Purdue CMS Analysis Facility

Dmitry Kondratyev, Stefan Piperov, Norbert Neumeister

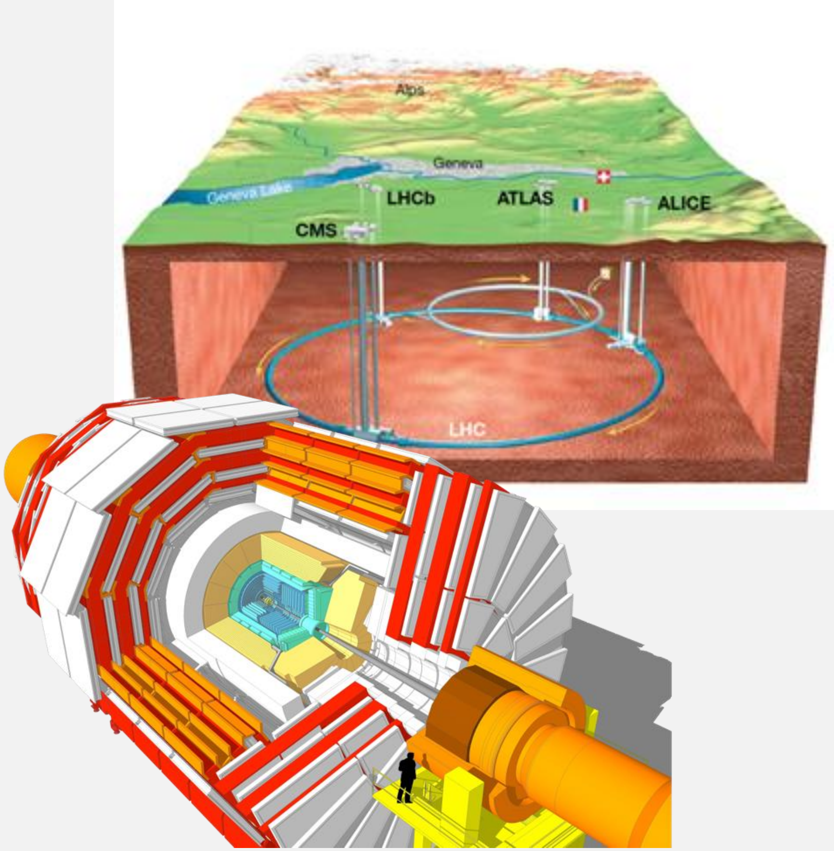
November 7, 2024
HEPiX Fall Workshop
Norman, OK

Outline

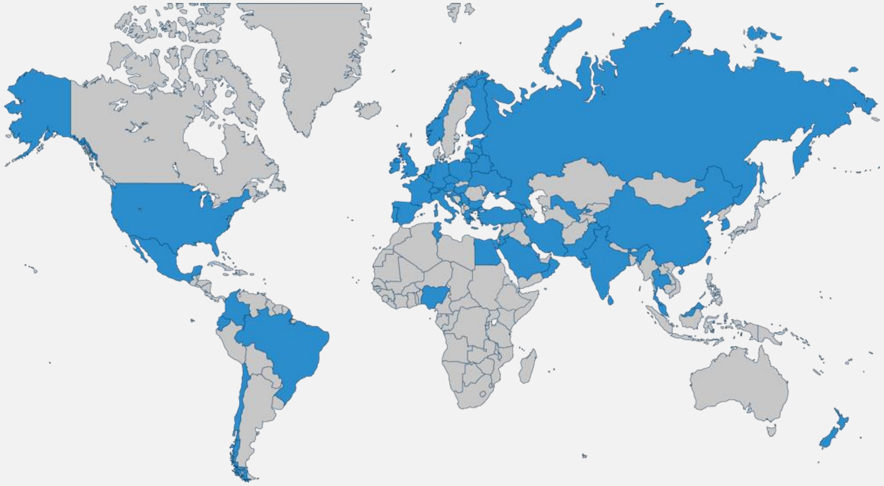
- CMS Experiment and its computing model
- Purdue Analysis Facility:
    - Architecture
    - User interface
    - Storage & data access
    - Scale-out methods
    - Monitoring
- Purdue AF applications

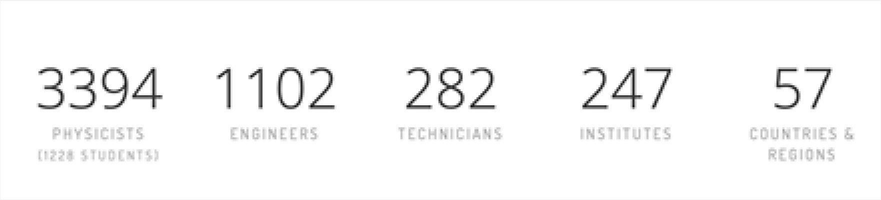
CMS Experiment

CMS detector at CERN LHC



CMS Collaboration



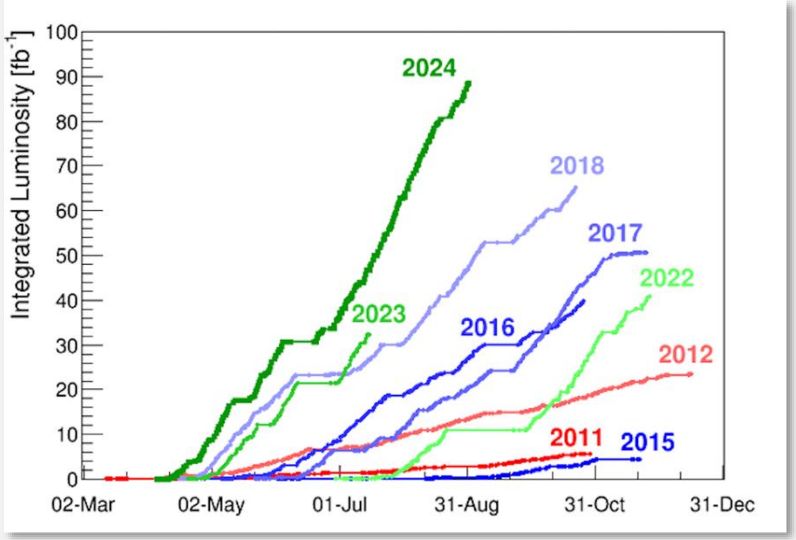


3394	1102	282	247	57
PHYSICISTS (1228 STUDENTS)	ENGINEERS	TECHNICIANS	INSTITUTES	COUNTRIES & REGIONS

CMS Experiment

LHC data collection "runs":

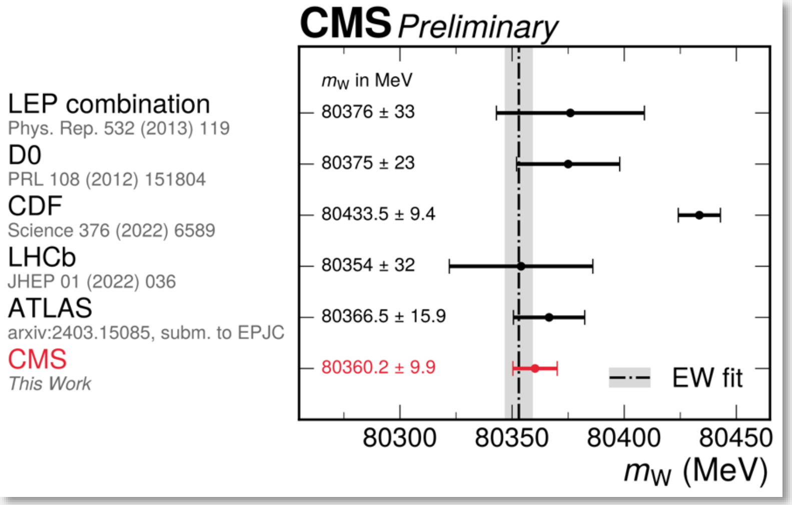
- Run 1: 2010 – 2012 (Higgs boson discovery)
- Run 2: 2015 – 2018
- **Run 3: 2022 – 2026 ← we are here**
- High Luminosity LHC: 2030s



CMS publishes **~100 papers per year**

Most recent important CMS result:

W mass measurement (2024)



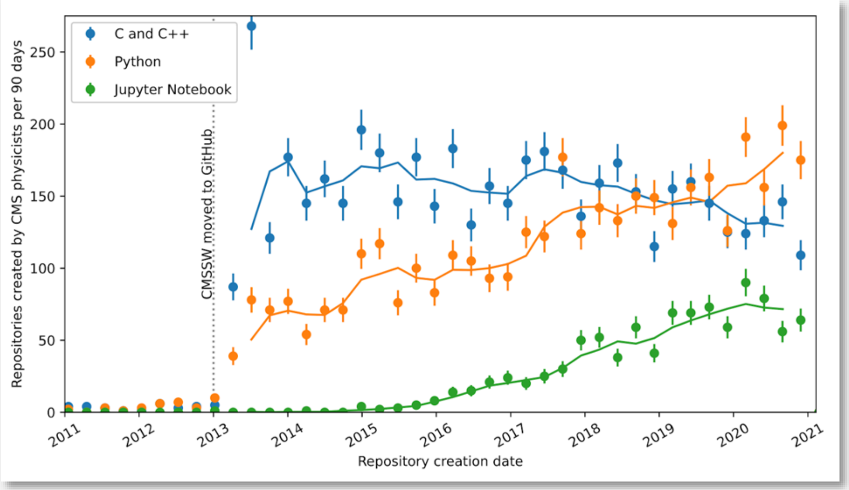
Source: SMP-23-002

Physics Analyses at CMS

- "Traditional" methods (still in use):
  - C++ based frameworks: CMS Offline Software (CMSSW), ROOT
  - Data processing implemented via event loops
  - User interface is just a command line

- Trends emerging in the past decade:
  - Python based frameworks
  - Array programming
  - Interactive interfaces (Jupyter)
  - Adoption of software from other domains (e.g. ML libraries)
  - Advancements in parallel & distributed computing

GitHub repositories of CMS physicists

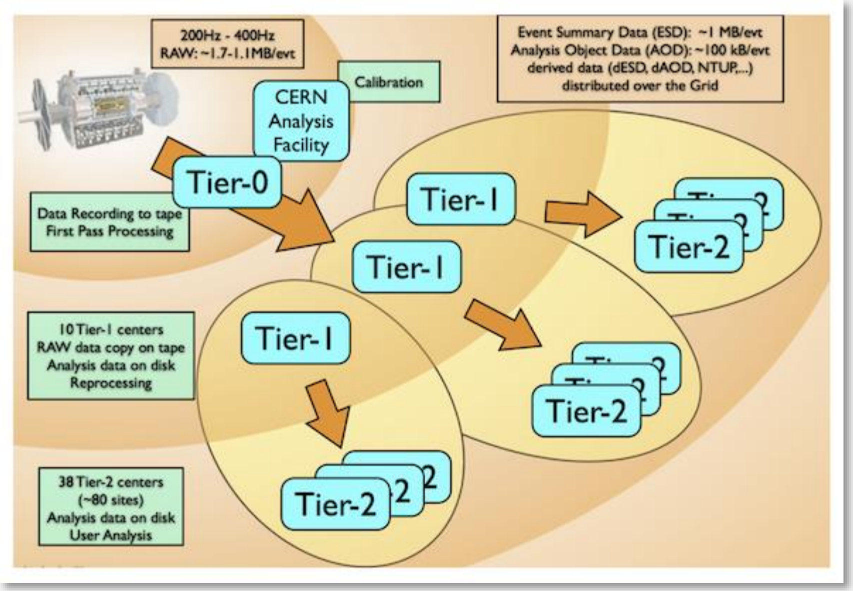


Source: IRIS-HEP

⇒ Motivation to develop Analysis Facilities that natively support these methods

CMS Computing Model

CMS data processing is distributed via **WLCG (Worldwide LHC Computing Grid)**, which has a tiered structure.



Purdue is a **Tier-2 site**, which means that we provide:

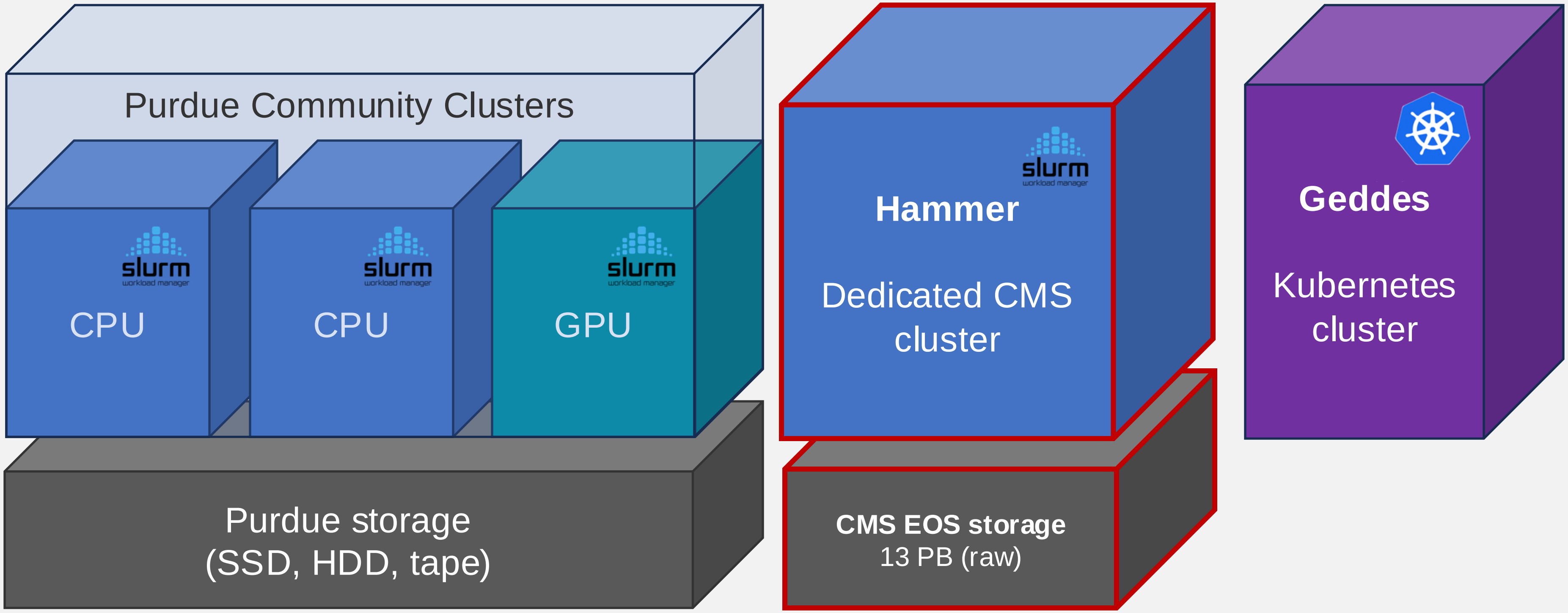
- Computing resources for "central" processing (MC simulations, RECO, etc.)
- Computing resources for **user analyses**.

Purdue Research Computing & CMS Tier-2

Purdue Community Clusters Program
Opportunistic access to job slots via **Slurm**;
A dedicated GPU cluster with various GPU models

Dedicated CMS resources
Cluster with 12k CPU cores;
Large storage element (13PB)

Kubernetes cluster
CMS owns 2k CPU cores
& several A100 GPUs

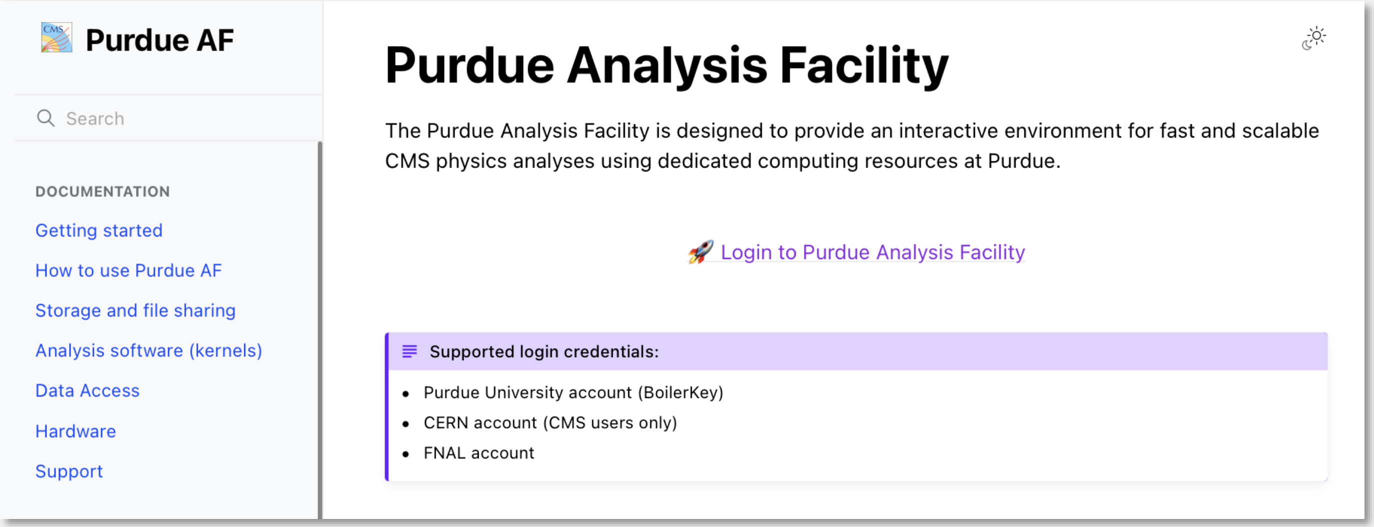


Purdue also hosts “Anvil” supercomputer – a part of NSF ACCESS program.

Purdue Analysis Facility (Purdue AF)

Purdue AF is an interactive environment for end-to-end CMS analyses.

- Primary work environment for majority of CMS researchers at Purdue.
- ~150 registered users, ~30 daily users
- In production since 2023
- Documentation & entry point: https://analysis-facility.physics.purdue.edu/
- Open to all CMS users worldwide
    - Must have CMS affiliation & either **Purdue, CERN, or FNAL account**.



Purdue Analysis Facility

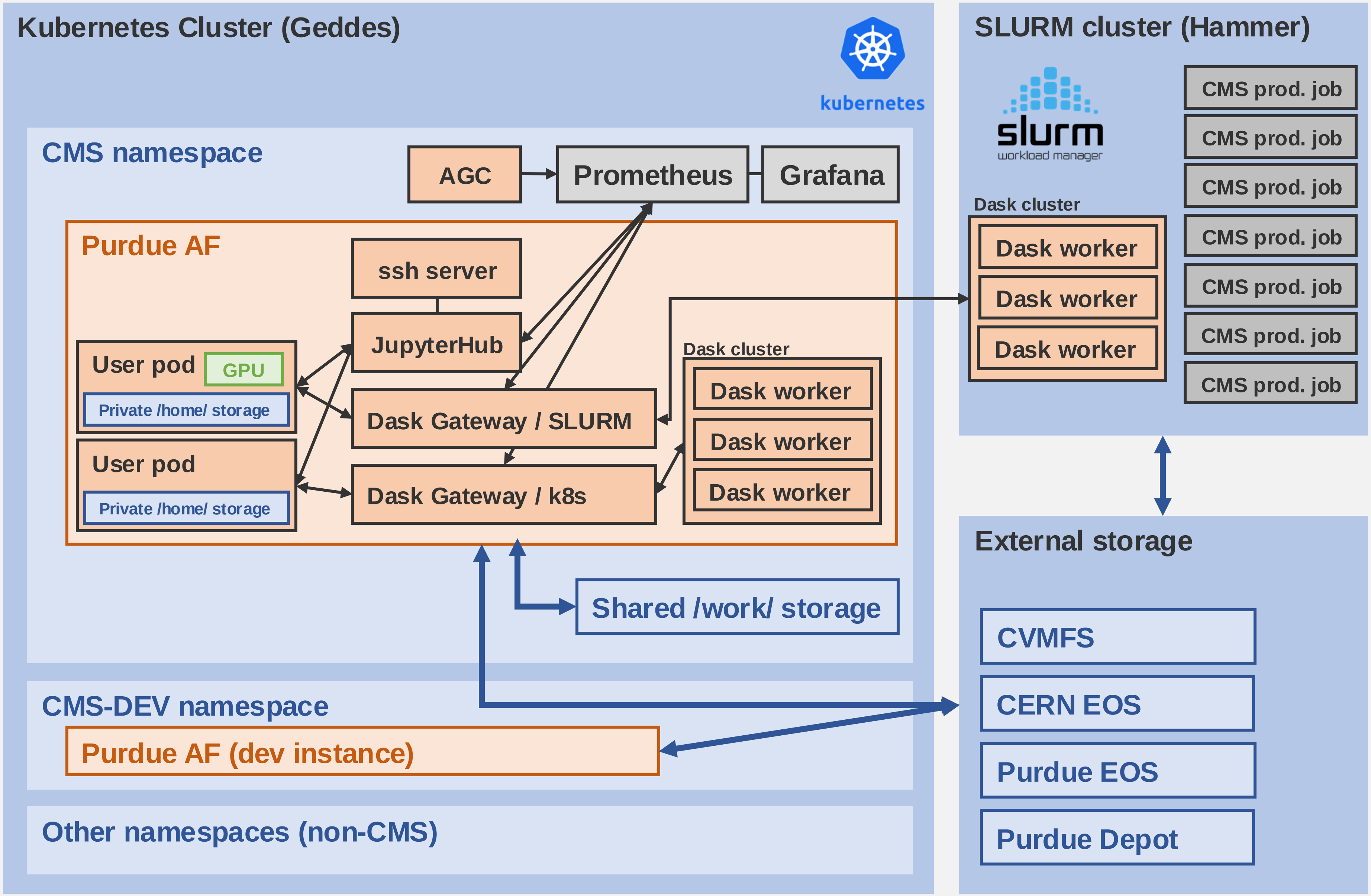
The Purdue Analysis Facility is designed to provide an interactive environment for fast and scalable CMS physics analyses using dedicated computing resources at Purdue.

🚀 Login to Purdue Analysis Facility

Supported login credentials:

- Purdue University account (BoilerKey)
- CERN account (CMS users only)
- FNAL account

Purdue AF: Architecture



Kubernetes Cluster (Geddes)

kubernetes

CMS namespace

AGC

Prometheus

Grafana

Purdue AF

ssh server

JupyterHub

User pod

GPU

Private /home/ storage

User pod

Private /home/ storage

Dask Gateway / SLURM

Dask Gateway / k8s

Dask cluster

Dask worker

Dask worker

Dask worker

Shared /work/ storage

CMS-DEV namespace

Purdue AF (dev instance)

Other namespaces (non-CMS)

SLURM cluster (Hammer)

slurm

workload manager

Dask cluster

Dask worker

Dask worker

Dask worker

CMS prod. job

CMS prod. job

CMS prod. job

CMS prod. job

CMS prod. job

CMS prod. job

CMS prod. job

External storage

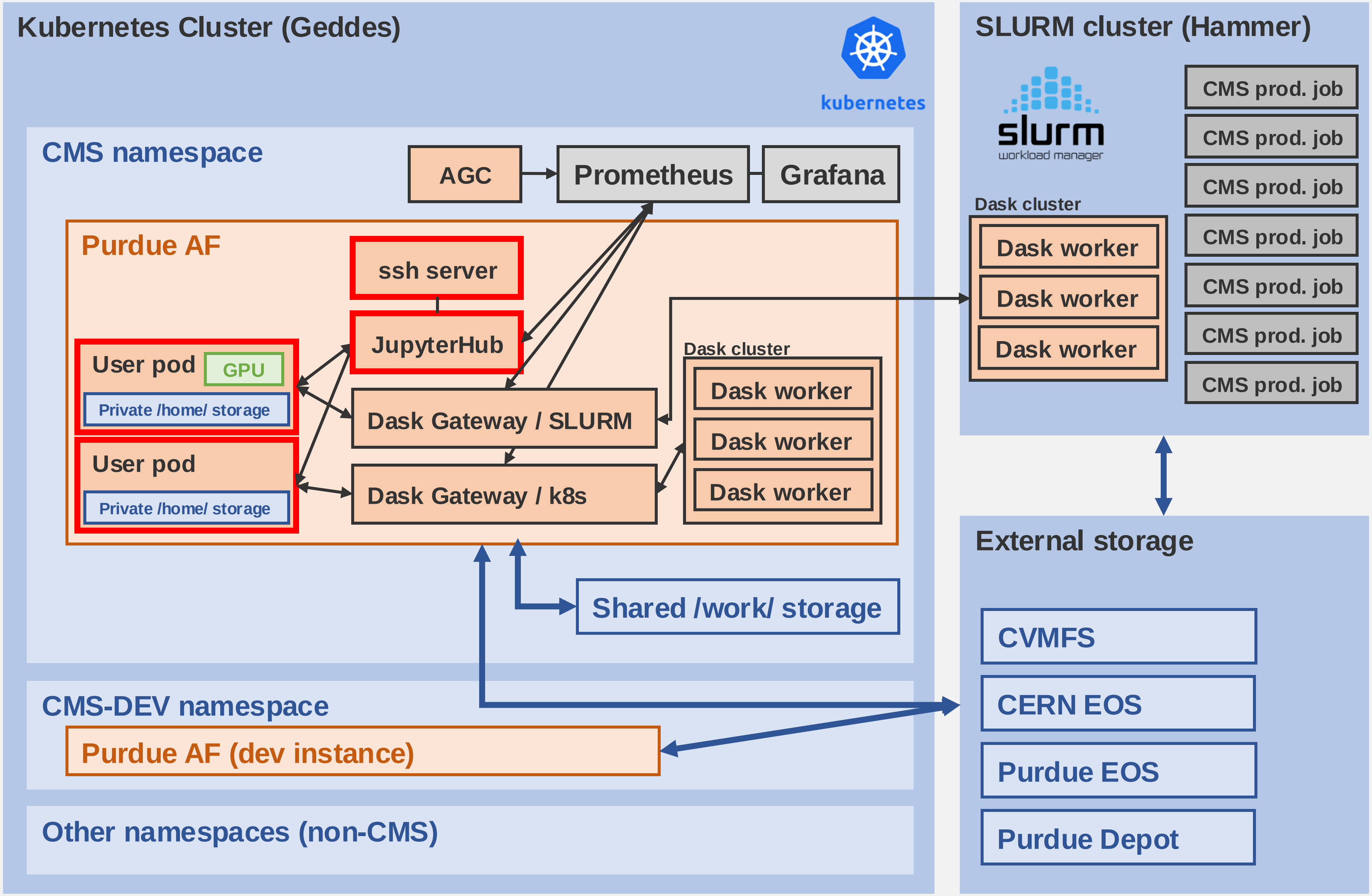
CVMFS

CERN EOS

Purdue EOS

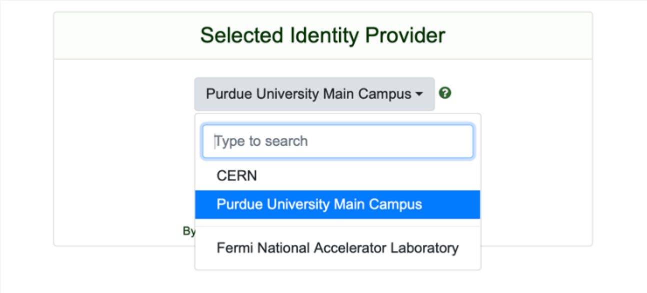
Purdue Depot

Purdue AF: User Interface



User Interface

- Access from any web browser
- At login: choice of account provider and resources (CPU / RAM / GPU)
- **JupyterLab** interface: notebooks, terminals, editors, file browser.
  - Session keeps running for up to 14 days if user closes the browser tab.

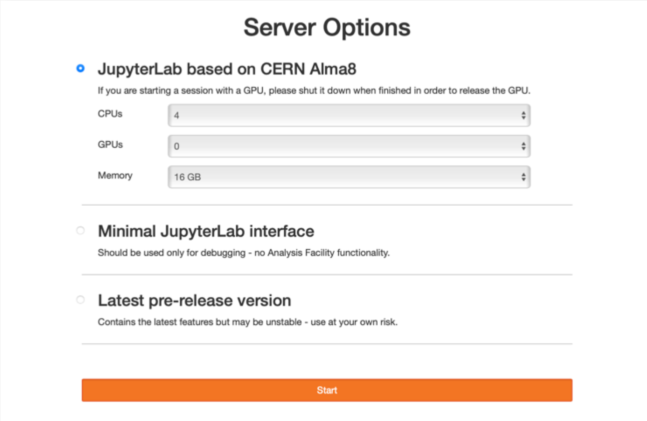


Selected Identity Provider

Purdue University Main Campus ▾

Type to search

- CERN
- Purdue University Main Campus
- Fermi National Accelerator Laboratory



Server Options

● JupyterLab based on CERN Alma8

If you are starting a session with a GPU, please shut it down when finished in order to release the GPU.

CPUs: 4

GPUs: 0

Memory: 16 GB

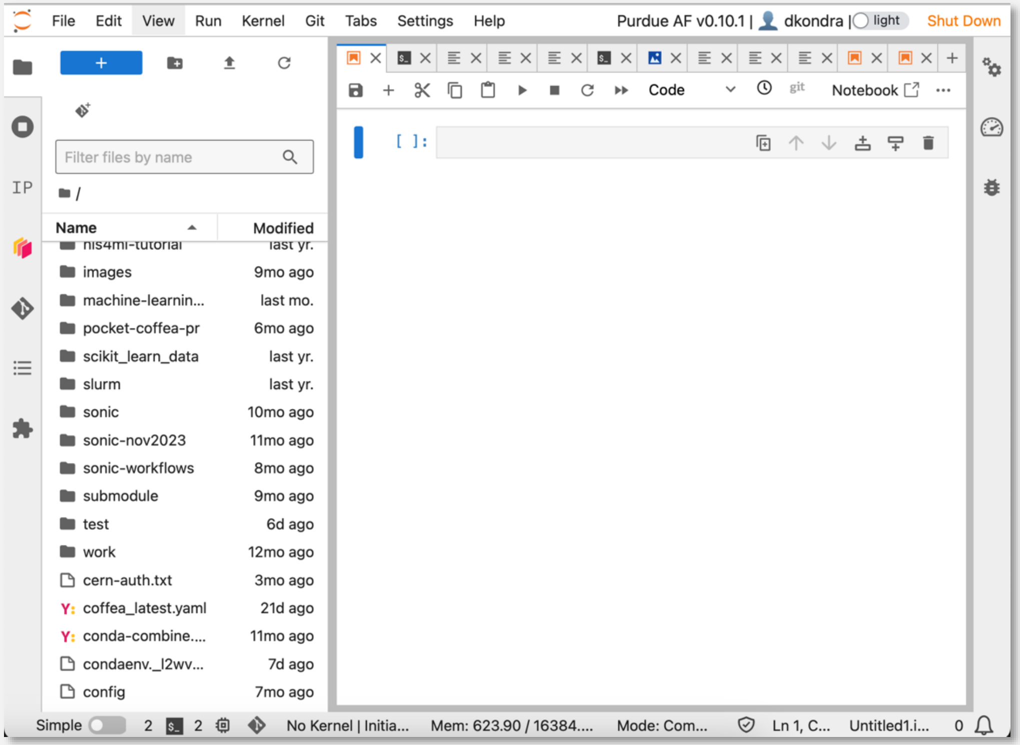
○ Minimal JupyterLab interface

Should be used only for debugging - no Analysis Facility functionality.

○ Latest pre-release version

Contains the latest features but may be unstable - use at your own risk.

Start

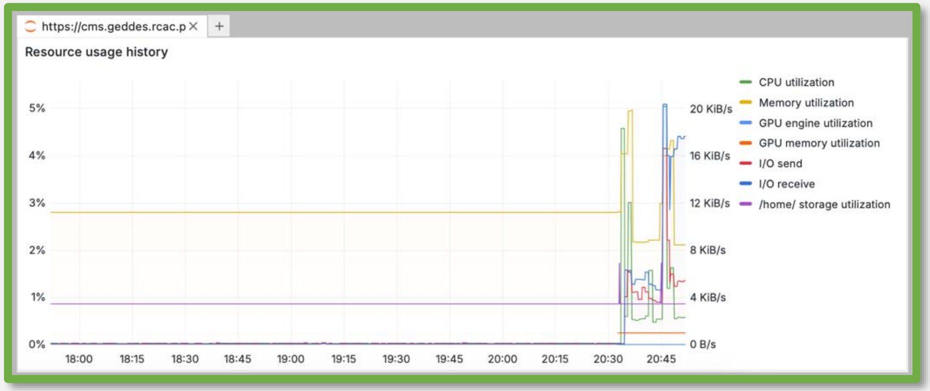


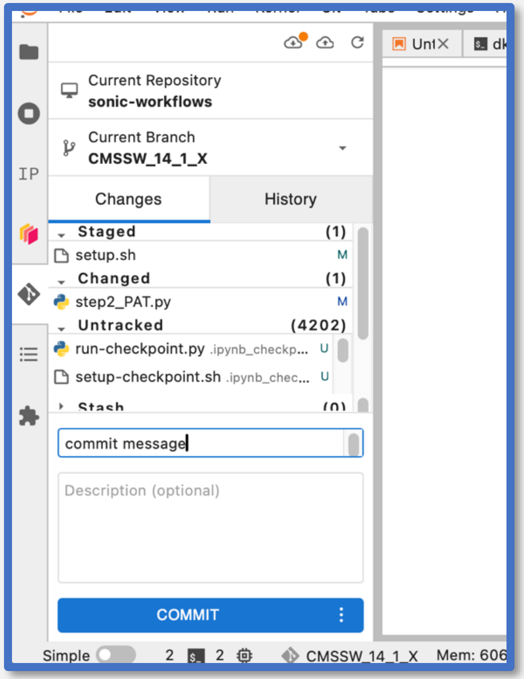
Name	Modified
hls4ml-tutorial	last yr.
images	9mo ago
machine-learnin...	last mo.
pocket-coffea-pr	6mo ago
scikit_learn_data	last yr.
slurm	last yr.
sonic	10mo ago
sonic-nov2023	11mo ago
sonic-workflows	8mo ago
submodule	9mo ago
test	6d ago
work	12mo ago
cern-auth.txt	3mo ago
coffea_latest.yaml	21d ago
conda-combine....	11mo ago
condaenv._l2wv...	7d ago
config	7mo ago

User Interface

Custom JupyterLab extensions:

- **Git**
- **Dask**
- **Resource usage monitoring**





Current Repository
sonic-workflows

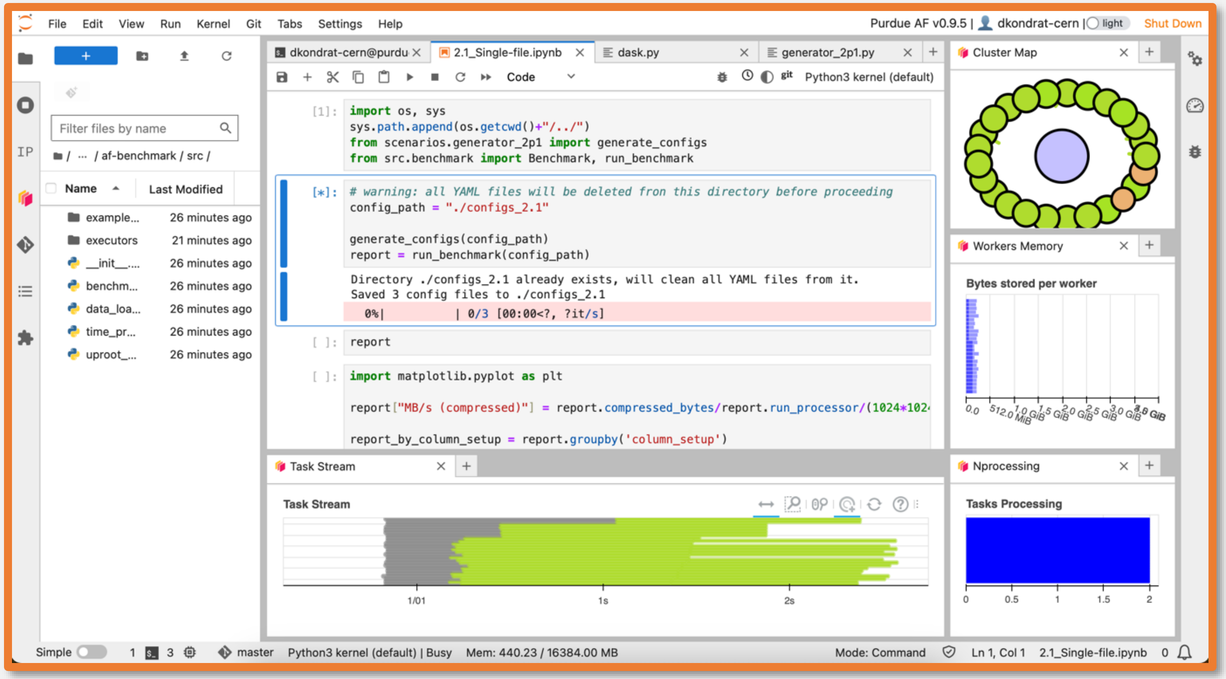
Current Branch
CMSSW_14_1_X

Staged		(1)
setup.sh		M
Changed		(1)
step2_PAT.py		M
Untracked		(4202)
run-checkpoint.py	.ipynb_checkp...	U
setup-checkpoint.sh	.ipynb_chec...	U
Stash		(0)

commit message

Description (optional)

CMSSW_14_1_X

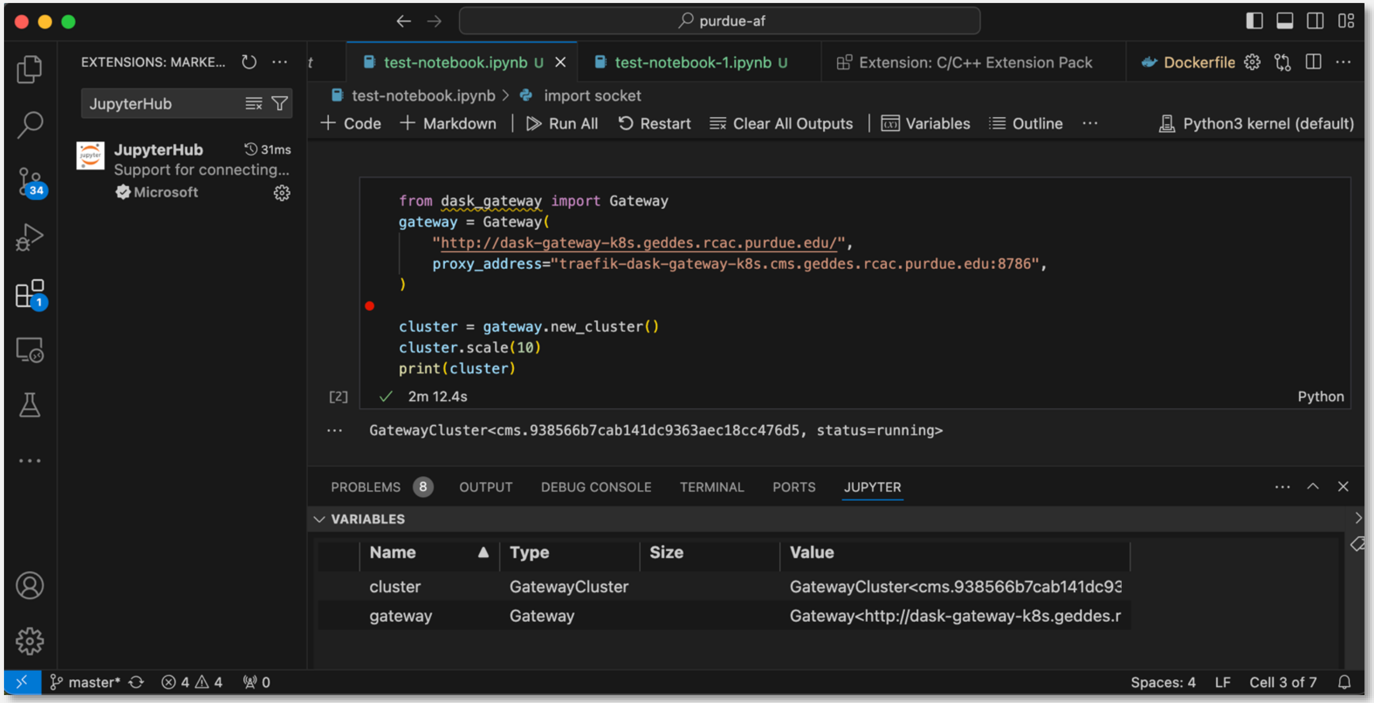


Name	Last Modified
example...	26 minutes ago
executors	21 minutes ago
__init__....	26 minutes ago
benchm...	26 minutes ago
data_loa...	26 minutes ago
time_pr...	26 minutes ago
uproot_...	26 minutes ago

```
[1]: import os, sys
     sys.path.append(os.getcwd()+"/../")
     from scenarios.generator_2p1 import generate_configs
     from src.benchmark import Benchmark, run_benchmark

[*]: # warning: all YAML files will be deleted fron this directory before proceeding
     config_path = "./configs_2.1"

     generate_configs(config_path)
     report = run_benchmark(config_path)

     Directory ./configs_2.1 already exists, will clean all YAML files from it.
     Saved 3 config files to ./configs_2.1
       0%|          | 0/3 [00:00<?, ?it/s]

[ ]: report

[ ]: import matplotlib.pyplot as plt

     report["MB/s (compressed)"] = report.compressed_bytes/report.run_processor/(1024*102

     report_by_column_setup = report.groupby('column_setup')
```

Advanced Access Methods

- Web browser is the default access method, but not the only one.
- With token-based authentication, users can:
  - Connect to a running session via **ssh**;
  - Connect **VSCode** to run local notebooks with remote kernels, or use Purdue AF scale-out resources bypassing the web interface.



```python
from dask_gateway import Gateway
gateway = Gateway(
    "http://dask-gateway-k8s.geddes.rcac.purdue.edu/",
    proxy_address="traefik-dask-gateway-k8s.cms.geddes.rcac.purdue.edu:8786",
)

cluster = gateway.new_cluster()
cluster.scale(10)
print(cluster)
```

```
GatewayCluster<cms.938566b7cab141dc9363aec18cc476d5, status=running>
```

Name	Type	Size	Value
cluster	GatewayCluster		GatewayCluster<cms.938566b7cab141dc93
gateway	Gateway		Gateway<http://dask-gateway-k8s.geddes.r

Software

Software for user analyses is provided in multiple ways:

1. **Curated software stacks** based on needs of Purdue CMS users;
2. Access to **centrally managed software stacks** via CVMFS (CERN’s distributed filesystem);
    - CMSSW releases
    - Singularity images
    - etc.
3. Custom **user-made environments**.

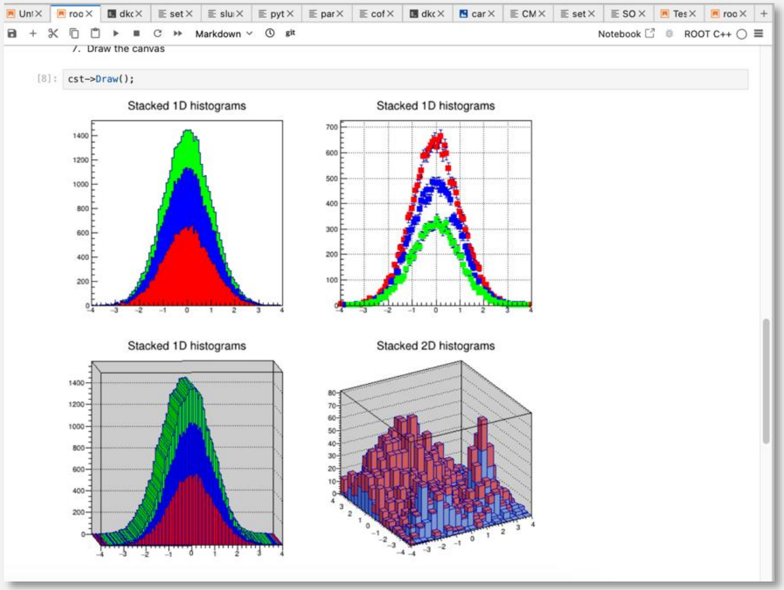
For Python workflows:

- **Conda environments & Jupyter kernels**
- Two curated environments include all popular tools for HEP analysis, such as Uproot, Coffea, Dask, pyROOT.

ROOT at Purdue AF

Multiple ways to use ROOT:

- ROOT console in terminal
- pyROOT in Conda environments / Jupyter kernels
- ROOT C++ kernel: turn notebook into a ROOT Console

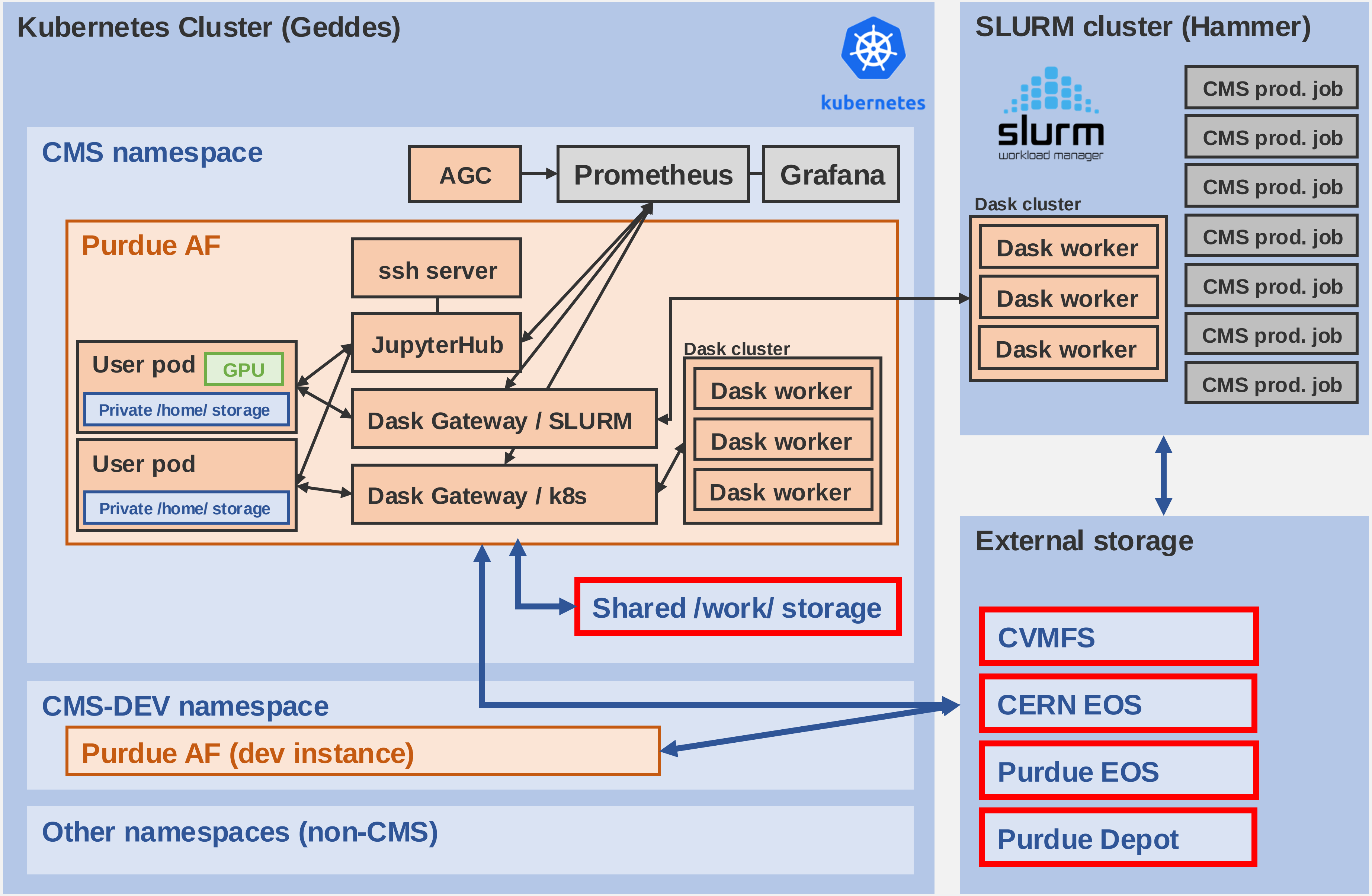


7. Draw the canvas

```
[8]: cst->Draw();
```

We have implemented **GPU acceleration** of ROOT components (RooFit).

Purdue AF: Storage & Data Access



Storage options

Users are directly connected to Purdue Tier-2 storage volumes;
working close to data ⇒ high throughput / low latency

Storage volumes:

- **Purdue EOS storage (13 PB HDD)** – for large datasets
- **Shared project storage (100+ GB per user, SSD)**
  - Two solutions with different permissions for non-Purdue users
- **Private home directories (25 GB)**

Additionally, remote mounts:

- CVMFS to access CERN software
- CERNBox (CERN’s cloud storage) to share work outside of Purdue AF

Data Access

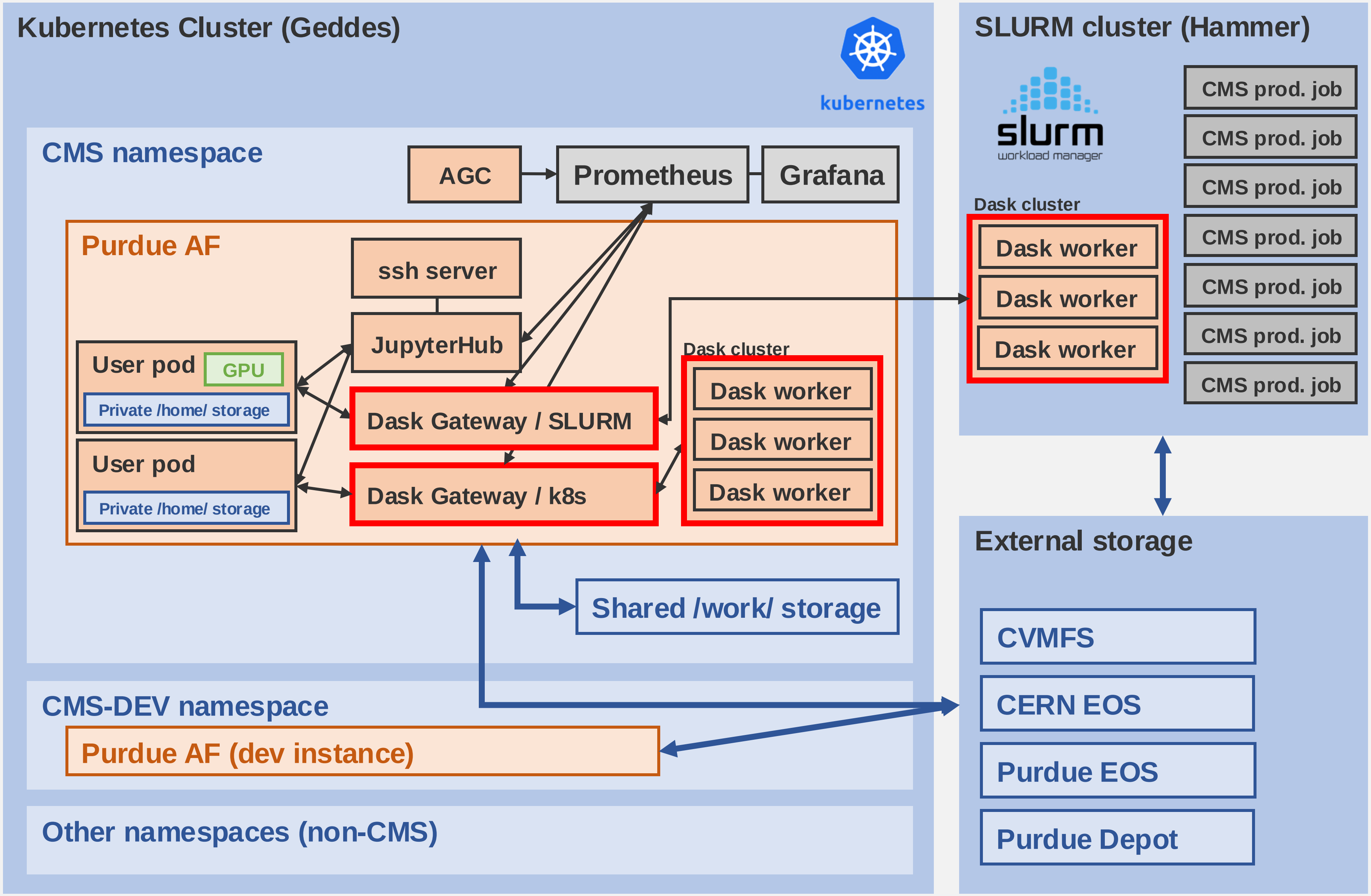
Users are directly connected to Purdue Tier-2 storage volumes;

working close to data ⇒ high throughput / low latency

Data access methods:

- Remote dataset access from anywhere via **XRootD** protocol
- **XCache** – server for local dataset cashing
  - dramatically accelerates data access if a dataset is read repeatedly
- Clone (“subscribe”) datasets to Purdue storage via **Rucio**

Purdue AF: Scale-Out Options



Kubernetes Cluster (Geddes)

kubernetes

CMS namespace

AGC

Prometheus

Grafana

Purdue AF

ssh server

JupyterHub

User pod

GPU

Private /home/ storage

User pod

Private /home/ storage

Dask Gateway / SLURM

Dask Gateway / k8s

Dask cluster

Dask worker

Dask worker

Dask worker

Shared /work/ storage

CMS-DEV namespace

Purdue AF (dev instance)

Other namespaces (non-CMS)

SLURM cluster (Hammer)

slurm workload manager

Dask cluster

Dask worker

Dask worker

Dask worker

CMS prod. job

CMS prod. job

CMS prod. job

CMS prod. job

CMS prod. job

CMS prod. job

CMS prod. job

External storage

CVMFS

CERN EOS

Purdue EOS

Purdue Depot

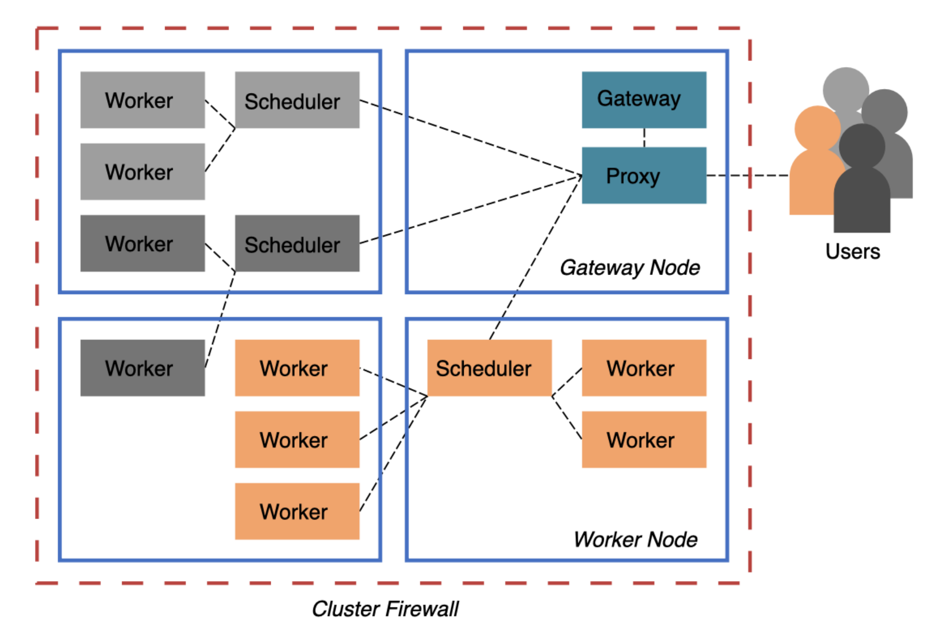
Scaling Out

Available scale-out resources:

- **Slurm** at Purdue clusters
  - 10k-40k cores (dedicated + opportunistic);
  - Users compete with CMS production jobs → slow scheduling.
- **Kubernetes** cluster
  - ~1k cores immediately available;
  - No scheduling mechanism → can't queue jobs.
- **CRAB**
  - Distribute CMSSW jobs to the Grid (1.4M cores);
  - Works for large workloads, but only specific types.
- **CMS-Connect**
  - Submit HTCondor jobs to US CMS Global Pool.

Dask Gateway

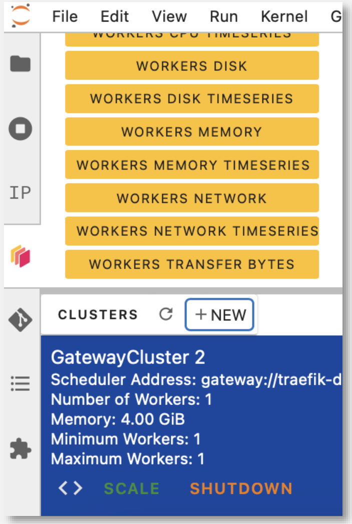
- **Dask** is a flexible Python library for parallel and distributed computing.
- **Dask Gateway** is a multi-tenant service for managing Dask clusters.
  - Clusters are managed via Gateway server(s), which run outside of user sessions.
  - A choice of backends (Local, Kubernetes, Slurm, PBS, Hadoop).
  - User interface is very similar to other Dask implementations.
  - Gateway server keeps track of user’s clusters ⇒ automatic cluster discovery



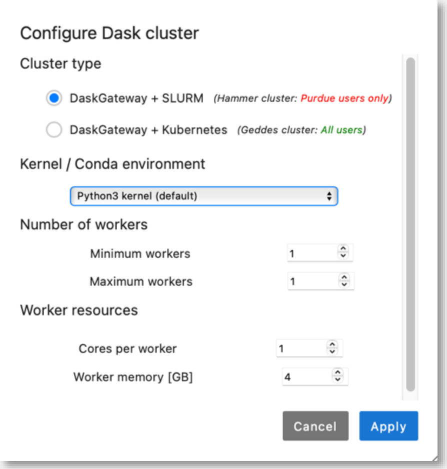
Dask Labextension

- Dask Labextension is an interactive GUI for managing Dask clusters and displaying monitoring dashboards.

- We have prepared our own version of the extension with more flexibility:
  - Choice of Dask Gateway backend (**Slurm** or **Kubernetes**)
  - Specify worker resources interactively



GatewayCluster 2
Scheduler Address: gateway://traefik-d
Number of Workers: 1
Memory: 4.00 GiB
Minimum Workers: 1
Maximum Workers: 1


Configure Dask cluster

Cluster type

● DaskGateway + SLURM *(Hammer cluster: Purdue users only)*

○ DaskGateway + Kubernetes *(Geddes cluster: All users)*

Kernel / Conda environment

Python3 kernel (default)

Number of workers

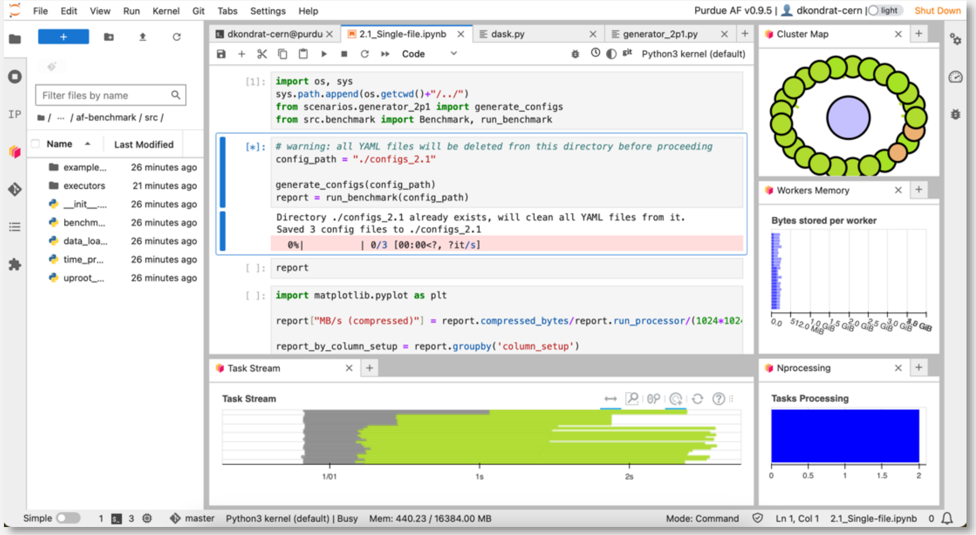
Minimum workers 1

Maximum workers 1

Worker resources

Cores per worker 1

Worker memory [GB] 4



Name	Last Modified
example...	26 minutes ago
executors	21 minutes ago
__init__....	26 minutes ago
benchm...	26 minutes ago
data_loa...	26 minutes ago
time_pr...	26 minutes ago
uproot_...	26 minutes ago

```
[1]: import os, sys
     sys.path.append(os.getcwd()+"/../")
     from scenarios.generator_2p1 import generate_configs
     from src.benchmark import Benchmark, run_benchmark

[*]: # warning: all YAML files will be deleted fron this directory before proceeding
     config_path = "./configs_2.1"

     generate_configs(config_path)
     report = run_benchmark(config_path)

     Directory ./configs_2.1 already exists, will clean all YAML files from it.
     Saved 3 config files to ./configs_2.1
       0%|          | 0/3 [00:00<?, ?it/s]

[ ]: report

[ ]: import matplotlib.pyplot as plt

     report["MB/s (compressed)"] = report.compressed_bytes/report.run_processor/(1024*102

     report_by_column_setup = report.groupby('column_setup')
```

Simple 1 3 master Python3 kernel (default) | Busy Mem: 440.23 / 16384.00 MB Mode: Command Ln 1, Col 1 2.1_Single-file.ipynb 0

Access to GPUs

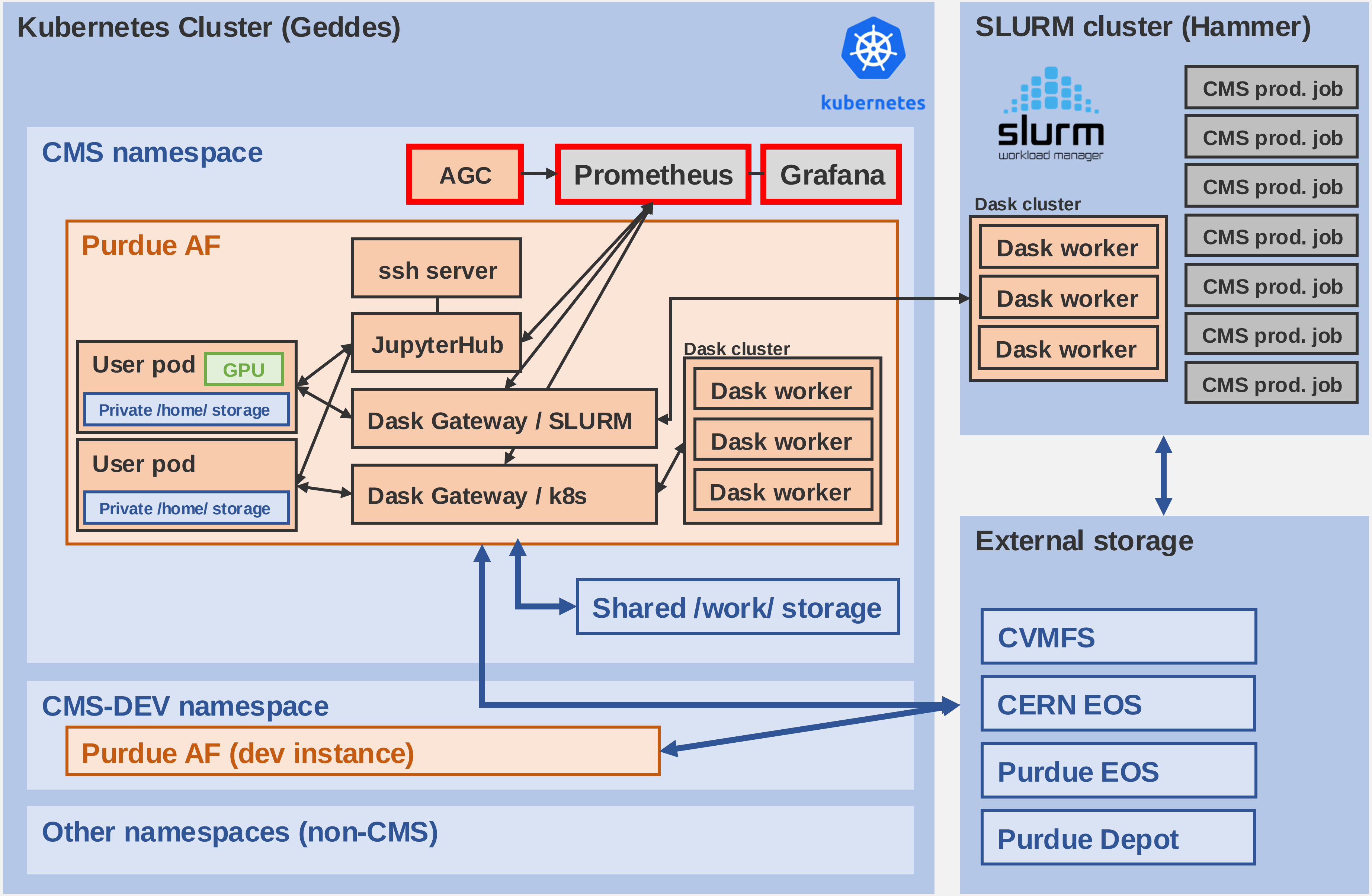
Main use cases for GPUs at Purdue AF:

- ML training
- ML inference
- Accelerating non-ML frameworks such as RooFit

GPU access modes:

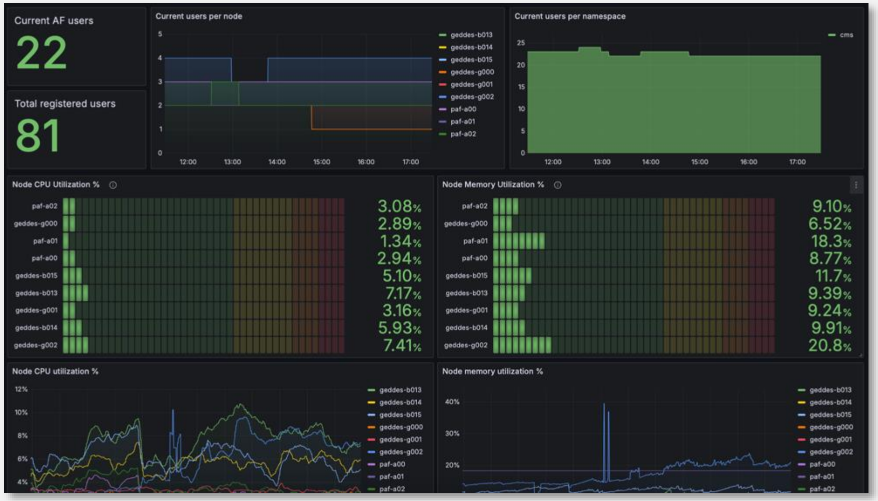
- **Direct access**:
  - 6xA100 40GB GPUs; 4 of them MiG-partitioned into 5GB “slices”
- GPU access via **Slurm jobs**:
  - Wide selection of NVIDIA GPUs Purdue Community Clusters
- SONIC – inference-as-a-Service implementation
  - WIP for CMS production, can also be used for user analyses in the future.

Purdue AF: Monitoring



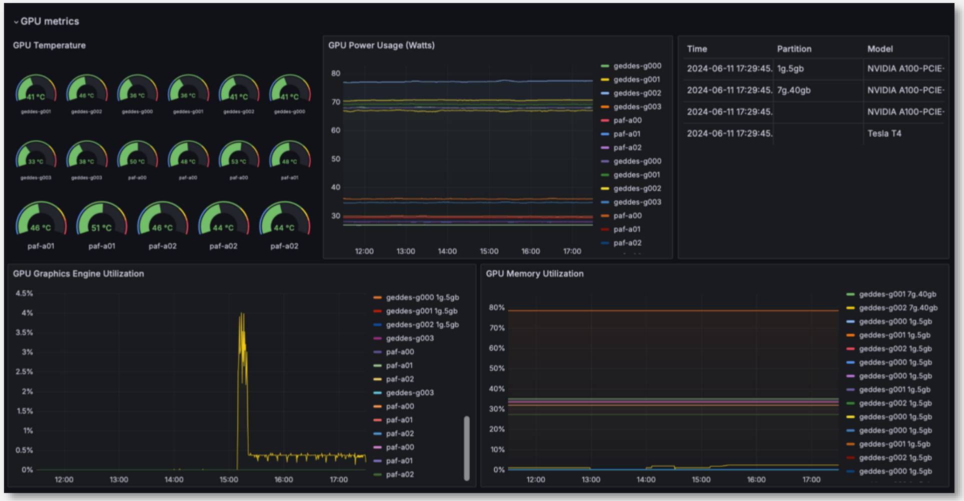
Monitoring

- Monitoring is implemented via **Prometheus** and **Grafana** services
  - User activity and resource utilization
  - Health of nodes and storage mounts
  - GPU utilization
- We also run **Analysis Grand Challenge** benchmark every 3 hours as a standalone CronJob, and monitor its performance at Purdue AF.



Current AF users 22

Total registered users 81



Time	Partition	Model
2024-06-11 17:29:45.	1g.5gb	NVIDIA A100-PCIE-
2024-06-11 17:29:45.	7g.40gb	NVIDIA A100-PCIE-
2024-06-11 17:29:45.		NVIDIA A100-PCIE-
2024-06-11 17:29:45.		Tesla T4

- Panels are configured via code developed using Grafonnet library.

Purdue AF in Action

- Papers for which Purdue AF was used (as of summer 2024):
  - 3 published papers [1, 2, 3]
  - 3 approved analyses awaiting journal submission
  - at least 4 ongoing analyses

- Purdue AF is useful outside of Purdue University:
  - Ongoing projects have external collaborators not affiliated with Purdue
  - This summer hosted six tutorials for CMS users by Fermilab (LPC HATS)

- We host regular tutorials locally at Purdue to help our users learn how to conduct analyses faster & more efficiently.

Summary

- Purdue Analysis Facility aims to provide an interactive environment and a complete toolset for physics analyses at CMS experiment.

- Purdue AF is
  - deployed on a Kubernetes cluster at Purdue
  - connected to Purdue CMS Tier-2 storage and Slurm queues.

- Adopted by almost all Purdue CMS researchers
- Has been instrumental in CMS analyses and tutorials