
A Kubernetes-based Analysis Facility at UChicago

Lincoln Bryant, Rob Gardner, **Fengping Hu**, David Jordan, Judith Stephen, Andrew Taylor, Ilija Vukotic, Farnaz Golnaraghi
University of Chicago

[HSF Analysis Facilities Forum Kickoff Meeting](#)

25 March 2022



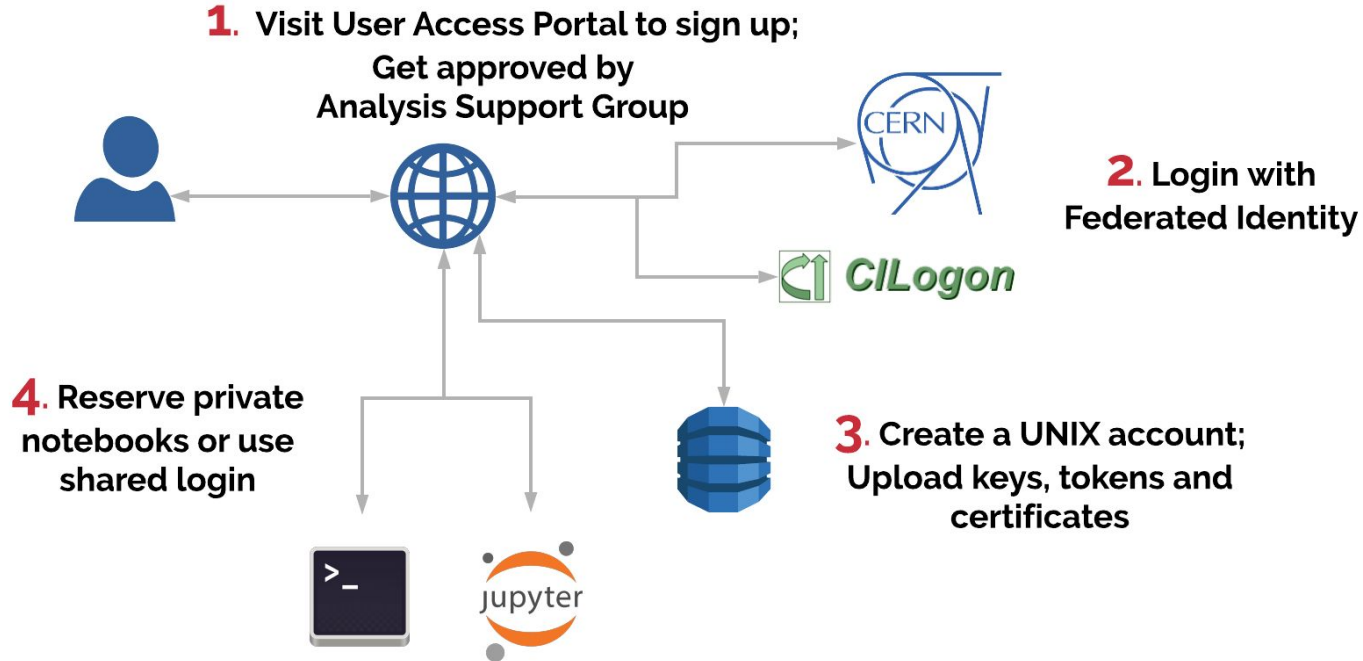
Why Analysis Facilities? Why Kubernetes?

- There is a need to provide infrastructure to support highly interactive, bursty workloads not covered by grid infrastructure (*)
- A number of interesting new analysis frameworks are being developed, many of which have adopted Kubernetes as an enabling technology
- Need flexibility to support both traditional batch and interactive capabilities for LHC Run 3, and forward-looking technologies for HL-LHC R&D

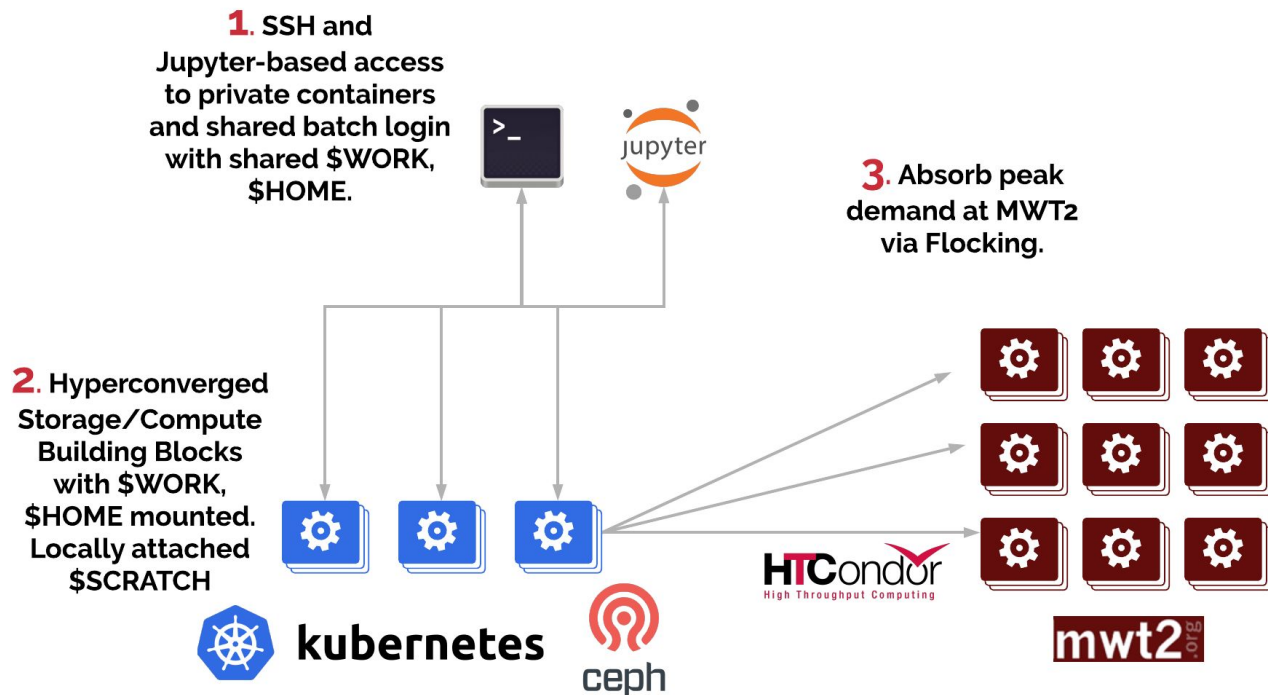
(*) [Future Analysis Systems and Facilities](#), IRIS-HEP blueprint meeting, October 26-27, 2020



Designed for easy onboarding and use



Architecture overview

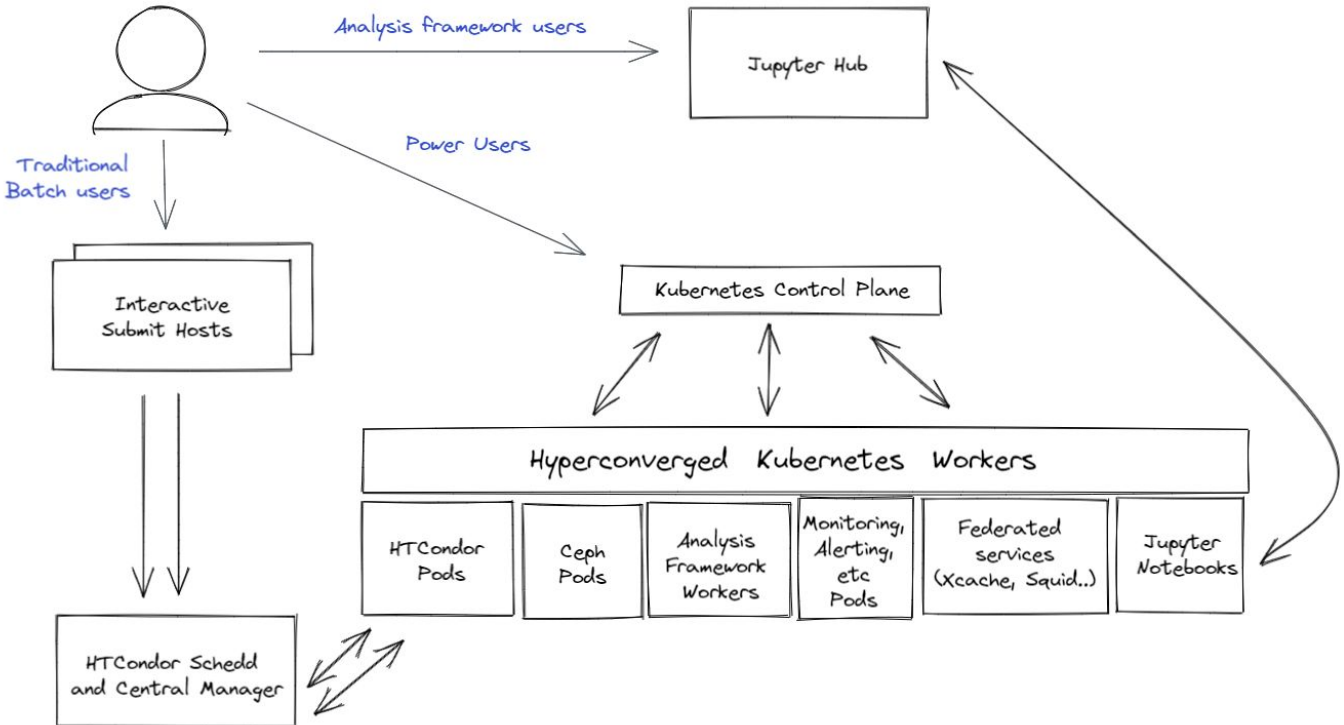


Hardware Description

- Equipment and design chosen to provide interactive logins, 1000 analysis cores, 1 PB of user storage and 25Gbps switching infrastructure within a fixed budget
- 16 "hyperconverged" nodes to provide **both** storage and compute capacity
 - Dual AMD Epyc 7402, 512 GB RAM, 16TB HDDs and 1TB NVMeS for Ceph, 2TB SSDs for dedicated scratch space for batch
- 16 "compute only" nodes – dell R750
 - Dual Intel Xenon Gold 6348, 28C/56T, 384 GB, 10x 3.2 TB NVMe, 25GbE
- 2 GPU nodes – Dell XE8545
 - Dual AMD 7543 2.8GHz (32-core/64-thread) CPU, 512GB RAM , 2x 960GB, SSD storage (~2T /scratch), Dual Port 10/25GbE SFP28 w/ 10Gbps optics
 - 4x A100 GPU cards
- 6 Interactive nodes for traditional batch logins plus Jupyter notebooks.
 - Dual AMD Epyc 7402, 256 GB RAM



Zooming in on the Kubernetes pieces



Kubernetes – basic configurations and extensions

- OIDC authentication
 - cilogon
- Metallb
 - Currently on layer 2 mode
 - BGP mode can be explored too when needed
- Calico
 - Without network overlay/encapsulation – highest performance and simplest network
 - ipv4/ipv6 dual stack configured
- Nginx Ingress controller and certificate cluster issuer with both http01 and dns01 solver



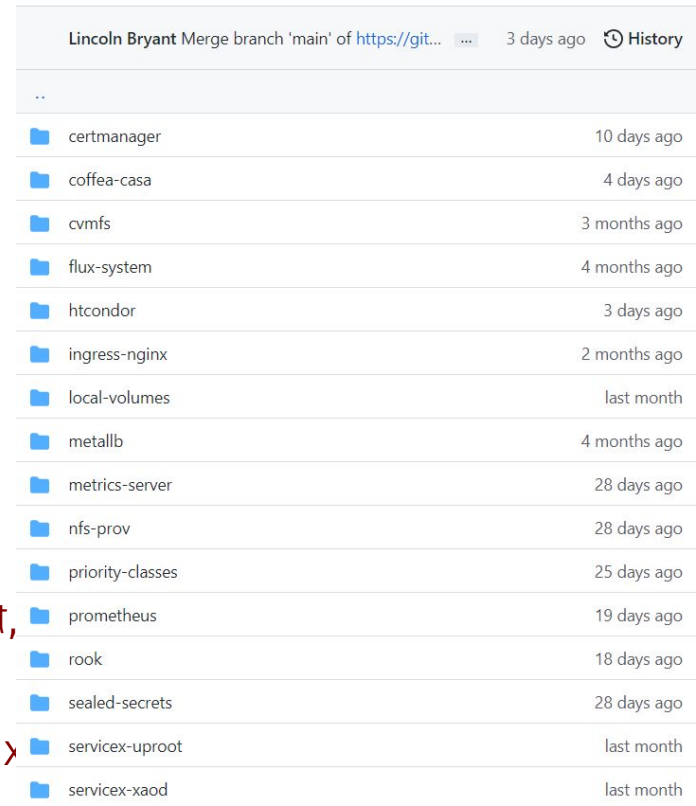
Kubernetes – extensions with operators

- Operators use crds and control loop, aims to capture the key aim of a human operator who is managing a service or set of services– you declare, I reconcile
- Many operators deployed
 - Fluxcd – gitops operator
 - Rook – storage operator
 - Kube–prometheus–stack – monitoring operator
 - Calico operator – network operator



Tools for declarative deployment – FluxCD

- **Flux CD** – “GitOps” style application deployment
 - All configuration lives in GitHub, installation/updates/removal all happen via the Flux operator that uses Git as a single source of truth for the cluster.
 - All of the basic Kubernetes extensions are loaded into the Flux repo (Ingress, Load Balancer, monitoring, certificate management, etc)
 - Ceph, HTCondor, etc are also managed by Flux



The screenshot shows a GitHub repository directory listing for FluxCD. The repository is named 'Lincoln Bryant Merge branch 'main' of https://git...' and was updated 3 days ago. The directory contains the following subdirectories:

Directory Name	Last Updated
certmanager	10 days ago
coffea-casa	4 days ago
cvmfs	3 months ago
flux-system	4 months ago
htcondor	3 days ago
ingress-nginx	2 months ago
local-volumes	last month
metallb	4 months ago
metrics-server	28 days ago
nfs-prov	28 days ago
priority-classes	25 days ago
prometheus	19 days ago
rook	18 days ago
sealed-secrets	28 days ago
servicex-uproot	last month
servicex-xaod	last month

Tools for declarative deployments – SLATE

- Federated service orchestration platform for K8S.
- Provides abstraction for users, groups using federated ID (CI Logon / Globus) across many K8S clusters
- Uses Helm to abstract away complex K8S APIs, simplifying to a set of configuration values for each app.
 - In SLATE, only trusted developers can build applications, while operators can configure them.
- System administrators can register their cluster with SLATE, allow groups and apps to deploy software.
- Operators can store deployments in GitHub and use GitOps

Cluster: uchicago-af

Cluster Information

Administering Group: atlas-af-ops
Contact: lincolnb@uchicago.edu
Organization: University of Chicago
Location: CHICAGO, United States of America
Number of Nodes: 21
Status: Reachable ✓

Groups with access

atlas-xcache
slate-dev
atlas-af-ops

Running Instances on AF

Show entries

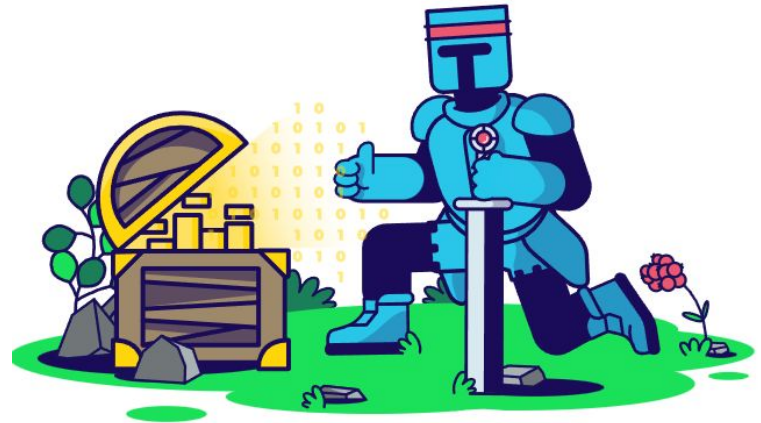
Search:

Name	Applications	Group	Cluster	ID
nginx-default	nginx	slate-dev	uchicago-af	instance_TYMtZWpLFhM
osg-frontier-squid-af-01	osg-frontier-squid	atlas-af-ops	uchicago-af	instance_JuVpDGq3enQ
redis-sysview	redis	atlas-af-ops	uchicago-af	instance_rCNFLUeINwl
xcache-slate01	xcache	atlas-xcache	uchicago-af	instance_HEGGi_f7XTI

A tour of some apps on the Analysis Facility

Rook

- Kubernetes-flavored Ceph, closely tracks upstream releases
- Fully manages Ceph cluster on the AF via Kubernetes Operator
- Very popular storage option in the Kubernetes community
- A "graduated" project in the Cloud Native Computing Foundation ecosystem.
 - i.e., multiple organizations committing code, completed security audit, explicitly defined project governance, etc.

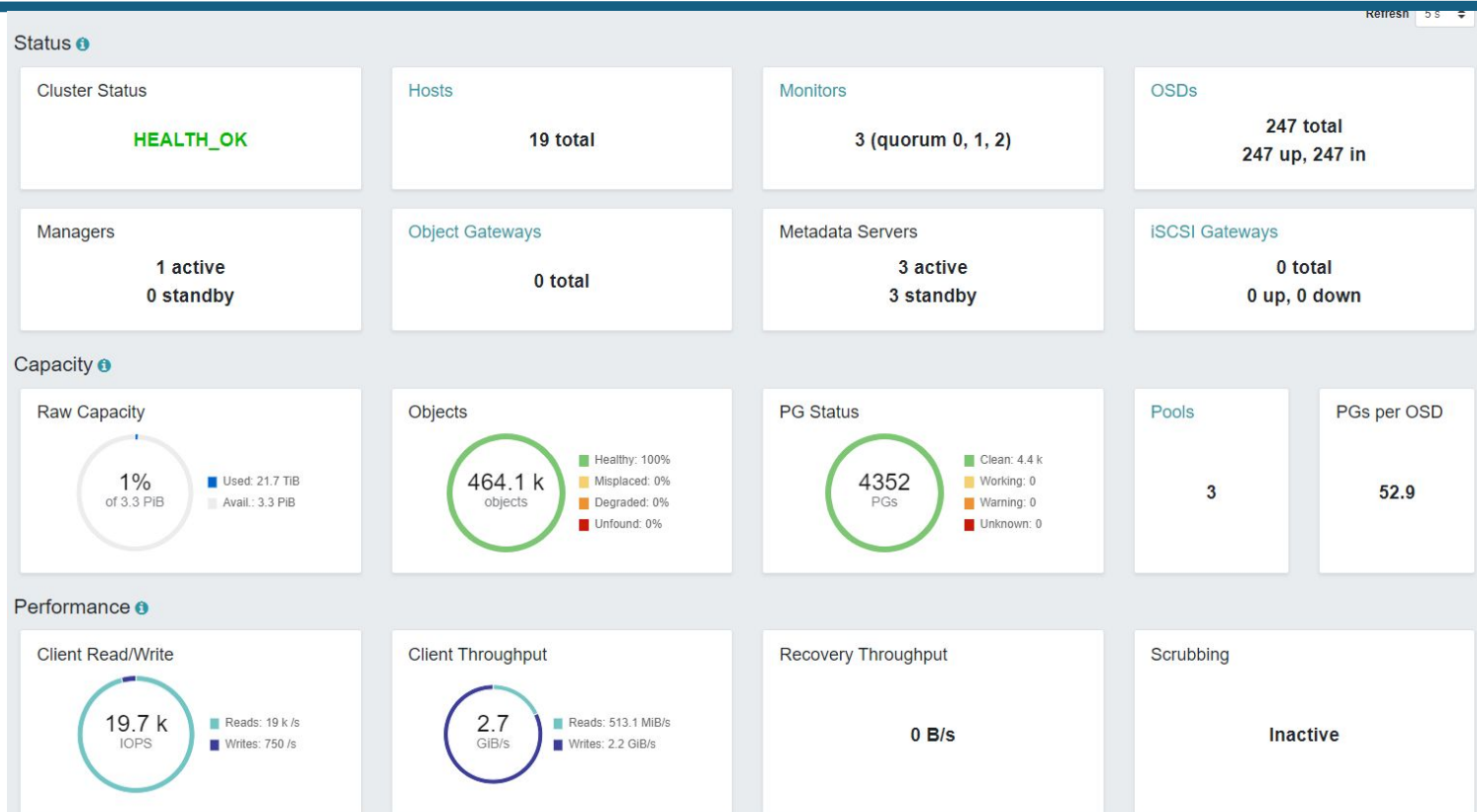


Rook/Ceph configuration on the UC AF

- **Goal: Provide a 1PB shared filesystem (\$DATA) for users of the AF**
- 228x 16TB HDDs configured for 3x replication
 - EC is tantalizing for the capacity gains, but we haven't had a good experience with it elsewhere.
- Each node has a dedicated NVMe for Bluestore database (Metadata).
- Each node has a second dedicated NVMe for CephFS metadata
- CephFS configured with 6 MDSes
 - 3 active, 3 standby – floating on the Hyperconverged nodes.
- Filesystem mountable within Kubernetes and outside.
- Currently we are not using RADOSGW or RBD.
 - **Tight focus on performant cluster filesystem for user data.**



Ceph Dashboard



HTCondor Setup

- Single, unified queue presented to users
 - Any login node, any notebook sees the same queue
- Fully tokenized authentication
 - Each user has a `$HOME/.condor` directory that holds a token allowing job submission to the remote schedd on a shared filesystem
- All execute nodes live in Kubernetes
 - Piecemeal approach to moving daemons into K8S



HTCondor – Submit

- **Goal: Provide O(1000) cores for batch analysis**
- HTCondor 9.x with fully tokenized configuration
- Single schedd on dedicated "head" node, users see a single queue.
 - In the future, we will run multiple schedd queues and hash users across queues, if needed.
- Currently two of the "interactive" nodes are used as traditional HTCondor submit points
- Every user has a submit token in `~/ .condor / tokens .d` to facilitate 'remote' submission
- Shared filesystem between submit points and schedd, so users don't need to transfer files to the schedd.

HTCondor – Execute

- Completely managed by Kubernetes with a lot of nice features
- 80 logical cores per Worker, partitionable slots
- HTCondor pods are dynamically configured based on values from the Kubernetes downward API, e.g.

```
resources:
  limits:
    cpu: "84"
    memory: "400G"
    ephemeral-storage: "10G"
  requests:
    cpu: "80"
    memory: "384G"
    ephemeral-storage: "10G"
  - name: _CONDOR_MEMORY
    valueFrom:
      resourceFieldRef:
        containerName: execute
        resource: requests.memory
        divisor: 1Mi
```

→

```
$ condor_status slot1@c001 -af Memory
366211
```

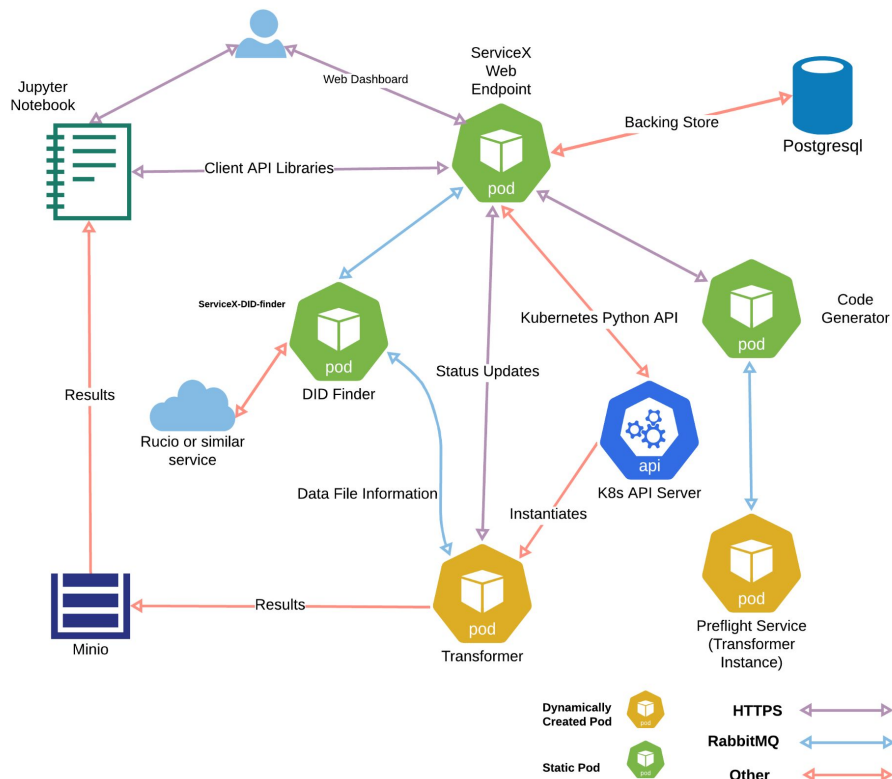
- CPU is a slightly trickier expression because Kubernetes can schedule at a sub-core level while HTCondor uses whole cores

HTCondor – Execute cont'd

- HTCondor pods are preemptable by other Kubernetes deployments
 - This is potentially a pain point! No tools available to enforce HTCondor-like fair share in K8S.
 - Global default priority class is non preempting, so pods by default will not preempt other running pods
- CVMFS is available via hostpath mount
- User identity (UNIX name/uid/gid) and shared filesystems (\$DATA, \$HOME) available in the containers

ServiceX

- A service that quickly **filters** and **delivers** data.
- **Filtering** here means skimming, slimming and augmenting input data. Input data can be xAODs or flat ROOT files.
- Resulting data can be **delivered** as PyArrow awkward arrays or flat ROOT files.



ServiceX deployments

- Ideally to be orchestrated to run near datasets to reduce data transfer volume (FABRIC CERN to investigate server side filtering)
- 2 Instances are deployed at AF
 - [ATLAS xAOD](#)
 - [ATLAS uproot](#)
- Endpoints are publicly accessible
- Authentication via Globus Auth. Account approval via a Slack channel `#servicex-signups`

- What is it?
 - A Prototype of an Analysis Facility for Columnar Object Framework For Effective Analysis (Coffea).
 - A bunch of python tools that greatly simplifies columnar data analysis.
 - Specially performant when parallelized using Dask.
 - Build on top of kubernetes and htcondor, users use Jupyter to do analysis.
 - Easily-scalable and user-friendly computational environment that will simplify, facilitate, and accelerate the delivery of HEP results.
- The SSL team works closely with the Coffea Casa team to support Analysis Grand Challenge
 - including direct code contributions, e.g. adding support for deploying HTCondor within Kubernetes
 - Previously required external HTCondor as a dependency
 - This allows easier deployment for sites with no existing HTCondor cluster/or HTCondor expertise.
- Deployment of Coffea Casa instance on SSL and the new ATLAS shared Tier3 / Analysis Facility

Coffea Casa Deployment

- Deployed via Fluxcd
- Docker images served from [OSG Harbor](#)
- User authentication via Indico IAM for Atlas
- Coffea-Casa Analysis Facility @ UChicago can support ATLAS users via CERN IdM service (<https://atlas-auth.web.cern.ch/>)
- Currently run as coffea-atlas user, will work on customizing the jupyter authenticator and notebook images to support running notebooks as provisioned user to give users a more unified experience(\$HOME \$DATA, directory access, fair share etc)



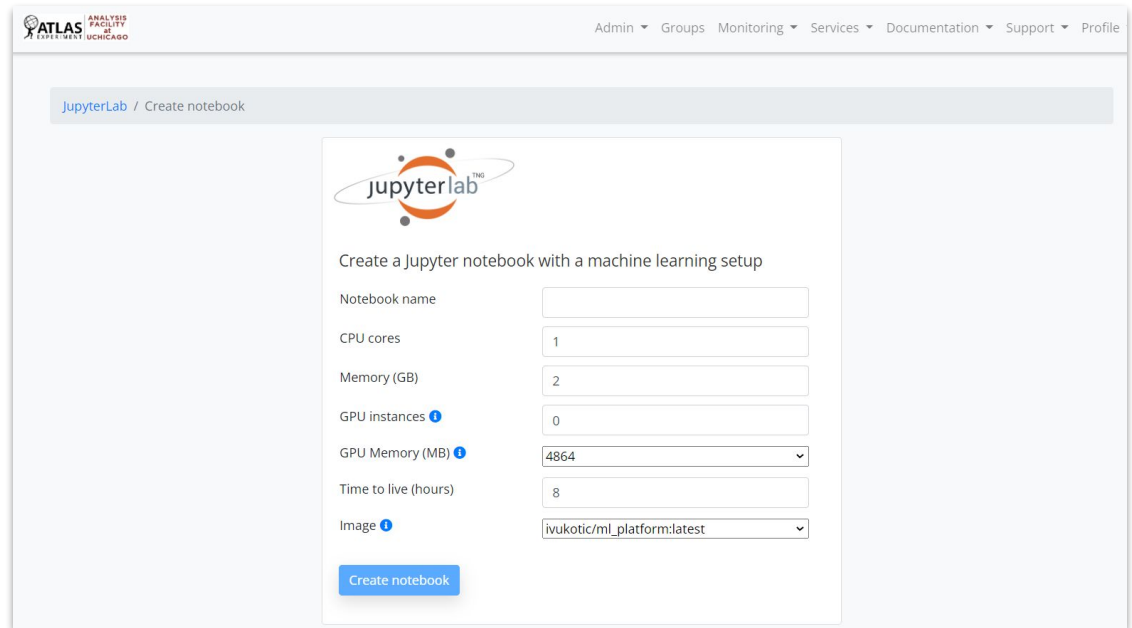
Resource management

- 3 main types of load: htcondor worker, interactive notebooks, servicex(transformers)
- most of the resources statically allocated to compute(condor workers)
- Login nodes are reserved for interactive notebooks
- Servicex transformers are bursty but doesn't have reserved resource – scales with HPA
- Configure priorityclass to allow transformers to explore unused cycles



ML Platform


- Customizable Jupyterlab environment with GPU support
- Node feature discovery to label nodes with their features and expose features to allow users to match gpus
- Currently 2 types of GPUs
 - 4 Full A100
 - 28 MIG1g.5gb(equivalent of 1/7 of full gpu)
- the notebook runs as provisioned user for unified user experience – with \$HOME and \$DATA directory access



ATLAS ANALYSIS FACILITY UNIVERSITY OF CHICAGO

Admin ▾ Groups Monitoring ▾ Services ▾ Documentation ▾ Support ▾ Profile

JupyterLab / Create notebook



Create a Jupyter notebook with a machine learning setup

Notebook name

CPU cores

Memory (GB)

GPU instances

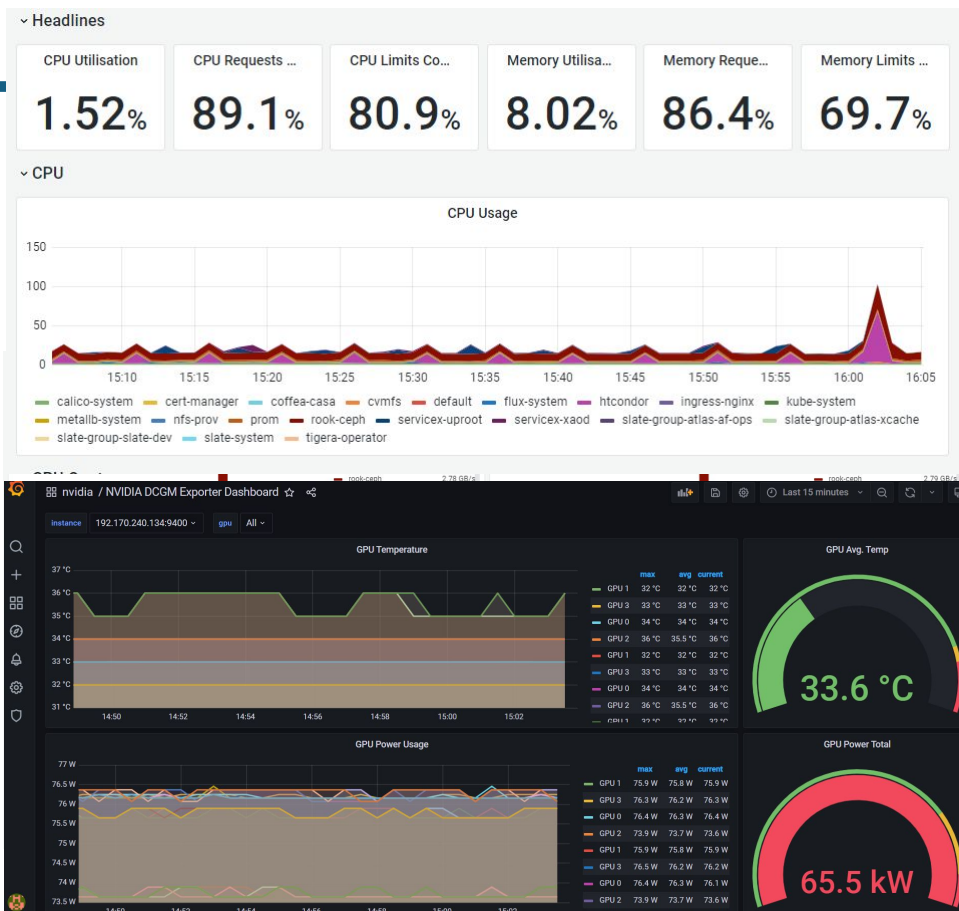
GPU Memory (MB)

Time to live (hours)

Image

Monitoring

- Prometheus operator and Grafana
- Configured for Federated ID login (CI Logon)
- Many dashboards: ceph cluster, coffea-casa users, ml users, kube networking, compute, workload, gpus ...
- Alerts send to slack channel



Closing thoughts

- Kubernetes is a portable, extensible, open-source platform for managing containerized workloads and services, a nice fit for building forward-looking analysis facility
 - A large and rapidly growing ecosystem
 - The operator pattern helps a lot in managing complex application deployments
 - The declarative nature of kubernetes and gitops tooling and practices offers peace of mind in managing the complexity
- There are many interesting new analysis frameworks in this space, and users are excited about notebook-driven analysis.
- Traditional batch is not going away, we need to support both.

Questions?