Kubernetes activities at UChicago

Lincoln Bryant

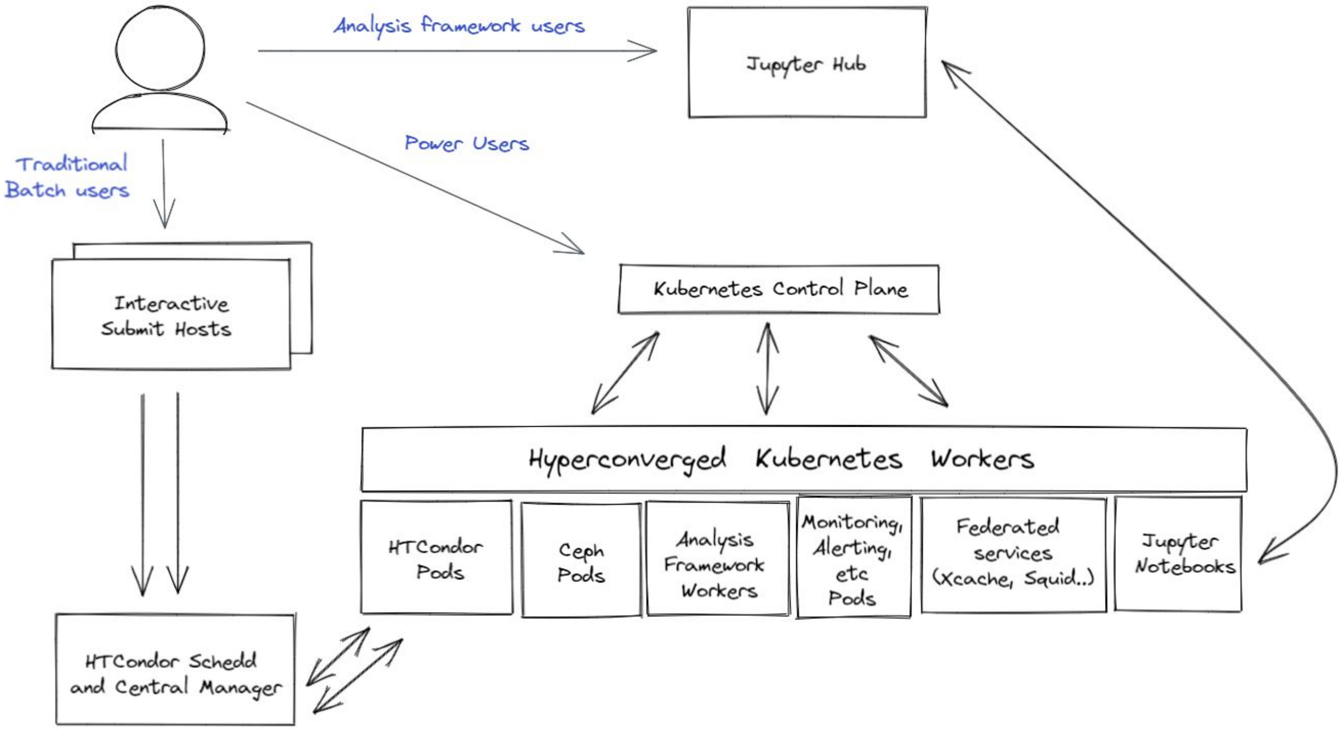
Pre-GDB

07 June 2022

ATLAS Analysis Facility

- We recently opened our doors to users for a new ATLAS Analysis Facility at UChicago
- About 1,000 cores / 1 PB of usable disk
- Traditional batch SSH access, as well as web-based Jupyter notebooks
- HTCondor workers, Ceph storage, other various applications managed via Kubernetes

UChicago AF Architecture



HTCondor – Execute

- Completely managed by Kubernetes
- 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
```

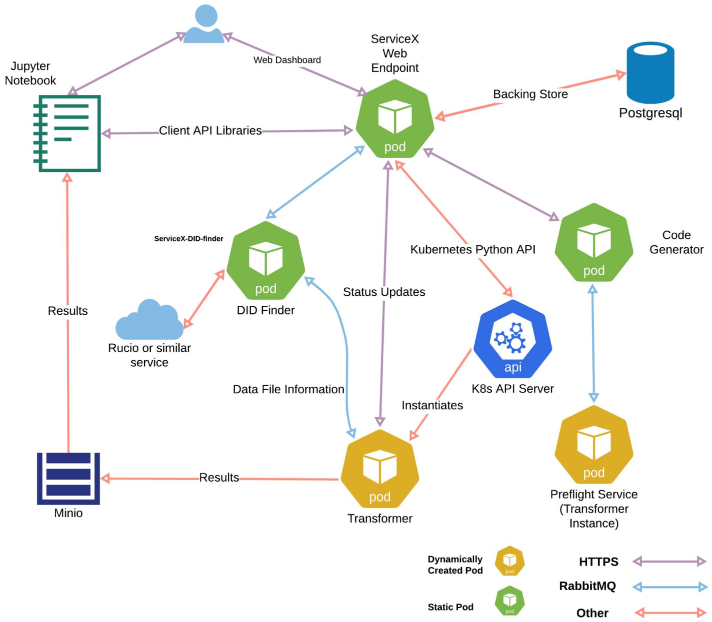
- CPU is a slightly trickier expression because Kubernetes can schedule at a sub-core level while HTCondor uses whole cores

Jupyter Notebooks

- Users can deploy Jupyter notebooks that run on K8S
- Instead of using JupyterHub, we built a simple in-house system that:
    - Allows users to specify CPU, Memory, GPUs, container image
    - Instantiates containers with JupyterLab, the CephFS mount, CVMFS, and the user's real UID and group membership
    - Sets a time limits on how long people can use a notebook and automatically culls old containers

Access to applications

- Jupyter users can access two Kube-native services on the AF today:
  - ServiceX
    - Data filtering and delivery service
    - Delivers slimmed/skimmed input data as PyArrow awkward arrays or flat ROOT files
  - Coffea Casa
    - Service for low-latency columnar analysis
    - Users can submit jobs through Coffea, which use Dask to send workloads to HTCondor



CVMFS experience in K8S

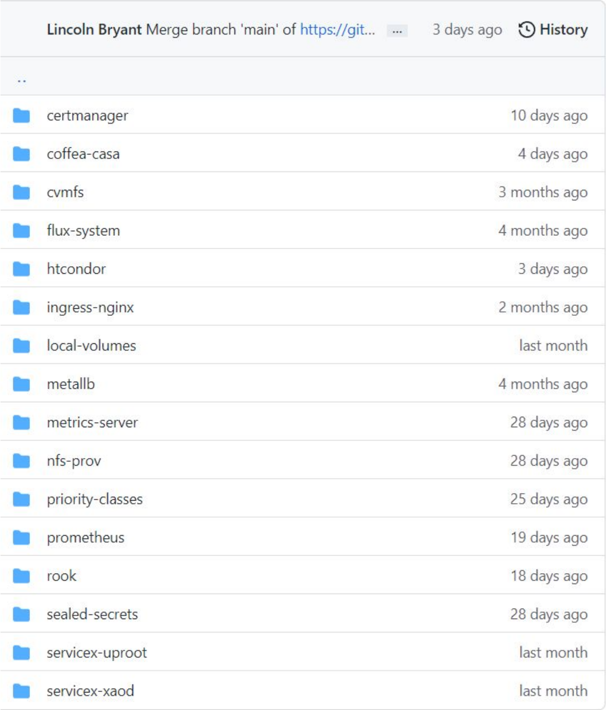
- We tried a number of options for CVMFS with various degrees of success
  - CVMFS CSI
    - Last tried a year or so ago – some issues with zombie processes, haven't tried recently enough to comment beyond that.
  - cvmfsexec
    - Only works for a single user (pilot-style) batch configuration
  - OSG K8S CVMFS
    - Encountered some race conditions where pods would be ready before CVMFS
- We've found that the most stable solution for CVMFS has been to simply install it normally on the workers and bind mount it.

GitOps (Flux, Argo, etc)

- We have found that GitOps is completely **essential** to using Kubernetes in production
  - **In my opinion,** K8S without GitOps is the equivalent of system administration without configuration management (Puppet, Chef, Ansible, etc).
- We are using 2 GitOps solutions for anything we do on Kubernetes
  - Flux v2 for all our clusters at UChicago
  - a custom GitOps solution for SLATE
- Why GitOps?
  - Track when changes are made
  - Review and approve changes in the usual PR model
  - Easily roll back bad deployments
  - ~Single source of truth for the cluster state
  - Immensely useful in disaster recovery scenarios

Components managed by Flux

- Kubernetes requires a lot of add-in functionality to get the most out of it
- On top of a normal **`kubeadm`** installation, we add:
  - Sealed Secrets Operator
  - Rook (Ceph)
  - Prometheus
  - Ingress controller(s)
  - Certificate Management
  - Priority classes
  - … and so on



Lincoln Bryant Merge branch 'main' of https://git... …	3 days ago History
..	
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

Implementing a federated operations model

- Creating tools and a **trust framework** to create distributed platforms such as CDNs to reduce operational costs and innovate more quickly
- **SLATE** (Services Layer At The Edge) implements distributed service operation and a trust model (close as we can get to a NetFlix model given institutional boundaries)
- **Helm packaged applications**
  - **OSG Entrypoints (both), HTCondor-worker, Frontier Squid, Globus, FTS, XCache, PerfSonar-test, Open OnDemand and more**
  - https://github.com/slateci/slate-catalog
    - usable via Helm even if you don't use SLATE

- SLATE-flavored GitOps
  - Deploy, manage SLATE applications via a single Git repository

Services running in SLATE / FedOps

- In the US, we have 2 applications that are run across the Tier 2 facility under the Federated Operations (FedOps) model
  - **XCache**
  - **Frontier Squid**
- Ideal services for deploying through SLATE
  - Focused on caching
  - Failure won’t take the site offline, but will probably get you a GGUS ticket
    - Keeps us accountable!

What has become easier?

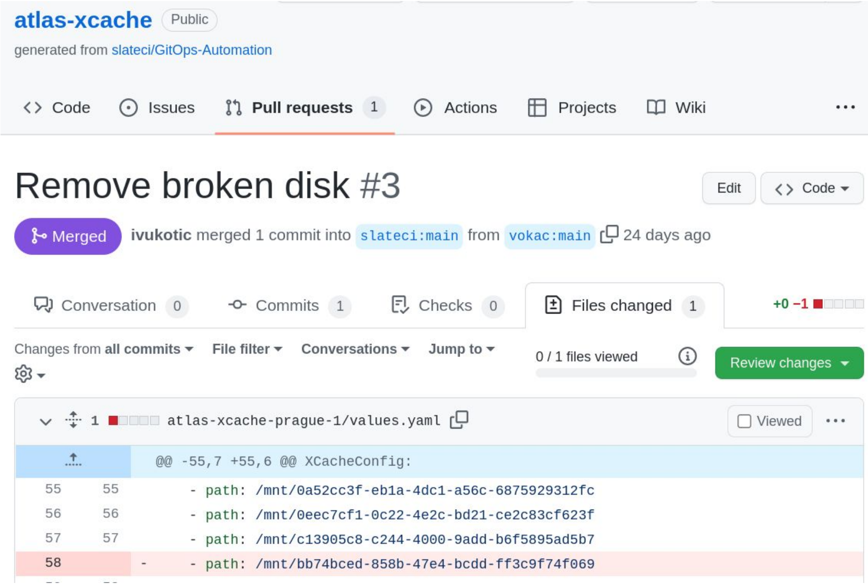
- Applications are generally easier to deploy, maintain, and upgrade
    - For the most part, getting the latest & greatest container version is really as simple as:
        - **`slate instance restart <instance id>`**
- Comparing configuration, versions, etc is all very simple.
- When urgent action is needed (e.g., security updates), it is very easy to get everyone up to date.

```
$ diff atlas-squid-uchicago-prod-3/values.yaml
atlas-squid-mwt2-iu-1/values.yaml
1c1
< Instance: "mwt2-uc-1"
---
> Instance: "mwt2-iu-1"
5c5
<   Hostname: sl-uc-xcache1.slateci.io
---
>   Hostname: iut2-slate.mwt2.org
14a15,17
> Pod:
>   UseHostTimezone: True
>
17,18c20,22
<   CacheMem: "32768"
<   CacheSize: 25000
---
>   CacheMem: "8192"
>   CacheSize: "25000"
>   MaximumObjectSizeInMemory: "1048704"
21d24
<   MaximumObjectSizeInMemory: "1048704"
29c32
<   Logfile_Rotate: "20"
---
>   Logfile_Rotate: "5"
```

Compare instacces by simply diff'ing high-level Helm config

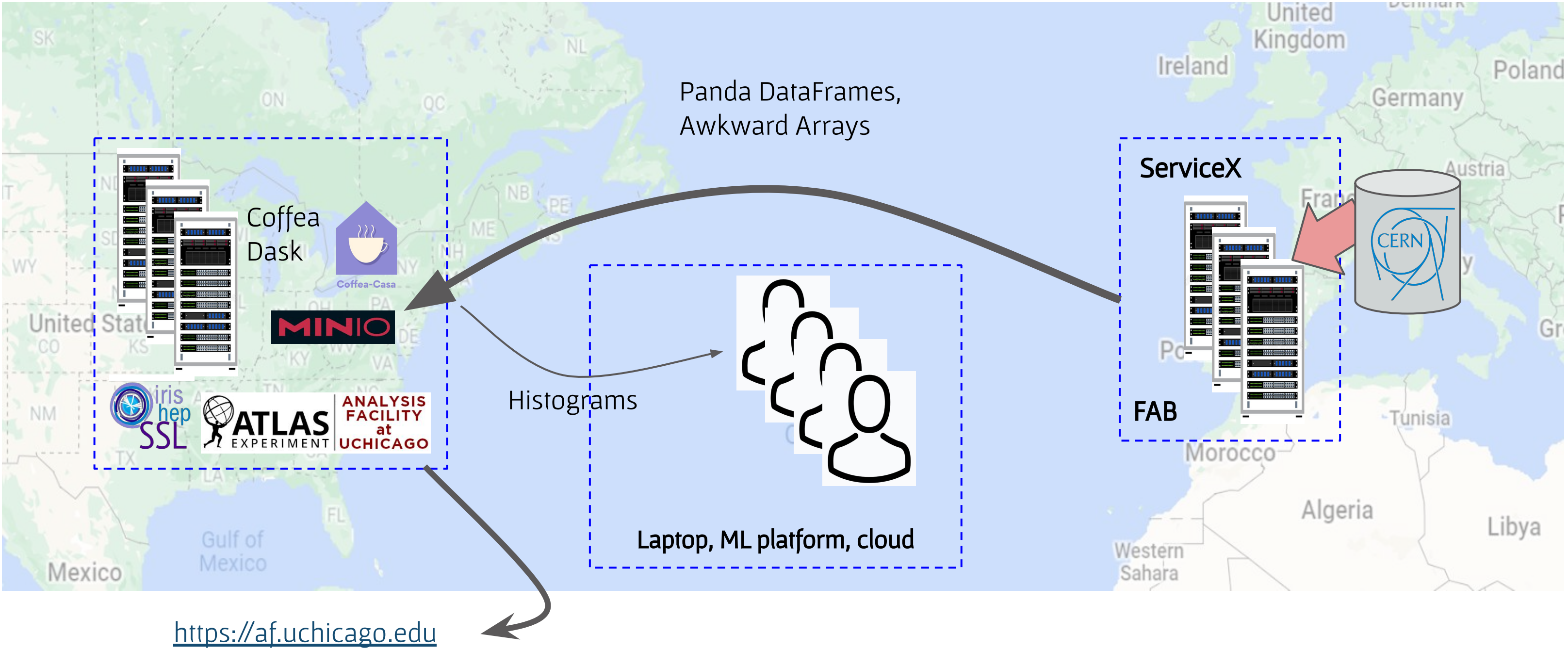
Tradeoffs and improvements

- Site admins give up a bit of autonomy under a SLATE-like model, have to coordinate with FedOps teams for changes
  - e.g. hardware replacement, network changes, etc
- New deployment and management mechanisms = new monitoring needed for when they fail
- Team receives regular emails when:
  - Any changes merged into SLATE GitOps
  - Kubernetes certificates are about to expire
  - K8S instances have restarted or otherwise dropped off of the network



Deploying into FABRIC

Working with **FAB** (FABRIC Across Borders) to demonstrate ServiceX deployment at CERN, delivery of analysis objects to analysis facilities in the U.S.



https://af.uchicago.edu

CERN–>FABRIC–>Analysis Facility–>Notebook

IRIS-HEP Analysis Grand Challenge Tools Workshop Example

```
[1]: from func_adl_servicex import ServiceXSourceUpROOT
     from hist import Hist
     import awkward as ak
```

We will process only one file from one of the samples. File is accessed using root protocol.

```
[2]: input_files = ['root://eospublic.cern.ch//eos/opendata/atlas/OutreachDatasets/2020-01-22/4lep/MC/mc_345060.ggH125_ZZ4lep.4lep.root']
     treename='mini'
```

CERN data source

The following command does almost everything.

First, it specifies data source by calling ServiceXSourceUpROOT and giving it filepath, root tree containing data, and a name of servicex service to use has to be listed in the file servicex.yaml. In this repo there are two servicex instances that can process this data: uproot-af

Secondly, for every event it gets lepton pT.

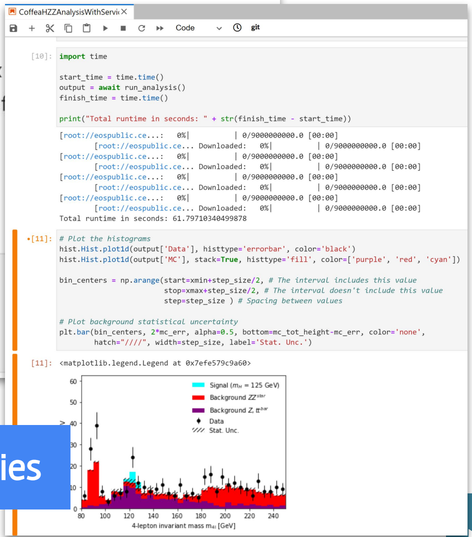
Finally, it specifies that the data should be returned as an Awkward Array.

```
[3]: data = ServiceXSourceUpROOT(input_files, treename, backend_name='uproot-fabric') \
         .Select("lambda e: {'lep_pt': e['lep_pt']}")
         .AsAwkwardArray() \
         .value()
```

Query

Output

ServiceX on FABRIC-NCSA



Notebooks on analysis facilities

All code can be found here.

Summary

- We are using Kubernetes more and more, and we see others doing the same
- We are having success with the Federated Operations model in the US
- Undoubtedly Kubernetes is a force multiplier for developers
- We have to be careful not to underestimate the investment needed to operate K8S clusters
  - especially on premises
  - especially at scale