Research Platform 1

WBS 2.3.5.3 Facility R&D

Aidan Rosberg
Indiana University
Throughput Computing Week 2026

Hello👋I’m Aidan

I am an employee of Indiana University and have been working with the collaboration for 2 years. My official title is Systems Administrator and my position is entirely funded by and dedicated* to US ATLAS

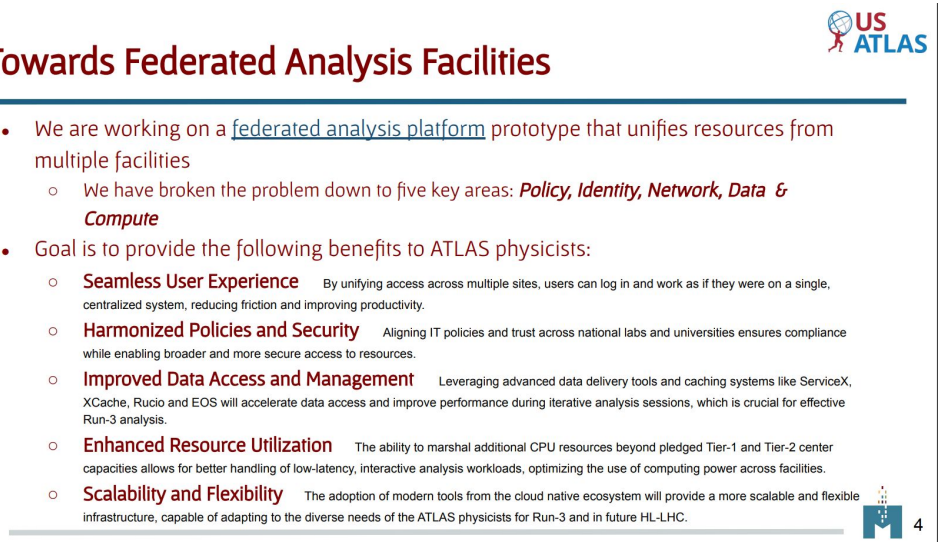
My time is materialized in the

- MWT2 (50%)
- UC AF and Facility R&D activities (50%)

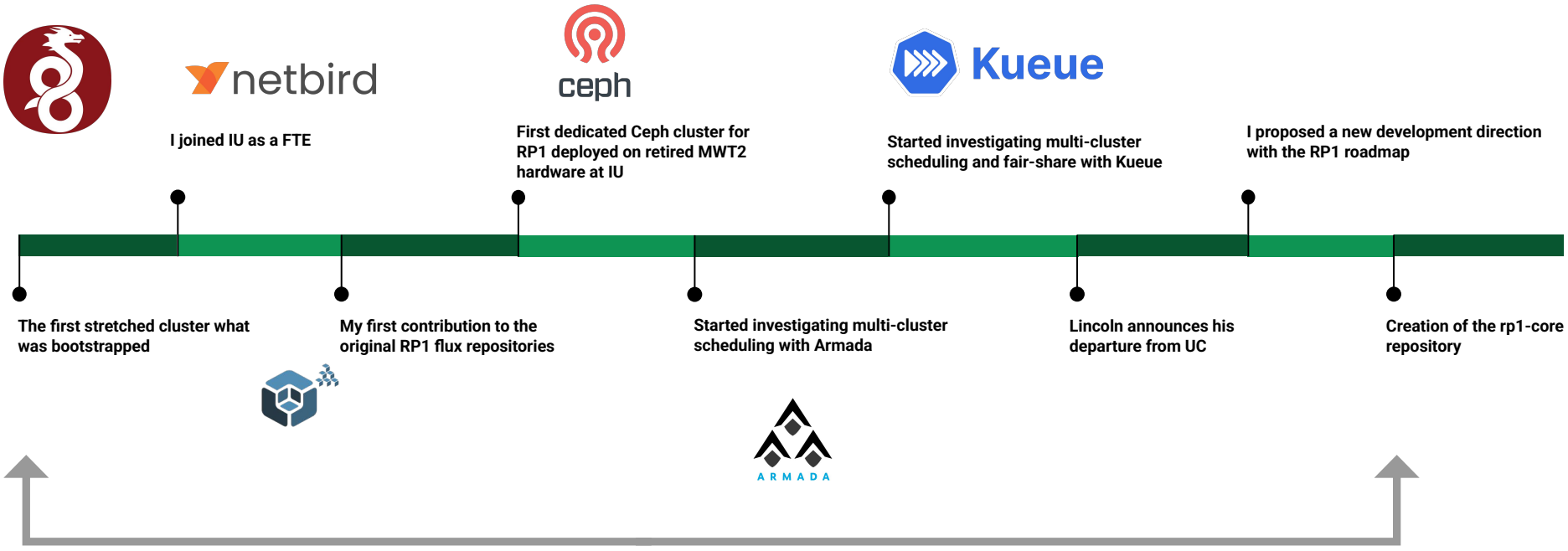
RP1 in the Facility R&D context

First realization of RP1 was presented by Lincoln Bryant “Building Scalable Analysis Infrastructure for ATLAS” [1] at CHEP 2024

- Set out goals and key areas that are still very much important to Facility R&D today



https://doi.org/10.1051/epjconf/202533701062


netbird
ceph
Kueue
I joined IU as a FTE
First dedicated Ceph cluster for RP1 deployed on retired MWT2 hardware at IU
Started investigating multi-cluster scheduling and fair-share with Kueue
I proposed a new development direction with the RP1 roadmap
The first stretched cluster what was bootstrapped
My first contribution to the original RP1 flux repositories
Started investigating multi-cluster scheduling with Armada
Lincoln announces his departure from UC
Creation of the rp1-core repository
ARMADA

Why the change in direction?

We were doing too much with too little

- Trying to recreate an AF, incorporate hardware from other sites, and add functionality beyond what AFs offer all with less dedicated FTE time than an AF.

For an R&D platform this is not ideal

- We spent a lot of time playing catch up with AFs on both hardware and software front which caused both to suffer
- Building up **maintenance burden** when our goal is to **develop**
- Ideally we would like to skip to having an AF. You cannot reason about an AF unless you have an AF.

A new approach to AF development and deployment

Traditionally we think of developing a cluster as an interactive process. You spin up the cluster and frantically develop for an indefinite amount of time until you have desired functionality. This process makes it easy to apply ad hoc changes to achieve the desired functionality without considering ramifications of technical debt.

If we can make creating an AF easy then exploring at the edge becomes much more manageable and the results much more broadly useful.

With the new rendition of RP1 we wanted to move the development phase to the front. We should have the **full functionality** of the platform **independent of the hardware** it runs on.

- How can we develop the platform functionality independent of hardware and still ensure it will work on our infrastructure?
    - To answer this question we must understand how our computing systems achieve their functionality. We must understand their operational context.

What is operational context?

A simple definition: Everything that affects the **functionality** of a **computing system**

Operational context can encapsulate many things

- Site admins, hardware, configuration management, monitoring, location of data center, network, the list goes on…
- What if we could capture the entire operational context of an Analysis Facility in a single repository?

Instead of “We have the hardware now hurry and spin up the software”

We flip “We have the software what hardware do you want it on?”

How can we achieve this in reality?

In reality it is impossible to capture all operational context that contributes to the functionality of a site

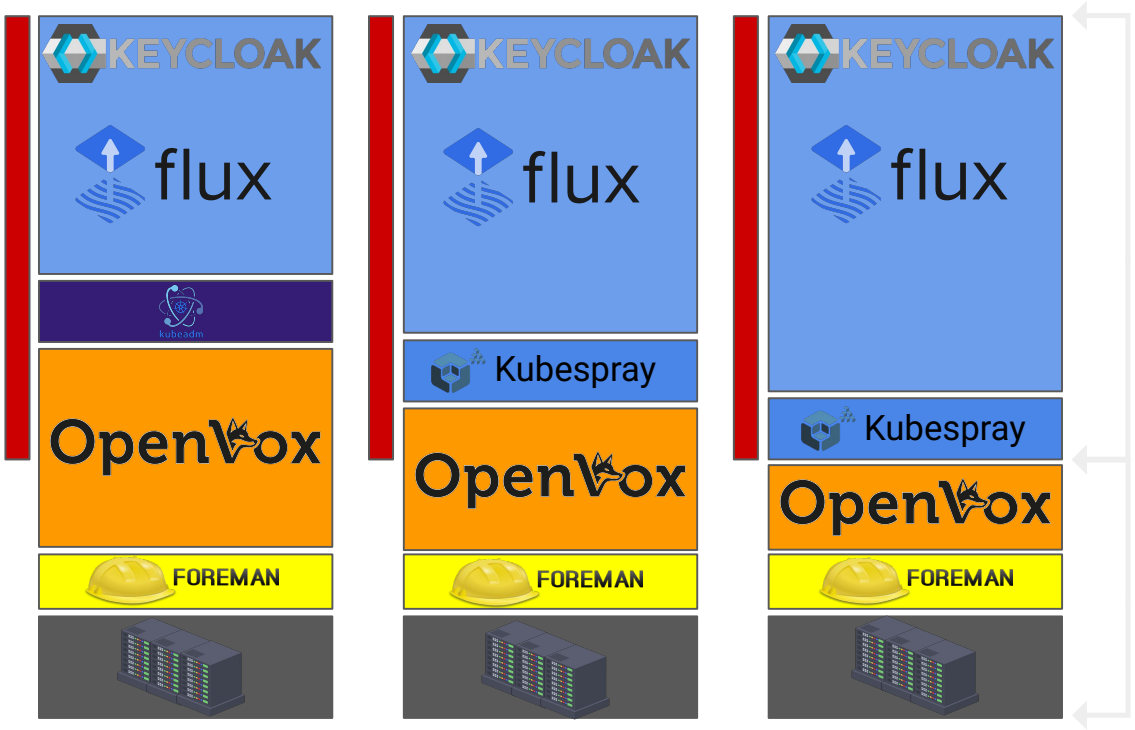
- Spoilers! - We are not to the point of “Press a button and spin up an AF” yet (Although this is potentially possible in a cloud environment)

We are able to get close by adopting a few key development principles:

- Fully embrace Kubernetes as the deployment platform, anything not captured by the cluster is captured by the docs
- Only worry about **where** services run when **absolutely necessary**
- Rely on upstream projects as much as possible, **avoid custom development** unless absolutely necessary
- Keep platform functionality as **self contained** as possible
- Documentation built **with** the platform
- Multi-cluster is unavoidable, need to make clusters easy to bootstrap, configure, and contribute to
- **Develop our infrastructure as if it is software**
    - Every component that affects the functionality of the platform **must** be IaC or documented

Distribution of operational context (Simplified)

Where platform specific operational context lives

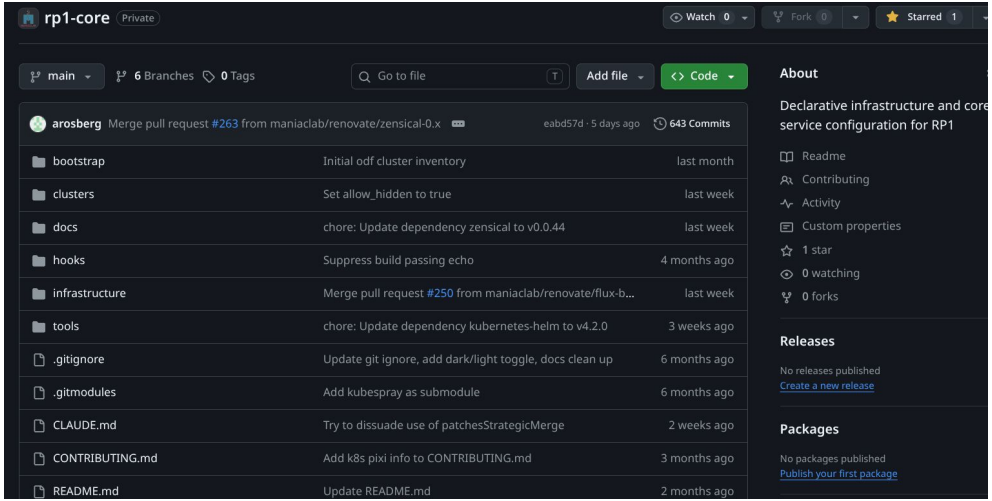


The platform functionality is independent of the hardware

rp1-core

Where the operational context for RP1 clusters live

- Contains all k8s manifests for the platform
- Contains all kubespray configurations
- Contains documentation for operational context not captured by those two layers
- link
  - Currently a private repo in Maniac Lab github organization. Planning on making a mirror in CERN GitLab



rp1-dev

First cluster following the roadmap

Consists of 6 VMs provisioned on IU hardware

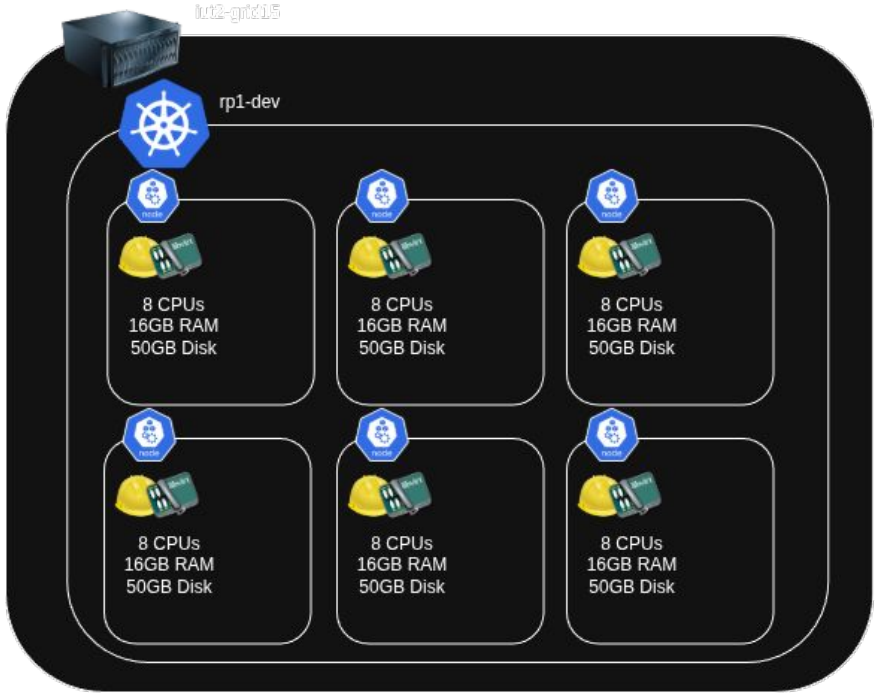
- Attempt to provision the same way we do our bare metal servers to make the dev -> bare metal transition much easier

kvm/qemu has a lot of flexibility, able to recreate an environment similar to a bare metal

- This is the whole point of a VM but it's true!

Acted as the primary development surface for the first ~5 months following the roadmap

- Now acts as a staging cluster for other RP1 clusters



First Demonstration on Bare Metal

Migrated all functionality from rp1-dev to bare metal at IU in roughly 8 hours* of dedicated human time

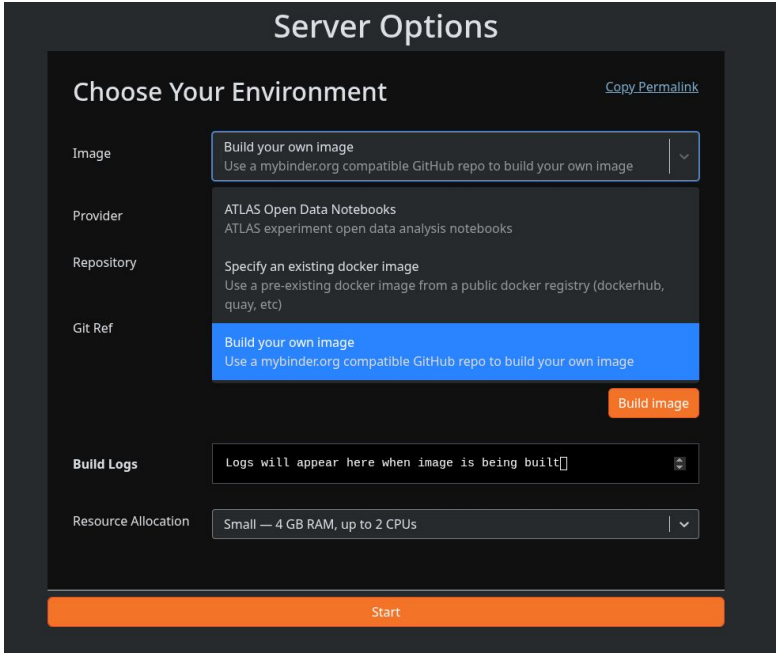
Where most of this time is spent:

- Bootstrapping cluster with kubespray
- Rotating secrets
- Configuring new clients for each service in Keycloak
- Tweaks in flux configuration

User Environment

Focusing our user facing surface (For now)

- The modus operandi of the UC AF is let users do everything
  - This is great for users but not an R&D platform with limited FTE resources
  - Previously had binderhub, coffea-casa, HTCondor
  - This was a large surface to develop, maintain, and operate
- JupyterHub, and the notebook server (JupyterLab) are very powerful tools
  - Integrated cvmfs, setupATLAS, voms-proxy-init, binderhub



Storage Model

Everything use PVs

Currently no traditional /data, /home mounts in user environment

Access method for data is expected to be through tools like rucio client, xrdcp

Auth Model

Keep all user state in Keycloak

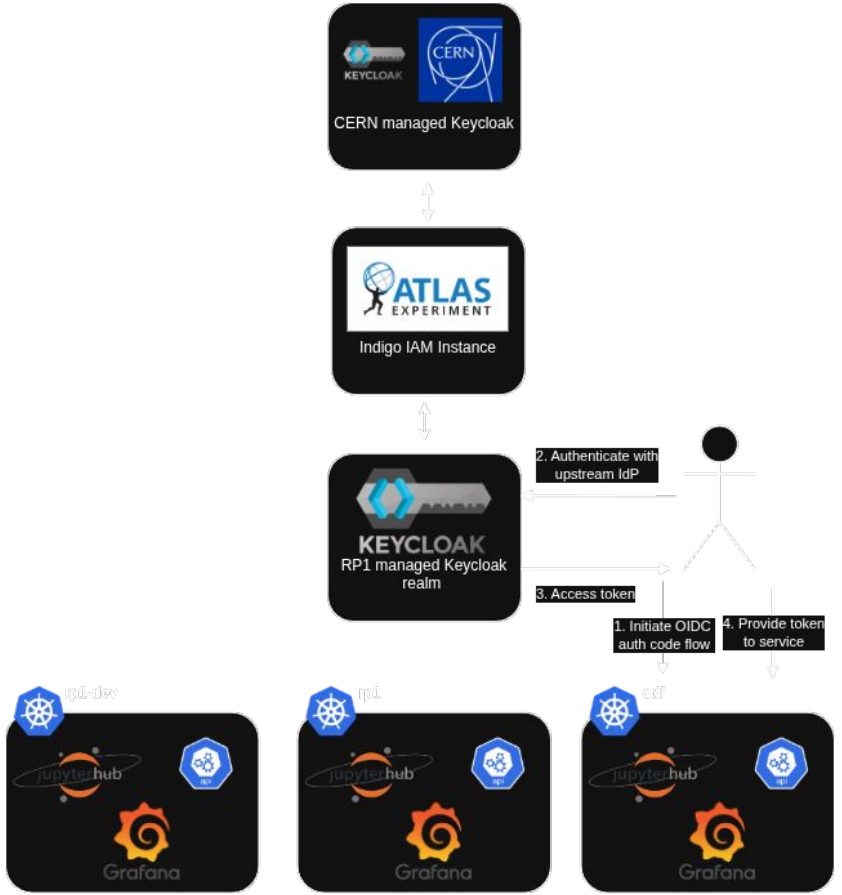
- Groups, roles, etc

Try to stick to OIDC as much as possible

Push authorization decisions down to applications by consuming claims in OIDC tokens

Integrated kubelogin to allow k8s API access authenticated with ATLAS IAM

- All manual cluster operations are tied to a user account



That's great and all but what about physics?

Open Data Facility (ODF) - *First use case driven application of RP1*

Deployed on hardware at UC in ~8 hours* of focused human time

Demonstrated cvmfs, setupATLAS, voms-proxy-init, resource requests, binderhub integration. All contained in JupyterHub and the notebook server.

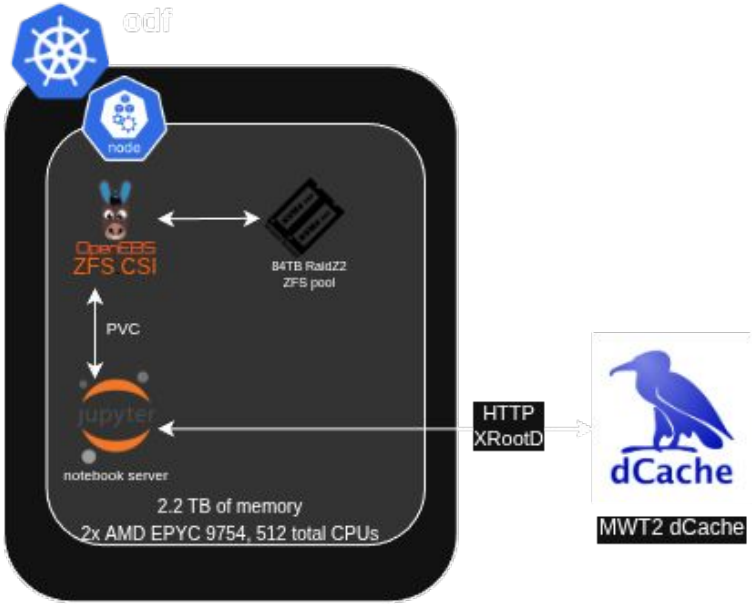
Validated ATLAS Open Data notebooks redirecting data access from CERN EOS to MWT2_OPENDATA mirror

Tested 20 notebook servers running lightweight open data analysis simultaneously

- Single node could easily support 30+ simultaneous notebook servers running analysis
- Need to find a better way to test notebooks to scale up
    - Likely using something like papermill

Ready to scale up workers to support open data research use cases

Will be available for Genesis Phase 1 demonstrators requiring ATLAS open data, to complement the resources of the AmSC



Is a stretched / distributed cluster out of the picture?

I don't think so

- This has been demonstrated successfully in industry and by other research computing platforms (namely NRP likely soon with AMsc)
- It will look different than our first approach

Future development directions

Scale out capability

- There is still large demand for batch which RP1 is currently neglecting

Agentic infrastructure management

- Perfect space to explore agentic infrastructure management, earn operator trust, then push into production deployments

Lots of ground to be trodden on the network side for both local and stretched infra

- Exploring features such as cilium cluster mesh, BGP-based load balancing

Velero

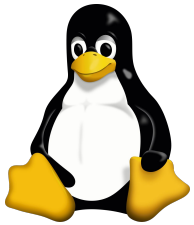
- Since our storage model relies entirely on PVs we are in need solution to safely backup and restore, perform disaster recovery, and migrate stateful workloads easily

What's the cost of all this?

<1 FTE worth of time

Software cost approaching $0

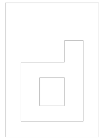
A handful of spare hardware at UC and IU sites







kubernetes



K
Kustomize

HELM

ANSIBLE



cilium


envoy
gateway

CERT
MANAGER


Grafana

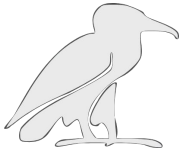



MetalLB

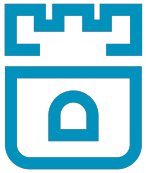

libvirt

QEMU

KVM

RUCIO



jupyter


ceph


XRootD

binderhub

Summary

Refocused our development path and user facing surface. Worked heavily on the **process** of developing and operating our clusters

Still playing catch up with AFs in terms of raw functionality but have the agility and momentum to explore the boundaries and inform platform development for US ATLAS

- Some of things RP1 has explored have already been integrated into the UC AF

Thank you

Our public footprint: https://rp1.hl-lhc.io

Backup

Key Security Considerations

Bootstrapping the cluster requires root access. This is not avoidable.

- NRP requires root access (Or even IPMI access)

The user-facing environment is mostly tied to the security of the notebook server and the images users run

Authentication is entirely managed with keycloak

- Authorization decisions are made by the applications based on claims in the OIDC tokens

Zero use of admin credentials for kubernetes API or other services

LLM Use

Been using LLMs in some capacity since I started here 2 years ago

- Up until ~January 2026 my usage was primarily with chat interfaces (Latest GPT models at the time)
- Got access to a claude subscription and started using claude code cli
  - This was a real game changer for productivity
  - Ability understand project structure and retrieve relevant context
    - Not just from in repo files but also from other sources like using kubectl cli to query k8s, or using gh cli to read issues about software
- rp1-core is 98+% LLM generated and 99% human reviewed
- I think the most effective use we can get out of AI/LLMs is empowering us as individuals
- LLM usage has definitely increased **my** productivity
  - Lots of variables through
    - Getting more familiar with working with our systems in general
    - Motivation from a new beginning with the roadmap to guide development
    - Anxiety from losing the my mentor guiding me through the process