.add_dependency(wc_job, parents=[curl_job])

# --- Write out the Abstract Workflow ----------
wf.write()
```

python™ Java jupyter



Abstract Workflow

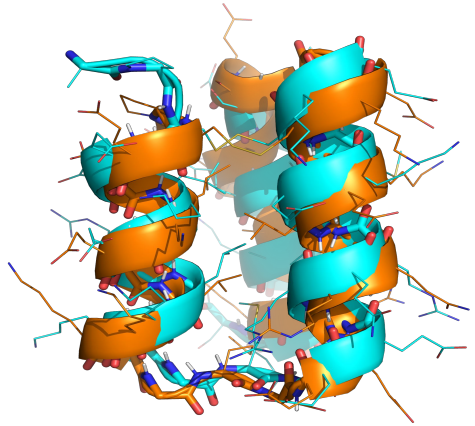
```yaml
x-pegasus:
  apiLang: python
  createdBy: vahi
  createdOn: 11-19-20T14:57:58Z
pegasus: '5.0'
name: pipeline
jobs:
- type: job
  name: curl
  id: ID0000001
  arguments:
  - -o
  - pegasus.html
  - http://pegasus.isi.edu
  uses:
  - lfn: pegasus.html
    type: output
    stageOut: false
    registerReplica: false
- type: job
  name: wc
  id: ID0000002
  stdout: count.txt
  arguments:
  - -l
  - pegasus.html
  uses:
  - lfn: count.txt
    type: output
    stageOut: true
    registerReplica: true
  - lfn: pegasus.html
    type: input
jobDependencies:
- id: ID0000001
  children:
  - ID0000002
```

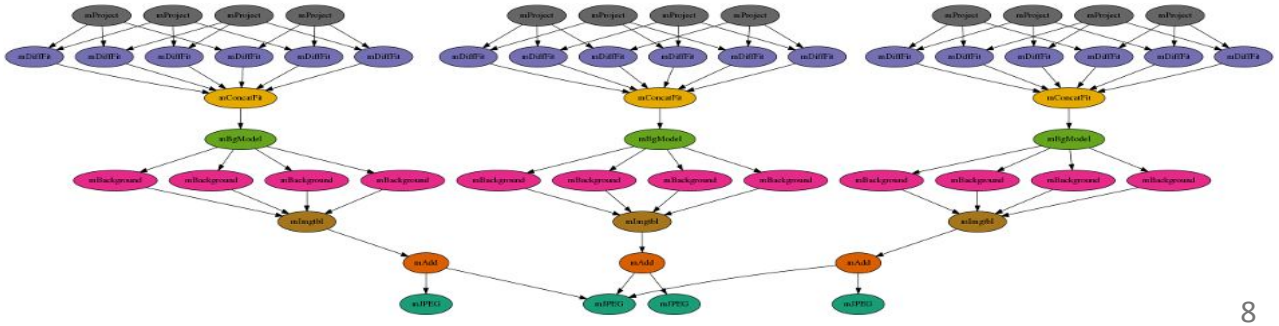
YAML Formatted

Example Workflows

In addition to tutorial workflows, a set of example workflows are automatically installed into each user account - easy to explore, execute and modify!

- Artificial Intelligence
  - Lung Segmentation
  - Mask Detection
  - Orca Sound
  - LLM + RAG
- Astronomy
  - Montage
- Bioinformatics
  - Alphafold
  - Rosetta
  - VariantCalling





LLM - RAG

LLM RAG (Large Language Model Retrieval-Augmented Generation) is a technique that enhances large language models by incorporating information retrieval mechanisms.

It involves retrieving relevant information from a database or document corpus and combining it with the original query to provide additional context. This augmented input is then processed by the large language model to generate more accurate and contextually relevant responses.

LLM RAG offers benefits such as improved accuracy, access to up-to-date information, and better contextual understanding, making it useful for applications like question answering, summarization, and conversational AI.

1. `gpu96`: This flavor provides 4 NVIDIA V100 GPUs (with 32GB of memory each) and 128GB of RAM on the host CPU node.
2. `gpu60`: This flavor offers 2 NVIDIA V100 GPUs (each with 32GB of memory) and 64GB of RAM on the host CPU node.

The Jetstream2 system offers several GPU instance flavors: g3.small, g3.medium, g3.large, and g3.xl. Here's a summary of their specifications:

- g3.small: 4 vCPUs, 15 GB RAM, 60 GB local storage, 20% GPU compute, 20 GB GPU RAM
- g3.medium: 8 vCPUs, 30 GB RAM, 60 GB local storage, 25% GPU compute, 10 GB GPU RAM
- g3.large: 16 vCPUs, 60 GB RAM, 60 GB local storage, 50% GPU compute, 20 GB GPU RAM
- g3.xl: 32 vCPUs, 125 GB RAM, 60 GB local storage, 100% GPU compute, 40 GB GPU RAM

Tutorial: LLM - RAG

Goal: Execute a set of LLMs, locally on NAIRR resources, and use RAG to augment the models and answer a set of predefined prompts.

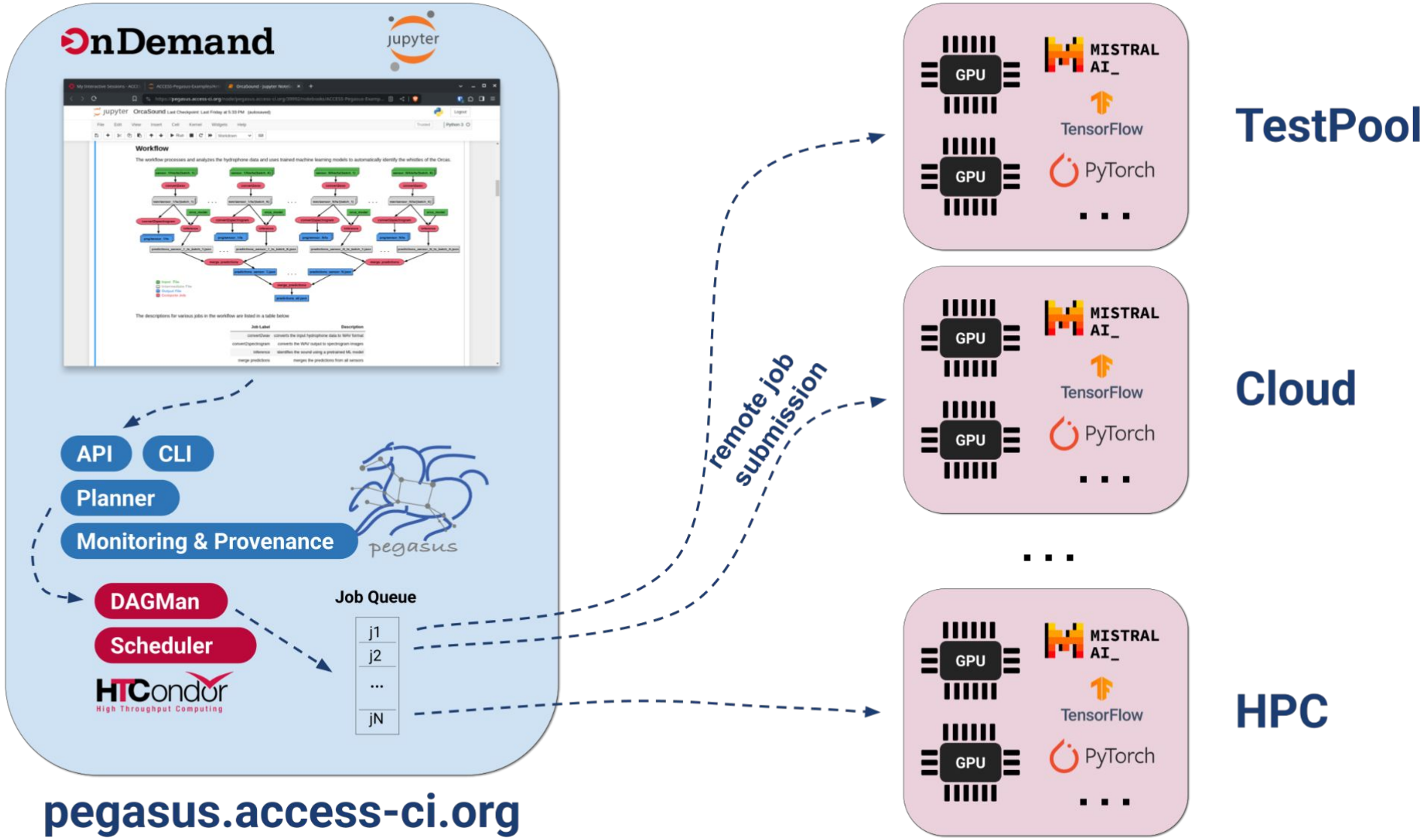
Container includes full LLM and toolchain (Mistral, LangChain, Chroma)

Workflow Input: Public domain books (Project Gutenberg)

Prompts:

- Please provide a one paragraph summary of the book.
- Who is the protagonist in the book?
- Who is the antagonist in the book?
- What time period is the book set in?

Can execute on GPUS from any of the capacity providers: TestPool, Cloud, HTCondor Annex, OSPool

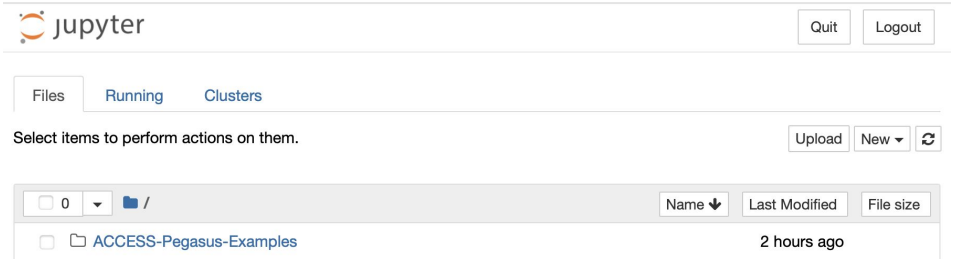

OnDemand
jupyter
Workflow
API
CLI
Planner
Monitoring & Provenance
pegasus
DAGMan
Scheduler
HTCondor
High Throughput Computing
Job Queue
j1
j2
...
jN
remote job submission
GPU
GPU
MISTRAL AI_
TensorFlow
PyTorch
TestPool
Cloud
HPC
pegasus.access-ci.org

Tutorial Setup

Set of Jupyter Notebooks, contains a mix of overview, pointers, workflows. Step through them one cell at the time, starting from the top.

Log in to ACCESS Pegasus, and start a Jupyter Notebook.

Tutorials can be found in the ***ACCESS-Pegasus-Examples*** directory.



Using your ACCESS account, verify that you can use a web browser to log into:

https://pegasus.access-ci.org

Hands on: Running our first workflow

https://pegasus.access-ci.org

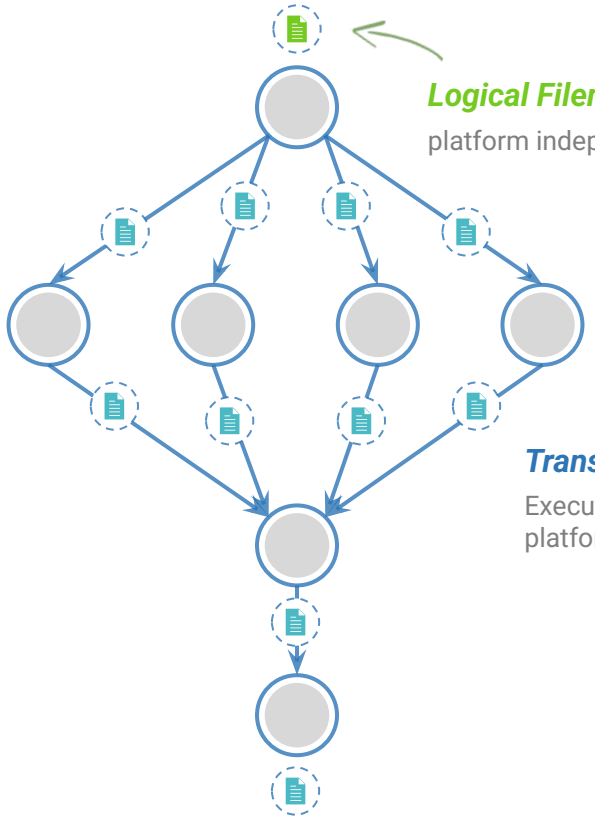
Input Workflow Specification YAML formatted

Output Workflow

Portable Description

Users do not worry about low level execution details

ABSTRACT WORKFLOW

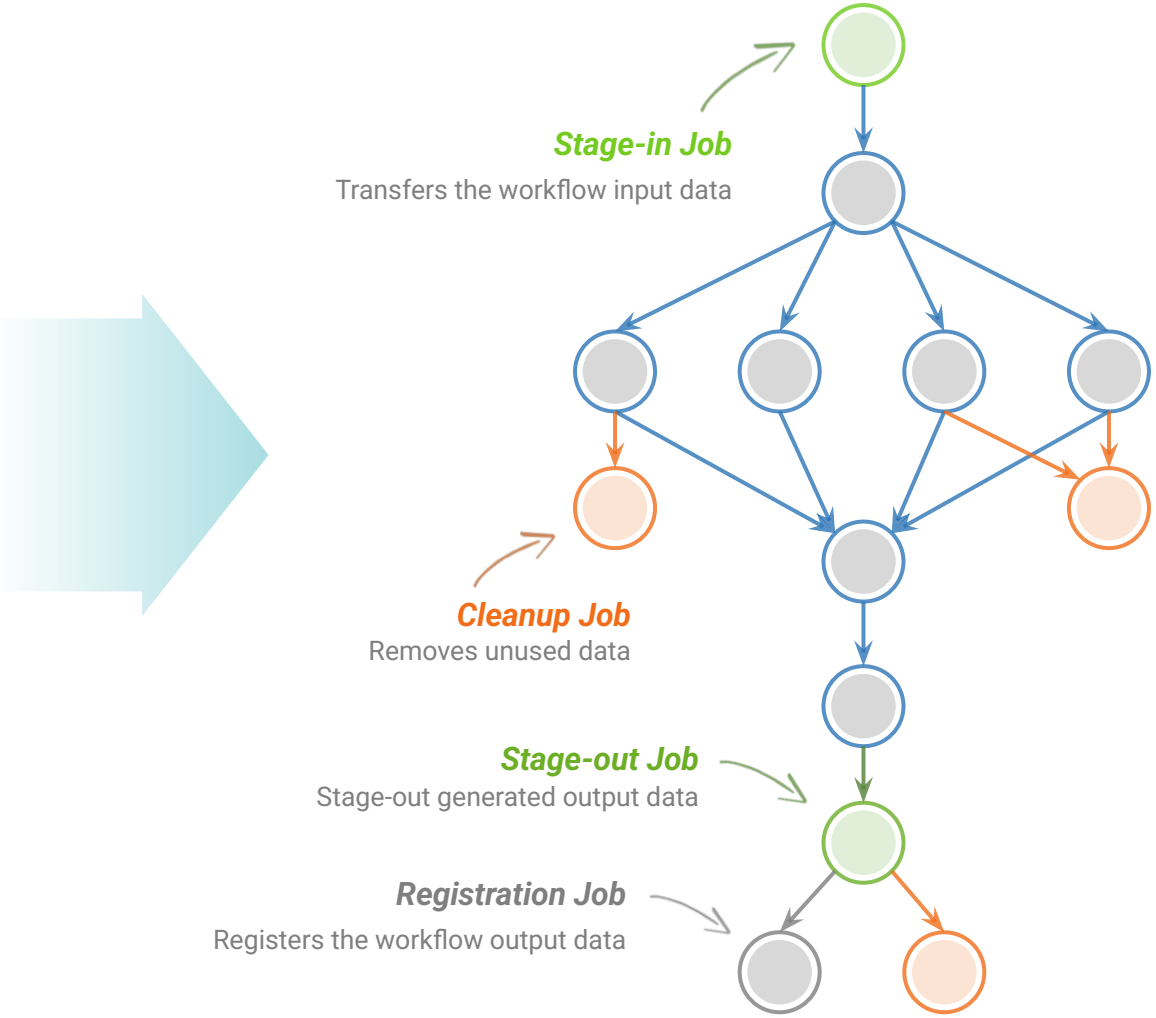


Logical Filename (LFN)

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

So, what other information does Pegasus need?

Site Catalog

Describes the sites where
The workflow jobs are to be executed

Transformation Catalog

Describes all of the executables
(called “transformations”)
used by the workflow

Replica Catalog

Describes all of the input data stored
on external servers

Hands on: Catalogs

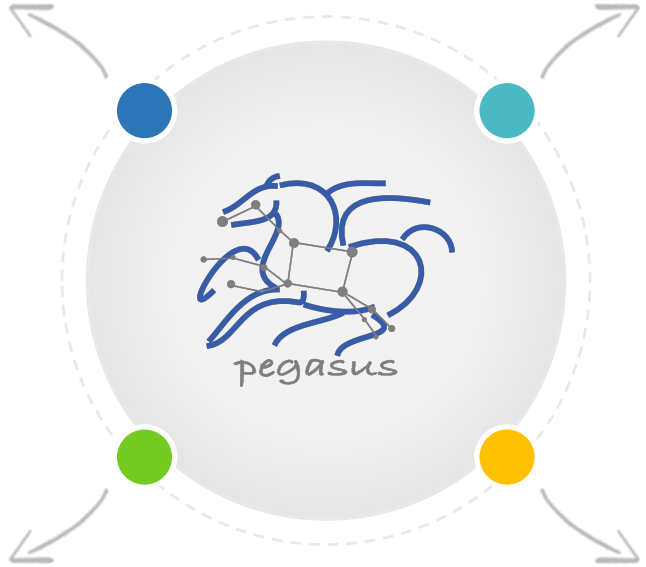
And if a job fails?

Postscript

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

Job Retry

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



Checkpoint Files

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

Rescue DAGs

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

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

Hands on: Statistics

Hands on: Statistics and Debugging

NAIRR Pilot

National Artificial Intelligence Research Resource Pilot

Resources Overlay

Advantages of Glidein (Pilot) Jobs in HTCondor

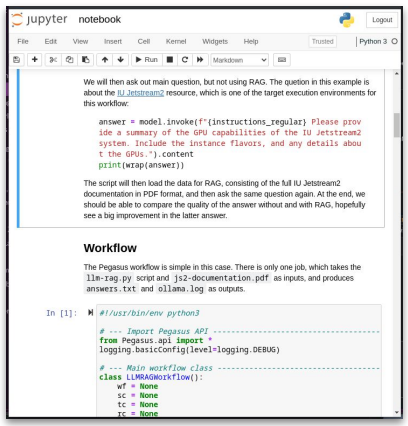
A glidein (pilot) job in HTCondor is a placeholder job that starts on a remote resource, sets up the HTCondor environment, and transforms the resource into a temporary worker node ready to run user jobs from the central HTCondor pool.

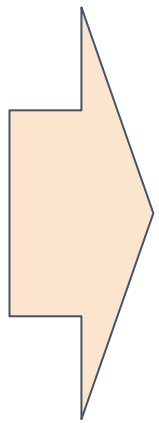
Dynamic Resource Provisioning: Jobs start only when needed, reducing idle time.

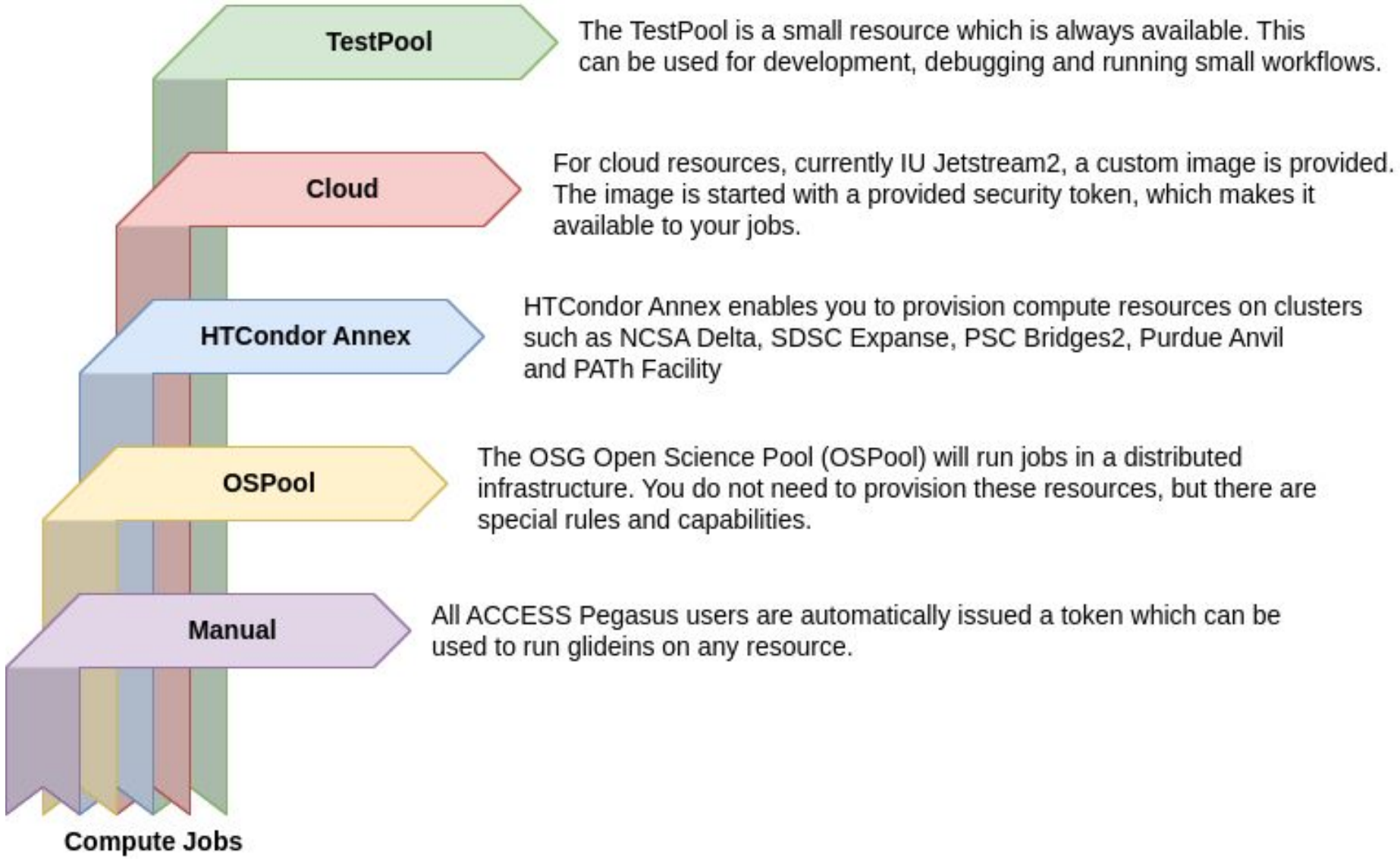
Late Binding: Matches jobs to the best resources after provisioning.

Central Management: Unified view of all jobs from a single scheduler.

Efficient Setup: Reuses pilots to run multiple jobs, reducing overhead.


jupyter notebook
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Markdown
We will then ask out main question, but not using RAG. The quetion in this example is about the IU Jetstream2 resource, which is one of the target execution environments for this workflow:
answer = model.invoke(f"{instructions_regular} Please provide a summary of the GPU capabilities of the IU Jetstream2 system. Include the instance flavors, and any details about the GPUs.").content
print(wrap(answer))
The script will then load the data for RAG, consisting of the full IU Jetstream2 documentation in PDF format, and then ask the same question again. At the end, we should be able to compare the quality of the answer without and with RAG, hopefully see a big improvement in the latter answer.
Workflow
The Pegasus workflow is simple in this case. There is only one job, which takes the llm-rag.py script and js2-documentation.pdf as inputs, and produces answers.txt and ollama.log as outputs.
In [1]:
#!/usr/bin/env python3
# --- Import Pegasus API ---
from Pegasus.api import *
logging.basicConfig(level=logging.DEBUG)
# --- Main workflow class ---
class LLMRAGWorkflow():
wf = None
sc = None
tc = None
rc = None




TestPool
The TestPool is a small resource which is always available. This can be used for development, debugging and running small workflows.
Cloud
For cloud resources, currently IU Jetstream2, a custom image is provided. The image is started with a provided security token, which makes it available to your jobs.
HTCondor Annex
HTCondor Annex enables you to provision compute resources on clusters such as NCSA Delta, SDSC Expanse, PSC Bridges2, Purdue Anvil and PATh Facility
OSPool
The OSG Open Science Pool (OSPool) will run jobs in a distributed infrastructure. You do not need to provision these resources, but there are special rules and capabilities.
Manual
All ACCESS Pegasus users are automatically issued a token which can be used to run glideins on any resource.
Compute Jobs

TestPool

Helps users with an ACCESS account but no allocation explore the capability

Small amount of compute resources attached to pegasus.access-ci.org

- CPU: 32 cores, 128 GB RAM, 256 GB disk
- GPU: 32 cores, 2 GPUs, 128 GB RAM, 256 GB disk
- Hosted on IU Jetstream2, provisioned when needed

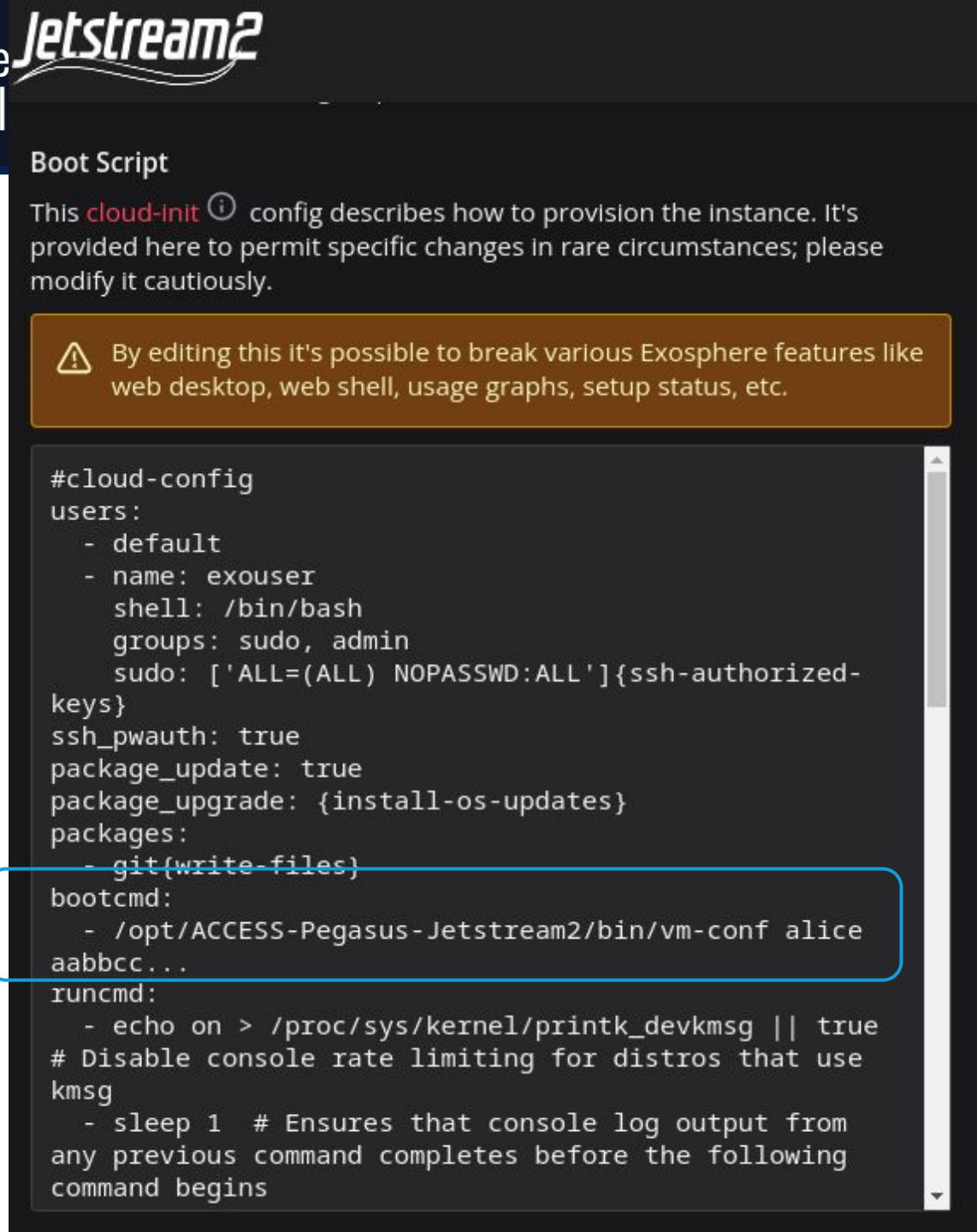
Always available, **no allocation needed**

Can be used for quick turnaround jobs

- workflow development and debugging
- tutorials (not all users might have an allocation at the time of the tutorial)

Cloud

- IU JetStream2
- Provided VM image
- Users have to add pegasus.access-ci.org username and token in the cloud-init yaml
- Instances self-terminates when there are no more jobs



HTCondor Annex

- Bring your own HPC allocation
- Semi-managed, submits glideins via SSH.
  - A glidein can run multiple user jobs - it stays active until no more user jobs are available or until end of life has been reached, whichever comes first.
  - A glidein is partitionable - job slots will dynamically be created based on the resource requirements in the user jobs. This means you can fit multiple user jobs on a compute node at the same time.
  - A glidein will only run jobs for the user who started it.
- Documentation: https://htcondor.org/experimental/ospool/byoc/

```
$ htcondor annex create --nodes 1 --lifetime 7200 \
  --project sta230005p --gpu-type v100-16 $USER GPU@bridges2
```

delta
- cpu
- cpu-interactive
- gpuA100x4
- gpuA100x4-preempt
- gpuA100x8
- gpuA40x4
- gpuA40x4-preempt

stampede2
- normal
- development
- skx-normal

expanse
- compute
- gpu
- shared
- gpu-shared

anvil
- wholenode
- wide
- shared
- gpu
- gpu-debug

bridges2
- RM
- RM-512
- RM-shared
- EM
- GPU
- GPU-shared

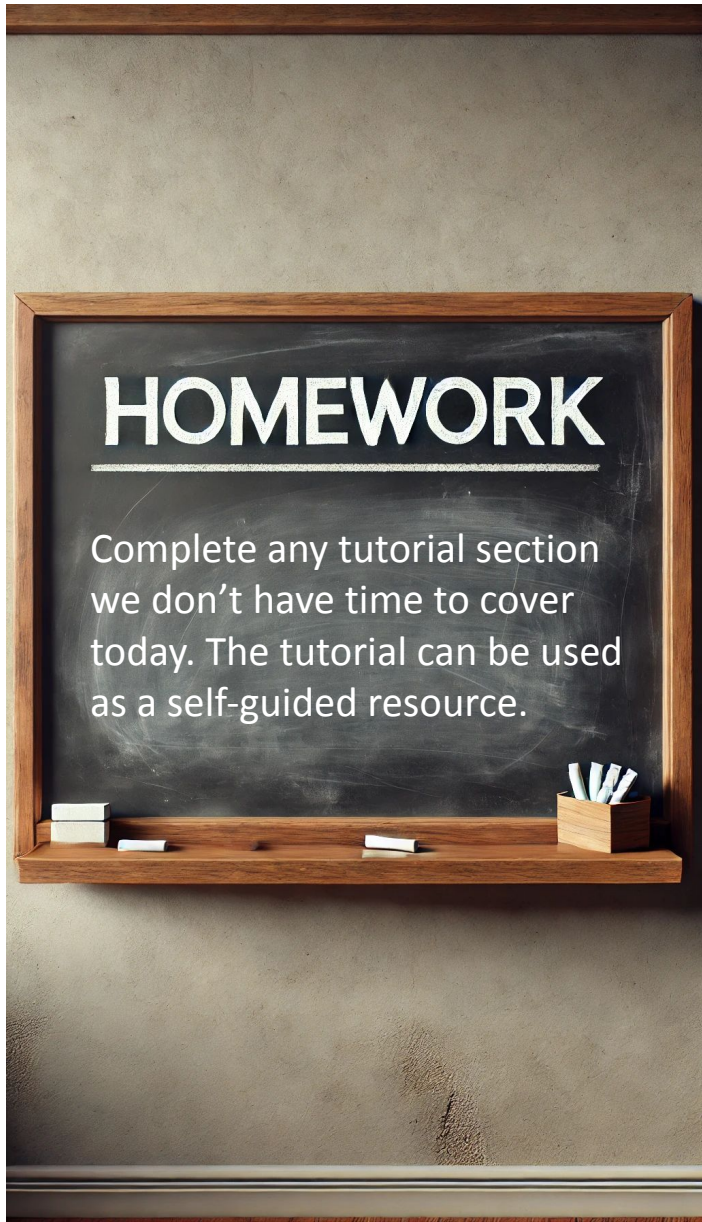
path-facility
- cpu

pegasus-glidein

- Simple glidein which can be run anywhere, as long as you have outbound network connectivity
- Ties in well with ACCESS Pegasus
- https://github.com/pegasus-isi/pegasus-glidein

```
#!/bin/bash
#SBATCH --job-name=glidein
#SBATCH --nodes=10
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=48
#SBATCH --time=24:00:00

curl -o pegasus-glidein https://raw.githubusercontent.com/pegasus-isi/pegasus-glidein/main/pegasus-glidein
chmod a+x pegasus-glidein
srun ./pegasus-glidein -c pegasus.access-ci.org \
                       -t mytoken \
                       -s 'RemoteOwner == "myusername"'
```

1. Use a workflow system for your AI workloads
2. Scheduler overlay to simplify resource management



Thank You!

https://pegasus.isi.edu

Pegasus Users Slack and mailing lists: https://pegasus.isi.edu/contact/

E-mail Support: pegasus-support@isi.edu

Office Hours: https://pegasus.isi.edu/office-hours/